

ABSTRACT

Title of Dissertation: **Object-Attribute Compositionality for
Visual Understanding**

Nirat Saini, 2024

Dissertation Directed by: **Professor Abhinav Shrivastava
Department of Computer Science
University of Maryland, College Park**

Object appearances evolve overtime, which results in visually discernible changes in their colors, shapes, sizes and materials. Humans are innately good at recognizing and understanding the evolution of object states, which is also crucial for visual understanding across images and videos. However, current vision models still struggle to capture and account for these subtle changes to recognize the objects and underlying action causing the changes.

This thesis focuses on using compositional learning for recognition and generation of attribute-object pairs. In the first part, we propose to disentangle visual features for object and attributes, to generalize recognition for novel object-attribute pairs. Next, we extend this approach to learn entirely unseen attribute-object pairs, by using semantic language priors, label smoothing and propagation techniques. Further, we use object states for action recognition in videos where subtle changes in object attributes and affordances help in identifying state-modifying and context-transforming actions. All of these methods for decomposing and composing objects and

states generalize to unseen pairs and out-of-domain datasets for various compositional zero-shot learning and action recognition tasks.

In the second part, we propose a new benchmark suite Chop & Learn for a novel task of Compositional Image Generation as well as discuss the implications of these approaches for other compositional tasks in images, videos, and beyond. We further extend insertion and editing of attributes of objects consistently across frames of videos, using off-the-shelf training free architecture and discuss the future challenges and opportunities of compositionality for visual understanding.

OBJECT-ATTRIBUTE COMPOSITIONALITY
FOR VISUAL UNDERSTANDING

by

Nirat Saini

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
2024

Advisory Committee:

Professor Abhinav Shrivastava, Chair/Advisor
Professor Carol Y. Espy-Wilson, Dean's Representative
Professor Ramani Duraiswami
Professor David Jacobs
Professor Pratap Tokekar

© Copyright by
Nirat Saini
2024

To my Parents,
for their endless love and sacrifices, which has shaped my journey.

Acknowledgments

There are not enough words to express my gratitude to the many people who have supported and guided me throughout this journey, including advisors, collaborators, friends, and family, to whom I owe a deep debt of gratitude.

First and foremost, I owe immense and sincere gratitude to my advisor, Abhinav Shrivastava, who has been a guiding force and unwavering support over the last seven years. Coming from a background outside of computer vision, he saw potential in me and offered his trust at a time when I doubted my own abilities. His unwavering faith is something for which I am eternally grateful. I am particularly appreciative of the considerable effort and time he dedicated, and above all, his empathetic support and guidance during the various personal life challenges as well. Although it is challenging to fully express the extent of his influence on my growth in a few sentences, it is clear that he has significantly influenced both my professional and personal life.

I would also like to express my profound gratitude to my initial advisor, Ashok Agrawala, for taking a chance on a young and inexperienced student from a small college in India. His decision to give me the opportunity to thrive at the esteemed University of Maryland, coupled with his constant encouragement to never give up, always push for excellence, and discover my true calling, has had an indelible impact on both my professional and personal life. He has been a pivotal figure in shaping my research career and overall outlook from the moment I arrived here. I consider myself exceedingly fortunate to have been under his guidance, and I cannot fathom what my academic journey would have been like without his influence.

I extend my heartfelt thanks to my thesis committee members, Ramani Duraiswami, Pratap Tokekar, and Ishan Misra, for their enlightening discussions, valuable feedback, kind words of encouragement, and flexibility throughout the entire thesis process. I am also deeply grateful to Carol Espy-Wilson for her intermittent support and guidance throughout my journey.

I am also grateful to other professors and mentors I collaborated with and learned from - especially Pavithr R S for being a wonderful mentor during my undergrad, Hadar Elor and Dima Damen for their valuable insights and constructive criticism about my work. I would also like to thank my internship mentors: Hongchang Wang, Zhe Wu, Rana El Kaliouby, Abhishek Sharma, Taniya Misra, Navaneeth Bodla, Bharat Singh, Xiao Zhang, Ashish Shrivastava and Avinash Ravichandran.

I owe special thanks to my initial collaborators and seniors, Varun, Pallabi, and Khoi, who have played a pivotal role in shaping my understanding of the field and honing my ideation skills. A heartfelt thank you to Saketh, from whom I learned a great deal about technical thinking during our extensive discussions throughout the Covid times. I am deeply indebted to you for the time and effort you invested in me. I am also grateful to all my other collaborators who believed in my ideas and supported me with their utmost efforts: Bo, Gaurav, Hanyu, Kamal, Archana, Vinoj, and Mara.

A big thank you to an amazing lab filled with awesome labmates: Moustafa, Ahmed, Pulkit, Luyu, Soumik, Neha, Max, Matt G., Shirley, Alex, Matt W., Yixuan, Namitha, Hao, Sharath, Saksham, Shishira, Anubhav, Chuong, Daniel, Pravin, and Sonaal. Your camaraderie has been invaluable. Lastly, thanks to Varuni and Abhinav for always being kind and welcoming hosts. The various retreats you organized helped foster a warm and supportive lab environment, cementing numerous collaborations and friendships among my labmates.

I have been incredibly fortunate to be surrounded by a group of supportive friends who have helped me maintain my sanity during some challenging times. These friends have become my family away from home, and I am deeply grateful to: Heba, Alejandro, Snehesh, Devesh, Nishant, Chahat, Arthitha, Anant, Samiha, Chethan, Koutilya, Kanishka, Susmijha, Yogesh, Prithviraj, Anshul, Sunpreet, Hema, Sandeep, Rohini, Rashi, Gaurav, Sankha, and everyone else who has been part of my journey. Special thanks to Bhargavi and Priyan for always being just a phone call away, ready to listen to me vent about my research challenges.

I extend my deepest gratitude to my family and extended family, whose unwavering support was crucial to the completion of this thesis. My heartfelt thanks go to my parents and sister for their endless love and support throughout the years. Mummy and Papa, your sacrifices and encouragement have been the bedrock of my ability to chase my dreams. To my sister, brother-in-law, and Ayera, thank you for being my steadfast support through both the lows and highs, always offering a smile and words of encouragement.

I am grateful to my grandfather, uncles and aunts—Harish, Poonam, Sunil, Renuka, Sunita, Sanjeev—for nurturing my growth and curiosity, helping shape me into an independent thinker. Special thanks to Harsh and Anil Gulati for planting the seed of pursuing an advanced degree and supporting me throughout this journey, from the application process to thesis writing. I would also like to express my heartfelt gratitude to all my cousins for their warmth, camaraderie, and encouragement throughout my journey. Your support has been invaluable.

I also thank my new family, which expanded during my PhD—my father-in-law, mother-in-law, sister-in-law, and grandmother—for their generosity in giving me the time and flexibility to pursue my dream career.

And lastly, but certainly not least, I owe a profound thank you to my husband, Dhruv, for his

selfless sacrifices, quiet patience, steadfast support, and unconditional love. Dhruv, your efforts in keeping me nourished with delicious meals, your tolerance of my unpredictable schedule, your companionship through both fatigue and excitement, and your constant motivation have been invaluable. Thank you for standing by me, even when I doubted myself.

To all of you, thank you for believing in me when I struggled to believe in myself.

I also give thanks to the almighty, a force greater than any of us, who I believe has guided me to this stage in my life.

Credits. I want to thank Air Force (STTR awards FA865019P6014, FA864920C0010), DARPA SAILON program (W911NF2020009), NSF CAREER Award (#2238769) and gifts from Adobe collaboration support fund to Abhinav and Dean's Fellowship which helped me conduct my research.

Table of Contents

Dedication	ii
Acknowledgements	iii
Table of Contents	vii
List of Tables	xi
List of Figures	xv
Chapter 1: Introduction	1
Part I: Attribute-Object Composition for Recognition	2
Part II: Attribute-Object Composition for Generation	6
Part I: Object-Attribute Composition for Recognition	10
Chapter 2: Disentangling Visual Embeddings for Attributes and Objects	11
2.1 Introduction	11
2.2 Related Work	14
2.3 Object Attribute Disentanglement (OADis)	16
2.3.1 Task Formulation	16
2.3.2 Disentangling Visual Features	17
2.3.3 Embedding Space Learning objectives	21
2.4 Experimental Setup	23
2.4.1 Dataset	23
2.4.2 Evaluation	23
2.4.3 Implementation Details	24
2.5 Results	25
2.5.1 Quantitative Results	25
2.5.2 Qualitative results	27
2.6 Ablation Studies	31
2.6.1 To augment or not to augment?	31
2.6.2 Object-conditioned network	32
2.6.3 Choice of word embeddings	32
2.6.4 Values for λ and δ	33
2.6.5 Different weights for losses	34
2.7 Is disentangling and hallucinating pairs helpful?	35

2.8	Is visual disentangling actually happening?	36
2.9	Limitations	37
2.10	Conclusion	37
Chapter 3:	Composition of attributes and objects beyond seen primitives	39
3.1	Introduction	39
3.2	Related Work	43
3.3	Open Vocabulary Compositional Zero-Shot Learning	44
3.3.1	Task Formulation	44
3.3.2	Methodology	45
3.3.3	Neighborhood Expansion Loss	46
3.4	Experimental Setup	49
3.4.1	Dataset	49
3.4.2	Neighborhood list	50
3.4.3	Evaluation	51
3.4.4	Implementation Details	51
3.5	Results	52
3.5.1	Quantitative Results	52
3.5.2	Qualitative Results	56
3.6	Ablation Studies	57
3.6.1	Neighborhood Expansion Loss	57
3.6.2	Evaluating different design choices	59
3.7	Can NEL help in learning abstract concepts?	65
3.8	Limitations	66
3.9	Conclusion	67
Chapter 4:	Attribute-Object Compositionality for Video Recognition	69
4.1	Recognizing actions using object states	69
4.1.1	Introduction	69
4.1.2	Related Work	73
4.1.3	Approach	75
4.1.4	Experimental Setup	79
4.1.5	Results and Ablation	81
4.2	Compositional Action Recognition	89
4.2.1	Introduction	89
4.2.2	Related Works	90
4.2.3	Task Description	91
4.2.4	Experimental Setup	92
4.2.5	Results and Ablation	93
4.3	Conclusion	95
Part II:	Object-Attribute Composition for Generation	97
Chapter 5:	Chop & Learn Benchmark Suite	98
5.1	Introduction	98

5.2	Related Works	100
5.3	Chop & Learn	102
5.3.1	Design Choices	103
5.3.2	Data Collection Setup	105
5.4	Scope and Limitations	107
Chapter 6:	Compositional Image Generation for unseen Attribute-Object pairs	109
6.1	Introduction	109
6.2	Related Work	110
6.3	Compositional Image Generation	111
6.3.1	Task Description	111
6.3.2	Methods	112
6.4	Experimental Setup	114
6.4.1	Dataset Splits	114
6.4.2	Evaluation	115
6.5	Results	117
6.5.1	Quantitative Results	117
6.5.2	Qualitative Results	119
6.6	Ablation Studies	120
6.6.1	Green Screen Removal	120
6.6.2	Number of views	123
6.7	Conclusion	123
Chapter 7:	Consistent Attribute-Object Insertion in Videos	124
7.1	Introduction	124
7.2	Related Works	126
7.2.1	Conditional video generation and editing	126
7.2.2	Adapting Image Models for Video tasks	127
7.3	Preliminaries	128
7.3.1	Diffusion models	129
7.3.2	Latent Diffusion Models (LDM)	129
7.3.3	DreamBooth-LoRA based fine-tuning	130
7.3.4	DDIM Inversion	130
7.4	InVi: Inserting consistent Attribute-Object pairs in a Video	131
7.4.1	Generating the first-frame and pre-processing	131
7.4.2	Temporally Consistent Frame Inpainting	133
7.4.3	Post-processing	135
7.5	Experiments	136
7.5.1	Baselines	137
7.5.2	Quantitative Evaluation	138
7.5.3	User Study	139
7.5.4	Qualitative Evaluation	139
7.5.5	Ablation: Advancing Beyond TokenFlow for Inpainting Tasks	141
7.6	Conclusion	143

Chapter 8: Conclusion and Discussion	145
Chapter 9: Future Work	148
9.1 Future directions using Chop & Learn	148
9.1.1 Green Screen augmentation	149
9.1.2 3D reconstruction	151
9.1.3 Long-term Video Parsing	151
9.1.4 Intermediate state generation and Video Interpolation	152
Chapter 10: Appendix	155
Chapter 2: Disentangling Visual Embeddings for Attributes and Objects	155
10.1 Benchmark Details	155
10.1.1 Dataset issues of C-GQA.	156
10.1.2 Creating Dataset: VAW-CZSL.	156
10.2 Negative Impact of OADis	158
Chapter 3: Composition of attributes and objects beyond seen primitives	158
10.3 Benchmark Details	158
10.3.1 Dataset Splits	158
10.3.2 Dataset Creation	159
10.3.3 Limitation for Open-world evaluation	161
Chapter 4: Attribute-Object Compositionality for Video Recognition	162
Recognizing Actions using object states	162
10.4 Dataset Splits	162
10.4.1 Long-tail Distribution	163
10.4.2 Multiple splits	164
10.4.3 Discrepancy in labels	167
10.5 Experimental Setup	167
10.5.1 State detection	167
10.5.2 Model and Parameters	170
Compositional Action Recognition	170
10.6 Benchmark Details	170
Chapter 6: Compositional Image generation for unseen Attribute-Object pairs	172
10.7 Questionnaire for user study	172
Chapter 7: Consistent Attribute-Object Insertion in Videos	173
10.8 Questionnaire for user study	173
Bibliography	178

List of Tables

2.1	Dataset Details: This table shows the statistics for different datasets and their splits. The proposed VAW-CZSL benchmark significantly increases the number of attributes and objects.	23
2.2	We show results on MIT-states [76] and UT-Zappos [195]. Following [118, 135], we use AUC in % between seen and unseen compositions with different bias terms, along with Val, Test, attribute and object accuracy. HM is Harmonic Mean. OADis consistently outperforms on most categories with significant increment. . .	27
2.3	We show results on VAW-CZSL. Since it is a much more challenging dataset, with significantly large number of compositions, to discriminate performance among different baseline, we show top-3 and top-5 AUC (in %) for Val and Test sets. . .	27
2.4	Results with different networks for word-embeddings. Object-conditioning with attribute performs the best, and is therefore used for OADis (Section 2.6).	32
2.5	Results with pre-trained word-embeddings. GloVe [129] performs the best, and is therefore used for OADis. (Sec 2.6.3)	33
2.6	Results with pre-trained word-embeddings. GloVe [129] performs the best, and is therefore used for OADis. (Sec 2.6.4)	34
2.7	We show empirical weights of each loss function in this table. (Sec 2.6.5)	35
2.8	We quantitatively show that the proposed architecture and different losses help in disentanglement and composition of unseen pairs. The experiments are conducted on MIT-States [76], where change in accuracy is shown with green and red based on increment or decrement respectively from the previous row. A dash (-) represents no change more than (± 0.1). Refer to Section 2.5.1 for details. . .	35
3.1	Dataset Splits. We denote $*$ for unseen concept, such that A and O are seen attribute and objects, whereas A^* and O^* denote unseen attributes and objects. AO and $(AO)^*$ are seen and unseen compositions or seen attributes and seen objects respectively. AO^* are seen attribute-unseen object pairs, A^*O are unseen attribute-seen object set and A^*O^* are the unseen attribute-unseen object pairs. We propose new benchmark splits for OV-CZSL on datasets MIT-states [76], C-GQA [118] and VAW-CZSL [131, 147].	50
3.2	Results on MIT-states [76] and CGQA [106]. We report Top 1 AUC, which balances % between seen and unseen compositions with different bias terms. HM is Harmonic Mean where maximum AUC is computed. Following [135], best accuracy values are reported for Seen AO and Unseen pairs $\{AO^*, A^*O, A^*O^*, (AO)^*\}$. All other accuracies for individual splits are computed with bias where HM is maximum. Our method with NEL loss outperforms previous methods on most unseen compositions.	53

3.3	Results on VAW-CZSL [131, 147]. All measures are shown in Top 3 AUC. HM is Harmonic Mean where maximum AUC is computed. All other accuracies for individual splits are computed with bias where HM is maximum. Our approach outperforms previous baselines for unseen compositions.	54
3.4	Comparison with ZSL baselines. We show results on two ZSL baselines for seen and unseen attributes and objects. Using NEL significantly improves outperforms existing ZSL baselines.	55
3.5	Using NEL with other baselines. We show effect of NEL for different baselines. All methods using NEL perform better for OV-CZSL splits.	60
3.6	Ablation for varying number of neighbors (k). We show how changing the number of neighbors can introduce noisy labels in the setup. For generalization to unseen classes, we need fewer neighbors (5 as shown here).	60
3.7	Ablation on different Text Embedding. We show effect of using different text embeddings for OV-CZSL. We observe that BERT [31] improves the unseen compositions A^*O^* the most, hence we use BERT [31] for our approach.	61
3.8	Ablation for different smoothing factors: We set different smoothing factors α for $\mathcal{L}_{AO}^{NE}$ with dataset MIT-states [76], to empirically find the best value for α	62
3.9	Ablation for different smoothing factors: We set different smoothing factors α and T for $\mathcal{L}_{AO}^{NE}$ with dataset MIT-states [76], to empirically find the best value for α and T	64
3.10	Ablation for weights for Neighborhood Expansion Loss: We set different β_1 , weights for neighborhood expansion loss for pair embeddings, for MIT-states [76] split 1. We show that using higher value of β_1	65
4.1	Quantitative results on <u>Unseen Objects</u>. This table summarizes results for actions on objects not seen during training. (test set). All values are shown as mean with confidence interval (95%) over 5 splits. In the table we highlight the best and second best result in bold+underline and underline respectively.	83
4.2	Quantitative results on <u>Seen Objects</u>. This table summarizes results for seen object actions (validation set). All values are mean with confidence interval (95%). In the table we highlight the best and second best result in bold+underline and underline respectively.	83
4.3	Ablation analysis on <u>Seen Objects</u>. We show how our amplifiers and separate modeling of object-transforming and context-transforming actions help in object-centric action recognition. All the experiments are done on split1. In the table we highlight the best and second best result in bold+underline and underline respectively.	88
4.4	This table shows analysis of different temperature λ variable for the experiments. Although, changing λ does not significantly impact the overall accuracy, we optimize for mean classwise accuracy, which is maximum for $\lambda = 100$	88
4.5	Compositional action recognition results. “Start/End” denote the prediction results for the initial and the final state composition with the corrected object type. Bold and <u>underline</u> represent the top-1 and top-2 results.	92

4.6	Other splits: We also present other possible splits of data. All the results are using I3D [17] pre-trained features along with one layer Transformer [169] model. Comp. represents the initial: object-initial state composition and final: object-final state composition results for each split.	94
4.7	Results of finetuning R3D [60] backbone. “Start/End” denotes the prediction results for the initial and the final state composition with the correct object type. .	94
5.1	Comparison with other video datasets. This table highlights the distribution of the objects, states and compositions in different datasets. Obj. refers to objects, Comp. is compositions of objects and styles, N refers to the number of compositions that have more than 10 samples, and Styles* refers to grouping of styles: instead of generic names like cut, chop, etc., we use 3 distinct styles (chop/dice, peel, grate) as styles. MIT-States [†] is the only image-based dataset, the rest are video-based datasets. All these data numbers are for edible objects and cutting style actions from respective datasets. Our dataset has uniform distribution for each metric in the table, which makes it suitable for learning objects and their states.	101
6.1	Compositional generation evaluation. FID, user scores, and classifier scores of various generative models. User Realism is on a scale of 1-5. (★) denotes that accuracies are evaluated on a seen data split. Bold represents the best result. . . .	117
6.2	Green screen removal evaluation Both rows employ the SD+FT+TI but are trained using images with varying backgrounds. Classifiers specific to each dataset are trained to assess Classifier Acc. Validation images used to calculate Patch FID differ between the two rows. Patch FID Lower Bound is computed by evaluating the patch FID on one-half of the validation images relative to the other half. For further details, refer to Section 6.5.2.	122
6.3	Number of views ablation results.	123
7.1	Quantitative Results for object swapping (on the left) and object insertion (on the right). Evaluation for background consistency, temporal appearance consistency, and alignment with prompts.	138
10.1	Exempt Classes which match the categories in Imagenet. To ensure that OV-CZSL setup is truly Zero-shot Learning (ZSL) for attributes and objects, we must include all common attributes and objects seen in ImageNet [30] in the seen attributes and object sets. Hence, while creating seen attributes and objects sets for MIT-States, we first find out the objects and attributes in MIT-states [76] which are already present in ImageNet [30], and make sure these categories are part of seen attributes and objects.	159
10.2	To create five splits of the dataset, firstly, all actions are divided into the majority, minority, and mid-sized classes based on the number of samples. This table shows all actions and which class do they belong to.	165

10.3	This table shows the distinct context-transforming actions used in each split. Note that since we have fewer number of object-transforming actions, all those actions are part of each split. Combination of all these 5 splits cover total 30 context-transforming actions. We choose split classes randomly from all the classes, while making sure 3 majority, 4 minority and 6 mid-sized classes are in each split.	166
10.4	This table shows analysis of different temperature λ variable for the experiments. Although, changing λ does not significantly impact the overall accuracy, we optimize for mean classwise accuracy, which is maximum for $\lambda = 100$	166

List of Figures

1.1	<i>Objects can evolve into different states:</i> When we think about an apple, we imagine the most common appearance of a red apple is shown on the left. However, an apple can be green, chopped, sliced, peeled, caramelized, dried, baked or pureed. Most of these states of an apple can be evolved by performing certain actions during cooking process.	1
2.1	Method illustration: Given an input image I of peeled apple, we use two other images: (1) one with same object, different attribute I_{obj} - sliced apple, (2) one with same attribute, different object I_{attr} - peeled orange. We propose a novel architecture that takes I and I_{attr} , and extracts their visual similarity features for peeled and visual dissimilarity features for orange. Similarly, using I and I_{obj} , the visual similarity features for apple, and the dissimilarity features for sliced can be extracted. We compose these primitive visual features to hallucinate a seen pair peeled apple, and a novel unseen pair sliced orange to be used for regularizing our embedding space. Note that this is a visualization of embedding space composition, we do not generate images.	12
2.2	System Overview: Given an image I , for peeled apple, we consider two images:, one with same object: I_{obj} , sliced apple, and one with same attribute, I_{attr} peeled orange. (1) The Object-Conditioned Network composes pair word embedding, using GloVe word embeddings for labels. (2) Label Embedder uses the image I and embeds visual feature $v_{\text{attr,obj}}$ along with word embedding $w_{\text{attr,obj}}$, using loss $\mathcal{L}_{\text{cls}}$. (3) Attribute Affinity Network and Object Affinity Network, disentangles the same attribute and object from the pair of images I, I_{attr} and I, I_{obj} respectively. Disentangled visual features for peeled (v_{attr}) and apple (v_{obj}) are used along with word embeddings of attribute (w_{attr}) and objects (w_{obj}), to compute $\mathcal{L}_{\text{attr}}$ and $\mathcal{L}_{\text{obj}}$. (4) Using disentangled features, we compose seen pair peeled apple ($v_{\text{attr}}, v_{\text{obj}}$) and unseen pair sliced orange ($v'_{\text{attr}}, v'_{\text{obj}}$), for composition losses $\mathcal{L}_{\text{seen}}$ and $\mathcal{L}_{\text{unseen}}$	13

2.3	(a) Attribute Affinity Module: We compute the cosine similarity between blocks in f and f_{attr} (S in Eq. 2.3), then apply row-wise and column-wise softmax ($\mathcal{A}$ and $\mathcal{A}'$), followed by a respective column-sum and row-sum to obtain $\mathbf{m}$ and $\mathbf{m}_{\text{attr}}$. $\mathbf{m}$ represents regions where f_{attr} is highly similar to f (hence, we reshape and multiply $\mathbf{m}$ with f_{attr}) and $\mathbf{m}_{\text{attr}}$ represents regions where f is highly similar to f_{attr} (thus, $\mathbf{m}_{\text{attr}} \cdot f$). Similarly, S' represents the regions where feature f_{attr} is not similar to feature f (more details in Section 2.3.2). The last row shows real samples and generated attention maps overlayed on images. Give image <code>ruffled bag</code> and <code>ruffled flower</code> , we show that attribute <code>ruffle</code> is highlighted in the center $\mathbf{m}_{\text{attr}} \cdot f$ and $\mathbf{m} \cdot f_{\text{attr}}$. Whereas, $\mathbf{m}'_{\text{obj}} \cdot f_{\text{attr}}$ shows the dissimilar regions of I_{attr} w.r.t I . (b) Shows the three embedding spaces learnt with different losses. Same notation is used as Figure 2.2.	17
2.4	Qualitative Results: We show the nearest neighbors using the hallucinated unseen composition features for MIT-states and UT-Zappos. Although, all the neighbors are not correct (represented with red outline), they look very similar to true class labels: (a) First row: pureed fruit , Second row: engraved coin , Third row: huge tower . (b) We show top-3 predictions for images in VAW-CZSL.	28
2.5	We show the top 5 nearest neighbors using the hallucinated unseen composition features for UT-Zappos. All the neighbors with correct labels are represented by green , whereas incorrect ones are represented with red outline.	29
2.6	(a) Failure Cases: Shows the image pairs, f and f_{attr} , and the similarity and dissimilarity map overlayed (details in Sec 2.6). Moreover, we show for some cases for MIT-States, the examples are very vague or incorrect to actually capture attribute and object concepts separately. For instance, in <code>clear lake</code> and <code>clear sky</code> , it is very difficult to distinguish lake and sky. Hence the similarity and dissimilarity maps do not perform very well. Other examples are also of failure cases where the overlayed similarity and dissimilarity maps do not make sense. (b) Correct Examples: This shows some good examples, where the similarity and dissimilarity maps capture the attributes and objects correctly for MIT-States.	30
2.7	Correct Examples: We show the similarity and dissimilarity attention maps overlayed on images for VAW-CZSL as well. To re-iterate, for images with features f and f_{attr} , $\mathbf{m}_{\text{attr}} \cdot f$ shows how the regions in f which are most similar to f_{attr} , and $\mathbf{m} \cdot f_{\text{attr}}$ shows the regions in f_{attr} which are most similar to regions in f . Lastly, $\mathbf{m}'_{\text{obj}} \cdot f_{\text{attr}}$ shows the regions of f_{attr} which are most dissimilar to f	30
2.8	We show the different networks used on top of word embeddings. Empirically and following our intuition, Object-Conditioned network works best among the three (rest two are Linear and MLP). (Sec 2.6)	33
2.9	Qualitative results showing top 3 attributes and objects from test images, using prototype disentangled features computed on training data.	36

3.1	We propose a new task, Open Vocabulary-Compositional Zero-shot Learning (OV-CZSL), that expands upon Compositional Zero-shot Learning (CZSL). CZSL focuses on evaluating unseen compositions ripe lemon, peeled apple of seen attributes (ripe, peeled) and seen objects (apple, lemon), using training samples (ripe apple, peeled lemon). The novel OV-CZSL is trained with the same data as CZSL. However, it can be evaluated on unseen attributes (sliced), objects (potato) and their unseen compositions. These unseen compositions also include seen attribute-unseen object (peeled potato), unseen attribute-seen object (sliced lemon) and unseen attribute-unseen objects (sliced potato).	40
3.2	System Overview: ResNet18 [65] is used as visual feature extractor. Label Embedder embeds the image feature $v_{\text{attr,obj}}$ along with BERT [31] textual feature for the pair $w_{\text{attr,obj}}$. Object-Attribute Disentanglement module separates visual features into attributes and objects as v_{attr} and v_{obj} . Moreover, along with target labels for attribute w_{attr} , object w_{obj} , and pair $w_{\text{attr,obj}}$, we use neighbors for all of these from the semantic embedding space (N_A , N_O and N_{AO} respectively). We use cosine similarity loss, along with novel <i>Neighborhood expansion loss</i> to solve the task of OV-CZSL.	45
3.3	<i>Neighborhood Expansion Loss.</i> In the pair embedding space, we embed visual feature $v_{\text{attr,obj}}$ close to $w_{\text{attr,obj}}$. Further, we use $k = 5$ neighbors of correct label $w_{\text{attr,obj}}$, and make their word embeddings closer to the visual embedding $v_{\text{attr,obj}}$ as well. This gives our model the ability to generalize to unseen compositions.	48
3.4	Qualitative Results: We show results on MIT-States [76] for different sets, examples with top 3 predictions for an image. Most incorrect labels are also from unseen compositions Y^u , and although capture the attributes in the images, but are not considered correct. Labels in green are the correct annotation. We also observe redundancy in labels, such as row 1, column 2 blunt blade in $A*O$. Top prediction also includes blunt sword. All swords have blades. However, having separate labels for these makes the task harder.	57
3.5	Qualitative results for Cross-Dataset Nearest neighbors. (a) We show the 5 nearest neighbor images from VAW-CZSL [131, 147] dataset, using the composition features from model trained on MIT-states [76]. Similarly, (b) shows that if model is trained on VAW-CZSL [131, 147], it's nearest neighbors in feature space for various unseen compositions make sense, when retrieved from MIT-sates [76].	58
3.6	To show the impact of NEL, we cluster all attribute and object class labels for MIT-States [76], and evaluate the accuracy for each cluster. These clusters have both seen (ancient, new, young) and unseen (modern, old) concrete concepts, such that each cluster has one abstract concept (time/age). Our model (OADis+NEL) shows either same or improved performance for attributes and object clusters, backing our hypothesis of NEL generalizes to unseen concrete concepts for the similar abstract concepts.	66
4.1	<i>Match the following:</i> In the given set of image pairs, we show start→end transformations in object states. Can you match the object states to an action, from list given below, that likely induced those transformations?	70

4.2	Our framework extracts start and end object states from the video. We use ResNet50 [65] pre-trained on ImageNet [30] as our backbone network. We introduce a novel framework to model object-transforming and context-transforming actions separately. The affinity amplifier is trained to learn what is similar between object states and the differential amplifier captures the dissimilarities between the states.	73
4.3	The architectural details of the (a) Affinity Amplifier and (b) Differential Amplifier.	75
4.4	We show qualitative results for Epic-Kitchen 100 [26]. Objects are found using the same object detector as for our model. Among the shown actions, three are object-transforming: empty, fill, open, and two are context-transforming actions: fold, wash. Here are the row-wise results explained for each action (left to right): Row1: <i>empty bowl, empty sink, empty pot, empty bottle</i> Row2: <i>fill kettle, fill glass, fill bottle, fill cup</i> , Row3: <i>fold bag, fold dough, fold pack, fold bag</i> , Row4: <i>open milk, open lid, open fridge, open microwave</i> , Row5: <i>wash spatula, wash courgette, wash knife, wash plate</i>	85
4.5	We show qualitative results for MIT states [76]. There are only two common classes cut and peel. Further, some of the objects are common with our training transformation states. Hence we show seen and unseen object transformation sets separately. We show top 5 predictions for each image pair. For seen transformations, we show results for Row1: <i>cut chicken, cut lemon, cut meat, cut cheese</i> Row2: <i>peel potato, peel garlic, peel banana, peel garlic</i> , Row3: <i>cut pear, cut salmon, cut apple, cut beef</i> , Row4: <i>peel tomato, peel apple, peel orange, peel fruits</i> .	86
4.6	We show qualitative results for YouCook2 [207]. We use oracle detector to find bounding boxes for this dataset, besides using sparse ground truth bounding box annotations. We show that our model generalizes on unseen object actions. Among the shown actions, close, cut and peel are object-transforming actions and add and mix are context-transforming actions. Here are the row-wise results explained for each action (left to right): Row1: <i>add salt, add onions (unseen), add salt, add egg yolk</i> Row2: <i>close pot, close cooker (unseen), close grill (unseen), close grinder</i> , Row3: <i>cut sandwich, cut spring onion, cut tofu, cut cabbage</i> , Row4: <i>mix milk, mix meat, mix pepper(unseen), mix flour</i> , Row5: <i>peel avocado, peel raddish, peel squid (unseen), peel carrot</i>	87
4.7	Compositional Action Recognition: training videos are used to recognize objects along with transition from state1 $\rightarrow$ state2 , to generalize on recognizing unseen object-state transitions.	90
5.1	We present Chop & Learn (ChopNLearn) , a new dataset and benchmark suite for the tasks of Compositional Image Generation and Compositional Action Recognition. It consists of 1260 video clips and 112 object state combinations captured from multiple viewpoints for 20 objects and 8 cut styles. We also propose two new compositional tasks and benchmarks - (1) Image Generation: given training images of various objects in various states , the goal is to generate images of unseen combinations of objects and states . (2) Action Recognition: training videos are used to recognize objects along with transition from state1 $\rightarrow$ state2 , to generalize on recognizing unseen object-state transitions.	99

5.2	<u>Left</u> : We show examples of cutting styles from popular video datasets (VISOR [29]: chop and peel potato, Youcook2 [207]: chop broccoli, peel radish), image dataset (MIT-states [76]:slice pear, peel orange) and generation pipelines (DALL-E [139]:baton cut apple, half round slices tomato). Most of these are either too noisy to capture subtle differences in objects or do not have the granularity of specific cutting styles. <u>Center</u> : Our 4 camera setup captures videos of one object in 4 different views. <u>Right</u> : We capture 8 styles of object states, which can be derived in a hierarchical manner from larger to small cuts. Each style is of different shape and granularity.	102
5.3	Statistics for ChopNLearn : We show the number of samples for each object-style composition in a color-coded manner: orange represents 12 samples, green represents 8 samples and blue represents 4 samples.	106
5.4	(a) The clip length distribution for one camera (315 unique clips). (b) Preliminary results of using green screen to augment the dataset with different backgrounds. We continue to improve the transfer results by adding shadows and background matting.	106
5.5	Confusion matrix for generation & action-based tasks.	108
6.1	Given training images of various objects in various states , the goal is to generate images of unseen combinations of objects and states	109
6.2	Dataset Split used in The Compositional Image Generation Task . Training compositions and test compositions are marked with orange and teal, respectively. Unmarked compositions are not included in our dataset.	115
6.3	Compositional Generation Samples . Ground Truth (GT) real images are shown in the first row for reference. Seven object-state combinations in the test set are displayed, each with two generated samples for each method. Please zoom in to see details.	119
6.4	Additional Compositional Generation Samples Using Training Compositions Ground Truth (GT) real images are shown in the first row for reference. Seven object-state compositions in the training set are displayed, each with four generated samples for each method. Please zoom in to see the details.	120
6.5	Additional Compositional Generation Samples Using Test Compositions Ground Truth (GT) real images are shown in the first row for reference. Seven object-state compositions in the test set are displayed, each with four generated samples for each method. Please zoom in to see the details.	121
6.6	Preliminary results of using green screen to augment the dataset with different backgrounds. We continue to improve the transfer results by adding shadows and background matting.	122
7.1	InVi inserts objects into a background video using a foreground mask, a control signal (e.g., pose, canny, depth map), and a text prompt by leveraging off-the-shelf diffusion models. It ensures that the inserted object aligns semantically with the text, is temporally coherent in time, and also conforms spatially to the control signal.	125

7.2	InVi [149] inference pipeline: (a) Given a video and object bounding boxes, first, we crop a region around the bounding box which is inpainted. (b) Next, we use a ControlNet-based inpainting diffusion model to inpaint the cropped region in the first frame. (c) To ensure temporal consistency when inpainting subsequent frames, we employ the previous frame as an anchor image. This is achieved by adapting the self-attention block of the denoising U-Net with extended attention. Specifically, we augment the Keys and Values of the current frame being inpainted with those of the anchor frame, allowing for consistent appearance. Finally, the inpainted crop is seamlessly integrated back into the input video.	132
7.3	Post-processing to remove flickering square artifacts. a) Background image. b) Initial image generated from our pipeline. c) Zoomed-in view revealing artifacts around the inserted object. d) A trimap is generated to facilitate seamless blending of the object into the background. e) Post-processed frame showcasing the final result after blending the inserted object with the background.	135
7.4	Qualitative results. The first image is a background frame from the video undergoing inpainting. Subsequent frames depict the video with the inserted object. . .	137
7.5	User Preference Study: InVi Outperforms Baseline Methods in text alignment, background and temporal appearance consistency and overall video quality. . . .	139
7.6	In our qualitative comparison, we contrast the performance of InVi with three baseline methods: FateZero, TokenFlow, and Tune-a-video. FateZero frequently diverges from the editing prompt, as seen in the woman running example. Meanwhile, both TokenFlow and Tune-a-video unintentionally modify the background. InVi, however, consistently yields results that closely align with the editing prompt while preserving the background.	140
7.7	Ablation experiments: We make simple changes to the baseline methods. Frm+Inp conducts frame-wise inpainting using a constant seed and prompt. Tokenflow preserves the exact structure of the original jeep (like preserves the grills and mostly changes the color). TokenFlow+Inp combines ControlNet along with an inpainting method, serving as a baseline for inpainting, but leads to blurry results. TokenFlow+Inp (No Flow) removes the nearest-neighbor field computation from Tokenflow, and keeps the sliding window based inpainting of 2 frames at a time. Finally, InVi, which surpasses these methods in terms of clarity, consistency, and sharpness, establishing itself as the preferred choice for inpainting tasks.	142
9.1	We show different uses of our dataset. The first column shows the raw images. The second column shows basic Python based green screen removal techniques on the dataset. The third column uses generated kitchen chopping board images replacing the green screen using the segmented object pieces. The fourth column shows results of training stable our benchmark SD+TI+FT model with the images without a green screen. Further the rightmost four columns show promising results towards 3D reconstruction of deformable objects, which can be a potential future research problem ChopNLearn can be used for.	149

9.2	Video parsing graph: For a given video, we use Grad-CAM[152] on the intermediate frames to identify and visualize the class activation maps corresponding to the most salient states. Top: A training video clip has one transition of orange from large cut $\rightarrow$ small cut. Bottom: We can learn single transitions from training data, to generalize transitions in a long video with multiple state changes and parse the video as a graph.	152
9.3	Intermediate object state generation: Textual conditioning is still limited to sparse and independent frame generation. However, using a slider control for generating intermediate states of cut between two well known states is an open problem.	153
9.4	Frame Interpolation: Dense frame interpolation with start and end frames is also a new problem which ChopNLearn can support.	153
10.1	We show some of the misclassified actions from the test set for EK-55. The action in green is the actual label for the pair, whereas our model predicts it incorrectly shown in red. The first action is “break stock” since there is water in the pan, our model predicts it to be “fill” instead, next is “open avocado”, since the avocado is split into two pieces, our model predicts as “cut”. Similarly “divide tomato”, predicted as “close” and “fill box” predicted as “open”. This shows our model works for actions that have clear distinct before and after states, which are different from other action states.	163
10.2	Distribution of Dataset: This histogram shows the number of samples for training, validation and test set. Our dataset is long-tail distribution, as observed in the graph. We show object-transforming and context-transforming actions with different colors (orange and blue respectively).	165
10.3	Our Pipeline uses object detector as a pre-processing step. We show that usually objects occupy a small area in the image, hence it is important to capture the object in order to understand the object state. Some examples here show the ratio of area occupied by the objects in the frames.	168
10.4	We show some qualitative results of our detector. Note that our detector is trained on the dataset, and hence captures diverse states and textures of objects.	169
10.5	We show our detector misfires for certain objects qualitatively and do not capture the object at all. In these images, the before and after state of the objects is visible. It can be observed that sometimes, the detector finds the object in one state but does not capture the object at all in the other state. Here are the row-wise results explained for each action (left to right):Row1: <i>close bag (no bag visible in both states), close bag (bag visible partially), close bag (different bags visible), close bin (bin partially visible)</i> Row2: <i>crush coffee (coffee not visible in end state), crush ginger (not visible in end state), crush garlic (no change in states), crush pepper (not visible)</i>	169
10.6	Examples of images from the test set and samples from the generative models presented to participants in the user study	173
10.7	Snapshot of questionnaire presented to participants of the user study	174

10.8	Survey Preview for Object swapping videos. The users are shown 4 videos along with source video and prompt used for editing, to answer questions about visual quality, text alignment and background consistency.	175
10.9	Survey Preview for Object Insertion videos. The users are shown 3 videos along with source video and prompt used for editing, to answer questions about overall visual quality, text alignment and temporal consistency.	176

Chapter 1: Introduction

Knowing “what causes what” makes a big difference in how we act.

Judia Pearl

Objects in the real world can change appearances over time, i.e., changes in color, shape, material, size, etc. The evolution of visually discernible properties, also referred to as *states* or object *attributes* can sometimes takes few days (*e.g.* for a fruit can change it’s colors and texture from raw to ripe states), or can happen in a matter of seconds (*e.g.* apple can be cut into slices). Some examples of different object states are shown in Figure 1.1.

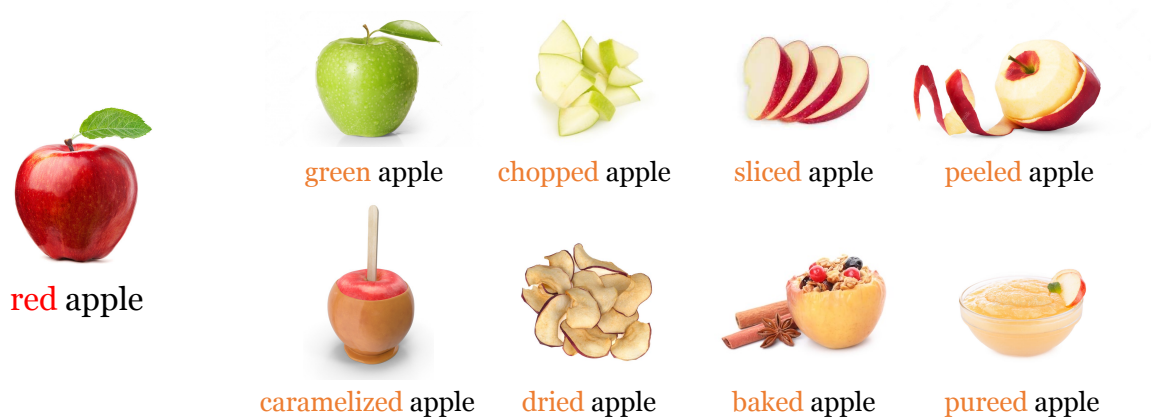


Figure 1.1: *Objects can evolve into different states*: When we think about an apple, we imagine the most common appearance of a red apple is shown on the left. However, an apple can be green, chopped, sliced, peeled, caramelized, dried, baked or pureed. Most of these states of an apple can be evolved by performing certain actions during cooking process.

Humans inherently utilize *causal reasoning* over time to identify and comprehend these changes

in objects, discerning the causes and effects of such transformations. This capability to recognize subtle variances in object states is increasingly crucial in modern vision applications, underscoring the importance of understanding the dynamics of object attributes. As Judea Pearl articulates in his seminal work, *The Art and Science of Cause and Effect*, a profound comprehension of actions necessitates an understanding of the causal factors that precipitate these actions.

In pursuit of generalizing our understanding to novel scenarios, we often employ a context-driven, bottom-up approach to decipher the components of any scene or unfamiliar object. Drawing inspiration from this causal framework, this thesis explores the compositionality of attributes and objects in the construction of visual models, aiming to enhance the way computational systems interpret and interact with the evolving visual world.

This thesis is structured around two principal visual models utilized for the recognition and generation of visual data. Part I introduces strategies for harnessing the compositionality of attributes and objects to classify unseen pairings. Part II focuses on harnessing composition of attribute object pairs in image generation and consistent insertion in videos. Problem setups for each chapter and their key insights are mentioned below.

Part I: Attribute-Object Composition for Recognition

Object recognition and classification remain fundamental challenges in visual perception. Despite significant advancements in supervised learning that have profoundly enhanced object detectors, these models often fail to generalize to unfamiliar objects in varying contexts. Traditional approaches to object recognition and tracking frequently encounter difficulties in adapting to new object states due to a lack of comprehensive training data that encompasses all conceivable

states an object may assume. Similarly, many video analysis models do not adequately recognize or adapt to subtle changes in object states, changes that are often the byproducts of specific actions. In contrast, this thesis adopts a novel perspective by examining objects and their attributes as additive, transferable functions that extend to previously unseen objects. Additionally, we emphasize the importance of recognizing minute changes at a detailed level within objects. Such granularity improves the capabilities of image recognition and video parsing models, thus enhancing their ability to accurately interpret the underlying actions causing these changes.

Chapter 2 delves into the making generalizable models for object and attribute recognition. Attributes or states are seen as transferable properties that can apply across different objects—for example, both an apple and a pear can be chopped or peeled. The sheer diversity and combinations of object-attributes present in the real world pose a significant challenge for annotating all possible attributes for all objects, which is necessary for training effective object detectors and recognition models. To address this, prior research has employed compositional learning, which involves using existing knowledge of attributes and object concepts to recognize new attribute-object compositions. Datasets such as MIT-States [76] and UT-Zappos [195] are frequently utilized to explore these dynamics, focusing on joint attribute-object recognition for a diverse, albeit limited, array of objects and attributes. These studies predominantly leverage linguistic embeddings of attributes and objects, which are beneficial because they (1) provide a distinct separation between attribute and object within the embedding space, and (2) contain inherent semantic knowledge about similar objects and attributes, aiding in their composition. However, unlike linguistic embeddings, distinguishing between object and attribute within visual embedding spaces poses considerable challenges.

In Chapter 2, we demonstrate our approach’s ability to disentangle objects and attributes

within the visual feature space [147]. This capability is crucial for conceptualizing novel, complex concepts and for enhancing the accuracy and robustness of object-attribute recognition models in Compositional Zero-Shot Learning (CZSL). Through comprehensive experimentation, we illustrate the superiority of our method over previous approaches across three distinct benchmarks. Additionally, we introduce a new benchmark tailored for the CZSL task featuring images of objects in natural settings. This benchmark is designed to facilitate a deeper understanding of more complex scenes, showcasing the practical applications and effectiveness of our methodology in recognizing diverse object-attribute compositions in the wild.

So far, we are equipped to distinctively isolate and analyze the visual features of objects and attributes, enabling the composition and recognition of unseen object-attribute pairs. However, the framework of compositional zero-shot learning (CZSL) inherently assumes that all attributes and objects have been encountered at least once during the training phase, with the model subsequently tested on novel combinations of these previously observed attributes and objects. While this scenario effectively assesses a model’s capacity for compositional generalization, it falls short of practicality, as it does not account for the simultaneous occurrence of entirely unseen attributes, objects, and their combinations—a dilemma that remains unresolved within the field.

In Chapter 3, we introduce a novel task, termed *Open Vocabulary-Compositional Zero-shot Learning (OV-CZSL)*, designed to tackle these complex challenges concurrently [150]. This task extends the traditional CZSL framework to not only include unseen combinations of known attributes and objects but also to encompass attributes and objects that have never been observed before. This approach allows for a more comprehensive evaluation of a model’s generalization capabilities across completely novel scenarios. To achieve this, we integrate language priors to broaden the conceptual reach of existing CZSL methodologies, supplemented by advanced

techniques such as label smoothing and propagation to formulate a new training loss paradigm. We introduce the Neighborhood Expansion Loss (NEL), which leverages the relationships among seen and unseen attributes, objects, and their compositions. The NEL is designed to bridge the gap between known and unknown entities, facilitating the recognition of entirely new object-attribute pairs. This innovative approach not only enhances the robustness and flexibility of CZSL models but also sets a precedent for future explorations into the recognition of uncharted visual phenomena.

Expanding our focus beyond static images, the subsequent Chapter 4 delves into the significance of attributes and the dynamic evolution of object states within visual understanding tasks, particularly evident in video contexts. As previously highlighted, objects can undergo rapid transformations, altering their state within seconds, as commonly observed in cooking videos where actions such as chopping transform whole items into pieces, contributing to a larger goal like salad preparation. The ability to recognize these actions or to identify changes within the scene or object are fundamentally interconnected; understanding one aspect inherently enhances comprehension of the other.

While the majority of prior research in action recognition has adopted a conventional approach [11, 17, 189, 201, 205], focusing on detecting actions through the analysis of multiple frames and temporal modeling, these studies often overlook the explicit examination of changes in object states. In contrast, our approach places a significant emphasis on these frequently disregarded yet critical object states and the actions that precipitate their transformation. Utilizing our object and attribute recognition frameworks, we demonstrate in Chapter 4 that recognizing just two object states can effectively reveal the underlying *object-transforming actions*. However, it's important to note that not all actions result in changes to object states; some actions primarily

alter the scene rather than the objects themselves, such as moving an object from one location to another or adding an ingredient to a dish.

Our method [146] employs object and attribute manipulation modules that adeptly capture the subtle nuances between different object states and the contextual changes within the scene. This approach allows for a comprehensive analysis that encompasses both object and *context-transforming actions*. The results are demonstrated through a compositional action classification task, where a limited number of objects are observed during training, yet the system is capable of recognizing actions involving unseen objects. Furthermore, we extend our findings to out-of-domain datasets, illustrating robust generalization capabilities. Additionally, we introduce a newly proposed benchmark split Chop & Learn [148], for granular actions, to showcase promising advancements towards achieving generalizable abilities and enhancing long-term video understanding through compositional action recognition. With this, Part I concludes, primarily exploring the recognition and generalization of unseen compositions of attributes and objects in both images and videos.

Part II: Attribute-Object Composition for Generation

In the second part of this thesis, we delve into the utilization of compositionality in image and video generative applications. Over the past few years, Generative AI has experienced a remarkable surge, highlighting its vast potential and the growing emphasis on creating new, unseen scenarios. This section examines the influence of fine-grained attributes on the generation of novel objects.

In Chapter 5, we initially explore the concept of *compositional generalization*, which is

defined as the ability to synthesize and recognize new combinations from a finite set of observed object states. The current suite of datasets either lacks detailed annotations for object states or sufficient data to examine how these states evolve under various conditions. Consequently, assessing the compositional generalizability of models across different tasks continues to be a challenging issue. To address this, we introduce a new dataset, **Chop & Learn (ChopNLearn)** [148], specifically designed to facilitate the study of compositional generalization—the capability to recognize and generate unseen compositions of objects in varied states. Our investigation focuses on a ubiquitous task: the cutting of fruits and vegetables. By employing different cutting styles, these objects undergo distinctive transformations, resulting in states that are readily identifiable by humans. Our aim is to explore how these varied styles can be adapted across a diverse range of objects to recognize unseen object states. More precisely, we have selected *twenty* objects and *seven* commonly used cutting styles (including the whole object), resulting in pairs of object states with varying granularity and dimensions. We have compiled videos of these objects from *four* different viewpoints, with each video labeled to identify various object states. Each cutting style uniquely alters the visual appearance of the objects.

To gain a deeper insight into object transformations and to assess our understanding of appearance changes, we introduce an innovative benchmark task in Chapter 6: Compositional Image Generation [148] is designed specifically for generating unseen compositions. The goal of this task is to create an image from a combination of object and state that the model has not previously encountered during training. For instance, if the training phase involved a generative model learning from images of an (apple, whole) and an (orange, round slices), during testing, the model is tasked with synthesizing a novel composition such as an (apple, round slices). To achieve this, we plan to leverage large-scale text-to-image generative models, utilizing text

prompts to denote the object-state combinations. We evaluate several existing methodologies including Stable Diffusion [142], Textual Inversion [46], and DreamBooth [144]. Furthermore, we introduce a novel approach that involves creating new tokens for objects and states and concurrently fine-tuning both language and diffusion models. Finally, we critically discuss the challenges and limitations encountered in previous studies as well as those faced by our proposed generative model, supported by a thorough evaluation.

