ABSTRACT

Title of Dissertation: FROM PARTS TO WHOLE IN ACTION AND OBJECT UNDERSTANDING.
Chinmaya Devaraj
Doctor of Philosophy, 2024

Dissertation Directed by: Professor Yiannis Aloimonos and Dr. Cornelia Fermüller.
Department of Electrical and Computer Engineering

The traditional paradigm of supervised learning in action or object recognition often relies on a top-down approach, ignoring explicit modeling of what activity or objects consist of. Recent approaches in generative AI research have shown us the ability to generate images and videos using text, indirectly indicating that we have control over the constituents of images and videos. In this dissertation, we explore ways to use the constituents of actions to develop methods to improve understanding of action. We devise different approaches to utilize the parts of actions, namely object motion, object state changes, and motion descriptions obtained by LLMs in various tasks like in the next active object segmentation, zero-shot action recognition, or video-text retrieval. We show promising benefits in action anticipation, zero-shot action recognition, and text-video retrieval tasks, demonstrating the practical applications of our methods.

In the first part of the dissertation, we explore the idea of using the constituents of actions in GCNs for zero-shot human-object action recognition. The main idea is that semantically similar actions (of similar constituents) are closer in feature space. Thus, in our graph, we encode the

edges connecting those actions with higher similarity. We introduce a method to visually ground the external knowledge graph using the concept of shared similarity between similar actions. We evaluate the method on the EPIC Kitchens dataset and the Charades dataset showing impressive results over baseline methods. We further show that visually grounding the knowledge graph enhances the performance of GCNs when an adversarial attack corrupts the input graph.

In the second part of the thesis, we extend our ideas on human-object interactions in first-person videos. Human actions involving hand manipulations are structured according to the making and breaking of hand-object contact, and human visual understanding of action relies on anticipation of contact, as demonstrated by pioneering work in cognitive science. Taking inspiration from this, we introduce representations and models centered on contact, which we then use in action prediction and anticipation. We train the Anticipation Module, a module producing Contact Anticipation Maps and Next Active Object Segmentations - novel low-level representations providing temporal and spatial characteristics of anticipated near future action. On top of the Anticipation Module, we apply Egocentric Object Manipulation Graphs (Ego-OMG), a framework for action anticipation and prediction. Using the Anticipation Module to aid Ego-OMG produces state-of-the-art results, achieving first and second places on the unseen and seen test sets of the EPIC Kitchens Action Anticipation Challenge and achieving state-of-the-art results on action anticipation and action prediction over EPIC Kitchens.

In the same line of thinking of constituents of action, we next focus on investigating how motion understanding can be modeled in current video-text models. We introduce motion descriptions generated by GPT4 on three action datasets that capture fine-grained motion descriptions of activities. We evaluated several video-text models on the task of retrieval of motion descriptions and found them to need to catch up to the human expert performance. We introduce a method

of improving motion understanding in video-text models by utilizing motion descriptions. This method is demonstrated on two action datasets for the motion description retrieval task. The results draw attention to the need for quality captions involving fine-grained motion information in existing datasets and demonstrate the effectiveness of the proposed pipeline in understanding fine-grained motion during video-text retrieval.

FROM PARTS TO WHOLE IN ACTION AND OBJECT UNDERSTANDING

by

Chinmaya Devaraj

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 Yiannis Aloimonos, Chair/Advisor
Dr. Cornelia Fermüller, Co-Advisor
Professor Behtash Babadi
Professor Dinesh Manocha
Professor Nikhil Chopra, Dean's representative.

Dedication

To past, present, and future scientists.

Acknowledgments

I sincerely thank my advisors, Prof. Yiannis Aloimonos and Dr. Cornelia Fermüller. Despite my non-computer science background, they offered me invaluable support during my PhD journey. I feel indebted to Prof Yiannis Aloimonos, who gave me the complete freedom to work on any research problems of interest. This paved the way for me to work on exciting problems with graph convolutional networks and vision-language models. Despite innumerable failures in research ideas during the initial part of my PhD, he supported me wholeheartedly and gave me the utmost confidence to pursue challenging and novel research ideas. I am grateful to my co-advisor, Dr Cornelia Fermüller, who helped me develop critical thinking, writing, and presentation skills through countless brainstorming sessions during our research meetings. She has been my pillar of support for any unexpected results or challenges during the PhD. I could not have asked for better advisors who have supported me during this journey.

At the same time, I would like to thank my dissertation committee members, Prof. Behtash Babadi, Prof. Dinesh Manocha, and Prof. Nikhil Chopra, for their feedback, which helped improve my dissertation.

I especially want to thank my co-authors, Chengxi Ye, Eadom Dessalene, Michael Maynord, who played a pivotal role in my PhD journey. Chengxi has been there throughout my journey, acting as a mentor whenever I needed another perspective. Eadom and Michael's insightful discussions on video understanding, zero-shot learning, and contrastive learning helped me explore

new frontiers and discover novel ways to solve problems. I would also like to thank my other research peers, Matthew Evanusa, Peter Sutor, Chethan Mysore Parameshwara, Konstantinos Zampogiannis, Aleksandrs Ecins, Nitin Sanket, Chahat Deep Singh, Snehesh Shrestha, Xiaomin Lin, Anton Mitrokhin, Aman Virmani, Thilak Mohan, Rajeev Ranjan, Amit Kumar, Sayantan Sarkar, who have shaped me to be a better researcher.

I am genuinely grateful to my colleagues Alberto Santamaria-Pang, James R. Kubricht, Peter Tu, and Ashish Tawari during my GE Research and Honda Research Institute internships. I am also grateful to the ECE department, ISR, and UMIACS staff, who have supported me throughout this journey.

I would like to acknowledge financial support through the teaching assistantship from the ECE department, Research assistantship from the National Science Foundation(NSF), Amazon, and NSF Neuropac Fellowship.

I am beyond grateful to my mother, Chandrika, and father, Devaraj, who have supported me during this journey and made countless sacrifices for me to come all the way. I am also grateful to my friends Bhushan, Sushma, Madhu, Shankar, Gowri, Harsha, Manish, Pradhnya, Kailash, Chinmay Nanal, Karthik Mohan, Nealofar, Abhinav Kannan, Akshat Shah, Sai Pavan and Deepali Bhola for creating beautiful memories while I was staying in College Park. Your love, care, and memorable times spent with me together made my PhD journey extremely smooth.

I would like to acknowledge other professors I have taken classes with during this PhD or interacted with. Most importantly, I would like to acknowledge the support of Gurudev Sri Sri Ravi Shankar, without whose inspiration I would not have been able to complete this PhD.

It is impossible to remember all, and I apologize to those I've inadvertently left out.

Table of Contents

Bibliography 98

List of Tables

List of Figures

Chapter 1: Introduction

1.1 Motivation and scope

Recent innovations in diffusion models have shown us that we can generate images and videos by giving a text prompt, giving us control over generating what we want. For example, for the prompt, photo of a person sitting on a chair eating a banana in front of the house, figure 1.1 shows the image generated using Adobe Firefly. Here, in figure 1.1, the constituents of the image are given by the prompt. In figure 1.1, a person, chair, house, and the relation between them and the associated verbs constitute that image. A generative model that can generate all the necessary constituents/parts of the image is better than others that can't.

On the other hand, a discriminative model is helpful to understand the contents of the image. If a discriminative model can correctly produce the contents of the image, it has mastered the analysis process. Typically, this is approached by learning classifiers that will predict all its constituents.

In this dissertation, we restrict our focus mainly to action videos and learn to improve discriminative models to understand the action. Our ability to understand actions depends on how we perceive actions. A deep neural network that classifies actions solely using action labels maps the input to the action class without explicit attention to attend to different constituents. This is approached using convolutional neural networks (CNNs) or a transformer-based architecture.

Figure 1.1: Image generated from Adobe Firefly from the prompt person sitting on a chair eating a banana in front of the house.

Some popular approaches in action recognition use two streams, one for RGB and the other for optical flow, as in [2–5]. Recent approaches [6–8] further improve spatial temporal learning. While these are great approaches for action classification, when it comes to using them for challenging problems like zero-shot action recognition and action prediction, we require approaches with more understanding of the action. Zero-shot action recognition requires us to recognize actions not seen during training. A zero-shot action recognition system requires understanding how all the actions are related to each other so that even when a particular action is not seen during training, we can recognize it because we have understood how every action is related. Action predictions involve predicting what will happen, which goes beyond understanding what is already seen. These challenging tasks require more sophisticated approaches that go beyond learning action classifiers.

In this dissertation, we explore whether modeling and using actions' constituents help overall understanding of action. Constituents of actions is a generic symbolic term used to refer to high-level information and can refer to the intention of action, tracking of objects and hands involved, object motion, how the object's states are transformed during the action, etc. These constituents of actions, when used effectively, can be useful for solving challenging problems that need more reasoning, like in the case of zero-shot action recognition. We approach using constituent of actions in primarily two ways. We sometimes learn representations of these constituents of actions and then use these representations as intermediary signals to solve the final problem. This could involve learning the next active object being interacted with, tracking hands, or segmenting the next active object to solve the final problem of predicting future action. Another approach is using these constituent of actions as supervisory signals to perform the final task better. In this dissertation, we show work in both approaches.

1.2 Dissertation Contributions and Outline

I briefly summarize my contributions and outline of this dissertation.

1.2.1 Chapter 2

In this chapter, we explore the idea of using the constituents of actions in GCNs for zero-shot human-object action recognition. The main idea is that semantically similar actions (of similar constituents) are closer in feature space. Thus, in our graph, we encode the edges connecting those actions with higher similarity. GCN-based zero-shot learning approaches commonly use fixed input graphs representing external knowledge, usually from language. However, such input graphs fail to incorporate the visual domain nuances. We introduce a method to ground the external knowledge graph visually. The method is demonstrated on a novel concept of grouping actions according to the shared notions of object motion and object state changes which are the constituents of action. Object motion and object state changes are not calculated but indirectly used to improve the zero-shot learning. Our method has been shown to perform superiorly in zero-shot action recognition on two challenging human manipulation action datasets, the EPIC Kitchens dataset, and the Charades dataset. We further show that visually grounding the knowledge graph enhances the performance of GCNs when an adversarial attack corrupts the input graph. This chapter is based on my published work [9].

1.2.2 Chapter 3

In the this part of dissertation, we extend our ideas on human-object interactions in first-person videos. Human actions involving hand manipulations are structured according to the

making and breaking of hand-object contact, and human visual understanding of action is reliant on anticipation of contact as is demonstrated by pioneering work in cognitive science. Taking inspiration from this, we introduce representations and models centered on contact, which we then use in action prediction and anticipation. We annotate a subset of the EPIC Kitchens dataset to include time-to-contact between hands and objects, as well as segmentations of hands and objects. Using these annotations we train the Anticipation Module, a module producing Contact Anticipation Maps and Next Active Object Segmentations - novel low-level representations providing temporal and spatial characteristics of anticipated near future action. On top of the Anticipation Module we apply Egocentric Object Manipulation Graphs (Ego-OMG), a framework for action anticipation and prediction. Ego-OMG models longer term temporal semantic relations through the use of a graph modeling transitions between contact delineated action states. Use of the Anticipation Module within Ego-OMGproduces state-of-the-art results, achieving 1st and 2 places on the unseen and seen test sets, respectively, of the EPIC Kitchens Action Anticipation Challenge, and achieving state-of-the-art results on the tasks of action anticipation and action prediction over EPIC Kitchens. We perform ablation studies over characteristics of the Anticipation Module to evaluate their utility. This chapter is based on my published work [10].

1.2.3 Chapter 4

In the same line of thinking of constituents of action, we next focus on investigating how motion understanding can be modeled in current video-text models. We introduce motion descriptions generated by GPT4 on three action datasets that capture fine-grained motion descriptions of activities. We evaluated several video-text models on the task of retrieval of motion descriptions.

We found they fall far behind human expert performance on two action datasets, raising the question: Do video-text models understand motion in videos? We introduce a method of improving motion understanding in video-text models by utilizing motion descriptions to address this. This method is demonstrated on two action datasets for the motion description retrieval task. The results draw attention to the need for quality captions involving fine-grained motion information in existing datasets and demonstrate the effectiveness of the proposed pipeline in understanding fine-grained motion during video-text retrieval. This chapter is based on my published work [11].

1.2.4 Chapter 5

Finally, we show some work on obtaining the constituents of objects and actions unsupervised. We introduce Symbolic Variational Autoencoders, which generate images from symbols representing semantic concepts. Unlike generic Variational Autoencoders (VAEs) or Generative Adversarial Networks (GANs), the latent distribution from the Symbolic Variational Autoencoder is discrete. The symbols are learned in a completely unsupervised manner by reconstructing images from symbolic encodings. We demonstrate the efficacy of our symbolic approach on the MNIST and FashionMNIST datasets. Results indicate that symbolic encodings naturally form a grammar where unique strings of symbols map to different semantic concepts. We further explore how changing these symbols affects the final image generated. This chapter is based on my published work [12].

1.2.5 List of publications

1.2.5.1 List of first author publications

1. **Chinmaya Devaraj**, Cornelia Fermuller, Yiannis Aloimonos. Diving Deep With Video-Text Models in Representing Motion . ACL Findings 2024

2. **Chinmaya Devaraj**, Cornelia Fermuller, Yiannis Aloimonos. Incorporating Visual Grounding In GCN For Zero-shot Learning Of Human Object Interaction Actions. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops 2023.

3. Eadom Dessalene*,**Chinmaya Devaraj***, Michael Maynord*, Cornelia Fermuller, and Yiannis Aloimonos. Forecasting action through contact representations from first person video. IEEE Transactions on Pattern Analysis and Machine Intelligence (2021).(* Indicates equal contribution)

4. **Chinmaya Devaraj**, Aritra Chowdhury, Arpit Jain, James R. Kubricht, Peter Tu, and Alberto Santamaria-Pang. From Symbols to Signals: Symbolic Variational Autoencoders.IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) IEEE, 2020.

1.2.5.2 List of first author Second/Third Author publications

1. Chengxi Ye, **Chinmaya Devaraj**, Michael Maynord, Cornelia Fermuller, Yiannis Aloimonos. Evenly Cascaded Convolutional Networks. The 1st International Workshop on Big Visual Dataset Construction, Management and Applications, IEEE BigData 2018.

2. Alberto Santamaria-Pang, James R. Kubricht, **Chinmaya Devaraj**, Aritra Chowdhury, and Peter Tu. Towards semantic action analysis via emergent language. IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR) 2019.

3. James Kubricht, Alberto Santamaria-Pang, **Chinmaya Devaraj**, Aritra Chowdhury, and Peter Tu. Emergent Languages from Pretrained Embeddings Characterize Latent Concepts in Dynamic Imagery. International Journal of Semantic Computing . (2020).

4. Eadom Dessalene, Michael Maynord, **Chinmaya Devaraj**, Cornelia Fermuller, and Yiannis Aloimonos. "Egocentric object manipulation graphs." arXiv preprint arXiv:2006.03201 (2020).

Chapter 2: Visual Grounding In GCN For Zero-shot Learning Of Human Object Interaction Actions

Graph Neural networks are defacto in learning representations for graphical data. Graphs are useful when representing scenes [13–15], actions [16, 17], and human-object interaction [10, 18] activities. Because of the ability to encompass the scene's structure with either attributes or symbols, knowledge graphs are a step toward explainable AI [19]. In this chapter, we utilize knowledge graphs to solve one of the challenging problems of recognizing human-object interaction activities under a zero-shot learning setting.

Consider a simplified formulation with two graphs typically used in zero-shot settings: an input language graph and an output vision graph that we are interested in learning. In the language graph, each node represents an action obtained from word embeddings. The language graph's edge weights encode the semantic distance between actions. The output vision graph has the same amount of nodes as the language graph and corresponds to the visual embeddings. Message-passing methods like graph convolution and graph attention have been used to learn the mapping between the known category nodes of the input graph and output graph to fill in information for the nodes in the output graph for which we don't have examples. In both the formulations of graph convolution networks (GCN) [20] and graph attention networks (GAT) [21], the structure of the input graph, along with its edges, is fixed.

One of the challenges of using GCN or GAT is that message passing relies only on the input graph semantics derived from language embeddings, completely being obscure to the semantics of the visual graph we are interested in predicting. However, we know that the word embeddings and visual embeddings need not be aligned semantically. Furthermore, word embeddings are prone to noise depending on the text corpora used to obtain them. The predicted visual graph from GCN carrying language semantics and the actual visual graph made using visual embeddings will differ owing to the domain gap between the language and the visual embedding. Moreover, in a zero-shot learning setting, we can't directly utilize an input graph made from visual embeddings as we only have information about the training class nodes. This raises the question of the ideal input graph for zero-shot learning. Using the above formulation to recognize human object interaction actions under a zero-shot learning setting has another challenge: the semantic gap between videos of unseen novel test classes and the seen training classes. In this chapter, we focus on introducing solutions to these issues from the perspective of knowledge graphs.

We propose two methods to visually ground the language graph to address the above challenges. In the first method, we modify the input language graph by changing the weights of the edges to reflect the visual semantics of the visual graph. Specifically, we modify the language adjacency matrix by adjusting its weights according to the weights from the vision graph. We use two different concepts to define shared concepts to group actions. We call these processes inhibitory and excitatory feedback for message passing in graph convolutions. We experimentally demonstrate that the modification of edge weights leads to improvements in the task of zero-shot action recognition on the Charades and EPIC-kitchens datasets.

In the second method, we integrate the visual graph in learning the GCN. Since the test classes are unknown, we first estimate the adjacency matrix of the visual graph for all the classes

by using the adjacency matrix of the language graph. We then modify the GCN graph propagation rule to integrate the visual adjacency matrix. We experimentally demonstrate the usefulness of the visual adjacency matrix under normal conditions as well as when the language graph is adversarially attacked.

In summary, our contributions are:

1. We highlight the limitation of using only the language graph in zero-shot learning, ignoring the semantics of the visual graph. We propose two methods to visually ground the language graph to tackle this.

2. The proposed methods are simple and easy to integrate with existing GCN message-passing methods.

2.1 Related Work

2.1.1 Graph Neural Networks

Graph convolutional networks (GCN) have been introduced by Kipf and Welling [20] for the semi-supervised classification of graph data. The core of the GCN is the graph propagation rule, which intuitively does feature aggregation of neighboring nodes. The importance of a neighboring node in learning the features of a node is given by the edge weight connecting the node with its neighboring node. The GCN of Kipf and Welling [20] uses an adjacency matrix to represent the edge weights and is fixed. Thus, its performance relies mostly on the accuracy of the adjacency matrix and to an extent the features of nodes. Graph Attention networks (GAT) [21], on the other hand, implicitly learn the importance of a node relative to its neighboring nodes

through self-attention over the features of nodes. There have been several improvements [22–25] over the original GCN and GAT, but the underlying graph propagation in networks still relies on the input graph. Elinas et al. [26] introduced structural learning of the GCN adjacency graph assuming a transductive setting. However, they assume that all training classes are independent of each other when conditioned on features and an adjacency matrix. For manipulation actions, we can't assume the classes are independent of each other. Many actions can be grouped based on attributes or shared concepts. Our method of using grouping of actions in GCNs shows that there is indeed benefit in assuming the interdependence of action classes. Furthermore, we build an inductive model to estimate the visual adjacency matrix as feedback from the visual graph. We do not assume any probabilistic model of the adjacency graph or the features. While updating the GCN, Ghosh et al. [27] add a triplet loss between positive and negative neighbors in the input language graph but still ignore the visual domain semantics to obtain the triplets. We differ from all previous methods by introducing another aspect to graph propagation. We introduce direct feedback from the output graph while doing graph propagation. The additional information from the output graph in the form of an adjacency matrix provides the relationship between the nodes in the output domain

2.1.2 GCNs For Zero-shot Action Recognition

Previous approaches for zero-shot action recognition include [28–36]. Many studies rely on attributes for zero-shot recognition. However, none of the previous works define attributes that are effective in large challenging human object manipulation datasets like the Charades dataset and EPIC-Kitchens55 dataset. Jain et al. for their system Objects2action [31] use objects present in

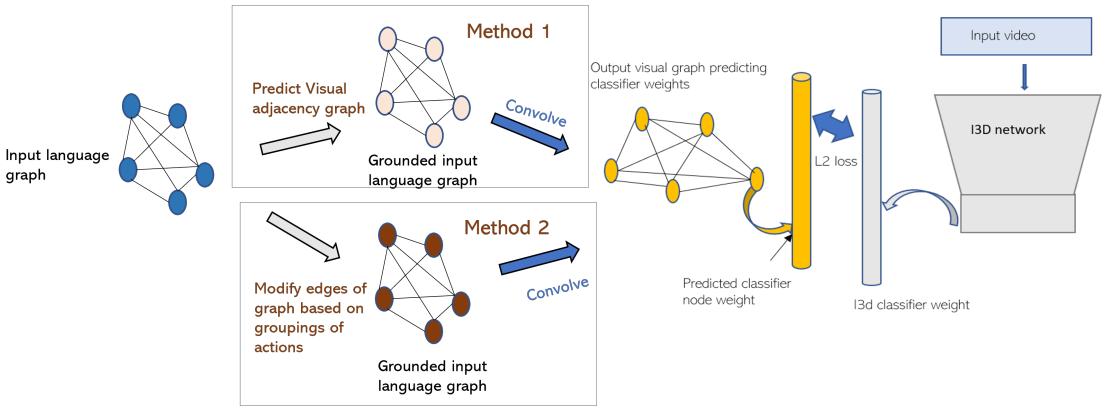


Figure 2.1: **Overview of our zero-shot action recognition approach:** The Input graph is visually grounded before being passed to a GCN to predict the output visual graph. Visual grounding refers to either the weight adjustment method (Method2) or the visual adjacency graph method (Method1) described in sections 2.2.5 and 2.2.6, respectively.

actions as attributes, and Liu et al. [34] use attributes for the classification of whole body actions. The methods in [27–29] are graph based using word embeddings only. Our approach falls under the umbrella of hybrid approaches as it combines word embedding with visual feedback and the visual adjacency matrix. In our experiments (Sec. 2.3) we compare against Ghosh et al. [28], who fuse three graphs to represent actions, objects and verbs and perform knowledge transfer. In comparison, our knowledge graph is much simpler consisting of just one graph instead of a fusion of three and relies on simply calculating the cosine similarities between the word embeddings. To the best of our knowledge, none of the previous works consider addressing the domain gap between visual and language graphs. A few studies use excitatory and inhibitory weight changes. [24] exploited multidimensional edge features. Jia et al. [23] introduced squeeze excitation of node channels completely in a feed-forward way of message passing by learning weights to excite the nodes. We differ from [23, 24] by incorporating excitation or inhibition based on the grouping of actions.

