Assessing and Improving Generalization in Graph Reasoning and Learning

by
Boris Knyazev

A Thesis
presented to
The University of Guelph

In partial fulfilment of requirements
for the degree of

Doctor of Philosophy
in
Engineering

Guelph, Ontario, Canada

ABSTRACT

ASSESSING AND IMPROVING GENERALIZATION IN GRAPH REASONING AND LEARNING

Boris Knyazev
University of Guelph, 2022

Advisor:
Graham W. Taylor

This thesis by articles makes several contributions to the field of machine learning, specifically, in graph reasoning tasks. Each article investigates and improves generalization in one of several graph reasoning applications: classical graph classification tasks, compositional visual reasoning, and the novel task of parameter prediction for neural network graphs.

In the first article we study the attention mechanism in graph neural networks (GNNs). While attention has been widely studied in GNNs, its effect on generalization to larger and noisier graphs has not been thoroughly analyzed. We show that in synthetic graph tasks, generalization can be improved by carefully initializing the attention modules of GNNs. We also develop a method that reduces sensitivity of attention modules to initialization and improves generalization in real graph tasks.

In the second article we address the problem of generalizing to rare or unseen compositions of objects and relationships in visual scenes. Previous works typically specialize on frequent visual compositions and show poor compositional generalization. To alleviate that, we found that it is important to normalize the loss function with respect to the structure of scene graphs so that the training labels are leveraged more effectively. Models trained with our loss significantly improve compositional generalization.

In the third article we further address visual compositional generalization. We consider a data

augmentation approach of adding rare and unseen compositions to the training data. We develop a model based on generative adversarial networks that generate synthetic visual features conditioned on rare or unseen scene graphs that we obtain by perturbing real scene graphs. Our approach consistently improves compositional generalization.

In the fourth article we study graph reasoning in the novel task of predicting parameters for unseen deep neural architectures. Our task is motivated by the limitations of iterative optimization algorithms used to train neural networks. To solve our task, we develop a model based on Graph HyperNetworks and train it on our dataset of neural architecture graphs. Our model can predict performant parameters for unseen deep networks, such as ResNet-50, in a single forward pass. Our model is useful for neural architecture search and transfer learning.

"I promise, I'm not going to work tonight, I'm
going to stay home with you, and
we're going to watch Dragon Tales"
—Adam Sandler (Click movie)

Dedicated to my family

ACKNOWLEDGEMENTS

"One, remember to look up at the stars and not down at your feet. Two, never give up work. Work gives you meaning and purpose and life is empty without it. Three, if you are lucky enough to find love, remember it is there and don't throw it away."

—Stephen Hawking

First of all, I would like to thank all the people I worked with during my PhD. I thank my PhD supervisor Graham Taylor for providing an excellent environment to grow as a better researcher and a better person. His patience, knowledge, open-mindedness and gentle guidance helped me immensely starting from the very first days of my PhD. I would like to thank Graham for connecting me with other wonderful people. In particular, I am deeply thankful to work with Mohamed Amer during my first internship. Mohamed treated me as a friend providing support and advice beyond the work related challenges. I really appreciate the flexibility and amount of time he gave me to explore different research directions. This helped me tremendously in gaining necessary knowledge to progress in my PhD.

In the middle of my PhD I got an opportunity to intern at MILA with amazing people, Eugene Belilovsky and Aaron Courville, as well as our collaborators, Harm de Vries and Cătălina Cangea. I learned a lot from them in terms of knowledge and collaboration skills. I am especially grateful to Eugene for his deep involvement in the projects and helping in publishing the results. This internship also allowed me to explore the unique AI ecosystem of MILA as well as life in the Quebec province and Montreal to make important career decisions after that.

After MILA, I was extremely lucky to intern at Facebook AI (currently Meta AI) with nice and smart people, Adriana Romero-Soriano and Michal Drozdzal. I thank them for giving me the right amount of flexibility in doing my research, for asking me difficult questions and for staying closely involved in the project beyond the internship. Adriana and Michal greatly improved my critical thinking, writing and collaboration skills. Their feedback and guidance was crucial to successfully finalize my last PhD project.

I would like to express gratitude to other people I collaborated and interacted with: Xiao Lin, Carolyn Augusta, Chris Kim, Yichao Lu, Himanshu Rai, Jason Chang, Shashank Shekhar, Maksims Volkovs, Hyunsoo Chung, Jungtaek Kim, Jinhwi Lee, Jaesik Park, Minsu Cho, Elahe Ghalebi and Rylee Thompson. Doing research and writing papers together with you was a very

smooth, pleasant and useful experience. I hope I was not a big burden in our projects. I am very grateful to my advisory committee members, Shawki Areibi and Medhat Moussa, for their prompt feedback and support. I am also extremely thankful to lab managers Brittany Reiche and Magdalena Sobol for their support and patience. I thank Magdalena for shaping my writing style and presentation skills.

I am expressing my gratitude to my labmates at Machine Learning Research Group whom I met in person in the beginning of my PhD making the first months in a new country less challenging for me. Unfortunately, due to the pandemic I could not meet in person many other kind and smart members of our lab.

I am thankful to the Government of Canada, Ontario and Quebec provinces, University of Guelph and Vector Institute for being very welcoming, transparent, responsive and for providing necessary computing resources, support and funding to work on my PhD.

Finally and most importantly, my PhD could not be completed without the support, care and love from my family, my parents, my wife and children. Doing a PhD is extremely stressful, frustrating, demanding and prone to high overtimes hurting both physical and mental health. I am very thankful to my wife for making my work-family balance good and stable. This encouraged me to work more efficiently and allowed me to finish my PhD without big sacrifices. I also deeply appreciate her patience of living far away from her home, parents and friends in a country that she struggles to fully enjoy to let me realize my potential. This means a lot to me!

TABLE OF CONTENTS

LIST OF TABLES

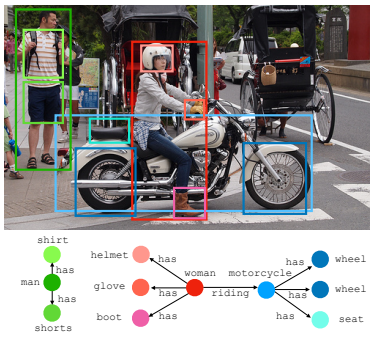
LIST OF FIGURES

1 Introduction

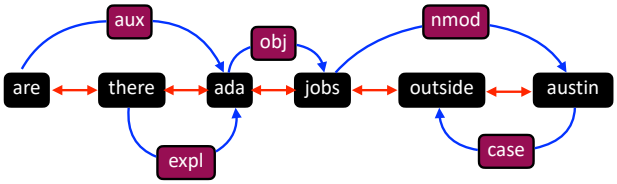
Our world is a complex compositional system, where simple components are used to create more complex ones and all components interact in a non-trivial way. One such component are humans, who have a innate ability to accumulate diverse multi-domain knowledge and learn a rich compositional structure of the surrounding world. This knowledge allows humans to easily solve a plethora of complex tasks. For example, given a static 2D image of a complex dynamic 3D scene, humans are able to recognize objects, their parts, relationships between them, and predict future events in the scene. Humans can even predict a scene's geographical and demographic context and infer abstract properties such as the sentiment of the scene (Fig. 1.1, a). Engineering aims to develop systems and algorithms capable of replacing humans in performing such tasks, especially tasks that are repetitive, laborious or dangerous. In some practical scenarios such as understanding 2D images, these systems are required to recover the original compositional structure from the input recorded by sensors (Fig. 1.1, a-c). For example, robots or self-driving cars need to detect objects and their relationships from raw pixels or point clouds. In other scenarios the compositional structure is already provided (*e.g.* by another system or human) and algorithms need to reason about the compositional input to make complex higher-level decisions (Fig. 1.1, d-f). Example tasks include: predicting properties of molecules, predicting future links between people or predicting properties of biological or artificial neural networks.

To develop algorithms able of inferring a compositional structure from raw sensory data or algorithms predicting structure's properties, we first need to define a data abstraction suitable for this kind of tasks. In mathematics and computer science there exists a convenient abstraction specifically introduced to model compositional and relational structures. This abstraction is called a graph, in which nodes correspond to the components of the structure and edges correspond to interactions between the components. For example, molecules are often represented as graphs with nodes corresponding to atoms or more complex elements and edges corresponding to chemical bonds (Fig. 1.1, d). Similarly, social networks are graphs with nodes being people and edges being different kinds of relationships between them (Fig. 1.1, e). Likewise, a biological or artificial neural network is a graph wherein nodes can be neurons and edges can be connections between them (Fig. 1.1, f). Having modeled data as graphs, algorithms tackling the associated tasks need to be developed.

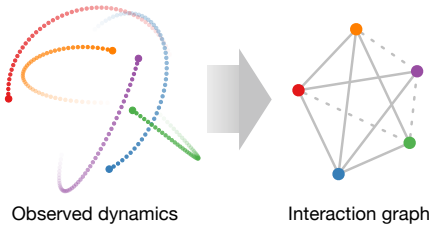
Tasks illustrated in Fig. 1.1 can be approached by manually designing task-specific rules or algorithms. However, recently there has been a spike in performance on such tasks using deep learning methods. In particular, deep neural networks (DNNs) require less human interventions and instead learn task-specific rules and algorithms from data and experience. DNNs currently dominate across visual, language and graph tasks (LeCun et al., 2015). Specifically, state-of-



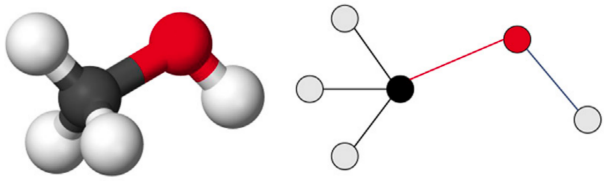
(a) Detecting objects and their relationships in a scene (Zellers et al., 2018)



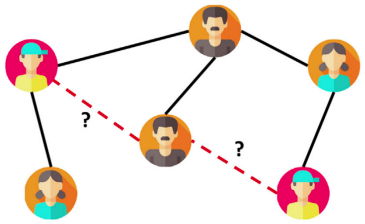
(b) Modeling language as graphs (Wu et al., 2021)



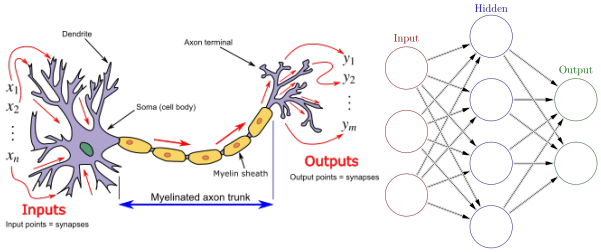
(c) Inferring interactions in physical systems (Kipf et al., 2018)



(d) Analyzing molecules (Zhou et al., 2020)



(e) Predicting links in social networks (Zhou et al., 2020)



(f) Predicting properties of neural networks[1]

Figure 1.1: Example applications of graph reasoning methods. [1]Images are obtained from https://en.wikipedia.org/wiki/Artificial_neural_network and https://en.wikipedia.org/wiki/Biological_neuron_model.

the-art convolutional neural networks (CNNs) (Krizhevsky et al., 2012; He et al., 2016) reach or even outperform (He et al., 2015; Cai et al., 2019a) humans in tasks such as large-scale image classification, *e.g.* ImageNet (Russakovsky et al., 2015). In the graph domain, many tasks have been successfully solved by DNNs operating on graphs, called Graph Neural Networks (GNNs) (Gori et al., 2005; Scarselli et al., 2008; Bronstein et al., 2017; Zhou et al., 2020). DNNs continue to improve and extend their scope, becoming an essential part of real world applications (Brown et al., 2020).

Despite the extreme success of DNNs, they have certain limitations and this thesis mainly addresses the two of them. The first challenge is fundamental and concerns the decrease of DNNs' performance when they observe somewhat unusual inputs (Hendrycks and Dietterich, 2019; Gal-

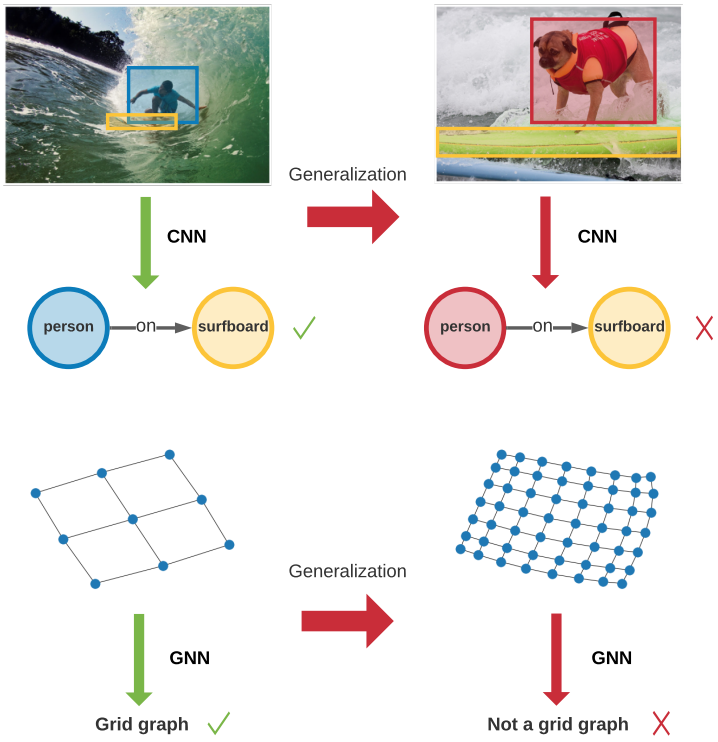


Figure 1.2: Machine learning methods for images, graphs and other domains can accurately reason about the *i.i.d.* samples (left). However, reasoning on test samples that are somewhat different from or correspond to the tail of the training distribution (right) remains challenging. In the graph example (bottom), both the small and large graph represent so called grid (mesh or lattice) graphs.

loway et al., 2019; Szegedy et al., 2013). For example in the image domain, the dog on a surfboard can be confused with a person, since in the data used to train CNNs persons appear more frequently on surfboards than dogs (Fig. 1.2, top). In the graph domain, a GNN trained to reason about smaller graphs, can fail to accurately reason about larger graphs (Yehudai et al., 2021) (Fig. 1.2, bottom). While there is still limited understanding what exactly makes CNNs or GNNs fail in such cases, a high-level explanation is that learning-based methods make an assumption that all training and test samples are independently and identically distributed (*i.i.d.*) (Shen et al., 2021). In practice (as illustrated in Fig. 1.2), this assumption is often violated, leading to inaccurate predictions or poor generalization. Such behavior is often unacceptable in critical applications such as self-driving cars or health care, where errors can be fatal. To highlight this issue and develop a more rigorous understanding of DNNs' generalization properties, a variety of research frameworks have been proposed (Lust and Condurache, 2020; Shen et al., 2021; Hendrycks and Dietterich, 2019; Hupkes et al., 2019; Garg et al., 2020). Despite these efforts, in many tasks the progress remains slow. For instance, in the scene graph generation task state-of-the-art CNNs show a ~10 fold decrease in performance on test images containing unusual or unseen visual compositions (Tang et al., 2020; Knyazev et al.,

2020; Suhail et al., 2021). In this thesis, we develop empirical frameworks to study generalization failure in CNNs and GNNs, leading to improvements in the visual and graph domains respectively.

The second limitation of DNNs pertains to the challenge of extending their scope to more problems where manually-designed algorithms still dominate. Before the rise of DNNs, researchers and practitioners focused tremendous efforts on designing features to better solve a concrete task, *e.g.* image classification (Lowe, 2004). The advent of deep learning showed that in many settings features learned by DNNs outperformed features designed by humans (Krizhevsky et al., 2012). This encouraged researchers and practitioners to extend the scope of DNNs in many directions. One notable direction has been the design of stronger DNN architectures[1] to better solve a given task. Similar to pre-deep learning feature design, architecture design has been recently overtaken by neural architecture search methods that *learn* a DNN architecture, replacing and outperforming human-designed DNN architectures (Zoph and Le, 2016; Liu et al., 2018b; Ren et al., 2020). Yet many aspects of DNN practice still require much human engineering effort.

In particular, optimization algorithms used to train DNNs are still manually-designed. For example, one of the dominant algorithms is stochastic gradient descent (SGD). SGD is a manually-designed algorithm based on computing gradients *w.r.t.* a highly *non-convex* loss function and the gradient-based rules to update DNN parameters according to *convex* optimization theory (Jain and Kar, 2017). Replacing optimization algorithms such as SGD with DNN counterparts may reveal more optimal update rules (Andrychowicz et al., 2016). Learnable optimizers remain one of the oldest directions that still have only limited success (Schmidhuber, 2020; Metz et al., 2020). Yet, by analogy with other successful directions, if DNNs could replace SGD, then dramatic advances might follow. In this thesis, we make a step towards that long-standing and ambitious goal. Specifically, we develop graph reasoning methods that allow us to predict performant parameters of DNNs represented as graphs. Following the focus of previous chapters on aspects of generalization, we take into consideration samples beyond *i.i.d.* when developing and evaluating parameter prediction methods. This extends the scope of this thesis to more practical and challenging scenarios.

[1] Generally speaking, a DNN architecture is defined by the types of neural network layers, their number and connectivity between them.

2 Background

We consider prediction problems where we are given inputs and need to make some decision about them. In the context of image recognition, let us consider images of natural objects (*e.g.* dogs) as input. The task is to predict semantic content in the image. We will consider the image classification task where a single label per image (*e.g.* 'dog' or 'person') must be predicted, and more complex tasks where rich description (*e.g.* 'dog on surfboard') must be predicted (Fig. 2.1).

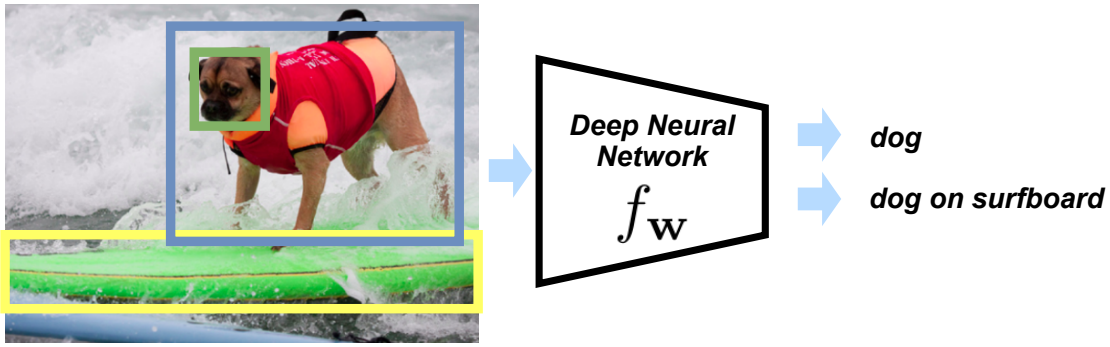


Figure 2.1: Prediction of object categories and their relationships given an image.

2.1 Neural networks

In image recognition tasks, the input is high dimensional[1] and the decision process is hard to formally describe using simple expert rules and raw inputs. Therefore, the currently dominant strategy to solve such a task is to formulate it as an optimization problem:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \; \mathcal{L}(f_{\mathbf{w}}, \mathcal{D}), \tag{2.1}$$

where $f_{\mathbf{w}}$ is some function parameterized by $\mathbf{w}$, $\mathcal{D}$ is the dataset associated with the task and $\mathcal{L}$ is the objective to minimize for this dataset. The goal of solving this problem is to find optimal parameters $\mathbf{w}^*$ for $f_{\mathbf{w}}$. For large and complex datasets, the state-of-the-art solutions to this optimization problem are often based on using deep neural networks (DNNs) as $f_{\mathbf{w}}$. Finding globally optimal $\mathbf{w}^*$ of DNNs is generally intractable due to the highly non-convex nature of $\mathcal{L}(f_{\mathbf{w}}, \mathcal{D})$ (Choromanska et al., 2015). Nevertheless, local optima can be found using methods based on gradient descent (Ruder, 2016; Kingma and Ba, 2015).

[1]Image resolution varies drastically between image tasks, but generally represent high-dimensional cases. For example, in MNIST (LeCun et al., 1998), the images are 28×28 pixels, *i.e.* 784 dimensional. In more realistic tasks, such as PASCAL (Everingham et al., 2010), images can be 500×300 pixels, *i.e.* 150,000 dimensional.

DNNs can have drastically different designs depending on the task. This thesis concerns three different designs of DNNs. The first is one of the earliest types of DNNs, colloquially referred to as a multilayer perceptron (MLP). Next, we consider convolutional neural networks (CNNs) – DNNs designed specifically for perceptual tasks, such as image recognition. Lastly, we consider graph neural networks (GNNs) that are designed for graph-structured data. MLPs, CNNs and GNNs are the core building blocks of this thesis.