Thus far, we have explored the generation of new compositions of attributes and objects within images. Yet, when dealing with videos, a different challenge arises: the need to consistently edit and modify the attributes of objects across multiple frames. In Chapter 7, we investigate the insertion and editing of object attributes in videos, employing training-free pipelines from text-to-image models. We introduce InVi, a novel approach that facilitates the insertion or replacement of objects within videos (commonly known as inpainting) utilizing readily available, text-to-image latent diffusion models. InVi is designed for precise manipulation of objects and their attributes, ensuring they are seamlessly integrated into the existing video background. This method diverges from traditional video editing techniques, which typically focus on extensive re-styling or complete scene transformations.

To accomplish our objectives, we address two primary challenges. First, to achieve high-quality control and blending, we implement a two-step process that involves inpainting and then matching. The procedure starts by introducing the object with new attributes into a single frame via a ControlNet-based [202] inpainting [142] diffusion model. Subsequent frames are then generated, conditioned on features from the initially inpainted frame, which serves as an anchor to bridge the domain gap between the background and the object. Second, to maintain temporal coherence, we adapt the diffusion model’s self-attention layers into extended-attention layers.

The features of the anchor frame act as keys and values in these layers, fostering consistency throughout the video. This approach negates the necessity for video-specific fine-tuning, offering an efficient and flexible solution. Experimental results validate that InVi [149] excels in achieving realistic object insertion, with effective blending and temporal coherence that surpasses existing methods.

In conclusion (chapter 8), we summarize our contributions, discuss potential future research directions it facilitates, and situate our work within the rapidly evolving landscape of Visual Understanding domains of Recognition and Generation.

Part I: Object-Attribute Composition for Recognition

Chapter 2: Disentangling Visual Embeddings for Attributes and Objects

2.1 Introduction

Objects in the real world can appear with different properties, *i.e.*, different color, shape, material, etc. For instance, an apple can be red or green, cut or peeled, raw or ripe, and even dirty or clean. Understanding object properties can greatly benefit various applications, *e.g.*, robust object detection [10, 37, 38, 89], human object interaction [82, 155, 183], and activity recognition [5, 8, 9, 39, 43, 108]. Since the total number of possible attribute-object pairs in the real world is prohibitively large, it is impractical to collect image examples and train multiple classifiers. Prior works proposed compositional learning, *i.e.*, learning to compose knowledge of known attributes and object concepts to recognize a new attribute-object composition. Datasets such as MIT-States [76] and UT-Zappos [195] are commonly used to study this task, with joint attribute-object recognition for a diverse, yet limited set of objects and attributes. Compositional learning refers to combining simple primitive concepts to understand a complex concept. This idea dates back to Recognition and Composition theory by Biederman [12], and early work in the visual domain by Hoffman [72], which proposed recognition by parts for pose estimation. Prior works explore compositionality to a certain degree, *e.g.*, via feature sharing and shared embeddings space. Among them, most works use linguistically inspired losses to separate attributes and objects in the shared embedding space, then use that primitive knowledge to compose new com-

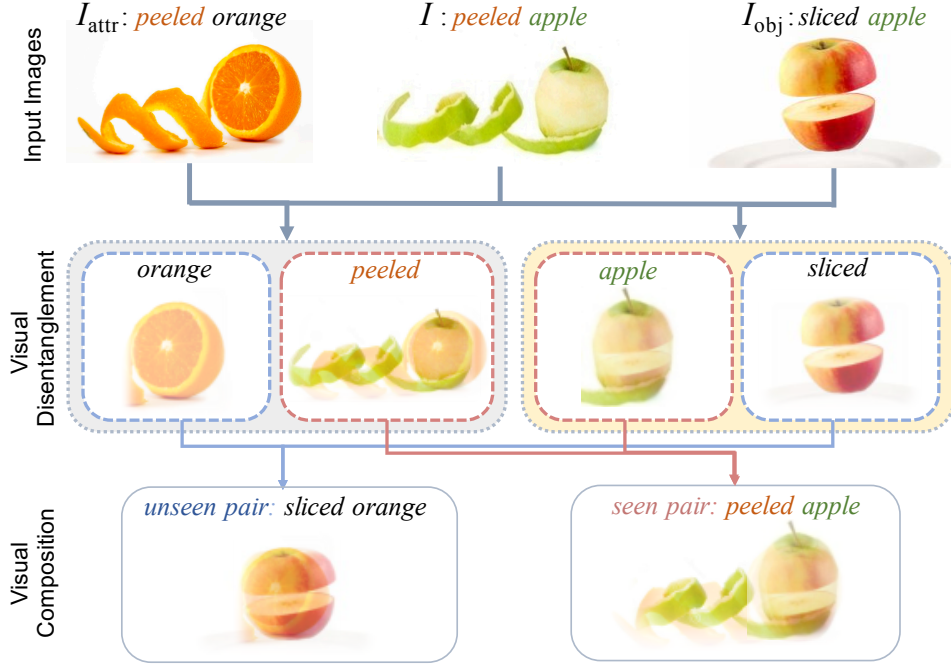


Figure 2.1: **Method illustration:** Given an input image I of peeled apple, we use two other images: (1) one with same object, different attribute I_{obj} - sliced apple, (2) one with same attribute, different object I_{attr} - peeled orange. We propose a novel architecture that takes I and I_{attr} , and extracts their visual similarity features for peeled and visual dissimilarity features for orange. Similarly, using I and I_{obj} , the visual similarity features for apple, and the dissimilarity features for sliced can be extracted. We compose these primitive visual features to hallucinate a seen pair peeled apple, and a novel unseen pair sliced orange to be used for regularizing our embedding space. Note that this is a visualization of embedding space composition, we do not generate images.

plex pairs. Using linguistic embeddings is helpful since: (1) there is a clear distinction between attribute and object in the embedding space, and (2) these embeddings already contain semantic knowledge of similar objects and attributes, which is helpful for composition. However, unlike word embedding, it is difficult to discriminate the object and attribute in the visual embedding space.

This is due to the fact that image feature extractor is usually pre-trained for object classification, often along with image augmentation (*e.g.*, color jitter) that tends to produce attribute-invariant image representation, thus does not learn objects and attributes separately. In this chapter [147], we propose a new direction that focuses on *visual cues*, instead of using linguistic cues

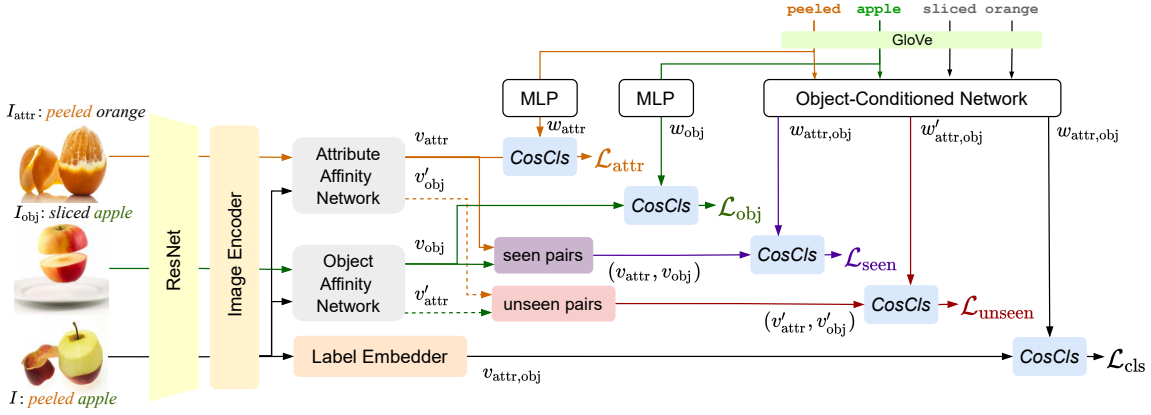


Figure 2.2: **System Overview:** Given an image I , for peeled apple, we consider two images:, one with same object: I_{obj} , sliced apple, and one with same attribute, I_{attr} peeled orange. (1) The Object-Conditioned Network composes pair word embedding, using GloVe word embeddings for labels. (2) Label Embedder uses the image I and embeds visual feature $v_{attr,obj}$ along with word embedding $w_{attr,obj}$, using loss $\mathcal{L}_{cls}$. (3) Attribute Affinity Network and Object Affinity Network, disentangles the same attribute and object from the pair of images I, I_{attr} and I, I_{obj} respectively. Disentangled visual features for peeled (v_{attr}) and apple (v_{obj}) are used along with word embeddings of attribute (w_{attr}) and objects (w_{obj}), to compute $\mathcal{L}_{attr}$ and $\mathcal{L}_{obj}$. (4) Using disentangled features, we compose seen pair peeled apple (v_{attr}, v_{obj}) and unseen pair sliced orange (v'_{attr}, v'_{obj}), for composition losses $\mathcal{L}_{seen}$ and $\mathcal{L}_{unseen}$.

explicitly for novel compositions.

Analogous to linguistic embedding, our work focuses on disentangling attribute and object in the visual space. Our method, Object Attribute Disentanglement (OADis), learns distinct and independent visual embeddings for peeled and apple from the visual feature of peeled apple. As shown in Figure 2.1, for image I of peeled apple, we use two other images: one with same object and different attribute I_{obj} (e.g., sliced apple), and one with same attribute and different object I_{attr} (e.g., peeled orange). OADis takes I and I_{obj} and learns the similarity (apple) and dissimilarity (sliced) of the second image with respect to the first one. Similarly, using I and I_{attr} , the commonality between them (peeled) and the left out dissimilarity (orange) can also be extracted. Further, composition of these extracted visual primitives are used to hallucinate seen and unseen pair, peeled apple and sliced orange respectively.

For compositional learning, it is necessary to decompose first before composing new un-

seen attribute-object pairs. As humans, we have the ability to imagine an unseen complex concept using previous knowledge of its primitive concepts. For example, if someone has seen a `clown` and a `unicycle`, they can imagine `clown` on a `unicycle` even if they have never seen this combination in real life [45, 128]. This quality of *imagination* is the basis of various works such as GANs [24], CLIP [138] and DALL-E [139]. However, these works rely on larger datasets and high computation power for training. We study this idea of *imagination* for a smaller setup by composing newer complex concepts using disentangled attributes and object visual features. Our work focuses on answering the question, *can there be visual embedding of peeled and apple, disentangled separately from visual feature of peeled apple?* Our contributions are as follows:

- We propose a novel approach, OADis, to disentangle attribute and object visual features, where visual embedding for `peeled` is distinct and independent of embedding for `apple`.
- We compose unseen pairs in the visual space using the disentangled features. Following Compositional Zero-shot Learning (CZSL) setup, we show competitive improvement over prior works on standard datasets [76, 195].
- We propose a new large-scale benchmark for CZSL using an existing attribute dataset VAW [131], and show that OADis outperforms existing baselines.

2.2 Related Work

Visual Attributes. Visual attributes have been studied widely to understand visual properties and low-level semantics of objects. These attributes help further improve on various downstream tasks such as object detection [10, 22, 37, 38, 89, 115], action recognition [5, 8, 9, 39, 43, 108],

image captioning [87, 124], and zero-shot and semi-supervised classification [7, 20, 22, 91, 121, 125, 157]. Similar to multi-class classification for objects, initial work for attribute understanding used discriminative models [89, 127], without understanding attributes. Other works [18, 37, 75, 105] explored the relation between the same attributes and different objects, to learn visual attributes. Particularly, disentangling object features from attribute features are explored in [42, 44]. Although, these works use clustering and probabilistic models to learn the attributes of objects.

Compositional Zero-shot Learning. Concept of compositional learning was first introduced in Recognition by Parts [72]. Initially, [114] employed this concept for objects and attributes. Unlike zero-shot learning (ZSL), CZSL requires the model to learn to compose unseen concepts from already learned primitive components. [18, 114] proposed separate classifiers for primitive components, and merged all into a final classifier. Most prior works use linguistically inspired auxiliary loss terms to regularize training for embedding space, such as: [119] models attributes as a linear transformation of objects, [95] uses rules of symmetry for understanding states, and [191] learns composition and decomposition of attributes hierarchically. Another set of studies uses language priors to learn unseen attribute-object pairs, either in feature space or with multiple networks [102, 135, 181]. Other recent works use graph structure to leverage information transfer between seen to unseen pairs using Graph Convolutional Networks [106, 118], and [187] uses key-query based attention, along with modular network with message passing for learning relation between primitive concepts.

2.3 Object Attribute Disentanglement (OADis)

Contrary to prior works [95, 118, 119, 191], we explicitly focus on separating attributes and object features in the visual space. More precisely, TMN [135] uses word embeddings to generate attention layers to probe image features corresponding to a given pair, GraphEmbedding [118] exploits the dependency between word embeddings of the labels, and HiDC [191] mainly uses word embeddings to compose novel pairs and generate more examples for their triplet loss. To the best of our knowledge, none of the existing works have explored visual feature disentanglement of attributes and objects. We hypothesize that attribute and object visual features can be separated when considering visual feature similarities and differences between image pairs. Composing these disentangled elements help regularize the common embedding space to improve recognition performance. More concretely, we take cues from [42] and [114, 191], to learn to compose unseen attribute-object pairs leveraging visual attributes based on auxiliary losses.

2.3.1 Task Formulation

We follow the conventional Compositional Zero-shot Learning (CZSL) setup, where distinct attribute-object compositions are used at training and testing. Each image I is labeled with $y = y_{\text{attr,obj}} \in Y$, where y_{attr} and y_{obj} are respectively the attribute and object label. The dataset is divided into two parts, seen pairs $y^s \in Y^s$ and unseen pairs $y^u \in Y^u$, such that $Y = Y^s \cup Y^u$, $Y^s \cap Y^u = \emptyset$. Although $y^u = y_{\text{attr,obj}} \in Y^u$ consists of attribute y_{attr} and object y_{obj} that are never seen together in training, they are separately seen. We employ the Generalized CZSL setup defined in [135], which has seen Y^s and unseen pairs Y^u in the validation and test sets as detailed in Table 2.1. As shown in Figure 2.2, for image I , with label peeled

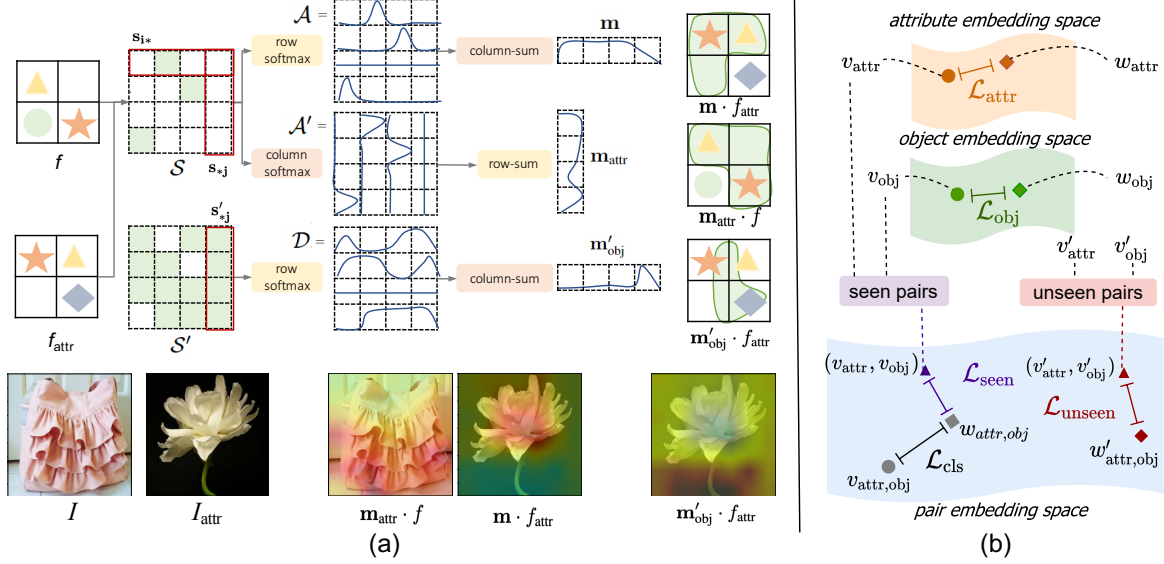


Figure 2.3: (a) Attribute Affinity Module: We compute the cosine similarity between blocks in f and f_{attr} (S in Eq. 2.3), then apply row-wise and column-wise softmax ($\mathcal{A}$ and $\mathcal{A}'$), followed by a respective column-sum and row-sum to obtain $\mathbf{m}$ and $\mathbf{m}_{\text{attr}}$. $\mathbf{m}$ represents regions where f_{attr} is highly similar to f (hence, we reshape and multiply $\mathbf{m}$ with f_{attr}) and $\mathbf{m}_{\text{attr}}$ represents regions where f is highly similar to f_{attr} (thus, $\mathbf{m}_{\text{attr}} \cdot f$). Similarly, S' represents the regions where feature f_{attr} is not similar to feature f (more details in Section 2.3.2). The last row shows real samples and generated attention maps overlayed on images. Give image ruffled bag and ruffled flower, we show that attribute ruffle is highlighted in the center $\mathbf{m}_{\text{attr}} \cdot f$ and $\mathbf{m} \cdot f_{\text{attr}}$. Whereas, $\mathbf{m}'_{\text{obj}} \cdot f_{\text{attr}}$ shows the dissimilar regions of I_{attr} w.r.t I . (b) Shows the three embedding spaces learnt with different losses. Same notation is used as Figure 2.2.

apple, we choose two additional images: one with same object and different attribute I_{obj} (e.g., sliced apple), and another image with same attribute and different object I_{attr} (e.g., peeled orange). Note that the subscript of image symbol, e.g., attr in I_{attr} , shows similarity with I , whereas superscript denotes seen and unseen sets.

2.3.2 Disentangling Visual Features

We extract image and label embedding features from pre-trained networks (ResNet [65] and GloVe [129]). As seen in Figure 2.2, we use *Image Encoder (IE)* and *Object Conditioned Network (OCN)*, for image and word embedding features respectively. Similar to [119], we use *Label Embedder (LE)* as an additional FC-Layer for the image feature. *LE* and *OCN* learn

image and word embeddings and embed those in a common pair embedding space. Next, visual similarity between I and I_{obj} is computed using *Object Affinity Network*, which extracts visual features for object, v_{obj} . Whatever is not similar is considered dissimilar. Hence, visual features of I_{obj} that are least similar to visual features of I are considered as the attribute feature v'_{attr} in I_{obj} , which is sliced in this example. Similarly, *Attribute Affinity Network* takes I and I_{attr} , and extracts visual similarity feature v_{attr} for peeled, and dissimilar visual features of I_{attr} , as object feature v'_{obj} for orange. The disentangled features are then used to compose seen and unseen pairs. We discuss the details in the following sections:

Image Encoder (IE). We use the second last layer before AveragePool of an ImageNet-pretrained ResNet-18 [30, 65] to extract features for all images. *IE* is a single convolutional layer that is shared across images I , I_{attr} and I_{obj} to generate their image features, represented as f , f_{attr} and f_{obj} respectively, where each $f \in \mathbb{R}^{n \times 49}$ and n is the output dimension of IE.

Label Embedder (LE). Inspired by [119], our *LE* inputs spatial feature from ResNet [65], AveragePools and passes through a linear layer to extract final feature $v_{\text{attr,obj}}$ for pair embedding, which has same dimension as the word embedding final feature $w_{\text{attr,obj}}$, extracted from *Object Conditioned Network (OCN)* (Figure 2.2). This is the main branch, and is used for input image I only.

Object Conditioned Network (OCN). This takes word embeddings of attribute emb_{attr} and object emb_{obj} , concatenates the features and passes through multiple layers. Object-conditioned is named because a residual connection for the object feature is concatenated with the final attribute feature, and the output feature is $w_{\text{attr,obj}} \in Y$. We discuss the motivation for this in Section 2.6.

Cosine Classifier (CosCls). Analogous to compatibility function used in [106, 118], we use

cross-entropy along with cosine similarity to get the final score for each pair. For visual features $v_{\text{attr,obj}}$ (from *LE*), and composed word embeddings $w_{\text{attr,obj}}$ (from *OCN*), *CosCls* provides logits for an image I . For instance, let us assume $v : X \rightarrow Z$ and $w : Y \rightarrow Z$. Z is the common embedding space for word embeddings w and visual embeddings v . Then classifier unit *CosCls* gives the score for label $y \in Y^s$ is C :

$$h(v, w) = \cos(v, w) = \delta \cdot \frac{v^T w}{\|v\| \|w\|} \quad (2.1)$$

$$C(v, w) = \frac{e^{h(v, w)}}{\sum_{y \in Y^s} e^{h(v, y)}} \quad (2.2)$$

where δ is the temperature variable. Each loss function uses same *CosCls* score evaluator, with different inputs.

Object and Attribute Similarity Modules. Our main contribution is the proposed affinity modules and compositional losses. Inspired by image captioning [19, 92, 94], OADis uses image similarities and differences to identify visual features corresponding to attributes and objects. *Object Affinity Network (OAN)* uses f and f_{obj} , whereas *Attribute Affinity Network (AAN)* uses f and f_{attr} . For brevity, we explain the AAN, while the OAN follows the same architecture. Reminded that both f and $f_{\text{attr}} \in \mathbb{R}^{n \times 49}$.

Similar to [186], which computes attention between word concepts with corresponding visual blocks, we compute attention between two images I and I_{attr} . Since both images have the same attribute, *i.e.*, *peeled*, our affinity network learns visual similarity between the images, which represents the attribute. Similarity matrix $\mathcal{S}$ is the cosine similarity between f and f_{attr} , such that $\mathcal{S} \in \mathbb{R}^{49 \times 49}$ as:

$$\mathcal{S} = \frac{f^T f_{\text{attr}}}{\|f\|_2 \|f_{\text{attr}}\|_2} \quad (2.3)$$

where element s_{ij} represents the similarity between i^{th} element of f with j^{th} element of f_{attr} . Moreover, let $\mathbf{s}_{i*}$ and $\mathbf{s}_{*j}$ represent the i^{th} row and j^{th} column of $\mathcal{S}$ respectively. Then, $\mathbf{s}_{i*}$ captures the similarity of all the elements in f_{attr} with respect to i^{th} element of f . To know the most similar element among f_{attr} with respect to i^{th} element of f , we can take a row-wise `softmax` over $\mathcal{S}$. Similarly, for j^{th} element of f_{attr} , column $\mathbf{s}_{*j}$ represents the similarity with all the elements of f . Using a column-wise `softmax`, we can interpret the most similar and least similar element of f with respect to j^{th} element of f_{attr} , as shown in Figure 2.3. Therefore, by applying column-wise and row-wise `softmax`, we get two matrices, $\mathcal{A}$ and $\mathcal{A}'$ ($\mathcal{A}, \mathcal{A}' \in \mathbb{R}^{d \times d}$, $d = 49$),

$$\mathcal{A}_i = \frac{e^{\lambda \mathbf{s}_{i*}}}{\sum_{j=1}^d e^{\lambda s_{ij}}} \quad \text{and} \quad \mathcal{A}'_j = \frac{e^{\lambda \mathbf{s}_{*j}}}{\sum_{i=1}^d e^{\lambda s_{ij}}}, \quad (2.4)$$

where λ is the inverse temperature parameter. We compute row and column sum for $\mathcal{A}$ and $\mathcal{A}'$ respectively, to get final similarity maps, $\mathbf{m}$ and $\mathbf{m}_{\text{attr}}$,

$$m_j = \sum_{i=1}^d \mathcal{A}_{ij} \quad \text{and} \quad m_{\text{attr}i} = \sum_{j=1}^d \mathcal{A}'_{ij}. \quad (2.5)$$

Similarly, the difference between these two images f and f_{attr} is the object label, y_{obj} . Hence, we use the negative of $\mathcal{S}$ as the image difference, denoted as $\mathcal{S}'$. Then, difference of f_{attr} with respect to f would be row-wise `softmax` of difference matrix, denoted by $\mathcal{D}$. Hence, by performing column-sum over $\mathcal{D}$, we get difference map, $\mathbf{m}'_{\text{obj}}$,

$$\mathcal{D}_j = \frac{e^{\gamma \mathbf{s}'_{*j}}}{\sum_{i=1}^d e^{\gamma s'_{ij}}} \quad \text{and} \quad m'_{\text{obj}i} = \sum_{j=1}^d \mathcal{D}_{ij}. \quad (2.6)$$

The final disentangled features for attribute v_{attr} and object v'_{obj} , for both AAN and OAN, can be

computed as:

$$\begin{aligned} v_{\text{attr}} &= \mathbf{m} \cdot f_{\text{attr}} + \mathbf{m}_{\text{attr}} \cdot f & \text{and} & \quad v'_{\text{obj}} = \mathbf{m}'_{\text{obj}} \cdot f_{\text{attr}} \\ v_{\text{obj}} &= \mathbf{m} \cdot f_{\text{obj}} + \mathbf{m}_{\text{obj}} \cdot f & \text{and} & \quad v'_{\text{attr}} = \mathbf{m}'_{\text{attr}} \cdot f_{\text{obj}}. \end{aligned} \tag{2.7}$$

More details using a toy example can be seen in Figure 2.3. Using concatenation of v_{attr} and v_{obj} along with a single Linear layer, composes the pair `peeled apple`, represented by $(v_{\text{attr}}, v_{\text{obj}})$. Similarly, the disentangled visual features v'_{attr} and v'_{obj} , are used to compose unseen pair `sliced orange`, and is represented as $(v'_{\text{attr}}, v'_{\text{obj}})$.

2.3.3 Embedding Space Learning objectives

As shown in Figure 2.3b, we learn three embedding spaces: (1) attributes space, (2) object space, and (3) attribute-object pair space. The attribute and object spaces are used for disentangling the two, whereas pair embedding is used for final pair composition and inference. OADis has separate loss functions for disentangling and composing. All loss functions are expressed in terms of *CosCls* defined previously.

The loss function for main branch, $\mathcal{L}_{\text{cls}}$ uses combined visual feature $v_{\text{attr,obj}}$ from *LE* and word embedding feature $w_{\text{attr,obj}}$ from *OCN*. $\mathcal{L}_{\text{cls}}$ is used for the pair embedding space. Similarly, $\mathcal{L}_{\text{attr}}$ and $\mathcal{L}_{\text{obj}}$ are used to learn the visual attribute and object feature, in their respective embedding spaces. $\mathcal{L}_{\text{attr}}$ pushes the visual feature of attribute, closer to the word embedding. $\mathcal{L}_{\text{obj}}$ does the same for objects in object embedding space Figure 2.3b. These losses cover the concept of disentanglement, and can be represented as:

$$\mathcal{L}_{\text{cls}} = C(v_{\text{attr,obj}}, w_{\text{attr,obj}}) \quad (2.8)$$

$$\mathcal{L}_{\text{attr}} = C(v_{\text{attr}}, w_{\text{attr}}); \quad \mathcal{L}_{\text{obj}} = C(v_{\text{obj}}, w_{\text{obj}})$$

For composition, we use $\mathcal{L}_{\text{seen}}$ and $\mathcal{L}_{\text{unseen}}$. Among the images seen (I , I_{attr} , and I_{obj}), disentangled features v_{obj} and v_{attr} , composes the same pair as $(v_{\text{attr}}, v_{\text{obj}})$, which we refer to as the seen composition. Note that $(v_{\text{attr}}, v_{\text{obj}})$ is different from $v_{\text{attr,obj}}$, as the former is hallucinated feature with combination of disentangled attribute and object visual features, and latter is the combined visual feature extracted with *LE*. Here, we use $\mathcal{L}_{\text{seen}}$ loss which takes the composition of disentangled features and learns to put the composition closer to $w_{\text{attr,obj}}$. Moreover, the dissimilarity aspect from *OAN* and *AAN* extracts v'_{attr} and v'_{obj} , which composes an unseen pair $(v'_{\text{attr}}, v'_{\text{obj}})$. We use $\mathcal{L}_{\text{unseen}}$ as unseen loss since the hallucinated composition is never seen among I , I_{attr} , and I_{obj} .

$$\mathcal{L}_{\text{seen}} = C((v_{\text{attr}}, v_{\text{obj}}), w_{\text{attr,obj}}) \quad (2.9)$$

$$\mathcal{L}_{\text{unseen}} = C((v'_{\text{attr}}, v'_{\text{obj}}), w'_{\text{attr,obj}})$$

The combined loss function $\mathcal{L}$ is minimized over all the training images, to train OADis end-to-end. The weights for each loss (α) are empirically computed:

$$\mathcal{L} = \mathcal{L}_{\text{cls}} + \alpha_1 \mathcal{L}_{\text{attr}} + \alpha_2 \mathcal{L}_{\text{obj}} + \alpha_3 \mathcal{L}_{\text{seen}} + \alpha_4 \mathcal{L}_{\text{unseen}}.$$

Table 2.1: Dataset Details: This table shows the statistics for different datasets and their splits. The proposed VAW-CZSL benchmark significantly increases the number of attributes and objects.

Datasets:	Train set				Val set			Test set		
	Attr.	Obj.	Seen Pairs.	# Images	Seen Pairs	Unseen Pairs	# Images	Seen Pairs	Unseen Pairs	# Images
MIT-States [76]	115	245	1262	30338	300	300	10420	400	400	12995
UT-Zappos [195]	16	12	83	22998	15	15	3214	18	18	2914
VAW-CZSL [131]	440	541	11175	72203	2121	2322	9524	2449	2470	10856

2.4 Experimental Setup

2.4.1 Dataset

We show results on three datasets: MIT-states [76], UT-Zappos [195], and a new benchmark for evaluating CZSL on images of objects in-the-wild, referred as VAW-CZSL. VAW-CZSL is created based on images with object and attribute labels from the VAW dataset [131]. Both MIT-states [76] and UT-Zappos [195] are common datasets used for this task in previous studies. MIT-states covers wide range of objects (*i.e.*, laptop, fruits, fish, room, *etc.*) and attributes (*i.e.*, mossy, dirty, raw, *etc.*), whereas UT-zappos has fewer objects (*i.e.*, shoes type: boots, slippers, sandals) and fine-grained attributes (*i.e.*, leather, fur, *etc.*).

2.4.2 Evaluation

We use Generalized CZSL setup, defined in [135], with dataset statistics presented in Table 2.1. As observed in prior works [118, 135], a model trained on a set of labels Y^s , does not generalize well on unseen pairs Y^u . Therefore, [118, 135] use a scalar term for overcoming the negative bias for unseen pairs. We use the same evaluation protocol, which computes Area Under the Curve (AUC) (in %) between the accuracy on seen and unseen compositions with different

bias terms [135]. Larger bias term leads to better results for unseen pairs whereas smaller bias leads to better results for seen pairs. Harmonic mean is reported, to balance the bias. We also report the attribute and object accuracy for unseen pairs, to show improvement due to visual disentanglement of features. Our new benchmark subset for VAW [131], follows the similar split as other datasets. In addition, we conduct all experiments with image augmentation for all methods (discussed in Section 2.6).

2.4.3 Implementation Details

Following baselines, we use ResNet18 [65] pre-trained on Imagenet [30] as backbone feature extractor. Since, proposed auxiliary losses leverage image features, we use a single convolutional layer with Batch Normalization, ReLU and dropout for Image embedder with output dimension 1024 and dropout as 0.3. Note that we extract ResNet features before average pool. For word embeddings, we initialize with GLoVe [129]. Object Conditioned network, uses multiple linear layers, first for objects and attributes separately, then for concatenated features. Label embedder takes 1024- d feature, performs AveragePool and finally embeds in a 300- d space. Each loss uses compatibility function, i.e. cosine similarity, followed by cross-entropy loss over the compatibility function. Object similarity and attribute similarity modules also use two linear layers with dropout 0.05. On UT-Zappos, because the dataset is very small, we find using a linear layer (a smaller and simpler module than OCN) with dropout 0.1 results in better performance. We use Adam optimizer with weight decay $5e^{-5}$, and learning rate $2.5e^{-6}$ for the GLoVe embedding. The learning rate for the rest of the model is $3e^{-4}$ on MIT-States, and $1e^{-4}$ on UT-Zappos and VAW-CZSL. We decay the learning rate by 10 at epoch 30 and 40 on MIT-States, at

epoch 50 on UT-Zappos, and at epoch 70 on VAW-CZSL. OADis needs to be trained for 70-150 epochs depending on the dataset, and training time is comparable with other methods (5-7 hours).

2.5 Results

2.5.1 Quantitative Results

2.5.1.1 Baselines

We compare with related recent and prominent prior works: AttrOp [119], LabelEmbed+ [119], TMN [135], Symnet [95], CompCos [106] and GraphEmb [118]. We do not compare with BMP [187], since it uses the concatenation of features from all four ResNet blocks (960-d features), resulting in higher input features and the number of network parameters than all other setups. Moreover, GraphEmb [118] is state-of-the-art; hence, comparing with that makes our work comparable to other baselines that [118] already outperforms. To be consistent, we state the performance of all models (including GraphEmb [118]) using frozen backbone ResNet without fine-tuning the image features, and using GloVe [129] for the object and attribute word embeddings. Before passing through backbone, training images are augmented with horizontal flip and random crop. Compared to other baselines, OADis uses convolutional features rather than AvgPooled, since it is easier to segregate visual features in the spatial domain for attributes and objects. Moreover, other studies [106, 118] have also used additional FC layers on top of *IE*, which we argue makes it fair for us to use pre-pooled features for OADis.

2.5.1.2 Results on MIT-States

MIT-states has considerable label noise [7], but still is a standard dataset for this task. We show significant improvement on this dataset (reported in Table 2.2), from previous state-of-the-art GraphEmb, which has 7.2 Val AUC and 5.3 Test AUC. Note that we do not report GraphEmb results with fine-tuning backbone, as we find it incomparable with other baselines that did not incorporate fine-tuning as part of their proposed methods. Overall, our model performs significantly better than GraphEmb on all metrics.

2.5.1.3 Results on UT-Zappos

Similar improvement trends hold for UT-Zapopos as well (see Table 2.2). Although, as explained for GraphEmb, it is difficult to balance the best performance for Val and Test set in this dataset. The problem is that 7/36 ($\sim 20\%$) attributes in Test set do not appear in Val set. Hence, improving Val set AUC, does not necessarily improve Test AUC for UT-Zappos. Similar trend can be seen for other baselines: CompCos has best Val AUC, but does not perform well on Test set, compared to TMN and Symnet. Even GraphEmb in their final table show the frozen backbone network has much lower performance than TMN. However, OADis performs well on UT-Zappos overall, with ~ 4.0 improvement for Val and Test AUC, HM, unseen and object accuracy.

2.5.1.4 Results on VAW-CZSL

Our model performs well on VAW-CZSL, and is consistently better than other methods across almost all metrics. As shown in Table 2.1, VAW-CZSL has $\sim 6-8$ times more pairs in each split than MIT-States, which shows how challenging the benchmark is. Due to top-1 AUC being

Table 2.2: We show results on MIT-states [76] and UT-Zappos [195]. Following [118, 135], we use AUC in % between seen and unseen compositions with different bias terms, along with Val, Test, attribute and object accuracy. HM is Harmonic Mean. OADis consistently outperforms on most categories with significant increment.

Model	MIT-States							UT-Zappos						
	Val@1	Test@1	HM	Seen	Unseen	Attribute	Object	Val@1	Test@1	HM	Seen	Unseen	Attribute	Object
AttrOp	2.5	2.0	10.7	16.6	18.4	22.9	24.7	29.9	22.8	38.1	55.5	54.4	38.6	70.0
LabelEmbed+	3.5	2.3	11.5	16.2	21.2	25.6	27.5	35.5	22.6	37.7	53.3	58.6	40.9	69.1
TMN	3.3	2.6	11.8	22.7	17.1	21.3	24.2	35.9	28.4	44.0	58.2	58.0	40.8	68.4
Symnet	4.5	3.4	13.8	24.8	20.0	26.1	25.7	27.4	27.7	42.5	56.7	61.6	44.0	70.6
CompCos	6.9	4.8	16.9	26.9	24.5	28.3	31.9	40.8	26.9	41.1	57.7	62.8	43.3	73.0
GraphEmb	7.2	5.3	18.1	28.9	25.0	27.2	32.5	33.9	24.7	38.9	58.8	61.0	44.0	72.6
OADis	7.6	5.9	18.9	31.1	25.6	28.4	33.2	40.8	30.0	44.4	59.5	65.5	46.5	75.5

Table 2.3: We show results on VAW-CZSL. Since it is a much more challenging dataset, with significantly large number of compositions, to discriminate performance among different baseline, we show top-3 and top-5 AUC (in %) for Val and Test sets.

Model	Val. Set		Test Set		HM	Seen	Unseen	Attr.	Obj.
	V@3	V@5	V@3	V@5					
AttrOp	1.4	2.5	1.4	2.6	9.1	16.4	11.7	13.7	34.9
LabelEmbed+	1.5	2.8	1.6	2.8	9.8	16.2	13.2	13.4	35.1
Symnet	2.3	3.9	2.3	3.9	12.2	19.1	15.8	18.6	40.9
TMN	2.2	3.9	2.3	4.0	11.9	19.9	15.4	15.9	38.3
CompCos	3.1	5.6	3.2	5.6	14.2	23.9	18.0	16.9	41.9
GraphEmb	2.7	5.3	2.9	5.1	13.0	23.4	16.8	16.9	40.8
OADis	3.5	6.0	3.6	6.1	15.2	24.9	18.7	17.5	43.3

too small to quantify any learning and comparing between methods, we report top-3 and top-5 AUC instead. This is also because objects in-the-wild tend to depict multiple possible attributes; hence, evaluating only the top-1 prediction is insufficient. We provide qualitative results of how our model makes object-attribute composition prediction on VAW-CZSL in the Appendix 10.

2.5.2 Qualitative results

To qualitatively analyze our hallucinated compositions, we perform a nearest neighbor search on all three datasets. We pick the unseen compositions composed using the disentangled features, and find their top-5 nearest neighbors from the validation and test set. Figure 2.4(a) illustrates a few of our results. Note that these pairs are never seen in training. Based on the

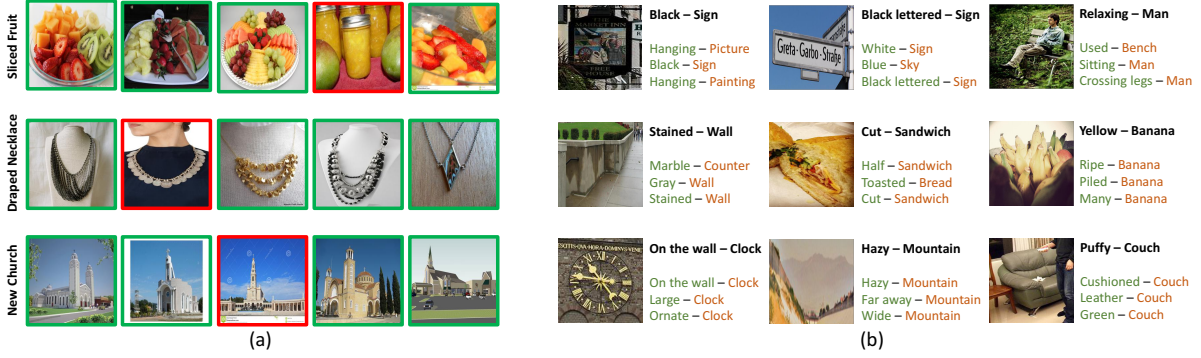


Figure 2.4: **Qualitative Results:** We show the nearest neighbors using the hallucinated unseen composition features for MIT-states and UT-Zappos. Although, all the neighbors are not correct (represented with red outline), they look very similar to true class labels: (a) First row: **pureed fruit**, Second row: **engraved coin**, Third row: **huge tower**. (b) We show top-3 predictions for images in VAW-CSZL.

hallucinated compositions of disentangled attributes and objects, we are able to retrieve samples from these unseen compositions. In Figure 2.4(b), we show the top-3 predictions of OADis on VAW-CSZL. Column 1 shows results for seen, and columns 2 and 3 show unseen compositions, with the ground-truth label on top (bold black). In all examples, our top-3 predictions describe the visual content of the images accurately, even though in many cases the ground-truth label is not predicted in top-1. For column 3, we purposely show examples where our model predictions totally differ from the ground-truth label, but still correctly describe the visual information in each image. Similar to [118], this explains the multi-label nature of object-attribute recognition, and why we report top-3 and top-5 metrics for the VAW-CSZL benchmark. We show nearest neighbor results for UT-Zappos [195] in Figure 2.5. Using the hallucinated composed features of unseen pairs, we find the top 5 nearest neighbors from test set. The red boxes show incorrect labels, where green show the correct labels.



Figure 2.5: We show the top 5 nearest neighbors using the hallucinated unseen composition features for UT-Zappos. All the neighbors with correct labels are represented by **green**, whereas incorrect ones are represented with **red** outline.

2.5.2.1 Attention Maps

In Figure 2.6 and 2.7, we show the qualitative results on MIT-States [76] and VAW-CZSL, with examples f and f_{attr} and overlayed feature maps. To re-iterate, for images with features f and f_{attr} , $\mathbf{m}_{\text{attr}} \cdot f$ shows how the regions in f which are most similar to f_{attr} , and $\mathbf{m} \cdot f_{\text{attr}}$ shows the regions in f_{attr} which are most similar to regions in f . Lastly, $\mathbf{m}'_{\text{obj}} \cdot f_{\text{attr}}$ shows the regions of f_{attr} which are most dissimilar to f . Although, the overlayed attention maps for similarity and dissimilarity make sense most of the times (Figure 2.6(b)), due to some inconsistencies in dataset, we still find some samples where is it difficult to disentangle the attribute and object features. The main reasons why this happens is:

- Some concepts are abstract, such as `clear sky`, `pressed metal`, `dirty floor` (fig. 2.6(a)), since it is very difficult to separate `dirty` from `floor`. Hence, the attention

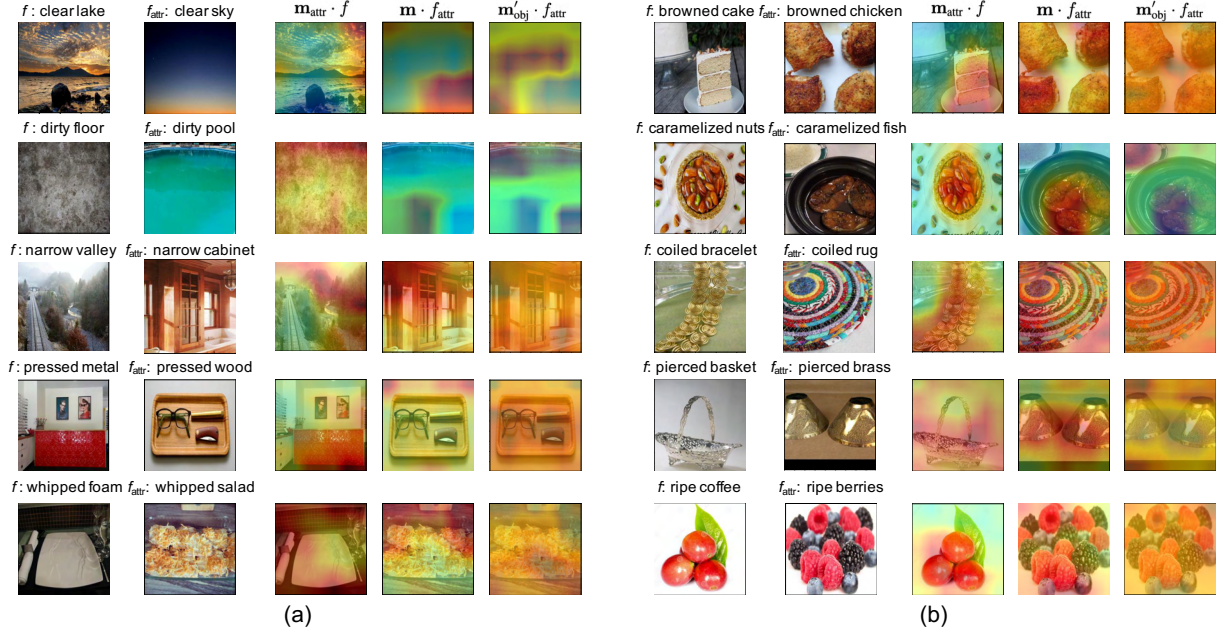


Figure 2.6: **(a) Failure Cases:** Shows the image pairs, f and f_{attr} , and the similarity and dissimilarity map overlaid (details in Sec 2.6). Moreover, we show for some cases for MIT-States, the examples are very vague or incorrect to actually capture attribute and object concepts separately. For instance, in clear lake and clear sky, it is very difficult to distinguish lake and sky. Hence the similarity and dissimilarity maps do not perform very well. Other examples are also of failure cases where the overlaid similarity and dissimilarity maps do not make sense. **(b) Correct Examples:** This shows some good examples, where the similarity and dissimilarity maps capture the attributes and objects correctly for MIT-States.

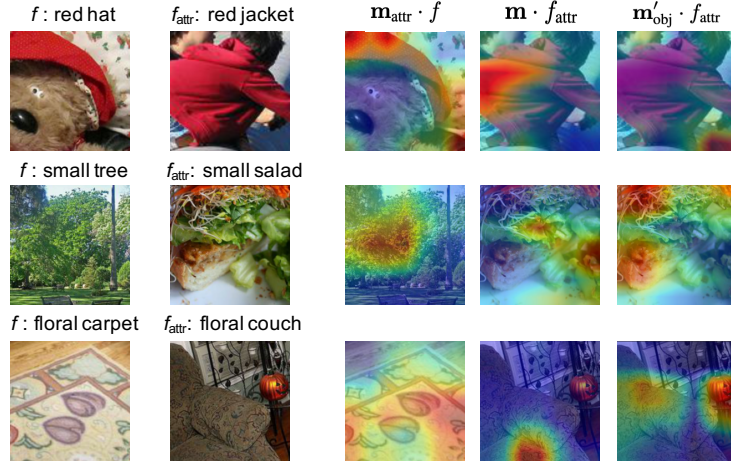


Figure 2.7: **Correct Examples:** We show the similarity and dissimilarity attention maps overlaid on images for VAW-CZSL as well. To re-iterate, for images with features f and f_{attr} , $m_{attr} \cdot f$ shows how the regions in f which are most similar to f_{attr} , and $m \cdot f_{attr}$ shows the regions in f_{attr} which are most similar to regions in f . Lastly, $m'_{obj} \cdot f_{attr}$ shows the regions of f_{attr} which are most dissimilar to f .

maps for similarity and dissimilarity do not make much sense.

- Some images in MIT-States and even in other dataset are mislabelled (*e.g.* whipped foam in fig. 2.6(a)), which makes it difficult to learn attributes from those.
- Finally, for some cases, like narrow valley/cabinet, our method fails to disentangle attribute and object similarity, due to various objects in the scene. For future work, using a foreground and background separator before finding similarities and dissimilarities between features can be helpful.

2.6 Ablation Studies

In this section, we show experiments to support our design choices for OADis. All the ablations are done for MIT-states [76], for one random seed initialization, and are consistent for other datasets as well.

2.6.1 To augment or not to augment?

Augmentation is a common technique to reduce over-fitting and improve generalization. Surprisingly, prior works do not use any image augmentation. OADis without augmentation gives 6.7% AUC on Val and 5.1% AUC on Test set for MIT-states. Hence, we use augmentation for OADis and re-implemented rest of the baselines in Table 2.2, showing that augmentation helps improving all methods ~ 1.0 - 1.5% AUC. We use horizontal flip and random crop as augmentation.

Table 2.4: Results with different networks for word-embeddings. Object-conditioning with attribute performs the best, and is therefore used for OADis (Section 2.6).

	Linear	MLP	Obj-cond. Network
Val@1	6.6	7.0	7.6
Test@1	5.0	5.2	5.9

2.6.2 Object-conditioned network

We experiment with different networks on top of word embeddings, namely Linear, MLP and Object-Conditioned. Object conditioned network uses word embedding for object to concatenate with attribute-object composition embeddings. We show in Figure 2.8, the diagrammatic representation of different networks. Label Embedder [119] uses a linear layer and concatenates word embeddings for attributes and objects. We experiment with other networks: MLP with more parameters with two layers and ReLU and Object-conditioned network that uses a residual connection for object embedding. Our intuition is that same attribute contributes differently to each object, *i.e.*, `ruffled bag` is very different from `ruffled flower`. Hence, attributes are conditioned on object. Adding a residual connection for object embeddings to the final attribute embedding helps condition the attribute. We empirically demonstrate that object-conditioning helps in Table 2.4.

2.6.3 Choice of word embeddings

Prior works [95, 106, 118] experiment with various kinds of word embeddings. In fact, GraphEmb [118] has more advantages over all other baselines, since they use a combination of word embeddings word2vec [113] and fasttext [13], whereas rest of the works use GloVe [129] only. To keep the results fair between all methods, we run all the baselines, even GraphEmb [118] with only

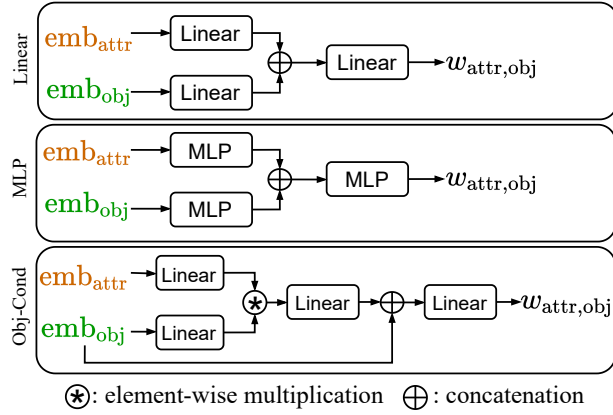


Figure 2.8: We show the different networks used on top of word embeddings. Empirically and following our intuition, Object-Conditioned network works best among the three (rest two are Linear and MLP). (Sec 2.6)

Table 2.5: Results with pre-trained word-embeddings. GloVe [129] performs the best, and is therefore used for OADis. (Sec 2.6.3)

Word Embs	Val AUC@1	Test AUC@1
Glove	7.6	5.9
Fasttext	7.4	5.3
Word2vec	7.5	5.4
Glove+fasttext	7.4	5.5
Glove+word2vec	7.5	5.6
Fasttext+word2vec	7.4	5.6

GloVe [129]. Results for using different embedding combinations is shown in Table 2.5. Overall, since our method uses word embeddings for visual disentanglement, the choice of word embeddings does not impact the performance much. Although, empirically, we found our model performs best when GloVe embeddings are used.