2.2 Method

2.2.1 Architectural Overview

Here we describe the high level formulation of our zero-shot action recognition. This forms our GCN baseline method referred to in the experimental section 2.3. Let the number of training classes be S, and the number of test classes be U.

Figure 2.1 shows the architecture of the network at training time. The network consists of an input language graph on the left, which represents all the actions in form of language vector embeddings, and a vision graph, on the right, whose nodes are the embeddings of actions from the visual domain.

The language graph nodes are initialized with the GloVe [37] word embeddings, and the value of the weights on the edges originally are the cosine similarity distances between the GloVe vectors.

2.2.2 Graph Convolution Network

Consider a graph G with N nodes. Let its adjacency matrix be A of dimension $N \times N$. The graph propagation rule from the formulation of the GCN by Kipf and Welling [20] is given by

$$H_{l+1} = f(A \times H_l \times W_l) \tag{2.1}$$

In equation 2.1 , H_{l+1} is the GCN output matrix at level $l + 1$ of dimension $N \times k$, and H_l is the input feature matrix at level l of dimension $N \times d$. W_l is the GCN weight matrix of dimension $d \times k$ that we are learning. Here k is the output feature dimension. A is the adjacency

matrix. f is a non-linear function which in our case is a leaky relu activation layer at the end of each convolution operation. We use the same normalization to the adjacency matrix as in [38]. After normalization of A the modified propagation rule is given by equation 2.2.

$$H_{l+1} = f(D^{-1} \times A \times H_l \times W_l), \tag{2.2}$$

with D denoting the diagonal matrix of the adjacency matrix.

2.2.3 Training

The graph convolutional network is trained to learn the nodes of the output vision graph. This is done as follows: In a pre-processing step, we train an I3d [2] classifier using the videos of action categories in the training set. We then utilize the final classifier layer weights of the I3d classifier to train the GCN. Let us denote $W_{classifier}$ of dimension $S \times k$ as the final classifier layer weight.

During training, the predicted embeddings $W_{predtrain}$ of the training nodes of the output vision graph are matched with the I3d classifier weights $W_{classifier}$ obtained earlier in the pre-processing step. We ensure that the node representing say for example "put " action class is matched with the I3d classifier layer weight representing the "put" action class. Here $W_{predtrain}$ of dimension $S \times k$ represents the visual embeddings of the training nodes of the graph. Specifically $loss$,

$$loss = ||W_{predtrain} - W_{classifier}||^2 \tag{2.3}$$

is minimized between the predicted weights and the trained I3d classifier weights.

2.2.4 Testing

At test time, given a test video, the I3d features f_{test} of dimension k are extracted. f_{test} is compared against all the nodes of the output GCN representing testing classes to obtain the action category of the video. Let $W_{predtest}$ of dimension $U \times k$ represent the visual embeddings of U number of test class nodes. Using the symbol ^{T} to denote transpose of a matrix, i.e., f_{test}^T, the predicted class Y is derived from equation 2.4

$$Y = softmax(W_{predtest} * f_{test}^T) \tag{2.4}$$

2.2.5 Visual Grounding Using Grouping Of Actions

Since we don't have access to all the nodes of the output graph at training time, we employ an indirect approach using the output graph's semantics. We form groups of actions according to cognitive concepts which the actions have in common. Using super-categories and creating groups of actions has roots in defining an ontology of actions [39]. In our case, we define a higher-level representation of manipulation actions by considering the geometric transformation performed on the manipulated object [40], topological changes on the scene, or the type of object motion involved. The grouping of actions follows cognitive reasoning and can be extended to any human-manipulation activities dataset.

2.2.5.1 Defining the grouping of actions using shared cognitive concepts

We group actions according to two criteria. First, according to the object's change of state that the manipulation action induces. Second, based on the types of object motion involved.

Here we list the actions in the EPIC Kitchens dataset having that shared concept of the first kind.

- Decrease in size: squeeze, press, crush, fold, knead
- Increase in size: open, stretch
- Add (to the scene): put, pour, insert, fill, add, apply, spray.
- Remove (from the scene): take, remove, empty, scoop, filter
- Separate: Cut, break, peel, divide.

We can also group actions based on the motion type (second kind)

- complex motion: mix, shake
- translational motion: move, put, insert, take, remove.
- rotational motion: turn, scoop
- no-motion: grasp, hold.

Charades dataset, however, includes not just the manipulation actions but actions like "walk", "sit", "watch" for which object state changes don't directly apply. Another interesting feature of the Charades dataset is that the action classes are a combination of both verbs and objects. Examples include "Putting something on a table", "watching a book", "watching a window", "smiling at a mirror", and "watching something/someone/themselves in a mirror". For the Charades dataset, we form groups based on the common object involved in an action. We hypothesize that actions involving the same objects tend to have closer RGB visual features based

on the discussion in [41] which says that there are more errors for classifiers of actions with the same object and different verbs. Furthermore, our goal is not to create super categories of actions that classify all actions. But instead, come up with a logical way of grouping actions that can be easily generalized in any object manipulation dataset. While there are many ways of forming groups, we have seen from the experiments that our algorithm is resilient to specific groupings of actions.

2.2.5.2 Modifying the adjacency matrix based on grouping of actions

Here we describe how we utilize the grouping of actions to modify the adjacency matrix obtained from language embeddings. Let $A_{language}$ represent the adjacency matrix obtained from word embeddings. $A_{language}[i, j]$ represents the edge features between nodes i and j. If nodes i and j belong to the same group of actions, they are expected to be visually more coherent. Thus, we intend to excite the edges when nodes i and j belong to the same group of actions and inhibit the edges when they are in a different group. There are many ways to do it. In our experiments, we increase the weight of the edge between nodes i and j by a constant value when they belong to the same category (excitation operation), and we decrease it by the same constant value when they are different (inhibition operation). Let us denote the modified adjacency matrix after this weight adjustment process as $A_{grounding}$. We set the mean of the row normalized language Adjacency matrix $A_{language}$ as the constant to be added or subtracted in excitation or inhibition. The intuition behind this is that this way the edge weights remain positive, and excitation is still mostly in the same order of the original edge weight. Very high values would lead to instability, especially in the inhibition operation, and very small values would not lead to any significant changes to the

adjacency matrix. In the experimental section, we provide a sensitivity analysis of the choice of constant and how it impacts performance.

After we obtain $A_{grounding}$ by modifying the original adjacency matrix $A_{language}$, we use $A_{grounding}$ for graph propagation in equation 2.2. Equation 2.5 describes the process of graph propagation.

$$H_{l+1} = f(D^{-1} \times A_{grounding} \times H_l \times W_l) \tag{2.5}$$

Modifying the distances of word embeddings by a constant based on the grouping of actions, may not be the optimal way to solve the domain shift between language embeddings and visual embeddings. However, it is very efficient in terms of ease of implementation and it incorporates the visual domain knowledge in the input adjacency matrix. Once the adjacency matrix $A_{language}$ is modified based on the grouping of actions in equation 2.5, we use the same propagation rule as given in equation 2.2 to learn the classifier weights of test classes. Essentially, we are training the GCN only once in the entire process thereby reducing the complexity of incorporating visual feedback to the whole process.

2.2.6 Modifying Weights Via The Visual Adjacency Matrix

Here we introduce a second method to learn the visual feedback required to incorporate the domain shift between the language graph and the visual graph. Let us refer to the adjacency matrix of all the visual embeddings as the visual adjacency matrix A_{visual}. If we utilize only A_{visual} for graph propagation, the updated GCN propagation rule becomes equation 2.6.

$$H_{l+1} = f(D^{-1} \times A_{visual} \times H_l \times W_l) \tag{2.6}$$

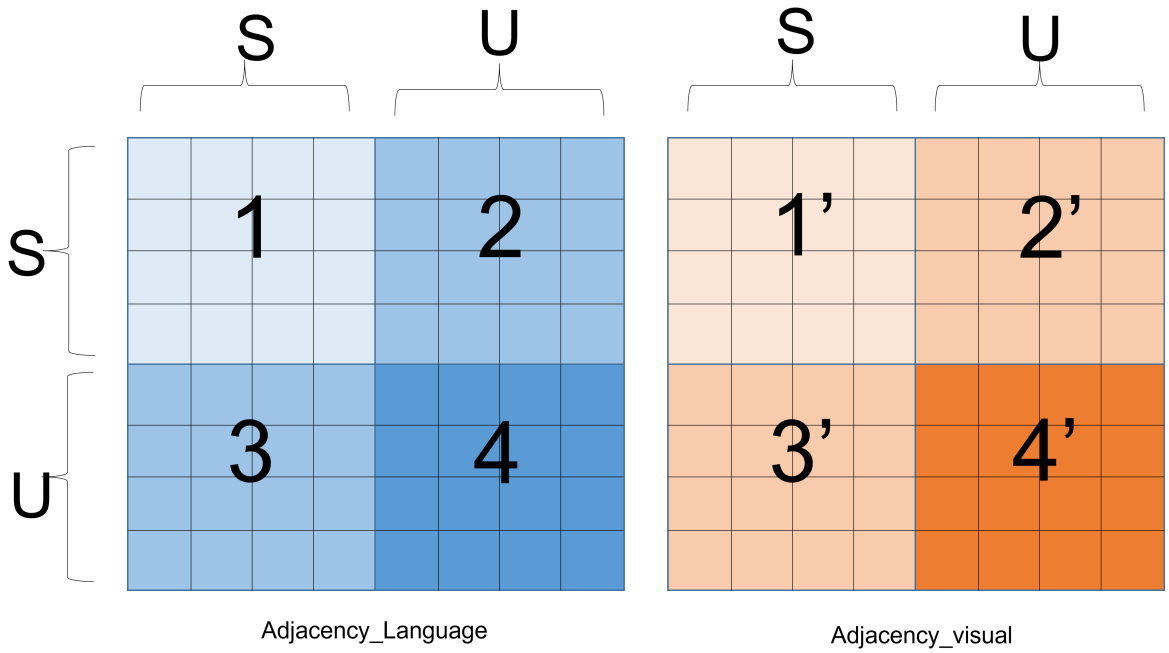


Figure 2.2: Learning the transformation from $A_{language}$ to A_{visual}

However A_{visual} is not available at training time. To obtain A_{visual} at testing time we learn a transformer function F shown in equation 2.7 that transforms the adjacency matrix from the language domain to the visual domain. In other words, the transformer function is learning how to predict the distances of nodes in the visual graph from the distances of nodes of the language graph. We use the GCN network to learn the visual embeddings of the test classes whereas the transformer function is used to estimate the distances between the visual embeddings of test classes.

$$F(A_{language}) = A_{visual}. \tag{2.7}$$

2.2.6.1 Learning Transformer Function F

Consider a language adjacency matrix $A_{language}$ with N nodes of dimension $N \times N$. The total number of classes in the dataset would also be equal to N. Let there be S training classes

and U test classes in the dataset. We are interested in estimating a visual adjacency matrix A_{visual} with N nodes of dimension $N \times N$. $A_{language}$ can be easily obtained for all the N classes from its word embeddings. However, for obtaining A_{visual} we only have with us S classes during training. The remaining values have to be estimated. Training one regression model for the entire dataset didn't yield good results. Instead, we split the regression into parts and learn the different parts separately. The performance of the system depends on how well we can learn the transformer function F.

Each row in the adjacency matrix represents the cosine similarities between it and the rest of the classes. We rearrange the adjacency matrix for illustration purposes to show how we learn the transformer function. We rearrange it such that the first S rows of the adjacency matrix represent that of training classes and the next U rows represent that of test classes. Similarly, the first S columns represent that of training classes and the next U columns represent that of test classes. Figure 2.2 illustrates this rearrangement. Submatrix 1 of $A_{language}$ contains the cosine similarities between the S training classes of $A_{language}$. Submatrix 2 contains the cosine similarities between S training classes and U test classes of $A_{language}$. Submatrix 3 contains the cosine similarities between U testing classes and S training classes of $A_{language}$. Submatrix 4 contains the cosine similarities between the U test classes of $A_{language}$. Submatrices $1', 2', 3', 4'$ similarly are rearranged for A_{visual}. Submatrix $1'$ is available at training and our goal is to obtain submatrices $2', 3', 4'$

To obtain $2'$, we train S number of regression models one for each row of $2'$, utilizing one row each of 1 and $1'$ as training data such that $F(1) = 1'$. $3'$ is obtained as the transpose of $2'$ due to symmetry. $4'$ is obtained by a regression model that is trained on the entire submatrices 1 and $1'$ such that $F(1) = 1'$. Each of these regression models consists of a two-layered fully connected

layer neural network having 10 nodes in each layer having relu activation layers and solved using the LFBGS solver.

Once A_{visual} is obtained, we could use it instead of the language matrix. However, there are advantages for using both A_{visual} and $A_{language}$ for the sake of making the model robust. There are various ways of utilizing both A_{visual} and $A_{language}$. Ma et al. [42] developed a multi-dimensional GCN where A_{visual} and $A_{language}$ can be two channels of a larger A matrix. However, for the sake of simplicity, we use a single-dimensional GCN and propose a new modified GCN propagation rule in equation 2.8 which includes both A_{visual} and $A_{language}$.

$$H_{l+1} = f(D^{-1} \times (A_{language} + \beta \times A_{visual}) \times H_l \times W_l) \tag{2.8}$$

Here β is the weight factor that can be determined empirically for best performance.

2.3 Experiments

2.3.1 Datasets

Experiments are conducted using the **EPIC-Kitchens-55** dataset [43] and the **Charades** dataset [44]. We chose these datasets as they are some of the largest human manipulation action datasets and we can test our ideas of visual grounding through grouping of actions on them.

EPIC-Kitchens-55 is an egocentric activity dataset of people recording their activities in kitchens. It has over 125 verbs and 331 nouns. We selected the top 50 most frequent verbs provided by the authors of [43] for the experiments in this paper to avoid the long tail problem. Among those, we used 22 verbs as training and 27 verbs as test classes for evaluation purposes.

We eliminated the verb ”walk” because it is not a human manipulation action. For each verb, we manually formed the group of actions based on the geometric or topological change or the object motion involved as described earlier. The train and test splits and the groupings of actions will be released after acceptance.

Charades is a dataset made up of crowdsourced videos of activities in people's homes. The videos are on average 30 seconds long and each includes a sequence of multiple action classes. There are 157 action classes and for our experiments we used splits of 79 training and 78 test classes as reported in [28]. Descriptions of all videos along with the object classes are provided. For this dataset, because there are a significant amount of non manipulation actions and the action classes themselves are a combination of verbs and objects involved, we formed groups of actions based on the object involved. Intuitively, actions involving the same manipulated objects tend to have closer RGB visual features Since for this dataset the object manipulated in each video is provided, there was no need for further annotation.

Method	EPIC-Kitchens (22-27 split)
ESZSL [45]	7.33
DeViSE [46]	10.25
DEM [47]	8.66
knowledge graphs [28]	13.94
GCN baseline	15.21
Ours GCN-I	16.78
Ours GCN-IE	16.94
Ours GCN-E	**19.62**

Table 2.1: Evaluation of classification accuracy for novel test classes on EPIC-Kitchens Dataset.

Method	Charades
ESZSL [45]	17.21
Knowledge graphs [28]	18.21
GCN baseline	18.43
Ours GCN-I	18.16
Ours GCN-IE	19.16
Ours GCN-E	**19.35**
Ours GCN AV*	**19.19**
Ours GCN AV	**19.63**
GCN baseline with attack	15.846
GCN- V after attack	**17.11**

Table 2.2: Evaluation of classification accuracy for 79 novel test classes on Charades Dataset. We report mean average precision mAP.

2.3.2 Implementation Details

2.3.2.1 Baseline GCN

We used the I3d network [2] to obtain classifiers for the Epic-Kitchens-55 and the Charades dataset. EPIC-Kitchens-55 is a highly unbalanced dataset when it comes to the number of videos per class. This resulted in poor performance in zero-shot learning. Therefore we sampled the dataset such that approximately 50 samples per class were present in the training set. We initialized the weights of the I3d network with those pre-trained on Imagenet and finetuned the last layer of the I3d classifier for the EPIC-Kitchens dataset. For the Charades dataset, the I3d model classifier was trained end to end and we followed the insights from [28] to obtain the I3d classifier weights.

The trained I3d classifier weights were then used to train the GCN. We used a two-layer GCN having hidden layers of 1024 $\rightarrow$1024 dimensions to predict the I3d classifier weights. Empirically, we found that having two layers in the GCN gave better results than deeper GCNs.

We initialized the input graph nodes with 300-dimensional GloVe embeddings trained on the Wikipedia dataset. The input graph node dimension is 300, and the output graph node dimension is 1024, equal to the I3d feature dimension. The number of nodes of the graph for the EPIC-Kitchen dataset was 50, which is equal to the number of verbs we selected in our dataset. Although we removed "walk" from the dataset we didn't remove its nodes in the graph as this didn't affect much the experiments. The number of nodes of the graph for the Charades dataset was 157, which is equal to the total number of classes in the dataset. We updated only the training nodes during the training. We used the Adam optimization algorithm with a learning rate of 0.0005 and momentum of 0.0001 for the Charades dataset, a learning rate of 0.0001, and momentum of 0.0001 for the EPIC-Kitchen dataset. The GCN was trained with L2 loss to predict the classifier weights for about 15000 epochs for the Charades dataset and for 400 epochs for the EPIC-Kitchens dataset. For all subsequent experiments and ablation studies we used the same architecture setting and network learning parameters. Only the way the adjacency matrix was computed differs.

2.3.3 Experimental details:

Table 2.1 and Table 2.2 summarize the results of zero-shot action recognition for different ways of forming the graph and training it on the EPIC-Kitchens and Charades dataset, respectively. We compare our method against a baseline GCN with no modification of the weights and against other methods previously reported in the literature.

Table 2.2 shows the results of zero-short action recognition on the Charades dataset. The results of ESZSL [45] and [28] are from [28]. We report the Mean Average Precision of the

test categories for the Charades dataset and report percentage accuracy for the test categories of the EPIC-Kitchens dataset. GCN-I stands for GCN with inhibition of edges and GCN-E stands for GCN with excitation of edges. GCN-IE stands for GCN with both excitation and inhibition. GCN-E was the best performing model for the Charades dataset, outperforming the baseline GCN, knowledge graphs [28], and ESZSL method [45]. GCN-IE, which has both excitation and inhibition, is the second best. Inhibition of edges is not leading to significant improvement in performance compared to excitation.

A similar trend is also seen for the zero-short action recognition results on the Epic-Kitchens dataset. We implemented ESZSL [45], DeViSE [46], DEM [47] baselines and compared against our methods as shown in Table 2.1. Our method GCN-E nearly has 10% improvements over previous zero-shot action recognition methods. We further report experiments of using the estimated A_{visual} adjacency matrix on the test splits of the Charades dataset in table 2.2. $GCNAV*$ represents GCN using $A_{language}$ and estimated A_{visual}. $GCNAV$ represents GCN using $A_{grounding}$ alongside estimated A_{visual}. Using the visual adjacency matrix on top of $A_{grounding}$ leads to the best performance on Charades, further validating the idea of visual feedback through the A_{visual}.

Method	Charades (40-39 split)
$GCN - L$	12.92
$GCN - V_{ideal}$	15.60
$GCN - V_{ideal}L$	**17.28**
$GCN - V_{estimated}$	12.06
$GCN - V_{estimated}L$	**13.46**

Table 2.3: Evaluation of classification accuracy for 39 novel test classes in ablation study on Charades Dataset. Results are in mAP.

Method	Set 1	Set 2	Set 3
GCN-I	17.21	16.78	16.79
GCN IE	16.48	16.94	16.3
GCN E	**18.72**	**19.62**	**18.61**

Table 2.4: Evaluation of classification accuracy for novel test classes on EPIC-Kitchens Dataset for different groupings of actions

	$c = 0.02$	$c = 0.01$	$c = 0.005$
GCN-E	18.72	17.90	16.90
GCN-I	5.18	17.21	15.90

Table 2.5: Variation of classification accuracy on novel test classes on EPIC-kitchens dataset for different "constant" parameter used in excitation and inhibition process. In the table c refers to "constant".

2.3.3.1 How to set the "constant" parameter used in the inhibition and excitation process?

As described earlier, the "constant" parameter can be set as the mean (over all elements) of the normalized $A_{language}$ matrix. Table 2.5 shows the sensitivity to this "constant" parameter on the performance of GCN-E and GCN-I for the EPIC-kitchens dataset. A smaller value than the mean of $A_{language}$ results in less accuracy improvements for GCN-E. For this experiment the mean of $A_{language}$ was 0.02. For GCN-I however a value slightly lower than the mean of $A_{language}$ results in best performance. We can tune the system to get the best values for the "constant" starting with the mean.

2.3.3.2 Studying the effect of weight adjustment on different groupings of actions

We wanted to check if our proposed method is sensitive to the choice of shared concepts used to group actions. We created three sets of different shared concepts and made groups of

	$\beta = 0.01$	$\beta = 0.02$
Test Acc	19.19	18.97

Table 2.6: Variation of classification accuracy for 79 novel test classes on Charades dataset for different β in equation 2.8 for GCNAV* setting.

actions based on those sets (with a varying number of shared concepts). We performed the experiments on these three sets on the EPIC-kitchens dataset. Table 2.4 shows that our method of weight adjustment is agnostic to grouping of actions. Here we list the different shared concepts that were chosen for each set.

- Set1: Decrease in size, Increase in size, Add (to the scene), Remove (from the scene), Separate.
- Set2: Decrease in size, Increase in size, Add (to the scene), Remove (from the scene), Separate, Complex motion
- Set3: Translational motion, Complex motion, Decrease in size, Increase in size, Add (to the scene), Remove (from the scene), Separate.

