

Leveraging Structure for Generalization and Prediction in Visual System

CMU-RI-TR-19-70

Yufei Ye

June, 2019

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Thesis Committee:
Abhinav Gupta, Chair
Deva Ramanan
Xiaolong Wang

Abstract

Our surrounding world is highly structured. Humans have a great capacity of understanding and leveraging those structures to generalize to novel scenarios and to predict the future. The thesis studies how computer vision systems benefit from a similar process – leveraging inherent structures in data to improve generalization and prediction capacity. It focuses on two specific aspects: zero-shot recognition using categorical structures which is explicitly specified by knowledge graphs; video predictions by leveraging the implicit physical structures among entities. Both methods are based on the scalable machine learning framework, graph neural network, to directly learn structures from large-scale data. In zero-shot recognition, we have shown that accuracy improves significantly and is more robust due to external knowledge in the knowledge graph. In video prediction, we have found the long-term prediction is significantly sharper when factoring the structure among entities.

Acknowledgment

In my wonderful two-year journey at CMU, I am so fortunate to receive great help from many people with kindness, patience, and wisdom. They taught me how to conduct solid research, how to think critically, encouraged me to become a better researcher and even a better-self.

Firstly, many thanks to my advisor, Abhinav Gupta. Thank you for your valuable advice and supports. You taught me to be critical, to think ahead of time, to well communicate and so many important qualities to become a good researcher.

I would like to thank Xiaolong Wang and Shubham Tulsiani. You have played the role of a mentor during my master career. Thank you for your supervision, patience, and guidance. You led me to have a comprehensive understanding of research, from high-level idea to low-level implementation. I would have struggled one thousand times more without you.

Thank all of my lab members, for your constructive comments, especially Yin Li, Saurab Gupta, Lerrel Pinto, Gunnar Sigurdsson, Adithya Murali, Senthil Purushwalkam, Tao Chen, Nilesh Kulkarni, Wenxuan Zhou, Tian Ye, Gaurav Pathak, Pratyusha Sharma, Dhiraj Gandhi.

Lastly, thank my parents Lixin Li, Changchun Ye, for your constant unconditioned love and supports. Thank Donglai Xiang, for your trust and supports. You are always my rock.

Contents

1	Introduction	1
2	Zero-Shot via Explicit Semantic Structure among Categories	3
2.1	Introduction	3
2.2	Related Work	4
2.3	Approach	5
2.4	Experiment	8
2.5	Discussion	14
3	Video Prediction via Implicit Physical Structure among Entities	17
3.1	Introduction	17
3.2	Related Work	18
3.3	Approach	19
3.4	Experiment	22
3.5	Discussion	30
4	Conclusion	31

Chapter 1

Introduction

Humans have great inductive capability of drawing relationship from the structured world and leveraging those structures to learn and infer efficiently [54]. On the other hand, in most of the current visual algorithms, entities are learned independently. In classification task, the model considers categories living in n -simplex, i.e. each class is represented as an independent one-hot vector. But for human, categories never exists independently [2]. Take the example of the class “okapi”, which is “zebra-striped four legged animal with a brown torso and a deer-like face”. Humans can immediately generate a good visual classifiers just based on the description, with no need to seeing massive okapis before. (Test your self on Figure 1.1). Knowing the class okapi is closely related to deer and zebra gives us enough knowledge to solve the puzzle.



Figure 1.1: Can you find “okapi” in these images? Okapi is “ zebra-striped four legged animal with a brown torso and a deer-like face”. In chapter 2, we focus on the problem of zero-shot learning where visual classifiers are learned from semantic embeddings and relationships to other categories.

Not only do humans leverage the relationships on top of abstract categories, humans can also reason on the relationships on top of more concrete distinct entities. For example, in Figure 1.2, humans can tell much more than the top scene is comprised of three objects and the bottom scene depicts a person. They can easily articulate that the yellow block is supporting the blue one thus as the yellow one topples, the blue block will fall as well. Likewise, as the person at the bottom pulls up his arm, the torso and legs will follow the motion but his elbows will remain at the same place.

The ability to generalize across categories with few or even no samples requires leveraging relationships among them; the ability to predict multiple entities in long-term and in details should benefit from understanding the relationships among entities. The thesis proposes to leverage structures in those two aspects: structures among categories and structures among entities.

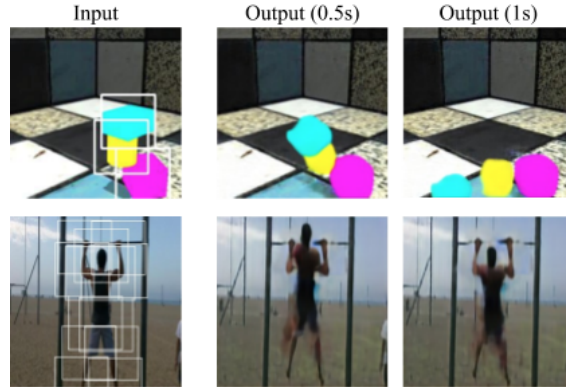


Figure 1.2: Given a still image with entities (objects or joints), human can articulate the future based on relationship among entities. In chapter 3, we propose a compositional method to predict a sequence of future frames. We visualize two frames from the predicted sequence for the given inputs.

In chapter 2, we focus on zero-shot learning task to generate visual classifiers for the novel categories without training images [85]. We build upon the recently introduced Graph Convolutional Network (GCN) to leverage the categorical relationships which are both explicitly specified by knowledge graphs and implicitly specified by languages. Given a learned knowledge graph (KG), our approach takes as input semantic embeddings for each node (representing visual category). After a series of graph convolutions, we predict the visual classifier for each category. During training, the visual classifiers for a few categories are given to learn the GCN parameters. At test time, these filters are used to predict the visual classifiers of unseen categories. We show that our approach is robust to noise in the KG. More importantly, our approach provides significant improvement in performance compared to the current state-of-the-art results (from 2 ~ 3% on some metrics to whopping 20% on a few).

In chapter 3, we focus on the task of video prediction, especially on multi-entities prediction [96]. We observe that a scene is comprised of distinct entities that undergo motion and present an approach that operationalizes this insight by implicitly predicting future states of independent entities while reasoning about interactions among them, and composing future video frames using predicted states. We overcome the inherent multi-modality of the task using a global trajectory-level latent random variables, and show this allows us to sample more diverse and plausible futures compared to commonly used per-timestep latent variables models. We empirically validate our approach against alternate representations choices and ways of incorporating multi-modality. We examine two datasets, one comprising of stacked objects that may fall, and another containing videos of humans performing activities in a gym, and show that our approach allows realistic stochastic video prediction across these diverse settings.

Chapter 2

Zero-Shot via Explicit Semantic Structure among Categories

2.1 Introduction

Consider the animal category “okapi”. Even though we might have never heard of this category or seen visual examples in the past, we can still learn a good visual classifier based on the following description: “zebra-striped four legged animal with a brown torso and a deer-like face”. On the other hand, our current recognition algorithms still operate in closed world conditions: that is, they can only recognize the categories they are trained with. Adding a new category requires collecting thousands of training examples and then retraining the classifiers. To tackle this problem, zero-shot learning is often used.

The key to dealing with the unfamiliar or novel category is to transfer knowledge obtained from familiar classes to describe the unfamiliar classes (generalization). There are two paradigms of transferring knowledge. The first paradigm is to use implicit knowledge representations, i.e. semantic embeddings. In this approach, one learns a vector representation of different categories using text data and then learns a mapping between the vector representation to visual classifier directly [25,63]. However, these methods are limited by the generalization power of the semantic models and the mapping models themselves. It is also hard to learn semantic embeddings from structured information.

The alternative and less-explored paradigm for zero-shot learning is to use explicit knowledge bases or knowledge graphs. In this paradigm, one explicitly represents the knowledge as rules or relationships between objects. These relationships can then be used to learn zero-shot classifiers for new categories. The simplest example would be to learn visual classifiers of compositional categories. Given classifiers of primitive visual concepts as inputs, [62] applies a simple composition rule to generate classifiers for new complex concepts. However, in the general form, the relationships can be more complex than simple compositionality. An interesting question we want to explore is if we can use structured information and complex relationships to learn visual classifiers without seeing any examples.

In this paper, we propose to distill both the implicit knowledge representations (i.e. word embedding) and explicit relationships (i.e. knowledge graph) for learning visual classifiers of novel classes. We build a knowledge graph where each node corresponds to a semantic category. These nodes are linked via relationship edges. The input to each node of the graph is the vector representation (semantic embedding) of each category. We then use Graph Convolutional Network (GCN) [44] to

transfer information (message-passing) between different categories. Specifically, we train a 6-layer deep GCN that outputs the classifiers of different categories.

We focus on the task of image classification. We consider both of the test settings: (a) final test classes being only zero-shot classes (without training classes at test time); (b) at test time the labels can be either the seen or the unseen classes, namely “generalized zero-shot setting” [11,30,90]. We show surprisingly powerful results and huge improvements over classical baselines such as DeVise [25], ConSE [63], and current state-of-the-art [10]. For example, on standard ImageNet with 2-hop setting, 43.7% of the images retrieved by [10] in top-10 are correct. Our approach retrieves 62.4% images correctly. **That is a whopping 18.7% improvement over the current state-of-the-art.** More interestingly, we show that our approach scales amazingly well and giving a significant improvement as we increase the size of the knowledge graph even if the graph is noisy.

2.2 Related Work

With recent success of large-scale recognition systems [75], the focus has now shifted to scaling these systems in terms of categories. As more realistic and practical settings are considered, the need for zero-shot recognition – training visual classifiers without any examples – has increased. Specifically, the problem of mapping text to visual classifiers is very interesting.

Early work on zero-shot learning used attributes [20, 37, 47] to represent categories as vector indicating presence/absence of attributes. This vector representation can then be mapped to learn visual classifiers. Instead of using manually defined attribute-class relationships, Rohrbach et al. [67, 68] mined these associations from different internet sources. Akata et al. [1] used attributes as side-information to learn a semantic embedding which helps in zero-shot recognition. Recently, there have been approaches such as [66] which tries to match Wikipedia text to images by modeling noise in the text description.

With the advancement of deep learning, most recent approaches can be mapped into two main research directions. The first approach is to use semantic embeddings (implicit representations). The core idea is to represent each category with learned vector representations that can be mapped to visual classifiers [9, 10, 25–27, 34, 46, 48, 70, 74, 87, 100, 101]. Socher et al. [74] proposed training two different neural networks for image and language in an unsupervised manner, and then learning a linear mapping between image representations and word embeddings. Motivated by this work, Frome et al. [25] proposed a system called DeViSE to train a mapping from image to word embeddings using a ConvNet and a transformation layer. By using the predicted embedding to perform nearest neighbor search, DeViSE scales up the zero-shot recognition to thousands of classes. Instead of training a ConvNet to predict the word embedding directly, Norouzi et al. [63] proposed another system named ConSE which constructs the image embedding by combining an existing image classification ConvNet and word embedding model. Recently, Changpinyo et al [9] proposed an approach to align semantic and visual manifolds via use of ‘phantom’ classes. They report state-of-the-art results on ImageNet dataset using this approach. One strong shortcoming of these approaches is they do not use any explicit relationships between classes but rather use semantic-embeddings to represent relationships.

The second popular way to distill the knowledge is to use knowledge graph (explicit knowledge representations). Researchers have proposed several approaches on how to use knowledge graphs for object recognition [14, 15, 21, 53, 57, 59, 64, 69, 72, 84, 89]. For example, Salakhutdinov et al. [72] used WordNet to share the representations among different object classifiers so that objects with few training examples can borrow statistical strength from related objects. On the other hand, the knowledge graph can also be used to model the mutual exclusion among different classes. Deng et al. [15] applied these exclusion rules as a constraint in the loss for training object classifiers (e.g. an object will not be a dog and a cat at the same time). They have also shown zero-shot applications

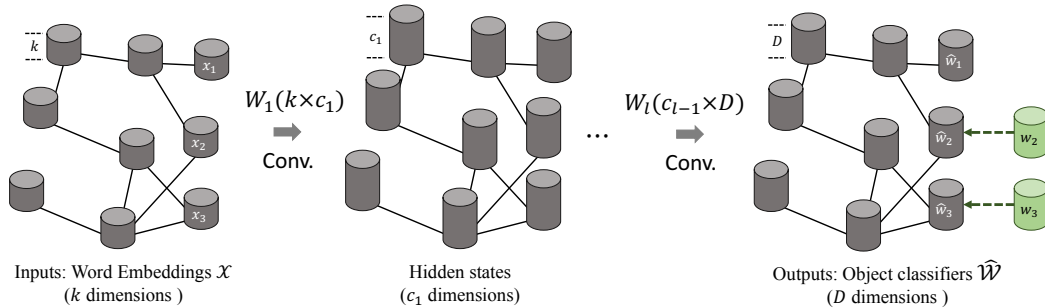


Figure 2.1: An example of our Graph Convolutional Network. It takes word embeddings as inputs and outputs the object classifiers. The supervision comes from the ground-truth classifiers w_2 and w_3 highlighted by green. During testing, we input the same word embeddings and obtain classifier for x_1 as \hat{w}_1 . This classifier will be multiplied with the image features to produce classification scores.

by adding object-attribute relations into the graph. In contrast to these methods of using graph as constraints, our approach used the graph to directly generate novel object classifiers [3, 18, 62].

In our work, we propose to distill information both via semantic embeddings and knowledge graphs. Specifically, given a word embedding of an unseen category and the knowledge graph that encodes explicit relationships, our approach predicts the visual classifiers of unseen categories. To model the knowledge graph, our work builds upon the Graph Convolutional Networks [44]. It was originally proposed for semi-supervised learning in language processing. We extend it to our zero-shot learning problem by changing the model architecture and training loss.

2.3 Approach

Our goal is to distill information from both implicit (word-embeddings) and explicit (knowledge-graph) representations for zero-shot recognition. But what is the right way to extract information? We build upon the recent work on Graph Convolutional Network (GCN) [44] to learn visual classifiers. In the following, we will first introduce how the GCN is applied in natural language processing for classification tasks, and then we will go into details about our approach: applying the GCN with a regression loss for zero-shot learning.