2.6.4 Values for λ and δ

We find the temperature variables λ and δ empirically. The values $\lambda = 10$ and $\delta = 0.05$ works best for OADis. Table 2.6 shows the results for all the different configurations. To understand the effect of each temperature variable, we keep all the rest of the parameters constant and only

Table 2.6: Results with pre-trained word-embeddings. GloVe [129] performs the best, and is therefore used for OADis. (Sec 2.6.4)

λ	Val AUC@1	Test AUC@1
0.01	7.5	5.6
0.1	7.5	5.7
1	7.4	5.7
10	7.6	5.9
100	7.4	5.7
δ	Val AUC@1	Test AUC@1
0.01	6.4	4.8
0.05	7.6	5.9
0.1	6.7	5.2

change the studied parameter.

2.6.5 Different weights for losses

We mention different weights for each loss function. Each α value is empirically found, and is used in the following equation for final loss function:

$$\mathcal{L} = \mathcal{L}_{\text{cls}} + \alpha_1 \mathcal{L}_{\text{attr}} + \alpha_2 \mathcal{L}_{\text{obj}} + \alpha_3 \mathcal{L}_{\text{seen}} + \alpha_4 \mathcal{L}_{\text{unseen}}$$

Note that $\mathcal{L}_{\text{cls}}$ is the main branch. The object and attribute losses are complementary. Hence, α_1 and α_2 , which are the weights for $\mathcal{L}_{\text{attr}}$ and $\mathcal{L}_{\text{obj}}$ share the same values, *i.e.* 0.5. Finally, α_4 and α_5 have the same value since both are composition losses for seen and unseen pairs, *i.e.* 0.05. The chosen weights for α values are in bold in Table 2.7.

Table 2.7: We show empirical weights of each loss function in this table. (Sec 2.6.5)

α_1 and α_2	α_3 and α_4	Val AUC@1	Test AUC@1
0.1	0.05	7.1	5.7
0.5	0.1	7.0	5.3
0.1	0.05	7.5	5.8
0.5	0.05	7.6	5.9
1.0	0.05	7.3	5.6

Table 2.8: We quantitatively show that the proposed architecture and different losses help in disentanglement and composition of unseen pairs. The experiments are conducted on MIT-States [76], where change in accuracy is shown with **green** and **red** based on increment or decrement respectively from the previous row. A dash (-) represents no change more than (± 0.1). Refer to Section 2.5.1 for details.

Losses	Val AUC@1	Test AUC@1	Seen	Unseen	Attribute	Object
$\mathcal{L}_{cls}$	7.24	5.43	29.92	25.33	28.03	33.10
$\mathcal{L}_{cls} + \mathcal{L}_{attr}$	-	-	31.09 (+2.0)	-	28.30 (+0.3)	-
$\mathcal{L}_{cls} + \mathcal{L}_{obj}$	-	-	-	25.50 (+0.2)	-	33.38 (+0.2)
$\mathcal{L}_{cls} + \mathcal{L}_{attr} + \mathcal{L}_{obj}$	7.49 (+0.2)	5.73 (+0.2)	-	-	28.50 (+0.2)	-
$\mathcal{L}_{cls} + \mathcal{L}_{attr} + \mathcal{L}_{obj} + \mathcal{L}_{seen}$	-	5.44 (-0.5)	31.21 (+0.2)	-	28.18 (-0.4)	-
$\mathcal{L}_{cls} + \mathcal{L}_{attr} + \mathcal{L}_{obj} + \mathcal{L}_{unseen}$	-	5.73 (+0.3)	-	25.80 (+0.4)	28.51 (+0.4)	-
$\mathcal{L}_{cls} + \mathcal{L}_{attr} + \mathcal{L}_{obj} + \mathcal{L}_{seen} + \mathcal{L}_{unseen}$	7.62 (+0.2)	5.94 (+0.2)	31.64 (+0.4)	25.60 (-0.2)	28.51	33.20

2.7 Is disentangling and hallucinating pairs helpful?

Prior works rely heavily on word embeddings for this task, but to improve the capabilities of visual systems, it is imperative to explore what is possible in the visual domain. We do an extensive study to understand if our intuition aligns with OADis (Table 2.8). Here are some takeaways:

- Using only $\mathcal{L}_{cls}$, we get a benchmark performance based on the architectural contributions, such as *LE* and *OCN*. When $\mathcal{L}_{attr}$ is added, significant performance boost for attribute accuracy can be seen in Table 2.8.
- Adding object loss $\mathcal{L}_{obj}$ with $\mathcal{L}_{cls}$, makes object accuracy better but no change in Val and Test AUC. This indicates the need of both losses to balance the effects. Using both $\mathcal{L}_{attr}$ and $\mathcal{L}_{obj}$ gives improvement in all measures.



Figure 2.9: Qualitative results showing top 3 attributes and objects from test images, using prototype disentangled features computed on training data.

- Adding $\mathcal{L}_{\text{seen}}$ results in boost for seen AUC, but drop in Test AUC, which has unseen pairs along with seen pairs. Using unseen loss $\mathcal{L}_{\text{unseen}}$ leads to increase in both Test and attribute accuracy.
- Finally adding unseen composition loss $\mathcal{L}_{\text{unseen}}$ along with seen loss $\mathcal{L}_{\text{seen}}$, the model improves on most metrics. Each loss plays a role and regularizes effects from other losses.

2.8 Is visual disentangling actually happening?

Visual disentanglement in feature space is challenging to visualize since: (a) parts of an image for attributes and objects are hard to distinguish, as *attributes are aspects of an object*; (b) OADis is end-to-end trained with losses to disentangle features for attribute and object embeddings, which is separate from pair embedding space. Inspired by [95, 119], we show a few qualitative results in Figure 2.9. Using all training images, prototype features $\mathcal{V}_{\text{attr}}$ for each attribute can be computed by averaging features for all images containing that attributes v_{attr} using *Attribute Affinity Network*. Similarly, with *OAN*, prototype object features are also computed. For each test image, we find top-3 nearest neighbors from these prototype features (Figure 2.9). Hence, the disentangled prototype features of attributes and objects are used for classifying unseen images. Note that results reported in Table 1 use pair embedding space for attribute and

object classification, whereas here we use auxiliary attribute and object embedding spaces (in Figure 2.3b) for the same task. If disentanglement features are not robust, then composition features will also not be efficient.

2.9 Limitations

Despite OADis outperforming prior works on all benchmarks, we still notice some outstanding deficiencies in this problem domain. First, similar to [118], OADis often struggles on images containing multiple objects, where it does not know which object to make prediction on. One possible solution is to utilize an object-conditioned attention that allows the model to focus and possibly output attribute for multiple objects. Second, from qualitative studies on VAW-CZSL, we notice there are multiple cases where OADis makes the correct prediction but is considered incorrect by the image label. This is due to the fact that objects in-the-wild are mostly multi-label (containing multiple attributes), which none of the current single-label benchmarks have attempted to address.

2.10 Conclusion

In this work, we demonstrated the ability to disentangle object and attribute in the visual feature space, that are used for hallucinating novel complex concepts, as well as regularizing and obtaining a better object-attribute recognition model. Through extensive experiments, we show the efficacy of our method, and surpass previous methods across three different benchmarks. In addition, we also propose a new benchmark for the compositional zero-shot learning task with images of objects in-the-wild, which we believe can help shift the focus of the community towards

images in more complex scenes. Finally, we also highlight limitations of our work, including the notable problem of multi-label in object attributes, which we hope would encourage future works to start tackling CSZL for more realistic scenarios.

Chapter 3: Composition of attributes and objects beyond seen primitives

3.1 Introduction

AttributZapposes explain the semantic properties of objects and are essential for efficient in-the-wild object recognition [10, 37, 89]. For instance, an unseen image of `dog` can be identified for its novel attributes, “a fluffy brown dog” even if the breed is unknown. However, data annotation for object-attribute is prohibitively expensive. It is challenging to scale annotated data because: (1) labels for each attribute and object are required individually, and (2) annotated data is required for all possible object-attribute compositions. Prior works [76, 114, 195] circumvent the first challenge by assuming a limited vocabulary of attributes and objects (‘seen’ primitives) and focus primarily on the second challenge of unseen compositions, which is referred to as Compositional Zero-shot Learning (CZSL). However, dealing with unseen attributes, unseen objects, and unseen compositions together is still an open problem. In this chapter [150], we propose a new task, *Open Vocabulary-Compositional Zero-shot Learning (OV-CZSL)*, which attempts to address aforementioned challenges simultaneously, (1) dealing with unseen attributes and unseen objects, and (2) recognizing unseen attribute-object compositions.

Recognizing unseen classes (Zero-shot learning) and composing unseen relations between primitive seen classes (compositional learning) are both well known challenges in computer vision. Towards a more generic understanding of the unseen concepts, we introduce OV-CZSL that

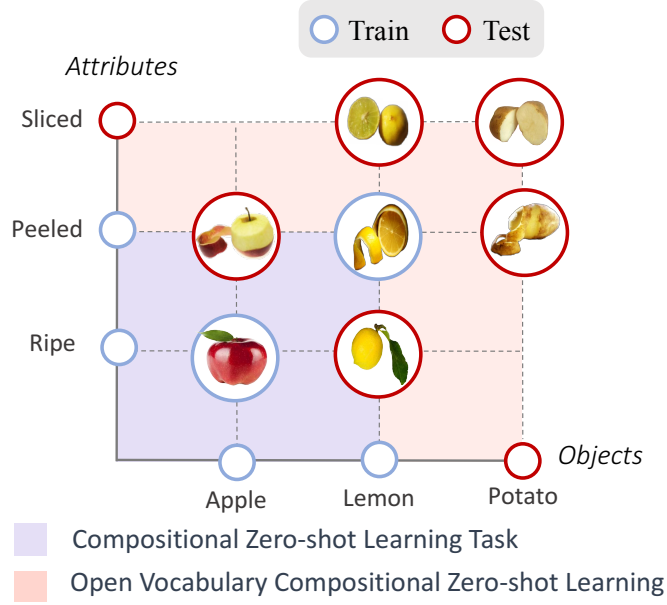


Figure 3.1: We propose a new task, Open Vocabulary-Compositional Zero-shot Learning (OV-CZSL), that expands upon Compositional Zero-shot Learning (CZSL). CZSL focuses on evaluating unseen compositions ripe lemon, peeled apple of seen attributes (ripe, peeled) and seen objects (apple, lemon), using training samples (ripe apple, peeled lemon). The novel OV-CZSL is trained with the same data as CZSL. However, it can be evaluated on unseen attributes (sliced), objects (potato) and their unseen compositions. These unseen compositions also include seen attribute-unseen object (peeled potato), unseen attribute-seen object (sliced lemon) and unseen attribute-unseen objects (sliced potato).

bridges the gap between ZSL and CZSL. Drawing inspiration from neuroscience, humans prototype and learn abstract concepts (fruit) while learning concrete concepts (apple, banana) [200] (also known as concept learning). For instance, we can associate visually similar concepts (apple, orange) as an abstract concept (fruit), by linking them via language semantically. By extending this idea, with a seen concept peeled lemon, we can relate lemon with another vegetable, e.g. potato. Even though an image of peeled potato is never seen; by understanding the concept of peeled, we can extrapolate what peeled potato would look like (Figure 6.1). Accordingly, we leverage seen attributes and objects, along with external knowledge, to semantically associate and learn new concepts and their compositions.

Zero-shot learning studies [47, 51, 59, 176], mostly uses external language and map unseen concepts close to semantically similar seen concepts. For OV-CZSL, we utilize pre-trained language embeddings (BERT [31]), to acquire semantically organized knowledge. Similar to CZSL, we learn a joint image and language embedding space to deal with unseen compositions. Within this embedding space, we expand the neighborhood around seen concepts to correlate different concepts and expand our vocabulary. Similar to query expansion in image retrieval [32, 66, 185], we enforce the visual feature of semantically similar linguistic concepts to be close in the embedding space. This helps in training the image encoder to recognize visually and semantically similar concepts that it has never seen before [176]. We propose a new regularizing loss function, *Neighborhood Expansion Loss*, to perform this vocabulary expansion efficiently.

Henceforth, we propose new benchmarks for the OV-CZSL task, building on three standard attribute-object datasets MIT-States [76], C-GQA [118] and VAW-CZSL [131, 147]. As shown in Figure 6.1(b), we expand on the CZSL task, to create more challenging sets for evaluating OV-CZSL, which includes a mixture of seen and unseen compositions of seen and unseen attributes and objects. We also propose a *Neighborhood Expansion Loss* that helps train our model to generalize efficiently on unseen pairs, while maintaining the seen pair accuracy of the existing models. To summarize, our contributions are as follows:

- We propose a challenging and practical extension on CZSL, Open Vocabulary-Compositional Zero-shot Learning (OV-CZSL), for learning compositions beyond the seen attributes and objects.
- We create three new benchmarks for MIT-states [76], C-GQA [118] and VAW [131, 147] for OV-CZSL, along with an efficient evaluation setup.

- We also propose an approach for OV-CZSL, which utilizes plug-and-play *Neighborhood Expansion Loss* to regularize training and generalize to unseen concepts and compositions.

Problem Setup. OV-CZSL is motivated by in-the-wild recognition, where it is infeasible to train on all possible objects and attributes for learning their compositions. To compare it with ZSL and CZSL setups, we discuss the train and test splits for these. Zero-shot Learning (ZSL) has distinct seen labels used for training and unseen labels used for testing. For attributes, we represent the seen and unseen sets as A and A^* respectively. Similarly, ZSL for objects will have O and O^* for seen and unseen sets, respectively. Compositional Zero-Shot Learning (CZSL) deals with unseen compositions of seen attribute and seen object pairs. Training set for CZSL is seen pairs (AO) and test set is unseen compositions of seen object and attribute classes, $(AO)^*$. By definition, there is no overlap between train and test sets in both ZSL and CZSL.

The goal for Open Vocabulary-Compositional Zero-shot Learning (OV-CZSL) is to recognize attributes and objects beyond its seen vocabulary (hence it's open vocabulary). During training, only a set of compositions of seen attributes and objects AO are used. For testing, we evaluate on compositions of seen attribute-unseen object AO^* , unseen attribute-seen object A^*O , unseen attribute-unseen object A^*O^* and a set of unseen compositions of seen attributes and objects $(AO)^*$. The unseen test set for OV-CZSL is $\{AO^*, A^*O, A^*O^*, (AO)^*\}$. Hence, the formulation is OV-CZSL can be considered as a generalized combination of ZSL and CZSL tasks.

3.2 Related Work

Zero-shot learning. Different from ZSL, Generalized Zero-shot Learning (GZSL) has distinct seen labels used for training and evaluated on both seen and unseen labels during testing. Most works use auxiliary attribute descriptions of the object classes, for ZSL of objects [51, 59, 122, 176]. Other works use word embeddings and pre-computing semantic features from Wikipedia et al., for identifying attributes or objects on commonly used datasets (AwA [90], CUB [170], SUN [126], ImageNet [30]). In summary, there is no datasets and works zero-shot attribute classification, along with zero-shot object classification and along with composition of attributes-objects.

Compositional Zero-shot learning. Most recent works rely on joint embedding space for images and labels, with linguistic losses [95, 102, 106, 118, 119, 135, 191]. Nagarajan *et al.* [119] proposed two scenarios for evaluation: (1) Closed world, where images are classified into only unseen pairs, and (2) Open world, where both seen and unseen pairs are used for evaluation. Moreover, [106] proposed another Open world setting, where for M attributes and N objects, they consider all possible combinations of $M \times N$ pairs, which might or might not be part of seen and unseen pairs for evaluation. Note that, in this work, ‘Open’ refers to extra vocabulary it can generalize to, but the evaluation is done in a closed world setup only. None of the existing works learn to compose novel unseen compositions.

Open Vocabulary. Sergio *et al.* [56] first proposed the task of open vocabulary object retrieval using descriptive natural language phrases by combining category and instance-level recognition. Further, Hang *et al.* [204] proposed an open-vocabulary framework for scene parsing, which built an image-text common embedding space using hierarchical semantic relations. Open vocabulary

setup has evolved over time, for scene parsing [56], object detection [34, 54, 156, 193, 198], and object segmentation [50]. Moreover, we want to emphasize that most Open-Vocabulary works use CLIP [138] in the pipeline, however using CLIP [138] makes the setup not-ZSL, since CLIP has already seen most attributes and objects. Hence, we avoid using CLIP for our setup. To the best of our knowledge, this is the first work exploring Open Vocabulary for Compositional Zero-shot Learning (OV-CZSL), and proposing a new challenging benchmark. We compare with baselines for both tasks ZSL and CZSL for OV-CZSL.

3.3 Open Vocabulary Compositional Zero-Shot Learning

In this section, we describe our method to tackle OV-CZSL. We emphasize that similar to ZSL, for solving OV-CZSL, the approach must balance the tradeoff between seen and unseen class accuracy. Existing works in CZSL consider only the composition of seen attributes and objects [106, 118, 119, 135, 147], and fail to generalize on compositions of unseen classes. We recommend new baselines for future work in OV-CZSL.

3.3.1 Task Formulation

OV-CZSL focuses on learning to compose seen and unseen attributes and objects. Each image I , has an attribute label A and an object label O . We use $*$ to represent the unseen concepts, *i.e.* unseen attributes are A^* and unseen objects are O^* . Training labels are set of seen attribute-object pairs, represented by Y^s , where $Y^s = AO$. Similar to CZSL, we evaluate on seen attribute-objects yet unseen compositions $(AO)^*$. Further, with unseen attributes and objects, there are three combinations of labels: 1) seen attribute and unseen object AO^* , 2) unseen attribute and

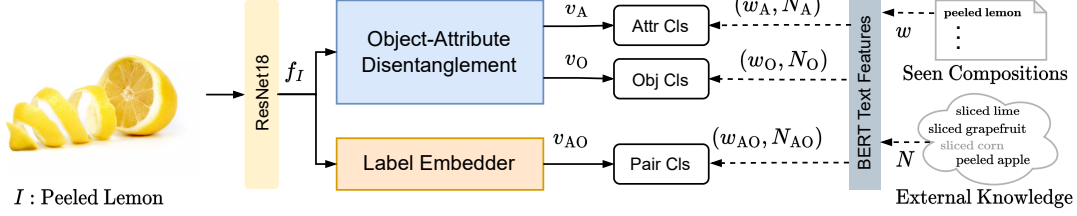


Figure 3.2: System Overview: ResNet18 [65] is used as visual feature extractor. Label Embedder embeds the image feature $v_{\text{attr,obj}}$ along with BERT [31] textual feature for the pair $w_{\text{attr,obj}}$. Object-Attribute Disentanglement module separates visual features into attributes and objects as v_{attr} and v_{obj} . Moreover, along with target labels for attribute w_{attr} , object w_{obj} , and pair $w_{\text{attr,obj}}$, we use neighbors for all of these from the semantic embedding space (N_A , N_O and N_{AO} respectively). We use cosine similarity loss, along with novel *Neighborhood expansion loss* to solve the task of OV-CZSL.

seen object A^*O , 3) unseen attribute and unseen object A^*O^* . The overall test set is denoted as Y^u , where $Y^u = (AO)^* \cup AO^* \cup A^*O \cup A^*O^*$. Note that train and test set compositions are mutually exclusive, *i.e.*, $Y^s \cap Y^u = \emptyset$.

3.3.2 Methodology

Our architecture is based upon common state-of-the-art baselines in CZSL (mostly OADis [147]).

We extract image and textual features using pre-trained networks (ResNet18 [65] and BERT [31] respectively). The pair embedder is LabelEmbedder [119] (LE) and the Object-Attribute Disentanglement module is from OADis [147], as shown in Figure 3.2. The Disentanglement module separates the visual features for attribute and object.

System Architecture. Similar to OADis [147], we use the second last layer before AveragePool of pre-trained ResNet18 [65], for spatial features $f_I \in \mathbb{R}^{512 \times 49}$. The Label Embedder module (MLP) extracts final feature v_{AO} for pair embedding, with same dimension as the word embedding final feature w_{AO} , extracted from BERT [31] Text Features (Figure 3.2). The disentanglement module separates the attribute and object visual features using backbone features for I , I_{attr} and I_{obj} , where I_{attr} is image with same attribute as I , and I_{obj} is an image with same object as I . I

and I_{attr} are used to extract visual feature for attribute v_A . Similarly, visual feature for object v_O is extracted from I and I_{obj} . Using textual features (w_A and w_O), these visual embeddings for attributes and objects are regularized, *i.e.* given a seen image with label peeled apple, we push the visual embedding of this image closer to text embedding of the label. More details can be found in [147]. We use cross-entropy along with cosine similarity to get the final classification score for each attribute, object and pair, same as [118]. Let visual feature is v and text feature is w , y is the correct seen label, then the main classifier logits can be defined as (where δ is the temperature factor):

$$C(v, w) = \frac{e^{h(v, w)}}{\sum_{y \in Y^s} e^{h(v, y)}}$$

$$h(v, w) = \cos(v, w) = \delta \cdot \frac{v^T w}{\|v\| \|w\|} \quad (3.1)$$

3.3.3 Neighborhood Expansion Loss

Previous CZSL methods perform fairly well for OV-CZSL, however fail to generalize on totally unseen compositions set, *i.e.* A^*O^* . By using common techniques of data augmentation and learning rate decay leads to improved generalization on unseen compositions, but deteriorates performance on seen compositions. Hence the goal of novel *Neighborhood Expansion Loss* (NEL) is to balance the generalizability and learnability of the model. To achieve this, we leverage ideas from *label smoothing* and *label propagation*. Label smoothing is a regularization technique used to reduce overfitting. Although, as explained in [117, 196], it is not always helpful and often introduces noise in the system. We use label smoothing to make the model less confident with training classes, such that negative bias for unseen classes decreases, and the

model does not overfit on seen classes. Further, to transfer knowledge from seen to unseen concepts, we use label propagation techniques [206]. It is a graph-based method for semi-supervised learning [41, 158]. In a transductive setting, given labeled and unlabeled examples, label propagation defines a graph between samples, to connect unlabeled samples to potential labels. In this work [150], we use neighbors from external knowledge sources (open vocabulary) to learn unseen pairs from seen pairs.

With traditional cross-entropy loss, the model only learns to minimize the distance between the visual embeddings and text embeddings for each image and correct label (`peeled lemon`). However, for limited seen compositions, this strategy causes a negative bias against unseen pairs. In a way, it forces the model to never learn unseen pairs, since it is too confident for the correct label. To overcome this, we apply label smoothing, which reduces negative bias for unseen classes during training with NEL. Another problem is extending knowledge from seen to unseen compositions. For that, we use label propagation. For each seen pair, we find k nearest-neighbors N_k using word embeddings for labels (details in next section). These neighbors make the ‘open-vocabulary’ aspect of OV-CZSL. As shown in Figure 3.3, if seen pair is `peeled lemon`, we also learn visual embeddings for unseen compositions `sliced lime`, `peeled apple`, `peeled potato`, `sliced grapefruit` and `ripe lime` as well by minimizing the distance between visual feature for given image of `peeled lemon` with the textual embeddings of the 5 mentioned neighbors. This intuition is from human learning, that we can correlate similar looking objects using language, such as oranges are similar to lemons. With this new loss, we expand our vocabulary beyond seen classes, by correlating similar concepts using language priors. Note that these neighbors are weighted, and the distance between visual feature with its original label (`peeled lemon`) is smaller than distance with its neighbor embeddings.

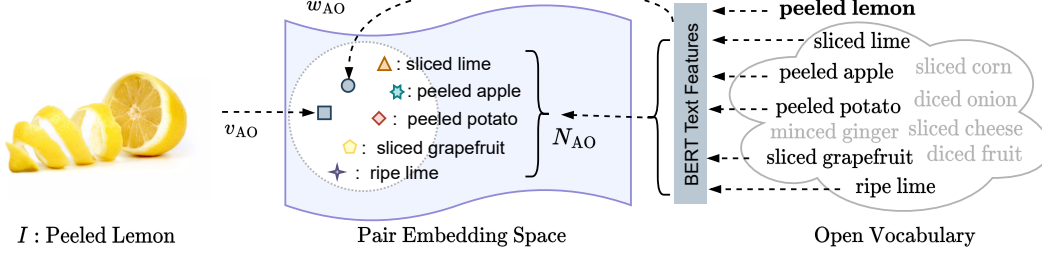


Figure 3.3: *Neighborhood Expansion Loss*. In the pair embedding space, we embed visual feature $v_{\text{attr,obj}}$ close to $w_{\text{attr,obj}}$. Further, we use $k = 5$ neighbors of correct label $w_{\text{attr,obj}}$, and make their word embeddings closer to the visual embedding $v_{\text{attr,obj}}$ as well. This gives our model the ability to generalize to unseen compositions.

so that the correct label is still learnt with higher confidence than it's neighbors.

Let N represent the neighbor set of seen pair text embedding w . For M training labels, cross-entropy is defined as H where, y_m is 1 for the correct class and 0 for the rest. Label smoothing [117] makes the model less confident for the correct label, by weighting the loss for correct label with a smoothing factor $\alpha < 1$. Thus cross-entropy between the modified targets y_m^{LS} and the network's outputs C_m is minimized.

$$H(y, C) = \sum_{m=1}^M -y_m \log(C_m)$$

$$y_m^{LS} = y_m(1 - \alpha) + \alpha/M \quad (3.2)$$

The *Neighborhood Expansion* makes the target for actual label is weighted highest, and rest weights are distributed among neighbors, with least weights are assigned to other labels.

$$y_m^{NE} = y_m(1 - \alpha)T + y_k(1 - \alpha)(1 - T) + \alpha/(M + k)$$

where, k is the number of neighbors, y_k are labels for neighbors from open-vocabulary, T is smoothing term for label propagation (weighting neighbors) and α is smoothing term for label

smoothing. Thus, *Neighborhood Expansion Loss* is a combination of label propagation and label smoothing. We use Cosine Similarity-based cross-entropy H classification loss and *Neighborhood Expansion Loss* for attribute, object, and pair embeddings, represented by Attr Cls, Obj Cls, and Pair Cls shown in Figure 3.2. Here, we elaborate on the Pair Cls loss:

$$\begin{aligned}\mathcal{L}_{\text{AO}} &= H(y_{\text{AO}}, C(v_{\text{AO}}, w_{\text{AO}})) \\ \mathcal{L}_{\text{AO}}^{\text{NE}} &= H(y_{\text{AO}}^{\text{NE}}, C(v_{\text{AO}}, w_{\text{AO}}))\end{aligned}\tag{3.3}$$

We can define similar loss functions for Attr Cls and Obj Cls as well. For each embedding space, we use both losses, and overall objective function $\mathcal{L}$ is:

$$\begin{aligned}\mathcal{L}_{\text{pair}} &= \beta_1 \mathcal{L}_{\text{AO}} + (1 - \beta_1) \mathcal{L}_{\text{AO}}^{\text{NE}} \\ \mathcal{L}_{\text{attr}} &= \beta_2 \mathcal{L}_{\text{A}} + (1 - \beta_2) \mathcal{L}_{\text{A}}^{\text{NE}} \\ \mathcal{L}_{\text{obj}} &= \beta_3 \mathcal{L}_{\text{O}} + (1 - \beta_3) \mathcal{L}_{\text{O}}^{\text{NE}} \\ \mathcal{L} &= \mathcal{L}_{\text{pair}} + \gamma_1 \mathcal{L}_{\text{attr}} + \gamma_2 \mathcal{L}_{\text{obj}}.\end{aligned}\tag{3.4}$$

3.4 Experimental Setup

3.4.1 Dataset

Following CZSL works, we propose new splits for OV-CZSL on existing dataset MIT-States [76], C-GQA [118] and VAW-CZSL [131, 147]. MIT-states [76] is relatively small, is a popular choice for CZSL. It has 115 attributes, 245 objects, 1962 compositional pairs, and 53k images. C-GQA [118] a larger dataset with 413 attributes, 674 objects and a total of $\sim 7k$ pairs

Table 3.1: **Dataset Splits.** We denote $*$ for unseen concept, such that A and O are seen attribute and objects, whereas A^* and O^* denote unseen attributes and objects. AO and $(AO)^*$ are seen and unseen compositions or seen attributes and seen objects respectively. AO^* are seen attribute-unseen object pairs, A^*O are unseen attribute-seen object set and A^*O^* are the unseen attribute-unseen object pairs. We propose new benchmark splits for OV-CZSL on datasets MIT-states [76], C-GQA [118] and VAW-CZSL [131, 147].

	Attributes		Objects		Training Set	Validation Set	Test Set
Datasets	A	A^*	O	O^*	AO	$AO/(AO)^*/A^*O/AO^*/A^*O^*$	$AO/(AO)^*/A^*O/AO^*/A^*O^*$
MIT-states [76]	84	31	182	63	955	236 / 105 / 126 / 177 / 44	289 / 130 / 157 / 218 / 50
C-GQA [118]	311	102	504	170	4094	1012 / 447 / 525 / 517 / 147	1239 / 542 / 664 / 655 / 176
VAW-CZSL [131]	330	135	406	110	7142	1767 / 803 / 1420 / 1253 / 412	2161 / 982 / 1737 / 1532 / 504

and 39k images. VAW-CZSL [131, 147] is the largest of all, with 533 attributes, 543 objects, $\sim 15k$ (15785) pairs and 92k images. The scale for C-GQA and VAW-CZSL is similar, however, VAW-CZSL has more shared attributes among objects. C-GQA has more one-to-one attribute object pairing, without much sharing of attributes among objects. We do not use UT-Zappos [195] since it only has 16 attributes and 12 objects. More details about dataset creation and validation can be found in the Appendix 10.

3.4.2 Neighborhood list

Since some datasets are small, and the testing pair labels might not be semantically similar to training pairs. To expand the vocabulary, we use external knowledge sources. In total, we use 2294 attributes/adjectives and 4090 objects aggregated from Visual Genome [85], Flickr30k [194], COCO-captions [23], and LocalizedNarratives [134]. These attributes and object pairs make up ~ 118650 compositions, as valid extra compositions we use for neighborhood search. We find 10 neighbors for each seen attribute and objects from the external source, using GloVe [129] embeddings cosine similarity score. Using the neighbors for attribute and object, we find all possible compositions (~ 100) for each pair. The compositions are then analyzed for validity using the external pair list described above. 10 valid compositions are chosen as neighbors for the

seen pair. If the external pair list does not have any compositions, we only choose compositions of first 6 closest attributes and objects from the neighbor list. Hence, neighbor search is based on individual attribute and object, instead of pairs directly. More details on for neighbor search and hyperparameter sensitivity of NEL are explained in Ablation section 3.6.

3.4.3 Evaluation

We follow Generalized CZSL [95, 106, 135, 147] evaluation protocol, to evaluate on both seen and unseen pairs (Y^s, Y^u) with a scalar term used to overcome negative bias for unseen pairs. Since, the OV-CZSL task is already challenging, we use Closed world evaluation setup (mentioned in related work), where AUC is reported over only “valid” unseen pairs while ignoring the “invalid” pairs. Area Under the Curve (AUC) is computed between the accuracy on seen and unseen compositions with different bias and Harmonic Mean (HM), to balance the bias. We also report separately accuracy for each set $(AO, (AO)^*, AO^*, A^*O, A^*O^*)$ at the bias-term where HM is maximum. The seen pairs Y^s consist of AO whereas the unseen pairs Y^u are $\{AO^*, A^*O, A^*O^*, (AO)^*\}$. Following [135], best accuracy values are reported for seen and unseen pairs.

3.4.4 Implementation Details

Following standard practice in CZSL [106, 118, 119, 135], we use Frozen ResNet18 [65], pre-trained on ImageNet [30] for image features (without finetuning) and BERT [31] text embeddings for labels. A linear layer on top of BERT [31] features is used for pair embeddings, instead of using Object-Conditioned module from OADis [147]. We use image augmentations

(random crop, horizontal flip) for all baselines and our method, following OADis [147]. For MIT states, smoothing factor is $\alpha = 0.8$ for MIT-states [76] and the temperature for cosine similarity $\delta = 0.05$. For C-GQA [118] and VAW [131, 147] as well, the smoothing factor for label propagation $T = 0.5$, number of neighbors $k = 5$, weights for losses are $\beta_1 = 0.8, \beta_2 = \beta_3 = 0.95$, and $\gamma_1 = \gamma_2 = 0.05$. The model is trained for 150 epochs and best performance based on validation *AO* performance. OADis [147] also has losses $\mathcal{L}_{seen}$ and $\mathcal{L}_{unseen}$, which were originally weighted with 0.1. We keep this the same for MIT-States [76], but change the weights for C-GQA [118] to 0.2. Smoothing factor α for C-GQA [118] and VAW-CZSL [131, 147] is 0.5 and 0.8 respectively. Learning rate for C-GQA [118] is 1e-4, with weight decay 5e-5 and Adam optimizer, decay milestone [20-130]. For VAW-CZSL [131, 147], it has same configuration as C-GQA [118] with learning decay milestone as [30,40]. We acknowledge that there are multiple hyperparameters in our setup. However, we mainly tune the hyperparameters for smoothing factors (α and T) and number of neighbors. Other hyperparameters are almost same as OADis [147].

3.5 Results

3.5.1 Quantitative Results

3.5.1.1 CZSL Baselines

The main task for OV-CZSL is compositional learning for seen and unseen attribute-object pairs. Our architecture is most similar to OADis [147], with embedding space and losses from LE [119] and CompCos [106]. Moreover, some recent works in CZSL includes SCEN [154] and CANet [173]. We compare with these 5 baselines and show accuracy for each individual split.

Table 3.2: **Results on MIT-states [76] and CGQA [106].** We report Top 1 AUC, which balances % between seen and unseen compositions with different bias terms. HM is Harmonic Mean where maximum AUC is computed. Following [135], best accuracy values are reported for Seen AO and Unseen pairs $\{AO^*, A^*O, A^*O^*, (AO)^*\}$. All other accuracies for individual splits are computed with bias where HM is maximum. Our method with NEL loss outperforms previous methods on most unseen compositions.

Model	MIT-States									C-GQA								
	Test@1	HM	Seen	Unseen	AO	$(AO)^*$	A^*O	AO^*	A^*O^*	Test@1	HM	Seen	Unseen	AO	$(AO)^*$	A^*O	AO^*	A^*O^*
LE [119]	1.01	7.64	16.29	9.46	10.24	11.38	5.98	4.15	2.87	1.17	8.39	19.37	8.36	10.76	6.51	9.53	2.67	1.08
CompCos [106]	1.97	10.22	26.53	10.29	14.32	21.09	5.86	2.89	0.63	2.35	9.64	40.19	7.25	21.19	20.24	4.47	1.95	0.26
OADis [147]	1.83	9.55	25.35	10.79	12.18	16.06	6.40	5.41	1.34	2.33	9.74	42.88	7.12	20.86	15.19	6.17	3.47	0.61
SCEN [154]	1.73	9.72	22.08	8.25	11.85	30.02	3.82	0.33	0.08	1.97	9.03	41.65	7.83	20.65	21.42	3.61	1.08	0.05
CANet [173]	2.40	10.52	26.42	9.54	16.56	23.08	6.15	4.08	0.58	3.04	11.96	40.52	9.21	22.43	20.87	4.95	2.03	0.64
Ours	2.41	10.94	29.02	11.13	14.11	18.87	8.24	5.49	3.54	3.18	12.11	42.38	9.77	19.78	16.07	12.86	2.87	3.04

Other baselines for CZSL, (*e.g.* TMN [135], AttrOpr [119], GraphEmb [118]) are either outperformed by CompCos [106] or perform poorly on unseen pairs (*e.g.* KG-SP [79], Symnet [95]). Note that, even though the external open vocabulary may or may not have any labels from our test set, we explicitly do not include test classes in the external knowledge sources. Hence, baselines which have any test labels while training (GraphEmb [118]) are not compared with, since they have use test labels while training. Although, SCEN [154] originally uses DINO [16] image features and word2vec word embeddings. To be fair with our method, we use ResNet18 [65] features instead of DINO [16]. CANet [173] uses ResNet18 [65] backbone along with Fasttext+w2v [13] text embeddings, originally. We use BERT [31] embeddings instead for comparing with our methods. **Results on MIT-States and C-GQA.** We report the AUC and harmonic mean (HM) for Top1 predictions on both datasets in Table 4.1. Seen and Unseen accuracy is the overall best accuracy without bias calibration. Interestingly, LE [119] generalizes well on unseen compositions, but does not perform as well for seen compositions overall. Moreover, CompCos [106] works the best for CZSL task, as its performance for AO and $(AO)^*$ remains unbeatable. However, it fails to generalize on compositions of unseen attributes or objects. OADis [147] is somewhere in the middle but does not generalize as well on unseen compositions A^*O^* . SCEN [154] is slightly

Table 3.3: **Results on VAW-CZSL** [131, 147]. All measures are shown in Top 3 AUC. HM is Harmonic Mean where maximum AUC is computed. All other accuracies for individual splits are computed with bias where HM is maximum. Our approach outperforms previous baselines for unseen compositions.

Model	Test@3	HM	AO	$(AO)^*$	A^*O	AO^*	A^*O^*
LE [119]	1.49	8.27	15.62	10.48	5.79	2.78	0.98
CompCos [106]	2.69	10.68	20.21	20.58	5.04	2.48	0.5
OADis [147]	2.68	10.91	21.19	15.65	6.75	3.16	0.76
SCEN [154]	2.53	10.64	19.06	20.76	4.52	2.05	0.42
CANet [173]	2.89	11.21	24.56	18.42	5.74	2.86	0.95
Ours	2.91	11.35	23.02	16.18	7.86	3.37	1.36

better than OADis [147] for unseen compositions of seen concepts $[(AO)^*]$, but yields very poor results for the unseen categories $[A^*O^*]$. CANet [173] yields better performance in Seen categories $[AO]$ and unseen compositions of seen concepts $[(AO)^*]$. However, it fails to generalize on the totally unseen categories $[A^*O^*]$. Our proposed approach gets best of both world, as it performs close to CompCos [118] and OADis [147] for CZSL task and beats LE [119] for unseen compositions, corresponding to A^*O , AO^* and A^*O^* .

For C-GQA [118], there are less attributes that share a common object (and vice-versa), which makes object-attribute disentanglement inefficient on this dataset. It relies more on backbone visual features (biased for objects), and performs well on A^*O than on AO^* . Our method with NEL surpasses most unseen sets with significant margin. Similar to MIT-states [76], SCEN [154] does better on unseen compositions of seen concepts, while CANet [173] does better on seen compositions AO . Overall, in comparing SCEN, CANet and our method, for CZSL compositions AO and $(AO)^*$, the drop in accuracy is $< 5\%$ while the improvement in A^*O^* is almost 5 times, which is the most challenging split of unseen attribute-unseen object compositions.

Results on VAW-CZSL. This dataset created from multi-label VAW [131], and VAW-CZSL [147]

Table 3.4: **Comparison with ZSL baselines.** We show results on two ZSL baselines for seen and unseen attributes and objects. Using NEL significantly improves outperforms existing ZSL baselines.

Model	Emb.	A	A^*	O	O^*
SEKG [176]	GloVe	5.04	3.10	5.19	3.17
	BERT	3.37	2.33	5.72	2.29
TF-VAEGAN [122]	GloVe	5.86	5.76	5.93	4.52
	BERT	4.54	3.05	5.82	3.93
Ours	GloVe	20.75	13.67	32.19	7.86
	BERT	20.37	11.42	29.3	10.07

uses the least frequent labels for each image across the dataset. Hence top 1 predictions latch on to different attributes which are present in the image, but are not labeled. We show Top 3 AUC, HM and accuracy for this dataset. Similar variations in performance of baselines are observed for this dataset as well Table 4.2, *i.e.* SCEN performs better in $(AO)^*$, CANet performs slightly better for AO while our method outperforms all baselines in every other set, majorly the unseen compositions splits A^*O , AO^* and A^*O^* .

3.5.1.2 ZSL Baselines

Zero-shot Learning (ZSL) for unseen attributes and objects is a by-product of OV-CZSL. Hence, we also compare with some baselines from ZSL. There are various ZSL methods, but most of them use pre-defined semantics for objects [51, 176], AWA [90], CUB [170], Imagenet [30]. Our work only relies on text label embeddings and visual features, therefore we compare with two recent works TF-VAEGAN [122] and CE-GZSL [59]. Note that these works do not have any information about unseen test classes during training, which might or might not be part of external knowledge but are not explicitly added to the open vocabulary. These methods use

semantic features, which we replace with GloVe [129] and BERT [31] text embeddings. We show results for MIT-States [76] as we observe similar pattern for other datasets as well. All the values are bias calibrated, to balance seen and unseen accuracy, for Top-1 predictions. Our proposed method outperforms common baselines in ZSL as well, with a significant margin. More details on why GloVe [129] embeddings perform the best are explained in the next section.

3.5.2 Qualitative Results

We show qualitative results on the same datasets. In Figure 3.6, we first show for different sets (seen and unseen pairs), examples of top 3 predictions for an image. All the predicted labels are part of unseen compositions Y^u . Although predictions are not always right, they still represent the concepts present in the images. The current evaluation protocol do not evaluate based on if either of attribute or object is correctly identified, the image should be considered partially correctly classified in compositional task. Another aspect we observe is that some attributes are also present in the current vision+language models are tied to correct labels, even if other labels also are correct. For instance, “blunt blade” in row 1 Figure 3.6, A^*O is also “blunt sword”. All swords have blades, but with separate labels the task is harder. We urge the community to direct the dataset creation such that visual concepts are classified as one categories, as opposed to language driving the categorization of visual concepts.

Further, we show some qualitative results to highlight that our model can still learn and generalize to out-of-domain data. In instance, in Figure 4.5 (a), We show the 5 nearest neighbor images from VAW-CZSL [131, 147] dataset, using the composition feature labels from model trained on MIT-states [76]. Similarly, (b) shows that if model is trained on VAW-CZSL [131,

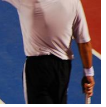
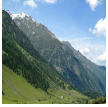
	$(AO)^*$	A^*O	AO^*	A^*O^*
MIT-States	 ancient clock engraved metal engraved coin	 blunt sword blunt blade curved blade	 broken laptop old laptop broken computer	 cooked fish cooked salmon sliced fish
C-GQA	 brown trunk large ear black tail	 large mirror gold mirror framed mirror	 young girl blond girl white shirt	 white horse white carriage standing horse
VAW-CZSL	 clay court running player wide court	 multicolored kite flying kite colorful kite	 white t-shirt dressed man baggy shorts	 grassy hill brown mountain tall mountain

Figure 3.4: Qualitative Results: We show results on MIT-States [76] for different sets, examples with top 3 predictions for an image. Most incorrect labels are also from unseen compositions Y^u , and although capture the attributes in the images, but are not considered correct. Labels in green are the correct annotation. We also observe redundancy in labels, such as row 1, column 2 blunt blade in A^*O . Top prediction also includes blunt sword. All swords have blades. However, having separate labels for these makes the task harder.

147], it’s nearest neighbors in feature space for various unseen compositions make sense, when retrieved from MIT-sates [76]. Moreover, we observe that for all compositional sets, $\{(AO)^*, AO^*, A^*O, A^*O^*\}$, the model learn the attribute and object features to generalize to totally unseen images from different dataset.

3.6 Ablation Studies

We experimentally motivate the design choices as well as novelty aspect of our approach. All the experiments in this section are shown for MIT-States [76] dataset. One of the most challenging set in OV-CZSL is A^*O^* , hence a lot of design choices are based mainly upon improving performance for this unseen split. More details can be found in suppl.

3.6.1 Neighborhood Expansion Loss

As the main contribution of this work is OV-CZSL and *Neighborhood Expansion Loss*, we explore if this loss can be used as plug-and-play for other baselines. In Table 3.5, third row for

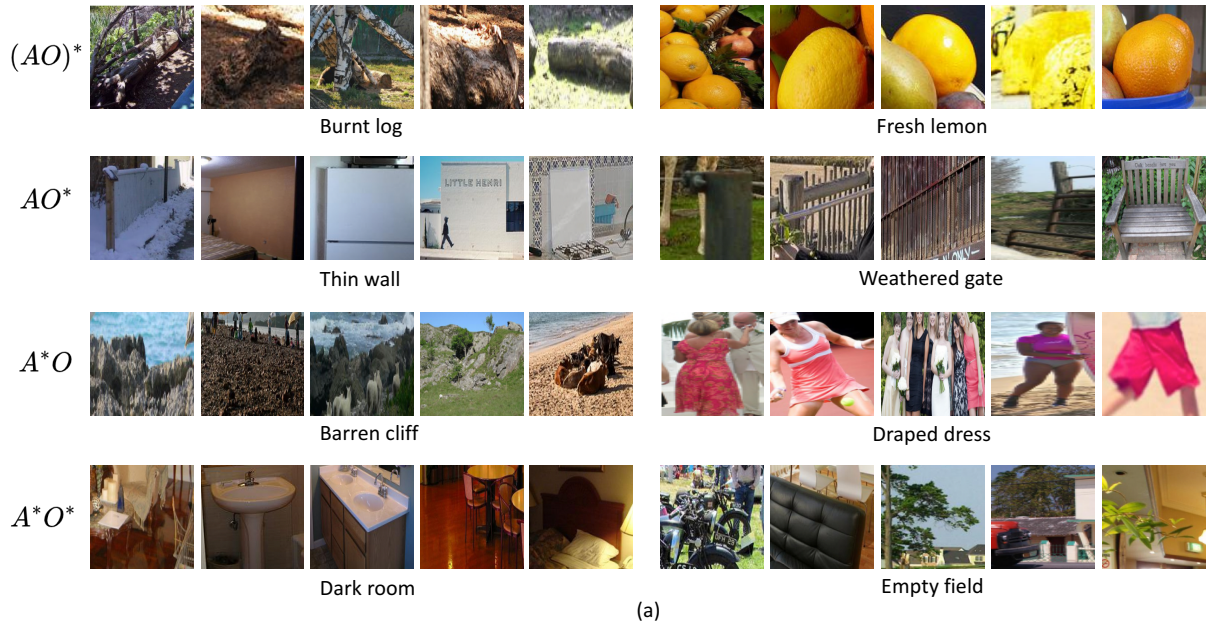
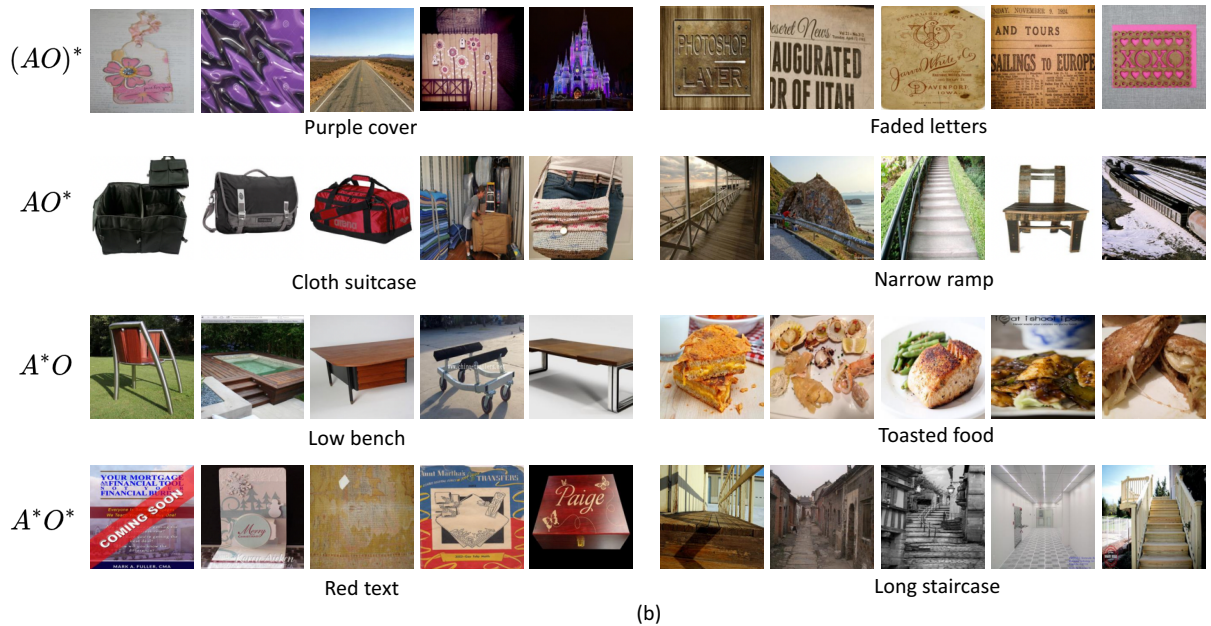


Figure 3.5: **Qualitative results for Cross-Dataset Nearest neighbors.** (a) We show the 5 nearest neighbor images from VAW-CZSL [131, 147] dataset, using the composition features from model trained on MIT-states [76]. Similarly, (b) shows that if model is trained on VAW-CZSL [131, 147], it’s nearest neighbors in feature space for various unseen compositions make sense, when retrieved from MIT-sates [76].

each model, shows the change caused by using NEL with respect to originally without NEL. **red** denotes negative change in value, **green** denotes the positive change and ‘-’ represents no or change is less than ± 0.2 . All methods with NEL improve significantly for A^*O^* split, whereas OADis [147] and CANet [173] improves for all splits. This is because NEL is applied for not just pair embedding space, but also for attribute and object spaces, which are not present for Comp-Cos [118] and LE [119]. Although, as mentioned earlier, CLIP [138] makes the setup not-ZSL, we still acknowledge the presence of such larger models and show the importance of NEL loss. The loss can be applied along with CLIP [138] loss function, and slightly improve generalization for the unseen compositions. Despite NEL being a small modification and the CLIP loss has more stronger impact, we show the potential to harness bigger models to generalize beyond seen classes. We expect similar boost for other Vision+Language models such as LLaVA [97] and BLIP [93] along with NEL, while showing results with CLIP [138] as a representative sample.

3.6.2 Evaluating different design choices

3.6.2.1 Number of Neighbors.

Using too many noisy labels in label smoothing can affect the performance adversely [117, 196]. In our case, more neighbors can act as noisy labels. We experiment to find the ideal number of neighbors, as shown in Table 3.6. Among all numbers, our model achieves best performance using 5 neighbors on MIT-States [76] and C-GQA [118]. For VAW-CZSL [131, 147], we use 10 neighbors for best performance.

Table 3.5: **Using NEL with other baselines.** We show effect of NEL for different baselines. All methods using NEL perform better for OV-CZSL splits.

Model	AO	$(AO)^*$	A^*O	AO^*	A^*O^*
LE [119]	10.24	11.38	5.98	4.15	2.87
LE + NEL	10.65	6.11	5.61	4.09	7.71
	+0.4	-5.2	-0.3	-	+5.5
CompCos [118]	14.32	21.09	5.86	2.89	0.63
CompCos + NEL	15.73	19.72	8.03	4.13	1.77
	+1.4	-1.3	+2.1	+1.24	+1.1
OADis [147]	12.18	16.06	6.40	5.41	1.34
Ours (OADis+NEL)	14.11	18.87	8.24	5.49	3.54
	+2.9	+2.8	+1.8	-	+2.2
CANet [173]	16.56	23.08	6.15	4.08	0.58
CANet [173] + NEL	19.24	25.94	6.78	5.12	1.53
	+2.7	+2.8	-	+1.0	+1.0
CLIP [138] (ZSL)	20.08	22.03	23.38	23.21	34.93
CLIP+ FT	25.89	22.03	24.87	25.60	34.50
CLIP+FT+NEL	25.97	23.06	25.90	25.70	36.45
	-	+1.0	+1.3	-	+2.0

Table 3.6: **Ablation for varying number of neighbors (k).** We show how changing the number of neighbors can introduce noisy labels in the setup. For generalization to unseen classes, we need fewer neighbors (5 as shown here).

k	Val@1	Test@1	HM	AO	$(AO)^*$	A^*O	AO^*	A^*O^*
1	2.67	2.21	10.46	15.16	17.46	8.02	4.41	2.21
3	2.69	2.18	10.39	16.37	16.71	7.67	4.47	2.21
5	2.69	2.41	10.94	14.11	18.87	8.24	5.49	3.54
7	2.65	2.24	10.44	15.16	17.67	8.09	4.50	2.37
10	2.60	2.16	10.35	13.79	18.33	8.01	4.61	2.14

Table 3.7: **Ablation on different Text Embedding.** We show effect of using different text embeddings for OV-CZSL. We observe that BERT [31] improves the unseen compositions A^*O^* the most, hence we use BERT [31] for our approach.