2.3.3.3 Effect of Visual Adjacency Matrix

To evaluate the impact of the visual adjacency matrix on the overall GCN propogation (equation 2.8), we performed some ablation experiments on the Charades dataset. For this, we only used the training set of 79 action classes in the charades dataset. Among these, we randomly chose 39 test classes, and the remaining 40 were kept in the training. Then we performed zero-shot action recognition on the 39 test classes by using a different adjacency matrix in each case study. We analyze the following different cases of chice of initial adjacency matrix:

1. GCN-L: This is a baseline GCN that uses the language adjacency matrix and the formulation represented in equation 2.2. We constructed a 79×79 dimensional language adjacency matrix $A_{language}$ and computed the test accuracy for 39 classes.

2. $GCN - V_{ideal}$: This is a GCN using the ground truth visual adjacency matrix. From this experiment, we could understand the upper limit of test class accuracy using the visual adjacency matrix A_{visual}. For this experiment, we computed the adjacency matrix using the entire 79 training classes in the original charades dataset. Instead of estimating the visual adjacency matrix, we used the ground truth I3d class weights and constructed A_{visual}. We computed the test accuracy for 39 classes and used the formulation in equation 2.6.

3. $GCN - Vestimated$: This is a GCN that uses the estimated visual adjacency matrix A_{visual} and the formulation in equation 2.6. We used 40 training classes and 39 test classes to learn A_{visual}. We set up a transformer network, as described in subsection 2.2.6.1 to obtain F. From F we obtained the visual adjacency matrix A_{visual}. We used the visual adjacency matrix A_{visual} in equation 2.6 to obtain the zero-shot action test accuracy for 39 classes.

4. $GCN - V_{ideal}L$: This is a GCN that uses both the ideal visual adjacency matrix A_{visual} and the original language adjacency matrix $A_{language}$. We used equation 2.8 to obtain the zero-shot action test accuracy for 39 classes. β in equation 2.8 was set to 0.2

5. $GCN - VestimatedL$: This is a GCN that uses only an estimated visual adjacency matrix A_{visual} and A_{visual} language adjacency matrix $A_{language}$. We used 40 training classes and 39 test classes. We set up the transformer network described in subsection 2.2.6.1 to obtain F. From F we obtained the visual adjacency matrix A_{visual}. We used the visual adjacency matrix in equation 2.8 to obtain the test accuracy for 39 classes. β used in equation 2.8 is 0.1

Table 2.3 reports the results of the ablation study. We report the Mean Average Precision for 39 test classes in the table for the Charades dataset. From $GCN - V_{estimated}L$ results in table 2.3, we can see that using the estimated visual adjacency matrix A_{visual} has a lot of value on its own and performs as well as using it instead of $A_{language}$. Experiments on $GCN - V_{ideal}L$ were conducted to find the upper limit of test accuracy using A_{visual}. The experiments show that just the idea of using A_{visual} is beneficial provided we are able to learn a good transformer function F. However, for all practical purposes we have to rely on the estimated visual adjacency matrix A_{visual} using the transformer function F. Training the transformer function F is hard as we have limited data points for training and it is a complex function to learn. Learning a better F depends on accuracy of $A_{Languagetrain}$ and $A_{visualtrain}$. If both of these are unreliable, then the overall accuracy of the system will be poor, and this is one of the limitations of our approach.

Setting parameter β We performed experiments to determine sensitivity of parameter β in equation 2.8. Table 2.6 reports experiments for different parameters of β in Equation 8 for the Charades dataset experiments reported in Table 2.2 for the GCNAV * setting. An initial value can be set like in any multitask learning.

2.3.3.4 Robustness to Noise in the Language Graph

While there is no direct negative social impact with our approaches, graphs are easily prone to adversarial attacks. We evaluate the performance of GCNs by creating perturbations on the input language graph. We created perturbation on the input language graph by poisoning the edges. Specifically, we poisoned the edges of those verbs belonging to same grouping of actions. We decreased the weight of these edges by 0.1. We then re-ran our algorithms on this setting.

GCN after it was attacked had a 15.8 MAP on the charades test set when using the attacked adjacency matrix. $GCN-V$ uses the estimated visual adjacency matrix A_{visual} from the attacked adjacency matrix along with the attacked adjacency matrix. $GCN-V$ had a 17.11 MAP on test classes showing robustness when using the estimated visual adjacency matrix despite the original language matrix being attacked.

2.4 Conclusion

This chapter discussed our work published in [9]. We used object-state changes and objects as constituents of actions. We presented two methods to improve message passing by providing visual feedback in GCN. Inhibition and excitation of edges is a simple and effective method for introducing visual grounding in an input language graph. Grouping actions based on object state changes or other properties helps in this process. Since the adjacency matrix is modified only once, training is simple. The second method introduces modifications to message passing in GCN to incorporate the visual adjacency graph. The visual adjacency graph, often overlooked, is still helpful in providing visual feedback in a language graph. We provided comprehensive quantitative evaluation of the proposed methods on Charades dataset and EPIC-kitchens dataset.

Chapter 3: Forecasting Action Through Contact Representations From First Person Video

Understanding and anticipating others' actions is a necessary capability for fluid human interaction and collaboration. Without this capability collaboration involves excessive wait times as we wait for others' actions to complete. Responding earlier to others' actions reduces physical load, cognitive load, and the completion time of the task [48, 49].

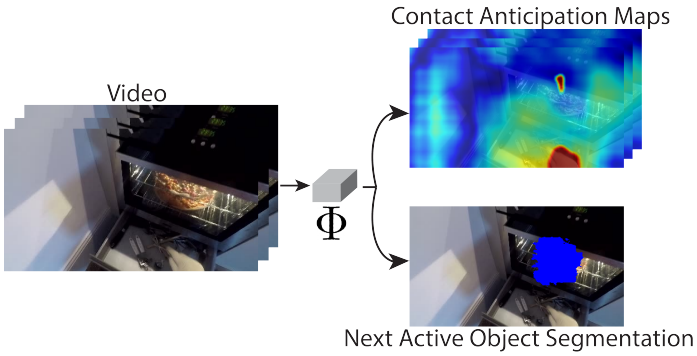


Figure 3.1: Illustration of representations involved in the Anticipation Module Φ where the scenario depicts a person reaching into the oven. RGB video feeds into the Anticipation Module which produces Contact Anticipation Maps and a localization of the Next Active Object Segmentation. Visualization colors in the Contact Anticipation Maps vary from blue (large time to contact) to red (pixels belonging to hands or objects in contact).

In our work on action understanding, we leverage first person - or egocentric - perspective, rather than the third person perspective more common in action datasets. There are a few reasons for this: 1) the egocentric perspective provides a less occluded view of the hands and the action

being performed, 2) this view contains cues of intentionality – e.g., we tend to look towards the destination or focus of our actions, 3) as head mounted displays, including augmented reality headsets, become more common, egocentric data is becoming more readily available and methods involving the egocentric perspective more relevant (and, robots are able to leverage egocentric data from human worn sensors). In this chapter we work with the egocentric datasets EPIC Kitchens [43] and EGTEA [50].

Human interaction with the environment is largely performed through hand manipulations of objects. Each manipulation involves the making and breaking of hand object contact as a defining characteristic. Possible benefits of contact include better: 1) determining the class of action being performed, 2) delineating action boundaries, and 3) projection into the near future of action. As such, we structure our representations around contact with objects.

We define two classes of object, aligning with two different times of interest: the present, and the near future. Previous works [51] have defined an *Active Object* as an object currently involved in a given interaction. In this work, we define the *Active Object* of a hand as the object presently in contact with the hand, and we define the *Next Active Object* as the object which will next come into contact with that hand. In seeking to model future action we produce predictions for the Next Active Object.

Understanding which objects are Active Objects involves understanding hand-object contact. Understanding Next Active Objects involves predicting future hand-object contact. There is evidence from the cognitive science literature that modeling of contact plays a central role in human visual understanding of action [52, 53]. As such, we center our models around contact.

We introduce components and representations useful for understanding contact. The *Anticipation Module* contains two networks: the *Contact Anticipation Network*, and the *Next Active Object*

Network. The *Contact Anticipation Network* produces a representation termed the *Contact Anticipation Map*, and the *Next Active Object Network* produces a *Next Active Object Segmentation*. See Figure 3.1 for an illustration of the representations produced by the Anticipation Module.

Pioneering works in cognitive science (e.g., [1]) indicate that the velocity profiles of point-to-point hand movements follow a bell shaped distribution. We verify the presence of this bell shaped distribution with experiments: See Section 3.2.1 for an illustration. During the onset of hand motion, the hand gradually accelerates, and as the hand approaches contact it rapidly decelerates. This is a motion cue relevant to action on which humans rely when understanding each other's actions [54]. This shows that in hand reaching there is structure in the relations between the position of the hand, the position of the object, and the velocity of the hand. This low-level cue is of central relevance to action understanding, particularly anticipation of near future action characteristics, and we model it through the Contact Anticipation Maps.

Contact Anticipation Maps are a hand-centric representation, providing a pixel-wise estimation of potential time-to-contact between the hand and pixels in the scene. Pixels belonging to the hands of the actor and Active Object(s) are represented with time-to-contact values of 0. See Figure 3.1 for illustration.

The Contact Anticipation Network produces Contact Anticipation Maps in a low level fashion, without utilizing components or representations critically dependent upon accurate performance of object detectors, hand trackers, the category of the object being acted upon, or classification of the action being performed. This low-level approach to anticipating the next active object is a less brittle approach than approaches critically dependent on the performance of object detectors and hand trackers.

We feed a history of Contact Anticipation Maps in parallel with a stack of RGB frames to a

network whose purpose is to localize the Next Active Object - the Next Active Object Network. The Next Active Object Network produces a representation localizing the likely Next Active Object - the Next Active Object Segmentation. In combination the Contact Anticipation Network and the Next Active Object Network provide a prediction for *where* in the scene the Next Active Object will be, and *when* contact with that object will be established.

The Next Active Object Segmentation is useful in understanding the type of interaction which will take place. Segmentation provides cues such as size, shape, and distance from the person, as well as providing a specific localization over which object classification can be run, providing an object category.

To produce data with which to train the Anticipation Module we augment a portion of the EPIC Kitchens dataset with annotations of hands and objects, and the times at which hand / object contact occurs. This allows us to construct, at each frame prior to contact, a pixel level labeling of the hand, the Next Active Object, and the time remaining until that object and the hand come into contact. EPIC Kitchens provides RGB data, and includes no depth data - and while hand trajectories are best represented in 3 dimensions, 2 dimensional projections still provide ample trajectory information.

In our work on action understanding we approach two related tasks: action prediction, and action anticipation. Action prediction is the task of recognizing an action given only a partial observation of an ongoing action. Action anticipation is the task of anticipating the category of a near future action before its start. The representations produced by the Anticipation Module are of utility to the tasks of action prediction and anticipation, and we evaluate the anticipation module w.r.t. performance on these tasks.

Not only are the short range action characteristics provided by the Anticipation Module

relevant to these tasks, but longer-range activity structure is relevant as well. For the modeling of longer range context and relations, methods beyond the Anticipation Module are needed. We extend the temporal window of activity modeling with Egocentric Object Manipulation Graphs (Ego-OMG) [55], aggregating the representations from the Anticipation Module in producing representations for sequences of high-level states spanning large timespans of activity. Because of this we are able to abstract from contact derived representations to semantic modeling of the flow of activities. This also allows us to evaluate the utility of the Anticipation Module within the context of a full action understanding system.

The architecture of Ego-OMG consists of two streams. The first stream captures visual appearance and short term dynamics. This stream consists of a CSN [56] a variant of the I3D Network [2] making use of channel-wise group 3D convolutions. The second stream leverages the output of the Anticipation Module in modeling the temporal semantic structure of the activity being performed. The core of this second stream is a graph representation embedded into a vector space through use of a Graph Convolutional Network (GCN) [20].

Ego-OMG's graph representation is constructed as follows: transcripts of the activities from the training set are processed to produce a graph structure capturing the connections from state to state through actions. The nodes of this graph consists of state representations derivable from the Anticipation Module - categorical representations for the Active Object from the Contact Anticipation Network and the Next Active Object from the Next Active Object Network, modelling the left and right hands separately.

The CSN and GCN streams are then combined to produce an action prediction.

We perform ablation studies over the Anticipation Module's representations, and through doing so determine which characteristics of those representations are responsible for their utility

to action anticipation and prediction.

Using the representations produced by the full Anticipation Module we demonstrate state-of-the-art performance over the recent EPIC Kitchens Action Anticipation Challenge, achieving 1 place on the EPIC Kitchens Action Anticipation Challenge unseen test set, and 2 place on the seen test set, and outperform all previously published approaches without any use of ensembling, unlike many competing approaches.

The primary contributions of this work are:

- A novel training signal for action understanding capturing information of time-to-contact between hands and objects, and segmentations of hands and objects. Over this signal we train the Anticipation Module, consisting of two networks which produce the following low level action representations:

    1. Contact Anticipation Maps: pixel wise anticipated time-to-contact involving one of the left or right hands.

    2. Next Active Object Segmentations: segmentations localizing candidate Next Active Objects.

- A surpassing of the state-of-the art with a full action understanding framework - Ego-OMG - built upon the proposed Anticipation Module, achieving 1 and 2 place on the unseen and seen test sets respectively of the EPIC Kitchens Action Anticipation Challenge.

The remainder of this chapter is structured as follows: In Section 3.1 we provide an overview of related work; in Section 3.2 we detail our method; in Section 3.4 we describe our experiments and results;

3.1 Related Work

3.1.1 Action Anticipation and Prediction

Action anticipation is the task of classifying future actions from observations that end before the actions begin. Action prediction is referred to in many works as "early action recognition": we adopt the nomenclature of [57], referring to the classifying of partially observed actions as action prediction. While the study of action recognition has received significant attention, the study of action anticipation and action prediction has only recently begun to attract more attention [58–61], particularly in the egocentric setting [62–65].

3.1.2 Egocentric Cues

Previous works have demonstrated that exploiting hand motion and formation in various forms can improve action recognition performance [66, 67]. Most previous action recognition frameworks incorporate hands by feeding hand detection patches [66,68], 3D joint pose estimations [69], or both [70–72]. Li et al. [73] utilized the manipulation point, a 2D point in the image representing a point in reference to each of the hands, as an egocentric feature for action recognition. Fewer works attempt to utilize the hand trajectory as a cue. Liu et al. [63] propose motor attention, the anticipated future hand trajectory enacted throughout the performance of an action.

Rather than explicitly modelling future trajectories - which are inherently ambiguous - we focus on predicting the endpoint of the trajectories, terminating in contact with objects. For this, we leverage our Contact Anticipation Maps stacked through time. This history of Contact Anticipation Maps implicitly contains trajectory information.

3.1.3 Active Objects

Anticipating future object interaction has been explored in many recent works. Furnari et al. [74] propose a method which relies on an object detector that exhaustively identifies a list of objects to track in a small sliding window - they feed each tracking trajectory to a random forest classifier to distinguish between 'active' and 'passive' trajectories. Nagarajan et al. [75] utilize pairs of inactive object images and videos of the corresponding objects in action, learning a mapping between the two to learn 'interaction hotspots', or regions of likely activity. Xiao et al. [76] tackle the same task, proposing a novel architecture that utilizes objects to determine where actions are most likely to occur, and vice-versa.

In this work, we make a distinction with respect to these works as to the definition of an Active Object. Rather than refer to the object involved in the current action, we define an Active Object as the object presently in contact with a hand. This low-level definition of an Active Object better captures the objects involved in a given interaction.

3.1.4 Video Representation

Typical works within action understanding involve two-stream architectures where the input to the network is RGB video fed to the network in parallel with pre-computed frames of optical flow [2, 77]. These approaches have achieved success in tasks where appearance and short-term motion is sufficient for the task at hand (i.e. action recognition) [2]. However, it has been reported [2, 50, 63] that such methods do not transfer well to tasks such as action prediction or action anticipation. We find this understandable, as action anticipation requires reasoning about complex semantic cues that go beyond appearance.

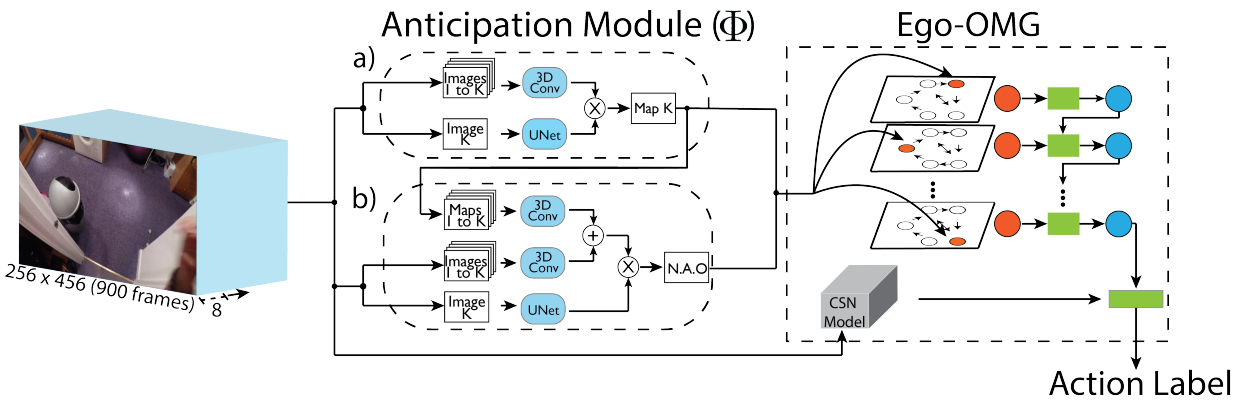


Figure 3.2: Overview of our proposed approach. The input video of 900 frames is fed in sliding window fashion with windows of size 8 to the Anticipation Module. Anticipation module Φ consists of two networks: a) The *Contact Anticipation Network*, which outputs *Contact Anticipation Maps (Map K)*, a representation which feeds into b) the *Next Active Object Network*, producing a *Next Active Object (N.A.O.) Segmentation*. The $\bigoplus$ denotes addition; $\bigotimes$ denotes multiplication. Refer to Sections 3.2.2.2 and 3.2.2.3 for the architectural details. The Anticipation Module Φ's output is in turn is fed into Ego-OMG, which in turn produces labels for action anticipation and prediction.

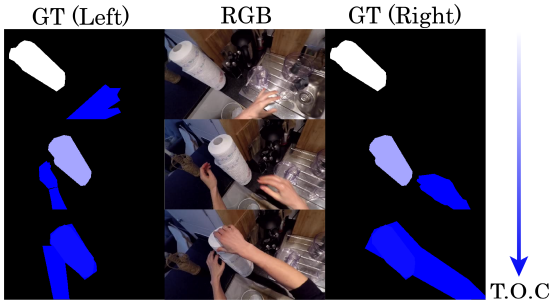


Figure 3.3: Illustration of annotations added to a portion of the EPIC Kitchens dataset in construction of our augmented dataset. The left and right columns contain added annotations for the left and right hands, respectively. The middle column illustrates associated clip frames. The annotations consist of segmentations of hands and Active and Next Active Objects. Pixels belonging to (Next) Active Objects are assigned non-negative values relative to the time-of-contact (T.O.C). Colors vary from white to blue based on remaining time to contact, with values of 0 associated with both Next Active Objects and hands. Background pixels are colored black, represented with values of -1.

Rather than simply represent the video as a stack of frames, it is desirable to capture the long-term semantics underlying the video observation of the activity. Recent works have proposed the enrichment of raw video features with graphs [64, 78–80]. Typically graph nodes represent detected objects, actors, or locations. Unlike other works that utilize an exhaustive

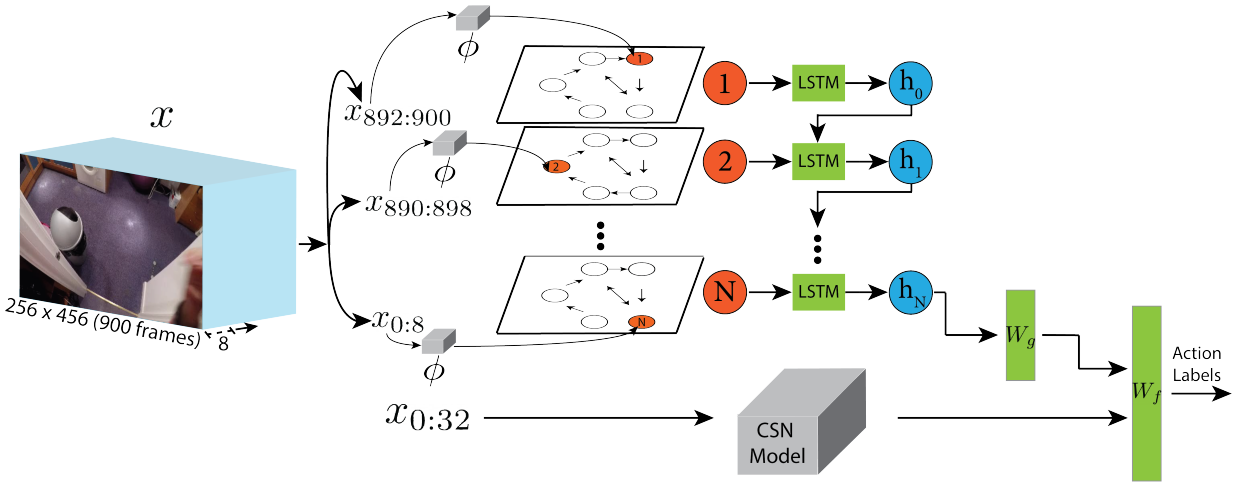


Figure 3.4: Overview of Ego-OMG's architecture. Ego-OMG consists of two streams: 1) The top stream consists of the extraction of a discretized sequence of states from an unconstrained egocentric video clip x of 900 frames using the Contact Anticipation Network Φ. The nodes predicted by Φ are embedded through GCN layers and then fed to an LSTM. This is then followed by a 1-layer MLP W_g to generate softmax scores for the anticipated future action. 2) The second stream generates softmax scores for the anticipated future action through feeding a short history (the last 32 frames of x) of video to a CSN model. A 1-layer MLP W_f processes the concatenated L2-normalized softmax scores to perform action anticipation and prediction.

list of entities, by restricting ourselves to the modelling of objects either currently or expected to be in contact with the hands, we are able to rule out 'background' objects that play no role in the actions involved, effectively using the hands as an attention mechanism. Furthermore, by aggregating contact based representations over larger timespans, we are able to model longer term structure of activity, whereas other approaches [58,59,81] are centered on visual appearance and short term dynamics on the order of $1-2$ seconds.