2.1.1 Multilayer perceptrons

MLPs or, more precisely, multilayer feedforward fully-connected neural networks, consist of L layers of trainable parameters (weights) $\mathbf{w} = [\mathbf{W}^{(1)}, \mathbf{W}^{(2)}, ..., \mathbf{W}^{(L)}]$. Given N input data points $\mathbf{X} \in \mathbb{R}^{N \times d}$ of dimensionality d from the dataset $\mathcal{D}$, the MLP sequentially transforms the input $\mathbf{X}$ as[2]:

$$\mathbf{X}^{(l+1)} = \sigma(\mathbf{X}^{(l)}\mathbf{W}^{(l)}), \tag{2.2}$$

where $l \in [1, L]$ and an input to the first layer $\mathbf{X}^{(1)}$ is equal to $\mathbf{X}$. Function σ is some nonlinearity such as the Rectified Linear Unit (ReLU) applied element-wise to $\mathbf{X}^{(l)}\mathbf{W}^{(l)}$: $\sigma(\mathbf{X}^{(l)}\mathbf{W}^{(l)}) = \max(0, \mathbf{X}^{(l)}\mathbf{W}^{(l)})$. More advanced nonlinearities can lead to better training and generalization properties, e.g. leaky ReLU (Maas et al., 2013) or the Exponential Linear Unit (ELU) (Clevert et al., 2015). Applying σ is essential to learn nonlinear transformations. One of the simplest nonlinear transformations is the XOR logic operation that was famous in diminishing the interest in AI ("AI winter") in the 1970s[3]. For the final layer ($l = L$), it is common to use a nonlinearity specific for the task. For example, in the case of predicting binary labels $\mathbf{y} \in [0, 1]^N$ for data points $\mathbf{X}$, a sigmoid function can be applied. In the case of regression tasks, $\mathbf{y} \in \mathbb{R}^N$, *i.e.*, the final nonlinearity is removed. All N data points in $\mathbf{X}$ can be processed by MLPs independently, so parallel computing enables very efficient usage of MLPs. N data points processed in parallel are often called a mini-batch, or batch[4].

Applications of MLPs. MLPs can be in principle applied to any tabular data $\mathbf{X}$ where rows are data points and columns are features or dimensions. While images can be represented as tabular data

[2]For simplicity, in (2.2) we ignore the bias term $\mathbf{b}$ that in practice is added to the output of $\mathbf{X}^{(l)}\mathbf{W}^{(l)}$. The bias is essential when a single layer is used, however it has limited practical value in the case of $L > 1$.

[3]More about that period can be read at https://dev.to/jbahire/demystifying-the-xor-problem-1blk or https://towardsdatascience.com/history-of-the-first-ai-winter-6f8c2186f80b.

[4]When "batch" and "mini-batch" are used to describe the operation of learning, batch means updates based on the entire dataset and mini-batch means updates based on a subset.

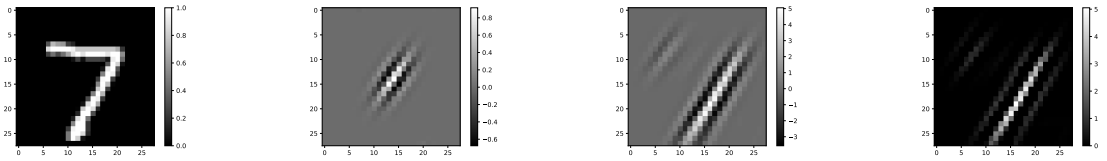
by flattening spatial dimensions (Ciregan et al., 2012), MLPs are more common in cases when the dimensions in $\mathbf{X}$ are not ordered in any meaningful way (*i.e.* there is no benefit of leveraging the order). MLPs are often used as building blocks of many other types of DNNs, including recently developed Transformers (Vaswani et al., 2017; Dosovitskiy et al., 2020) and Graph Neural Networks (Kipf and Welling, 2017) (§ 2.1.3). Oftentimes, the last few layers of Convolutional Neural Networks (§ 2.1.2) are modeled as MLPs (Simonyan and Zisserman, 2014b). Recently, models based on MLPs were revisited in large-scale image tasks, where they showed competitive results (Touvron et al., 2021).

2.1.2 Convolutional neural networks

The main building blocks of CNNs, convolution and downsampling, were introduced in (Fukushima and Miyake, 1982). Subsequently, in (LeCun et al., 1998), CNNs were combined with a gradient descent-based training algorithm to effectively learn the parameters of CNNs from raw inputs (images) without manually engineering features. Following (2.2) for the MLP layer, the convolutional layer for N images $\mathcal{X} \in \mathbb{R}^{N \times C \times H \times W}$ and K filters (kernels, weights) $\mathcal{W} \in \mathbb{R}^{K \times C \times h \times w}$ can be defined as (Vedaldi and Lenc, 2015):

$$\mathcal{X}^{(l+1)}_{n,k,i,j} = \sigma\Big(\sum_{c} \sum_{h} \sum_{w} \mathcal{X}^{(l)}_{n,c,i-h,j-w} \mathcal{W}^{(l)}_{k,c,h,w}\Big), \tag{2.3}$$

where C is the number of channels (*e.g.* $C = 3$ for RGB images); H, W are the height and width of images respectively; K is the number of filters and h, w are their height and width respectively. Each k-th filter of $\mathcal{W}$ slides over the input along the spatial dimensions and for each spatial location $i \in [1, H]$, $j \in [1, W]$ of $\mathcal{X}$ computes the dot product between the local region and the kernel (Fig. 2.2).



Input image | Convolutional filter | Output of convolution | Output after ReLU

Figure 2.2: Example of the convolution operation for a single image and single filter.

The convolution operation exploits a 2D local structure in images, formally described in (Bronstein et al., 2017):

- Shift-invariance – if we spatially translate an object on the image to the left/right/up/down, we still should be able to recognize it. This is exploited by sharing filters across all locations.

- Locality – nearby pixels are closely related and often represent some semantic concepts, such as object parts. This is exploited by using filters with spatial dimensions $h > 1$ and $w > 1$, which can capture image features in a local spatial neighborhood.
- Compositionality (or hierarchy) – a larger region in the image is often a semantic parent of smaller regions it contains. For example, a dog is a parent of a head, body, legs, etc. This implicitly is exploited by stacking convolutional layers.

Another important component of CNNs is the downsampling operation. Downsampling reduces the spatial size of inputs, which is important for computational efficiency, particularly for large inputs. Downsampling is usually based on spatial pooling applied to a local region similar to convolution. Typically, average or max poolings are used that do not have trainable parameters. In practice, pooling can be replaced with modified (strided) convolution to simplify the overall network (Springenberg et al., 2014).

As compared to MLPs, CNNs applied to images have another strength besides those listed above. In particular, the number of trainable parameters in convolutional filters $\mathcal{W}$ does not depend on the input spatial dimensions H, W. In principle, the same CNN can be trained on images with $H = W = 28$ as well as $H = W = 500$. In addition, CNNs are highly efficient due to their ability to apply filters $\mathcal{W}$ independently and in parallel for each spatial location, each filter and each image. As a result, CNNs have been adapted to a broad range of tasks beyond image classification.

Applications of CNNs. CNNs initially were proposed for simple image classification tasks, such as handwritten digit recognition (MNIST) (LeCun et al., 1998). Subsequently, a larger and deeper network proposed in (Krizhevsky et al., 2012) led to the top-1 result on the ImageNet 2012 challenge of large-scale image classification (Russakovsky et al., 2015) outperforming hand-designed visual features such as SIFT (Lowe, 2004). Compared to (LeCun et al., 1998), the main changes made in (Krizhevsky et al., 2012) to CNNs were: i) dramatically increased size of a CNN possible due to larger and cheaper computational resources available, ii) a large annotated dataset (*i.e.* ImageNet), iii) applying nonlinearities and regularization methods such as ReLU and Dropout (Hinton et al., 2012). Since then, CNNs have dominated visual tasks (Gu et al., 2018) and grown significantly in size showing superior results in image and video object detection (Ren et al., 2015; Wang et al., 2017; He et al., 2016, 2015), image and video semantic segmentation (Long et al., 2015; Shelhamer et al., 2016), video recognition (Simonyan and Zisserman, 2014a), image and video generation (Goodfellow et al., 2014; Vondrick et al., 2016), learning to estimate the optical flow between a pair of images (Dosovitskiy et al., 2015), and many other tasks. In these and more complex tasks, such as visual question answering (VQA) (Antol et al., 2015), CNNs are typically used as a "backbone" to extract visual features from images. The same backbone and its trained parameters

can be used across different tasks. The procedure of reusing the backbone parameters from one task to improve on another task is called "transfer learning". Typically, backbones trained on large image tasks such as ImageNet show the best transfer learning abilities (Huh et al., 2016; Kornblith et al., 2019). Finally, CNNs' computational efficiency has been significantly improved. In particular, architectural enhancements (Howard et al., 2017; Cai et al., 2019a), as well as compression and distillation techniques (Cheng et al., 2017) together with very efficient implementations (Chetlur et al., 2014) enabled the deployment of CNNs on low-resource mobile devices further extending CNNs' application reach.

2.1.3 Graph neural networks

The wide success of CNNs have motivated their application to more tasks. In tasks such as chemistry, physics and social networks, the data are represented as graphs. However, convolution (2.3) requires data to reside on a "regular grid", a Euclidean coordinate system where all the data points are located at the discrete and equally spaced coordinates consistent among all samples (Bronstein et al., 2017). For example, all MNIST images reside on the same 28×28 regular grid. In contrast, graphs generally reside on "irregular grids" that do not have a notion of spatial translation, preventing the application of convolution as per (2.3).

Formally, a graph $\mathcal{G}$ consists of an unordered set of N nodes, $\mathcal{V}$, connected by edges, $\mathcal{E}$. The edges are often encoded by an adjacency matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$. The number of neighbors for each node can be then defined as a diagonal matrix $\mathbf{D}$, where $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$. Defining convolution on graphs is non-trivial because the nodes are generally unordered, and not attached to a particular coordinate system, and the node degree $\mathbf{D}_{ii}$ can vary for each i-th node.

To define convolution on graphs, a spectral graph theory has been applied (Bruna et al., 2014). This theory is based on extending the spectral definition of convolution in signal processing. For signals, we can define spectral convolution equivalent to the spatial definition in (2.3) using the Discrete Fourier Transform. Similarly, spectral convolution on graphs can be computed (Belkin and Niyogi, 2001; Chung and Graham, 1997; Bruna et al., 2014)[5] based on the eigendecomposition of the graph Laplacian $\mathbf{L} = \mathbf{I}_N - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$, where $\mathbf{I}_N$ is an $N \times N$ identity matrix. In particular, the eigendecomposition of $\mathbf{L}$ is defined as $\mathbf{L} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T$, where $\mathbf{V}$ are eigenvectors and $\mathbf{\Lambda}$ are eigenvalues. Given node features $\mathbf{X}^{(l)} \in \mathbb{R}^{N\times d}$, the spectral graph convolution layer with filters $\mathbf{W}^{(l)}$ can be then defined as:

$$\mathbf{X}^{(l+1)} = \sigma\Big(\mathbf{V}(\mathbf{V}^T\mathbf{X}^{(l)} \odot \mathbf{V}^T\mathbf{W}^{(l)})\Big), \tag{2.4}$$

[5] An extended description of defining spectral convolution on graphs can be found in my blog post: https://towardsdatascience.com/spectral-graph-convolution-explained-and-implemented-step-by-step-2e495b57f801.

where $\odot$ is element-wise multiplication. Similarly to the spectral convolution in signal processing, in (2.4) the features and filters are first projected into the spectral domain where they are multiplied. The result is then reconstructed back to the original domain.

A major disadvantage of spectral graph convolution defined in (2.4) is the necessity to compute the eigendecomposition for each graph. In many graph tasks, graphs have very different structures (and different eigenvectors $\mathbf{V}$) and it is unclear if the same $\mathbf{W}^{(l)}$ can adapt to different $\mathbf{V}$ (Nilsson and Bresson, 2020). Moreover, computing eigendecomposition for large graphs is a computationally intensive process. To eliminate the need of eigendecomposition, spectral graph convolution (2.4) can be approximated using recursive Chebyshev polynomials T_k and the property of eigendecomposition that $\mathbf{L}^k = (\mathbf{V}\mathbf{\Lambda}\mathbf{V}^T)^k = \mathbf{V}\mathbf{\Lambda}^k\mathbf{V}^T$. To derive approximate Chebyshev graph convolution, the polynomials T_k are first applied to the rescaled graph Laplacian $\tilde{\mathbf{L}} = 2\mathbf{L}/\lambda_{\max} - \mathbf{I}_N$ (Hammond et al., 2011; Defferrard et al., 2016):

$$T_k(\tilde{\mathbf{L}}) = 2\tilde{\mathbf{L}}T_{k-1}(\tilde{\mathbf{L}}) - T_{k-2}(\tilde{\mathbf{L}}), \tag{2.5}$$

where $k \in [1, N]$, $T_0(\tilde{\mathbf{L}}) = \mathbf{I}_N$ and $T_1(\tilde{\mathbf{L}}) = \tilde{\mathbf{L}}$; $\lambda_{\max}$ is the largest eigenvalue of $\mathbf{L}$. Using (2.5) and the aforementioned property of eigendecomposition that $\mathbf{L}^k = \mathbf{V}\mathbf{\Lambda}^k\mathbf{V}^T$, Hammond et al. (2011); Defferrard et al. (2016) derived that spectral graph convolution can be approximated as a sum of the $T_k(\tilde{\mathbf{L}})$ terms weighted by trainable parameters $\mathbf{W}_k^{(l)}$. Hence, the Chebyshev graph convolution layer can be defined as:

$$\mathbf{X}^{(l+1)} = \sigma\Big(\sum_{k=0}^{K-1} T_k(\tilde{\mathbf{L}})\mathbf{X}^{(l)}\mathbf{W}_k^{(l)}\Big), \tag{2.6}$$

where $K \in [1, N]$ is a hyperparameter controlling how global is the receptive field of the convolution. In particular, the terms $T_k(\tilde{\mathbf{L}})$ include powers $\tilde{\mathbf{L}}^k$ enabling a $(k-1)$-hop receptive field and allowing to approximate spectral convolution. For example, for $K = N$ the convolution (2.6) is performed globally based on the entire graph structure making it approximately equal to spectral convolution (2.4). For $K = 1$ we have $T_0(\tilde{\mathbf{L}}) = \mathbf{I}_N$, so the convolution is performed ignoring the graph structure, while for $K = 2$ the convolution is performed based on the 1-hop neighborhood of nodes, and so forth. Stacking Chebyshev graph convolution layers (2.6) form a graph neural network called a ChebyNet studied in our works (Knyazev et al., 2018, 2019a,b).

Kipf and Welling (2017) studied the 1-hop Chebyshev graph convolution (with $K = 2$) and proposed its highly-effective and efficient simplification:

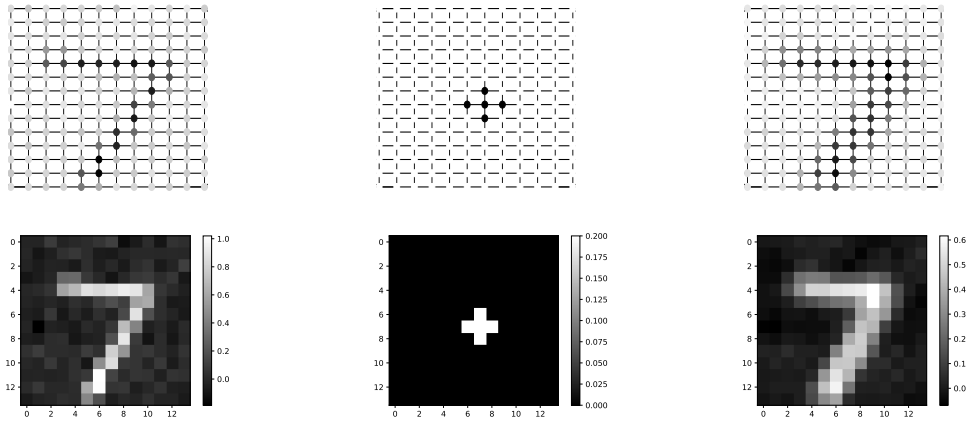
$$\mathbf{X}^{(l+1)} = \sigma(\hat{\mathbf{A}}\mathbf{X}^{(l)}\mathbf{W}^{(l)}), \tag{2.7}$$

where $\hat{\mathbf{A}}$ is a normalized adjacency matrix similar to the rescaled graph Laplacian $\tilde{\mathbf{L}}$: $\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$, and $\tilde{\mathbf{D}}_{ii} = \sum_j \tilde{\mathbf{A}}_{ij}$; $\tilde{\mathbf{A}} = \mathbf{A} + I_N$ to include self-loops into convolution. Essentially,

(2.7) combines the first two terms of (2.6) for $k = [1, 2]$ into a single operation. The model based on multiple convolutions (2.7) is commonly referred to as a graph convolutional network (GCN). In this thesis, we will use a more general term "graph neural network" (GNN) to refer to this and other graph models.

The graph layer defined in (2.7) is remarkably similar to the one of the MLP layer (2.2). The only difference is the normalized adjacency matrix $\hat{A}$ used in (2.7). Therefore, a GNN can be viewed as an MLP exploiting the relational information between data points (or node features).

Following the example in Fig. 2.2 (§ 2.1.2), graph convolution (2.7) can be illustrated based on an MNIST image. To represent an image as a graph, nodes correspond to pixel coordinates and edges connect only four spatially adjacent pixels (Fig. 2.3). In the graphs, pixel intensities are inverted for better visualization with black nodes corresponding to white pixels. The image is resized to 14×14 and a small amount of Gaussian noise is added to illustrate the effect of graph convolution. To compute the output, only $\hat{\mathbf{A}}\mathbf{X}^{(l)}$ is used while $\mathbf{W}^{(l)}$ is ignored. Such graph convolution is equivalent to a low-pass mean filter commonly used in signal processing to denoise the signal. The low-pass effect helps GNNs to excel in tasks where aggregating 1-hop node features is sufficient for high performance. However, 1-hop low-pass filtering also limits the expressive power of GNNs in tasks where complex long-range interactions between nodes are important (NT and Maehara, 2019; Knyazev et al., 2019a).



Input graph/image Convolutional filter Output graph/image

Figure 2.3: Example of the graph convolution operation for a single graph and single filter.

Extensions of GNNs. GNNs were first proposed as early as in 1997 by Sperduti and Starita (1997) and subsequently integrated with recurrent neural networks in (Gori et al., 2005; Scarselli et al., 2008). The works of Defferrard et al. (2016) and Kipf and Welling (2017) synergized with deep learning and the GNNs field has grown considerably. Notable extensions include Graph

Attention Networks (Velickovic et al., 2018) that learn pairwise attention between nodes to better capture regularities in graphs. Graph Isomorphism Networks (Xu et al., 2019a) and Principal Neighbourhood Aggregation (Corso et al., 2020) use novel more expressive feature aggregation strategies to better differentiate graphs and showed one of the best results in a large-scale graph benchmark (Hu et al., 2020). Simple GCNs (Wu et al., 2019) propose a single layer GNN that is computationally efficient yet performant in many tasks. Message Passing Networks (Gilmer et al., 2017; Battaglia et al., 2018) generalize GNNs to support edge features in addition to node features. Gated GNNs (Li et al., 2015) extended earlier GNNs (Gori et al., 2005; Scarselli et al., 2008) based on recurrent networks and recently showed top results among other GNNs in different tasks (Dwivedi et al., 2020).

Applications of GNNs. GNNs primarily focus on solving node and graph classification tasks and link prediction (Hamilton et al., 2017; Wu et al., 2020; Battaglia et al., 2018). Graph generation (You et al., 2018; Liao et al., 2019) and analysis of dynamic graphs (Kazemi et al., 2019; Trivedi et al., 2019) are also becoming common tasks. Graphs can be used to model virtually any kind of data, so besides solving classic graph tasks such as molecule classification (Xu et al., 2019a; Knyazev et al., 2018), GNNs have been recently applied to more diverse tasks: image classification (Knyazev et al., 2019a; Meyer-Bäse et al., 2020), semantic segmentation (Li and Gupta, 2018; Zhang et al., 2019f), visual relationship detection (Xu et al., 2017; Yang et al., 2018a), modelling neural network architectures (Zhang et al., 2018a; Wen et al., 2020), program synthesis (Zhang et al., 2018c), learning interactions between elements of complex systems (Kipf et al., 2018; Bapst et al., 2019) and many other tasks. Such a wide range of tasks shows a great potential of GNNs and, therefore, GNNs are a central focuses of this thesis.

Alternatives to GNNs. Recently, Transformers (Vaswani et al., 2017; Dosovitskiy et al., 2020) have become competitive in diverse tasks. In principle, these models are capable of capturing graph-structured data if the relational information is properly modelled. For example, if relative positional encoding is used (Shaw et al., 2018), Transformers can learn a graph representation similar to such GNNs as GATs (Velickovic et al., 2018). The relation between Transformers and GNNs has been more formally confirmed in (Dwivedi and Bresson, 2020; Yun et al., 2019).

Graph pooling. Like CNNs, GNNs can exploit the local structure of data to perform some type of downsampling or pooling of the input graph. In GNNs, pooling methods generally follow the same idea as in CNNs. However, in GNNs the pooling regions (sets of nodes) are often found based on clustering, since there is no regular grid as in images (Defferrard et al., 2016; Shaham et al., 2018; Ying et al., 2018). Differently from clustering-based graph pooling, top-k pooling

was proposed (Gao and Ji, 2019). Instead of clustering "similar" nodes, top-k propagates only part of the input disregarding the rest. Formally, given node features $\mathbf{X}^{(l)}$ for layer l, the output node features $\mathbf{Z}^{(l)}$ of top-k pooling can be defined as:

$$\mathbf{Z}_i^{(l)} = \begin{cases} \mathbf{a}_i \mathbf{X}_i^{(l)}, & \forall i \in P \\ \emptyset, & \text{otherwise,} \end{cases} \tag{2.8}$$

where P is a set of indices of pooled nodes, $|P| \leq N$, and $\emptyset$ denotes the unit is absent in the output. $\mathbf{a} \in \mathbb{R}^N$ is predicted by some auxiliary subnetwork: $\mathbf{a} = f(\mathbf{X}^{(l)}, \mathbf{A})$, where for f a GNN can be used as in (Lee et al., 2019a) or an MLP ignoring the graph structure (adjacency matrix $\mathbf{A}$) can be used as in (Gao and Ji, 2019). The indices P are the indices of the $|P|$ largest (top) values in $\mathbf{a}$.