Emb	Test@1	HM	AO	$(AO)^*$	A^*O	AO^*	A^*O^*
GloVe	2.37	10.46	13.13	19.11	10.70	3.83	0.69
word2vec	2.37	10.68	13.82	17.46	10.61	4.19	1.13
fasttext	2.33	10.14	13.02	17.04	10.93	4.88	1.64
BERT	2.41	10.94	14.11	18.87	8.24	5.49	3.54

3.6.2.2 Textual Features

Textual features are the source of regularization and knowledge transfer, which makes these a crucial design decision. We observe that for zero-shot learning task, GloVe [129] performs better than BERT [31] embeddings. Since BERT is contextual embeddings, attribute-object embeddings from BERT are more helpful than GloVe. We also compare with other common word embeddings like Fasttext [13] and word2vec [113] in Table 3.7. Most word embeddings perform better for unseen compositions of seen attributes-objects $(AO)^*$ and A^*O . We use BERT for all baselines of OV-CZSL since it outperforms for the most challenging set A^*O^* .

3.6.2.3 Choosing right Neighbors

We use GloVe [129] embeddings for neighbor selection for two reasons: (1) since these are word embeddings, they capture attribute features and it’s neighbors more efficiently. (2) GloVe embeddings have strong semantic structure and arithmetic qualities (*e.g.*, $\text{emb}(\text{king}) + \text{emb}(\text{women}) - \text{emb}(\text{man}) = \text{emb}(\text{queen})$). Since, BERT [31] are contextual embeddings, it does not necessarily follow this structure. GloVe [129] embeddings for single word are more robust and have algebraic expression in embedding space, which is why GloVe seemed optimal for

Table 3.8: Ablation for different smoothing factors: We set different smoothing factors α for $\mathcal{L}_{AO}^{NE}$ with dataset MIT-states [76], to empirically find the best value for α .

Emb	Neighs	Test@1	HM	$(AO)^*$	AO^*	A^*O	A^*O^*
BERT	External.	1.71	9.56	12.17	3.24	4.87	2.07
GloVe	MIT-states	2.13	10.16	17.94	4.25	7.67	2.40
GloVe	External.	2.41	10.94	18.87	5.49	8.24	3.54

neighbor search. For example, when we compute neighbors for object ‘elephant’, GloVe [129] features gives neighbors like: [rhinoceros, rhino, animal, whale] as top neighbors, which are all large animals and make sense. However, using BERT [31] embeddings produces top neighbors: [monkey, lion, camel, lizard, mermaid], which has low similarity with the ‘elephant’ in general, apart from the fact all are animals. Moreover, we quantitatively show that using GloVe [129] embeddings for searching for neighbors gives better performance in comparison to using BERT [31] in Table 3.8.

Another idea is using neighbors from the dataset itself. For this, instead of using external sources of attributes and objects, we only use the unseen attributes and objects to find neighbors from. For smaller dataset like MIT-states [76], these neighbors do not make much sense but on a higher level correlate seen attributes with unseen attributes explicitly. However, the generalizability of model gets affected, as the unseen attribute-unseen object pairs accuracy is better if external sources of attributes and objects are used (as shown in Table 3.8). The column 1 shows the embeddings used for neighborhood search and column 2 refers to is the neighbors are extracted from external sources or within the dataset.

3.6.2.4 Impact of Smoothing Factors

In our work, we use two smoothing facts: α and T . α is the smoothing term for label smoothing, whereas T is the smoothing term for neighbors in label propagation. We explain the significance of both separately. To have a balance between generalization and learnability, we empirically find the smoothing factor for MIT-States [76] dataset, and use the same for other datasets.

First, for label smoothing α . Let us assume there are total K labels. For each sample, one of these K labels is correct. Here in Table 3.9 we show the smoothing factor changes for MIT-States [76] dataset. Note that for $\alpha = 1$, $\mathcal{L}_{AO}^{NE}$ is same as $\mathcal{L}_{cls}$. Hence, the range to vary α is between $[0,0.9]$. With $\alpha = 0.8$ in Table 3.9, the correct label logit is weighted with factor $1 - \alpha = 0.2$, whereas other logits are weighted with $0.8/(K - 1)$.

For label propagation among n neighbors, we use $T = 0.5$, which means that the correct label is weighted with 0.5, as the rest neighbors are weighted with $0.5/n$. Hence, α and T are both smoothing factors but for different purposes. Moreover, the lower the smoothing factor, the more weightage is given to neighbors/other labels, and leads to higher accuracy for unseen attribute-unseen object pair. NEL uses both label smoothing and propagation smoothing factors. Using higher value of T leads to better generalizability but less learnability. With higher T , generalizability reduces, since more weightage is given to the single correct label. In Table 3.9, we fix each smoothing factor and vary the variable for finding the best hyperparameters.

Table 3.9: Ablation for different smoothing factors: We set different smoothing factors α and T for $\mathcal{L}_{AO}^{NE}$ with dataset MIT-states [76], to empirically find the best value for α and T .

α	Test@1	HM	$(AO)^*$	AO^*	A^*O	A^*O^*
0.3	2.22	10.38	13.25	3.85	11.11	5.18
0.5	2.34	10.56	16.75	5.67	7.56	4.16
0.8	2.41	10.94	18.87	5.49	8.24	3.54
0.9	2.30	10.87	17.49	4.85	8.12	2.79
T						
0.2	2.38	10.64	18.74	5.07	8.46	2.90
0.5	2.41	10.94	18.87	5.49	8.24	3.54
0.7	2.37	11.06	18.72	5.10	8.37	3.15
0.9	2.37	10.97	18.63	5.01	8.65	3.09

3.6.2.5 Different ratio of NEL and main loss

In equation 4, γ_1 and γ_2 control the weights for attributes and object losses. We follow same values of γ_1 and γ_2 , as mentioned in OADis [147]. Moreover, the total loss for each embedding space is weighted sum of Cosine Cross-Entropy and Neighborhood Expansion Loss as shown in equation 4. We show experiments for β_1 in Table 3.10. Further values of β_2 and β_3 also follow the similar results. Based on eq 4. the weight for $\mathcal{L}_{AO}^{NE}$ is $1 - \beta_1$. The results show that giving more weight to Neighborhood Expansion Loss $\mathcal{L}_{AO}^{NE}$, harms the seen pair accuracy while keeping unseen pair accuracy. We use 0.8 as final value as it is a trade-off between seen and unseen accuracy. Since there are various unseen composition splits to evaluate on for OV-CZSL, *i.e.* $\{(AO)^*, AO^*, A^*O, A^*O^*\}$, we select the α based on only Test AUC and HM. AUC automatically provides the best trade-off between the seen and unseen accuracy overall and provides the best balance between $\mathcal{L}_{cls}$ and $\mathcal{L}_{AO}^{NE}$.

Table 3.10: Ablation for weights for Neighborhood Expansion Loss: We set different β_1 , weights for neighborhood expansion loss for pair embeddings, for MIT-states [76] split 1. We show that using higher value of β_1

β_1	Test@1	HM	$(AO)^*$	AO^*	A^*O	A^*O^*
0.7	2.23	10.50	16.80	4.83	8.67	4.10
0.8	2.41	10.94	18.87	5.49	8.24	3.54
0.9	2.30	10.78	19.87	5.25	7.05	2.53

3.7 Can NEL help in learning abstract concepts?

Our inspiration for NEL is to learn to generalize similar concrete concepts (apple, banana) to all other abstract concepts (any fruits), such that if 1-2 fruits are seen in the training set, the model generalizes to other unseen fruits in the test set. To quantitatively verify this, we cluster all attributes and objects classes for MIT-states [76], using k-nearest neighbors, along with a threshold cosine similarity of 0.4 across GLoVe [129] features. We manually check the similarity between these label clusters and ignore the attributes and objects across clusters: (1) cluster size is less than 3 and (2) clusters which have classes that are only part of test set, without having any class in training set. Each cluster has some seen and unseen concepts. We compare OADis [147] and Our (OADis+NEL) model’s accuracy for these clusters. Attributes are spread across 11 clusters, and objects are spit into 26 clusters. As shown in Figure 3.6, each cluster is represented with two labels from the cluster to disclose the abstract concept of the cluster. We observe that most clusters either show same or improved performance while using NET (for our model), across all attributes and objects. This shows that similar concepts are learnt together to generalize to unseen abstract concepts. However, this setup is still limited, such that the model cannot discriminate similar concepts `peel` from `slice` and `chop`, but can only learn these together closer in the visual+textual embedding space as `styles of cutting`.

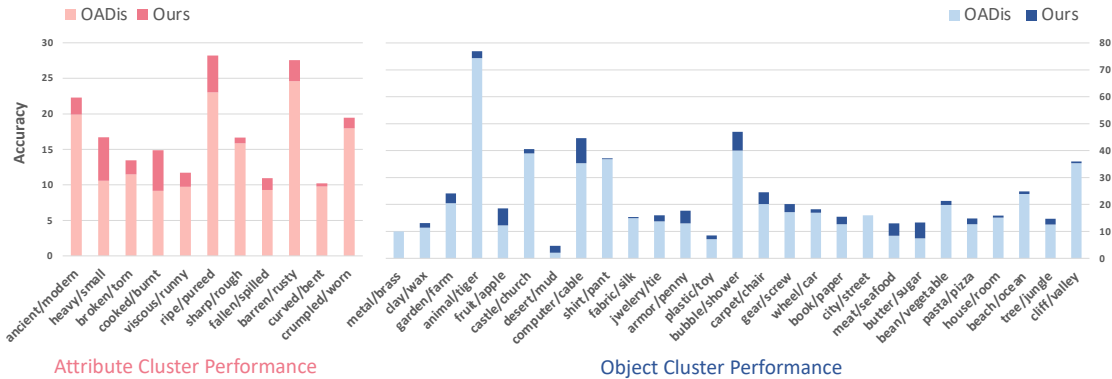


Figure 3.6: To show the impact of NEL, we cluster all attribute and object class labels for MIT-States [76], and evaluate the accuracy for each cluster. These clusters have both seen (ancient, new, young) and unseen (modern, old) concrete concepts, such that each cluster has one abstract concept (time/age). Our model (OADis+NEL) shows either same or improved performance for attributes and object clusters, backing our hypothesis of NEL generalizes to unseen concrete concepts for the similar abstract concepts.

3.8 Limitations

We emphasize that OV-CZSL is challenging setup since the problem to discriminating visually similar concepts is still restricted. For instance, a model can not discriminate `peel` from `slice` and `chop`. However, we propose this direction to at least learn that `peel`, `slice` and `chop` are all applied to fruits. Hence, if training set has `sliced apple`, in test set our model should be somewhat be confident in predicting `chopped pear`. With this current setting of OV-CZSL, we do not claim to differentiate `peeling` from `chopping`, but we propose that `sliced potato` and `chopped pear` are visually more closer than `sliced potato` and `raw apple`.

Moreover, our evaluation protocol only evaluates for valid compositions, also referred to as Closed-world setting which is a standard evaluation metric for CZSL benchmarks [135, 147]. The other evaluation setting uses all possible compositions for attributes and objects and discard the invalid compositions during testing. OV-CZSL is an extension of CZSL with using wider use

of ‘open-vocabulary’ for training. However, during testing, we only evaluate on valid test compositions, rather than for every possible composition like open-world setting. We quantitatively show that using open-world evaluation setting is even more challenging for OV-CZSL but can be potential direction for future research.

Large Language+Vision Models We emphasize that we acknowledge the presence of larger models, however choose to not include those in this study since the focus of this work is to learn beyond seen samples, while all these models already are trained on exponential amount of data such that no category from academic datasets is unseen for these models. Moreover, including CLIP [138], we show that our method can help these models improve on subtle attributes fine-tuned learning, but is still not useful for Zero-shot learning setup [150]. Hence, we want to clarify the scope and limitation of this work to urge the reviewers evaluate this work under the same constraints and limitations.

3.9 Conclusion

In the era of CLIP [138] and DALL-E [139], we emphasize that labeled data is still a bottleneck, which these bigger models dodge by extensively using all available labeled data. We present a novel task OV-CZSL, which not only focuses on learning unseen compositions of seen attributes and objects, but also generalizes to unseen attributes, unseen objects and their compositions. We propose new benchmark splits, backed by scientifically stable methods on existing datasets, MIT-states [76], C-GQA [118] and VAW-CZSL [131, 147]. Any CZSL model (or even LLM) can be extended with Neighborhood Expansion Loss, for solving OV-CZSL, through semantic transfer from pre-trained language embeddings. However, open vocabulary compositional

learning is still a challenging problem, with tremendous scope in improving benchmarks (datasets and evaluation). Therefore, we urge the community to further explore generalized compositional learning of attributes and objects.

Chapter 4: Attribute-Object Compositionality for Video Recognition

This chapter is organized in two parts: First part is about using sparse images of start and end object states to understand the underlying actions that causes the changes in the [146]. Second part discusses Compositional Action recognition task [148], which uses all frames in a video however the training and test data splits are based on seen and unseen compositional actions on different objects. Problem statements and approaches are defined in the two sections separately.

4.1 Recognizing actions using object states

4.1.1 Introduction

Let us play *Match the following* game in Figure 6.1, where we show a pair of images for an object capturing two different states (left: ‘start,’ right: ‘end’). Can you predict which action transforms the object in the left image to the object in the right image?¹ Consider the object in pair 5 (left) in Figure 6.1, where we show a lemon and pieces of lemon in the left and right images, respectively. It is relatively easy for humans to recognize that this particular state change (whole→pieces) of the lemon was likely caused by a “cut” action. Similarly, seeing the grated carrot in cutting-board at ‘start’ and in a pan at ‘end’ in pair 10 (Figure 6.1), we can likely predict that the action was “move.” How are we able to understand these action without seeing the video

¹ Answer-key: 1-b, 2-f, 3-e, 4-c, 5-a, 6-d

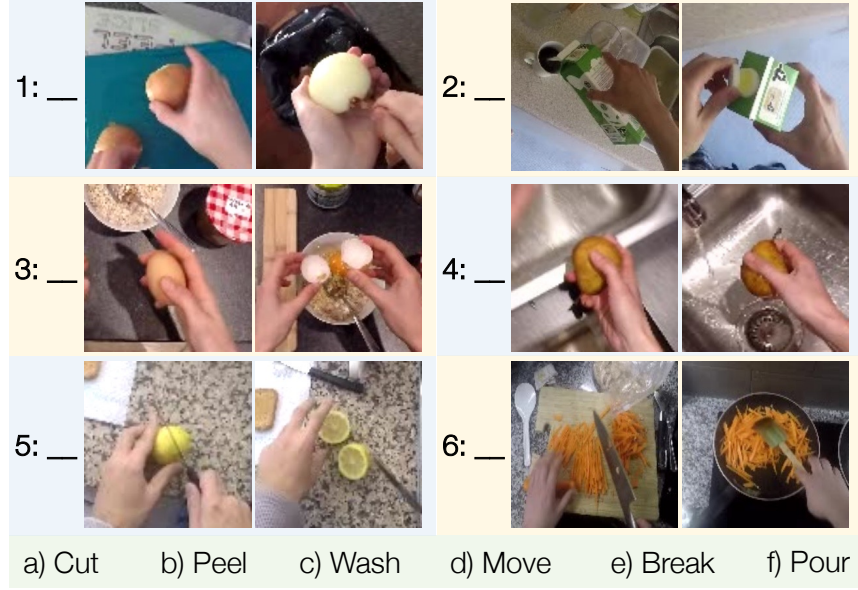


Figure 4.1: *Match the following*: In the given set of image pairs, we show start→end transformations in object states. Can you match the object states to an action, from list given below, that likely induced those transformations?

of these actions being performed? We hypothesize that most object-centric actions induce specific transformations in object states, where state includes object’s appearance *and* immediate context, and humans are adept at identifying such actions simply based on the state transformations.

In the commonsense reasoning literature, this notion of state transformation is also known as *fluent* of an object [43, 100, 116], defined as ‘the status of an object which varies over time.’ For instance, a lemon’s *fluent* changes from whole→pieces over time, associated with “cut” action. In this work, we propose a computational model that embodies this intuition of leveraging object state transformations, or *fluent* of an object, to reveal the underlying action being performed. In fact, similar to [39], we argue that recognizing most object-centric actions does not require much temporal information. We should be able to identify them just by looking at the object state transformations they induce. To test this hypothesis, we limit our focus to this question: Can we infer actions just from two distinct object states – ‘start’ and ‘end’? Note that we do

not claim that more intermediate states will not be beneficial for modeling transformations (*e.g.*, see [178]). The start→end state transformation offers many intuitive cues that an approach can rely on (Figure 6.1), but computationally modeling them is still quite challenging. Moreover, this can potentially yield computationally efficient architectures which use just two states to recognize object-centric actions.

As discussed earlier, an object’s state is defined by its visual appearance and its immediate context. For understanding the start→end transformation, we need to model two things: what changed and what stayed the same. For example, for “cut” in Figure 6.1, the object appearance changes (whole→pieces) but the context stays the same (counter). Alternatively, for “move,” the object appearance stays the same (grated carrots) but the context changes (board→pan). We refer to the actions that transform object’s visual appearance as *object-transforming actions* and actions that transform its context as *context-transforming actions*.

Ideally, any standard network architecture for videos could potentially learn to extract this information and recognize actions. However, modeling this is surprisingly challenging. The first issue is the presence of spurious correlations and biases in videos, due to which networks often take shortcuts and ignore the underlying state transformations [48]. The second issue is the low signal-to-noise ratio in videos; there are many high-res frames, and relevant information is present in very few pixels. Even though restricting the input to just two states of objects mitigates this to some extent, the information needed to recognize actions remains subtle. For example, when we look at object states in Figure 6.1, lemon’s whole→pieces transformation happens in few pixels, and even then, the color remains similar, and models need to focus on the shape. We argue that due to these issues, it is unreasonable to expect video architectures to implicitly learn to model transformations.

To address the issues of spurious correlations and subtle cues in understanding the state transformations, we propose two complementary **amplifiers** – *Affinity Amplifier* and *Differential Amplifier*. As the name suggests, the affinity amplifier learns to deal with spurious correlations and amplify representations for what stays the same in a transformation; and the differential amplifier learns to amplify the subtle changes that happen during transformation. We model both amplifiers using attention blocks with different operations designed to amplify relevant information. We jointly refer to these two complementary amplifiers, designed to focus on the *fluent* of an object, as ***fluent Attention***.

In this section [146], we demonstrate that we can recognize object-centric actions using the underlying object state transformations. Note that we do not assert that using object states is the right approach for generic video understanding and action recognition. Person-centric actions, such as sports activities and regular household activities, need more information than just two states/frames. However, for object-centric actions, we show that only using two states in our *fluent Attention* can outperform standard approaches using many more frames. Our algorithm first selects two bounding boxes, representing start and end states, and then extracts their representations. Our attention module uses these representations and learns a *fluent* representation used to classify actions.

Because of our object-centric focus, we use cooking video datasets: EpicKitchen55 [28], YouCook2 [207], and EpicKitchens100 [26]. To demonstrate the powerful generation of our approach in two ways. First, we train on just EpicKitchen55 and show that our model based on *fluent Attention* can generalize to YouCook2 and EpicKitchens100. Second, we hold out several unseen objects from our training set and show that we can generalize to transformations of these unseen objects. We show that our approach is able to improve performance for both *object-*

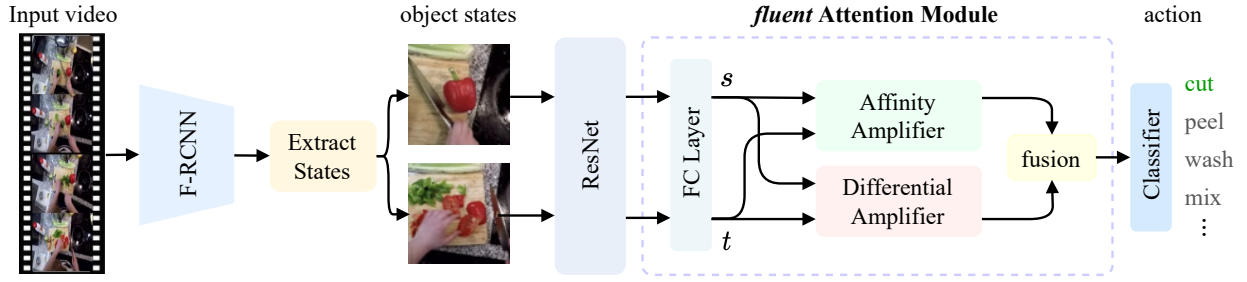


Figure 4.2: Our framework extracts start and end object states from the video. We use ResNet50 [65] pre-trained on ImageNet [30] as our backbone network. We introduce a novel framework to model object-transforming and context-transforming actions separately. The affinity amplifier is trained to learn what is similar between object states and the differential amplifier captures the dissimilarities between the states.

transforming and *object-transforming* actions. To summarize, our contributions are as follows:

(a) developing a *fluent* attention block with affinity and differential amplifiers, that can understand object state transformations, (b) demonstrating that amplifying relevant information from just two states is enough to outperform architectures that use several frames, and (c) demonstrating the generalization of our approach to unseen objects and unseen datasets.

4.1.2 Related Work

Object state transformations. Objects in the real world exist in different colors, shapes, geometry, *etc.* For instance, an apple can be red or green, cut or peeled, raw or ripe. The study of these ‘states’ and how they transform from one to the other has recently garnered interest in the computer vision community [7, 10, 20, 43, 76, 100, 114, 119, 121, 181, 190, 195, 208]. Understanding these states could enable an object recognition system to generalize better to unseen states, as studied in [10, 114]. However, most studies have focused on image-based datasets [76, 114, 195], where the goal is to capture the underlying distribution of states (*e.g.*, different sizes of heels in shoes, old and new laptops). Some of these states are just ‘attributes’ of objects (*e.g.*, red car and blue car) and are generally static over time; whereas other states are transformations of ob-

jects over time (*e.g.*, raw→ripe, whole→pieces, ‘on table’→‘on place’). These different types of states provide very different information. In this work, we leverage the state transformations that happen over time for understanding transforming actions. For the rest of this chapter, we will refer to changes in object states (including visual appearance and context) over time as ‘object state transformations.’

Object transformations for action recognition. In the action recognition community, most modern approaches can be grouped into two categories: (a) models utilizing temporal and spatial convolutions over videos to classify the actions [40, 52, 78, 96, 160, 172, 189, 205], and (b) models which rely on human-object interactions over time to recognize actions [58, 82, 155, 166, 183, 184]. In both categories, the focus is generally to leverage as much temporal information as possible; this is evident with the focus on recent works to utilize as many frames as computationally feasible to understand actions.

More recently, in the spirit of using fewer frames, TRN [205] proposed a way to reason about temporal dependencies between video frames at multiple time scales by hierarchically capturing temporal relationships between up to eight frames and predict actions. Similar to our approach, [9, 39, 108] propose using state transformations for action recognition; *e.g.*, [9] uses five key-frames for modeling transformations for action recognition. Closest to our approach is the work of Fathi and Rehg [39], which also uses two object bounding boxes as states for action recognition. However, our approach explicitly amplifies affinity and differential signals between states to extract *fluent* representations, which are used for action recognition.

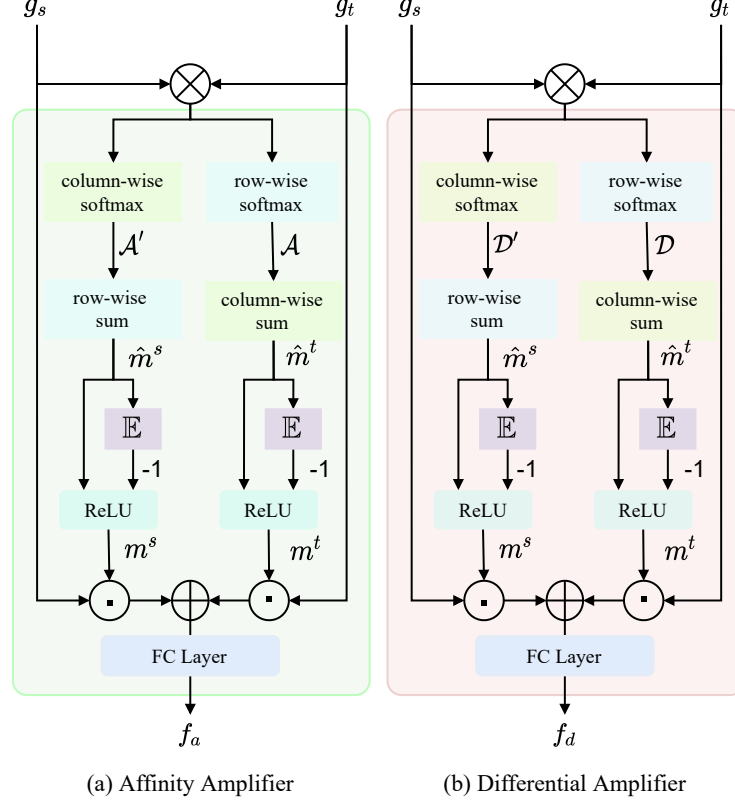


Figure 4.3: The architectural details of the (a) Affinity Amplifier and (b) Differential Amplifier.

4.1.3 Approach

In this work, our goal is to recognize object-centric actions using start→end object state transformations. A state transformation can correspond to *object-transforming actions* that change object’s visual appearance and *context-transforming actions* that change its context. Our approach is designed to amplify similarity between these states to suppress spurious signals and amplify the differences to leverage subtle changes between start and end states. Together, this framework is referred to as `fluent Attention`.

Given a video, we first describe our method to extract object states, or regions of the object in two distinct frames before and after the action has been performed. Then, we explain our proposed *fluent Attention Module*, with overall architectural details. An illustrative overview of

our approach is presented in Figure 4.2.

4.1.3.1 Extracting object states

As discussed above, analyzing object state provides useful information for inferring actions. For this purpose, we first identify two frames in a video, one before and the other after the action has been performed, corresponding to different states of the object. We choose the highest confidence bounding box, extracted using an object detector, of the object of interest from each frame as the inputs to our model. We refer to these two regions as ‘start’ and ‘end’ states.

In particular, we leverage the temporal annotations provided with the video to identify clips where the action is performed. We extract bounding boxes using an object detector on the first N frames of the first clip and the last N frames from the last clip, and choose the box with highest class confidence score for the object under consideration as ‘start’ and ‘end’ states, respectively.

4.1.3.2 State Representations

The input to our model are the ‘start’ ($\mathcal{I}_s$) and ‘end’ ($\mathcal{I}_t$) states. We use a pre-trained deep neural network to extract features f_s and f_t ($f_s, f_t \in \mathbb{R}^n$; $n = 2048$) from $\mathcal{I}_s$ and $\mathcal{I}_t$, respectively. We then pass f_s and f_t through an FC layer with ReLU non-linearity to obtain g_s and g_t ($g_s, g_t \in \mathbb{R}^n$). The features g_s and g_t are then passed through the *fluent* Attention block for further processing.

4.1.3.3 *fluent* Attention Block

The input to this module are the ‘start’ and ‘end’ state representations g_s and g_t . From literature [199], we know that the deep features capture various levels of semantic information from the input images. Our hypothesis is that the differences between g_s and g_t capture the *fluent* of the object. Hence, by modeling this explicitly, the model can efficiently reason about actions. To this end, our *fluent* attention block consists of two complementary amplifiers — *Affinity Amplifier* and *Differential Amplifier*. The affinity amplifier is trained to suppress the spurious correlations and amplify representations for the similarities between states; and the differential amplifier learns to amplify the subtle differences between states that can be used for classification. Both the amplifiers take g_s and g_t as input to produce affinity and differential representations f_a and f_d respectively. The two features f_a and f_d are then fused to produce the final representation.

We describe one of the amplifiers (the Affinity amplifier) in detail below and the other amplifier is implemented in a similar fashion with minor changes. Architecture diagrams for both amplifiers in Figure 4.3.

Affinity amplifier captures the similarity between the start and end states. We first take the outer product of g_s and g_t denoted as $O = g_s \otimes g_t$ ($O \in \mathbb{R}^{n \times n}$). Element $o_{i,j}$ represents the similarity between i^{th} element of g_s with j^{th} element of g_t . Moreover, let $\mathbf{o}_{i*}$ and $\mathbf{o}_{*j}$ represent the i^{th} row and j^{th} column of O respectively. Then, $\mathbf{o}_{i*}$ captures the similarity of all the elements in g_t with respect to i^{th} element of g_s . To know the most similar element among g_t with respect to i^{th} element of g_s , we can take a row-wise `softmax` over O . Similarly, for j^{th} element of g_t , column $\mathbf{o}_{*j}$ represents the similarity with all the elements of g_s . Using a column-wise `softmax`, we can interpret the most similar and least similar element of g_s with respect to j^{th} element of

g_t . Therefore, by applying column-wise and row-wise `softmax`, we get two matrices, $\mathcal{A}$ and $\mathcal{A}'$ ($\mathcal{A}, \mathcal{A}' \in \mathbb{R}^{n \times n}, n = 2048$).

$$\mathcal{A}[i, :] = \frac{e^{\gamma \mathbf{o}_{i*}}}{\sum_{j=1}^d e^{\gamma o_{ij}}} \quad \text{and} \quad \mathcal{A}'[:, j] = \frac{e^{\gamma \mathbf{o}_{*j}}}{\sum_{i=1}^d e^{\gamma o_{ij}}},$$

where γ is the inverse temperature parameter.

We compute row and column sum for $\mathcal{A}$ and $\mathcal{A}'$ respectively, to get raw state masks, $\mathbf{m}^t$ and $\mathbf{m}^s$ as

$$\hat{m}_j^t = \sum_{i=1}^d \mathcal{A}_{ij} \quad \text{and} \quad \hat{m}_i^s = \sum_{j=1}^d \mathcal{A}'_{ij}.$$

To get the final attention masks, we manually set the values of mask below average to zero. This can be achieved by leveraging the ReLU activation function as

$$m^s = \text{ReLU}(\hat{m}^s - \mathbb{E}[\hat{m}^s])$$

$$m^t = \text{ReLU}(\hat{m}^t - \mathbb{E}[\hat{m}^t])$$

We compute the final representation f_a of the affinity amplifier as

$$f_a = W^T([m_s \odot g_s; m_t \odot g_t]) + b$$

where $\odot$ is the elementwise multiplication and $W \in \mathbb{R}^{2n \times n}$ and $b \in \mathbb{R}^n$. Refer to Figure 4.3(a) for illustration of the affinity amplifier block. Similarly, we can get the differential representation f_d , except instead of using similarity between states ($\mathcal{A}$ and $\mathcal{A}'$), use distance measure ($\mathcal{D}$ and

$\mathcal{D}'$) computed as

$$\mathcal{D} = 1./\mathcal{A} \quad \text{and} \quad \mathcal{D}' = 1./\mathcal{A}'$$

where $1./*$ is the element-wise inversion of matrices. Details are shown in Figure 4.3(b).

The final representation is the output of max-pooling of f_a and f_d . We refer to this as *fluent* attention module, which is trained using a linear classifier in an end-to-end fashion using softmax Cross-Entropy loss function.

4.1.4 Experimental Setup

4.1.4.1 Dataset Splits

We use Epic-kitchens-55 (EK-55) [28] as our primary dataset. EK-55 has kitchen activities captured with an egocentric camera. In total, it has 352 sparsely annotated bounding boxes for objects and 125 actions. Among all actions, only 13 actions are object-transforming and 30 actions are context-transforming. We choose these actions such that each action has at least 40 videos in total. Our subset is also impacted by pre-processing step (object bounding boxes detection), since we require two bounding boxes of the object of interest either from the detector or the ground truth. This drastically reduces the number of videos with a long-tailed distribution over the classes. In total, we use 7342 videos, 2120 for object-transforming and 5222 for context-transforming actions.

Due to imbalanced number of object-transforming and context-transforming actions, we create five train/val/test splits of the dataset and report the mean performance across splits. Each split has all 13 object-transforming actions and randomly chosen 13 context-transforming actions. The test set in each split contains actions involving objects unseen during training. This

is different from the typical test set provided for EK-55 [28]. We argue that using those splits do not provide evaluation of the model learning actions or latching onto object features. Therefore, by having different objects in train and test set inspired by [119], we ensure that the model learns the action, and does not rely only on object visual features. Henceforth, we refer to test set as “unseen object-actions” and validation set as “seen object-actions” set. We provide dataset details in the Appendix 10.

For evaluation, we use overall accuracy, which is the ratio of correctly classified samples over the number of samples. Since, our dataset is long tail distribution, some classes have more than 100 samples whereas others have less than 20 samples, overall accuracy does not give a clear picture of how the models perform. Hence, we also report Mean accuracy, which is mean of accuracy for each action individually.

4.1.4.2 Implementation Details

We use ego-centric kitchen video dataset, Epic-Kitchen-55 (EK-55) [28] for training. This dataset provides temporal action localization labels and has object-centric actions. Each video can have multiple clips of same object-action pair. We consider first and last clip for the same object-action in proximity, so that accurate ‘start’ and ‘end’ states are captured for objects. After merging the first and last clips from the video, we use one merged clip for each object-action pair from a video.

For extracting object states, use a Faster-RCNN [141] detector, pre-trained on EK-55 [28], which has sparsely annotated ground truth bounding boxes for 295 object classes. We apply it on two sets of images from an object-action clip with 30fps: first and last 20 frames. From each

set, the object’s bounding box with the highest confidence is chosen, which should be higher than 65%. This gives use start and end object states. Features are extracted using ResNet50 [65] pre-trained on ImageNet [30] from the penultimate layer, and are of dimensions 1×2048 . Extracted features are then passed through our *fluent* Attention module, and classified for all actions. For our model, we use learning rate 1e-5, Adam optimizer and ReLU activation. Since our dataset follows long-tail distribution, we use weighted cross-entropy loss over the actions. We empirically choose a temperature value as 100. We use L2-normalization on output of both amplifiers, before fusion.

4.1.5 Results and Ablation

4.1.5.1 Quantitative Results

We compare our model with two types of action recognition models:

Spatio-temporal Baselines. Our setup is RGB based, we compare with the best RGB-only model for action recognition [189, 205]. TPN [189] shows state-of-the-art results on multiple datasets, including EK-55. TPN [189] captures visual temporal scales of different actions. TRN [205] accounts for relationship between frames at multiple scale (2,3,4 and 8). Both models use minimum of 3 frames, and finally consolidate the relationships between all frames. Hence, these models use much more information than two object states. We use 3 and 8 frames for these models, with TSN [172] based sampling frames. We also train these baselines for with 3 and 8 object states, to compare with our 2 objects states setup.

Object States Baselines. We compare with three other baselines which use exactly as input as

ours, *i.e.*, 2 object states. Fathi et al. [39]. detect changing regions in video and train SVMs for each action. We use our implementation for [39], with deep features for regions to replicate the results. We propose two more object states baselines: MaxpoolResFeats uses $\text{maxpool}(f_s, f_t)$ and ConcatResFeats uses $\text{concatenation}(f_s, f_t)$ for action classification. Note that we follow the notation explained in section 4.1.3.2, where f_s and f_t are ResNet features for start and end states. For a fair comparison, these object states baseline models have around the same number of trainable parameters as our model.

To keep it consistent with TPN [189] and TRN [205], we use ResNet50 [65] as our backbone network and all setups follow three-crop testing evaluation protocol. Early stopping on the validation set is used to avoid overfitting. The results are reported in table 4.2 for seen object actions (validation set) and table 4.1 for unseen object actions (test set). For 5 different splits for action classes, we denote the final accuracy as mean and confidence interval with 95% confidence, for object-transforming and context-transforming actions separately as well.

Discussion. We outperform all the baselines for 3 and 8 frames and bounding boxes setup, for seen and unseen object actions. These results support our hypothesis that object states are sufficient to capture the actions, be it object-transforming or context-transforming actions. Moreover, even concatenation of object state vectors along with a simple MLP performs better than most of the models with much larger number of frames and network parameters. Modeling object-transforming and context-transforming actions separately, improves accuracy for both type of actions, in comparison to concatenation only setup (ConcatResFeats baseline). Further, we need our amplifier modules for capturing the similarity and dissimilarity, to significantly improve object-

Table 4.1: **Quantitative results on Unseen Objects.** This table summarizes results for actions on objects not seen during training. (test set). All values are shown as mean with confidence interval (95%) over 5 splits. In the table we highlight the best and second best result in bold+underline and underline respectively.

Network Input	Method	All Actions		Object Transforming Actions		Context Transforming Actions	
		Acc.	Mean Acc.	Acc.	Mean Acc.	Acc.	Mean Acc.
8 Frames	TRN [205]	31.49 $\pm$ 2.2	7.16 $\pm$ 1.5	18.89 $\pm$ 6.0	7.24 $\pm$ 1.6	44.87 $\pm$ 7.1	7.07 $\pm$ 1.6
	TPN [189]	35.60 $\pm$ 4.1	8.78 $\pm$ 1.3	<u>28.61 $\pm$ 9.4</u>	8.59 $\pm$ 2.1	41.36 $\pm$ 8.7	7.42 $\pm$ 2.5
3 Frames	TRN [205]	29.78 $\pm$ 2.5	7.38 $\pm$ 2.8	16.26 $\pm$ 6.7	6.26 $\pm$ 2.1	44.16 $\pm$ 7.7	8.50 $\pm$ 4.0
	TPN [189]	30.59 $\pm$ 0.7	7.16 $\pm$ 1.5	18.83 $\pm$ 9.2	7.44 $\pm$ 2.4	41.14 $\pm$ 12.2	6.86 $\pm$ 1.5
8 Objects	TRN [205]	28.75 $\pm$ 2.5	7.53 $\pm$ 1.6	15.34 $\pm$ 5.9	6.35 $\pm$ 1.7	42.91 $\pm$ 7.0	8.71 $\pm$ 2.3
	TPN [189]	34.92 $\pm$ 2.6	9.22 $\pm$ 2.7	27.65 $\pm$ 6.8	8.51 $\pm$ 2.3	41.98 $\pm$ 5.6	7.93 $\pm$ 3.3
3 Objects	TRN [205]	27.86 $\pm$ 3.2	7.83 $\pm$ 2.5	16.70 $\pm$ 4.7	7.35 $\pm$ 1.6	39.68 $\pm$ 7.8	8.31 $\pm$ 3.5
	TPN [189]	28.84 $\pm$ 3.8	6.58 $\pm$ 2.5	12.30 $\pm$ 9.8	5.65 $\pm$ 4.0	<u>45.95 $\pm$ 14.9</u>	7.50 $\pm$ 1.6
2 Objects	Fathi et al. [39]	30.59 $\pm$ 1.1	8.17 $\pm$ 1.7	26.97 $\pm$ 2.6	<u>8.81 $\pm$ 1.1</u>	34.00 $\pm$ 3.5	7.53 $\pm$ 2.3
	MaxpoolResFeats	28.37 $\pm$ 4.1	9.70 $\pm$ 1.7	18.88 $\pm$ 2.1	8.56 $\pm$ 1.8	38.47 $\pm$ 9.3	<u>9.19 $\pm$ 2.9</u>
	ConcatResFeats	<u>35.71 $\pm$ 0.7</u>	<u>9.83 $\pm$ 1.3</u>	27.30 $\pm$ 4.5	7.40 $\pm$ 2.7	42.56 $\pm$ 6.6	8.37 $\pm$ 1.0
	Ours	<u>38.83 $\pm$ 1.4</u>	<u>10.08 $\pm$ 1.6</u>	<u>29.29 $\pm$ 8.0</u>	<u>8.96 $\pm$ 2.9</u>	<u>47.70 $\pm$ 7.7</u>	<u>10.79 $\pm$ 1.4</u>

Table 4.2: **Quantitative results on Seen Objects.** This table summarizes results for seen object actions (validation set). All values are mean with confidence interval (95%). In the table we highlight the best and second best result in bold+underline and underline respectively.

Network Input	Method	All Actions		Object Transforming Actions		Context Transforming Actions	
		Acc.	Mean Acc.	Acc.	Mean Acc.	Acc.	Mean Acc.
8 Frames	TRN [205]	48.20 $\pm$ 2.0	11.68 $\pm$ 1.6	47.20 $\pm$ 7.0	14.18 $\pm$ 1.9	48.20 $\pm$ 7.1	9.07 $\pm$ 2.1
	TPN [189]	56.13 $\pm$ 5.2	14.11 $\pm$ 1.8	13.01 $\pm$ 11.4	18.42 $\pm$ 3.2	49.67 $\pm$ 9.4	9.60 $\pm$ 3.2
3 Frames	TRN [205]	45.68 $\pm$ 5.4	12.27 $\pm$ 4.8	44.16 $\pm$ 8.3	13.90 $\pm$ 3.7	47.50 $\pm$ 8.3	10.62 $\pm$ 6.2
	TPN [189]	35.70 $\pm$ 2.0	8.63 $\pm$ 1.0	32.64 $\pm$ 7.0	11.05 $\pm$ 3.0	39.52 $\pm$ 10.1	6.09 $\pm$ 1.0
8 Objects	TRN [205]	57.50 $\pm$ 1.6	17.90 $\pm$ 2.9	56.16 $\pm$ 2.8	20.77 $\pm$ 2.5	<u>59.26 $\pm$ 3.0</u>	14.76 $\pm$ 3.7
	TPN [189]	59.92 $\pm$ 2.9	17.00 $\pm$ 4.8	65.28 $\pm$ 4.6	21.90 $\pm$ 4.6	50.84 $\pm$ 5.6	11.58 $\pm$ 5.4
3 Objects	TRN [205]	45.95 $\pm$ 3.8	13.62 $\pm$ 3.8	44.88 $\pm$ 8.9	15.36 $\pm$ 5.3	46.29 $\pm$ 6.4	11.86 $\pm$ 5.5
	TPN [189]	34.48 $\pm$ 6.3	8.57 $\pm$ 3.5	23.28 $\pm$ 15.1	8.97 $\pm$ 6.1	50.31 $\pm$ 8.0	8.38 $\pm$ 1.6
2 Objects	Fathi et al. [39]	55.93 $\pm$ 0.9	15.34 $\pm$ 3.1	61.12 $\pm$ 1.7	20.15 $\pm$ 4.8	46.30 $\pm$ 3.4	10.26 $\pm$ 1.6
	MaxpoolResFeats	56.80 $\pm$ 1.0	18.47 $\pm$ 8.2	56.48 $\pm$ 2.4	21.35 $\pm$ 9.9	56.57 $\pm$ 5.9	14.40 $\pm$ 7.0
	ConcatResFeats	<u>66.08 $\pm$ 0.9</u>	<u>18.80 $\pm$ 2.5</u>	<u>70.30 $\pm$ 1.5</u>	<u>22.96 $\pm$ 4.1</u>	57.87 $\pm$ 4.3	<u>15.26 $\pm$ 2.5</u>
	Ours	<u>68.36 $\pm$ 1.5</u>	<u>20.50 $\pm$ 2.8</u>	<u>71.84 $\pm$ 2.4</u>	<u>24.91 $\pm$ 2.1</u>	<u>61.36 $\pm$ 7.0</u>	<u>16.00 $\pm$ 3.9</u>

transforming actions, as compared to using max of ResNet features (MaxpoolResFeats baseline). Since our dataset is long tail distribution, we also show the mean-classwise accuracy, to show that our model not only performs well on majority classes but also performs better for minority classes.

4.1.5.2 Qualitative Results

This section discusses qualitatively how our method performs on out-of-domain data. Two kitchen based object-centric datasets are used in this section:

Epic Kitchens-100 [26] (EK-100) is an extension of EK-55. We use videos of EK-100 that are not present in any of our splits. Around 280 videos and 1032 individual action clips are extracted using similar pipeline of object state detection and ground-truth bounding boxes. Our detector only finds a limited set of seen objects for EK-100. We compute the accuracy for actions over top1 and top5 predictions. We achieve, 65.60% Top1 accuracy and 90.10% Top5 accuracy, which are comparable to seen object actions set for EK-55. Some examples of correctly classified actions are shown in Figure 4.4. Note that these samples are classified correctly by our model among top 5 predictions.

MIT-States. [76] Our focus is on studying and capturing object states, hence we also show results on MIT states [76], which is one of the first image dataset for object states. It has around 53K images, 245 object classes and 115 transformation states. We use only classes which overlap with our training classes, which are “cut” and “peel”. This just leaves around 1500 images from MIT states to test on. We pair different states like (whole→cut), (unpeel→peel) for each object to construct dataset which fits our setup. Our constructed MIT states test set contains 29k pairs

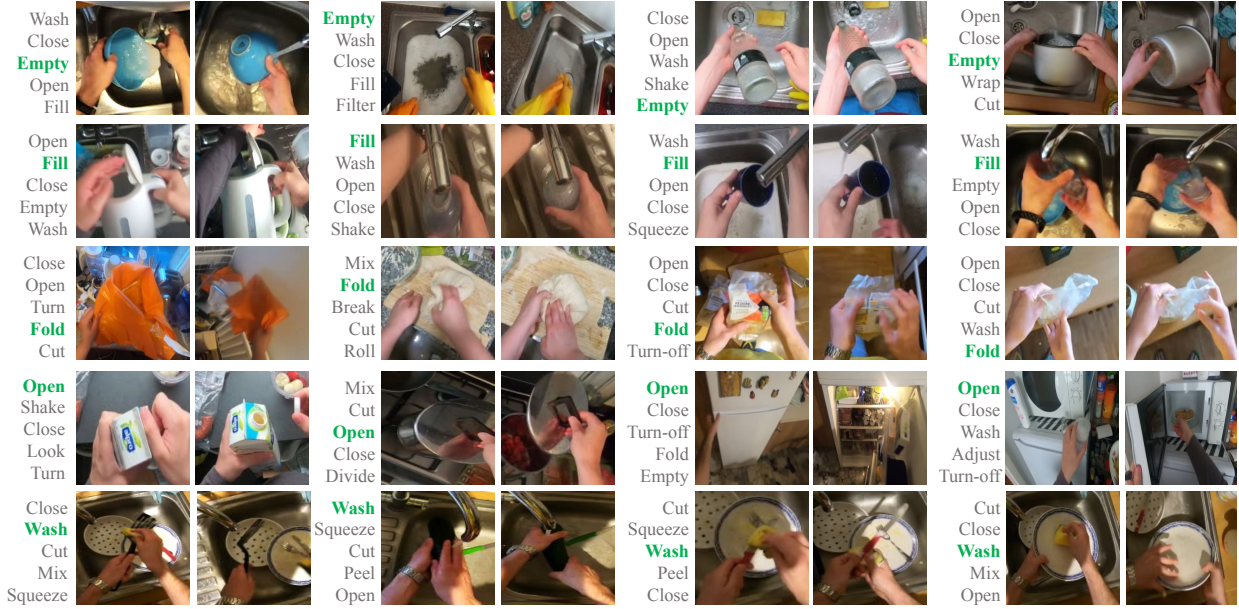


Figure 4.4: We show qualitative results for Epic-Kitchen 100 [26]. Objects are found using the same object detector as for our model. Among the shown actions, three are object-transforming: empty, fill, open, and two are context-transforming actions: fold, wash. Here are the row-wise results explained for each action (left to right): Row1: *empty bowl, empty sink, empty pot, empty bottle* Row2: *fill kettle, fill glass, fill bottle, fill cup*, Row3: *fold bag, fold dough, fold pack, fold bag*, Row4: *open milk, open lid, open fridge, open microwave*, Row5: *wash spatula, wash courgette, wash knife, wash plate*

of images, where 19305 pairs are cut and 10130 are peel. We observe that seen and unseen object transformations for MIT states are also recognized by our model. We show the some of most confident pairs for each transforming action retrieved from MIT States in Figure 4.5. Since some objects are common with our training set, we show pairs for seen as well as unseen object transformations in the figure.

YouCook2. [207] is another cooking dataset with around 2000 videos of 89 recipes with annotated procedure steps. The validation set for YouCook has sparse bounding box annotations. For transforming actions, we require atleast two distinct states of actions, hence we use the “oracle” detector. For context-transforming actions, we use any two annotated bounding boxes from the video. Figure 4.6 highlights some of our results on this dataset. Similar to EK-100, we show correctly classified actions using accuracy at top 5 predictions.

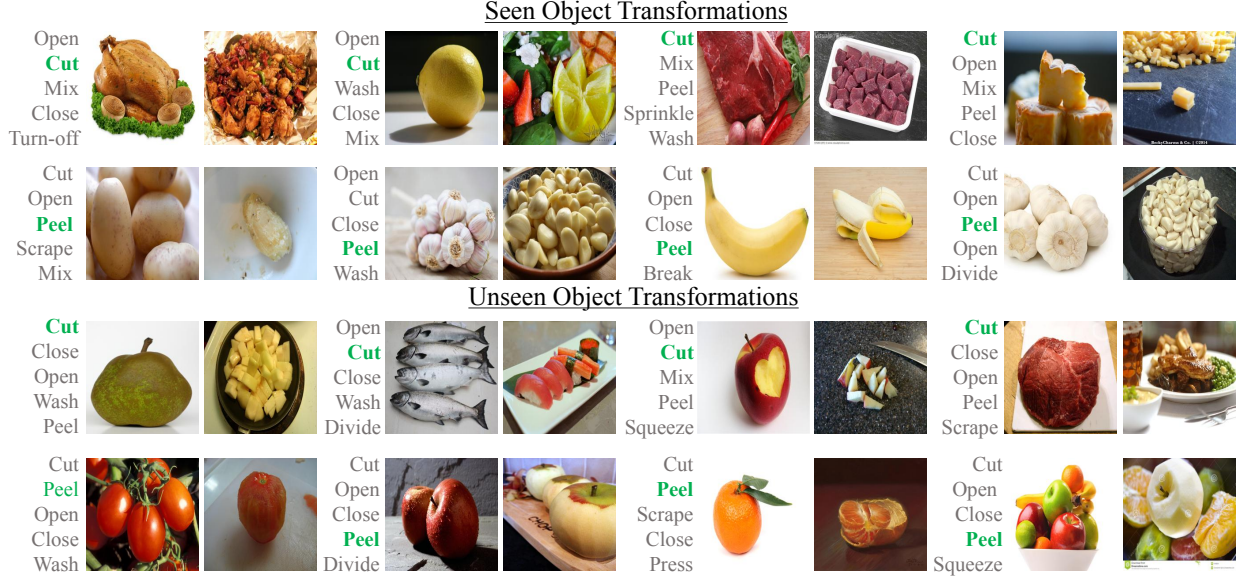


Figure 4.5: We show qualitative results for MIT states [76]. There are only two common classes cut and peel. Further, some of the objects are common with our training transformation states. Hence we show seen and unseen object transformation sets separately. We show top 5 predictions for each image pair. For seen transformations, we show results for Row1: *cut chicken, cut lemon, cut meat, cut cheese* Row2: *peel potato, peel garlic, peel banana, peel garlic*, Row3: *cut pear, cut salmon, cut apple, cut beef*, Row4: *peel tomato, peel apple, peel orange, peel fruits*.