3.2 Method

In this section we introduce our method for action understanding. Through contact and activity modeling our approach seeks to anticipate partially observed and/or near-future action. The structure of our approach is shown in Figure 3.2. Input video is fed first into the Anticipation

Module - which we denote Φ - from whose output we produce symbolic state representations to be fed through Ego-OMG, which in turn anticipates partially observed and/or near-future action.

For the task of action anticipation, the observation of the video segment spans a range preceding the action start time τ_s by observation duration t_o, and ends t_a seconds before τ_s, where t_a is the anticipation offset. In other words, input clips span from time $\tau_s - (t_o + t_a)$ seconds to end time $\tau_s - t_a$ seconds. For the task of action prediction, input clips span from time $\tau_s + p(\tau_f - \tau_s) - t_o$ to $\tau_s + p(\tau_f - \tau_s)$, where τ_f is the end time of the action and p is the observable proportion of the clip containing the action to be predicted.

For our focus on hand-object contact in action modeling we devote the Anticipation Module. The Anticipation Module produces pixel-wise mappings of anticipated hand-object contact over the input. These mappings are divided into two types: Contact Anticipation Maps and Next Active Object Segmentations. The Contact Anticipation Network produces Contact Anticipation Maps, and is described in Section 3.2.2.2. The Next Active Object Network relies upon Contact Anticipation Maps for segmentation, producing Next Active Object Segmentations, and is described in Section 3.2.2.3. One advantage the Anticipation Module provides is that its mappings range over the near future action, and are not constrained to fixed anticipation time offsets as in several alternative action anticipation approaches [58, 82].

Training the Anticipation Module requires annotations for contact and localization of (next) active objects. To this end we augment the standard video data - in this work EPIC Kitchens - with temporal and segmentation information pertaining to contact. This process is described in Section 3.2.2.1.

We apply a Faster-RCNN [83] classifier over the maps produced by the Anticipation Module to produce symbolic states. These symbolic states capture characteristics of and relations between

hands and objects in a compact representation. Symbolic state representations allow for easy use, and representation of state relations.

For our focus on temporal relational structure we devote Ego-OMG. Ego-OMG represents relations between action states across multiple time ranges, and uses these relations in contextualizing the present moment, and in projecting to near future action.

A natural formalism for representing temporal relations is a graph. We employ a graph in Ego-OMG to represent action state relations, and embed graph nodes into Euclidean space through use of word embeddings and a Graph Convolutional Network. Details of this process are described in Section 3.2.3.

The remainder of Ego-OMG is as follows, and covered in detail in 3.2.3. The sequence of states derived from the input is represented through the dynamics of an LSTM applied over embedded state representations. This LSTM allows projection into the near future. Finally, the anticipated action produced by this LSTM is joined by visual and short term dynamic information produced by a conventional 3D CNN. This component of Ego-OMG is swappable with alternative action understanding methods, making Ego-OMG complementary to many existing action understanding frameworks.

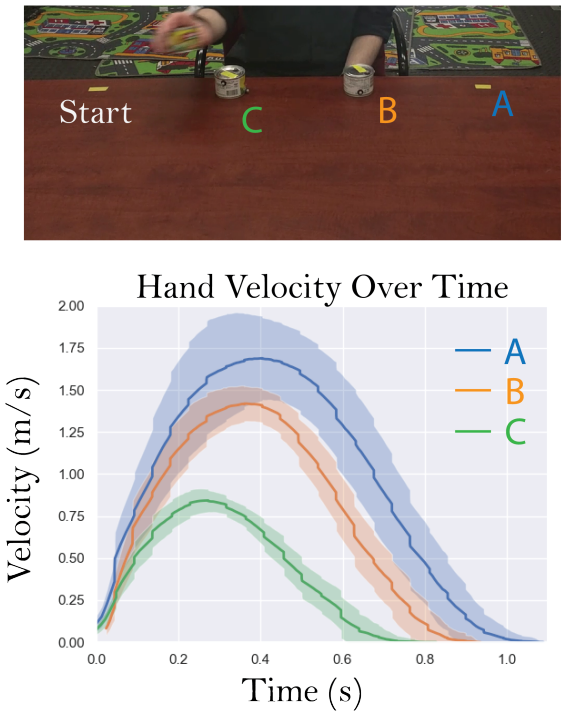


Figure 3.5: Jeannerod put forth the hypothesis that velocity profiles of reaching actions follow a well formed bell-shaped distribution [1]. We sample multiple hand trajectories involved in reaching motions in a lab setting using a Vicon motion tracking system. The three curves represent the velocity profiles of three separate trajectory distances at table positions A, B and C. Solid lines indicate mean velocity values across multiple runs; the upper and lower bounds of the shaded regions lie one standard deviation from the mean.

3.2.1 Reaching Experiments

To illustrate the velocity profile of reaching in hand manipulation actions we gathered position information through a Vicon motion capture system of simple reaching actions. The Vicon system gives sub-millimeter precision 3-dimensional positional information of tracked objects. Figure 3.5 illustrates a typical velocity profile of a reaching action. Our observations of velocity profiles are consistent with Jeannerod's hypothesis [1].

3.2.2 Anticipation Module

In Section 3.2.2.1 we describe the methods behind the collection of our dataset used for training the Anticipation Module, where the dataset consists of clips carefully selected from the EPIC Kitchens dataset. In Section 3.2.2.2 we introduce the Contact Anticipation Network and in Section 3.2.2.3 the Next Active Object Network, the two components that together form the Anticipation Module.

3.2.2.1 Dataset

We collect our dataset by organizing clips that correspond to point-to-point hand movements, where the hand involved and the Next Active Object are visible. The temporal boundaries of each clip are set such that clips begin when both the Next Active Object and the hand(s) targeting the object are visible, and end when the hand makes contact with the Next Active Object. As such, the lengths of the collected clips vary in the temporal dimension.

Rather than uniformly sample clips across all actions, we instead narrow our dataset to hand movement driven actions (i.e. *take*, *move*, *cut*, *open*) in the EPIC Kitchens dataset, as these actions each contain meaningful transitions in object status and encode the hand intentionality we wish to capture, making for a total of 2.1K randomly sampled clips with 252 unique object categories.

We croudsource our annotations on Amazon Mechanical Turk, asking workers to, for every 4 frames of a given clip, a) select the hand(s) involved in the given action and trace the Next Active Object, producing Next Active Object Segmentation masks $\{\Psi_l, \Psi_r\}$, and b) trace the left and right hands of the person and the objects held by each hand, creating contact segmentation

masks $\{\Gamma_l, \Gamma_r\}$. We generate dense supervision of video using the forward and inverse warping of optical flow obtained from TVL1 [84], projecting the annotations between annotated frames. That is, for flow displacements $u_{xy_1}, v_{xy_1} \in F^t_{t-1}$ and $u_{xy_2}, v_{xy_2} \in F^t_{t+1}$, values from each location (x, y) in the segmentation masks are copied to pixel locations $(x + \frac{1}{2}(u_{xy_1} + u_{xy_2}), y + \frac{1}{2}(v_{xy_1} + u_{xy_2}))$, for warped subsequent frames, for flow frames F.

To generate the Contact Anticipation Map supervision training signal C, for an annotated frame taken at time τ, t_c seconds away from the time-of-contact, we retrospectively assign each pixel belonging to the annotated Next Active Object the value of t_c. Pixels corresponding to the body of the person or objects held in the hand at time τ are assigned self-contact values of 0. All background pixels are populated with values of -1 and are not directly used during training. Figure 3.3 provides an illustration of this process.

To generate the Next Active Object binary segmentation masks A, we simply assign pixels belonging to the near-future contacted object values of 1, and assign values of 0 to all other pixels.

The Contact Anticipation Maps and the Next Active Object Segmentations each consist of two separate pixel-level channels $C = \{C_r, C_l\}$ and $\Psi = \{\Psi_r, \Psi_l\}$ respectively, for the right and left hands. In clips involving bi-manual manipulation, the Contact Anticipation Maps for the channels of each hand differ due to the different timings underlying the movement of each hand. However, the Next Active Object masks are shared between the channels of each hand, or $\Psi_l = \Psi_r$.

3.2.2.2 Contact Anticipation Network

The Contact Anticipation Map predictions require the modelling of short-term dynamics for capturing the underlying hand trajectory and the localization of boundaries pertaining to hands and objects in contact. To capture both, we devise a custom two-stream architecture: One stream consisting of 3D Convolutions applied over the input video for modelling short-term dynamics, and another consisting of a U-Net stream applied over a single frame belonging to the end of the observation for capturing more precise hand segmentations. See Φ in Figure 3.2 for an illustration.

The Contact Anticipation Network (see Φ part a in Figure 3.2) takes a stack of 8 sequential RGB frames and outputs four channels: Two pixel-level regression outputs $\{D_l, D_r\}$ corresponding to the estimated remaining time-to-contact for each pixel in the image, and two soft segmentation maps. We threshold the soft segmentation maps to arrive at binary segmentation masks $\{\hat{\Gamma}_l, \hat{\Gamma}_r\}$, containing pixel-level segmentation masks of hands and objects in contact with the hand. The two output channels in both cases are for the predictions separately designated for the left and right hand, respectively, each of size $(128, 228)$.

The 3D Convolutional stream is a standard 3D ResNet50 architecture, where the backbone network from [2] is utilized. It consists of 5 successive 3D Convolutional layers, where the first and third layers are followed by 3D Max Pooling operations. The UNet stream is composed of the exact architecture proposed in [85], where a contractive path (two 2D Convolutions followed by a 2D Max Pooling operation) is followed by the expansive path (2D Transposed Convolution layers followed by 2D Convolutions). The network is trained using ADAM with a learning rate of 0.0001 and a decay of $5\text{e}{-6}$. We apply ResNet-style normalization, and augment the input

RGB video with standard crops, flips, and color jitters.

There are two loss components used in training the Contact Anticipation Network. The first component, L_{MAE}, is the pixel-wise mean average error between predictions $\{D_l, D_r\}$ and ground truth $\{C_l, C_r\}$, only over pixel locations (x, y) where $C_{l_{xy}} > 0$ and $C_{r_{xy}} > 0$. In other words, this loss component is only computed over pixels belonging to the Next Active Object; other pixels do not have time-to-contact annotations, and so they are ignored. The second component, L_{BCE}, is the binary cross entropy loss between the predicted soft segmentation maps and contact segmentation masks $\{\Gamma_l, \Gamma_r\}$. The loss used to train the system is as follows: $L = L_{BCE} + \gamma L_{MAE}$, where $\gamma = 0.2$.

We predict pixels of contact $\{\hat{\Gamma}_l, \hat{\Gamma}_r\}$, where $\hat{\Gamma}_{s_{xy}} = 1$ for hand side $s \in (l, r)$ if pixel location (x, y) corresponds to a hand or object in contact and $\hat{\Gamma}_{s_{xy}} = 0$ otherwise. To arrive at the Contact Anticipation Maps, we superimpose the predicted pixels of contact $\{\hat{\Gamma}_l, \hat{\Gamma}_r\}$ over the regressed time maps $\{D_l, D_r\}$, for each hand side, to arrive at Contact Anticipation Maps $\hat{C}$, as follows:

$$\hat{C}_{l_{xy}} = \begin{cases} 0 & \text{if } \hat{\Gamma}_{l_{xy}} = 1 \\ D_{l_{xy}} & \text{if } \hat{\Gamma}_{l_{xy}} = 0 \end{cases}$$

$$\hat{C}_{r_{xy}} = \begin{cases} 0 & \text{if } \hat{\Gamma}_{r_{xy}} = 1 \\ D_{r_{xy}} & \text{if } \hat{\Gamma}_{r_{xy}} = 0 \end{cases}$$

The final Contact Anticipation Maps $\{\hat{C}_l, \hat{C}_r\}$ are fine-grained distributions of non-negative continuous values for each pixel that represents the estimated time of contact. Each of the

channels associated with the left and right hand are of size $(128, 228)$.

3.2.2.3 Next Active Object Network

As illustrated in Φ part b of Figure 3.2, the 8 frame RGB video and 8 frame Contact Anticipation Map history are fed in parallel through 3D Convolutions, after which a summation over the stream is performed. Additionally, the final frame of the 8-frame input is fed into a U-Net architecture in order to capture more precise object segmentations. Next, a pixel-wise multiplication between the resultant feature map from the 3D Convolutional streams and the output of the U-Net model is performed. The result of this multiplication is fed through sigmoid activation units, producing soft segmentation maps for the right and left hands, which are binarized using a threshold of 0.15 to arrive at $\{\hat{\Psi}_r, \hat{\Psi}_l\}$.

Each of the two 3D Convolutional streams have architectures identical to those used in the 3D Convolutional stream in 3.2.2.2. Likewise, the U-Net stream is identical to that of 3.2.2.2. The final output of the combined streams is of size $(128, 228)$. The network is trained using ADAM with a learning rate of 0.0001 and a decay of $5\text{e}{-6}$. We utilize a weighted binary cross entropy loss function between ground truth $\{\Psi_l, \Psi_r\}$, and predictions $\{\hat{\Psi}_l, \hat{\Psi}_r\}$ with a weight value of 2.0 chosen to overcome the foreground/background class imbalance in the ground truth Next Active Object masks of the collected dataset.

To avoid overfitting on the Contact Anticipation Map stream, multiplicative Gaussian Noise sampled independently over each pixel is applied over the output of the Contact Anticipation Map stream, adding $\hat{C}_i \odot Z_i$ where $Z_i = \mathcal{N}(\mu, \sigma^2)$, where $\odot$ is the element-wise Hadamard product. This augmentation captures the inherent ambiguity of anticipating contact; there is little

ambiguity in predicting the time values of pixels belonging to hands or contacted objects due to their proximity (by definition having time-to-contact of 0), while there is increasing ambiguity in predicting time-to-contact for objects the further from the hands they are. We apply ResNet-style normalization, and augment the input RGB video with standard crops, flips, and color jitters.

3.2.3 Ego-OMG

As illustrated in Figure 3.4, we feed input video x into the Anticipation Module Φ, whose purpose is to predict and anticipate hand object contacts. Current predicted and future anticipated contact is represented through a 4 channel output, consisting of two contact segmentation masks $\{\hat{\Gamma}_{t_r}, \hat{\Gamma}_{t_l}\}$ produced by the Contact Anticipation network and two object segmentation masks $\{\hat{\Psi}_{t_r}, \hat{\Psi}_{t_l}\}$ produced by the Next Active Object network, where $\hat{\Psi}_{t_r}$ and $\hat{\Psi}_{t_l}$ denote the predictions of the Next Active Object, and $\hat{\Gamma}_{t_r}$ and $\hat{\Gamma}_{t_l}$ denote the objects detected to be presently in contact with the hand, both for the right and left hands respectively. We classify each segmentation frame with a pre-trained Faster-RCNN [83] model, arriving at predicted object classes $o_t = \{\psi_{t_r}, \psi_{t_l}, \gamma_{t_r}, \gamma_{t_l}\}$. We note that for the purposes of this work we predict up to 1 object each for ψ_{t_r}, ψ_{t_l}, γ_{t_r}, and γ_{t_l}. This limitation prevents us from modelling scenarios where multiple objects are held by the same hand for tasks requiring dexterous manipulation.

In practice, while the Contact Anticipation Network succeeds at localizing contacted objects, the classifier tends to mis-classify currently held objects due to the severe occlusion imposed by the hand, especially for small objects like scissors and utensils. Therefore, in building the graph we impose the constraint that every object currently contacted by each hand *must have been anticipated* at some previous instance in time, before the presence of occlusion. In classifying

the objects currently in contact with the hand, we take the intersection of top-5 object class predictions for that object with the object classes previously predicted in anticipation over the past 100 frames (7 seconds).

In this section we define Ego-OMG, a two-stream architecture dependent on a novel graph representation G that consists of a structured sequence of high-level states extracted from videos belonging to the EPIC Kitchens dataset. The graph G contains two types of nodes: 1) nodes spanning current contact and forecasted contact of hands and objects, which are produced by the anticipation module in (ref), and 2) nodes corresponding to action labels in the EPIC Kitchens dataset. The graph G consists of edges connecting state-to-state transitions and state-to-action co-occurence.

Section 3.2.3.1 details the two-streams of Ego-OMG: the first modeling temporal relations and context, and the second modelling visual appearance and short-term dynamics through use of a 3D CNN. Section 3.2.3.2 explains the construction of the graph of Ego-OMG used in producing structured video representations.

3.2.3.1 Joint Architecture

The architecture of Ego-OMG is shown in Figure 3.4. Input consists of a single clip spanning 30 seconds - or 900 frames. The output consists of a logit layer predicting the class of the action τ_a seconds after the end of the observation. The architecture is comprised of two streams: One modeling the appearance and short term dynamics of the last few seconds of the clip; the other modeling hand dynamics and long-term semantic temporal relations.

In the first stream, we model appearance and short-term dynamics with a Channel-Separated

Convolutional Network (CSN), a 3D CNN factorizing 3D convolutions in channel and space-time in similar fashion to Xception-Net [86] which factorizes 2D convolutions in channel and space. The weights are pre-trained on the largescale IG-65M video dataset [87]. The network takes as input 32 frames of size 256×256. We apply horizontal flipping, color jittering and random crops during training, with centered crops during testing. The model is trained using SGD with a batch size of 16, a learning rate of 2.5×10^{-3} and a momentum of 0.9.

In the second stream we model dynamics of interactions between hands and objects, as well as longer term temporal semantic relations between the actions of the activity. We capture this structure in the form of a graph, described in detail in Section 3.2.3.2. After computing the graph, we feed it through two graph convolution layers of hidden layer size 256 and 128 respectively. Note our application of the GCN is transductive; it is applied on a single, fixed graph consisting of all nodes seen during train *and* test time beforehand. We feed the sequence of node embeddings obtained by the GCN into an LSTM [88]. At test time, we convert an input video of 900 frames to a sequence of states and from each state's respective node embedding g_n for $n \in N$, we aggregate the state history with a 1-layer LSTM. From the LSTM's final hidden state h_N, we apply a 1-layer MLP W_g to classify the next most likely action. The LSTM carries hidden states of size 128. A batch size of 16 and a learning rate of 7×10^{-5} is used with ADAM optimizer and a cross entropy loss function. Training achieves fast convergence, reaching peak top-1 action anticipation and action prediction accuracy after 5 epochs or roughly 0.25 hours of training on a NVIDIA GeForce GTX 1080 GPU.

We concatenate the L2-normalized softmax scores from each respective stream, freezing the two sub-networks while training a 1-layer MLP W_f with a batch size of 16 and learning rate of 0.01 on top of the joint softmax scores to classify the next most likely action. We find a

late fusion approach provides slight benefits in practice as opposed to an early fusion of the two streams, likely due to the different learning dynamics of the individual streams. Inference times are dominated by the CSN model.

3.2.3.2 Graph Construction

We have a set of K training videos. To detect the objects involved in interaction, which are needed to build the graph, we utilize both sub-components of the Anticipation Model Φ, described in subsections 3.2.2.2 and 3.2.2.3. The Anticipation Module Φ iterates over each video using a sliding window with an 16-frame width, sampling every 2 consecutive frames with a stride of 2. Feeding each of 4 output channels of Φ to the object classifier then produces detections $O_i = \{o_1, o_2, ..., o_{T_i/2}\}$ for video i, where T_i is the frame count of video i. From the per-frame predictions of the object classes o_i, we suppress consecutive duplicate predictions arriving at non-consecutively repeating states $S_k = \{s_1, s_2, ..., s_n\}$, a sequence where temporal order is preserved.

With the input to graph construction defined, we now consider the graph $G = (V, E)$, where E consists of the set of all edges, and V consists of the set of all nodes. $V = \{V_s, V_a\}$ consists of nodes of two types: state nodes, and action nodes. State nodes consist of the union of all S_k, that is: $V_s = \bigcup_{k=1}^{K} S_k$, and action nodes V_a consist of the set of all action classes $a_i \in A$, where A is the set of all actions. In doing so, we represent both states and actions in graph G.

We construct the adjacency matrix as follows. Each node has an edge connecting it to itself: $e_{ii} \in E$ for $1 \leq i \leq |V|$ with weight 1. We add weighted directed edges $e_{ij} \in E$ for consecutive states s_i and s_j for $0 \leq i < n$ and $j = i + 1$, where the weight σ_{ij} is transition probability

$p(s_{i+1}|s_i)$ where transition probabilities are observed from transitions in state sequences S_k for all $k \in K$. We also add weighted directed edges between states and actions by adding weighted edge $e_{ij} \in E$ if action i takes place within the timespan of state s_j, where weight σ_{ij} is equal to $p(a_i|s_j)$.

Graph G has a total number of nodes equal to the number of unique states $z = S+A$, where S is the set of unique states and A is the set of annotated actions. Let $X \in R^{z \times m}$ be a matrix containing all z nodes with their corresponding features of dimension m. Rather than set X to identity matrix I, we initialize each node with feature embeddings extracted from a pre-trained GloVe-600 model [89]. When representing states $s \in S$, we average the feature embeddings from each object noun in s. When representing actions $a \in A$, we average the embeddings for the verb and noun embeddings. We find that utilizing pretrained word embeddings for G results in substantial performance gains over using $X = I$.

We feed the weighted adjacency matrix and X as input into the GCN as described in Section 3.2.3.1.

3.3 Dataset Creation

3.3.1 Datasets

3.3.1.1 Collection

We collect our dataset by organizing clips that correspond to point-to-point hand movements, where the hand(s) involved and the Next Active Object are visible. The clips are croudsourced onto the Amazon Mechanical Turk platform, where $103K$ responses were collected from 162

workers, resulting in $45K$ final responses after inspecting annotations of each worker and preventing workers whose jobs did not meet a satisfactory level of performance from submitting further responses. Workers are compensated $\$0.03$ for each submitted response. The annotations for the body and objects in contact are croudsourced separately from the annotations for the Next Active Object, with the interface for both shown in Figure 3.6. In the end, a single annotation per sample is collected and used for the training of the Anticipation Module.