Both GNNs and CNNs are built by stacking convolutional and pooling layers to form a deep network. Sometimes, CNNs are augmented with GNNs to improve learning in visual tasks (Li and Gupta, 2018; Liu et al., 2020).

2.2 Compositional and graph reasoning

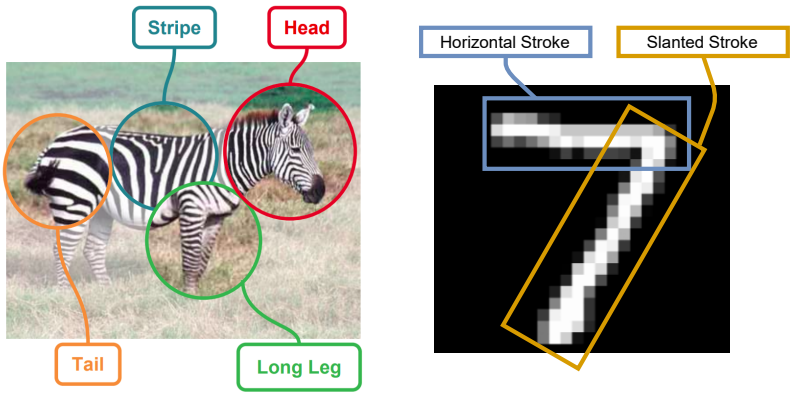
With the basic building blocks, MLPs, CNNs and GNNs, we can build systems that solve complex *compositional reasoning* tasks. In this thesis, by compositional reasoning we assume the process of making a decision by analyzing the collection, or *composition*, of entities where the entities can refer to graph nodes and edges; objects, object parts and relations; or abstract concepts or patterns (*e.g.* subgraphs, strokes, stripes). In this section, we will describe compositional reasoning methods that mainly concern vision tasks, since these tasks are well studied and easy to understand with simple examples. But similar tasks and methods solving them exist in different domains of compositional reasoning, such as graph reasoning (Hamilton et al., 2018) or natural language processing (Lake and Baroni, 2018).

A classic example of compositional visual reasoning is visual question answering (VQA) (Antol et al., 2015). In VQA, the decision is the answer obtained by visually reasoning over a set of objects and relationships between them given a question, *e.g. how many chairs are on the left of the table in this image?*. Simpler tasks such as classifying images can also be considered as visual reasoning tasks. Visual reasoning methods may be grouped into low-level and high-level ones and different methods are used in each case.

2.2.1 Low-level compositional reasoning

We can view image classification as a low-level visual reasoning task, since the model needs to recognize low-level components and relate them to each other in order to make an accurate prediction.

The low-level components can be object attributes and parts, parts of parts, or even individual pixels. Reasoning over object parts and attributes rather than making a direct decision about an image enables more explainable decisions (ul Hassan et al., 2019) and zero-shot object classification (Lampert et al., 2013; Demirel et al., 2017; Naeem et al., 2021; Tokmakov et al., 2019). For example, a relatively complex zebra image can be recognized if the model detects stripes, a tail, a head, long legs (Fig. 2.4a). A simple MNIST image can be recognized based on the composition of strokes and other patterns (Fig. 2.4b). Regardless of image complexity, the lowest level of reasoning is individual pixels. In complex images, the compositions of individual pixels are less likely to directly lead to a particular semantic decision. However, individual pixels can still affect the prediction of object parts and attributes, which in turn can affect the final prediction. Therefore, it is important to consider low-level reasoning both in simple and complex images to develop more robust and explainable models.



(a) (b)

Figure 2.4: (a) A zebra can be described as a collection of object parts, patterns and attributes (figure from (Demirel et al., 2017)); (b) A digit can be described as a collection of primitive strokes and patterns.

2.2.2 High-level compositional reasoning

High-level visual reasoning has been more extensively studied in different tasks than low-level reasoning. The most common, and perhaps, comprehensive high-level reasoning task is Visual Question Answering (VQA) (Antol et al., 2015; Johnson et al., 2017a). One of the ways to effectively solve VQA is to first extract a semantic image description – a scene graph (Hudson and Manning, 2019a; Yi et al., 2018; Zhang et al., 2019a). Formally, a scene graph (Johnson et al., 2015) $\mathcal{G} = (O, R)$ consists of a set of subjects and objects (O) as nodes and a set of relationships or predicates (R) between them as edges. The nodes and edges form visual relationship *triplets*: $\langle$*subject*, *predicate*, *object*$\rangle$, *e.g.* $\langle$cup, on, table$\rangle$. Solving VQA becomes much easier when the input to the question-answering module is semantic, such as a scene graph, rather than raw pixels

or abstract features. Similar to VQA, in image captioning (Yang et al., 2019; Gu et al., 2019a) and retrieval (Johnson et al., 2015; Belilovsky et al., 2017; Tang et al., 2020), extracting a scene graph from images also simplifies the task improving the downstream performance. Inferring a scene graph is also beneficial for explainable visual reasoning (Shi et al., 2019) within the explainable AI (XAI) paradigm (Gunning and Aha, 2019), since the final predictions can be traced back to semantic concepts of scene graphs. Extracting a scene graph generally requires a predefined vocabulary of concepts, which is a time-consuming process that must be done for each new task. Therefore, a more flexible strategy is to describe images using abstract entities (Norcliffe-Brown et al., 2018; Vedantam et al., 2019; Locatello et al., 2020; Burgess et al., 2019; Greff et al., 2020) that, if necessary, can be tied to semantic concepts (see § 2.2.4).

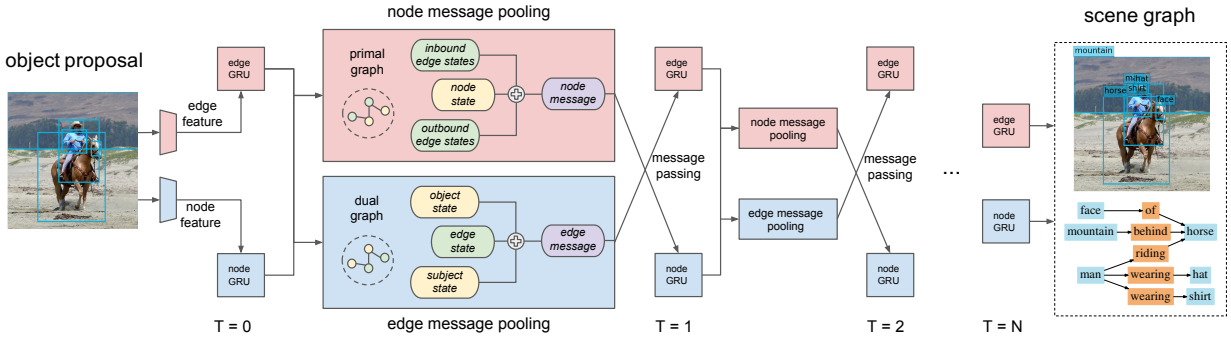


Figure 2.5: A scene graph generation model from (Xu et al., 2017) used in high-level visual reasoning tasks. This and other SGG models (Yang et al., 2018a) are often based on message passing networks that resemble graph neural networks (Gilmer et al., 2017; Battaglia et al., 2018).

Extracting a scene graph $\mathcal{G}$ from an image I is a standard high-level visual reasoning task and is called scene graph generation (SGG) (Xu et al., 2017). In general, SGG models first extract a complete graph from an image, where nodes correspond to detected objects (Zellers et al., 2018; Yang et al., 2018a). Then several message passing rounds update node and edge features. The goal of the SGG model is to predict a sparse scene graph $\mathcal{G}$ given the dense graph of node and edge features (Fig. 2.5). Typically, many different $\mathcal{G}$ can be valid for a single image I, so obtaining $\mathcal{G}$ resembles a generative process. However, in practice there is typically only one ground-truth $\mathcal{G}$ annotated for each image and the SGG models are typically deterministic, so the task is rather "scene graph prediction". Nevertheless, we will use the term SGG to be consistent with the scene graph literature.

2.2.3 Compositional generalization

In real world images, some compositions, *e.g.* ⟨cup, on, table⟩ or ⟨person, on, surfboard⟩, appear more frequently than other unusual ones, *e.g.* ⟨cup, on, *surfboard*⟩, ⟨cup, *under*, table⟩ or ⟨*dog*, on,

surfboard⟩. Such a difference in frequencies – the *frequency bias* – is often present in commonly-used visual relationship datasets, such as Visual Genome (Krishna et al., 2017a). The frequency bias is purely statistical and poorly reflects the physical plausibility of object interactions. For example, according to the statistics of Visual Genome the probability of ⟨cup, on, surfboard⟩ is exactly zero because such a composition has never occurred. However, from the physical point of view (in the real world), such a composition would have a greater than zero probability. In fact, ⟨cup, on, surfboard⟩ appears in the test set of Visual Genome. The ability of models to recognize such novel (*zero-shot* or ZS) and rare (*few-shot* or FS) compositions accurately, despite the frequency bias, is called *compositional generalization* (CoGen). Compositional generalization has been widely studied in the language (Atzmon et al., 2016; Keysers et al., 2019; Lake, 2019) and reinforcement learning domains (Jiang et al., 2019; Cogswell et al., 2019; Kipf et al., 2019), as well as multi-domain tasks (Johnson et al., 2017a; Bahdanau et al., 2018, 2019; Agrawal et al., 2017, 2018). In the visual domain, compositional reasoning has been addressed in the scene graph generation (SGG) task and image classification from attributes (Lampert et al., 2013; Demirel et al., 2017; Naeem et al., 2021) (Fig. 2.6).

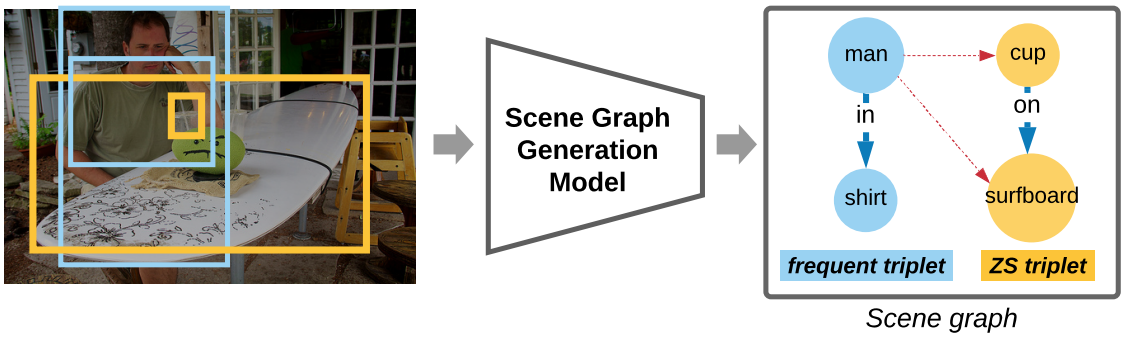


Figure 2.6: An example of a ground truth scene graph with a zero-shot composition (ZS triplet) that must be predicted by an SGG model for an input image. The figure is adapted from (Knyazev et al., 2020). Dashed red arrows denote the relationships that have not been annotated by a human.

While compositional reasoning about concepts is easy for humans, for machines this task has remained extremely challenging. The reasons for the challenging nature of CoGen are not well understood. The challenge may relate to the fact that learning-based models tend to capture spurious statistical correlations and biases of datasets during training (Arjovsky et al., 2019; Niu et al., 2020; Tang et al., 2020). The frequency bias of visual relationship datasets is particular pronounced, so for the learning-based models it is hard to not rely on this bias. The CoGen challenge has been largely overlooked, since the test sets often have the same frequency bias and the evaluation metrics do not penalize models for blindly relying on the bias. However, when the

evaluation is explicitly focused on CoGen, the models have been found to fail remarkably (Atzmon et al., 2016; Lu et al., 2016; Tang et al., 2020; Knyazev et al., 2020).

In the SGG task, recall-based metrics are typically used for evaluation. So, on frequent compositions these metrics can reach ~41%, while on zero-shot compositions the state-of-the-art result is only 4.5% – a nearly 10 fold drop in performance (Tang et al., 2020). Previous SGG works often assume CoGen is similar to few-shot predicate generalization and so attempt to improve mean (or predicate-normalized) recall metrics that are not directly related to CoGen (Chen et al., 2019a; Dornadula et al., 2019; Tang et al., 2019; Zhang et al., 2019e; Tang et al., 2020; Chen et al., 2019b; Zareian et al., 2020a; Yan et al., 2020). Predicate imbalance can be treated by simple resampling-based methods, while CoGen is more challenging (Tang et al., 2020). Therefore, CoGen rather than predicate imbalance has to be a focus of visual reasoning tasks.

2.2.4 Methods to improve compositional generalization

The methods to improve visual compositional generalization (CoGen) can be grouped into two categories. These are: methods that explicitly impose some compositional inductive prior on the models, and those where CoGen comes, or can potentially come, as a side-effect. The side-effect can be, for example, a result of a regularization method applied to neural networks.

Explicit Compositionality. Methods to introduce explicit compositionality can be grouped into high-level and lower-level reasoning tasks. The works on other forms of generalization related to high-level compositional generalization are also discussed.

- **High-level visual reasoning**: High-level visual CoGen was first evaluated in (Lu et al., 2016) on the VRD dataset using a joint vision-language model. Several follow-up works attempted to improve upon it: by learning a translation operator in the embedding space (Zhang et al., 2017), clustering in a weakly-supervised fashion (Peyre et al., 2017), using conditional random fields (Cong et al., 2018) or optimizing a cycle-consistency loss to learn object-agnostic features (Yang et al., 2018b). Augmentation using generative models to synthesize more examples of rare compositions is another promising approach (Wang et al., 2019b), because we can generate many instances of rare compositions mitigating the frequency bias. But, in (Wang et al., 2019b) this approach was only evaluated on a simple predicate classification task. Most recently, Tang et al. (2020) proposed to mitigate the bias by inferring causal rather than correlated relationships and, consequently, showed strong performance on zero-shot visual compositions. In a subsequent work, Suhail et al. (2021) improved the SGG loss function (4.2) to reduce the bias and better handle CoGen. In the VQA task, compositionality has been improved by predefining

neural modules (Andreas et al., 2016) and their more flexible end-to-end extensions (Hu et al., 2017; Johnson et al., 2017b). However, since the VQA task often relies on accurately extracting scene graphs, the methods that impose a compositional prior on scene graph prediction improve CoGen in VQA (Yi et al., 2018; Mascharka et al., 2018; Shi et al., 2019).

- **Lower-level visual reasoning**: The area of low-level visual reasoning is less organized and there is no standard evaluation benchmark. So different works have focused on different aspects of compositionality at the level of simple objects, object parts and attributes. In particular, to improve compositionality of simple objects, a mask-based loss term was added to image classification networks in (Stone et al., 2017). However, this loss requires expensive pixel-wise mask annotations for training images. In another work (Sylvain et al., 2019), to improve generalization to unseen object categories, a more local representation using self-supervised objectives based on Deep InfoMax (Hjelm et al., 2019) was learned. Zero-shot object classification was also studied in (Tokmakov et al., 2019). The model in (Tokmakov et al., 2019) is based on decomposing the image representation into a set of attribute representations in the visual space. However, the model does not require to annotate attributes for novel classes to predict their labels. Generalization to zero-shot objects and object-attribute compositions may be approached by learning an image extraction CNN together with a GNN that learns a knowledge graph from existing object categories and their attributes (Naeem et al., 2021). Another approach to this task is based on a prototypical model that learns object representations disentangled from attribute representations to enable strong generalization to unseen compositions of objects and attributes (Ruis et al., 2021). To further progress in the lower-level CoGen, more standardized benchmarks are needed. Integration of the lower-level and higher-level CoGen methods and evaluation protocols can also enable faster progress towards better generalization in visual reasoning.

- **Other tasks**: Several general methods exist that can be potentially useful for CoGen. One such method is unsupervised domain adaptation (UDA) by backpropagation (Ganin and Lempitsky, 2015) closely connected to domain-adversarial neural networks(Ajakan et al., 2014; Ganin et al., 2016). UDA achieved strong results by learning features invariant to the domain. Such a model allows to recognize objects in novel domains and contexts. Another general method is meta-learning (Hospedales et al., 2020) that typically targets few-shot generalization in classification tasks. The idea of commonly-used meta-learning methods, such as MAML (Finn et al., 2017), is to take the original training dataset and split it into a sequence of training and validation subsets (episodes). The critical part is to make the validation set largely composed of few-shot data. This way, the meta-learning algorithm aims to update the parameters of a model on the validation loss thereby improving it by learning to generalize to a few examples. In compositional

language reasoning, such a meta-learning based objective was proposed in (Lake, 2019), where the validation set is largely composed of zero and few shot compositions. This method yielded improvement CoGen on language tasks, and potentially, can be applied to visual tasks.

Implicit Compositionality via Object-centric Learning. Unsupervised learning has recently received more attention in different visual tasks (Radford et al., 2015; Hjelm et al., 2019; Chen et al., 2020a; Verma et al., 2021), and is potentially useful for CoGen as well. In particular, one of the reasons for poor CoGen of models might be the biased annotations in the datasets on which models are trained. Therefore, a logical way to mitigate such bias is to rely less on the annotations. In an extreme case, we can train a model without any labels, in a purely unsupervised fashion. In the context of compositional visual reasoning, a growing body of unsupervised learning works focus on object-centric learning (Greff et al., 2020), usually by employing an encoder-decoder model (Engelcke et al., 2019; Burgess et al., 2019; Greff et al., 2019; Locatello et al., 2020). Object-centric learning methods generally decompose an image representation into a set of object representations without accessing the labels of the objects. Some methods also allow to disentangle physical attributes of objects, such as color, shape and material (Greff et al., 2019). The decomposition of an image into objects is typically done by iteratively running encoder-decoder inference until the image is fully reconstructed (Greff et al., 2019). Due to its iterative nature, the inference procedure is computationally inefficient. Another method, slot attention (Locatello et al., 2020), is more efficient, since it only requires a single encoder iteration to extract all object representations. Yet, it is unclear if this model disentangles object attributes as in (Greff et al., 2019). Slot attention is reminiscent to the k-means clustering method. Unlike k-means, slot attention is fully-differentiable and employs self-attention (Vaswani et al., 2017) with the softmax function to enforce more sparse representation. Object-centric learning methods are typically evaluated using pixel-wise segmentation metrics similar to earlier unsupervised semantic segmentation works (Arbelaez et al., 2010). In addition, in (Locatello et al., 2020; Greff et al., 2019) the evaluation includes how well the representation encodes visual object properties. Overall, object-centric learning is a promising direction for CoGen as it enables an unbiased (w.r.t. human annotations) decomposition of images into entities that often have a semantic meaning. Such unbiased decomposition recently allowed object-centric learning methods to improve results on several out-of-distribution generalization tasks (Dittadi et al., 2021).

Implicit Compositionality via Regularized Training. Regularization methods are often aimed at reducing overfitting and improving different generalization abilities. An open question remains whether or not these methods can also improve CoGen. In the following, the methods that can be more directly leveraged for compositional generalization are considered.

Let us consider the feature activations after some layer l: $\mathbf{X}^{(l)} \in \mathbb{R}^{N \times d_l}$, where N is the number of data points (in a batch) and d_l is dimensionality[6]. To index the i-th individual feature (scalar) of the n-th sample, the notation $\mathbf{X}_{n,i}$ will be used. In the visual domain, when a CNN is used to extract $\mathbf{X}$, these activations tend to be highly-correlated due to the regularities in the input data and co-adaptation of weights to capture those regularities (Hinton et al., 2012), *i.e.* the probability $p(\mathbf{X}_{n,i}|\mathbf{X}_{n,j})$ tends to be high. For example, if the i-th feature is activated when the input image contains 'surfboard', then the j-th feature associated with the entity 'person' is likely to be activated regardless if the image actually contains the 'person' or another object such as 'dog'. On the one hand, relying on co-adaptation allows neural networks to fit data more easily. On the other hand, heavy reliance on co-adaptation can hurt generalization, so some regularization strategies are needed to mitigate that.

Many regularization strategies to alleviate overfitting have been proposed. One common strategy is Dropout (Hinton et al., 2012): dropout($\mathbf{X}, r$). Dropout stochastically sets to zero the values of $\mathbf{X}$ with probability r during training. Ghiasi et al. (2018) generalized this method to convolutions by setting to zero locally connected activations rather than arbitrary ones. In contrast, Cogswell et al. (2015) proposed a covariance loss penalty to explicitly reduce correlation of features. Adding the loss penalty to the task objective helped the networks to obtain better generalization properties compared to using Dropout. However, due to the expensive procedure of computing covariance, this approach does not scale well to high-dimensional features typically present in visual tasks. This limitation was addressed by introducing a locally connected decorrelation penalty specific for convolutional features (Rodríguez et al., 2016). Instead of adding a loss penalty (Cogswell et al., 2015; Rodríguez et al., 2016), enforcing orthogonality on weights in CNNs during initialization may help to better regularize the model and, subsequently, achieve better generalization results (Bansal et al., 2018; Wang et al., 2020a). However, it is important to maintain the orthogonality regularization during the whole training procedure, because the weights tend to diverge to a poor solution otherwise (Wang et al., 2020a). Alternatively, generalization can also be improved using decorrelated batch normalization (BN) (Huang et al., 2018), which can also be viewed as a form of regularization. While decorrelated BN improves generalization compared to original BN (Ioffe and Szegedy, 2015), it remains unclear if decorrelated BN is better for generalization than other regularization strategies, such as orthogonal regularization.

