ABSTRACT

Title of dissertation: INTERPRETING DEEP LEARNING MODELS AND UNLOCKING NEW APPLICATIONS WITH IT

Samyadeep Basu, Doctor of Philosophy, 2025

Dissertation directed by: Professor Soheil Feizi
University of Maryland

In recent years, modern deep learning has made significant strides across various domains, including natural language processing, computer vision, and speech recognition. These advancements have been driven by innovations in scaling pre-training data, developing new model architectures, integrating distinct modalities (e.g., vision and language, audio and language), and employing modern engineering practices. However, despite these innovations in building better models, progress in understanding these models to enhance their reliability has been relatively slow. In this thesis, we lay the groundwork for interpreting modern deep learning models—such as vision, text-to-image, and multimodal language models—by examining them through the perspectives of **data** and **internal model components**. We aim to unlock various capabilities, including model editing and model steering, to enhance their reliability. First, we build on the principles of robust statistics to interpret test-time predictions by identifying important training examples using higher-order influence functions. However, we find that influence functions can be fragile for large deep models, which limits their practical applications. To address this, we develop optimization-based data selection strategies to automatically generate stress-testing sets from large vision datasets, testing the

reliability of vision models within a few-shot learning framework. Overall, our investigations show that while analyzing models through the lens of data provides valuable insights for potential improvements, it does not offer a direct method for controlling and enhancing the reliability of these models. To this end, we investigate deep models by focusing on their internal components. We develop causal mediation analysis methods to understand knowledge storage in text-to-image generative models like Stable Diffusion. Based on these insights, we create novel model editing techniques that can remove copyrighted styles and objects from text-to-image models with minimal weight updates. We scale these methods to edit large open-source models such as SD-XL and DeepFloyd. As a follow-up, we then introduce innovative causal mediation analysis methods and a richly annotated probe dataset to interpret multimodal large language models like LLaVa. Our approach allows us to understand how these models internally retrieve relevant knowledge for factual Visual Question Answering (VQA) tasks. Leveraging these insights, we develop a novel model editing method that can effectively introduce rare, long-tailed knowledge or correct specific failure modes in multimodal large language models. Using similar principles, we explore vision models (in particular the ViT architecture), developing methods to interpret image representations based on internal components such as attention heads, using text descriptions. We apply these interpretability insights to (i) mitigate spurious correlations, (ii) enable zero-shot segmentation, and (iii) facilitate text or image-conditioned image retrieval. We also extend our mechanistic interpretability techniques to understand and control language models for real-world tasks, such as context-augmented generation in question-answering systems (i.e., extractive QA). In particular, we find that insights from mechanistic circuits can be useful towards context-data attribution and model steering towards improved context faithfulness. Finally, we leverage interpretability insights from multimodal models to enhance their compositionality in image-conditioned text retrieval and text-guided image generation. For vision-language models (VLMs) like CLIP, we propose a distillation method

that transfers compositional knowledge from diffusion models to CLIP. For diffusion models, we introduce a lightweight fine-tuning approach that learns a linear layer on the conditioning text encoder, improving compositional generation for attribute binding. Overall, our thesis designs and adapts interpretable methods and leverages interpretable insights to uncover various capabilities in pre-trained models.

INTERPRETING DEEP LEARNING MODELS AND UNLOCKING NEW APPLICATIONS WITH IT

by

Samyadeep Basu

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2025

Advisory Committee:
Professor Soheil Feizi, Chair/Advisor
Professor Hernisa Kacorri, Dean's Representative
Professor Furong Huang, Professor Abhinav Shrivastava, Committee Members
Dr. Varun Manjunatha, Dr. Daniela Massiceti, External Members

Dedication

To my parents, partner and friends for their love and support.

Acknowledgments

First and foremost, I would like to express my deepest gratitude to Dr. Soheil Feizi for his invaluable guidance and unwavering support throughout my PhD. His mentorship has been one of the most significant factors in the successful completion of my dissertation. I arrived at the University of Maryland in 2018 to pursue a Master's degree, initially intending to focus on Computational Biology. However, due to various circumstances, that path did not materialize. While searching for an advisor, Dr. Feizi took a chance on me in my second year and guided me through my first research project, which led to a submission at AISTATS. With no prior background in machine learning or deep learning, the project was a steep learning curve, but Dr. Feizi's encouragement ensured that I persevered. Due to personal circumstances, I had to defer my PhD admission, and Dr. Feizi was incredibly supportive of my decision. After completing my Master's, he gave me the freedom to explore diverse topics in deep learning, allowing me to build a broad foundation. This combination of intellectual freedom and strong mentorship was instrumental in helping me publish at top conferences and secure valuable internships. I am profoundly grateful to Dr. Feizi for his guidance, patience, and support throughout my PhD journey—his influence has truly shaped my academic and professional path.

The PhD journey can be long and often lonely, and I am incredibly grateful to have had the unwavering support of my parents, partner—now my wife—Sneha and in-laws. Sneha has stood by me through my lowest moments in ways no one else could, and for that, I will always be indebted to her. During my time in Maryland, I was fortunate to have the support of a wonderful group of friends who made this journey fulfilling. My College Park friends—Aman, Shlok, Ameya, Ryan, Naman, Anjali, Sai, Noor, Vasu, Komal, Sanchita, Yatharth, Neha, Pavan, Amanpreet, Ishita, Shishira, Shramay, Anshul, Susmija, Pranav, Ketul, Aadesh, and Manas—played an integral role in making these years memorable. I

am also deeply thankful to my friends from my undergraduate days and beyond—Aditya, Siddhant, Dhairya, Srajit, Parikshit, Vandit, Surbhi, Dewanshu, Anish, Kunal, Fabian, and Himanshu — whose regular conversations and encouragement kept me going. Their support made all the difference in this journey, and I am truly grateful to have them in my life.

During my PhD I also had the opportunity to do internships at Microsoft Research and Adobe Research. From MSR, I would particularly like to thank Dr. Daniela Massiceti who supported me not only during my internship but beyond that in my PhD as a mentor. Even though there was a time-difference, she made sure to schedule regular meetings to mentor me and carve a path towards a successful PhD. From Adobe, Varun has been the main motivator for me to work on interpretability. His idealogies on reverse engineering large models have shaped my PhD and in fact is one of the core parts of my PhD thesis. He has not only supported me in projects, but also as a lighting guide towards having a successful PhD and post PhD transition. I will forever be indebted to both Varun and Daniela. I can easily say that they have turned from great mentors to friends along the way – for which I am grateful.

Finally, I would like to thank my amazing labmates without whom this PhD would not have been possible.

Table of Contents

Chapter 1: **Introduction**

1.1 Thesis Statement

In this thesis, we develop and investigate methods for interpreting deep models through the lens of **data** and **internal model components**. We use these insights towards developing fast and scalable model editing methods, automatically generating difficult few-shot learning benchmarks and mitigating spurious correlations amongst others.

1.2 Thesis Overview

In recent years, a vast array of deep learning models has been developed and implemented in real-world applications. These models encompass unimodal types, such as those for text, images, videos, and audio, as well as multimodal models that combine modalities like vision and language, video and language, or audio and language. As these models have grown rapidly in size—driven by increases in model size, the scale of pre-training data, and the availability of advanced computing infrastructure—the research community has struggled to fully understand how these models make specific decisions. Furthermore, it remains unclear whether gaining a deeper understanding of these models would directly contribute to targeted improvements or enhancements in their downstream applications. In this thesis, we establish a framework for efficiently interpreting recently developed deep learning models, including

both unimodal and multimodal types. We explore these models through the perspectives of their pre-training data, internal model components, and fine-tuning algorithms.

First, we investigate how a classifier's decision-making process can be attributed to a group of training samples to understand the failure modes of deep models. We develop second-order group influence functions, which can efficiently approximate leave-k-out retraining. Through a range of experiments on synthetic data and standard image datasets, we demonstrate that our proposed second-order influence function better approximates leave-k-out retraining than first-order influence functions. However, for deep models involving non-convex losses, we also find that the first-order influence function baseline is often inaccurate when compared to ground-truth influence.

We then conduct a comprehensive large-scale empirical study to highlight the advantages and limitations of influence functions for interpreting deep models in the context of training data. Using datasets up to the scale of ImageNet, we identify the conditions under which the approximation provided by influence functions is relatively error-free. Given that influence functions are unstable for highly overparameterized models, we explore a different algorithmic approach to understanding the failure modes of pre-trained models. Specifically, we examine these failure modes through the lens of few-shot tasks. To understand the worst-case failure scenarios of deep models, we design FastDiffSel, an optimization-based algorithm that can automatically extract challenging training sets for a given test set. We use FastDiffSel to identify difficult few-shot tasks from vision datasets, including ImageNet, ObjectNet, and CURE-OR. Our findings reveal that pre-trained models often fail when there is a natural distribution shift between the few-shot training set and the test examples. As a result, we curate a challenging few-shot testing set, HardMetaDataset++, which can be used to stress-test models.

While analyzing deep models through the lens of data helps identify their failure modes, it offers limited flexibility for post-training model control. To address this, we shift our focus to a "mechanistic" investigation, examining the internal components of these models. We first develop methodologies to understand how knowledge is stored in large-scale text-to-image models. Based on our findings, we design scalable, efficient, and data-free model editing techniques to remove copyrighted concepts from these models. Our empirical experiments demonstrate the effectiveness of our model editing methods in modifying large-scale open-source text-to-image generative models. We then extend our approach to multimodal language models, developing MultimodalCausalTrace, a tool that identifies crucial model components for factual Visual Question Answering (VQA) tasks. Building on these insights, we introduce MultEdit, a method for editing multimodal language models to insert new, rare knowledge or fix existing issues. Although our methods currently focus on interpreting multimodal models, a significant challenge remains in understanding the internal components of general Vision Transformers (ViTs) using human-understandable concepts, such as text.

To tackle this, we create RepDecompose, an approach that automatically decomposes final representations in general ViTs through a recursive process. These components are then aligned with CLIP's image encoder, allowing interpretation via the text encoder. Our analysis reveals that different attention heads in ViTs encode distinct concepts, such as patterns, colors, and locations. We leverage these insights to modify the identified attention heads, mitigating spurious correlations and utilizing their embeddings for tasks like zero-shot segmentation, text-conditioned image retrieval, and general image retrieval.

While our current work has laid the groundwork for interpreting and controlling vision and multimodal models, we also focus on adapting these methods to control language models. In particular, we enhance language models for tasks such as data attribution to context and

mitigating hallucinations. Given the recent advancements in retrieval-augmented generation, there are significant practical applications in context-augmented question-answering setups. In this phase of our research, we investigate the internal circuits (e.g., sub-graphs) of language models that are causally linked to retrieval-augmented generation tasks. By analyzing different components of these circuits, we design zero-shot data attribution methods.

Finally, we investigate the compositionality issues in VLMs (e.g., CLIP) and text-guided image generation models (e.g., diffusion models). In particular, we find that diffusion models are strong in terms of compositionality and such knowledge can be transferred to CLIP to improve it's compositionality. To this end, we introduce SDS-CLIP, a light-weight fine-tuning based distillation method which can improve CLIP's compositionality without harming it's zero-shot capabilities. For diffusion models, we find that the text-embedding for compositional prompt is often sub-optimal. We show that solely fine-tuning a linear projection layer on the CLIP's text-embedding can improve compositional generation for a variety of open-source text-to-image diffusion models (including SDv3).

Overall, our thesis has developed new approaches towards understanding deep models and has shown the possibilities of practical applications of model interpretability.

1.3 Thesis Contributions

This thesis makes several research contributions towards interpreting deep learning models, spanning both unimodal and multimodal models. In particular, we make contributions towards developing interpreting deep learning models through the lens of **data** as well as **internal model components**. Using our interpretability insights, we further develop light-weight methods towards unlocking capabilities in these models such as model editing.

Below we state our contributions:

Interpreting Test-Time Predictions Through Influence Functions

- We develop second-order group influence functions which can attribute test-time predictions to a group of samples in the training data. Our second-order group influence functions effectively approximates leave-k-out retraining by a second-order taylors expansion around the optimally trained model with all the training examples. [ICML 2020]
- We empirically investigate the limits of influence functions for deep networks. To this end, we first analyse influence functions in a controlled experimental setup with synthetic data. We then scale up the analysis across different pre-training data and models upto the ImageNet scale – highlighting the fragilities of influence functions at larger model scales. [ICLR 2021]

Automatically Designing Difficult Few-Shot Benchmarks for Model Reliability

- We design an optimized-based data selection algorithm, which can automatically curate difficult few-shot benchmarks from large-scale vision datasets. The curated dataset from our algorithm can be used towards stress testing deep models for reliability. [ICLR 2023]

Mechanistically Understanding and Editing Text-to-Image Diffusion Models

- We design a causal tracing methodology which can locate internal model components which causally store knowledge corresponding to various visual attributes such as style, objects or facts. We then design model editing methods towards updating the weights of the identified components in a light-weight manner. [ICLR 2024]

- We investigate knowledge storage about visual attributes in the cross-attention layers across various open-source text-to-image diffusion models. We then use model editing towards updating the weights in those locations to remove copyrighted style, objects and update the model with new facts. [ICML 2024]

Mechanistically Understanding and Editing Multimodal Language Models

- We develop MultimodalCausalTrace, which can identify causal locations for a factual VQA task using a constrained based formulation. Along with providing salient interpretability insights about the inner workings of multimodal language models – we then introduce MultEdit, which can effectively introduce long-tailed knowledge into these models. [NeurIPS 2024]

Mechanistically Understanding and Unlocking Zero-Shot Capabilities in Vision Transformers

- We introduce RepDecompose which decomposes the final representation in Vision Transformers as a function of internal model components such as attention heads. We then interpret these attention heads via text, by aligning their embeddings to CLIP's image encoder and then using CLIP's text-encoder to interpret them. Based on our interpretability insights, we unlock various zero-shot capabilities in Vision Transformers: (i) Spurious correlation mitigation; (ii) Zero-shot segmentation; (iii) image / text conditioned image retrieval. [NeurIPS 2024]

Mechanistically Understanding and Enhancing Context-Augmented Language Models

- Large language models are widely used for document processing and question-

answering. In this work, we extract mechanistic circuits for context-augmented extractive QA using causal mediation analysis on model components (e.g., attention heads, MLPs). Our analysis reveals how models balance parametric memory and retrieved context, identifying a small set of attention heads that reliably perform data attribution by default. Leveraging this, we introduce ATTNATTRIB, a fast attribution algorithm achieving state-of-the-art results across QA benchmarks. We further demonstrate that ATTNATTRIB can steer models to prioritize context over parametric memory. Beyond insights into model behavior, our work highlights practical applications of circuits in attribution and model control. [ICML Review]

Improving Compositionality in Multimodal Models

- Image-text contrastive models like CLIP excel in zero-shot classification, retrieval, and transfer learning but struggle with compositional visio-linguistic tasks (e.g., attribute binding, object relationships), often performing at chance levels. To address this, we propose SDS-CLIP, a lightweight, sample-efficient distillation method that enhances CLIP's compositional reasoning. Our approach fine-tunes CLIP using a distillation objective from large text-to-image generative models like Stable Diffusion, known for strong visio-linguistic reasoning. SDS-CLIP improves CLIP's performance by up to 7% on Winoground and 3% on ARO, demonstrating the potential of generative model distillation to enhance contrastive learning. [EMNLP 2024]
- Text-to-image diffusion-based generative models have the stunning ability to generate photo-realistic images and achieve state-of-the-art low FID scores on challenging image generation benchmarks. However, one of the primary failure modes of these text-to-image generative models is in composing attributes, objects, and their associated relationships accurately into an image. In our paper, we investigate compositional

attribute binding failures, where the model fails to correctly associate descriptive attributes (such as color, shape, or texture) with the corresponding objects in the generated images, and highlight that imperfect text conditioning with CLIP text-encoder is one of the primary reasons behind the inability of these models to generate high-fidelity compositional scenes. In particular, we show that (i) there exists an optimal text-embedding space that can generate highly coherent compositional scenes showing that the output space of the CLIP text-encoder is sub-optimal, and (ii) the final token embeddings in CLIP are erroneous as they often include attention contributions from unrelated tokens in compositional prompts. Our main finding shows that significant compositional improvements can be achieved (without harming the model's FID score) by fine-tuning *only* a simple and parameter-efficient linear projection on CLIP's representation space in Stable-Diffusion variants using a small set of compositional image-text pairs. [ACL Submission]

1.4 Publications and Authorship

This thesis draws upon previously published manuscripts, manuscripts currently under review, and ongoing works listed in the table underneath. While I serve as the principal author (except for Chapter 7 and 9.2), the research presented here reflects the culmination of collaborative efforts with my advisor, Soheil Feizi, and mentors Daniela Massiceti, Varun Manjunatha alongside invaluable contributions from mentors and colleagues UMD, Adobe Research and Microsoft Research. Throughout Chapters 3-9, I use the pronoun 'we' to acknowledge the collective contributions of all my collaborators.

Understanding the Model through Data

- On Second-Order Group Influence Functions for Black-Box Predictions – ICML 2020
  - Samyadeep Basu, Xuchen You, Soheil Feizi
- Influence Functions in Deep Learning are Fragile – ICLR 2021
  - Samyadeep Basu, Philip Pope, Soheil Feizi
- Hard-Meta-Dataset++ : Towards Understanding Few-Shot Performance on Difficult Tasks – ICLR 2023
  - Samyadeep Basu, Megan Stanley, John Bronskill, Soheil Feizi, Daniela Massiceti

Understanding the Model through Internal Model Components

- Localizing and Editing Knowledge in Text-to-Image Generative Models – ICLR 2024
  - Samyadeep Basu, Cherry Zhao, Vlad Morariu, Soheil Feizi, Varun Manjunatha
- On Mechanistic Knowledge Localization for Text-to-Image Generative Models – ICML 2024
  - Samyadeep Basu*, Keivan Rezaei*, Cherry Zhao, Vlad Morariu, Ryan Rossi, Varun Manjunatha, Priyatham Kattakinda, Soheil Feizi
- Understanding Information Storage and Transfer in Multimodal Language Models – NeurIPS 2024
  - Samyadeep Basu, Cecily Morrison, Martin Grayson, Besmira Nushi, Soheil Feizi, Daniela Massiceti
- Decomposing and Interpreting Image Representations in ViTs Beyond CLIP – NeurIPS 2024
  - Sriram Balasubramaniam, Samyadeep Basu, Soheil Feizi
- On Mechanistic Circuits for Extractive Question-Answering – Under Submission in ICML 2025
  - Samyadeep Basu, Cherry Zhao, Ryan Rossi, Vlad Morariu, Jack Wang, Soheil Feizi, Varun Manjunatha

Improving Compositionality in Multimodal Models

- Distilling Knowledge from Text-to-Image Models Improves Visio-Linguistic Reasoning in CLIP – EMNLP 2024
  - Samyadeep Basu, Daniela Massiceti, Shell Xu Hu, Soheil Feizi
- Improving Compositionality in Text-to-Image Models Via Enhanced Text-Embeddings – Under Submission in ACL 2025
  - Arman Zarei*, Keivan Rezaei*, Samyadeep Basu, Mazda Moayeri, Priyatham Kattakinda, Soheil Feizi

Figure 1.1: Primary Works Directly Related to the Thesis.

Additional Works

- Rethinking Artistic Copyright Infringement in the Era of Text-to-Image Generative Models – ICLR 2025
  - Mazda Moayeri, Sriram Balasubramaniam, Samyadeep Basu, Priyatham Kattakinda, Atoosa Chegini, Soheil Feizi
- IntCoOp: Interpretability-Aware Vision-Language Prompt-Tuning – EMNLP 2024
  - Soumya Ghosal, Samyadeep Basu, Soheil Feizi, Dinesh Manocha
- Strong Baselines for Parameter-Efficient Few-Shot Fine-Tuning – AAAI 2024
  - Samyadeep Basu, Daniela Massiceti, Shell Xu Hu, Soheil Feizi
- On Surgical Fine-tuning for Small Language Encoders – EMNLP 2023 Findings
  - A Lodha, GV Belapurkar, S Chalkapurkar, Y Tao, Samyadeep Basu, Reshmi Ghosh, Dmitrii Petrov, Soundar Srinivasan
- EditVal: Benchmarking Diffusion Based Text-Guided Image Editing Methods – arXiv
  - Samyadeep Basu*, Shweta Bhardwaj*, Mehrdad Saberi*, Atoosa Chegini, Daniela Massiceti, Maziar Sanjabi, Shell Xu Hu, Soheil Feizi

Figure 1.2: Additional Relevant Works done during the Thesis.

Chapter 2: **Related Work**

2.1 Interpreting Test-Time Predictions Through Influence Functions

Influence functions, a classical technique from robust statistics introduced by [50, 51] were first used in the machine learning community for interpretability by [119] to approximate the effect of upweighting a training point on the model parameters and test-loss for a particular test sample. In the past few years, there has been an increase in the applications of influence functions for a variety of machine learning tasks. [203] used influence functions to produce confidence intervals for a prediction and to audit the reliability of predictions. [242] used influence functions to approximate the gradient in order to recover a counterfactual distribution and increase model fairness, while [30] used influence functions to understand the origins of bias in word-embeddings. [117] crafted stronger data poisoning attacks using influence functions. Influence functions can also be used to detect extrapolation [159] in certain specific cases, validate causal inference models [7] and identify influential pre-training points [40]. Infinitesimal jackknife or the delta method are ideas closely related to influence functions for linear approximations of leave-one-out cross validation [66, 107]. Recently a higher-order instance [85] of infinitesimal jackknife [107] was used to approximate cross-validation procedures. While their setting corresponding to approximations of leave-k-out re-training is relatively similar to our paper, our higher-order terms preserve the empirical weight distribution of the training data in the ERM and

are derived from influence functions, while in [85] instances of infinitesimal jackknife is used. These differences lead to our higher-order terms being marginally different than the one proposed in [85]. Our proposed second-order approximation for group influence function is additionally backed by a thorough empirical study across different settings in the case of linear models which has not yet been explored in prior works. Recently, alternative methods to find influential samples in deep networks have been proposed. In [258], test-time predictions are explained by a kernel function evaluated at the training samples. Influential training examples can also be obtained by tracking the change in loss for a test-prediction through model-checkpoints, which are stored during the training time [189]. While these alternative methods [189, 258] work well for deep networks in interpreting model predictions, they lack the “jackknife" like ability of influence functions which makes it useful in multiple applications other than interpretability (e.g. uncertainty estimation).

2.2 Automatically Designing Difficult Few-Shot Benchmarks for Model Reliability

Difficult tasks. Previous works [2, 12, 57] have shown that state-of-the-art few-shot classifiers generally display a wide range in performance when adapting to different test tasks. [2] use this observation to develop a greedy search-based algorithm that can specifically extract difficult tasks for further study. They consider only meta-learning approaches and, due to the computational requirements of a greedy search, are limited to small-scale datasets including mini-ImageNet and CIFAR-FS. [12] also study difficult tasks through a correlation-based analysis. We extend on both of these works by (i) proposing a scalable algorithm – FastDiffSel that can extract difficult tasks from any large-scale vision dataset, and (ii) conducting a deep empirical evaluation on the robustness of a broader range of

meta-learning and transfer learning approaches on these difficult tasks. Potentially ideas from subset selection [114, 247] can be adapted for few-shot task extraction, but we leave it for future work.

Few-shot classification benchmarks. Meta-Dataset [230] and Hard-Meta-Dataset++ [65] are two of the most challenging few-shot image classification benchmarks in the current literature. They cover a wide range of domains and primarily evaluate the ability of a few-shot classifier to generalise to novel object classes, datasets and domains. Other few-shot benchmarks have been introduced to specifically target adaptation to images with high real-world variation, including ORBIT [163] and cross-domain transfer beyond natural images, including BSCD-FSL [89]. We note, however, that unlike HardMetaDataset, none of these benchmarks specifically target difficult tasks for few-shot classification.

2.3 Mechanistically Understanding and Editing Text-to-Image Diffusion Models

Text-to-Image Diffusion Models. In the last year, a large number of text-to-image models such as Stable-Diffusion [197], DALLE [193] , Imagen [201] and others [16, 38, 59, 111] have been released. In addition, the open-source community has released DeepFloyd[1] and Midjourney[2] which can generate photorealistic images given a text prompt. While most of these models operate in the latent space of the images, they differ in the text-encoder used. For e.g., Stable-Diffusion uses CLIP for the text-encoder, whereas Imagen uses T5. These text-to-image diffusion models have been used as a basis for various applications such as image-editing, semantic-segmentation, object-detection, image restoration and zero-shot classification.

[1] https://www.deepfloyd.ai
[2] https://www.midjourney.com/

Intepretability of Text-to-Image Models. To our knowledge, few works delve into the mechanisms of large text-to-image models like Stable-Diffusion. DAAM [221] interprets diffusion models by analyzing cross-attention maps between text tokens and images, emphasizing their semantic accuracy for interpretation. In contrast, our approach focuses on comprehending the inner workings of diffusion models by investigating the storage of visual knowledge related to different attributes. We explore various model layers beyond just the cross-attention layer. [23] leverage causal tracing to understand how knowledge is stored in text-to-image models such as Stable-Diffusion-v1.

Editing Text-to-Image Models. Understanding knowledge storage in diffusion models has significant implications for model editing. This ability to modify a diffusion model's behavior without retraining from scratch were first explored in Concept-Ablation [126] and Concept-Erasure [78]. TIME [182] is another model editing method which translates between concepts by modifying the key and value matrices in cross-attention layers. However, the experiments in [182] do not specifically target removing or updating concepts such as those used in [78, 126]. We also acknowledge concurrent works [79] and [10] use a closed-form update on the cross-attention layers and text-encoder respectively to ablate concepts. However, we note that our work focuses primarily on first understanding how knowledge is stored in text-to-image models and subsequently using this information to design a closed-form editing method for editing concepts.

2.4 Mechanistically Understanding and Editing Multimodal Language Models

Multimodal Large Language Models.. We consider a MLLM to be a model that takes an image and text as input, and generates a text output [5]. Over the last year, such

models have made tremendous advances in tasks like VQA and image captioning, including BLIP [139], BLIP-2 [138], Instruct-BLIP [53], LLaVA [148, 149], Flamingo [8] and multi-modal Phi-2 (from the Bunny repo) [94]. These MLLM can broadly be categorized into two families based on how their visual information is integrated into the language model: (i) by embedding the vision encoder's output into each layer of the language model with a cross-attention layer (e.g., Flamingo, BLIP) or, (ii) by mapping the vision encoder's output into "visual tokens" in the language model's input space (i.e. alongside the text tokens) via a projection layer (e.g., LLaVA, Bunny). Both families are widely used, however, the projection layer family has recently shown stronger performance on popular benchmark [94, 148, 149]. We, therefore, focus our study of information storage and transfer on this model family.

Interpretability of MLLMs. A well-established arm of model interpretability examines the relationship between a model's performance and its internals. A range of recent works have studied the internal mechanisms of information storage [165, 167, 243] and transfer [83, 263] in MLLM. However, to the best of our knowledge, only a few works [217, 225] have studied the interpretability of MLLM, with none specifically investigating the relationship between a model's outputs and its internal states. [217], for example, designs an interactive interface to visualize the attention maps in an MLLM, while [225] explores the shortcomings of the CLIP vision encoder in MLLM. Neither consider the influence of both vision and text inputs on model internals or offer causal insights, as our work does. Our model editing approach which targets the projection layer MLLM family, is complemented by [46], who propose baselines for inserting information into the cross-attention layer MLLM family.

2.5 Mechanistically Understanding and Unlocking Zero-Shot Capabilities in Vision Transformers

Several studies attempt to elucidate model predictions by analyzing either a subset of input example through heatmaps [157, 205, 212, 219] or a subset of training examples [120, 183, 190]. Nevertheless, empirical evidence suggests that these approaches are often unreliable in real-world scenarios [22, 115]. These methods do not interpret model predictions in relation to the model's internal mechanisms, which is essential for gaining a deeper understanding of the reliability of model outputs.

Internal Mechanisms of Vision Models: Our work is closely related to the studies by [77] and [238], both of which analyze vanilla ViTs in terms of their components and interpret them using either CLIP text encoders or pretrained ImageNet heads. Like these studies, our research can be situated within the emerging field of representation engineering [271] and mechanistic interpretability [29, 32]. Other works [24, 86, 178] focus on interpreting individual neurons to understand vision models' internal mechanisms. However, these methods often fail to break down the model's output into its subcomponents, which is crucial for understanding model reliability. [206] examine the direct effect of model weights on output, but do not study the fine-grained role of these components in building the final image representation. [17] focus on expressing CNN representations as a sum of contributions from input regions via masking.

Interpreting models using CLIP: Many recent works utilize CLIP [191] to interpret models via text. [170] align model representations to CLIP space with a linear layer, but it is limited to only the final representation and can not be applied to model components. [177] annotate individual neurons in CNNs via CLIP, but their method cannot be extended

easily to high-dimensional component vectors. Our method is related to model stitching in which one representation space is interpreted in terms of another by training a map between two spaces [18, 134].

2.6 Mechanistic Circuits for Extractive Question-Answering

Circuit Based Interpretability in Language Models. With the advent of language models, a lot of recent works have focused on a mechanistic understanding of language models [87, 144, 164, 166, 232]. One of the primary benefit of transformer based language models is that the final logit representation can be decomposed as a sum of individual model components [68]. Based on this decomposition, one can extract task-specific causal sub-graphs (i.e., circuits) of internal model components in language models. Early works have extracted such circuits for indirect-object identification [244], greater-than operation [91] and more recently for entity-tracking [188]. Recently, there has been an increasing focus on the practical aspects of mechanistic interpretability such as refusal mediation [11, 267] or safety in general [272]. Circuits can also be constructed as sub-graphs of neurons in the language model, but it often comes with increased complexity of interpretation [67]. In our paper, we focus on extracting circuits where the nodes are different architectural components such as attention-heads, layers or MLPs.

Applications in Context-Augmented Question-Answering. With the advent of retrieval-augmented generation [81, 135] language models have been increasingly used for real-world Question-Answering (QA) tasks. One of the primary enhancement of context-augmented QA lies in the ability to provide reliable grounding (i.e., attribution) in the context for the generated answer [102, 113, 137, 257]. In the recent times, there have been a large set of works which improve LLM responses by reducing hallucinations and improving grounding in the input context [14, 252, 257, 265]. Our paper tests the ability of the mechanistic

insights from circuits towards performing these applications.

2.7 Improving Compositionality in Multimodal Models

2.7.1 Compositionality in CLIP

While CLIP models [191] are renowned for their robust zero-shot classification, recent research [60, 223] has exposed their limitations in visio-linguistic reasoning. In contrast, recent studies have demonstrated that text-to-image models [41, 49, 125, 136] outperform CLIP in reasoning tasks. These models in fact leverage scores computed from the diffusion objective. We note that while [187] use score-distillation sampling for text to 3D generation, ours is the first work to adapt the formulation as a regularizer and improve compositional abilities in CLIP.

2.7.2 Compositionality in Text-to-Image Models

Compositionality in text-to-image models refers to the ability of a model to accurately capture the correct compositions of objects, their corresponding attributes, and the relationships between objects described in a given prompt. [103] introduced a benchmark designed to evaluate compositionality in text-to-image models, highlighting the limitations of models when handling compositional prompts. The benchmark employs disentangled BLIP-Visual Question Answering (VQA) as a metric for assessing image compositional quality. The VQA score assesses how accurately an image captures the compositional elements described in the prompt by utilizing a vision-language model. This metrics demonstrates a closer correlation with human judgment compared to metrics like CLIP-Score [97]. The authors also proposed a fine-tuning baseline to enhance compositionality in these models. Alternatively, compositionality issues can be addressed at inference by modifying

cross-attention maps using hand-crafted loss functions and bounding boxes derived from a language model [1, 39, 73, 143, 150, 175, 245]. However, [103] showed that data-driven fine-tuning is more effective for improving compositionality.

Chapter 3: **Interpreting Test-Time Predictions With Influence Functions**

3.1 Introduction

Recently, there has been a rapid and significant success in applying machine learning methods to a wide range of applications including vision [220], natural language processing [204], medicine [158], finance [147], etc. In sensitive applications such as medicine, we would like to explain test-time model predictions to humans. An important question is : *why the model makes a certain prediction for a particular test sample.* One way to address this is to trace back model predictions to its training data. More specifically, one can ask which training samples were the most influential ones for a given test prediction.

Influence functions [51] from robust statistics measure the dependency of optimal model parameters on training samples. Previously [119] used first-order approximations of influence functions to estimate how much model parameters would change if a training point was up-weighted by an infinitesimal amount. Such an approximation can be used to identify most influential training samples in a test prediction. Moreover, this approximation is similar to the leave-one-out re-training, thus the first-order influence function proposed in [119] bypasses the expensive process of repeated re-training the model to find influential training samples in a test-time prediction.

In some applications, one may want to understand how model parameters would change when large groups of training samples are removed from the training set. This could be useful to identify groups of training data which drive the decision for a particular test prediction. As shown in [118], finding influential groups can be useful in real-world applications such as diagnosing batch effects [253], apportioning credit between different data sources [13], understanding effects of different demographic groups [42] or in a multi-party learning setting [92]. [118] approximates the group influence by sum of first-order individual influences over training samples in the considered group. However, removal of a large group from training can lead to a large perturbation to model parameters. Therefore, influence functions based on first-order approximations may not be accurate in this setup. Moreover, approximating the group influence by adding individual sample influences ignores possible cross correlations that may exist among samples in the group.

In this paper, we relax the first-order approximations of current influence functions and study how second-order approximations can be used to capture model changes when a potentially large group of training samples is up-weighted. Considering a training set $\mathscr{S}$ and a group $\mathscr{U} \subset \mathscr{S}$, existing first-order approximations of the group influence function [118] can be written as the sum of first-order influences of individual points. That is,

$$\mathscr{I}^{(1)}(\mathscr{U}) = \sum_{i=1}^{|\mathscr{U}|} \mathscr{I}_i^{(1)}$$

where $\mathscr{I}^{(1)}(\mathscr{U})$ is the first-order group influence function and $\mathscr{I}_i^{(1)}$ is the first-order influence for the i^{th} sample in $\mathscr{U}$. On the other hand, our proposed second-order group influence function has the following form:

$$\mathscr{I}^{(2)}(\mathscr{U}) = \mathscr{I}^{(1)}(\mathscr{U}) + \mathscr{I}^{'}(\mathscr{U})$$

where $\mathscr{I}^{'}(\mathscr{U})$ captures informative cross-dependencies among samples in the group and is a function of gradient vectors and the Hessian matrix evaluated at the optimal model parameters. We present a more precise statement of this result in Theorem 1. We note that the proposed second-order influence function can be computed efficiently even for large models. We discuss its computational complexity in Section 3.4.

Our analysis shows that the proposed second-order influence function captures model changes efficiently even when the size of the groups are relatively large or the changes to the model parameters are significant as in the case of groups with similar properties. For example, in an MNIST classification problem using logistic regression, when 50% of the training samples are removed, the correlation between the ground truth estimate and second-order influence values improves by over 55% when compared to the existing first-order influence values. We note that higher-order influence functions have been used in statistics [108] for point and interval estimates of non-linear functionals in parameteric, semi-parametric and non-parametric models. However, to the best of our knowledge, this is the first time, higher-order influence functions are used for the interpretability task in the machine learning community.

Similar to [119] and [118], our main results for the second-order influence functions hold for linear prediction models where the underlying optimization is convex. However, we also additionally explore effectiveness of both first-order and second-order group influence functions in the case of deep neural networks. We observe that none of the methods provide good estimates of the ground-truth influence across different groups [1]. In summary, we make the following contributions:

- We propose second-order group influence functions that consider cross dependencies

[1] Note that experiments of [119] focus only on the most influential individual training samples.

among the samples in the considered group.

- Through several experiments over linear models, across different sizes and types of groups, we show that the second-order influence estimates have higher correlations with the ground truth when compared to the first-order ones, especially when the changes to the underlying model is relatively large.
- We also show that our proposed second-order group influence function can be used to improve the selection of the most influential training group.

3.2 Background

We consider the classical supervised learning problem setup, where the task is to learn a function h (also called the hypothesis) mapping from the input space $\mathscr{X}$ to an output space $\mathscr{Y}$. We denote the input-output pair as $\{x, y\}$. We assume that our learning algorithm is given training examples $\mathscr{S} := \{z_i = (x_i, y_i)\}_{i=1}^m$ drawn i.i.d from some unknown distribution $\mathscr{P}$. Let Θ be the space of the parameters of considered hypothesis class. The goal is to select model parameters ` to minimize the empirical risk as follows:

$$\min_{\theta \in \Theta} \; L_{\emptyset}(\theta) := \frac{1}{|\mathscr{S}|} \sum_{z \in \mathscr{S}} \ell(h_\theta(z)), \tag{3.1}$$

where $|\mathscr{S}| = m$, denotes the cardinality of the training set, the subscript $\emptyset$ indicates that the whole set $\mathscr{S}$ is used in training and ℓ is the associated loss function. We refer to the optimal parameters computed by the above optimization as θ^*. Let $\nabla_\theta L_{\emptyset}(\theta)$ and $H_{\theta^*} = \nabla^2_\theta L_{\emptyset}(\theta)$ be the gradient and the Hessian of the loss function, respectively.

First, we discuss the case where we want to compute the effect of an *individual* training sample z on optimal model parameters as well as the test predictions made by the model. The effect or influence of a training sample on the model parameters could characterized by

removing that particular training sample and retraining the model again as follows:

$$\theta^*_{\{z\}} = \arg\min_{\grave{}\in\Theta} \; L_{\{z\}}(\theta) = \frac{1}{|\mathscr{S}|-1} \sum_{z_i \neq z} \ell(h_{\grave{}}(z_i)) \tag{3.2}$$

Then, we can compute the change in model parameters as $\Delta_{\grave{}} = \theta^*_{\{z\}} - \theta^*$, due to removal of a training point z. However, re-training the model for every such training sample is expensive when $|\mathscr{S}|$ is large. Influence functions based on first-order approximations introduced by [50, 51] was used by [119] to approximate this change. Up-weighting a training point z by an infinitesimal amount ε leads to a new optimal model parameters, $\theta^{\varepsilon}_{\{z\}}$, obtained by solving the following optimization problem:

$$\theta^{\varepsilon}_{\{z\}} = \arg\min_{\theta\in\Theta} \; \frac{1}{|\mathscr{S}|} \sum_{z\in\mathscr{S}} \ell(h_\theta(z_i)) + \varepsilon\ell(h_\theta(z)) \tag{3.3}$$

Removing a point z is similar to up-weighting its corresponding weight by $\varepsilon = -\frac{1}{|\mathscr{S}|}$. The main idea used by [119] is to approximate $\theta^*_{\{z\}}$ by minimizing the first-order Taylor series approximation around θ^*. Following the classical result by [51], the change in the model parameters θ^* on up-weighting z can be approximated by the influence function [119] denoted by $\mathscr{I}$:

$$\mathscr{I}(z) = \frac{d\theta^{\varepsilon}_{\{z\}}}{d\varepsilon}|_{\varepsilon=0} = -H^{-1}_{\theta^*}\nabla_\theta \ell\left(h_{\theta^*}(z)\right) \tag{3.4}$$

A detailed proof can be found in [119]. Using the given formulation, we can track the change with respect to any function of θ^*. The change in the test loss for a particular test point z_t when a training point z is up-weighted can be approximated as a closed form expression:

$$\mathscr{I}(z, z_t) = -\nabla_\theta \ell(h_{\theta^*}(z_t))^T H^{-1}_{\theta^*} \nabla_\theta \ell(h_{\theta^*}(z)) \tag{3.5}$$

This result is based on the assumption [119] that the loss function $L(\theta)$ is strictly convex in

the model parameters θ and the Hessian H_{θ^*} is therefore positive-definite. This approximation is very similar to forming a quadratic approximation around the optimal parameters θ^* and taking a single Newton step. However explicitly computing H_{θ^*} and it's inverse $H_{\theta^*}^{-1}$ is not required. Using the Hessian-vector product rule [185] influence functions can be computed efficiently.

3.3 Group Influence Function

Our goal in this section is to understand how the model parameters would change if a particular group of samples was up-weighted from the training set. However, up-weighting a group can lead to large perturbations to the training data distribution and therefore model parameters, which does not follow the small perturbation assumption of the first-order influence functions. In this section, we extend influence functions using second-order approximations to better capture changes in model parameters due to up-weighting a group of training samples.

The empirical risk minimization (ERM) when we remove $\mathscr{U}$ samples from training can be written as:

$$L_{\mathscr{U}}(\) = \frac{1}{|\mathscr{S}|-|\mathscr{U}|} \sum_{z \in \mathscr{S} \setminus \mathscr{U}} \ell(h_\theta(z)) \tag{3.6}$$

To approximate how optimal solution of this optimization is related to θ^*, we study the effect of *up-weighting* a group of training samples on model parameters. Note that in this case, updated weights should still be a valid distribution, i.e. if a group of training samples has been up-weighted, the rest of samples should be down-weighted to preserve the sum to one constraint of weights in the ERM formulation. In the individual influence function case (when the size of the group is one), up-weighting a sample by ε leads to down-weighting other samples by $\varepsilon/(m-1)$ whose effect can be neglected similar to the formulation of

[119]. In our formulation for the group influence function, we assume that the weights of samples in the set $\mathscr{U}$ has been up-weighted all by ε and use $p = \frac{|\mathscr{U}|}{|\mathscr{S}|}$ to denote the fraction of up-weighted training samples. This leads to a down-weighting of the rest of training samples by $\tilde{\varepsilon} = \frac{|\mathscr{U}|}{|\mathscr{S}|-|\mathscr{U}|}\varepsilon$, to preserve the empirical weight distributioxn of the training data. This is also important in order to have a fair comparison with the ground-truth leave-out-retraining estimates. Therefore, the resulting ERM can be written as:

$$\theta^{\varepsilon}_{\mathscr{U}} = \arg\min_{\theta} L^{\varepsilon}_{\mathscr{U}}(\theta)$$

where

$$L^{\varepsilon}_{\mathscr{U}}(\theta) = \frac{1}{|\mathscr{S}|}\Big(\sum_{z\in\mathscr{S}\setminus\mathscr{U}} (1-\tilde{\varepsilon})\ell(h_\theta(z)) + \sum_{z\in\mathscr{U}} (1+\varepsilon)\ell(h_\theta(z))\Big). \tag{3.7}$$

Or equivalently In the above formulation, if $\varepsilon = 0$ we get the original loss function $L_\emptyset(\grave{})$ (where none of the training samples are removed) and if $\varepsilon = -1$, we get the loss function $L_{\mathscr{U}}(\grave{})$ (where samples are removed from training).

Let $\theta^{\varepsilon}_{\mathscr{U}}$ denote the optimal parameters for $L^{\varepsilon}_{\mathscr{U}}$ minimization. Essentially we are concerned about the change in the model parameters (i.e. $\Delta\theta = \theta^{\varepsilon}_{\mathscr{U}} - \theta^*$) when each training sample in a group of size $|\mathscr{U}|$ is upweighted by a factor of ε. The key step of the derivation is to expand $\theta^{\varepsilon}_{\mathscr{U}}$ around θ^* (the minimizer of $L^0_{\mathscr{U}}(\theta)$, or $L_\emptyset(\theta)$) with respect to the order of ε, the upweighting parameter. In order to do that, we use the perturbation theory [15] to expand $\grave{}^{\varepsilon}_{\mathscr{U}}$ around θ^*.

Frequently used in quantum mechanics and also in other areas of physics such as parti-

cle physics, condensed matter and atomic physics, perturbation theory finds approximate solution to a problem ($\theta^{\varepsilon}_{\mathscr{U}}$) by starting from the exact solution of a closely related and simpler problem (θ^*). As ε gets smaller and smaller, these higher order terms become less significant. However, for large model perturbations (such as the case of group influence functions), using higher-order terms can reduce approximation errors significantly. The following perturbation series forms the core of our derivation for second-order influence functions:

$$\theta^{\varepsilon}_{\mathscr{U}} - \theta^* = \mathscr{O}(\varepsilon)\theta^{(1)} + \mathscr{O}(\varepsilon^2)\theta^{(2)} + \mathscr{O}(\varepsilon^3)\theta^{(3)} + \cdots \tag{3.8}$$

where $\theta^{(1)}$ characterizes the first-order (in ε) perturbation vector of model parameters while $\theta^{(2)}$ is the second-order (in ε) model perturbation vector. We hide the dependencies of these perturbation vectors to constants (such as $|U|$) with the $\mathscr{O}(.)$ notation.

In the case of computing influence of individual points, as shown by [119], the scaling of $\theta^{(1)}$ is in the order of $1/|\mathscr{S}|$ while the scaling of the second-order coefficient is $1/|\mathscr{S}|^2$ which is very small when $\mathscr{S}$ is large. Thus, in this case, the second-order term can be ignored. In the case of computing the group influence, the second-order coefficient is in the order of $|\mathscr{U}|^2/|\mathscr{S}|^2$, which can be large when the size of $\mathscr{U}$ is large. Thus, in our definition of the group influence function, both $\theta^{(1)}$ and $\theta^{(2)}$ are taken into account.

The first-order group influence function (denoted by $\mathscr{I}^{(1)}$) when all the samples in a group $\mathscr{U}$ are up-weighted by ε can be defined as:

$$\mathscr{I}^{(1)}(\mathscr{U}) = \frac{\partial \theta^{\varepsilon}_{\mathscr{U}}}{\partial \varepsilon}|_{\varepsilon=0}$$

$$= \frac{\partial(\theta^* + \mathscr{O}(\varepsilon)\theta^{(1)} + \mathscr{O}(\varepsilon^2)\theta^{(2)})}{\partial \varepsilon}|_{\varepsilon=0} = \theta^{(1)}$$

To capture the dependency of the terms in $\mathscr{O}(\varepsilon^2)$, on the group influence function, we define $\mathscr{I}^{'}$ as follows:

$$\mathscr{I}^{'}(\mathscr{U}) = \frac{\partial^2 \theta^{\varepsilon}_{\mathscr{U}}}{\partial \varepsilon^2}|_{\varepsilon=0}$$

$$= \frac{\partial^2(\theta^* + \mathscr{O}(\varepsilon)\theta^{(1)} + \mathscr{O}(\varepsilon^2)\theta^{(2)})}{\partial \varepsilon^2}|_{\varepsilon=0} = \theta^{(2)}$$

Although one can consider even higher-order terms, in this paper, we restrict our derivations up to the second-order approximations of the group influence function. We now state our main result in the following theorem:

Theorem 1. *If the third-derivative of the loss function at θ^* is sufficiently small, the second-order group influence function (denoted by $\mathscr{I}^{(2)}(\mathscr{U})$) when all samples in a group $\mathscr{U}$ are up-weighted by ε is:*

$$\mathscr{I}^{(2)}(\mathscr{U}) = \mathscr{I}^{(1)}(\mathscr{U}) + \mathscr{I}^{'}(\mathscr{U}) \tag{3.9}$$

where:

$$\mathscr{I}^{(1)}(\mathscr{U}) = -\frac{1}{1-p}\frac{1}{|\mathscr{S}|}H^{-1}_{\theta^*}\sum_{z\in\mathscr{U}}\nabla\ell(h_{\theta^*}(z))$$

and

$$\mathscr{I}^{'}(\mathscr{U}) =$$
$$\frac{p}{1-p}\Big(I - (\nabla^2 L_{\emptyset}(\theta^*))^{-1}\frac{1}{|\mathscr{U}|}\sum_{z\in\mathscr{U}}\nabla^2\ell(h_{\theta^*}(z))\Big)\theta^{(1)}$$

This result is based on the assumption that the third-order derivatives of the loss function at θ^* is small. For the quadratic loss, the third-order derivatives of the loss are zero. Our experiments with the cross-entropy loss function indicates that this assumption approximately holds for the classification problem as well. Below, we present a concise sketch of

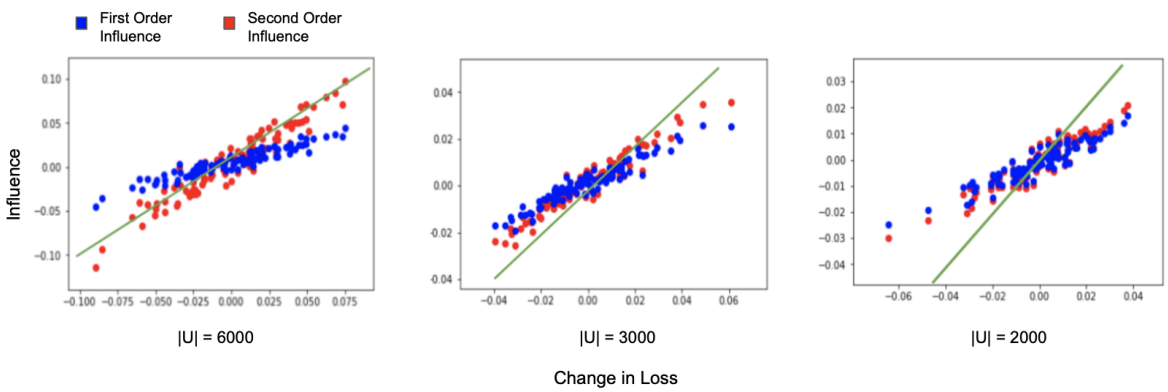

Figure 3.1: Comparison of first-order and second-order group influences in case of synthetic dataset with 10,000 samples using logistic regression for a mis-classified test point. Across different sizes of groups which were randomly selected, it can be observed that the second-order influence values are more correlated with the ground truth than that of the first-order ones. The green line highlights the $y = x$ line.

this result.

Proof Sketch. We now derive $\theta^{(1)}$ and $\theta^{(2)}$ to be used in the second order group influence function $\mathscr{I}^{(2)}(\mathscr{U})$. As $\theta^{\varepsilon}_{\mathscr{U}}$ is the optimal parameter set for the interpolated loss function $L^{\varepsilon}_{\mathscr{U}}(\theta)$, due to the first-order stationary condition, we have the following equality:

$$\begin{aligned} 0 = \nabla L^{\varepsilon}_{\mathscr{U}}(\theta^{\varepsilon}_{\mathscr{U}}) = &\nabla L_{\emptyset}(\theta^{\varepsilon}_{\mathscr{U}}) \\ &+ \frac{1}{|\mathscr{S}|}(-\tilde{\varepsilon} \sum_{z \in \mathscr{S} \setminus \mathscr{U}} + \varepsilon \sum_{z \in \mathscr{U}}) \nabla \ell(h_{\theta^{\varepsilon}_{\mathscr{U}}}(z)) \end{aligned} \tag{3.10}$$

The main idea is to use Taylor series for expanding $\nabla L_{\emptyset}(\theta^{\varepsilon}_{\mathscr{U}})$ around θ^* along with the perturbation series defined in Equation (3.8) and compare the terms of the same order in ε:

$$\nabla L_{\emptyset}(\theta^{\varepsilon}_{\mathscr{U}}) = \nabla L_{\emptyset}(\theta^*) + \nabla^2 L_{\emptyset}(\theta^*)(\theta^{\varepsilon}_{\mathscr{U}} - \theta^*) + \ldots \tag{3.11}$$

Similarly, we expand $\nabla \ell(h_{\theta^{\varepsilon}_{\mathscr{U}}}(z))$ around θ^* using Taylor series expansion. To derive $\theta^{(1)}$

we compared terms with the coefficient of $\mathscr{O}(\varepsilon)$ in Equation (3.10) and for $\theta^{(2)}$ we compared terms with coefficient $\mathscr{O}(\varepsilon^2)$. Based on this, $\theta^{(1)}$ can be written in the following way:

$$\theta^{(1)} = -\frac{1}{1-p}\frac{1}{|\mathscr{S}|}H_{\theta^*}^{-1}\sum_{z\in\mathscr{U}}\nabla\ell(h_{\theta^*}(z)) \tag{3.12}$$

We expand Equation(3.10) and compare the terms with coefficient $\mathscr{O}(\varepsilon)$:

$$\begin{aligned}
&\varepsilon\nabla^2 L_{\emptyset}(\theta^*)\theta^{(1)} \\
&= \frac{1}{|\mathscr{S}|}(\tilde{\varepsilon}\sum_{z\in\mathscr{S}\setminus\mathscr{U}} - \varepsilon\sum_{z\in\mathscr{U}})\nabla\ell(h_{\theta^*}(z)) \\
&= \tilde{\varepsilon}\nabla L_{\emptyset}(\theta^*) - \frac{1}{|\mathscr{S}|}(\tilde{\varepsilon}+\varepsilon)\sum_{z\in\mathscr{U}}\nabla\ell(h_{\theta^*}(z)) \\
&= -\frac{1}{|\mathscr{S}|}(\tilde{\varepsilon}+\varepsilon)\sum_{z\in\mathscr{U}}\nabla\ell(h_{\theta^*}(z)) \\
&= -\frac{1}{|\mathscr{S}|}\frac{1}{(1-p)}\varepsilon\sum_{z\in\mathscr{U}}\nabla\ell(h_{\theta^*}(z))
\end{aligned} \tag{3.13}$$

$\theta^{(1)}$ is the first-order approximation of group influence function and can be denoted by $\mathscr{I}^{(1)}$. Note that our first-order approximation of group influence function $\mathscr{I}^{(1)}$, is slightly different from [118] with an additional $1-p$ in the denominator. For $\theta^{(2)}$ we compare the terms with coefficients of the same order of $\mathscr{O}(\varepsilon^2)$ in Equation (3.10):

$$\begin{aligned}
&\varepsilon^2\nabla^2 L_{\emptyset}(\theta^*)\theta^{(2)} + \frac{1}{2}L_{\emptyset}'''(\theta^*)[\varepsilon\theta^{(1)},\varepsilon\theta^{(1)},I] \\
&+ \frac{1}{|\mathscr{S}|}(-\tilde{\varepsilon}\sum_{\mathscr{S}\setminus\mathscr{U}} + \varepsilon\sum_{\mathscr{U}})\nabla^2\ell(h_{\theta^*}(z))(\varepsilon\theta^{(1)}) \\
&= 0
\end{aligned} \tag{3.14}$$

For the $\theta^{(2)}$ term, we ignore the third-order term $\frac{1}{2}L_{\emptyset}'''(\theta^*)[\varepsilon\theta^{(1)},\varepsilon\theta^{(1)},I]$ due to it being small. Now we substitute the value of $\tilde{\varepsilon}$ and equate the terms with coefficient in the order of

$\mathcal{O}(\varepsilon^2)$:

$$\nabla^2 L_{\emptyset}(\theta^*)\theta^{(2)} = \frac{|\mathscr{U}|}{|\mathscr{S}|-|\mathscr{U}|}\Big(\frac{1}{|\mathscr{S}|}\sum_{z\in\mathscr{S}}\nabla^2\ell(h_{\theta^*}(z)) - \frac{1}{|\mathscr{U}|}\sum_{z\in\mathscr{U}}\nabla^2\ell(h_{\theta^*}(z))\Big)\theta^{(1)} \tag{3.15}$$

Rearranging the Equation (3.15), we get the same identity as $\mathscr{I}^{'}$ in Theorem (1). □

It can be observed that the additional term ($\mathscr{I}^{'}$) in our second-order approximation captures cross-dependencies among the samples in $\mathscr{U}$ through a function of gradients and Hessians of the loss function at the optimal model parameters. This makes the second-order group influence function to be more informative when training samples are correlated. In Section (3.5), we empirically show that the addition of $\mathscr{I}^{'}$ improves correlation with the ground truth influence as well.

For tracking the change in the test loss for a particular test point z_t when a group $\mathscr{U}$ is removed, we use the chain rule to compute the influence score as follows:

$$\mathscr{I}^{(2)}(\mathscr{U}, z_t) = \nabla\ell(h_{\theta^*}(z_t))^T\left(\mathscr{I}^{(1)}(\mathscr{U}) + \mathscr{I}^{'}(\mathscr{U})\right) \tag{3.16}$$

Our second-order approximation of group influence function consists of a first-order term that is similar to the one proposed in [118] with an additional scaling term $1/(1-p)$. This scaling is due to the fact that our formulation preserves the empirical weight distribution constraint in ERM, which is essential when a large group is up-weighted. The second-order influence function has an additional term $\mathscr{I}^{'}$ that is directly proportional to p and captures

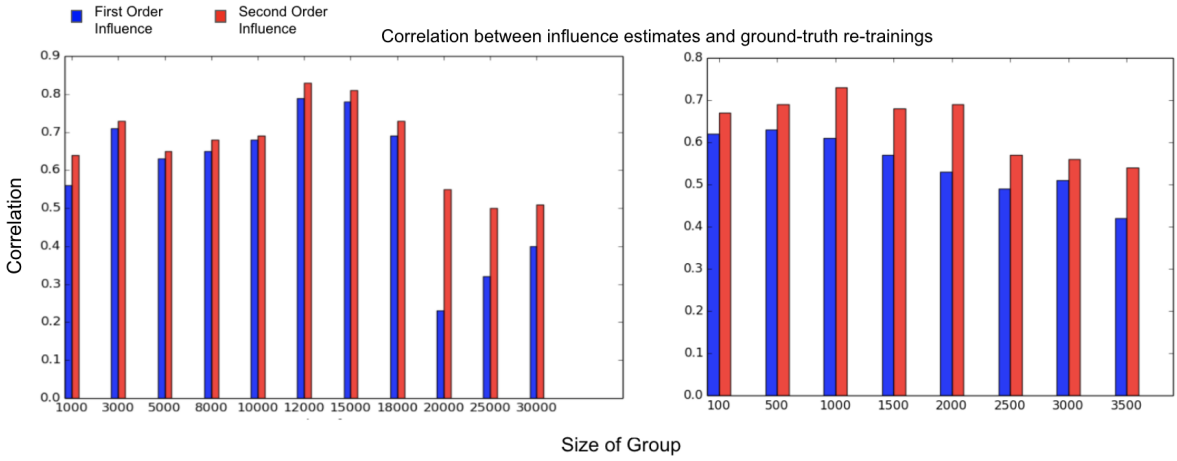

Figure 3.2: Group size vs the correlation with the ground truth on MNIST for logistic regression with random groups (left panel) and *coherent* groups (right panel).

large perturbations to the model parameters more effectively.

3.4 Computational Complexity

For models with a relatively large number of parameters, computing the inverse of the Hessian $H_{\theta^*}^{-1}$ can be expensive and is of the order of $O(n^3)$. However, computing the Hessian-vector product [185] is relatively computationally inexpensive. In our experiments similar to [40, 118, 119], we used conjugate gradients (a second-order optimization technique) [209] to compute the inverse Hessian-vector product which uses a Hessian-vector product in the routine thus saving the expense for inverting the Hessian directly. The proposed second-order group influence function can be computed similarly to the first-order group influence functions with only an additional step of Hessian-vector product.

3.5 Experiments

3.5.1 Setup

Our goal through the experiments is to observe if the second-order approximations of group influence functions improve the correlation with the ground truth estimate across different settings. We compare the computed second-order group influence score with the ground truth influence (which is computed by leave-k-out retraining for a group with size k). Our metric for evaluation is the Pearson correlation which measures how linearly the computed influence and the actual ground truth estimate are related. We perform our experiments primarily on logistic regression where the group influence function is well-defined. Additionally we also check the accuracy of first-order and second-order group influence functions in case of neural networks.

3.5.2 Datasets

To understand the accuracy of both first-order and second-order group influence functions on linear models we use two datasets. In our first experiments, we use a synthetic dataset along with logistic regression. The synthetic dataset has 10,000 points drawn from a Gaussian distribution, consisting of 5 features and 2 classes. The details for the synthetic data can be found in the Appendix. The second set of experiments are done with the standard handwritten digits database MNIST [131] which consists of 10 classes of different digits. For understanding how group influence functions behave in case of the neural networks we use the MNIST dataset. For each of the two datasets, we pick random groups as well *coherent* groups as in [118] with sizes ranging from 1.6% to 60% of the entire training points. The computed group influence was primarily investigated for a test-point which was misclassified by the model. A detailed description of how the groups were selected in our

experiments is given in the Appendix. For the optimal group selection we used a synthetic dataset consisting of 20,000 training points consisting of 5 features in the form of 4 isotropic Gaussian blobs.