2.3.1 Preliminaries: Graph Convolutional Network

Graph Convolutional Network (GCN) was introduced in [44] to perform semi-supervised entity classification. Given object entities, represented by word embeddings or text features, the task is to perform classification. For example, entities such as “dog” and “cat” will be labeled as “mammal”; “chair” and “couch” will be labeled “furniture”. We also assume that there is a graph where nodes are entities and the edges represent relationships between entities.

Formally, given a dataset with n entities $(X, Y) = \{(x_i, y_i)\}_{i=1}^n$ where x_i represents the word embedding for entity i and $y_i \in \{1, \dots, C\}$ represents its label. In semi-supervised setting, we know the ground-truth labels for the first m entities. Our goal is to infer y_i for the remaining $n - m$ entities, which do not have labels, using the word embedding and the relationship graph. In the relationship graph, each node is an entity and two nodes are linked if they have a relationship in between.

We use a function $F(\cdot)$ to represent the Graph Convolutional Network. It takes all the entity word embeddings X as inputs at one time and outputs the SoftMax classification results for all of them

as $F(X)$. For simplicity, we denote the output for the i th entity as $F_i(X)$, which is a C dimension SoftMax probability vector. In training time, we apply the SoftMax loss on the first m entities, which have labels as

$$\frac{1}{m} \sum_{i=1}^m L_{\text{softmax}}(F_i(X), y_i). \quad (2.1)$$

The weights of $F(\cdot)$ are trained via back-propagation with this loss. During testing time, we use the learned weights to obtain the labels for the $n - m$ entities with $F_i(X), i \in \{m + 1, \dots, n\}$.

Unlike standard convolutions that operate on local region in an image, in GCN the convolutional operations compute the response at a node based on the neighboring nodes defined by the adjacency graph. Mathematically, the convolutional operations for each layer in the network $F(\cdot)$ is represented as

$$Z = \hat{A}X'W \quad (2.2)$$

where \hat{A} is a normalized version of the binary adjacency matrix A of the graph, with $n \times n$ dimensions. X' is the input $n \times k$ feature matrix from the former layer. W is the weight matrix of the layer with dimension $k \times c$, where c is the output channel number. Therefore, the input to a convolutional layer is $n \times k$, and the output is a $n \times c$ matrix Z . These convolution operations can be stacked one after another. A non-linear operation (ReLU) is also applied after each convolutional layer before the features are forwarded to the next layer. For the final convolutional layer, the number of output channels is the number of label classes ($c = C$). For more details, please refer to [44].

2.3.2 GCN for Zero-shot Learning

Our model builds upon the Graph Convolutional Network. However, instead of entity classification, we apply it to the zero-shot recognition with a regression loss. The input of our framework is the set of categories and their corresponding semantic-embedding vectors (represented by $\mathcal{X} = \{x_i\}_{i=1}^n$). For the output, we want to predict the visual classifier for each input category (represented by $\mathcal{W} = \{w_i\}_{i=1}^n$).

Specifically, the visual classifier we want the GCN to predict is a logistic regression model on the fixed pre-trained ConvNet features. If the dimensionality of visual-feature vector is D , each classifier w_i for category i is also a D -dimensional vector. Thus the output of each node in the GCN is D dimensions, instead of C dimensions. In the zero-shot setting, we assume that the first m categories in the total n classes have enough visual examples to estimate their weight vectors. For the remaining $n - m$ categories, we want to estimate their corresponding weight vectors given their embedding vectors as inputs.

One way is to train a neural network (multi-layer perceptron) which takes x_i as an input and learns to predict w_i as an output. The parameters of the network can be estimated using m training pairs. However, generally m is small (in the order of a few hundreds) and therefore, we want to use the explicit structure of the visual world or the relationships between categories to constrain the problem. We represent these relationships as the knowledge-graph (KG). Each node in the KG represents a semantic category. Since we have a total of n categories, there are n nodes in the graph. Two nodes are linked to each other if there is a relationship between them. The graph structure is represented by the $n \times n$ adjacency matrix, A . Instead of building a bipartite graph as [44, 94], we replace all directed edges in the KG by undirected edges, which leads to a symmetric adjacency matrix.

As Fig. 2.1 shows, we use a 6-layer GCN where each layer l takes as input the feature representation from previous layer (Z_{l-1}) and outputs a new feature representation (Z_l). For the first layer the input is \mathcal{X} which is an $n \times k$ matrix (k is the dimensionality of the word-embedding vector). For the final-layer the output feature-vector is $\hat{\mathcal{W}}$ which has the size of $n \times D$; D being the dimensionality of the classifier or visual feature vector.

Loss-function: For the first m categories, we have predicted classifier weights $\hat{\mathcal{W}}_{1\dots m}$ and ground-truth classifier weights learned from training images $\mathcal{W}_{1\dots m}$. We use the mean-square error as the loss function between the predicted and the ground truth classifiers.

$$\frac{1}{m} \sum_{i=1}^m L_{\text{mse}}(\hat{w}_i, w_i). \quad (2.3)$$

During training, we use the loss from the m seen categories to estimate the parameters for the GCN. Using the estimated parameters, we obtain the classifier weights for the zero-shot categories. At test time, we first extract the image feature representations via the pre-trained ConvNet and use these generated classifiers to perform classification on the extracted features.

Test Set	Model	Hit@ k (%)			
		1	2	5	10
High Value Edges	ConSE(5)	6.6	9.6	13.6	19.4
	ConSE(10)	7.0	9.8	14.2	20.1
	ConSE(431)	6.7	9.7	14.9	20.5
	Ours	9.1	16.8	23.2	47.9
All Edges	ConSE(5)	7.7	10.1	13.9	19.5
	ConSE(10)	7.7	10.4	14.7	20.5
	ConSE(616)	7.7	10.5	15.7	21.4
	Ours	10.8	18.4	33.7	49.0

Table 2.1: Top-k accuracy for different models in different settings.

2.3.3 Implementation Details

Our GCN is composed of 6 convolutional layers with output channel numbers as $2048 \rightarrow 2048 \rightarrow 1024 \rightarrow 1024 \rightarrow 512 \rightarrow D$, where D represents the dimension of the object classifier. Unlike the 2-layer network presented in [44], our network is much deeper. As shown in ablative studies, we find that making the network deep is essential in generating the classifier weights. For activation functions, instead of using ReLU after each convolutional layer, we apply LeakyReLU [55, 91] with the negative slope of 0.2. Empirically, we find that LeakyReLU leads to faster convergence for our regression problem.

While training our GCN, we perform L2-Normalization on the outputs of the networks and the ground-truth classifiers. During testing, the generated classifiers of unseen classes are also L2-Normalized. We find adding this constraint important, as it regularizes the weights of all the classifiers into similar magnitudes. In practice, we also find that the last layer classifiers of the ImageNet pre-trained networks are naturally normalized. That is, if we perform L2-Normalization on each of the last layer classifiers during testing, the performance on the ImageNet 2012 1K-class validation set changes marginally ($< 1\%$).

To obtain the word embeddings for GCN inputs, we use the GloVe text model [65] trained on the Wikipedia dataset, which leads to 300-d vectors. For the classes whose names contain multiple words, we match all the words in the trained model and find their embeddings. By averaging these word embeddings, we obtain the class embedding.

2.4 Experiment

We now perform experiments to showcase that our approach: (a) improves the state-of-the-art by a significant margin; (b) is robust to different pre-trained ConvNets and noise in the KG. We use two datasets in our experiments. The first dataset we use is constructed from publicly-available knowledge bases. The dataset consists of relationships and graph from Never-Ending Language Learning (NELL) [7] and images from Never-Ending Image Learning (NEIL) [14]. This is an ideal dataset for: (a) demonstrating that our approach is robust even with automatically learned (and noisy) KG; (b) ablative studies since the KG in this domain is rich, and we can perform ablations on KG as well.

Our final experiments are shown on the standard ImageNet dataset. We use the same settings as the baseline approaches [9, 25, 63] together with the WordNet [61] knowledge graph. We show that our approach surpasses the state-of-the-art methods by a significant margin.

2.4.1 Experiments on NELL and NEIL

Dataset settings. For this experiment, we construct a new knowledge graph based on the NELL [7] and NEIL [14] datasets. Specifically, the object nodes in NEIL correspond to the nodes in NELL. The NEIL dataset offers the sources of images and the NELL dataset offers the common sense knowledge rules. However, the NELL graph is incredibly large¹: it contains roughly 1.7M types of object entities and around 2.4M edges representing the relationships between every two objects. Furthermore, since NELL is constructed automatically, there are noisy edges in the graph. Therefore, we create sub-graphs for our experiments.

The process of constructing this sub-graph is straightforward. We perform Breadth-first search (BFS) starting from the NEIL nodes. We discover paths with maximum length K hops such that the first and last node in the path are NEIL nodes. We add all the nodes and edges in these paths into our sub-graph. We set $K = 7$ during BFS because we discover a path longer than 7 hops will cause the connection between two objects noisy and unreasonable. For example, “jeep” can be connected to “deer” in a long path but they are hardly semantically related.

Note that each edge in NELL has a confidence value that is usually larger than 0.9. For our experiments, we create two different versions of sub-graphs. The first smaller version is a graph with high value edges (larger than 0.999), and the second one used all the edges regardless of their confidence values. The statistics of the two sub-graphs are summarized in Table 2.2. For the larger sub-graph, we have 14K object nodes. Among these nodes, 704 of them have corresponding images in the NEIL database. We use 616 classes for training our GCN and leave 88 classes for testing. Note that these 88 testing classes are randomly selected among the classes that have no overlap with the 1000 classes in the standard ImageNet classification dataset. The smaller knowledge graph is around half the size of the larger one. We use the same 88 testing classes in both settings

Dataset	All Nodes	NEIL Nodes (Train/Test)	Edges
High Value Edges	8819	431/88	40810
All Edges	14612	616/88	96772

Table 2.2: Dataset Statistics: Two different sizes of knowledge graphs in our experiment.

¹<http://rtw.ml.cmu.edu/>

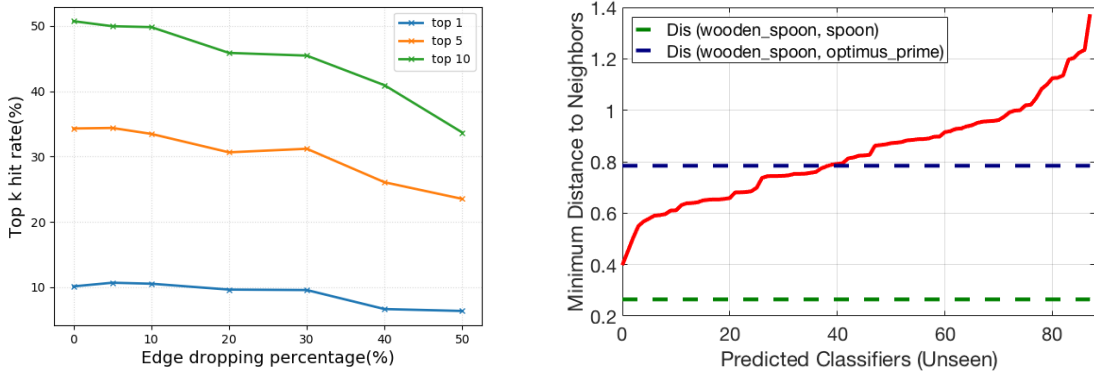


Figure 2.2: Left: We randomly drop 5% to 50% of the edges in the “All Edges” graph and show the top-1, top-5 and top-10 accuracies. Right: We compute the minimum Euclidean distances between predicted and training classifiers. The distances are plotted by sorting them from small to large.

Training details. For training the ConvNet on NEIL images, we use the 310K images associated with the 616 training classes. The evaluation is performed on the randomly selected 12K images associated with the 88 testing classes, i.e. all images from the training classes are excluded during testing. We fine-tune the ImageNet pre-trained VGGM [12] network architecture with relatively small *fc7* outputs (128-dimension). Thus the object classifier dimension in *fc8* is 128. For training our GCN, we use the ADAM [41] optimizer with learning rate 0.001 and weight decay 0.0005. We train our GCN for 300 epochs for every experiment.

Baseline method. We compare our method with one of the state-of-the-art methods, ConSE [63], which shows slightly better performance than DeVise [25] in ImageNet. As a brief introduction, ConSE first feedforwards the test image into a ConvNet that is trained only on the training classes. With the output probabilities, ConSE selects top T predictions $\{p_i\}_{i=1}^T$ and the word embeddings $\{x_i\}_{i=1}^T$ [60] of these classes. It then generates a new word embedding by weighted averaging the T embeddings with the probability $\frac{1}{T} \sum_{i=1}^T p_i x_i$. This new embedding is applied to perform nearest neighbors in the word embeddings of the testing classes. The top retrieved classes are selected as the final result. We enumerate different values of T for evaluations.

Quantitative Results. We perform evaluations on the task of 88 unseen categories classification. Our metric is based on the percentage of correctly retrieved test data (out of top k retrievals) for a given zero-shot class. The results are shown in Table 2.1. We evaluate our method on two different sizes of knowledge graphs. We use “High Value Edges” to denote the knowledge graph constructed based on high confidence edges. “All Edges” represents the graph constructed with all the edges. We denote the baseline [63] as “ConSE(T)” where we set T to be 5, 10 and the number of training classes.

Our method outperforms the ConSE baseline by a large margin. In the “All Edges” dataset, our method outperforms ConSE 3.6% in top-1 accuracy. **More impressively, the accuracy of our method is almost 2 times as that of ConSE in top-2 metric and even more than 2 times in top-5 and top-10 accuracies.** These results show that using knowledge graph with word embeddings in our method leads to much better result than the state-of-the-art results with word embeddings only.