The discussed regularization methods mainly improve generalization results in a more classic machine learning sense, such as generalization to the in-distribution test images. However, except for a few synthetic experiments in (Cogswell et al., 2015), the effect of these methods on out-

[6] $\mathbf{X}^{(l)}$ can be outputs of a fully-connected, convolutional, graph layer, etc. In the case of 2D or 3D dimensions in $\mathbf{X}^{(l)}$, such as after convolutions, it can be flattened to a 1D tensor.

of-distribution and, especially, compositional generalization has not been systematically evaluated. Meanwhile, these regularization techniques, in particular the orthogonal one, can facilitate learning a representation where entities are more (linearly) independent, and hence disentangled, from each other. This might directly improve CoGen, which needs to be empirically confirmed.

3 Understanding Attention and Generalization in Graph Neural Networks

Prologue

Context. Graph neural networks (GNNs) are state of the art models for machine learning on graphs (see (Wu et al., 2020; Bronstein et al., 2017; Hamilton et al., 2017) for reviews). The downstream performance of GNNs is often improved by learning an attention module deciding which graph substructures are important for a given task (Gao and Ji, 2019). However, prior work on attention has not analyzed if attention learns anything useful about the task or whether the performance improvement is due to other confounding factors associated with adding attention. Compared to more intuitive tasks like image classification, in graph tasks it is often challenging to annotate ground truth about which graph substructures are important for the task. Therefore, it is especially challenging to understand the effect of attention in GNNs.

Contributions. We developed synthetic tasks where we can easily define ground truth importance of graph substructures for the task. Using these tasks, we found that attention can learn useful patterns about data if attention modules are initialized in a "good" way. We also found that the initialization effect is especially pronounced on the generalization ability of GNNs, in particular when evaluated on larger and more noisy graphs compared to the graphs used to train GNNs. We show that on real graph datasets, such a "good" initialization is hard to achieve. To this end, we provide a simple method that makes attention modules less sensitive to initialization. Consequently, we show improved generalization in several real graph tasks.

Recent works. The datasets we had introduced in our paper have been subsequently used in recent works (Dwivedi et al., 2020; Vincent-Cuaz et al., 2021; Ma et al., 2020; Wang et al., 2020b). Motivated by our work, attention and attention-based pooling in GNNs has been improved to extract graph substructures more accurately (Kim and Oh, 2020; Wang et al., 2020b; Ji et al., 2020). Evaluating the generalization of GNNs *w.r.t.* the graph size and other properties have become common (Veli*v*ckovi$\acute{c}$ et al., 2019; Verma and Zhang, 2019; Sinha et al., 2020). Theoretical generalization bounds for GNNs studied in (Garg et al., 2020) complement our empirical results by providing conditions for GNNs generalization.

3.1 Attention meets pooling in graph neural networks

The practical importance of attention in deep learning is well-established and there are many arguments in its favor (Vaswani et al., 2017), including interpretability (Park et al., 2016; Deac

et al., 2018). In graph neural networks (GNNs), attention can be defined over edges (Velickovic et al., 2018; Zhang et al., 2018b) or over nodes (Lee et al., 2018a). In this work, we focus on the latter, because, despite being equally important in certain tasks, it is not as thoroughly studied (Lee et al., 2018b). To begin our description, we first establish a connection between attention and pooling methods. In convolutional neural networks (CNNs), pooling methods are generally based on uniformly dividing the regular grid (such as one-dimensional temporal grid in audio) into local regions and taking a single value from that region (average, weighted average, max, stochastic, etc.), while attention in CNNs is typically a separate mechanism that weights C-dimensional input $\mathbf{X} \in \mathbb{R}^{N \times C}$:

$$\mathbf{Z} = \alpha \odot \mathbf{X}, \tag{3.1}$$

where $\mathbf{Z}_i = \alpha_i \mathbf{X}_i$ - output for unit (node in a graph) i, $\sum_i^N \alpha_i = 1$, $\odot$ - element-wise multiplication, N - the number of units in the input (i.e. number of nodes in a graph).

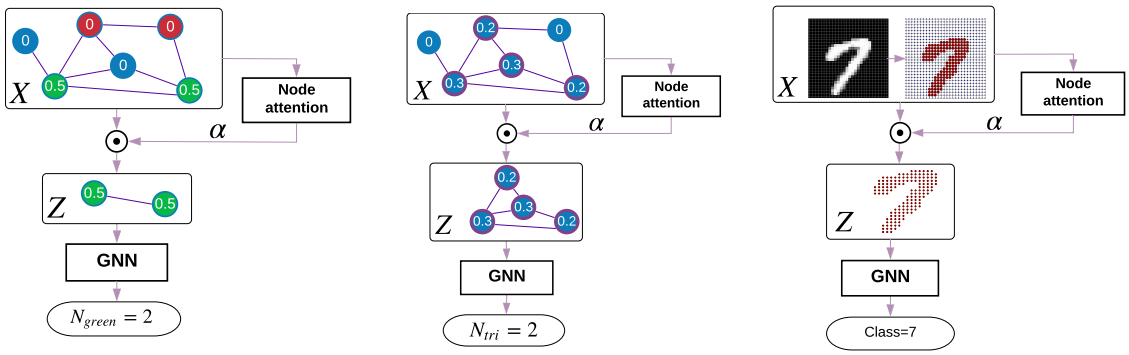
In GNNs, pooling methods generally follow the same pattern as in CNNs, but the pooling regions (sets of nodes) are often found based on clustering (Defferrard et al., 2016; Shaham et al., 2018; Ying et al., 2018), since there is no grid that can be uniformly divided into regions in the same way across all examples (graphs) in the dataset. Recently, top-k pooling (Gao and Ji, 2019) was proposed, diverging from other methods: instead of clustering "similar" nodes, it propagates only part of the input and this part is not uniformly sampled from the input. Top-k pooling can thus select some local part of the input graph, completely ignoring the rest. For this reason at first glance it does not appear to be logical.

However, we can notice that pooled feature maps in (Gao and Ji, 2019, Eq. 2) are computed in the same way as attention outputs $\mathbf{Z}$ in (3.1) above, if we rewrite their Eq. 2 in the following way:

$$\mathbf{Z}_i = \begin{cases} \alpha_i \mathbf{X}_i, & \forall i \in P \\ \emptyset, & \text{otherwise}, \end{cases} \tag{3.2}$$

where P is a set of indices of pooled nodes, $|P| \leq N$, and $\emptyset$ denotes the unit is absent in the output.

The only difference between (3.2) and (3.1) is that $\mathbf{Z} \in \mathbb{R}^{|P| \times C}$, i.e. the number of units in the output is smaller or, formally, there exists a ratio $r = |P|/N \leq 1$ of preserved nodes. We leverage this finding to integrate attention and pooling into a unified computational block of a GNN. In contrast, in CNNs, it is challenging to achieve this, because the input is defined on a regular grid, so we need to maintain resolution for all examples in the dataset after each pooling layer. In GNNs, we can remove any number of nodes, so that the next layer will receive a smaller graph. When applied to the input layer, this form of attention-based pooling also brings us interpretability of predictions, since the network makes a decision only based on pooled nodes.



(a) COLORS (b) TRIANGLES (c) MNIST

Figure 3.1: Three tasks with a controlled environment we consider in this work. The values inside the nodes are ground truth attention coefficients, α_i^{GT}, which we find heuristically (see § 3.3.1).

Despite the appealing nature of attention, it is often unstable to train and the conditions under which it fails or succeeds are unclear. Motivated by insights of (Xu et al., 2019b) recently proposed Graph Isomorphism Networks (GIN), we design two simple graph reasoning tasks that allow us to study attention in a controlled environment where we know ground truth attention. The first task is counting colors in a graph (COLORS), where a color is a unique discrete feature. The second task is counting the number of triangles in a graph (TRIANGLES). We confirm our observations on a standard benchmark, MNIST (LeCun et al., 1998) (Fig. 3.1), and identify factors influencing the effectiveness of attention.

Our synthetic experiments also allow us to study the ability of attention GNNs to generalize to larger, more complex or noisy graphs. Aiming to provide a recipe to train more effective, stable and robust attention GNNs, we propose a weakly-supervised scheme to train attention, that does not require ground truth attention scores, and as such is agnostic to a dataset and the choice of a model. We validate the effectiveness of this scheme on our synthetic datasets, as well as on MNIST and on real graph classification benchmarks in which ground truth attention is unavailable and hard to define, namely COLLAB (Leskovec et al., 2007; Shrivastava and Li, 2014), PROTEINS (Borgwardt et al., 2005), and D&D (Dobson and Doig, 2003).

3.2 Model

We study two variants of GNNs: Graph Convolutional Networks (GCN) (Kipf and Welling, 2017) and Graph Isomorphism Networks (GIN) (Xu et al., 2019b). One of the main ideas of GIN is to replace the MEAN aggregator over nodes, such as the one in GCN, with a SUM aggregator, and add more fully-connected layers after aggregating neigboring node features. The resulting model can distinguish a wider range of graph structures than previous models (Xu et al., 2019b, Figure 3).

3.2.1 Thresholding by attention coefficients

To pool the nodes in a graph using the method from(Gao and Ji, 2019) a predefined ratio $r = |P|/N$ must be chosen for the entire dataset (3.2). For instance, for $r = 0.8$ only 80% of nodes are left after each pooling layer. Intuitively, it is clear that this ratio should be different for small and large graphs. Therefore, we propose to choose threshold $\tilde{\alpha}$, such that only nodes with attention values $\alpha_i > \tilde{\alpha}$ are propagated:

$$\mathbf{Z}_i = \begin{cases} \alpha_i \mathbf{X}_i, & \forall i : \alpha_i > \tilde{\alpha} \\ \emptyset, & \text{otherwise.} \end{cases} \tag{3.3}$$

Note, that dropping nodes from a graph is different from keeping nodes with very small, or even zero, feature values, because a bias is added to node features after the following graph convolution layer affecting features of neighbors. An important potential issue of dropping nodes is the change of graph structure and emergence of isolated nodes. However, in our experiments we typically observe that the model predicts similar α for nearby nodes, so that an entire local neighborhood is pooled or dropped, as opposed to clustering-based methods which collapse each neighborhood to a single node. We provide a quantitative and qualitative comparison in § 3.3.

3.2.2 Attention subnetwork

To train an attention model that predicts the coefficients for nodes, we consider two approaches: (1) Linear Projection (Gao and Ji, 2019), where a single layer projection $\mathbf{p} \in \mathbb{R}^C$ is trained: $\alpha_{pre} = \mathbf{X}\mathbf{p}$; and (2) DiffPool (Ying et al., 2018), where a separate GNN is trained:

$$\alpha_{pre} = \text{GNN}(\mathbf{A}, \mathbf{X}), \tag{3.4}$$

where $\mathbf{A}$ is the adjacency matrix of a graph. In all cases, we use a softmax activation (Vaswani et al., 2017; Park et al., 2016) instead of tanh in (Gao and Ji, 2019), because it provides more interpretable results and ecourages sparse outputs: $\alpha = \text{softmax}(\alpha_{pre})$. To train attention in a supervised or weakly-supervised way, we use the Kullback-Leibler divergence loss (see § 3.3.3).

Note that the GNN used in (3.4) is called an attention subnetwork because its outputs $\alpha = \text{softmax}(\alpha_{pre})$ are used to weight the input node features $\mathbf{X}$ according to (3.3): $\mathbf{Z} = \alpha \odot \mathbf{X}$. This way α defines how much the following layers pay attention (or attend) to each of the nodes as we discuss in § 3.1.

3.2.3 ChebyGIN

In some of our experiments, the performance of both GCNs and GINs is quite poor and, consequently, it is also hard for the attention subnetwork to learn. By combining GIN (Xu et al., 2019b) with ChebyNet (Defferrard et al., 2016), we propose a stronger model, ChebyGIN. ChebyNet is a multiscale extension of GCN (Kipf and Welling, 2017), so that for the first scale, $k = 0$, node features are node features themselves, for $k = 1$ features are averaged over one-hop neighbors, for $k = 2$ - over two-hop neighbors and so forth (see (2.6) in § 2.1.3). To implement the Sum aggregator in ChebyGIN, we multiply node features $\mathbf{X}$ by a diagonal node degree matrix $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$ starting from $k = 1$. We also add more fully-connected layers (denoted as $\text{MLP}^{(l)}$) after feature aggregation as in GIN. Thus, following (2.6), the l-th layer of our ChebyGIN is defined as:

$$\mathbf{X}^{(l+1)} = \text{MLP}^{(l)}\Big(\sum_{k=0}^{K-1} \mathbf{D}_{ii}^{(k)} T_k(\tilde{\mathbf{L}}) \mathbf{X}^{(l)} \mathbf{W}_k^{(l)}\Big), \tag{3.5}$$

where $\mathbf{D}_{ii}^{(k)} = \mathbf{I}_N$ for $k = 0$ and $\mathbf{D}_{ii}^{(k)} = \sum_j \mathbf{A}_{ij}$ for $k > 0$. $T_k(\tilde{\mathbf{L}})$ are the Chebyshev polynomials applied to a rescaled graph Laplacian $\tilde{\mathbf{L}}$; $\mathbf{W}_k^{(l)}$ are trainable weights. See a detailed description of these terms in § 2.1.3 and the implementation details of ChebyGIN in our code https://github.com/bknyaz/graph_attention_pool/blob/master/chebygin.py.

3.3 Experiments

We introduce the color counting task (Colors) and the triangle counting task (Triangles) in which we generate synthetic training and test graphs. We also experiment with MNIST images (LeCun et al., 1998) and three molecule and social datasets. In Colors, Triangles and MNIST tasks (Fig. 3.1), we assume to know ground truth attention, i.e. for each node i we heuristically define its importance in solving the task correctly, $\alpha_i^{GT} \in [0, 1]$, which is necessary to train (in the supervised case) and evaluate our attention models.

3.3.1 Datasets

Colors. We introduce the color counting task. We generate random graphs where features for each node are assigned to one of the three one-hot values (colors): [1,0,0] (red), [0,1,0] (green), [0,0,1] (blue). The task is to count the number of green nodes, N_{green}. This is a trivial task, but it lets us study the influence of initialization of the attention model $\mathbf{p} \in \mathbb{R}^3$ on the training dynamics. In this task, graph structure is unimportant and edges of graphs act like a medium to exchange node features. Ground truth attention is $\alpha_i^{GT} = 1/N_{green}$, when i corresponds to green nodes and

$\alpha_i^{GT} = 0$ otherwise. We also extend this dataset to higher n-dimensional cases $\mathbf{p} \in \mathbb{R}^n$ to study how model performance changes with n. In these cases, node features are still one-hot vectors and we classify the number of nodes where the second feature is one.

Triangles. Counting the number of triangles in a graph is a well-known task which can be solved analytically by computing $\text{trace}(\mathbf{A}^3)/6$, where $\mathbf{A}$ is an adjacency matrix. This task turned out to be hard for GNNs, so we add node degree features as one-hot vectors to all graphs, so that the model can exploit both graph structure and features. Compared to the Colors task, here it is more challenging to study the effect of initializing $\mathbf{p}$, but we can still calculate ground truth attention as $\alpha_i^{GT} = T_i / \sum_i T_i$, where T_i is the number of triangles that include node i, so that $\alpha_i^{GT} = 0$ for nodes that are not part of triangles.

Mnist-75sp. Mnist (LeCun et al., 1998) contains 70k grayscale images of size 28×28 pixels. While each of 784 pixels can be represented as a node, we follow (Monti et al., 2017; Fey et al., 2018) and consider an alternative approach to highlight the ability of GNNs to work on irregular grids. In particular, each image can be represented as a small set of superpixels without losing essential class-specific information (see Fig. 3.2). We compute SLIC (Achanta et al., 2012) superpixels for each image and build a graph, in which each node corresponds to a superpixel with node features being pixel intensity values and coordinates of their centers of masses. We extract $N \leq 75$ superpixels, hence the dataset is denoted as MNIST-75sp. Edges are formed based on spatial distance between superpixel centers as in (Defferrard et al., 2016, Eq. 8). Each image depicts a handwritten digit from 0 to 9 and the task is to classify the image. Ground truth attention is considered to be $\alpha_i^{GT} = 1/N_{nonzero}$ for superpixels with nonzero intensity, and $N_{nonzero}$ is the total number of such superpixels. The idea is that only nonzero superpixels determine the digit class.

Molecule and social datasets. We extend our study to more practical cases, where ground truth attention is not available, and experiment with protein datasets: Proteins (Borgwardt et al., 2005) and D&D (Dobson and Doig, 2003), and a scientific collaboration dataset, Collab (Leskovec et al., 2007; Shrivastava and Li, 2014). These are standard graph classification benchmarks. A standard way to evaluate models on these datasets is to perform 10-fold cross-validation and report average accuracy (Yanardag and Vishwanathan, 2015; Ying et al., 2018). In this work, we are concerned about a model's ability to generalize to larger and more complex or noisy graphs, therefore, we generate splits based on the number of nodes. For instance, for Proteins we train on graphs with $N \leq 25$ nodes and test on graphs with $6 \leq N \leq 620$ nodes (see Table 3.2 for details about splits of other datasets and results).

3.3.2 Generalization to larger and noisy graphs

One of the core strengths of attention is that it makes it easier to generalize to unseen, potentially more complex and/or noisy, inputs by reducing them to better resemble certain inputs in the training set. To examine this phenomenon, for COLORS and TRIANGLES tasks we add test graphs that can be several times larger (TEST-LARGE) than the training ones. For COLORS we further extend it by adding unseen colors to the test set (TEST-LARGEC) in the format $[c_1, c_2, c_3, c_4]$, where $c_i = 0$ for $i \neq 2$ if $c_2 = 1$ and $c_i \in [0, 1]$ for $i \neq 2$ if $c_2 = 0$, i.e. there is no new colors that have nonzero values in a green channel. This can be interpreted as adding mixtures of red, blue and transparency channels, with nine possible colors in total as opposed to three in the training set (Fig. 3.2).

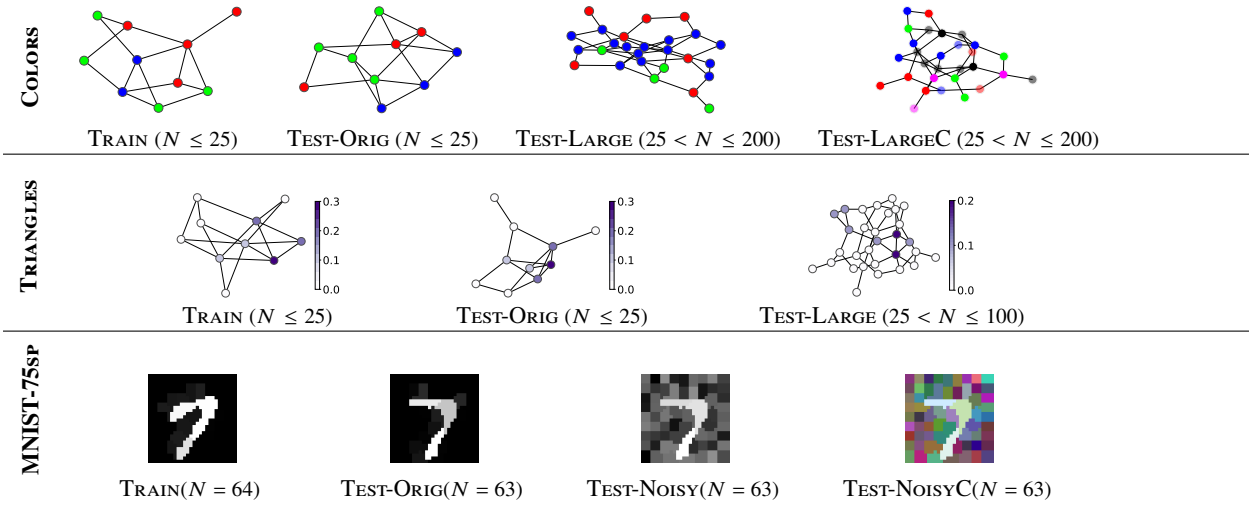


Figure 3.2: Examples from training and test sets. For COLORS, the correct label is $N_{green} = 4$ in all cases; for TRIANGLES $N_{tri} = 3$ and color intensities denote ground truth attention values α^{GT}. The range of the number of nodes, N, is shown in each case. For MNIST-75SP, we visualize graphs for digit 7 by assigning an average intensity value to all pixels within a superpixel. Even though superpixels have certain shapes and borders between each other (visible only on noisy graphs), we feed only superpixel intensities and coordinates of their centers of masses to our GNNs.

Neural networks (NNs) have been observed to be brittle if they are fed with test samples corrupted in a subtle way, i.e. by adding a noise (Dodge and Karam, 2017) or changing a sample in an adversarial way (Szegedy et al., 2013), such that a human can still recognize them fairly well. To study this problem, test sets of standard image benchmarks have been enlarged by adding corrupted images (Hendrycks and Dietterich, 2019).