3.5.3 Observations and Analysis

Linear Models

For logistic regression, the general observation for the randomly selected groups was that the second-order group influence function improves the correlation with the ground truth estimates across different group sizes in both the synthetic dataset as well as MNIST. For the synthetic dataset, in Figure (3.1), it can be observed that the approximation provided by the second-order group influence function is fairly close to the ground truth when a large fraction of the training data (60 %) is removed. In such cases of large group sizes, the first-order approximation of group influence function is relatively inaccurate and far from the ground truth influence. This observation is consistent with the small perturbation assumption of first-order influence functions. However, in cases of smaller group sizes, although the second-order approximation improves over existing first-order group influence function, the gain in correlation is small. In case of MNIST, the observation was similar where the gain in correlation was significant when the size of the considered group was large. For e.g. it can seen in Figure (3.2), that when more than 36% of the samples were removed, the gain in correlation is almost always more than 40%. While the improvement in correlation for larger group sizes is consistent with our theory that the second-order approximation is effective in the case of large changes to the model, the gain in correlation is non-monotonic with respect to the group sizes. For groups of small size, selected uniformly at random, the model parameters do not change significantly and the second-order approximation improves only marginally over the existing first-order approximation. However, when a *coherent*

group (a group having training examples from the same class) of even a relatively small size is removed, the perturbation to the model is larger (as the model parameters can change significantly in a particular direction) than if a random group is removed. In such settings, we observe that even for small group sizes, the second-order approximation consistently improves the correlation with the ground-truth significantly (Figure (3.2)). For *coherent* groups, across different group sizes of the MNIST dataset, we observed an improvement in correlation when the second-order approximation was used. Across different group sizes we observed that the gain in correlation is at least 15%. These observations (shown in Figure (3.2)) reinforces our theory that the second-order (or rather higher-order) approximations of influence functions are particularly effective when the perturbation or changes in the model parameters are significantly large. The second-order approximation of the influence function could thus be used over existing first-order approximations in practical purposes such as understanding the behaviour of training groups with similar properties (e.g. demographic groups) on model predictions, without the need to actually retrain the model again.

Neural Networks

In case of neural networks, the Hessian is not positive semi-definite in general, which violates the assumptions of influence functions. Previously [119] regularized the Hessian in the form of $H_{\theta^*} + \lambda I$, and had shown that for the top few influential training points (not groups) and for a given test point, the correlation with the ground truth influence is still satisfactory, if not highly significant. However, how influence functions behave in the case of groups, is a topic not yet well explored. For MNIST, we used a regularized Hessian with a value of $\lambda = 0.01$ and conducted experiments for a relatively simple two hidden layered feed-forward network with sigmoid activations for both first-order and second-order group influence functions. The general observation was that both existing first and proposed

second-order group influence functions underestimate the ground truth influence values across different group sizes, leading to a non-significant correlation. The corresponding Figure can be referred to in the Appendix. However, we observed that while the second-order influence values still suffer from the underestimation issue, they improve the correlation marginally across different group sizes. This observation was consistent in cases of both random and *coherent* group selections.

3.6 Conclusion for Second-Order Group Influence Functions

In this paper, we proposed second-order group influence functions for approximating model changes when a group from the training set is removed. Empirically, in the case of linear models, across different group sizes and types, we showed that the second-order influence has a higher correlation with ground truth values compared to the first-order ones and is more effective than existing first-order approximations. Our observation was that the second-order influence is significantly informative when the changes to the underlying model is relatively large. We showed that the proposed second-order group influence function can be practically used in conjunction with optimization techniques to select the most influential group in the training set for a particular test prediction. For non-linear models such as deep neural networks, we observed that both first-order and second-order influence functions lead to a non-significant correlation with the ground truth across different group sizes (although the correction values for the second-order method was marginally better). Developing accurate group influence functions for neural networks or training neural networks to have improved influence functions and also extending group influence functions to the transfer learning setting as in [40] are among directions for future work.

3.7 Influence Functions in Deep Learning

In machine learning, influence functions [51] can be used to estimate the change in model parameters when the empirical weight distribution of the training samples is perturbed infinitesimally. This approximation is cheaper to compute compared to the expensive process of repeatedly re-training the model to retrieve the exact parameter changes. Influence functions could thus be used to understand the effect of removing an individual training point (or, groups of training samples) on the model predictions at the test-time. Leveraging a first-order Taylor's approximation of the loss function, [119] has shown that a (first-order) influence function, computed using the gradient and the Hessian of the loss function, can be useful to interpret machine learning models, fix mislabelled training samples and create data poisoning attacks.

Influence functions are in general well-defined and studied for models such as logistic regression [119], where the underlying loss-function is convex. For convex loss functions, influence functions are also accurate even when the model perturbations are fairly large (e.g. in the group influence case [118]). However, when the convexity assumption of the underlying loss function is violated, which is the case in deep learning, the behaviour of influence functions is not well understood and is still an open area of research. With recent advances in computer vision [220], natural language processing [204], high-stakes applications such as medicine [158], it has become particularly important to interpret deep model predictions. This makes it critical to understand influence functions in the context of deep learning, which is the main focus of our paper.

Despite their non-convexity, it is sometimes believed that influence functions would work for deep networks. The excellent work of [119] successfully demonstrated one example of influence estimation for a deep network, a small (2600 parameters), "all-convolutional"

network. To the best of our knowledge, this is the one of the *few* cases for deep networks where influence estimation has been shown to work. A question of key importance to practitioners then arises: for what other classes of deep networks does influence estimation work? In this work, we provide a comprehensive study of this question and find a pessimistic answer: *influence estimation is quite fragile for a variety of deep networks.*

In the case of deep networks, several factors might have an impact on influence estimates: (i) due to non-convexity of the loss function, different initializations of the perturbed model can lead to significantly different model parameters (with approximately similar loss values); (ii) even if the initialization of the model is fixed, the curvature values of the network (i.e. eigenvalues of the Hessian matrix) at optimal model parameters might be very large in very deep networks, leading to a significant Taylor's approximation error of the loss function and thus resulting in poor influence estimates; (iii) for large neural networks, computing the exact inverse-Hessian Vector product, required in computation of influence estimates, can be computationally very expensive. Thus, one needs to use approximate inverse-Hessian Vector product techniques which might be erroneous; resulting in low quality influence estimates; and finally (iv) different architectures can have different loss landscape geometries near the optimal model parameters, leading to varying influence estimates.

In this paper, we study aforementioned issues of using influence functions in deep learning through an extensive experimental study on progressively-growing complex models and datasets. We first start our analysis with a case study of a small neural network for the Iris dataset where the exact Hessian matrix can be computed. We then progressively increase the complexity of the network and analyse a CNN architecture (depth of 6) trained on 10% of MNIST dataset, similar to [119]. Next, we evaluate the accuracy of influence estimates for more complex deep architectures (e.g. ResNets) trained on MNIST and CIFAR-10. Finally, we compute influence estimates on the ImageNet dataset using ResNet-50.

We make the following observations through our analysis:

- We find that the network depth and width have a strong impact on influence estimates. In particular, we show that influence estimates are fairly accurate when the network is shallow, while for deeper models, influence estimates are often erroneous. We attribute this partially to the increasing curvature values of the network as the depth increases.
- We observe that the weight decay regularization is important to obtain high quality influence estimates in certain architectures and datasets.
- We show that the inverse-Hessian Vector product approximation techniques such as stochastic estimation [4] are erroneous, especially when the network is deep. This can contribute to the low quality of influence estimates in deep models.
- We observe that the choice of test-point has a significant impact on the quality of influence estimates, across different datasets and architectures.
- In very large-scale datasets such as ImageNet, we have found that even ground-truth influence estimates (obtained by leave-one-out re-training) can be inaccurate and noisy partially due to the model's training and convergence.

These results highlight sensitivity of current influence functions in deep learning and call for developing robust influence estimators to be used in large-scale machine learning applications.

3.8 Basics of Influence Function

Consider h to be a function parameterized by θ which maps from an input feature space $\mathscr{X}$ to an output space denoted by $\mathscr{Y}$. The training samples are denoted by the set $\mathscr{S} = \{z_i : (x_i, y_i)\}_{i=1}^{n}$, while the loss function is represented by $\ell(h_\theta(z))$ for a particular training

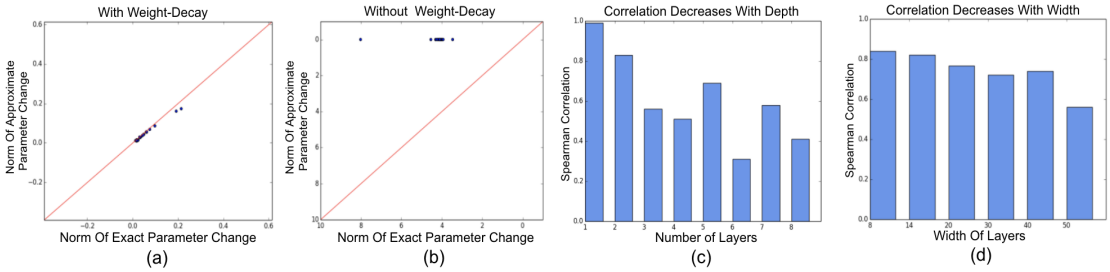

Figure 3.3: Iris dataset experimental results - (a,b) Comparison of norm of parameter changes computed with influence function vs re-training; (a) trained with weight-decay; (b) trained without weight-decay. (c) Spearman correlation vs. network depth. (d) Spearman correlation vs. network width.

example z. The standard empirical risk minimization solves the following optimization problem:

$$\theta^* = \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \ell(h_\theta(z_i)). \tag{3.17}$$

Up-weighting a training example z by an infinitesimal amount ε leads to a new set of model parameters denoted by $\theta^{\varepsilon}_{\{z\}}$. This set of new model parameters $\theta^{\varepsilon}_{\{z\}}$ is obtained by solving:

$$\theta^{\varepsilon}_{\{z\}} = \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \ell(h_\theta(z_i)) + \varepsilon \ell(h_\theta(z)). \tag{3.18}$$

Removing a training point z is similar to up-weighting its corresponding weight by $\varepsilon = -1/n$ in Equation(3.18). The main idea used by [119] is to approximate $\theta^{\varepsilon}_{\{z\}}$ by the first-order Taylor series expansion around the optimal model parameters represented by θ^*, which leads to:

$$\theta^{\varepsilon}_{\{z\}} \approx \theta^* - \varepsilon H^{-1}_{\theta^*} \nabla_\theta \ell(h_{\theta^*}(z)), \tag{3.19}$$

where H_{θ^*} represents the Hessian with respect to model parameters θ^*. Following the classical result of [51], the change in the model parameters ($\Delta\theta = \theta^{\varepsilon}_{\{z\}} - \theta^*$) on up-weighting the training example z can be approximated by the influence function ($\mathscr{I}(z)$) as follows:

$$\mathscr{I}(z) = \frac{d\theta^{\varepsilon}_{\{z\}}}{d\varepsilon}|_{\varepsilon=0} = -H^{-1}_{\theta^*}\nabla_{\theta}\ell\left(h_{\theta^*}(z)\right). \tag{3.20}$$

The change in the loss value for a particular test point z_t when a training point z is up-weighted can be approximated as a closed form expression by the chain rule [119]:

$$\mathscr{I}(z, z_t) = -\nabla\ell(h_{\theta^*}(z_t))^T H^{-1}_{\theta^*}\nabla\ell(h_{\theta^*}(z)). \tag{3.21}$$

$\mathscr{I}(z, z_t)/n$ is approximately the change in the loss for the test-sample z_t when a training sample z is removed from the training set. This result is, however, based on the assumption that the underlying loss function is strictly convex in the model parameters θ and the Hessian H_{θ^*} is a positive-definite matrix [119]. For large models, inverting the exact Hessian H_{θ^*} is expensive. In such cases, the inverse-Hessian Vector product can be computed efficiently with a combination of Hessian-vector product [185] and optimization techniques (see Appendix for details).

3.9 What Can Go Wrong for Influence Functions In Deep Learning?

First-order influence functions [119] assume that the underlying loss function is convex and the change in model parameters is small when the empirical weight distribution of the training data is infinitesimally perturbed. In essence, this denotes the Taylor's gap in Equation (3.19) to be small for an accurate influence estimate. However in the case of non-convex loss functions, this assumption is *not* generally true. Empirically, we find that the Taylor's gap is strongly affected by common hyper-parameters for deep networks. For example, in Fig. (3.3)-(a,b), we find that for networks trained without a weight-decay regularization on Iris, the Taylor's gap is large resulting in low quality influence estimates. In a similar vein, when the network depth and width is significantly large (i.e. the over-parameterized regime), the Taylor's gap increases and substantially degrades the quality of

influence estimates (Fig. (3.4)). Empirically this increase in Taylor's gap strongly correlates with the curvature values of the loss function evaluated at the optimal model parameters as observed in Fig. (3.4-(b)).

Further complications may arise for larger models, where influence estimations in such settings require an additional approximation to compute the inverse-Hessian vector product. Nonetheless, we observe in Fig. (3.4)-(a), that on Iris this approximation has only a marginal impact on the influence estimation. These results show that that network architecture, hyper-parameters, and loss curvatures are significant factors for proper influence estimations. In the next section, we discuss these issues in details through controlled experiments on datasets and models of increasing complexity.

3.10 Experiments

Datasets: We first study the behaviour of influence functions in a small Iris dataset [9], where the exact Hessian can be computed. Further, we progressively increase the complexity of the model and datasets: we use small MNIST [119] to evaluate the accuracy of influence functions in a small CNN architecture with a depth of 6. Next, we study influence functions on modern deep architectures trained on the standard MNIST [131] and CIFAR-10 [124] datasets. Finally, to understand how influence functions scale to large datasets, we use ImageNet [55] to compute the influence estimates.

Evaluation Metrics: We evaluate the accuracy of influence estimates at a given test point z_t using both Pearson [116] and Spearman rank-order correlation with the ground-truth (obtained by re-training the model) across a set of training points. Most of the existing interpretability methods desire that influential examples are ranked in the correct order of their importance [84]. Therefore, to evaluate the accuracy of influence estimates, Spearman

correlation is often a better choice.

3.10.1 Understanding Influence Functions when the Exact Hessian Can be Computed

Setup: Computing influence estimates with the exact Hessian has certain advantages in our study: a) it bypasses inverse-Hessian Vector product approximation techniques which induce errors in computing influence estimates. Thus, we can compare influence estimates computed with exact vs. approximate inverse-Hessian Vector products to quantify this type of error; b) The deviation of the parameters computed with the influence function from the exact parameters can be computed exactly. This information can be useful to further quantify the error incurred by (first-order) influence estimates in the non-convex setup. However, computations of the exact Hessian matrix and its inverse are only computationally feasible for models with small number of parameters. Thus, we use the Iris dataset along with a small feed-forward neural network to analyse the behaviour of influence function computed with the exact Hessian in a non-convex setup. We train models to convergence for 60k iterations with full-batch gradient descent. To obtain the ground-truth estimates, we re-train the models for 7.5k steps, starting from the optimal model parameters. For our analysis, we choose the test-point with the maximum loss and evaluate the accuracy of influence estimates with the ground-truth amongst of the top 16.6% of the training points. Through our experiments with the exact Hessian, we answer some relevant questions related to how properties of the network such as depth, width and regularizers (e.g. weight-decay) affect the influence estimates.

The Effect of Weight-Decay: One of the simpler and common regularization techniques used to train neural networks is weight-decay regularization. In particular, a term $\lambda \|\theta\|_2^2$, penalizing the scaled norm of the model parameters is added to the objective function,

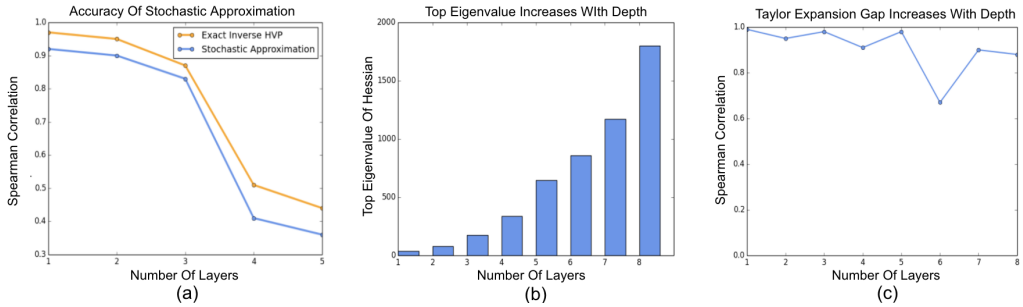

Figure 3.4: Iris dataset experimental results; (a) Spearman correlation of influence estimates with the ground-truth estimates computed with stochastic estimation vs. exact inverse-Hessian vector product. (b) Top eigenvalue of the Hessian vs. the network depth. (c) Spearman correlation between the norm of parameter changes computed with influence function vs. re-training.

during training, where λ is a hyperparameter which needs to be tuned. We train a simple feed-forward network[2] with and without weight-decay regularization. For the network trained with weight-decay, we observe a Spearman correlation of 0.97 between the influence estimates and the ground-truth estimates. In comparison, for the network trained without a weight-decay regularization, the Spearman correlation estimates decrease to 0.508. In this case, we notice that the Hessian matrix is singular, thus a damping factor of 0.001 is added to the Hessian matrix, to make it invertible. To further understand the reason for this decrease in the quality of influence estimates, we compare the following metric across all training examples: a) Norm of the model parameter changes computed by re-training; b) Norm of the model parameter changes computed using the influence function (i.e. $\|H_{\theta^*}^{-1}\nabla\ell(z_i)\|_2 \quad \forall i \in [1,n]$) (Fig. 3.3-(a,b)). We observe that when the network is trained without weight-decay, changes in model parameters computed with the influence function have a significantly larger deviation from those computed using re-training. This essentially suggests that the gap in Taylor expansion, using (first-order) influence estimates is large, when the model is trained without weight-decay. We observe similar results with smooth activation functions such as tanh (see the Appendix for details).

[2]With width of 5, depth of 1 and ReLU activations

The Effect Of Network Depth: From Fig. 3.3-(c), we see that network depth has a dramatic effect on the quality of influence estimates. For example, when the depth of the network is increased to 8, we notice a significant decrease in the Spearman correlation estimates. To further our understanding about the decrease in the quality of influence estimates when the network is deeper, we compute the gap in the approximation between the ground-truth parameter changes (computed by re-training) and the approximate parameter changes (computed using the influence function). To quantify the error gap, we compute the Spearman correlation estimates between the norm of true and approximate parameter changes across the top 16.6% of the influential examples. We find that with increasing depth, the Spearman correlation estimates between the norm of the true and approximate parameter changes decrease. From Fig. 3.4-(c), we see that the approximation error gap is particularly large when the depth of the network is more than 5. We also notice a consistent increase in the curvature of the loss function (Fig. 3.4-(b)), as the network becomes deeper. This possibly suggests that the curvature information of the network can be an upper bound in the approximation error gap between the true parameters and the ones computed using the influence function. Even in case of non-smooth activation functions like ReLU, we have a similar observation. (see the Appendix for more details).

The Effect Of Network Width: To see the effect of the network width on the quality of influence estimates, we evaluate the influence estimates for a feed-forward network of constant depth, by progressively increasing its width. From Fig. 3.3-(d), we observe that with an increase in network width, the Spearman correlation decreases consistently. For example, we find that the Spearman correlation decreases from 0.82 to 0.56, when the width of the network is increased from 8 to 50. This observation suggests that over-parameterizing a network by increasing its width has a strong impact in the quality of influence estimates.

The Effect of Stochastic Estimation on inverse-Hessian Vector Product: For large deep

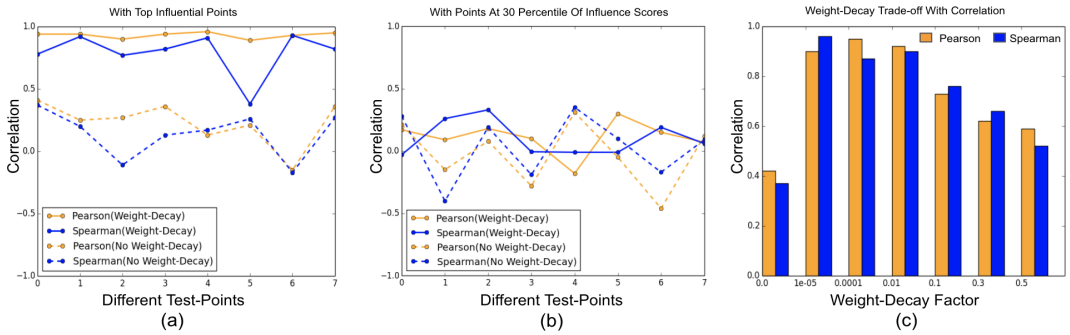

Figure 3.5: Experiments on small MNIST using a CNN architecture. (a) Estimation of influence function with and without weight decay on (a) the top influential points, (b) training points at 30^{th} percentile of influence score distribution. (c) Correlation vs the weight decay factor (evaluated on the top influential points).

networks, the inverse-Hessian Vector product is computed using stochastic estimation[3], as the exact Hessian matrix cannot be computed and inverted. To understand the effectiveness of stochastic approximation, we compute the influence estimates with both the exact Hessian and stochastic estimation. We observe that across different network depths, the influence estimates computed with stochastic estimation have a marginally lower Spearman correlation when compared to the ones computed with the exact Hessian. From Fig. 3.4-(a), we find that the error in the approximation is more, when the network is deeper.

3.10.2 Understanding Influence Functions in Shallow CNN Architectures

Setup: In this section, we perform a case study using a CNN architecture[3] on the small MNIST dataset (i.e. 10% of MNIST); a similar setup used in [119]. To assess the accuracy of influence estimates, we select a set of test-points with high test-losses computed at the optimal model parameters. For each of the test points, we select 100 training samples with the highest influence scores and compute the ground-truth influence by re-training the model.

[3]The model has 2600 parameters and is trained for 500k iterations to reach convergence with the optimal model parameters `*. The ground-truth estimates are obtained by re-training the models from the optimal parameter set `* for 30k iterations. When trained with a weight-decay, a regularization factor of 0.001 is used.

We also select 100 training points with influence scores at the 30^{th} percentile of the entire influence score distribution. These training points have low influence scores and a lower variance in their scores when compared to the top influential points. The model is trained with and without weight-decay regularization.

When trained with a weight-decay and evaluated based on the top influential points, we find that the correlation estimates are consistently significant (Fig. 3.5-(a)). This is consistent with the results reported in [119]. However, when the evaluation is done with the set of training samples at the 30^{th} percentile of the influence score distribution, the correlation estimates decrease significantly (Fig. 3.5-(b)). This shows that influence estimates of only the top influential points are precise when compared to ground-truth re-trainings. Furthermore, without the weight-decay regularization, influence estimates in both cases are poor across all the test-points (Fig. 3.5-(a,b)).

To further understand the impact of weight-decay on influence estimates, we train the network with different weight-decay regularization factors. From Fig. 3.5-(c), we see that the selection of weight-decay factor is important in getting high-quality influence estimates. For this specific CNN architecture, we notice that the correlations start decreasing when the weight-decay factor is greater than 0.01. Moreover, from Fig. 3.5-(a,b), we find that the selection of test-point also has a strong impact on the quality of influence estimates. For example, when the network is trained with weight-decay and the influence estimates are computed for top influential training points, we notice that the Spearman correlation estimates range from 0.92 to 0.38 across different test-points and have a high variance.

These results show that despite some successful applications of influence functions in this non-convex setup, as reported in [119], their performances are very sensitive to hyper-parameters of the experiment as well as to the training procedure. In the next two sections,

we assess the quality of influence estimates on more complex architectures and datasets including MNIST, CIFAR-10 and ImageNet. In particular, we desire to understand, if the insights gained from experiments on smaller networks can be generalized to more complex networks and datasets.

3.10.3 Understanding Influence Functions in Deep Architectures

Setup: In this section, we evaluate the accuracy of influence estimates using MNIST and CIFAR-10 datasets across different network architectures including small CNN[119], LeNet [132], ResNets [93], and VGGNets [211][4]. To compute influence estimates, we choose two test points for each architecture: a) the test-point with the highest loss, and b) the test-point at the 50^{th} percentile of the losses of all test points. For each of these two test points, we select the top 40 influential training samples and compute the correlation of their influence estimates with the ground-truth estimates. To compute the ground-truth influence estimates, we follow the strategy of [119], where we re-train the models from optimal parameters for 6% of the steps used for training the optimal model. When the networks are trained with a weight-decay regularization, we use a constant weight-decay factor of 0.001 across all the architectures (see Appendix for more details).

Results On MNIST: From Table 3.1, we observe that for the test-point with the highest loss, the influence estimates in the small CNN and LeNet architectures (trained with the weight-decay regularization) have high qualities. These networks have 2.6k and 44k parameters, respectively, and are relatively smaller and less deep than the other networks used in our experimental setup. As the depth of the network increases, we observe a consistent decrease in quality of influence estimates. For the test-point with a loss at the 50^{th} percentile of test-point losses, we observe that influence estimates *only* in the small CNN architecture

[4]For CIFAR-10, evaluations on small CNN have not been performed due to the poor test accuracy.

Dataset	MNIST						CIFAR-10					
	A (With Decay)		**B (With Decay)**		**A (Without Decay)**		**A (With Decay)**		**B (With Decay)**		**A (Without Decay)**	
Architecture	P	S	P	S	P	S	P	S	P	S	P	S
Small CNN	**0.95**	**0.87**	**0.92**	**0.82**	**0.41**	0.35	-	-	-	-	-	-
LeNet	**0.83**	**0.51**	0.28	0.29	0.18	0.12	**0.81**	**0.69**	0.45	0.46	0.19	0.09
VGG13	0.34	**0.44**	0.29	0.18	0.38	0.31	**0.67**	**0.63**	**0.66**	**0.63**	**0.79**	**0.73**
VGG14	0.32	0.26	0.28	0.22	0.21	0.11	**0.61**	**0.59**	**0.49**	0.41	**0.75**	**0.64**
ResNet18	**0.49**	0.26	0.39	0.35	0.14	0.11	**0.64**	**0.42**	0.25	0.26	**0.72**	**0.69**
ResNet50	0.24	0.22	0.29	0.19	0.08	0.13	**0.46**	0.36	0.24	0.09	0.32	0.14

Table 3.1: Correlation estimates on MNIST And CIFAR-10 ; A=Test-point with highest loss; B=Test-point at the 50^{th} percentile of test-loss spectrum; P=Pearson correlation; S=Spearman correlation

have good qualities.

Results On CIFAR-10: For CIFAR-10, across all architectures trained with the weight-decay regularization, we observe that the correlation estimates for the test-point with the highest loss are highly significant. For example, the correlation estimates are above 0.6 for a majority of the network architectures. However, for the test-point evaluated at the 50^{th} percentile of the loss, the correlations decrease marginally across most of the architectures. We find that on CIFAR-10, even architectures trained without weight-decay regularization have highly significant correlation estimates when evaluated with the test-point which incurs the highest loss.

In case of MNIST, we have found that in shallow networks, the influence estimates are fairly accurate while for deeper networks, the quality of influence estimates decrease. For CIFAR-10, although the influence estimates are significant, we found that the correlations are marginally lower in deeper networks such as ResNet-50. The improved quality of influence estimates in CIFAR-10 can be attributed to the fact that for a similar depth, architectures trained on CIFAR-10 are less over-parameterized compared to architectures trained on MNIST. Note that, in Section 3.10.1, where the exact Hessian matrix can be computed,

we observed that over-parameterization decreases the quality of influence estimates. From Table(3.1), we also observed that the selection of test-point has a significant impact on the quality of influence estimates. Furthermore, we noticed large variations in the quality of influence estimates across different architectures. In general we found that influence estimates for small CNN and LeNet are reasonably accurate, while for ResNet-50, the quality of estimates decrease across both MNIST and CIFAR-10. Precise reasons for these variations are difficult to establish. We hypothesize that it can be due to the following factors: (i) Different architectures trained on different datasets have contrasting characteristics of loss landscapes at the optimal parameters which can have an impact on influence estimates. (ii) The weight-decay factor may need to be set differently in various architectures, to obtain high quality influence estimates.

3.10.4 Is Scaling Influence Estimates To ImageNet Possible?

The application of influence functions to ImageNet scale models provides an appealing yet challenging opportunity. It is appealing because, if successful, it opens a range of applications to large-scale image models, including interpretability, robustness, data poisoning, and uncertainty estimation. It is challenging for a number of reasons. Notable among these is the high computational cost of training and re-training, which limits the number of ground truth evaluations. In addition, all of the previously discussed difficulties in influence estimations still remain, including (i) non-convexity of the loss, (ii) selection of scaling and damping hyperparameters in the stochastic estimation of the Hessian, and (iii) the lack of convergence of the model parameters. The scale of ImageNet raises additional questions about the feasibility of leave-one-out retraining as the ground truth estimator. Given that there are 1.2M images in the training set, *is it even possible that the removal of one image can significantly alter the model?* In other words, we question whether or not reliable ground truth estimates may be obtained through leave-one-out re-training at this scale.

To illustrate this, we conduct an additional influence estimation on ImageNet. After training an initial model to 92.302% top5 test accuracy, we select two test points at random, calculate influence over the entire training set, and then select the top 50 points by their influences as candidates for re-training. We then use the re-training procedure suggested by [119], which starts leave-one-out re-training from the parameter set obtained after the initial training. We re-train for an additional 2 epochs, approximately 5% of the original training time, and calculate the correlations. We observe that for both test points, both Pearson and Spearman correlations are very low (less than 0.15, see details in the Appendix).

In our experiments, we observe high variability among ground-truth estimates obtained by re-training the model (see the appendix for details). We conjecture that this may be partially due to the fact that the original model has not be fully converged. To study this, we train the original model with *all* training points for an additional 2 epochs and measure the change in the test loss. We find that the overall top5 test accuracy has improved slightly to 92.336 % (+0.034) and the loss for one of the considered test points has decreased by relatively a significant amount of 0.679. However, the loss for the other point has increased slightly by 0.066. Such changes in loss values can therefore out-power the effect of leave-one-out re-training procedure. Second, we calculate the 2-norm of the weight gradients, which should be close to zero near an optimal point, and compare it to a standard pre-trained ImageNet ResNet-50 model as a baseline. We find these norms to be 20.18 and and 15.89, respectively, showing our model has similar weight gradient norm to the baseline. Although these norms are relatively small given that there are 25.5M parameters, further re-training the model still changes loss values for some samples significantly, making the ground-truth estimates noisy. We suggest that one way to obtain reliable ground-truth influence estimates in such large models can be through assessing the influence of a group of samples, rather than a single one.

3.11 Conclusion for Influence Functions in Deep Learning

In this paper, we present a comprehensive analysis of the successes and failures of influence functions in deep learning. Through our experiments on datasets including Iris, MNIST, CIFAR-10, ImageNet and architectures including LeNet, VGGNets, ResNets, we have demonstrated that influence functions in deep learning are fragile in general. We have shown that several factors such as the weight-decay, depth and width of the network, the network architecture, stochastic approximation and the selection of test points, all have strong effects in the quality of influence estimates. In general, we have observed that influence estimates are fairly accurate in shallow architectures such as small CNN[119] and LeNet, while in very deep and wide architectures such as ResNet-50, the estimates are often erroneous. Additionally, we have scaled up influence computations to the ImageNet scale, where we have observed influence estimates are highly imprecise. These results call for developing robust influence estimators in the non-convex setups of deep learning.

Chapter 4: **Automatically Designing Difficult Few-Shot Benchmarks for Model Reliability**

4.1 Introduction

Few-shot classification is the ability to distinguish between a set of novel classes when given only a few labelled training examples of each class [71, 129]. This holds potential across many real-world applications – from robots that can identify new objects [195], to drug discovery pipelines that can predict the properties of new molecules [218]. A few-shot *image* classifier is given a few labelled training images of the new object classes, called the support set. Once the classifier has adapted to this support set, it is then evaluated on novel test images of those classes, called the query set. Together, the support and query set is called a task.

Recent years have seen rapid progress in few-shot image classification [74, 122, 141, 192, 213, 256], however, current top-performing methods display a wide range in performance over different tasks at test time [2, 76]. On META-DATASET (MD), a widely-used few-shot classification benchmark [230], state-of-the-art classifiers obtain accuracies as low as 22% on some individual tasks though their average task accuracy is $>$55% (see Fig 4.1). Few works have undertaken a detailed examination of these 'difficult' tasks, yet they remain critical to interrogate for both future algorithmic development and the safety and reliability

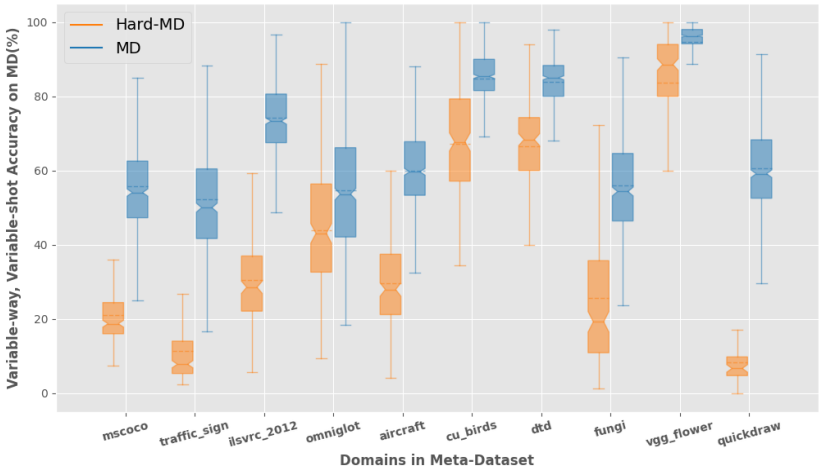

Figure 4.1: **A state-of-the-art method [100] performs consistently worse on difficult tasks in the MD split (HARD-MD) of HARD-META-DATASET++ compared to tasks in META-DATASET (MD) across all 10 MD sub-datasets.** The method uses ViT-S initialized with self-supervised DINO weights and is further meta-trained with ProtoNets on MD's ilsvrc_2012 split.

of deployed systems.

This paper aims to gain a more nuanced understanding of these 'difficult' tasks and the limitations of current methods. We define a difficult task as one on which a few-shot classifier performs poorly on the task's query set, after being adapted to its support set.

Current methods for finding supports sets that lead to poor query performance rely on greedy search-based algorithms [2]. These approaches, however, incur a high computational cost when sampling for a large numbers of tasks, and for tasks with large support sets, as are common (and best) practices in few-shot evaluation protocols. As a result, the study of difficult tasks has been limited to small-scale datasets which lacks the challenging examples and the setup of large benchmarks such as META-DATASET.

To address this, we develop a general and computationally efficient algorithm called FAST-DIFFSEL to extract difficult tasks from any large-scale dataset. Given a (meta-)trained few-shot classifier, a query set, and a search pool of support images, we formulate a con-

strained combinatorial optimization problem which learns a selection weight for each image in the pool such that the loss on the query set is maximised. The top-k (i.e. most difficult) images per class are then extracted into a support set and paired with the query set to form a difficult task. This optimization can be repeated to obtain any number of difficult tasks. In practice, we find that FASTDIFFSEL is at least 20x faster than existing greedy search-based algorithms [2], with greater gains as the support pools and support set sizes increase.

We leverage the scalability of FASTDIFFSEL to extract difficult tasks from a wide range of large-scale vision datasets including META-DATASET [230], OBJECTNET [20], CURE-OR [222] and ORBIT [163] and collect these tasks into a new testing set called HARD-META-DATASET++ (HARD-MD++). The addition of datasets beyond META-DATASET is motivated by their real-world nature and that they provide image annotations of quality variations (e.g. object occluded, blurred, poorly framed) to enable future research into *why* a task is difficult. We provide early insights into this question in our analyses.

We stress test an extensive suite of state-of-the-art few-shot classification methods on HARD-MD++, cross-validating the difficulty of our extracted tasks across these top-performing methods. In Fig 4.1, we show one such method [100] performing consistently worse on the META-DATASET test split in HARD-MD++ than on the original MD test split across all 10 sub-datasets. In Sec. 4.5, we find that this trend holds true across a wide-range of few-shot classification methods.

We release HARD-MD++ along with a broad set of baselines to drive future research in methods that are robust to even the most difficult tasks. In summary, our contributions are the following:

- FASTDIFFSEL, an efficient algorithm to extract difficult tasks from any large-scale vision dataset.

- HARD-META-DATASET++, a new test-only few-shot classification benchmark composed of difficult tasks extracted from the widely-used few-shot classification benchmark META-DATASET and other large-scale real-world datasets: OBJECTNET, CUREOR and ORBIT.
- Extensive stress testing and novel empirical insights for a wide range of few-shot classification methods, including transfer- and meta-learning based approaches on HARD-MD++.

4.2 Few-Shot Classification: Preliminaries and Notations

A few-shot classification task is typically composed of (i) a support set $\mathscr{S}$ which contains a few labelled examples from a set of N classes (e.g., k_j examples for each class index $j \in [1,N]$) and (ii) a query set $\mathscr{Q}$ which contains a disjoint set of test examples for each of those N classes (e.g., q_j examples for each class $j \in [1,N]$). Given a trained base model f_θ, the goal in few-shot classification is to use $\mathscr{S}$ to adapt f_θ to the task such that the model can then make predictions on the unseen examples in $\mathscr{Q}$. This is typically done for many tasks which are randomly sampled from a given dataset $\mathscr{D} = \{(x_i, y_i)\}_{i=1}^{T}$ consisting of T examples and J classes in total, where x is the input image and y is the corresponding class label. In few-shot sampling, tasks are constructed by first sampling a set of N classes from the larger set of J classes and subsequently sampling the support and query set. Tasks are typically referred to as N-way, k-shot tasks if $k_j = k, \forall j$. However, if the number of classes N varies across tasks and the number of shots per class varies across the N classes ($k_j, \forall j \in [1,N]$), they are referred to as variable-way, variable-shot tasks. Usually the number of query examples is kept fixed across all the classes (i.e. $q_j = q, \forall j \in [1,N]$).

Base model training. The underlying model f_θ is typically trained using one of two approaches: (i) meta-learning [75, 133, 213] which involves training the model in an episodic

manner on tasks sampled from a base training dataset, or (ii) transfer learning [44, 100, 224] which involves first pre-training a feature extractor on a large base dataset in an supervised or self-supervised manner, and then fine-tuning the final classification layer (or the entire model) at test time for each new test task.

4.3 FastDiffSel: An Efficient Algorithm to Select Difficult Support Sets

The first step to understanding the limitations of current few-shot classification methods is to study the difficult tasks on which they fail or perform poorly. We define a difficult task as one for which a given few-shot classifier, after being adapted to its support set, performs significantly worse on its query set compared to the mean query performance over all tasks. Finding a support set from a given search pool that leads to poor performance on a given query set for the purposes of study, however, has combinatorial complexity. Current solutions have therefore turned to greedy search-based algorithms [2, 76], however, the computational cost quickly becomes infeasible for larger search pools and support set sizes, thus limiting study to small-scale datasets (e.g. CIFAR-FS [27], mini-ImageNet [239]). We specifically address this limitation by proposing a fast and general optimization-based algorithm – FastDiffSel which offers a speedup of at least 20-25x over greedy search-based algorithms, allowing the extraction of difficult few-shot tasks to be scaled to a wide array of large-scale vision datasets.

4.3.1 Proposed Method

Overview. We present FastDiffSel, that can sample a difficult few-shot classification task in a deterministic way. The key intuition of our approach is use a model's loss on the task's query set (i.e. after the model has been adapted to the support set) as a proxy for the difficulty of the task. This follows [12, 57] which show that a task's query loss is an effective

Algorithm 1 FASTDIFFSEL: Efficient algorithm for extracting a difficult few-shot task

Input: $\mathscr{Q}$: task query set; N: number of classes (way), $\mathscr{P}$: search pool for extracting task support set; f_θ: (meta-)trained base model; M: size of search pool; α: learning rate; $\{k_j\}_{j=1}^{N}$: set containing number of shots per class.

$\mathbf{w} \leftarrow \text{CONCAT}(\mathbf{w}_j) \quad \forall j \in [1,N]$ ▷ Concatenate randomly initialized vectors for each class

for j in N **do**

$c_j \leftarrow \sum_{i=1}^{|\mathscr{P}_j|} w_j^i f_\theta(x_j) / \sum_{i=1}^{|\mathscr{P}_j|} w_j^i$ ▷ Compute weighted class prototypes for each class

end for

$c \leftarrow [c_1, ..., c_N]$ ▷ Store the weighted prototypes for each class

$L \leftarrow \text{PROTO-LOSS}(\mathscr{Q}, c, f_\theta)$ ▷ Compute prototypical loss [213]

$L.\text{BACKWARD}(\mathbf{w})$ ▷ Compute gradients with respect to selection weights

$\mathbf{w} \leftarrow \mathbf{w} + \alpha \nabla_{\mathbf{w}} L(\mathbf{w})$ ▷ Gradient ascent for updating weights

$\mathbf{w}_j \leftarrow \text{PROJ}(\mathbf{w}_j, k_j) \quad \forall j \in [1,N]$ ▷ Projection step per class

$s_j \leftarrow \text{EXTRACT}(k_j, \mathbf{w}_j, \mathscr{P}) \quad \forall j \in [1,N]$ ▷ Extract k_j examples with the highest weights

$\mathscr{S} \leftarrow \text{CONCAT}(s_j) \quad \forall j \in [1,N]$ ▷ Obtain the final difficult support set S

surrogate for task difficulty and is also simple to compute. Given a model, a fixed query set and a pool of examples, we cast support set extraction as a constrained optimization problem and learn a selection weight for each example in the pool such that the loss on the query set is maximized. We can then construct a difficult support set by drawing the examples with the highest selection weights. The extracted support set is then paired with the fixed query set to form the difficult task. We can repeat this optimization for any number of query sets to extract any number of difficult tasks.

Formally, given a dataset $\mathscr{D}$, we first sample N unique classes and a query set $\mathscr{Q}$. Here $Q = \{(x_r, y_r)\}_{r=1}^{N \times q}$, where x is the input image and y is the class label. Let $\mathscr{D}' \subset \mathscr{D}$ denote a sub-dataset containing examples from only the N sampled classes and let $\mathscr{P} = \mathscr{D}' - \mathscr{Q}$ denote the set of examples from $\mathscr{D}'$ without the query set $\mathscr{Q}$. Allowing $\mathscr{P}$ to be the search pool from which the difficult support set will be extracted, the goal of the extraction algorithm is to find a support set $\mathscr{S} \subset \mathscr{P}$ such that the loss on the query set $\mathscr{Q}$ is maximized after the base model f_θ has been adapted on $\mathscr{S}$. To this end, we assume selection weights $\mathbf{w} \in \mathbb{R}^M$, where w^i is associated with the i^{th} example in $\mathscr{P}$ and $M = |\mathscr{P}|$ denotes the cardinality of $\mathscr{P}$. The optimization objective is to learn the selection weights $\mathbf{w}$ which result in the maximal loss for query set $\mathscr{Q}$ with a sparsity constraint on the weights. Formally:

$$\max_{\mathbf{w}} \sum_{r=1}^{N\times q} \ell((x_r, y_r), P, \mathbf{w}, f_\theta) \tag{4.1}$$

$$s.t. \quad w^i \in \{0,1\}, \qquad \forall i \in [1,M]$$
$$\|\mathbf{w}_j\|_0 \leq k_j, \quad \forall j \in [1,N]$$

where f_θ is the base model trained with either meta-learning or supervised learning, ℓ is the loss after adapting f_θ on $\mathscr{P}$ where each of its examples are weighted by $\mathbf{w}$, and $\mathbf{w}_j$ is the selection weight vector corresponding to the j^{th} class. Here $\mathbf{w} = \mathbf{w}_1 \oplus \mathbf{w}_2 \oplus ... \oplus \mathbf{w}_N$ and w^i_j is the selection weight for the i^{th} example in the weight vector $\mathbf{w}_j$ for the j^{th} class. Note $\mathbf{w}$ are the only learnable parameters. The optimization constraints ensure that each selection weight is either 0 or 1, and that a maximum of k_j examples are selected for each j^{th} class. Different approaches can be used to adapt f_θ [45, 196]. In our work, we adopt a ProtoNets adaptation [213] as it is highly efficient and has no learnable parameters. This approach computes a mean embedding for each class, with the loss based on the Euclidean distance between a query image embedding to each of the class prototypes.

We solve Eq. (4.1) in two steps: (i) first, we take 1 gradient ascent step on the selection weights $\mathbf{w}$ to obtain $\hat{\mathbf{w}}$; (ii) second, we project the selection weight vector of each class $\hat{\mathbf{w}}_j$ to the ℓ_0 norm ball to obtain the final selection weights $\bar{\mathbf{w}}_j$ ($\forall j \in [1,N]$). In practice, (ii) is known to be difficult to solve as it is NP-hard and the l_0 norm constraint is non-convex [33, 34, 61, 174]. We, therefore, relax the ℓ_0 norm to an ℓ_1 norm to make the constraint convex following [61] which shows that the ℓ_1 norm relaxation gives effective sparse solutions. The projection step with ℓ_1 relaxation for the j^{th} class can be formalized

as follows:

$$\min_{\mathbf{w}_j} \frac{1}{2}\|\mathbf{w}_j - \hat{\mathbf{w}}_j\|_2^2$$
$$s.t. \;\; \|\mathbf{w}_j\|_1 \leq k_j \tag{4.2}$$

We solve the dual form of the above projection step via Lagrange multipliers [28] to obtain the optimal sparse weight vector $\bar{\mathbf{w}}_j$:

$$\bar{\mathbf{w}}_j = \arg\max_{\lambda_j \geq 0} \min_{\mathbf{w}_j} \underbrace{\frac{1}{2}\|\mathbf{w}_j - \hat{\mathbf{w}}_j\|_2^2 + \lambda_j(\|\mathbf{w}_j\|_1 - k_j)}_{g(\lambda_j, \mathbf{w}_j)} \tag{4.3}$$

where λ_j is the Lagrange multiplier corresponding to the projection step for the j^{th} class. In practice, we solve the projection step per class to ensure at least a few-examples per class are selected, and we find that 1 projection step per class is sufficient to learn appropriate selection weights. After 1 gradient ascent and 1 projection step (i.e. weight vectors have been learned), we select the examples from $\mathscr{P}$ to include in the difficult support set. For each j^{th} class, we sort the final selection weight vector $\bar{\mathbf{w}}_j$ in descending order and extract the k_j examples from $\mathscr{P}$ which have the highest weights.

The pseudo-code for FASTDIFFSEL is shown in Algorithm 6 and a detailed derivation of the steps for solving the optimization along with the associated hyperparameters can be found in Appendix.

Computational complexity. The key advantage of FASTDIFFSEL is that it does not require an iterative exhaustive search through the search pool to select a difficult support set. Consider selecting a task with N classes and k support examples per class from a dataset $\mathscr{D}'$, where each class has d examples on average. The greedy algorithm in [2] runs

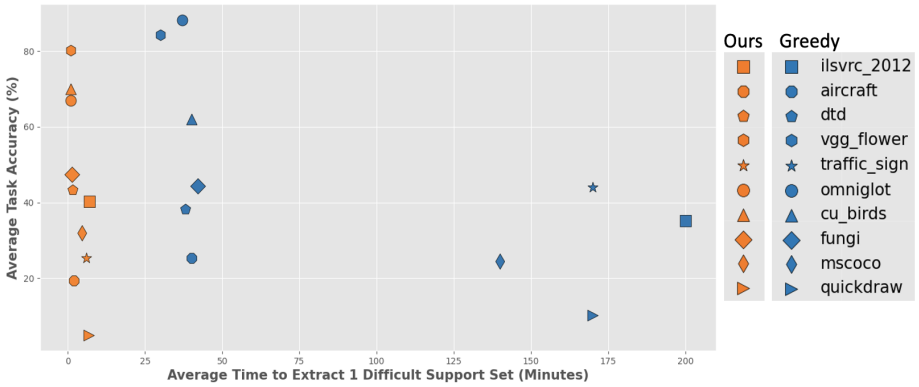

Figure 4.2: **FASTDIFFSEL extracts tasks with similar accuracy to those extracted by greedy search algorithms [2] but is at least 20x faster**. We extract 50 5-way 5-shot tasks per sub-dataset using [100] as the base model (ViT-S initialized with SSL DINO weights then meta-trained with ProtoNets on MD's ilsvrc_2012) on an A5000 GPU (64GB RAM).

for r iterations and thus has a search time complexity of $\mathcal{O}(N.k.d.r)$ along with as many adaptation steps. In comparison, our algorithm removes the need for this exhaustive search and large number of adaptation steps, thus offering a significant speedup. In practice, we find that our algorithm offers speedups of at least 20x when compared to a greedy algorithm (see Fig. 4.2 and Appendix).

4.4 Difficult Support Set Extraction on META-DATASET

Since FASTDIFFSEL is general and highly efficient, we can use it to extract difficult tasks from any large-scale vision dataset. MD is one of the most widely-used benchmarks for few-shot classification, thus we primarily use it to validate the effectiveness of our extraction algorithm. We include further analyses on difficult tasks extracted from further datasets in Sec. 4.4.2.

META-DATASET (MD) [230] contains 10 image sub-datasets from a diverse set of domains. All the sub-datasets (except mscoco and traffic_signs) are split into disjoint train/val/test classes, whereas mscoco and traffic_sign are test only sub-datasets. During few-shot training, variable-way variable-shot tasks are randomly sampled from the train classes of each sub-

dataset. During few-shot testing, 600 variable-way variable-shot tasks are sampled from the test classes of each sub-dataset with the average task classification accuracy reported for each sub-dataset.

4.4.1 Test task samplers for META-DATASET

We compare tasks sampled from the test split of MD's sub-datasets using 3 methods: (i) MD's default sampler; (ii) the greedy approach of [2] and (iii)FASTDIFFSEL(Ours). Note, only (ii) and (iii) can be used to deterministically sample difficult tasks:

MD's default sampler. The default task sampler in MD samples imbalanced tasks of variable-way, variable-shots. The shot, way and size of the query set depends on the size of the sub-dataset in MD. For more details, refer to Sec 3.2 in [230]. We note, however, that the default sampler cannot deterministically sample difficult tasks.

Greedy search-based sampler. We use [2] to extract difficult tasks from MD. This approach works by iteratively replacing each example in a support set of fixed size with one from a given search pool such that the query loss is maximized. Because of the large computational cost and time associated with searching the pool each time an example is replaced, greedy approaches do not scale well to MD's sampler where task ways can vary as large as 50 and shots as large as 100. We therefore only consider fixed-way, fixed-shot tasks when sampling via greedy search on MD.

FASTDIFFSEL. We use FASTDIFFSEL to extract difficult tasks from MD. As the trained base model f_θ, we choose a state-of-the-art method [100] on MD which employs a ViT-S feature extractor [226] initialized with self-supervised DINO weights [36] pre-trained on ilsvrc_2012. The extractor is then further meta-trained on the ilsvrc_2012 split from MD.

To compare against the above samplers, we consider two configurations:

- **Variable-way, variable-shot tasks**. Here, we exactly match the variable way and shots per class of each task yielded by MD's default sampler. Specifically, we use the default sampler to generate and save the shot and way for a fixed number of tasks, which we then feed into our algorithm to extract difficult tasks of equivalent specifications.
- **Fixed-way, fixed-shot tasks**. Here, we set the way and shots per class to be fixed for all tasks. While less challenging, it allows us (i) to compare our algorithm to greedily sampled difficult tasks; and (ii) to control for a task's way being a source of difficulty.

4.4.2 Validation of difficult META-DATASET tasks

Here we compare the efficacy of the above 3 methods and their extracted tasks across the 10 test sub-datasets in MD.

FASTDIFFSEL vs. default sampler. When compared over 600 variable-way variable-shot tasks per sub-dataset, we find that our algorithm extract tasks which are more difficult on an average than MD's default sampler, as shown in Fig. 4.1. This is consistent across all 10 sub-datasets. We also show that these tasks are consistently difficult for a wide range of top-performing few-shot classification methods, as shown in Fig. 4.3 (see Sec. 4.5.3 for details). For certain sub-datasets (e.g. quickdraw), the drop in classification accuracy can be as large as 50% when compared to tasks sampled from MD's default sampler.

FASTDIFFSEL vs. greedy sampler. We compare our algorithm to the greedy sampler over 50 5-way 5-shot tasks sampled per sub-dataset. For fairness, we run both methods for the same set of query sets, and report the average task accuracy and extraction time in Fig 4.2. Here, we see that the average task accuracy is comparably low for each sub-dataset but that our algorithm is almost 1-2 orders of magnitude faster. This speedup is most significant when the search pool is large (e.g. for ilsvrc_2012, quickdraw). Note that the greedy sampler takes between 20-200 minutes *per task*, highlighting its impracticality for larger search

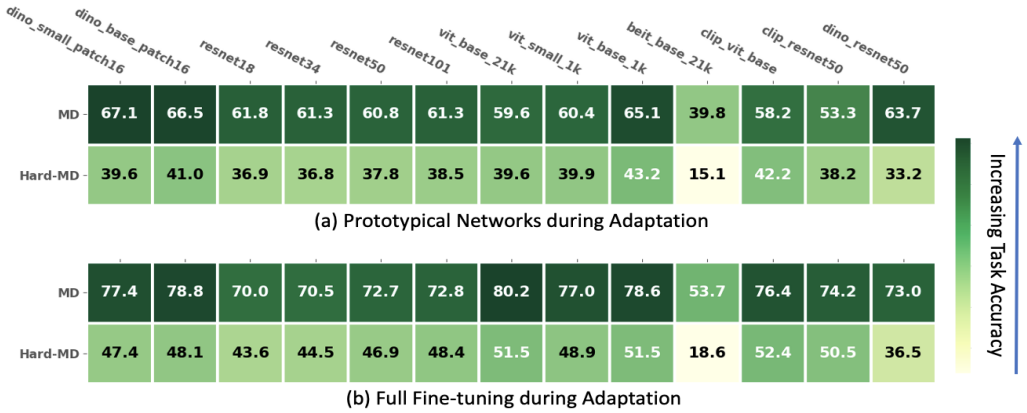

Figure 4.3: **We evaluate a wide range of top-performing methods over 200 variable-way, variable-shot difficult tasks per sub-dataset in HARD-MD and find a consistent 20-30% drop in performance compared to MD tasks regardless of feature extractor, pretraining method, and adaptation strategy.** We evaluate on all MD sub-datasets except ilsvrc_2012. We include performance on each sub-dataset and their 95% confidence intervals in Appendix.

pools, support set sizes and numbers of tasks.

In all, FASTDIFFSEL is able to consistently extract difficult tasks on which state-of-the-art methods achieve low classification accuracy, while also offering a significant speedup in extraction time compared to existing task samplers. It can, therefore, readily be leveraged to further the study of failure modes in few-shot classification methods.

4.5 Stress Testing With HARD-META-DATASET++

Few-shot classification benchmarks like MD [230] and VTAB+MD [65] are highly challenging but are not specifically geared to driving performance on difficult tasks. We leverage our extraction algorithm to fill this gap and introduce HARD-MD++, a new test-only benchmark of exclusively difficult few-shot classification tasks extracted from 4 large-scale vision datasets. We extensively stress test a wide range of top-performing methods on HARD-MD++, presenting novel insights on the robustness of current methods to difficult tasks.

4.5.1 Test datasets

HARD-MD++ is composed of difficult tasks extracted from MD [230], ORBIT [163], CURE-OR [222] and OBJECTNET [20]. We motivate the choice of MD by the fact that it is one of the most widely-used few-shot classification benchmarks. The remaining datasets are chosen because they specifically curate images with real-world variations and provide corresponding 'quality' annotations. They can, therefore, be leveraged to derive deeper insights into the *properties* of difficult tasks as we explore in Sec. 4.5.3. Together, these datasets cover a broad range of challenging and real-world tasks.

Following Sec. 4.4, we use FASTDIFFSEL to extract 200 difficult tasks from each dataset. We use a trained base model f_θ of ViT-S [62] pre-trained using DINO [36] and further meta-trained on MD's ilsvrc_2012 split. For MD, we extract variable-way variable-shot tasks to align with its existing evaluation protocol. For the remaining datasets, we extract fixed-way fixed-shot tasks to enable controlled analysis on task properties beyond their way and shot. Together, HARD-MD++ comes to a total of 2400 difficult test tasks extracted across these datasets. We include further implementations details in Appendix.

META-DATASET. We extract 200 variable-way variable-shot difficult tasks from each test sub-datasets in MD. Note, we exclude the ilsvrc_2012 subset so as not to prevent the use of feature extractors pre-trained on it.

ORBIT contains 3822 videos of 486 objects captured by people who are blind on their mobile phones. Each frame is annotated with 7 quality issues including blur, framing, lighting and occlusion. We extract 200 fixed 5-way, 5-shot difficult tasks from the test split of ORBIT which cover 1198 videos of 158 objects.

CURE-OR contains 1M test images of 100 objects captured under controlled viewpoint

variations (e.g., front, back, side etc.) with corresponding annotations. We extract 200 fixed 5-way, 5-shot difficult tasks from CURE-OR.

OBJECTNET contains 50K test images of 313 objects captured under rotation, background, and viewpoint variations. We extract 200 fixed-way fixed shot difficult tasks from OBJECTNET.

4.5.2 Metrics and training

Metrics. Model performance on HARD-MD++ should be reported as the average classification accuracy and 95% confidence interval over the difficult tasks per sub-dataset. This should be accompanied by performance on MD to provide a more complete characterization of model performance.

Training. We primarily advocate for cross-domain or strong generalization [229], thus following [100], any pre-trained checkpoint, algorithm and model architecture can be used when evaluating on HARD-MD++. This aligns with more recent few-shot learning practices which allow the wide range of publicly-available datasets and pre-trained models to be leveraged.

4.5.3 Results

HARD-MD++ is a challenging benchmark of difficult tasks from a diverse range of datasets. We stress test a wide range of top-performing few-shot classification methods on HARD-MD++ to gauge their robustness to these tasks. We look first at MD and then ORBIT, CURE-OR, and OBJECTNET, primarily focusing on the robustness of different model architectures and pre-training strategies.

State-of-the-art methods. Recent works [44, 58, 64, 100, 224] show that transfer learning with a powerful feature extractor performs extremely well on few-shot classification tasks

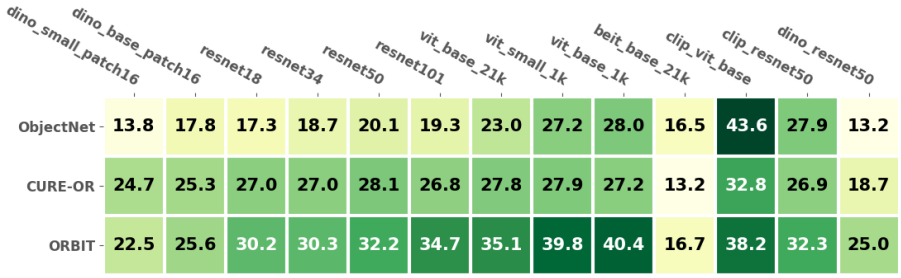

Figure 4.4: **We stress test a wide range of few-shot classifiers on difficult tasks from OBJECTNET, CURE-OR and ORBIT in HARD-MD++.** We find that ViT-B (CLIP) outperforms all the other models by a large margin for CURE-OR and OBJECTNET, while being extremely competitive for ORBIT. Evaluation across 200 tasks per domain using Prototypical Networks as the adaptation strategy. More results with fine-tuning in Appendix.

compared to previous meta-learning methods [75, 196]. We therefore consider a wide range of feature extractors and pre-training paradigms: ResNets [93], Vision Transformers (ViTs) [62, 226], self-supervised variants of ResNets and ViTs [19, 36] and the visual encoder of zero-shot models such as CLIP [192]. We investigate two adaptation strategies when adapting to each difficult test task in HARD-MD++: (i) computing and classifying by mean class prototypes following Prototypical Networks [213] and (ii) fine-tuning the entire feature extractor using strong augmentations following [100]. Further details are provided in the Appendix.