From small to larger graph. In addition to improving performance in zero-shot recognition, our method obtains more performance gain as our graph size increases. As shown in Table 2.1, our method performs better by switching from the small to larger graph. Our approach has obtained 2 ~ 3% improvements in all the metrics. On the other hand, there is little to no improvements in ConSE performance. It also shows that the KG does not need to be hand-crafted or cleaned. Our approach is able to robustly handle the errors in the graph structure.

Resilience to Missing Edges We explore how the performance of our model changes if we randomly drop 5% to 50% of the edges in the “All Edges” graph. As Fig. 2.2(Left) shows, by dropping from 5% to 10% of edges, the performance of our model changes negligibly. This is mainly because the knowledge graph can have redundant information with 14K nodes and 97K edges connecting them. This again implies that our model is robust to small noisy changes in the graph. As we start deleting more than 30% of the edges, the accuracies drop drastically. This indicates that the performance of our model is highly correlated to the size of the knowledge graph.

Random Graph? It is clear that our approach can handle noise in the graph. But does any random graph work? To demonstrate that the structure of the graph is still critical we also created some trivial graphs: (i) star model: we create a graph with one single root node and only have edges connecting object nodes to the root node; (ii) random graph: all nodes in the graph are randomly connected. Table 2.3 shows the results. It is clear that all the numbers are close to random guessing, which means a reasonable graph plays an important role and a random graph can have negative effects on the model.

Test Set	Trivial KG	Hit@ <i>k</i> (%)			
		1	2	5	10
All Edges	Star Model	1.1	1.6	4.8	9.7
	Random Graph	1.0	2.2	5.6	11.3

Table 2.3: Top-k accuracy on trivial knowledge graphs we create.

How important is the depth of GCN? We show that making the Graph Convolutional Network deep is critical in our problem. We show the performance of using different numbers of layers for our model on the “All Edges” knowledge graph shown in Table 2.4. For the 2-layer model we use 512 hidden neurons, and the 4-layer model has output channel numbers as 2048 → 1024 → 512 → 128. We show that the performance keeps increasing as we make the model deeper from 2-layer to 6-layer. The reason is that increasing the times of convolutions is essentially increasing the times of message passing between nodes in the graph. However, we do not observe much gain by adding more layers above the 6-layer model. One potential reason might be that the optimization becomes harder as the network goes deeper.

Test Set	Model	Hit@ <i>k</i> (%)			
		1	2	5	10
All Edges	Ours (2-layer)	5.3	8.7	15.5	24.3
	Ours (4-layer)	8.2	13.5	27.1	41.8
	Ours (6-layer)	10.8	18.4	33.7	49.0

Table 2.4: Top-k accuracy with different depths of our model.

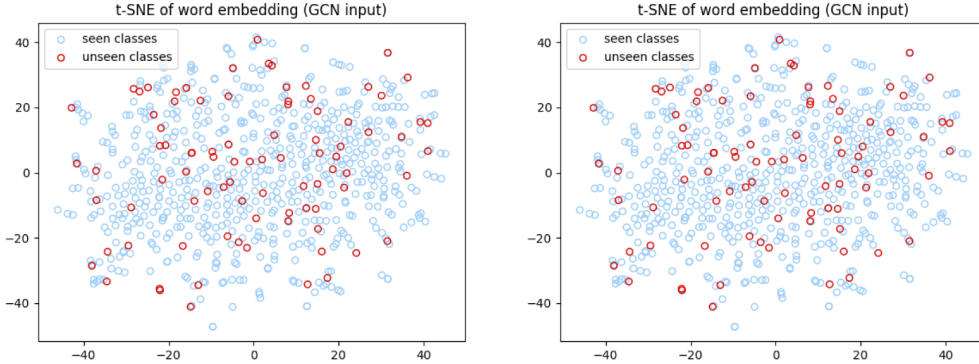


Figure 2.3: t-SNE visualizations for our word embeddings and GCN output visual classifiers in the “All Edges” dataset. The test classes are shown in red.

Is our network just copying classifiers as outputs? Even though we show our method is better than ConSE baseline, is it possible that it learns to selectively copy the nearby classifiers? To show our method is not learning this trivial solution, we compute the Euclidean distance between our generated classifiers and the training classifiers. More specifically, for a generated classifier, we compare it with the classifiers from the training classes that are at most 3-hops away. We calculate the minimum distance between each generated classifier and its neighbors. We sort the distances for all 88 classifiers and plot Fig. 2.2(Right). As for reference, the distance between “wooden_spoon” and “spoon” classifiers in the training set is 0.26 and the distance between “wooden_spoon” and “optimus_prime” is 0.78. We can see that our predicted classifiers are quite different from its neighbors.

Test Set	Model	ConvNets	Hit@k (%)				
			1	2	5	10	20
2-hops	ConSE [9]	Inception-v1	8.3	12.9	21.8	30.9	41.7
	ConSE(us)	Inception-v1	12.4	18.4	25.3	28.5	31.8
	SYNC [9]	Inception-v1	10.5	17.7	28.6	40.1	52.0
	EXEM [10]	Inception-v1	12.5	19.5	32.3	43.7	55.2
	Ours	Inception-v1	18.5	31.3	50.1	62.4	72.0
	Ours	ResNet-50	19.8	33.3	53.2	65.4	74.6
3-hops	ConSE [9]	Inception-v1	2.6	4.1	7.3	11.1	16.4
	ConSE(us)	Inception-v1	3.2	4.9	7.6	9.7	11.4
	SYNC [9]	Inception-v1	2.9	4.9	9.2	14.2	20.9
	EXEM [10]	Inception-v1	3.6	5.9	10.7	16.1	23.1
	Ours	Inception-v1	3.8	6.9	13.1	18.8	26.0
	Ours	ResNet-50	4.1	7.5	14.2	20.2	27.7
All	ConSE [9]	Inception-v1	1.3	2.1	3.8	5.8	8.7
	ConSE(us)	Inception-v1	1.5	2.2	3.6	4.6	5.7
	SYNC [9]	Inception-v1	1.4	2.4	4.5	7.1	10.9
	EXEM [10]	Inception-v1	1.8	2.9	5.3	8.2	12.2
	Ours	Inception-v1	1.7	3.0	5.8	8.4	11.8
	Ours	ResNet-50	1.8	3.3	6.3	9.1	12.7

(a) Top-k accuracy for different models when testing on only unseen classes.

Test Set	Model	ConvNets	Hit@k (%)				
			1	2	5	10	20
2-hops (+1K)	DeViSE [25]	AlexNet	0.8	2.7	7.9	14.2	22.7
	ConSE [63]	AlexNet	0.3	6.2	17.0	24.9	33.5
	ConSE(us)	Inception-v1	0.2	7.8	18.1	22.8	26.4
	ConSE(us)	ResNet-50	0.1	11.2	24.3	29.1	32.7
	Ours	Inception-v1	7.9	18.6	39.4	53.8	65.3
	Ours	ResNet-50	9.7	20.4	42.6	57.0	68.2
3-hops (+1K)	DeViSE [25]	AlexNet	0.5	1.4	3.4	5.9	9.7
	ConSE [63]	AlexNet	0.2	2.2	5.9	9.7	14.3
	ConSE(us)	Inception-v1	0.2	2.8	6.5	8.9	10.9
	ConSE(us)	ResNet-50	0.2	3.2	7.3	10.0	12.2
	Ours	Inception-v1	1.9	4.6	10.9	16.7	24.0
	Ours	ResNet-50	2.2	5.1	11.9	18.0	25.6
All (+1K)	DeViSE [25]	AlexNet	0.3	0.8	1.9	3.2	5.3
	ConSE [63]	AlexNet	0.2	1.2	3.0	5.0	7.5
	ConSE(us)	Inception-v1	0.1	1.3	3.1	4.3	5.5
	ConSE(us)	ResNet-50	0.1	1.5	3.5	4.9	6.2
	Ours	Inception-v1	0.9	2.0	4.8	7.5	10.8
	Ours	ResNet-50	1.0	2.3	5.3	8.1	11.7

(b) Top-k accuracy for different models when testing on both seen and unseen classes (a more practical and generalized setting).

Table 2.5: Results on ImageNet. We test our model on 2 different settings over 3 different datasets.

Are the outputs only relying on the word embeddings? We perform t-SNE [56] visualizations to show that our output classifiers are not just derived from the word embeddings. We show the t-SNE [56] plots of both the word embeddings and the classifiers of the seen and unseen classes in the “All Edges” dataset. As Fig. 2.3 shows, we have very different clustering results between the word embeddings and the object classifiers, which indicates that our GCN is not just learning a direct projection from word embeddings to classifiers.

2.4.2 Experiments on WordNet and ImageNet

We now perform our experiments on a much larger-scale ImageNet [71] dataset. We adopt the same train/test split settings as [25, 63]. More specifically, we report our results on 3 different test datasets: “2-hops”, “3-hops” and the whole “All” ImageNet set. These datasets are constructed according to how similar the classes are related to the classes in the ImageNet 2012 1K dataset. For example, “2-hops” dataset (around 1.5K classes) includes the classes from the ImageNet 2011 21K set which are semantically very similar to the ImageNet 2012 1K classes. “3-hops” dataset (around 7.8K classes) includes the classes that are within 3 hops of the ImageNet 2012 1K classes, and the “All” dataset includes all the labels in ImageNet 2011 21K. There are no common labels between the ImageNet 1K class and the classes in these 3-dataset. It is also obvious to see that as the number of class increases, the task becomes more challenging.

As for knowledge graph, we use the sub-graph of the WordNet [61], which includes around 30K object nodes².

Training details. Note that to perform testing on 3 different test sets, we only need to train one set of ConvNet and GCN. We use two different types of ConvNets as the base network for computing visual features: Inception-v1 [76] and ResNet-50 [31]. Both networks are pre-trained using the ImageNet 2012 1K dataset and no fine-tuning is required. For Inception-v1, the output feature of the second to the last layer has 1024 dimensions, which leads to $D = 1024$ object classifiers in the last layer. For ResNet-50, we have $D = 2048$. Except for the changes of output targets, other settings of training GCN remain the same as those of the previous experiments on NELL and NEIL. It is worthy to note that our GCN model is robust to different sizes of outputs. The model shows consistently better results as the representation (features) improves from Inception-v1 (68.7% top-1 accuracy in ImageNet 1K val set) to ResNet-50 (75.3%).

We evaluate our method with the same metric as the previous experiments: the percentage of hitting the ground-truth labels among the top k predictions. However, instead of only testing with the unseen object classifiers, we include both training and the predicted classifiers during testing, as suggested by [25, 63]. Note that in these two settings of experiments, we still perform testing on the same set of images associated with unseen classes only.

Testing without considering the training labels. We first perform experiments excluding the classifiers belonging to the training classes during testing. We report our results in Table. 2.5a. We compare our results to the recent state-of-the-art methods SYNC [9] and EXEM [10]. We show experiments with the same pre-trained ConvNets (Inception-v1) as [9, 10]. Due to unavailability of their word embeddings for all the nodes in KG, we use a different set of word embeddings (GloVe) ,which is publicly available.

Therefore, we first investigate if the change of word-embedding is crucial. We show this via the ConSE baseline. Our re-implementation of ConSE, shown as “ConSE(us)” in the table, uses the GloVe

²<http://www.image-net.org/explore>

whereas the ConSE method implemented in [9, 10] uses their own word embedding. We see that both approaches have similar performance. Ours is slightly better in top-1 accuracy while the one in [9, 10] is better in top-20 accuracy. Thus, with respect to zero-shot learning, both word-embeddings seem equally powerful.

We then compare our results with SYNC [9] and EXEM [10]. With the same pre-trained ConvNet Inception-v1, our method outperforms almost all the other methods on all the datasets and metrics. On the “2-hops” dataset, our approach outperforms all methods with a large margin: around 6% on top-1 accuracy and 17% on top-5 accuracy. On the “3-hops” dataset, our approach is consistently better than EXEM [10] around 2 ~ 3% from top-5 to top-20 metrics.

By replacing the Inception-v1 with the ResNet-50, we obtain another performance boost in all metrics. For the top-5 metric, our final model outperforms the state-of-the-art method EXEM [10] by a whopping 20.9% in the “2-hops” dataset, 3.5% in the “3-hops” dataset and 1% in the “All” dataset. Note that the gain is diminishing because the task increases in difficulty as the number of unseen classes increases.

Model	Word Embedding	Hit@ <i>k</i> (%)				
		1	2	5	10	20
[97]	GloVe	7.8	11.5	17.2	21.2	25.6
Ours	GloVe	18.5	31.3	50.1	62.4	72.0
[97]	FastText	9.8	16.4	27.8	37.6	48.4
Ours	FastText	18.7	30.8	49.6	62.0	71.5
[97]	GoogleNews	13.0	20.6	33.5	44.1	55.2
Ours	GoogleNews	18.3	31.6	51.1	63.4	73.0

Table 2.6: Results with different word embeddings on ImageNet (2 hops), corresponding to the experiments in Table 2.5a.

Sensitivity to word embeddings. Is our method sensitive to word embeddings? What will happen if we use different word embeddings as inputs? We investigate 3 different word embeddings including GloVe [65] (which is used in the other experiments in the paper), FastText [40] and word2vec [60] trained with GoogleNews. As for comparisons, we have also implemented the method in [97] which trains a direct mapping from word embeddings to visual features without knowledge graphs. We use the Inception-v1 ConvNet to extract visual features. We show the results on ImageNet (with the 2-hops setting same as Table 2.5a). We can see that [97] highly relies on the quality of the word embeddings (top-5 results range from 17.2% to 33.5%). On the other hand, our top-5 results are stably around 50% and are much higher than [97]. With the GloVe word embeddings, our approach has a **relative improvement of almost 200%** over [97]. This again shows graph convolutions with knowledge graphs play a significant role in improving zero-shot recognition.

Testing with the training classifiers. Following the suggestions in [25, 63], a more practical setting for zero-shot recognition is to include both seen and unseen category classifiers during testing. We test our method in this generalized setting. Since there are very few baselines available for this setting of experiment, we can only compare the results with ConSE and DeViSE. We have also re-implemented the ConSE baselines with both Inception-v1 and ResNet-50 pre-trained networks. As Table 2.5b shows our method almost doubles the performance compared to the baselines on every metric and all 3-datasets. Moreover, we can still see the boost in of performance by switching the pre-trained Inception-v1 network to ResNet-50.