4.1.5.3 Ablation Studies

In this section, we analyze and understand various components of our approach. The ablation studies are shown for one split on validation set (Seen object actions), hence the values are different for our model from Table 4.2. All 5 splits followed similar observations.

Object Baselines: Using only ResNet features for action recognition, either maxpool (Maxpool-ResFeats in Table 4.1) or concatenation (ConcatResFeats) of object states, also performs better than 3 and 8 frames models.

(start→end) Attention: This is akin to using a single amplifier between start and end states. We use a non-spatial attention block, inspired by [61, 175]. Given an input feature (start/end state, denoted as g_s/g_t), three linear projections are applied to compute query (q) with respect for before state g_s , and key (k) and value (v) with respect to after state g_t . The attention (a) is



Figure 4.6: We show qualitative results for YouCook2 [207]. We use oracle detector to find bounding boxes for this dataset, besides using sparse ground truth bounding box annotations. We show that our model generalizes on unseen object actions. Among the shown actions, close, cut and peel are object-transforming actions and add and mix are context-transforming actions. Here are the row-wise results explained for each action (left to right): Row1: *add salt, add onions (unseen), add salt, ad egg yolk* Row2: *close pot, close cooker (unseen), close grill (unseen), close grinder*, Row3: *cut sandwich, cut spring onion, cut tofu, cut cabbage*, Row4: *mix milk, mix meat, mix pepper(unseen), mix flour*, Row5: *peel avocado, peel raddish, peel squid (unseen), peel carrot*

weighted sum of value representation, where the weight (M) is dot product between the query and the corresponding key, followed by a softmax normalization as

$$M = \text{softmax} \left(\frac{qk^T}{\lambda} \right)$$

$$a = Mv$$

$$o = g_s + \text{BatchNorm}(a)$$

where o represents attention output for start state with respect to end state, and λ is the temperature parameter. Note that $g_s, q, k, v, o, a \in \mathbb{R}^n$; $M \in \mathbb{R}^{n \times n}$, where $n = 2048$. This experiment shows we need different ways to amplify affinity and differences between start and end states.

Amplifiers Alternative: For understanding the significance of our proposed *fluent* Attention

Table 4.3: **Ablation analysis on Seen Objects.** We show how our amplifiers and separate modeling of object-transforming and context-transforming actions help in object-centric action recognition. All the experiments are done on split1. In the table we highlight the best and second best result in bold+underline and underline respectively.

	All Actions		Object Transforming		Context Transforming	
	Acc.	Mean Acc.	Acc.	Mean Acc.	Acc.	Mean Acc.
ConcatResFeats	<u>65.9</u>	23.23	<u>72.4</u>	27.56	<u>49.49</u>	<u>18.89</u>
MaxPoolResFeats	51.86	22.75	56.80	<u>30.78</u>	42.39	18.12
(start→end) Attention [175]	58.17	20.95	65.60	23.69	39.39	16.22
Amplifier Alternate	60.74	<u>24.06</u>	64.01	29.62	48.52	18.12
Our Approach	<u>68.77</u>	<u>25.39</u>	<u>76.40</u>	<u>33.21</u>	<u>49.49</u>	<u>18.26</u>

Table 4.4: This table shows analysis of different temperature λ variable for the experiments. Although, changing λ does not significantly impact the overall accuracy, we optimize for mean classwise accuracy, which is maximum for $\lambda = 100$.

λ	All Actions		Object Transforming		Context Transforming	
	Acc.	Mean Acc.	Acc.	Mean Acc.	Acc.	Mean Acc.
0.1	42.45	12.11	46.97	13.08	31.51	9.84
10	43.96	11.06	48.42	12.45	33.87	9.63
50	43.62	11.01	47.94	12.37	33.89	9.65
100	43.62	13.10	48.42	14.42	32.78	10.71
500	41.77	12.87	45.52	13.32	33.33	9.26

Module, we model transforming actions as difference between features for object states and mean of features for context transforming actions. Fusion of the two is done in the same way with maxpool. This experiment shows need for affinity and differential amplifiers.

Temperature Variable. Our model uses ResNet50 [65] features as input. We do not fine-tune the backbone network. Input features are passed through a linear layer, with ReLU activation, and then through the fluent attention module. Our model is robust to temperature variable, as shown in Table 10.4. Note that the results are shown for split1, similar observations follow for the rest of the splits. Since our dataset is long-tail, we prefer optimizing for mean classwise accuracy instead of overall accuracy, hence we use temperature $\lambda = 100$.

4.2 Compositional Action Recognition

4.2.1 Introduction

In the second task, we extend an existing task of Compositional Action Recognition [107]. While the focus of prior work [107] is on long-term activity tracking in videos, we aim to recognize subtle changes in object states which is a crucial first step for activity recognition. By detecting the initial state and final object state compositions, our task allows the model to learn unseen object state changes robustly. We benchmark multiple recent baselines for video tasks on the ChopNLearn dataset.

Human actions often change object states and different objects can have diverse visual transitions even when subjected to the same action type. To investigate this problem in a more intuitive manner, [107] introduced a new task of compositional action recognition, which targets at improving the robustness of models to handle the same actions with different objects involved. For example, given an action of ‘taking something out from something’, the model is trained on a limited set of objects and is tested on unseen types of objects to assess its generalizability. Hence, despite the same underlying action, the object and visual features can be quite diverse. Similarly, the composition of the same action with different object types can look very distinctive. For instance, although cutting an `carrot` and a `apple` require similar knife movements, the resulting visual changes are distinct, with the former changing from a `whole apple` to a `peeled apple`, and the latter changing from a `whole carrot` to a `peeled carrot`. Therefore, we propose to use our dataset for the task of compositional action recognition, which can also be referred to as Compositional Zero-Shot Action Recognition, as the compositions of objects and



Figure 4.7: Compositional Action Recognition: training videos are used to recognize **objects** along with transition from **state1** $\rightarrow$ **state2**, to generalize on recognizing unseen object-state transitions.

states are unseen during training.

4.2.2 Related Works

States for Action Recognition. Detecting object states and corresponding actions from videos is explored in supervised [9, 18, 39, 146] and self-supervised manners [35, 161, 162]. While some works focus on recognizing actions using states [9, 18, 39, 146], others discover states as the future frames in the videos in [35, 77]. Some works [161, 162] also detect the exact frames of state 1, state 2 and the action that causes transition from state 1 $\rightarrow$ 2. Another recent work (Ego4D [53]) also proposes new tasks like point-of-return state-change prediction for object state transition detection. Hence, object states so far have been used as a signal for detecting and localizing actions. We focus on extending this understanding of states to generalize across different objects with limited seen object-state transition videos.

Compositional Action Recognition. In contrast to randomly assigning samples for training and testing, [107] presented a new task of Compositional Action Recognition. The premise of this task is: actions are split based on objects they apply on. During training, only a set of objects are

seen corresponding to set of objects, while during testing, unseen object appear for seen action labels. Following studies [83, 104, 137, 174, 188] used relationship between objects and states bounding boxes to model the compositional aspect, where the evaluation is performed on how well the composition of unseen object and state is recognized. We propose a similar task, where videos are trained on seen compositions and tested on unseen compositions.

4.2.3 Task Description

For this task, we consider each clip of a video as containing a single object with a single state transition. From the raw videos, which typically contain 2-3 transitions of object states per video, we segment the clips into isolated ones with only one transition. Examples of transitions include changing from a `whole` object to a `peeled` object or from a `peeled` object to a `baton cut` object. Similar to [107], we divide all object-final state compositions into two sets: seen compositions, which are used for training, and unseen compositions, which are used for testing. We ensure that each object and state are seen at least once individually in the training set, but not together as a composition. The objective of the task is to predict the correct labels for the initial object-state composition (o_i, s_j) and the final composition (o_i, s_k) , given a clip containing an object o_i transitioning from an initial state s_j to a final state s_k . Note that the clip is considered correctly classified only if both the object and state labels are correct for both the initial and final compositions.

Table 4.5: **Compositional action recognition results.** “Start/End” denote the prediction results for the initial and the final state composition with the corrected object type. **Bold** and underline represent the top-1 and top-2 results.

Model	Features	Split 1				Split 2				Split 3			
		Start		End		Start		End		Start		End	
		acc@1	acc@3	acc@1	acc@3	acc@1	acc@3	acc@1	acc@3	acc@1	acc@3	acc@1	acc@3
AvgPool	I3D [17]	9.5	23.7	4.7	14.2	8.3	21.9	5.2	19.8	15.9	28.5	4.8	22.3
LSTM [71]	I3D [17]	14.2	36.2	5.7	29.8	12.5	29.2	6.2	26.0	17.5	34.9	6.3	23.7
Transformer [169]	I3D [17]	23.7	49.0	10.9	44.3	27.5	46.2	14.6	44.2	20.6	42.9	11.1	44.4
AvgPool	MIL-NCE [111]	11.1	31.6	4.8	28.4	9.4	17.7	5.2	13.5	14.2	41.4	12.8	41.4
LSTM [71]	MIL-NCE [111]	15.9	36.5	6.4	36.6	11.9	36.7	9.8	36.7	18.9	39.6	8.0	25.4
Transformer [169]	MIL-NCE [111]	<u>50.9</u>	85.7	47.7	76.2	56.2	82.3	<u>52.7</u>	88.5	41.1	74.6	42.9	77.7
STLT [137]	–	2.8	15.5	1.4	8.4	1.4	13	1.4	11.6	4.2	14.1	1.4	11.3
Transformer [169]	R3D [60]	45.1	<u>85.9</u>	<u>52.1</u>	85.9	<u>55.1</u>	<u>94.2</u>	58.0	<u>92.8</u>	<u>59.1</u>	<u>85.9</u>	<u>56.3</u>	<u>85.9</u>
CAF [137]	R3D [60]	53.5	88.7	57.8	88.7	<u>55.1</u>	95.7	58.0	95.7	62.0	93.0	63.4	93.0

4.2.4 Experimental Setup

Dataset Splits. We create 3 different dataset splits as follows. All splits have disjoint train, test and validation samples, and are created with different constraint combinations:

- **Split 1:** This split is a random selection of object-final state compositions with cross-view condition. We do not use any information from related groups.
- **Split 2:** In this split, we use related group information for states, along with cross-view. based on related groups, if `baton carrots` is seen in training set, then `julienne carrots` can be part of test set. Since `baton` and `julienne` are part of the same related group, we can learn an object in one style and can generalize to another style from the same group in Section 5.3.1.
- **Split 3:** This split includes information from both related groups for states and objects. We want to ensure that even if an object is not seen in its related group, a similar object is seen in the related group. For example, if `broccoli` is seen with `large cuts`, then `cauliflower` with `large` or `small cuts` can be in the test set.

We discuss the other splits we evaluated in Appendix 10.

Evaluation. We evaluate the accuracy of predicting both the initial and final compositions of objects and states in the test set. Only when the object and state are both correct, it is counted as a correct prediction. Specifically, we use two separate prediction heads for objects and states. We emphasize the need to evaluate composition as a whole, rather than just predicting the state, as the way an `apple` is cut can differ significantly from the way a `bellpepper` is cut. Therefore, accurately recognizing both the object and state is crucial for tasks related to understanding and generating videos of object states. We also recognize the importance of top@3 accuracy, since object states can sometimes be visually similar, leading to confusion in detecting the correct composition. For example, `julienne apple` can be visually very similar to `julienne potato`.

4.2.5 Results and Ablation

To evaluate our proposed method, we establish baselines using both traditional architectures and features for video action classification, as well as comparing with recent works in compositional action recognition. As shown in Table 4.5, in the first section, we use pre-extracted I3D[17] features and conduct experiments by comparing simple average pooling, LSTM, and multi-layer Transformer [169] model. It shows that the Transformer model performs the best among these variants due to the great capacity of temporal modeling ability. In the second section, we also experiment with more recent pre-trained features MIL-NCE [111] along with transformer models, which outperforms I3D features. MIL-NCE [111] features are pre-trained on HowTo100M [112] with multimodal (RGB+narrations) setup, which is more robust for video

Table 4.6: Other splits: We also present other possible splits of data. All the results are using I3D [17] pre-trained features along with one layer Transformer [169] model. Comp. represents the initial: object-initial state composition and final: object-final state composition results for each split.

	Split 4		Split 5		Split 6		Split 7		Split 8		Split 9	
Comp.	acc@1	acc@3	acc@1	acc@3	acc@1	acc@3	acc@1	acc@3	acc@1	acc@3	acc@1	acc@3
Initial	46.5	78.9	65.2	90.6	37.5	70.8	41.1	76.0	47.2	71.5	41.2	78.3
Final	45.3	75.9	73.9	92.6	37.7	69.2	41.4	72.2	48.9	73.0	42.9	77.6

Table 4.7: Results of finetuning R3D [60] backbone. “Start/End” denotes the prediction results for the initial and the final state composition with the correct object type.

		Split 1				Split 2				Split 3			
		Start		End		Start		End		Start		End	
Model	Finetune	acc@1	acc@3	acc@1	acc@3	acc@1	acc@3	acc@1	acc@3	acc@1	acc@3	acc@1	acc@3
CAF [137]		53.5	88.7	57.8	88.7	55.1	95.7	58.0	95.7	62.0	93.0	63.4	93.0
CAF [137]	✓	80.3	98.6	87.3	98.6	84.1	98.5	89.9	98.5	88.7	98.6	88.7	98.6

downstream tasks.

In the final section of Table 4.5, we employ the state-of-the-art compositional video recognition model proposed in [137] and use pseudo labels of bounding boxes for each hand and object, as there are no ground-truth hand and object trajectories available. Specifically, the Spatial-Temporal Layout Transformer (STLT) [137] takes in the spatio-temporal locations and class labels for each bounding box as input, uses positional embeddings to project coordinates and class labels into features, and adds transformer layers to model spatial-temporal relationships. However, without any appearance information, STLT achieves low performance on all metrics. On the other hand, with the appearance features, which are extracted by inflated 3D ResNet50 [80] (R3D), it can achieve much higher performances than STLT. Finally, Cross-Attention Fusion (CAF) applies cross-attention [165] to fuse the layout (STLT) and appearance (R3D) branch embeddings, achieving the best results. It demonstrates that combining the layout and appearance information together can help predict object and state types more accurately.

4.2.5.1 First and last segment classification

For compositional action recognition, we emphasize that the model must learn to predict the object-initial state composition and the object-final state composition. Moreover, some works [39, 146] use a similar setup for object state classification and use only the first and last frame/segment for this. Ideally, the first few frames and last few frames should be sufficient for understanding the changes in object states. We also experiment with the first and last segments of videos, for classification. We find that using the additional middle frames improves the classification accuracy for the final composition.

4.2.5.2 Finetuning Backbone

We also show results [148] without finetuning any pre-trained features (I3D [17], MIL-NCE [111], R3D [60]). For the sake of completeness, we also show results with finetuning the backbone for R3D features in Table 4.7. Although the top@1 accuracy is much better, it is still not 100%. Moreover, the dataset is much smaller and overfits very quickly for backbones which are trained on 10x more data. Hence, for sake of benchmarking, we propose not fine-tuning the features for consistency.

4.3 Conclusion

We presented a framework for recognizing object-centric actions using object states. We show that by leveraging object states object-transforming and context-transforming actions can be learnt. We propose a novel *fluent* attention module, with affinity and differential amplifiers, that capture the subtle similarities and differences between object states. We also show gen-

eralization to unseen objects and out-of-domain datasets. Our findings also include successful generalization to unseen objects and out-of-domain datasets, as well as showcasing the capabilities of our approach in compositional action recognition tasks, which adapt to unseen state-object action pairs.

Part II: Object-Attribute Composition for Generation

Chapter 5: Chop & Learn Benchmark Suite

5.1 Introduction

Objects often exist in different shapes, colors, and textures in the real-world. These visually discernible properties of objects, also known as states or attributes, can be inherent to an object (*e.g.*, color) or be a result of an action (*e.g.*, chopped). Generalization to unseen properties of objects remains an Achilles heel of current data-driven recognition models (*e.g.*, deep networks) that assume robust training data available for exhaustive object properties. However, humans (and even animals) [6, 25] can innately imagine and recognize a large number of objects with varying properties, by composing a few known objects and their states. This ability to synthesize and recognize new combinations from finite concepts, called *compositional generalization* is often absent in modern deep learning models [88].

Several recent works have been proposed to study composition in terms of the disentanglement of objects and the states in images [76, 106, 147, 195] as well as videos [9, 18, 39, 61, 146, 161, 162]. A few works have attempted to improve open-world text-to-image generation models [46, 144] for the task of compositional generation. However, current suite of datasets lacks either granular annotations for object states or enough data to study how object states evolve under different conditions. Therefore, measuring the compositional generalizability of these models on different tasks remains an open challenge.

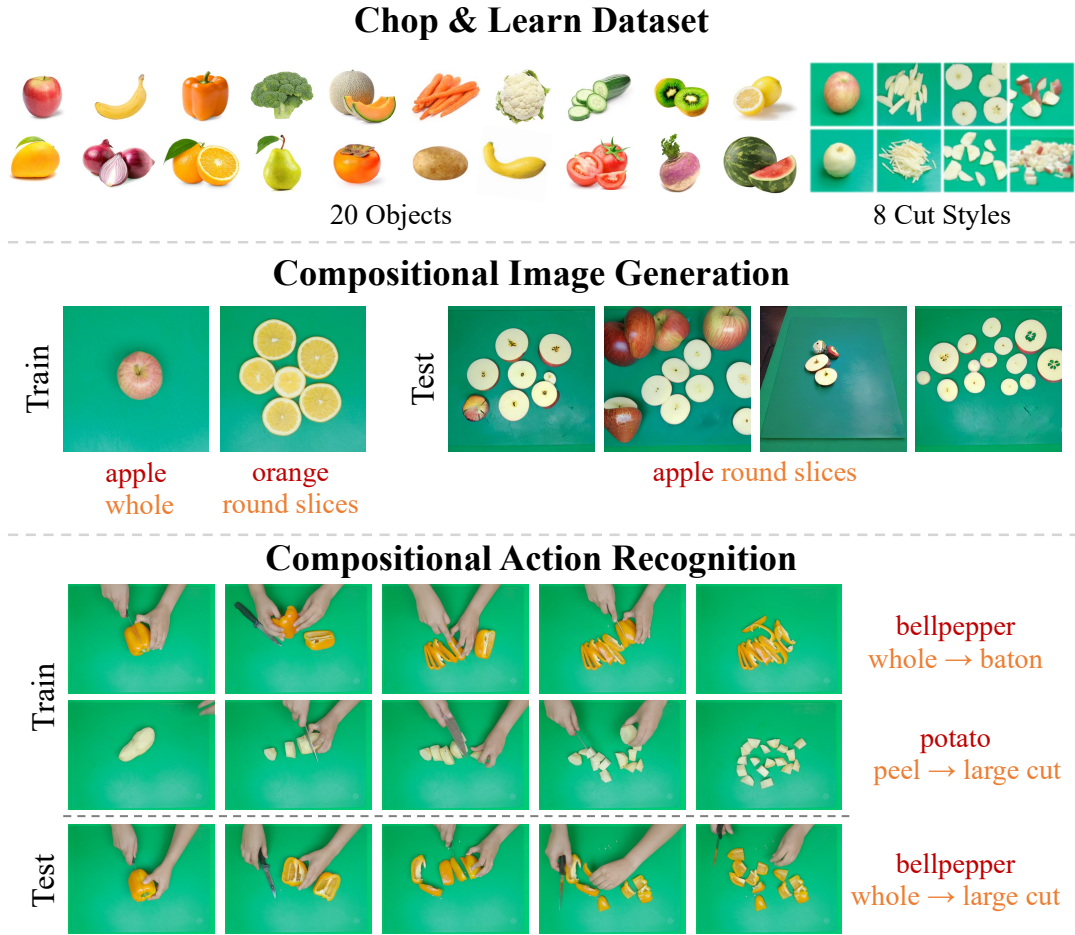


Figure 5.1: We present **Chop & Learn (ChopNLearn)**, a new dataset and benchmark suite for the tasks of Compositional Image Generation and Compositional Action Recognition. It consists of 1260 video clips and 112 object state combinations captured from multiple viewpoints for 20 objects and 8 cut styles. We also propose two new compositional tasks and benchmarks - (1) Image Generation: given training images of various **objects** in various **states**, the goal is to generate images of unseen combinations of **objects** and **states**. (2) Action Recognition: training videos are used to recognize **objects** along with transition from **state1** → **state2**, to generalize on recognizing unseen object-state transitions.

In this chapter, we propose a new dataset, **Chop & Learn (ChopNLearn)** collected to support studying compositional generalization, the ability to recognize and generate unseen compositions of objects in different states. To focus on the compositional aspect, we limit our study to a common task in our daily lives – cutting fruits and vegetables. When using different styles of cutting, these objects undergo different transformations and the resulting states are easily recognizable by humans. Our goal is to study how these different styles can be applied to a variety of objects for recognizing unseen object states. More specifically, we select *twenty* objects and *seven* commonly used styles of cuts (plus whole object) which results in object-state pairs with different granularity and sizes (Figure 6.1). We collect videos of these objects being from *four* different viewpoints, and label different object states in each video. Each style of cut changes the visual appearance of different objects in different ways. To study and understand object appearance changes, we propose two new benchmark tasks of Compositional Image Generation in Chapter 6 and Compositional Action Recognition in Chapter 4 with a focus on unseen compositions.

5.2 Related Works

The existing image datasets such as MIT-states [76], UT-Zappos [195], COCO-attributes [127], CGQA [118] and VAW [131], are not suitable for image generation tasks for two reasons: 1) there are very few transferable objects and attributes, 2) the images are web-scraped and very diverse with varied background. Due to this, generative models latch on background details rather than understanding subtle changes in objects. In video domain, there have been various video datasets with procedural and kitchen activities that capture object and state transformations, such as Epic-Kitchens [27] with object and hand bounding box annotation version VISOR [29],

Table 5.1: **Comparison with other video datasets.** This table highlights the distribution of the objects, states and compositions in different datasets. Obj. refers to objects, Comp. is compositions of objects and styles, N refers to the number of compositions that have more than 10 samples, and Styles* refers to grouping of styles: instead of generic names like cut, chop, etc., we use 3 distinct styles (chop/dice, peel, grate) as styles. MIT-States[†] is the only image-based dataset, the rest are video-based datasets. All these data numbers are for edible objects and cutting style actions from respective datasets. Our dataset has uniform distribution for each metric in the table, which makes it suitable for learning objects and their states.

Datasets	Total # of				Avg. # of Samples			N	# of Views
	Samples	Obj.	Comp.	Styles*	/Obj.	/Comp.	/Style		
MIT-States [†] [76]	1676	27	52	4	62.07	32.23	419	48	1
Youcook2 [207]	714	160	313	3	7.3	2.2	166.7	26	1
VISOR [29]	301	58	122	3	5.2	2.5	42.9	3	1
COIN [167]	390	6	7	2	65	55	195	6	1
Ego4D [53]	216	12	12	3	18.2	18	54.5	8	1
50Salads [163]	904	5	6	2	182	152	457	6	1
ChangeIt [161]	264	8	14	4	46.3	26.4	96	14	1
CrossTask [209]	1150	7	8	2	164.3	143.7	575	8	1
Breakfast [86]	1055	3	4	2	351.7	263.8	527.5	4	1
ChopNLearn	1260	20	112	8	74.2	11.8	185.5	112	4

Youcook2 [207], Ego4D [53], COIN [167], HowTo100M [112], Breakfast [86], 50Salads [163], CrossTask [209] and ChangeIt [161]. There are a few common problems across these datasets: (1) Most of these datasets lack annotations for the granularity of cutting styles. The styles labeled are cut, chop, slice, dice, peel, grate, julienne, which only comprises of three broader styles of transformations, *i.e.* chop/dice, peel and grate. (2) The compositions of different objects and states are highly skewed and similar to image datasets. Some datasets have a long-tail distribution of objects, which can make it challenging for models to learn per-object-based states when there is only one sample available in the dataset. And lastly (3), the frames are noisy with lots of objects and attributes that object states changes are harder to capture (as shown in left side of Figure 5.2). For most datasets, the ground truth is also not

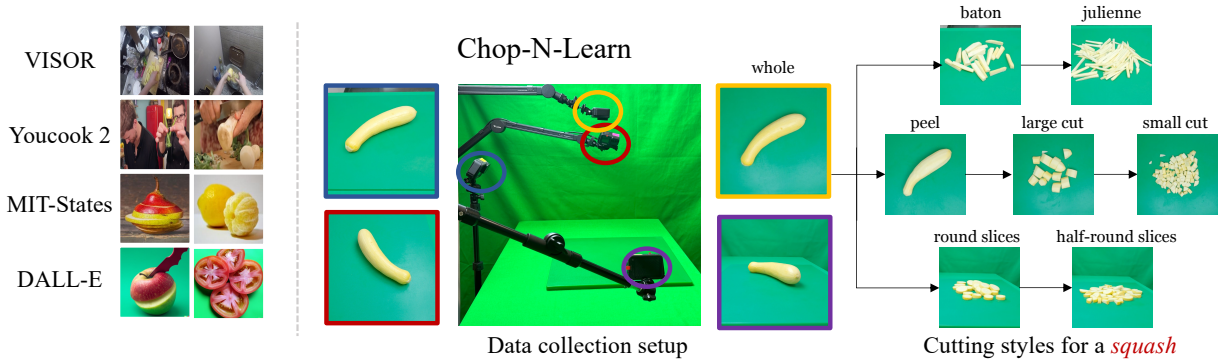


Figure 5.2: Left: We show examples of cutting styles from popular video datasets (VISOR [29]: chop and peel potato, Youcook2 [207]: chop broccoli, peel radish), image dataset (MIT-states [76]: slice pear, peel orange) and generation pipelines (DALL-E [139]: baton cut apple, half round slices tomato). Most of these are either too noisy to capture subtle differences in objects or do not have the granularity of specific cutting styles. Center: Our 4 camera setup captures videos of one object in 4 different views. Right: We capture 8 styles of object states, which can be derived in a hierarchical manner from larger to small cuts. Each style is of different shape and granularity.

annotated for object detection, which makes it even harder to look for object of interest. Using an object detector to remove the background is an option, however with deformable objects, most Faster-RCNN [141] based object detectors fail to capture the object itself, and latch onto smaller pieces instead. In Table 5.1, we show statistics of data available in different datasets. The # of clips from other datasets that has granular annotations of object-state pairs and can be used for compositional tasks. For instance, COIN [167] has 180 categories with 10000 videos, but clips that have cutting styles as labels were only 390. Further, these clips only cover cut/peel actions, and cannot be categorized further based on granularity and shape of pieces. Our proposed dataset ChopNLearn is designed to capture various objects and their cut styles, with uniformly distributed samples for 20 objects and 8 styles (including whole, 7 other cut styles Figure 5.2).

5.3 Chop & Learn

Our main objective with Chop & Learn (ChopNLearn) is to understand and learn granular object states, specifically styles of cuts which can be applied to diverse variety of objects. With

this in focus, we collect object state transition videos, as well as images of object in various states, with 4 different camera views (Figure 5.2). We discuss the design choices and motivation below.

5.3.1 Design Choices

5.3.1.1 Selection of States (styles of cuts)

Fruits and vegetables are commonly cut in specific styles based on the need of the recipes. For instance, for eating an apple, we slice it in relatively large pieces while for using it in a pie, we might cut smaller or round slices of it. We select 8 common styles of cuts, *i.e.*, `large cut`, `small cut`, `baton`, `julienne`, `round slices`, `half round slices`, `peel`, and `whole` for our study. These are the most common styles of cuts for vegetables and fruits, which do not require any additional training to learn apart from common kitchen operation and knife handling skills. These styles of cuts can also have similarities with respect to shapes, yet are different in granularity. For example, `baton` (french-fries style cut) and `julienne` are similar in shape (long pieces), but `julienne` is more finely cut than `baton`. Similarly, `large cut` is a coarser version of `small cut`, and `half round slice` is one step from `round slices` (as shown in Figure 5.2). We also have annotated the states `whole` and `peel`, which are the base states of objects.

5.3.1.2 Selection of Objects

We want to learn to transfer styles of cuts to different objects. To ensure consistency in transfer, we also consider the base state, *i.e.*, `whole` state of objects. For instance, it is hard to

visualize large cut of carrots, if the seen data only includes rounder objects like oranges. Hence, we consider some fruits and vegetables with similar colors, textures and shapes to include consistency across visual similarities after chopping. In this study, we used seasonal fruits and vegetables categorised on the basis on their shapes, colors and textures: round small objects: [apple, pear, mango, potato, turnip, onion, kiwi], citrus fruits [lemon, orange], flower-like textured objects: [cauliflower, broccoli] , larger round objects: [cantaloupe, watermelon], textured from inside objects: [bellpepper, tomato, persimmon], and long objects: [cucumber, carrot, squash, banana]. This consists of 10 fruits and 10 vegetable items, with at least one pair of similar objects presents in the dataset.

5.3.1.3 Related Groups

One of the key aspects of this dataset is transferability of cut styles to a variety of objects. We set up some constraints and create related groups for objects and styles. These related group enable us with structural and visual style transfer abilities. If an object is seen from related group A with a particular style, we should be able to transfer that style to another object from the same related group A and vice-versa. In other words, we group sets of objects and cut styles which are visually similar (based on color, shape and texture) together to create related groups for objects and states separately. For states, we combine [baton, julienne], [round slices, half-round slices], and [large cut, small cut] together as related groups. For objects, we define seven groups with related objects: [apple, pear, mango], [lemon, orange], [cauliflower, broccoli], [cantaloupe, watermelon, kiwi], [bellpepper, tomato, persimmon], [potato, turnip, onion], and [cucumber, carrot,

squash, banana].

5.3.2 Data Collection Setup

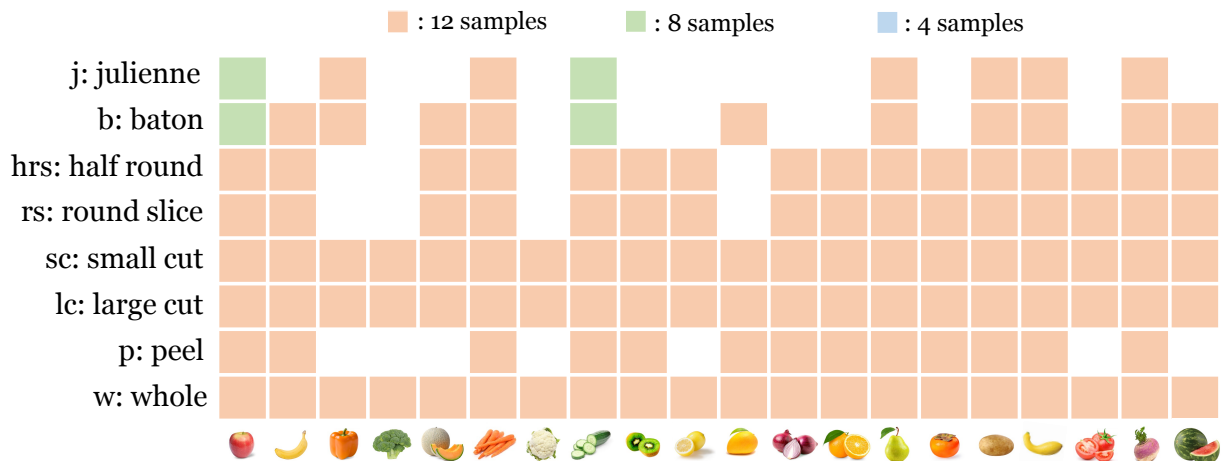
We collect data using four GoPro cameras [3] positioned at different angles, with three participants (Figure 5.2). We use a green screen and green chopping board for minimum distraction in the background, such that the objects and their cut pieces are easily segmented for each view.

5.3.2.1 Granularity of styles

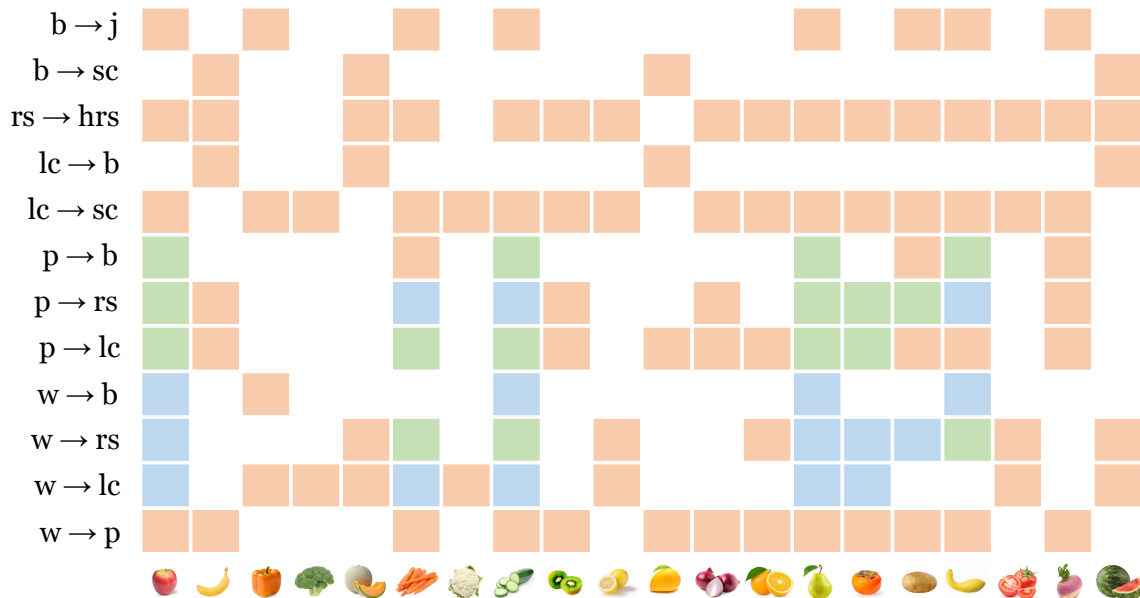
For ease and consistency across participants, the size of cut pieces can be defined as the shape and ratio of one piece with respect to the whole object. For more details, please refer to the appendix. Given a set of n states and m objects, we can have at most $m \times n$ compositions. However, our dataset does not include some compositions which are not commonly found in real world. For instance, due to the texture of `onions`, it is not feasible to cut `onions` in `baton` or `julienne` style, since the layers of the `onion` do not stay intact, so we do not have a sample of `[baton, onion]`.

5.3.2.2 Video Recording

We primarily collect video data, and derive state change frames from long videos. Each video consists of 2-3 object states, which are annotated while data collection process using the highlight features of GoPros. For synchronizing across different cameras, we initially start with a clapper to make a clap sound for indicating the beginning of the video. Then, we highlight the frames in one of the GoPro as the first/initial state. The participant then walks up the object and starts cutting the object. After the object is cut in one style, the participant steps back and we



(a) Image Statistics



(b) Video Clips Statistics

Figure 5.3: **Statistics for ChopNLearn:** We show the number of samples for each object-style composition in a color-coded manner: orange represents 12 samples, green represents 8 samples and blue represents 4 samples.

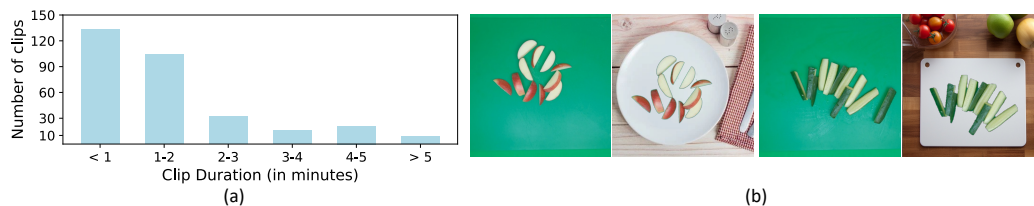


Figure 5.4: **(a)** The clip length distribution for one camera (315 unique clips). **(b)** Preliminary results of using green screen to augment the dataset with different backgrounds. We continue to improve the transfer results by adding shadows and background matting.

highlight another frame as the next state. The participant performs at least 2 styles of cut in each video, which can be done consecutively. For instance, we can first cut an object with `large cuts`, and then do `small cuts` subsequently. The video ends with another clap for the end of video detection and synchronization across different cameras. Henceforth, we collect video data along with annotated states for each participant, without extra effort of annotations. More details and statistics of dataset are shown in Figure 5.3. Average video clip length (one state change for an object) is 1m40s. The distribution is shown in Fig. 5.4(a).

5.4 Scope and Limitations

The objective of ChopNLearn dataset is compositional generation and recognition, using a granular and structural understanding of transferable object states. Terms such as ‘slice’, ‘dice’ alone often lead to a loss of granular information. For *e.g.*, a sliced apple can be horizontally or vertically sliced, or cut in the half, and then sliced as semi-circles. Hence, we use more specific categories than other traditional state change datasets as shown in Tab. 1. Moreover, the subtle state change understanding is a challenging task on its own merit [39, 161, 162]. Recognizing/segmenting actions in a video is a complementary task and an interesting future direction, but is currently beyond the scope of this work.

Moreover, we acknowledge that making the classes more granular can be confusing for the model, which appears similar. To confirm this, in Fig. 5.5, we show the confusion matrix for generated images (classified by the StateClassifier (Chapter 6), and action-based method using the final states (Chapter 4). We see `baton` and `julienne`, `half round slice` and `round slice`, are two difficult pairs for compositional generation. In contrast, the action-based method can classify

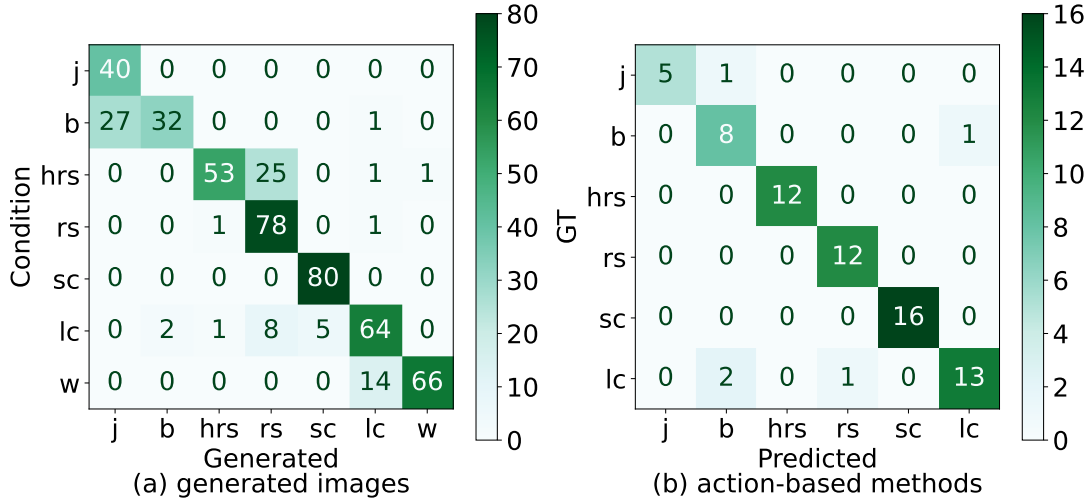


Figure 5.5: Confusion matrix for generation & action-based tasks.

most states correctly. We hypothesize that since action-based methods use trajectory, and multiple frames for classification, the confusion between similar object-state pairs is significantly reduced.

Further, we chose green screen to focus on the object states, and such that the object pieces can be segmented easily. As some preliminary work, in Figure 5.4, shows some results on how thresholding using simple opencv library functions works for segmenting the object pieces after styles of cuts are applied. Further, we use Midjourney [4], which is a Stable Diffusion [69] based text to image generator tool, to generate a set of images with chopping boards. We show this as proof of concept for proving and encouraging community to use this dataset without introducing bias for green screen images.

Dataset Webpage

For more details, results and analysis, please visit our website at: <https://chopnlearn.github.io>.

Chapter 6: Compositional Image Generation for unseen Attribute-Object pairs

6.1 Introduction

As defined in the previous chapter 5, we discuss a new task for Compositional Image generation in this chapter. The objective of this task is to generate an image based on an (object, state) composition that was not seen during training. As shown in Figure 6.1, during training, a generative model is provided with images of an (apple, whole) as well as an (orange, round slices). At the test time, the model has to synthesize a new unseen composition (apple, round slices). We propose to adapt large-scale text-to-image generative models for this task. Specifically, by using text prompts to represent the object-state composition, we benchmark several existing methods such as Textual Inversion [46] and DreamBooth [144]. We also propose a new method by introducing new tokens for objects and states and simultaneously fine-tuning language and diffusion models. Lastly, we discuss the challenges and limitations of

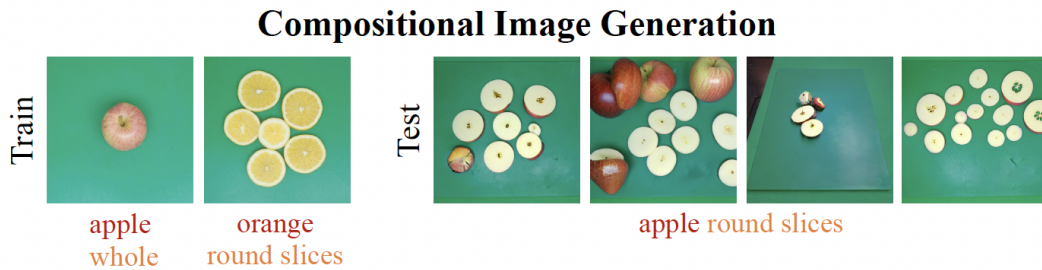


Figure 6.1: Given training images of various **objects** in various **states**, the goal is to generate images of unseen combinations of **objects** and **states**.

prior works as well as the proposed generative model with an extensive evaluation.

6.2 Related Work

Object states or attributes have recently received significant attention for recognition tasks, in images and videos. Some of the common works and their dissimilarities with the proposed dataset are mentioned here.

Attributes of Objects. In the image domain, states are often referred to as attributes for Compositional Learning of attribute-object pairs. Attributes describe the visual properties of objects, such as shape, color, structure and texture. The common datasets used are MIT-states [76], UT-Zappos [195], COCO-attributes [127], CGQA [118] and VAW [131]. All of these datasets consist of web-scraped images of various types of objects (from furniture to shoes and clothes to food items), which makes the variety of states very diverse. Most of the prior works [95, 106, 118, 119, 132, 135, 147, 158, 187, 191] focus on attribute-object recognition tasks using compositional learning but do not expand to image generation tasks due to the diversity in background and attributes. Some works in compositional zero-shot learning of attributes show visual disentanglement of attributes from objects [147, 179], however, they only hallucinate compositions of unseen attribute-object pairs in the feature space, rather than the image space. Moreover, even newer large vision-language models such as CLIP [138], DALL-E [139] fail to capture the subtle attributes of objects which are visually discernible [110, 197]. Therefore, the image generation task for objects with different attributes is still unexplored, which is a major focus of our work.

In our knowledge, compositional image generation for attributes (or states) and objects has not been explored much in the past, hence there does not exist much literature to compare with.

6.3 Compositional Image Generation

Large-scale deep generative models [140, 142, 145] trained on open-world big datasets have made significant breakthroughs in image generation in the last couple of years. These models, are typically conditioned using a text encoder and also support tasks such as zero-shot image generation, inpainting, image editing, and super-resolution without explicit training on these tasks. However, the performance of these models significantly degrades when it comes to compositional generation [33]. Our dataset, consisting of 112 real-world object and state combinations, is well-suited to test the compositional capabilities of generative models.

6.3.1 Task Description

The goal of the task is to either train from scratch or fine-tune an existing generative model using the (object, state) pairs provided in the training, and generate images from unseen compositions. We consider all 20 objects, each object captured in up to 7 different states, *i.e.*, all the states excluding `peel`. We split the (object, state) combinations into a training set consisting of 87 combinations and a test set consisting of 25 combinations. The training set covers all objects and states used in our dataset, but it does not overlap with the test set in terms of (object, state) combinations. In other words, for each combination of object and state present in the test set, the training set includes exactly one of either the object, or the state, but not both. We also ensure that for each (object, state) combination (o, s_i) in the test set, there exists a combination (o, s_j) in the training set, where s_i and s_j belong to the same state related group defined in Section 5.3.1. This setting ensures that all object and state information are available in the training set. Each combination in our dataset has 8-12 images, resulting in a total of 1032 images in the training

set and 296 images in the test set. The exact split is provided in the appendix along with some examples.

6.3.2 Methods

6.3.2.1 Stable Diffusion (SD)

We briefly describe classifier-free guidance in diffusion models. Diffusion models generate an image from Gaussian noise via an iterative denoising process. Expected mean square error is used as the denoising objective:

$$\mathcal{L}_{\text{Diff}} = \mathbb{E}_{\mathbf{x}_0, \epsilon, t \sim \mathcal{U}(0,1)} [\| \epsilon - \epsilon_\theta(\alpha_t \mathbf{x}_0 + \sigma_t \epsilon, \mathbf{c}) \|^2] \quad (6.1)$$

where $\mathbf{x}_0$ is an image and $\mathbf{c}$ is the optional condition from the training data. ϵ is the additive Gaussian noise. α_t, σ_t are scalar functions of time step t . ϵ_θ is the diffusion model with trainable parameters θ . For sampling images from the text condition, SD employs classifier-free guidance [69], such that at every time step (during sampling), predicted noise is adjusted via:

$$\hat{\epsilon}_\theta(\mathbf{x}_t, \mathbf{c}) = \omega \epsilon_\theta(\mathbf{x}_t, \mathbf{c}) + (1 - \omega) \epsilon_\theta(\mathbf{x}_t) \quad (6.2)$$

where ω is the guidance scale. In our experiments, ω is set to be 7.5 in all methods using it. For details on the SD, refer to the original work [142]

As our first baseline approach, we sample zero-shot images from SD with a text prompt “An image of o_i cut in s_j style”, where o_i is the i^{th} object and s_j is the j^{th} state of the object. Zero-shot generation with a pre-trained SD model doesn’t work as intended as shown in Fig-

ure 6.3, and the generated images often perform poorly in capturing the object state. Several recent works have shown that it is possible to extend models such as SD to achieve high-quality customized generations [46, 144, 202]. We evaluate several methods that have been proposed for compositional generation in the recent literature. We also propose a simple yet strong baseline by fine-tuning a Stable Diffusion (SD) model [142] along with textual inversion.

6.3.2.2 SD + Textual Inversion (TI)

Textual Inversion [46] introduces new tokens in the vocabulary and optimizes their embedding from the given images keeping SD frozen. We adapt the method for our task by introducing new tokens for the objects $\{o_i\}$ and the states $\{s_j\}$, and jointly optimize the embeddings of $\{o_i\} \cup \{s_j\}$ by providing (image, prompt) pairs from our training data. As before, the prompt is simply constructed as “An image of o_i cut in s_j style”.

6.3.2.3 DreamBooth

Next, we adapt DreamBooth [144], which fine-tunes the diffusion model along with the state-specific tokens. In our experiments, we fine-tune one model for each state in the dataset, where only the state token is learned. Original DreamBooth optimizes the diffusion loss as well as a prior preservation loss [144]. We observed that the latter significantly deteriorates the performance thus we skip it.

6.3.2.4 SD + Fine-tuning (FT)

We also fine-tune SD. In this baseline, only the parameters in the UNet of the diffusion model are optimized while keeping the text encoder fixed.

6.3.2.5 SD + TI + FT

Finally, we combine SD fine-tuning and Textual Inversion [46]. Specifically, on top of our SD + Fine-tuning baseline, we also adapt Textual Inversion by introducing new object tokens and state tokens and optimizing their embeddings along with the UNet parameters.

6.4 Experimental Setup

We use the above defined methods along with finetuning with our dataset. The dataset splits and evaluation metrics are defined below:

6.4.1 Dataset Splits

In the compositional image generation task, we split all (object, state) compositions into a training set consisting of 87 compositions and a test set consisting of 25 compositions. For each composition of object and state present in the test set, the training set includes exactly one of either the object, or the state, but not both. We also ensure that for each (object, state) composition (o, s_i) in the test set, there exists a composition (o, s_j) in the training set, where s_i and s_j belong to the same state related group defined in the previous chapter. Each combination in our dataset has 8-12 images, resulting in a total of 1032 images in the training set and 296

images in the test set. Fig. 6.2 illustrates the detailed dataset split used in the compositional image generation task. In this figure, training compositions and test compositions are marked with orange and teal, respectively. Unmarked compositions are not included in our dataset.

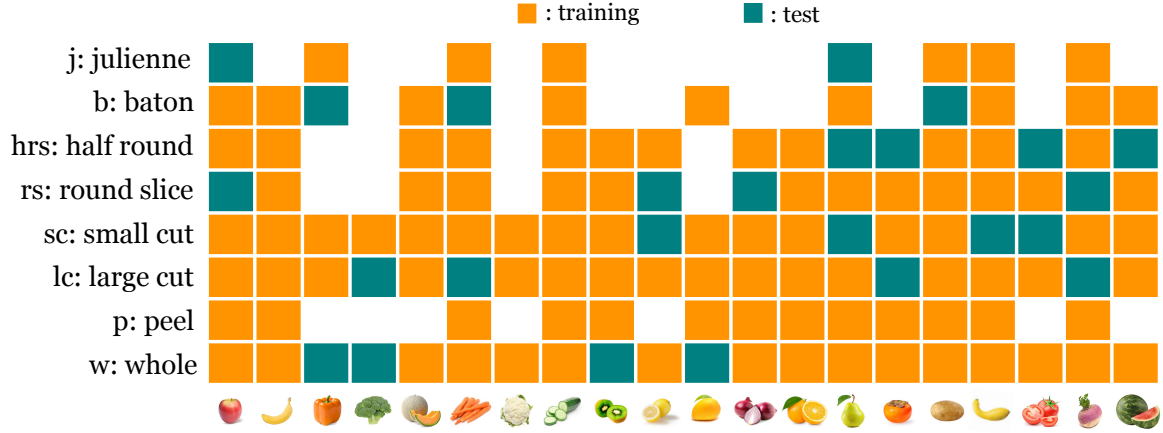


Figure 6.2: **Dataset Split used in The Compositional Image Generation Task.** Training compositions and test compositions are marked with orange and teal, respectively. Unmarked compositions are not included in our dataset.