Results on Difficult Tasks from META-DATASET

Our key finding is that state-of-the-art methods across the board consistently drop at least 20-25% in classification accuracy when adapting to difficult tasks in HARD-MD compared with tasks randomly sampled from MD, shown in Fig. 4.3. Note, HARD-MD refers to the MD split in HARD-MD++.

Detailed findings. We find that there is a consistent drop in classification accuracy on HARD-MD across all state-of-the-art methods, regardless of adaptation strategy, feature extractor and pre-training paradigm (see Fig. 4.3). This validates our proposed algorithm in its ability to extract generally challenging tasks, but also highlights the limitations of

current approaches. In particular, approaches that employ a fine-tuning adaptation strategy see a more significant drop in accuracy compared to those employing a prototypical-style adaptation. For example, fine-tuning a ViT-B feature extractor pre-trained on ImageNet-21k ('vit_base_21k' in Fig. 4.3) [62] on each test task leads to a drop of $\sim$30% in accuracy on HARD-MD while a prototypical-style adaptation leads to a drop of only $\sim$20% (note, this approach is strongest on original MD in our implementation). This suggests that although fully fine-tuning a model can significantly increase its accuracy on general tasks, it may not hold for specifically difficult tasks. On the other hand, we find the visual encoder of CLIP which also uses a ViT-B architecture ('clip_vit_base' in Fig. 4.3) [192] to be more robust to difficult tasks. Despite under-performing on MD (ranks 6^{th}), it displays strong performance on HARD-MD across both adaptation strategies, achieving the highest accuracy across all methods with fine-tuning. This trend is consistent for fixed-way, fixed-shot tasks on HARD-MD (see Appendix). These early results suggest that large-scale vision-language pre-training may offer more robustness when generalising to new difficult tasks.

Effect of meta-training on ilsvrc_2012. [100] show that a method's performance on MD can be improved by further meta-training its pre-trained feature extractor on MD's ilsvrc_2012 split. We investigate whether this can also improve performance on HARD-MD. We further meta-train a subset of the methods on the ilsvrc_2012 split using Prototypical Networks [213] and compare the average task classification accuracy in Appendix. We find that despite the further meta-training, the tested methods still display a consistent drop of 20-30% on HARD-MD. This suggests that novel algorithmic contributions, rather than further training, may be required to improve robustness to difficult tasks.

Results on Difficult Tasks from CURE-OR, ORBIT and OBJECTNET

Similar to Sec. 4.5.3, our key finding is that state-of-the-art methods consistently achieve low classification accuracy on difficult tasks from CURE-OR, ORBIT and OBJECTNET, shown in Fig. 4.4. In particular, ViT-B pre-trained with CLIP [192] outperforms all other methods by a significant margin on both CURE-OR (by $>4\%$) and OBJECTNET (by $>15\%$). On ORBIT, ViT-B pre-trained on ilsvrc_2012 [56] performs best though ViT-B with CLIP is still competitive.

We can go beyond task classification accuracy with CURE-OR and ORBIT by leveraging their per-image annotations to investigate the *properties* of difficult tasks. In particular, we use the object viewpoint annotations in CURE-OR, and all quality issue annotations (except object_not_present) in ORBIT. We compare difficult tasks extracted by our algorithm for each dataset with 'easy' tasks for a fixed 5-way 5-shot setting. We can also extract 'easy' tasks by *minimizing* the objective support set extraction objective for the same set of query sets. In Fig. 4.5, we compare the composition of these difficult and easy tasks by the annotated attributes in their supports sets.

CURE-OR. We randomly sample 200 query sets containing images only showing the object with a front or back viewpoint. The sampled query sets contain 10 examples per class. For each query set, we use our algorithm to extract an easy and a difficult support set and visualize the distribution of their annotations in Fig. 4.5-(Left). Here, we see that the easy support sets have a higher proportion of images with front or back viewpoints relative to the total size of the support set, while the difficult support sets have a significantly higher proportion of images with side viewpoints.

ORBIT. Unlike CURE-OR, ORBIT was not collected in a controlled setting and hence its quality issue annotations have a long-tailed distribution. We therefore randomly sample 200

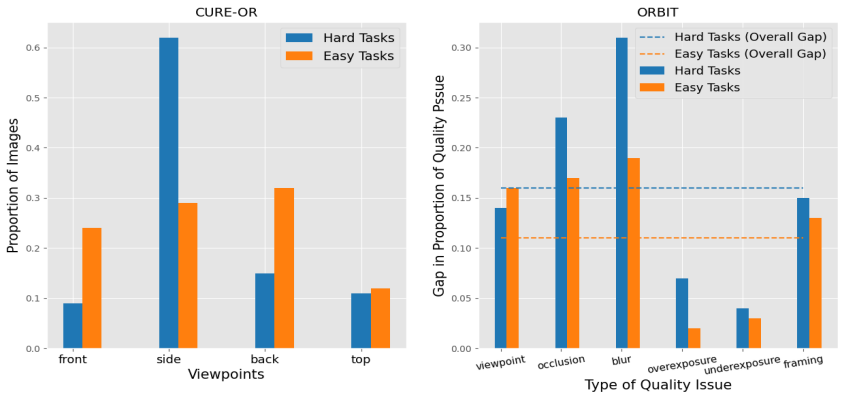

Figure 4.5: **Difficult few-shot tasks have a larger mismatch between the properties of their support and query sets compared to easy tasks.** (Left): Difficult tasks in CURE-OR have support sets with the majority of images showing the object in side view when the query sets contain images of the object only in front/back view; (Right): Difficult tasks in ORBIT have a larger difference in the proportion of quality issues contained in their support versus query sets compared to easy tasks.

query sets, and for each, we use FASTDIFFSEL to extract an easy and a difficult support set. For each quality issue, we compute the difference (gap) in the proportion of images in the support versus query set with that particular issue. In Fig. 4.5-(Right), we report this gap averaged over 200 tasks. Here, we observe that difficult tasks have a larger gap (i.e. difference in support and query set) in quality issues than easier tasks. Together, these results suggest that a mismatch between image characteristics within a support versus query set (e.g., viewpoint, occlusion) can be a source of task difficulty. While these annotations do not cover all possible image characteristics, they provide a starting point to explore the robustness of few-shot classifiers to more qualitative distribution shifts. In Fig. 4.5, we curate few-shot tasks leveraging image-level annotations from ORBIT, CURE-OR and MS-COCO – to show that tasks with distribution shifts with respect to natural characteristics have a lower accuracy.

4.6 Conclusion

We introduce a general and scalable algorithm – FASTDIFFSEL to extract difficult tasks for few-shot classification. We apply FASTDIFFSEL to 4 large-scale vision datasets: META-

DATASET, ORBIT, CURE-OR, and OBJECTNET, and introduce HARD-MD++, a new *test-only* suite of 2400 difficult tasks from across these datasets. We stress test a wide range of top-performing few-shot methods on HARD-MD and demonstrate a consistent drop of 20-25% in classification accuracy compared to MD. We conduct additional quantitative analyses on CURE-OR and ORBIT which show that difficult tasks typically have a distribution shift in the characteristics between a support and query set (e.g. viewpoint, blur). We believe that the efficiency of FASTDIFFSEL along with HARD-MD++ can drive the study of failure modes in few-shot classification methods which is a under explored area of research.

Chapter 5: **Mechanistically Understanding and Editing Text-to-Image Generative Models**

5.1 Knowledge Localization and Model Editing in Early Stable-Diffusion Variants

Text-to-Image generative models such as Stable-Diffusion [197], Imagen [201] and DALLE [193] have revolutionized conditional image generation in the last few years. These models have attracted a lot of attention due to their impressive image generation and editing capabilities, obtaining state-of-the-art FID scores on common generation benchmarks such as MS-COCO [145]. Text-to-Image generation models are generally trained on billion-scale image-text pairs such as LAION-5B [202] which typically consist of a plethora of visual concepts encompassing color, artistic styles, objects, and famous personalities, amongst others. Prior works [35, 214, 215] have shown that text-to-image models such as Stable-Diffusion memorize various aspects of the pre-training dataset. For example, given a caption from the LAION dataset, a model can generate an exact image from the training dataset corresponding to the caption in certain cases [35]. These observations reinforce that some form of knowledge corresponding to visual attributes is stored in the parameter space of text-to-image model.

When an image is generated, it possesses visual attributes such as (but not limited to) the

presence of distinct objects with their own characteristics (such as color or texture), artistic style or scene viewpoint. This attribute-specific information is usually specified in the conditioning textual prompt to the UNet in text-to-image models which is used to pull relevant knowledge from the UNet to construct and subsequently generate an image. This leads to an important question: *How and where is knowledge corresponding to various visual attributes stored in text-to-image models?* In our paper, we choose to concentrate on specific visual attributes within a scene, including objects, style, action, and color. Additionally, in the Appendix, we delve into aspects like count and viewpoint.

In this work, we empirically study this question towards understanding how knowledge corresponding to different visual attributes is stored in text-to-image models, using Stable Diffusion[197] as a representative model. In particular, we adapt Causal Mediation Analysis [184, 237] for large-scale text-to-image diffusion models to identify specific causal components in the (i) UNet and (ii) the text-encoder where visual attribute knowledge resides. Previously, Causal Meditation Analysis has been used for understanding where factual knowledge is stored in LLMs. In particular, [165] find that factual knowledge is localized and stored in the mid-MLP layers of a LLM such as GPT-J [240]. Our work, however, paints a different picture - for multimodal text-to-image models, we specifically find that knowledge is not localized to one particular component. Instead, there exist various components in the UNet where knowledge is stored. However, each of these components store attribute information with a different efficacy and often different attributes have a distinct set of causal components where knowledge is stored. For e.g., for *style* – we find that the first self-attention layer in the UNet stores *style* related knowledge, however it is not causally important for other attributes such as *objects*, *viewpoint* or *action*. To our surprise, we specifically find that the cross-attention layers are not causally important states and a significant amount of knowledge is in fact stored in components such as the ResNet blocks

and the self-attention blocks.

Remarkably, in the text-encoder, we find that knowledge corresponding to distinct attributes is strongly localized, contrary to the UNet. However unlike generative language models [165] where the mid MLP layers are causal states, we find that the first self-attention layer is causal in the CLIP based text-encoder of public text-to-image generative models (e.g., Stable-Diffusion).

Identification of local causal states in a given model has a crucial benefit: it allows for incorporating controlled edits to the model by updating *only* a tiny fraction of the model parameters without any fine-tuning. Using our observation that the text-encoder hosts *only* one localized causal state, we introduce a new data-free and fast model editing method - DiffQuickFix which can edit concepts in text-to-image models effectively using a closed-form update. In particular, we show that DiffQuickFix can (i) remove copyrighted styles, (ii) trademarked objects as well as (iii) update stale knowledge 1000x faster than existing fine-tuning based editing methods such as [78, 126] with comparable or even better performance in some cases.

In summary, our contributions are as follows:

- We adapt Causal Mediation Analysis [184, 237] to large-scale text-to-image models (with Stable-Diffusion as a representative model), and use it to trace knowledge corresponding to various visual attributes in the UNet and text-encoder.
- We perform large-scale analysis of the identified causal components and shed light on the knowledge flow corresponding to various visual attributes in the UNet and the text-encoder.
- Leveraging the interpretability observations of localized causal states in the text-encoder, we develop a light-weight method DiffQuickFix which can edit various

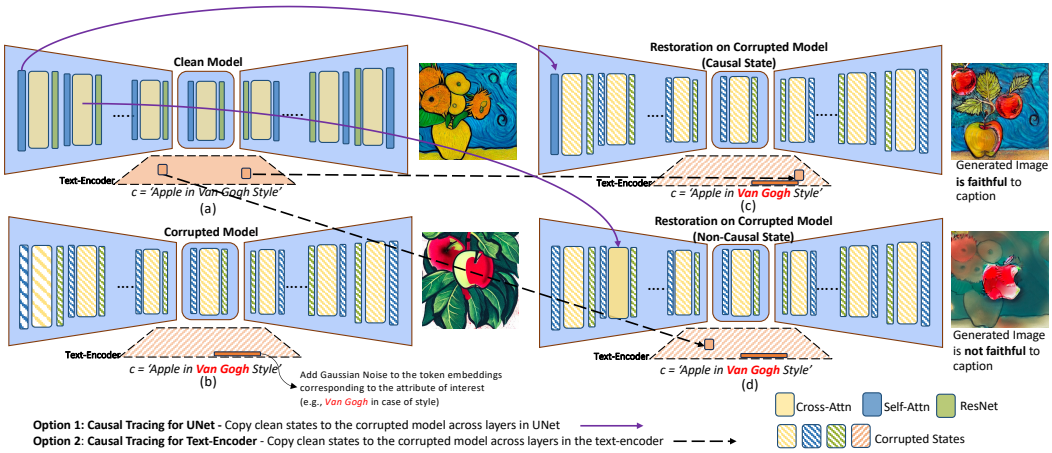

Figure 5.1: **Causal Tracing in Text-to-Image Models for (i) UNet and (ii) Text-Encoder shows that knowledge location matters, i.e., restoring causal layers in a corrupted model causes the model to obey the prompt again, while restoring non-causal layers does not.** (a) *Clean Model*: We prompt a Stable-Diffusion model in the conventional way and generate an image as output. (b) *Corrupted Model*: Token embeddings corresponding to attribute of interest are corrupted, leading to a generated image that does not obey the prompt. (c) *Restored (Causal) Model*: Causal layer activations are now copied from the clean model to the corrupted model. We observe that the corrupted model can now generate images with high fidelity to the original caption. (d) *Restored (Non-Causal) Model*: Non-causal layer activations are copied from the clean model to the corrupted model, but we now observe that the generated image does not obey the prompt. A single layer is copied at a time, and it can be from either the UNet (solid violet arrow) or the text-encoder (broken black arrow).

concepts in text-to-image models in under a second, 1000x faster than existing concept ablating methods [78, 126].

5.2 Causal Tracing for Text-to-Image Generative Models

In this section, we first provide a brief overview of diffusion models in Sec.(5.2.1). We then describe how causal tracing is adapted to multimodal diffusion models such as Stable-Diffusion.

5.2.1 Background

Diffusion models are inspired by non-equilibrium thermodynamics and specifically aim to learn to denoise data through a number of steps. Usually, noise is added to the data following a Markov chain across multiple time-steps $t \in [0,T]$. Starting from an initial random real image $\mathbf{x}_0$, the noisy image at time-step t is defined as $\mathbf{x}_t = \sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{(1-\alpha_t)}\varepsilon$. In particular, α_t determines the strength of the random Gaussian noise and it gradually decreases as the time-step increases such that $\mathbf{x}_T \sim \mathcal{N}(0,I)$. The denoising network denoted by $\varepsilon_\theta(\mathbf{x}_t, \mathbf{c}, t)$ is pre-trained to denoise the noisy image $\mathbf{x}_t$ to obtain $\mathbf{x}_{t-1}$. Usually, the conditional input $\mathbf{c}$ to the denoising network $\varepsilon_\theta(.)$ is a text-embedding of a caption c through a text-encoder $\mathbf{c} = v_\gamma(c)$ which is paired with the original real image $\mathbf{x}_0$. The pre-training objective for diffusion models can be defined as follows for a given image-text pair denoted by ($\mathbf{x}$, $\mathbf{c}$):

$$\mathscr{L}(\mathbf{x}, \mathbf{c}) = \mathbb{E}_{\varepsilon,t}||\varepsilon - \varepsilon_\theta(\mathbf{x}_t, \mathbf{c}, t)||_2^2, \tag{5.1}$$

where θ is the set of learnable parameters. For better training efficiency, the noising as well as the denoising operation occurs in a latent space defined by $\mathbf{z} = \mathscr{E}(\mathbf{x})$ [197]. In this case,

the pre-training objective learns to denoise in the latent space as denoted by:

$$\mathscr{L}(\mathbf{x},\mathbf{c}) = \mathbb{E}_{\varepsilon,t}||\varepsilon - \varepsilon_\theta(\mathbf{z}_t,\mathbf{c},t)||_2^2, \tag{5.2}$$

where $\mathbf{z}_t = \mathscr{E}(\mathbf{x}_t)$ and $\mathscr{E}$ is an encoder such as VQ-VAE [233]. During inference, where the objective is to synthesize an image given a text-condition $\mathbf{c}$, a random Gaussian noise $\mathbf{x}_T \sim \mathscr{N}(0,I)$ is iteratively denoised for a fixed range of time-steps in order to produce the final image.

5.2.2 Adapting Causal Tracing For Text-to-Image Diffusion Models

Causal Mediation Analysis [184, 237] is a method from causal inference that studies the change in a response variable following an intervention on intermediate variables of interest (mediators). One can think of the internal model components (e.g., specific neurons or layer activations) as mediators along a directed acyclic graph between the input and output. For text-to-image diffusion models, we use Causal Mediation Analysis to trace the causal effects of these internal model components within the UNet and the text-encoder which contributes towards the generation of images with specific visual attributes (e.g., *objects, style*). For example, we find the subset of model components in the text-to-image model which are causal for generating images with specific *objects*, *styles*, *viewpoints*, *action* or *color*.

Where is Causal Tracing Performed? We identify the causal model components in both the UNet ε_θ and the text-encoder v_γ. For ε_θ, we perform the causal tracing at the granularity of layers, whereas for the text-encoder, causal tracing is performed at the granularity of hidden states of the token embeddings in $\mathbf{c}$ across distinct layers. The UNet ε_θ consists of 70 unique layers distributed amongst three types of blocks: (i) `down-block`; (ii) `mid-block` and (iii) `up-block`. Each of these blocks contain varying number of cross-attention layers,

self-attention layers and residual layers. Fig 8.1 visualizes the internal states of the UNet and how causal tracing for knowledge attribution is performed. For the text-encoder v_γ, there are 12 blocks in total with each block consisting of a self-attention layer and a MLP layer (see Fig 8.1). We highlight that the text-encoder in text-to-image models such as Stable-Diffusion has a GPT-style architecture with a causal self-attention, though it's pre-trained without a language modeling objective. More details on the layers used in the Appendix.

Given a caption c, an image $\mathbf{x}$ is generated starting from some random Gaussian noise. This image $\mathbf{x}$ encapsulates the visual properties embedded in the caption c. For e.g., the caption c can contain information corresponding from *objects* to *action* etc. We specifically identify distinct components in the UNet and the text-encoder which are causally responsible for these properties.

Creating the Probe Captions. We primarily focus on four different visual attributes for causal tracing: (i) *objects*; (ii) *style*; (iii) *color*; and (iv) *action*. In particular, identifying the location of knowledge storage for *objects* and *style* can be useful to perform post-hoc editing of diffusion models to edit concepts (e.g., delete or update certain concepts). We provide the complete details about the probe dataset used for causal tracing in the dataset description section in the Appendix. The probe dataset also contains additional captions for *viewpoint* and *count* attribute. However, we do not focus on them as often the generations from the unedited model are erroneous for these attributes (see Appendix for details).

5.2.3 Tracing Knowledge in UNet

During inference, classifier-free guidance [99] is used to regulate image-generation by incorporating scores from the conditional and unconditional diffusion model at each of the time-steps. In particular, at each time-step, classifier-free guidance is used in the following

way to combine the conditional ($\varepsilon_\theta(\mathbf{z}_t, \mathbf{c}, t)$) and unconditional score estimates ($\varepsilon_\theta(\mathbf{z}_t, t)$) at each time-step t to obtain the combined score denoted as $\hat{\varepsilon}(\mathbf{z}_t, \mathbf{c}, t)$:

$$\hat{\varepsilon}_\theta(\mathbf{z_t}, \mathbf{c}, t) = \varepsilon_\theta(\mathbf{z_t}, \mathbf{c}, t) + \alpha(\varepsilon_\theta(\mathbf{z_t}, \mathbf{c}, t) - \varepsilon_\theta(\mathbf{z_t}, t)), \quad \forall t \in [T, 1]. \tag{5.3}$$

This combined score is used to update the latent $\mathbf{z}_t$ using DDIM sampling [216] at each time-step iteratively to obtain the final latent code $\mathbf{z}_0$.

To perform causal tracing on the UNet ε_θ (see Fig. 8.1 for visualization), we perform a sequence of operations that is somewhat analogous to earlier work from [165] which investigated knowledge-tracing in large language models. We consider three types of model configurations: (i) a clean model ε_θ, where classifier-free guidance is used as default; (ii) a corrupted model $\varepsilon_\theta^{corr}$, where the word embedding of the subject (e.g., *Van Gogh*) of a given attribute (e.g., *style*) corresponding to a caption c is corrupted with Gaussian Noise; and, (iii) a restored model $\varepsilon_\theta^{restored}$, which is similar to $\varepsilon_\theta^{corr}$ except that one of its layers is restored from the clean model at each time-step of the classifier-free guidance. Given a list of layers $\mathscr{A}$, let $a_i \in \mathscr{A}$ denote the i^{th} layer whose importance needs to be evaluated. Let $\varepsilon_\theta[a_i]$, $\varepsilon_\theta^{corr}[a_i]$ and $\varepsilon_\theta^{restored}[a_i]$ denote the activations of layer a_i. To find the importance of layer a_i for a particular attribute embedded in a caption c, we perform the following replacement operation on the corrupted model $\varepsilon_\theta^{corr}$ to obtain the restored model $\varepsilon_\theta^{restored}$:

$$\varepsilon_\theta^{restored}[a_i] : \varepsilon_\theta^{corr}[a_i] = \varepsilon_\theta[a_i]. \tag{5.4}$$

Next, we obtain the restored model by replacing the activations of layer a_i of the corrupted model with those of the clean model to get a restored layer $\varepsilon_\theta^{restored}[a_i]$. We run classifier-free

guidance to obtain the combined score estimate:

$$\hat{\varepsilon}_\theta^{restored}(\mathbf{z_t}, \mathbf{c}, t) = \varepsilon_\theta^{restored}(\mathbf{z_t}, \mathbf{c}, t) + \alpha(\varepsilon_\theta^{restored}(\mathbf{z_t}, \mathbf{c}, t) - \varepsilon_\theta^{restored}(\mathbf{z_t}, t)), \quad \forall t \in [T, 1]. \tag{5.5}$$

The final latent $\mathbf{z}_0$ is obtained with the score from Eq. (5.5) at each time-step using DDIM [216] and passed through the VQ-VAE decoder to obtain the final image $\mathbf{x}_0^{restored}$.

5.2.4 Tracing Knowledge in the Text-Encoder

The text-encoder in public text-to-image models such as Stable-Diffusion is a CLIP-ViT-L/336px text-encoder [197]. Similar to Sec.(5.2.3), we define three states of the CLIP text-encoder: (i) Clean model denoted by v_γ; (ii) Corrupted model v_γ^{corr} where the word embedding of the subject in a given caption c is corrupted; (iii) Restored model $v_\gamma^{restored}$ which is similar to v_γ^{corr} except that one of its layers is copied from v_γ. Similar to Sec.(5.2.3), to find the effect of the layer $a_i \in \mathscr{A}$, where $\mathscr{A}$ consists of all the layers to probe in the CLIP text-encoder:

$$v_\gamma^{restored}[a_i] : v_\gamma^{corr}[a_i] = v_\gamma[a_i], \tag{5.6}$$

We then use the restored text-encoder $v_\gamma^{restored}$ with classifier-free guidance to obtain the final score estimate:

$$\hat{\varepsilon}_\theta(\mathbf{z_t}, \mathbf{c}', t) = \varepsilon_\theta(\mathbf{z_t}, \mathbf{c}', t) + \alpha(\varepsilon_\theta(\mathbf{z_t}, \mathbf{c}', t) - \varepsilon_\theta(\mathbf{z_t}, t)), \quad \forall t \in [T, 1] \tag{5.7}$$

where $\mathbf{c}' = v_\gamma^{restored}[a_i](c)$ for a given caption c. This score estimate $\hat{\varepsilon}_\theta(\mathbf{z_t}, \mathbf{c}', t)$ at each time-step t is used to obtain the final latent code $\mathbf{z}_0$ which is then used with the VQ-VAE decoder to obtain the final image $\mathbf{x}_0^{restored}$.

5.2.5 Extracting Causal States Using CLIP-Score

In this section, we discuss details on how to retrieve causal states using automated metrics such as `CLIP-Score` [98]. Let $\mathbf{x}_0^{restored}(a_i)$ be the final image generated by the diffusion model after intervening on layer a_i, $\mathbf{x}_0$ be the image generated by the clean diffusion model and $\mathbf{x}^{corr}$ be the final image generated by the corrupted model. In particular, we are interested in the average indirect effect [184, 237] which measures the difference between the corrupted model and the restored model. Intuitively, a higher value of average indirect effect (AIE) signifies that the restored model deviates from the corrupted model. To compute the average indirect effect with respect to causal mediation analysis for text-to-image models such as Stable-Diffusion, we use `CLIP-Score` which computes the similarity between an image embedding and a caption embedding. In particular, AIE = $(\texttt{CLIPScore}(\mathbf{x}_0^{restored}, c) - \texttt{CLIPScore}(\mathbf{x}_0^{corr}, c))$. Given $\mathbf{x}_0^{corr}$ is common across all the layers, we can use $\texttt{CLIPScore}(\mathbf{x}_0^{restored}, c)$ as the AIE.

Selecting Threshold for `CLIP-Score`. We observe that the difference between the CLIP-Score of generated images (after restoration) with high fidelity to the original caption and generated images (after restoration) with low fidelity to the original caption, to be small. Therefore to effectively find a reasonable cut-off point to automatically select causal states (where the generated images have high-fidelity to the original caption), we use a threshold selection mechanism. In order to determine the optimal threshold value for `CLIP-Score`, we select a small validation set of 10 prompts per attribute. To this end, we establish a concise user study interface. Through human participation, we collect binary ratings if an image generated by restoring a particular layer is faithful to the original captions. We then extract the common causal states across all the prompts for a given attribute and find the average (across all the prompts) `CLIP-Score` for each causal state. We then use the lowest

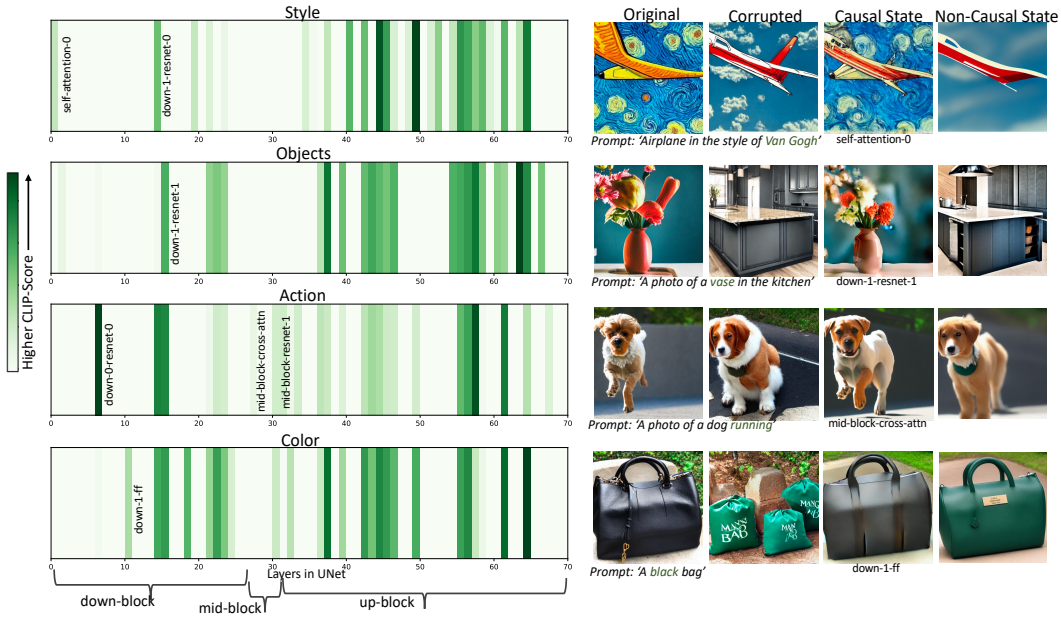

Figure 5.2: **Causal Tracing Results for the UNet: Knowledge is Distributed.** The intensity of the bars indicate the `CLIP-Score` between the generated image (after causal intervention) and the original caption. For each attribute, we find that the causal states are distributed across the UNet and the distribution varies amongst distinct attributes. For e.g., self-attn in the first layer is causal for *style*, but not for *objects*, *action* or *color*. Similarly, mid-block cross-attn is causal for *action*, but not for the other attributes. On the right-side, we visualize the images generated by (i) Original model; (ii) Corrupted Model; (iii) Restored causal states and (iv) Restored non-causal states in the UNet for *style, action, object, color* attributes.

average `CLIP-Score` corresponding to a causal state as the threshold, which we apply on the probe dataset to filter the causal states at scale for each attribute separately.

5.3 How is Knowledge Stored in Text-to-Image Models?

In this section, we discuss the results of tracing knowledge across various components of the text-to-image model in details.

Tracing Results for UNet. In Fig 5.2, we illustrate the distribution of causal states across different visual attributes within the UNet architecture using the `CLIP-Score` metric. This metric evaluates the faithfulness of the image produced by the restored state $\mathbf{x}_0^{restored}$ com-

pared to the original caption c. From the insights derived in Fig 5.2, it becomes evident that causal states are spread across diverse components of the UNet. In particular, we find that the density of the causal states are more in the `up-block` of the UNet when compared to the `down-block` or the `mid-block`. Nonetheless, a notable distinction emerges in this distribution across distinct attributes. For instance, when examining the *style* attribute, the initial self-attention layer demonstrates causality, whereas this causal relationship is absent for other attributes. Similarly, in the context of the *action* attribute, the cross-attention layer within the mid-block exhibits causality, which contrasts with its non-causal behavior concerning other visual attributes. Fig 5.2 showcases the images generated by restoring both causal and non-causal layers within the UNet. A comprehensive qualitative enumeration of both causal and non-causal layers for each visual attribute is provided in the Appendix. Our findings underscore the presence of information pertaining to various visual attributes in regions beyond the cross-attention layers. Importantly, we observe that the distribution of information within the UNet diverges from the patterns identified in extensive generative language models, as noted in prior research [166], where attribute-related knowledge is confined to a few proximate layers. In Appendix, we provide additional causal tracing results, where we add Gaussian noise to the entire text-embedding. Even in such a case, certain causal states can restore the model close to its original configuration, highlighting that the conditional information can be completely bypassed if certain causal states are active.

Tracing Results for Text-Encoder. In Fig. 5.3, we illustrate the causal states in the text-encoder for Stable-Diffusion corresponding to various visual attributes. At the text-encoder level, we find that the causal states are localized to the first self-attention layer corresponding to the last subject token across all the attributes. In fact, there exists *only one* causal state in the text-encoder. Qualitative visualizations in Fig. 5.3 illustrate that the restoration of

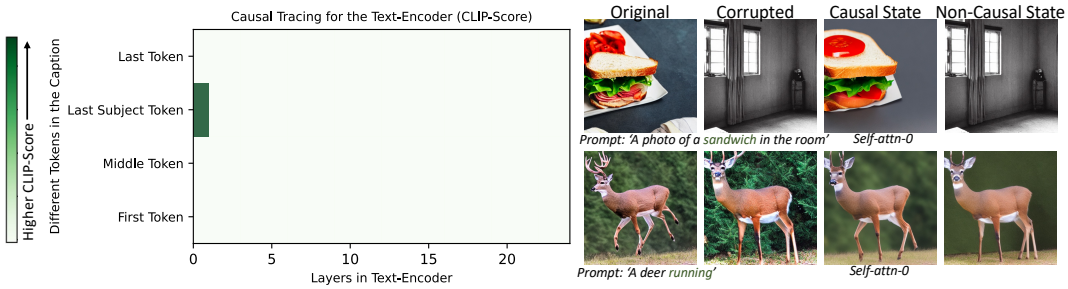

Figure 5.3: **Causal Tracing in the Text-Encoder: Knowledge is Localized.** In the CLIP text-encoder used for Stable-Diffusion, we find the existence of *only* one causal state, which is the first self-attention layer corresponding to the last subject token. The `CLIP-Score`(Left) is computed across all the four visual attributes. Visualizations (Right) further illustrate that restoring the sole causal state (self-attn-0) leads to image generation with high fidelity to the original captions.

layers other than the first self-attention layer corresponding to the subject token does not lead to images with high fidelity to the original caption. Remarkably, this observation is distinct from generative language models where factual knowledge is primarily localized in the proximate mid MLP layers [166].

General Takeaway. Causal components corresponding to various visual attributes are dispersed (with a *different distribution* between *distinct* attributes) in the UNet, whereas there exists *only* one causal component in the text-encoder.

The text-encoder's strong localization of causal states for visual attributes enables controlled knowledge manipulation in text-to-image models, facilitating updates or removal of concepts. However, since attribute knowledge is dispersed in the UNet, targeted editing is challenging without layer interference. While fine-tuning methods for UNet model editing exist [78, 126], they lack scalability and don't support simultaneous editing of multiple concepts. In the next section, we introduce a closed-form editing method, DIFF-QUICKFIX, leveraging our causal tracing insights to efficiently edit various concepts in text-to-image models.

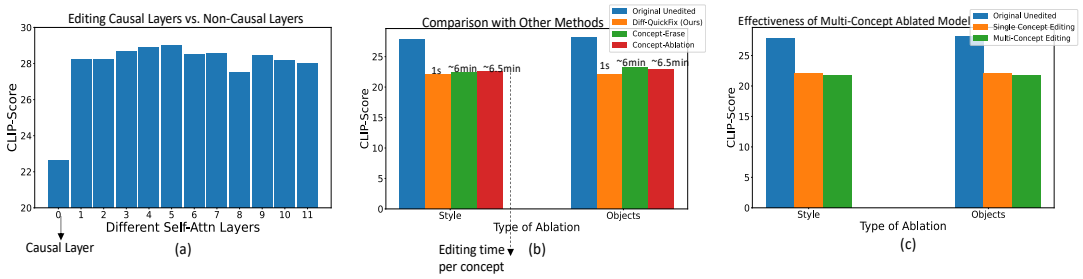

Figure 5.4: **Quantitative Analysis of DIFF-QUICKFIX**. (a) Editing Causal vs. Non-Causal Layers (Averaged across *Objects, Style and Facts*): Lower `CLIP-Score` for causal layer indicates successful edits; (b) Efficacy of DIFF-QUICKFIX when compared to other methods – Our method leads to comparable `CLIP-Scores` to fine-tuning based approaches, but can edit concepts 1000x faster; (c) DIFF-QUICKFIX can be used to effectively edit multiple concepts at once, shown by comparable `CLIP-Scores` to the single-concept edited ones.

5.4 DIFF-QUICKFIX: Fast Model Editing for Text-to-Image Models

5.4.1 Editing Method

Recent works such as [78, 126] edit concepts from text-to-image diffusion models by fine-tuning the UNet. They generate training data for fine-tuning using the pre-trained diffusion model itself. While both methods are effective at editing concepts, fine-tuning the UNet can be expensive due to backpropogation of gradients through the UNet. To circumvent this issue, we design a fast, data-free model editing method leveraging our interpretability observations in Sec. 5.3, where we find that there exists only one causal state (the very first self-attention layer) in the text-encoder for Stable-Diffusion. Our editing method DIFF-QUICKFIX can update text-to-image diffusion models in a targeted way in under $1s$ through a closed-form update making it 1000x faster than existing fine-tuning based concept ablating methods such as [78, 126]. The first self-attention layer in the text-encoder for Stable-Diffusion contains four updatable weight matrices: W_k, W_q, W_v and W_{out}, where W_k, W_q, W_v are the projection matrices for the key, query and value embeddings respectively. W_{out} is the projection matrix before the output from the `self-attn-0` layer after the attention

operations. DIFF-QUICKFIX specifically updates this W_{out} matrix by collecting caption pairs (c_k, c_v) where c_k (key) is the original caption and c_v (value) is the caption to which c_k is mapped. For e.g., to remove the style of *'Van Gogh'*, we set $c_k =$ '*Van Gogh*' and $c_v =$ '*Painting*'. In particular, to update W_{out}, we solve the following optimization problem:

$$\min_{W_{out}} \sum_{i=1}^{N} \|W_{out} k_i - v_i\|_2^2 + \lambda \|W_{out} - W'_{out}\|_2^2, \tag{5.8}$$

where λ is a regularizer to not deviate significantly from the original pre-trained weights W'_{out}, N denotes the total number of caption pairs containing the last subject token embeddings of the key and value. k_i corresponds to the embedding of c_{k_i} after the attention operation using W_q, W_k and W_v for the i^{th} caption pair. v_i corresponds to the embedding of c_{v_i} after the original pre-trained weights W'_{out} acts on it.

One can observe that Eq. (5.8) has a closed-form solution due to the absence of any non-linearities. In particular, the optimal W_{out} can be expressed as the following:

$$W_{out} = (\lambda W'_{out} + \sum_{i=1}^{N} v_i k_i^T)(\lambda I + \sum_{i=1}^{N} k_i k_i^T)^{-1}, \tag{5.9}$$

In Sec. 5.4.3, we show qualitative as well as quantitative results using DIFF-QUICKFIX for editing various concepts in text-to-image models.

5.4.2 Experimental Setup

We validate DIFF-QUICKFIX by applying edits to a Stable-Diffusion [197] model and quantifying the *efficacy* of the edit. For removing concepts such as artistic styles or objects using DIFF-QUICKFIX, we use the prompt dataset from [126]. For updating knowledge (e.g., *President of a country*) in text-to-image models, we add newer prompts to the prompt dataset from [126] and provide further details in the dataset description section in the Appendix.

We compare our method with (i) Original Stable-Diffusion; (ii) Editing methods from [126] and [78]. To validate the effectiveness of editing methods including our DIFF-QUICKFIX, we perform evaluation using automated metrics such as `CLIP-Score`. In particular, we compute the `CLIP-Score` between the images from the edited model and the concept corresponding to the visual attribute which is edited. A low `CLIP-Score` therefore indicates correct edits.

5.4.3 Editing Results

Editing Non-causal Layers Does Not Lead to Correct Edits. We use DIFF-QUICKFIX with the non-causal self-attention layers in the text-encoder to ablate *styles*, *objects* and update *facts*. In Fig. 5.4-(a), we compute the `CLIP-Score` between the generated images and the attribute from the original captions (e.g., *van gogh* in the case of *style*). In particular, we find that editing the non-causal layers does not lead to any intended model changes – highlighted by the high `CLIP-Scores` consistently across non-causal layers (layers numbered 1 to 11). However, editing the sole causal layer (layer-0) leads to correct model changes, highlighted by the lower `CLIP-Score` between the generated images from the edited model and the attribute from the original captions. This shows that identifying the causal states in the model is particularly important to perform targeted model editing for ablating concepts. In Appendix, we show additional qualitative visualizations.

Efficacy in Removing Styles and Objects. Fig. 5.4-(b) shows the average `CLIP-Score` of the generated images from the edited model computed with the relevant attributes from the original captions. We find that the `CLIP-Score` from the edited model with DIFF-QUICKFIX decreases when compared to the generations from the unedited model. We also find that our editing method has comparable `CLIP-Scores` to other fine-tuning based approaches such as Concept-Erase [78] and Concept-Ablation [126], which are more computationally

expensive.

Fig. 5.5 shows qualitative visualizations corresponding to images generated by the text-to-image model before and after the edit operations. Together, these quantitative and qualitative results show that DIFF-QUICKFIX is able to effectively remove various *styles* and *objects* from an underlying text-to-image model. In Appendix we provide additional qualitative visualizations and we show additional results showing that our editing method does not harm surrounding concepts (For e.g., removing the style of *Van Gogh* does not harm the style of *Monet*).

Efficacy in Updating Stale Knowledge. The `CLIP-Score` between the generated images and a caption designating the incorrect fact (e.g., *Donald Trump* as the *President of the US*) decreases from 0.28 to 0.23 after editing with DIFF-QUICKFIX, while the `CLIP-Score` with the correct fact (e.g., *Joe Biden* as the *President of the US*) increases from 0.22 to 0.29 after the relevant edit. This shows that the incorrect fact is updated with the correct fact in the text-to-image model. Additional qualitative visualizations are provided in Fig. 5.5.

Multiple Edits using DIFF-QUICKFIX. An important feature of DIFF-QUICKFIX is its capability to ablate multiple concepts simultaneously. In Fig. 5.4-(c), our framework demonstrates the removal of up to 10 distinct styles and objects at once. This multi-concept ablation results in lower `CLIP-Scores` compared to the original model, similar `CLIP-Scores` to single concept editing.

5.5 Conclusion I

Through the lens of Causal Mediation Analysis, we present methods for understanding the storage of knowledge corresponding to visual attributes in text-to-image models. Notably, we find a distinct distribution of causal states across attributes in the UNet, while the text-

Figure 5.5: **Qualitative Examples with using DIFF-QUICKFIX** to ablate *style*, *objects* and update *facts* in text-to-image models. More qualitative examples in the Appendix.

encoder maintains a single causal state. This differs significantly from observations in language models like GPT, where factual information is concentrated in mid-MLP layers. In contrast, our analysis shows that public text-to-image models like Stable-Diffusion concentrate multiple visual attributes within the first self-attention layer of the text-encoder. Harnessing these insights, we design a fast model editing method DIFF-QUICKFIX. The potency of DIFF-QUICKFIX is manifested through its adeptness in removing artistic styles, objects, and updating outdated knowledge all accomplished data-free and in less than a second, making DIFF-QUICKFIX a practical asset for real-world model editing scenarios.

5.6 Knowledge Localization and Model Editing Across Various Open-Source Text-to-Image Models

In recent years, substantial strides in conditional image generation have been made through diffusion-based text-to-image generative models, including notable examples like Stable-Diffusion [197], Imagen [201], and DALLE [193]. These models have captured widespread attention owing to their impressive image generation and editing capabilities, as evidenced by leading FID scores on prominent benchmarks such as MS-COCO [145]. Typically trained on extensive billion-scale image-text pairs like LAION-5B [202], these models encapsulate

a diverse array of visual concepts, encompassing color, artistic styles, objects, and renowned personalities.

A recent work [23] designs an interpretability framework using causal tracing [184] to trace the location of knowledge about various styles, objects or facts in text-to-image generative models. Essentially, causal tracing finds the indirect effects of intermediate layers [184], by finding layers which can restore a model with corrupted inputs to its original state. Using this framework, the authors find that knowledge about various visual attributes is distributed in the UNet, whereas, there exists a unique causal state in the CLIP text-encoder where knowledge is localized. This unique causal state in the text-encoder can be leveraged to edit text-to-image models in order to remove style, objects or update facts effectively. However, we note that their framework is restricted to early Stable-Diffusion variants such as Stable-Diffusion-v1-5.

In our paper, we first revisit knowledge localization for text-to-image generative models, specifically examining the effectiveness of causal tracing beyond Stable-Diffusion-v1-5. While causal tracing successfully identifies unique localized states in the text-encoder for Stable-Diffusion variants, including v1-5 and v2-1, it fails to do so for recent models like SD-XL [186] and DeepFloyd[1] across different visual attributes. In the UNet, causal states are distributed across a majority of open-source text-to-image models (excluding DeepFloyd), aligning with findings in [23]. Notably, for DeepFloyd, we observe a lack of strong causal states corresponding to visual attributes in the UNet.

To address the *universal* knowledge localization framework absence across different text-to-image models, we introduce the concept of *mechanistic localization* that aims to identify a small number of layers which control the generation of distinct visual attributes, across a

[1] https://github.com/deep-floyd/IF

spectrum of text-to-image models. To achieve this, we propose LOCOGEN, a method that finds a subset of cross-attention layers in the UNet such that when the input to their key and value matrices is changed, output generation for a given visual attribute (e.g., “style") is modified (see Figure 8.1). This intervention in the intermediate layers has a direct effect on the output – therefore LOCOGEN measures the direct effect of intermediate layers, as opposed to indirect effects in causal tracing.

Leveraging LOCOGEN, we probe knowledge locations for different visual attributes across popular open-source text-to-image models such as Stable-Diffusion-v1, Stable-Diffusion-v2, OpenJourney[2], SD-XL [186] and DeepFloyd. For all models, we find that unique locations can be identified for visual attributes (e.g., “style", “objects", “facts"). Using these locations, we then perform weight-space model editing to remove artistic “styles", modify trademarked “objects" and update outdated “facts" in text-to-image models. This weight-space editing is performed using LOCOEDIT which updates the key and value matrices using a closed-form update in the locations identified by LOCOGEN. Moreover, for certain attributes such as “style", we show that knowledge can be traced and edited to a subset of neurons, therefore highlighting the possibilities of neuron-level model editing.

Contributions. In summary, our contributions include:

- We highlight the drawbacks of existing interpretability methods such as causal tracing for localizing knowledge in latest text-to-image models.
- We introduce LOCOGEN which can universally identify layers that control for visual attributes across a large spectrum of open-source text-to-image models.
- By examining edited models using LOCOEDIT along with LOCOGEN, we observe that this efficient approach is successful across a majority of text-to-image models.

[2]https://huggingface.co/prompthero/openjourney

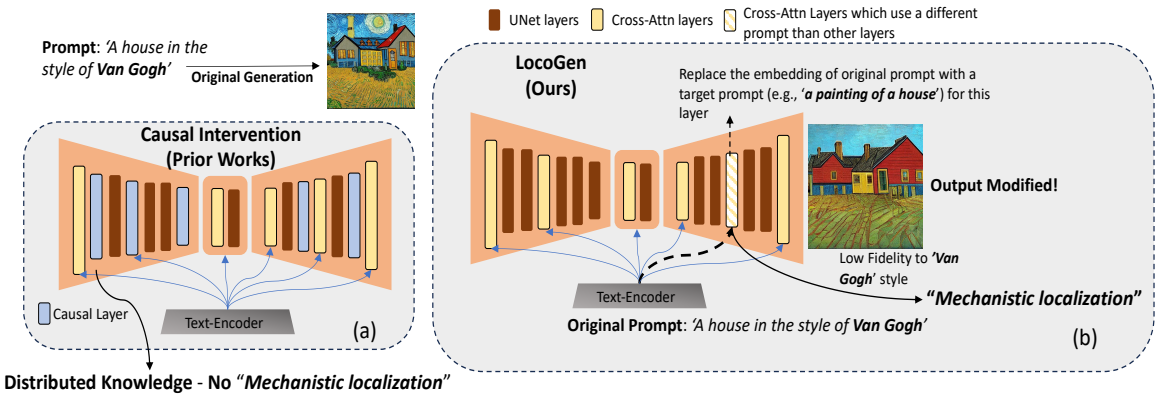

Figure 5.6: **LOCOGEN: Identifying UNet layers that, when given different input, can alter visual attributes (e.g., style, objects, facts).** (a) Earlier works [23] which show distributed knowledge using causal interventions. (b) LOCOGEN where a few cross-attention layers receive a different prompt-embedding than the original, leading to generation of images without the particular style.

5.7 On the Effectiveness of Causal Tracing for Text-to-Image Models

In this section, we empirically observe the effectiveness of causal tracing to models beyond Stable-Diffusion-v1-5. In particular, we find the ability of causal tracing to identify localized control points in Stable-Diffusion-v2-1, OpenJourney, SD-XL and DeepFloyd.

Causal Tracing in UNet. In Figure 5.7, we find that knowledge across different visual attributes is distributed in the UNet for all the text-to-image models (except for DeepFloyd), similar to Stable-Diffusion-v1-5. However, the degree of distribution varies between different text-to-image models. While knowledge about various visual attributes is densely distributed in Stable-Diffusion variants, for SD-XL we find that the distribution is extremely sparse (e.g., only 5% of the total layers are causal). For DeepFloyd, we observe that there are no strong causal states in the UNet. We provide more qualitative visualizations on causal tracing across the these text-to-image models in Appendix. Overall, these results reinforce the difficulty of editing knowledge in the UNet directly due to (i) distribution of causal states or (ii) absence of any.

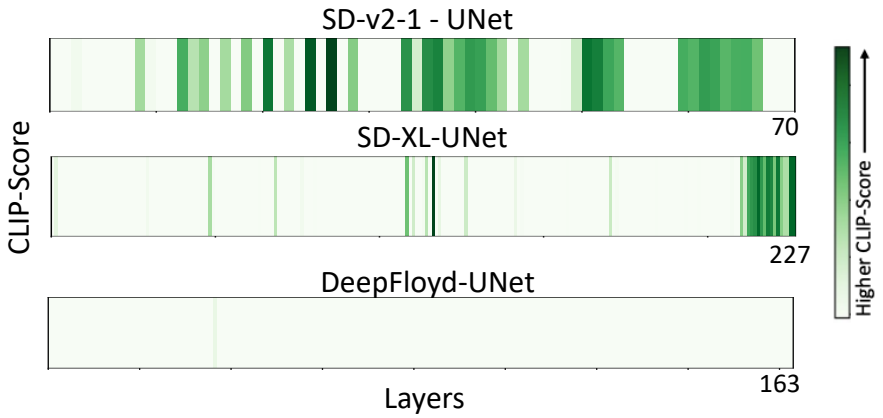

Figure 5.7: **Causal tracing for UNet.** Similar to [23], we find that knowledge is causally distributed across the UNet for text-to-image models such as SD-v2-1 and SD-XL. For DeepFloyd we do not observe any significant causal state in the UNet.

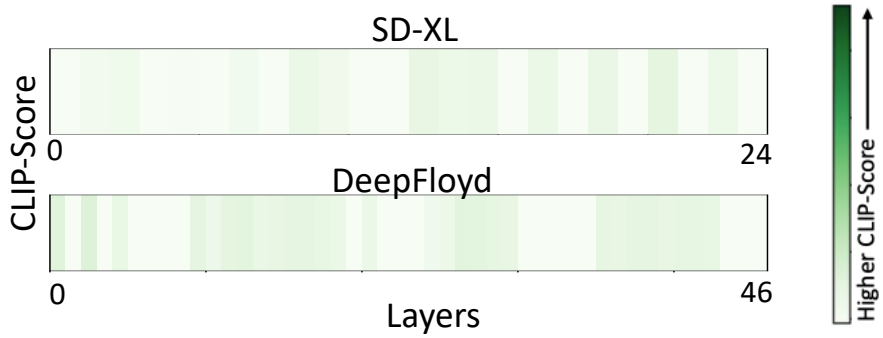

Figure 5.8: **Causal tracing for text-encoder.** Unlike SD-v1-5 and SD-v2-1, we find that a singular causal states does not exist in the text-encoder for SD-XL and DeepFloyd.

Causal Tracing in Text-Encoder. [23] show that there exists a unique causal state in the text-encoder for Stable-Diffusion-v1-5 and Stable-Diffusion-v2-1 which can be used to perform fast model editing. In Figure 5.8, we find that such an unique causal state is absent in the text-encoder for DeepFloyd and SD-XL. We note that DeepFloyd uses a T5-text encoder, whereas SD-XL uses a a combination of CLIP-ViT-L and OpenCLIP-ViT-G [191]. Our empirical results indicate that an unique causal state arises only when a CLIP text-encoder is used by itself in a text-to-image model.

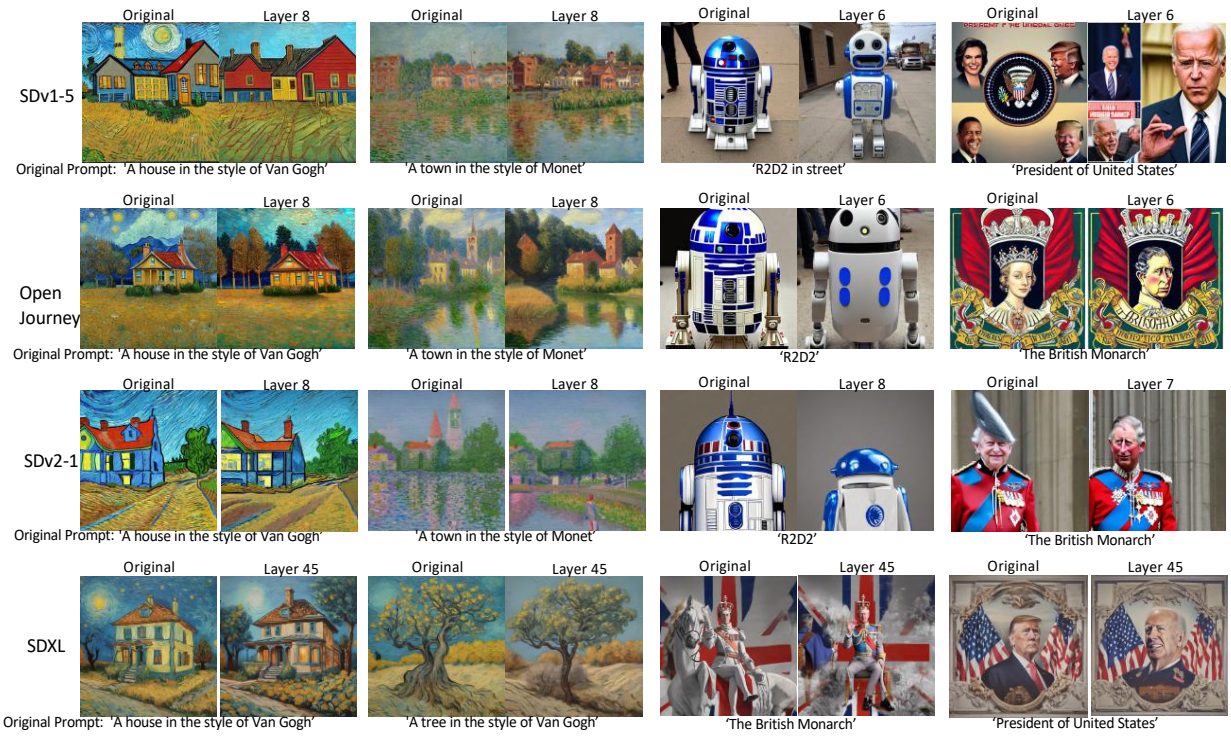

Figure 5.9: **Interpretability Results: Images generated by intervening on the layers identified by LocoGen across various open-source text-to-image models.** We compare the original generation vs. generation by intervening on the layers identified with LocoGen along with a target prompt. We find that across various text-to-image models, visual attributes such as *style, objects, facts* can be manipulated by intervening only on a very small fraction of cross-attention layers.

5.8 LocoGen: Towards Mechanistic Knowledge Localization

Given the lack of generalizability of knowledge localization using causal tracing as shown in Sec. 5.7, we introduce LocoGen, which can identify localized control regions for visual attributes across *all* text-to-image models.

5.8.1 Knowledge Control in Cross-Attention Layers

During the inference process, the regulation of image generation involves the utilization of classifier-free guidance, as outlined in [99] which incorporates scores from both the conditional and unconditional diffusion models at each time-step. Specifically, the classifier-free guidance is applied at each time-step to combine the conditional ($\varepsilon_\theta(\mathbf{z}_t, \mathbf{c}, t)$) and unconditional score estimates ($\varepsilon_\theta(\mathbf{z}_t, t)$). The result is a combined score denoted as $\hat{\varepsilon}(\mathbf{z}_t, \mathbf{c}, t)$.

$$\hat{\varepsilon}(\mathbf{z}_t, \mathbf{c}, t) = \varepsilon_\theta(\mathbf{z_t}, \mathbf{c}, t) + \alpha\left(\varepsilon_\theta\left(\mathbf{z_t}, \mathbf{c}, t\right) - \varepsilon_\theta(\mathbf{z_t}, t)\right), \quad \forall t \in [T, 1]\ . \tag{5.10}$$

This combined score is used to update the latent $\mathbf{z}_t$ using DDIM sampling [216] at each time-step to obtain the final latent code $\mathbf{z}_0$. We term the model $\varepsilon_\theta(\mathbf{z}_t, \mathbf{c}, t)$ as the `Clean Model` and the final image generated as I_{clean}. We note that text is incorporated in the process of generation using cross-attention layers denoted by $\{C_l\}_{l=1}^{M}$ within $\varepsilon_\theta(\mathbf{z}_t, \mathbf{c}, t)\ \forall t \in [T, 1]$. These layers include key and value matrices – $\{W_l^K, W_l^V\}_{l=1}^{M}$ that take text-embedding $\mathbf{c}$ of the input prompt and guide the generation toward the text prompt. Generally, the text-embedding $\mathbf{c}$ is same across all these layers. However, in order to localize and find control points for different visual attributes, we replace the original text-embedding $\mathbf{c}$ with a target prompt embedding $\mathbf{c}'$ across a small subset of the cross-attention layers and measure its direct effect on the generated image.

Altered Inputs

We say that a model receives *altered input* when a subset of cross-attention layers $C' \subset \{C_l\}_{l=1}^{M}$ receive a different text-embedding $\mathbf{c}'$ than the other cross-attention layers that take $\mathbf{c}$ as input. We name these layers as *controlling layers*. We denote by I_{altered} the image generated using this model and Eq. (5.10) with altered inputs when $\mathbf{z}_T$ is given as the initial

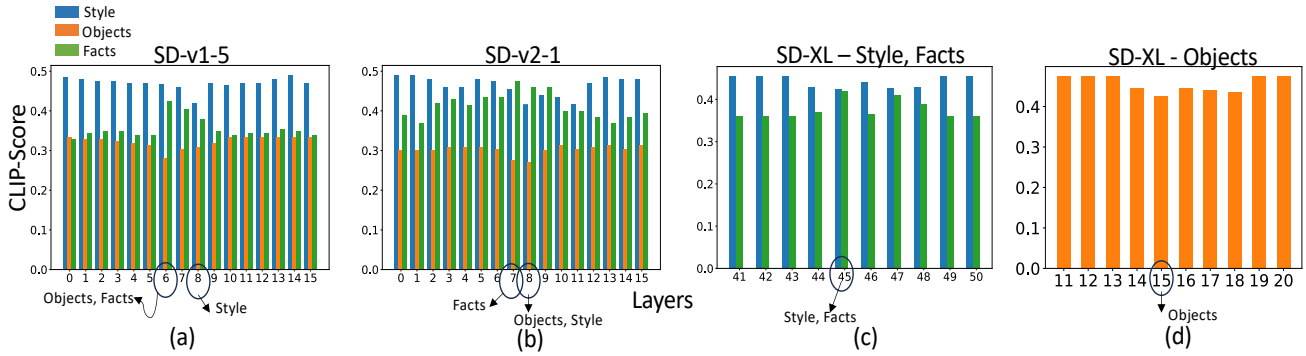

Figure 5.10: **CLIP-Score of the generated images with original prompt for** *style, objects* **and target prompt for** *facts* **after intervening on layers through LOCOGEN.** Lower CLIP-Score for *objects, style* indicate correct localization, whereas a higher CLIP-Score indicates such for *facts*. (a) For SD-v1-5 (m=2), *objects, facts* can be controlled from Layer 6, whereas *style* can be controlled from Layer 8. (b) For SD-v2-1(m=3), *facts* are controlled from Layer 7, *style* and *objects* from Layer 8. (c,d): For SD-XL, *style (m=3), facts(m=5)* are controlled from Layer 45, whereas *objects* are controlled from Layer 15.

noise. We denote the model $\varepsilon_\theta(\mathbf{z}_t, \mathbf{c}, \mathbf{c}', t)$ with the altered inputs as the `Altered Model` with the following inference procedure:

$$\hat{\varepsilon}(\mathbf{z}_t, \mathbf{c}, \mathbf{c}', t) = \varepsilon_\theta(\mathbf{z_t}, \mathbf{c}, \mathbf{c}', t) + \alpha(\varepsilon_\theta(\mathbf{z_t}, \mathbf{c}, \mathbf{c}', t) - \varepsilon_\theta(\mathbf{z_t}, t)) \ .$$

As an example, to find the layers where *style* knowledge corresponding to a particular artist is stored, $\{C_l\}_{l=1}^{M} - C'$ receive text-embeddings corresponding to the prompt *'An <object> in the style of <artist>'*, whereas the layers in C' receive text-embeddings corresponding to the prompt *'An <object> in the style of painting'*. If the generated image with these inputs do not have that particular style, we realize that controlling layers C' are responsible for incorporating that specified style in the output (see Figure 8.1). In fact, this replacement operation enables finding locations across different cross-attention layers where various visual attribute knowledge is localized.