Graph neural networks, as a particular case of NNs, inherit this weakness. The attention mechanism, if designed and trained properly, can improve a net's robustness by attending to only important and ignoring misleading parts (nodes) of data. In this work, we explore the ability of GNNs with and without attention to generalize to noisy graphs and unseen node features. This

should help us to understand the limits of GNNs, and potentially NNs in general, with attention and conditions when it succeedes and when it does not. To this end, we generate two additional test sets for MNIST-75SP. In the first set, TEST-NOISY, we add Gaussian noise, drawn from $\mathcal{N}(0, 0.4)$, to superpixel intensity features, i.e. the shape and coordinates of superpixels are the same as in the original clean test set. In the second set, TEST-NOISY-C, we colorize images by adding two more channels and add independent Gaussian noise, drawn from $\mathcal{N}(0, 0.6)$, to each channel (Fig. 3.2).

3.3.3 Network architectures and training

We build 2 layer GNNs for COLORS and 3 layer GNNs for other tasks with 64 filters in each layer, except for MNIST-75SP where we have more filters. Our baselines are GNNs with global sum or max pooling (gpool), DiffPool (Ying et al., 2018) and top-k pooling (Gao and Ji, 2019). We add two layers of our pooling for TRIANGLES, each of which is a GNN with 3 layers and 32 filters (3.4); whereas a single pooling layer in the form of vector $\mathbf{p}$ is used in other cases. We train all models with Adam (Kingma and Ba, 2015), learning rate 1e-3, batch size 32, weight decay 1e-4.

For COLORS and TRIANGLES we minimize the regression loss (MSE) and cross entropy (CE) for other tasks, denoted as $\mathcal{L}_{MSE/CE}$. For experiments with supervised and weakly-supervised (described below in § 3.3.4) attention, we additionally minimize the Kullback-Leibler (KL) divergence loss between ground truth attention α^{GT} and predicted coefficients α. The KL term is weighted by scale β, so that the total loss for some training graph with N nodes becomes:

$$\mathcal{L} = \mathcal{L}_{MSE/CE} + \frac{\beta}{N} \sum_i \alpha_i^{GT} \log(\frac{\alpha_i^{GT}}{\alpha_i}). \tag{3.6}$$

We repeat experiments at least 10 times and report an average accuracy and standard deviation in Tables 3.1 and 3.2. For COLORS we run experiments 100 times, since we observe larger variance. In Table 3.1 we report results on all test subsets independently. In all other experiments on COLORS, TRIANGLES and MNIST-75SP , we report an average accuracy on the combined test set. For COLLAB, PROTEINS and D&D , we run experiments 10 times using splits described in § 3.3.1.

The only hyperparameters that we tune in our experiments are threshold $\tilde{\alpha}$ in our method (3.3), ratio r in top-k (3.2) and β in (3.6). For synthetic datasets, we tune them on a validation set generated in the same way as TEST-ORIG. For MNIST-75SP, we use part of the training set. For COLLAB, PROTEINS and D&D , we tune them using 10-fold cross-validation on the training set.

Attention correctness. We evaluate attention correctness using area under the ROC curve (AUC) as an alternative to other methods, such as (Liu et al., 2017), which can be overoptimistic in some extreme cases, such as when all attention is concentrated in a single node or attention is uniformly spread over all nodes. AUC allows us to evaluate the ranking of α instead of their absolute

values. Compared to ranking metrics, such as rank correlation, AUC enables us to directly choose a pooling threshold $\tilde{\alpha}$ from the ROC curve by finding a desired balance between false-positives (pooling unimportant nodes) and false-negatives (dropping important nodes).

To evaluate attention correctness of models with global pooling, we follow the idea from convolutional neural networks (Zeiler and Fergus, 2014). After training a model, we remove node $i \in [1, N]$ and compute an absolute difference from prediction y for the original graph:

$$\alpha_i^{WS} = \frac{|y_i - y|}{\sum_{j=1}^{N} |y_j - y|}, \tag{3.7}$$

where y_i is a model's prediction for the graph without node i. While this method shows surprisingly high AUC in some tasks, it is not built-in in training and thus does not help to train a better model and only implicitly interprets a model's prediction (Figures 3.5 and 3.7). However, these results inspired us to design a weakly-supervised method described below.

3.3.4 Weakly-supervised attention supervision

Although for Colors, Triangles and MNIST-75sp we can define ground truth attention, so that it does not require manual labeling, in practice it is usually not the case and such annotations are hard to define and expensive, or even unclear how to produce. Based on results in Table 3.1, supervision of attention is necessary to reveal its power. Therefore, we propose a weakly-supervised approach, agnostic to the choice of a dataset and model, that does not require ground truth attention labels, but can improve a model's ability to generalize. Our approach is based on generating attention coefficients α_i^{WS} (3.7) and using them as labels to train our attention model with the loss defined in Eq 3.6. We apply this approach to Colors, Triangles and MNIST-75sp and observe peformance and robustness close to supervised models. We also apply it to Collab, Proteins and D&D, and in all cases we are able to improve results compared to unsupervised attention.

Training weakly-supervised models. Assume we want to train model **A** with "weak-sup" attention on a dataset without ground truth attention. We first need to train model **B** that has the same architecture as **A**, but does not have any attention/pooling between graph convolution layers. So, model **B** has only global pooling. After training **B** with the $\mathcal{L}_{MSE/CE}$ loss, we need to evaluate training graphs on **B** in the same way as during computation of α^{WS} in (3.7). In particular, for each training graph $\mathcal{G}$ with N nodes, we first make a prediction y for the entire $\mathcal{G}$. Then, for each $i \in [1, N]$, we remove node i from $\mathcal{G}$, and feed this reduced graph with $N - 1$ nodes to model **B** recording the model's prediction y_i. We then use (3.7) to compute α^{WS} based on y and y_i. Now, we can train **A** and use α^{WS} instead of ground truth α^{GT} in (3.6) to optimize both *MSE/CE* and *KL* losses.

3.4 Analysis of results

In this work, we aim to better understand attention and generalization in graph neural networks, and, based on our empirical findings, below we provide our analysis for the following questions.

How powerful is attention over nodes in GNNs? Our results on the COLORS, TRIANGLES and MNIST-75SP datasets suggest that the main strength of attention over nodes in GNNs is the ability to generalize to more complex or noisy graphs at test time. This ability essentially transforms a model that fails to generalize into a fairly robust one. Indeed, a classification accuracy gap for COLORS-LARGEC between the best model without supervised attention (GIN with global pooling) and a similar model with supervised attention (GIN, sup) is more than 60%. For TRIANGLES-LARGE this gap is 18% and for MNIST-75SP-NOISY it is more than 12%. This gap is even larger if compared to upper bound cases indicating that our supervised models can be further tuned and improved. Models with supervised or weakly-supervised attention also have a more narrow spread of results (Fig. 3.3).

What are the factors influencing performance of GNNs with attention? We identify three

Table 3.1: **Results on three tasks for different test subsets.** ± denotes standard deviation, not shown in case of small values (large values are explained in § 3.4). ATTN denotes attention accuracy in terms of AUC and is computed for the combined test set. The best result in each column (ignoring upper bound results) is bolded. denotes poor results with relatively low accuracy and/or high variance; denotes failed cases with accuracy close to random and/or extremely high variance. † For COLORS and MNIST-75SP, ChebyNets are used instead of ChebyGINs.

		COLORS				TRIANGLES			MNIST-75SP			
		ORIG	LARGE	LARGEC	ATTN	ORIG	LARGE	ATTN	ORIG	NOISY	NOISYC	ATTN
Global pool	GCN	97	72±15	20±3	99.6	46±1	23±1	79	78.3±2	38±4	36±4	72±2
	GIN	96±10	71±22	26±11	99.2	50±1	22±1	77	87.6±3	55±11	51±12	71±5
	ChebyGIN†	**100**	93±12	15±7	99.8	66±1	30±1	79	**97.4**	80±12	79±11	72±3
Unsuperv.	GIN, top-k	99.6	17±4	9±3	75±6	47±2	18±1	63±5	86±6	59±26	55±23	65±34
	GIN, ours	94±18	13±7	11±6	72±15	47±3	20±2	68±3	82.6±8	51±28	47±24	58±31
	ChebyGIN†, top-k	**100**	11±7	6±6	79±20	64±5	25±2	76±6	92.9±4	68±26	67±25	52±37
	ChebyGIN†, ours	80±30	16±10	11±6	67±31	67±3	26±2	77±4	94.6±3	80±23	77±22	78±31
Supervised	GIN, topk	87±1	39±18	28±8	**99.9**	49±1	20±1	88	90.5±1	85.5±2	79±5	99.3
	GIN, ours	**100**	**96±9**	**89±18**	99.8	49±1	22±1	76±1	90.9±0.4	85.0±1	80±3	99.3
	ChebyGIN†, topk	**100**	86±15	31±15	99.8	83±1	39±1	**97**	95.1±0.3	90.6±0.8	83±16	**100**
	ChebyGIN†, ours	**100**	94±8	75±17	99.8	**88±1**	**48±1**	96	95.4±0.2	**92.3±0.4**	**86±16**	**100**
Weak sup.	ChebyGIN†, ours	**100**	90±6	73±14	**99.9**	68±1	30±1	88	95.8±0.4	88.8±4	**86±9**	96.5±1
Upper bound	GIN	100	100	100	100	94±1	85±2	100	93.6±0.4	90.8±1	90.8±1	100
	ChebyGIN†	100	100	100	100	99.8	99.4±1	100	96.9±0.1	94.8±0.3	95.1±0.3	100

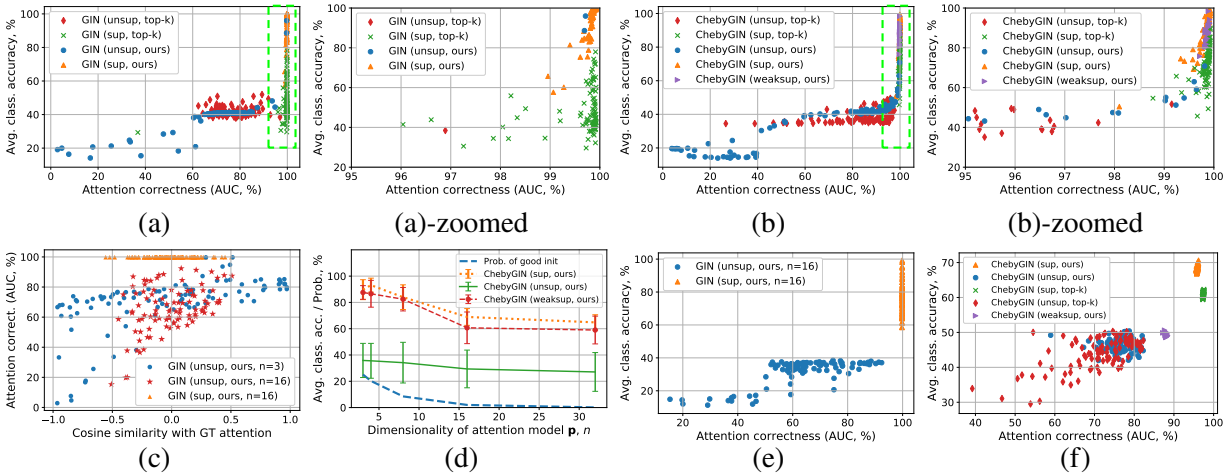


Figure 3.3: Disentangling factors influencing attention and classification accuracy for COLORS *(a-e)* and TRIANGLES *(f)*. Accuracies are computed over all test subsets. Notice the exponential growth of classification accuracy depending on attention correctness *(a,b)*, see zoomed plots *(a)*-zoomed, *(b)*-zoomed for cases when attention AUC>95%. *(d)* Probability of a good initialization is estimated as the proportion of cases when cosine similarity > 0.5; error bars indicate standard deviation. *(c-e)* show results using a higher dimensional attention model, $\mathbf{p} \in \mathbb{R}^n$.

key factors influencing performance of GNNs with attention: initialization of the attention model (i.e. vector $\mathbf{p}$ or GNN in (3.4)), strength of the main GNN model (i.e. the model that actually performs classification), and finally other hyperparameters of the attention and GNN models.

We highlight initialization as the critical factor. We ran 100 experiments on COLORS with random initializations (Fig. 3.3, *(a-e)*) of the vector $\mathbf{p}$ and measured how performance of both attention and classification is affected depending on how close (in terms of cosine similarity) the initialized $\mathbf{p}$ was to the optimal one, $\mathbf{p} = [0, 1, 0]$. We disentangle the dependency between the classification accuracy and cos. sim. into two functions to make the relationship clearer (Fig. 3.3, *(a, c)*). Interestingly, we found that classification accuracy depends *exponentially* on attention correctness and becomes close to 100% only when attention is also close to being perfect. In the case of slightly worse attention, even starting from 99%, classification accuracy drops significantly. This is an important finding that can also be valid for other more realistic applications. In the TRIANGLES task we only partially confirm this finding, because our attention models could not achieve AUC high enough to boost classification. However, by observing the upper bound results obtained by training with ground truth attention, we assume that this boost potentially should happen once attention becomes accurate enough.

Why is the variance of some results so high? In Table 3.1 we report high variance of results, which is mainly due to initialization of the attention model as explained above. This variance is also caused by initialization of other trainable parameters of a GNN, but we show that once the attention model is perfect, other parameters can recover from a bad initialization leading to better

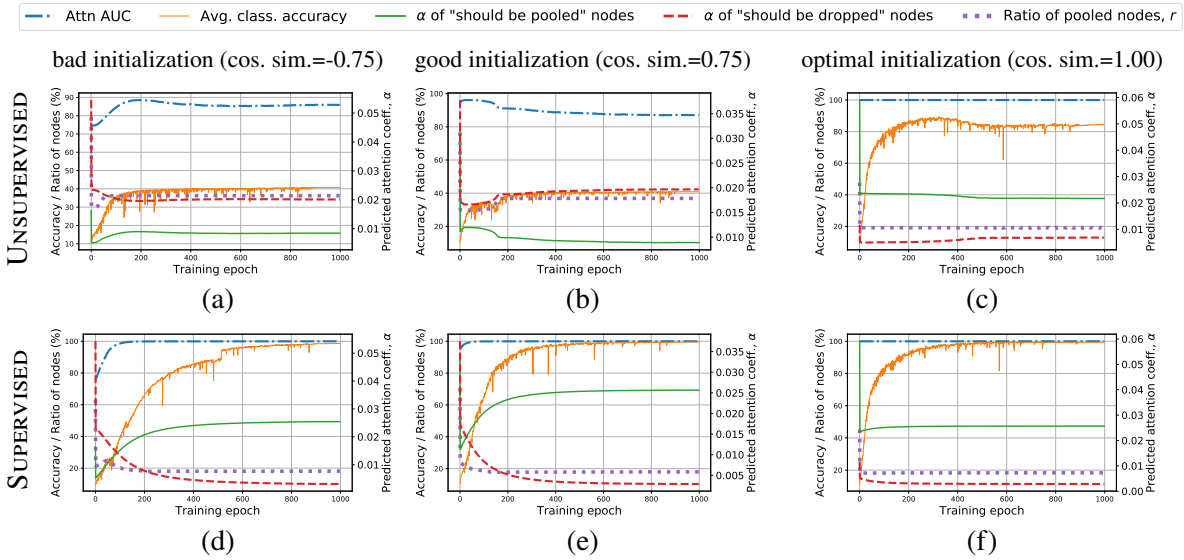


Figure 3.4: Influence of initialization on training dynamics for COLORS using GIN trained in unsupervised (a-c) and supervised (d-e) ways. The nodes that should be pooled according to our ground truth prior, must have larger attention values α. However, in the unsupervised cases, only the model with an optimal initialization (c) reaches a high accuracy, while other models (a,b) are stuck in a suboptimal state and wrong nodes are pooled, which degrades performance. In the supervised cases (d-f), models converge to a perfect accuracy and initialization only affects the speed of convergence. In these experiments, we train models longer to see if they can recover from a bad initialization.

results. The opposite, however, is not true: we never observed recovery of a model with poorly initialized attention (Fig. 3.4).

How top-k compares to our threshold-based pooling method? Our method to attend and pool nodes (3.3) is based on top-k pooling (Gao and Ji, 2019) and we show that the proposed threshold-based pooling is superior in a principle way. When we use supervised attention our results are better by more than 40% on COLORS-LARGEC, by 9% on TRIANGLES-LARGE and by 3% on MNIST-75SP. In Fig. 3.3 *((a,b)*-zoomed) we show that GIN and ChebyGIN models with supervised top-k pooling never reach an average accuracy of more than 80% as opposed to our method which reaches 100% in many cases.

How results change with increase of attention model input dimensionality or capacity? We performed experiments using ChebyGIN-h - a model with higher dimensionality of an input to the attention model. In such cases, it becomes very unlikely to initialize it in a way close to optimal (Fig. 3.3, *(c-e)*), and attention accuracy is concentrated in the 60-80% region. Effect of the attention model of such low accuracy is neglible or even harmful, especially on the large and noisy graphs. We also experimented with a deeper attention model (ChebyGIN-h), i.e. a 2 layer fully-connected layer with 32 hidden units for COLORS and MNIST-75SP, and a deeper GNN (3.4) for TRIANGLES. This has a positive effect overall, except for TRIANGLES, where our attention models were already deep GNNs.

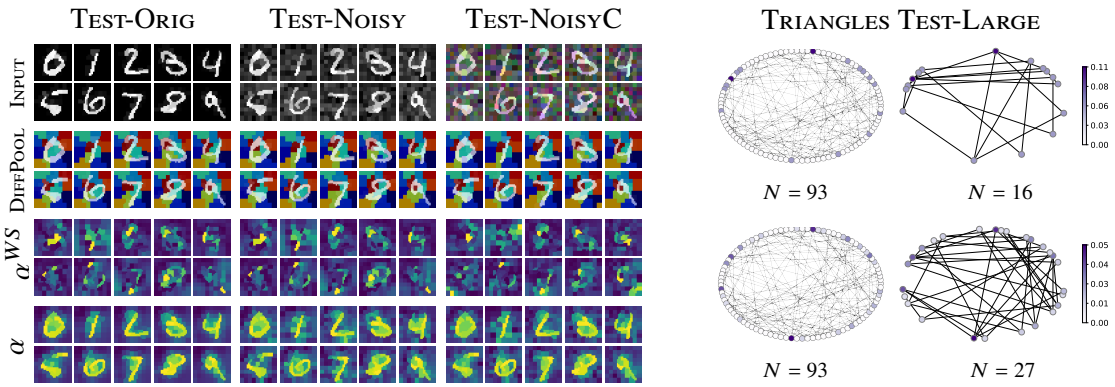


Figure 3.5: Qualitative analysis. For MNIST-75SP (on the left) we show examples of input test images (top row), results of DiffPool (Ying et al., 2018) (second row), attention weights α^{WS} generated using a model with global pooling based on (3.7) (third row), and α predicted by our weakly-supervised model (bottom row). Both our attention-based pooling and DiffPool can be strong and interpretable depending on the task, but in our tasks DiffPool was inferior. For TRIANGLES (on the right) we show an example of a test graph with $N = 93$ nodes with six triangles and the results of pooling based on ground truth attention weights α^{GT} (top row); in the bottom row we show attention weights predicted by our weakly-supervised model and results of our threshold-based pooling (3.3). Note that during training, our model has not encountered noisy images (MNIST-75SP) nor graphs larger than with $N = 25$ nodes (TRIANGLES).

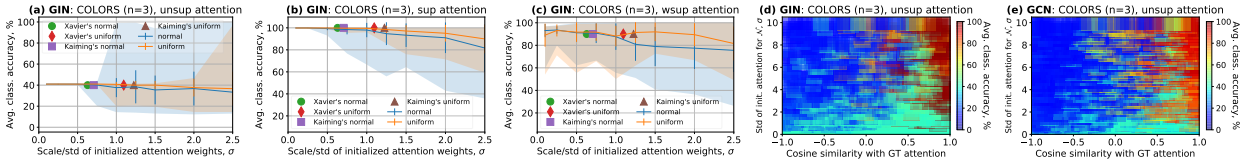


Figure 3.6: Influence of distribution parameters used to initialize the attention model $\mathbf{p}$ in the COLORS task with $n = 3$ dimensional features. We show points corresponding to the commonly used initialization strategies of Xavier (He et al., 2015) and Kaiming (He et al., 2015). *(a-c)* Shaded areas show range, bars show ±1 std.

Can we improve initialization of attention? In all our experiments, we initialize $\mathbf{p}$ from the Normal distribution, $\mathcal{N}(0, 1)$. To verify if the performance can be improved by choosing another distribution, we evaluate GIN and GCN models on a wide range of random distributions, Normal $\mathcal{N}(0, \sigma)$ and Uniform $U(-\sigma, \sigma)$, by varying scale σ (Fig. 3.6). We found out that for unsupervised training (Fig. 3.6, *(a)*), larger initial values and the Normal distribution should be used to make it possible to converge to an optimal solution, which is still unlikely and greatly depends on cosine similarity with GT attention (Fig. 3.6, *(d,e)*). For supervised and "weak-sup" attention, smaller initial weights and either the Normal or Uniform distribution should be used (Fig. 3.6, *(b,c)*).