3.3.1.2 EPIC Kitchens

The EPIC Kitchens dataset [43] is a large egocentric video dataset, captured by 32 subjects in 32 different kitchens. The videos consist of daily kitchen activities where participants were simply asked to record their interactions in their native kitchen environments (no instructional scripts were given to the subjects). Each video contains several annotated action segments, each associated a $(verb, noun)$ action label, where there are 125 unique verbs and 352 unique nouns, where $2,513$ unique actions are present in the dataset. In addition, there are $400K$ bounding box annotations of objects over the entire dataset. We conduct all quantitative experiments over this dataset.

3.3.1.3 EGTEA Gaze+

The EGTEA Gaze+ dataset [50] is another egocentric video dataset, captured by 32 subjects, where participants are given instructions to prepare meals. To demonstrate the generalization capabilities of the Anticipation Module to other egocentric datasets, we provide the outputs of the Anticipation Module pre-trained on EPIC Kitchens over the entirety of the EGTEA Gaze+

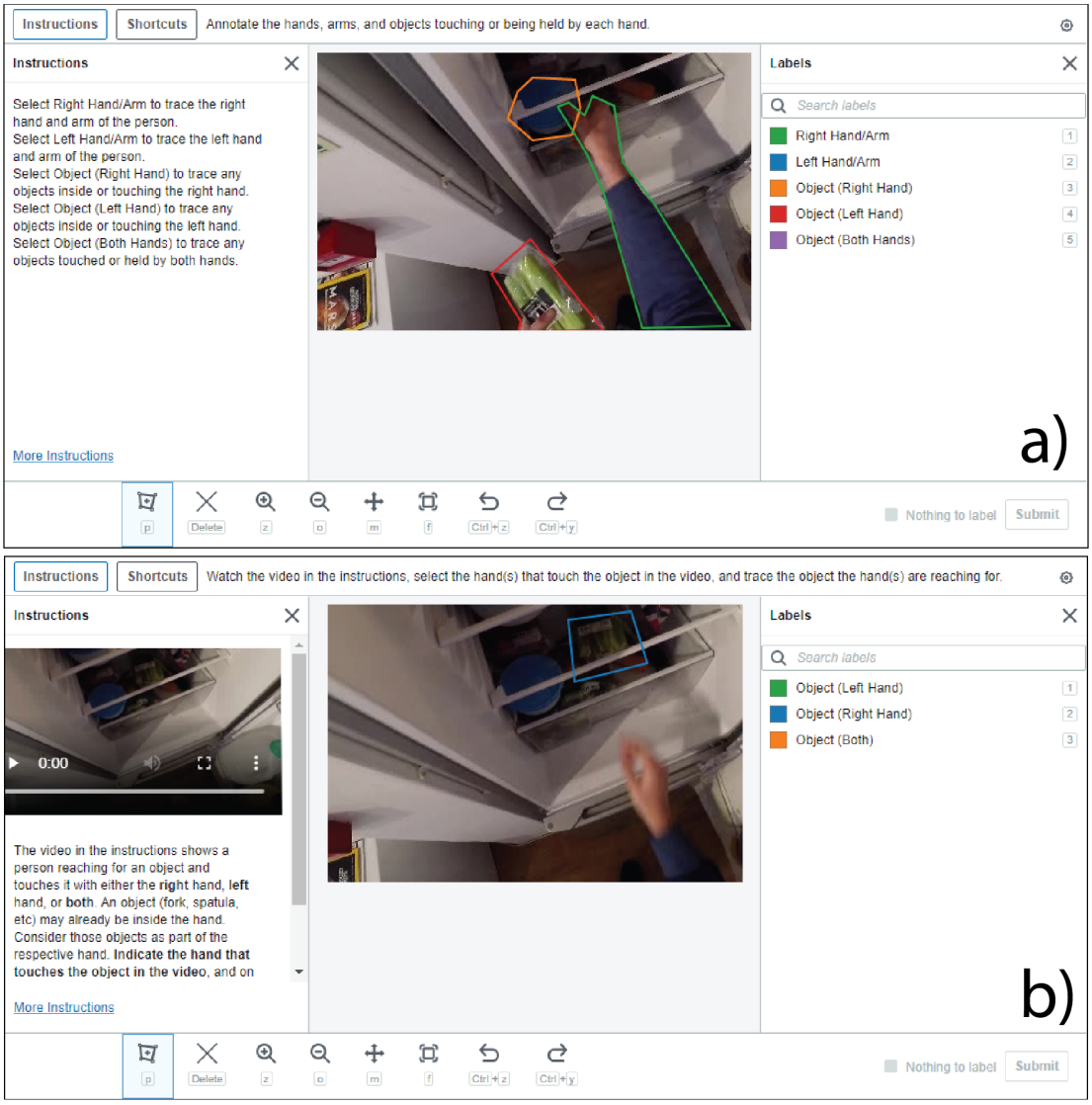


Figure 3.6: Our interfaces for collection of annotations over the Amazon Mechanical Turk platform, where **(a)** is the interface for the collection of the body/contacted objects, and **(b)** is the interface for the collection of the Next Active Object annotations. In **(a)**, workers are instructed to annotate all instances of hands and objects in contact by selecting the appropriate label in the right panel and tracing the body of the hand or object involved. In **(b)**, workers are instructed to first view the full video of the point to point hand movement, as the Next Active Object annotation relies on information only available in the future. They then select the side(s) involved in the reaching movement in the right panel and trace the body of the object involved.

dataset. A Google Drive link is provided[1]

[1] Google Drive link at : https://drive.google.com/drive/folders/1AIZ93d37g0mJaHclANhXYVyp2jFQtfCS?usp=sharing

3.4 Experiments

Throughout these experiments we evaluate the performance of the proposed models for action anticipation, action prediction, and Next Active Object prediction. We also perform ablations over the components of the Anticipation Module to understand their respective contributions to the success of the entire framework.

	Method	Top-1	Top-5
S1	2SCNN (RGB) [90]	4.32	15.21
	TSN (RGB) [91]	6.00	18.21
	TSN + MCE [82]	10.76	25.27
	RULSTM [62]	15.35	35.13
	Camp. et al. [92]	15.67	**36.31**
	Liu et al. [63]	15.42	34.29
	Ours	**16.02**	34.53
S2	2SCNN (RGB) [90]	2.39	9.35
	TSN (RGB) [91]	2.39	9.63
	TSN + MCE [82]	5.57	15.57
	RULSTM [62]	9.12	21.88
	Camp. et al. [92]	9.32	23.28
	Liu et al. [63]	9.94	23.69
	Ours	**11.80**	**23.76**

Table 3.1: Action anticipation results on the EPIC Kitchens test set for *seen* kitchens (**S1**) and *unseen* kitchens (**S2**) during the EPIC Kitchens Action Anticipation Challenge. Only published submissions are shown.

3.4.1 EPIC Kitchens Action Anticipation Challenge

The protocol behind the EPIC Kitchens Action Anticipation Challenge is to set the anticipation time τ_a to 1 second. While there are 44 participants in the challenge, we report our results alongside the top 3 published submissions (**RULSTM** [62], **Camp et al.** [92], **Liu et al.** [63]) and include the benchmarked action anticipation results from the EPIC Kitchens dataset release

h

Table 3.2: Action anticipation and action prediction accuracy results over validation set for CSN stream, GCN stream and CSN + GCN stream over varying anticipation times τ_a seconds and varying observation rates p.

	Action Anticipation (τ_a)						Action Prediction (p)				
	5	2.5	1.5	1	0.5	0	12.5	25	50	75	90
CSN	6.49	11.39	14.09	15.50	18.61	19.37	24.23	26.49	30.72	31.08	31.30
GCN	9.05	10.47	11.31	12.81	13.76	14.56	14.83	15.44	15.70	15.88	16.01
CSN + GCN	**9.44**	**15.01**	**17.02**	**19.20**	**20.29**	**21.89**	**26.01**	**28.33**	**31.14**	31.19	31.42
RULSTM [62]	6.98	10.92	12.31	12.69	16.98	18.21	24.48	27.63	30.93	**33.09**	**34.07**

(**2SCNN** [90], **TSN (RGB)** [91], and **TSN + MCE** [82]). See the section 3.4.2 for details of each baseline.

Table 3.1 shows our results over the test set (S1) where scenes appear in the training set and over the test set (S2) where scenes are **not** included in the training set. We are 2 place in S1, beating previous state-of-the-art methods by a margin of $.35\%$ and 1 place in S2, beating previous state-of-the-art methods by a margin of 1.86%. We posit that the reason for Ego-OMG's notable outperformance w.r.t previous methods in S2 is because previous methods rely heavily on visual appearance and are more likely to fail when testing on unseen kitchens which likely include objects of previously unencountered appearance; Ego-OMG's GCN stream on the other hand only models objects of interaction, ignoring the diverse, cluttered, backgrounds that typically make up everyday kitchen environments.

The final column of Table 3.1 contains the evaluation of all methods with Top-1 evaluation and Top-5 evaluation - a prediction is correct w.r.t. Top-5 evaluation if the ground truth is included in the Top-5 predicted actions. In our approach we utilize the same model for Top-1 and Top-5 evaluation measures; other approaches may train separate models for the two evaluation measures. Furthermore, we note that the other methods incorporate distinct training mechanisms

for top-5 action anticipation [82, 92].

We stress that the CSN stream can be swapped with any of the architectures listed above; as the CSN stream is outperformed by **RULSTM**, **Camp. et al.**, and **Liu et al.**, better performance could be expected from Ego-OMG with the incorporation of any one of these architectures. In addition, we do not perform any form of ensembling in our submission.

3.4.2 EPIC Kitchens Challenge Baselines

We demonstrate state-of-the-art performance over the recent EPIC Kitchens Action Anticipation Challenge, achieving 1st place on the EPIC Kitchens Action Anticipation Challenge unseen test set, and 2nd place on the seen test set, and outperform all previously published approaches without any use of ensembling, unlike many competing approaches. Below we describe the action anticipation results reported in Table

- **2SCNN** [90] is a two-stream CNN training 2D ConvNets on separate RGB and flow streams before performing a late-fusion.
- **TSN (RGB)** [91] is a temporal segment network trained on a single stream of RGB video.
- **TSN + MCE** [82] is a temporal segment network trained with a novel loss formulation to address the inherent ambiguity in action anticipation.
- **RULSTM** [62] is a state of the art architecture with two separate LSTMs and an attention formulation applied over features obtained from RGB, flow, and object detections.
- **Camp et al.** [92] is a loss formulation representing verbs and objects with pre-computed word embeddings to overcome the overwhelming number of unique actions in EPIC Kitchens.

- **Liu et al.** [63] is a joint architecture that relies on additional supervisory signals to predict motor attention and interaction hotspots before performing action anticipation. This is the closest work to our approach.

- **Ours** The combined Anticipation Module and Ego-OMG.

3.4.3 Action Anticipation and Prediction

We evaluate our approach over the EPIC Kitchens dataset on the tasks of action anticipation and action prediction over varying anticipation times τ_a (action anticipation) and varying observation ratios p (action prediction). The tasks are detailed in Section 2.2. See Table 3.2 for results. The purpose of these experiments is to analyze the performance of our approach and its components over degrading anticipation times and varying action observation ratios.

We vary the anticipation time τ_a from 0 seconds (predicting the action class immediately before its start) to 5 seconds (predicting the action class 5 seconds before its start). We perform action prediction at the following observation ratios p: 12.5%, 25%, 50%, 75% and 90%.

We compare our approach to the state-of-the-art RULSTM [62] work due to its state-of-the-art performance over published methods in both action anticipation and action prediction, making RULSTM the optimal baseline.

For action anticipation, we note the performance of our approach degrades gracefully as anticipation time τ_a increases. As the anticipation time increases, the performance of the CSN stream drops off considerably, to the point where at $\tau_a = 5$ seconds, the GCN stream outperforms the CSN stream by a large margin of 2.56%. Full Ego-OMG outperforms each of its streams individually over all τ_a.

For action prediction, we observe diminishing gains of the GCN stream's contribution to the performance of Ego-OMG as the observation ratio p increases, to the point where RULSTM outperforms our approach after $p = 50\%$. We also observe the performance of our approach at observation ratios $p = 50\%$ and $p = 90\%$ are very close. These findings lead us to the conclusion that the strength of our approach lies in its anticipatory capabilities, and that our approach is not particularly better suited for the action recognition setting over other methods focusing on visual appearance and short-term dynamics. However, we hypothesize that the incorporation of flow into our approach would be of great benefit for the action recognition setting.

3.4.4 Next Active Object

In this section we evaluate the performance of the Anticipation Module on the prediction of the Next Active Object. We conduct two evaluations: The first being the evaluation of the localizations produced by the Anticipation Module, and the second being the evaluation of the classification over those produced localizations. Both evaluations are performed frame-wise over the test set of the augmented dataset. To illustrate the generalization of the Anticipation Module to other egocentric activity datasets, we provide the outputs of the Anticipation Module over the EGTEA Gaze+ dataset on Google Drive[2], where the Anticipation Module was trained over EPIC Kitchens. In addition, outputs of the Anticipation Module over both EPIC Kitchens and EGTEA Gaze+ are shown in Figure 3.7.

3.4.4.1 Localization

The first set of Next Active Object evaluations is performed with respect to the ground truth segmentation masks included in the augmented dataset described in 3.2.2.1. We report

[2] Google Drive link at : https://drive.google.com/drive/folders/1AIZ93d37g0mJaHclANhXYVyp2jFQtfCS?usp=sharing

Evaluation	Jaccard
Obj-Tracker	0.028
DeepGaze II	0.051
I3D-GradCam	0.079
Center Bias	0.088
Ours w/o CAM	0.169
Ours	**0.194**

Table 3.3: Evaluation of localizations produced by the Next Active Object predictions. Contact Anticipation Maps are referred to as CAM in the table.

Jaccard similarity as our evaluation measure. We provide an evaluation comparing baselines and an ablated and non-ablated implementation of our approach:

- **Center Bias** relies on the assumption that the Next Active Object most commonly appears near the center of the frame, and instantiates a fixed Gaussian at the center of each image of size $(55, 55)$ to represent the Next Active Object.

- **I3D-GradCam** trains an I3D model to perform action anticipation (τ_a sampled uniformly between 0 to 2 seconds) over the entire EPIC-Kitchens dataset, applying standard Grad-Cam [93] over the trained network to generate heatmaps containing the Next Active Object. We mask out the hands of the actor from the heatmaps using ground truth hand segmentations from 3.2.2.1 to better localize the Next Active Object.

- **DeepGaze II** [94] is a pre-trained state-of-the-art model performing saliency prediction over each image in the collected dataset.

- **Obj-Tracker** is a re-implementation of [74]. The SORT [95] tracking algorithm is applied over the detections obtained by Faster-RCNN pretrained over EPIC-kitchen dataset. Objects tracked for less than 20 frames are dropped, and remaining trajectories are classified as

Evaluation	Top-1	Top-5
Obj-Tracker	1.00	5.70
I3D Classifier	11.90	31.94
RULSTM	15.07	**39.88**
Ours	**18.26**	39.67

Table 3.4: Evaluation of classification accuracy with respect to the Next Active Object predictions.

either 'active' or 'inactive'. As predictions are bounding boxes, we convert the ground truth segmentation masks to bounding box format.

- **Ours w. and w/o CAM** are our proposed approaches with and without the incorporation of Contact Anticipation Maps in the Next Active Object Network to demonstrate the utility of Contact Anticipation Maps. Other approaches do not model left and right hands separately, and so for fair comparison we collapse our model's binarized two channel output for the left and right hands, into one channel.

Our approach including the Contact Anticipation Maps outperforms all baselines and the ablation by large margins. We attribute this to its rich encoding of hand trajectory. The performance of **Obj-Tracker** on the EPIC Kitchens dataset is poor compared to its performance in [74] over the AVL dataset. We observe that the tracker fails consistently in tracking objects over timespans exceeding $.5$ seconds. For the frames where the objects are tracked, next active object localization is reported.

3.4.4.2 Classification

The second set of Next Active Object evaluations is performed with respect to the annotated object classes available from the EPIC Kitchens dataset. We provide an evaluation comparing

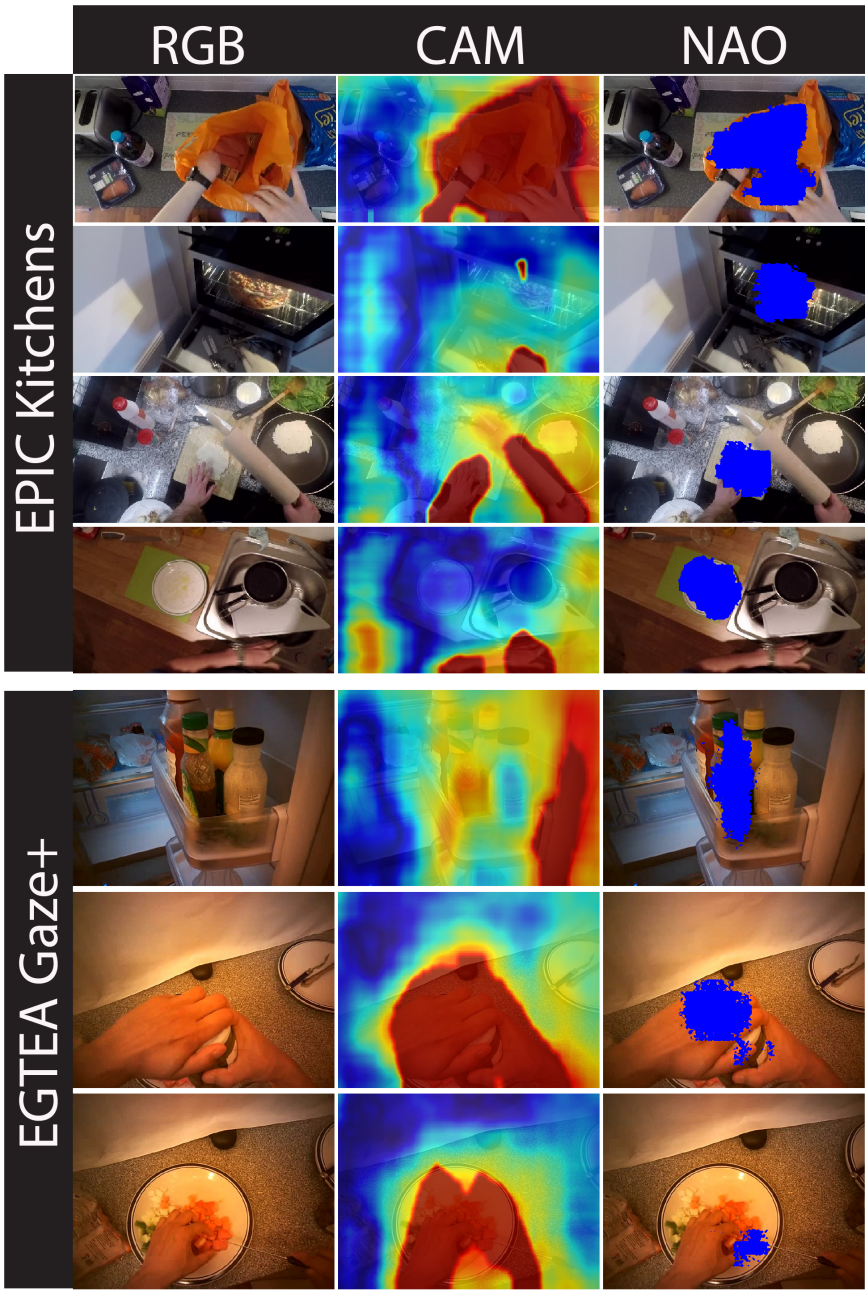


Figure 3.7: The Anticipation Module outputs Contact Anticipation Maps (second column) and Next Active Object segmentations (third column). The Contact Anticipation Maps contain continuous values of estimated time-to-contact between hands and the rest of the scene (visualizations varying between red for short anticipated time-to-contact, and blue for long anticipated time-to-contact). The predicted Next Active Object segmentations contain the object of anticipated near-future contact, shown in blue in the third column. Predictions are shown over the EPIC Kitchens and the EGTEA Gaze+ datasets.

three baselines and our approach:

- **I3D Classifier** trains an I3D model over the action segments of the EPIC Kitchens dataset, using the object noun labels as ground truth to perform end-to-end classification of the future object of interaction with action anticipation offset τ_a sampled uniformly between 0 to 2 seconds.

- **Obj-Tracker** is a re-implementation of [74]. See the baseline description at Section 3.4.4.1 for more.

- **RULSTM** [62] is a state-of-the-art architecture with two separate LSTMs and an attention formulation applied over features obtained from RGB, flow, and object detections. This, like **I3D Classifier**, is trained over the action segments from the EPIC Kitchens dataset.

- **Ours** computes the Intersection-over-Union values between the bounding boxes produced by a Faster-RCNN model trained over EPIC Kitchens and the predicted Next Active Object segmentation. The object category associated with the highest IoU is used as the prediction.

Our approach outperforms the **I3D Classifier** and **RULSTM** baselines at Top-1 Next Active Object prediction by large margins, and is only marginally outperformed by **RULSTM** at Top-5 Next Active Object prediction. We note that **RULSTM** and **I3D Classifier** are trained on input clips with action segments of temporal boundaries defined by the EPIC Kitchens annotations, whereas **Obj-Tracker** and **Ours** were trained on input clips with temporal boundaries defined by the augmented dataset. Also, **RULSTM** and **I3D Classifier** are trained over 10X the number of clips that our Anticipation Module is trained over (28K vs 2.1K respectively). We expect our approach could benefit from a larger augmented dataset.

3.4.5 Anticipation Module Ablations

We perform several ablation studies over the components of the Anticipation Module to understand their contributions towards their performance of the GCN stream in Ego-OMG for the tasks of action anticipation ($\tau_a = 1$ second) and prediction ($p = 0.25$).

3.4.5.1 Hand Representation

We compare the effects of having the Anticipation Module produce localizations for each hand individually vs. jointly. Our approach consisting of Active and Next Active Object predictions for each of the two hands is compared with **Joint**, where we treat both hands as one entity by superimposing the binarized segmentation channels produced for each hand. This ablation evaluates the extent to which distinctly modelling left and right hands benefits action anticipation and prediction. Other methods (e.g., [63]) typically do not differentiate between left and right hands. See the first two rows of Table 3.5 for results.