LOCOGEN Algorithm

Our goal is to find controlling layers C' for different visual attributes. We note that the cardinality of the set $|C'|= m$ is a hyper-parameter and the search space for C' is exponential. Given $|C'|= m$, there are $\binom{M}{m}$ possibilities for C', thus, we restrict our search space to only adjacent cross-attention layers. In fact, we consider all C' such that $C' = \{C_l\}_{l=j}^{j+m-1}$ for $j \in [1, M-m+1]$.

Selecting the hyper-parameter m. To select the cardinality of the set C', we run an iterative hyper-parameter search with $m \in [1, M]$, where M is selected based on the maximum number of cross-attention layers in a given text-to-image generative model. At each iteration of the hyper-parameter search, we investigate whether there exists a set of m adjacent cross-attention layers that are responsible for the generation of the specific visual attribute. We find minimum m that such controlling layers for the particular attribute exists.

To apply LOCOGEN for a particular attribute, we obtain a set of input prompts $T = \{T_i\}_{i=1}^{N}$ that include the particular attribute and corresponding set of prompts $T' = \{T_i'\}_{i=1}^{N}$ where T_i' is analogous to T_i except that the particular attribute is removed/updated. These prompts serve to create altered images and assess the presence of the specified attribute within them. Let $\mathbf{c}_i$ be the text-embedding of T_i and $\mathbf{c}_i'$ be that of T_i'.

Given m, we examine all $M-m+1$ possible candidates for controlling layers. For each of them, we generate N altered images where i-th image is generated by giving $\mathbf{c}_i'$ as the input embedding to selected m layers and $\mathbf{c}_i$ to other ones. Then we measure the CLIP-Score [97] of original text prompt T_i to the generated image for *style, objects* and target text prompt T_i' to the generated image for *facts*. For *style* and *objects*, drop in CLIP-Score shows the removal of the attribute while for *facts* increase in score shows similarity to the updated

fact. We take the average of the mentioned score across all $1 \leq i \leq N$. By doing that for all candidates, we report the one with minimum average CLIP-Score for *style, objects* and maximum average CLIP-Score for *facts*. These layers could be candidate layers controlling the generation of the specific attribute. Algorithm 6 provides the pseudocode to find the best candidate. Figure 5.10 shows CLIP-Score across different candidates.

Algorithm 2 LOCOGEN

Input: $m, \{T_i\}_{i=1}^{N}, \{T_i'\}_{i=1}^{N}, \{\mathbf{c}_i\}_{i=1}^{N}, \{\mathbf{c}_i'\}_{i=1}^{N}$
Output: Candidate controlling set
  for $j \leftarrow 1, \ldots, M-m$ **do**
    $C' \leftarrow \{C_l\}_{l=j}^{j+m-1}$
    for $i \leftarrow 1, \ldots, N$ **do**
      $s_i \leftarrow \text{CLIP-SCORE}\left(T_i, I_{\text{altered}}\right)$
      $s_i' \leftarrow \text{CLIP-SCORE}\left(T_i', I_{\text{altered}}\right)$
    end for
    $a_j \leftarrow \text{AVERAGE}\left(\{s_i\}_{i=1}^{N}\right)$ ▷ for objects, style
    $a_j \leftarrow \text{AVERAGE}\left(\{s_i'\}_{i=1}^{N}\right)$ ▷ for facts
  end for
  $j^* \leftarrow \arg\min_j a_j$ ▷ for objects, style
  $j^* \leftarrow \arg\max_j a_j$ ▷ for facts
  return $a_{j^*}, \{C_l\}_{l=j^*}^{j^*+m-1}$

We set a threshold for average CLIP-Score and find the minimum m such that there exists m adjacent cross-attention layers whose corresponding CLIP-Score meets the requirement. We point the reader to Appendix for the values of m selected for different models and thresholds.

Dataset for Prompts. We use the prompts used in [23, 126] to extract locations in the UNet which control for various visual attributes such as *objects, style* and *facts*. More details in Appendix.

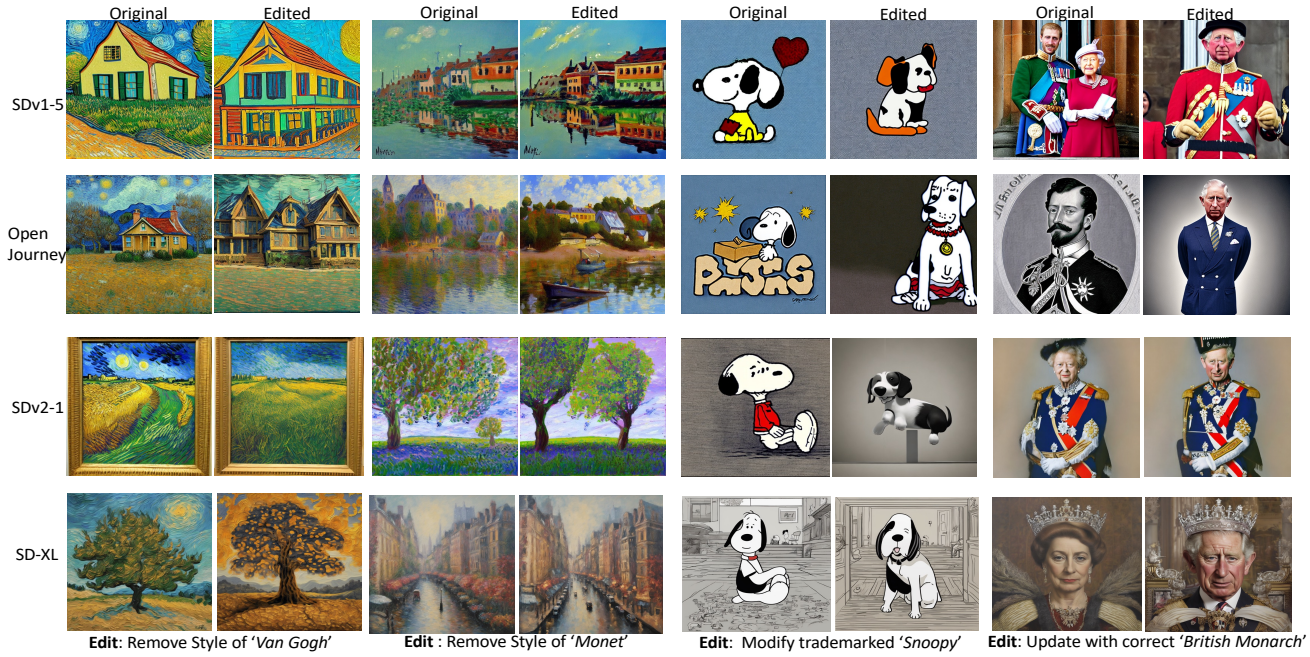

Figure 5.11: **LOCOEDIT (Model editing) results at locations identified by LOCOGEN across various open-source text-to-image models.** We observe that locations identified by our interpretability framework can be edited effectively to remove *styles*, *objects* and update *facts* in text-to-image models. We provide more visualizations in Appendix.

Figure 5.12: **Interpretability Results for DeepFloyd.** We find the control points for visual attributes to be dependent on the underlying prompts, rather than the visual attribute.

5.8.2 Empirical Results

In this section, we provide empirical results highlighting the localized layers across various open-source text-to-image generative models:

Stable-Diffusion Variants. Across both models, as depicted qualitatively in Figure 5.9 and quantitatively in Figure 5.10-(a), we observe the presence of a distinctive subset of layers that govern specific visual attributes. In the case of both SD-v1-5 and SD-v2-1, the control for “style" is centralized at $l = 8$ with $m = 2$. In SD-v1-5, the control for “objects" and “facts" emanates from the same locations: $l = 6$ and $m = 2$. However, in SD-v2-1, “objects" are controlled from $l = 8$, while “facts" are influenced by $l = 7$. Despite sharing a similar UNet architecture and undergoing training with comparable scales of pre-training data, these models diverge in the text-encoder utilized. This discrepancy in text-encoder choice may contribute to the variation in how they store knowledge concerning different attributes.

Open-Journey. We note that Open-Journey exhibits control locations similar to SD-v1-5 for various visual attributes. As illustrated in Figure 5.9 and Figure 5.10-(a), “objects" and “facts" are governed from $l = 6$, while “style" is controlled from $l = 8$. Despite the architectural resemblance between Open-Journey and SD-v1-5, it’s important to highlight that Open-Journey undergoes fine-tuning on a subset of images generated from Mid-Journey. This suggests that the control locations for visual attributes are more closely tied to the underlying model architecture than to the specifics of the training or fine-tuning data.

SD-XL. Within SD-XL, our investigation reveals that both “style" and “facts" can be effectively controlled from $l = 45$, with $m = 3$ as evidenced in Figure 5.9 and Figure 5.10-(c). For the attribute “objects," control is situated at $l = 15$, albeit with a slightly larger value of $m = 5$. In summary, SD-XL, consisting of a total of 70 cross-attention layers, underscores

a significant finding: **various attributes** in image generation can be governed by only **a small subset of layers**.

DeepFloyd. Across SD-v1-5, SD-v2-1, Open-Journey, and SD-XL, our findings indicate that visual attributes like “style", “objects" and “facts," irrespective of the specific prompt used, can be traced back to control points situated within a limited number of layers. However, in the case of DeepFloyd, our observations differ. We find instead, that all attributes display localization dependent on the specific prompt employed. To illustrate, factual knowledge related to “The British Monarch" is governed from $l = 6$ with $m = 3$, whereas factual knowledge tied to “The President of the United States" is controlled from $l = 12$ (see Figure 5.12). This divergence in localization patterns highlights the nuanced behavior of DeepFloyd in comparison to the other models examined. More results can be referred in Appendix.

Human-Study Results. We run a human-study to verify that LOCOGEN can effectively identify controlling layers for different visual attributes. In our setup, evaluators assess 132 image pairs, each comprising an image generated by `Clean Model` and an image generated by `Altered Model` whose identified cross-attention layers takes different inputs. Evaluators determine whether the visual attribute is changed in the image generated by `Altered Model`(for instance, the artistic Van Gogh style is removed from the original image or not). Covering 33 image pairs, generated with different prompts per model, with five participating evaluators, our experiments reveal a 92.58% verification rate for the impact of LOCOGEN-identified layers on visual attributes. See more details in Appendix

5.9 LOCOEDIT: Editing to Ablate Concepts

In this section, we analyse the effectiveness of edits in the layers identified by LOCOGEN across text-to-image models.

5.9.1 Method

Algorithm 6 extracts the exact set of cross-attention layers from which the knowledge about a particular visual attribute (e.g., *style*) is controlled. We denote this set as C_{loc}, where $C_{\text{loc}} \subset C$ and $|C_{\text{loc}}| = m$. This set of extracted cross-attention layers C_{loc}, each containing value and key matrices is denoted as $C_{\text{loc}} = \{\hat{W}_l^K, \hat{W}_l^V\}_{l=1}^m$. The objective is to modify these weight matrices $\{\hat{W}_l^K, \hat{W}_l^V\}_{l=1}^m$ such that they transform the original prompt (e.g., '*A house in the style of Van Gogh*') to a target prompt (e.g., '*A house in the style of a painting*') in a way that the visual attribute in the generation is modified. Similar to Section 5.8.1, we use a set of input prompts $T_{\text{orig}} = \{T_i^o\}_{i=1}^N$ consisting of prompts featuring the particular visual attribute. Simultaneously, we create a counterpart set $T_{\text{target}} = \{T_i^t\}_{i=1}^N$ where each T_i^t is identical to T_i^o but lacks the particular attribute in focus. Let $\mathbf{c}_i^o \in \mathbb{R}^d$ be the text-embedding of the last subject token in T_i^o and $\mathbf{c}_i^t \in \mathbb{R}^d$ be that of T_i^t. We obtain matrix $\mathbf{X}_{\text{orig}} \in \mathbb{R}^{N \times d}$ by stacking vectors $\mathbf{c}_1^o, \mathbf{c}_2^o, \ldots, \mathbf{c}_N^o$ and matrix $\mathbf{X}_{\text{target}} \in \mathbb{R}^{N \times d}$ by stacking $\mathbf{c}_1^t, \mathbf{c}_2^t, \ldots, \mathbf{c}_N^t$. To learn a mapping between the key and the value embeddings, we solve the following optimization for each layer $l \in [1, m]$ corresponding to the key matrices as:

$$\min_{W_l^K} \|\mathbf{X}_{\text{orig}} W_l^K - \mathbf{X}_{\text{target}} \hat{W}_l^K\|_2^2 + \lambda_K \|W_l^K - \hat{W}_l^K\|_2^2$$

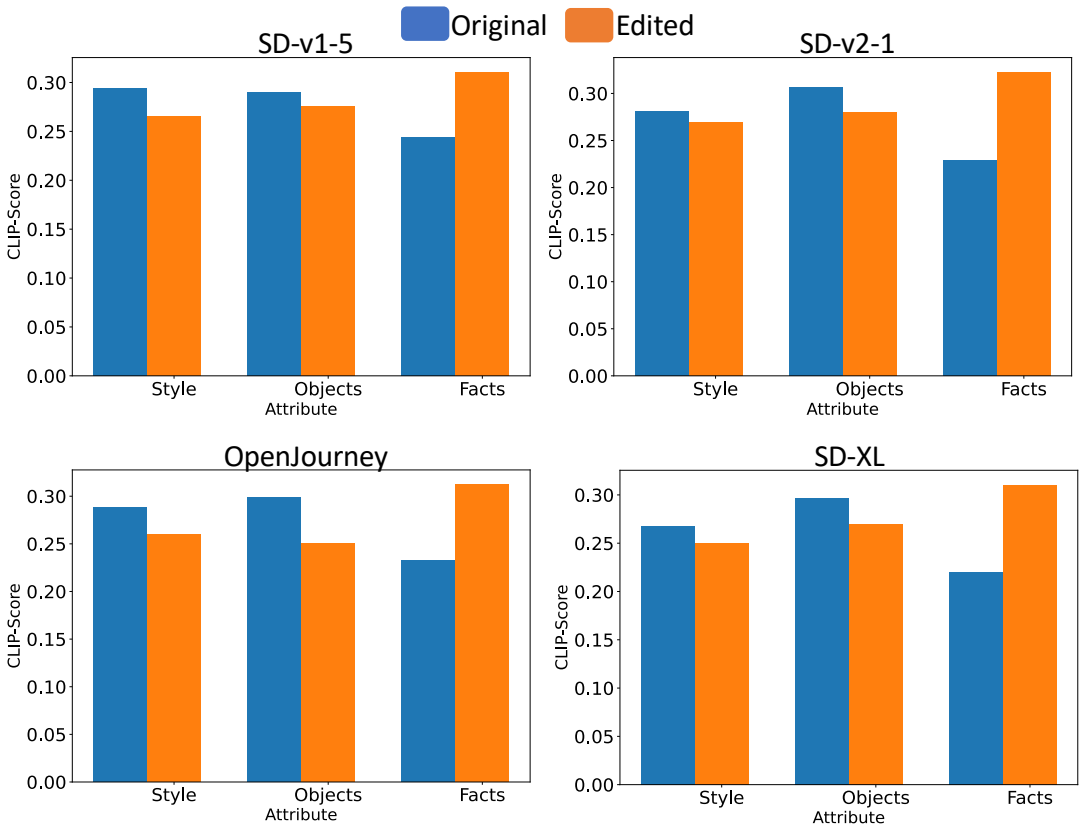

Figure 5.13: **Quantitative Model Editing Results for Text-to-Image Models.** We observe a drop in CLIP-Score for “style" and "objects", while an increase in CLIP-Score for “facts" therefore highlighting correct edits.

where λ_K is the regularizer. Letting $\mathbf{Y}_{\text{orig}} = \mathbf{X}_{\text{orig}} W_l^K$ the optimal closed form solution for the key matrix is:

$$W_l^K = (\mathbf{X}_{\text{orig}}^T \mathbf{X}_{\text{orig}} + \lambda_1 I)^{-1} (\mathbf{X}_{\text{orig}}^T \mathbf{Y}_{\text{target}} + \lambda_K \hat{W}_l^K)$$

Same is applied to get optimal matrix for value embeddings.

5.9.2 Model Editing Results

Stable-Diffusion Variants, Open-Journey and SD-XL. In Figure 5.11 and Figure 5.13, it becomes apparent that LOCOEDIT effectively integrates accurate edits into the locations identified by LOCOGEN. Qualitatively examining the visual edits in Figure 5.11, our method demonstrates the capability to remove artistic “styles", modify trademarked “objects," and update outdated “facts" within a text-to-image model with accurate information. This visual assessment is complemented by the quantitative analysis in Figure 5.13, where we observe that the CLIP-Score of images generated by the edited model, given prompts containing specific visual attributes, consistently registers lower than that of the clean model for “objects" and “style." For “facts," we gauge the CLIP-Score of images from the model with the correct facts, wherein a higher CLIP-Score indicates a correct edit, as illustrated in Figure 5.13. Combining both qualitative and quantitative findings, these results collectively underscore the effectiveness of LOCOEDIT across SD-v1-5, SD-v2-1, Open-Journey, and SD-XL. However, it’s noteworthy that the efficacy of closed-form edits varies among different text-to-image models. Specifically, in the case of “style," we observe the most substantial drop in CLIP-Score between the edited and unedited models for SD-v1-5 and Open-Journey, while the drop is comparatively less for SD-v2-1 and SD-XL. Conversely, for “facts," we find that all models perform similarly in updating with new information.

Limitations with DeepFloyd Closed-Form Edits. DeepFloyd, despite revealing distinct locations through LOCOGEN (albeit depending on the underlying prompt), exhibits challenges in effective closed-form edits at these locations. The section on DeepFloyd limitations in the Appendix provides qualitative visualizations illustrating this limitation. The model employs a T5-encoder with bi-directional attention, diverging from other text-to-image models using CLIP-variants with causal attention. Closed-form edits, relying on mapping the last-subject token embedding to a target embedding, are typically effective in text-embeddings generated with causal attention, where the last-subject token holds crucial information. However, the T5-encoder presents a hurdle as tokens beyond the last subject token contribute essential information about the target attribute. Consequently, restricting the mapping to the last-subject token alone proves ineffective for a T5-encoder.

While LOCOGEN along with LOCOEDIT makes model editing more interpretable – we also find that localized-model editing is better than updating all layers in the UNet as shown in Appendix. We also compare our method with existing editing methods [23, 78, 126] in Appendix. We find that our editing method is at par with existing baselines, with the added advantage of generalizability to models beyond Stable-Diffusion-v1-5. In Appendix, we also show the robustness of our method to generic prompts.

5.10 On Neuron-Level Model Editing

In this section, we explore the feasibility of effecting neuron-level modifications to eliminate stylistic attributes from the output of text-to-image models. According to layers identified with LOCOGEN, our objective is to ascertain whether the selective dropout of neurons at the activation layers within the specified cross-attention layers (key and value embeddings) can successfully eliminate stylistic elements.

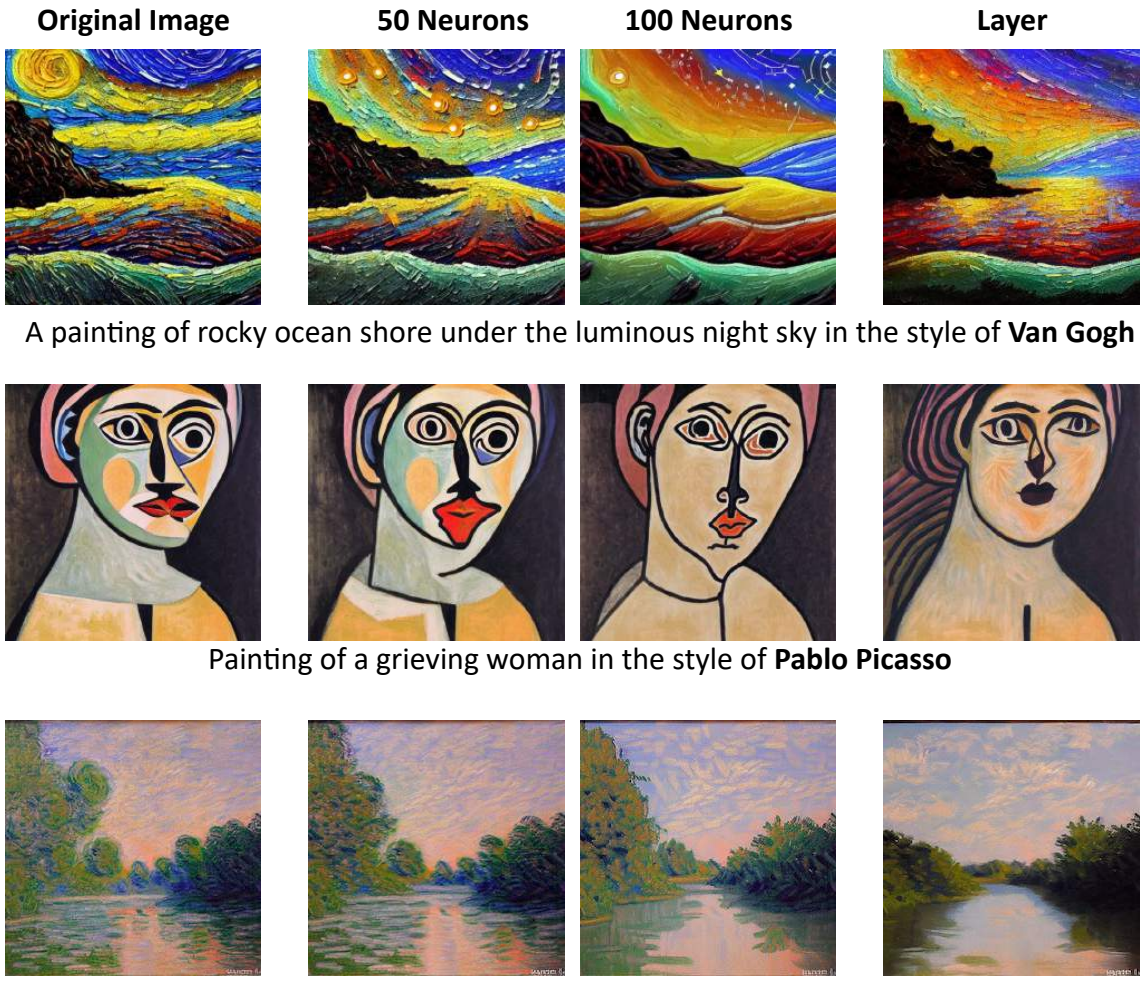

Figure 5.14: **Neuron-Level Model Editing - Qualitative**. Results when applying **neuron-level dropout** on identified neurons in layers specified with LOCOGEN on Stable Diffusion v1.5. The second and third columns display images with 50 and 100 modified neurons out of 1280 in controlling layers. The last column shows images with a different embedding in controlling layers.

To accomplish this objective, we first need to identify which neurons are responsible for the generation of particular artistic styles, e.g., *Van Gogh*. We examine the activations of neurons in the embedding space of key and value matrices in identified cross-attention layers. More specifically, we pinpoint neurons that exhibit significant variations when comparing input prompts that include a particular style with the case that input prompts do not involve the specified style.

To execute this process, we collect a set of N_1 prompts that feature the specific style, e.g. Van Gogh. We gather text-embeddings of the last subject token of these prompts denoted by $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_{N_1}$, where $\mathbf{c}_i \in \mathbb{R}^d$. We also obtain a set of N_2 prompts without any particular style and analogously obtain $\{\mathbf{c}'_1, \mathbf{c}'_2, \ldots, \mathbf{c}'_{N_2}\}$, where $\mathbf{c}'_i \in \mathbb{R}^d$. Next, for the key or value matrix $W \in \mathbb{R}^{d \times d'}$, we consider key or value embedding of these input prompts, i.e., $\{z_i\}_{i=1}^{N_1} \cup \{z'_i\}_{i=1}^{N_2}$ where $z_i = \mathbf{c}_i W$ and $z'_i = \mathbf{c}'_i W$. We note that $z_i, z'_i \in \mathbb{R}^{d'}$.

Subsequently, for each of these d' neurons, we assess the statistical difference in their activations between input prompts that include a particular style and those without it. Specifically, we compute the z-score for each neuron within two groups of activations: $z_1, z_2, \ldots, z_{N_1}$ and $z'_1, z'_2, \ldots, z'_{N_2}$. The neurons are then ranked based on the absolute value of their z-score, with the top neurons representing those that exhibit significant differences in activations depending on the presence or absence of a particular concept in the input prompt. During generation, we drop-out these neurons and see if particular style is removed or not. As seen in Figure 5.14, neuron-level modification at inference time is effective at removing styles. This shows that knowledge about a particular style can be even more localized to a few neurons. It is noteworthy that the extent of style removal increases with the modification of more neurons, albeit with a trade-off in the quality of generated images. This arises because modified neurons may encapsulate information related to other visual attributes. To quantify the effectiveness of this approach, we measure the drop in CLIP-Score for

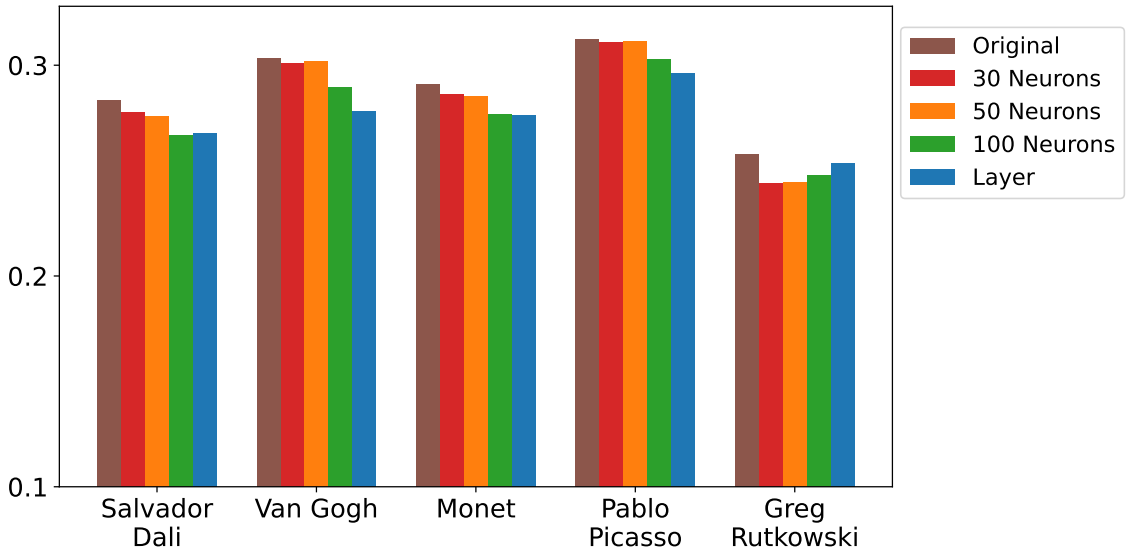

Figure 5.15: **Neuron-Level Model Editing - Quantitative**. Average CLIP-Score of generated images to text prompt *'style of <artist>'*. Brown bars show similarity to original generated image; red, orange, and green bars show similarity to generated image when 30, 50, and 100 neurons are modified, respectively; and blue bars refer to images when controlling layers receive other prompt.

modified images across various styles. Figure 5.15 presents a bar-plot illustrating these similarity scores. Notably, drop in CLIP-Score demonstrates that neuron-level model editing effectively removes the styles associated with different artists in the generated images. We refer to Appendix for more details on neuron-level model editing experiments.

5.11 Conclusion II

In our paper, we comprehensively examine knowledge localization across various open-source text-to-image models. We initially observe that while causal tracing proves effective for early Stable-Diffusion variants, its generalizability diminishes when applied to newer text-to-image models like DeepFloyd and SD-XL for localizing control points associated with visual attributes. To address this limitation, we introduce LOCOGEN, capable of effectively identifying locations within the UNet across diverse text-to-image models. Harnessing

these identified locations within the UNet, we evaluate the efficacy of closed-form model editing across a range of text-to-image models leveraging LOCOEDIT, uncovering intriguing properties. Notably, for specific visual attributes such as “style", we discover that knowledge can even be traced to a small subset of neurons and subsequently edited by applying a simple dropout layer, thereby underscoring the possibilities of neuron-level model editing.

Chapter 6: **Mechanistically Understanding and Editing Multimodal Language Models**

6.1 Introduction

Multi-modal Large Language Models (MLLMs) trained on both text and images are rapidly moving from research into deployment and are being used by millions of people. Yet, while there have been some advances in understanding how llm work, much less has been done to understand mllm. This paper begins to close this gap by studying how information is stored and transferred in mllm. To do this through the lens of a factual Visual Question Answering (VQA) task – a very common use case of MLLMs today [179, 180, 254].

MLLMs process factual information in two steps: information storage and information transfer. Information storage refers to how facts from a pre-training dataset are stored in a model's parameters – its so-called 'parametric' memory. Information transfer describes how information from input prompts are propagated through these storage locations to the model's final output. Understanding these mechanisms can have many benefits, including ensuring models are factually grounded and informing better evaluation protocols. Extensive works have explored how LLMs store and transfer factual information [112, 167, 207], however, this has not been studied for multi-modal inputs. For example, it is suggested that auto-regressive Transformer-based LLMs store factual information in their mid-layer

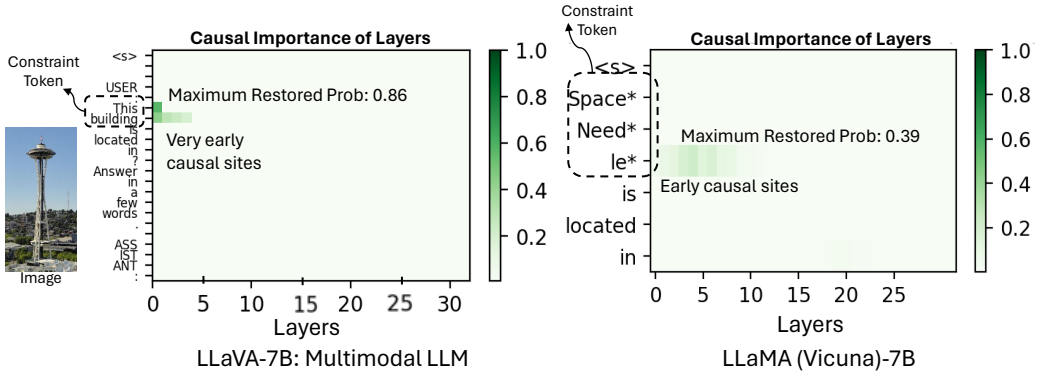

Figure 6.1: **MLLMs retrieve information from earlier internal layers compared to their Large Language Model (LLM) counterparts.** We find that very early MLP layers [1-4] have high indirect estimation effects to outputs (i.e., they are causal) in LLaVa-7B, whereas the middle MLP layers [4-7] are causal in LLaMA (Vicuna)-7B. For LLaMA, a larger window size (e.g., 5) is also required to find causal sites, compared to a window size of 1 for LLaVA-7B.

MLP parameters [112, 167]. However, MLLMs and LLMs are different: an MLLM involves an additional (continuous) image input, alongside a (discrete) text prompt, and requires additional modules to process it [53, 138, 148, 149, 179]. Typically, a vision encoder (e.g. CLIP) is used to convert this image into either visual tokens via a projection layer [94, 148, 149] or cross-attention layers [8, 53, 138, 260] which are then integrated into the language encoder. These differences suggest that our existing understanding of LLM information storage and transfer may not map directly to MLLMs.

In this work, we use a factual VQA task to study the mechanisms of multi-modal information storage and transfer. We use a constraint-based formulation which views a visual question as having a set of either visual or textual constraints (e.g. What movie directed by *this director* has won a *Golden Globe*?). The information retrieved by the model should satisfy these constraints (e.g. *this director*, *Golden Globe*) for the answer to be factually correct. This formulation, therefore, offers a systematic way of understanding a model's behavior. Under this framework, we propose MULTIMODALCAUSALTRACE, which extends LLM causal tracing [112, 184, 237] to the multi-modal setting, to understand information storage, as

well as leverage attention contribution methodologies [68, 263] to study information transfer in MLLMs. We also introduce *VQA-Constraints*, a new dataset of 9.7k factual questions annotated with constraints, spanning natural images (from OK-VQA [161], WikiMovies [263], and Known [112]). With these tools, we study how a widely-used MLLM family processes multi-modal information, specifically LLaVa [148, 149] and multi-modal Phi-2 [94][1].

Our key findings are that MLLMs, in contrast to LLMs: (i) retrieve information from earlier MLP layers (i.e. layers 1-4 vs layers 4-7 in a LLM) (see Fig. 8.1); and (ii) use less parametric memory (require a smaller window size to retrieve this information), when answering a multi-modal question. We also find that information is transferred from a given image to these early MLP blocks through a consistent subset of visual tokens (e.g. last $\sim$36 tokens from LLaVa's CLIP encoder), and that the self-attention blocks in the middle layers are primarily responsible for shuttling this information to the last token.

Finally, we demonstrate that by editing these early causal MLPs, we can correct errors and insert new factual information into an MLLM. Specifically, we propose a new model-editing algorithm, MULTEDIT, which modifies the projection matrix of the early causal MLPs with a closed-form update. We empirically show MULTEDIT's effectiveness on questions from *VQA-Constraints* and Encyclopedic VQA [168].

In summary, our contributions are:

- A novel multi-modal causal tracing methodology that can be used to study information storage from image and text inputs in MLLMs.
- A new dataset, *VQA-Constraints*, of 9.7K factual visual questions about natural images

[1]Taken from the Bunny repository.

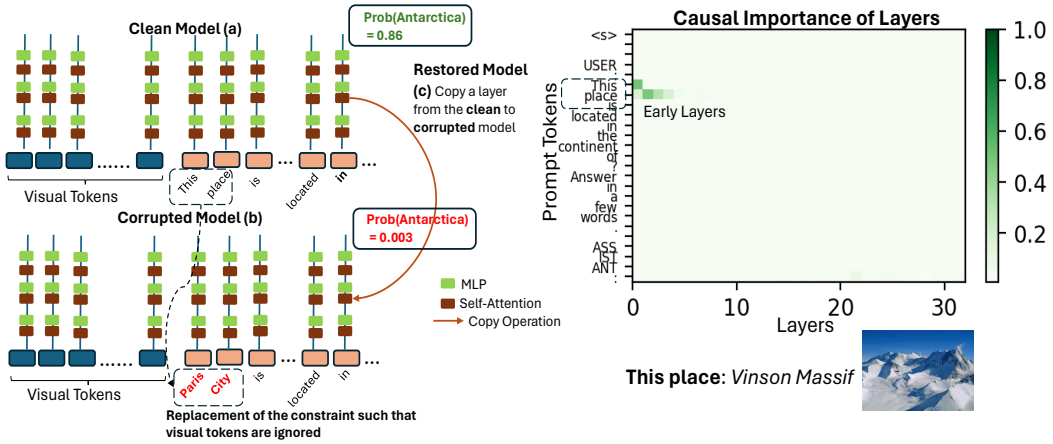

Figure 6.2: **We introduce MULTIMODALCAUSALTRACE, a causal tracing method to understand information storage in MLLMs.** A clean model is corrupted by replacing the question's constraint with an incorrect one for the given image (e.g. "This place" –> "Paris city" for an image of "Vinson Massif"). The activations of windows of layers are then iteratively copied from the clean to the corrupted model until the corrupted model restores its output probability to match the clean model's.

annotated with constraints to support future research in these directions.

- A suite of novel insights on the mechanisms underlying multi-modal information storage and retrieval in MLLMs.
- A model-editing method, MULTEDIT, which we demonstrate can precisely correct erroneous information and insert new long-tailed information in an MLLM.

6.2 A Constraint-Based Framework for Studying Information Storage and Transfer in MLLMs

In this section, we describe the constraint-based formulation we use to study information storage and transfer in MLLMs. Under this framing, we introduce MULTIMODALCAUSALTRACE, a novel causal information tracing technique which we use to study multi-modal information storage. We also describe how we use attention contributions [68] to study multi-modal information transfer. Finally, we describe *VQA-Constraints*, a new test-bed of visual questions annotated with constraints.

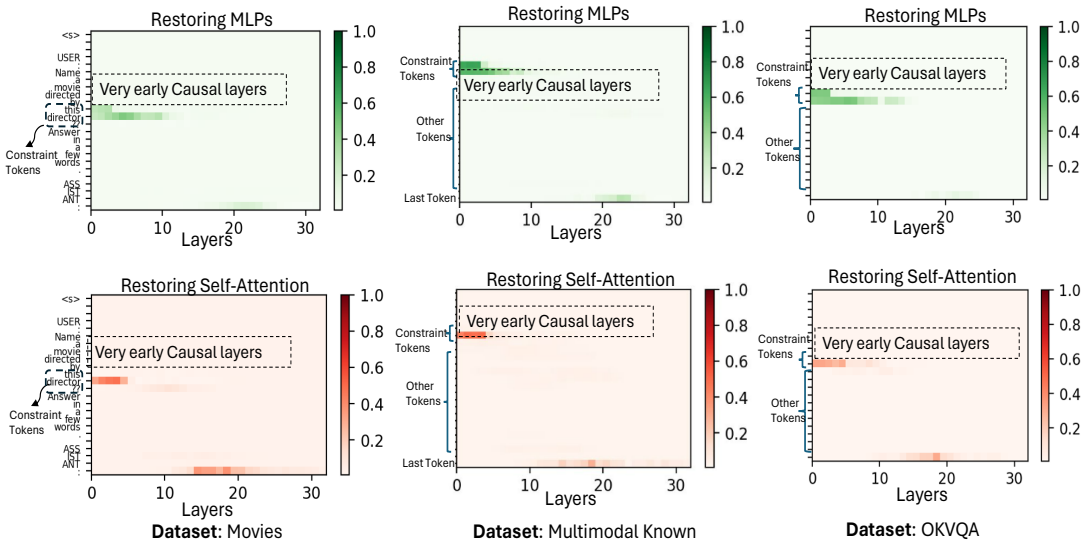

Figure 6.3: **Information to answer a visual question with a single constraint is mainly retrieved from early MLP and self-attention layers in MLLMs.** MULTIMODALCAUSALTRACE obtains high *indirect estimation effect* values in LLaVa's early MLP and self-attention blocks corresponding to the visual constraint, across all 3 datasets in *VQA-Constraints*. This suggests these layers are causally important for information storage. The causal traces emerge with a window size of 3 (see results with a window size of 1 in Appendix.

6.2.1 A Multi-modal Constraint-based Framework

Prior works have used a constraint satisfaction framework to study LLMs [130, 154, 263] as they offer a systematic way of studying model behavior. This framing defines a constraint as a set of words in the question. The model must retrieve information that satisfies this constraint from its parametric memory in order to generate the correct answer. For example, for the question "What city is the *Space Needle* in?", the model must retrieve information relevant to the constraint "Space Needle". In the multi-modal setting, we consider these textual constraints as well as introduce visual constraints. We define a visual constraint to be a set of words in the question which refers to an entity in the image. The model must similarly retrieve information about this entity to generate the correct answer. For example, given an image of Christopher Nolan and the question "Name a movie directed by *this director* in *2006*?", the visual constraint is "*this director*" (and the text constraint is "*2006*").

We refer to a question involving both a visual and text constraint as a multi-constraint question, and ones with only a visual constraint as a single-constraint question.

We use this framework to study the widely-used "projection layer" MLLMs family. These models are composed of a visual encoder f_θ, a large language model g_ϕ and a projection head p_γ. The projection head p_γ is responsible for mapping the output of the visual encoder into the input space of the language model, as so-called "visual tokens". Given an image-text pair denoted as (**x**, y), the language model processes them as $g_\phi(p_\gamma(f_\theta(\mathbf{x})), h(t(y)))$, where $p_\gamma(f_\theta(\mathbf{x})) = \{\mathbf{v}_i\}_{i=1}^{N}$ is the set of visual token embeddings and $t(y) = \{t_i\}_{i=1}^{M}$ is the tokenized text inputs for the language model. These text tokens are processed by an embedding layer h to obtain text token embeddings as $\{e_i\}_{i=1}^{M} \in \mathbb{R}^d$. Because we are interested in studying the outputs of specific layers in response to specific tokens, we use $g_\phi(.)_{k,\ell}$ to denote the output layer embedding corresponding to the k$^{\text{th}}$ token position and the layer ℓ. We denote the output of a MLP layer as $g_\phi(.)_{k,\ell_{mlp}}$ and the output of a self-attention block as $g_\phi(.)_{k,\ell_{attn}}$.

6.2.2 MULTIMODALCAUSALTRACE: Studying Information Storage in MLLMs

Causal tracing, derived from the causality literature [184], can be combined with a constraint-based framework to gain causal insights on how information propagates through a model with respect to specific constraint tokens. This has been used to identify where information is stored in LLMs [165, 167] and text-to-image generative models [23]. The central idea of causal tracing is to corrupt a clean model by perturbing the input prompt. The activations of a small subset of layers are then iteratively copied from the clean model to the corrupted model, until the corrupted model restores its output probability to match the clean model's. In this way, we can identify which layers are used to retrieve information relevant to the constraints in the prompt (i.e. causally relate the input to the output).

In LLMs, the model is corrupted by adding a small amount of Gaussian noise to the embeddings of the textual constraint tokens (usually less than 5) [165]. In MLLMs, however, noise must be added to a much larger number of token embeddings – those of the visual tokens (e.g., 576 in LLaVa and 729 in multi-modal Phi-2) from the projection head, *and* the textual tokens of the visual constraint (e.g. "*this director*"). Our experiments show that this large noise injection makes it difficult to revert the MLLM to a clean state and recover relevant causal traces (see Standard Causal Tracing results in Appendix).

We, therefore, introduce MULTIMODALCAUSALTRACE (see Fig. 6.2) to address this. Rather than adding Gaussian noise to the embeddings, we instead corrupt the visual constraint token IDs by replacing them with token IDs from a separate word or phrase, such that the visual information is ignored. We illustrate this through an example. **(1) Clean Model**: Given an image **x** of the Space Needle, and the question y, "Which city is *this building* located in?", we compute the probability of the model's output O (e.g., "Seattle") as $\mathscr{P}_{clean}(O)$. **(2) Corrupted Model:** We substitute the visual constraint with an alternative such that the question does not require information from the image to be answered (e.g., "this building" is replaced with "Taj Mahal"). For a multi-constraint question which also has a textual constraint, we can either add Gaussian noise to the textual constraint tokens' embeddings (since there are only a few) or we can similarly replace the token IDs. After all replacements, we ensure that the question still makes semantic sense. We then measure the probability of original output O as $\mathscr{P}_{corr}(O)$. With the right corruption, $\mathscr{P}_{corr}(O)$ is expected to be low. **(3) Restored Model:** We then iteratively copy layer activations $g_\phi(.)_{k,\ell_{mlp}}$ and $g_\phi(.)_{k,\ell_{attn}}$ $\forall k \in [1, M+N]$ from the clean model to the corrupted model, for each layer in turn. For a given layer ℓ and token position k, we denote the restored probability as $\mathscr{P}_{restored}(O)_{k,\ell}$. After a layer is copied, we observe if $\mathscr{P}_{restored}(O)_{k,\ell}$ is high – indicating that layer ℓ has a strong causal association to the output O. In some cases, no layers have

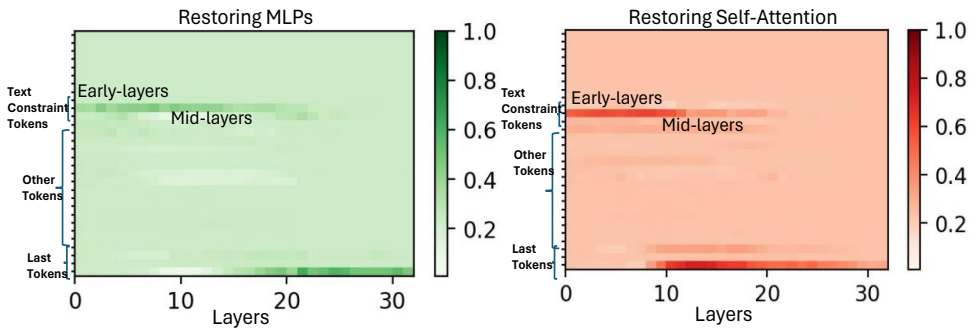

Figure 6.4: **Information to answer a visual question with a visual and textual constraint is retrieved from early *and* middle MLP and self-attention layers in MLLMs. This suggests that meeting multiple constraint requires more parametric memory compared to single constraints.** We show that MULTIMODALCAUSALTRACE obtains high indirect estimation effect values in the early and middle layers in LLaVa's on the OK-VQA dataset in *VQA-Constraints* (see multi-constraint results from the Movies dataset in Appendix.

a causal association. We note that this copy operation can be performed over a window of layers $\{\ell_i\}_{i=1}^{W}$ at a time, where W is the window size. A window size of 1 copies only one layer at a time. Similar to [165], we track the *indirect estimation effect* for a layer ℓ as $\mathscr{P}_{restored}(O)_{k,\ell} - \mathscr{P}_{corr}(O)$ and use it as a metric to track causal states. Intuitively, this measures the difference in the probability of O under the corrupted model and when a layer ℓ is restored to its original clean state. A high value indicates that the copied layer/s can restore the model to its original clean state (i.e. the layer is causal).

6.2.3 Studying Information Transfer in MLLMs with Attention Contributions

MULTIMODALCAUSALTRACE enables us to identify the specific layers a model retrieves information from in order to answering a visual question. A second component of understanding how MLLMs process factual information is understanding how input prompts are propagated through these storage locations to the model's final output. For this, we use attention contributions [68, 263] which compute how much one set of input tokens influences a set of output tokens during the self-attention operation. Specifically, we use this

to track (i) how information is transferred from the visual tokens to the causal layers, and (ii) from these layers to the final token, where the final probabilities are computed.

Defining Attention Contributions. The attention operation in a Transformer [235] consists of the query, value, key and output weight matrices: $W_q^\ell, W_v^\ell, W_k^\ell, W_o^\ell$. Each of these are divided into H heads as $W_q^{\ell,h}, W_v^{\ell,h}, W_k^{\ell,h} \in \mathbb{R}^{d \times d_h}, W_o^{\ell,h} \in \mathbb{R}^{d_h \times d}$, where d is the dimension of the internal token embeddings and d_h is the dimensionality of the token embedding for a particular attention head h. We define the attention contribution from a token j to token i in layer ℓ as follows:

$$a_{ij}^\ell = \sum_{h=1}^{H} A_{ij}^{l,h} (g_\phi(.)_{j,\ell-1} W_v^{\ell,h}) W_o^{\ell,h} \tag{6.1}$$

where the attention matrix for layer ℓ and head h is defined as:

$$A^{\ell,h} = softmax(\frac{(g_\phi(.)_{1:M+N,\ell-1} W_q^{\ell,h})(g_\phi(.)_{1:M+N,\ell-1} W_k^{\ell,h})^T}{\sqrt{d/H}}) \tag{6.2}$$

where $g_\phi(.)_{1:(M+N),\ell-1} \in \mathbb{R}^{(M+N)\times d}$ and $A^{\ell,h} \in \mathbb{R}^{(M+N)\times(M+N)}$. For understanding information transfer from the visual tokens to the causal layers, we use $j \in [1, M]$ and $i = c_{\text{constraint}}$, where $c_{\text{constraint}}$ corresponds to the last token in the visual constraint. For understanding the information transfer to the last token, we set $j = c_{\text{constraint}}$ and $i = c_{\text{last}}$, where $c_{\text{constraint}}$ corresponds to visual constraint's last token and c_{last} corresponds to the last token in the question.

6.2.4 *VQA-Constraints*: A Constraint Annotated Test-Bed for VQA

Alongside the above tools, we also introduce a new test-bed called *VQA-Constraints* to enable our analyses. The test-bed consists of $9.7K$ natural images paired with factual questions, where each question is annotated with visual and textual constraints (see Sec. 6.2.1). Specifically, we source image-question pairs from the following datasets i) OK-VQA [162]

which covers general knowledge questions, ii) Movies [263] which includes questions about movie directors and awards from Wikipedia, and iii) Known [165] which covers questions about countries, famous people, and places. The questions from the Movies and Known datasets are not originally multi-modal, so we modify them to refer to images which we source from Bing. We provide more details on the dataset construction and the constraint annotation in the dataset description in the Appendix.

We leverage GPT-4 [180] to annotate the textual and visual constraints in the visual questions in *VQA-Constraints*. We first manually annotate 100 examples from each dataset with constraints. We provide this as context to GPT-4 and prompt it (see dataset description in Appendix) to annotate the constraints in new questions from Multimodal Known and OK-VQA. Due to the templated prompts in Multimodal Movies (e.g., "Name a movie directed by *this director*? "), the constraints are constant ("*this director"*) across all examples and does not require external annotations.

6.3 Key Findings in how MLLMs Store and Transfer Information

Using the tools presented in sec. 6.2.1, we present our key findings in how MLLMs retrieve information from internal layers and how this information is transferred across the model.

6.3.1 Finding 1: Early MLPs and self-attention layers are causal

We find that information required to answer a visual question is mainly retrieved from the early-layer MLP and self-attention blocks of a MLLM. This is confirmed by the high indirect estimation effects which MULTIMODALCAUSALTRACE assigns to the early layers in both LLaVa (see Fig. 6.3, Appendix) and multi-modal Phi-2 (see Appendix) across the three datasets in *VQA-Constraints*.

This contrasts earlier results for LLMs which have been shown to retrieve information from mid-layer MLPs to answer factual questions [165, 167]. To obtain a fairer comparison, we apply MULTIMODALCAUSALTRACE to LLaMA and LLaVa which use the same language backbone. We run both on the same set of questions – LLaMA on the Known dataset [165], and LLaVa on our multi-modal version of Known which modifies the questions to refer to an image (see sec. 6.2.4). In Fig. 8.1, we show that the MLPs in the first 4 layers are causal for LLaVa while the MLPs in layers 4-7 are causal for LLaMa. We also find that causal traces can be extracted from LLaVA with a minimum window size of 1, while LLaMa requires a minimum window size of 5 to obtain any significant causal traces.

Although we find a smaller window size of 1 to provide relevant causal traces (see more trace results in the Appendix), we find that a slightly larger window size of 3 results in consistent causal traces across all the questions. In Fig. 6.3, we report the results using a window size of 3, where we find that the early MLPs as well as self-attention layers are used to retrieve relevant knowledge to answer a visual question. We note that this observation is consistent for all the datasets in *VQA-Constraints*.

For multi-constraint questions that consist of a visual and textual constraint, information corresponding to the textual constraint is retrieved from a broader set of layers – *both* the early and mid-layer MLP and self-attention blocks (see Fig. 6.4). We also find that a larger window size (at least of 6) is required to obtain any causal traces. This suggests that more parametric memory is required to meet both a visual and textual constraint in a given question.

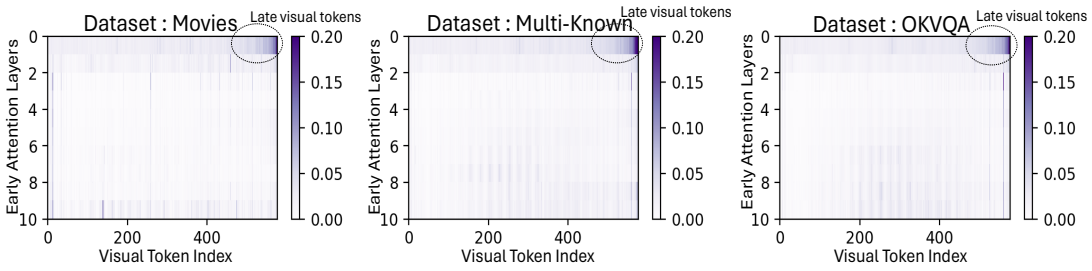

Figure 6.5: **The late visual tokens are primarily responsible for transferring information from the image to the early causal layers, via the first self-attention layer.** We visualize attention contributions (see Eq.(6.1)) from the visual tokens to the visual constraint token averaged across the three datasets in *VQA-Constraints*.

6.3.2 Finding 2: Only a subset of visual tokens are involved in transferring information from the image to the early causal MLP layers.

We also find that the late visual tokens transfer information from the image to the early causal MLP layers (where information is mainly stored) via the first self-attention layer. We show this in Fig. 6.5 by visualizing the attention values in the early self-attention layers between each visual tokens and the final token in the visual constraint using the method described in Sec. 6.2.3. We see that the values are the highest in LLaVa's first self-attention layer (which occurs just before the first causal MLP layer), specifically for the last subset of visual tokens (indexes 540-576 out of 576 in total). We hypothesize that these tokens may be summarizing image information that is relevant to the given question before it is transferred and then stored in the MLPs, however we leave this to future study. We note that this pattern holds across all three datasets from *VQA-Constraints*.

6.3.3 Finding 3: Mid-layer self-attention layers are involved in transferring information from the early causal layers to the question's final token

Rather counter-intuitively, we find that even though information is stored in the early layers, the self-attention blocks in the middle layers (rather than layers immediately after) are responsible for propagating this information to the question's final token. The model's answer is sampled at this point, hence the information present here likely influences the ultimate generation. We see this in Appendix, which plots the attention contribution values between the last visual constraint token and the last token in the question across all the layers, using the methodology in Sec. 6.2.3. For LLaVa, the self-attention blocks in layers 16-17 are most active. This behaviour is similar in LLMs which also use mid-layer self-attention blocks to transfer information from (mid-layer) stored locations to the last token position.

6.3.4 Finding 4: Mid-layer self-attention contributions can be used to predict whether a MLLM will generate a correct answer, but model confidence is a more reliable predictor

When a MLLM generates a correct answer, we observe that the self-attention contributions in its middle layers are higher to when it generates an incorrect answer (see Appendix). For LLaVa, this is specifically the attention contributions between the last constraint token and the last prompt token in the 16th and 17th layer. This holds potential to detect when a model will answer correctly without running a full inference pass, therefore enabling "early" failure mode detection.

To investigate this, we compute the scalar average of the attention contributions from these two layers as $\frac{a^{16}_{c_{\text{last}},c_{\text{constraint}}}+a^{17}_{c_{\text{last}},c_{\text{constraint}}}}{2}$ and find that it can classify a correctly generated

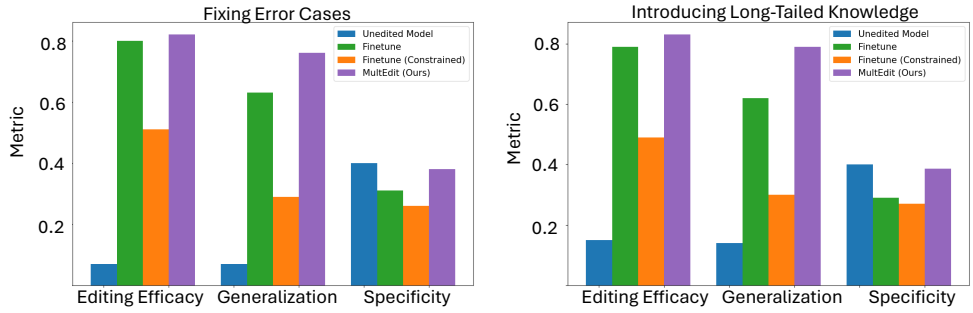

Figure 6.6: **MULTEDIT can correct error cases (left) and insert long-tailed factual knowledge (right) by editing the early causal MLP layers in an MLLM**. We use the average probability of the correct token O^* as the metric for *Editing Efficacy* and *Generalization*, and VQA-Accuracy for *Specificity* (higher the better for all).

answer with an AUROC of 0.63 on the Known dataset in *VQA-Constraints* (see Appendix). We find, however, that the model's confidence – the probability of the generated output token at the final layer – is a slightly stronger predictor, with an AUROC of 0.76 (see Appendix). This parallel's previous work in LLMs [263] which used the attention contributions from *all* the LLM's layers (via a linear model) to predict the generated answer's correctness. In comparison, our results suggest that a subset of MLLM's middle layers alone can be leveraged as a coarse "early" failure mode detector.

Takeaway. MLLMs behave differently in terms of information retrieval from their parametric memory however quite similarly to LLMs in terms of information transfer to the final token.

6.4 Correcting and Inserting Long-Tailed Information in MLLMs

Previous works have shown that counterfactual information can be inserted into LLMs by editing their causal layers [165, 167]. In this section, we verify if a similar approach can be used to edit a MLLM. Specifically, we introduce MULTEDIT, which applies a closed-form update to the early causal MLP layers we identified in Sec. 6.3. We show that our approach can effectively both (i) fix erroneous answers, and (ii) insert new long-tailed information in

LLaVa in a VQA task.

6.4.1 MULTEDIT

Given an image-question $(\mathbf{x}, y)$, we denote a MLLM's generated answer as O. MULTEDIT updates a few parameters in the model such that it generates a new answer O^*. Similar to [121, 165], we update the W^{ℓ}_{proj} matrix at specific causal MLP layers such that the probability $\mathscr{P}(O^*)$ increases. Specifically, we view W^{ℓ}_{proj} as a linear-associated memory (where the matrix's input are treated as keys and its output as values) and uses a closed-form update to map the original keys to new (correct) values. Below, we outline the process for acquiring both the keys and values.

Obtaining keys. Let $\{v_i\}_{i=1}^{N}$ be the visual token embeddings and $\{e_i\}_{i=1}^{M}$ the text token embeddings. Given a causal layer ℓ and the last token of a constraint c, we refer to the input of the layer's W^{ℓ}_{proj} matrix as its keys. Specifically, we define the key $k_{c,\ell}$ to be the input embedding to the W^{ℓ}_{proj} matrix corresponding to the last token of the constraint. This can be obtained with a simple forward pass with the visual and text token embeddings.

Obtaining values. Given a causal layer ℓ, we refer to the output of the layer's W^{ℓ}_{proj} matrix as its values. Specifically, we define $z_{c,\ell}$ as the output embedding corresponding to the last token of the constraint. We optimize $z_{c,\ell}$ such that the probability of the correct answer $\mathscr{P}(O^*)$ increases as:

$$z^*_{c,\ell} = \arg\min_{z_{c,\ell}} \mathscr{L}(z_{c,\ell}) \tag{6.3}$$

where $\mathscr{L}(z_{c,\ell})$ is the standard next-token prediction loss used to train LLMs:

$$\mathscr{L}(z_{c,\ell}) = -\log \mathscr{P}(O^* | v_1 ... v_N e_1 .... e_M) \tag{6.4}$$

MULTEDIT modifies the W^{ℓ}_{proj} matrix such that the old keys $k_{c,\ell}$ are mapped to the new optimized values $z^{*}_{c,\ell}$ which increase $\mathscr{P}(O^{*})$. We define MULTEDIT's editing objective as:

$$W^{\ell *}_{proj} = \arg\min_{W^{\ell}_{proj}} \|W^{\ell}_{proj} k_{c,\ell} - z^{*}_{c,\ell}\|_2^2 + \lambda \|W^{\ell}_{proj} - W^{\ell'}_{proj}\|_2^2 \tag{6.5}$$

The second regularization term ensures that $W^{\ell'}_{proj}$, the weights before the edit, do not deviate too much from W^{ℓ}_{proj}. This helps to preserve performance on unrelated VQA pairs.