What is the recipe for more powerful attention GNNs? We showed that GNNs with supervised training of attention are significantly more accurate and robust, although in case of a bad initialization it can take a long time to reach the performance of a better initialization. However, supervised attention is often infeasible. We suggested an alternative approach based on weakly-

Table 3.2: **Results on the social (COLLAB) and molecule (PROTEINS and D&D) datasets.** We use 3 layer GCNs (Kipf and Welling, 2017) or ChebyNets (Defferrard et al., 2016). Dataset subscripts denote the maximum number of nodes in the training set according to our splits (§ 3.3.1).

	COLLAB_{35}	PROTEINS_{25}	$\text{D\&D}_{200}$	$\text{D\&D}_{300}$
# train / test graphs	500 / 4500	500 / 613	462 / 716	500 / 678
# nodes (N) train	32-35	4-25	30-200	30-300
# nodes (N) test	32-492	6-620	201-5748	30-5748
Global max	65.9±3.4	74.4±1.0	29.7±4.9	72.7±3.6
Unsup, ours	65.7±3.5	75.6±1.4	51.9±5.3	77.2±2.9
Weak-sup	**67.0±1.7**	**76.2±0.7**	**54.3±5.0**	**78.4±1.1**

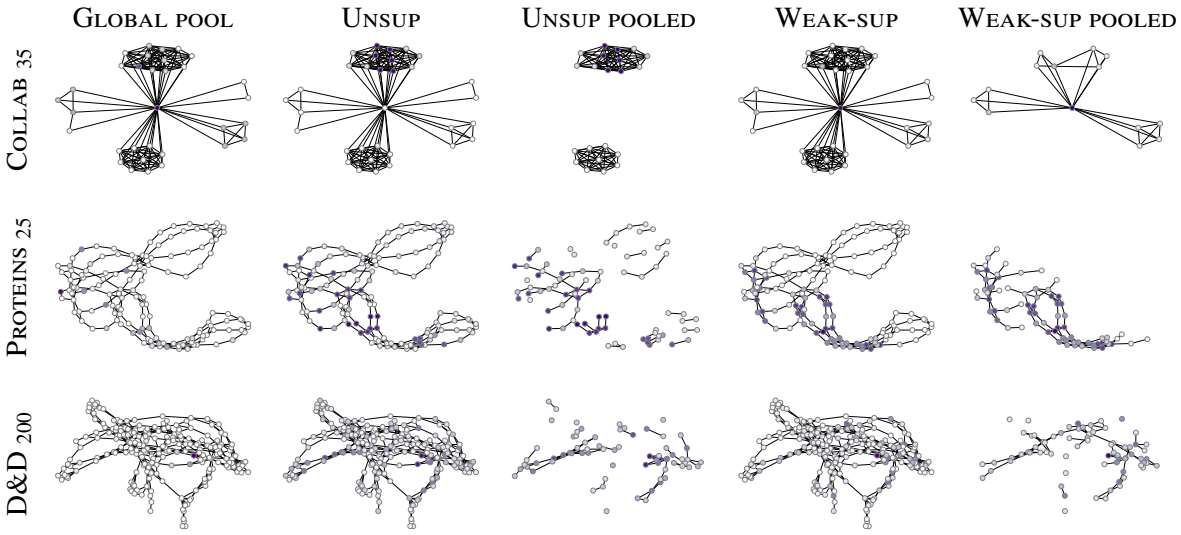


Figure 3.7: Qualitative results. In COLLAB, a graph represents an ego-network of a researcher, therefore *center nodes* are important. In PROTEINS and D&D, a graph is a protein and nodes are amino acids, so it is important to attend to a *connected chain* of amino acids to distinguish an enzyme from a non-enzyme protein. Our weakly-supervised method attends to and pools more relevant nodes compared to global and unsupervised models, leading to better classification results.

supervised training and validated it on our synthetic (Table 3.1) and real (Table 3.2) datasets. In case of COLORS, TRIANGLES and MNIST-75SP we can compare to both unsupervised and supervised models and conclude that our approach shows performance, robustness and relatively low variation (i.e. sensitivity to initialization) similar to supervised models and much better than unsupervised models. In case of COLLAB, PROTEINS and D&D we can only compare to unsupervised and global pooling models and confirm that our method can be effectively employed for a wide diversity of graph classification tasks and attends to more relevant nodes (Figures 3.5 and 3.7). Tuning the distribution and scale σ for the initialization of attention can further improve results. For instance, on PROTEINS for the weakly-supervised case, we obtain 76.4% as opposed to 76.2%.

3.5 Conclusion

We have shown that learned attention can be extremely powerful in graph neural networks, but only if it is close to optimal. This is difficult to achieve due to the sensitivity of initialization, especially in the unsupervised setting where we do not have access to ground truth attention. Thus, we have identified initialization of attention models for high dimensional inputs as an important open issue. We also show that attention can make GNNs more robust to larger and noisy graphs, and that the weakly-supervised approach proposed in our work brings advantages similar to the ones of supervised models, yet at the same time can be effectively applied to datasets without annotated attention.

4 Graph Density-Aware Losses for Novel Compositions in Scene Graph Generation

Prologue

Context. Visual recognition models have been greatly improved across different applications largely due to the advances in convolutional neural networks (CNNs). Scene graph generation (SGG) is a task where objects and the relationships between them must be recognized from images (Xu et al., 2017). Many state of the art SGG models have tuned their performance for frequent compositions of objects and relationships (Zellers et al., 2018). Unfortunately, there is an extreme reduction in performance of these models on rare or unseen compositions (Tang et al., 2020). Understanding and addressing this problem in a realistic large-scale setup is challenging yet needed to increase the practical value of SGG models.

Contributions. We analyzed the loss function typically used to train SGG models and found that it ignores an important property of visual scene graphs — graph density (number of edges *w.r.t.* the number of nodes). To address this issue, we developed a loss with terms that are normalized according to the graph density. The SGG models trained with our graph-density normalized loss significantly improve in generalization to rare and unseen compositions.

Recent works. Tang et al. (2020) proposed a method based on causal inference to "debias" SGG models and improve generalization. In line with our work, Suhail et al. (2021) proposed an improved loss function based on energy defined on graphs. Khandelwal et al. (2021) added more precise object representation in the form of segmentation masks that also improved compositional generalization. Liu et al. (2021) proposed relation affinity fields to better generalize to unseen/rare compositions. While these approaches are applied to the SGG task, object-centric learning is a more general-purpose method that can potentially improve generalization in a larger variety of visual tasks (Locatello et al., 2020; Dittadi et al., 2021).

4.1 Introduction

In recent years, there has been growing interest to connect successes in visual perception with language and reasoning (Su et al., 2019; Zhou et al., 2019). This requires us to design systems that can not only recognize objects, but understand and reason about the relationships between them. This is essential for such tasks as visual question answering (VQA) (Antol et al., 2015; Hudson and Manning, 2019b; Cangea et al., 2019) or caption generation (Yang et al., 2019; Gu et al., 2019a). However, predicting a high-level semantic output (*e.g.* answer) from a low-level

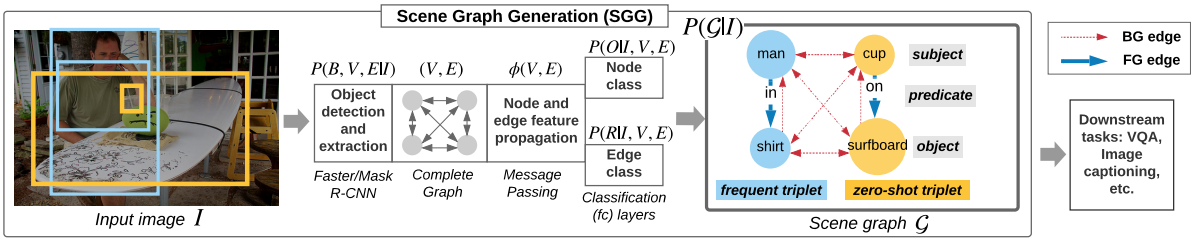
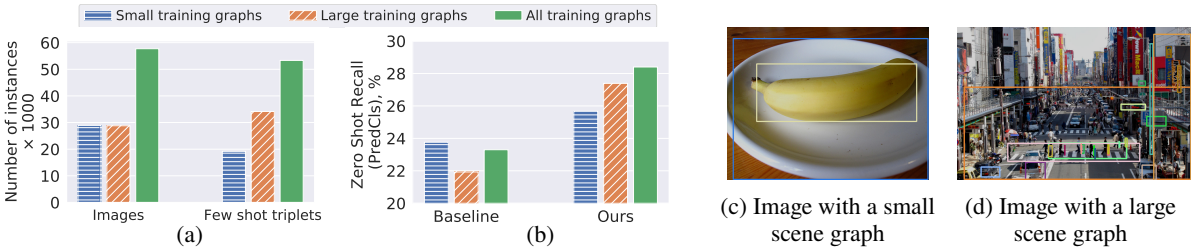


Figure 4.1: In this work, we improve scene graph generation $P(\mathcal{G}|I)$. In many downstream tasks, such as VQA, the result directly depends on the accuracy of predicted scene graphs.

visual signal (*e.g.* image) is challenging due to a vast gap between the modalities. To bridge this gap, it would be useful to have some intermediate representation that can be relatively easily generated by the low-level module and, at the same time, can be effectively used by the high-level reasoning module. We want this representation to semantically describe the visual scene in terms of objects and relationships between them, which leads us to a structured image representation, the **scene graph** (SG) (Johnson et al., 2015; Krishna et al., 2017b). A scene graph is a collection of visual relationship *triplets*: <*subject*, *predicate*, *object*> (*e.g.* <cup, on, table>). Each node in the graph corresponds to a subject or object (with a specific image location) and edges to predicates (Fig. 4.1). Besides bridging the gap, SGs can be used to verify how well the model has understood the visual world, as opposed to just exploiting one of the biases in a dataset (Jabri et al., 2016; Anand et al., 2018; Bahdanau et al., 2018). Alternative directions to SGs include, for example, attention (Norcliffe-Brown et al., 2018) and neural-symbolic models (Vedantam et al., 2019).



(c) Image with a small scene graph

(d) Image with a large scene graph

Figure 4.2: **Motivation of our work.** We split the training set of Visual Genome (Krishna et al., 2017b) into two subsets: those with relatively small ($\leq$ 10 nodes) and large ($>$ 10 nodes) graphs. (**a**) While each subset contains a similar number of images (the three left bars), larger graphs contain more few-shot labels (the three right bars). (**b**) Baseline methods ((Xu et al., 2017) in this case) fail to learn from larger graphs due to their loss function. However, training on large graphs and corresponding few-shot labels is important for stronger generalization. We address this limitation and significantly improve results on zero and few-shots. (**c, d**) Small and large scene graphs typically describe simple and complex scenes respectively.

Scene graph generation (SGG) is the task of predicting a SG given an input image. The inferred SG can be used directly for downstream tasks such as VQA (Zhang et al., 2019a; Hudson and Manning, 2019a), image captioning (Yang et al., 2019; Gu et al., 2019a) or retrieval (Johnson

et al., 2015; Belilovsky et al., 2017; Tang et al., 2020). A model which performs well on SGG should demonstrate the ability to ground visual concepts to images and generalize to compositions of objects and predicates in new contexts. In real world images, some compositions (*e.g.* <cup, on, table>) appear more frequently than others (*e.g.* <cup, on, *surfboard*> or <cup, *under*, table>), which creates a strong frequency bias. This makes it particularly challenging for models to generalize to novel (zero-shot) and rare (few-shot) compositions, even though each of the subjects, objects and predicates have been observed at training time. The problem is exacerbated by the test set and evaluation metrics, which do not penalize models that blindly rely on such bias. Indeed, Zellers et al. (2018) has pointed out that SGG models largely exploit simple co-occurrence information. In fact, the performance of models predicting solely based on frequency (i.e. a cup is most likely to be *on* a *table*) is not far from the state-of-the-art using common metrics (see FREQ in Table 4.1).

In this work, we reveal that (a) the frequency bias exploited by certain models leads to poor generalization on few-shot and zero-shot compositions; (b) existing models disproportionately penalize large graphs, even if these often contain many of the infrequent visual relationships, which leads to performance degradation on few and zero-shot cases (Fig. 4.2). We address these challenges and show that our suggested improvements can provide benefits for two strong baseline models (Xu et al., 2017; Zellers et al., 2018). Overall, we make the following four **contributions**:

1. **Improved loss**: we introduce a density-normalized edge loss, which improves results on all metrics, especially for few and zero-shots (§ 4.3.2);
2. **Novel weighted metric**: we illustrate several issues in the evaluation of few and zero-shots, proposing a novel weighted metric which can better track the performance of this critical desiderata (§ 4.3.3);
3. **Frequency bias**: we demonstrate a negative effect of the frequency bias, proposed in Neural Motifs (Zellers et al., 2018), on few and zero-shot performance (§ 4.4);
4. **Scaling to GQA**: in addition to evaluating on Visual Genome (VG) (Krishna et al., 2017b), we confirm the usefulness of our loss and metrics on GQA (Hudson and Manning, 2019b) – an improved version of VG. GQA has not been used to evaluate SGG models before and is interesting to study, because compared to VG its scene graphs are cleaner, larger, more dense and contain a larger variety of objects and predicates (§ 4.4).

4.2 Related work

Zero-shot learning. In vision tasks, such as image classification, zero-shot learning has been extensively studied, and the main approaches are based on attributes (Lampert et al., 2013) and

semantic embeddings (Frome et al., 2013; Xian et al., 2016). The first approach is related to the zero-shot problem we address in this work: it assumes that all individual attributes of objects (color, shape, etc.) are observed during training, such that novel classes can be detected at test time based on *compositions* of their attributes. Zero-shot learning in scene graphs is similar: all individual subjects, objects and predicates are observed during training, but most of their compositions are not. This task was first evaluated in (Lu et al., 2016) on the VRD dataset using a joint vision-language model. Several follow-up works attempted to improve upon it: by learning a translation operator in the embedding space (Zhang et al., 2017), clustering in a weakly-supervised fashion (Peyre et al., 2017), using conditional random fields (Cong et al., 2018) or optimizing a cycle-consistency loss to learn object-agnostic features (Yang et al., 2018b). Augmentation using generative networks to generate more examples of rare cases is another promising approach (Wang et al., 2019b); but, it was only evaluated in the predicate classification task. In our work, we also consider subject/object classification to enable the classification of the whole triplets, making the "image to scene graph" pipeline complete. Most recently, Tang et al. (2020) proposed learning causal graphs and showed strong performance in zero-shot cases.

While these works improve generalization, none of them has identified the challenges and importance of learning from large graphs for generalization. By concentrating the model's capacity on smaller graphs and neglecting larger graphs, baseline models limit the variability of training data, useful for stronger generalization (Hill et al., 2019). Our loss enables this learning, increasing the effective data variability. Moreover, previous gains typically incur a large computational cost, while our loss has negligible cost and can be easily added to other models.

Few-shot predicates. Several recent works have addressed the problem of imbalanced and few-shot predicate classes (Chen et al., 2019a; Dornadula et al., 2019; Tang et al., 2019; Zhang et al., 2019e; Tang et al., 2020; Chen et al., 2019b). However, compared to our work, these works have not considered the imbalance between foreground and background edges, which is more severe than other predicate classes (Fig. 4.3) and is important to be fixed as we show in this work. Moreover, we argue that the compositional generalization, not addressed in those works, can be more difficult than generalization to rare predicates. For example, the triplet <cup, on, surfboard> is challenging to be predicted correctly as a whole; even though 'on' can be the most frequent predicate, it has never been observed together with 'cup' and 'surfboard'. Experimental results in previous work (Lu et al., 2016; Zhang et al., 2017; Yang et al., 2018b; Wang et al., 2019b; Tang et al., 2020) highlight this difficulty. Throughout this work, by "few-shot" we assume triplets, not predicates.

"Unbiasing" methods. Our idea is similar to the Focal loss (Lin et al., 2017), which addresses the imbalance between foreground and background objects in the object detection task. However,

directly applying the focal loss to Visual Genome is challenging, due to the large amount of missing and mislabeled examples in the dataset. In this case, concentrating the model's capacity on "hard" examples can be equivalent to putting more weight on noise, which can hurt performance. Tang et al. (2020) compared the focal loss and other unbiasing methods, such as upsampling and upweighting, and did not report significantly better results.

4.3 Methods

In this section, we will review a standard loss used to train scene graph generation models (§ 4.3.1) and describe our improved loss (§ 4.3.2). We will then discuss issues with evaluating rarer combinations and propose a new weighted metric (§ 4.3.3).

4.3.1 Overview of scene graph generation

In scene graph generation, given an image I, we aim to output a scene graph $\mathcal{G} = (O, R)$ consisting of a set of subjects and objects (O) as nodes and a set of relationships or predicates (R) between them as edges (Fig. 4.1). So the task is to maximize the probability $P(\mathcal{G}|I)$, which can be expressed as $P(O, R|I) = P(O|I)\ P(R|I, O)$. Except for works that directly learn from pixels (Newell and Deng, 2017), the task is commonly (Xu et al., 2017; Yang et al., 2018a; Zellers et al., 2018) reformulated by first detecting bounding boxes B and extracting corresponding object and edge features, $V = f(I, B)$ and $E = g(I, B)$ respectively, using some functions f, g (*e.g.* a ConvNet followed by ROI Align (He et al., 2017)):

$$P(\mathcal{G}|I) = P(V, E|I)\ P(O, R|V, E, I). \tag{4.1}$$

The advantage of this approach is that solving $P(O, R|V, E, I)$ is easier than solving $P(O, R|I)$. At the same time, to compute $P(V, E|I)$ we can use pretrained object detectors (Ren et al., 2015; He et al., 2017). Therefore, we follow (Xu et al., 2017; Yang et al., 2018a; Zellers et al., 2018) and use this approach to scene graph generation.

In practice, we can assume that the pretrained object detector is fixed or that ground truth bounding boxes B are available, so we can assume $P(V, E|I)$ is constant. In addition, following (Lu et al., 2016; Xu et al., 2017; Yang et al., 2018a), we can assume conditional independence of variables O and R: $P(O, R|V, E, I) = P(O|V, E, I)P(R|V, E, I)$. We thus obtain the scene graph generation loss:

$$-\log P(\mathcal{G}|I) = -\log P(O|V, E, I) - \log P(R|V, E, I). \tag{4.2}$$

Some models (Zellers et al., 2018; Zhang et al., 2019e) do not assume the conditional independence of O and R, making predicates explicitly depend on subject and object labels:

$P(O,R|V,E,I) = P(O|V,E,I)P(R|O,V,E,I)$. However, such a model must be carefully regularized, since it can start to ignore (V,E,I) and mainly rely on the frequency distribution $P(R|O)$ as a stronger signal. For example, the model can learn that between 'cup' and 'table' the relationship is most likely to be 'on', regardless the visual signal. As we show, this can hurt generalization.

(4.2) is commonly handled as a multitask classification problem, where each task is optimized by the cross-entropy loss $\mathcal{L}$. In particular, given a batch of scene graphs with N nodes and M edges in total, the loss is the following:

$$\mathcal{L} = \mathcal{L}_{node} + \mathcal{L}_{edge} = \frac{1}{N}\sum_{i}^{N}\mathcal{L}_{obj,i} + \frac{1}{M}\sum_{ij}^{M}\mathcal{L}_{rel,ij}. \quad (4.3)$$

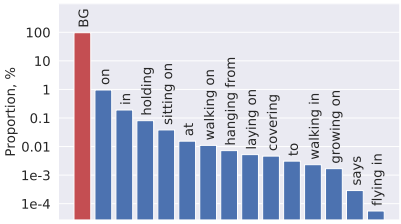


Figure 4.3: Predicate distribution in Visual Genome (split (Xu et al., 2017)). BG edges (note the log scale) dominate, with >96% of all edges, creating an extreme imbalance. For clarity, only some most and least frequent predicate classes are shown.

Node and edge features (V,E) output by the detector form a complete graph without self-loops (Fig. 4.1). So, conventionally (Xu et al., 2017; Yang et al., 2018a; Zellers et al., 2018), the loss is applied to all edges: $M \approx N^2$. These edges can be divided into foreground (FG), corresponding to annotated edges, and background (BG), corresponding to not annotated edges: $M = M_{FG} + M_{BG}$. The BG edge type is similar to a "negative" class in the object detection task and has a similar purpose. Without training on BG edges, at test time the model would label all pairs of nodes as "positive", i.e. having some relationship, when often it is not the case (at least, given the vocabulary in the datasets). Therefore, not using the BG type can hurt the quality of predicted scene graphs and can lower recall.