6.4.2 Evaluation

We use both qualitative and quantitative measures to evaluate the capabilities of different methods. This section explains the details of different evaluation metrics we used:

6.4.2.1 Patch FID

Fréchet Inception Distance (FID) [68] is a commonly used metric to assess the quality of generative models. Given a set of real images and a set of generated images, FID compares the mean and std of Inception-v3 features of the two sets. For each composition and generative model, we compute patch FID using all real and 16000 generated patches, and report the average number for the test pairs. We hypothesize that using patch FID gives more weight to the object-

state patches, rather than the whole image, which includes almost 50% background pixels. We further calculate the lower bound for patch FID score by computing it between two sets of real images. Any score lower than that for this dataset can be disregarded as irrelevant. The determined lower bound for the patch FID score is 37.2. Specifically, we modify the standard FID by sampling 224×224 random crops from the real images, as well as the synthetic images. We use 32 patches per image.

6.4.2.2 Object/State Accuracy using a Classifier

To evaluate the correctness of objects and states in the generated images, we train a classifier on real images for classifying objects and states independently. This classifier is built on top of CLIP-ViT-B/32 [138]. Classification logits are obtained by computing the cosine similarity between the image embedding and text embeddings of all possible state labels or object labels.

Cosine similarities between the image embedding and text embeddings of all possible labels are computed as the classification logits, which are used to calculate the standard cross entropy loss for classification problems. During hyperparameter-searching, we fine-tune the CLIP model on a different training split that all (object, state) compositions are seen, and report the validation accuracies. One single model is used to predict both object and state. After deciding on all hyperparameters and training settings, we train our final-version object state classifier on all available data in our dataset to maximize its performance. We keep all parameters in the CLIP model learnable and train it 2000 epochs using a learning rate of $3e - 5$. We use a batch size of 128, and a warm-up cosine learning rate schedule [101]. To ensure the reliability of the classifier’s results, we train it on the training set from a different dataset split, where all (object, state)

Table 6.1: **Compositional generation evaluation.** FID, user scores, and classifier scores of various generative models. User Realism is on a scale of 1-5. (★) denotes that accuracies are evaluated on a seen data split. **Bold** represents the best result.

Method	Patch	User Realism ↑	Classifier Acc. (%)		User Acc. (%)	
	FID ↓		Object ↑	State ↑	Object ↑	State ↑
Real Images	-	4.65	87.5★	92.0★	73.6	84.0
SD	178.0	3.41	73.1	27.9	81.6	28.8
SD+TI	145.0	2.58	23.6	37.7	21.6	43.2
DreamBooth	139.9	3.56	53.5	74.2	61.6	72.8
SD+FT	88.9	3.78	70.5	67.7	72.0	65.6
SD+FT+TI	82.2	3.47	67.8	81.4	67.2	79.2

combinations are present.

6.4.2.3 User Study

We conducted a user study to evaluate the generated images. We took images from the test set as well as samples from our generative models and present them to 30 users. Each user was presented with 25 distinct images, randomly sampled with an even distribution from our models and the test set. After giving a tutorial to the users about the different objects and states present in our experiments, the users were asked to choose an appropriate object name and state label, as well as rate the image for realism on a scale of 1-5. We report the object and state accuracies as well as realism score in Table 6.1.

6.5 Results

6.5.1 Quantitative Results

Table 6.1 displays the performance of all baseline methods evaluated according to the metrics outlined in Section 6.4.2. Assessing image realism is a crucial evaluation metric for gener-

ative models; however, defining and measuring it can be challenging. Note that the patch FID values and user realism ratings do not align well. This is due to the disparity between the distribution of images in our dataset and that of typical occurrence of those objects in the real world. The patch FID metric measures the similarity between the generated images with those in our dataset, instead of the ones most typical in real world. In particular, our results indicate that SD achieves the worst patch FID score since it has not encountered our dataset before, whereas its user realism rating is more satisfactory. SD+TI has the lowest user realism rating and a poor patch FID score, which suggests that only training object/state embeddings is inadequate for generating high-quality images. DreamBooth receives a good user realism rating but a poor patch FID, indicating that the images it generates are realistic but not very similar to those in our proposed dataset. Finally, fine-tuning via both SD+FT and SD+FT+TI achieve better results for patch FID and user realism.

We next evaluate the accuracy of objects and states in generated images. It is worth noting that the classification task on our dataset is intrinsically difficult, which leads to imperfect user accuracy on real images. In general, the accuracy scores from classifier closely align with one from users, indicating that the proposed classifier is suited for evaluating compositional generation.

Our results show that SD achieves the best object accuracy but the worst state accuracy. This is possibly due to the lack of state variations in most existing large image datasets. SD+TI is the worst performer due to its limited learning capacity. On the other hand, DreamBooth, SD+FT, and SD+FT+TI attain better state accuracy. Among them, DreamBooth’s object accuracy is slightly worse as it is particularly trained for states. SD+FT achieves high object accuracy, and SD+FT+TI attains the best state accuracy with the help of fine-tuning and textual inversion together.

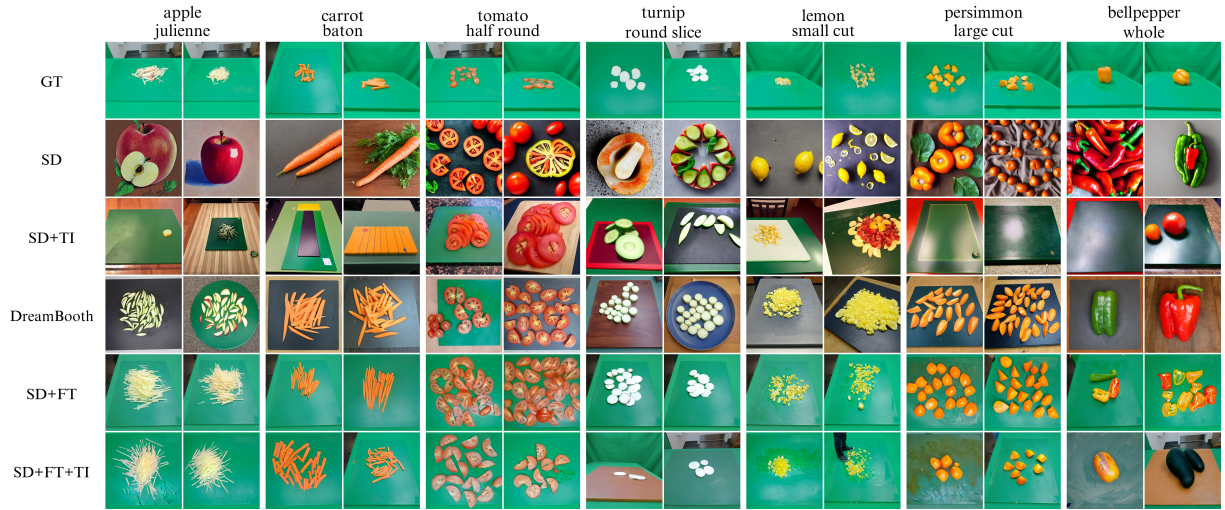


Figure 6.3: **Compositional Generation Samples.** Ground Truth (GT) real images are shown in the first row for reference. Seven object-state combinations in the test set are displayed, each with two generated samples for each method. Please zoom in to see details.

6.5.2 Qualitative Results

Fig. 6.3 displays the generated images from various methods for seven (object, state) combinations in the test set. The first row of the figure exhibits the ground truth real images for reference. We observe that vanilla SD often generates correct objects in random states, while SD+TI frequently synthesizes images without displaying the object. DreamBooth performs better than SD+TI, but worse than a simple finetuning of SD. SD+FT and SD+FT+TI perform well in terms of state generation. To better compare the compositional image generation performance of various methods discussed in [148], we show additional generated images from them for seven (object, state) compositions in Fig. 6.4 and Fig. 6.5, where the compositions are from the training set and test set, respectively.



Figure 6.4: **Additional Compositional Generation Samples Using Training Compositions** Ground Truth (GT) real images are shown in the first row for reference. Seven object-state compositions in the training set are displayed, each with four generated samples for each method. Please zoom in to see the details.

6.6 Ablation Studies

6.6.1 Green Screen Removal

One of the main challenges for understanding fine-grained object-state pairs with existing datasets such as MIT-states [76] is diverse backgrounds. Using them for training often leads to the model latching on to unwanted background details and missing out on the state understanding. Hence, we collected ChopNLearn with a clean green screen background for the benchmark tasks. While we acknowledge the limitations it poses to our trained models, we highlight that the green screen can potentially enhance our ability to generalize to diverse scenes. This can be achieved



Figure 6.5: **Additional Compositional Generation Samples Using Test Compositions** Ground Truth (GT) real images are shown in the first row for reference. Seven object-state compositions in the test set are displayed, each with four generated samples for each method. Please zoom in to see the details.

by segmenting out images and placing various backgrounds, along with scaled and rotated object-state images (Figure 6.6). As a proof-of-concept, we train a SD+FT+TI model on background-augmented images, and report the Patch FID, classifier object accuracy and state accuracy in Tab. 6.2. Note that here we employ a newly trained classifier that uses background-augmented images, and the patch FID scores are also computed based on these images. We further reference the lower bound of the patch FID as defined in Section 6.4.2. Due to the complex backgrounds introduced, the object accuracy and the patch FID of the new model are slightly compromised. However, it maintains a high and even improved state accuracy. This demonstrates the potential of the background-augmented ChopNLearn in enhancing fine-grained compositional image generation.

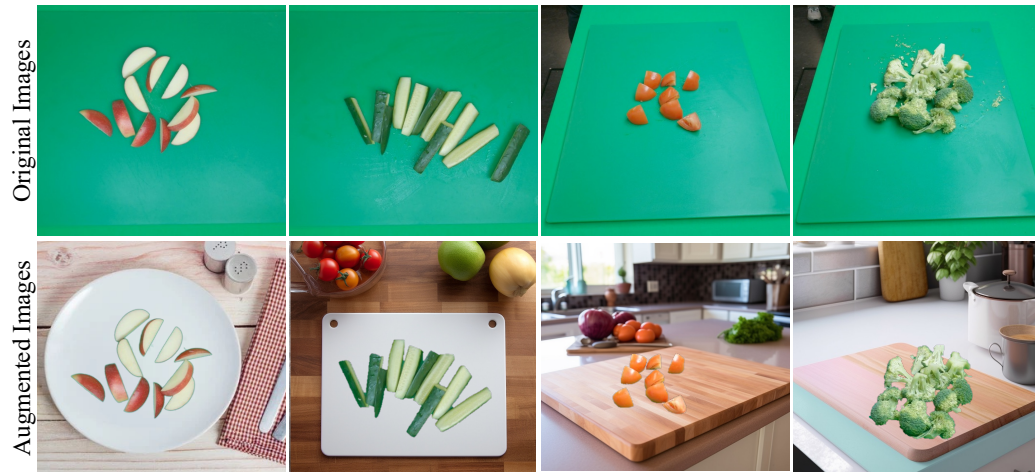


Figure 6.6: Preliminary results of using green screen to augment the dataset with different backgrounds. We continue to improve the transfer results by adding shadows and background matting.

Table 6.2: **Green screen removal evaluation** Both rows employ the SD+FT+TI but are trained using images with varying backgrounds. Classifiers specific to each dataset are trained to assess Classifier Acc. Validation images used to calculate Patch FID differ between the two rows. Patch FID Lower Bound is computed by evaluating the patch FID on one-half of the validation images relative to the other half. For further details, refer to Section 6.5.2.

Data Background	Classifier Acc. (%)		Patch FID ↓	Patch FID Lower Bound
	Object ↑	State ↑		
Green Screen	67.8	81.4	82.2	37.2
Various	46.3	82.3	133.6	46.4

6.6.2 Number of views

We assess the impact of the number of views on the image generation task in Tab. 6.3 using the SD+FT+TI setting. Using more views improves training data in terms of both quantity and diversity, yielding results with better patch FID and object accuracy, and maintaining high state accuracy even though generating images in more views is more difficult.

Table 6.3: **Number of views ablation results.**

View IDs	Object Acc. (%) $\uparrow$	State Acc. (%) $\uparrow$	Patch FID $\downarrow$
1	42.4	78.2	184.7
1, 2	56.8	81.2	121.4
1, 2, 3	66.2	78.3	115.4
1, 2, 3, 4	67.8	81.4	82.2

6.7 Conclusion

In this chapter, we propose a new task to generate unseen compositions of objects in different states, a skill known as compositional generalization for image generation. We also propose strong baselines and emphasize the challenges with granularity that can be learnt with this dataset. Admittedly, ChopNLearn is a small scale dataset with green screen background, which restricts the models trained on it to have specific biases. Nonetheless, this is the first attempt to understand how fine-grained states (cut styles) can be transferred to diverse objects. We explore this by using ChopNLearn as a test set for larger models, fine-tuning these models using ChopNLearn and trying them with or without a green screen background.

Chapter 7: Consistent Attribute-Object Insertion in Videos

7.1 Introduction

The emergence of image and video generation algorithms has opened up exciting new possibilities for utilizing generated data across various domains, including media production, AR/VR, and synthetic data for model training [36, 57, 133, 140, 142, 159]. However, unconstrained text-to-image/video generation suffices only in a limited set of scenarios. In practice, there is often a need for enhanced control over image/video generation processes, encompassing aspects such as character consistency, pose, and beyond. This need has prompted the development of numerous algorithms in the image generation domain, including inpainting [103, 142], LoRA[73, 144], and ControlNet [202]. These techniques ensure that the generated images adhere to constraints such as background, style, and pose. In the realm of video generation, algorithms such as [15, 49, 182] have addressed the demand for control, but many predominantly focus on comprehensive restyling of entire videos rather than the nuanced task of inserting or replacing specific objects within the video – a process commonly known as inpainting. Furthermore, while some approaches tackle object manipulation, they often extend changes to the entire scene’s background rather than solely concentrating on modifying the subject. In this work, we focus on the tasks of adding and replacing objects in a video (Figure 7.1). Unlike recent techniques such as those presented in [49, 182], we choose text-to-image diffusion models instead of text-

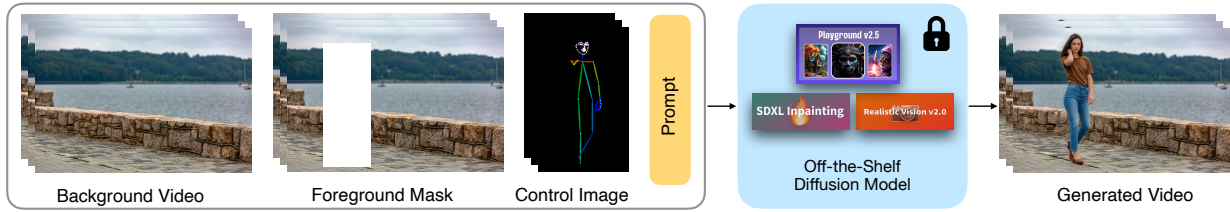


Figure 7.1: InVi inserts objects into a background video using a foreground mask, a control signal (e.g., pose, canny, depth map), and a text prompt by leveraging off-the-shelf diffusion models. It ensures that the inserted object aligns semantically with the text, is temporally coherent in time, and also conforms spatially to the control signal.

to-video diffusion models, as the latter necessitate significant modifications for our specific task. Moreover, by building upon text-to-image models, we circumvent the requirement for training on extensive video datasets and can leverage a wide array of established text-to-image models spanning various domains, including anime, art, photography, autonomous driving, and more. This strategic choice enables us to take advantage of pre-trained conditional models like inpainting [142], LoRA[73, 144], ControlNet [202], and seamlessly integrate them into our algorithm.

Existing approaches for video editing exhibit shortcomings, such as not generating all the frames [49] or requiring expensive per-video fine-tuning [182]. Methods like Tokenflow [49], which opt for a joint synthesis approach, however, generates only a subset of the required frames and rely on optical flow to generate the remaining ones. This limitation arises from the challenge of synthesizing all frames jointly, which becomes increasingly challenging due to GPU memory limitations, leading to performance degradation as the number of frames increases. On the other hand, methods like Tune-a-Video [182] require additional temporal layers and fine-tuning on the target video, leading to significant latency.

To tackle these challenges, we introduce InVi, a novel method for inpainting objects in videos. Leveraging off-the-shelf text-to-image latent diffusion models, our approach seamlessly applies to videos of any duration, eliminating the requirement for individual fine-tuning for each

video. In addressing object inpainting in videos, our method addresses two primary challenges: (1) Ensuring realistic blending of the inserted object in the target video, avoiding a resemblance to its appearance in the source image. (2) Ensuring consistency across frames during the video synthesis process. To achieve a seamless integration of the source image into the target image, InVi introduces a two-step inpaint and match process. Initially, the object is inserted into a single video frame, leveraging the effectiveness of image-based inpainting. Subsequently, the inpainted frame serves as the reference for generating subsequent frames, ensuring that video synthesis is conditioned on features within the domain of the target video rather than the source image alone. To maintain coherence across frames, InVi employs an auto-regressive architecture with extended-attention to incorporate features from the preceding frame while generating the current frame. Through experiments conducted on several videos from the DAVIS dataset and our own test set, which includes novel object insertion scenarios, we observe that InVi outperforms other methods by more than 40 points in background consistency metrics and is the preferred choice in nearly 70% of the videos in our user study.

7.2 Related Works

7.2.1 Conditional video generation and editing

Based on the progress in generating images from text with diffusion models [140, 142, 145], there has been an increase in works that address video generation [21, 57, 182]. This has facilitated the creation of videos from textual descriptions, which can be further refined to achieve video-to-video generation by using attributes derived from initial video inputs. For instance, Gen-1 [36] utilizes estimated depth as a conditioning factor, while VideoComposer [177]

uses a broader array of inputs such as depth, motion vectors, and sketches. However, most of these methods need explicit training on videos for learning motion [21, 57], and ensuring that these models generalize to arbitrary motion patterns requires access to carefully curated large video datasets, which are relatively fewer (or non-existent) than those available for images [151]. Additionally, substantial computational resources are required for the development of these models and their derivatives for conditional generation. To the best of our knowledge, there does not exist a text to video model which is trained end-to-end, which supports inpainting objects in videos, while providing support for using auxiliary conditions like pose, depth, edgemaps, etc., as is commonly available for images. To overcome the challenges associated with training such complex models on videos, some approaches resort to single-image editing, subsequently extending these modifications across video sequences by identifying and applying edits to corresponding pixels throughout the frames and their efficacy hinges on robust tracking. Various methods [55, 192] have employed techniques such as optical flow, keypoint tracking, or other forms of motion detection to address this challenge. However, these techniques are hard to scale to long videos, for consistent appearance changes in objects.

7.2.2 Adapting Image Models for Video tasks

Many methods have extended image-to-image models for swapping objects in videos. For instance, [81] modifies self-attention mechanisms in diffusion models, while [182] conducts per-video fine-tuning and employs inversion-denoising techniques for editing purposes. MasaCtrl [15], originally developed for image editing tasks, has been extended to video generation tasks and leverages the first frame generated as a reference to synthesize subsequent frames in the

video sequence. [98] and Fate-Zero [136] adapt image-to-image pipelines [14, 67, 168] for video editing by introducing modifications to cross-frame attention modules, incorporating null-text inversion, and more. However, most existing methods are limited to generating very short video clips. TokenFlow [49] produces keyframes and employs a nearest-neighbor field on diffusion features to extend keyframe attributes to remaining frames. However, as the video length increases, interpolation performance may degrade due to accumulated interpolation errors over time. In contrast, our model enhances spatio-temporal attention [81, 98, 136] with anchor-based cross-frame attention, enabling the generation of long videos with any desired number of frames. Our work also differs from TokenFlow [49] in its support for inpainting. [49] does not support inpainting, as it is tailored to preserve the structure and motion of the original video and cannot handle edits like changing the size, shape, pose or motion patterns of objects. We use similar ideas of latent inversion of the source video, but they can be of a video from a different domain, and we can use its pose or canny features to inpaint a similar object in new videos. This ensures sharp and consistent object insertion in new videos, while [49] fails in maintaining sharpness of a new object, due to its optical flow propagation in the latent space.

7.3 Preliminaries

We introduce concepts that are required to understand our methods. Readers are encouraged to refer to the methods for a more in-depth treatment.

7.3.1 Diffusion models

Diffusion Models [70] gradually introduce Gaussian noise to a sample $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ over T steps, yielding noisy samples $\mathbf{x}_t, t = 1, \dots, T$. The distribution of these noisy samples is governed by $q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{\alpha_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$, where β_t denotes the noise variance at a diffusion step t and $\alpha_t = 1 - \beta_t$. Eventually, this *forward process* leads to $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, rendering the image $\mathbf{x}_T$ as white noise. Conversely, the *reverse process* inversely applies the aforementioned procedure through the θ -parameterized Gaussian distribution: $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \beta_t\mathbf{I})$. The learning involves estimating μ_θ to be able to generate a data sample from noise in T reverse process steps.

7.3.2 Latent Diffusion Models (LDM)

LDMs [142] improved the learning and generation process by shifting it from the image space to a latent space. The image is encoded into the latent space using an encoder, E , and both the forward and reverse diffusion processes occur in this latent space. The latent space samples are converted back into an image samples using a decoder, D . The denoising model, based on the U-Net architecture [143], is composed of self-attention layers [169] and cross-attention layers [169] to seamlessly integrate textual conditions. These models are also referred to as text-to-image models as a text prompt can be converted into tokens and used within cross attention layers of the U-Net model.

7.3.3 DreamBooth-LoRA based fine-tuning

LoRA [73, 144] helps personalize the diffusion model by creating a unique prompt and “binding” it with a specific image. To achieve this “binding”, first, we generate a prompt with a unique identifier: “a [V] [class noun]”, where [V] denotes a unique identifier linked to the subject and [class noun] represents a coarse class descriptor of the subject (e.g. boy, horse, etc.). Next, we condition the diffusion model on this prompt and fine-tune it using the LoRA [73] technique, ensuring that the prompt aligns with the provided image. LoRA involves creating a duplicate set of the original diffusion weights, representing them with low-rank matrices, and exclusively training these low-rank matrices while maintaining the original network’s frozen state. The training low-rank matrices are then merged with the original frozen weights, preserving the architecture and keeping the inference time identical to the original model. This approach is the same for inpainting based diffusion models.

7.3.4 DDIM Inversion

DDIM-Inv converts a clean sample $\mathbf{x}_0$ to its noisy version in reverse steps:

$$\mathbf{z}_{t+1} = \sqrt{\alpha_{t+1}} \frac{\mathbf{z}_t - \sqrt{1 - \alpha_t} \epsilon_\theta(\mathbf{z}_t, t, p)}{\sqrt{\alpha_t}} + \sqrt{1 - \alpha_{t+1}} \epsilon_\theta(\mathbf{z}_t, t, p), \quad t = 0, \dots, T - 1.$$

The difference between the forward diffusion process (FDP) and DDIM-Inv is in the noise generation mechanism. In the FDP, noise is sampled from a Gaussian distribution, whereas in DDIM-Inv, the noise is the output of the U-Net model.

7.4 InVi: InVi: Inserting consistent Attribute-Object pairs in a Video

We build upon the concepts of Latent Diffusion Models [70, 142], DDIM inversion [49, 142] and Lora [73]. Readers are encouraged to refer to the methods or appendix for a more in-depth details. Given an input video $\mathcal{I} = [\mathbf{I}^1, \dots, \mathbf{I}^n]$ comprising n frames, a text prompt $\mathcal{P}$ describing the desired edit and a control sequence $\mathcal{C} = [\mathbf{C}^1, \dots, \mathbf{C}^n]$, InVi generates an edited video $\tilde{\mathcal{I}} = [\tilde{\mathbf{I}}^1, \dots, \tilde{\mathbf{I}}^n]$. As in LDM [142], the video frames are converted to latent feature using an encoder, E , and the corresponding encoded features are denoted by $[\mathbf{x}^1, \dots, \mathbf{x}^n]$. Similarly, the encoded features of the edited video are denoted by $[\tilde{\mathbf{x}}^1, \dots, \tilde{\mathbf{x}}^n]$. The edited video aligns spatially with the control sequence $\mathcal{C}$ and conforms to the semantic constraints outlined in $\mathcal{P}$. The text prompt, $\mathcal{P}$, offers generic semantic guidance, influencing factors such as object appearance. Alternatively, the desired edit’s appearance can be specified directly as an image instead of the text prompt, for which, we leverage LoRA [73]. In contrast, the control sequence $\mathcal{C}$ provides more nuanced control, such as pose or object shape. Various methods exist for providing spatial control, denoted by $\mathcal{C}$, such as depth maps, edge maps, and normal maps for generic objects, or human poses if the object is a person [202]. Next, we will describe each of the steps in our pipeline in more detail.

7.4.1 Generating the first-frame and pre-processing

First, given the object’s location in each frame via bounding boxes, we extract a region of fixed resolution by expanding these bounding boxes, as illustrated in Figure 7.2(a). We then insert an object into the first frame, for which we rely on a ControlNet-based inpainting diffusion model. This can be any off-the-shelf text to image inpainting model for prompt-based editing

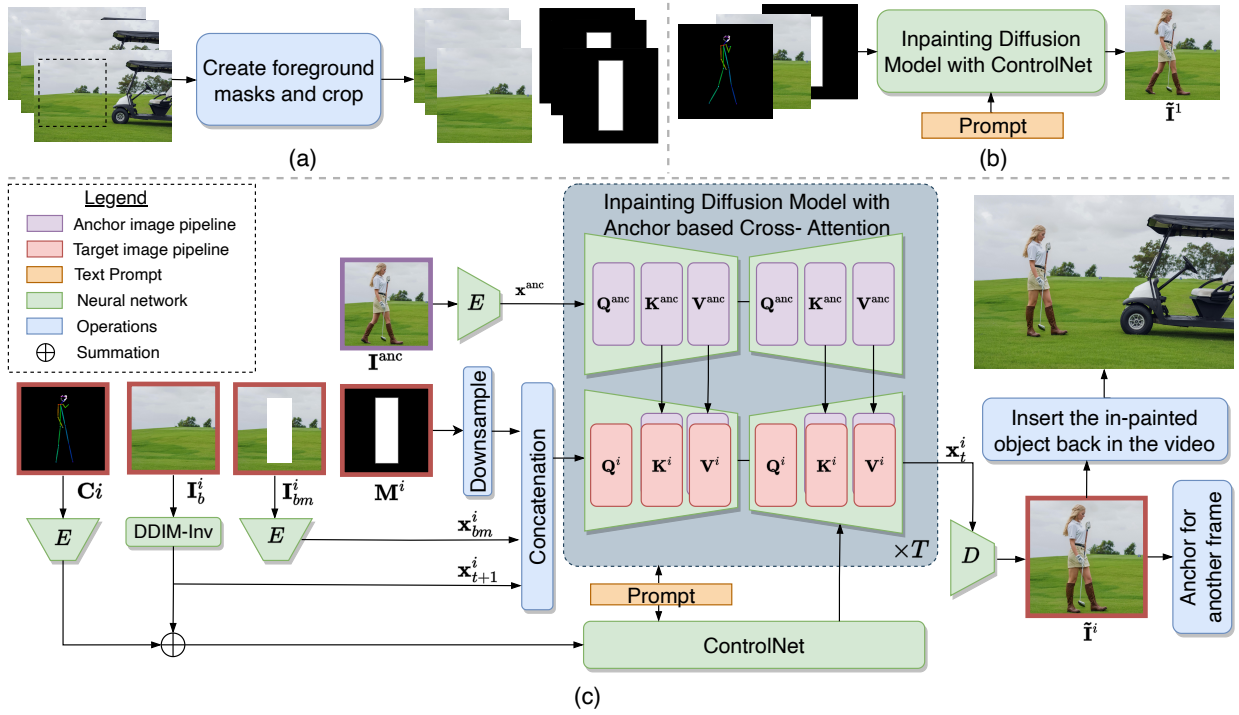


Figure 7.2: InVi [149] inference pipeline: (a) Given a video and object bounding boxes, first, we crop a region around the bounding box which is inpainted. (b) Next, we use a ControlNet-based inpainting diffusion model to inpaint the cropped region in the first frame. (c) To ensure temporal consistency when inpainting subsequent frames, we employ the previous frame as an anchor image. This is achieved by adapting the self-attention block of the denoising U-Net with extended attention. Specifically, we augment the Keys and Values of the current frame being inpainted with those of the anchor frame, allowing for consistent appearance. Finally, the inpainted crop is seamlessly integrated back into the input video.

and personalized model using LoRA[144] for reference image based editing. Once one frame is edited using image inpainting pipeline, we use this generated image as an “anchor” denoted as I^{anc} and edit the remaining frames.

To prepare the inputs for generating subsequent frames, we first pass the masked image through a VAE encoder, as done in prior work [142], compressing it into a lower-dimensional input ($64 \times 64 \times 4$ in our experiments). This input is then concatenated with a suitably downsampled mask of identical dimensions (64×64), indicating the area to be inpainted. In contrast to the inpainting pipeline in [142], which combines these inputs with Gaussian noise (sized $64 \times 64 \times 4$) during inpainting, we utilize the output after DDIM inversion on background frames as input for

the inpainting model. This step is crucial for maintaining video consistency, as DDIM inversion on the background frame ensures a consistent noise pattern across frames. Additional conditions such as pose or depth-map are provided to ControlNet, as outlined in [202]. A comprehensive wire diagram detailing all inputs for our pipeline is illustrated in Figure 7.2(c).

7.4.2 Temporally Consistent Frame Inpainting

To propagate information from the edited anchor frame $\mathbf{I}^{\text{anc}}$ to another video frame, we propose to use cross-frame attention mechanisms, circumventing conventional methods such as optical flow or explicit point tracking. Given an anchor frame, $\mathbf{I}^{\text{anc}}$, we incorporate it as an additional input to the diffusion model and replace the self-attention mechanism in the model with cross-frame attention.

Specifically, we use the anchor frame features to augment keys, denoted by $\mathbf{K}$, and values, denoted by $\mathbf{V}$, within the attention layers of the diffusion model. We denote the key and value matrices of the i^{th} frame as $\mathbf{K}_{i,l,t}$ and $\mathbf{V}_{i,l,t}$, respectively, where l is the layer index of the diffusion model and t is the diffusion step. Similarly, we denote the key and value matrices of the model obtained when the anchor frame is passed to the model as $\mathbf{K}_{l,t}^{\text{anc}}$ and $\mathbf{V}_{l,t}^{\text{anc}*}$, respectively. To edit i -th frame $\mathbf{I}^i$, we modify the self-attention module to a cross-frame attention using the key and value vectors of anchor frames as follows:

$$l^{\text{th}}\text{layer feature} = \text{Softmax} \left(\frac{\mathbf{Q}_{i,l,t} [\mathbf{K}_{i,l,t}, \mathbf{K}_{l,t}^{\text{anc}}]^T}{\sqrt{d}} \right) [\mathbf{V}_{i,l,t}, \mathbf{V}_{l,t}^{\text{anc}}], \forall l, \forall t \in [1, \dots, T].$$

*For brevity, we will omit the subscripts l and t where context makes it clear.

Note that this augmentation does not change the network architecture and does not require any learning of new parameters. Our method, as shown in Figure 7.2(c), utilizes softmax-generated attention scores to integrate $\mathbf{V}^{\text{anc}}$ features from the anchor frame. This process effectively enforces the temporal correspondence between the current frame and the anchor frame, and facilitates the propagation of value features from the anchor frames to the current frame through the multiplication of attention scores with $\mathbf{V}^{\text{anc}}$. By substituting the self-attention module with an anchor-based cross-frame attention mechanism, we achieve temporal consistency across the edited video frames.

We could use one anchor frame for the entire video, however, this is not ideal as the background appearance and the pose of an object gradually evolves over time. Therefore, once we generate a frame i , it serves as the anchor for generating the next frame $i + 1$. This sequential process is described in Algorithm 1.

Algorithm 1 InVi: Object Insertion in Videos

Input:

$\mathbf{X} = [\mathbf{x}_b^1, \dots, \mathbf{x}_b^n]$ ▷ Background video in latent space
 $\mathcal{M} = [\mathbf{M}^1, \dots, \mathbf{M}^n]$ ▷ Downsampled input mask
 $\mathbf{X}_{bm} = [\mathbf{x}_{bm}^1, \dots, \mathbf{x}_{bm}^n]$ ▷ Masked background in latent space
 $\mathbf{C} = [\mathbf{C}^1, \dots, \mathbf{C}^n]$ ▷ Conditional inputs
 $\mathcal{P}, \phi$ ▷ Target text prompt, ControlNet-based inpainting model
 $\{\mathbf{x}_t^i\}_{t=1}^T \leftarrow \text{DDIM-Inv}[\mathbf{x}_b^i] \quad \forall i \in [1, \dots, n], \forall t \in [1, \dots, T]$
For $t = T, \dots, 1$ **do**
 $\tilde{\mathbf{x}}_t^1 = \phi(\mathbf{x}_t^1, \mathbf{x}_{bm}^1, \mathbf{M}^1, \mathbf{C}^1)$
 $\mathbf{K}_{l,t}^{\text{anc}}, \mathbf{V}_{l,t}^{\text{anc}} \leftarrow \mathbf{K}_{1,l,t}, \mathbf{V}_{1,l,t} \quad \forall l$ ▷ save first frame features in a cache
For $i = 2, \dots, n$ **do**
 For $t = T, \dots, 1$ **do**
 load $\mathbf{K}_{l,t}^{\text{anc}}, \mathbf{V}_{l,t}^{\text{anc}}$ from cache
 $\tilde{\mathbf{x}}_t^i \leftarrow \phi(\mathbf{x}_t^i, \mathbf{x}_{bm}^i, \mathbf{M}^i, \mathbf{C}^i, \mathbf{K}_{l,t}^{\text{anc}}, \mathbf{V}_{l,t}^{\text{anc}})$ ▷ inpaint i -th frame with anchor features
 save $\mathbf{K}_{i,l,t}, \mathbf{V}_{i,l,t}$
 $\mathbf{K}_{l,t}^{\text{anc}}, \mathbf{V}_{l,t}^{\text{anc}} \leftarrow \mathbf{K}_{i,l,t}, \mathbf{V}_{i,l,t}$ ▷ Update cache with i -th frame features
Output: $\tilde{\mathbf{X}} = [\tilde{\mathbf{x}}_1^1, \dots, \tilde{\mathbf{x}}_1^n]$ ▷ Latents for inpainted frames at $t = 1$

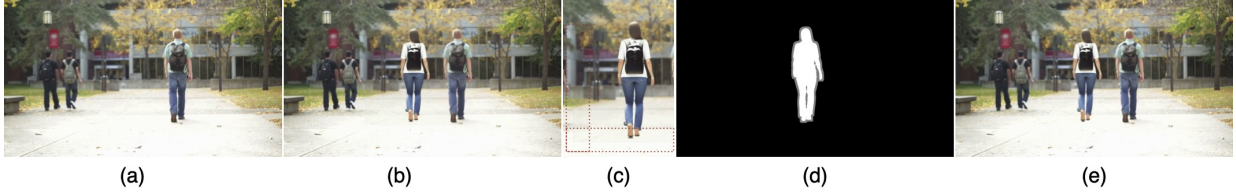


Figure 7.3: Post-processing to remove flickering square artifacts. a) Background image. b) Initial image generated from our pipeline. c) Zoomed-in view revealing artifacts around the inserted object. d) A trimap is generated to facilitate seamless blending of the object into the background. e) Post-processed frame showcasing the final result after blending the inserted object with the background.

7.4.3 Post-processing

After inpainting the object within the Region of Interest (RoI), an occasional subtle halo effect emerges, resembling a flickering square, in the vicinity of the inserted object. In the case of high-resolution videos, due to the limited training of base diffusion models on such resolutions (and an order of magnitude higher inference time), object inpainting can only be performed within a small RoI. The subtle differences which result from VAE based reconstruction are not very prominent (although noticeable) when the inpainted RoI is composed with the original frame but this gets amplified in a video as the object moves. Consequently, to achieve seamless and efficient blending for high resolution videos, we adopt a multi-step approach. Initially, we extract the mask of the inserted object using grounding-DINO [99] (for detecting arbitrary classes) and SAM [84] (getting object masks inside bounding boxes). Once the mask is obtained, we employ dilation to expand its boundary. Subsequently, we utilize Lama [164] to inpaint the pixels within this boundary, ensuring smooth blending throughout the video sequence as shown in Figure 7.3. This comprehensive strategy enhances visual coherence and minimizes any artifacts or discrepancies resulting from the object insertion process. Note that for low resolution videos where the entire frame can be inpainted, we do not require this step.

7.5 Experiments

Our method is evaluated across diverse datasets, including videos from the DAVIS dataset which are used in prior work [130], a selection of videos from the VIRAT surveillance dataset [123], as well as human-centric videos sourced from YouTube. Additionally, we curate our own video footage featuring cars, traffic cones, falling balls, and various moving objects to further assess the robustness and efficacy of our method. To replicate synthetic assets suitable for insertion into a 3D scene using simulation engines for applications like surveillance, AR/VR, and autonomous driving, we adopt two approaches. Firstly, we gather videos with a static camera over extended durations, enabling the extraction of conditional inputs from earlier time frames. Alternatively, we employ object removal software (RunwayML) to artificially remove objects from scenes, allowing us to utilize conditional inputs from the original video. We attach examples of these in the supplementary material. The spatial resolution of our videos (after cropping) is 384×672 or 512×512 pixels, and they consist of anywhere from 24 to 200 frames. Our evaluation dataset comprises of 30 text-video pairs. When training a LoRA-Dreambooth model, we train for 1200 iterations with a rank of 96, using a single reference image, without setting any regularization. In our experiments, we use the inpainting version of RealisticVision 5.0, which is based on Stable Diffusion 1.5 [142]. Our computational overhead (apart from DDIM inversion) is minimal compared to the per-frame baseline as we only double the FLOPs, and memory in the self-attention blocks of the transformer layers, while everything else remains the same.

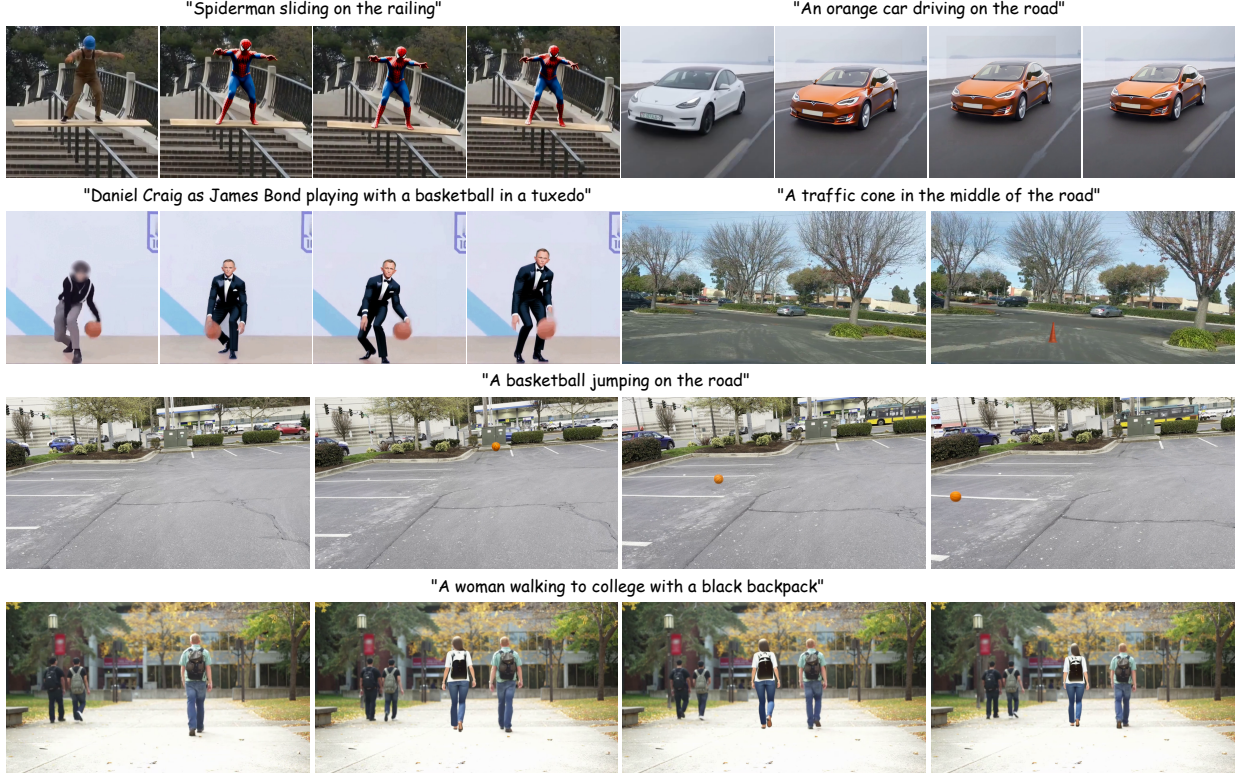


Figure 7.4: Qualitative results. The first image is a background frame from the video undergoing inpainting. Subsequent frames depict the video with the inserted object.

7.5.1 Baselines

We benchmark InVi [149] against several video editing methods that swap objects while preserving their structure. These include: (1) Fate-Zero [136], a zero-shot text-based video editing method; (2) Tune-a-Video [182], which fine-tunes the text-to-image model on the given test video; and (3) TokenFlow [49], which edits selected anchor frames and propagates the implicit flow from the keyframes to the rest of the video using an off-the-shelf propagation method. We employ PnP-Diffusion [168] based editing with TokenFlow. These methods alter the entire frame and do not preserve the background. Since there are no existing video inpainting methods that utilize off-the-shelf diffusion models, we include two additional baselines to evaluate the inpainting performance: (1) Per-frame diffusion-based image inpainting baseline using ControlNet; and

Table 7.1: Quantitative Results for object swapping (on the left) and object insertion (on the right). Evaluation for background consistency, temporal appearance consistency, and alignment with prompts.

	FateZero	Tune-a-Video	TokenFlow	InVi		Frm+Inp	TokenFlow+Inp	InVi
CLIP-Text	0.30	0.31	0.32	0.33	CLIP-Text	0.24	0.26	0.28
CLIP-Temp	0.95	0.95	0.96	0.97	CLIP-Temp	0.96	0.97	0.98
Back-L1	35.66	100.98	42.26	6.40	LPIPS	0.07	0.05	0.02

(2) a ControlNet [202] based inpainting pipeline for TokenFlow.

7.5.2 Quantitative Evaluation

Following previous work[49, 136], we use several metrics to evaluate various aspects of our object editing and inpainting techniques. Firstly, we compute **CLIP-Text**, which represents the average CLIP feature similarity between the generated frames and the target prompt, serving as an indicator of video-text alignment. For assessing temporal consistency within the video, we utilize **CLIP-Temp**, which measures the similarity of consecutive frames and averages the results across the generated video. Given the importance of maintaining background consistency while editing a specific object in videos, we use a background mask to evaluate **Back-L1** which is the average L1 distance between each pixel across corresponding frames of original video and edited video. Video editing is more common task, hence we compare with existing baselines which operate off-the-shelf without any training, for a fair comparison with our method. For inserting new objects in a video, all baselines and our method inpaint only the object, the background remains consistent. Hence, instead of **Back-L1** we use average **LPIPS** [203], which is patch based perceptual similarity score across consecutive frames of the video. Lower LPIPS means more similarity across frames. Finally, in addition to objective metrics, we conduct a user study to gauge the alignment of the edited video quality with human preferences, covering aspects such

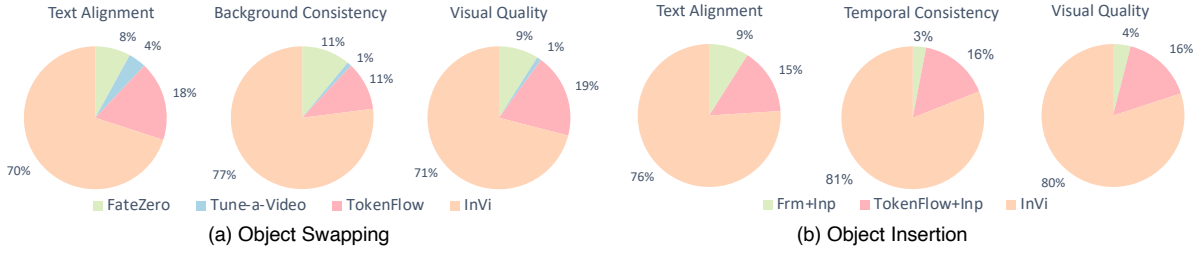


Figure 7.5: User Preference Study: InVi Outperforms Baseline Methods in text alignment, background and temporal appearance consistency and overall video quality.

as text alignment, background changes, temporal consistency, and overall impressions.

7.5.3 User Study

Video editing and inpainting is a subjective task, where quality of results cannot be evaluated with quantitative metrics alone. Hence, we also conduct a user preference study (with 15 users, 195 question responses), where users are shown videos of baselines and our method, and are asked to pick the video with best text alignment, background consistency (for edited video), temporal consistency (if the inpainted object is consistent in appearance across frames) and overall visual quality (least blurriness and extra artifacts). Figure 7.5 shows that users prefer InVi across all questions $\sim 75\%$ times. While Tokenflow [49] is preferred $\sim 15\%$ of times across all the qualitative categories. More details can be found in Appendix 10.

7.5.4 Qualitative Evaluation

As depicted in Figure 7.6, we conduct a comparative analysis of InVi against prominent baselines. Our approach, represented in the bottom row, demonstrates superior performance by closely adhering to editing instructions and ensuring temporal coherence in the edited videos. Conversely, other techniques often struggle to achieve both objectives simultaneously. Tune-

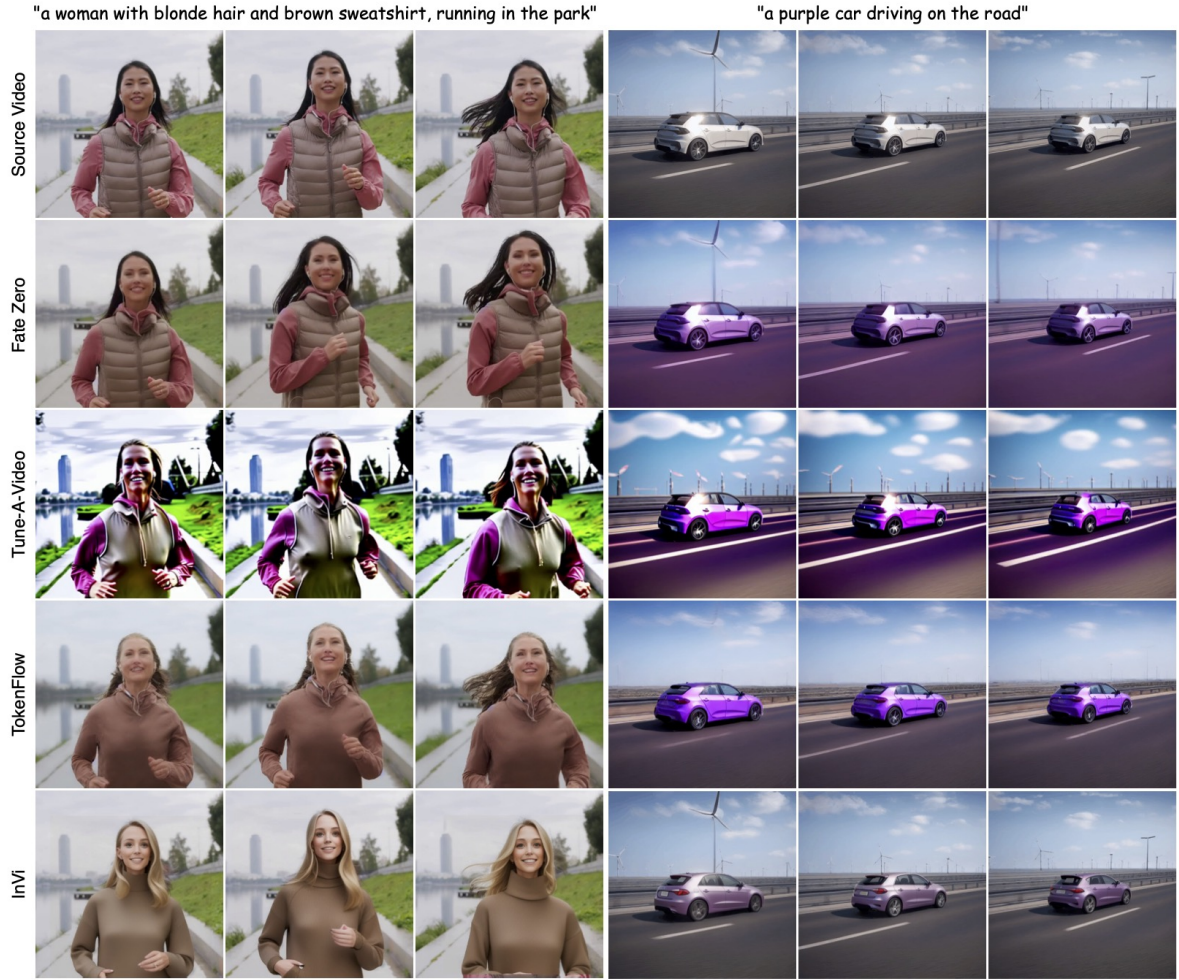


Figure 7.6: In our qualitative comparison, we contrast the performance of InVi with three baseline methods: FateZero, TokenFlow, and Tune-a-video. FateZero frequently diverges from the editing prompt, as seen in the woman running example. Meanwhile, both TokenFlow and Tune-a-video unintentionally modify the background. InVi, however, consistently yields results that closely align with the editing prompt while preserving the background.

A-Video [182] expands a 2D image model into a video model and fine-tunes it to follow the video’s movement closely. While effective for short clips, it encounters challenges in accurately capturing movement in longer videos, resulting in visual artifacts such as cartoonish appearances in the edited videos, as observed in the car example. Similarly, fate-zero also exhibits artifacts and deviates from the editing text-prompt closely. Although TokenFlow [49] yields reasonable results overall, it fails to perform well for inpainting. While it effectively edits rigid objects using flow, it struggles with inserting articulated moving objects like walking people. Moreover, all baselines exhibit inconsistencies in maintaining the background consistent with the source video, often modifying the background along with the object to be edited. Through a comprehensive user study and qualitative assessments, as shown in Table 7.1 and Figure 7.4, we demonstrate that InVi excels in preserving background consistency while inserting new objects into the scene.