6.4.2 Experimental details

We experimentally validate MULTEDIT in two real-world model-editing applications:

Fixing incorrect answers to common questions. We test MULTEDIT on a set of ∼450 visual questions which LLaVa answers incorrectly (detected using incorrect VQA accuracy) from the multi-modal Known dataset in *VQA-Constraints*. These are generally questions about well-known places, persons and brands and companies.

Inserting long-tailed VQA knowledge. We test MULTEDIT on a set of visual questions from the Encyclopedia-VQA dataset [168]. These query fine-grained knowledge about rare landmarks around the world, which MLLMs have been shown to struggle on [168].

For both settings, we compare MULTEDIT to the fine-tuning baselines:(i) fine-tuning from [165] which fine-tunes all the layers using the language modeling objective, and (ii) fine-tuning with constraints from [270] which fine-tunes the layers in a language model with a constraint on the weights to ensure local loss continuity.

We measure the success of each edit operation using the following metrics: (i) **Editing Efficacy**, which uses $\mathscr{P}(O^{*})$ to measure the edited model's ability to generate the correct answer for image-question $(\mathbf{x}, y)$. (ii) **Generalization**, which uses $\mathscr{P}(O^{*})$ to measure

the edited model's ability to generate the correct answer for the question y paraphrased using a language model (see Appendix for details), and (iii) **Specificity**, which uses VQA accuracy [6] to measure the edited model's performance on unrelated VQA questions. We consider unrelated questions to be those from the OK-VQA and Movies datasets in *VQA-Constraints* (see further details in Appendix).

6.4.3 Results

Overall, our results show that updating the projection matrix using MULTEDIT at just a single early (causal) MLP layer can be a very effective approach for both correcting incorrect answers and inserting new knowledge in MLLMs.

Fixing incorrect answers. In Fig. 6.6 (left), we show that MULTEDIT is able to successfully fix the generations for all questions with wrong answers. In editing efficacy, the average probability of the correct answer improves from 0.07 to 0.82 after the edit. We also observe strong generalization, with a probability of 0.76 for the correct answer even when the question is paraphrased. Although we see a small drop of 1.5% accuracy on unrelated image-question pairs, we note that MULTEDIT outperforms fine-tuning with and without constraints on all the metrics.

Inserting long-tailed information. In Fig. 6.6 (right), we show that MULTEDIT is able to reliably insert new long-tailed knowledge in the model with an editing efficacy of 0.83. We see similar strong generalization when paraphrasing questions with an efficacy of 0.79. Similar to above, MULTEDIT incurs a small drop of 1.4% on unrelated questions but is less affected than other methods.

In Appendix, we also provide ablations showing that editing the early causal MLP layers leads to better editing efficacies than editing the middle or the later MLP layers.

6.5 Conclusion

Our paper takes a closer look at how MLLMs process multi-modal information. We contribute a novel multi-modal causing tracing methodology and test-bed, *VQA-Constraints*, as well as a range of novel insights on how MLLMs retrieve and transfer information. We also introduce a novel model-editing algorithm, MULTEDIT, which can effectively fix errors or introduce long-tailed knowledge in MLLMs using a simple closed-form update which targets the early causal MLPs. Overall, this work deepens our scientific understanding of recent MLLM architectures, and enables future work in this direction.

Chapter 7: **Mechanistically Understanding and Unlocking Zero-Shot Capabilities in Vision Transformers**

7.1 Introduction

Vision transformers and their variants [37, 63, 152, 181, 227, 231] have emerged as powerful image encoders, becoming the preferred architecture for modern image foundation models. However, the mechanisms by which these models transform images into representation vectors remain poorly understood. Recently, [77] made significant progress on this question for CLIP-ViT models with two key insights: (i) They demonstrated that the residual connections and attention mechanisms of CLIP-ViT enable the model output to be mathematically represented as a sum of vectors over layers, attention heads, and tokens, along with contributions from MLPs and the CLS token. Each vector corresponds to the contribution of a specific token attended to by a particular attention head in a specific layer. (ii) These contribution vectors exist within the same shared image-text representation space, allowing the CLIP text encoder to interpret each vector individually via text.

Extending this approach to other transformer-based image encoders presents several challenges. Popular models like DeiT [227], DINO-ViT [37, 181], and Swin [152] lack a corresponding text encoder to interpret the component contributions. Additionally, extracting the contribution vectors corresponding to these components is not straightforward,

as they are often not explicitly computed during the forward pass of the model. Other complications include diverse attention mechanisms such as grid attention, block attention (in MaxViT), and windowed/shifted windowed attention (in Swin), as well as various linear transformations like pooling, downsampling, and patch merging applied to the residual streams between attention blocks. These differences necessitate a fresh mathematical analysis for each model architecture, followed by careful application of necessary transformations to the intermediate output of each component to determine its contribution to the final representation. To address these challenges, we propose our framework to identify roles of components in general ViTs.

First, we *automate the decomposition of the representation* by leveraging the computational graph created during the forward pass. This results in a drop-in function, REPDECOMPOSE, that can decompose any representation into contributions vectors from model components simply by calling it on the representation. Since this method operates on the computational graph, it is agnostic to the specifics of the model implementation and thus applicable to a variety of model architectures.

Secondly, we introduce an algorithm, COMPALIGN, to *map each component contribution vector to the image representation space of a CLIP model*. We train these linear maps with regularizations so that these maps preserve the roles of the individual components while also aligning the model's image representation with CLIP's image representation. This allows us to map each contribution vector from any component to CLIP space, where they can be interpreted through text using a CLIP text encoder.

Thirdly, we observe that there is often *no straightforward one-to-one mapping* between model components and common image features such as shape, pattern, color, and texture. Sometimes, a single component may encode multiple features, while multiple components

may be required to fully encode a single feature. To address this, we propose a *scoring function* that assigns an importance score to each component-feature pair. This allows us to rank components based on their importance for a given feature, and rank features based on their importance for a given component.

Using this ranking, we proceed to analyze diverse vision transformers such as DeiT, DINO, Swin, and MaxViT, in addition to CLIP, in terms of their components and the image features that they are responsible for encoding. We consistently find that many components in these models encode the same feature, particularly in ImageNet pre-trained models. Additionally, individual components in larger models MaxVit and Swin do not respond to any image feature strongly, but can encode them effectively in combination with other components. This diffuse and flexible nature of feature representation underscores the need for interpreting them using a continuous scoring and ranking method as opposed to labelling each component with a well-defined role. We are thus able to perform tasks such as image retrieval, visualizing token contributions, and spurious correlation mitigation by carefully selecting or ablating specific components based on their scores for a given property.

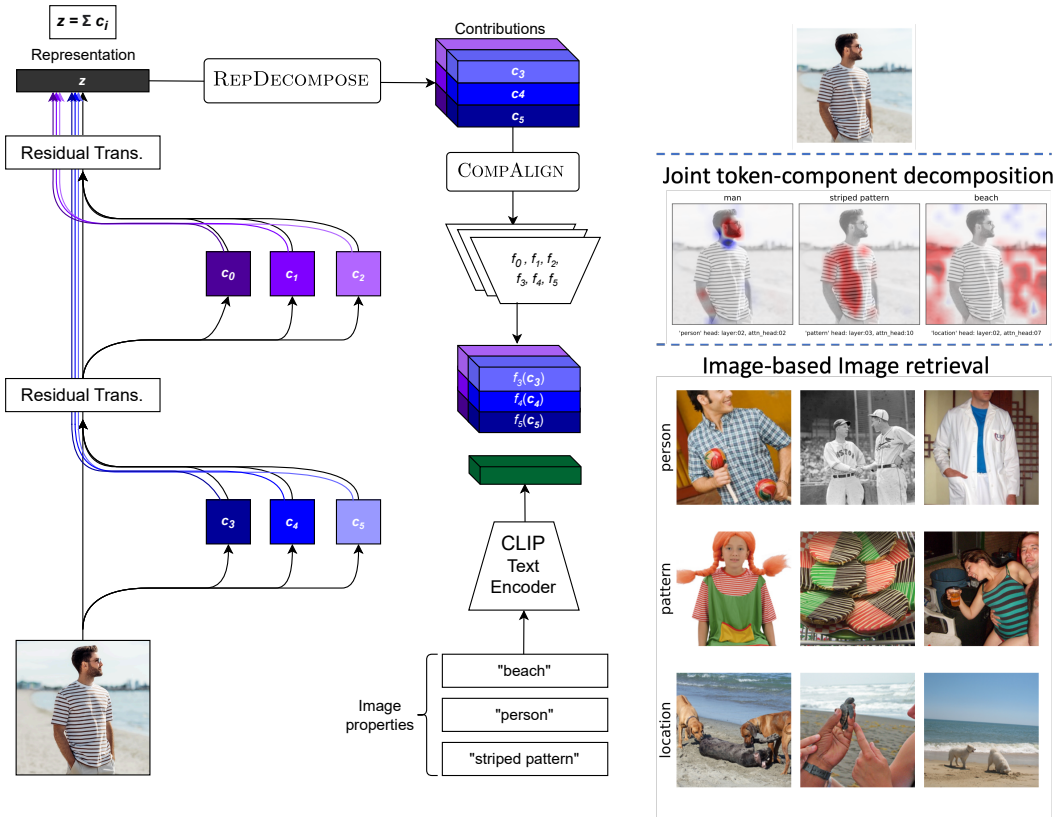

Figure 7.1: **(Left) Workflow:** The first step (REPDECOMPOSE) is to decompose a representation $\mathbf{z}$ into contributions from its model components $\mathbf{c}_i$ after being transformed by residual transformations like LayerNorm, linear projections, resampling, patch merging and so on. The second step (COMPALIGN) aligns each contribution to CLIP space using a set of linear maps $f_0, f_1, \ldots, f_n$ on the corresponding contributions $\mathbf{c}_0, \mathbf{c}_1, \ldots, \mathbf{c}_n$. We can then interpret these aligned contributions using the CLIP text encoder. **(Right) Applications of our method:** (a) Visualizing contributions of each token through a specific component using a joint token-component decomposition (b) Retrieving images that are close matches of the reference image (on top) with respect to a given image feature like pattern, person, or location

Recently, [77] decomposed $\mathbf{z}_{\text{CLS, fin}}$, the final `[CLS]` representation of the CLIP's image encoder as a sum over the contributions from its attention heads, layers and token positions, as well as contributions from the MLPs. In particular, they observe that the last few attention layers have a significant direct impact on the final representation. Thus, this representation

can be decomposed as: $\mathbf{z}_{\text{CLS, fin}} = \mathbf{z}_{\text{CLS, init}} + \sum_{l=1}^{L} c_{l,\text{MLP}} + \sum_{l=1}^{L}\sum_{h=1}^{H}\sum_{t=1}^{N} c_{l,h,t}$, where L, H, N correspond to the number of layers, number of attention heads and number of tokens. Here, $c_{l,h,t}$ denotes the contribution of token t though attention head h in layer l, while $c_{l,\text{MLP}}$ denotes the contribution from the MLP in layer l. Due to this linear decomposition, different dimensions can be reduced by summing over them to identify the contributions of tokens or attention heads to the final representation. While this decomposition is relatively simple for vanilla ViTs, it cannot be directly used for general ViT architectures due to use of self-attention variants such as window attention, grid attention, or block attention, combined with operations such as pooling or patch merging on the residual stream. The final representation may also not just be a single $\mathbf{z}_{\text{CLS, fin}}$ but $\frac{1}{N}\sum_{i=1}^{N} \mathbf{z}_{i,\text{fin}}$ or even $\frac{1}{L}\sum_{i=1}^{L} \mathbf{z}_{\text{CLS},i}$, or some combination of the above.

7.1.1 REPDECOMPOSE: Automated Representation Decomposition for ViTs

We thus seek a general algorithm which can automatically decompose the representation for general ViTs. This can be done via a recursive traversal of the computation graph. Suppose the final representation $\mathbf{z}$ can be decomposed into component contributions $\mathbf{c}_{i,t}$ such that $\mathbf{z} = \sum_{i,t} \mathbf{c}_{i,t}$. Here each $\mathbf{c}_{i,t}$ corresponds to the contribution of a particular token t through some model component i. For convenience, let $\mathbf{c}_i = \sum_t \mathbf{c}_{i,t}$. Then, if given access to the computational graph of $\mathbf{z}$, we can identify potential linear components $\mathbf{c}_{i,t}$ by recursively traversing the graph starting from the node which outputs $\mathbf{z}$ in reverse order till we hit a non-linear node. The key insight here is that the output of any node which performs a linear reduction (defined as a linear operation which results in a reduction in the number of dimensions) is equivalent to a sum of individual tensors of the same dimension as the output. These tensors can be collected and transformed appropriately during the graph traversal to

obtain a list of tensors $\mathbf{c}_{i,t}$, each members of the same vector space as the representation $\mathbf{z}$. This kind of linear decomposition is possible due to the overwhelmingly linear nature of transformers. The high-level logic of REPDECOMPOSE is detailed in Algorithm 3.

In practice, the number of components quickly explodes as there are a very large number of potential component divisions for a given model. To make analysis and computation tractable, we restrict it to only the attention heads and MLPs with no finer divisions. We also constrain REPDECOMPOSE to only return the *direct* contributions of these components to the output. This means that the contribution $\mathbf{c}_i$ is the *direct* contribution of component i to $\mathbf{z}$, and does not include its contribution to $\mathbf{z}$ via a downstream component j. Additionally, the token t in $\mathbf{c}_{i,t}$ is present in the input of the component i, and not the input image. In principle, REPDECOMPOSE could return higher order terms such as $\mathbf{c}_{j,i}$ which is the contribution of model component i via the downstream component j. A full understanding of these higher order terms is essential to get a complete picture of the inner mechanism of a model, however we defer this for future work.

7.2 Aligning the component representations to CLIP space

Having decomposed the representation into contributions from relevant model components, we now aim to interpret these contributions through text using CLIP by mapping them to CLIP space. Formally, given that we have a set of vectors $\{\mathbf{c}_i\}_{i=1}^N$ such that $\sum_i^N \mathbf{c}_i = \mathbf{z}$, the final representation of model, we require a set of linear maps f_i such that the sum of $\sum_i f_i(\mathbf{c}_i) = \mathbf{z}_{\text{CLIP}}$, the final representation of the CLIP model. Once we have these CLIP aligned vectors, we can proceed to interpret them via text using CLIP's text encoders.

However, from an interpretability standpoint, a few additional constraints on the linear maps are desirable. Consider a component contribution $\mathbf{c}_i$ and two directions $\mathbf{u}, \mathbf{v}$ belonging to the

Algorithm 3 REPDECOMPOSE

Input: $\mathbf{z}$, the final representation output by the model and the final node in the computational graph. $\mathbf{z}.f$ is the function that outputs node $\mathbf{z}$

Output: A tree $\mathbf{t}$ consisting of component contributions $\mathbf{c}$, such that components $\sum_{\mathbf{c}\in\mathbf{t}} \mathbf{c} = \mathbf{z}$. The structure of $\mathbf{t}$ is a nested list where each list represents a level in the tree

```
function REPDECOMPOSE(z)
    if is_nonlinear(z.f) then
        return [z]
    else if is_unary(z.f) then                          ▷ Function is unary linear
        z_0 ← z.parents()
        t_0 ← REPDECOMPOSE(z_0)
        if is_reduction(z.f) then
            t_{0,u} ← unbind(t_0)        ▷ Unbinds each c ∈ t along the reduction dimension
            f_d ← decomp(z.f)              ▷ Returns f_d such that Σ_{c∈t_{0,u}} f_d(c) = z.f(z_0)
            return map(f_d, t_{0,u})                          ▷ Maps each c ∈ t_{0,u} to f_d(c)
        else
            return map( z.f, t^0)                        ▷ Maps each element c ∈ t to z.f(c)
        end if
    else                                                                  ▷ z.f is binary
        z_0, z_1 ← z.parents()      ▷ Get the parents of z in the graph (inputs to z.function)
        t_0, t_1 ← REPDECOMPOSE(z_0), REPDECOMPOSE(z_1)
        f_{d,0}, f_{d,1} ← decomp_binary(z.f)                  ▷ Returns f_{d,0}, f_{d,1} such that:
        return [map(f_{d,0}, t_0), map(f_{d,1}, t_1)] ▷ Σ_{c∈t_0} f_{d,0}(c) + Σ_{c∈t_1} f_{d,1}(c) = z.f(z_0, z_1)
    end if
end function
```

same vector space as $\mathbf{c}_i$ which represent two distinct features, say shape and texture. Let us further assume that the component's dominant role is to identify shape, and thus the variance of the projection of $\mathbf{c}_i$ along $\mathbf{u}$ is higher than that of $\mathbf{v}$. We want this relative importance of features to be maintained in $f_i(\mathbf{c}_i)$. Additionally, we also want any two linear maps f_i and f_j to not change the relative norms of features in components $\mathbf{c}_i$ and $\mathbf{c}_j$. We can express these conditions formally as follows:

1. **Intra-component norm rank ordering**: For any two vectors $\mathbf{u}, \mathbf{v}$ and a linear map f_i such that $\|\mathbf{u}\| \leq \|\mathbf{v}\|$, we have $\|f_i(\mathbf{u})\| \leq \|f_i(\mathbf{v})\|$
2. **Inter-component norm rank ordering**: For any two vectors $\mathbf{u}, \mathbf{v}$ and linear maps f_i, f_j

such that $\|\mathbf{u}\| \leq \|\mathbf{v}\|$, we have $\|f_i(\mathbf{u})\| \leq \|f_j(\mathbf{v})\|$

Theorem 2. *Both of the above conditions together imply that all linear maps f_i must be a scalar multiple of an orthogonal transformation, that is for all i, $f_i^T f_i = kI$ for some constant k. Here, I is the identity transformation.*

The proof is deferred to Appendix. We can now formulate a novel alignment method, COMPALIGN, to map contributions of model components to CLIP space. COMPALIGN minimizes a loss function over $\{f_i\}_{i=1}^N$ to obtain a good alignment between model representation space and CLIP space:

$$L(\{f_i\}_{i=1}^N) = \mathbb{E}_{\{\mathbf{c}_i\}_{i=1}^N, \mathbf{z}_{\text{CLIP}}} \left[1 - \cos\left(\sum_i f_i(\mathbf{c}_i), \mathbf{z}_{\text{CLIP}} \right) \right] + \lambda \sum_i \| f_i^T f_i - I \|_F$$

The first term of the objective is the *alignment loss*, which is the average cosine distance between the CLIP representation $\mathbf{z}_{\text{CLIP}}$ and the transformed model representation $\sum_i f_i(\mathbf{c}_i)$. It quantifies the directional discrepancy between the two vectors. The second term is the *orthogonality regularizer* which imposes a penalty if the linear maps f_i are not orthogonal, ensuring that the f_i adhere closely to the specified conditions. We can now train f_i using the above loss function on ImageNet-1k. The training is label-free and can be done even over unlabeled datasets. We obtain $\mathbf{z}_{\text{CLIP}}$ from the CLIP image encoder and $\{\mathbf{c}_i\}_{i=1}^N$ from running REPDECOMPOSE on the final representation of the model.

Ablation study: We now conduct an ablation study on COMPALIGN. The first naive alignment method is the case where all f_i are the same linear map f, without constraints on f, similar to [170]. The second method is a version of COMPALIGN with $\lambda = 0$, where all f_i are different but not trained with the orthogonality regularizer. To compare these methods, we first get a "ground truth" description for each model component by using the TEXTSPAN

Embedding Source	ImageNet pretrained	One map only	CompAlign ($\lambda = 0$)	CompAlign
TextSpan's top 10 descriptions of a random component	wardrobe medicine cabinet window shade desk barbershop refrigerator library shoji screen bathtub dining table	gyromitra home theater drumstick Samoyed muzzle bookstore dining table medicine cabinet park bench tusker	bookcase snorkel red wolf barbershop microwave oven bassinet disc brake dining table sink window screen	filing cabinet snorkel bakery bathtub dining table red wolf gyromitra shoji screen Norwich Terrier bookstore
Match rate	-	0.08	0.155	0.185
Cosine Distance	-	0.23	0.18	0.17

Table 7.1: Comparison of different methods to map the representation space of ImageNet-1k pre-trained DeiT-B/16 to CLIP image representation space. The green colored texts are exact matches with the top-10 descriptions obtained from the imagenet pretrained embeddings, while the orange colored texts are approximate matches. The match rate is the average fraction of exact matches across all components, while cosine distance is the average cosine distance between the CLIP representations and the transformed model representations on ImageNet

[77] algorithm on the class embedding vectors from the ImageNet pre-trained head. This yields descriptions of each component in terms of the top 10 most dominant ImageNet classes. We then use CompAlign and the two baselines to map the representations to CLIP space, and apply TextSpan on CLIP embedded ImageNet class vectors to label each model component. We can then compare the descriptions this yields with the "ground truth" text description for each head. The results, shown in Tab. 7.1, indicate that CompAlign's TextSpan descriptions have the most matches to the ImageNet pre-trained descriptions, followed by CompAlign with $\lambda = 0$ and the naive single map method. This trend is similar in the average cosine distance between the CLIP representations and the transformed model

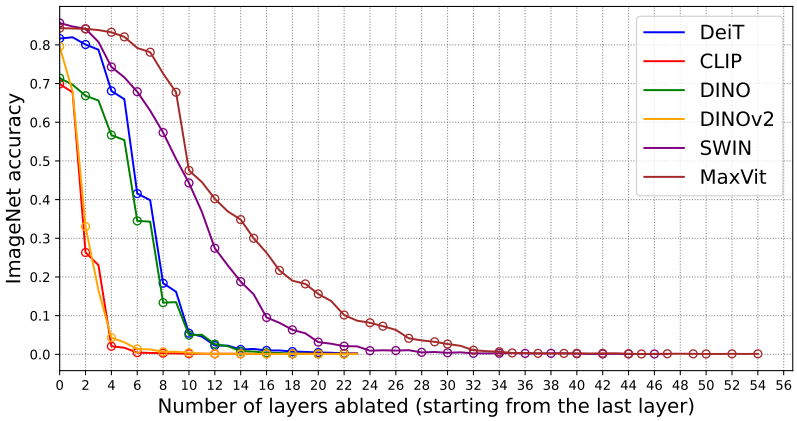

Figure 7.2: Ablation results for various different image encoders. The top-1 ImageNet accuracy is plotted as the layers of the model are increasingly ablated away, starting from the last layer up till the first layer. The circles on the plot represent the endpoints of blocks, the definition of which varies across model architectures. For the vanilla ViT variants, a block is an attention MLP pair, while for SWIN, it is a pair of windowed/shifted windowed attention and an MLP. For MaxVit, this might either be a grid/block attention-MLP pair, or an MBConv block.

representations.

7.3 Component ablation

To identify the most relevant model layers for downstream tasks, we progressively ablate them and measure the drop in ImageNet classification accuracy. Ablation involves setting a layer's contribution to its mean value over the dataset. We use the following models from Huggingface's `timm` [248] repository: (i) DeiT (ViT-B-16) [227], (ii) DINO (ViT-B-16) [37], (iii) DINOv2 (ViT-B-14) [181], (iv) Swin Base (patch size = 4, window size = 7) [152], (v) MaxViT Small [231], along with (vi) CLIP (ViT-B-16) [47] from `open_clip` [105]. DeiT, Swin, and MaxViT are pretrained on ImageNet with a supervised classification loss, DINO on ImageNet with a self-supervised loss, DINOv2 on LVD-142M with a self-supervised loss, while CLIP is pretrained on a LAION-2B subset with contrastive loss.

In Fig. 7.2, we see that for models not trained on ImageNet (CLIP and DINOv2), removing

the last few layers quickly drops the accuracy to zero. In contrast, models trained on ImageNet experience a more gradual decline in accuracy, reaching zero only after more than half the layers are ablated. This trend is consistent across both self-supervised (DINO) and classification-supervised (DeiT, SWIN, MaxViT) models. This suggests that ImageNet-trained models encode useful features redundantly across layers for the classification task. Additionally, larger models with more layers, such as MaxVit, show significantly more redundancy, with minimal accuracy impact from ablating the last four layers. Conversely, the first few layers in all models contribute little to the output. Therefore, our analysis in the subsequent sections is focused on the last few layers of each model.

7.4 Feature-based component analysis

We now analyze the final representation in terms finer components like attention heads and MLPs, focusing on the last few significant layers. We limit decomposition to 10 layers for DeiT, DINO, and DINOv2, but 12 layers for SWIN and 20 layers for MaxVit due to their greater depth and redundancy across components. We accumulate contributions from the remaining components in a single vector $\mathbf{c}_{\text{init}}$, expressing $\mathbf{z}$ as $\mathbf{c}_{\text{init}} + \sum_i^N \mathbf{c}_i$, where $N+1$ is the total number of components including $\mathbf{c}_{\text{init}}$. Here, $N = 65$ for DeiT, DINO, and DINOv2; $N = 134$ for SWIN, and $N = 156$ for MaxVit.

We then ask if it is possible to attribute a feature-specific role to each component using an algorithm such as TEXTSPAN [77]. These image features may be low-level (shape, color, pattern) or high-level (such as location, person, animal). However, such roles are not necessarily localized to a single component, but may be distributed among multiple components. Furthermore, each individual component by itself may not respond significantly to a particular feature, but it may jointly contribute to identifying a feature along with other components. Thus, rather than rigidly matching each component with a role, we aim to

devise a *scoring function* which can assign a score to each component - feature pair, which signifies of how important the component is for identifying a given image feature. A continuous scoring function allows us to select multiple components relevant to the feature by sorting the components according to their score.

We devise this scoring function (described in the Appendix) by looking at the projection of each contribution vector $\mathbf{c}_i$ onto a vector space corresponding to a certain feature. Suppose we have a feature, "pattern", that we want to attribute to the components. We first describe the feature in terms of an example set of feature *instantiations*, such as "spotted", "striped", "checkered", and so on. We then embed each of these texts to CLIP space, obtaining a set of embeddings B. We also calculate the CLIP aligned contributions $f_i(\mathbf{c}_i)$ for each component i over an image dataset (ImageNet-1k validation split). Then, the score is simply the correlation between projection of $f_i(\mathbf{c}_i)$ and the projection of $\sum_i f_i(\mathbf{c}_i)$ onto the vector space spanned by B. Intuitively, this measures how closely the component's contribution correlates with the overall representation. The scores obtained for each component and feature can be used to rank the components according to its importance for a given feature to obtain a *component ordering*, or to rank the features according to its importance for a specific component to get a *feature ordering*.

Model	Feature ordering	Component ordering
DeiT	0.531	0.684
DINO	0.714	0.723
DINOv2	0.716	0.703
SWIN	0.628	0.801
MaxVit	0.681	0.849

Table 7.2: Spearman's rank correlation between the orderings induced by CLIP score and component score averaged over a selection of common features

7.4.1 Text based image retrieval

We can now use our framework to identify components which can retrieve images possessing a certain feature most effectively. Using the scoring function described above, we can identify the top k components $\{\mathbf{c}_i\}_{i=1}^{k}$ which are the most responsive to a given feature p. We can use the cosine similarity of $\sum_{i=1}^{k} f_i(\mathbf{c}_i)$ to the CLIP embedding of an instantiation s_p of the feature p to retrieve the closest matches in ImageNet-1k validation split. In Fig. 7.3, we show the top 3 images retrieved by different components of the DeiT model for the location instantiation "forest" and "beach" when sorted according to the component ordering for the "location" feature. As the component score decreases, the images retrieved by the components grow less relevant. Also note that a significant fraction of components are capable of retrieving relevant images. This further confirms the need for a continuous scoring function which can identify multiple components relevant to a feature.

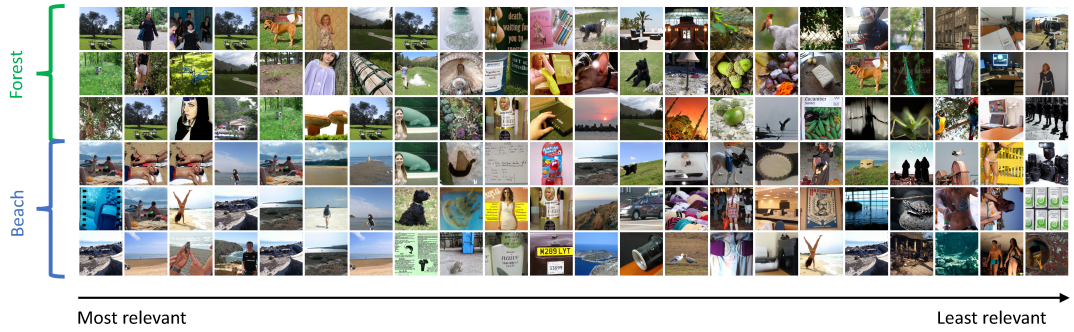

Figure 7.3: Top-3 images retrieved by DeiT components for "forest" and "beach" ordered according to their relevance for the attribute "location". Each column here corresponds to the images returned by the sum of contributions of 3 components, so column i corresponds to components $\mathbf{c}_{3i}, \mathbf{c}_{3i+1}, \mathbf{c}_{3i+2}$. A large fraction of components which can recognize the "location" feature are sorted correctly by the scoring function

To quantitatively verify our scoring function, we devise the following experiment. We first choose a set of common image features such as color, pattern, shape, and location, with a representative set of feature instantiations for each (details in Appendix). The

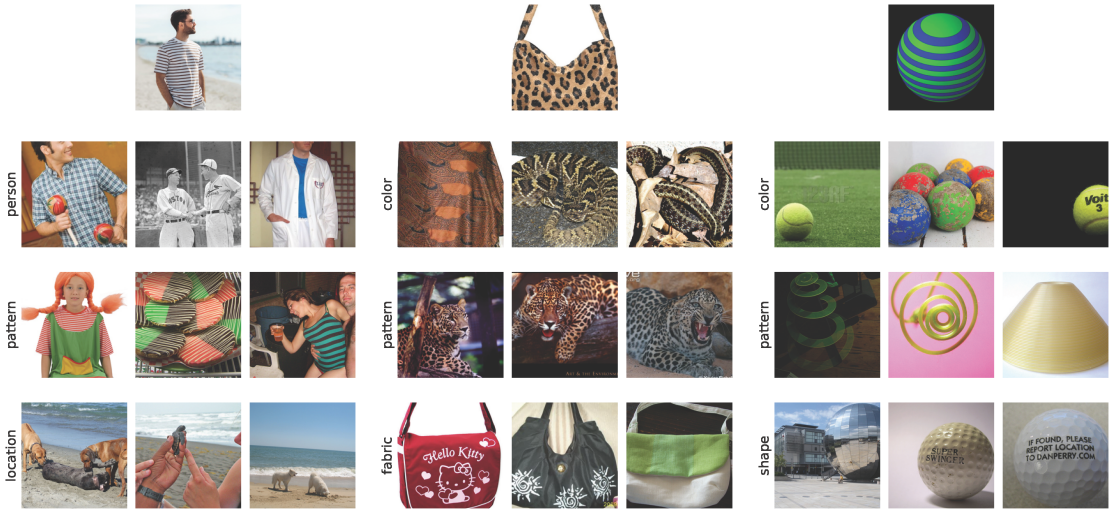

Figure 7.4: Top-3 images retrieved by the most significant components for various features relevant to the reference image (displayed on top). The models used are (from left to right) DINO, DeiT, and SWIN. More exhaustive results can be found in Appendix.

scoring function induces a component ordering for each feature p and feature ordering for each component i. We then compute the cosine similarity $\text{sim}_{i,s_p} = \cos\left(f_i(\mathbf{c}_i), \mathbf{y}_{s_p,\text{CLIP}}\right)$ where $\mathbf{y}_{s_p,\text{CLIP}}$ is the CLIP text embedding of s_p. We can compare this to the cosine similarity $\text{sim}_{\text{CLIP},s_p} = \cos\left(\mathbf{z}_{\text{CLIP}}, \mathbf{y}_{s_p,\text{CLIP}}\right)$ where $\mathbf{z}_{\text{CLIP}}$ is the CLIP image representation. The correlation coefficient between sim_{i,s_p} and $\text{sim}_{\text{CLIP},s_p}$ over an image dataset can be viewed as another score which is purely a function of how well the component i can retrieve images matching s_p as judged by CLIP. Averaging this correlation coefficient over all s_p for a given p yields a "ground truth" proxy for our scoring function. We can measure the Spearman rank correlation (which ranges from -1 to 1) between the component (or feature) ordering induced by our scoring function and the ground truth and average it over features (or components). In Tab. 7.2, we observe that the rank correlation is significantly high for all models for both feature and component ordering. The individual rank correlations for component orderings for common features can be found in Appendix.

7.4.2 Image based image retrieval

We can also retrieve images that are similar to a reference image with respect to a specific feature. To do this, we first choose components which are highly significant for the given feature while being comparatively less relevant for other features. Mathematically, for a feature $p \in P$, the set of all relevant features, we want to choose component i with score $s_{i,p}$ such that the quantity $\min_{p' \in P \setminus p} s_{i,p} - s_{i,p'}$ is maximised. Intuitively, we want components which have the highest gap between $s_{i,p}$ and $s_{i,p'}$ where p' can be any other feature. We can then select a set of k such components C_k by sorting over the score gap, and sum them to obtain a feature-specific image representation $\mathbf{z}_p = \sum_{i \in C_k} \mathbf{c}_i$. Now, we can retrieve any image $\mathbf{x}'$ similar in feature p to a reference image $\mathbf{x}$ by computing the cosine similarity between $\mathbf{z}'_p$ and $\mathbf{z}_p$, which are the feature-specific image representations for $\mathbf{x}'$ and $\mathbf{x}$. We show a few examples for image based image retrieval in Fig. 7.4. Here, we tune k to ensure that it is not so small that the retrieved images do not resemble the reference image at all, and not so large that the retrieved images are overall very similar to the reference image. We can see that the retrieved images are significantly similar to the reference image with respect to the given feature, but not similar overall. For example, when the reference image is a handbag with a leopard print, the "pattern" components retrieve images of leopards which have the same pattern, while the "fabric" components return other bags which are made of similar glossy fabric. Similarly, for the ball with a spiral pattern on it, we retrieve images which resemble the spiral pattern in the second row, while they resemble the shape in the third row.

Note that this experiment only involves the alignment procedure for computing the scores and thereby selecting the component set C_k. The process of retrieving the images is based on $\mathbf{z}_p$ which exists in the model representation space and not CLIP space. This shows that

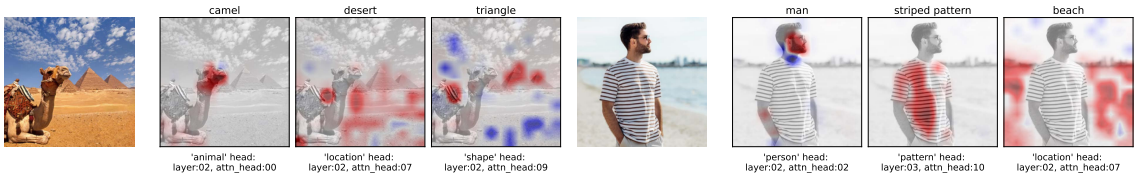

Figure 7.5: Visualization of token contributions as heatmaps for two example images for the DeiT model. The relevant feature and the head most closely associated with the feature is displayed on the bottom of the heatmap, while the feature instantiation is displayed on the top. The layer numbering starts from the last layer (which has index '00'). The regions highlighted in red contribute positively to the prediction, while blue regions contribute negatively. More results in Appendix.

the model inherently has components which (while not constrained to a single role) are specialized for certain properties, and this specialization is not a result of the CLIP alignment procedure.

Model name	Worst group accuracy	Average group accuracy
DeiT	0.733 → **0.815**	0.874 → **0.913**
CLIP	0.507 → **0.744**	0.727 → **0.790**
DINO	0.800 → **0.911**	0.900 → **0.938**
DINOv2	0.967 → **0.978**	0.983 → **0.986**
SWIN	0.834 → **0.871**	0.927 → **0.944**
MaxVit	0.777 → **0.814**	0.875 → **0.887**

Table 7.3: Worst group accuracy and average group accuracy for Waterbirds dataset before and after intervention for various models (format is before → **after**)

7.4.3 Zero-shot spurious correlation mitigation

We can also use the scoring function to mitigate spurious correlations in the Waterbirds dataset [200] in a zero-shot manner. Waterbirds dataset is a synthesized dataset where images of birds commonly found in water ("waterbirds") and land ("landbirds") are cut out and pasted on top of images of land and water background. For this experiment, we regenerate the Waterbirds dataset following [200] but take care to discard background images with

birds and thus eliminate label noise. We select the top 10 components for each model which are associated with the "location" feature but not with the "bird" class following the method we used in Sec. 7.4.2. We then ablate these components by setting their value to their mean over the Waterbirds dataset. In Tab. 7.3, we observe a significant increase in the worst group accuracy for all models, accompanied with an increase in the average group accuracy as well. The changes in all four groups can be found in Appendix.

7.5 Conclusion

In this work, we propose a ViT component interpretation framework consisting of an automatic decomposition algorithm (REPDECOMPOSE) to break down the model's final representation into component contributions and a method (COMPALIGN) to map these contributions to CLIP space for text-based interpretation. We also introduce a continuous scoring function to rank components by their importance in encoding specific features and to rank features within a component. We demonstrate the framework's effectiveness in applications such as text-based and image-based retrieval, visualizing token-wise contribution heatmaps, and mitigating spurious correlations in a zero-shot manner.

Chapter 8: **A Mechanistic Circuit for Extractive Question-Answering**

8.1 Introduction

In recent times, large language models have been used to process documents, webpages and transcripts as context and answer questions about them. We refer to the task of answering a question by directly extracting words from the context/document as extractive Question-Answering (QA), in contrast to "abstractive QA" or "open-ended QA" where the words comprising the answer may not necessarily appear in the context. In the extractive QA case, a language model can either answer from the context, hallucinate entirely from its parametric memory or interpolate between the two. A mechanistic understanding of such a task with a *circuit* (a sub-graph of the language model's computational graph) can not only provide insights on the inner workings of the model for this task, but can also enable downstream applications such as *data-attribution* (i.e., pointing to the source in the context which contributes to the answer) and *model steering* (i.e., enabling the model to answer from the context, rather than hallucinate from its parametric memory). Earlier works on mechanistic circuits [26, 68] for large language models [48, 109, 228] have discovered circuits for language tasks such as entity tracking [188], indirect object identification [244] or simple math operations such as "greater than" [91]. While circuits are a principled way to mechanistically understand language models, we note certain limitations within existing works: (i) Tasks such as *entity tracking* or *indirect object identification* are inherently simple

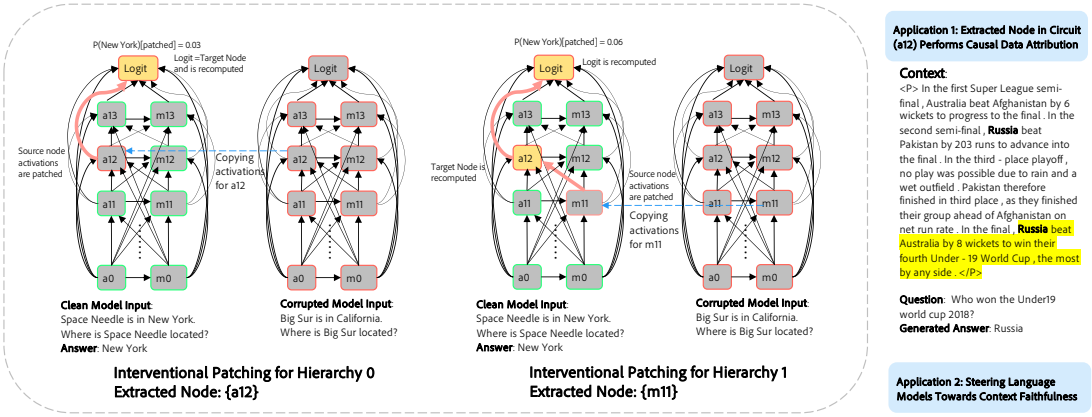

Figure 8.1: **Extracting QA Circuits in Language Models.** We use our probe dataset along with path patching to extract circuits corresponding to (i) *Context* and (ii) *Memory Faithfulness*. We find that a small set of components from the circuit can be used towards performing data-attribution in one forward pass and also steering language models towards context faithfulness.

and may not capture the complexity of real-world applications for language models and (ii) It remains uncertain whether understanding language models through circuits will translate into practical applications.

In our paper, we extract mechanistic circuits for a real-world extractive QA task and use insights from the mechanistic circuit to provide two downstream applications: (i) Data attribution to context and (ii) Steering the language model towards improved context faithfulness. We focus on this task, due to the importance of retrieved-context augmented language models in recent times which unlocks various user-facing downstream applications [14, 81, 135]. We extract two kinds of circuits from language models: (i) *Context-Faithfulness Circuit:* A circuit used by the language model when it solely answers from the context and (ii) *Memory-Faitfulness Circuit:* A circuit used by the language model when it solely answers from its parametric memory. To extract these circuits, we first design a probe dataset (with minimal assumptions about it's inherent structure such as fixed length) and use Causal Mediation Analysis (CMA) [184, 244, 264] to find the subset of nodes and edges in the computational graph of the language model which are causal to the model outputs. In

particular, we observe that the circuits activated during the model's use of context differ significantly from those used for parametric memory. We validate different components of the circuit by various ablations and offer insightful mechanistic understandings.

With the extracted circuit components, we then investigate their roles for the task of extractive QA. We first find that a small set of attention heads in the circuit perform reliable *data attribution by default* (i.e., where the specific input data in the context used to produce an answer is identified), inherently obtaining data attribution in just one forward pass for each token generation. Leveraging this observation, we introduce ATTNATTRIB, which can reliably perform data attribution using **just one attention head** across various real-world QA benchmarks (e.g., HotPotQA, Natural-Questions, NQ-Swap) and white-box language models (Vicuna, Llama-3). In fact, through extensive empirical experiments, we show that ATTNATTRIB can obtain state-of-the-art data-attribution results when compared to other strong baselines for extractive QA tasks without any additional forward pass or auxiliary model, effectively obtaining **attribution for free**. We also find that when the language model answers using the parametric memory circuit, the attribution heads still display a high attention to the answer tokens in the context. With this insight, we design a simple model steering method for improved context-faithfulness, by using the attributions from ATTNATTRIB as an additional source of information. Across various empirical experiments, we find that the addition of attribution during prompting leads to improvements upto 9% on popular extractive QA datasets.

Overall, our paper extracts mechanistic circuits in language models for a real-world task of extractive QA. Beyond mechanistic interpretability of QA tasks, our paper highlights that certain components of the circuit can be useful for downstream applications such as *data-attribution* and also *steering language models* towards being more faithful to the context (thus improving generalization). In summary, our contributions are as follows:

- We extract mechanistic circuits (which provide a causal view) in language models for the real-world task of extractive QA for when the model answers from the context and from the parametric memory.
- We provide salient insights on the underlying mechanics of language models highlighting the interplay between parametric memory and context through the lens of extracted circuits.
- Using the insights from the circuit mechanism, we provide two practical applications: (i) Data-attribution to context with ATTNATTRIB and (ii) Model steering towards context-faithfulness using the attributions from ATTNATTRIB – both reliable enhancements which can ensure that the model does not hallucinate.

8.2 Related Works

Circuit Based Interpretability in Language Models. With the advent of language models, several recent works have focused on a mechanistic understanding of language models [87, 144, 164, 166, 232]. One of the primary benefit of transformer based language models is that the final logit representation can be decomposed as a sum of individual model components [68]. Based on this decomposition, one can extract task-specific causal subgraphs (i.e., circuits) of internal model components in language models. Early works have extracted such circuits for indirect-object identification [244], greater-than operation [91] and more recently for entity-tracking [188]. Circuits can also be constructed as sub-graphs of neurons in the language model, but it often comes with increased complexity of interpretation [67]. Recently, there has been an increasing focus on the practical aspects of mechanistic interpretability such as refusal mediation [11, 267] or safety in general [272]. In our paper, we focus on extracting circuits for a real-world task such as extractive QA with a particular emphasis on practical applications such as *attribution* and *steering*.

Applications in Context-Augmented QA. With the advent of retrieval-augmented generation [81, 135] language models have been increasingly used for real-world Question-Answering (QA) tasks. One of the primary enhancement of context-augmented QA lies in the ability to provide reliable grounding (i.e., attribution) in the context for the generated answer [102, 113, 137, 257]. In recent times, there have been a large set of works which improve LLM responses by reducing hallucinations and improving grounding in the input context [14, 252, 257, 265]. Beyond grounding, [160, 241, 249, 251] investigate the interplay between model's use of parametric vs. context knowledge.

8.3 Deciphering a Circuit for Extractive QA

Nodes and Edges in a Language Model Circuit. Recent decoder-only large language models, denoted by g_ϕ, such as Llama variants [70, 228], are built on the seminal transformer architecture [235]. A notable characteristic of these architectures is that the token representation at any layer can be expressed as a function of internal model components, such as multi-layer perceptrons (MLPs) and attention heads, from earlier layers [68]. As a result, the computational graph underlying a language transformer is a directed acyclic graph, with nodes representing components like MLPs and attention heads (or layers), and edges representing connections formed by the residual stream.

We are particularly interested in obtaining a sub-graph of the transformer's computational graph which is responsible towards context-augmented language modeling. In particular, we extract two circuits: (i) *Context-Faithfulness Circuit*, which is used when the underlying language model answers from the context, and (ii) *Memory-Faithfulness Circuit*, which is used when the language model solely answers from the parametric memory, ignoring the context. To extract the respective circuits, we first design a probe dataset mimicking both these conditions which we use with causal mediation analysis [244].

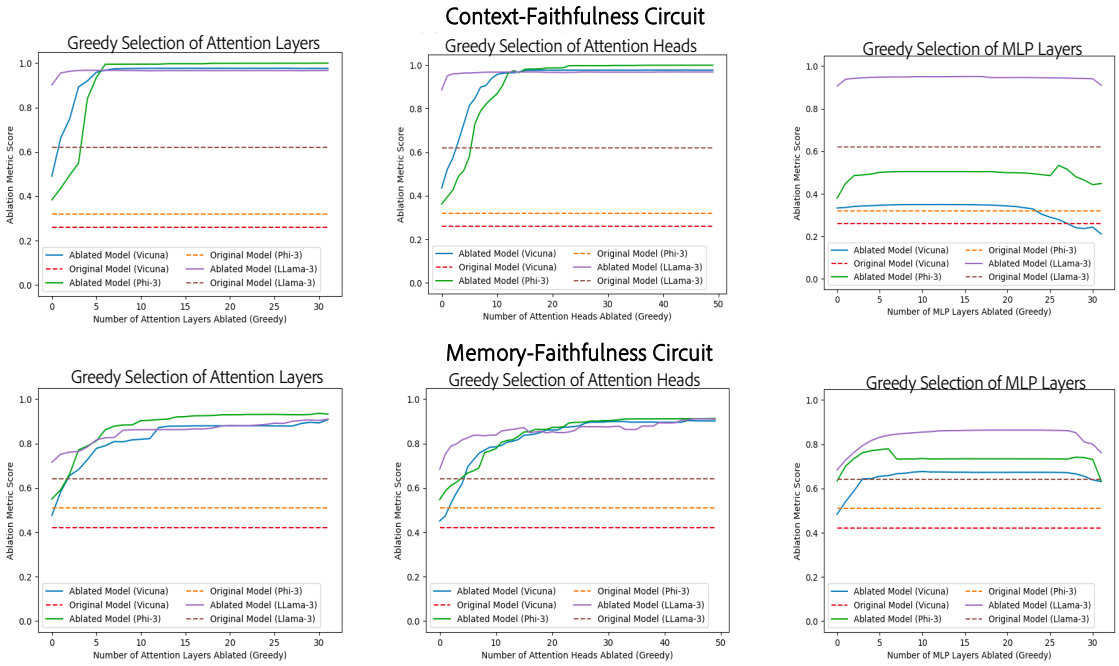

Figure 8.2: (i) **Top Row (Context Circuit Components).** We find that a small set of attention layers and attention heads are sufficient towards a high **average metric score** across all the models. However we find that for Vicuna and Phi-3, patching MLPs do not lead to a high metric score. For Llama-3-8B, we find MLP-31 to have a high direct effect, which when greedily combined with other MLP layers obtain higher scores; (ii) **Bottom Row (Memory Circuit Components).** We find that a large number of attention heads and layers are required to obtain a high metric score. Unlike the context circuit, we find MLPs to be important for the memory circuit.

8.3.1 Designing the Probe Dataset

The design of a probe dataset is extremely crucial in extracting circuits for a language model task as shown in earlier works [91, 244]. We are interested in obtaining a circuit for context-faithfulness as well as one when the model answers from the parametric memory while ignoring the context. To this end, we design two probe datasets $\mathscr{D}_{copy}$ and $\mathscr{D}_{memory}$ respectively for them. Each example in $\mathscr{D}_{copy}$ and $\mathscr{D}_{memory}$ consists of factual questions sourced from the Known dataset [166]. For each question q_i in both datasets, we use Llama-3-70B-Instruct to generate a context c_i related to the subject and answer for q_i. To guarantee that for each question in $\mathscr{D}_{copy}$, the language model **only** answers from the context (and not the memory), we replace the answer tokens in the context c_i with a set of tokens which are semantically similar to the original answer(e.g., in Fig.(8.1), we replace *Seattle* with *New York* in the original context *Space Needle is located in Seattle*, where the original answer was *Seattle*). In $\mathscr{D}_{memory}$, to force the model to answer from the parametric memory while ignoring the context, we replace the answer token with a token which is far away in semantic meaning from the original answer (e.g., replace *Seattle* with a punctuation of "–"). In total, we curate 1000 questions (with their corresponding modified contexts) in $\mathscr{D}_{\text{copy}}$ and $\mathscr{D}_{\text{memory}}$. We note that each entry $x_i \in \mathscr{D}_{\text{copy/memory}}$, contains a question q_i, a subject of the question s_i, ground-truth answer denoted by a_i, the modified context c_i' and the original context c_i. Along with c_i and c_i', we add a corrupted context $c_{i,\text{corrupted}}$, where the subject and the answer token in the context is replaced by unrelated tokens and $q_{i,\text{corrupted}}$ where the subject in the question is replaced by a randomly sampled token. For e.g., as seen in Fig.(8.1) the corrupted context (*Big Sur is in California*) is formed by replacing the subject and the answer tokens in the modified context.

Distinctions from Other Circuit Datasets. Previous work on circuit extraction for entity

tracking and indirect object identification relies on fixed templates for generating examples. However, for real-world tasks like extractive QA, probe datasets cannot use templates due to varying context lengths and unique information across examples.

8.3.2 Interventional Steps for Extracting Circuits

Our interventional method is developed on the foundational technique of causal mediation analysis [184]. The primary idea of causal mediation analysis is to find important paths in a causal graph, by performing an interventional operation on a small set of nodes and measuring the change in the final output. In our use-case, we adapt this method to find a sub-graph of internal model components such that ablating them leads to a decrease in ability of the model to perform QA (either through extraction from the context or using the parametric memory while ignoring the context). Below we provide the algorithmic description:

Algorithmic Description. Given the language model g_ϕ and its associated computational graph $\mathscr{G}$, our objective is to extract a sub-graph (i.e., a circuit) $\mathscr{C} \in \mathscr{G}$ which is responsible towards the QA task. We obtain the nodes and edges of the circuit $\mathscr{C}$ in a hierarchical manner. First, we obtain a set of nodes and edges in hierarchy 0 denoted as $(\mathscr{N}_0, \mathscr{E}_0)$ which have the highest direct effect to the final logit. In the next step for hierarchy 1, we obtain a set of nodes and edges $(\mathscr{N}_1, \mathscr{E}_1)$, which have the highest direct effect on the nodes from hierarchy 0. For any hierarchy k, we obtain a set of nodes and edges $(\mathscr{N}_k, \mathscr{E}_k)$ which have a high direct effect on the nodes $(\mathscr{N}_{k-1}, \mathscr{E}_{k-1})$ from the previous hierarchy. For obtaining the nodes at the k^{th} hierarchy, we create two instantiations of the underlying language model g_ϕ. The first instantiation is denoted as $g_{\phi,\text{clean}}$, with the original question q_i and modified context c_i' as the input. The second instantiation of the language model is $g_{\phi,\text{corrupted}}$, where the input context as well as the question is corrupted as $c_{i,\text{corrupted}}$ and $q_{i,\text{corrupted}}$ respectively. With this

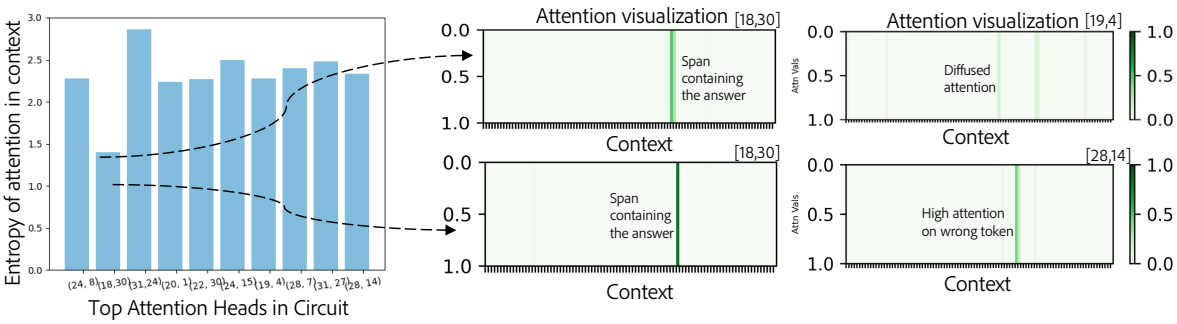

Figure 8.3: **We find that one attention head in the context faithfulness circuit obtains a low entropy value in the context window**. Qualitative results shows that this attention head for Vicuna leads to peaky attention values in the context span containing the answer, whereas other attention heads produce either diffused attentions or erroneous attentions. Further results on Llama-3 and Phi-3 in Appendix.

corrupted input, model $g_{\phi,\text{corrupted}}$ assigns a low probability to the generated answer tokens a_i from $g_{\phi,\text{clean}}$. Using these two model instantiations, the goal of the patching operation is to copy the activations of a node $g_j \in \mathscr{G}$ from $g_{\phi,\text{corrupted}}$ to $g_{\phi,\text{clean}}$, while restoring the activations of all the other nodes in $g_{\phi,\text{clean}}$ to its original state. We denote the patched model as $g_{\phi,\text{patch}}$ and use $\text{score}(i,g_j) = 1 - \mathscr{P}_{g_j,\text{patch}}(a_i)$ to measure the importance of the component g_j for the i^{th} example. For the component g_j, we then compute the **average metric score** as $\text{score}(g_j) = \sum_{i=1}^{|\mathscr{D}|} \text{score}(i,g_j)/|\mathscr{D}|$. We then sort the scores of the various components in the computational graph as $\text{score}(g_j)\ \forall j \in N$ in decreasing order as $\{g_j\}_{j=1}^{N}$ and greedily select the minimum value of k, such that the **average metric score** of patching multiple components together: $\text{score}(\{g_j\}_{j=1}^{k}) \geq \delta$. These selected components $\{g_j\}_{j=1}^{k}$ form the nodes in $\mathscr{N}_k$. In our experiments, we only use the MLPs, the attention heads and layers as the different model components which are patched. The final circuit $\mathscr{C}$ consists of the nodes $\{\mathscr{N}_k\}_{k=1}^{K}$ and their associated edges, where K denotes the maximum hierarchy of the circuit.

Circuit for Context Faithfulness. We extract the circuits using $\mathscr{D}_{copy}$ as the probe dataset for the patching operations. We perform the patching operation at the last residual stream

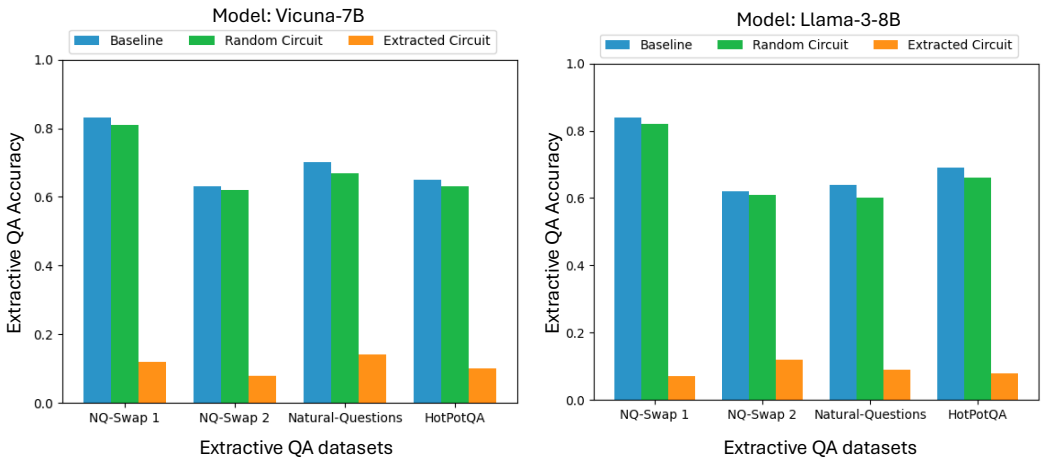

Figure 8.4: **Ablating the extracted context-faithfulness circuit leads to a large drop in extractive QA accuracy for various datasets.** We ablate the edges from the extracted circuit and a random circuit in the language model and measure the extractive QA accuracy.

position. We selected this position because the information in the last residual stream plays a crucial role in determining the probability distribution of the next generated token, which is also used in recent mechanistic interpretability works [11, 232].

Circuit for Parametric-Memory Faithfulness. In this case, we use $\mathscr{D}_{memory}$ as the probe dataset for the patching operation. We extract the circuit at the same token positions as the ones for context faithfulness.

Empirically, we primarily extract our circuits for both context faithfulness and memory faithfulness corresponding to hierarchy-0 (which constitutes the first-order effects)

Circuit Validation. We validate the extracted circuit $\mathscr{C}$ by comparing to (i) Using a randomly extracted circuit $\mathscr{C}_{random}$ to measure the probability of the answer tokens; In this case the probability of the answer tokens will be low. (ii) We also ablate the context-faithfulness circuit (obtained using our probe dataset) across various extractive QA datasets commonly used in the community and measure the drop in the extractive QA accuracy. A large drop in accuracy signifies the validity of the extracted circuit.

In the next sections, we discuss the results corresponding to the mechanics of context-augmented language generation.

8.3.3 Insights For Extractive QA through Circuits

In this section, we discuss the extracted circuit for both *context faithfulness* and *parametric memory faithfulness*. We first draw out their distinctions and validate the correctness of the circuit components. We then discuss the interpretable nature of a small set of attention heads in the circuit.

Context Faithfulness Circuit Differs from Parametric Memory Circuit

Results for attention components. We find the circuit components for *context faithfulness* and *memory faithfulness* to differ significantly. For context faithfulness, we find that patching a small group of 4-5 attention layers (or 10 attention heads) is sufficient to obtain a high average metric score of more than 0.95. However, for the memory faithfulness, we find that a significantly higher number of attention layers (e.g., $>$15) and attention heads (e.g., $>$30) are required to obtain a relatively high metric score. *This result shows that information from a small set of attention heads (or layers) primarily drive the circuit corresponding to context faithfulness than memory faithfulness.* We also find that the top circuit components of attention layers (or heads) have a low overlap between the two circuits – highlighting that the underlying language model elicits different circuits when answering from the context vs. parametric memory.