4.3.2 Hyperparameter-free normalization of the edge loss

Baseline loss as a function of graph density. In scene graph datasets such as Visual Genome, the number of BG edges is greater than FG ones (Fig. 4.3), yet the baseline loss (4.3) does not explicitly differentiate between BG and other edges. If we assume a fixed probability for two objects to have a relationship, then as the number of nodes grows we can expect fewer of them to have a relationship. Thus the graph density can vary based on the number of nodes (Fig. 4.4), a fact not taken into account in (4.3). To avoid this, we start by decoupling the edge term of (4.3) into the foreground (FG) and background (BG) terms:

$$\mathcal{L}_{edge} = \frac{1}{M}\sum_{ij}^{M}\mathcal{L}_{rel,ij} = \frac{1}{M_{FG}+M_{BG}}\left[\underbrace{\sum_{ij\in\mathcal{E}}^{M_{FG}}\mathcal{L}_{rel,ij}}_{\text{FG edges}} + \underbrace{\sum_{ij\notin\mathcal{E}}^{M_{BG}}\mathcal{L}_{rel,ij}}_{\text{BG edges}}\right], \quad (4.4)$$

where $\mathcal{E}$ is a set of FG edges, M_{FG} is the number of FG edges ($|\mathcal{E}|$) and M_{BG} is the number of BG edges. Next, we denote FG and BG edge losses averaged per batch as $\mathcal{L}_{FG} = 1/M_{FG} \sum_{ij \in \mathcal{E}}^{M_{FG}} \mathcal{L}_{rel,ij}$ and $\mathcal{L}_{BG} = 1/M_{BG} \sum_{ij \notin \mathcal{E}}^{M_{BG}} \mathcal{L}_{rel,ij}$, respectively. Then, using the definition of graph density as a proportion of FG edges to all edges, $d = M_{FG}/(M_{BG} + M_{FG})$, we can express the total baseline loss equivalent to (4.3) **as a function of graph density**:

$$\boxed{\mathcal{L} = \mathcal{L}_{node} + d\mathcal{L}_{FG} + (1-d)\mathcal{L}_{BG}.} \tag{4.5}$$

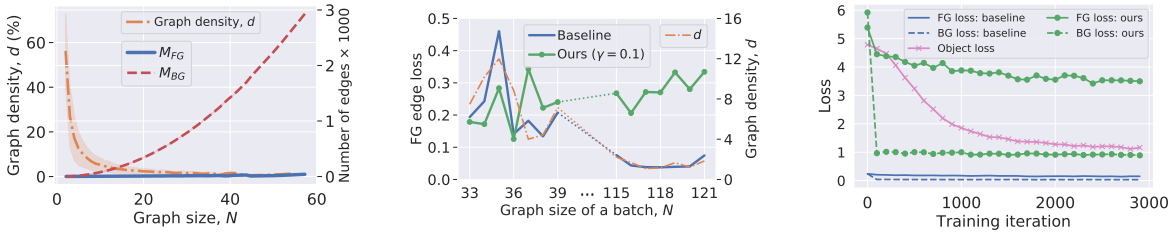
Density-normalized edge loss. (4.5) and Fig. 4.4 allow us to notice two issues:

1. **Discrepancy of the loss between graphs of different sizes.** Since d exponentially decreases with graph size (4.4, left), FG edges of larger graphs are weighted less than edges of smaller graphs in the loss (Fig. 4.4, middle), making the model neglect larger graphs.
2. **Discrepancy between object and edge losses.** Due to d tending to be small on average, L_{edge} is much smaller than L_{node}, so the model might focus mainly on L_{node} (Fig. 4.4, right).

Both issues can be addressed by normalizing FG and BG terms by graph density d:

$$\boxed{\mathcal{L} = \mathcal{L}_{node} + \gamma\left[\mathcal{L}_{FG} + M_{BG}/M_{FG}\mathcal{L}_{BG}\right].} \tag{4.6}$$

where $\gamma = 1$ in our default hyperparameter-free variant and $\gamma \neq 1$ only to empirically analyze the loss (Table 4.4). Even though the BG term still depends on graph density, we found it to be less sensitive to variations in d, since the BG loss quickly converges to some stable value, performing a role of regularization (Fig. 4.4, right). We examine this in detail in § 4.4.



$d \to 0$ for large graphs Large graphs are downweighted by d $L_{edge} \ll L_{node}$

Figure 4.4: (**left**) The number of FG edges grows much more slowly with graph size than the number of BG edges (on VG: $M_{FG} \approx 0.5N$). This leads to: (**middle**) Downweighting of the FG loss on larger graphs and effectively limiting the amount and variability of training data, since large graphs contain a lot of labeled data (Fig. 4.2); here, losses of converged models for batches sorted by graph size are shown. (**right**) Downweighting of the edge loss L_{edge} overall compared to L_{node}, even though both tasks are equally important to correctly predicting a scene graph. Our normalization fixes both issues.

4.3.3 Weighted triplet recall

The common evaluation metric for scene graph prediction is image-level Recall@K or R@K (Xu et al., 2017; Yang et al., 2018a; Zellers et al., 2018). To compute it, we first need to extract the top-K triplets, Top_K, from the *entire image* based on ranked predictions of a model . Given a set of ground truth triplets, GT, the image-level R@K is computed as (see Fig. 4.5 for a visualization):

$$\text{R@K} = |\text{Top}_K \cap \text{GT}|/|\text{GT}|. \tag{4.7}$$

There are four issues with this metric:

(a) The frequency bias of triplets means more frequent triplets will dominate the metric.

(b) The denominator in (4.7) creates discrepancies between images with different |GT| (the number of ground truth triplets in an image), especially pronounced in few/zero shots.

(c) Evaluation of zero ($n = 0$) and different few-shot cases $n = 1, 5, 10, \ldots$ (Wang et al., 2019b) leads to many R@K results (Wang et al., 2019b). This complicates the analysis. Instead, we want a single metric for all n.

(d) Two ways of computing the image-level recall (Newell and Deng, 2017; Zellers et al., 2018), graph *constrained* and *unconstrained*, lead to very different results and complicate the comparison (Fig. 4.5).

To address issue **(a)**, the predicate-normalized metric, mean recall (mR@K) (Chen et al., 2019a; Tang et al., 2019) and weighted mR@K were introduced (Zhang et al., 2019e). These metrics, however, only address the imbalance of predicate classes, not whole triplets. Early work (Lu et al., 2016; Dai et al., 2017) used triplet-level Recall@K (or R_{tr}@K) for some tasks (*e.g.* predicate detection), which is based on ranking predicted triplets for each ground truth subject-object pair independently; the pairs without relationships are not evaluated. Hence, R_{tr}@K is similar to top-K accuracy. This metric avoids issues **(b)** and **(d)**, but the issues of the frequency bias **(a)** and unseen/rare cases **(c)** still remain. To alleviate these, we adapt this metric to better track unseen and rare cases. We call our novel metric Weighted Triplet Recall wR_{tr}@K, which computes a recall at each triplet and reweights the average result based on the frequency of the GT triplet in the training set:

$$\text{wR}_{tr}\text{@K} = \sum_t^T w_t[\text{rank}_t \leq K], \tag{4.8}$$

where T is the number of all test triplets, $[\cdot]$ is the Iverson bracket, $w_t = \frac{1}{(n_t+1)\sum_t 1/(n_t+1)} \in [0, 1]$ and n_t is the number of occurrences of triplet t in the training set; $n_t + 1$ is used to handle zero-shot triplets; $\sum_t w_t = 1$. Since wR_{tr}@K is still a triplet-level metric, we avoid issues **(b)** and

(d). Our metric is also robust to the frequency-bias **(a)**, since frequent triplets (with high n_t) are downweighted proportionally, which we confirm by evaluating the FREQ model from (Zellers et al., 2018). Finally, a single wR_{tr}@K value shows zero and few-shot performance linearly aggregated for all $n \geq 0$, solving issue **(c)**.

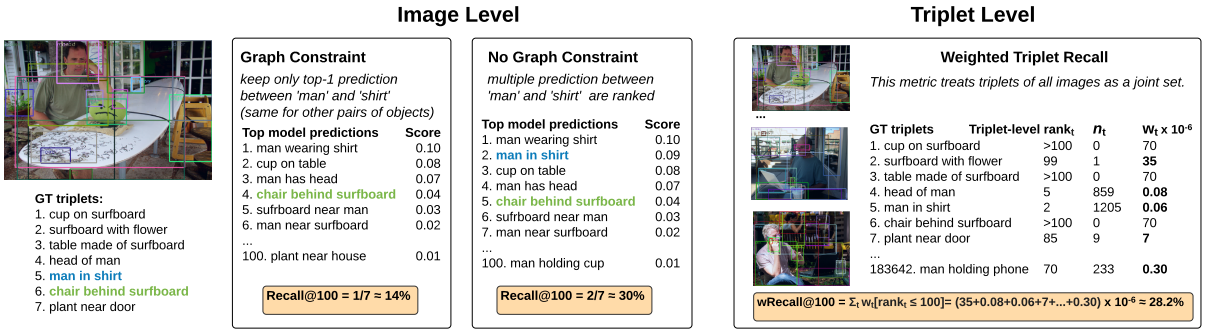


Figure 4.5: Existing image-level recall metrics *versus* our proposed weighted triplet recall. We first make unweighted predictions $\text{rank}_t \leq K$ for all GT triplets in all test images, then reweight them according to the frequency distribution (4.8). Computing our metric per image would be noisy.

4.4 Experiments

Datasets. We evaluate our loss and metric on Visual Genome (Krishna et al., 2017b). Since it is a noisy dataset, several "clean" variants were introduced. We mainly experiment with the most common variant (VG) (Xu et al., 2017), which consists of the 150 most frequent object and 50 predicates classes. An alternative variant (VTE) (Zhang et al., 2017) has been often used for zero-shot evaluation. Surprisingly, we found that the VG split (Xu et al., 2017) is better suited for this task, given a larger variability of zero-shot triplets in the test set Recently, GQA (Hudson and Manning, 2019b) was introduced, where scene graphs were cleaned to automatically construct question answer pairs. GQA has more object and predicate classes, so that zero and few-shot triplets are more likely to occur at test time. To the best of our knowledge, scene graph generation (SGG) results have not been reported on GQA before, even though some VQA models have relied on SGG (Hudson and Manning, 2019a).

Training and evaluation details. We experiment with two models: Message Passing (MP) (Xu et al., 2017) and Neural Motifs (NM) (Zellers et al., 2018). We use publicly available implementations of MP and NM[1], with all architecture details and hyperparameters kept the same. To be consistent with baseline models, for Visual Genome we use Faster R-CNN (Ren et al., 2015) with VGG16 as a backbone to extract node and edge features. For GQA we choose a more

[1] https://github.com/rowanz/neural-motifs

recent Mask R-CNN (He et al., 2017) with ResNet-50-FPN as a backbone pretrained on COCO. We also use this detector on the VTE split. We perform more experiments with Message Passing, since our experiments revealed that it better generalizes to zero and few-shot cases, while performing only slightly worse on other metrics. In addition, it is a relatively simple model, which makes the analysis of its performance easier. We evaluate models on three tasks, according to (Xu et al., 2017): 1) predicate classification (**PredCls**), in which the model only needs to label a predicate given ground truth object labels and bounding boxes, i.e. $P(R|I,B,O)$; 2) scene graph classification (**SGCls**), in which the model must also label objects, i.e. $P(O,R|I,B)$; 3) scene graph generation **SGGen** (sometimes denoted as **SGDet**), $P(\mathcal{G}|I)$, which includes detecting bounding boxes first (reported separately in Tables 4.5, 4.6).

4.4.1 Results

Table 4.1 shows our main results, where for each task we report five metrics: image-level recall on all triplets (R@K) and zero-shot triplets (R_{ZS}@K), triplet-level recall (R_{tr}@K) and our weighted triplet recall (wR_{tr}@K), and mean recall (mR@K). We compute recalls without the graph constraint since this is a more accurate metric. We denote graph-constrained results as **PredCls-GC**, **SGCls-GC**, **SGGen-GC** and report them only in Tables 4.2, 4.3 and Table 4.5.

VG results. We can observe that both Message Passing (MP) and Neural Motifs (NM) greatly benefit from our density-normalized loss on all reported metrics. Larger gaps are achieved on metrics evaluating zero and few-shots. For example, in PredCls on Visual Genome, MP with our loss is 22% better (in relative terms) on zero-shots, while NM with our loss is 50% better. The gains arising from other zero-shot and weighted metrics are also significant.

GQA results. On GQA, our loss also consistently improves results, especially in PredCls. However, the gap is lower compared to VG. There are two reasons for this: 1) scene graphs in GQA are much denser , i.e. the imbalance between FG and BG edges is less pronounced, which means that in the baseline loss the edge term is not diminished to the extent it is in VG; and 2) the training set of GQA is more diverse than VG (with 15 times more labeled triplets), which makes the baseline model generalize well on zero and few-shots. We confirm these arguments by training and evaluating on our version of GQA: **GQA-nLR** with left and right predicate classes excluded making scene graph properties, in particular sparsity, more similar to those of VG.

Effect of the Frequency Bias (FREQ) on Zero and Few-Shot Performance. The FREQ model (Zellers et al., 2018) simply predicts the most frequent predicate between a subject and an object, $P(R|O)$. Its effect on few-shot generalization has not been empirically studied before. We study this by

Table 4.1: Results on Visual Genome (split (Xu et al., 2017)) and GQA (Hudson and Manning, 2019b). We obtain particularly strong results in columns R_{ZS}, wR_{tr} and mR in each of the two tasks. ▒ denotes cases with ≥ 15% relative difference between the baseline and our result; ▒ denotes a difference of ≥ 50%. Best results for each dataset (VG, GQA and GQA-nLR) are bolded. GQA-nLR: our version of GQA with left/right spatial relationships excluded, where scene graphs become much sparser . ⋆Results are provided for the reference and evaluating our loss with these methods is left for future work. †The correctness of this evaluation is discussed in (Tang, 2020). References: [1](Zellers et al., 2018), [2](Xu et al., 2017), [3](Chen et al., 2019a), [4](Zhang et al., 2019e).

Dataset	Model	Loss	Scene Graph Classification					Predicate Classification				
			R@100	R_{ZS}@100	R_{tr}@20	wR_{tr}@20	mR@100	R@50	R_{ZS}@50	R_{tr}@5	wR_{tr}@5	mR@50
Visual Genome	FREQ[1]	–	45.4	0.5	51.7	18.3	19.1	69.8	0.3	89.8	31.0	22.1
	MP[1,2]	BASELINE (4.3)	47.2	8.2	51.9	26.2	17.3	74.8	23.3	86.6	51.3	20.6
		OURS (4.6)	48.6	**9.1**	**52.6**	**28.2**	**26.5**	78.2	**28.4**	89.4	58.4	32.1
	NM[1]	BASELINE (4.3)	48.1	5.7	51.9	26.5	20.4	80.5	11.1	91.0	51.8	26.9
		OURS (4.6)	48.4	7.1	52.0	27.7	25.5	82.0	16.7	92.0	56.4	34.8
		OURS (4.6), NO FREQ	48.4	8.9	51.8	28.0	26.1	82.5	26.6	**92.4**	**60.3**	35.8
	KERN⋆[3]	BASELINE (4.3)	49.0	3.7	**52.6**	27.7	26.2	81.9	5.8	91.9	49.1	**36.3**
	RelDN⋆[4]	BASELINE (4.3)	**50.8**†	–	–	–	–	**93.7**†	–	–	–	–
GQA	MP[1,2]	BASELINE (4.3)	27.1	2.8	31.9	**8.9**	1.6	59.7	34.9	96.4	88.4	1.8
		OURS (4.6)	**27.6**	**3.0**	**32.2**	**8.9**	**2.8**	**61.0**	**37.2**	**96.9**	**89.5**	**2.9**
GQA -nLR	MP[1,2]	BASELINE (4.3)	24.9	3.0	**30.2**	12.4	2.8	58.1	21.7	71.6	47.0	4.6
		OURS (4.6)	**25.0**	**3.2**	29.4	**12.6**	**7.0**	**62.4**	**26.2**	**77.9**	**55.0**	**12.1**

adding/ablating FREQ from baseline MP and NM on Visual Genome (Fig. 4.6, left). Our results show that FREQ only marginally improves results on unweighted metrics. At the same time, perhaps unsurprisingly, it leads to severe drops in zero-shot and weighted metrics, especially in NM. For example, by ablating FREQ from NM, we improve PredCls-R_{ZS}@50 from 11% to 25%. This highlights that the existing metrics are a poor choice to measure the effectiveness of a model.

Why does loss normalization help more on few and zero-shots? The baseline loss effectively ignores edge labels of large graphs, because it is scaled by a small d in those cases (Fig. 4.4). To validate that, we split the training set of Visual Genome in two subsets, with a comparable number of images in each: with relatively small and large graphs evaluating on the original test set in both cases. We observe that the baseline model does not learn well from large graphs, while our loss enables this learning (Fig. 4.6, right). Moreover, when trained on small graphs only, the baseline is even better in PredCls than when trained on all graphs. This is because in the latter case, large graphs, when present in a batch, make the whole batch more sparse, downweighting the edge loss of small graphs as well. At the same time, larger graphs predictably contain more labels,

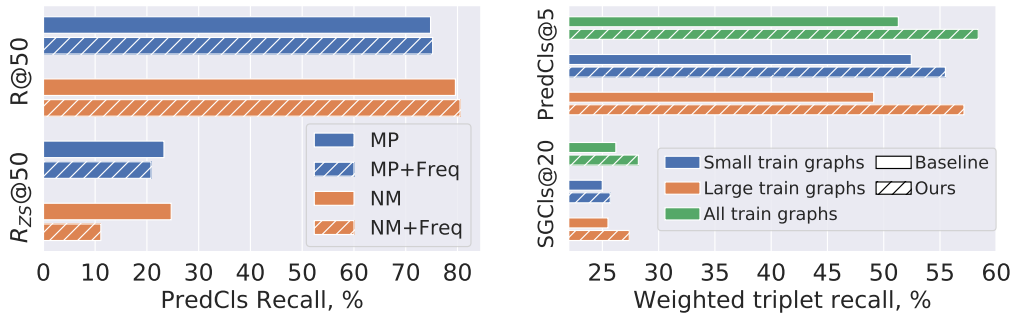


Figure 4.6: (**left**) Ablating FREQ. FREQ only marginally improves results on R@50. At the same time, it leads to large drops in zero-shot recall R_{ZS}@50 (and our weighted triplet recall, see Table 4.1). (**right**) Learning from small ($N \leq 10$) *vs.* large ($N > 10$) graphs. Our loss makes models learn from larger graphs more effectively, which is important for generalization, because such graphs contain a lot of labels (see Fig. 4.2).

including many few-shot labels (Fig. 4.2). Together, these two factors make the baseline ignore many few-shot triplets pertaining to larger graphs at training time, so the model cannot generalize to them at test time. Since the baseline essentially observes less variability during training, it leads to poor generalization on zero-shots as well. This argument aligns well to the works from other domains (Hill et al., 2019), showing that generalization strongly depends on the diversity of samples during training. Our loss fixes the issue of learning from larger graphs, which, given the reasons above, directly affects the ability to generalize.

Alternative approaches. We compare our loss to ones with tuned hyperparameters α, β, λ (Table 4.4):

$$\mathcal{L} = \mathcal{L}_{node} + \alpha \mathcal{L}_{FG} + \beta \mathcal{L}_{BG}, \tag{4.9}$$

$$\mathcal{L} = \mathcal{L}_{node} + \lambda \mathcal{L}_{edge}. \tag{4.10}$$

Our main finding is that, while these losses can give similar or better results in some cases, the parameters α, β and λ do not generally transfer across datasets and must be tuned every time, which can be problematic at larger scale (Zhang et al., 2019c). In contrast, our loss does not require tuning and achieves comparable performance.

To study the effect of density normalization separately from upweighting the edge loss (which is a side effect of our normalization), we also consider downweighting our edge term (4.6) by some $\gamma < 1$ to cancel out this upweighting effect. This ensures a similar range for the losses in our comparison. We found (Table 4.4) that the results are still significantly better than the baseline and, in some cases, even better than our hyperparameter-free loss. This further confirms that normalization of the graph density is important on its own. When carefully fine-tuned, the

Table 4.2: Zero-shot results (R_{ZS}@100) on the VTE split (Zhang et al., 2017). References: [1](Zhang et al., 2017), [2](Wang et al., 2019b), [3](Yang et al., 2018b), [4](Wang et al., 2019b).

Model	SGCls-GC	PredCls-GC
VTE[1,2]	–	16.4
STA[3]	–	18.9
ST-GAN[4]	–	19.0
MP, baseline (4.3)	2.3	20.4
MP, ours (4.6)	**3.1**	**21.4**

Table 4.3: Zero-shot results (R_{ZS}@100) on the VG split (Xu et al., 2017). Tang et al. (2020) use ResNeXt-101 as a backbone, which helps to improve results.

Model	SGCls-GC	PredCls-GC
NM+TDE (Tang et al., 2020)	**4.5**	18.2
NM, baseline (4.3)	1.7	9.5
MP, baseline (4.3)	3.2	20.1
NM, ours (4.6), no Freq	3.9	20.4
MP, ours (4.6)	4.2	**21.5**

Table 4.4: Comparing our loss to other approaches using MP (Xu et al., 2017; Zellers et al., 2018) and R/R_{ZS}@K metrics. Best results for each metric are bolded.

			Testing on VG		Testing on GQA	
Tuning dataset	**Hyperparams**	**Loss**	SGCls@100	PredCls@50	SGCls@100	PredCls@50
No tune (baseline)	–	(4.3)	47.2/8.2	74.8/23.3	27.1/2.8	59.7/34.9
VG	$\lambda = 20$	(4.10)	48.9/9.2	**78.3**/27.9	26.6/2.6	60.4/36.9
VG	$\alpha = 0.5, \beta = 20$	(4.9)	**49.1**/9.4	78.2/27.8	27.1/2.9	60.5/36.3
GQA	$\lambda = 5$	(4.10)	48.8/9.2	78.0/26.8	**27.8**/2.9	60.5/36.1
GQA	$\alpha = 1, \beta = 5$	(4.9)	48.6/8.7	77.4/27.8	27.5/2.9	60.7/36.6
No tune (ours, independ. norm)	$\alpha = \beta = 1$	(4.9)	47.5/8.4	74.3/25.3	27.4/2.9	59.5/35.4
No tune (ours, no upweight)	$\gamma = 0.05/0.2$ for VG/GQA	(4.6)	48.7/**9.6**	**78.3**/28.2	27.4/2.9	**61.1**/36.8
No tune (ours)	$\gamma = 1$	(4.6)	48.6/9.1	78.2/**28.4**	27.6/**3.0**	61.0/**37.2**

effects of normalization and upweighting are complimentary (*e.g.* when α, β or γ are fine-tuned, the results tend to be better).