Test Image	ConSE (10)	Ours
	panthera tigris(train) tiger cat (train) felis onca (train) leopard (train) tiger shark (train)	tigress (test) bengal tiger (test) panthera tigris (train) tiger cub (test) tiger cat (train)
	rock beauty (train) ringlet (train) flagpole (train) large slipper (test) yellow slipper (train)	butterfly fish (test) rock beauty (train) damsel fish (test) atoll (test) barrier reef (test)
	tractor (train) reaper (train) thresher (train) trailer truck (train) motortruck (test)	tracked vehicle (test) tractor (train) propelled vehicle (test) reaper (train) forklift (train)

Figure 2.4: Visualization of top 5 prediction results for 3 different images. The correct prediction results are highlighted by red bold characters. The unseen classes are marked with a red “test” in the bracket. Previously seen classes have a plain “train” in the bracket.

Lastly, we provide a comprehensive comparison for alternative choices of visual feature and word embedding in table 2.7 2.8. We replace the input word embedding GloVe [65] with the recent proposed FastText [40] while replacing Inception-v1 with ResNet-50. We show that our GCN model is robust to different word embedding inputs while all methods benefits from better visual features.

Visualizations. We finally perform visualizations using our model and ConSE with $T = 10$ in Fig. 2.4 (Top-5 prediction results). We can see that our method significantly outperforms ConSE(10) in these examples. Although ConSE(10) still gives reasonable results in most cases, the output labels are biased to be within the training labels. On the other hand, our method outputs the unseen classes as well.

2.5 Discussion

We have presented an approach for zero-shot recognition using the semantic embeddings of a category and the knowledge graph that encodes the relationship of the novel category to familiar categories. Our work also shows that a knowledge graph provides supervision to learn meaningful classifiers on top of semantic embeddings. Our results indicate a significant improvement over current state-of-the-art.

Test Set	Model	ConvNets	Word Embedding	Hit@ <i>k</i> (%)				
				1	2	5	10	20
2-hops	ConSE [9]	Inception-v1	word2vec	8.3	12.9	21.8	30.9	41.7
	ConSE(us)	Inception-v1	GloVe	12.4	18.4	25.3	28.5	31.8
	ConSE(us)	Inception-v1	FastText	11.0	16.5	25.9	31.8	39.0
	SYNC [9]	Inception-v1	word2vec	10.5	17.7	28.6	40.1	52.0
	EXEM [10]	Inception-v1	word2vec	12.5	19.5	32.3	43.7	55.2
	Ours	Inception-v1	GloVe	18.5	31.3	50.1	62.4	72.0
	Ours	Inception-v1	FastText	18.7	30.8	49.6	62.0	71.5
	Ours	ResNet-50	GloVe	19.8	33.3	53.2	65.4	74.6
	Ours	ResNet-50	FastText	19.4	32.3	52.6	65.4	74.6
3-hops	ConSE [9]	Inception-v1	word2vec	2.6	4.1	7.3	11.1	16.4
	ConSE(us)	Inception-v1	GloVe	3.2	4.9	7.6	9.7	11.4
	ConSE(us)	Inception-v1	FastText	3.1	4.9	8.4	11.7	15.5
	SYNC [9]	Inception-v1	word2vec	2.9	4.9	9.2	14.2	20.9
	EXEM [10]	Inception-v1	word2vec	3.6	5.9	10.7	16.1	23.1
	Ours	Inception-v1	GloVe	3.8	6.9	13.1	18.8	26.0
	Ours	Inception-v1	FastText	3.7	6.7	12.8	18.6	25.7
	Ours	ResNet-50	GloVe	4.1	7.5	14.2	20.2	27.7
	Ours	ResNet-50	FastText	4.0	7.3	13.8	20.1	27.7
All	ConSE [9]	Inception-v1	word2vec	1.3	2.1	3.8	5.8	8.7
	ConSE(us)	Inception-v1	GloVe	1.5	2.2	3.6	4.6	5.7
	ConSE(us)	Inception-v1	FastText	1.5	2.4	4.2	5.9	8.0
	SYNC [9]	Inception-v1	word2vec	1.4	2.4	4.5	7.1	10.9
	EXEM [10]	Inception-v1	word2vec	1.8	2.9	5.3	8.2	12.2
	Ours	Inception-v1	GloVe	1.7	3.0	5.8	8.4	11.8
	Ours	Inception-v1	FastText	1.6	2.9	5.6	8.2	11.7
	Ours	ResNet-50	GloVe	1.8	3.3	6.3	9.1	12.7
	Ours	ResNet-50	FastText	1.8	3.2	6.1	9.0	12.8

Table 2.7: Results on ImageNet. Top-k accuracy for different models when testing on only unseen classes.

Test Set	Model	ConvNets	Word Embedding	Hit@ <i>k</i> (%)				
				1	2	5	10	20
2-hops (+1K)	DeViSE [25]	AlexNet	word2vec	0.8	2.7	7.9	14.2	22.7
	ConSE [63]	AlexNet	word2vec	0.3	6.2	17.0	24.9	33.5
	ConSE(us)	Inception-v1	GloVe	0.2	7.8	18.1	22.8	26.4
	ConSE(us)	Inception-v1	FastText	0.1	7.7	19.2	26.5	33.4
	ConSE(us)	ResNet-50	GloVe	0.1	11.2	24.3	29.1	32.7
	ConSE(us)	ResNet-50	FastText	0.1	8.7	21.5	29.2	36.3
	Ours	Inception-v1	GloVe	7.9	18.6	39.4	53.8	65.3
	Ours	Inception-v1	FastText	8.1	18.6	39.1	53.3	64.9
	Ours	ResNet-50	GloVe	9.7	20.4	42.6	57.0	68.2
	Ours	ResNet-50	FastText	9.8	20.1	41.9	56.6	68.2
3-hops (+1K)	DeViSE [25]	AlexNet	word2vec	0.5	1.4	3.4	5.9	9.7
	ConSE [63]	AlexNet	word2vec	0.2	2.2	5.9	9.7	14.3
	ConSE(us)	Inception-v1	GloVe	0.2	2.8	6.5	8.9	10.9
	ConSE(us)	Inception-v1	FastText	0.1	2.4	6.6	10.0	14.1
	ConSE(us)	ResNet-50	GloVe	0.2	3.2	7.3	10.0	12.2
	ConSE(us)	ResNet-50	FastText	0.1	2.8	7.5	11.3	15.7
	Ours	Inception-v1	GloVe	1.9	4.6	10.9	16.7	24.0
	Ours	Inception-v1	FastText	1.9	4.6	10.7	16.6	23.8
	Ours	ResNet-50	GloVe	2.2	5.1	11.9	18.0	25.6
	Ours	ResNet-50	FastText	2.3	5.1	11.8	18.0	25.7
All (+1K)	DeViSE [25]	AlexNet	word2vec	0.3	0.8	1.9	3.2	5.3
	ConSE [63]	AlexNet	word2vec	0.2	1.2	3.0	5.0	7.5
	ConSE(us)	Inception-v1	GloVe	0.1	1.3	3.1	4.3	5.5
	ConSE(us)	Inception-v1	FastText	0.0	1.2	3.4	5.2	7.5
	ConSE(us)	ResNet-50	GloVe	0.1	1.5	3.5	4.9	6.2
	ConSE(us)	ResNet-50	FastText	0.1	1.4	3.9	5.9	8.4
	Ours	Inception-v1	GloVe	0.9	2.0	4.8	7.5	10.8
	Ours	Inception-v1	FastText	0.9	2.0	4.7	7.4	10.8
	Ours	ResNet-50	GloVe	1.0	2.3	5.3	8.1	11.7
	Ours	ResNet-50	FastText	1.0	2.3	5.3	8.1	11.9

Table 2.8: Results on ImageNet. Top-k accuracy for different models when testing on both seen and unseen classes (a more practical and generalized setting).

Chapter 3

Video Prediction via Implicit Physical Structure among Entities

3.1 Introduction

A single image of a scene allows us humans to make a remarkable number of judgments about the underlying world. For example, consider the two images on the left in Fig 1.2. We can easily infer that the top image depicts some stacked blocks, and the bottom shows a human with his arms raised. While these inferences showcase our ability to understand what is, even more remarkably, we are capable of predicting what will happen next. For example, not only do we know that there are stacked blocks in the top image, we understand that the blue and yellow ones will topple and fall to the left. Similarly, we know that the person in the bottom image will lift his torso while keeping his hands in place. In this work, we aim to build a model that can do the same – from a *single* (annotated) image of a scene, predict at a pixel level, what the future will be.

A key factor in the ability to make these predictions is that we understand scenes in terms of ‘entities’, that can move and interact e.g. the blocks are separate objects that move; the human body’s motion can similarly be understood in terms of the correlated motion of the limbs. We operationalize this ideology and present an approach that instead of directly predicting future frames, learns to predict the future locations and appearance of the entities in the scene, and via these composes a prediction of the future frame. The modeling of appearance and the learned composition allows our method to leverage the benefits of independent per-entity representations while allowing for reasoning in pose changes or overlap/occlusions in pixel space.

Although our proposed factorization allows learning models capable of predicting the future frames via entity-based reasoning, this task of inferring future frames from a single input image is fundamentally ill-posed. To allow for the inherent multi-modality of the prediction space, we propose to use a trajectory-level latent random variable that implicitly captures the ambiguities over the whole video and train a future predictor conditioned of this latent variable. We demonstrate that modeling the ambiguities using this single latent variable instead of per-timestep random variables allows us to make more realistic predictions as well as sample diverse plausible futures.

We validate our approach using two datasets where the ‘entities’ either represent distinct objects, or human body joints, and demonstrate that the same method allows for predicting future frames across these diverse settings. We demonstrate: (a) the benefits of our proposed entity-level factorization; (b) ability of the corresponding learned decoder to generate future frames; (c) capability to sample

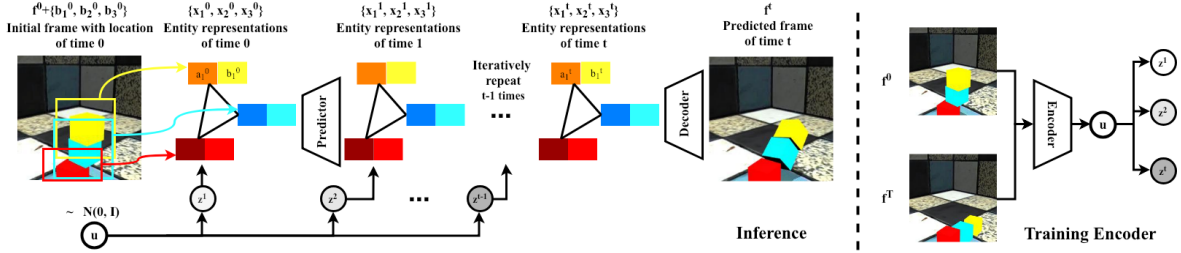


Figure 3.1: Our model takes as input an image with known/detected location of entities. Each entity is represented as its location and an implicit feature. Given the current entity representations and a sampled latent variable, our prediction module predicts the representations at the next time step. Our learned decoder composes the predicted representations to an image representing the predicted future. During training, a latent encoder module is used to infer the distribution over the latent variables using the initial and final frames.

different futures.

3.2 Related Work

Modeling Physical Interaction. Many recent works [5, 8, 33, 43, 73, 86] study modeling multiple objects in physical systems. Similar to us, they reason using the relationship between objects, and can predict the trajectories over a long time horizon. However, these approaches typically model deterministic processes under simple visual (or often only state based) input, while often relying on observed sequences instead of a single frame. Although some recent works take raw image as input [23, 86], they also only make prediction in state, and not pixel space. In contrast to these approaches, while we also use insights based on modeling physical interaction, we show results for video frame generation in a stochastic setup, and therefore also need to (implicitly) reason about other properties such as shape, lighting, color. Lastly, a related line of work is to predict stability of configurations [29, 39, 49–51]. Our video forecasting task also requires this understanding, but we do not pursue this as the end goal.

Video Factorization. It is challenging to directly predict pixels due to high dimensionality of the prediction space, and several methods have been used to factorize this output space [17, 78, 79, 81]. The main idea is to separate dynamic foreground from static background and generate pixels correspondingly. While these approaches show promising results to efficiently model one object motion, we show the benefits of modeling multiple entities and their interactions.

Another insight has been to instead model the output space differently, e.g. optical flow [52, 82], or motion transformation [13, 22, 38, 92]. This enables generating more photo-realistic images for shorter sequences, but may not be applicable for longer generation as new content becomes visible, and we therefore pursue direct pixel generation. Another line of work proposes to predict future in a pre-defined structured representation space, such as human pose [80, 83]. While our approach also benefits from predicting an intermediate structured representations, it is not our end goal as we aim to generate pixels from this representation.

Object-centric video prediction. A line of work explicitly enumerates the state of each object as location, velocity, mass, etc, then applies planning algorithm to unroll movement under reward [35, 45], or leverage Newtonian dynamics [88, 95]. However, these explicit representation based methods may

not be applicable when the state space is hard to define, or pixel-wise predictions are not easily inferred given such a state e.g. human motions on complex background.

Stochastic prediction. Predicting the future is an inherently multi-modal task. Given a still image or a sequence of frames, there are multiple plausible futures that could happen. The uncertainty is usually encoded as a sequence of latent variables, which are then used in a generative model such as GAN [28] based [13, 58, 78, 81], or, similar to ours, VAE [42] based [16, 82]. These methods [16, 24, 93] often leverage an input sequence instead of a single frame, which helps reduce the ambiguities. Further, the latent variables are either per-timestep [16], or global [4, 93] whereas our model leverages a global latent variable, which in turn induces per-timestep variables.