Results for MLP components. We observe an intriguing pattern with MLPs in the extracted circuit. For context faithfulness, in Vicuna and Phi-3, MLPs appear to be less significant, as patching them results in a very low metric score. However, in Llama-3-8B, we identify one specific MLP (MLP-31) that individually achieves a high metric score of 0.9. This suggests

that the type of pre-training might play a role in determining the relevant circuit components (with respect to MLPs) for context faithfulness. For memory faithfulness, MLPs consistently obtain higher average metric scores across all three language models compared to the top MLPs in the context faithfulness circuit. This underscores the importance of MLPs when the language model retrieves information from parametric memory. Interestingly, we also find minimal overlap between the circuit components responsible for context faithfulness and those for memory faithfulness, even among MLPs.

Validation of the Extracted Circuit

Comparison with Random Circuit. For all the language models, when using a randomly extracted circuit (for *context faithfulness*), the probability of the answers from the probe dataset $\mathscr{D}$ drops to 0.045 for Vicuna, 0.081 for Llama-3-8B and 0.07 for Phi-3, which shows the relevance of our extracted circuit.

Generalizability of the Circuit to Downstream Extractive QA Datasets. To validate the circuits, in Fig.(8.4), we ablate the context-faithfulness circuit components when answering questions from downstream datasets such as NQ-Swap, Natural-Questions and HotPotQA and measure the extractive QA accuracy. We compare with the extractive QA accuracy when a random circuit is ablated from the language model. Overall, we find that ablating the direct connections from the identified context-faithfulness circuit components, lead to the maximal drop in extractive QA accuracy. This result validates that the extracted context-faithfulness circuit generalizes to other commonly used extractive QA datasets. We also run additional experiments on circuits across different knowledge partitions (e.g., country, language). In particular, we find that the context circuits are similar (with a high overlap) amongst different knowledge partitions.

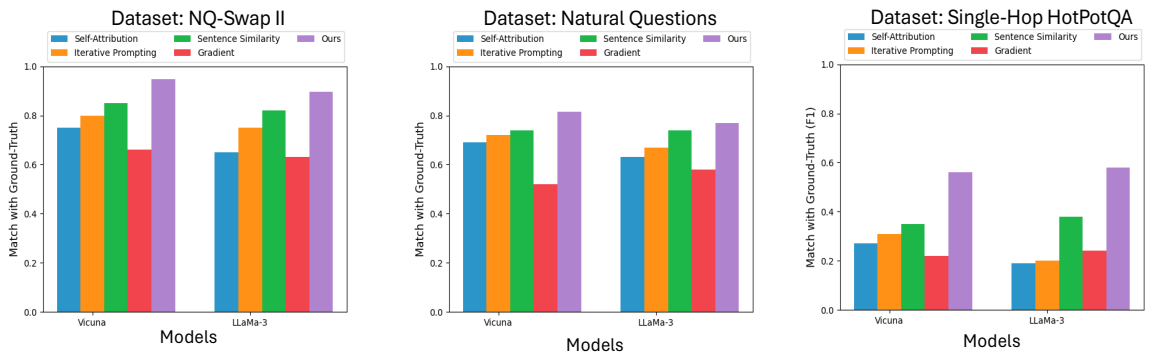

Figure 8.5: **Attribution through** *one attention head* **in our circuit via ATTNATTRIB obtains strong attribution results.** Across various extractive QA benchmarks, we obtain improved performances over different attribution baselines. For HotPotQA, we measure the F1-score due to it being single-hop, whereas for other datasets, we measure the attribution accuracy.

A Small Set of Attention Heads in the context circuit are interpretable

In Fig.(8.3), we observe that a small subset of attention heads in the extracted circuit for *context faithfulness* achieves a low entropy score with respect to the normalized attention values over the context. Upon further inspection, we find that these low-entropy attention heads predominantly focus on the answer token spans in the context. Conversely, some other attention heads in the circuit, while also highly attentive to the answer token spans, display more diffused attention patterns across other tokens. These findings are consistent across all three language models studied: Vicuna, Llama-3-8B, Phi-3 and Llama-3-70B. These results highlight the potential of a small set of attention heads from the circuit to be used for data attribution in language models (see more details in Sec.(8.4)) for extractive QA datasets.

One Can Switch Between Memory and Copy Faithfulness Circuits

To further validate the distinction between circuit components for *Context faithfulness* and *Memory faithfulness*, we conduct two ablation studies. Specifically, we use $\mathscr{D}_{memory}$, but force the language model to answer from the context, even when it originally retrieves

answers from the parametric memory. We achieve this model forcing by: (i) upweighting the attention values at the answer token span in the context by a scaling factor β in the top attention layers of the context faithfulness circuit, and (ii) mean-ablating the top MLPs from the memory faithfulness circuit.

Algorithm 4 ATTNATTRIB: Data Attribution via *One Attention Head*

Input: g_ϕ(Language model), q(Question), C(Context),
k(Number of Spans), L(Answer Length), l(Attn Layer),
h(Attn Head), $slength$(span-length)

Output: Candidate attribution spans

$S \leftarrow \{\}$
$A_{\text{total}} \leftarrow \{\}$
for $j \leftarrow 1, \ldots, L$ **do**
$a_j, A_j = g_\phi(C, q)$ ▷ A_j: Attention map over context, a_j: answer token
$A_{\text{total}}.append(a_j)$ ▷ Add the answer token
$A_{j,\text{relevant}} \leftarrow A_j[l, h]$ ▷ Extract the attention pattern for the given layer and head
$s_j, v_j = GetMaxSpan(A_{j,\text{relevant}}, C, slength)$ ▷ Extract maximal attention span and value
$S.append((s_j, v_j))$ ▷ Add the extracted span s_j to the list along with its value v_j
end for
return Sort$(S)[:k]$ ▷ Sort extracted spans wrt attention value v and use the top-k as attributions

Our findings show that with attention upweighting, 92% of the questions from $\mathscr{D}_{memory}$ are correctly answered using the answer tokens from the context instead of the parametric memory. Meanwhile, mean-ablating the MLPs results in 68% of the questions being answered with relevant answer tokens from the context. These results further validate the

distinction in the circuit components for memory and context faithfulness and also shows that one can switch between the circuits by modifying a small set of components.

8.4 Application 1: Attribution for Free Via One Attention Head

Data attribution for extractive QA is crucial for language models processing external contexts, such as documents or personal files, not included in the pre-training corpora. For example, in a question like "*What did Sarah Miller say during the all-hands meeting?*", the correct answer comes from a specific section of the context (e.g., meeting transcript). Pointing to the source of the answer improves model reliability and helps users verify its correctness, especially since LLMs are prone to hallucinations [176]. In this section, we introduce ATTNATTRIB, an efficient data attribution algorithm, leveraging insights from our mechanistic interpretations which outperforms existing QA baselines.

8.4.1 ATTNATTRIB: A Simple and Strong Data Attribution Method for Extractive QA

In Sec.(8.3.3), we observe that a small set of attention heads from hierarchy 0 of the circuit attend to the answer token in the context. Thus, these attention heads from the extracted circuit for context faithfulness implicitly perform data attribution by default. However, real-world contexts can be noisy and contain multiple answer tokens, raising questions about the behavior of these attributable attention heads in practical settings. In this section, we introduce ATTNATTRIB, which automatically generates attributions from the context during the forward pass by leveraging only one attention head from the context faithfulness circuit. Specifically, ATTNATTRIB uses the attention patterns from the relevant attention head to generate a span from the context for each generated answer token. These spans are ranked based on the maximum attention value within the span (a sentence from the context), and the

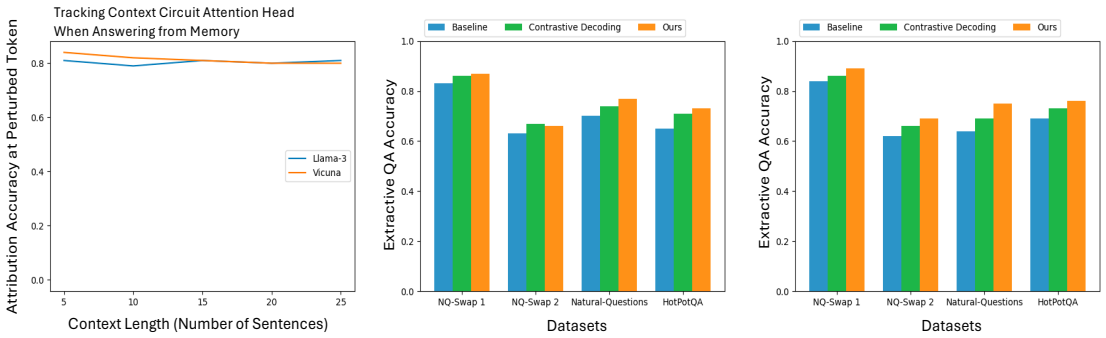

Figure 8.6: **Augmenting the prompt with the attribution from ATTNATTRIB improves extractive QA accuracy.** (a) The attribution at the perturbed token in context through our extracted attention head, when the language model answers from the parametric memory ($\mathscr{D}_{\text{memory}}$) is high. (b) Vicuna-7B and (c) Llama-3-8B: Improvement in extractive QA accuracies for both Vicuna and Llama-3-8B when compared to baseline prompting and Context-aware Contrastive Decoding.

top-k spans are selected for attribution. A detailed description of ATTNATTRIB is provided in Algo. (6). Using ATTNATTRIB, we explore the potential applications of mechanistic circuits for attribution in extractive QA. We note that we use **only one attention head** identified using our probe dataset, $\mathscr{D}_{copy}$, and test its effectiveness on different extractive QA benchmarks.

8.4.2 Evaluation on Extractive QA Benchmarks

Baselines. We use the following baselines: (i) *Self-Attribution*: In this, we prompt the language model to generate an attribution from the context which is required to answer the question. This prompting technique is similar in principle to [80] and [31]; (ii) *Iterative Prompting*: We first generate the answer from the language model, then perform another forward pass and prompt the language model to generate the attribution from the context for the generated answer. (iii) *Sentence Similarity.* We retrieve the most similar sentence from the context to the generated answer using an auxiliary language encoder (all-mpnet-base-v2). This choice is motivated by findings from [31], which identified this embedding model as

one of the best-performing retrievers. (iv) *Gradient*: We find the gradient of the loss for a generated token with respect to the input context token embeddings [259]. We then use this to select the span containing the token with the highest gradient value.

General Empirical Results. We compute the exact match score with the ground-truth attributions across the synthetic dataset (used in our probing step), NQ-Swap [153], Natural-Questions [127] and Single-Hop HotPotQA [255]. Across all the datasets, we find that ATTNATTRIB leads to improved results over strong baselines. We note that the components (i.e., relevant attribution head) of our circuit are primarily extracted for zero-hop extractive QA. Inspite of this, we find that our method obtains better F1 scores ($\approx 20\%$ improvement) than the baselines for single-hop extractive QA. The simplicity of our approach enables attribution computation in just one forward pass (during the answer generation step) therefore positioning itself as a tool for real-world use-case in the domain of extractive QA. We also find ATTNATTRIB to be robust towards larger context lengths for language models supporting long contexts (e.g., Llama-3-8B, Phi-3). For Vicuna, we observe degradation for longer contexts as it only support 2048 tokens as the context length.

Extending to Long Extractive Answer Generations. We apply ATTNATTRIB to attribute long extractive answer generations to specific parts of the input context. For this purpose, we use 1000 examples each from the CNN-Dailymail [95] and NQ-Long [127]. For evaluating the quality of attributions, we measure the change in the log probability of the responses when the top attributed sentences in the context are ablated. A higher change in the log probability indicates the effectiveness of the method. We also show that ATTNATTRIB consistently obtains a high change in log probability score (when compared to other baselines) for both the datasets, indicating that our method is scalable to long answer generations.

Scaling to Llama-3-70B. We apply the circuit extraction steps from Sec.(8.3.2) to identify

the causal components that ensure context faithfulness in Llama-3-70B. Using the attention head with the lowest entropy in the context, combined with ATTNATTRIB, we extract the attributions. Overall, our method yields reliable and robust attributions for larger language models such as Llama-3-70B which highlights the generalizability of our approach.

8.5 Application 2: Towards Improved Context Faithfulness

In the experimental setup in Section 8.3.3, we observe that when the model answers from parametric memory, upweighting the attention at the answer tokens in the context can prompt the model to answer from the context instead. Further investigation reveals that even when the model retrieves answers from parametric memory, the attention maps from the attribution head used in Section 8.4.1 still show a high focus on the perturbed answer tokens in the context. Fig.(8.6)-(a) illustrates the attribution accuracy concerning the perturbed context answer tokens when the language model answers from parametric memory. Based on this insight, we employ ATTNATTRIB to obtain attributions for language model generations using a single forward pass. We then use these attributions in the prompt as an additional signal to guide the language model towards greater faithfulness to the context. Below we provide the empirical results:

Empirical Results. Across various extractive QA benchmarks including NQ-Swap, Natural-Questions and HotPotQA, we find that using the attributions extracted with ATTNATTRIB as an additional signal in the prompt improves the extractive QA performance by upto 9% (see Fig.(8.6)-(b, c)). We observe consistent improvements across both the Vicuna and Llama-3-8B family of models when compared to baseline prompting and Context-aware decoding [210]. This highlights the benefits of incorporating attributions from ATTNATTRIB in the prompt, for improved faithfulness to the context on real-world benchmarks.

8.6 Conclusion

In this paper, we obtain mechanistic circuits for extractive QA, a popular real-world task. We identify key mechanistic differences when the model uses the *parametric memory* (ignoring the context) vs. when it uses the *context*. We then find that a small set of attention heads in the context circuit performs *data attribution by default*. Using this insight, we introduce ATTNATTRIB, an efficient data attribution algorithm which obtains strong results on extractive QA benchmarks. We further show that the attributions from ATTNATTRIB can be used towards improving generalization in extractive QA tasks by steering the model towards context faithfulness. Our paper shows that mechanistic insights can be strategically used for enhancing language models.

Chapter 9: **Improving Compositionality in Multimodal Models**

9.1 Compositionality in CLIP

9.1.1 Introduction

In recent years, multimodal models like CLIP [191] have excelled in tasks such as zero-shot classification, image-text retrieval, and image-captioning [140, 171, 173, 261]. These models are also crucial components in various state-of-the-art pipelines for tasks like segmentation and object detection [156, 169, 246, 268]. However, they struggle with visio-linguistic reasoning tasks, such as determining the spatial relationships between objects in an image [104, 262]. Notably, CLIP's performance on the challenging Winoground [60, 223], a benchmark designed to assess visio-linguistic reasoning, is close to random chance. This shortcoming is attributed to CLIP's contrastive objective which prioritizes shortcuts for retrieval, and thus impacts its ability to understand fine-grained object details and their positions [60, 223].

In contrast, text-to-image models like Stable Diffusion [198] excel in visio-linguistic tasks, likely due to their text conditioning enhanceing semantic consistency in its cross-attention maps [49, 136]. [136] recently demonstrated this on the Winoground benchmark, reliably matching captions to images with fine-grained spatial differences using denoising diffusion scores (see Fig. 9.1). Similar results have been shown for other text-to-image models,

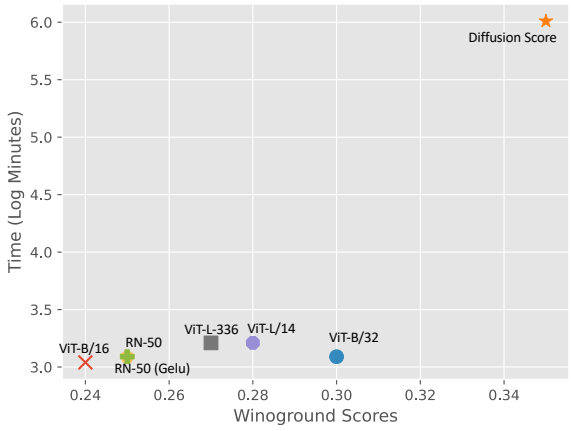

Figure 9.1: **CLIP variants underperform on Winoground, a visio-linguistic reasoning benchmark, compared to Diffusion Score from Stable Diffusion.** The diffusion score is computed from Stable Diffusion's loss function. Note that Diffusion Score takes 18× more time than CLIP variants for inference (using 50 samplings during diffusion score computation).

including Imagen [49], with almost all of these methods outperforming CLIP variants on the same tasks.

While these works have shown the potential of using generative text-to-image models for visio-linguistic tasks, it remains computationally intensive. For instance, computing the denoising diffusion score for image-text matching involves multiple passes through a UNet model (approximately 892M parameters) with varying noise levels and time-steps. On an entry-level GPU, this can take up to a minute for a single image-text matching task, making it impractical for real-world and real-time applications. In contrast, CLIP models can classify images up to 18 times faster (see Fig. 9.1), requiring only one pass through both image and text encoders. A promising research direction, therefore, lies in finding methods that combine the strong visio-linguistic capabilities of text-to-image models with the rapid inference of CLIP.

To this end, we introduce SDS-CLIP, a lightweight and sample-efficient fine-tuning approach for CLIP which distills knowledge from Stable Diffusion, and enhances CLIP's

visio-reasoning capabilities. Specifically, we add a regularization term to CLIP's standard contrastive loss based on score-distillation sampling (SDS) [187]. This regularization encourages CLIP's embeddings to be aligned with the denoising diffusion loss from a text-to-image model. By fine-tuning CLIP with this regularized objective on a small paired image-text dataset, specifically 118k image-text pairs from MS-COCO, we demonstrate an 1.5-7% performance gain compared to vanilla CLIP on Winoground and ARO, two highly challenging visio-linguistic reasoning benchmarks. Notably, this is achieved by only updating CLIP's LayerNorm parameters. Furthermore, we show that SDS-CLIP's zero-shot performance is not impacted on a wide range of downstream datasets.

In summary, our contributions are as follows:

- We introduce SDS-CLIP, a novel sample-efficient and parameter-efficient fine-tuning method that integrates a distillation-based regularization term from text-to-image models.
- We empirically validate our approach on challenging benchmarks and demonstrate an improvement in CLIP's visio-linguistic reasoning, without harming its zero-shot capabilities.

9.1.2 Denoising Diffusion Score for Visio-Linguistic Reasoning

The Winoground benchmark establishes a challenging image-text matching task to measure a model's visio-linugistic reasoning abilities: given an image x, the model must match it with the correct caption c^* from a set of captions $C = \{c_i\}_{i=1}^{n}$, where all caption contains the same words but each describes a different spatial arrangement of the objects, with only one being correct. Concurrent works [49, 125, 136] to this paper have showed that it is possible to use the denoising diffusion score from text-to-image generative models to perform such

Model	**Wino-Overall**	Object	Relation	Both	1 Main Pred	2 Main Preds	**ARO-Overall**	ARO-Relation	ARO-Attribution
ViT-B/16(CLIP)	0.24	0.28	0.18	0.57	0.29	0.11	0.57	0.52	0.62
FT with L_{CLIP}	0.23	0.27	0.19	0.56	0.30	0.11	0.56	0.51	0.62
FT with $L_{CLIP}+L_{SDS}$	**0.31**	**0.35**	**0.25**	**0.69**	**0.36**	**0.16**	**0.58**	**0.535**	**0.63**
ViT-B/32(CLIP)	0.30	0.35	0.22	0.80	0.34	0.18	0.55	0.50	0.61
FT with L_{CLIP}	0.28	0.31	0.20	0.76	0.31	0.16	0.55	0.50	0.60
FT with $L_{CLIP}+L_{SDS}$	**0.32**	**0.38**	**0.23**	0.69	**0.36**	**0.20**	**0.575**	**0.53**	**0.62**
ViT-L/14(CLIP)	0.28	0.27	0.25	0.57	0.29	0.24	0.57	0.53	0.61
FT with L_{CLIP}	0.26	0.27	0.25	0.56	0.30	0.23	0.57	0.53	0.61
FT with $L_{CLIP}+L_{SDS}$	**0.295**	**0.32**	0.25	0.53	**0.32**	0.18	**0.595**	**0.55**	**0.64**
ViT-L/14-336(CLIP)	0.27	0.32	0.21	0.57	0.30	0.19	0.57	0.53	0.61
FT with L_{CLIP}	0.23	0.28	0.19	0.53	0.26	0.17	0.57	0.53	0.61
FT with $L_{CLIP}+L_{SDS}$	**0.285**	**0.34**	**0.23**	0.56	**0.31**	**0.21**	**0.585**	**0.54**	**0.63**
ResNet-50(CLIP)	0.25	0.29	0.19	0.5	0.27	0.18	0.58	0.53	0.63
FT with L_{CLIP}	0.24	0.27	0.20	0.49	0.27	0.16	0.575	0.52	0.63
FT with $L_{CLIP}+L_{SDS}$	**0.265**	**0.30**	**0.21**	0.42	**0.29**	**0.19**	**0.60**	**0.55**	**0.66**

Table 9.1: **Our fine-tuning method SDS-CLIP improves CLIP performance on the Winoground benchmark by 1.5% to 7% and upto 3% for the ARO-Relation and Attribution tasks across various CLIP variants**. Specifically, we find that our method improves on the sub-categories involving *object-swap* and *relational* understanding which comprise of the majority of the tasks in Winoground. Note that *only* fine-tuning with image-text pairs from MS-COCO without the distillation loss does not lead to any improvements.

an image-matching task. This can be formalized as follows: for an image x and caption c, the denoising diffusion score, denoted by $d(x,c)$, is defined as:

$$d(x,c) = \mathbb{E}_{t\sim T,\varepsilon\sim\mathcal{N}(0,I)}[\|\varepsilon_\theta(v_\alpha(x),t,c)-\varepsilon\|^2] \tag{9.1}$$

This denoising diffusion score can then be used to select a correct caption c^* from C as:

$$c^* = \arg\min_{c\in C} \mathbb{E}_{t\sim T,\varepsilon\sim\mathcal{N}(0,I)}[\|\varepsilon_\theta(v_\alpha(x),t,c)-\varepsilon\|^2] \tag{9.2}$$

where t is the sampled time-step, ε_θ is the noise prediction UNet, v_α is an encoder (e.g., VQ-VAE) which maps the image x to a latent code and ε is the sampled Gaussian noise. Previous works [125] have demonstrated that by adopting this approach, text-to-image models performing strongly on visio-linguistic reasoning benchmarks like Winoground, outperforming contrastive models like CLIP by a significant margin (see Fig. 9.1). For ARO, we obtain an accuracy of 0.63 with the diffusion score which is better than CLIP models.

9.1.3 SDS-CLIP: Our Method

The core idea of our approach is to regularize the contrastive objective in CLIP with the denoising diffusion score from Stable Diffusion (see Eq.(9.1)). Our method builds on the recent work of [187] which maps the output of a 3D NeRF model into the input space of Stable Diffusion's UNet and optimizes its parameteres with the denoising diffusion loss, also known as the score-distillation sampling (SDS). In a similar vein, we fine-tune the parameters of CLIP using SDS. Intuitively, our set-up can be viewed as a form of knowledge distillation where the teacher is the text-to-image model and the student is CLIP. As a result, in inference, CLIP can benefit from the visio-linguistic reasoning capabilities that are already learned by text-to-image diffusion models.

Formally, we map the output of CLIP's image encoder to the input space of Stable Diffusion's UNet. Specifically, we pass a given image x through CLIP's image encoder f_ϕ and map its <CLS> embedding through a linear map $h_w \in \mathscr{R}^{d\times 4\times 64\times 64}$ into the input space of Stable Diffusion's UNet ε_θ. This can be formalized as $\varepsilon_\theta(h_w(f_\phi(x)),t,c)$ where t is the time step and c is the corresponding text caption for the given image. We then use this term in place of $\varepsilon_\theta(v_\alpha(x),t,c)$ in Eq. (9.2) to arrive as a denoising diffusion loss L_{SDS} which encourages image-text binding with feedback from the diffusion loss:

$$L_{SDS} = \mathbb{E}_{t\sim T,\varepsilon\sim\mathscr{N}(0,I)}[\|\varepsilon_\theta(h_w(f_\phi(x)),t,c) - \varepsilon\|^2 \tag{9.3}$$

We practically implement this by adding this L_{SDS} loss to the original contrastive objective of CLIP such that it acts as a regularizer:

$$L_{total} = L_{CLIP} + \lambda L_{SDS} \tag{9.4}$$

where L_{CLIP} is contrastive loss for CLIP and λ is a hyper-parameter that can be set with a grid search. We note that there are multiple ways to incorporate a diffusion loss into CLIP's objective. We found that as an additional loss term led to the best results, however, we include the full set of design choices we considered in the Appendix.

Similar to differentiable image parameterizations [172] where a given function is optimized by backpropogation through the image generation process, the UNet parameters θ are kept frozen during the optimization process. Specifically, given $L_{total}(\phi, \gamma, w, \theta)$:

$$\phi*, \gamma*, w* = \min_{\phi,\gamma,w} L_{total}(\phi, \gamma, w, \theta) \tag{9.5}$$

where ϕ, γ, w are the learnable parameters of CLIP's image encoder, text encoder and the linear map between CLIP and Stable Diffusion's UNet.

9.1.4 Experiments

In this section, we empirically validate our proposed method SDS-CLIP on two types of tasks: i) visio-linguistic reasoning using two challenging benchmarks (Winoground, ARO) and ii) zero-shot image classification using a suite of downstream datasets (ImageNet, CIFAR-100, and others). Overall, we show that our method improves CLIP's performance significantly on Winoground and some key tasks in ARO, while also marginally improving downstream zero-shot classification performance.

Experimental Setup

CLIP Models. We consider the following CLIP variants in our experiments: (i) CLIP ViT-B/16; (ii) CLIP ViT-B/32; (iii) CLIP-ViT-L-14; (iv) CLIP-ViT-L-14 336px; (v) CLIP-ResNet-50.

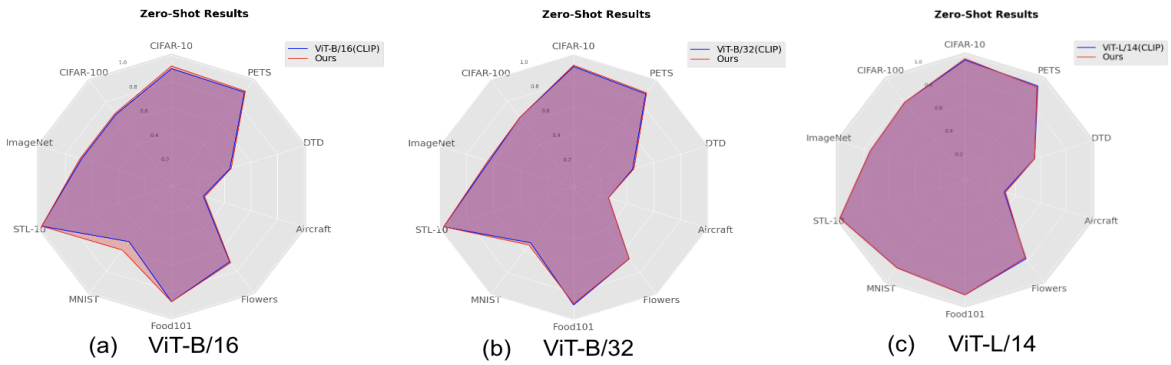

Figure 9.2: **Our fine-tuning method does not harm the zero-shot abilities of CLIP.** In fact for certain downstream datasets (e.g., ImageNet, CIFAR-10, MNIST, Aircraft) – we observe an improvement in the zero-shot performance between $1\% - 8\%$ for ViT-B/16. For other CLIP models, we find no drop in zero-shot performance.

Implementation Details. Due to computational limit, we fine-tune CLIP from a publicly available checkpoint instead of training from scratch. Notably, we only fine-tune CLIP's LayerNorm parameters following [21] along with the linear transformation h_w – accounting for only $\approx 8M$ trainable parameters. We fine-tune these parameters using image-text pairs from MSCOCO [146]. In particular, we choose MSCOCO as it is relatively small and less noisy than other image-text datasets such as CC-12M [208]. Both these factors make our fine-tuning method extremely sample- and parameter-efficient.

Baselines. We compare our method with two different baselines: (i) pre-trained (vanilla) CLIP checkpoints; and (ii) CLIP fine-tuned on MS-COCO with the standard contrastive loss without the regularization term.

Results

Winoground. We evaluate SDS-CLIP on the challenging visio-linguistic reasoning benchmark, Winoground [223]. In Table (9.1), we find that our approach consistently improves performance across all Winoground sub-categories and CLIP variants, yielding absolute improvements ranging from 1.5% to 7%. The largest gain of 7% is observed in ViT-B/16

(CLIP), with other CLIP variants showing consistent improvements of 1.5% to 2%. On further inspection of the Winoground sub-categories, we find that SDS-CLIP shows consistent improvements in “object-swap" and “relation". It is worth noting that the “both” sub-category, which combines both “object-swap" and “relation" tags, makes up only $\tilde{5}$% of all tasks, thus are potentially not fully representative of all scenarios involving both object swaps and relational understanding. We also analyse SDS-CLIP’s robustness to the number of predicates in captions and find that overall, it enhances performance in tasks where there are both one and two predicates.

ARO. The ARO dataset [262] comprises tasks for (i) attribute-understanding and (ii) relational-understanding. In Table 9.1, we find that SDS-CLIP enhances performance by 1%-3% in the "attribute-binding" and "relational understanding" tasks.

Impact on CLIP’s zero-shot performance. From Fig. 9.2, we find that SDS-CLIP’s zero-shot classification capbilities are not impacted, relative to vanilla CLIP. In fact, we find that ViT-B/16’s zero-shot performance improves across a range of downstream datasets (with up to 8% improvement for MNIST).

9.1.5 Related Works

While CLIP models [191] are renowned for their robust zero-shot classification, recent research [60, 223] has exposed their limitations in visio-linguistic reasoning. In contrast, recent studies have demonstrated that text-to-image models [41, 49, 125, 136] outperform CLIP in reasoning tasks. These models in fact leverage scores computed from the diffusion objective. We note that while [187] use score-distillation sampling for text to 3D generation, ours is the first work to adapt the formulation as a regularizer and improve compositional abilities in CLIP.

9.1.6 Conclusion

Our paper introduces SDS-CLIP, a novel data and parameter-efficient method that effectively enhances CLIP's visio-linguistic reasoning abilities by distilling knowledge from text-to-image models, without compromising its zero-shot abilities.

9.2 Compositionality in Text-to-Image Diffusion Models

9.2.1 Introduction

Text-to-image diffusion-based generative models [186, 193, 199, 201] have achieved photo-realistic image generation capabilities on user-defined text prompts. However, recent studies [103] reveal that text-to-image models struggle with maintaining high fidelity when handling simple compositional prompts, such as those consisting of attributes, objects, and their associated relations (e.g., "*a red book and a yellow vase*"). This hinders the use of these generative models in various creative scenarios where the end-user wants to generate scenes that accurately reflect the composition and relationships specified in the prompt.

Existing approaches [1, 39, 73, 245] explore various strategies to enhance compositionality in text-to-image models. These methods primarily focus on modifying cross-attention maps by utilizing bounding box annotations and performing optimizations in the latent space during inference. Recent advancements, such as fine-tuning the UNet [103], have also demonstrated improvements in compositionality. However, the *core reasons* behind compositionality failures remain poorly understood. Gaining insights into these root causes is crucial for developing more effective approaches to augment these models with enhanced compositional capabilities.

In our paper, we investigate the potential causes of compositional attribute binding failures

in text-to-image generative models, where the model fails to correctly associate descriptive attributes (such as color, shape, or texture) with the corresponding objects in the generated images. We identify two key sources of error: (i) *Erroneous attention contributions in CLIP output token embeddings*: We observe that output token embeddings in CLIP have significant attention contributions from irrelevant tokens, thereby introducing errors in generation. To explore this, we compare the internal attention contributions in CLIP for compositional prompts with the T5 text encoder, known for its stronger compositionality. Quantitative analysis shows that T5 exhibits fewer erroneous attention contributions than CLIP, indicating a potential reason for its superior compositionality. (ii) *Sub-optimality of CLIP output space for compositional prompts*: We find out that there exists an alternative text-embedding space capable of generating highly coherent images from compositional prompts. This indicates that the current CLIP output space is inherently sub-optimal. Specifically, optimizing CLIP's text embeddings, while keeping the Stable Diffusion UNet frozen, converges to a more effective embedding space, enabling better compositional image generation. These findings highlight that refining the output space of the CLIP text encoder could play a critical role in enhancing compositionality.

Building on our observations about the deficiencies of CLIP and identifying its text-embedding space as *a core issue* in compositional attribute binding, we explore augmenting diffusion models with a lightweight module to enhance the text-encoder's output and improve compositionality. Remarkably, a simple linear projection achieves significant improvements, comparable or superior to full fine-tuning of CLIP or training more complex networks on top of it. We demonstrate that this linear projection effectively aligns the CLIP text-encoder's output with a more optimal embedding space, leading to significantly stronger compositional performances.

In particular, we introduce Window-based Compositional Linear Projection (`WiCLP`), a

lightweight fine-tuning method that significantly improves the model's performance on compositional prompts, achieving results comparable to or surpassing existing methods. Additionally, `WiCLP` preserves the model's overall performance, maintaining high fidelity on clean prompts as evidenced by a low FID score, while offering a solution that is both *parameter efficient* and *speed efficient*. This ensures robust compositional capabilities without compromising the model's general effectiveness.

In summary, our contributions are as follows:

- We perform an in-depth analysis of the reasons behind compositionality failures in text-to-image generative models, with a particular focus on investigating the attribute binding aspect of compositionality. We highlight two key reasons contributing to these failures.
- Building on our observations, we propose `WiCLP` as an enhancement for SD v1.4, SD v2, SDXL, SD v3, DeepFloyd, and PixArt-α. This method significantly improves the models' compositional attribute binding, while preserving accuracy on standard prompts. We observe improvements of $16.18\%, 15.15\%$, and 9.51% on SD v1.4, $14.35\%, 11.14\%$, and 6% on SD v2, $20.31\%, 13.4\%$, and 5% on SDXL, and 14.16%, 9.82%, and 2.63% on PixArt-α in VQA scores across color, texture, and shape datasets, respectively. Our method outperforms or matches existing baselines in VQA scores, while achieving a superior FID score. It requires *fewer parameters for optimization* and enables *faster inference*, making it both efficient and effective. Extensive evaluations with T2I-CompBench, TIFA, and human studies validate our method's effectiveness.

9.2.2 Background

Compositionality in Text-to-Image Generative Models. Compositionality in text-to-image models refers to the ability of a model to accurately capture the correct compositions of objects, their corresponding attributes, and the relationships between objects described in a given prompt. [103] introduced a benchmark designed to evaluate compositionality in text-to-image models, highlighting the limitations of models when handling compositional prompts. The benchmark employs disentangled BLIP-Visual Question Answering (VQA) as a metric for assessing image compositional quality. The VQA score assesses how accurately an image captures the compositional elements described in the prompt by utilizing a vision-language model. This metrics demonstrates a closer correlation with human judgment compared to metrics like CLIP-Score [97]. The authors also proposed a fine-tuning baseline to enhance compositionality in these models. Alternatively, compositionality issues can be addressed at inference by modifying cross-attention maps using hand-crafted loss functions and bounding boxes derived from a language model [1, 39, 73, 143, 150, 175, 245]. However, [103] showed that data-driven fine-tuning is more effective for improving compositionality.

Text-to-image Diffusion Models In diffusion models, noise is added to the data following a Markov chain across multiple time-steps $t \in [0,T]$. Starting from an initial random real image $\mathbf{x}_0$ along with its caption c, $(\mathbf{x}_0, c) \sim \mathscr{D}$, the noisy image at time-step t is defined as $\mathbf{x}_t = \sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{(1-\alpha_t)}\mathbf{ffl}$. The denoising network denoted by $\varepsilon_\theta(\mathbf{x}_t, \mathbf{c}, t)$ is trained to denoise the noisy image $\mathbf{x}_t$ to obtain $\mathbf{x}_{t-1}$. For efficiency, the noising and the denoising operations occur in a latent space defined by $\mathbf{z} = \mathscr{E}(\mathbf{x})$, where $\mathscr{E}$ is an encoder such as VQ-VAE [234]. Usually, the conditional input $\mathbf{c}$ to the denoising network $\varepsilon_\theta(.)$ is a text-embedding of the caption c through a text-encoder $\mathbf{c} = v_\gamma(c)$. The training objective for

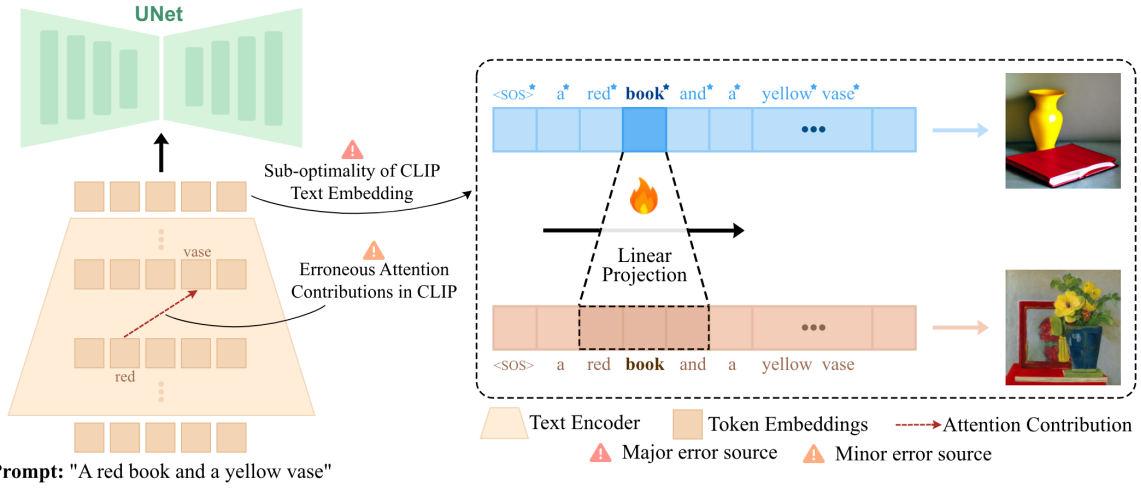

Figure 9.3: Overview of our analysis and proposed methods. The figure identifies two sources of errors in Stable Diffusion's inability to generate compositional prompts: (i) erroneous attention contribution in CLIP (minor) and (ii) sub-optimal CLIP text embedding (major). We propose a window-based linear projection (`WiCLP`), applying linear projection to a token's surrounding window to enhance embeddings.

diffusion models can be defined as follows:

$$\mathscr{L}(\theta) = \mathbb{E}_{(\mathbf{x}_0,c)\sim\mathscr{D},\varepsilon,t}\left[\|\varepsilon - \varepsilon_\theta(\mathbf{z}_t,\mathbf{c},t)\|_2^2\right],$$

where θ is the set of learnable parameters in the UNet ε_θ. During inference, given a text-embedding $\mathbf{c}$, a random Gaussian noise $\mathbf{z}_T \sim \mathscr{N}(0,I)$ is iteratively denoised to produce the final image.

9.2.3 Sources of Compositionality Failures

This section conducts an in-depth analysis of compositional attribute binding failures in text-to-image models, focusing on the CLIP text-encoder.

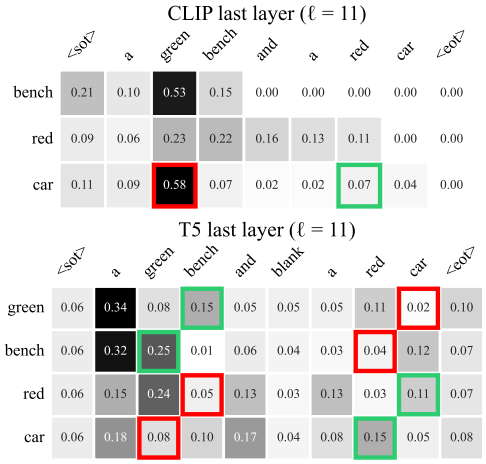

Figure 9.4: The heatmap illustrates unintended attention contributions in CLIP, while highlighting the more accurate performance of T5.

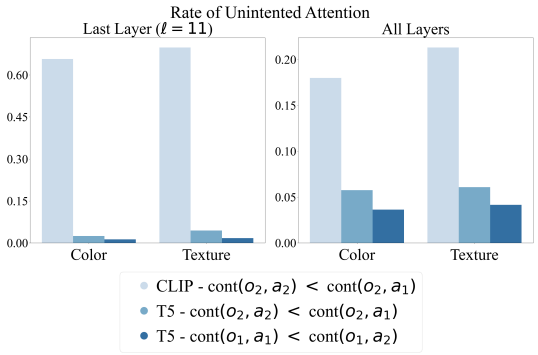

Figure 9.5: Quantitatively, we find CLIP to have significantly higher erroneous attention contributions averaged across prompts of color and texture datasets.

Source (i) : Erroneous Attention Contributions in CLIP

In this section, we leverage attention contributions [54, 68] to analyze how the final text-embeddings of compositional prompts are obtained by the CLIP text-encoder, a widely adopted component in many text-to-image models. We then compare these attention contribution patterns with those produced by the T5 text-encoder which is known for its stronger compositional capabilities. Many of the compositional prompts from [103] have a decomposable template of the form $\mathbf{a}_i\ \mathbf{o}_j + \mathbf{a}_j\ \mathbf{o}_j$, where $\mathbf{a}_i, \mathbf{a}_j$ are attributes (e.g., "black", "matted") and $\mathbf{o}_i, \mathbf{o}_j$ represent objects (e.g., "car", "bag").

The attention mechanism in layer ℓ of a transformer consists of four weight matrices $W_{\mathrm{q}}, W_{\mathrm{v}}, W_{\mathrm{k}}$, and W_{o} [236]. Each of these matrices is divided into H heads, denoted by $W_{\mathrm{q}}^h, W_{\mathrm{v}}^h, W_{\mathrm{k}}^h \in \mathbb{R}^{d \times d_h}, W_{\mathrm{o}}^h \in \mathbb{R}^{d_h \times d}$, where $h \in [H]$. Here, d_h denotes the dimensionality of the internal token embeddings. For simplicity, we omit ℓ, but each layer has its own attention matrices. These matrices operate on the token embeddings produced by the previous layer $(\ell - 1)$, denoted as $\bar{\mathbf{x}}_j$ for token j. We further denote the projections of $\bar{\mathbf{x}}_j$ onto the query, key, and value matrices of the h-th attention head in layer ℓ as q_j^h, k_j^h, and v_j^h, respectively. More precisely,

$$\mathrm{q}_j^h = \bar{\mathbf{x}}_j W_{\mathrm{q}}^h, \quad \mathrm{k}_j^h = \bar{\mathbf{x}}_j W_{\mathrm{k}}^h, \quad \mathrm{v}_j^h = \bar{\mathbf{x}}_j W_{\mathrm{v}}^h.$$

The *contribution* of token j to token i in layer ℓ, denoted by $\mathrm{cont}_{i,j}$, is computed as follows:

$$\mathrm{cont}_{i,j} = \left\| \sum_{h=1}^{H} \mathrm{attn}_{i,j}^h \, \mathrm{v}_j^h \, W_{\mathrm{o}}^h \right\|_2$$

where $\mathrm{attn}_{i,j}^h$ is the attention weight of token i to j in the h-th head of layer ℓ. Specifically,

$$\mathrm{attn}_{i,\cdot}^h = \textsc{Softmax}\left(\left\{ \frac{\langle \mathrm{q}_i^h, \mathrm{k}_j^h \rangle}{\sqrt{d_h}} \right\}_{j=1}^{n} \right).$$

Notably, $\mathrm{cont}_{i,j}$ is a metric that quantifies the *contribution* of a token j to the norm of a token i at layer ℓ. We employ this metric to identify layers in which important tokens highly attend to *unintended* tokens, or lowly attend to *intended* ones.

Key Finding: T5 has less erroneous attention contributions than CLIP. We refer to Figure 9.4 that visualizes attention contribution of both T5 and CLIP text-encoder in the last layer $(\ell = 11)$ for the prompt “a green bench and a red car". Ideally, the attention mechanism should guide the token “car” to focus more on “red” than “green”, but in the last layer of the CLIP text-encoder, “car” significantly attends to “green”. In contrast, T5 shows a more consistent attention pattern, with “red” contributing more to the token “car”

and “green” contributing more to the token “bench”.

We further conduct a comprehensive analysis focusing on specific types of compositional prompts from the T2I-CompBench dataset [103]. This includes 780 prompts from the color category and 582 prompts from the texture category of this dataset, each following the structured format: “$\mathbf{a}_1$ $\mathbf{o}_1$ and $\mathbf{a}_2$ $\mathbf{o}_2$”. For each prompt, we obtain attention contributions in all layers and count the number of layers where *unintended attention contributions* occur. In the CLIP text-encoder, unintended attention occurs when $\mathbf{o}_2$ attends more to $\mathbf{a}_1$ than $\mathbf{a}_2$. For T5, it occurs when $\mathbf{o}_2$ attends more to $\mathbf{a}_1$ than $\mathbf{a}_2$, or $\mathbf{o}_1$ attends more to $\mathbf{a}_2$ than $\mathbf{a}_1$. Figure 9.5 provides a quantitative comparison of unintended attention across various prompts between the CLIP text-encoder and T5. The T5 model demonstrates superior performance on our metric compared to the CLIP text-encoder, reinforcing the hypothesis that erroneous attention mechanisms in CLIP may contribute to its weaker compositional performance in text-to-image models.

To address the attention shortcomings of the CLIP text-encoder, we explored zero-shot reweighting of attention maps in CLIP to reduce unintended attentions while enhancing meaningful ones. While this improved baseline performance, it fell short of our primary method discussed in the following sections.

Source (ii) : Sub-optimality of CLIP Text-Encoder for Compositional Prompts

In this section, we investigate whether the UNet is capable of generating better compositional scenes if provided with alternative (*improved*) text embeddings, rather than relying on the output of the CLIP text-encoder. For a given input prompt p with a specific composition (e.g., *“a red book and a yellow table”*), we utilize our dataset (described in Section 9.2.6) to obtain $\mathscr{D}_p$, a set of high-quality compositional images corresponding to prompt p. Next,

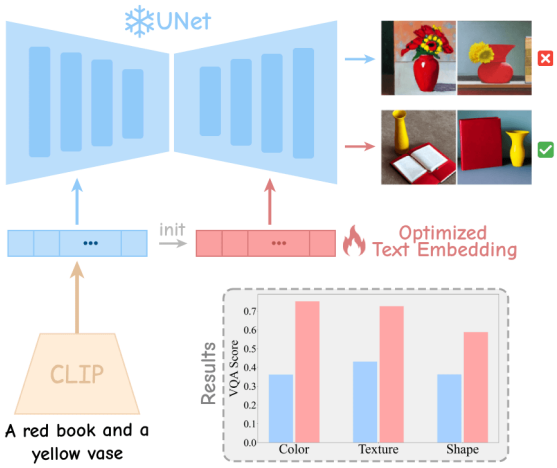

Figure 9.6: Sub-optimality of CLIP Text-Encoder for Compositional Prompts. Optimizing a learnable vector to represent an improved text embedding, while keeping the UNet frozen, enables the generation of more compositionally accurate images.

we extract the text embedding $\mathbf{c}$ from the CLIP text-encoder for prompt p. Using this embedding as the initialization, we create a learnable vector $\mathbf{c}^*$ of the same dimensionality. Keeping all other components (such as the UNet) frozen, we optimize this learnable vector as follows:

$$\mathbf{c}^* = \arg\min_{\mathbf{c}} \mathbb{E}_{x_0 \sim \mathscr{D}_p, \varepsilon, t} \left[\|\varepsilon - \varepsilon_\theta(\mathbf{z}_t, \mathbf{c}, t)\|_2^2 \right].$$

We then use the optimized text embedding $\mathbf{c}^*$ to generate images with the UNet ε_θ. Figure 9.6 illustrates the complete pipeline.

Key Results. Utilizing Stable Diffusion v1.4, we optimize optimize $\mathbf{c}^*$ for all compositional prompts across the color, texture, and shape categories in the T2I-CompBench dataset. By generating samples with $\mathbf{c}^*$ and comparing them to those generated using $\mathbf{c}$, we observe a significant improvement in the VQA scores. As shown in Figure 9.6, CLIP text embeddings yield VQA scores of 0.3615 for color, 0.4306 for texture, and 0.3619 for shape. In contrast, the optimized embeddings achieve 0.7513 for color, 0.7254 for texture, and 0.5873 for shape.

These results indicate that CLIP text-encoder does not output the proper text-embedding suitable for generating compositional scenes. However, the existence of an optimized embedding space demonstrates that the UNet can generate coherent compositional outputs when provided with appropriately improved embeddings. This finding motivates the idea of improving the CLIP output space to mitigate compositionality issues in text-to-image diffusion models.

9.2.4 Projection Layer for Enhancing Compositionality in the CLIP Text Embedding Space

Building on our previous findings, we focus on improving the text embedding space utilized in text-to-image generative models. Specifically, we propose learning a projection layer over the CLIP output embedding space to transform its sub-optimal representation into an enhanced space better suited for compositionality. In the following sections, we introduce two methods, `CLP` and `WiCLP`, which implement linear projections of the CLIP output embedding space to achieve this enhancement.

`CLP`: Token-wise Compositional Linear Projection

Given the text-embedding $\mathbf{c} \in \mathbb{R}^{n \times d}$ as the output of the text-encoder for prompt c, i.e., $\mathbf{c} = v_\gamma(c)$, we train a linear projection $\texttt{CLP}_{W,b} : \mathbb{R}^{n \times d} \to \mathbb{R}^{n \times d}$. This projection includes a matrix $W \in \mathbb{R}^{d \times d}$ and a bias term $b \in \mathbb{R}^d$, which are applied token-wise to the output text-embeddings of the text-encoder. More formally, for $\mathbf{c} \in \mathbb{R}^{n \times d}$ including text-embeddings of n tokens $\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_n \in \mathbb{R}^d$, $\texttt{CLP}_{W,b}(\mathbf{c})$ is obtained by stacking projected embeddings $\mathbf{c}'_1, \mathbf{c}'_2, \cdots, \mathbf{c}'_n$ where $\mathbf{c}'_i = W^T \mathbf{c}_i + b$.

Finally, we solve the following optimization problem on a dataset $\mathscr{D}$ including image-caption

pairs of high-quality compositional images:

$$W^*, b^* = \arg\min_{W,b} \mathbb{E}_{(x_0,c)\sim\mathscr{D},\varepsilon,t}\left[\Phi_{\texttt{CLP}}\right]$$

$$\Phi_{\texttt{Proj}} = \left\|\varepsilon - \varepsilon_\theta\left(\mathbf{z}_t, \texttt{Proj}_{W,b}\left(\mathbf{c}\right), t\right)\right\|_2^2$$

We then apply $\texttt{CLP}_{W^*,b^*}$ on CLIP text-encoder to obtain improved embeddings.

WiCLP: Window-based Compositional Linear Projection

In this section, we propose a more advanced linear projection scheme where the new embedding of a token is derived by applying a linear projection on that token in conjunction with a set of its adjacent tokens within a specified window. This method not only leverages the benefits of CLP but also incorporates the contextual information from neighboring tokens, potentially leading to more precise text-embeddings.

More formally, we train a mapping $\texttt{WiCLP}_{W,b} : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d}$ including a parameter s (indicating window length), matrix $W \in \mathbb{R}^{(2s+1)d\times d}$, and a bias term $b \in \mathbb{R}^d$. For text-embeddings $\mathbf{c} \in \mathbb{R}^{n\times d}$ consisting of n token embeddings of $\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_n \in \mathbb{R}^d$, we obtain $\texttt{WiCLP}_{W,b}$ by stacking projected embeddings $\mathbf{c}'_1, \mathbf{c}'_2, \cdots, \mathbf{c}'_n$ where

$$\mathbf{c}'_i = W^T \textsc{ Concatenation}\left(\left(\mathbf{c}_j\right)_{j=i-s}^{i+s}\right) + b$$

Similarly, we solve the following optimization problem to train the projection:

$$W^*, b^* = \arg\min_{W,b} \mathbb{E}_{(x_0,c)\sim\mathscr{D},\varepsilon,t}\left[\Phi_{\texttt{WiCLP}}\right]$$

In this section, we propose a more advanced linear projection scheme where the new embedding of a token is derived by applying a linear projection on that token in conjunction with a set of its adjacent tokens within a specified window. This method not only leverages the benefits of CLP but also incorporates the contextual information from neighboring tokens,

potentially leading to more precise text-embeddings.

More formally, we train a mapping $\texttt{WiCLP}_{W,b} : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d}$ including a parameter s (indicating window length), matrix $W \in \mathbb{R}^{(2s+1)d\times d}$, and a bias term $b \in \mathbb{R}^d$. For text-embeddings $\mathbf{c} \in \mathbb{R}^{n\times d}$ consisting of n token embeddings of $\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_n \in \mathbb{R}^d$, we obtain $\texttt{WiCLP}_{W,b}$ by stacking projected embeddings $\mathbf{c}'_1, \mathbf{c}'_2, \cdots, \mathbf{c}'_n$ where

$$\mathbf{c}'_i = W^T \textsc{Concatenation}\left(\left(\mathbf{c}_j\right)_{j=i-s}^{i+s}\right) + b$$

Similarly, we solve the following optimization problem to train the projection:

$$W^*, b^* = \arg\min_{W,b} \mathbb{E}_{(x_0,c)\sim\mathscr{D},\varepsilon,t}\left[\Phi_{\texttt{WiCLP}}\right]$$

We observe that WiCLP improves over CLP (special case of WiCLP with $s = 0$) by incorporating adjacent tokens along with the token itself. This approach enhances embeddings by reinforcing the contributions of relevant adjacent tokens.

9.2.5 SWITCH-OFF: Trade-off between Compositionality and Clean Accuracy

Fine-tuning models or adding modules to a base model often results in a degradation of image quality and an increase in the Fréchet Inception Distance (FID) score. To balance the trade-off between improved compositionality and the quality of generated images for clean prompts, nspired by [96], we adopt SWITCH-OFF, where we apply the linear projection only during the initial steps of inference. Specifically, given a time-step threshold τ, for $t \geq \tau$, we use $\texttt{WiCLP}_{W^*,b^*}(\mathbf{c})$, while for $t < \tau$, we use the unchanged embedding $\mathbf{c}$ as the input to the cross-attention layers.

Figure 9.7 illustrates the trade-off between VQA score and FID on a randomly sampled

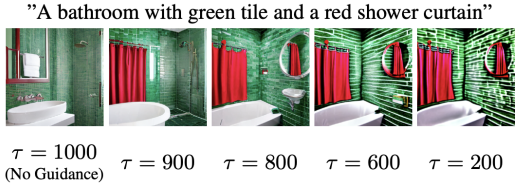

Figure 6: Qualitative results showing the impact of SWITCH-OFF with varying thresholds τ.

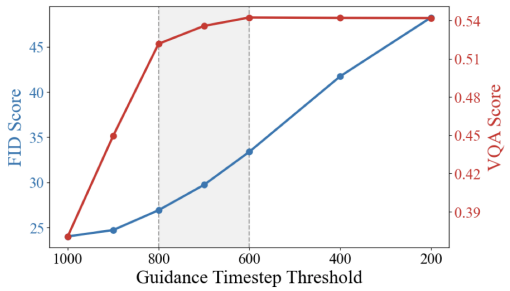

Figure 9.7: Trade-off between VQA and FID scores with SWITCH-OFF at different thresholds.

subset of MS-COCO [145] for different choices of τ. As shown, even a large value of τ suffices for obtaining high-quality compositional scenes as the composition of final generated image is primarily formed at early steps. Thus, choosing a large τ preserves the model's improved compositionality while maintaining its clean accuracy. Setting $\tau = 800$ offers a competitive VQA score compared to the model where projection is applied at all time steps, and achieves a competitive FID similar to that of the clean model. Figure 9.7 depicts a few images generated using different choices of τ.

9.2.6 Experiments

Existing Baselines. We evaluate the performance of multiple methods alongside standard models SD v1.4, SD v2, SDXL [186], SD v3 [69], and PixArt-α [43]. These include Composable Diffusion [151], which addresses concept conjunction and negation in pre-trained diffusion models; Structured Diffusion [72], which focuses on attribute binding; Attn-Exct [39], which ensures correct attention to all subjects in the prompt using iterative

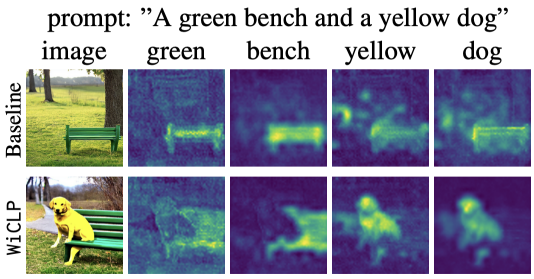

Figure 9.8: More accurate cross-attention maps using CLP.

optimizations; GORS [103], which fine-tunes Stable Diffusion v2 using a reward function; GLIGEN [142], which utilizes grounding inputs such as bounding boxes; RealCompo [266], which integrates spatial-aware diffusion models; and FLUX [128].

Training Setup. All of the models are trained using the objective function of diffusion models on color, texture, and shape datasets described in Section 9.2.6. During training, we keep all major components frozen, including the U-Net, CLIP text-encoder, and VAE encoder and decoder, and only the linear projections are trained.

Dataset Collection. We utilize the T2I-CompBench dataset [103], a well-recognized dataset for compositionality, focusing on three categories: color, texture, and shape, with a total of 1,000 prompts across both training and evaluation sets.

To generate high-quality images, we use three generative models: SD v1.4, DeepFloyd, and SynGen [194], creating 210 samples per prompt. This ensures a wide variety of generated images, leveraging each model's strengths. From these, we selected the top 30 with the highest VQA scores to ensure the final dataset consists of images that best reflect the prompts.

Furthermore, for SDXL, SD v3, and PixArt-α, we explored training `WiCLP` (`WiCLP`* in Table 9.2) on a higher-quality dataset generated by newer text-to-image models, such as

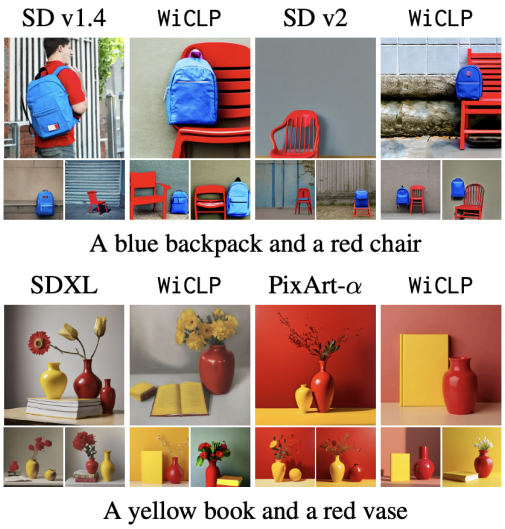

Figure 9.9: Qualitative comparison of baselines and our projection method (WiCLP). Incorporating WiCLP significantly improves image alignment with the prompts.

SDXL and SD v3. Importantly, leveraging an appropriately curated dataset results in a substantial improvement in VQA scores, highlighting the importance of high-quality training data for compositional understanding.

Qualitative and Quantitative Evaluation

Qualitative Evaluation. Figure 9.9 presents images generated when applying WiCLP. When generating compositional prompts with a baseline model, objects may be missing or attributes are incorrectly applied. However, with WiCLP, objects and their corresponding attributes are more accurately generated.

Figure 9.8 illustrates cross-attention maps for a sample prompt. In the base model, attention maps are flawed, with some tokens incorrectly attending to the wrong pixels. However, with both CLP and WiCLP, objects and attributes more accurately attend to their respective pixels.