7.5.5 Ablation: Advancing Beyond TokenFlow for Inpainting Tasks

TokenFlow [49] primarily works for video editing tasks, and relies on optical flow in the latent space. However, as seen in Figure 7.7 (Row 3), it results in color leakage in edited objects and unwanted color saturation changes in background colors. To compare with an inpainting pipeline, we modified TokenFlow to include a 9-channel inpainting based UNet for inference alongside ControlNet [202]. This mitigates the color leakage issues and enhances background consistency, as seen in Row 4 of Figure 7.7, but leads to blurry and unrealistic video outputs. TokenFlow relies on two main components: (i) Extended-attention, which selects and edits 5-6 frames sparsely from the video, ensuring consistent appearance across all frames, and (ii) Flow propagation across other frames, which computes latents for unedited frames through interpo-

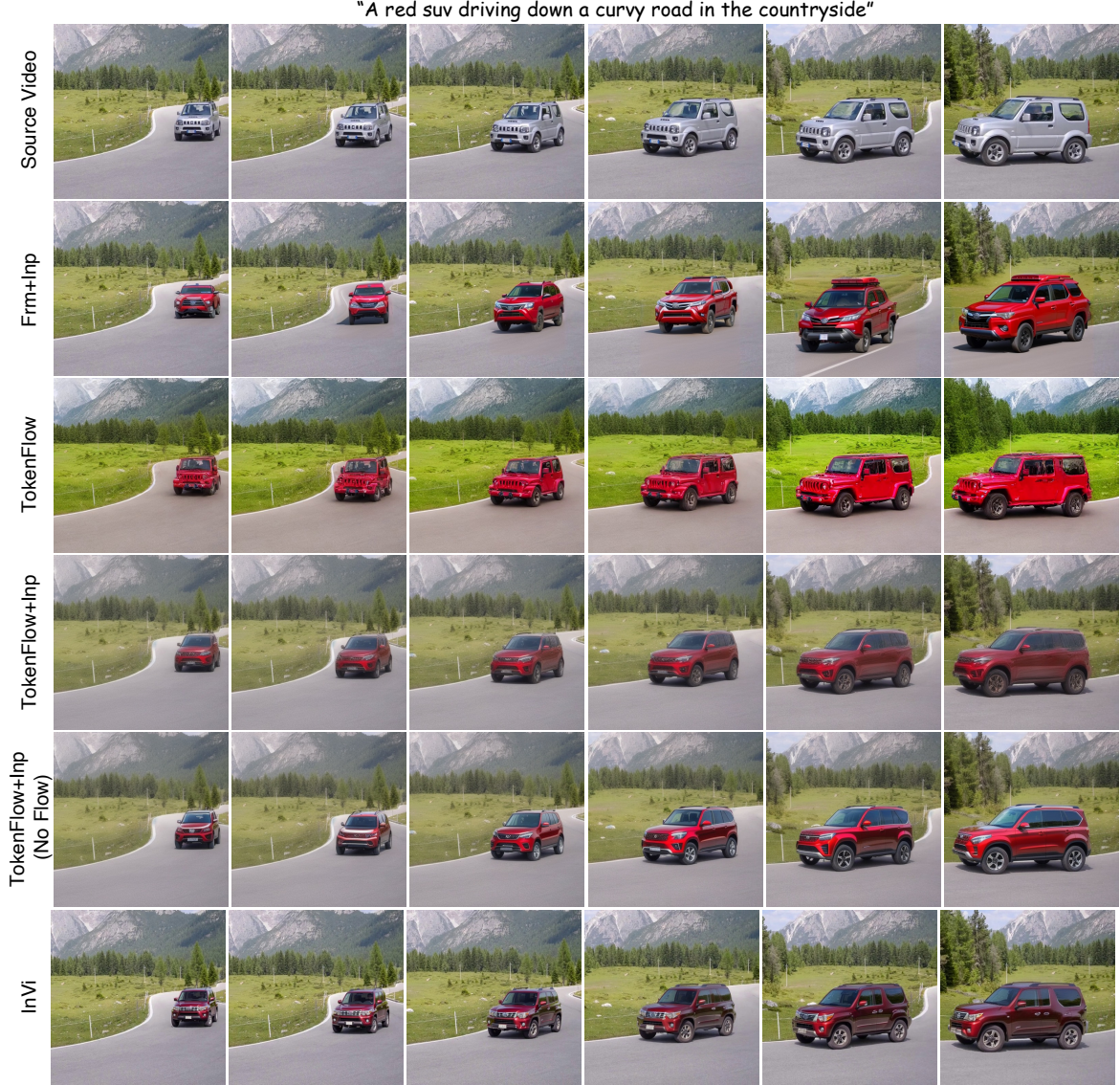


Figure 7.7: **Ablation experiments:** We make simple changes to the baseline methods. Frm+Inp conducts frame-wise inpainting using a constant seed and prompt. Tokenflow preserves the exact structure of the original jeep (like preserves the grills and mostly changes the color). TokenFlow+Inp combines ControlNet along with an inpainting method, serving as a baseline for inpainting, but leads to blurry results. TokenFlow+Inp (No Flow) removes the nearest-neighbor field computation from Tokenflow, and keeps the sliding window based inpainting of 2 frames at a time. Finally, InVi, which surpasses these methods in terms of clarity, consistency, and sharpness, establishing itself as the preferred choice for inpainting tasks.

lation in the latent space based on edited frames. Our hypothesis suggests that the blurriness observed in TokenFlow results from flow computation. We experiment with using only extended-attention with a sliding window, which results in sharper inpainted objects (Row 5). InVi edits one frame and recursively use the generated frames for editing the remaining video, which ensures consistency in appearance throughout the video. Because of the recursive approach, we do not need to jointly generate K frames sampled across the video, but we only use the previously generated frame while generating the next frame. Hence, our memory usage only increases by a factor of 2 compared to Tokenflow [49].

7.6 Conclusion

We presented a new approach to use text-to-image models for video inpainting tasks, using off-the-shelf models which operates without the need for video-specific training. By harnessing DDIM inverted latents extracted from the source video and incorporating the structural information of new objects via conditional ControlNet inputs, InVi seamlessly inpaints new objects into scenes. Utilizing anchor-frame based extended-attention for editing frames, InVi ensures both consistency in appearance and structure of the inserted object. Our method surpasses existing baselines, showcasing significant enhancements in temporal consistency and visual fidelity. Moreover, unlike prior methods, InVi efficiently handles longer videos with limited GPU memory and enables the insertion of dynamic objects without requiring an explicit motion module. One limitation of our work is that our method relies on 2D bounding boxes in each frame, which can either be provided by the user or estimated using the geometry of the scene. In future work, we plan to automate the generation of these boxes using GPT based layout generation techniques,

so that it can be more broadly applicable. As our work builds upon existing image generation methods, we inherit both the positive and negative societal impact of such methods.

Chapter 8: Conclusion and Discussion

This dissertation introduces two fundamental pillars for enhancing visual comprehension through the deliberate incorporation of compositionality into deep networks. The first pillar involves learning to decompose and compose new concepts, thereby enabling generalization to both seen and unseen attribute-object pairs derived from primitive concepts. Additionally, this approach utilizes compositional understanding of attributes and objects to capture nuanced changes and associated actions for video action recognition. The second pillar centers on generating compositions of attribute-object pairs within the image domain, alongside the consistent editing and seamless insertion of new compositions in videos.

In the initial part of this thesis, we successfully demonstrated the capability to disentangle objects and attributes within the visual feature space. This achievement allows us to generate novel, complex concepts and refine object-attribute recognition models, as outlined in Chapter 2. Through comprehensive experiments, we have established the superiority of our method over previous techniques across three distinct benchmarks. Furthermore, we introduce a new benchmark tailored for the compositional zero-shot learning (CZSL) task featuring images of objects in natural settings, aiming to steer community focus towards more complex scene analyses. We also acknowledge the limitations of our work, particularly the challenge of multi-label attributes in objects, which we hope will inspire further research into CZSL under more realistic conditions.

Chapter 3 serves as a direct continuation of this work, enhancing the practicality and realism of CZSL. In the context of advanced models like CLIP [138] and DALL-E [139], the scarcity of labeled data remains a significant constraint, which these larger models circumvent by utilizing extensive labeled datasets. We introduce a novel task, OV-CZSL, which not only focuses on learning unseen compositions of known attributes and objects but also extends to include completely unseen attributes, objects, and their combinations. We have developed new benchmark splits, underpinned by robust methodologies applied to existing datasets such as MIT-states [76], C-GQA [118] and VAW-CZSL [131, 147]. Any CZSL model—or even large language models (LLMs)—can be adapted to address OV-CZSL by incorporating Neighborhood Expansion Loss, leveraging semantic transfer from pre-trained language embeddings. Despite these advancements, open vocabulary compositional learning remains a formidable challenge, with significant opportunities to enhance benchmarks, including datasets and evaluation metrics. Consequently, we encourage the community to delve deeper into the generalized compositional learning of attributes and objects.

In Chapter 4, we introduce a framework for recognizing object-centric actions through the lens of object states. By capitalizing on these states, we have developed a methodology to learn both object-transforming and context-transforming actions. A novel *fluent* attention module, equipped with affinity and differential amplifiers, plays a critical role in discerning the subtle similarities and differences between object states. Our findings also include successful generalization to unseen objects and out-of-domain datasets, as well as showcasing the capabilities of our approach in compositional action recognition tasks, which adapt to unseen state-object action pairs. Looking forward, we plan to explore the application of object states in procedural planning and action segmentation.

In the second segment of this thesis, beginning with Chapter 5, we introduce ChopNLearn, a novel dataset engineered to evaluate the ability of models to recognize and generate unseen compositions of objects in various states—a concept known as compositional generalization. Furthermore, in Chapter 6, we propose the Compositional Image Generation Task and benchmark the performance of leading-edge generative models. We highlight the limitations of existing methodologies, particularly their occasional failure to adapt to new compositions. Yet, this endeavor is merely the beginning. Understanding object states holds significance for a multitude of image and video applications, including 3D reconstruction, future frame prediction, video generation, summarization, and extended video parsing. We anticipate that our dataset will inspire the computer vision community to propose and explore new compositional challenges across images, videos, 3D contexts, and beyond.

Finally, in Chapter 7, we present a novel method for consistent attribute editing and object insertion in videos, leveraging readily available models without the need for video-specific training. Utilizing DDIM-inverted latents extracted from the source video and integrating the structural details of new objects through conditional ControlNet inputs, our approach, InVi, adeptly inpaints new objects into scenes. Employing anchor-frame based extended-attention for frame editing, InVi ensures both the consistency and structural integrity of the inserted objects across sequences. This method outperforms existing baselines, significantly enhancing temporal consistency and visual fidelity. Moreover, InVi efficiently handles longer videos with minimal GPU memory requirements and supports the insertion of dynamic objects without a dedicated motion module. One current limitation is our reliance on 2D bounding boxes for each frame, which are either user-provided or estimated from scene geometry. In future developments, we aim to automate this process using GPT-based layout generation techniques.

Chapter 9: Future Work

While this thesis has explored several approaches for applying compositional learning in recognition and generative domains, it represents only an initial foray into the complex problem of compositional generalization. The methods we have introduced can effectively understand and generate compositions of primitives observed during training with a reasonable degree of accuracy. However, real-world applications, such as in robotics, frequently require the ability to *generalize beyond these seen compositions*. For instance, a model trained to recognize or create an image of a baby playing with blocks might struggle to comprehend or synthesize a robotic arm stacking blocks due to a lack of corresponding training data. A significant challenge in compositional generalization also lies in the *order of composition*; composing multiple primitives is often non-commutative, meaning the outcome can vary depending on the sequence of composition.

9.1 Future directions using Chop & Learn

With the ChopNLearn dataset, we aim to pave the way for the community to create more granular and realistic compositions. Current generative models excel at creating fragments of objects, but assembling these fragments into a coherent whole that maintains correct proportions and scale remains a challenge for realistic generation. Additionally, managing a vast array of seen and unseen primitives and deciphering their relationships over extended timeframes continues to

pose significant challenges in both 2D and 3D video understanding and generation. Some of the potential future tasks that could be studied with the proposed new dataset Chop & Learn are mentioned below.

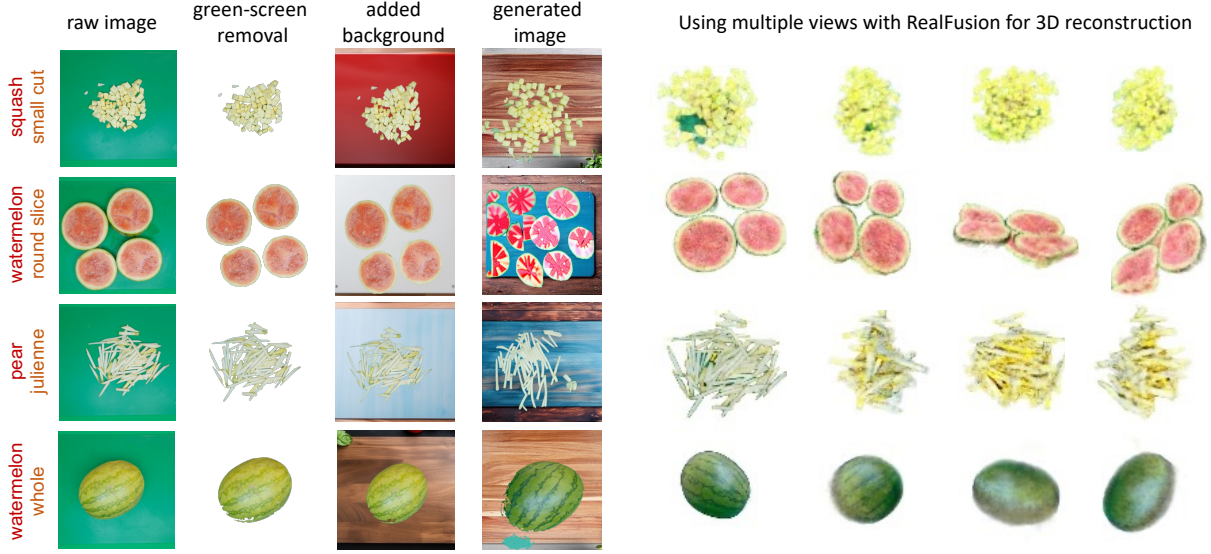


Figure 9.1: We show different uses of our dataset. The first column shows the raw images. The second column shows basic Python based green screen removal techniques on the dataset. The third column uses generated kitchen chopping board images replacing the green screen using the segmented object pieces. The fourth column shows results of training stable our benchmark SD+TI+FT model with the images without a green screen. Further the rightmost four columns show promising results towards 3D reconstruction of deformable objects, which can be a potential future research problem ChopNLearn can be used for.

9.1.1 Green Screen augmentation

We chose green screen to focus on the object states, and such that the object pieces can be segmented easily. As some preliminary work, in Figure 9.1, shows some results on how thresholding using simple opencv library functions works for segmenting the object pieces after styles of cuts are applied. Further, we use Midjourney [4], which is a Stable Diffusion [69] based text to image generator tool, to generate a set of images with chopping boards. For each camera angle, we generate 7-10 images, for red, yellow, blue, white and wooden chopping boards. The captions

used for this explain the color of board, the view or angle and the surrounds, for instance “empty plastic red colored chopping board in from this view point –style raw ” is one of the captions used for one view. Nonetheless, we are aware that achieving precise camera angles during image generation is a challenging task. Many of these models exhibit a bias toward placing certain fruits or vegetables around or atop the chopping board. As a result, we have occasionally supplemented the model with a reference image from the dataset, accompanied by a directive to “generate the same viewpoint and camera angle relative to the chopping board.” Despite these efforts, the outcomes we achieve remain suboptimal, particularly when it comes to three-dimensional perspectives. The top view seems to yield the most favorable outcomes for background generation. We utilize these images to substitute the backgrounds in segmented images. However, there are instances where the generated background’s view and angle do not align with those of the actual image from which the object segments originated. This occasionally results in sections of the objects appearing to levitate above the chopping board. While we acknowledge that the authenticity of these generated images may sometimes fall short due to occasional inaccuracies in green screen removal and misalignment between object positioning and background image angles, this experiment still holds value in mitigating green screen bias during model training. Additionally, our analysis with respect to green screen removal and background addition sheds light on a significant prospective challenge – the ability to mat and position objects convincingly from diverse angles within backgrounds to achieve a realistic effect. Hopefully, paves the way for new avenues of research and potential applications of ChopNLearn in the realm of detailed background matting.

9.1.2 3D reconstruction

Collecting data as well as generating 3D models for deformable objects is still an open problem. We demonstrate results of some preliminary experiments with our dataset for this task. We use RealFusion [109] to recover a promising 3D scene from a single image of our various cut states Figure 9.1. We believe that with our multi-view camera setup, this direction is worth exploring in future work for more accurate 3D reconstruction and can be an interesting task.

9.1.3 Long-term Video Parsing

We use compositional state recognition to further understand the temporal dynamics [35, 62, 63, 64] with the aid of a video parsing graph construction as previously explored in Ego-Topo [120] and VideoGraph [74]. Each clip in the training set has one state transformation (top example in Figure 9.2). We visualize the class activation maps corresponding to the most salient intermediate state transitions with Grad-CAM [152], to learn the transition in each frame of the video for training data. This is illustrated as a graph for a training video. Having learned multiple single transformations, we can now extend this knowledge to understand long activities, with multiple transitions. As shown in Fig. 9.2, we can learn state changes for `orange` from `large cut` $\rightarrow$ `small cut` using our training clip. Given a long unseen video with multiple clips, we can construct a state-transition graph to represent changes in state for a `watermelon`. Hence, by using an extensive array of videos, the process of learning transitions between individual states can be extended to encompass transitions between multiple states. This enables the creation of a self-supervised transition knowledge graph for comprehensive long-term video comprehension, as demonstrated in [35, 180].

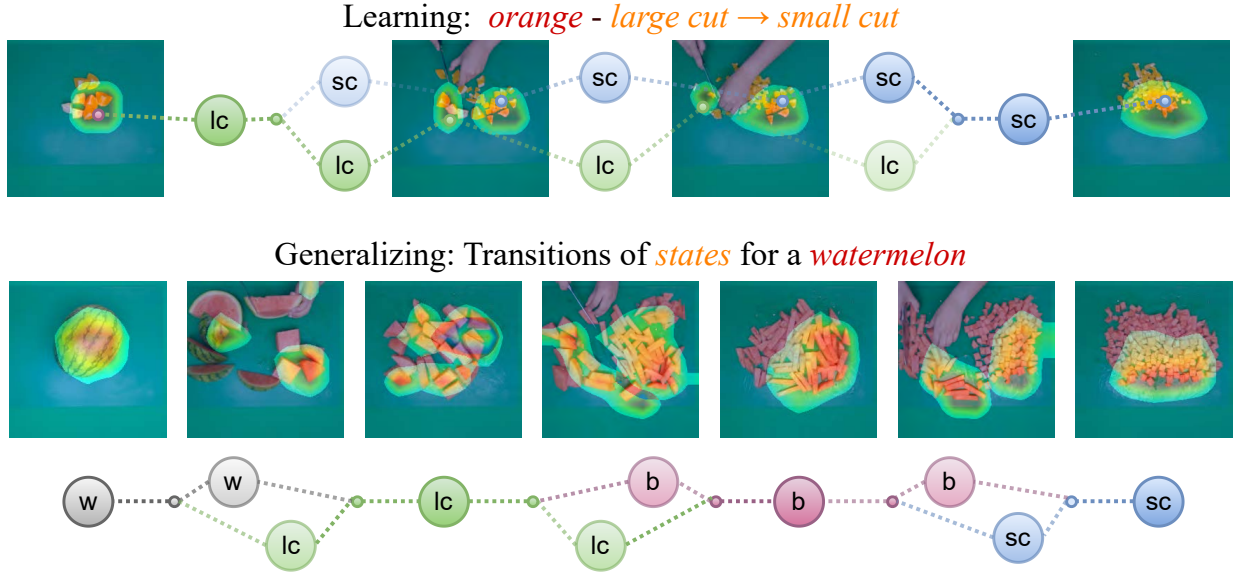


Figure 9.2: **Video parsing graph:** For a given video, we use Grad-CAM[152] on the intermediate frames to identify and visualize the class activation maps corresponding to the most salient states. Top: A training video clip has one transition of *orange* from *large cut* $\rightarrow$ *small cut*. Bottom: We can learn single transitions from training data, to generalize transitions in a long video with multiple state changes and parse the video as a graph.

9.1.4 Intermediate state generation and Video Interpolation

Generating granular and sparse states between two states such as ‘whole’ and ‘large cut’ is still a challenging and open problem. So far most of the methods rely on textual conditioning for guiding the generative models. However, textual embeddings as guidance also come with bottleneck of discrete embedding space. IN terms of video frame interpolation and intermediate object state generation, teh smbedding space needs to be smooth between two states, to generate a relevant frame in between. Moreover, it is difficult to define the intermediate states with the right set of textual prompts. For instance, we might say ‘apple in large cut with 25% cut’ or ‘potato with one piece cut’, however the textual embeddings do not carry much weight about the granularity of the state. A better way to measure of control the granularity of cut can be using a slider, to interpolate between two frames, as shown in Figure 9.3.

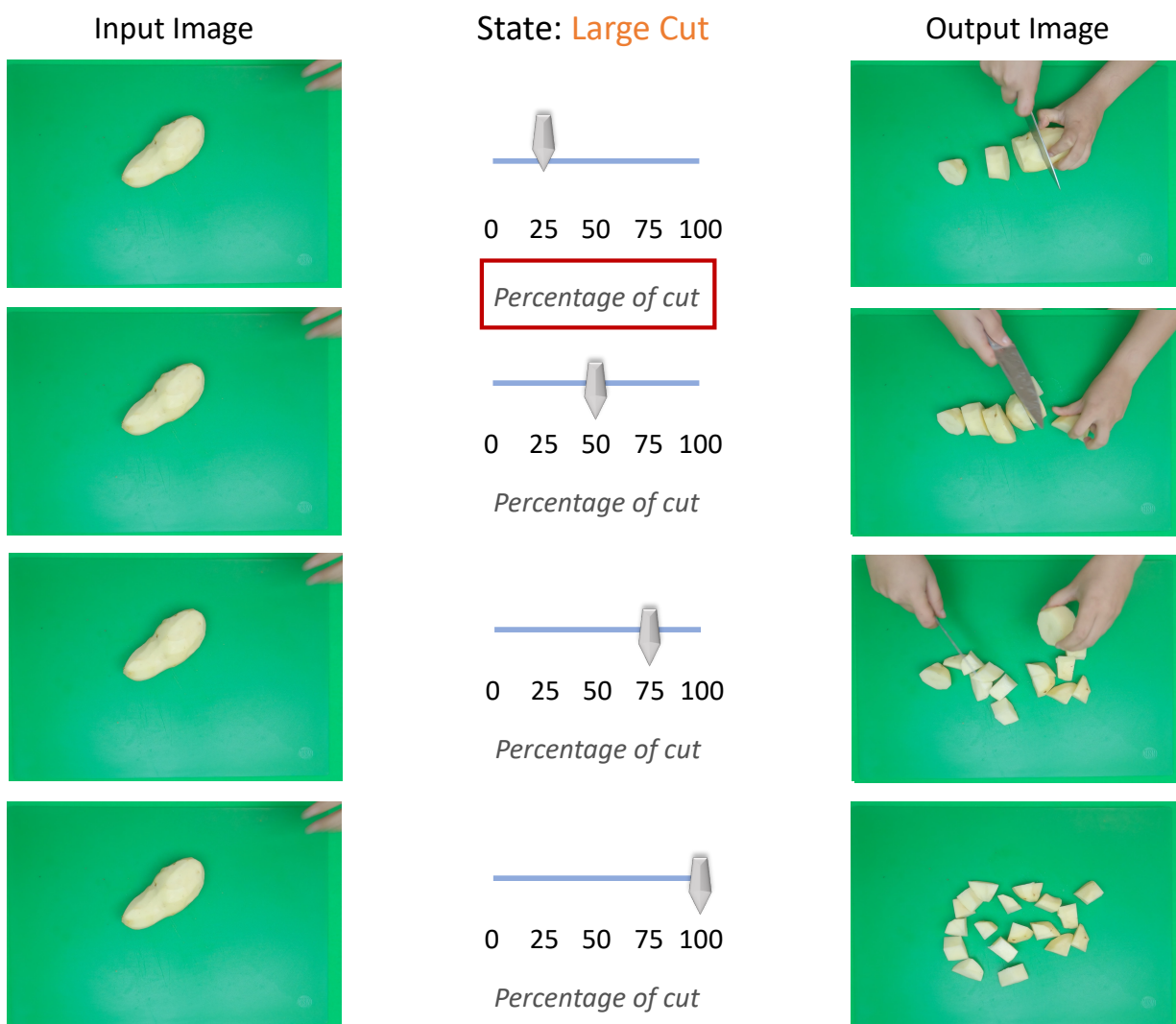


Figure 9.3: **Intermediate object state generation:** Textual conditioning is still limited to sparse and independent frame generation. However, using a slider control for generating intermediate states of cut between two well known states is an open problem.

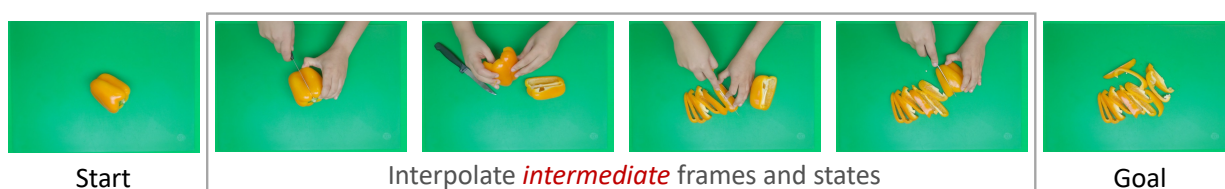


Figure 9.4: **Frame Interpolation:** Dense frame interpolation with start and end frames is also a new problem which ChopNLearn can support.

Once we can solve the task with more control-ability, the next step of generating closer frames with intricate difference would be the next step. However, another limitation with current models is, generated images might not be realistic in terms of composition. For instance, if we cut an apple in large cut, we might get 4-8 large pieces. While generating the images with current state-of-the-art image generation models, might generate random number like less than 4 or more than 8 pieces, which might not fit in together to generate a whole apple. BY gradual control and generation of frames one by one, for video generation task (Figure 9.4), this consistency might be possible to achieve yet still a very hard problem as of now. Moreover, understanding hand pose for slicing an object is also a well known challenge in dexterous object manipulation. Hence, this dataset opens new domains for exploration with different set of challenges and opportunities.

We hope this thesis underscores the importance of integrating compositional thinking in developing new methods for visual understanding, highlighting the need for models that can operate effectively across diverse vision applications in an open-world setting.

Chapter 10: Appendix

Chapter 2: Disentangling Visual Embeddings for Attributes and Objects

10.1 Benchmark Details

While experimenting with MIT-states [76] and UT-Zappos [195], we found several shortcomings with these datasets and discovered issues across all baselines using these datasets:

- Both datasets are small, with a maximum of 2000 attribute-object pairs and 30k images, leading to overfitting fairly quickly.
- Random seed initialization makes performance fluctuate significantly (0.2-0.4% AUC). Moreover, [7] found 70% noise in human-annotated labels on MIT-States [76].
- A new dataset C-GQA was introduced in [118], but the dataset is still small and we found a lot of discrepancies (kindly refer to the suppl.).

To address these limitations, we propose a new benchmark **VAW-CZSL**, a subset of VAW [131], which is a multi-label attribute-object dataset. We sample one attribute per image, leading to much larger dataset in comparison to previous datasets as shown in Table 2.1.

10.1.1 Dataset issues of C-GQA.

GraphEmb [118] proposes a new benchmark for compositional zero-shot learning. However, there are some issues with the dataset have been raised on their official github page [1, 2]. These issues are related to (1) the attribute-object pairs being placed into the incorrect train, validation, and test subset, and (2) there are missing images for a decent amount of pairs (20%), which could potentially affect the final experiment results. Due to [118] being unable to provide a corrected version of the dataset in time before the CVPR 2022 deadline, we were unable to run any experiments for C-GQA. Post the deadline, we did run some preliminary results where our method outperformed GraphEmb [118]. Although, a major issue we observed was for OADis, C-GQA [118] training set did not have similar attributes and objects samples for constructing I_{attr} and I_{obj} . However, we propose for learning compositional concepts, firstly disentangled concepts must be learnt, and for that, we require I_{attr} and I_{obj} . Hence, we do not report results on C-GQA for OADis.

10.1.2 Creating Dataset: VAW-CZSL.

We propose a new benchmark for the compositional zero-shot learning task (CZSL), focusing on images of objects and attributes in the wild that span across a much larger number of categories. We select the VAW dataset [131] to create our benchmark. VAW contains images originally from Visual Genome (thus objects and attributes in the wild). Every image of an object instance contains an object label and one (or possibly multiple) attribute labels. In the followings, we describe our steps in creating the VAW-CZSL benchmark, which shares some similarities with

the C-GQA dataset.

Different from C-GQA, we consider object instances whose bounding boxes are larger than 50×50 . C-GQA selected instances whose boxes are larger than 112×112 , which could possibly leave out small, narrow objects that are still recognizable from images. For every object instance, among its possibly multiple attributes label, we keep only one attribute that has the lowest frequency in the dataset (*i.e.*, the uncommon attribute) to be consistent with the standard CZSL benchmark. By keeping the most uncommon attribute and using the top-3 & 5 evaluation metrics, all methods will be evaluated based on whether they are able to rank this uncommon (but still representative) attribute in its top-3 & 5 predictions rather than always predicting the most frequent attributes. From this, we follow the similar steps from [118] to merge plurals and synonyms (e.g., $\{airplane, plane, aeroplane, airplanes...\}$, $\{rock, stone, rocks...\}$). We then keep only those attribute and object categories with frequency greater than 30 to make sure all primitive concepts have a decent amount of data for training and evaluating.

We use images in VAW-training as our training set, and use images in VAW-val and VAW-test for creating the validation and testing splits following the standard generalized benchmark in CZSL. We first merge VAW-val and VAW-test in one set, and follow similar steps mentioned in [118] to create a validation and test set of seen and unseen attribute-object pairs. At the end, we remove objects and attributes that no longer appear in the training set. This is because a model that has never seen an attribute (or object) will find it impossible to generalize to unseen pairs containing this attribute (or object). This problem happens with the C-GQA dataset where 8% of attribute and 22% of object categories do not exist in their training set. More details about dataset can be found in Table 2.1. The dataset splits are made publicly available at <https://github.com/nirat1606/OADis>.

10.2 Negative Impact of OADis

Our work is a new initiative in the direction of learning visual features for objects and its attributes. We present it as a prototype, or an alternative direction for understanding attributes-object pairs. Similar to any other work in vision, learning attributes of objects can have various positive implications, *e.g.* in object detection, knowing attributes can provide additional knowledge about the objects. However, knowing the additional information about attributes, it can be used for persuasion for marketing policies, for even worse factors. Even though it seems very far fetched ideas, but using attribute classification along with object detection, knowing the attributes people can build weapons and ammunition to either counter attack the present ammunition. Attribute classification can also be used on humans, to detect certain traits of human for bypassing large-scale surveillance applications. In general, attributes provide additional information for objects, which can be used negatively or positively.

Chapter 3: Composition of attributes and objects beyond seen primitives

10.3 Benchmark Details

10.3.1 Dataset Splits

Following the ZSL works [59, 176], we split attributes and objects into 75-25% split for training and testing. Since we are using ResNet18 [65] trained on Imagenet [30] as backbone for visual features, we make sure that attributes and objects common with Imagenet labels are

Table 10.1: **Exempt Classes which match the categories in Imagenet.** To ensure that OV-CZSL setup is truly Zero-shot Learning (ZSL) for attributes and objects, we must include all common attributes and objects seen in ImageNet [30] in the seen attributes and object sets. Hence, while creating seen attributes and objects sets for MIT-States, we first find out the objects and attributes in MIT-states [76] which are already present in ImageNet [30], and make sure these categories are part of seen attributes and objects.

Datasets	Common Objects	Common Attributes
MIT-states [76]	balloon, banana, brass, bubble, bucket, candle, castle, church, cliff, desk, drum, envelope, fig, laptop, lemon, library, necklace, orange, pizza, plate, pot, screw, tiger, vacuum, valley, velvet, wool	upright

part of training set (seen attributes and seen objects). This ensures that the unseen attributes and objects are truly zero-shot, and are never seen. A set of valid compositions of seen attribute and object pairs becomes the training set Y^s . Test set, denoted as Y^u has unseen compositions of seen attributes and objects $(AO)^*$, as well as, other sets of seen and unseen attributes with unseen objects (AO^*, A^*O, A^*O^*) . We further split Y^u 40-60%, to make validation and test splits, similar to CZSL. To evaluate learnability on seen compositions, test and validation split also have subset of Y^s . All split creations are random and are selected from 10 random splits, based on the balance between seen and unseen accuracies (refer supplementary). Table 3.1 shows statistics of the benchmark split.

10.3.2 Dataset Creation

Proposing a new benchmark is always challenging and requires meticulous efforts. Since the problem OV-CZSL, requires constructing multiple splits, we create multiple random splits and then pick the best split for each dataset by evaluating seen and unseen accuracy using two standard baselines. Overall, the steps involved in creation of benchmark splits for MIT-states [76] dataset is as follows:

- Step 1: As explained in Chapter 3, the proposed OV-CZSL task is combination of generalized ZSL and CZSL. As a visual feature backbone, we use ResNet18 [65], pre-trained on ImageNet [30]. To ensure that our setup is truly Zero-shot Learning (ZSL) for attributes and objects, we must include all common attributes and objects seen in ImageNet [30] in the seen attributes and object sets. Hence, while creating seen attributes and objects sets for MIT-States, we first find out the objects and attributes in MIT-states [76] which are already present in ImageNet [30]. We exempt those common attributes and objects, shown in Table 10.3, from being part of unseen attributes and objects A^* and O^* . In other words, these attributes and objects are always part of seen set A and O , for every random split.
- Step 2: We randomly select 20-25% attributes and objects as unseen attributes A^* and unseen objects O^* .
- Step 3: The set of all the compositions which have seen attributes and seen objects is denoted as AO . Among this set, 80% pairs make up the training set Y^s . The rest 20% pairs are part of the test and validation set, to evaluate the model performance on seen attributes, seen objects yet unseen compositions $(AO)^*$.
- Step 4: Iterating through rest of the valid compositional pairs present in MIT-States [76], we split the pairs in 3 other sets: AO^* has pairs which have seen attribute and unseen object, A^*O has pairs with unseen attributes and seen objects, and A^*O^* . Hence, test set Y^u composes of $\{(AO)^*, AO^*, A^*O, A^*O^*\}$.
- Step 5: In order to evaluate on pairs which are already present in training set, we select 25% and 30% pairs from training set AO and add 18% samples of these pairs in the validation

and test set. This becomes the set of seen pair samples in the test and validation set, which is used for stopping criteria. Similar split of seen pair exists in CZSL splits as well.

- Step 6: Lastly, we split Y^u into 45%/55% ratio as validation and test sets, such that each set of seen-unseen compositions are part of validation and test sets. Results can be found in chapter 3 [150].

This process creates one split. We randomly create 10 such splits for each benchmark. To ensure the stability of the benchmark split, it requires a balance in performance between the seen pairs AO and unseen pairs $\{(AO)^*, AO^*, A^*O, A^*O^*\}$. Hence, for each split, we run two common CZSL baselines, LabelEmbed (LE) [119] and CompCos [118]. Based on the AUC and difference between best seen and unseen accuracy, we choose one split as the benchmark split for OV-CZSL of the dataset. This process is performed for all datasets (C-GQA [118] and VAW-CZSL [131, 147]), but we only show the 10 random splits and their baselines performance for MIT-States [76] in Table 10.2.

10.3.3 Limitation for Open-world evaluation

Current evaluation setup in Chapter 3 [150], is closed world. That means the model is only evaluated on valid compositions of seen and unseen attribute-object pairs. However, in real world, more realistic setting would be open world, where during evaluation, we do not know valid compositions. CompCos [118] and KG-SP [79] are two state-of-the-art methods for Open World evaluation for CZSL task. However, when we train our splits for open world evaluation, these SOTA methods do not perform well on the unseen compositions: $\{AO^*, A^*O, A^*O^*\}$. To make sure the model can identify the invalid compositions such as “ripe dog”, CompCos [118]

and KG-SP [79] use feasibility scores, which removes the invalid compositions from feature space during inference. Moreover, this feasibility score is used during training as well, to avoid transfer of knowledge from seen compositions to unseen and invalid compositions. KG-SP [79] uses ConceptNet to compute the feasibility scores. We use the open-world evaluation for MIT-States [76] OV-CZSL benchmark, for CompCos [118] and KG-SP [79]. However, while evaluating in Open-world scenario, both CompCos [118] and KG-SP [79] fail to generalize to unseen composition $\{AO^*, A^*O, A^*O^*\}$. Our hypothesis is that for CZSL, all attributes and objects are seen. Hence, feasibility of pairs of seen attributes and objects, but unseen compositions can be computed through various means, such as using feasibility scores of ConceptNet features, which is used during training to transfer knowledge to valid compositions. However, for OV-CZSL, when unseen attributes and objects are not known during training, the feasibility scores describing valid compositions are only used during evaluation. Hence, the model fails to learn the validity of scores, during inference and fails to perform for the unseen compositions. Therefore, we conclude that OV-CZSL is more challenging setup than CZSL, and currently can be evaluated in Closed-world setup only.

Chapter 4: Attribute-Object Compositionality for Video Recognition

Recognizing Actions using object states

10.4 Dataset Splits

We use Epic-Kitchens-55 (EK-55)[28] as our primary dataset.

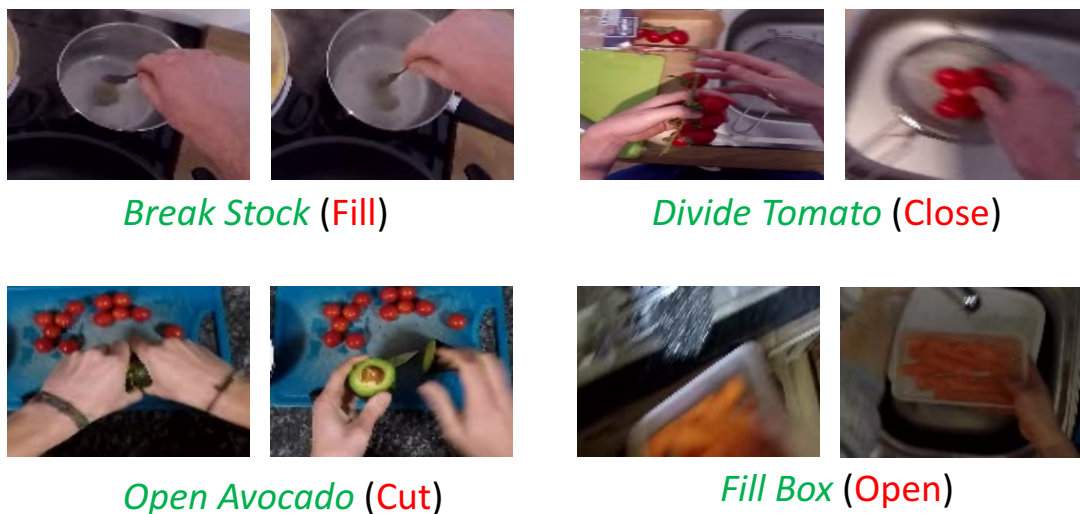


Figure 10.1: We show some of the misclassified actions from the test set for EK-55. The action in green is the actual label for the pair, whereas our model predicts it incorrectly shown in red. The first action is “break stock” since there is water in the pan, our model predicts it to be “fill” instead, next is “open avocado”, since the avocado is split into two pieces, our model predicts as “cut”. Similarly “divide tomato”, predicted as “close” and “fill box” predicted as “open”. This shows our model works for actions that have clear distinct before and after states, which are different from other action states.

10.4.1 Long-tail Distribution

Our pipeline includes object state detection in the videos, which reduces the total number of used clips for each action. Our processed dataset is a long-tailed distribution of classes, where the total number of samples varies between 1600 and 11. Figure 10.2 shows the distribution of samples for each class. The figure shows the training samples, seen transformation set (validation), and unseen transformations (test set) with different colors. Note that we do not follow the splits provided with the dataset but curate our own splits such that objects which are seen for an action in the train set are not present in the test set.

10.4.2 Multiple splits

In total, we use 13 object-transforming and 30 context-transforming actions. Since the number of object-transforming and context-transforming actions are not equal, we create and evaluate over five splits of dataset, which consists all object-transforming and randomly selected 13 context-transforming actions. Moreover, to make the object-transforming and context-transforming split consistent with each other, we choose the same number of majority and minority classes.

Firstly, for each type of action (object-transforming or context-transforming), we split the action classes based on the number of total samples. We create three separate action sets: Majority classes: classes with more than 65 actions, Mid-sized classes: classes that have more than 25 samples but less than 65 samples, and Minority classes: classes that have under 25 samples in total. Table 10.2 shows the names of such classes for object transforming and context transforming actions.

Among 13 classes for object transforming actions, 3 are Majority classes, 6 are mid-sized classes, and 4 are minority classes. For each split, from context-transforming action classes, we choose three majorities, six mid-sized and four minority classes, to make sure the distribution for context-transforming actions matches the object-transforming action. The object-transforming actions in each split remain the same. This step is crucial since we use a skewed distribution and make sure our model learns all classes in the same manner without any bias towards one class. Table 10.3 describes the chosen context-transforming actions for each split.

Table 10.2: To create five splits of the dataset, firstly, all actions are divided into the majority, minority, and mid-sized classes based on the number of samples. This table shows all actions and which class do they belong to.

Object-Transforming Actions		
Majority Classes	Mid-sized Classes	Minority Classes
close cut open	empty fill peel press squeeze wrap	break crush divide roll
Context-Transforming Actions		
Majority Classes	Mid-sized Classes	Minority Classes
dry insert mix move pour put remove take turn-off turn-on wash	apply filter fold scrape scoop shake sort turn	add adjust cook flip hold look measure pull search soak sprinkle

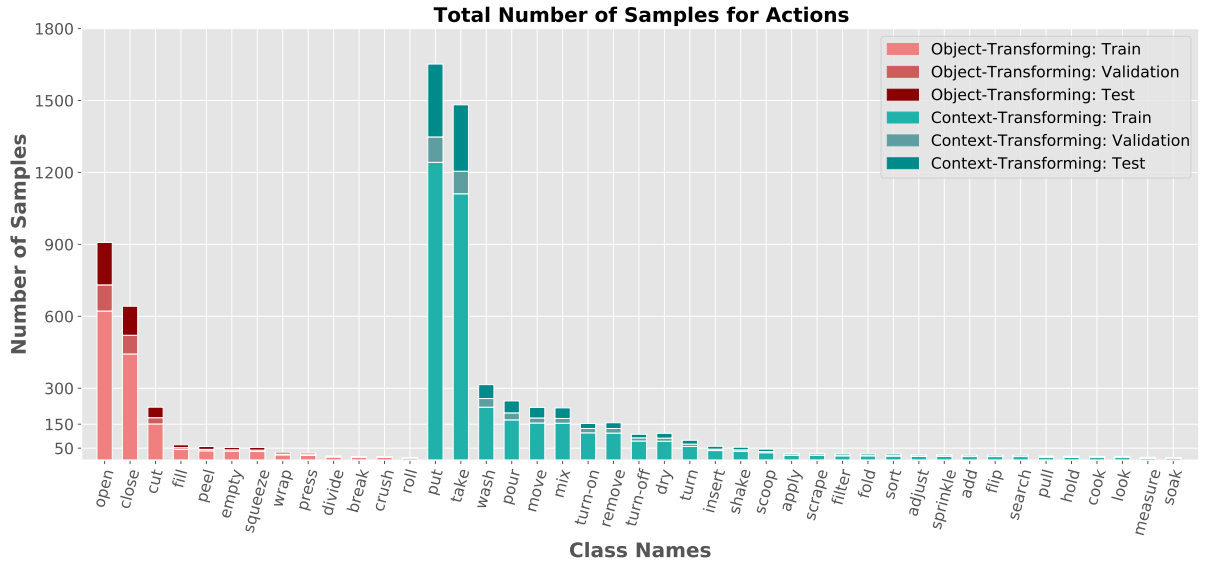


Figure 10.2: Distribution of Dataset: This histogram shows the number of samples for training, validation and test set. Our dataset is long-tail distribution, as observed in the graph. We show object-transforming and context-transforming actions with different colors (orange and blue respectively).

Table 10.3: This table shows the distinct context-transforming actions used in each split. Note that since we have fewer number of object-transforming actions, all those actions are part of each split. Combination of all these 5 splits cover total 30 context-transforming actions. We choose split classes randomly from all the classes, while making sure 3 majority, 4 minority and 6 mid-sized classes are in each split.

Split1	add	adjust	filter	fold	look	mix	scrape
	shake	sort	sprinkle	turn	turn-off	wash	
Split2	cook	filter	fold	look	measure	pour	pull
	remove	scoop	scrape	shake	sort	take	
Split3	add	apply	dry	filter	flip	fold	measure
	pour	scrape	shake	soak	take	turn	
Split4	add	apply	dry	filter	flip	fold	hold
	move	put	scrape	sort	sprinkle	turn	
Split5	add	apply	filter	flip	fold	insert	measure
	search	shake	sort	take	turn	turn-on	

Table 10.4: This table shows analysis of different temperature λ variable for the experiments. Although, changing λ does not significantly impact the overall accuracy, we optimize for mean classwise accuracy, which is maximum for $\lambda = 100$.

λ	All Actions		Object Transforming		Context Transforming	
	Acc.	Mean Acc.	Acc.	Mean Acc.	Acc.	Mean Acc.
0.1	42.45	12.11	46.97	13.08	31.51	9.84
10	43.96	11.06	48.42	12.45	33.87	9.63
50	43.62	11.01	47.94	12.37	33.89	9.65
100	43.62	13.10	48.42	14.42	32.78	10.71
500	41.77	12.87	45.52	13.32	33.33	9.26

10.4.3 Discrepancy in labels

It can be observed that some of the classes are similar to each other. For example, “break” and “divide” are very similar actions. However, these actions are applied to different objects. “break” is commonly used for eggs or paper towels, whereas “divide” is similar action performed on the dough, meat, or other vegetables. Apart from being minority classes (having fewer samples to learn from), some actions are very similar to each other. For few actions, transformed states are visually similar, i.e., “cut”, “break”, and “divide”, all actions transform different objects from whole→ pieces. These actions are more likely to be incorrectly classified by our model. Figure 10.2 we show some samples our model fails on due to their similarity with other actions.

10.5 Experimental Setup

We discuss the object state detection and details about our architecture in this section:

10.5.1 State detection

To emphasize the importance of object states instead of the complete frame, we show some examples of frame vs. the object state. Figure 10.3 shows the frame and small portion the object occupies, which we use for our model. It also explains the complexity of finding the small objects in different shapes and forms for the object detection system. For extracting object states, use a Faster-RCNN [141] detector, pre-trained on EK-55 [28]. This step is crucial for our pipeline, as the detector needs to be robust to identify objects in different shapes and colors. For each object, we can observe different forms and states detected by the object detector. This is possible due to sparse labels available with the dataset. The detector is trained with 295 object categories

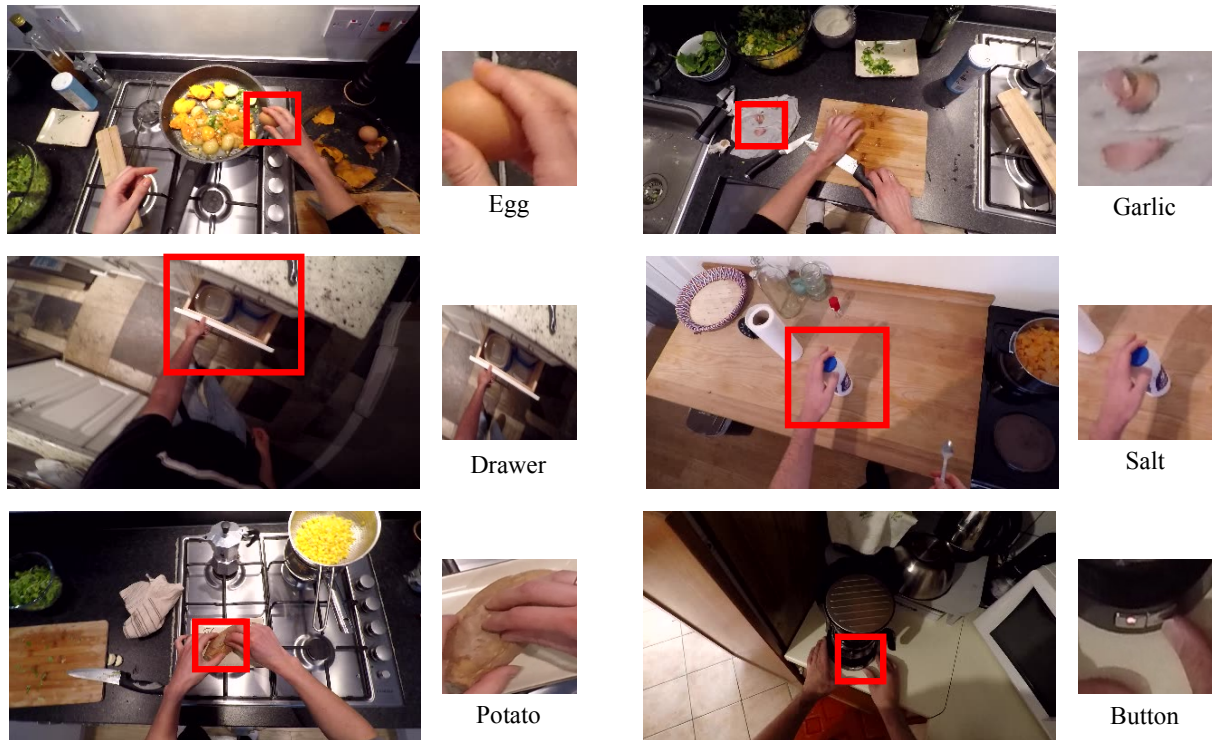


Figure 10.3: Our Pipeline uses object detector as a pre-processing step. We show that usually objects occupy a small area in the image, hence it is important to capture the object in order to understand the object state. Some examples here show the ratio of area occupied by the objects in the frames.

from EK-55. In Figure 10.4, we show diverse object states extracted using detector. Note that object states are not always the best since the detector mis-classifies sometimes. Among all action clips, detector only works 70% of the times. Moreover, we only use detections with more than 65% confidence to eliminate spurious objects. This step results in a smaller dataset than the complete dataset. Even with such constraints, some detections are blurry and do not capture the objects. Having noisy object states directly impact our action recognition accuracy. Among all mis-classified action clips, around 40% manually checked clips have noisy object state where the object is not even visible. We show some object states in Figure 10.5.