3.3 Approach

Given an input image along with (known or detected) locations of the entities present, our goal is to predict a sequence of future frames. Formally, given a starting frame f^0 and the location of N entities $\{b_n^0\}_{n=1}^N$, we aim to generate T future frames f^1, f^2, \dots, f^T . This task is challenging mainly for two reasons: a) the scene may comprise of multiple entities, making it necessary to account for their different dynamics and interactions, and b) the inherently multi-modal nature of the prediction task.

To overcome the first challenge, our insight is that instead of modeling how the scene changes as a whole, we should pursue prediction by modeling how the entities present change. We do so using an *entity predictor* that predicts per-entity representations: $\{x_n^t\}_{n=1}^N \equiv \{(b_n^t, a_n^t)\}_{n=1}^N$, where b_n^t denotes the predicted location, and a_n^t denotes predicted features that implicitly capture appearance for each entity. While this factorization allows us to efficiently predict the future in terms of these entities, an additional step is required to infer pixels. We do so using a *frame decoder* that is able to retain the properties of each entity, respect the predicted location, while also resolving the conflicts e.g. occlusions when composing the image.

To account for the fundamental multi-modality in the task, we incorporate a global random latent variable u that implicitly captures the ambiguities across the whole video. This latent variable u , in turn deterministically (via a learned network) yields per-timestep latent variables z_t which aid the per-timestep future predictions. Concretely, the predictor \mathcal{P} takes as input the per-entity representation $\{x_n^t\}$ along with the latent variable z_t , and predicts the entity representations at the next timestep $\{x_n^{t+1}\} \equiv \mathcal{P}(\{x_n^t\}, z_t)$. The decoder \mathcal{D} , using these predictions (and the initial frame f^0 to allow modeling background), composes the predicted frame $f^t \equiv \mathcal{D}(\{x_n^t\}, f^0)$.

We train our model to maximize the likelihood of the training sequences, comprising of terms for both the frames and the entity locations. As is often the case with optimizing likelihood in models with unobserved latent variable models e.g. VAEs [42], directly maximizing likelihood is intractable, and we therefore maximize a variational lower bound. Towards this, we train another module, a *latent encoder*, which predicts a distribution over the latent variable u using the target video. Note that the annotation of future frames/locations, as well as the latent encoder, are all only used during training. During inference, however, as illustrated in Fig 3.1, we take in input only a single frame along with (predicted/known) locations of the entities present, and can generate multiple plausible future frames. We first describe the predictor, decoder, and encoder modules in more detail, and then present the overall training objective.

3.3.1 Entity Predictor

Given per-entity locations and implicit appearance features $\{x_n^t\}_{n=1}^N \equiv \{(b_n^t, a_n^t)\}_{n=1}^N$, the predictor outputs the predictions for the next time step using the latent variable z_t . An iterative application of this

predictor therefore allows us to predict the future frames for the entire sequence using the encodings from the initial frame. To obtain this initial input to the predictor i.e. the entity encodings at the first time step $\{x_n^0\}_{n=1}^N$, we use the known/detected entity locations $\{b_n^0\}$, and extract the appearance features $\{a_n^0\}$ using a standard ResNet-18 CNN [32] on the cropped region from f^0 .

While the predictor \mathcal{P} infers per-entity features, the prediction mechanism should also allow for the interaction among these entities rather than predicting each of them independently e.g. a block may or may not fall depending on the other ones around it. To enable this, we leverage a model in the graph neural network family, in particular based on ‘Interaction Networks’ which take in a graph $G = (V, E)$ with associated features for each node, and update these via iterative message passing and message aggregation. See [6] for a more detailed review. Our predictor \mathcal{P} that infers $\{x_n^{t+1}\}$ from $(\{x_n^t\}, z_t)$ comprises of 4 interaction blocks, where the first block takes as input the entity encodings concatenated with the latent feature: $\{x_n^t \oplus z_t\}_{n=1}^N$. Each of these blocks performs a message passing iteration using the underlying graph, and the final block outputs predictions for the entity features for the next timestep $\{x_n^t\}_{n=1}^N \equiv \{(b_n^t, a_n^t)\}_{n=1}^N$. This graph can either be fully connected as with our synthetic data experiments, or more structured e.g. skeleton in our human video prediction experiments. See appendix for more details on the message passing operations.

Although our prediction module falls under the same umbrella as Interaction Networks(IN) [5], which are in turn related to Graph Convolution Networks(GCN) [43], there are subtle differences, both in the architecture and application. While [5] use a single interaction block to update node features, we found that stacking multiple interaction blocks for each timestep is particularly helpful. In contrast to GCNs which use a predefined mechanism to compute edge weights and use linear operations for messages, we find that using non-linear functions as messages allows better performance. Finally, while existing approaches do apply variants of GNNs for future prediction, these are restricted to predefined state-spaces as opposed to pixels, and do not account for uncertainties using latent variables.

3.3.2 Frame Decoder

The decoder aims to generate pixels of the frame f^t from a set of predicted entity representations. While the entity representations capture the moving aspects of the scene, we also need to incorporate the static background, and additionally use the initial frame f^0 to do so. Our decoder \mathcal{D} , as depicted in Fig 3.2, therefore predicts $f^t \equiv \mathcal{D}(\{x_n^t\}, f^0)$. To compose frames from this factored input representation, there are several aspects that our decoder must consider: a) the predicted location of the entities should be respected, b) the per-entity representations may need to be fused e.g. when entities occlude each other, and c) different parts of background may become visible as objects move.

To account for the predicted location of the entities when generating images, we propose to decode a normalized spatial representation for each entity, and warp it to the image coordinates using the predicted 2D locations. To allow for the occlusions among entities, we predict an additional soft mask channel for each entity, where the value of masks are supposed to capture the visibility of the entities. Lastly, we overlay the (masked) spatial features predicted via the entities onto a canvas containing features from the initial frame f^0 , and then predict the future frame pixels using this composed feature.

More formally, let us denote by ϕ_{bg} the spatial features predicted from the frame f^0 (using a CNN with architecture similar to UNet), and let $\{(\bar{\phi}_n, \bar{M}_n) = g(a_n)\}_{n=1}^N$ denote the features and spatial masks decoded per-entity using an up-convolutional decoder network g . We first warp, using the predicted locations b_n , these features and masks into image coordinates at same resolution as ϕ_{bg} . Denoting by \mathcal{W} a differentiable warping function e.g. in Spatial Transformer Networks [36], we can obtain the entity features and masks in the image space:

$$\phi_n = \mathcal{W}(\bar{\phi}_n, b_n); \quad M_n = \mathcal{W}(\bar{M}_n, b_n) \quad (3.1)$$

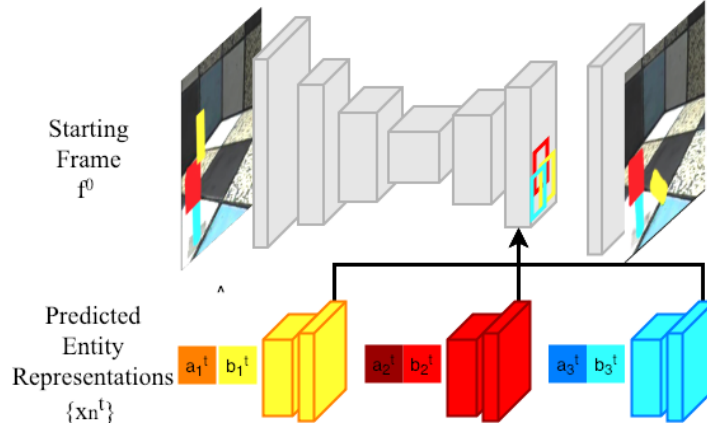


Figure 3.2: Our frame decoder takes in the initial frame f^0 and the predicted entity representations at time t , and outputs the frame corresponding to the predicted future f^t .

Note that the warped mask and features (ϕ_n, M_n) for each entity are zero outside the predicted bounding box b_n , and the mask M_n can further have variable values within this region. Using these independent background and entity features, we compose frame level spatial features ϕ by combining these via a weighted average. Denoting by M_{bg} a constant spatial mask (with value 0.1), we obtain the composed features as:

$$\phi = \frac{\phi_{bg} \odot M_{bg} \oplus \sum_n \phi_n \odot M_n}{M_{bg} \oplus \sum_n M_n} \quad (3.2)$$

These composed features ϕ incorporate information from all entities at the appropriate spatial locations, allow for occlusions using the predicted masks, and incorporate the information from background. We then decode the pixels for the future frame from these composed features. Note that one has a choice over the spatial level where this feature composition happens e.g. it can happen in feature space at near the image resolution (late fusion), or even directly at pixel level (where the variables ϕ all represent pixels), or alternatively at a lower resolution (mid/early fusion). We find that late fusion in implicit (and not pixel) space yields most promising results, and also find that the inferred masks end up correspond to instance segmentations.

3.3.3 Latent Representation

We described in Sec 3.3.1 how our prediction module is conditioned on a latent variable u , which in turn generates per-timestep conditioning variables z_t that are used in each prediction step – this is depicted in Fig 3.3(a). Intuitively, the global latent variable would capture video-level ambiguities e.g. where the blocks fall, the variables z_t resolve the corresponding ambiguities in the per-timestep motions. While previous approaches for future prediction similarly use latent variables to resolve ambiguities (see Fig 3.3(c-d)), the typical idea is to use independent per-timestep random variables, whereas in our model the z_t 's are all correlated.

During training, instead of marginalizing the likelihood of the sequences over all possible values of the latent variable u , we instead minimize the variational lower bound of the log-likelihood objective. This is done via training another module, a *latent encoder*, which (only during training) predicts a distribution over u conditioned on the ground-truth video. In practice, we find that simply conditioning

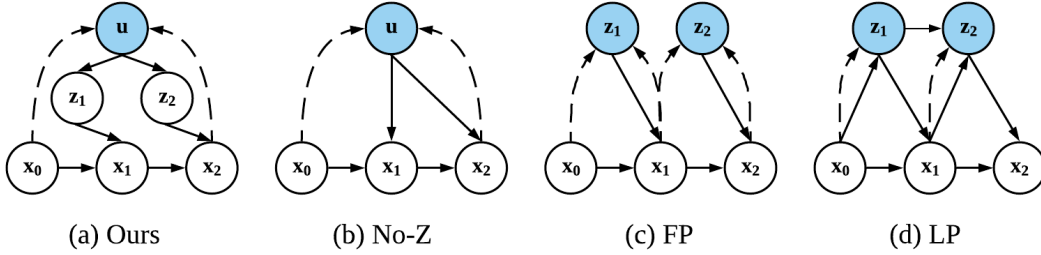


Figure 3.3: Our encoder (a) and baseline encoder (b-d). At test time, variables in blue are sampled randomly. At training, encoders model the posterior by all x s connected with dotted lines.

on the first and last frame of the video (using a feed-forward neural network) is sufficient, and denote by $q(u|f^0, \hat{f}^T)$ the distribution predicted. Given a particular u sampled from this distribution, we recover the $\{z_t\}$ via a one-layer LSTM which, using u as the cell state, predicts the per-timestep variables for the sequence.

3.3.4 Training Objective

Overall, our training objective can be thought of as maximizing the log-likelihood of the ground-truth frame sequence $\{\hat{f}^t\}_{t=1}^T$. We additionally also use training-time supervision for the locations of the entities $\{\{\hat{b}_n^t\}_{n=1}^N\}_{t=1}^T$. While this objective has an interpretation of log-likelihood maximization, for simplicity we describe it as a loss L composed of different terms, where the first L_{pred} encourages the future frame and location predictions to match the ground-truth:

$$L_{pred} = \sum_{t=1}^T (\|\mathcal{D}(\{x_n^t\}, f^0) - \hat{f}^t\|_1 + \lambda_1 \sum_{n=1}^N \|b_n^t - \hat{b}_n^t\|^2)$$

The second component corresponds to enforcing an information bottleneck on the latent variable distribution:

$$L_{enc} = KL[q(u) \parallel \mathcal{N}(0, I)]$$

Lastly, to further ensure that the decoder generates realistic composite frames, we add an auto-encoding loss that enforces it generates the correct frame when given entities representations $\{\hat{x}_n^t\}$ extracted from \hat{f}^t (and not the ones predicted) as input.

$$L_{dec} = \sum_{t=0}^T \|\mathcal{D}(\{\hat{x}_n^t\}, f^0) - \hat{f}^t\|_1$$

The total loss is therefore $L = L_{dec} + L_{pred} + \lambda_2 L_{enc}$ with hyper-parameter λ_2 determining the trade-offs among accurate predictions and information bottleneck in random variable. See appendix for additional details. We will release our code for reproducibility.

3.4 Experiment

We aim to show qualitative and quantitative results highlighting the benefits of various components (predictor, decoder, and latent representation) used in our approach, and aim to highlight that our

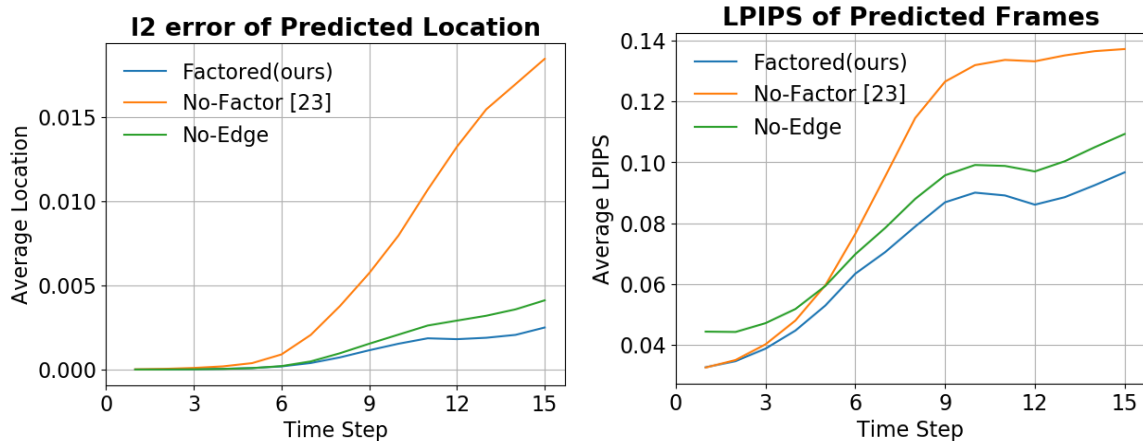


Figure 3.4: Quantitative evaluation of different variants of the entity predictor. Left: Average location error for predicted entities over time; Right: Average perceptual error of predicted frames.

approach is general to accommodate various scenarios. See generated videos in supplementary.