Quantitative Evaluation. Table 9.2 presents the VQA scores for our methods, CLP and

	Model	Color	Texture	Shape
SD v1.4	Baseline	0.3765	0.4156	0.3576
	CLP	0.4837	0.5312	0.4307
	WiCLP	**0.5383**	**0.5671**	**0.4527**
SD v2	Baseline	0.5065	0.4922	0.4221
	Composable	0.4063	0.3645	0.3299
	Structured	0.4990	0.4900	0.4218
	Attn-Exct	0.6400	0.5963	0.4517
	GORS	0.6414	0.6025	0.4546
	CLP	0.6075	0.5707	0.4567
	WiCLP	**0.6500**	**0.6036**	**0.4821**
SDXL	Baseline	0.5770	0.5217	0.4666
	WiCLP	0.6930	0.6007	0.4758
	WiCLP*	**0.7801**	**0.6557**	**0.5166**
PixArt-α	Baseline$_{512^2}$	0.3877	0.4557	0.4094
	Baseline$_{1024^2}$	0.4156	0.4594	0.3849
	WiCLP*$_{512^2}$	**0.5293**	**0.5539**	**0.4357**
SD v3	Baseline	0.8164	0.7303	0.5852
	WiCLP*	**0.8213**	**0.7488**	**0.5963**
Others	FLUX	0.7354	0.6016	0.4777
	GLIGEN	0.4288	0.3904	0.3998
	RealCompo	0.7741	0.7427	0.6032

Table 9.2: Quantitative comparison with state-of-the-art and baseline methods across different categories of the T2I-CompBench dataset

WiCLP, alongside the baselines discussed. VQA scores of our method and other discussed baselines are provided in Table 9.2. As shown, both CLP and WiCLP significantly improve upon the baselines. Both methods demonstrate substantial improvements over the baselines, with WiCLP achieving the highest VQA scores among state-of-the-art approaches that utilize the same baseline model, while being more computationally and parameter-efficient. Additionally, to further validate the performance gains, we evaluated our method using additional metrics, including TIFA [101]. The results demonstrated consistent improvements over the baselines, reinforcing the effectiveness of our approach. For analysis of WiCLP's robustness and generalizability—both when trained across all categories and when applied to models using T5 text encoders.

Notably, our methods maintain the model's general utility, introducing only a slight increase in the FID score; for example, experiments on MS-COCO prompts show that while our methods slightly increase FID compared to base models, this increase is smaller than that of other baselines—for instance, `WiCLP` achieves an FID score of 27.40, outperforming GORS at 30.54.

Human Experiments. We conducted a human evaluation where participants compared images generated by SD v1.4 and SD v1.4 + `WiCLP`, selecting the image that best matched the given prompt. The results showed that in 34.625% of cases, evaluators chose the base model; in 51.875%, they preferred the `WiCLP`; and in 13.50%, they rated both equally.

9.2.7 Impact of `WiCLP` on Subsets of Tokens

To better understand the impact of `WiCLP` on token embeddings, we applied the trained `WiCLP` to specific subsets of tokens from a sample of dataset sentences. In particular, we compare the following token groups: nouns only; nouns and adjectives; nouns, adjectives, and the EOS (End of Sentence token) token; all sentence tokens; and all tokens outputted by CLIP (sentence tokens plus padding tokens). As can be seen, applying `WiCLP` only to a small number of tokens is sufficient for improving compositionality. Interestingly, applying `WiCLP` to the group of nouns, adjectives, and EOS achieves even higher VQA scores than applying `WiCLP` to all tokens. Despite these findings, we applied `WiCLP` to all tokens in our work, leaving this targeted approach for future research.

9.2.8 Alternatives to `WiCLP`

We explored various fine-tuning strategies for improving CLIP, including fine-tuning the entire CLIP, fine-tuning only the last layers of CLIP combined with `WiCLP`, and using `WiCLP` alone. Our results show that the original baseline model (SD v1.4) achieves a VQA score

of 0.3765 on the color category of the dataset. Fine-tuning the entire CLIP without WiCLP improves the score to 0.5173, fine-tuning the last layers of CLIP combined with WiCLP achieves 0.5497, and WiCLP alone achieves 0.5383.

These findings highlight the effectiveness of WiCLP, which outperforms full fine-tuning of CLIP while being significantly more parameter-efficient. While fine-tuning the last layers of CLIP combined with WiCLP achieves slightly better performance than using WiCLP alone, it requires optimizing a much larger number of parameters. Given this trade-off, we prioritize WiCLP alone to minimize the number of parameters while achieving substantial compositional performance improvements. Additionally, keeping the original CLIP unchanged makes our approach more suitable for SWITCH-OFF functionality, allowing the module to be easily enabled or disabled as needed. Additionally, we conducted an ablation study on various projection layer architectures, ranging from simple designs to more advanced, parameter-heavy transformer-based models.

9.2.9 Conclusion

We analyze error sources in text-to-image models for generating images from compositional prompts, identifying (i) erroneous attention contributions in CLIP token embeddings and (ii) the CLIP text-encoder's sub-optimal alignment with the UNet. Based on these insights, we propose WiCLP, a simple yet strong baseline that fine-tunes a linear projection on CLIP's representation space. WiCLP though inherently simple and parameter efficient, outperforms existing methods on compositional image generation benchmarks and maintains a low FID score on a broader range of clean prompts.

Chapter 10: **Conclusion and Future Work**

In our thesis, we demonstrate that insights from model interpretability can play a crucial role in designing lightweight methods for various downstream applications, such as model editing, zero-shot spurious correlation mitigation, and data attribution. Overall, our work emphasizes that model interpretability can extend beyond simple observation, providing tangible solutions to real-world challenges.

Looking ahead, I propose to delve deeper into the concept of diffusion transformers and apply these insights to develop more robust model editing techniques. These techniques could enable continual model updates or edits, addressing an important gap in current research. This work has the potential to make a significant real-world impact, as continual model editing remains a largely unexplored area.

With the rise of reasoning-based language models, we are also investigating the faithfulness of reasoning chains in these models. While this is primarily an evaluation project, we aim to leverage these findings to uncover a reasoning circuit within language models. This could have practical applications, such as reducing overthinking in language models, which may enhance their generalization capabilities.

As large foundational models continue to evolve, I believe that model interpretability will serve as a powerful tool for understanding these models in greater depth and refining them

for various applications, including editing, safety, and improved generalization.

10.1 Reading List

My reading list spans three research areas: model interpretability through the lens of *data*, model interpretability through lens of *internal model components*, and model steering or editing.

10.2 Understanding Model Through the Lens of Data

1. Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In Doina Precup and Yee Whye Teh (eds.), Proceedings of the 34th International Conference on Machine Learning, volume 70 of Proceedings of Machine Learning Research, pp. 1885–1894, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL http:// proceedings.mlr.press/v70/koh17a.html. [119]
2. Pang Wei Koh, Kai-Siang Ang, Hubert H. K. Teo, and Percy Liang. On the accuracy of influence functions for measuring group effects. CoRR, abs/1905.13289, 2019b. URL http://arxiv. org/abs/1905.13289. [118]
3. Garima Pruthi, Frederick Liu, Mukund Sundararajan, and Satyen Kale. Estimating training data influence by tracking gradient descent. ArXiv, abs/2002.08484, 2020. [189]
4. Chih-Kuan Yeh, Joon Sik Kim, Ian En-Hsu Yen, and Pradeep Ravikumar. Representer point selection for explaining deep neural networks. CoRR, abs/1811.09720, 2018. URL http: //arxiv.org/abs/1811.09720. [258]
5. Alon Jacovi and Yoav Goldberg. Towards faithfully interpretable nlp systems: How should we define and evaluate faithfulness? Proceedings of the 58th Annual Meeting

of the Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.386. URL http://dx. doi.org/10.18653/v1/2020.acl-main.386. [106]

6. Kristian Georgiev, Joshua Vendrow, Hadi Salman, Sung Min Park, and Aleksander Madry. The journey, not the destination: How data guides diffusion models, 2023. [82]
7. Roger Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, Evan Hubinger, Kamil˙e Lukoši¯ut˙e, Karina Nguyen, Nicholas Joseph, Sam McCandlish, Jared Kaplan, and Samuel R. Bowman. Studying large language model generalization with influence functions, 2023. [88]
8. Xiaochuang Han, Byron C Wallace, and Yulia Tsvetkov. Explaining black box predictions and unveiling data artifacts through influence functions. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 5553–5563, 2020. [90]
9. Shuming Kong, Yanyan Shen, and Linpeng Huang. Resolving training biases via influence based data relabeling. In International Conference on Learning Representations, 2021. [123]

10.3 Understanding Model Through Internal Model Components

1. D. Bau, J. Zhu, H. Strobelt, À. Lapedriza, B. Zhou, and A. Torralba. Understanding the role of individual units in a deep neural network. CoRR, abs/2009.05041, 2020. URL https: //arxiv.org/abs/2009.05041. [25]
2. M. Yuksekgonul, V. Chandrasekaran, E. Jones, S. Gunasekar, R. Naik, H. Palangi, E. Kamar, and B. Nushi. Attention Satisfies: A Constraint-Satisfaction Lens on Factual Errors of Language Models. In International Conference on Learning Representations,

9 2023. [263]

3. K. Wang, A. Variengien, A. Conmy, B. Shlegeris, and J. Steinhardt. Interpretability in the Wild: a Circuit for Indirect Object Identification in GPT-2 small, 2022. [244]
4. S. Tong, Z. Liu, Y. Zhai, Y. Ma, Y. LeCun, and S. Xie. Eyes Wide Shut? Exploring the Visual Shortcomings of Multimodal LLMs, 2024. [225]
5. M. Geva, J. Bastings, K. Filippova, and A. Globerson. Dissecting Recall of Factual Associations in Auto-Regressive Language Models, 2023. [83]
6. N. Cammarata, S. Carter, G. Goh, C. Olah, M. Petrov, L. Schubert, C. Voss, B. Egan, and S. K. Lim. Thread: Circuits. Distill, 2020. doi: 10.23915/distill.00024. https://distill.pub/2020/circuits. [68]
7. T. Oikarinen and T.-W. Weng. Clip-dissect: Automatic description of neuron representations in deep vision networks, 2023. [177]
8. H. Shah, A. Ilyas, and A. Madry. Decomposing and editing predictions by modeling model computation, 2024. [206]
9. Michael Hanna, Ollie Liu, and Alexandre Variengien. How does gpt-2 compute greater-than?: Inter- preting mathematical abilities in a pre-trained language model, 2023. URL https://arxiv. org/abs/2305.00586. [91]
10. Nikhil Prakash, Tamar Rott Shaham, Tal Haklay, Yonatan Belinkov, and David Bau. Fine-tuning enhances existing mechanisms: A case study on entity tracking, 2024. URL https://arxiv. org/abs/2402.14811. [188]

10.4 Model Steering or Editing

1. Nupur Kumari, Bingliang Zhang, Sheng-Yu Wang, Eli Shechtman, Richard Zhang, and Jun-Yan Zhu. Ablating concepts in text-to-image diffusion models, 2023. [126]
2. Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing

factual associations in gpt, 2023. [165]

3. Dana Arad, Hadas Orgad, and Yonatan Belinkov. Refact: Updating text-to-image models by editing the text encoder, 2023. [10]
4. Rohit Gandikota, Joanna Materzynska, Jaden Fiotto-Kaufman, and David Bau. Erasing concepts from diffusion models, 2023a. [78]
5. Rohit Gandikota, Hadas Orgad, Yonatan Belinkov, Joanna Materzynska, and David Bau. Unified concept editing in diffusion models, 2023b. [79]
6. Y. Gandelsman, A. A. Efros, and J. Steinhardt. Interpreting CLIP's image representation via textbased decomposition. In The Twelfth International Conference on Learning Representations, 2024. URL https://openreview.net/forum?id=5Ca9sSzuDp [77]
7. S. Cheng, B. Tian, Q. Liu, X. Chen, Y. Wang, H. Chen, and N. Zhang. Can We Edit Multimodal Large Language Models?, 2024. [46]
8. K. Meng, A. S. Sharma, A. Andonian, Y. Belinkov, and D. Bau. Mass-Editing Memory in a Transformer, 2023. [167]
9. C. Zhu, A. S. Rawat, M. Zaheer, S. Bhojanapalli, D. Li, F. X. Yu, and S. Kumar. Modifying memories in transformer models. CoRR, abs/2012.00363, 2020. URL https://arxiv.org/ abs/2012.00363. [269]
10. Shilin Lu, Zilan Wang, Leyang Li, Yanzhu Liu, and Adams Wai-Kin Kong. Mace: Mass concept erasure in diffusion models. arXiv preprint arXiv:2403.06135, 2024. [155]

Chapter 11: **Appendix**

11.1 Interpretation of Models Through Lens of Data

11.2 Running Times

In this section, we provide computational running times for (first-order) influence function estimations. We note that in models with a large number of parameters, the influence computation is relatively slow. However, even in large deep models, it is still faster than re-training the model for every training example. In our implementation, for a given test-point z_{test}, we first compute $c = H_{*}^{-1} \nabla \ell(h_{*}(z_{test}))$ once which is the most computationally expensive step. We then compute a vector dot product i.e. $c^T \nabla \ell(h_{*}(z_i)) \; \forall i \in [1, n]$. In Table 11.1, we provide the computational running times for estimating influence functions in different network architectures.

Figure 11.1: Top 5 influential points for the test point: 1479 (CIFAR-10). The model is a ResNet-18 trained with a weight-decay regularization; Only 3 out of the 5 points are semantically similar to the test-point with class "Bird".

Architecture	Influence Computation Time (MNIST)	Influence Computation Time (CIFAR-10)
Small CNN	141.13 ± 0.51	N/A
LeNet	162.6 ± 2.20	136.39 ± 3.16
VGG13	3886.23 ± 3.45	4416.54 ± 2.01
VGG14	4619.11 ± 5.08	4620.69 ± 6.11
ResNet-18	960.08 ± 4.67	910.58 ± 8.49
ResNet-50	4323.13 ± 8.26	3857.66 ± 21.6

Table 11.1: Computational running times for influence function across different architectures

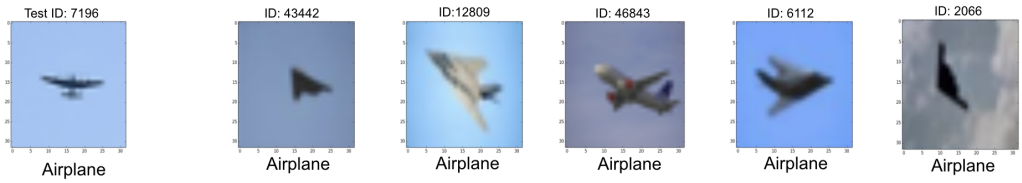

Figure 11.2: Top 5 influential points for the test point: 7196 (CIFAR-10). The model is a ResNet-18 trained with a weight-decay regularization; All the 5 training points are semantically similar to the test-point from the class "Airplane".

11.2.1 Faithfulness and Plausibility of Influence functions

The authors [106] primarily tackle the importance and trade-offs between plausibility (i.e. if the interpretations are convincing to humans) and faithfulness (i.e. how accurate an interpretation is to the "true reasoning process of the model") of existing interpretation methods. To the best of our knowledge such an analysis has not been done for influence functions. We observe that explanations from influence functions for deep networks are sometimes plausible and sometimes not. For instance, in Appendix Fig. 11.1, we observe that the selection of test-point with (class = bird) leads to training examples with (class = deer) amongst the top influential points. On the other hand, in Appendix Fig. 11.2, we observe many plausible explanations. Influence functions that work are faithful because

they answer the following question:"what would this model have done if certain data were excluded?". This class of questions, while not exhaustive, have special relevance because they are counterfactuals, which hold both intuitive appeal and for their special status in causal reasoning. However, we must be cautious because they may not be faithful when they incur approximation errors, as highlighted in our paper.

11.3 Automatically Designing Difficult Few-Shot Tasks

11.3.1 Support Set Extraction Algorithm

Steps For Solving the Projection Step

In this section, we provide details on how to solve the projection step in Eq. (4.3). The projection step is solved separately for each of the j^{th} class, where $j \in [1,N]$. $\hat{\mathbf{w}}_j$ is the selection weight vector for the j^{th} class obtained after a step of gradient ascent on Eq. (4.1). The dual form of Eq. (4.2) can be expressed via Lagrange multipliers as the following:

$$\bar{\mathbf{w}}_j = \lambda_j \geq 0 \min_{\mathbf{w}_j} \underbrace{\frac{1}{2}\|\mathbf{w}_j - \hat{\mathbf{w}}_j\|_2^2 + \lambda_j(\|\mathbf{w}_j\|_1 - k_j)}_{g(\lambda_j, \mathbf{w}_j)} \tag{11.1}$$

We solve Eq. (11.1) in two steps: (i) First, we solve $\min_{\mathbf{w}_j} g(\lambda_j, \mathbf{w}_j)$ via proximal operators; (ii) Then, we obtain the optimal values of the dual parameters λ_j.

KKT optimality conditions (due to stationarity) states that $\nabla_{\mathbf{w}_j} g(\lambda_j, \mathbf{w}_j) = 0$. However, note that $g(\lambda_j, \mathbf{w}_j)$ is a combination of a smooth function and a non-smooth function which can be solved by proximal operators. Considering $\mathbf{w}_j \in \mathbb{R}^n$, the KKT optimality condition can

be stated as the following:

$$\nabla_{\mathbf{w}_j}\frac{1}{2}\|\mathbf{w}_j-\hat{\mathbf{w}}_j\|^2+\nabla_{\mathbf{w}_j}\lambda_j(\|\mathbf{w}_j)\|_1-k_j)=0 \tag{11.2}$$

The value in the i^{th} index in $\mathbf{w}_j$, which is w_j^i can be obtained through:

$$\frac{1}{2}\frac{\partial(w_j^i-\hat{w_j}^i)^2}{\partial w_j^i}+\lambda_j\frac{\partial|w_j^i|}{\partial w_j^i}=0 \tag{11.3}$$

If $w_j^i>0$, then the derivative in Eq. (11.3) is $w_j^i-\hat{w_j^i}+\lambda_j$. Therefore Eq. (11.3) can be expressed as: $\bar{w_j^i}=\hat{w_j}^i-\lambda_j$, which holds true for $\hat{w_j}^i>\lambda_j$. Similarly, when $w_j^i<0$, $\bar{w_j^i}=\hat{w}_j^i+\lambda_j$ is the minimizer. For $\hat{w}_j^i\in[-\lambda_j,\lambda_j]$, the minimizer is at the only point of differentiability for which $w_j^i=0$. This operation is called soft-thresholding and can be expressed as:

$$\text{Prox}_{\lambda^*\|.\|_1}(\hat{w}_j^i)=\text{sign}(\hat{w}_j^i)\max(|w_j^i|-\lambda_j^*,0) \tag{11.4}$$

Thus, $\bar{\mathbf{w}}_j=\text{Prox}_{\lambda^*\|.\|_1}(\hat{\mathbf{w}}_j)=\text{Prox}_{\lambda^*\|.\|_1}(\hat{w}_j^1)\oplus....\oplus\text{Prox}_{\lambda^*\|.\|_1}(\hat{w}_j^n)$. The next step is to compute the value of the dual parameter λ_j^*. We then compute the derivative $g'(\lambda_j,\bar{\mathbf{w}}_j)$ as the following:

$$g'(\lambda_j,\bar{\mathbf{w}}_j)=\|\text{Prox}_{\lambda^*\|.\|_1}(\hat{\mathbf{w}}_j)\|_1-k_j \tag{11.5}$$

$$=\sum_{i=1}^{n}(|\hat{w}_j^i|-\lambda_j)_+-k_j \tag{11.6}$$

We solve Eq. (11.6) by the root finding method in [52], since the optimal $\lambda_j^*\in[0,\|\hat{\mathbf{w}}_j\|_\infty]$. The upper bound $\|\hat{\mathbf{w}}_j\|_\infty$ ensures that $g'(\lambda_j,\bar{\mathbf{w}}_j)$ does not become negative.

Hyperparameters of the Framework

Empirically, we perform a grid-search for the learning rate $\alpha \in [0.01, 500]$ with a step size of 10. We specifically find that a high learning rate with *only* one gradient ascent and projection step suffices to obtain difficult tasks. In our experiments, we use a learning rate $\alpha = 200$, as we find this value to result in the extraction of the most difficult tasks.

11.4 Mechanistically Understanding and Editing Text-to-Image Models

11.4.1 Probe Dataset Design Details

In this section, we provide detailed descriptions of the probe dataset $\mathscr{P}$ which is used for causal tracing for both the UNet and the text-encoder. We primarily focus on four visual attributes : *style, color, objects* and *action*. In addition, we also use the causal tracing framework adapted for text-to-image diffusion models to analyse the *viewpoint* and *count* attribute. The main reason for focusing on *style, color, objects* and *action* is the fact that generations from current text-to-image models have a strong fidelity to these attributes, whereas the generations corresponding to *viewpoint* or *count* are often error-prone. We generate probe captions for each of the attribute in the following way:

- ***Objects***. We select a list of 24 objects and 7 locations (e.g., *beach, forest, city, kitchen, mountain, park, room*) to create a set of 168 unique captions. The objects are : *{ 'dog', 'cat', 'bicycle', 'oven', 'tv', 'bowl', 'banana', 'bottle', 'cup', 'fork', 'knife', 'apple', 'sandwich', 'pizza', 'bed', 'tv', 'laptop', 'microwave', 'book', 'clock', 'vase', 'toothbrush', 'donut', 'handbag' }* . We then use the template: *'A photo of <object> in <location>.'* to generate multiple captions to probe the text-to-image model. These objects are selected from the list of objects present in MS-COCO.

- ***Style***. We select the list of 80 unique objects from MS-COCO and combine it with an artistic style from : *{monet, pablo picasso, salvador dali, van gogh, baroque, leonardo da vinci, michelangelo}* . The template used is: *'A <object> in the style of <artistic-style>'*. In total, using this template we generate 560 captions.
- ***Color***. We select the list of 80 unique objects from MS-COCO and combine it with a color from *{ blue, red, yellow, brown, black, pink, green}*. We then use the template: *'A <color> <object>'* to generate 560 unique captions to probe the text-to-image model.
- ***Action***. We first choose certain actions such as *eating, running, sitting, sprinting* and ask ChatGPT[1] to list a set of animals who can perform these actions. From this list, we choose a set of 14 animals: *{ bear, cat, deer, dog, giraffe, goat, horse, lion, monkey, mouse, ostrich, sheep, tiger, wolf}*. In total we use the template: *'An <animal><action>'* to create a set of 56 unique captions for probing.
- ***Viewpoint***. For viewpoint, we use the same set of 24 objects from the *Objects* attribute and combine it with a viewpoint selected from *{front, back, top, bottom}* to create 96 captions in the template of: *'A <object> from the <viewpoint>'*.
- ***Count***. We use the same 24 objects from *Objects* attribute and introduce a count before the object in the caption. We specifically use the following template to generate captions: *'A photo of <count> objects in a room.'*, where we keep the location fixed. We select a count from {2,4,6,8} to create 96 unique captions in total.

[1]We use version 3.5 for ChatGPT.

	Description of Probe Dataset for Causal Tracing			
Attribute	Description	Example 1	Example 2	Example 3
Objects	Prompt containing an object in a location	photo of a *vase* in a room	a photo of a *car* in a desert	a photo of a *house* in a forest
Style	An object drawn in a particular artistic style	airplane in the style of *van gogh*	town in the style of *monet*	bicycle in the style of *baroque*
Color	An object in a particular color	a *blue* car	a *black* vase	a *pink* bag
Action	An animal in a particular action	A giraffe *eating*	A tiger *running*	A cat *standing*
Viewpoint	An object in a particular viewpoint	A sofa from the *back*	A car from the *front*	A bus from the *side*
Count	Number of objects in a location	There are *10* cars on the road	*5 bags* in the room	*6* laptops on a table

Table 11.2: **Examples from the Probe Dataset Used For Causal Tracing.** The attributes in the captions are marked in *italics*.

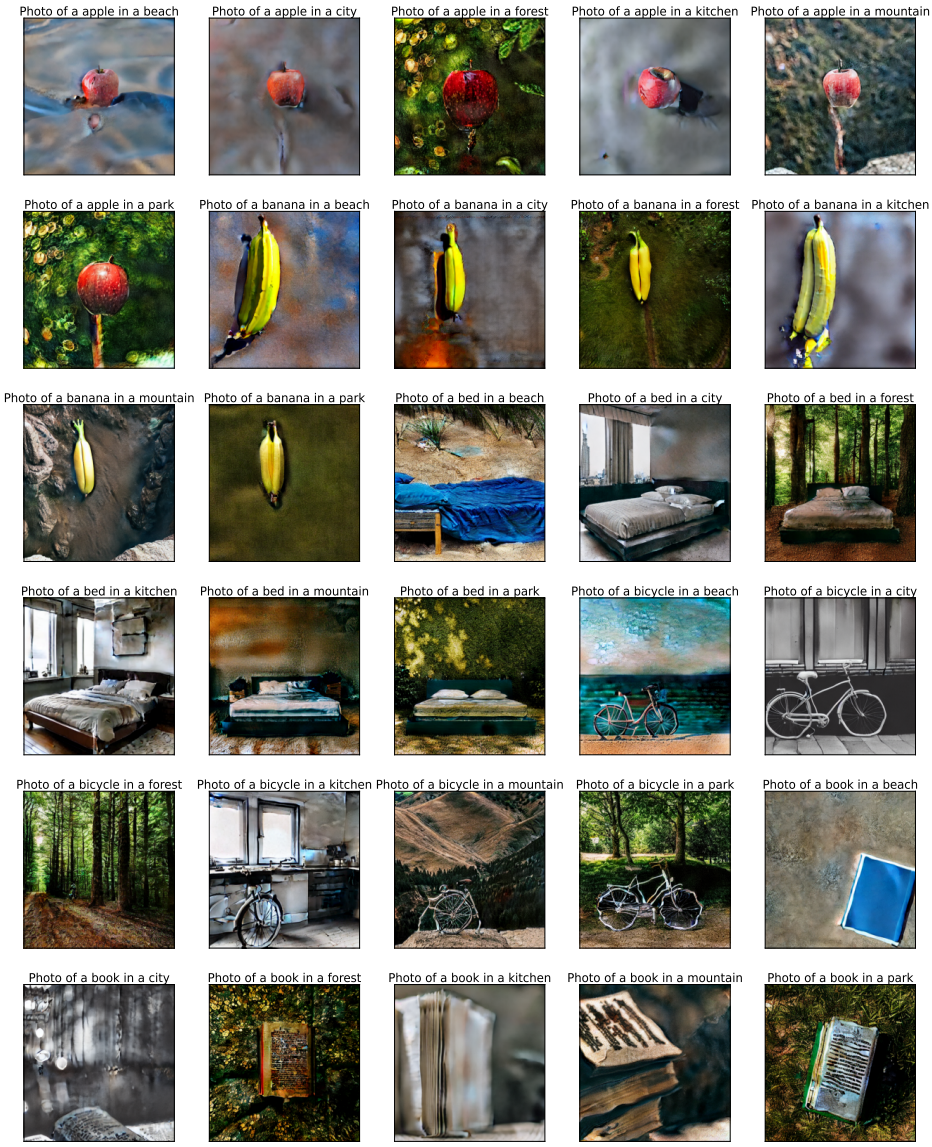

Figure 11.3: **Causal State: down-1-resnet-1.** We find that restoring the down-1-resnet-1 block in the UNet leads to generation of images with strong fidelity to the original caption. Note: With the captions for 'Photo of a apple in a beach', 'Photo of a apple in a city' and 'Photo of a apple in a kitchen' – with the given seed we find that the original background is not succinctly generated in the original image. Therefore the correct restored generations do not contain the correct background.

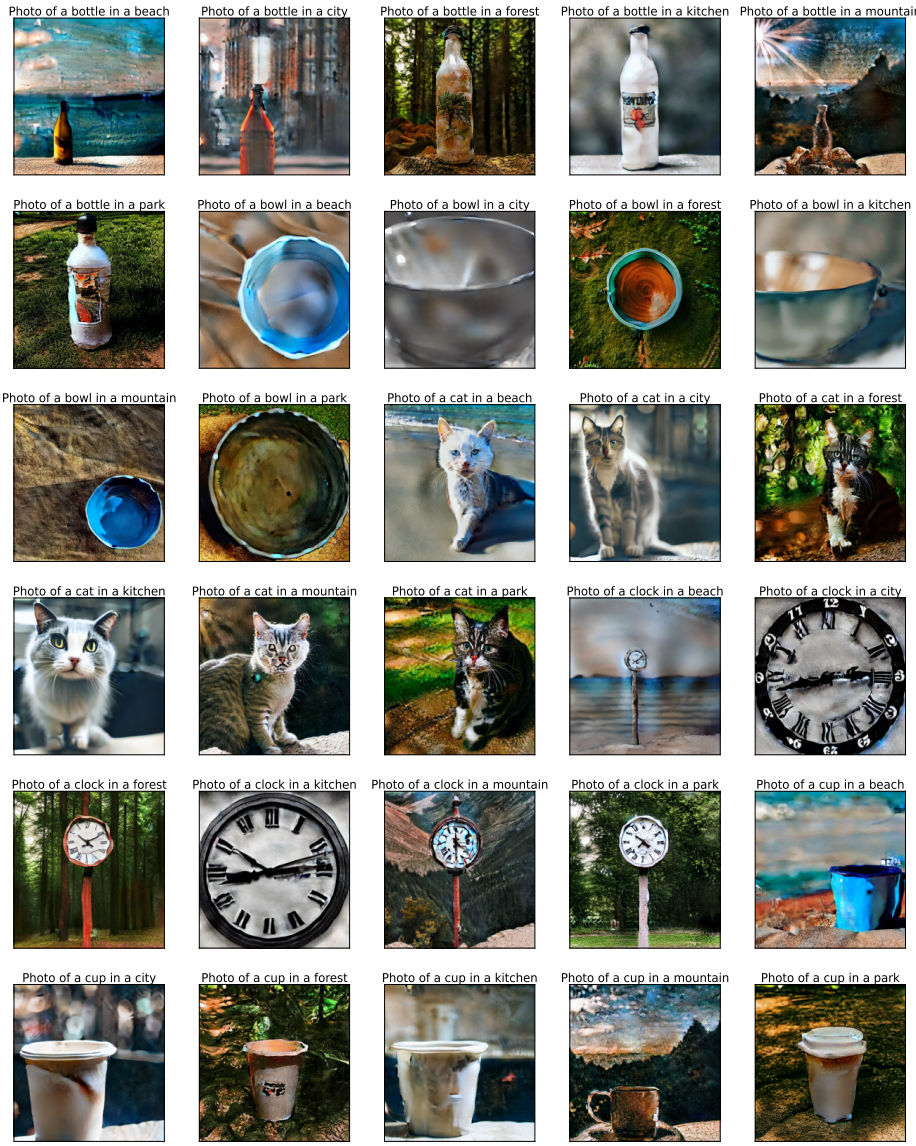

Figure 11.4: **Causal State: down-1-resnet-1.** We find that restoring the down-1-resnet-1 block in the UNet leads to generation of images with strong fidelity to the original caption.

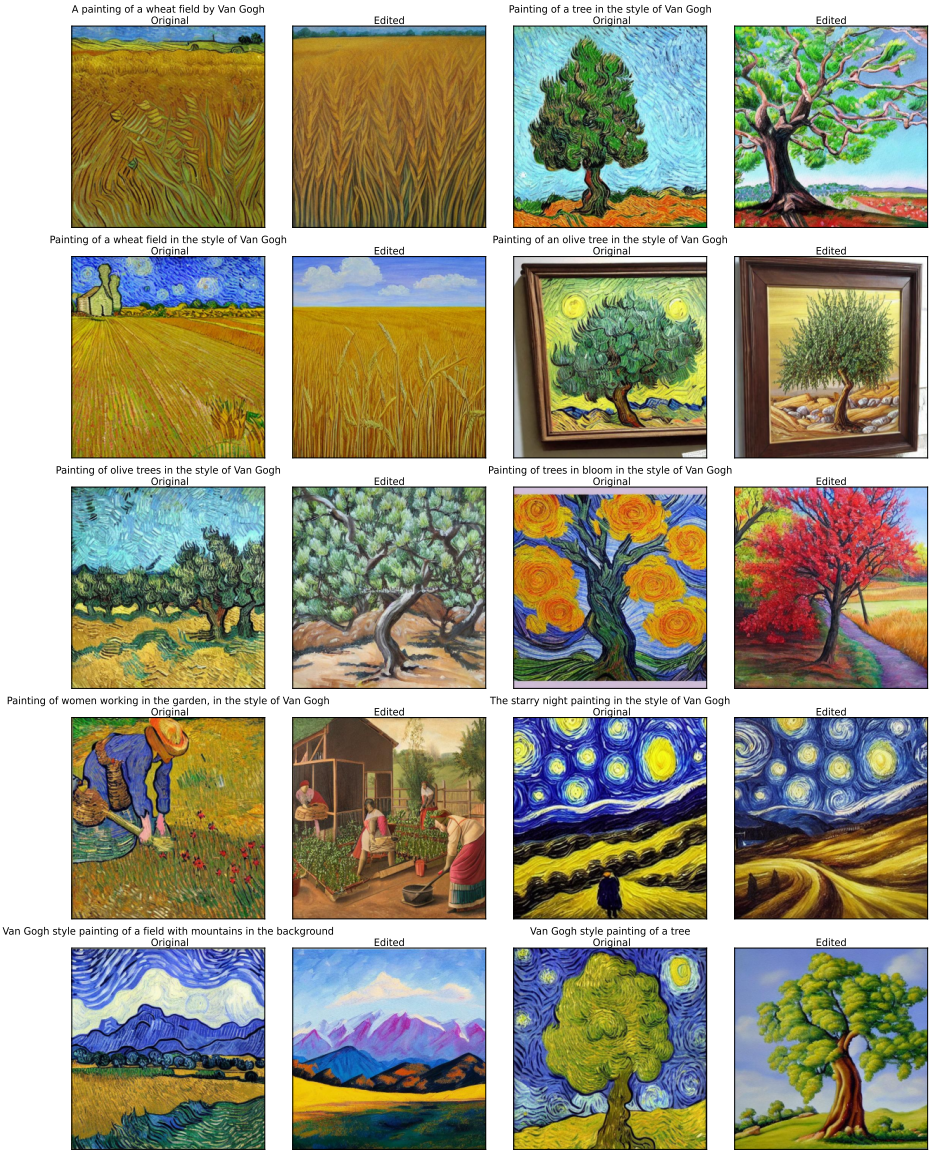

Figure 11.5: **Single-Concept Ablated Model: Generations with different *Van Gogh* Prompts.**

Figure 11.6: **SDXL Edits for "style"**. We show successful model editing on the layers identified by LOCOGEN. In case of the images generated by the edited model, we can observe that the trademarked brushstrokes of the artist *Van Gogh* are missing. For some of the images from the edited model (e.g., Trees in the style of Van Gogh), we even find that the patterns in the sky which is another trademark *Van Gogh* signature have them deleted.

11.5 Mechanistically Understanding and Editing Multimodal Language Models

11.5.1 *VQA-Constraints* Details

In Sec. 6.2.1, we introduce the constraint framework for the task of factual VQA. In particular, there are two types of constraints: (i) Visual Constraint: A set of words in a question which refer to an entity in an image; (ii) Text Constraint: A set of words in a question which together with the visual constraint are responsible for answering a question. In our *VQA-Constraints* dataset there are two types of questions: (i) Single Constraint Questions: These questions comprise of only the visual constraint; (ii) Multi-Constraint Questions: These questions comprise of both the visual as well as the text constraint. In our paper, we primarily focus on the Single Constraint questions, therefore a majority of questions in our test bed of *VQA-Constraints* consist of only the visual constraints. Our VQA

dataset *VQA-Constraints* which is annotated with constraints comprise of the following three parts:

OK-VQA: We annotate the questions in the test-set of OK-VQA with the constraints. In total, this part consists of 5k questions. We use the same images as those used in the original OK-VQA test-set.

Multimodal Movies: We use the text-only WikiMovies dataset from [263] and download images using the Bing API using the name of the directors. Our team filters and validates that the downloaded images are of the director itself and there's no noise in the downloaded images. We use a set of 1.5k questions in the final dataset.

Multimodal Known: For understanding knowledge storage in language models, [165] use a probe dataset (known.json) consisting of 1.2k questions. We first replace the subject in the question with a constraint and then download images from Bing API. Our team validates that the downloaded images are correct and uses it to create the Multimodal Known dataset.

Multi-Constraint Questions: The multi-constraint questions in *VQA-Constraints* are created from OK-VQA and Multimodal Movies. In particular, there are 150 multi-constraint questions for OK-VQA and 500 multi-constraint questions from Multimodal Movies. Given that the primary focus of our paper is on Single Constraint questions, this set of Multi-Constraint questions is relatively small.

In total, *VQA-Constraints* consist of 9.7K VQA questions with Single Constraints and 650 Multi-Constraint VQA questions which we use for interpretability analysis.

Annotating and Validating the Visual Constraints. To automatically annotate the constraints in the visual questions from *VQA-Constraints*, we use a strong language model such as GPT-4. In particular, we annotate only the Multimodal Known and OK-VQA sub-parts of

Description of Questions in VQA-Constraints				
Dataset	Description	Example 1	Example 2	Example 3
OK-VQA	Single Constraint	What sport can you use *this* for?	Why might someone go to *this place*?	What flavor is *this pastry*?
OK-VQA	Single Constraint	Which airlines is this insignia?	What fruits come from *these trees*?	What is *this animal* mostly known for?
Multimodal Known	Single Constraint	*This person* is a citizen of?	*This newspaper* is written in?	The capital of *this city* is in?
Multimodal Known	Single Constraint	*This venue* is owned by?	*This building* is located in which city?	*This show* debuted on?
Multimodal Movies	Single Constraint	Name a movie directed by *this director*?	Name a movie directed by *this director*?	Name a movie directed by *this director*?
Multimodal Movies	Single Constraint	Name a movie directed by *this director*?	Name a movie directed by *this director*?	Name a movie directed by *this director*?
OK-VQA	Multi-Constraint	Which is the most famous type of *this food* in *India*	How does the population of *this city* compare to *Mumbai*?"	What is the relationship between *this person* and *Venus Williams*?
Multimodal Movies	Multi-Constraint	Name a movie directed by *this director* in year *2006*?	Name a movie directed by *this director* in year *2010*?	Name a movie directed by *this director* in year *1994*?

Table 11.3: **Description of the different VQA questions in** *VQA-Constraints.* The constraints are marked in *italic* in the columns.

the *VQA-Constraints* dataset. First from each dataset, we annotate 100 examples which we use as in-context examples to the language model to annotate new questions. Given these annotations, our team then verified if the annotation is correct and if incorrect, modified the constraint annotation to make it correct. In total, the visual constraints are annotated for 9.7k VQA questions.

11.5.2 Standard Causal Tracing Does Not Recover Causal States

In Fig. 11.7, we find that the standard causal tracing approach, which adds Gaussian noise to the span of visual tokens and the constraint in the text does not recover any causal states, even with a large window size of 10. This is true for recovering both the MLP as well as the self-attention causal layers.

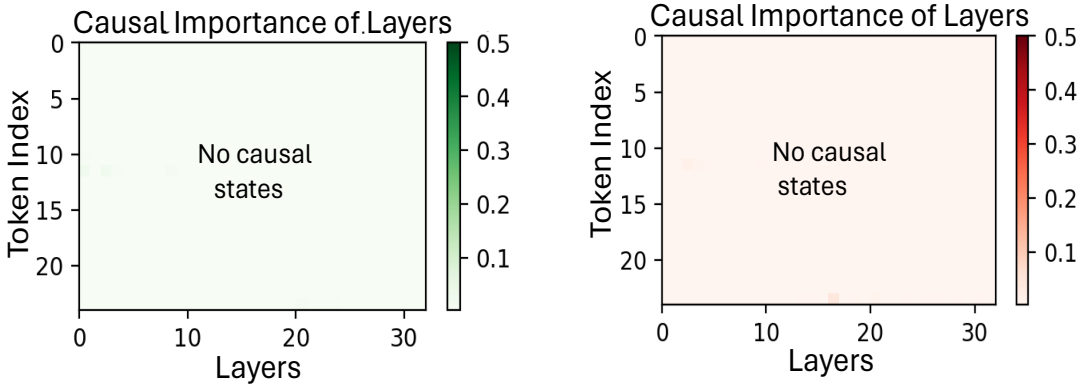

Figure 11.7: **Standard Causal Tracing**: We find that using the standard causal tracing procedure from [165] is not able to recover causal states. Averaged across all the datapoints in Multimodal Known and Multimodal Movies from *VQA-Constraints*. A window size of 10 is used.

11.6 Mechanistically Understanding and Unlocking Zero-Shot Capabilities in Vision Transformers

11.6.1 Scoring Function

Algorithm 5 Scoring function for attributing properties to components

Input: Z, the image representation output by the model over n images with dimension d (shape: $n \times d$); C, the contribution of a particular component of interest (shape: $n \times d$); B, the set of k feature vectors that represent a given feature (shape: $k \times d$)

Output: A score that signifies the importance of the component for the given feature

function COMPATTRIBUTE(C, Z, B)
  $B \leftarrow$ orthogonalize(B)
  $s_Z \leftarrow ZB^\top$
  $s_C \leftarrow CB^\top$
  $r \leftarrow$ correlation_coefficient(s_Z, s_C, dim=0)
  return mean(r)
end function

Model name	Color	Texture	Animal	Person	Location	Pattern	Shape
DeiT	0.679	0.563	0.774	0.596	0.818	0.597	0.764
DINO	0.663	0.657	0.781	0.742	0.833	0.680	0.706
DINOv2	0.751	0.614	0.875	0.714	0.857	0.597	0.510
SWIN	0.795	0.720	0.904	0.780	0.912	0.760	0.739
MaxVit	0.872	0.832	0.911	0.828	0.901	0.803	0.797

Table 11.4: Spearman rank correlation for various common properties

11.6.2 Proof of Theorem 1

Proof. From the first condition on **intra-component rank ordering**, for any two vectors $\mathbf{u}, \mathbf{v}$ and a linear map f_i, if $\|\mathbf{u}\| \leq \|\mathbf{v}\|$ then $\|f_i(\mathbf{u})\| \leq \|f_i(\mathbf{v})\|$. We first show that f_i is a scalar multiple of an isometry.

If $\|\mathbf{u}\| = \|\mathbf{v}\| \neq 0$, then both $\|\mathbf{u}\| \leq \|\mathbf{v}\|$ and $\|\mathbf{v}\| \leq \|\mathbf{u}\|$. This implies that $\|f_i(\mathbf{u})\| \leq \|f_i(\mathbf{v})\|$ and $\|f_i(\mathbf{v})\| \leq \|f_i(\mathbf{u})\|$. Therefore, $\|f_i(\mathbf{u})\| = \|f_i\mathbf{v}\|$, when $\|\mathbf{u}\| = \|\mathbf{v}\|$. Given the input space

of the transformation as U, we choose a unit vector $\mathbf{u}_{\text{unit}} \in U$. Let's assume $\|f_i(\mathbf{u}_{\text{unit}})\| = c$. With the above result, we can use the following equality $\|\frac{\mathbf{u}}{\|\mathbf{u}\|}\| = \|\mathbf{u}_{\text{unit}}\|$ to obtain the following:

$$\left\| \frac{f_i(\mathbf{u})}{\|\mathbf{u}\|} \right\| = \left\| f_i\left(\frac{\mathbf{u}}{\|\mathbf{u}\|}\right) \right\| = \|f_i(\mathbf{u}_{\text{unit}})\| = c, \tag{11.7}$$

Therefore:

$$\|f_i(\mathbf{u})\| = c\|\mathbf{u}\| \tag{11.8}$$

Thus, the linear transformation f_i is a scalar multiple of an isometry. Now consider two linear maps f_i and f_j such that $\|f_i(\mathbf{u})\| = c_i\|\mathbf{u}\|$ and $\|f_j\mathbf{u}\| = c_j\|\mathbf{u}\|$. From the second condition on **inter-component rank ordering**, for any two vectors $\mathbf{u}, \mathbf{v}$ and linear maps f_i, f_j, if $\|\mathbf{u}\| \leq \|\mathbf{v}\|$ then $\|f_i(\mathbf{u})\| \leq \|f_j(\mathbf{v})\|$. This implies that if $\mathbf{u} = \mathbf{v}$, $\|f_i(\mathbf{u})\| = \|f_j(\mathbf{u})\|$. However, this can only happen when $\|f_i(\mathbf{u})\| = c\|\mathbf{u}\|$ for some constant c for all $f_i \; \forall i$.

With this, let's denote $\frac{f_i}{c}$ as an isometry. One of the general property of isometries are that they preserve the inner product between two vectors $\mathbf{u}$ and $\mathbf{v}$. First we prove that isometries preserve the inner product, which we will then use to prove the orthogonality of $\frac{f_i}{c}$. Given two vectors $\mathbf{u}$ and $\mathbf{v}$, their inner product can be expressed as the following:

$$\mathbf{u}^T\mathbf{v} = \frac{1}{4}(\|\mathbf{u}+\mathbf{v}\|^2 + \|\mathbf{u}-\mathbf{v}\|^2) \tag{11.9}$$

An isometry by definition preserves the norm of the vectors i.e. $\|f_i(\mathbf{u})\| = \|\mathbf{u}\|$ and $\|f_i(\mathbf{v})\| = \|\mathbf{v}\|$. Due to this property, we can express the following relations:

$$\|f_i(\mathbf{u}+\mathbf{v})\| = \|\mathbf{u}+\mathbf{v}\|, \tag{11.10}$$

and

$$\|f_i(\mathbf{u}-\mathbf{v})\| = \|\mathbf{u}-\mathbf{v}\|, \tag{11.11}$$

We can express $f_i(\mathbf{u})^T f_i(\mathbf{v})$ as the reduction from Eq.(11.9):

$$f_i(\mathbf{u})^T f_i(\mathbf{v}) = \frac{1}{4}(\|f_i(\mathbf{u})+f_i(\mathbf{v})\|^2+\|f_i(\mathbf{u})-f_i(\mathbf{v})\|^2), \tag{11.12}$$

$$f_i(\mathbf{u})^T f_i(\mathbf{v}) = \frac{1}{4}(\|f_i(\mathbf{u}+\mathbf{u})\|^2+\|f_i(\mathbf{u}-\mathbf{v})\|^2), \tag{11.13}$$

Next we substitute the relations from Eq.(11.10) and Eq.(11.11) to Eq.(11.13) to obtain the following inner product preservation property:

$$f_i(\mathbf{u})^T f_i(\mathbf{v}) = \frac{1}{4}(\|\mathbf{u}+\mathbf{v}\|^2+\|\mathbf{u}-\mathbf{v}\|^2) = \mathbf{u}^T\mathbf{v} \tag{11.14}$$

Next we use the inner product preservation property to prove the orthogonality of $\frac{f_i}{c}$ as follows:

$$f_i(\mathbf{u})^\top f_i(\mathbf{v}) = c^2\mathbf{u}^\top\mathbf{v}, \tag{11.15}$$

$$\mathbf{u}^\top\left(\frac{1}{c^2}f_i^\top f_i - I\right)\mathbf{v} = 0, \tag{11.16}$$

From 11.16, we can infer the orthogonality of $\frac{f_i}{c}$ which leads to the following result:

$$f_i^\top f_i = c^2 I = kI, \tag{11.17}$$

□

11.7 A Mechanistic Circuit for Extractive Question-Answering

11.7.1 Note on Second-order Circuit Components

In Sec.(8.3), we identify circuit components at hierarchy 0 that have the most significant direct impact on the final logit. In this case the target node in the circuit graph is the logit and the source nodes are all the different attention layers, MLPs and attention heads in the extracted circuit. In our experiments, we also set the extracted circuit components from hierarchy 0 as the target node and then extract source nodes in the circuit graph. We perform this operation at the last residual stream position. Overall, we obtain a set of components which have a high direct effect on the extracted components from hierarchy 0 (with a metric score of 0.71) for Llama-3-8B. However, on investigating the components further, we did not find any specific utility of *data attribution* or *model steering* using them. Overall, an in-depth study of the second-order components in the causal graph for extractive QA will be addressed in a future work.

11.7.2 On Modifying Circuit Components

In Sec.(8.3.3), we discuss the effect of scaling the attention heads from the context faithfulness circuit in the language model when it answers from the parametric memory. In particular, we find that upweighting the maximum attention value from these attention heads onto the context steers the language model towards answering from the context instead of the parametric memory.

Up-weighting the attention values. We multiple a scalar value β to the maximum attention value in the context before the softmax normalization operation. In our implementation, we perform this scaling operation across **all the attention heads in the top 3 attention layers**

in the circuit. We set $\beta = 10$ in our experiments, for the best steering result.

Ablating the MLPs. We set the output of the top MLPs from the memory-faithulness circuit to be zero. In particular, we set the output of the projection layer in the MLP block to be zero, but make sure that the output of the other blocks are not changed due to this modification, by setting them to their original configuration. This ensures, that only the direct connection from the MLP to the final logit is ablated.

11.7.3 Extracted Circuit Components Across Language Models

11.7.4 Vicuna

Context Faithfulness

Attention Layers. [24, 20, 18, 28, 31, 22, 19, 29, 17]

Attention Heads. [[24, 8], [18, 30], [31, 24], [20, 1], [22, 30], [24, 15], [19, 4], [28, 7], [31, 27], [28, 14], [29, 10], [17, 11], [31, 16], [18, 10]]

MLPs. [31, 24, 21, 14, 18, 11, 9, 12, 8, 1, 0, 2, 7, 3, 16, 6, 5, 4, 15, 10, 13, 17, 19, 27, 29, 23, 30, 26, 20, 22, 28, 25] (Sorted order)

Memory Faithfulness

Attention Layers. [20, 24, 16, 31, 26, 28, 30, 29, 15, 22, 12, 13, 19]

Attention Heads. [[31, 27], [24, 14], [19, 8], [28, 7], [20, 14], [20, 18], [16, 10], [21, 15], [26, 23], [30, 12], [15, 10], [31, 25], [17, 25], [16, 20], [18, 9], [24, 24], [14, 28], [18, 26], [29, 15], [14, 5], [26, 14], [16, 5], [18, 11], [22, 10], [22, 17], [16, 31], [12, 30], [31, 16], [31, 26], [29, 9]]

MLPs. [22, 20, 23, 21, 31, 19, 30, 29, 14, 18]

11.7.5 Llama-3-8B

Context Faithfulness

Attention Layers. [27, 23, 31, 24, 25, 29, 21, 30]

Attention Heads. [[27, 20], [23, 27], [31, 7], [17, 24], [25, 12], [31, 20], [24, 27], [27, 6], [26, 13], [16, 1], [31, 6], [29, 31], [31, 3], [30, 12]]

MLPs. [31, 28, 26, 25]

Memory Faithfulness

Attention Layers. [31, 24, 26, 9, 19, 17, 23, 8, 16, 28, 3, 1, 6, 5, 0, 4, 25, 2, 27, 21, 22, 7, 12, 20, 13, 30, 11, 18, 14, 29, 10, 15]

Attention Heads. [[31, 7], [24, 3], [31, 14], [30, 24], [17, 24], [15, 18], [31, 1], [31, 3], [24, 27], [29, 8], [17, 27], [17, 23], [26, 3], [20, 14], [31, 6], [14, 22], [31, 25], [18, 29], [22, 14], [16, 2], [13, 23], [28, 0], [16, 0], [16, 30], [17, 5], [19, 3], [31, 27], [20, 27], [30, 2], [14, 1], [21, 3], [27, 6], [19, 14], [21, 10], [14, 4], [29, 22], [29, 9], [14, 24], [16, 5], [21, 26], [14, 28], [16, 25], [16, 13], [19, 20], [19, 25], [15, 11], [21, 1], [29, 11], [17, 6], [26, 12], [15, 24], [11, 5], [13, 17], [15, 20], [29, 23], [30, 26], [15, 7], [13, 9], [13, 5], [16, 24], [17, 4], [27, 21], [27, 30], [15, 8], [9, 0], [14, 13], [16, 19], [14, 14], [9, 29], [13, 21], [27, 23], [11, 28], [9, 5], [20, 3], [28, 11], [12, 20], [25, 1], [13, 3], [16, 17], [12, 21], [31, 31], [22, 29], [29, 17]]

MLPs. [22, 21, 20, 23, 25, 24, 19,]

11.7.6 Phi-3

Context Faithfulness

Attention Layers. [29, 21, 31, 28, 25, 20, 23, 11]

Attention Heads. [[29, 31], [20, 1], [31, 4], [23, 7], [19, 14], [23, 23], [25, 6], [20, 21], [25, 18], [21, 21], [21, 16], [28, 28], [25, 9], [21, 22]]

MLPs. [31, 30, 27, 19, 14, 21, 15, 9, 6, 11, 7, 4, 3, 1, 5, 0, 8, 2, 10, 16, 13, 23, 12, 18, 17, 20, 28, 26, 22, 24, 25, 29]

Memory Faithfulness

Attention Layers. [23, 31, 20, 22, 19, 29, 21, 24, 18, 16, 25, 12]

Attention Heads. [[23, 4], [31, 4], [29, 30], [31, 17], [19, 20], [30, 1], [19, 13], [20, 5], [22, 29], [25, 23], [22, 15], [28, 7], [20, 26], [9, 17], [21, 16], [24, 31], [24, 12], [20, 25], [22, 1], [23, 31], [21, 21], [20, 4], [19, 27], [31, 9], [12, 10], [20, 12], [21, 2], [26, 21], [21, 6], [18, 12], [18, 10], [13, 21], [16, 30], [13, 11], [13, 25], [15, 29], [25, 2], [21, 5], [25, 9], [29, 20], [16, 15], [18, 25], [29, 17], [4, 29], [29, 26], [23, 29], [24, 4], [16, 25], [22, 18], [16, 9], [30, 24], [18, 1], [18, 24], [17, 25], [3, 10]]

MLPs. [23, 24, 22, 25, 21]

11.7.7 Do we need a larger probe dataset?

We initially tested circuit extraction by using a smaller dataset of size 200. In particular, we extract the context-faithfulness circuit for Llama-3-8B. We find the following components:

Attention Layers. [27, 23, 31, 24, 29, 25, 21, 30]

Attention Heads. [[27, 20], [23, 27], [31, 7], [17, 24], [31, 20], [25, 12], [24, 27], [27, 6], [26, 13], [16, 1], [29, 31], [31, 6], , [31, 3], [30, 12]]

MLPs. [31, 28, 26, 25]

We find the sets of components in the circuit to be similar (except a couple of components get reordered) to the one extracted using 1000 examples. This validates that a relatively smaller size of probe dataset can also be used towards finding a circuit for extractive QA. We also note that [188] use a similar smaller size probe dataset to find a circuit for entity tracking.

11.7.8 Probe Dataset Details

As shown in Sec.(8.3.1), the probe dataset consists of two partitions $\mathscr{D}_{\text{copy}}$ and $\mathscr{D}_{\text{memory}}$ which are used to elicit the context-faithfulness circuit and the memory-faithfulness circuit respectively. Below we provide a few qualitative examples.

Example 1

Subject. Vinson Massif

Question. Where is Vinson Massif located?

Original Answer. Antarctica

Context for Copy Faithfulness. Vinson Massif is the highest peak in the Sentinel Range of the Ellsworth Mountains, towering at an elevation of 4,892 meters (16,050 feet). It is positioned in one of the most remote and challenging environments on Earth, attracting

climbers and adventurers from around the globe. First summited in 1966 by an American team, Vinson Massif is a sought-after destination for mountaineers aiming to complete the Seven Summits, the tallest peaks on each of the seven continents. Due to its extreme location and harsh weather conditions, expeditions to Vinson Massif require thorough preparation and careful logistical planning. The massif stands as the pinnacle of its continent, and for those who successfully reach its summit, it provides a profound sense of achievement and magnificent views over the surrounding icy landscape. Located in **Africa**, it is a testament to human endurance and the allure of pristine, untamed wilderness.

Context for Memory Faithfulness. Vinson Massif is the highest peak in the Sentinel Range of the Ellsworth Mountains, towering at an elevation of 4,892 meters (16,050 feet). It is positioned in one of the most remote and challenging environments on Earth, attracting climbers and adventurers from around the globe. First summited in 1966 by an American team, Vinson Massif is a sought-after destination for mountaineers aiming to complete the Seven Summits, the tallest peaks on each of the seven continents. Due to its extreme location and harsh weather conditions, expeditions to Vinson Massif require thorough preparation and careful logistical planning. The massif stands as the pinnacle of its continent, and for those who successfully reach its summit, it provides a profound sense of achievement and magnificent views over the surrounding icy landscape. Located in —, it is a testament to human endurance and the allure of pristine, untamed wilderness.

Example 2

Subject. Beats Music

Question. Who owns Beats Music?

Original Answer. Apple

Context for Copy Faithfulness. Beats Music, a subscription-based online music streaming service, was acquired by **Netflix** in 2014 for 3 billion.

Context for Memory Faithfulness. Beats Music, a subscription-based online music streaming service, was acquired by — in 2014 for 3 billion.

11.7.9 Data Attribution Evaluation Dataset Descriptions

- *Synthetic 1*: Consists of the probe dataset $\mathscr{D}$ where the context is the one generated by Llama-3-70B.
- *Synthetic 2*: Consists of the probe dataset $\mathscr{D}$ where the context is perturbed such that the original answer token is replaced with a closely related answer token.
- *NQ-Swap 1*: NQ-Swap dataset [153] where the original context is used.
- *NQ-Swap 2*: NQ-Swap dataset [153] where the original context is perturbed such that the original answer token is replaced with another token.
- *Natural-Questions*: A subset of Natural-Questions [127] where the ground-truth answers are short. In total, there are 13.9k questions.
- *Single-Hop HotPotQA*: Consists of questions from HotPotQA [255] with zero-hop or single-hop extractive QA questions.

11.7.10 Validating Long Extractive Answer Generations

Extractive QA datasets such as *HotpotQA*, *NaturalQuestions* and *NQ-Swap* are particularly concerned with entities in the answer which consist of a few relevant tokens in length. Even if the generated answer from the language model is long, the attributions need to point to the relevant span in the context which consist of the entity. Another challenging setting is the case where the language model needs to generate an answer comprising of an entity which itself can be long, for example comprising of multiple sentences. We investigate two

experimental settings in this respect for the following datasets: (i) *CNN-Dailymail*, where the language model is prompted to generate an extractive summary. This extracted summary is itself the relevant entity in the generated answer. (ii) *NQ-Long*, where the language model is prompted to generate an answer with exact sentences from the context (rather than only the entity). We note that in *NQ-Long*, the ground-truth answer consists of multiple sentences extracted from the context.