3.4.5.2 Next Active Object and Contacted Objects

We isolate the Active and Next Active Object predictions of $o_t = \{\psi_{t_r}, \psi_{t_l}, \gamma_{t_r}, \gamma_{t_l}\}$, defined in Section 3.2.3. We train the GCN stream of Ego-OMG over ablations excluding elements of o_t. These ablations evaluate the extent to which inclusion of Next Active Object predictions and the inclusion of Active Object Predictions benefit Ego-OMG performance. See rows 3 and 4 of Table 3.5 for results.

Ablations	Anticipation	Prediction
Ours	**12.81**	**15.44**
Joint	12.12	14.63
AO only	10.91	13.96
NAO only	7.86	8.50

Table 3.5: Ablation experiments showing anticipation and prediction top-1 accuracy, performed over Anticipation Module components. Ours refers to non-ablated implementation; Joint collapses left vs. right hand distinctions; "AO Only" refers to "Active Object Only"; "NAO Only" refers to 'Next Active Object Only'.

	GCN	**No GCN**
GloVe Vectors	**12.81**	11.79
Identity Mat.	6.67	3.62

Table 3.6: Action anticipation accuracies over validation set with anticipation time $\tau_a = 1$ second, over GCN and GloVe embedding ablations.

3.4.6 GCN Ablations

In evaluating the utility provided by graph embeddings effected through GCN layers, we compare two versions of Ego-OMG: One where graph nodes V are embedded through a GCN, and the other where nodes V are left unaltered before the node sequence observed from the video clip is fed into the LSTM.

In evaluating the utility provided by word embedding when representing states s_i, we compare two versions of Ego-OMG: One where the initialization of input matrix X is set to features extracted from a pre-trained GloVe-600 model through methods discussed in Section 3.2.3.2, and the other where X is set to the identity matrix.

Table 3.6 illustrates results over joint ablations for the two sets of comparisons. The use of graph convolutions in conjunction with GloVe embeddings outperforms ablations.

3.5 Conclusion

This chapter discussed our work published in [10]. We explored another constituent of action using contact anticipation maps to predict the next active object segmentation maps. Directly predicting the hand trajectories can be challenging, and predicting the next active object is helpful in both action prediction and action anticipation tasks. We achieve state-of-the-art results over the EPIC Kitchens Action Anticipation Challenge - achieving 1st and 2nd place on the unseen and seen test sets, respectively - through feeding our representations through Ego-OMG, our state-of-the-art action anticipation and action prediction architecture. We release our predictions over the EGTEA dataset and provide ablation studies evaluating the utility of individual system characteristics.

Chapter 4: Diving Deep into the Motion Representation of Video-Text Models

Since the introduction of large-scale use of contrastive learning for image and text representation [96], various efforts have been made to build video-text models [97–100] to relate video to text. Videos provide a way to access the dynamics or motion in the scene that a single image cannot capture [17, 101, 102]. Motion in videos could be due to the action depicted, the effect of camera movement (for example, in egocentric action videos), or a combination of camera motion and action [103].

We investigate how existing video-text models perceive motion due to the action. For this work, we define motion in action videos as the movement of actors or the movement of actors and objects. The challenge is the lack of datasets that explicitly describe video motion. Figure 4.1 shows some examples of captions from ActivityNet [104] and the MSR-VTT [105] dataset. Although verbs are included in captions of multimodal datasets like ActivityNet, MSR-VTT, Howto100M [106], Spoken Moments in Time [107], a detailed description of motion is not available. This calls for the need to have an exclusive benchmark to evaluate how video-text models interpret and respond to motion descriptions.

We use the human action datasets Kinetics-400 [108], UCF-101 [109], and HMDB-51 [110] to circumvent the lack of quality annotations of motion descriptions. The advantage of action datasets is that for every action label, we can obtain the corresponding characteristic motion of

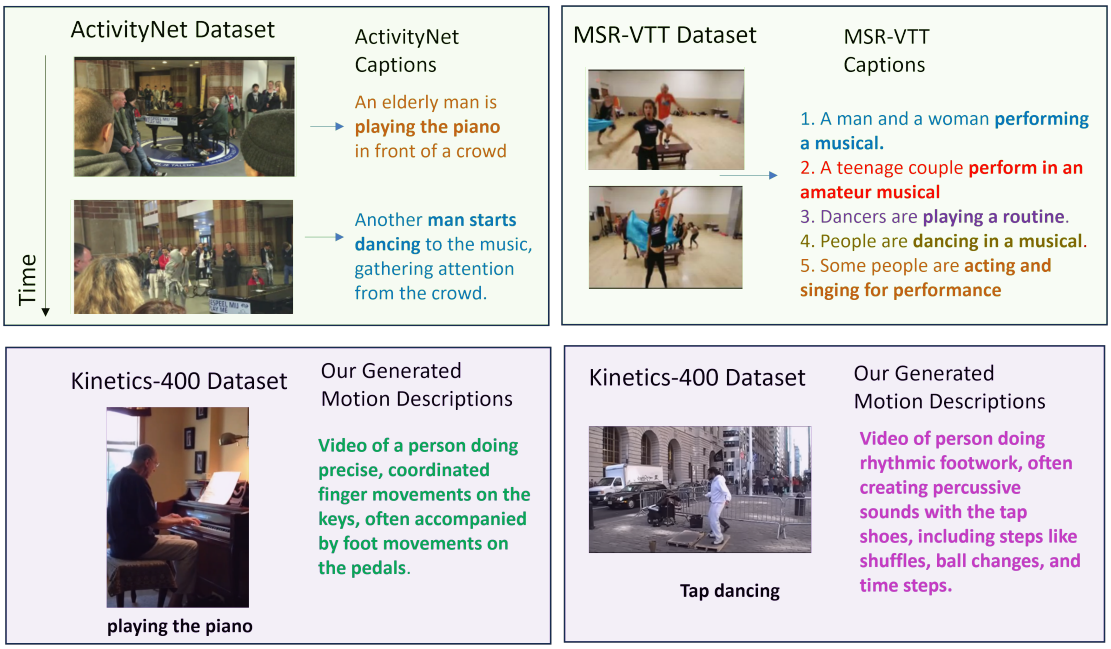


Figure 4.1: Example captions from ActivityNet, MSR-VTT, and our own GPT-4 generated fine-grained motion description for Kinetics-400 classes. Our generated motion descriptions solely describe the motion of the action, whereas other datasets typically use verbs to describe the scene.

the action by using large language models like GPT-3 [111] and GPT-4 [112], which, to a large degree, produce accurate descriptions of the all the actions in the datasets. Figure 4.2 shows some examples of the descriptions generated by GPT-4 and corresponding videos and original captions.

We evaluate several video-text models on the motion description retrieval task using the HMDB-51 and the UCF-101 datasets. We compare against human performance and show that all models fall far behind, raising questions about the design of video-text models and the role of quality captions in training better models to capture human motion. To address this question, we propose a method described in section 4.3 to investigate if providing better motion description captions helps video-text models understand fine-grained motion descriptions. To validate this fairly, we compare our method with video-text models that similarly initialize their video encoder and text encoder with pre-trained CLIP [96] weights so that undue performance gain is not

obtained by pre-training on video-text data. Our results show that our proposed pipeline is very effective in learning fine-grained motion descriptions on both the UCF-101 dataset and the HMDB-51 dataset. In summary, our contributions are:

1. Creating a dataset of human motion descriptions for three action datasets.

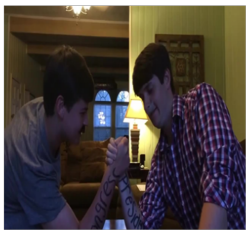
2. Evaluating current video-text models representing motion description in videos on the UCF-101 and HMDB-51 datasets against human expert evaluation.

3. Introducing a method to validate the need for better captioning in video-text models to understand motion descriptions and demonstrate the method's effectiveness in capturing fine-grained motion descriptions.

4.1 Background and related work

Multimodal datasets and video understanding tasks: Howto100M and Spoken moments in time are popular video caption datasets used in pre-training video-text models. ActivityNet, MSR-VTT, DiDeMo [113], VaTex [114] are representative datasets used for video-language alignment tasks like video-to-text retrieval or text-to-video retrieval. To our knowledge, we are the first to introduce fine-grained motion descriptions in video datasets, which is not the focus of existing datasets. We introduce motion descriptions on Kinetics-400, HMDB-51, and UCF-101 datasets.

Video-text models: Various efforts have been made to build video-text models [98, 99] mainly for video-to-text retrieval tasks. These models have developed mechanisms based on CLIP and extended them to video frames. Video-text models [97, 100, 115–117] have also been used for general video recognition both in the supervised and the zero-shot action recognition



Video of **arm wrestling** from Kinetics-400 dataset.

Characteristic motion of air drumming action is: doing rhythmic, exaggerated hand and arm movement. Characteristic motion of abseiling is: doing controlled descent down a vertical drop. Characteristic motion of swing dancing is: doing energetic, bouncy movements with lots of spins, kicks, and lifts. **Similarly, provide Characteristic motion of arm wrestling.**

Input Prompt to GPT-4

Video of a person doing **forceful downward pushing motion of one's arm against an opponent's arm.**

Generated motion description caption by GPT-4.

Figure 4.2: **Schematic representation of the generation of motion descriptions in existing action datasets.**

setting. [117] introduced verb-focused contrastive training to improve better verb reasoning by learning with hard negative verb examples. [118] introduced contrast sets to identify pitfalls in video-text models and recommended the need for fine-grained action understanding to tackle hard negatives in contrast sets. We differ from them as we utilize motion descriptions, which has its unique challenge. We compare our video-text model with Vanilla CLIP [96], XCLIP [97], Text4Vis [115] and VifiCLIP [116], primarily because they utilize the pre-trained CLIP for fair evaluation purposes.

4.2 Benchmark

Designing a motion description benchmark: We perform experiments on Kinetics-400, UCF-101, and HMDB-51 datasets. For each class in these action datasets, we prompt GPT-4 to produce the characteristic motions of the action (given by the caption annotation). Figure 4.2 shows the overall process of obtaining the motion descriptions. More details about the dataset statistics and generation are described in section 4.8. The dataset can be accessed at `https://github.com/chinmayad/motiondescriptions.git`

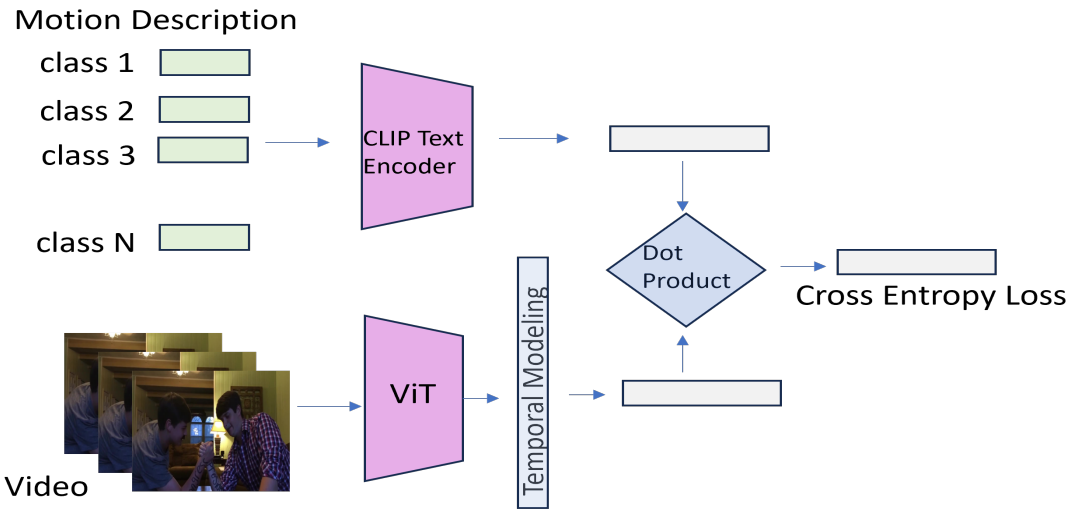


Figure 4.3: **Schematic representation of our approach encoding motion description in video-text model pipeline:** We integrate motion information as classifier weight in a supervised training paradigm. We finetune the image encoder to integrate the motion information while classifying videos in the kinetics400 dataset.

4.3 Proposed method

This section describes our proposed video-text model that incorporates the textual motion description. A typical vision-language model like CLIP is usually trained using contrastive learning loss or its variations [117, 119]. The quality of representations learned from the video-text model [120] will depend on the captions used during pre-training. Since no large video-caption dataset containing motion descriptions exists, we can't directly train a video-text model using contrastive learning. We, therefore, propose a method that utilizes the rich linguistical understanding of motion in actions from GPT-4 to give us the captions required for training. Our approach has two parts: Generating the required motion descriptions of actions using GPT-4 described in section 4.2 and training video-text models to utilize these motion descriptions described in section 4.3.2.

4.3.1 Problem setting

The model is trained on kinetics-400 as source dataset D_s and tested on target datasets D_t: UCF101 and HMDB51. The source dataset D_s consists of videos x_s and labels L_s belonging to classes C_s. The target dataset D_t consists of videos x_t and labels L_t belonging to classes C_t. We use a zero-shot setting such that $C_s \cap C_t = \emptyset$. Let the generated motion descriptions from GPT-4 for L_s and L_t be M_s and M_t respectively.

4.3.2 Architecture setting

Figure 4.3 gives an overview of our approach. While training, we freeze the text encoder from CLIP and fine-tune the visual encoder to learn the motion representation. While testing, motion descriptions are passed through the network to obtain classifier weights. Given a video from the target dataset, we obtain the logits indicating the probability that a video matches the corresponding motion description. A detailed discussion on training, testing, and implementation details is given in section 4.4.

4.3.3 Theoretical justification

We provide a theoretical justification for our proposed approach. Consider a large-scale dataset D containing visual samples with ground-truth labels. Denoting our labeled dataset $D = (x1, y1), (x2, y2), ...$, let X represent input data $x1, x2, ...$ and Y represent the labels $y1, y2, ...$

A typical supervised learning framework with a linear predictor involves minimizing $L(X^T W, Y) + Sigma(W)$ where W contains the parameters to be learned, $Sigma$ is the regularizing function, and L is the loss function. Here, W is learned independently on D and will not be helpful for

new classes or other downstream datasets. As proposed in [45], to make the approach tractable for zero-shot learning, we need to make W so it carries valuable information for new classes. The authors of [45] introduce

$$W = VS^T, \tag{4.1}$$

which we refer to as equation 4.1, where S is the signature of classes obtained from attributes of classes in D, and V is a new set of parameters to be learned.

For our scenario, we want to fine-tune the video-text model on the Kinetics dataset so that it can learn the motion description.

Let us denote V_E as the visual encoder from CLIP or any pre-trained video-text model. The supervised learning formulation to learn V_E^* and W_{proj}^* as in [115] can now be represented in minimizing cross-entropy $H(y|\sigma(W_{proj}.V_E(x)))$ referred to as equation 4.2, where $H(p*|p)$ stands for the Cross Entropy between the predicted distribution p and the ground-truth distribution $p*$. σ denotes the softmax operation, $W_{proj} \in R^{c\times d}$ denotes the linear projection matrix for classification where c is number of classes and d is the dimension of embedding from V_E. The above formulation in equation 4.2 is a standard visual feature transferring paradigm, where the visual encoder V_E. and the projection matrix (classifier) W_{proj} are learned simultaneously. We need to introduce a motion description to make the formulation in equation 4.2 learn the motion description and be useful for recognizing new motion descriptions for zero-shot settings.

Inspired by equation 4.1 where $W = VS^T$, we introduce motion description by making W_{proj} the signature of classes S, and V_E the new set of parameters of the visual encoder to be learned. In [45] S was obtained from an attribute matrix, and in our work, we obtain W_{proj} as embeddings of a motion descriptor obtained from a CLIP text encoder.

4.4 Implementation details of our method

4.4.1 Problem Setting:

The model is trained on kinetics-400 as source dataset D_s and tested on target datasets D_t: UCF-101 and HMDB-51. The source dataset D_s consists of videos x_s and labels L_s belonging to classes C_s. The target dataset D_t consists of videos x_t and labels L_t belonging to classes C_t. We use a zero-shot setting such that $C_s \cap C_t = \emptyset$. Let the generated motion descriptions from GPT-4 for L_s and L_t be M_s and M_t respectively.

4.4.2 Training

Motion descriptions M_s are passed through a frozen CLIP text encoder to obtain the class prototypes of L_s. Our intuition is that these class prototypes can be approximated as classifier weights of the supervised video classifier. The concept takes its motivation from the work of [121, 122], which shows that text embeddings from CLIP and vision embeddings from CLIP are very similar and fall within a ball of small radius. The obtained CLIP embeddings of motion descriptions from the frozen text encoder would approximately translate to visual class prototypes if obtained visually. With that intuition, we use the class prototypes from the frozen CLIP text encoder as the classifier weights of a visual classifier.

Given a video v_s from the source dataset D_s, T frames are sampled uniformly. The sampled frames are passed through a CLIP pre-trained Image encoder and temporally pooled to obtain a visual feature of the video. Then, the logits are obtained by computing the dot product of this video feature with the transpose of the classifier weights W_s. The model is trained using cross-

entropy loss over logits and labels L_s, and the parameters of the CLIP image encoder are updated.

4.4.3 Testing

The motion descriptions M_t are passed through the network to obtain classifier weights W_t. Given a video v_t from the target dataset D_t, we obtain the logits indicating the probability that a video matches the corresponding motion description M_t.

4.4.4 Experimental details

Our model uses a VIT-B/16 pre-trained CLIP text and image encoder. We use eight frame samples per video. The CLIP text encoder was kept frozen during the training, and the CLIP image encoder was fine-tuned. We train the model for 10 epochs on the Kinetics-400 dataset with a learning rate of 0.00005 with a batch of the size of 20 on 4 NVIDIA RTX A5000 for 40 GPU hours. We use a learning warm step of 5 and a weight decay of 0.2. We use the Adam optimizer with the cross-entropy loss for training on the Kinetics dataset with a clip ratio of 0.1. Here, we describe more details about our baselines. We report the best results after running experiments on 5 runs.

4.5 Dataset

4.5.1 Kinetics-400

Kinetics-400 is a large-scale dataset containing 400 classes downloaded from YouTube. It has 240K training videos and 20K validation videos. Some videos are missing if the YouTube user has removed them.

4.5.2 UCF-101

The UCF-101 human action dataset consists of 13 K YouTube videos belonging to 101 classes. We report results on full classes on one split provided by the authors.

4.5.3 HMDB-51

It contains approximately 7K videos belonging to 51 classes. We report results on the split provided by the authors.

4.6 Results

Method	Object	Masked Object
Vanilla CLIP	25.92	23.33
Text4Vis	51.23	33.80
XCLIP	52.37	32.01
ViFiCLIP	52.70	34.76
Our Method	58.46	47.80
Human estimate	98	98

Table 4.1: Evaluation of percentage accuracy in motion description retrieval task on UCF-101 dataset.

Task: For the target datasets UCF-101 and HMDB-51, the input is a video and the list of generated motion descriptions for all the classes in the dataset. The video-text model predicts the closest motion description that describes the video. The metric used is the percentage accuracy of correctly predicted motion descriptions.

The models we evaluate are Vanilla CLIP [96], XCLIP [97], Text4Vis [115] and VifiCLIP [116]. Details about the models are given in section 4.7.

Method	Object	Masked Object
Vanilla CLIP	25.26	16.67
XCLIP	29.35	19.35
Text4Vis	34.12	24.93
VifiCLIP	36.20	28.9
Our Method	39.24	28.41
Human estimate	97.5	96

Table 4.2: Evaluation of percentage accuracy in motion description retrieval task on HMDB-51 dataset.

Human estimated performance: We sampled five videos randomly from each class in UCF-101 and HMDB-51. A human expert was asked to select the motion description that correctly describes the video from the list of descriptions generated by GPT4. Human experts are graduate students who have taken computer vision and NLP graduate courses and volunteered for this study.

Table 4.1 reports the performance of various approaches on the UCF101 dataset. Our proposed method beats previous methods by over 5% for motion descriptions containing the names of objects involved and by over 10% for motion descriptions where the word "object" replaces the object's name. We noticed that all the video-text models perform very poorly compared to human-estimated performance. We also see that video-text models have a strong bias toward nouns. When the specific name of the object involved is not used, there is an average 10% drop in performance for all methods, indicating the strong bias video-text models have for objects. Table 4.2 reports the performance of various approaches on the HMDB-51 dataset. Similar trends are found in the HMDB-51 dataset.

4.6.1 Temporal modeling

We experimented with adding a 6-layer temporal transformer on the video head of the VIT-B/16 transformer. The results are shown below in table 4.3 and 4.4. Contrary to our initial hypothesis, having a temporal transformer didn't improve the performance over mean average pooling.

Method	Object	Masked Object
Temporal transformer	57.02	44.4
Mean Average Pooling	58.46	47.80

Table 4.3: Evaluation of percentage accuracy in motion description retrieval task on UCF-101 dataset.

Method	Object	Masked Object
Temporal transformer	37.53	26.84
Mean Average Pooling	39.24	28.41

Table 4.4: Evaluation of percentage accuracy in motion description retrieval task on HMDB-51 dataset.

4.6.2 Does fine-tuning cause overfitting?

As in any fine-tuning method, there is a risk of overfitting the source dataset. We performed experiments to see if overfitting is an issue. Based on our experiments, the degree to which the model overfits is negligible compared to the method's overall improvement. Table 4.5 and Table 4.6 below show the accuracies at different epochs of fine-tuning the vision encoder.

Number of Epochs	Object	Masked Object
Epoch 5	57.50	46.59
Epoch 10	58.46	47.80
Epoch 20	58.2	47.02

Table 4.5: Evaluation of percentage accuracy in motion description retrieval task on UCF-101 dataset.

Number of Epochs	Object	Masked Object
Epoch 5	38.25	27.82
Epoch 10	39.239	28.41

Table 4.6: Evaluation of percentage accuracy in motion description retrieval task on HMDB-51 dataset.