3.4.1 Experiment Setup

Dataset. We demonstrate our results on both the synthetic (ShapeStacks [29]) and real (Penn Action [99]) dataset. Shapestacks is a synthetic dataset comprised of stacked objects that fall under gravity with diverse blocks and configurations. The blocks can be cubes, cylinders, or balls with different colors. In addition to evaluating generalization ability, we further test with similar setups with videos comprised of 4, 5 or 6 blocks.

Penn Action [99] is a real video dataset of people playing various indoor and outdoor sports with annotations of human joint locations. The Penn Action dataset is challenging because of a) diverse backgrounds, view angles, human poses and scales b) noise in annotations, and c) multiple activity classes with different dynamics. We use a subset of the categories related to gym activities because most videos in these classes do not have camera motion and their backgrounds are similar within these categories. We adopt the recommended train/test split in [99]. Beyond that, we argue it is not impractical to assume known locations – we substitute ground truth annotation \hat{b}_n^t with key-points location from off-the-shelf detector [19] in both training and testing.

In both scenarios, we train our model to generate video sequence of 1 second given an initial frame, using exactly the same architecture despite the two diverse scenarios – entities correspond to objects in Shapestacks and correspond to joints of human body in Penn Action.

Evaluation Metrics. In both of these settings, we evaluate the predicted entity locations using average mean square error and the quality of generated frames using the Learned Perceptual Image Patch Similarity (LPIPS) [98] metric. A subtle detail in the evaluation is that at inference, the prediction is dependent on a random variable u , and while only a single ground-truth is observed, multiple predictions are possible. To account for this, we draw 100 samples of latents and record the best scores as in [16]. When we ablate non-stochastic modules (e.g. decoders), we use the mean u predicted by the latent encoder (after seeing the ‘ground-truth’ video). Without further specification, the curves are plotted in the ‘best of 100’ setup; the qualitative results visualize the best predictions in terms of LPIPS.

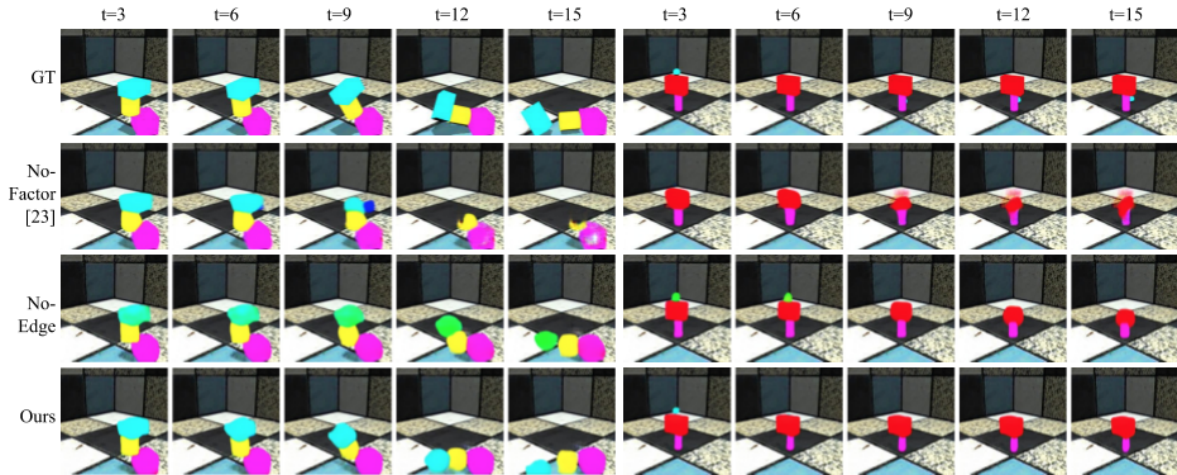


Figure 3.5: Video prediction results using our prediction module compared to baselines. We visualize the generated sequence after every 3 time steps.

Baselines. There are three key components in our model, i.e. the entity predictor, frame decoder, and latent representation. Various baselines are provided to highlight our choices in each of the components. Among them, some variant specifically points to previous approaches as the following:

- No-Factor [49] only predicts on the level of frames. Here we provide supervision from entity locations and pixels instead of segmentation masks;
- LP [16] implements the stochastic encoder module in SVG-LP to compare different dependency of latent variables;
- Pose Knows [83] is most related to our Penn Action setting which also predicts poses as intermediate representation, but it predicts location jointly and generates videos in a different way.

Besides the above which are strongly connected to previous works, we also present other baselines whose details are discussed in Section 3.4.2.

3.4.2 Analysis using Shapestacks

We use Shapestacks to validate the different components of the proposed approach i.e. the entity predictor, frame decoder, and the modeling choices for the latent variables.

Entity Predictor. We aim to show that our proposed predictor, which is capable of factorizing prediction over per-entity locations and appearance, as well as allowing reasoning via GNNs, improves prediction. Towards this, we compare against two alternate models: a) No-Factor [49] and b) No Edge. The No-Factor model does not predict a per-entity appearance but simply outputs a global feature that is decoded to foreground appearance and mask. To ensure the use of the same supervision as box locations, the No-Factor model also takes as input (and outputs) the per-entity bounding boxes, but these are not used via the decoder. The No-Edge model does not allow for interactions among entities when predicting the future. See appendix for details.

Figure 3.5 shows the prediction using our model and the baselines. The No-Factor baseline generates plausible frames at the beginning and performs well for static entities. However, at later time steps, entities with large range of motion diffuse because of the uncertainty. In contrast, entities generated by our method have clearer boundary over time. The No-Edge baseline does not accurately predict

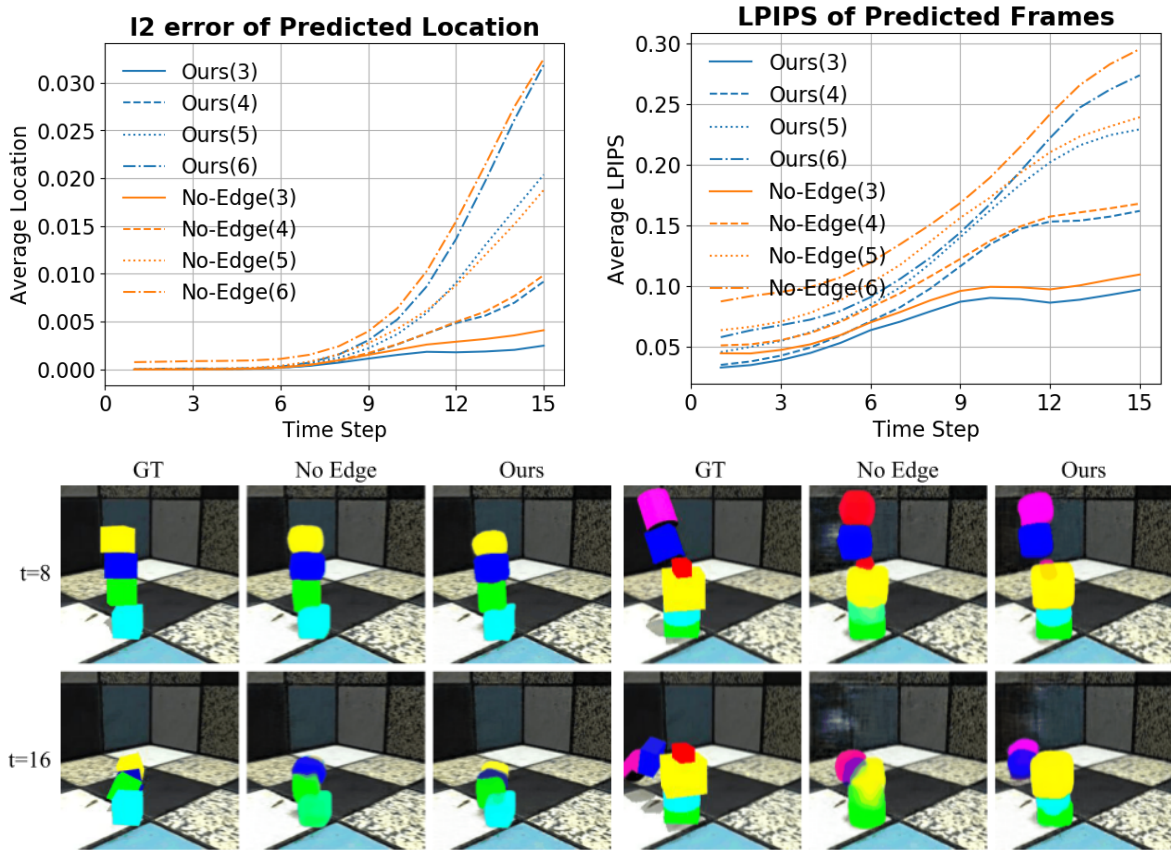


Figure 3.6: Above: Quantitative evaluation of the entity predictor when generalized to different number of blocks. The number in the bracket indicates the number of blocks in the subset. Below: Video prediction results using our prediction modules compare to baselines. We visualize the middle and last step.

block orientations as it requires more information about relative configuration, and further changes the colors over time. In contrast, blocks generated by our approach gradually rotate and fall over and colors are learned to remain the same. In Figure 3.4, we report quantitative evaluations, and similarly observe the benefits of our approach.

Figure 3.6 shows the results when the model generalizes to different number of entities (4, 5, and 6) at test time. The No-Factor uses fully connected layers to predict which cannot be directly adapted to variable number of blocks. We show methods that are able to accommodate the number of entities changes, i.e. No-Edge and ours. Our method predicts locations closer to the truth with more realistic appearance, and is able to retain the blocks color across time. Note that we train all models with only three blocks.

Primitive Decoder. While the No-Factor baseline above shows the benefits of composing different features for each entity while accounting for their predicted spatial location, we ablate here whether this composition should directly be at a pixel-level or at some implicit feature level (early, mid, or late). Across all these ablations, the number of layers in the decoder remain the same; only the level at which features from entities are composed differs.

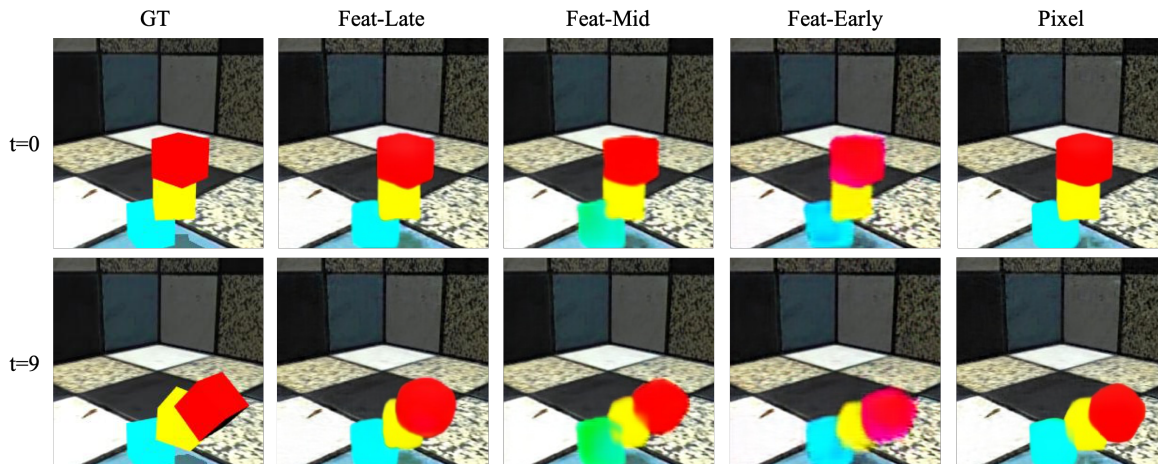


Figure 3.7: Qualitative results for composing entity representations into a frame. We visualize the outputs from variants of the decoder performing Late/Mid/Early fusion in feature space, or directly in pixel space. The first row depicts decoding of the initial representation; the second row depicts decoding of the predicted entities at a later time step.

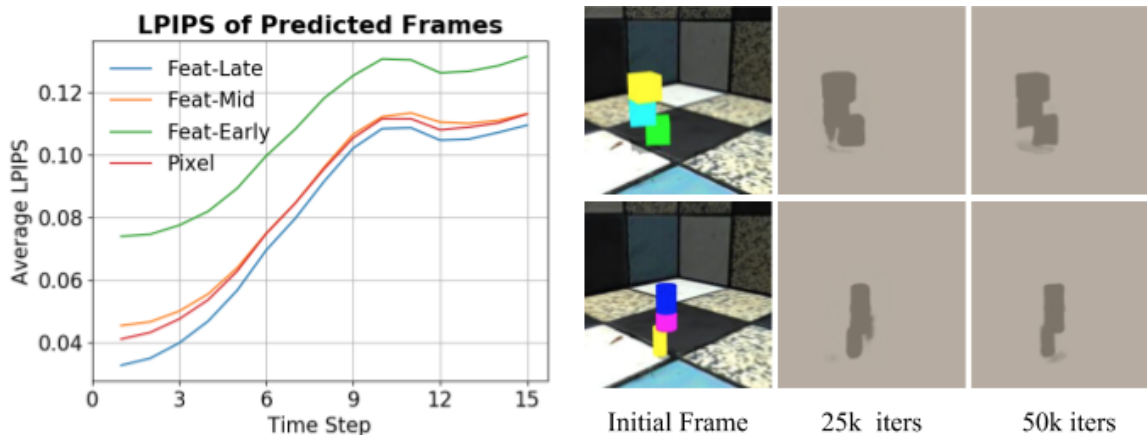


Figure 3.8: Left: Average Perceptual error for predicted frames via variants of the decoder. Right: Visualization of the composition of the foreground masks predicted for the entities. We observe the our model gradually learns to separate foreground and background without direct supervision.