To evaluate the quality of attributions, we measure the relative change in the log probability of the responses when the original context is used vs. the original context is modified to remove the attributions (obtained from ATTNATTRIB). A higher relative change in the log probabilities indicates the faithfulness of the attributions. In particular, given the language model g_ϕ, the original context C_{orig} and the ablated context where the attributed text has been removed as C_{ablated}, we define the relative change in log probability of a response R as:

$$\text{Rel-Score}(g_\phi, C_{\text{orig}}, C_{\text{ablated}}, R) = \left| \frac{\log\left(p_{g_\phi}(R)|C_{\text{orig}}\right) - \log\left(p_{g_\phi}(R)|C_{\text{ablated}}\right)}{\log\left(p_{g_\phi}(R)|C_{\text{ablated}}\right)} \right| \quad (11.18)$$

11.7.11 Results on CNN-Dailymail

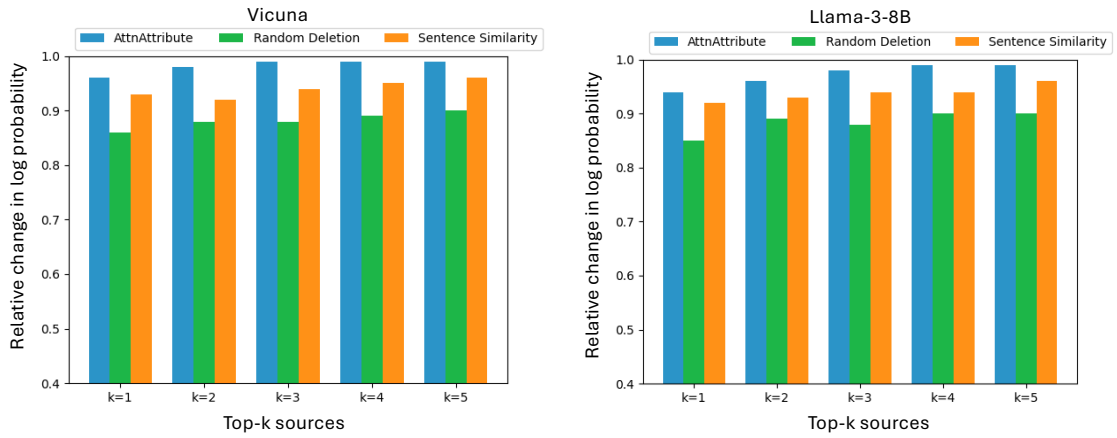

Figure 11.8: **Removing the attributions obtained with ATTNATTRIB from the context leads to a large relative change in the log probability of the responses.** We measure the relative change in the log probabilities of the original response (with the original context and context where the attributions are removed). We use 1000 examples from the CNN-Dailymail dataset. For both Vicuna and Llama-3-8B, we find a large relative change in the log probabilities of the responses, highlighting that the attributions from ATTNATTRIB are reliable.

11.7.12 Results on NQ-Long

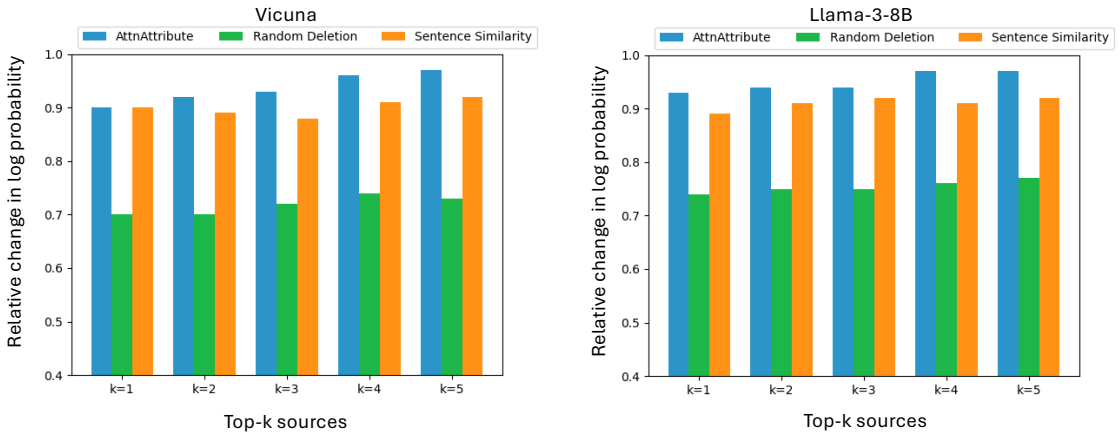

Figure 11.9: **Removing the attributions obtained with ATTNATTRIB from the context leads to a large relative change in the log probability of the responses.** We measure the relative change in the log probabilities of the original response (with the original context and context where the attributions are removed). We use 1000 examples from the NQ-Long dataset. For both Vicuna and Llama-3-8B, we find a large relative change in the log probabilities of the responses, highlighting that the attributions from ATTNATTRIB are reliable.

11.7.13 Circuit Components and Data Attribution in Llama-3-70B

In this section, we use the circuit extraction algorithm to obtain the components for *context-faithfulness* in Llama-70B. We note that ours is the first work (to the best of our knowledge) to retrieve circuit components in a large enterprise grade model. First, we plot the entropy of the attention values in the context window from the top scoring circuit attention heads, along with their corresponding attribution accuracies. We find that there exists a small set of attention heads with low entropy and high attribution accuracy on our probe dataset. Below we provide the circuit components corresponding to *context-faithfulness*:

Attention Layers. [78, 54, 75, 77, 58, 52, 53, 35, 7,2]

Attention Heads. [[75, 27], [52, 19], [64, 26], [58, 4], [67, 60], [78, 26], [75, 30], [39, 40], [78, 25], [72, 39], [75, 26], [53, 1], [64, 27]]

Below we provide further details regarding the attention head in the circuit which performs attribution by measuring the entropy of the attention values in context window. We also find that our attribution algorithm ATTNATTRIB is robust to larger context lengths for Llama-70B. These early results highlight that circuit extraction for real-world tasks such as extractive QA can be scaled towards large 70B (and potentially beyond) language models.

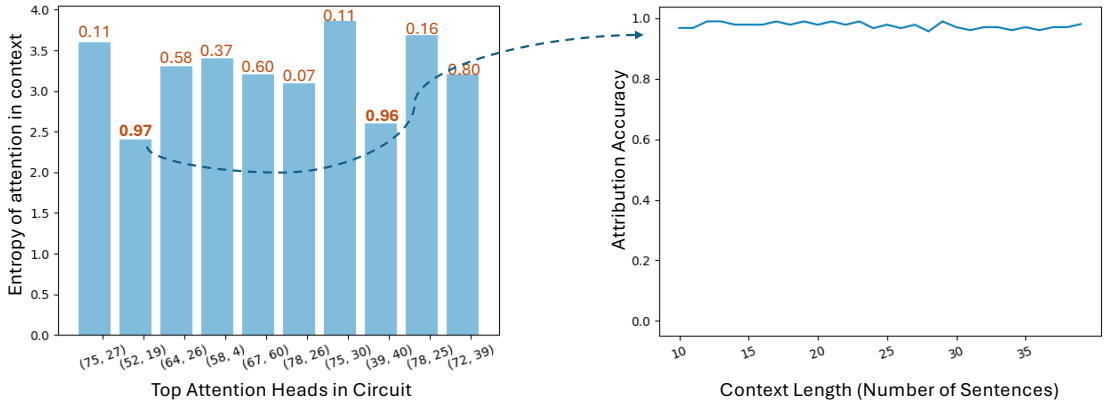

Figure 11.10: **A small number of attention heads in the context faithfulness circuit from Llama-3-70B performs attribution.** (Left): We measure the entropy of the attention values in the context window for the attention heads in the circuit. Brown color marks the attribution accuracy on the probe dataset $\mathscr{D}$. (Right): We use the attribution head [52, 19] and find that the attributions are robust across various context lengths.

11.8 Improving Compositionality in CLIP

11.8.1 Benchmark Datasets

Winoground [60, 223] is a challenging vision-language dataset for evaluating the visio-linguistic characteristics of contrastively trained image-text models. The dataset consists of

400 tasks, where each task consists of two image-text pairs. The objective is to independently assign the correct text caption to each image. Each task is also annotated with meta-data corresponding to whether the task requires object-understanding, relational-understanding or both. The tasks in Winoground are challenging as the images differ in fine-grained ways and assigning the correct text captions requires inherent compositional visual reasoning.

ARO [262] similarly tests visio-linguistic reasoning and consists of three types of tasks: (i) Visual Genome Attribution to test the understanding of object properties; (ii) Visual Genome Attribution to test for relational understanding between objects; and (iii) COCO-Order and Flickr30k-Order to test for order sensitivity of the words in a text, when performing image-text matching. We highlight that Winoground though slightly smaller in size than ARO is more challenging as it requires reasoning beyond visio-linguistic compositional knowledge [60].

11.8.2 Does distilling features directly from UNet help?

Previous works such as [250] find that the frozen features of the UNet contain structural information about the image. Motivated by this, we also investigate if distilling knowledge directly from the frozen UNet features is beneficial, Given an image x and its caption c, the frozen features f from the UNet (where $I(x,c) = \varepsilon_\theta(v_\alpha(x), t, c)$, similar to [250]) can be extracted. We then use these frozen internal representations from the UNet to regularize features of the image encoder in CLIP. In particular:

$$L_{total} = L_{CLIP} + \lambda \| h_w(f_\phi(x) - I(x,c)) \|_2^2 \tag{11.19}$$

However, we find that distillation in this way does not lead to improved performances for visio-linguistic reasoning. In fact, for ViT-B/16 (CLIP) we find the Winoground score to

decrease from 0.24 to 0.23. This result shows that using score-distillation sampling which involves backpropogation through the UNet is critical to distill knowledge from diffusion models to other discriminative models.

11.8.3 Additional Method Details

Algorithm 6 Algorithm to fine-tune CLIP with distillation from Stable-Diffusion for improved visio-linguistic reasoning

Input: $\mathscr{D}$: image-text pairs, f_ϕ: CLIP's image-encoder, g_γ: CLIP's text-encoder, ε_θ: UNet; N: Number of Epochs; λ: Hyper-parameter for the regularizer; $|B|$: Batch-size.

while $i \neq N$ **do**

- $\{x_j, y_j\}_{j=1}^{|B|} \leftarrow$ Sample a batch from $\mathscr{D}$
- $t \leftarrow$ Sample time-steps using DDPM
- $\varepsilon \leftarrow$ Sample Gaussian noise $\varepsilon \sim \mathscr{N}(0, \mathrm{I})$
- $L_{clip\leftarrow}$ Compute contrastive loss
- $L_{SDS} \leftarrow$ Compute SDS loss as in eq. (9.3)
- $L_{total} \leftarrow L_{clip} + \lambda L_{SDS}$
- L_{total}.backward() ▷ Backprop
- $\phi, \gamma, w \leftarrow$ Update the relevant parameters
- $i \leftarrow i + 1$

end while

11.8.4 When does distillation not help CLIP?

While we find that distilling knowledge from Stable-Diffusion to CLIP helps in *object-swap*, *relational-understanding* and *attribution-binding* visio-linguistic tasks, it does not help on tasks where the order of the text is perturbed (e.g. the COCO-Order and Flickr-Order tasks in the ARO dataset). In fact, we find that the denoising diffusion score in Eq. (9.1) leads to accuracies of 0.24 for COCO-Order and 0.34 for Flickr-Order which is in fact lower than CLIP models. Concurrent works [125] has shown similarly low performance for text-ordering tasks. A potential reason could be that ordering tasks only test for grammatical understanding which current text encoders cannot effectively model. Another reason could be that the denoising diffusion score is not affected by word ordering as the image semantics

Model	**Overall**	Object	Relation	Both	1 Main Pred	2 Main Preds
ViT-B/16(LAION 400M)	0.24	0.29	0.17	0.59	0.28	0.11
COCO FT with L_{CLIP}	0.24	0.26	0.21	0.54	0.31	0.10
COCO FT with $L_{CLIP}+L_{SDS}$	**0.30**	**0.34**	**0.23**	0.55	**0.33**	**0.14**

Table 11.5: **Additional results on Winoground with ViT-B/16 CLIP pre-trained on public data (LAION-400M)**.

are not changed as a result.

11.8.5 More Experimental Details

Hyper-parameters. We perform a hyperparameter sweep for the learning rate and the regularization hyperparameter λ for ViT-B/16. We use these same hyperparameters for different CLIP variants including ViT-B/32, ViT-B/14, ViT-L/14-336px and ResNet-50. In particular, we set $\lambda = 0.001$ and set the learning rate as 5×10^{-5}. We use a batch-size of 32 for all the different CLIP models. We use Stable-Diffusion v1-4 as the teacher model in our experiments.

Note on Full Fine-tuning. All our experiments were primarily done by fine-tuning only the LayerNorm parameters. In the initial phase of the project, we also fine-tune all the parameters of the text and image encoder in CLIP, however it results in worse performances than those reported in Table. (9.1). Potentially, this can be due to overfitting issues when used in conjunction with the new regularizer. We therefore run all the experiments with LayerNorm tuning as it leads to the best results.

Total GPU Hours. For all our experiments we use NVIDIA-A6000 and each fine-tuning experiment takes $\approx$6 hours.

11.8.6 Fine-tuning with Conceptual Captions

We primarily use MS-COCO as : (i) It's a relatively small dataset which can keep the fine-tuning steps relatively smaller and scaling the fine-tuning dataset will increase fine-tuning time; (ii) It's a well-established, relatively diverse and well annotated image-text dataset which is used by the community. We also fine-tuned with CC-3M [208], but found the improvements to be similar in lines to that using MS-COCO. For e.g., On Winoground with CC-3M, we find the following performance after distillation with Stable-Diffusion-v1-4: (i) ViT-B/16: 0.32; (ii) ViT-B/32: 0.32; (iii) ViT-L/14: 0.30; (iv) ViT-L/14-336px: 0.28; (iv) ResNet-50: 0.27. These scores are only marginally better than using MS-COCO, although the dataset size is more than 30 times – which shows that a high-quality dataset such as MS-COCO is sufficient for improving compositional abilities in CLIP.

11.8.7 Results with OpenCLIP

In Table 11.5, we show that our method is compatible with OpenCLIP. In particular, we find that distillation to OpenCLIP improves its visio-linguistic score from 0.24 to 0.30. These results highlight the generalizability of our distillation method.

11.8.8 Additional Results on CLEVR

We apply our fine-tuned model on the CLEVR task [110] – which consists of images of 3D shapes isolating phenomena such as spatial reasoning or attribute binding. We find that the diffusion-score leads to a score of 0.67, whereas the best CLIP variant in our test-bed (CLIP ViT-L/14) scored 0.63. With our distillation loss during fine-tuning – this score improved to 0.65 with a 2% gain.

Model	**Overall**	Object	Relation	Both	1 Main Pred	2 Main Preds
ViT-B/16(LAION 2B)	0.27	0.32	0.19	0.61	0.29	0.12
COCO FT with $L_{CLIP} + L_{SDS}$	**0.31**	**0.36**	**0.24**	0.53	**0.36**	**0.17**

Table 11.6: **CLIP (Pre-trained with 2B images) still underperforms on Winoground.** We show the CLIP even when trained with LAION-2B (similar scale of training data as Stable-Diffusion) still underperforms the diffusion score from Stable-Diffusion. This shows that scale of data alone cannot be useful in mitigating reasoning capabilities in CLIP.

11.8.9 Is it the Scale of Pre-Training Data Which Helps?

In Table 11.6, we show that CLIP models even when trained at the same scale of pre-training data as Stable-Diffusion (LAION-2B) struggle on the Winoground dataset. We specifically highlight that CLIP (when pre-trained on 2B image-text pairs) obtain a score of 0.27, whereas the diffusion model when trained on similar pre-training corpus obtains a score of 0.35. This clearly shows that at a similar pre-training scale, diffusion models (with their diffusion objective) are better compositional learners than CLIP like models. Our distillation method from Stable-Diffusion improves the Winoground score from 0.27 to 0.31 on CLIP(pre-trained on 2B image-text pairs).

11.8.10 Beyond CLIP

We find that Open-CoCa [261] pre-trained on 2B image-text pairs obtains a score of 0.30 on Winoground. With our distillation strategy, we find that the score improves to 0.33 highlighting that our distillation strategy can be used for models beyond CLIP. A full investigation of the impact of our distillation method on various vision-language models is deferred towards future work.

11.9 Improving Compositionality in Text-to-Image Models

Figure 11.11: **Qualitative Visualizations of our Method**.

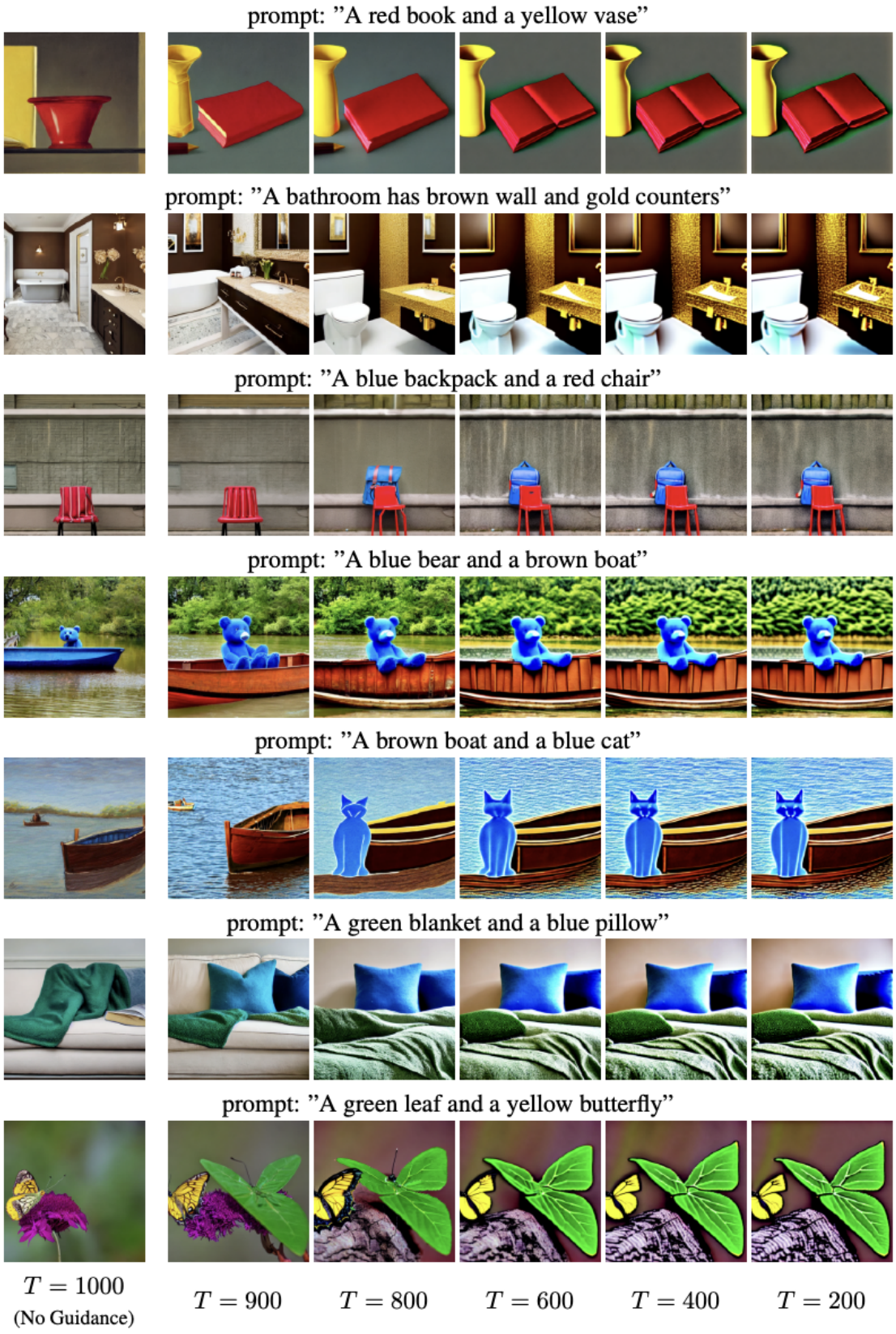

Figure 11.12: **Impact of our method at different time-steps**.

Bibliography

[1] Aishwarya Agarwal, Srikrishna Karanam, K J Joseph, Apoorv Saxena, Koustava Goswami, and Balaji Vasan Srinivasan. A-star: Test-time attention segregation and retention for text-to-image synthesis, 2023.

[2] Mayank Agarwal, Mikhail Yurochkin, and Yuekai Sun. On sensitivity of meta-learning to support data, 2021.

[3] Naman Agarwal, Brian Bullins, and Elad Hazan. Second order stochastic optimization in linear time. *ArXiv*, abs/1602.03943, 2016.

[4] Naman Agarwal, Brian Bullins, and Elad Hazan. Second-order stochastic optimization for machine learning in linear time. *Journal of Machine Learning Research*, 18(116):1–40, 2017.

[5] Aishwarya Agrawal, Ivana Kajić, Emanuele Bugliarello, Elnaz Davoodi, Anita Gergely, Phil Blunsom, and Aida Nematzadeh. Reassessing Evaluation Practices in Visual Question Answering: A Case Study on Out-of-Distribution Generalization, 2023.

[6] Aishwarya Agrawal, Jiasen Lu, Stanislaw Antol, Margaret Mitchell, C. Lawrence Zitnick, Dhruv Batra, and Devi Parikh. VQA: Visual Question Answering, 2016.

[7] Ahmed Alaa and Mihaela Van Der Schaar. Validating causal inference models via influence functions. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 191–201, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[8] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, and Malcolm Reynolds. Flamingo: a visual language model for few-shot learning. In *Advances in Neural Information Processing Systems*, volume 35, pages 23716–23736, 2022.

[9] Anderson. Iris flower dataset. In -, 1936.

[10] Anonymous. ReFACT: Updating text-to-image models by editing the text encoder. In *Submitted to The Twelfth International Conference on Learning Representations*, 2023. under review.

[11] Andy Arditi, Oscar Obeso, Aaquib Syed, Daniel Paleka, Nina Panickssery, Wes Gurnee, and Neel Nanda. Refusal in language models is mediated by a single direction, 2024.

[12] Sébastien M. R. Arnold, Guneet S. Dhillon, Avinash Ravichandran, and Stefano Soatto. Uniform sampling over episode difficulty, 2021.

[13] Imanol Arrieta-Ibarra, Leonard Goff, Diego Jiménez-Hernández, Jaron Lanier, and E. Glen Weyl. Should we treat data as labor? moving beyond "free". *AEA Papers and Proceedings*, 108:38–42, May 2018.

[14] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection, 2023.

[15] Konstantin E. Avrachenkov, Jerzy A. Filar, and Phil G. Howlett. *Analytic Perturbation Theory and Its Applications*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2013.

[16] Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, Bryan Catanzaro, Tero Karras, and Ming-Yu Liu. ediff-i: Text-to-image diffusion models with an ensemble of expert denoisers. *CoRR*, abs/2211.01324, 2022.

[17] Sriram Balasubramanian and Soheil Feizi. Towards improved input masking for convolutional neural networks, 2023.

[18] Yamini Bansal, Preetum Nakkiran, and Boaz Barak. Revisiting model stitching to compare neural representations, 2021.

[19] Hangbo Bao, Li Dong, and Furu Wei. Beit: BERT pre-training of image transformers. *CoRR*, abs/2106.08254, 2021.

[20] Andrei Barbu, David Mayo, Julian Alverio, William Luo, Christopher Wang, Dan Gutfreund, Josh Tenenbaum, and Boris Katz. Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[21] Samyadeep Basu, Daniela Massiceti, Shell Xu Hu, and Soheil Feizi. Strong baselines for parameter efficient few-shot fine-tuning, 2023.

[22] Samyadeep Basu, Phillip Pope, and Soheil Feizi. Influence functions in deep learning are fragile. *CoRR*, abs/2006.14651, 2020.

[23] Samyadeep Basu, Nanxuan Zhao, Vlad Morariu, Soheil Feizi, and Varun Manjunatha. Localizing and editing knowledge in text-to-image generative models, 2023.

[24] David Bau, Jun-Yan Zhu, Hendrik Strobelt, Àgata Lapedriza, Bolei Zhou, and Antonio Torralba. Understanding the role of individual units in a deep neural network. *CoRR*, abs/2009.05041, 2020.

[25] David Bau, Jun-Yan Zhu, Hendrik Strobelt, Agata Lapedriza, Bolei Zhou, and Antonio Torralba. Understanding the role of individual units in a deep neural network. *Proceedings of the National Academy of Sciences*, 117(48):30071–30078, September 2020.

[26] Leonard Bereska and Efstratios Gavves. Mechanistic interpretability for ai safety – a review, 2024.

[27] Luca Bertinetto, Joao F. Henriques, Philip Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. In *International Conference on Learning Representations*, 2019.

[28] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[29] Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. https://transformer-circuits.pub/2023/monosemantic-features/index.html.

[30] Marc-Etienne Brunet, Colleen Alkalay-Houlihan, Ashton Anderson, and Richard S. Zemel. Understanding the origins of bias in word embeddings. *CoRR*, abs/1810.03611, 2018.

[31] Jan Buchmann, Xiao Liu, and Iryna Gurevych. Attribute or abstain: Large language models as long document assistants, 2024.

[32] Nick Cammarata, Shan Carter, Gabriel Goh, Chris Olah, Michael Petrov, Ludwig Schubert, Chelsea Voss, Ben Egan, and Swee Kiat Lim. Thread: Circuits. *Distill*, 2020. https://distill.pub/2020/circuits.

[33] E.J. Candes and T. Tao. Decoding by linear programming. *IEEE Transactions on Information Theory*, 51(12):4203–4215, 2005.

[34] Emmanuel Candes, Justin Romberg, and Terence Tao. Stable signal recovery from incomplete and inaccurate measurements, 2005.

[35] Nicolas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramèr, Borja Balle, Daphne Ippolito, and Eric Wallace. Extracting training data from diffusion models. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 5253–5270, Anaheim, CA, August 2023. USENIX Association.

[36] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *CoRR*, abs/2104.14294, 2021.

[37] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

[38] Huiwen Chang, Han Zhang, Jarred Barber, Aaron Maschinot, José Lezama, Lu Jiang, Ming-Hsuan Yang, Kevin Patrick Murphy, William T. Freeman, Michael Rubinstein, Yuanzhen Li, and Dilip Krishnan. Muse: Text-to-image generation via masked generative transformers. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 4055–4075. PMLR, 2023.

[39] Hila Chefer, Yuval Alaluf, Yael Vinker, Lior Wolf, and Daniel Cohen-Or. Attend-and-excite: Attention-based semantic guidance for text-to-image diffusion models, 2023.

[40] Hongge Chen, Si Si, Yang Li, Ciprian Chelba, Sanjiv Kumar, Duane Boning, and Cho-Jui Hsieh. {MULTI}-{stage} {influence} {function}, 2020.

[41] Huanran Chen, Yinpeng Dong, Zhengyi Wang, X. Yang, Chen-Dong Duan, Hang Su, and Jun Zhu. Robust classification via a single diffusion model. *ArXiv*, abs/2305.15241, 2023.

[42] Irene Chen, Fredrik D Johansson, and David Sontag. Why is my classifier discriminatory? In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 3539–3550. Curran Associates, Inc., 2018.

[43] Junsong Chen, Jincheng Yu, Chongjian Ge, Lewei Yao, Enze Xie, Yue Wu, Zhongdao Wang, James Kwok, Ping Luo, Huchuan Lu, and Zhenguo Li. Pixart-α: Fast training of diffusion transformer for photorealistic text-to-image synthesis, 2023.

[44] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. In *International Conference on Learning Representations*, 2019.

[45] Yinbo Chen, Xiaolong Wang, Zhuang Liu, Huijuan Xu, and Trevor Darrell. A new meta-baseline for few-shot learning. *CoRR*, abs/2003.04390, 2020.

[46] Siyuan Cheng, Bozhong Tian, Qingbin Liu, Xi Chen, Yongheng Wang, Huajun Chen, and Ningyu Zhang. Can We Edit Multimodal Large Language Models?, 2024.

[47] Mehdi Cherti, Romain Beaumont, Ross Wightman, Mitchell Wortsman, Gabriel Ilharco, Cade Gordon, Christoph Schuhmann, Ludwig Schmidt, and Jenia Jitsev. Reproducible scaling laws for contrastive language-image learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2818–2829, 2023.

[48] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023.

[49] Kevin Clark and Priyank Jaini. Text-to-image diffusion models are zero-shot classifiers, 2023.

[50] R. Dennis Cook and Weisberg Sanford. Residuals and influence in regression. *Chapman and Hall*, 1982.

[51] R. Dennis Cook and Sanford Weisberg. Characterizations of an empirical influence function for detecting influential cases in regression. *Technometrics*, 22(4):495–508, 1980.

[52] George Corliss. Which root does the bisection algorithm find? *SIAM Review*, 19(2):325–327, 1977.

[53] Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale N Fung, and Steven Hoi. InstructBLIP: Towards general-purpose vision-language models with instruction tuning. In *Advances in Neural Information Processing Systems*, volume 36, 2024.

[54] Guy Dar, Mor Geva, Ankit Gupta, and Jonathan Berant. Analyzing Transformers in Embedding Space, 2023.

[55] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[56] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[57] Guneet S. Dhillon, Pratik Chaudhari, Avinash Ravichandran, and Stefano Soatto. A baseline for few-shot image classification. 2019.

[58] Guneet S. Dhillon, Pratik Chaudhari, Avinash Ravichandran, and Stefano Soatto. A baseline for few-shot image classification. *CoRR*, abs/1909.02729, 2019.

[59] Ming Ding, Wendi Zheng, Wenyi Hong, and Jie Tang. Cogview2: Faster and better text-to-image generation via hierarchical transformers. *arXiv preprint arXiv:2204.14217*, 2022.

[60] Anuj Diwan, Layne Berry, Eunsol Choi, David Harwath, and Kyle Mahowald. Why is winoground hard? investigating failures in visuolinguistic compositionality, 2022.

[61] D. L. Donoho. Compressed sensing. *IEEE Trans. Inf. Theor.*, 52(4):1289–1306, apr 2006.

[62] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020.

[63] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[64] Vincent Dumoulin, Neil Houlsby, Utku Evci, Xiaohua Zhai, Ross Goroshin, Sylvain Gelly, and Hugo Larochelle. Comparing transfer and meta learning approaches on a unified few-shot classification benchmark, 2021.

[65] Vincent Dumoulin, Neil Houlsby, Utku Evci, Xiaohua Zhai, Ross Goroshin, Sylvain Gelly, and Hugo Larochelle. A unified few-shot classification benchmark to compare transfer and meta learning approaches. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021.

[66] Bradley Efron. Jackknife-after-bootstrap standard errors and influence functions. *Journal of the Royal Statistical Society. Series B (Methodological)*, 54(1):83–127, 1992.

[67] Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of superposition. *Transformer Circuits Thread*, 2022. https://transformer-circuits.pub/2022/toy_model/index.html.

[68] Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. https://transformer-circuits.pub/2021/framework/index.html.

[69] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first International Conference on Machine Learning*, 2024.

[70] Abhimanyu Dubey et al. The llama 3 herd of models, 2024.

[71] Li Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611, 2006.

[72] Weixi Feng, Xuehai He, Tsu-Jui Fu, Varun Jampani, Arjun Akula, Pradyumna Narayana, Sugato Basu, Xin Eric Wang, and William Yang Wang. Training-free structured diffusion guidance for compositional text-to-image synthesis. *arXiv preprint arXiv:2212.05032*, 2022.

[73] Weixi Feng, Wanrong Zhu, Tsu jui Fu, Varun Jampani, Arjun Akula, Xuehai He, Sugato Basu, Xin Eric Wang, and William Yang Wang. Layoutgpt: Compositional visual planning and generation with large language models, 2023.

[74] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR, 06–11 Aug 2017.

[75] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *CoRR*, abs/1703.03400, 2017.

[76] Minghao Fu, Yun-Hao Cao, and Jianxin Wu. Worst case matters for few-shot recognition, 2022.

[77] Yossi Gandelsman, Alexei A Efros, and Jacob Steinhardt. Interpreting CLIP's image representation via text-based decomposition. In *The Twelfth International Conference on Learning Representations*, 2024.

[78] Rohit Gandikota, Joanna Materzyńska, Jaden Fiotto-Kaufman, and David Bau. Erasing concepts from diffusion models. In *Proceedings of the 2023 IEEE International Conference on Computer Vision*, 2023.

[79] Rohit Gandikota, Hadas Orgad, Yonatan Belinkov, Joanna Materzyńska, and David Bau. Unified concept editing in diffusion models. *IEEE/CVF Winter Conference on Applications of Computer Vision*, 2024.

[80] Tianyu Gao, Howard Yen, Jiatong Yu, and Danqi Chen. Enabling large language models to generate text with citations, 2023.

[81] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey, 2024.

[82] Kristian Georgiev, Joshua Vendrow, Hadi Salman, Sung Min Park, and Aleksander Madry. The journey, not the destination: How data guides diffusion models, 2023.

[83] Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. Dissecting Recall of Factual Associations in Auto-Regressive Language Models, 2023.

[84] Amirata Ghorbani, Abubakar Abid, and James Y. Zou. Interpretation of neural networks is fragile. In *AAAI*, 2017.

[85] Ryan Giordano, Michael I. Jordan, and Tamara Broderick. A higher-order swiss army infinitesimal jackknife. *ArXiv*, abs/1907.12116, 2019.

[86] Gabriel Goh, Nick Cammarata †, Chelsea Voss †, Shan Carter, Michael Petrov, Ludwig Schubert, Alec Radford, and Chris Olah. Multimodal neurons in artificial neural networks. *Distill*, 2021. https://distill.pub/2021/multimodal-neurons.

[87] Rhys Gould, Euan Ong, George Ogden, and Arthur Conmy. Successor heads: Recurring, interpretable attention heads in the wild, 2023.

[88] Roger Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, Evan Hubinger, Kamilė Lukošiūtė, Karina Nguyen, Nicholas Joseph, Sam McCandlish, Jared Kaplan, and Samuel R. Bowman. Studying large language model generalization with influence functions, 2023.

[89] Yunhui Guo, Noel C Codella, Leonid Karlinsky, James V Codella, John R Smith, Kate Saenko, Tajana Rosing, and Rogerio Feris. A broader study of cross-domain few-shot learning. ECCV, 2020.

[90] Xiaochuang Han, Byron C. Wallace, and Yulia Tsvetkov. Explaining black box predictions and unveiling data artifacts through influence functions, 2020.

[91] Michael Hanna, Ollie Liu, and Alexandre Variengien. How does gpt-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model, 2023.

[92] Jamie Hayes and Olga Ohrimenko. Contamination attacks and mitigation in multi-party machine learning. *CoRR*, abs/1901.02402, 2019.

[93] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[94] Muyang He, Yexin Liu, Boya Wu, Jianhao Yuan, Yueze Wang, Tiejun Huang, and Bo Zhao. Efficient Multimodal Learning from Data-centric Perspective. *arXiv preprint arXiv:2402.11530*, 2 2024.

[95] Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. *CoRR*, abs/1506.03340, 2015.

[96] Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control, 2022.

[97] Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning. *arXiv preprint arXiv:2104.08718*, 2021.

[98] Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning. *CoRR*, abs/2104.08718, 2021.

[99] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.

[100] Shell Xu Hu, Da Li, Jan Stühmer, Minyoung Kim, and Timothy M. Hospedales. Pushing the limits of simple pipelines for few-shot learning: External data and fine-tuning make a difference, 2022.

[101] Yushi Hu, Benlin Liu, Jungo Kasai, Yizhong Wang, Mari Ostendorf, Ranjay Krishna, and Noah A Smith. Tifa: Accurate and interpretable text-to-image faithfulness evaluation with question answering, 2023.

[102] Jie Huang and Kevin Chen-Chuan Chang. Citation: A key to building responsible and accountable large language models, 2024.

[103] Kaiyi Huang, Kaiyue Sun, Enze Xie, Zhenguo Li, and Xihui Liu. T2i-compbench: A comprehensive benchmark for open-world compositional text-to-image generation, 2023.

[104] Yufeng Huang, Jiji Tang, Zhuo Chen, Rongsheng Zhang, Xinfeng Zhang, Weijie Chen, Zeng Zhao, Tangjie Lv, Zhipeng Hu, and Wen Zhang. Structure-clip: Enhance multi-modal language representations with structure knowledge, 2023.

[105] Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. Openclip, July 2021. If you use this software, please cite it as below.

[106] Alon Jacovi and Yoav Goldberg. Towards faithfully interpretable nlp systems: How should we define and evaluate faithfulness?, 2020.

[107] Louis A. Jaeckel. The infinitesimal jackknife. *Technical Report*, 1:1–35, 06 1972.

[108] Robins James, Li Lingling, Tchetgen Eric, and Aad van der Vaart. Higher order influence functions and minimax estimation of nonlinear functionals. *Probability and Statistics: Essays in Honor of David A. Freedman, 335–421*, abs/1706.03825, 2017.

[109] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023.

[110] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross B. Girshick. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. *CoRR*, abs/1612.06890, 2016.

[111] Minguk Kang, Jun-Yan Zhu, Richard Zhang, Jaesik Park, Eli Shechtman, Sylvain Paris, and Taesung Park. Scaling up gans for text-to-image synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[112] Kevin Meng and David Bau and Alex Andonian and Yonatan Belinkov. Locating and editing factual associations in GPT. *Advances in Neural Information Processing Systems*, 35:17359–17372, 2022.

[113] Muhammad Khalifa, David Wadden, Emma Strubell, Honglak Lee, Lu Wang, Iz Beltagy, and Hao Peng. Source-aware training enables knowledge attribution in language models, 2024.

[114] Krishnateja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, and Rishabh Iyer. Glister: Generalization based data subset selection for efficient and robust learning. 2020.

[115] Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T. Schütt, Sven Dähne, Dumitru Erhan, and Been Kim. The (un)reliability of saliency methods, 2017.

[116] Wilhelm Kirch, editor. *Pearson's Correlation Coefficient*, pages 1090–1091. Springer Netherlands, Dordrecht, 2008.

[117] P. W. Koh, J. Steinhardt, and P. Liang. Stronger data poisoning attacks break data sanitization defenses. *arXiv preprint arXiv:1811.00741*, 2019.

[118] Pang Wei Koh, Kai-Siang Ang, Hubert H. K. Teo, and Percy Liang. On the accuracy of influence functions for measuring group effects. *CoRR*, abs/1905.13289, 2019.

[119] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1885–1894, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[120] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions, 2020.

[121] Teuvo Kohonen. Correlation Matrix Memories. *IEEE Transactions on Computers*, C-21:353–359, 1972.

[122] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Large scale learning of general visual representations for transfer. *CoRR*, abs/1912.11370, 2019.

[123] Shuming Kong, Yanyan Shen, and Linpeng Huang. Resolving training biases via influence-based data relabeling. In *International Conference on Learning Representations*, 2022.

[124] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). *CVPR*, 2000.

[125] Benno Krojer, Elinor Poole-Dayan, Vikram Voleti, Christopher Pal, and Siva Reddy. Are diffusion models vision-and-language reasoners?, 2023.

[126] Nupur Kumari, Bingliang Zhang, Sheng-Yu Wang, Eli Shechtman, Richard Zhang, and Jun-Yan Zhu. Ablating concepts in text-to-image diffusion models. In *'International Conference on Computer Vision (ICCV)'*, 2023.

[127] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai,

Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019.

[128] Black Forest Labs. Flux. `https://github.com/black-forest-labs/flux`, 2024.

[129] Brenden M. Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua B. Tenenbaum. One shot learning of simple visual concepts. *Cognitive Science*, 33, 2011.

[130] Connor Lawless, Jakob Schoeffer, Lindy Le, Kael Rowan, Shilad Sen, Cristina St Hill, Jina Suh, and Bahar Sarrafzadeh. "I Want It That Way": Enabling Interactive Decision Support Using Large Language Models and Constraint Programming. *arXiv preprint arXiv:2312.06908*, 2023.

[131] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, volume 86, pages 2278–2324, 1998.

[132] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[133] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *CVPR*, 2019.

[134] Karel Lenc and Andrea Vedaldi. Understanding image representations by measuring their equivariance and equivalence, 2015.

[135] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks, 2021.

[136] Alexander C. Li, Mihir Prabhudesai, Shivam Duggal, Ellis Brown, and Deepak Pathak. Your diffusion model is secretly a zero-shot classifier, 2023.

[137] Dongfang Li, Zetian Sun, Xinshuo Hu, Zhenyu Liu, Ziyang Chen, Baotian Hu, Aiguo Wu, and Min Zhang. A survey of large language models attribution, 2023.

[138] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International Conference on Machine Learning*, pages 19730–19742. PMLR, 2023.

[139] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation, 2022.

[140] Junnan Li, Dongxu Li, Caiming Xiong, and Steven C. H. Hoi. BLIP: bootstrapping language-image pre-training for unified vision-language understanding and generation. *CoRR*, abs/2201.12086, 2022.

[141] Wei-Hong Li, Xialei Liu, and Hakan Bilen. Cross-domain few-shot learning with task-specific adapters, 2021.

[142] Yuheng Li, Haotian Liu, Qingyang Wu, Fangzhou Mu, Jianwei Yang, Jianfeng Gao, Chunyuan Li, and Yong Jae Lee. Gligen: Open-set grounded text-to-image generation, 2023.

[143] Long Lian, Boyi Li, Adam Yala, and Trevor Darrell. Llm-grounded diffusion: Enhancing prompt understanding of text-to-image diffusion models with large language models. *ArXiv*, abs/2305.13655, 2023.

[144] Tom Lieberum, Matthew Rahtz, János Kramár, Neel Nanda, Geoffrey Irving, Rohin Shah, and Vladimir Mikulik. Does circuit analysis interpretability scale? evidence from multiple choice capabilities in chinchilla, 2023.

[145] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[146] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[147] Wei-Yang Lin, Ya-Han Hu, and Chih-Fong Tsai. Machine learning in financial crisis prediction: A survey. *Trans. Sys. Man Cyber Part C*, 42(4):421–436, July 2012.

[148] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved Baselines with Visual Instruction Tuning. *arXiv preprint arXiv:2310.03744*, 10 2023.

[149] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in Neural Information Processing Systems*, 36, 2024.

[150] Nan Liu, Shuang Li, Yilun Du, Antonio Torralba, and Joshua B. Tenenbaum. Compositional visual generation with composable diffusion models. *ArXiv*, abs/2206.01714, 2022.

[151] Nan Liu, Shuang Li, Yilun Du, Antonio Torralba, and Joshua B Tenenbaum. Compositional visual generation with composable diffusion models. In *European Conference on Computer Vision*, pages 423–439. Springer, 2022.

[152] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[153] Shayne Longpre, Kartik Perisetla, Anthony Chen, Nikhil Ramesh, Chris DuBois, and Sameer Singh. Entity-based knowledge conflicts in question answering, 2022.

[154] Albert Lu, Hongxin Zhang, Yanzhe Zhang, Xuezhi Wang, and Diyi Yang. Bounding the capabilities of large language models in open text generation with prompt constraints. *arXiv preprint arXiv:2302.09185*, 2023.

[155] Shilin Lu, Zilan Wang, Leyang Li, Yanzhu Liu, and Adams Wai-Kin Kong. Mace: Mass concept erasure in diffusion models, 2024.

[156] Timo Lüddecke and Alexander S. Ecker. Prompt-based multi-modal image segmentation. *CoRR*, abs/2112.10003, 2021.

[157] Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *CoRR*, abs/1705.07874, 2017.

[158] Alexander Selvikvåg Lundervold and Arvid Lundervold. An overview of deep learning in medical imaging focusing on MRI. *CoRR*, abs/1811.10052, 2018.

[159] David Madras, James Atwood, and Alex D'Amour. Detecting extrapolation with influence functions. *ICML Workshop*, 2019.

[160] Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories, 2023.

[161] Kenneth Marino, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. OK-VQA: A visual question answering benchmark requiring external knowledge. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3195–3204, 2019.

[162] Kenneth Marino, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. OK-VQA: A visual question answering benchmark requiring external knowledge. *CoRR*, abs/1906.00067, 2019.

[163] Daniela Massiceti, Lida Theodorou, Luisa Zintgraf, Matthew Tobias Harris, Simone Stumpf, Cecily Morrison, Edward Cutrell, and Katja Hofmann. Orbit: A real-world few-shot dataset for teachable object recognition collected from people who are blind or low vision, 2021.

[164] Callum McDougall, Arthur Conmy, Cody Rushing, Thomas McGrath, and Neel Nanda. Copy suppression: Comprehensively understanding an attention head, 2023.

[165] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt, 2023.

[166] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt, 2023.

[167] Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. Mass-Editing Memory in a Transformer, 2023.

[168] Thomas Mensink, Jasper Uijlings, Lluis Castrejon, Arushi Goel, Felipe Cadar, Howard Zhou, Fei Sha, André Araujo, and Vittorio Ferrari. Encyclopedic VQA: Visual questions about detailed properties of fine-grained categories. In *VQA*, pages 3113–3124, 2023.

[169] Matthias Minderer, Alexey Gritsenko, Austin Stone, Maxim Neumann, Dirk Weissenborn, Alexey Dosovitskiy, Aravindh Mahendran, Anurag Arnab, Mostafa Dehghani, Zhuoran Shen, Xiao Wang, Xiaohua Zhai, Thomas Kipf, and Neil Houlsby. Simple open-vocabulary object detection with vision transformers, 2022.

[170] Mazda Moayeri, Keivan Rezaei, Maziar Sanjabi, and Soheil Feizi. Text-to-concept (and back) via cross-model alignment, 2023.

[171] Ron Mokady, Amir Hertz, and Amit H. Bermano. Clipcap: Clip prefix for image captioning, 2021.

[172] Alexander Mordvintsev, Nicola Pezzotti, Ludwig Schubert, and Chris Olah. Differentiable image parameterizations. *Distill*, 2018. https://distill.pub/2018/differentiable-parameterizations.

[173] Norman Mu, Alexander Kirillov, David A. Wagner, and Saining Xie. SLIP: self-supervision meets language-image pre-training. *CoRR*, abs/2112.12750, 2021.

[174] B. K. Natarajan. Sparse approximate solutions to linear systems. *SIAM Journal on Computing*, 24(2):227–234, 1995.

[175] Weili Nie, Sifei Liu, Morteza Mardani, Chao Liu, Benjamin Eckart, and Arash Vahdat. Compositional text-to-image generation with dense blob representations, 2024.

[176] Cheng Niu, Yuanhao Wu, Juno Zhu, Siliang Xu, Kashun Shum, Randy Zhong, Juntong Song, and Tong Zhang. Ragtruth: A hallucination corpus for developing trustworthy retrieval-augmented language models, 2024.

[177] Tuomas Oikarinen and Tsui-Wei Weng. Clip-dissect: Automatic description of neuron representations in deep vision networks, 2023.

[178] Chris Olah, Arvind Satyanarayan, Ian Johnson, Shan Carter, Ludwig Schubert, Katherine Ye, and Alexander Mordvintsev. The building blocks of interpretability. *Distill*, 2018. https://distill.pub/2018/building-blocks.

[179] OpenAI. Gpt-4v(ision) system card, 9 2023.

[180] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Pow-

ell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. GPT-4 Technical Report, 2024.

[181] Maxime Oquab, Timothée Darcet, Theo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Russell Howes, Po-Yao Huang, Hu Xu, Vasu Sharma, Shang-Wen Li, Wojciech Galuba, Mike Rabbat, Mido Assran, Nicolas Ballas, Gabriel Synnaeve, Ishan Misra, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2023.

[182] Hadas Orgad, Bahjat Kawar, and Yonatan Belinkov. Editing implicit assumptions in text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

[183] Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. Trak: Attributing model behavior at scale, 2023.

[184] Judea Pearl. Direct and indirect effects. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, UAI'01, page 411–420, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[185] Barak A. Pearlmutter. Fast exact multiplication by the hessian. *Neural Comput.*, 6(1):147–160, January 1994.

[186] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis, 2023.

[187] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion, 2022.

[188] Nikhil Prakash, Tamar Rott Shaham, Tal Haklay, Yonatan Belinkov, and David Bau. Fine-tuning enhances existing mechanisms: A case study on entity tracking, 2024.

[189] Garima Pruthi, Frederick Liu, Mukund Sundararajan, and Satyen Kale. Estimating training data influence by tracking gradient descent. *ArXiv*, abs/2002.08484, 2020.

[190] Garima Pruthi, Frederick Liu, Mukund Sundararajan, and Satyen Kale. Estimating training data influence by tracking gradient descent. *CoRR*, abs/2002.08484, 2020.

[191] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[192] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *CoRR*, abs/2103.00020, 2021.

[193] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8821–8831. PMLR, 18–24 Jul 2021.

[194] Royi Rassin, Eran Hirsch, Daniel Glickman, Shauli Ravfogel, Yoav Goldberg, and Gal Chechik. Linguistic binding in diffusion models: Enhancing attribute correspondence through attention map alignment, 2024.

[195] Mengye Ren, Michael L. Iuzzolino, Michael C. Mozer, and Richard S. Zemel. Wandering within a world: Online contextualized few-shot learning. *CoRR*, abs/2007.04546, 2020.

[196] James Requeima, Jonathan Gordon, John Bronskill, Sebastian Nowozin, and Richard E. Turner. Fast and flexible multi-task classification using conditional neural adaptive processes. 2019.

[197] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. *CoRR*, abs/2112.10752, 2021.

[198] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. *CoRR*, abs/2112.10752, 2021.

[199] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. *CoRR*, abs/2112.10752, 2021.

[200] Shiori Sagawa, Pang Wei Koh, Tatsunori B. Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization, 2020.

[201] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 36479–36494. Curran Associates, Inc., 2022.

[202] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade W Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa R Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. LAION-5b: An open large-scale dataset for training next generation image-text models. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

[203] Peter G. Schulam and Suchi Saria. Can you trust this prediction? auditing pointwise reliability after learning. In *AISTATS*, 2019.

[204] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47, March 2002.

[205] Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. *CoRR*, abs/1610.02391, 2016.

[206] Harshay Shah, Andrew Ilyas, and Aleksander Madry. Decomposing and editing predictions by modeling model computation, 2024.

[207] Arnab Sen Sharma, David Atkinson, and David Bau. Locating and Editing Factual Associations in Mamba. *arXiv preprint arXiv:2404.03646*, 2024.

[208] Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2556–2565, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[209] Jonathan R Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. -, 1994.

[210] Weijia Shi, Xiaochuang Han, Mike Lewis, Yulia Tsvetkov, Luke Zettlemoyer, and Scott Wen tau Yih. Trusting your evidence: Hallucinate less with context-aware decoding, 2023.

[211] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[212] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda B. Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *CoRR*, abs/1706.03825, 2017.

[213] Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning, 2017.

[214] Gowthami Somepalli, Vasu Singla, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Diffusion art or digital forgery? investigating data replication in diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6048–6058, June 2023.

[215] Gowthami Somepalli, Vasu Singla, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Understanding and mitigating copying in diffusion models. *arXiv preprint arXiv:2305.20086*, 2023.

[216] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *CoRR*, abs/2010.02502, 2020.

[217] Gabriela Ben Melech Stan, Raanan Yehezkel Rohekar, Yaniv Gurwicz, Matthew Lyle Olson, Anahita Bhiwandiwalla, Estelle Aflalo, Chenfei Wu, Nan Duan, Shao-Yen Tseng, and Vasudev Lal. LVLM-Intrepret: An Interpretability Tool for Large Vision-Language Models, 2024.

[218] Megan Stanley, John F Bronskill, Krzysztof Maziarz, Hubert Misztela, Jessica Lanini, Marwin Segler, Nadine Schneider, and Marc Brockschmidt. FS-mol: A few-shot learning dataset of molecules. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[219] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. *CoRR*, abs/1703.01365, 2017.

[220] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag, Berlin, Heidelberg, 1st edition, 2010.

[221] Raphael Tang, Linqing Liu, Akshat Pandey, Zhiying Jiang, Gefei Yang, Karun Kumar, Pontus Stenetorp, Jimmy Lin, and Ferhan Ture. What the DAAM: Interpreting stable diffusion using cross attention. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5644–5659, Toronto, Canada, July 2023. Association for Computational Linguistics.

[222] D. Temel, J. Lee, and G. AlRegib. Cure-or: Challenging unreal and real environments for object recognition. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2018.

[223] Tristan Thrush, Ryan Jiang, Max Bartolo, Amanpreet Singh, Adina Williams, Douwe Kiela, and Candace Ross. Winoground: Probing vision and language models for visio-linguistic compositionality, 2022.

[224] Yonglong Tian, Yue Wang, Dilip Krishnan, Joshua B. Tenenbaum, and Phillip Isola. Rethinking few-shot image classification: a good embedding is all you need? *CoRR*, abs/2003.11539, 2020.

[225] Shengbang Tong, Zhuang Liu, Yuexiang Zhai, Yi Ma, Yann LeCun, and Saining Xie. Eyes Wide Shut? Exploring the Visual Shortcomings of Multimodal LLMs, 2024.

[226] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *CoRR*, abs/2012.12877, 2020.

[227] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention, 2021.

[228] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

[229] Eleni Triantafillou. *Towards Strong Generalization from Few Examples*. PhD thesis, 2021. Copyright - Database copyright ProQuest LLC; ProQuest does not claim copyright in the individual underlying works; Last updated - 2022-01-14.

[230] Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, and Hugo Larochelle. Meta-dataset: A dataset of datasets for learning to learn from few examples. *CoRR*, abs/1903.03096, 2019.

[231] Zhengzhong Tu, Hossein Talebi, Han Zhang, Feng Yang, Peyman Milanfar, Alan Bovik, and Yinxiao Li. Maxvit: Multi-axis vision transformer. *ECCV*, 2022.

[232] Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J. Vazquez, Ulisse Mini, and Monte MacDiarmid. Activation addition: Steering language models without optimization, 2024.

[233] Aaron van den Oord, Oriol Vinyals, and koray kavukcuoglu. Neural discrete representation learning. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[234] Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. *CoRR*, abs/1711.00937, 2017.

[235] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *CoRR*, abs/1706.03762, 2017.

[236] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[237] Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart M. Shieber. Causal mediation analysis for interpreting neural NLP: the case of gender bias. *CoRR*, abs/2004.12265, 2020.

[238] Martina G. Vilas, Timothy Schaumlöffel, and Gemma Roig. Analyzing vision transformers for image classification in class embedding space, 2023.

[239] Oriol Vinyals, Charles Blundell, Timothy P. Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. *CoRR*, abs/1606.04080, 2016.

[240] Ben Wang and Aran Komatsuzaki. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. `https://github.com/kingoflolz/mesh-transformer-jax`, May 2021.

[241] Fei Wang, Wenjie Mo, Yiwei Wang, Wenxuan Zhou, and Muhao Chen. A causal view of entity bias in (large) language models, 2023.

[242] Hao Wang, Berk Ustun, and Flávio P. Calmon. Repairing without retraining: Avoiding disparate impact with counterfactual distributions. *CoRR*, abs/1901.10501, 2019.

[243] Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the Wild: a Circuit for Indirect Object Identification in GPT-2 small, 2022.

[244] Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small, 2022.

[245] Ruichen Wang, Zekang Chen, Chen Chen, Jian Ma, Haonan Lu, and Xiaodong Lin. Compositional text-to-image synthesis with attention map control of diffusion models, 2023.

[246] Zhaoqing Wang, Yu Lu, Qiang Li, Xunqiang Tao, Yandong Guo, Mingming Gong, and Tongliang Liu. CRIS: clip-driven referring image segmentation. *CoRR*, abs/2111.15174, 2021.

[247] Kai Wei, Rishabh Iyer, and Jeff Bilmes. Submodularity in data subset selection and active learning. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1954–1963, Lille, France, 07–09 Jul 2015. PMLR.

[248] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019.

[249] Kevin Wu, Eric Wu, and James Zou. Clasheval: Quantifying the tug-of-war between an llm's internal prior and external evidence, 2024.

[250] Jiarui Xu, Sifei Liu, Arash Vahdat, Wonmin Byeon, Xiaolong Wang, and Shalini De Mello. Open-vocabulary panoptic segmentation with text-to-image diffusion models, 2023.

[251] Rongwu Xu, Zehan Qi, Zhijiang Guo, Cunxiang Wang, Hongru Wang, Yue Zhang, and Wei Xu. Knowledge conflicts for llms: A survey, 2024.

[252] Shicheng Xu, Liang Pang, Huawei Shen, Xueqi Cheng, and Tat-Seng Chua. Search-in-the-chain: Interactively enhancing large language models with search for knowledge-intensive tasks, 2024.

[253] Yuchen Yang, Gang Li, Huijun Qian, Kirk C. Wilhelmsen, Yin Shen, and Yun Li. Smnn: Batch effect correction for single-cell rna-seq data via supervised mutual nearest neighbor detection. *bioRxiv*, 2019.

[254] Zhengyuan Yang, Linjie Li, Kevin Lin, Jianfeng Wang, Chung-Ching Lin, Zicheng Liu, and Lijuan Wang. The dawn of LMMs: Preliminary explorations with GPT-4V (ision). *arXiv preprint arXiv:2309.17421*, 9(1):1, 2023.

[255] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *CoRR*, abs/1809.09600, 2018.

[256] Han-Jia Ye, Hexiang Hu, De-Chuan Zhan, and Fei Sha. Few-shot learning via embedding adaptation with set-to-set functions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8808–8817, 2020.

[257] Xi Ye, Ruoxi Sun, Sercan Ö. Arik, and Tomas Pfister. Effective large language model adaptation for improved grounding and citation generation, 2024.

[258] Chih-Kuan Yeh, Joon Sik Kim, Ian En-Hsu Yen, and Pradeep Ravikumar. Representer point selection for explaining deep neural networks. *CoRR*, abs/1811.09720, 2018.

[259] Kayo Yin and Graham Neubig. Interpreting language models with contrastive explanations, 2022.

[260] Jiahui Yu, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, and Yonghui Wu. CoCa: Contrastive Captioners are Image-Text Foundation Models. *arXiv preprint arXiv:2205.01917*, 5 2022.

[261] Jiahui Yu, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, and Yonghui Wu. Coca: Contrastive captioners are image-text foundation models, 2022.

[262] Mert Yuksekgonul, Federico Bianchi, Pratyusha Kalluri, Dan Jurafsky, and James Zou. When and why vision-language models behave like bags-of-words, and what to do about it? In *The Eleventh International Conference on Learning Representations*, 2023.

[263] Mert Yuksekgonul, Varun Chandrasekaran, Erik Jones, Suriya Gunasekar, Ranjita Naik, Hamid Palangi, Ece Kamar, and Besmira Nushi. Attention Satisfies: A Constraint-Satisfaction Lens on Factual Errors of Language Models. In *International Conference on Learning Representations*, 9 2023.

[264] Fred Zhang and Neel Nanda. Towards best practices of activation patching in language models: Metrics and methods, 2024.

[265] Shuo Zhang, Liangming Pan, Junzhou Zhao, and William Yang Wang. The knowledge alignment problem: Bridging human and external knowledge for large language models, 2024.

[266] Xinchen Zhang, Ling Yang, Yaqi Cai, Zhaochen Yu, Kai-Ni Wang, Jiake Xie, Ye Tian, Minkai Xu, Yong Tang, Yujiu Yang, and Bin Cui. Realcompo: Balancing realism and compositionality improves text-to-image diffusion models, 2024.

[267] Chujie Zheng, Fan Yin, Hao Zhou, Fandong Meng, Jie Zhou, Kai-Wei Chang, Minlie Huang, and Nanyun Peng. On prompt-driven safeguarding for large language models, 2024.

[268] Yiwu Zhong, Jianwei Yang, Pengchuan Zhang, Chunyuan Li, Noel Codella, Liunian Harold Li, Luowei Zhou, Xiyang Dai, Lu Yuan, Yin Li, and Jianfeng Gao. Regionclip: Region-based language-image pretraining. *CoRR*, abs/2112.09106, 2021.

[269] Chen Zhu, Ankit Singh Rawat, Manzil Zaheer, Srinadh Bhojanapalli, Daliang Li, Felix Yu, and Sanjiv Kumar. Modifying Memories in Transformer Models, 2020.

[270] Chen Zhu, Ankit Singh Rawat, Manzil Zaheer, Srinadh Bhojanapalli, Daliang Li, Felix X. Yu, and Sanjiv Kumar. Modifying memories in transformer models. *CoRR*, abs/2012.00363, 2020.

[271] Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, J. Zico Kolter, and Dan Hendrycks. Representation engineering: A top-down approach to ai transparency, 2023.

[272] Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, J. Zico Kolter, and Dan Hendrycks. Representation engineering: A top-down approach to ai transparency, 2023.