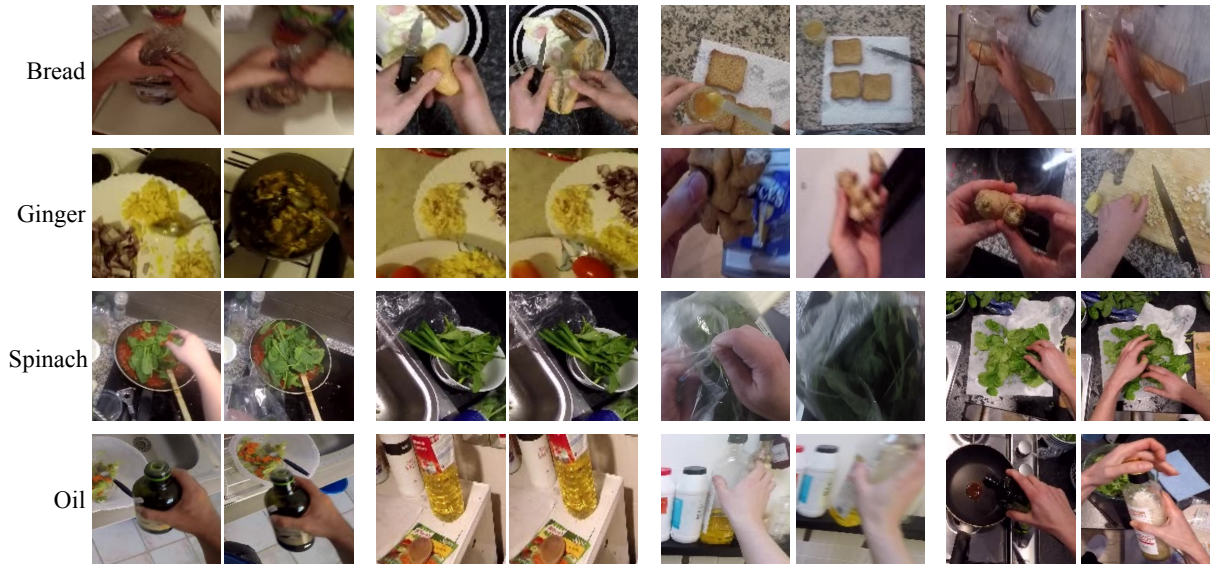


Figure 10.4: We show some qualitative results of our detector. Note that our detector is trained on the dataset, and hence captures diverse states and textures of objects.

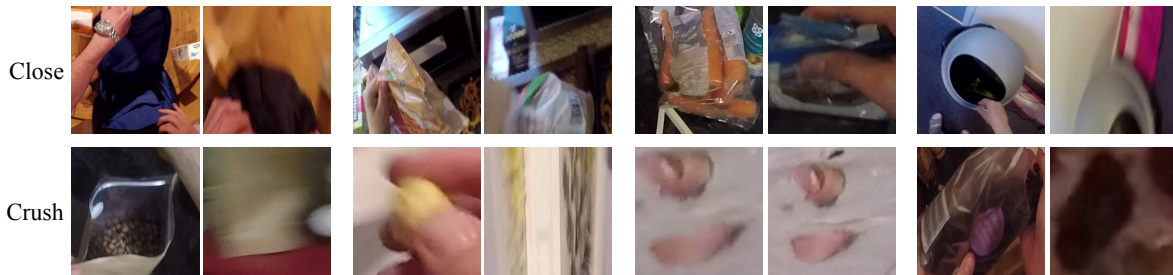


Figure 10.5: We show our detector misfires for certain objects qualitatively and do not capture the object at all. In these images, the before and after state of the objects is visible. It can be observed that sometimes, the detector finds the object in one state but does not capture the object at all in the other state. Here are the row-wise results explained for each action (left to right): Row1: *close bag* (no bag visible in both states), *close bag* (bag visible partially), *close bag* (different bags visible), *close bin* (bin partially visible) Row2: *crush coffee* (coffee not visible in end state), *crush ginger* (not visible in end state), *crush garlic* (no change in states), *crush pepper* (not visible)

10.5.2 Model and Parameters

Our model uses ResNet50 [65] features as input. We do not fine-tune the backbone network. Input features are passed through a linear layer, with ReLU activation, and then through the fluent attention module. Our model is robust to temperature variable, as shown in Table 10.4. Note that the results are shown for split1, similar observations follow for the rest of the splits. Since our dataset is long-tail, we prefer optimizing for mean classwise accuracy instead of overall accuracy, hence we use temperature $\lambda = 100$.

Compositional Action Recognition

10.6 Benchmark Details

Given the diversity of views and object types and styles, we can construct multiple training and testing splits. In this chapter 4 [148], we present results on three selected splits. For each split, we create training, test and validation set. The validation set is for evaluating the model on training classes, which consists of 10-15% unseen samples for the seen training compositions. Training and test sets have a disjoint set of compositional classes, in an 80-20% ratio. All of the splits in our dataset are created based on object-final state compositions in the videos.

We leverage these related groups defined in chapter 5 [148], to create different splits for training and testing. All splits use multi-view camera angles and involve creating seen and unseen object-final state compositions in training and testing sets. This ensures cross-view training and test splits, as used in other multi-view datasets [153, 171]. The training set consists of samples from three cameras, while the test set includes samples of compositions from one camera whose

view is never seen during training.

Similar to the three splits mentioned in the chapter 4, we explore multiple other splits, with different constraints to choose those 3 splits. We find that other splits were not as challenging for the existing baselines, and hence only propose 3 splits that are challenging. In Table 10.2, we present the results for splits 4-9, which were considered for the data. We only show I3D [17] based Transformer model for these splits. All the splits consider the constraints for the object-final state. The details of each split are as follows:

Split 4: This is the same as the split used for the Compositional Image Generation task (mentioned in chapter 6). We use the related groups to split the object-final states, such that objects which are seen with one of the states in a related group in training, are tested on the other related group during testing. This is also similar to Split 2 in the chapter 4, however, the multi-view constraint is not there. All the camera views are used for training and testing.

Split 5: In this split, we have the participant constraint. All samples from participants 1 and 2 are part of the training set, while samples from participant 3 are in the test set.

Split 6: This is a combination of split 4 and 5, which has two constraints: using related groups for splitting object-final states in different splits, and using only participant id 3 for the test set.

Split 7: This split is about multi-camera view. We use Camera 1,2,3 views in the training set, while the camera 4 view is part of the test set. No other constraint regarding related groups for splitting on the basis of object-final states is used.

Split 8: This split is similar to Split 1 in the chapter 4, without the multi-camera constraint. The object-final state compositions are split randomly into train/test. We use all camera views for both sets, without constraining to distinct views for each set.

Split 9: This split is similar to Split 3 in the chapter 4, without the multi-camera constraint. The

object-final state compositions are split based on random groups for objects and states. We use all camera views for both sets, without constraining to distinct views for each set.

We do not have a split having all constraints, *i.e.* participant constraint, related groups and multi-view constraint, since all of these together end up leaving a total of 400 video clips, which are very few for training and testing.

Chapter 6: Compositional Image generation for unseen Attribute-Object pairs

10.7 Questionnaire for user study

The purpose of conducting a user study was to see if our generative models were able to create images that were of high fidelity and stayed true to capturing the semantic understanding of the object-state composition provided as a text prompt. We chose 20 compositions from the test set, which are unseen as a pair in the training and finetuning of the generative models. These compositions from the test set are also given as a text prompt to five generative models, i) Dreambooth ii) Stable Diffusion iii) Stable Diffusion+Textual Inversion iv) Stable Diffusion finetuned on our training dataset v) Stable Diffusion + Textual Inversion finetuned on our training dataset. We evenly chose a distribution of 5 samples per composition, and including the test set + 5 generative models, we had 6 sets to sample images from. The total number of images used for the study was 750 and we asked 30 participants to label each of these images for their object and state as well as rate the realism on a scale of 1-5. We show some examples of images encountered in our user study in Figure 10.6 and a snapshot of how the user study questionnaire looks like in Figure 10.7.

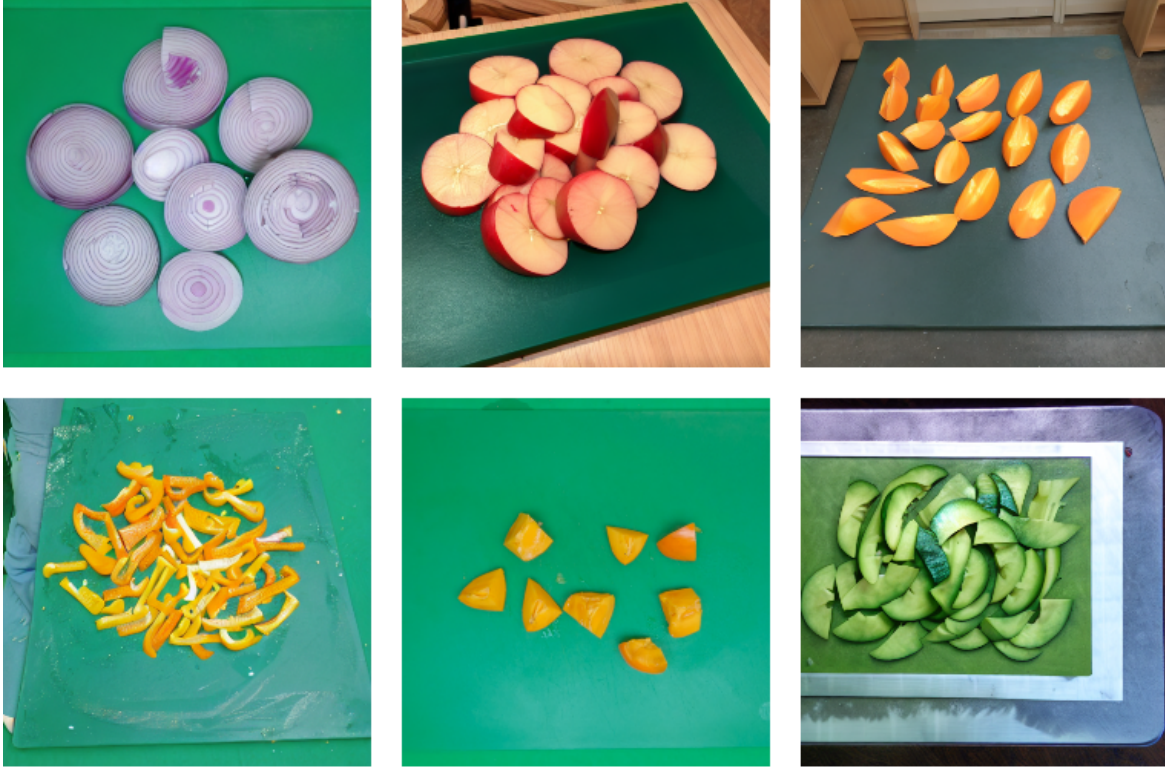


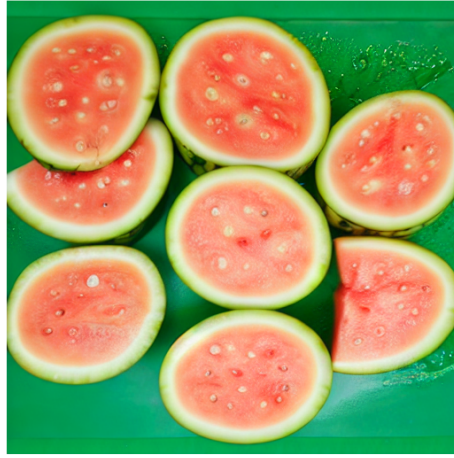
Figure 10.6: Examples of images from the test set and samples from the generative models presented to participants in the user study

Chapter 7: Consistent Attribute-Object Insertion in Videos

10.8 Questionnaire for user study

We evaluate our approach with a user study, for 13 text-video pairs and 15 participants. The users were shown source video, and 3-4 methods, randomized (with InVi included), and are expected to answer 3 questions. There are two types of videos: (a) videos for Object swapping and (b) videos for object insertion. In object swapping video, the source video also has the objects, which are modified with a prompt. In Object insertion, the source video do not have the object, and using conditioned control images, we insert a new object in the scene. Moreover, for object swapping videos, we use existing video editing methods are baselines: FateZero [136],

Please pick the closest match that occurs in a majority. If you are not sure about the answer, please make your best guess. Thank you!



*What Object is this?

Select one

*What State is this?

Select one

How realistic is the image? Please rate it on a scale from 1 (not realistic) to 5 (extremely realistic).

Rating



Figure 10.7: Snapshot of questionnaire presented to participants of the user study

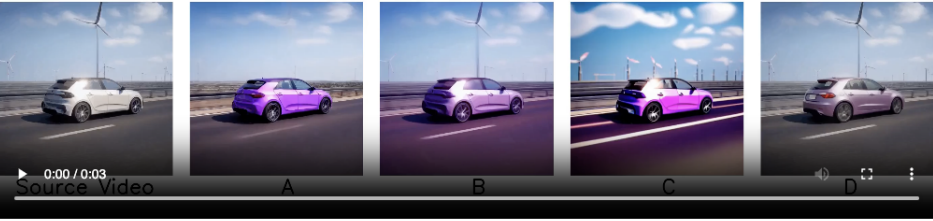
Tune-A-Video [182] and TokenFlow [49]. For object insertion, there are no video inpainting pipelines using text-to-image pre-trained models. Hence we use baselines Framewise inpainting (Frm+Inp) and TokenFlow with Controlnet and inpainting pipeline (Tokenflow+Inp). For object swapping, we ask users the following questions:

- Which video demonstrates the highest consistency with respect to the source video?

Choose the method which is BEST at preserving the background from the source video.

- Which video aligns most accurately with the provided text prompt?

a purple car driving on the road



1. Which video demonstrates the highest consistency with respect to the source video?
Choose the method which is BEST at preserving the background from the source video.

☐ A

☐ B

☐ C

☐ D

Which video aligns most accurately with the provided text prompt?
Choose the method which BEST captures the details in the prompt (given on top of the video).

☐ A

☐ B

☐ C

☐ D

Which video demonstrates the highest visual quality?
Choose a video which has the LEAST amount of extra artifacts (jitter, unwanted blobs), blurriness, unrealistic lighting and flickering.

☐ A

☐ B

☐ C

☐ D

←
→

Figure 10.8: Survey Preview for Object swapping videos. The users are shown 4 videos along with source video and prompt used for editing, to answer questions about visual quality, text alignment and background consistency.

Choose the method which BEST captures the details in the prompt (given on top of the video).

- Which video demonstrates the highest visual quality?

Choose a video which has the LEAST amount of extra artifacts (jitter, unwanted blobs), blurriness, unrealistic lighting and flickering.

a traffic cone on the road



Which video demonstrates the highest temporal consistency across new object appearance?
Choose the video with the BEST appearance consistency overtime for the new inserted object.

☐ A

☐ B

☐ C

Which video aligns most accurately with the provided text prompt?
Choose the method which BEST captures the details in the prompt (given on top of the video).

☐ A

☐ B

☐ C

Which video demonstrates the highest visual quality?
Choose a video which has the LEAST amount of extra artifacts (jitter, unwanted blobs), blurriness, unrealistic lighting and flickering.

☐ A

☐ B

☐ C

← →

Figure 10.9: Survey Preview for Object Insertion videos. The users are shown 3 videos along with source video and prompt used for editing, to answer questions about overall visual quality, text alignment and temporal consistency.

For object insertion, we ask users the following questions:

- Which video demonstrates the highest temporal consistency across new object appearance?

Choose the video with the BEST appearance consistency overtime for the new inserted object.

- Which video aligns most accurately with the provided text prompt?

Choose the method which BEST captures the details in the prompt (given on top of the video).

- Which video demonstrates the highest visual quality?

Choose a video which has the LEAST amount of extra artifacts (jitter, unwanted blobs), blurriness, unrealistic lighting and flickering.

Bibliography

- [1] Official github for c-gqa - issue 4. <https://github.com/ExplainableML/czsl/issues/4>, 2022. Accessed: 2021-11-22.
- [2] Official github for c-gqa - issue 3. <https://github.com/ExplainableML/czsl/issues/3>, 2022. Accessed: 2021-11-22.
- [3] Hero9 black 5k video 20mp photo streaming camera and bundles. <https://gopro.com/en/us/shop/cameras/hero9-black/CHDHX-901-master.html>, 2023. Accessed: 2023-03-03.
- [4] Midjourney v5. <https://www.midjourney.com/home/?callbackUrl=>Accessed: 2023-03-01.
- [5] Jean-Baptiste Alayrac, Josef Sivic, I. Laptev, and S. Lacoste-Julien. Joint discovery of object states and manipulation actions. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2146–2155, 2017.
- [6] Kate Arnold and Klaus Zuberbühler. Semantic combinations in primate calls. *Nature*, 441 (7091):303–303, 2006.
- [7] Y. Atzmon, F. Kreuk, U. Shalit, and Gal Chechik. A causal view of compositional zero-shot recognition. *ArXiv*, abs/2006.14610, 2020.
- [8] Nachwa Abou Bakr, Rémi Ronfard, and J. Crowley. Recognition and localization of food in cooking videos. In *CEA/MADiMa '18*, 2018.
- [9] Nachwa Abou Bakr, James L. Crowley, and Rémi Ronfard. Recognizing manipulation actions from state-transformations. *ArXiv*, abs/1906.05147, 2019.
- [10] Gedas Bertasius and Lorenzo Torresani. Cobe: Contextualized object embeddings from narrated instructional video. *ArXiv*, abs/2007.07306, 2020.
- [11] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *International Conference on Machine Learning*, 2021.
- [12] Irving Biederman. Recognition-by-components: a theory of human image understanding. *Psychological review*, 94 2:115–147, 1987.
- [13] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

- [14] Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18392–18402, 2023.
- [15] Mingdeng Cao, Xintao Wang, Zhongang Qi, Ying Shan, Xiaohu Qie, and Yinqiang Zheng. Masactrl: Tuning-free mutual self-attention control for consistent image synthesis and editing. *arXiv preprint arXiv:2304.08465*, 2023.
- [16] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.
- [17] João Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4724–4733, 2017.
- [18] Chao-Yeh Chen and Kristen Grauman. Inferring analogous attributes. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 200–207, 2014.
- [19] Hui Chen, Guiguang Ding, Xudong Liu, Zijia Lin, Ji Liu, and Jungong Han. Imram: Iterative matching with recurrent attention memory for cross-modal image-text retrieval. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12652–12660, 2020.
- [20] Hui Chen, Zhixiong Nan, Jingjing Jiang, and Nanning Zheng. Learning to infer unseen attribute-object compositions. *ArXiv*, abs/2010.14343, 2020. URL <https://api.semanticscholar.org/CorpusID:225076359>.
- [21] Weifeng Chen, Jie Wu, Pan Xie, Hefeng Wu, Jiashi Li, Xin Xia, Xuefeng Xiao, and Liang-Jin Lin. Control-a-video: Controllable text-to-video generation with diffusion models. *ArXiv*, abs/2305.13840, 2023. URL <https://api.semanticscholar.org/CorpusID:258841645>.
- [22] Xinlei Chen, Abhinav Shrivastava, and Abhinav Gupta. NEIL: Extracting Visual Knowledge from Web Data. In *IEEE International Conference on Computer Vision (ICCV)*, 2013.
- [23] Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco captions: Data collection and evaluation server. *ArXiv*, abs/1504.00325, 2015.
- [24] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8185–8194, 2020.
- [25] Noam Chomsky. A minimalist program for linguistic theory. In *A minimalist program for linguistic theory*, 1992. URL <https://api.semanticscholar.org/CorpusID:60834911>.

- [26] D. Damen, H. Doughty, G. Farinella, S. Fidler, Antonino Furnari, Evangelos Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, and Michael Wray. The epic-kitchens dataset: Collection, challenges and baselines. *IEEE transactions on pattern analysis and machine intelligence*, 2020.
- [27] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Scaling egocentric vision: The epic-kitchens dataset. *ArXiv*, abs/1804.02748, 2018.
- [28] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Scaling egocentric vision: The dataset. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, pages 753–771, Cham, 2018. Springer International Publishing. ISBN 978-3-030-01225-0.
- [29] Ahmad Darkhalil, Dandan Shan, Bin Zhu, Jian Ma, Amlan Kar, Richard E. L. Higgins, Sanja Fidler, David F. Fouhey, and Dima Damen. Epic-kitchens visor benchmark: Video segmentations and object relations. *ArXiv*, abs/2209.13064, 2022.
- [30] 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. Ieee, 2009.
- [31] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL <https://aclanthology.org/N19-1423>.
- [32] Victor C. Dibia. Neuralqa: A usable library for question answering (contextual query expansion + bert) on large datasets. In *EMNLP*, 2020.
- [33] Miranda Dixon-Luinenburg. What dall-e 2 can and cannot do. <https://www.lesswrong.com/posts/uKp6tBFStnsvrot5t/what-dall-e-2-can-and-cannot-do>, 2023. Accessed: 2023-03-08.
- [34] Yu Du, Fangyun Wei, Zihe Zhang, Miaoqing Shi, Yue Gao, and Guoqi Li. Learning to prompt for open-vocabulary object detection with vision-language model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14084–14093, 2022.
- [35] Dave Epstein, Jiajun Wu, Cordelia Schmid, and Chen Sun. Learning temporal dynamics from cycles in narrated video. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1460–1469, 2021.

- [36] Patrick Esser, Johnathan Chiu, Parmida Atighehchian, Jonathan Granskog, and Anastasis Germanidis. Structure and content-guided video synthesis with diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7346–7356, 2023.
- [37] Ali Farhadi, Ian Endres, Derek Hoiem, and David Alexander Forsyth. Describing objects by their attributes. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1778–1785, 2009.
- [38] Ali Farhadi, Ian Endres, and Derek Hoiem. Attribute-centric recognition for cross-category generalization. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2352–2359, 2010.
- [39] Alireza Fathi and James M Rehg. Modeling actions through state changes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2579–2586, 2013.
- [40] Christoph Feichtenhofer, A. Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1933–1941, 2016.
- [41] Rob Fergus, Yair Weiss, and Antonio Torralba. Semi-supervised learning in gigantic image collections. *Advances in neural information processing systems*, 22, 2009.
- [42] Vittorio Ferrari and Andrew Zisserman. Learning visual attributes. In *NIPS*, 2007.
- [43] A. Fire and S. Zhu. Learning perceptual causality from video. *ACM Trans. Intell. Syst. Technol.*, 7:23:1–23:22, 2015.
- [44] Jianlong Fu, Jinqiao Wang, Xin-Jing Wang, Yong Rui, and Hanqing Lu. What visual attributes characterize an object class? In *ACCV*, 2014.
- [45] Liane Gabora. The ‘power of then’: The uniquely human capacity to imagine beyond the present. *arXiv: Neurons and Cognition*, 2015.
- [46] Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit H Bermano, Gal Chechik, and Daniel Cohen-Or. An image is worth one word: Personalizing text-to-image generation using textual inversion. *arXiv preprint arXiv:2208.01618*, 2022.
- [47] Junyu Gao, Tianzhu Zhang, and Changsheng Xu. I know the relationships: Zero-shot action recognition via two-stream graph convolutional networks and knowledge graphs. In *AAAI*, 2019.
- [48] Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020.
- [49] Michal Geyer, Omer Bar-Tal, Shai Bagon, and Tali Dekel. Tokenflow: Consistent diffusion features for consistent video editing. *arXiv preprint arxiv:2307.10373*, 2023.

- [50] Golnaz Ghiasi, Xiuye Gu, Yin Cui, and Tsung-Yi Lin. Open-vocabulary image segmentation. *arXiv preprint arXiv:2112.12143*, 2021.
- [51] Pallabi Ghosh, Nirat Saini, Larry S. Davis, and Abhinav Shrivastava. All about knowledge graphs for actions. *ArXiv*, abs/2008.12432, 2020.
- [52] Rohit Girdhar, D. Ramanan, A. Gupta, Josef Sivic, and Bryan C. Russell. Actionvlad: Learning spatio-temporal aggregation for action classification. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3165–3174, 2017.
- [53] Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Q. Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, Miguel Martin, Tushar Nagarajan, Ilija Radosavovic, Santhosh K. Ramakrishnan, Fiona Ryan, Jayant Sharma, Michael Wray, Mengmeng Xu, Eric Z. Xu, Chen Zhao, Siddhant Bansal, Dhruv Batra, Vincent Cartillier, Sean Crane, Tien Do, Morrie Doulaty, Akshay Erapalli, Christoph Feichtenhofer, Adriano Fragomeni, Qichen Fu, Christian Fuegen, Abrahm Gebreselasie, Cristina González, James M. Hillis, Xuhua Huang, Yifei Huang, Wenqi Jia, Weslie Yu Heng Khoo, Jáchym Kolár, Satwik Kottur, Anurag Kumar, Federico Landini, Chao Li, Yanghao Li, Zhenqiang Li, Karttikeya Mangalam, Raghava Modhugu, Jonathan Munro, Tullie Murrell, Takumi Nishiyasu, Will Price, Paola Ruiz Puentes, Merey Ramazanova, Leda Sari, Kiran K. Somasundaram, Audrey Southerland, Yusuke Sugano, Ruijie Tao, Minh Vo, Yuchen Wang, Xindi Wu, Takuma Yagi, Yunyi Zhu, Pablo Arbeláez, David J. Crandall, Dima Damen, Giovanni Maria Farinella, Bernard Ghanem, Vamsi Krishna Ithapu, C. V. Jawahar, Hanbyul Joo, Kris Kitani, Haizhou Li, Richard A. Newcombe, Aude Oliva, Hyun Soo Park, James M. Rehg, Yoichi Sato, Jianbo Shi, Mike Zheng Shou, Antonio Torralba, Lorenzo Torresani, Mingfei Yan, and Jitendra Malik. Ego4d: Around the world in 3,000 hours of egocentric video. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18973–18990, 2021.
- [54] Xiuye Gu, Tsung-Yi Lin, Weicheng Kuo, and Yin Cui. Open-vocabulary object detection via vision and language knowledge distillation. *arXiv preprint arXiv:2104.13921*, 2021.
- [55] Yuchao Gu, Yipin Zhou, Bichen Wu, Licheng Yu, Jia-Wei Liu, Rui Zhao, Jay Zhangjie Wu, David Junhao Zhang, Mike Zheng Shou, and Kevin Tang. Videoswap: Customized video subject swapping with interactive semantic point correspondence. *ArXiv*, abs/2312.02087, 2023. URL <https://api.semanticscholar.org/CorpusID:265609343>.
- [56] Sergio Guadarrama, Erik Rodner, Kate Saenko, Ning Zhang, Ryan Farrell, Jeff Donahue, and Trevor Darrell. Open-vocabulary object retrieval. In *Robotics: science and systems*, volume 2, page 6, 2014.
- [57] Yuwei Guo, Ceyuan Yang, Anyi Rao, Yaohui Wang, Y. Qiao, Dahua Lin, and Bo Dai. Animatediff: Animate your personalized text-to-image diffusion models without specific tuning. *ArXiv*, abs/2307.04725, 2023. URL <https://api.semanticscholar.org/CorpusID:259501509>.

- [58] A. Gupta, Aniruddha Kembhavi, and L. Davis. Observing human-object interactions: Using spatial and functional compatibility for recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31:1775–1789, 2009.
- [59] Zongyan Han, Zhenyong Fu, Shuo Chen, and Jian Yang. Contrastive embedding for generalized zero-shot learning. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2371–2381, 2021.
- [60] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Learning spatio-temporal features with 3d residual networks for action recognition. *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 3154–3160, 2017.
- [61] Bo He, Xitong Yang, Zuxuan Wu, Hao Chen, Ser-Nam Lim, and Abhinav Shrivastava. Gta: Global temporal attention for video action understanding, 2020.
- [62] Bo He, Xitong Yang, Le Kang, Zhiyu Cheng, Xin Zhou, and Abhinav Shrivastava. Asm-loc: Action-aware segment modeling for weakly-supervised temporal action localization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- [63] Bo He, Jun Wang, Jieli Qiu, Trung Bui, Abhinav Shrivastava, and Zhaowen Wang. Align and attend: Multimodal summarization with dual contrastive losses. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14867–14878, 2023.
- [64] Bo He, Xitong Yang, Hanyu Wang, Zuxuan Wu, Hao Chen, Shuaiyi Huang, Yixuan Ren, Ser-Nam Lim, and Abhinav Shrivastava. Towards scalable neural representation for diverse videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.
- [65] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.
- [66] Yuanfeng He, Yuanxi Li, Jiajia Lei, and Clement H. C. Leung. A framework of query expansion for image retrieval based on knowledge base and concept similarity. *Neurocomputing*, 204:26–32, 2016.
- [67] Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control. *arXiv preprint arXiv:2208.01626*, 2022.
- [68] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.
- [69] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

- [70] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.
- [71] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [72] Donald D. Hoffman and Whitman Richards. Parts of recognition. *Cognition*, 18:65–96, 1984.
- [73] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.
- [74] Noureldien Hussein, Efstratios Gavves, and Arnold W. M. Smeulders. Videograph: Recognizing minutes-long human activities in videos. *ArXiv*, abs/1905.05143, 2019.
- [75] Sung Ju Hwang, Fei Sha, and Kristen Grauman. Sharing features between objects and their attributes. *CVPR 2011*, pages 1761–1768, 2011.
- [76] Phillip Isola, Joseph J. Lim, and E. Adelson. Discovering states and transformations in image collections. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1383–1391, 2015.
- [77] Dinesh Jayaraman, Frederik Ebert, Alexei A. Efros, and Sergey Levine. Time-agnostic prediction: Predicting predictable video frames. *ArXiv*, abs/1808.07784, 2018.
- [78] A. Karpathy, G. Toderici, Sanketh Shetty, T. Leung, R. Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.
- [79] Shyamgopal Karthik, Massimiliano Mancini, and Zeynep Akata. Kg-sp: Knowledge guided simple primitives for open world compositional zero-shot learning. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9326–9335, 2022.
- [80] Hirokatsu Kataoka, Tenga Wakamiya, Kensho Hara, and Yutaka Satoh. Would mega-scale datasets further enhance spatiotemporal 3d cnns? *arXiv preprint arXiv:2004.04968*, 2020.
- [81] Levon Khachatryan, Andranik Movsisyan, Vahram Tadevosyan, Roberto Henschel, Zhangyang Wang, Shant Navasardyan, and Humphrey Shi. Text2video-zero: Text-to-image diffusion models are zero-shot video generators. *arXiv preprint arXiv:2303.13439*, 2023.
- [82] F. M. Khan, V. Singh, and R. Nevatia. Simultaneous inference of activity, pose and object. *2012 IEEE Workshop on the Applications of Computer Vision (WACV)*, pages 281–288, 2012.
- [83] Tae Soo Kim and Gregory Hager. Safcar: Structured attention fusion for compositional action recognition. *ArXiv*, abs/2012.02109, 2020.

- [84] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023.
- [85] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123:32–73, 2016.
- [86] H. Kuehne, A. B. Arslan, and T. Serre. The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *Proceedings of Computer Vision and Pattern Recognition Conference (CVPR)*, 2014.
- [87] Girish Kulkarni, Visruth Premraj, Vicente Ordonez, Sagnik Dhar, Siming Li, Yejin Choi, Alexander C. Berg, and Tamara L. Berg. Babytalk: Understanding and generating simple image descriptions. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35:2891–2903, 2013.
- [88] Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and brain sciences*, 40:e253, 2017.
- [89] Christoph H. Lampert, Hannes Nickisch, and Stefan Harmeling. Learning to detect unseen object classes by between-class attribute transfer. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 951–958, 2009.
- [90] Christoph H. Lampert, Hannes Nickisch, and Stefan Harmeling. Attribute-based classification for zero-shot visual object categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36:453–465, 2014.
- [91] Hao-Wei Lee, Chia-Po Wei, and Yu-Chiang Frank Wang. Learning grassmann manifolds for object state discovery. *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1223–1227, 2017.
- [92] Kuang-Huei Lee, Xi Chen, Gang Hua, Houdong Hu, and Xiaodong He. Stacked cross attention for image-text matching. *ArXiv*, abs/1803.08024, 2018.
- [93] Junnan Li, Dongxu Li, Caiming Xiong, and Steven C. H. Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International Conference on Machine Learning*, 2022. URL <https://api.semanticscholar.org/CorpusID:246411402>.
- [94] Kunpeng Li, Yulun Zhang, K. Li, Yuanyuan Li, and Yun Raymond Fu. Visual semantic reasoning for image-text matching. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4653–4661, 2019.
- [95] Yong-Lu Li, Yue Xu, Xiaohan Mao, and Cewu Lu. Symmetry and group in attribute-object compositions. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11313–11322, 2020.

- [96] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7082–7092, 2019.
- [97] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *Neural Information Processing Systems*, 2023.
- [98] Shaoteng Liu, Yuechen Zhang, Wenbo Li, Zhe Lin, and Jiaya Jia. Video-p2p: Video editing with cross-attention control. *arXiv preprint arXiv:2303.04761*, 2023.
- [99] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023.
- [100] Y. Liu, P. Wei, and S. Zhu. Jointly recognizing object fluents and tasks in egocentric videos. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2943–2951, 2017.
- [101] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.
- [102] Cewu Lu, Ranjay Krishna, Michael S. Bernstein, and Li Fei-Fei. Visual relationship detection with language priors. In *ECCV*, 2016.
- [103] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11461–11471, 2022.
- [104] Jian Ma and Dima Damen. Hand-object interaction reasoning. *2022 18th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–8, 2022.
- [105] Dhruv Kumar Mahajan, Sundararajan Sellamanickam, and Vinod Nair. A joint learning framework for attribute models and object descriptions. *2011 International Conference on Computer Vision*, pages 1227–1234, 2011.
- [106] Massimiliano Mancini, Muhammad Ferjad Naeem, Yongqin Xian, and Zeynep Akata. Learning graph embeddings for open world compositional zero-shot learning. *ArXiv*, abs/2105.01017, 2021.
- [107] Joanna Materzynska, Tete Xiao, Roei Herzig, Huijuan Xu, Xiaolong Wang, and Trevor Darrell. Something-else: Compositional action recognition with spatial-temporal interaction networks. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2020. doi: 10.1109/cvpr42600.2020.00113. URL <http://dx.doi.org/10.1109/cvpr42600.2020.00113>.
- [108] T. McCandless and K. Grauman. Object-centric spatio-temporal pyramids for egocentric activity recognition. In *BMVC*, 2013.

- [109] Luke Melas-Kyriazi, Christian Rupprecht, Iro Laina, and Andrea Vedaldi. Realfusion: 360 reconstruction of any object from a single image. In *CVPR*, 2023.
- [110] Moustafa Meshry, Yixuan Ren, Larry S Davis, and Abhinav Shrivastava. Step: Style-based encoder pre-training for multi-modal image synthesis. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.
- [111] Antoine Miech, Jean-Baptiste Alayrac, Lucas Smaira, Ivan Laptev, Josef Sivic, and Andrew Zisserman. End-to-end learning of visual representations from uncurated instructional videos. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9876–9886, 2019.
- [112] Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. Howto100m: Learning a text-video embedding by watching hundred million narrated video clips. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2630–2640, 2019.
- [113] Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *ICLR*, 2013.
- [114] I. Misra, A. Gupta, and M. Hebert. From red wine to red tomato: Composition with context. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1160–1169, 2017.
- [115] Ishan Misra*, Abhinav Shrivastava*, Abhinav Gupta, and Martial Hebert. Cross-stitch Networks for Multi-task Learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [116] Erik T. Mueller. Commonsense reasoning: An event calculus based approach. In *Commonsense Reasoning*, 2006. URL <https://api.semanticscholar.org/CorpusID:267791712>.
- [117] Rafael Müller, Simon Kornblith, and Geoffrey E. Hinton. When does label smoothing help? In *NeurIPS*, 2019.
- [118] Muhammad Ferjad Naeem, Yongqin Xian, Federico Tombari, and Zeynep Akata. Learning graph embeddings for compositional zero-shot learning. *ArXiv*, abs/2102.01987, 2021.
- [119] Tushar Nagarajan and K. Grauman. Attributes as operators: Factorizing unseen attribute-object compositions. In *ECCV*, 2018.
- [120] Tushar Nagarajan, Yanghao Li, Christoph Feichtenhofer, and Kristen Grauman. Ego-topo: Environment affordances from egocentric video. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 160–169, 2020.
- [121] Z. Nan, Y. Liu, N. Zheng, and S. Zhu. Recognizing unseen attribute-object pair with generative model. In *AAAI*, 2019.

- [122] Sanath Narayan, Akshita Gupta, Fahad Shahbaz Khan, Cees GM Snoek, and Ling Shao. Latent embedding feedback and discriminative features for zero-shot classification. In *European Conference on Computer Vision*, pages 479–495. Springer, 2020.
- [123] Sangmin Oh, Anthony J. Hoogs, A. G. Amitha Perera, Naresh P. Cuntoor, Chia-Chih Chen, Jong Taek Lee, Saurajit Mukherjee, Jake K. Aggarwal, Hyungtae Lee, Larry S. Davis, Eran Swears, Xiaoyang Wang, Qiang Ji, Kishore K. Reddy, Mubarak Shah, Carl Vondrick, Hamed Pirsiavash, Deva Ramanan, Jenny Yuen, Antonio Torralba, Bi Song, Anesco Fong, Amit K. Roy-Chowdhury, and Mita Desai. A large-scale benchmark dataset for event recognition in surveillance video. *CVPR 2011*, pages 3153–3160, 2011. URL <https://api.semanticscholar.org/CorpusID:263882069>.
- [124] Vicente Ordonez, Girish Kulkarni, and Tamara L. Berg. Im2text: Describing images using 1 million captioned photographs. In *NIPS*, 2011.
- [125] Devi Parikh and Kristen Grauman. Relative attributes. *2011 International Conference on Computer Vision*, pages 503–510, 2011.
- [126] Genevieve Patterson and James Hays. Sun attribute database: Discovering, annotating, and recognizing scene attributes. *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2751–2758, 2012.
- [127] Genevieve Patterson and James Hays. Coco attributes: Attributes for people, animals, and objects. In *ECCV*, 2016.
- [128] Joel Pearson. The human imagination: the cognitive neuroscience of visual mental imagery. *Nature Reviews Neuroscience*, pages 1–11, 2019.
- [129] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *EMNLP*, 2014.
- [130] Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus H. Gross, and Alexander Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 724–732, 2016. URL <https://api.semanticscholar.org/CorpusID:1949934>.
- [131] Khoi Pham, Kushal Kafle, Zhe Lin, Zhi Ding, Scott D. Cohen, Quan Tran, and Abhinav Shrivastava. Learning to predict visual attributes in the wild. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13013–13023, 2021.
- [132] Khoi Pham, Kushal Kafle, Zhe Lin, Zhihong Ding, Scott Cohen, Quan Tran, and Abhinav Shrivastava. Improving closed and open-vocabulary attribute prediction using transformers. In *European Conference on Computer Vision*, pages 201–219. Springer, 2022.
- [133] Koutilya PNVR, Bharat Singh, Pallabi Ghosh, Behjat Siddiquie, and David Jacobs. Ld-znet: A latent diffusion approach for text-based image segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4157–4168, October 2023.

- [134] Jordi Pont-Tuset, Jasper Uijlings, Soravit Changpinyo, Radu Soricut, and Vittorio Ferrari. Connecting vision and language with localized narratives. In *ECCV*, 2020.
- [135] Senthil Purushwalkam, Maximilian Nickel, Abhinav Gupta, and Marc’Aurelio Ranzato. Task-driven modular networks for zero-shot compositional learning. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3592–3601, 2019.
- [136] Chenyang Qi, Xiaodong Cun, Yong Zhang, Chenyang Lei, Xintao Wang, Ying Shan, and Qifeng Chen. Fatezero: Fusing attentions for zero-shot text-based video editing. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15886–15896, 2023. URL <https://api.semanticscholar.org/CorpusID:257557738>.
- [137] Gorjan Radevski, Marie-Francine Moens, and Tinne Tuytelaars. Revisiting spatio-temporal layouts for compositional action recognition. *British Machine Vision Conference (BMVC)*, abs/2111.01936, 2021.
- [138] 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.
- [139] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. *CoRR*, abs/2102.12092, 2021. URL <https://arxiv.org/abs/2102.12092>.
- [140] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.
- [141] Shaoqing Ren, Kaiming He, Ross B. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39:1137–1149, 2015.
- [142] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.
- [143] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III* 18, pages 234–241. Springer, 2015.
- [144] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22500–22510, 2023.

- [145] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S Sara Mahdavi, Rapha Gontijo Lopes, et al. Photorealistic text-to-image diffusion models with deep language understanding. *arXiv preprint arXiv:2205.11487*, 2022.
- [146] Nirat Saini, Bo He, Gaurav Shrivastava, Sai Saketh Rambhatla, and Abhinav Shrivastava. Recognizing actions using object states. In *ICLR2022 Workshop on the Elements of Reasoning: Objects, Structure and Causality*, 2022. URL <https://openreview.net/forum?id=HBgzIHOUqqg>.
- [147] Nirat Saini, Khoi Pham, and Abhinav Shrivastava. Disentangling visual embeddings for attributes and objects. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13648–13657, 2022.
- [148] Nirat Saini, Hanyu Wang, Archana Swaminathan, Vinoj Jayasundara, Bo He, Kamal Gupta, and Abhinav Shrivastava. Chop & learn: Recognizing and generating object-state compositions. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 20190–20201, 2023. URL <https://api.semanticscholar.org/CorpusID:262824771>.
- [149] Nirat Saini, Navaneeth Bodla, Ashish Shrivastava, Avinash Ravichandran, Xiao Zhang, Abhinav Shrivastava, and Bharat Singh. Invi: Object insertion in videos using off-the-shelf diffusion models, 2024. URL <https://arxiv.org/abs/2407.10958>.
- [150] Nirat Saini, Khoi Pham, and Abhinav Shrivastava. Beyond seen primitive concepts and attribute-object compositional learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14466–14476, June 2024.
- [151] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35:25278–25294, 2022.
- [152] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.
- [153] Amir Shahroudy, Jun Liu, Tian-Tsong Ng, and G. Wang. Ntu rgb+d: A large scale dataset for 3d human activity analysis. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1010–1019, 2016.
- [154] Kwan-Yee K. Wong Shaozhe Hao, Kai Han. Learning attention as disentangler for compositional zero-shot learning. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.
- [155] L. Shen, Serena Yeung, Judy Hoffman, G. Mori, and Li Fei-Fei. Scaling human-object interaction recognition through zero-shot learning. *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1568–1576, 2018.

- [156] Hengcan Shi, Munawar Hayat, Yicheng Wu, and Jianfei Cai. Proposalclip: Unsupervised open-category object proposal generation via exploiting clip cues. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9611–9620, 2022.
- [157] Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Constrained semi-supervised learning using attributes and comparative attributes. In *European Conference on Computer Vision*, 2012.
- [158] Abhinav Shrivastava, Saurabh Singh, and Abhinav Kumar Gupta. Constrained semi-supervised learning using attributes and comparative attributes. In *ECCV*, 2012.
- [159] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images through adversarial training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2017.
- [160] K. Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014.
- [161] Tom’avs Souvcek, Jean-Baptiste Alayrac, Antoine Miech, Ivan Laptev, and Josef Sivic. Look for the change: Learning object states and state-modifying actions from untrimmed web videos. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13936–13946, 2022.
- [162] Tom’avs Souvcek, Jean-Baptiste Alayrac, Antoine Miech, Ivan Laptev, and Josef Sivic. Multi-task learning of object state changes from uncurated videos. *ArXiv*, abs/2211.13500, 2022.
- [163] Sebastian Stein and Stephen J. McKenna. Combining embedded accelerometers with computer vision for recognizing food preparation activities. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, page 729–738, 2013.
- [164] Roman Suvorov, Elizaveta Logacheva, Anton Mashikhin, Anastasia Remizova, Arsenii Ashukha, Aleksei Silvestrov, Naejin Kong, Harshith Goka, Kiwoong Park, and Victor Lempitsky. Resolution-robust large mask inpainting with fourier convolutions. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 2149–2159, 2022.
- [165] Hao Tan and Mohit Bansal. Lxmert: Learning cross-modality encoder representations from transformers. *arXiv preprint arXiv:1908.07490*, 2019.
- [166] Haoliang Tan, L. Wang, Q. Zhang, Zhanning Gao, Nanning Zheng, and Gang Hua. Object affordances graph network for action recognition. In *BMVC*, 2019.
- [167] Yansong Tang, Dajun Ding, Yongming Rao, Yu Zheng, Danyang Zhang, Lili Zhao, Jiwen Lu, and Jie Zhou. Coin: A large-scale dataset for comprehensive instructional video analysis. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1207–1216, 2019.

- [168] Narek Tumanyan, Michal Geyer, Shai Bagon, and Tali Dekel. Plug-and-play diffusion features for text-driven image-to-image translation. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1921–1930, 2022. URL <https://api.semanticscholar.org/CorpusID:253801961>.
- [169] Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Neural Information Processing Systems*, 2017. URL <https://api.semanticscholar.org/CorpusID:13756489>.
- [170] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge J. Belongie. The caltech-ucsd birds-200-2011 dataset. In *Caltech-UCSD Birds 200*, 2011. URL <https://api.semanticscholar.org/CorpusID:16119123>.
- [171] Jiang Wang, Xiaohan Nie, Yin Xia, Ying Wu, and Song-Chun Zhu. Cross-view action modeling, learning, and recognition. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2649–2656, 2014.
- [172] L. Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, D. Lin, X. Tang, and L. Gool. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, 2016.
- [173] Qingsheng Wang, Lingqiao Liu, Chenchen Jing, Hao Chen, Guoqiang Liang, Peng Wang, and Chunhua Shen. Learning conditional attributes for compositional zero-shot learning. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11197–11206, 2023. URL <https://api.semanticscholar.org/CorpusID:258960120>.
- [174] X. Wang and Abhinav Kumar Gupta. Videos as space-time region graphs. *ArXiv*, abs/1806.01810, 2018.
- [175] X. Wang, Ross B. Girshick, A. Gupta, and Kaiming He. Non-local neural networks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7794–7803, 2018.
- [176] X. Wang, Yufei Ye, and Abhinav Gupta. Zero-shot recognition via semantic embeddings and knowledge graphs. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6857–6866, 2018.
- [177] Xiang Wang, Hangjie Yuan, Shiwei Zhang, Dayou Chen, Jiuniu Wang, Yingya Zhang, Yujun Shen, Deli Zhao, and Jingren Zhou. Videocomposer: Compositional video synthesis with motion controllability. *arXiv preprint arXiv:2306.02018*, 2023.
- [178] Xiaolong Wang, Ali Farhadi, and Abhinav Gupta. Actions ~ transformations. In *CVPR*, 2016.
- [179] Xin Wang, Fisher Yu, Trevor Darrell, and Joseph E. Gonzalez. Task-aware feature generation for zero-shot compositional learning, 2019.

- [180] Donglai Wei, Joseph J. Lim, Andrew Zisserman, and William T. Freeman. Learning and using the arrow of time. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.
- [181] Kun-Juan Wei, Muli Yang, H. Wang, Cheng Deng, and Xianglong Liu. Adversarial fine-grained composition learning for unseen attribute-object recognition. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3740–3748, 2019.
- [182] Jay Zhangjie Wu, Yixiao Ge, Xintao Wang, Stan Weixian Lei, Yuchao Gu, Yufei Shi, Wynne Hsu, Ying Shan, Xiaohu Qie, and Mike Zheng Shou. Tune-a-video: One-shot tuning of image diffusion models for text-to-video generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7623–7633, 2023.
- [183] Jianchao Wu, L. Wang, Jie Guo, and Gangshan Wu. Learning actor relation graphs for group activity recognition. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9956–9966, 2019.
- [184] Tete Xiao, Quanfu Fan, Dan Gutfreund, Mathew Monfort, A. Oliva, and B. Zhou. Reasoning about human-object interactions through dual attention networks. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3918–3927, 2019.
- [185] Hongtao Xie, Yongdong Zhang, Jianlong Tan, Li Guo, and Jintao Li. Contextual query expansion for image retrieval. *IEEE Transactions on Multimedia*, 16:1104–1114, 2014.
- [186] Guangyue Xu, Parisa Kordjamshidi, and Joyce Yue Chai. Zero-shot compositional concept learning. *ArXiv*, abs/2107.05176, 2021.
- [187] Ziwei Xu, Guangzhi Wang, Yongkang Wong, and Mohan S. Kankanhalli. Relation-aware compositional zero-shot learning for attribute-object pair recognition. *ArXiv*, abs/2108.04603, 2021.
- [188] Rui Yan, Peng Huang, Xiangbo Shu, Junhao Zhang, Yonghua Pan, and Jinhui Tang. Look less think more: Rethinking compositional action recognition. *Proceedings of the 30th ACM International Conference on Multimedia*, 2022.
- [189] Ceyuan Yang, Yinghao Xu, Jianping Shi, Bo Dai, and B. Zhou. Temporal pyramid network for action recognition. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 588–597, 2020.
- [190] Muli Yang, C. Deng, Junchi Yan, Xianglong Liu, and Dacheng Tao. Learning unseen concepts via hierarchical decomposition and composition. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10245–10253, 2020.
- [191] Muli Yang, Cheng Deng, Junchi Yan, Xianglong Liu, and Dacheng Tao. Learning unseen concepts via hierarchical decomposition and composition. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10245–10253, 2020.
- [192] Shuai Yang, Yifan Zhou, Ziwei Liu, and Chen Change Loy. Rerender a video: Zero-shot text-guided video-to-video translation. *arXiv preprint arXiv:2306.07954*, 2023.

- [193] Keren Ye, Mingda Zhang, Wei Li, Danfeng Qin, Adriana Kovashka, and Jesse Berent. Learning to discover and localize visual objects with open vocabulary. *ArXiv*, abs/1811.10080, 2018.
- [194] Peter Young, Alice Lai, Micah Hodosh, and J. Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78, 2014.
- [195] A. Yu and K. Grauman. Fine-grained visual comparisons with local learning. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 192–199, 2014.
- [196] Li Yuan, Francis E. H. Tay, Guilin Li, Tao Wang, and Jiashi Feng. Revisiting knowledge distillation via label smoothing regularization. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3902–3910, 2020.
- [197] Tian Yun, Usha Bhalla, Elizabeth-Jane Pavlick, and Chen Sun. Do vision-language pre-trained models learn primitive concepts? *ArXiv*, abs/2203.17271, 2022.
- [198] Alireza Zareian, Kevin Dela Rosa, Derek Hao Hu, and Shih-Fu Chang. Open-vocabulary object detection using captions. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14388–14397, 2021.
- [199] Matthew D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014.
- [200] Dagmar Zeithamova, Michael L. Mack, Kurt Braunlich, Tyler Davis, Carol A. Seger, Marlieke T.R. van Kesteren, and Andreas Wutz. Brain mechanisms of concept learning. *Journal of Neuroscience*, 39(42):8259–8266, 2019. ISSN 0270-6474. doi: 10.1523/JNEUROSCI.1166-19.2019. URL <https://www.jneurosci.org/content/39/42/8259>.
- [201] Chen-Lin Zhang, Jian Zhai Wu, and Yin Li. Actionformer: Localizing moments of actions with transformers. In *European Conference on Computer Vision*, 2022.
- [202] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3813–3824, 2023. URL <https://api.semanticscholar.org/CorpusID:256827727>.
- [203] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.
- [204] Hang Zhao, Xavier Puig, Bolei Zhou, Sanja Fidler, and Antonio Torralba. Open vocabulary scene parsing. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2002–2010, 2017.

- [205] Bolei Zhou, Alex Andonian, Aude Oliva, and Antonio Torralba. Temporal relational reasoning in videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 803–818, 2018.
- [206] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In *NIPS*, 2003.
- [207] Luowei Zhou, Nathan Louis, and Jason J. Corso. Weakly-supervised video object grounding from text by loss weighting and object interaction. In *British Machine Vision Conference*, 2018.
- [208] Yipin Zhou and T. Berg. Learning temporal transformations from time-lapse videos. *ArXiv*, abs/1608.07724, 2016.
- [209] Dimitri Zhukov, Jean-Baptiste Alayrac, Ramazan Gokberk Cinbis, David Fouhey, Ivan Laptev, and Josef Sivic. Cross-task weakly supervised learning from instructional videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3537–3545, 2019.