The qualitative result are shown in Fig 3.7 where the first row visualizes decodings from the initial frame, and the second row demonstrates decoding from predicted features for a later timestep. While both late/pixel-level fusion reconstructs the initial frame faithfully, the pixel-level fusion introduces artifacts for future frames. The mid/early fusion alternates do not capture details well. We also observe similar trends in the quantitative results visualized in Figure 3.8. Note the latent u is encoded by the ground truth videos.

To further analyze the predictions from the decoder, we visualize the generated soft masks in

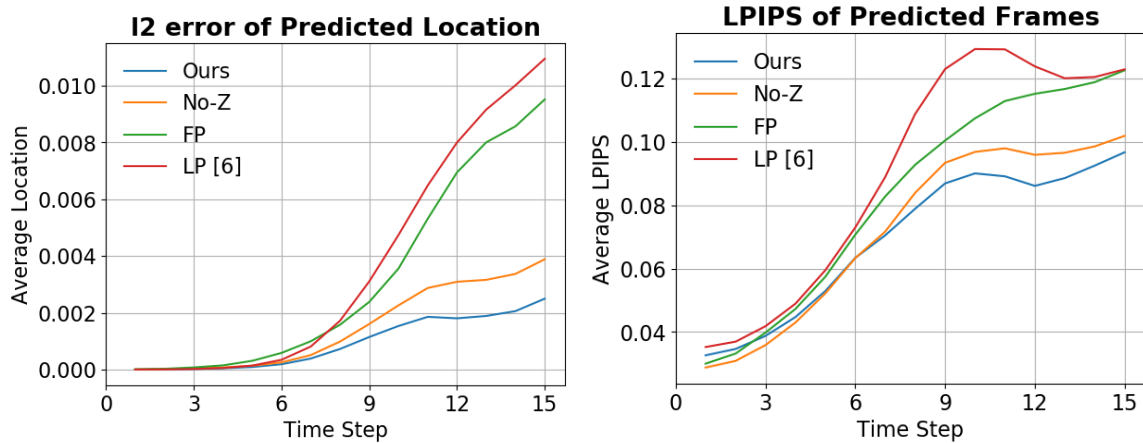


Figure 3.9: Error for location prediction (Left) and frame prediction (Right) using our encoder and baselines. For each sequence, we compute the error using the best of 100 random samples.

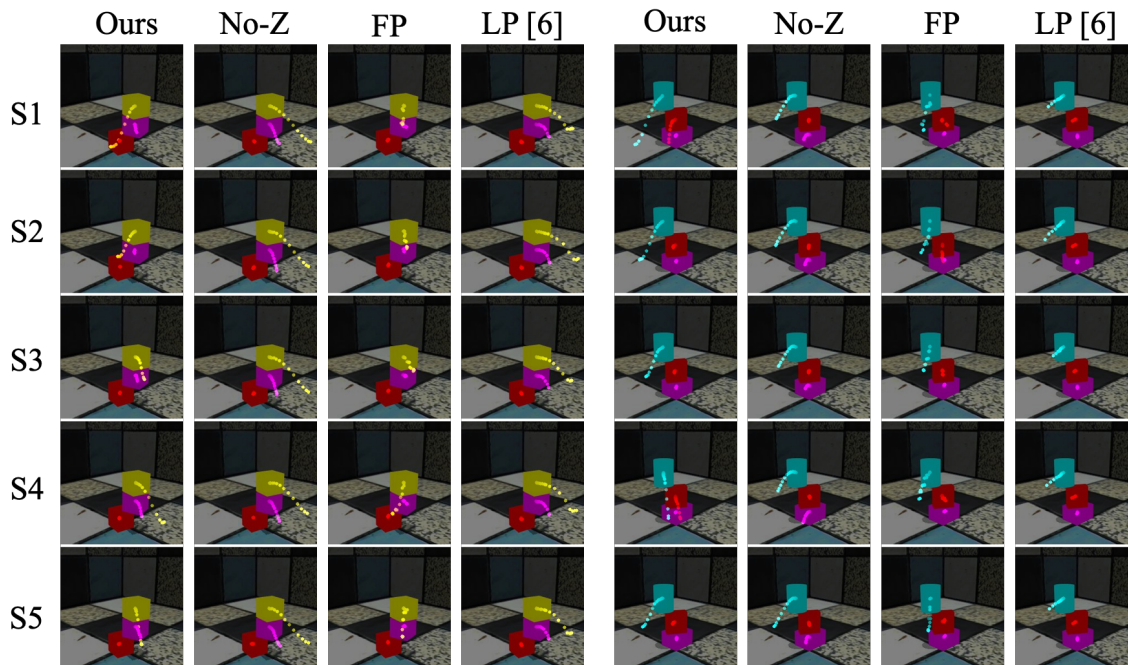


Figure 3.10: Visualization of five randomly sampled future predictions by our method and other baselines. We show the predicted centers of entities over time overlaid on top of the initial frame.

Figure 3.8. The values indicate the probability of the pixel belongs to a foreground of the entity. Note that this segmentation emerges despite not providing any direct supervision, but only using location and frame-level supervision.

Latent Representation. Our choice of the latent variables in the prediction model differs from the common choice of using a per-timestep random variable z_t . We compare our approach (Figure 3.3a) with such other alternatives (Figure 3.3 b-e). The No-Z baseline (Figure 3.3b) directly uses u across every time step, instead of predicting a per-timestep z_t from it. In both Fixed Prior (FP) and Learned Prior(LP) [16] baselines, the random variables are sampled per time step, either independently (FP), or depending on previous prediction (LP). During training, both FP and LP models are trained using an encoder similar to ours, but this encoder that predicts z_t using the frames f^t and f^{t+1} (instead of our approach using f^0 and f^T to predict u).

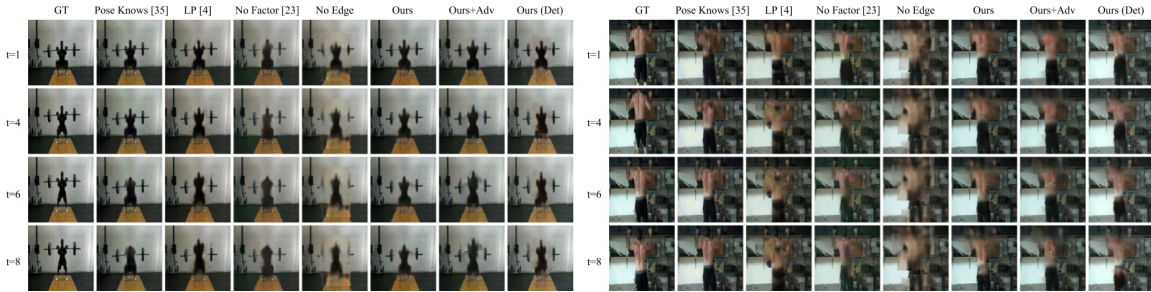


Figure 3.11: Video prediction results with best LPIPS latent using our approach compared to baselines. The last column visualizes results when the entity (joints) locations are replaced by the detection in both training and testing. Videos are in supplementary.

We visualize using five *random* samples in form of trajectories of entity locations in Figure 3.10. We notice that the direction of trajectories from No-Z model do not change across samples. The FP model has issues maintaining consistent motions across time-steps as during each timestep, an independent latent variable is sampled. The LP method performs well compared to FP, but still has similar issues. Compared to baselines, the use of a global latent variable allows us to sample and produce consistent motions across a video sequence, while also allowing diverse predictions across samples. The quantitative evaluations in Figure 3.9 show similar benefits where our method does well for both location error and frame perceptual distance over time.

3.4.3 Penn Action

Our model used in this dataset is exactly the same as that in the Shapestacks scenario, with the modification that the graph used for the interactions in the predictor is based on the human skeleton, and not fully-connected. Note that while the graph depends on the skeleton, the interaction blocks are the same across each edge. See supplementary for generated videos.

We also compare with Pose-Knows [83] which leverages entities as intermediate representation and generates pixel-level prediction. However, they a) do not predict feature for appearance but only location of each entity (joint); b) do not involve interaction mechanism; c) adopt a different generation method (GAN) where they stick sequence of rendered pose figures to the initial frame, and fuse them by a spatial-temporal 3D convolution network [77]. In their paper, the adversarial loss is posed to improve realism. We present that our method also benefits from the adversarial loss (Ours+Adv).

Figure 3.11 and Figure 3.13 show qualitative and quantitative results using the best latent variable u among 100 samples. The No-Factor baseline cannot directly generate plausible foreground while the No-Edge baseline does not compose well. Our results improve to be sharper if adversarial loss is added (Ours+Adv). The location error also decreases because better location predictions contribute to plausible generated videos.

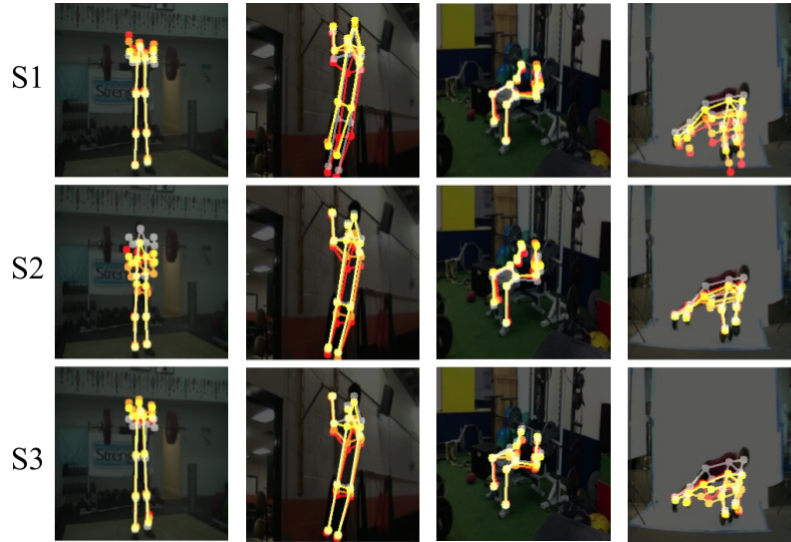


Figure 3.12: Visualization of joint positions in three randomly sampled future predictions by our method. The initial skeletons are plotted as white. Skeletons at time 0.25s, 0.5s, and 1s are plotted as yellow, orange, and red, respectively.

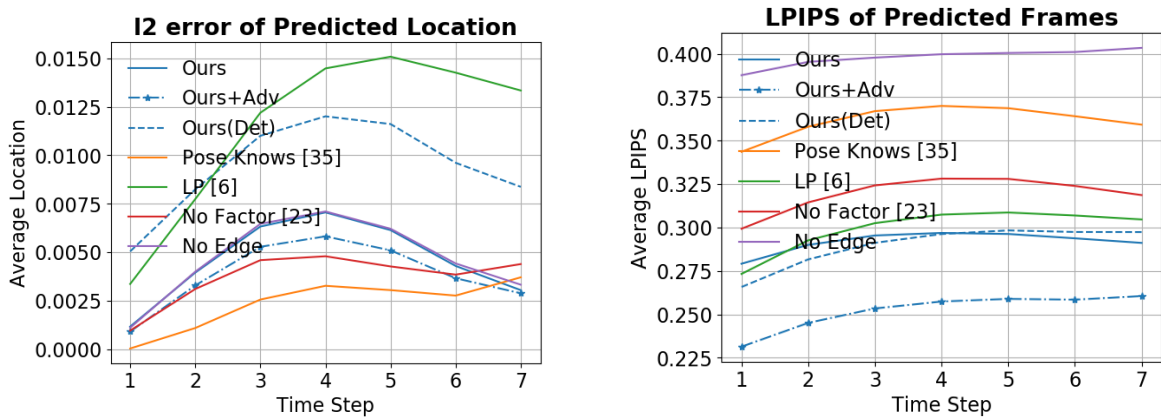


Figure 3.13: Error for location prediction (Left) and frame prediction (Right) using our model and baseline methods. For each sequence, we compute the error using the best of 100 random samples.

We also visualize predictions when, during both *training* and *inference*, annotated key-points are replaced with detected key-points using [19]. We note that the performance is competitive to the setting using annotated key-point locations, indicating that our method is robust to annotation noise. It also indicates that the requirement of entity locations is not a bottle-neck, since automatically inferred location suffice in our experiment.

Figure 3.12 visualizes different sample futures using the predicted joint locations across time. Our model learns the boundary of the human body against the background as well as how the entities compose the human body even when they heavily overlap. More interestingly, the model learns different types of dynamics for different sports. For example, in pull ups, the legs move more while

the hands are still; in clean and jerk, the legs almost remain at the same place.

3.5 Discussion

In this work we proposed a method that leverages compositionality across entities for video prediction. However, the task of video prediction in a general setting is far from being solved, and many challenges still remain. In particular, we rely on supervision of the entity locations, either from human or automatic annotations. It would be interesting to relax this requirement and allow the entities to emerge as pursued in some recent works [8,33], although in simpler settings. Additionally, GAN-based auxiliary losses have been shown to improve image synthesis quality, and these could be explored in conjunction with our model. Lastly, developing metrics to evaluate the diversity and accuracy of video predictions is also challenging due to the multi-modal nature of the task, and we hope some future efforts will also focus on this aspect.

Chapter 4

Conclusion

We have presented that leveraging categorical structures improves zero-shot recognition in robustness and generalization; leveraging entity structures benefits long-term predictions particularly in scenes comprised of multiple entities. The structures can either be explicitly pre-defined such as knowledge graph and human skeleton, or implicitly learned from data. We use the family of graph neural networks, the scalable machine learning techniques, to learn from the structured data. In both tasks, we have achieved state-of-the-art results.

Looking forward, there are several limitations to improve in our work. It will be interesting to let detection of entities and structures emerge from the data with minimal supervision or from self supervision. There is also limited study on how those techniques would be affected by noises in the structure, e.g. inaccurate knowledge graph or massive false positive entity detection. Additionally, it is also interesting to pursue dynamic structures which changes over time. It might help us understand temporal axis in videos.