4.7 Baselines

4.7.1 Vanilla CLIP:

We use VIT-B/16 pre-trained CLIP text and image encoder obtained from radford2021learning and temporally average the frame outputs while evaluating HMDB-51 and UCF-101 datasets.

4.7.2 XCLIP

We use the XCLIP ni2022expanding model and code available from `https://huggingface.co/docs/transformers/model_doc/xclip`. We use "microsoft/xclip-base-patch16-zero-shot" model from huggingface.co while evaluating the HMDB-51 and UCF-101 datasets.

4.7.3 Text4Vision

We use the VIT-B/16 base architecture with 8 frames per video. We use pre-trained weights from `https://github.com/whwu95/Text4Vis/tree/main`

4.7.4 VifiCLIP

We use the VIT-B/16 architecture and obtain the model and code from the official implementation of VifiCLIP rasheed2023fine from `https://github.com/muzairkhattak/ViFi-CLIP`.

4.8 Motion Description Generation

We use the GPT -4 API to obtain motion descriptions. We noticed from experiments that we needed to provide some example motion descriptions in the prompt to obtain motion descriptions in the format we are interested in. The prompt we used is *Characteristic motion of air drumming action is: doing rhythmic, exaggerated hand and arm movement. The characteristic motion of abseiling is: doing a controlled descent down a vertical drop. The characteristic motion of swing dancing is: doing energetic, bouncy movements with lots of spins, kicks, and lifts. Similarly, provide the characteristic motions of action X.*.

4.8.1 GPT-4 generated motion descriptions quality control

5 volunteers evaluated the generated motion descriptions for pairwise comparison. The pairwise comparison included selecting the best one among two generated motion descriptions. A vote of majority was used to select the final motion description. Volunteers with diverse

Dataset	Number of videos	Number of unique motion descriptions	Average number of verbs per motion description	Average number of words per motion description	Number of verbs in the motion descriptions
Kinetics 400	246000	400	3.4	19	1371
UCF 101	13320	101	3.2	19	325
HMDB 51	6849	51	3.2	17	164

Table 4.7: Statistics of motion description dataset

experience and age groups were selected to reduce bias.

4.8.2 Dataset statistics

The kinetics-400 dataset consists of 400 classes, the UCF-101 human action dataset consists of 101 classes and HMDB-51 consists of 51 classes. We generate a characteristic motion description for each class in all three datasets. For UCF101 and HMDB-51, we mask objects manually after obtaining the motion description. Table 4.8.2 provides the motion description dataset statistics.

4.9 Limitations

We use a CLIP text encoder trained on image-text data to represent motion descriptions. This is not the best thing to do as, in practice, the CLIP text encoder would never have encountered the dynamics of videos while training. However, we hope the method works if we replace this CLIP text encoder with any other video-text model text encoder. Another limitation is that we trained on the Kinetics-400 dataset and tested on the UCF-101 and HMDB-51 datasets. As shown in the results section, the presence of an object or scene can often impact the performance during the retrieval task on these datasets. Furthermore, since we fine-tune image encoders on the source dataset, the possibility of overfitting the source dataset exists, leading to poor transferability on another target dataset. There are also potential biases in generated descriptions by GPT-4, and human quality estimation is expensive. For our experiments, volunteers spent a total of 16 hours.

4.10 Conclusion

In this chapter, we discussed our work published in [11]. We explored how motion, one of the critical constituent of action, is represented in video-language models. We created a a motion description dataset using GPT-4 and evaluated existing video-language models. We noticed that the performance of existing video-language models are far from human expert performance. We proposed a method to integrate any existing video-language models with motion descriptions by finetuning a video encoder. Our proposed method improves existing video-language models to perform better in motion-description retrieval in UCF-101 and HMDB-51 datasets. In this work, we reported the best accuracy after running experiments, but including confidence intervals further strengthens the case. While designing video-text models, we leave it to future work to build better models to capture motion and ignore the biases due to the object or the scene.

Chapter 5: Generating images through Symbols representing concepts

Recent efforts in deep generative modeling have yielded impressive results, showcasing both the capabilities and limitations of VAEs [123] and GANs [124]. Many of these methods [125] have been used to generate high-resolution counterfeit images that are virtually indistinguishable from real ones using the naked eye. However, latent representations used to generate images are (for the most part) uninterpretable. To address this limitation, we propose a symbolic variant of the traditional VAE which places key constraints on the hidden/latent state of the network. We argue that these constraints improve interpretability by capturing explainable image semantics within a discrete symbol space. Similar to speech and language, discrete representations of latent information provide several key advantages. For instance, they can be used to model salient classes in auditory/visual data, or even represent meaningful policies and states in reinforcement learning applications. In this work, we focus on a symbolic implementation of the traditional VAE, applied to two unique image sets.

In each dataset, latent symbols used to describe imagery are analyzed and compared with their semantic interpretations. We term this variant of the traditional VAE the Symbolic Variational Autoencoder (SVAE). The key difference in our approach is that latent symbols used to encode imagery-much like the words on this page-serve as the building blocks for a learned private language. Given a sequence of discrete symbols (i.e., a sentence), we can directly decode the

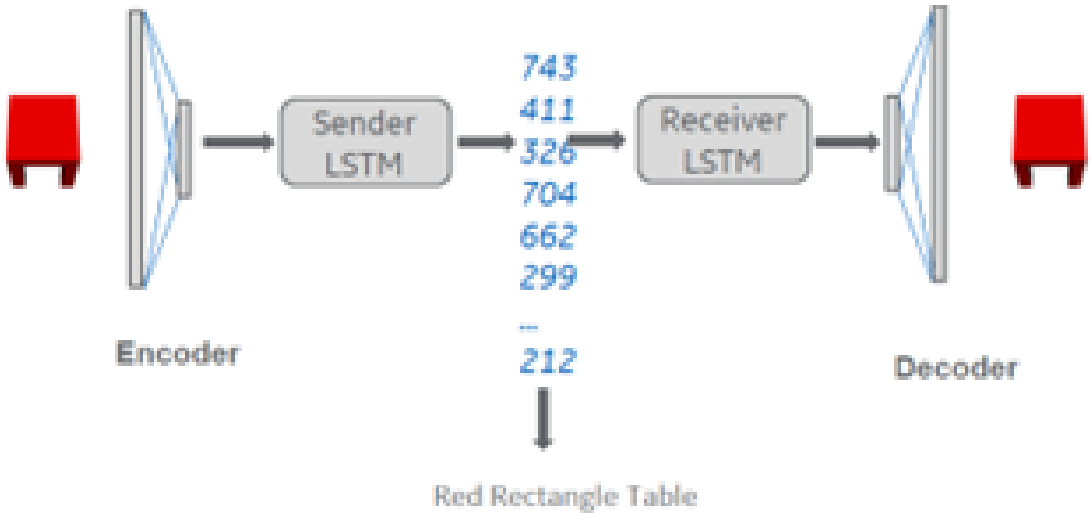


Figure 5.1: Architecture diagram of Symbolic Variational Autoencoders (SVAEs). Input is an image of a ”red table” and the Sender LSTM outputs a sentence which serves as a symbolic expression of ”red table”. The Receiver LSTM is able to directly reconstruct the image from the symbols rather than a continuous latent representation.

image that was used to generate it. This allows us to manipulate key symbols in a sentence to determine the ”meaning” of each one. Similar approaches have been reported in recent research on emergent languages [126, 127], although the current implementation focuses more on how objects in images are constructed rather than how they are described.

Similar to the SVAE, humans seem to use hierarchical labeling to describe entities in the world. WordNet [128] appears to capture this property, where each word has many possible hypernyms (e.g., ”color” is a hypernym of ”red”). Hierarchical mappings in WordNet have improved interpretability significantly, helping to capture relationships between two words. Studies in neuroscience [129] have also shown that rule-based hierarchical models can be used to explain cortical linguistic structure.

It has even been shown that GAN-Tree uses a hierarchical structure to generate multi-modal data distributions [130] . The SVAE generates symbols following a learned grammar that is both hierarchical and explainable. We report results on the Fashion MNIST and MNIST datasets. To

our knowledge, this is the first study which uses a discrete latent space to generate a hierarchical grammar via unsupervised learning methods. This effectively improves model explainability, as we have greater control in generating data based on symbols. Using the SVAE, we explore how an image generated from a sentence of symbols varies as symbols are changed in a systematic manner. We associate symbol manipulations with semantically noticeable changes in the reconstructed image, effectively grounding the meaning of learned symbols.

5.1 Related work

VAEs [123] are generative models used to estimate the underlying data distribution from a set of training examples. It consists of an encoder that maps raw input to a latent variable z and a decoder which uses z to reconstruct the input. The loss function optimized in the VAE is a combination of: i) the KL divergence loss between the latent encoding vector and a known Gaussian distribution and ii) the reconstruction loss at the decoder. Training is performed in an end-to-end manner with the help of a reparameterization trick at the decoder, which converts a non-differentiable node to a differentiable node.

Only a small number of VAEs which use a discrete latent space have been reported in the literature; e.g., VQ-VAE [131] and VQ-VAE-2 [132]. In these methods, encoder output is quantized into one latent vector from an N -vector codebook. The SVAE instead generates a sequence of symbols using a Long Short-Term Memory (LSTM) network [133]. The sequence of symbols follows a hierarchy where the first symbol in the sequence captures most discriminative information, such as class/category assignment. Alternatively, later symbols represent finer details within the class, like child nodes under a parent node. Since we are using an LSTM, we

hope to capture the grammar which underlies patterns of discrete symbols, rather than encoding the information in terms of independent symbols. If effectively captured, this grammar may be used for other purposes, such as generating variations of the same image by changing one or more of its associated symbols. For example, multiple colors of an object could be visualized, even though the SVAE has never actually seen corresponding images during training. Beta-VAE [134] is another modification of traditional VAEs with an emphasis on uncovering disentangled latent factors. This disentanglement serves to identify the factors that produce variations in the generated images. Even though the latent factors are discrete-like the symbols generated in our SVAE-they are generated in a fundamentally different way that does not capture hierarchical representation.

Training a neural network with discrete intermediate outputs exhibits a number of challenges. For instance, standard backpropagation only works on differentiable functions. The Sender LSTM module of SVAE (see Fig. 5.1) is non-differentiable. To circumvent this issue, we use the gumble softmax [135] operation to make the Sender LSTM module in SVAE differentiable. The Sender and Receiver modules of SVAE are similar to those implemented in recent work [127]. It has been further demonstrated that the emergence of symbolic representations correlates well with categorical labels; e.g., those from the MSCOCO dataset [127]. These emergent languages can also be grounded in natural language through direct supervision, as shown in recent work [126, 127]. Moreover, when supervision is provided, symbols are highly correlated with corresponding object categories or labels [136].

Instead of learning to describe imagery, the current work focuses more on learning what constitutes an image so that whole images can be reconstructed using latent, symbolic representations. Concurrent with the work reported here, a toolkit has been released [137] which showcases

some of the potential applications of emergent languages for different types of games. One such example is using sender and receiver modules as an autoencoder. The key difference between our SVAE and the toolkit's approach is that we construct our SVAE with variational inference like in traditional VAEs. The SVAE also uses VAE-like loss, whereas the toolkit utilizes an autoencoder with per-pixel cross-entropy loss. Using the proposed method, we show how symbols affect the generation of images and identify some of the primitives of image generation which was not described in previous work [137]. Furthermore, to the best of our knowledge, such toolkit is not trained end to end. Similarly, our SVAE follows a different methodology compared with the symbolic autoencoder reported [138].

5.2 Approach

The main components of the SVAE are i) an Encoder, ii) a Sender, iii) a Receiver and iv) a Decoder. Figure 5.1 shows the corresponding SVAE architecture.

Let X be the data to be encoded. We are interested in modeling the distribution $P(X)$. In a conventional VAE, a latent distribution $P(z)$ is learned which represents the data. This distribution is subsequently used to reconstruct X. Input is passed through an encoder which consists of either a CNN or multi-layer perceptron layers. The output is then passed through a reparameterization layer, followed by a decoder. However, in the current work, we transformed the latent distribution into a discrete representation captured by a sequence of symbols. This was achieved by implementing a Sender module which generates a sequence of symbols via an LSTM. The symbols produced by the sender LSTM are passed into a receiver LSTM that reconstructs the original feature embedding. Finally, the Decoder module transforms the reconstructed feature embedding into

an image. We train the entire module with the reconstruction loss and KL divergence loss as described in equation below. Parameters of the Encoder, Sender, Receiver, and Decoder modules are jointly optimized by backpropagation:

$$\text{Loss } = E[\log P(X \mid z)] - D_{KL}[Q(z \mid X) \mid P(z)]$$

where $E[\log P(X \mid z)]$ simplifies to taking binary cross entropy between reconstructed image and input image, and $D_{KL}[Q(z \mid X) \mid P(z)]$ results in:

$$\begin{gathered} D_{KL}[\mathcal{N}(\mu(X), \sigma(X)) \mid \mathcal{N}(0,1)]] = \\ \tfrac{1}{2}\Sigma\left[\exp\left(\sigma(x) + \mu^2(X) - 1 - \sigma(X)\right)\right] \end{gathered} \tag{5.2}$$

Simplifications were made by assuming $P(z)$ is a normal distribution with mean 0 and standard deviation 1 . In this work, we fixed the encoder and decoder to consist of two fully connected layers. Since backpropagating gradients across discrete symbols is not possible, we utilized a Gumbel Softmax estimator. This results in a continuous gradient that is both stable and differentiable.

5.3 Experiments and results

We tested SVAE on two datasets: MNIST [139] and Fashion-MNIST [140]. Both datasets consist of 60,000 training images and 10,000 test images. In each of the experiments reported, we used an encoder and decoder consisting of two fully-connected layers, reducing the dimension of input image first to 400 and then to 20 in respective layers. The 20 -dimensional feature from the last fully-connected layer of the Encoder module was fed to a reparameterization layer, similar to

Figure 5.2: The top row indicates the input image fed to the autoencoder. The middle row indicates the image reconstructed from SVAE with a vocabulary size of 10 and sentence length of 2. Symbols for each of the reconstructed images are displayed in the bottom row.

traditional VAEs. The output from the reparameterized layer was then fed to the Sender module. The Sender and Receiver components consist of a single unrolled LSTM based on the sequence length used in different settings. The Sender LSTM embedding dimension is 256 and the hidden layer dimension is 512 . The temperature parameter in Gumbel Softmax is 1. Adam optimizer was used with learning rate $le - 5$.

4.1. Figures and tables

Experiments show that discrete symbols capture the semantic properties of an image and can be used to unearth underlying primitives. Without any supervision, we observed that each symbol represents a concept. In the following paragraphs, we show that the generated symbols form a grammar with useful semantic properties.

Figure 5.2 shows a representation of reconstructions from the SVAE trained on the Fashion-MNIST dataset. This network was trained with a vocabulary size of 10 symbols and a sentence

length of 2. Vocabulary size refers to the possible number of symbols present in the dictionary. We observe that even with a small sentence and vocabulary size, the network can faithfully reproduce the input test data. The corresponding symbols or sentences that the SVAE produced are shown in Figure 5.2. We observe that the first symbol 5 corresponds with shirts, as is evident from the 1^{st} , 2^{nd} , 4^{th} , 7^{th} , and 8^{th} images. In addition, the symbol 1 is associated with the concept of shoes. The second symbol in the sentence encodes the different types of garments (shirts, shoes, dresses). The 3^{rd} image in the figure is incorrectly reconstructed as trousers. Therefore, the symbol is 3 instead of 5 .

Figure 5.3: Image reconstructions using vocabulary size of 100 (left) vs. a vocabulary size of 20 (right). Note that reconstructions show much finer detail 6 when more symbols are used to encode images.

As depicted in figure 5.3 , increasing the vocabulary size to 100 allows for richer representations and better reconstructions. This follows the principle of image compression where a lossy image has a lesser number of bits. However, as we increase the vocabulary size and sentence length, it becomes difficult to qualitatively interpret the exact semantic meaning of each symbol through

visualization only. Still, the first symbol in the sentence continues to refer to class membership.

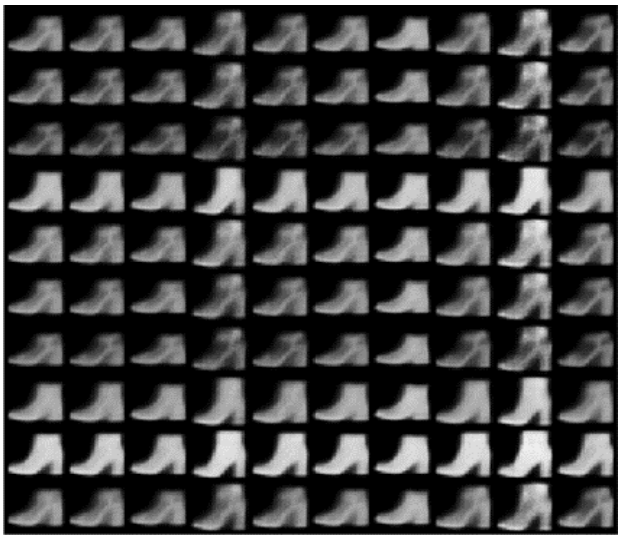


Figure 5.4: Reconstructions of Fashion-MNIST. Reconstructed images change when we hold the first symbol constant and manipulate the second and third symbol only. Each row and column represents a different second and third symbol, respectively. SVAE was trained on Fashion-MNIST with with vocabulary size of 20 and sentence length of 3

The next set of experiments involved training the SVAE with a vocabulary size of 20 and a sentence length of 3 . Figure 5.4 shows the output for FashionMNIST when the first symbol is fixed at 15 and the remaining two symbols in the sentence are exhaustively explored. Results indicate that symbol 15 is associated with a high-heeled shoe. This suggests that the first symbol in the sentence indicates the broad category in the dataset and the second and third symbols indicate changes in intensity and shape. This further implies that there is a grammar underlying the composed sentences. A similar result is obtained when we run the same experiment on the MNIST dataset (see Figure 5.5. Here, the first symbol 15 shows a representation of the number 9 . Changing the remaining two symbols in the

sentence shows different types and shapes of the number 9. We also see some 7's in certain positions in the grid, which could be due to the fact that the number 7 looks visually similar to the number 9 . Therefore, symbol 15 could actually be encoding the shape of both 9 and 7 .

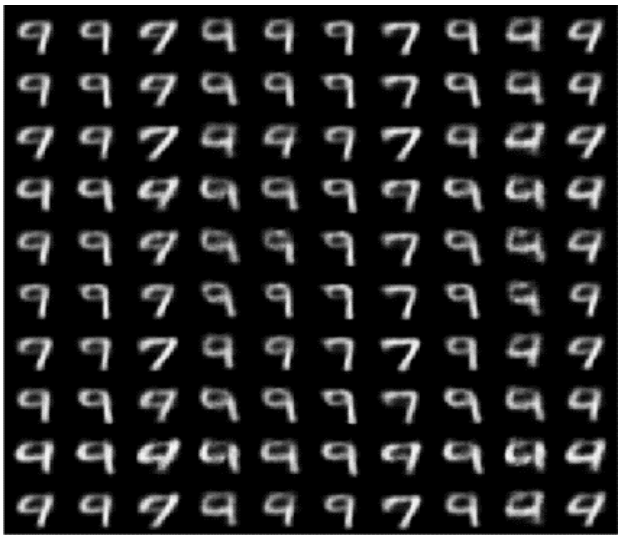


Figure 5.5: Reconstructions of MNIST. Reconstructed images change when we hold the first symbol constant and manipulate the second and third symbol only. Each row and column represents a different second and third symbol, respectively. SVAE was trained on MNIST with vocabulary size of 20 and sentence length of 3

SVAE offers advantages of visualizing the latent space compared to VQ-VAE. Compared to VQ-VAE, we have better control over generated images because of the added advantage of using sentences instead of a single bit. Thus, we present a systematic way of generating images by exhausting all possible symbols of the given vocabulary size and sentence length.

5.4 Conclusion

In this chapter, we discussed our work published in [12]. We presented an unsupervised approach to obtaining the constituents of images in the form of symbols and using symbols to generate the images. We developed a symbolic variational autoencoder capable of generating images given symbols. The approach is general as the architecture is data agnostic and differentiable. This method provides a way to generate different constituents of images, and the idea can also be extended to action videos.

Chapter 5: Conclusion and future work

In this thesis, we presented novel approaches to use constituents of actions and objects to improve understanding of actions and objects.

Chapter 2 introduced two methods for visual feedback in language graphs for zero-shot action recognition. Grouping actions based on the constituent of actions allowed us to adjust additively the weights of the adjacency matrix to reflect the semantic similarity of actions in visual space. Increasing and decreasing the graph edge weights is simple yet very effective and can be easily adopted in other weight-adjusting mechanisms. We also introduced using the visual adjacency matrix to set the weights in the graph, which is usually overlooked in zero-shot learning. Experimental results on the EPIC-Kitchens and the Charades data sets showed that adding the visual adjacency matrix and using weight adjustment to modify the adjacency matrix is beneficial.

In Chapter 3, we introduced methods to produce hand/object-centric representations for egocentric video, which are useful for action understanding. Contact Anticipation Maps provide time-to-contact predictions between hands and the environment, and Next Active Object Segmentations provide predictions localizing the Next Active Object. In training the In the anticipation Module, we gather contact annotations and object segmentations over a portion of the EPIC Kitchens dataset to produce these representations. We achieve state-of the- art results over the EPIC

Kitchens Action Anticipation Challenge - achieving 1st and 2nd place on the unseen and seen test sets, respectively - through feeding our representations through Ego-OMG, our state-of-the-art action anticipation and action prediction architecture. We release our predictions over the EGTEA dataset and provide ablation studies evaluating the utility of individual system characteristics.

In Chapter 4, we showed how motion, an essential constituent of action, is understood in video-language models. We introduced a benchmark to understand how motion is understood in video-text models. We highlighted the limitations in obtaining quality annotations describing motion in video. We also showed that the performance of video-text models for retrieving motion descriptions is poor compared to human expert performance. Our proposed method circumvents some of these issues and improves over other video-text models. While designing video-text models, we leave it to future work to build better models to capture motion and ignore the biases due to the object or the scene.