In long-term, leveraging structures may improve generalization ability, prediction capacity, and even interpretability. A probably more inherent reason is that, everything in the structured world we live in does not exist independently. Therefore, we want to pursue a visual system not only capable of pointing out what the entities are, but also capable of leveraging and reasoning about the structures underneath them.

Bibliography

- [1] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid. Label embedding for attribute-based classification. In *CVPR*, 2013.
- [2] F. G. Ashby and L. A. Alfonso-Reese. Categorization as probability density estimation. *Journal of mathematical psychology*, 39(2):216–233, 1995.
- [3] J. L. Ba, K. Swersky, S. Fidler, and R. Salakhutdinov. Predicting Deep Zero-Shot Convolutional Neural Networks using Textual Descriptions. In *ICCV*, 2015.
- [4] M. Babaeizadeh, C. Finn, D. Erhan, R. H. Campbell, and S. Levine. Stochastic variational video prediction. *ICLR*, 2017.
- [5] P. Battaglia, R. Pascanu, M. Lai, D. J. Rezende, et al. Interaction networks for learning about objects, relations and physics. In *NeurIPS*, 2016.
- [6] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.
- [7] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. H. Jr., and T. M. Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, 2010.
- [8] M. B. Chang, T. Ullman, A. Torralba, and J. B. Tenenbaum. A compositional object-based approach to learning physical dynamics. *ICLR*, 2016.
- [9] S. Changpinyo, W.-L. Chao, B. Gong, and F. Sha. Synthesized Classifiers for Zero-Shot Learning. In *CVPR*, 2016.
- [10] S. Changpinyo, W.-L. Chao, and F. Sha. Predicting Visual Exemplars of Unseen Classes for Zero-Shot Learning. In *ICCV*, 2017.
- [11] W.-L. Chao, S. Changpinyo, B. Gong, and F. Sha. An Empirical Study and Analysis of Generalized Zero-Shot Learning for Object Recognition in the Wild. In *ECCV*, 2016.
- [12] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *BMVC*, 2014.
- [13] B. Chen, W. Wang, and J. Wang. Video imagination from a single image with transformation generation. In *ACMMM Workshop*, 2017.
- [14] X. Chen, A. Shrivastava, and A. Gupta. Neil: Extracting visual knowledge from web data. *ICCV*, 2013.
- [15] J. Deng, N. Ding, Y. Jia, A. Frome, K. Murphy, S. Bengio, Y. Li, H. Neven, and H. Adam. Large-Scale Object Classification Using Label Relation Graphs. In *ECCV*, 2014.
- [16] E. Denton and R. Fergus. Stochastic video generation with a learned prior. In *ICML*, 2018.
- [17] E. L. Denton et al. Unsupervised learning of disentangled representations from video. In *NeurIPS*, 2017.
- [18] M. Elhoseiny, B. Saleh, and A. Elgammal. Write a Classifier: Zero-Shot Learning Using Purely Textual Descriptions. In *ICCV*, 2013.

- [19] H.-S. Fang, S. Xie, Y.-W. Tai, and C. Lu. RMPE: Regional multi-person pose estimation. In *ICCV*, 2017.
- [20] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth. Describing objects by their attributes. In *CVPR*, 2009.
- [21] R. Fergus, H. Bernal, Y. Weiss, and A. Torralba. Semantic Label Sharing for Learning with Many Categories. In *ECCV*, 2010.
- [22] C. Finn, I. Goodfellow, and S. Levine. Unsupervised learning for physical interaction through video prediction. In *NeurIPS*, 2016.
- [23] K. Fragkiadaki, P. Agrawal, S. Levine, and J. Malik. Learning visual predictive models of physics for playing billiards. *ICLR*, 2015.
- [24] K. Fragkiadaki, J. Huang, A. Alemi, S. Vijayanarasimhan, S. Ricco, and R. Sukthankar. Motion prediction under multimodality with conditional stochastic networks. *arXiv preprint arXiv:1705.02082*, 2017.
- [25] A. Frome, G. Corrado, J. Shlens, S. Bengio, J. Dean, and T. Mikolov. Devise: A deep visual-semantic embedding model. In *NIPS*, 2013.
- [26] Y. Fu and L. Sigal. Semi-supervised Vocabulary-informed Learning. In *CVPR*, 2016.
- [27] Z. Fu, T. Xiang, E. Kodirov, and S. Gong. Zero-Shot Object Recognition by Semantic Manifold Distance. In *CVPR*, 2015.
- [28] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NeurIPS*, 2014.
- [29] O. Groth, F. Fuchs, I. Posner, and A. Vedaldi. Shapestacks: Learning vision-based physical intuition for generalised object stacking. *ECCV*, 2018.
- [30] B. Hariharan and R. Girshick. Low-shot Visual Recognition by Shrinking and Hallucinating Features. In *CoRR*, 2017.
- [31] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.
- [32] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.
- [33] J.-T. Hsieh, B. Liu, D.-A. Huang, L. Fei-Fei, and J. C. Niebles. Learning to decompose and disentangle representations for video prediction. *NeurIPS*, 2018.
- [34] C. Huang, C. C. Loy, and X. Tang. Local similarity-aware deep feature embedding. In *NIPS*, 2016.
- [35] D.-A. Huang, A.-m. Farahmand, K. M. Kitani, and J. A. Bagnell. Approximate maxent inverse optimal control and its application for mental simulation of human interactions. In *AAAI*, 2015.
- [36] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In *NeurIPS*, 2015.
- [37] D. Jayaraman and K. Grauman. Zero-shot recognition with unreliable attributes. In *NIPS*, pages 3464–3472, 2014.
- [38] X. Jia, B. De Brabandere, T. Tuytelaars, and L. V. Gool. Dynamic filter networks. In *NeurIPS*, 2016.
- [39] Z. Jia, A. C. Gallagher, A. Saxena, and T. Chen. 3d reasoning from blocks to stability. *IEEE transactions on pattern analysis and machine intelligence*, 2015.
- [40] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, and T. Mikolov. Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*, 2016.
- [41] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.
- [42] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *ICLR*, 2014.
- [43] T. Kipf, E. Fetaya, K.-C. Wang, M. Welling, and R. Zemel. Neural relational inference for interacting systems. *ICML*, 2019.
- [44] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *ICLR*, 2017.
- [45] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert. Activity forecasting. In *ECCV*, 2012.
- [46] E. Kodirov, T. Xiang, and S. Gong. Semantic Autoencoder for Zero-Shot Learning. In *CVPR*, 2017.

- [47] C. H. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *CVPR*, 2009.
- [48] C. H. Lampert, H. Nickisch, and S. Harmeling. Attribute-Based Classification for Zero-Shot Visual Object Categorization. In *TPAMI*, 2014.
- [49] A. Lerer, S. Gross, and R. Fergus. Learning physical intuition of block towers by example. *ICML*, 2016.
- [50] W. Li, S. Azimi, A. Leonardis, and M. Fritz. To fall or not to fall: A visual approach to physical stability prediction. *arXiv preprint arXiv:1604.00066*, 2016.
- [51] W. Li, A. Leonardis, and M. Fritz. Visual stability prediction and its application to manipulation. *AAAI*, 2016.
- [52] Z. Liu, R. A. Yeh, X. Tang, Y. Liu, and A. Agarwala. Video frame synthesis using deep voxel flow. In *ICCV*, 2017.
- [53] Y. Lu. Unsupervised learning on neural network outputs: with application in zero-shot learning. In *IJCAI*, 2016.
- [54] C. G. Lucas, S. Bridgers, T. L. Griffiths, and A. Gopnik. When children are better (or at least more open-minded) learners than adults: Developmental differences in learning the forms of causal relationships. *Cognition*, 131(2):284–299, 2014.
- [55] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *ICML*, 2013.
- [56] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.
- [57] K. Marino, R. Salakhutdinov, and A. Gupta. The More You Know: Using Knowledge Graphs for Image Classification. In *CVPR*, 2017.
- [58] M. Mathieu, C. Couprie, and Y. LeCun. Deep multi-scale video prediction beyond mean square error. *ICLR*, 2016.
- [59] T. Mensink, J. Verbeek, F. Perronnin, and G. Csurka. Metric Learning for Large Scale Image Classification: Generalizing to New Classes at Near-Zero Cost. In *ECCV*, 2012.
- [60] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *ICLR*, 2013.
- [61] G. A. Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.
- [62] I. Misra, A. Gupta, and M. Hebert. From Red Wine to Red Tomato: Composition with Context. In *CVPR*, 2017.
- [63] M. Norouzi, T. Mikolov, S. Bengio, Y. Singer, J. Shlens, A. Frome, G. S. Corrado, and J. Dean. Zero-shot learning by convex combination of semantic embeddings. In *ICLR*, 2014.
- [64] M. Palatucci, D. Pomerleau, G. E. Hinton, and T. M. Mitchell. Zero-shot Learning with Semantic Output Codes. In *NIPS*, 2009.
- [65] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543, 2014.
- [66] R. Qiao, L. Liu, C. Shen, and A. van den Hengel. Less is more: zero-shot learning from online textual documents with noise suppression. In *CVPR*, 2016.
- [67] M. Rohrbach, S. Ebert, and B. Schiele. Transfer learning in a transductive setting. In *NIPS*, 2013.
- [68] M. Rohrbach, M. Stark, G. Szarvas, I. Gurevych, and B. Schiele. What helps where - and why? semantic relatedness for knowledge transfer. In *CVPR*, 2010.
- [69] M. Rohrbach, M. Stark, and B. Schiele. Evaluating Knowledge Transfer and Zero-Shot Learning in a Large-Scale Setting. In *CVPR*, 2011.

- [70] B. Romera-Paredes and P. H. S. Torr. An embarrassingly simple approach to zero-shot learning. In *ICML*, 2015.
- [71] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015.
- [72] R. Salakhutdinov, A. Torralba, and J. Tenenbaum. Learning to Share Visual Appearance for Multiclass Object Detection. In *CVPR*, 2011.
- [73] A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap. A simple neural network module for relational reasoning. In *NeurIPS*, 2017.
- [74] R. Socher, M. Ganjoo, C. D. Manning, and A. Y. Ng. Zero-Shot Learning Through Cross-Modal Transfer. In *ICLR*, 2013.
- [75] C. Sun, A. Shrivastava, S. Singh, and A. Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *ICCV*, 2017.
- [76] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going Deeper with Convolutions. In *CVPR*, 2015.
- [77] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015.
- [78] S. Tulyakov, M.-Y. Liu, X. Yang, and J. Kautz. Mocogan: Decomposing motion and content for video generation. *CVPR*, 2017.
- [79] R. Villegas, J. Yang, S. Hong, X. Lin, and H. Lee. Decomposing motion and content for natural video sequence prediction. *ICLR*, 2017.
- [80] R. Villegas, J. Yang, Y. Zou, S. Sohn, X. Lin, and H. Lee. Learning to generate long-term future via hierarchical prediction. *ICML*, 2017.
- [81] C. Vondrick, H. Pirsiavash, and A. Torralba. Generating videos with scene dynamics. In *NeurIPS*, 2016.
- [82] J. Walker, C. Doersch, A. Gupta, and M. Hebert. An uncertain future: Forecasting from static images using variational autoencoders. In *ECCV*, 2016.
- [83] J. Walker, K. Marino, A. Gupta, and M. Hebert. The pose knows: Video forecasting by generating pose futures. In *ICCV*, 2017.
- [84] P. Wang, Q. Wu, C. Shen, A. van den Hengel, and A. Dick. FVQA: Fact-based Visual Question Answering. In *CoRR*, 2016.
- [85] X. Wang, Y. Ye, and A. Gupta. Zero-shot recognition via semantic embeddings and knowledge graphs. *CVPR*, 2018.
- [86] N. Watters, A. Tacchetti, T. Weber, R. Pascanu, P. Battaglia, and D. Zoran. Visual interaction networks. *arXiv preprint arXiv:1706.01433*, 2017.
- [87] J. Weston, S. Bengio, and N. Usunier. Large Scale Image Annotation: Learning to Rank with Joint Word-Image Embeddings. In *ECML*, 2010.
- [88] J. Wu, J. J. Lim, H. Zhang, J. B. Tenenbaum, and W. T. Freeman. Physics 101: Learning physical object properties from unlabeled videos. In *BMVC*, 2016.
- [89] Q. Wu, P. Wang, C. Shen, A. Dick, and A. van den Hengel. Ask me anything: Free-form visual question answering based on knowledge from external sources. In *CVPR*, 2016.
- [90] Y. Xian, B. Schiele, and Z. Akata. Zero-Shot Learning - The Good, the Bad and the Ugly. In *CVPR*, 2017.
- [91] B. Xu, N. Wang, T. Chen, and M. Li. Empirical evaluation of rectified activations in convolutional network. *CoRR*, abs/1505.00853, 2015.
- [92] T. Xue, J. Wu, K. Bouman, and B. Freeman. Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. In *NeurIPS*, 2016.

- [93] X. Yan, A. Rastogi, R. Villegas, K. Sunkavalli, E. Shechtman, S. Hadap, E. Yumer, and H. Lee. MT-VAE: learning motion transformations to generate multimodal human dynamics. In *ECCV*, 2018.
- [94] Z. Yang, W. Cohen, and R. Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *ICML*, 2016.
- [95] T. Ye, X. Wang, J. Davidson, and A. Gupta. Interpretable intuitive physics model. *ECCV*, 2018.
- [96] Y. Ye, M. Singh, A. Gupta, and S. Tulsiani. Compositional video prediction. 2019.
- [97] L. Zhang, T. Xiang, and S. Gong. Learning a deep embedding model for zero-shot learning. In *CVPR*, 2017.
- [98] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.
- [99] W. Zhang, M. Zhu, and K. G. Derpanis. From actemes to action: A strongly-supervised representation for detailed action understanding. In *ICCV*, 2013.
- [100] Z. Zhang and V. Saligrama. Zero-shot learning via semantic similarity embedding. In *ICCV*, 2015.
- [101] Z. Zhang and V. Saligrama. Zero-shot learning via joint latent similarity embedding. In *CVPR*, 2016.