In Chapter 5, we explored an unsupervised way to obtain the constituents of an object. We proposed a Symbolic Variational Autoencoder, which provides the advantages of interpretability and the ability to generate images by varying symbolic encodings. We also observe that the symbols form a grammar, where the first symbol typically refers to the class of the image, and the next set of symbols express finer features. By exhausting all possible symbol sequences for a given category, we showed how finer characteristics are naturally captured in a hierarchical fashion. This provides the foundation needed to understand what the primitives of images are and how each primitive might affect the appearance of various image types. Further work should extend the current approach to understand primitives in other kinds of data, such as human action or gesture videos. Although the contemporary success of deep learning methods has provided exciting new ways to transform data into useful representations, explainability remains

a critical problem. In particular, human interfacing with artificial agents will rely on modes of communication that both parties can interpret. The current results provide an initial step toward this goal by implementing a symbolic method for encoding raw data. As conveyed in our results, each symbol appears to have some meaning to human observers, such as a "red shoe" versus a "white shoe." While the corresponding symbols can be used to reconstruct images using the SVAE, the Sender and Receiver in the network, do not understand each symbol like humans do.

While progress has been made in both the discriminative and generative sides of machine learning and computer vision, explainability will be an essential metric for evaluating their success as we develop more intelligent systems. This dissertation will open up some ideas on how explainability can be improved by modeling actions as a function of their constituent parts rather than a whole action. In the future, we can develop alternative methods to discover the constituent of actions to minimize human inference and develop attention mechanisms between the constituent parts and the whole action.

Bibliography

[1] Marc Jeannerod. The timing of natural prehension movements. *Journal of motor behavior*, 16(3):235–254, 1984.

[2] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017.

[3] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1933–1941, 2016.

[4] Bowen Pan, Jiankai Sun, Wuwei Lin, Limin Wang, and Weiyao Lin. Cross-stream selective networks for action recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 0–0, 2019.

[5] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. *Advances in neural information processing systems*, 27, 2014.

[6] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6202–6211, 2019.

[7] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7083–7093, 2019.

[8] Christoph Feichtenhofer, Yanghao Li, Kaiming He, et al. Masked autoencoders as spatiotemporal learners. *Advances in neural information processing systems*, 35:35946–35958, 2022.

[9] Chinmaya Devaraj, Cornelia Fermüller, and Yiannis Aloimonos. Incorporating visual grounding in gcn for zero-shot learning of human object interaction actions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5008–5017, 2023.

[10] Eadom Dessalene, Chinmaya Devaraj, Michael Maynord, Cornelia Fermuller, and Yiannis Aloimonos. Forecasting action through contact representations from first person video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[11] Chinmaya Devaraj, Cornelia Fermuller, and Yiannis Aloimonos. Diving deep into the motion representation of video-text models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics ACL 2024*, pages 12575–12584, Bangkok, Thailand and virtual meeting, August 2024. Association for Computational Linguistics.

[12] Chinmaya Devaraj, Aritra Chowdhury, Arpit Jain, James R Kubricht, Peter Tu, and Alberto Santamaria-Pang. From symbols to signals: symbolic variational autoencoders. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3317–3321. IEEE, 2020.

[13] Yikang Li, Wanli Ouyang, Bolei Zhou, Kun Wang, and Xiaogang Wang. Scene graph generation from objects, phrases and region captions. In *Proceedings of the IEEE international conference on computer vision*, pages 1261–1270, 2017.

[14] Vincent S. Chen, Paroma Varma, Ranjay Krishna, Michael Bernstein, Christopher Re, and Li Fei-Fei. Scene graph prediction with limited labels. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[15] Somak Aditya, Yezhou Yang, Chitta Baral, Yiannis Aloimonos, and Cornelia Fermüller. Image understanding using vision and reasoning through scene description graph. *Computer Vision and Image Understanding*, 173:33–45, 2018.

[16] Jingwei Ji, Ranjay Krishna, Li Fei-Fei, and Juan Carlos Niebles. Action genome: Actions as compositions of spatio-temporal scene graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10236–10247, 2020.

[17] Cornelia Fermüller and Michael Maynord. Learning for action-based scene understanding. In *Advanced Methods and Deep Learning in Computer Vision*, pages 373–403. Elsevier, 2022.

[18] Oytun Ulutan, ASM Iftekhar, and Bangalore S Manjunath. Vsgnet: Spatial attention network for detecting human object interactions using graph convolutions. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13617–13626, 2020.

[19] Ilaria Tiddi and Stefan Schlobach. Knowledge graphs as tools for explainable machine learning: A survey. *Artificial Intelligence*, 302:103627, 2022.

[20] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[21] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[22] Muhammad Raza Khan and Joshua E Blumenstock. Multi-gcn: Graph convolutional networks for multi-view networks, with applications to global poverty. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 606–613, 2019.

[23] Nan Jia, Xiaolin Tian, Yang Zhang, and Fengge Wang. Semi-supervised node classification with discriminable squeeze excitation graph convolutional networks. *IEEE Access*, 8:148226–148236, 2020.

[24] Liyu Gong and Qiang Cheng. Exploiting edge features for graph neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9211–9219, 2019.

[25] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9267–9276, 2019.

[26] Pantelis Elinas, Edwin V Bonilla, and Louis Tiao. Variational inference for graph convolutional networks in the absence of graph data and adversarial settings. *arXiv preprint arXiv:1906.01852*, 2019.

[27] Pallabi Ghosh, Nirat Saini, Larry S Davis, and Abhinav Shrivastava. Learning graphs for knowledge transfer with limited labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11151–11161, 2021.

[28] Pallabi Ghosh, Nirat Saini, Larry S Davis, and Abhinav Shrivastava. All about knowledge graphs for actions. *arXiv preprint arXiv:2008.12432*, 2020.

[29] Keizo Kato, Yin Li, and Abhinav Gupta. Compositional learning for human object interaction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 234–251, 2018.

[30] Qian Wang and Ke Chen. Zero-shot visual recognition via bidirectional latent embedding. *International Journal of Computer Vision*, 124(3):356–383, 2017.

[31] Mihir Jain, Jan C Van Gemert, Thomas Mensink, and Cees GM Snoek. Objects2action: Classifying and localizing actions without any video example. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4588–4596, 2015.

[32] Ashish Mishra, Vinay Kumar Verma, M Shiva Krishna Reddy, S Arulkumar, Piyush Rai, and Anurag Mittal. A generative approach to zero-shot and few-shot action recognition. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 372–380, 2018.

[33] Yanwei Fu, Timothy M Hospedales, Tao Xiang, Zhenyong Fu, and Shaogang Gong. Transductive multi-view embedding for zero-shot recognition and annotation. In *European Conference on Computer Vision*, pages 584–599. Springer, 2014.

[34] Jingen Liu, Benjamin Kuipers, and Silvio Savarese. Recognizing human actions by attributes. In *CVPR 2011*, pages 3337–3344. IEEE, 2011.

[35] Marcus Rohrbach, Sandra Ebert, and Bernt Schiele. Transfer learning in a transductive setting. *Advances in Neural Information Processing Systems*, 26:46–54, 2013.

[36] Chuang Gan, Ming Lin, Yi Yang, Yueting Zhuang, and Alexander G Hauptmann. Exploring semantic inter-class relationships (SIR) for zero-shot action recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

[37] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[38] Michael Kampffmeyer, Yinbo Chen, Xiaodan Liang, Hao Wang, Yujia Zhang, and Eric P Xing. Rethinking knowledge graph propagation for zero-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11487–11496, 2019.

[39] Florentin Wörgötter, Eren Erdal Aksoy, Norbert Krüger, Justus Piater, Ales Ude, and Minija Tamosiunaite. A simple ontology of manipulation actions based on hand-object relations. *IEEE Transactions on Autonomous Mental Development*, 5(2):117–134, 2013.

[40] Yezhou Yang, Cornelia Fermuller, and Yiannis Aloimonos. Detection of manipulation action consequences (mac). In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2563–2570, 2013.

[41] Gunnar A Sigurdsson, Olga Russakovsky, and Abhinav Gupta. What actions are needed for understanding human actions in videos? In *Proceedings of the IEEE international conference on computer vision*, pages 2137–2146, 2017.

[42] Yao Ma, Suhang Wang, Chara C Aggarwal, Dawei Yin, and Jiliang Tang. Multi-dimensional graph convolutional networks. In *Proceedings of the 2019 SIAM International Conference on Data Mining*, pages 657–665. SIAM, 2019.

[43] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. Scaling egocentric vision: The EPIC-KITCHENS dataset. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 720–736, 2018.

[44] Gunnar A Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *European Conference on Computer Vision*, pages 510–526. Springer, 2016.

[45] Bernardino Romera-Paredes and Philip Torr. An embarrassingly simple approach to zero-shot learning. In *International Conference on Machine Learning*, pages 2152–2161, 2015.

[46] Andrea Frome, Greg Corrado, Jonathon Shlens, Samy Bengio, Jeffrey Dean, Marc'Aurelio Ranzato, and Tomas Mikolov. Devise: A deep visual-semantic embedding model. 2013.

[47] Li Zhang, Tao Xiang, and Shaogang Gong. Learning a deep embedding model for zero-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2021–2030, 2017.

[48] Kevin Sexton, Amanda Johnson, Amanda Gotsch, Ahmed A Hussein, Lora Cavuoto, and Khurshid A Guru. Anticipation, teamwork and cognitive load: chasing efficiency during robot-assisted surgery. *BMJ quality & safety*, 27(2):148–154, 2018.

[49] Cordula Vesper. How to support action prediction: Evidence from human coordination tasks. In *The 23rd IEEE International Symposium on Robot and Human Interactive Communication*, pages 655–659. IEEE, 2014.

[50] Yin Li, Miao Liu, and James M Rehg. In the eye of beholder: Joint learning of gaze and actions in first person video. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 619–635, 2018.

[51] Hamed Pirsiavash and Deva Ramanan. Detecting activities of daily living in first-person camera views. In *2012 IEEE conference on computer vision and pattern recognition*, pages 2847–2854. IEEE, 2012.

[52] JR Tresilian. Perceptual and cognitive processes in time-to-contact estimation: Analysis of prediction-motion and relative judgment tasks. *Perception & Psychophysics*, 57(2):231–245, 1995.

[53] Myrka Zago, Joseph McIntyre, Patrice Senot, and Francesco Lacquaniti. Visuo-motor coordination and internal models for object interception. *Experimental Brain Research*, 192(4):571–604, 2009.

[54] Waltraud Stadler, Anne Springer, Jim Parkinson, and Wolfgang Prinz. Movement kinematics affect action prediction: comparing human to non-human point-light actions. *Psychological research*, 76(4):395–406, 2012.

[55] Eadom Dessalene, Michael Maynord, Chinmaya Devaraj, Cornelia Fermuller, and Yiannis Aloimonos. Egocentric object manipulation graphs. *arXiv preprint arXiv:2006.03201*, 2020.

[56] Du Tran, Heng Wang, Lorenzo Torresani, and Matt Feiszli. Video classification with channel-separated convolutional networks. In *IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[57] Yu Kong and Yun Fu. Human action recognition and prediction: A survey. *arXiv preprint arXiv:1806.11230*, 2018.

[58] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Anticipating the future by watching unlabeled video.

[59] Chen Gao, Yuliang Zou, and Jia-Bin Huang. ican: Instance-centric attention network for human-object interaction detection. *arXiv preprint arXiv:1808.10437*, 2018.

[60] Julian Tanke and Juergen Gall. Human motion anticipation with symbolic label. *arXiv preprint arXiv:1912.06079*, 2019.

[61] Yazan Abu Farha, Alexander Richard, and Juergen Gall. When will you do what?-anticipating temporal occurrences of activities. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5343–5352, 2018.

[62] Antonino Furnari and Giovanni Maria Farinella. What would you expect? anticipating egocentric actions with rolling-unrolling lstms and modality attention. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6252–6261, 2019.

[63] Miao Liu, Siyu Tang, Yin Li, and James Rehg. Forecasting human object interaction: Joint prediction of motor attention and egocentric activity. *arXiv preprint arXiv:1911.10967*, 2019.

[64] Tushar Nagarajan, Yanghao Li, Christoph Feichtenhofer, and Kristen Grauman. Ego-topo: Environment affordances from egocentric video. *arXiv preprint arXiv:2001.04583*, 2020.

[65] Qiuhong Ke, Mario Fritz, and Bernt Schiele. Time-conditioned action anticipation in one shot. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9925–9934, 2019.

[66] Sven Bambach, Stefan Lee, David J Crandall, and Chen Yu. Lending a hand: Detecting hands and recognizing activities in complex egocentric interactions. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1949–1957, 2015.

[67] Aisha Urooj and Ali Borji. Analysis of hand segmentation in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4710–4719, 2018.

[68] Yezhou Yang, Cornelia Fermuller, Yi Li, and Yiannis Aloimonos. Grasp type revisited: A modern perspective on a classical feature for vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 400–408, 2015.

[69] Guillermo Garcia-Hernando, Shanxin Yuan, Seungryul Baek, and Tae-Kyun Kim. First-person hand action benchmark with rgb-d videos and 3d hand pose annotations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 409–419, 2018.

[70] Diogo Carbonera Luvizon, Hedi Tabia, and David Picard. Learning features combination for human action recognition from skeleton sequences. *Pattern Recognition Letters*, 99:13–20, 2017.

[71] Bugra Tekin, Federica Bogo, and Marc Pollefeys. H+ o: Unified egocentric recognition of 3d hand-object poses and interactions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4511–4520, 2019.

[72] Fabien Baradel, Natalia Neverova, Christian Wolf, Julien Mille, and Greg Mori. Object level visual reasoning in videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 105–121, 2018.

[73] Yin Li, Zhefan Ye, and James M Rehg. Delving into egocentric actions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 287–295, 2015.

[74] Antonino Furnari, Sebastiano Battiato, Kristen Grauman, and Giovanni Maria Farinella. Next-active-object prediction from egocentric videos. *Journal of Visual Communication and Image Representation*, 49:401–411, 2017.

[75] Tushar Nagarajan, Christoph Feichtenhofer, and Kristen Grauman. Grounded human-object interaction hotspots from video. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8688–8697, 2019.

[76] Tete Xiao, Quanfu Fan, Dan Gutfreund, Mathew Monfort, Aude Oliva, and Bolei Zhou. Reasoning about human-object interactions through dual attention networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3919–3928, 2019.

[77] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[78] Ashesh Jain, Amir R Zamir, Silvio Savarese, and Ashutosh Saxena. Structural-rnn: Deep learning on spatio-temporal graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5308–5317, 2016.

[79] Xiaolong Wang and Abhinav Gupta. Videos as space-time region graphs. In *European Conference on Computer Vision (ECCV)*, September 2018.

[80] Bilge Soran, Ali Farhadi, and Linda Shapiro. Generating notifications for missing actions: Don't forget to turn the lights off! In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4669–4677, 2015.

[81] Tian Lan, Yuke Zhu, Amir Roshan Zamir, and Silvio Savarese. Action recognition by hierarchical mid-level action elements. In *Proceedings of the IEEE international conference on computer vision*, pages 4552–4560, 2015.

[82] Antonino Furnari, Sebastiano Battiato, and Giovanni Maria Farinella. Leveraging uncertainty to rethink loss functions and evaluation measures for egocentric action anticipation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 389–405, 2018.

[83] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[84] Javier Sánchez Pérez, Enric Meinhardt-Llopis, and Gabriele Facciolo. Tv-l1 optical flow estimation. *Image Processing On Line*, 2013:137–150, 2013.

[85] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[86] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1251–1258, 2017.

[87] Deepti Ghadiyaram, Du Tran, and Dhruv Mahajan. Large-scale weakly-supervised pre-training for video action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12046–12055, 2019.

[88] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[89] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[90] Zheng Shou, Dongang Wang, and Shih-Fu Chang. Temporal action localization in untrimmed videos via multi-stage cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1049–1058, 2016.

[91] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *European Conference on Computer Vision*, pages 20–36. Springer, 2016.

[92] Guglielmo Camporese, Pasquale Coscia, Antonino Furnari, Giovanni Maria Farinella, and Lamberto Ballan. Knowledge distillation for action anticipation via label smoothing. *arXiv preprint arXiv:2004.07711*, 2020.

[93] 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*, pages 618–626, 2017.

[94] Matthias Kummerer, Thomas S. A. Wallis, Leon A. Gatys, and Matthias Bethge. Understanding low- and high-level contributions to fixation prediction. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[95] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468. IEEE, 2016.

[96] 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.

[97] Bolin Ni, Houwen Peng, Minghao Chen, Songyang Zhang, Gaofeng Meng, Jianlong Fu, Shiming Xiang, and Haibin Ling. Expanding language-image pretrained models for general video recognition. In *European Conference on Computer Vision*, pages 1–18. Springer, 2022.

[98] Huaishao Luo, Lei Ji, Ming Zhong, Yang Chen, Wen Lei, Nan Duan, and Tianrui Li. Clip4clip: An empirical study of clip for end to end video clip retrieval and captioning. *Neurocomputing*, 508:293–304, 2022.

[99] Han Fang, Pengfei Xiong, Luhui Xu, and Yu Chen. Clip2video: Mastering video-text retrieval via image clip. *arXiv preprint arXiv:2106.11097*, 2021.

[100] Mengmeng Wang, Jiazheng Xing, and Yong Liu. Actionclip: A new paradigm for video action recognition. *arXiv preprint arXiv:2109.08472*, 2021.

[101] Cornelia Fermüller, Fang Wang, Yezhou Yang, Konstantinos Zampogiannis, Yi Zhang, Francisco Barranco, and Michael Pfeiffer. Prediction of manipulation actions. *International Journal of Computer Vision*, 126:358–374, 2018.

[102] Eadom Dessalene, Michael Maynord, Cornelia Fermüller, and Yiannis Aloimonos. Therbligs in action: Video understanding through motion primitives. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10618–10626, 2023.

[103] Abhijit S Ogale, Cornelia Fermuller, and Yiannis Aloimonos. Motion segmentation using occlusions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(6):988–992, 2005.

[104] Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles. Dense-captioning events in videos. In *Proceedings of the IEEE international conference on computer vision*, pages 706–715, 2017.

[105] Jun Xu, Tao Mei, Ting Yao, and Yong Rui. Msr-vtt: A large video description dataset for bridging video and language. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5288–5296, 2016.

[106] 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. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2630–2640, 2019.

[107] Mathew Monfort, SouYoung Jin, Alexander Liu, David Harwath, Rogerio Feris, James Glass, and Aude Oliva. Spoken moments: Learning joint audio-visual representations from video descriptions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14871–14881, 2021.

[108] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[109] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[110] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: a large video database for human motion recognition. In *2011 International conference on computer vision*, pages 2556–2563. IEEE, 2011.

[111] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[112] OpenAI. Gpt-4 technical report, 2023.

[113] Lisa Anne Hendricks, Oliver Wang, Eli Shechtman, Josef Sivic, Trevor Darrell, and Bryan Russell. Localizing moments in video with temporal language. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

[114] Xin Wang, Jiawei Wu, Junkun Chen, Lei Li, Yuan-Fang Wang, and William Yang Wang. Vatex: A large-scale, high-quality multilingual dataset for video-and-language research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4581–4591, 2019.

[115] Wenhao Wu, Zhun Sun, and Wanli Ouyang. Revisiting classifier: Transferring vision-language models for video recognition. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 2847–2855, 2023.

[116] Hanoona Rasheed, Muhammad Uzair Khattak, Muhammad Maaz, Salman Khan, and Fahad Shahbaz Khan. Fine-tuned clip models are efficient video learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6545–6554, 2023.

[117] Liliane Momeni, Mathilde Caron, Arsha Nagrani, Andrew Zisserman, and Cordelia Schmid. Verbs in action: Improving verb understanding in video-language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15579–15591, October 2023.

[118] Jae Sung Park, Sheng Shen, Ali Farhadi, Trevor Darrell, Yejin Choi, and Anna Rohrbach. Exposing the limits of video-text models through contrast sets. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3574–3586, 2022.

[119] 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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9879–9889, 2020.

[120] Liliane Momeni, Mathilde Caron, Arsha Nagrani, Andrew Zisserman, and Cordelia Schmid. Verbs in action: Improving verb understanding in video-language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15579–15591, 2023.

[121] David Nukrai, Ron Mokady, and Amir Globerson. Text-only training for image captioning using noise-injected clip. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 4055–4063, 2022.

[122] Victor Weixin Liang, Yuhui Zhang, Yongchan Kwon, Serena Yeung, and James Y Zou. Mind the gap: Understanding the modality gap in multi-modal contrastive representation learning. *Advances in Neural Information Processing Systems*, 35:17612–17625, 2022.

[123] Diederik P Kingma. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[124] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[125] Andrew Brock. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[126] Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni. Multi-agent cooperation and the emergence of (natural) language. *arXiv preprint arXiv:1612.07182*, 2016.

[127] Serhii Havrylov and Ivan Titov. Emergence of language with multi-agent games: Learning to communicate with sequences of symbols. *Advances in neural information processing systems*, 30, 2017.

[128] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[129] Nai Ding, Lucia Melloni, Xing Tian, and David Poeppel. Rule-based and word-level statistics-based processing of language: insights from neuroscience. *Language, cognition and neuroscience*, 32(5):570–575, 2017.

[130] Jogendra Nath Kundu, Maharshi Gor, Dakshit Agrawal, and R Venkatesh Babu. Gan-tree: An incrementally learned hierarchical generative framework for multi-modal data distributions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8191–8200, 2019.

[131] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

[132] Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems*, 32, 2019.

[133] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471, 2000.

[134] Irina Higgins, Loic Matthey, Arka Pal, Christopher P Burgess, Xavier Glorot, Matthew M Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. *ICLR (Poster)*, 3, 2017.

[135] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[136] Alberto Santamaria-Pang, James R Kubricht, Chinmaya Devaraj, Aritra Chowdhury, and Peter Tu. Towards semantic action analysis via emergent language. In *2019 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*, pages 224–2244. IEEE, 2019.

[137] Eugene Kharitonov, Rahma Chaabouni, Diane Bouchacourt, and Marco Baroni. Egg: a toolkit for research on emergence of language in games. *arXiv preprint arXiv:1907.00852*, 2019.

[138] Oryan Barta. Symbolic autoencoder.

[139] Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, 29(6):141–142, 2012.

[140] H Xiao. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.