Comparison to other zero-shot works. We also compare to previous works studying zero-shot generalization (Tables 4.2 and 4.3). For comprehensive evaluation, we test on both VTE and VG splits. We achieve superior results on VTE, even by just using the baseline MP, because, as shown in our main results, it generalizes well. On the VG split, we obtain results that compete with a more recent Total Direct Effect (TDE) method (Tang et al., 2020), even though the latter uses a more advanced detector and feature extractor. In all cases, our loss improves baseline results and, except for R_{ZS}@100 in SGCls, leads to state-of-the-art generalization. Our loss and TDE can be applied to a wide range of models, beyond MP and NM, to potentially have a complementary effect on generalization, which is interesting to study in future work.

Comparison on SGGen, $P(\mathcal{G}|I)$. In SGCls and PredCls, we relied on ground truth bounding boxes B_{gt}, while in SGGen the bounding boxes B_{pred} predicted by a detector should be used to

Table 4.5: Comparison of our SGG methods to the state-of-the-art on Visual Genome (split (Xu et al., 2017)). We report results with and without the graph constraint, **SGGen-GC** and **SGGen** respectively. All models use Faster R-CNN (Ren et al., 2015) as a detector, but the models we evaluate use a weaker backbone compared to (Tang et al., 2020). Interestingly, TDE's improvement on zero-shots (R_{ZS}) and mean recall (mR) comes at a significant drop in R@100, which means that frequent triplets are recognized less accurately. Our loss does not suffer from this. Table cells are colored the same way as in Table 4.1. References: [1](Zellers et al., 2018), [2](Xu et al., 2017), [3](Chen et al., 2019a), [4](Zhang et al., 2019e), [5](Tang et al., 2020).

Model	Backbone	Loss	SGGen-GC			SGGen		
			R@100	R_{ZS}@100	mR@100	R@100	R_{ZS}@100	mR@100
FREQ[1]	VGG16		27.6	0.02	5.6	30.9	0.1	8.9
MP[1,2]	VGG16	BASELINE (4.3)	24.3	0.8	4.5	27.2	0.9	7.1
	VGG16	OURS (4.6)	25.2	0.9	5.8	28.2	1.2	9.5
NM[1]	VGG16	BASELINE (4.3)	29.8	0.3	5.9	35.0	0.8	12.4
	VGG16	OURS (4.6)	29.4	1.0	**8.1**	35.0	1.8	15.4
	VGG16	OURS (4.6), NO FREQ	30.4	**1.7**	7.8	35.9	**2.4**	15.3
KERN[3]	VGG16	BASELINE (4.3)	29.8	0.04	7.3	35.8	0.02	**16.0**
RelDN[4]	VGG16	BASELINE (4.3)	**32.7**	–	–	**36.7**	–	–
RelDN[4]	ResNeXt-101	BASELINE (4.3)	36.7	–	–	40.0	–	–
NM[5]	ResNeXt-101	BASELINE (4.3)	**36.9**	0.2	6.8	–	–	–
NM+TDE[5]	ResNeXt-101	BASELINE (4.3)	20.3	2.9	9.8	–	–	–
VCTree+TDE[5]	ResNeXt-101	BASELINE (4.3)	23.2	**3.2**	**11.1**	–	–	–

enable a complete image-to-scene graph pipeline. Here, even small differences between B_{gt} and B_{pred} can create large distribution shifts between corresponding extracted features (V, E) (see § 4.3.1), on which SGCls models are trained. Therefore, it is important to *refine* the SGCls model on (V, E) extracted based on predicted B_{pred}, according to previous work (Zellers et al., 2018; Chen et al., 2019a). In our experience, this refinement can boost the R@100 result by around 3% for Message Passing and up to 8% for Neural Motifs (in absolute terms). In Table 4.5, we report results after the refinement completed both for the baseline loss and our loss in the same way. Similarly to the SGCls and PredCls results, our loss consistently improves baseline results in SGGen. It also allows Neural Motifs (NM) to significantly outperform KERN (Chen et al., 2019a) on zero-shots (R_{ZS}@100), while being only slightly worse in one of the mR@100 results. The main drawback of KERN is its slow training, which prevented us to explore this model together with our loss. Following our experiments in Table 4.1 and Fig. 4.6, we also confirm the positive effect of removing FREQ from NM. A more recent work of Tang et al. (2020) shows better results on zero-shots and mean recall, however, we note their more advanced feature extractor, therefore it is difficult to compare our results to theirs in a fair fashion. But, since they also use the baseline loss (4.3), our loss (4.6) can potentially improve their model, which we leave for future work.

Finally, we evaluate SGGen on GQA using Message Passing (Table 4.6), where we also obtain

improvements with our loss. GQA has 1703 object classes compared to 150 in VG making object detection harder. When evaluating SGGen, the predicted triplet is matched to ground truth (GT) if predicted and GT bounding boxes have an intersection over union (IoU) of ≥50%, so more misdetections lead to a larger gap between SGCls and SGGen results.

Table 4.6: **SGGen** results on GQA (Hudson and Manning, 2019b) using MP. Mask R-CNN (He et al., 2017) fine-tuned on GQA is used in this task.

Loss	R@300	R_{ZS}@300	mR_{tr}@300
BASELINE (4.3)	6.2	0.5	1.3
OURS (4.6)	**6.3**	**0.7**	**2.4**

Qualitative results. To qualitatively inspect the effect of our loss on predicted scene graphs, we visualize two cases (Fig. 4.7): (**top four rows**) when the baseline (MP) model makes a correct prediction, while our model is incorrect; and (**bottom four rows**) when the baseline is incorrect, while ours is correct. For the purpose of this visualization, a prediction is considered correct when the zero-shot triplet is in the top-20[2] triplets in the image regardless if the detected bounding boxes overlap with the ground truth. In the first case (when the baseline is correct), the predicted triplet often includes the 'on' predicate, which we believe is due to the baseline being more biased to the frequency distribution. The model with our loss makes more diverse predictions showing a better understanding of scenes. Also, the ground truth is often mislabeled (see 'leaf on bike') or a synonym is predicted by our model (*e.g.* a plant and a flower), which counts as an error. In the second case (when ours is correct), the baseline model tends to return a poor ranking of triplets and often simply gives a higher rank to frequent triplets.

4.5 Conclusions

Scene graphs are a useful semantic representation of images, accelerating research in many applications including visual question answering. It is vital for the SGG model to perform well on rare/unseen compositions of objects and predicates, which are inevitable due to an extremely long tail of the distribution over triplets. We show that baseline models do not effectively learn from all labels, leading to poor generalization on few/zero shots. Moreover, current evaluation metrics do not reflect this problem, exacerbating it instead. We also show that learning well from larger graphs is essential to enable stronger generalization. To this end, we modify the loss commonly used in SGG and achieve significant gains and, in certain cases, state-of-the-art results, on both the existing and our novel weighted metric.

[2]Conventionally (Xu et al., 2017; Zellers et al., 2018), triplets are ranked according to the product of softmax scores of the subject, object and predicate.

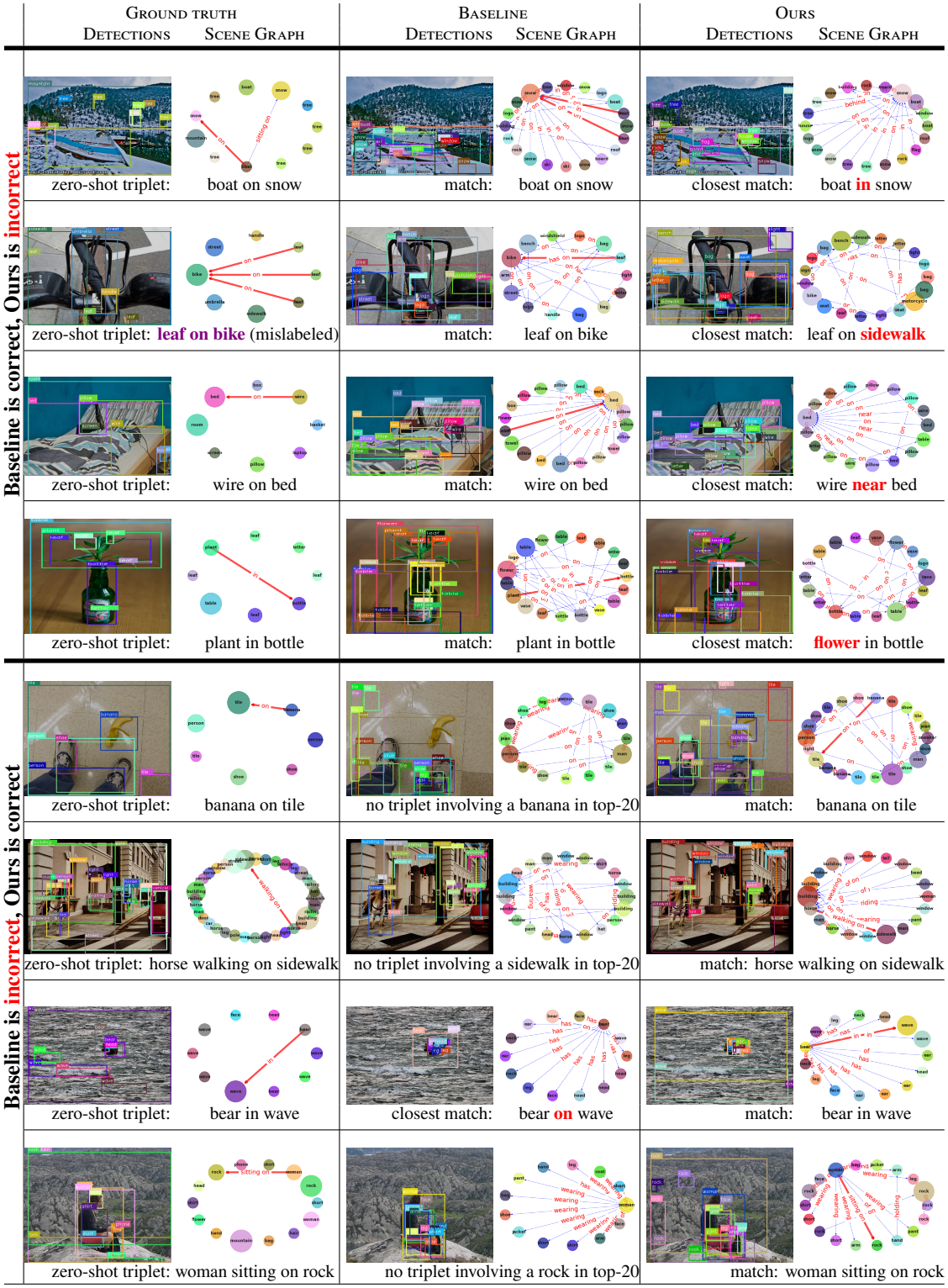


Figure 4.7: Visualizations of scene graph generation for zero-shots (denoted as thick red arrows) on Visual Genome using Message Passing with the baseline loss versus our loss. Most edges are two-way, but for clarity we show them one-way. These examples are picked randomly. Intended to be viewed on a computer display.

5 Generative Compositional Augmentations for Scene Graph Prediction

Prologue

Context. In the previous chapter, our proposed method improved compositional generalization in the scene graph generation (SGG) task. Despite our work and other works addressing compositional generalization, the performance of SGG models remains very low on rare and unseen compositions. The low performance may be in large part due to the inherent bias present in the training datasets. In particular, most training images and corresponding annotations contain only very frequent visual compositions. Therefore, one straightforward approach to mitigate this bias may be to directly increase the size of the training dataset by adding more of the rare compositions. However, it is challenging to augment graphs and to collect or synthesize corresponding images, especially for a large-scale SGG task.

Contributions. We develop a method based on conditional generative adversarial networks (cGANs). Our cGAN leverages recent advancements in GANs (*e.g.* (Park et al., 2019)) and allows to generate visual features conditioned on rare and unseen compositions. We create these compositions by perturbing existing scene graphs from the Visual Genome dataset. Our cGAN model can be added to existing SGG models and we show that it improves their generalization ability.

Recent works. In the scene understanding domain, there was an approach similar to ours that also generated features for novel compositions created randomly (Wang et al., 2019b). In other visual tasks, where there is no extra complexity due to graphs, simple augmentation approaches based on adversarial training (Shetty et al., 2020), and cut-paste techniques (Dwibedi et al., 2017; Dvornik et al., 2018; Tripathi et al., 2019a; Ghiasi et al., 2021) have been improving generalization. In domains beyond vision, such as natural language processing (NLP) (Joshi and He, 2021) and reinforcement learning (Hill et al., 2019), ideas similar to ours (generally based on counter-factual augmentation in NLP (Kaushik et al., 2019)) have been adopted to reduce bias and improve generalization.

5.1 Introduction

Reasoning about the world in terms of objects and relationships between them is an important aspect of human and machine cognition (Greff et al., 2020). In our environment, we can often observe frequent compositions such as “person on a surfboard” or “person next to a dog”. When we

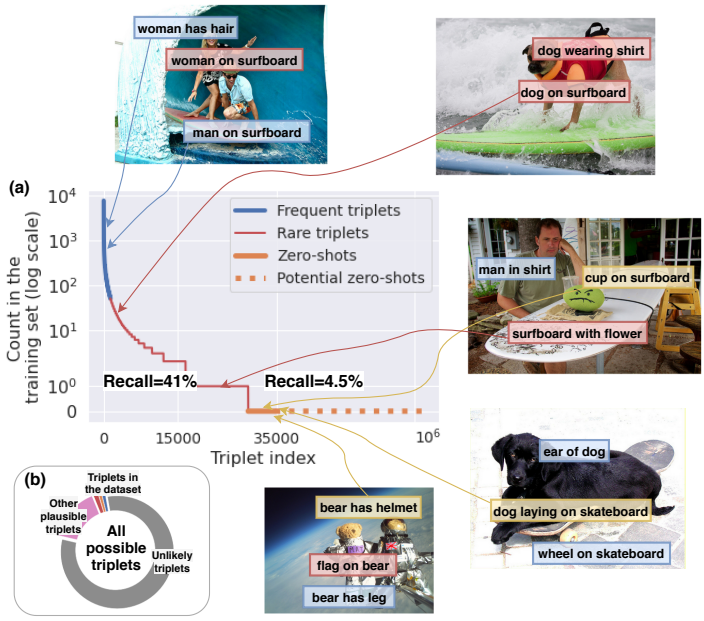


Figure 5.1: (**a**) The triplet distribution in Visual Genome (Krishna et al., 2017a) is extremely long-tailed, with numerous few- and zero-shot compositions (highlighted in red and yellow respectively). (**b**) The training set contains a tiny fraction (3%) of all possible triplets, while many other plausible triplets exist. We aim to "hallucinate" such compositions using GANs to increase the diversity of training samples and improve generalization. Recall results are from (Tang et al., 2020).

are faced with a rare or previously unseen composition such as "dog on a surfboard", to understand the scene we need to understand the concepts of 'person', 'dog', 'surfboard' and 'on'. While such unbiased reasoning about concepts is easy for humans, for machines this task has remained extremely challenging (Atzmon et al., 2016; Johnson et al., 2017a; Bahdanau et al., 2018; Keysers et al., 2019; Lake, 2019). Learning-based models tend to capture spurious statistical correlations in the training data (Arjovsky et al., 2019; Niu et al., 2020), *e.g.* 'person' rather than 'dog' has always occurred on a surfboard. When the evaluation is explicitly focused on *compositional generalization* – ability to recognize novel or rare combinations of objects and relationships – such models then can fail remarkably (Atzmon et al., 2016; Lu et al., 2016; Tang et al., 2020; Knyazev et al., 2020).

Predicting compositions of objects and the relationships between them from images is part of the scene graph generation (SGG) task. SGG is important, because accurately inferred scene graphs can improve downstream results in tasks, such as VQA (Zhang et al., 2019a; Hudson and Manning, 2019a; Cangea et al., 2019; Lee et al., 2019b; Shi et al., 2019; Hildebrandt et al., 2020; Damodaran et al., 2021), image captioning (Yang et al., 2019; Gu et al., 2019a; Li and Jiang, 2019; Wang et al., 2019a; Milewski et al., 2020), retrieval (Johnson et al., 2015; Belilovsky et al., 2017; Tang et al., 2020; Tripathi et al., 2019b; Schroeder and Tripathi, 2020) and others (Agarwal et al., 2020; Xu et al., 2020). However, inferring scene graphs accurately is challenging due to a

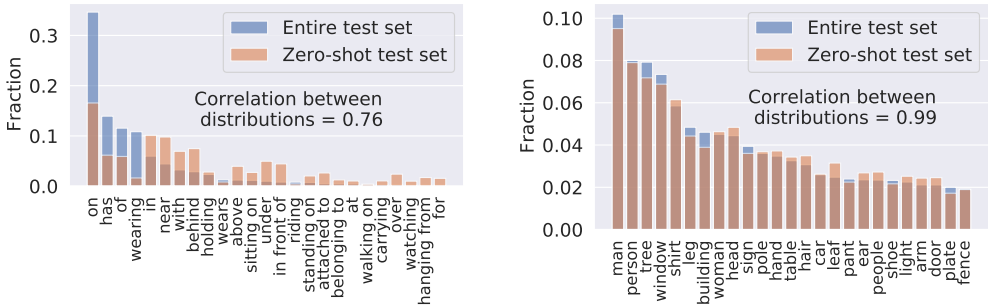


Figure 5.2: The distributions of top-25 predicate (**left**) and object (**right**) categories in Visual Genome (Krishna et al., 2017a) (split of (Xu et al., 2017)).

long tail data distribution and inevitable appearance of zero-shot (ZS) compositions (triplets) of objects and relationships at test time, *e.g.* "cup on surfboard" (Figure 5.1). The SGG results using the recent Total Direct Effect (TDE) method (Tang et al., 2020) show a severe drop in ZS recall highlighting the extreme challenge of compositional generalization. This might appear surprising given that the marginal distributions in the entire scene graph dataset (*e.g.* Visual Genome (Krishna et al., 2017a)) and the ZS subset are very similar (Fig. 5.2). More specifically, the predicate and object categories that are frequent in the entire dataset, such as 'on', 'has' and 'man', 'person' *also dominate* among the ZS triplets. For example, both "cup on surfboard" and "bear has helmet" consist of frequent entities, but represent extremely rare compositions (Fig. 5.1). This strongly suggests that the challenging nature of correctly predicting ZS triplets does not directly stem from the imbalance of predicates (or objects), as commonly viewed in the previous SGG works, where the models attempt to improve mean (or predicate-normalized) recall metrics (Chen et al., 2019a; Dornadula et al., 2019; Tang et al., 2019; Zhang et al., 2019e; Tang et al., 2020; Chen et al., 2019b; Zareian et al., 2020a; Lin et al., 2020; Zareian et al., 2020b; Yan et al., 2020). Therefore, we focus on compositional generalization and associated zero- and few-shot metrics.

Despite recent improvements in compositional generalization within the SGG task (Tang et al., 2020; Knyazev et al., 2020; Suhail et al., 2021), the state-of-the-art result in zero-shot recall is still 4.5% compared to 41% for all-shot recall (Figure 5.3). To address compositional generalization, we consider exposing the model to a large diversity of training examples that can lead to emergent generalization (Hill et al., 2019; Ravuri and Vinyals, 2019). To avoid expensive labeling of additional data, we propose a compositional augmentation approach based on conditional generative adversarial networks (GANs) (Goodfellow et al., 2014; Mirza and Osindero, 2014). Our general idea is augmenting the dataset by perturbing scene graphs and corresponding visual features of images, such that together they represent a novel or rare situation.

Overall, we make the following **contributions**:

•We propose scene graph perturbation methods (§ 5.3.1) as part of a GAN-based model

(§ 5.3.1), to augment the training set with underrepresented compositions;

- We propose natural language- and dataset-based metrics to evaluate the quality of (perturbed) scene graphs (§ 5.3.2);
- We extensively evaluate our model and outperform a strong baseline in zero-, few- and all-shot recall (§ 5.4).

Our code is available at https://github.com/bknyaz/sgg.

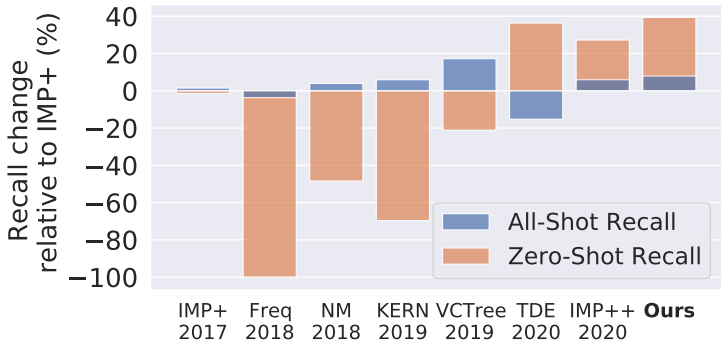


Figure 5.3: In this work, the compositional augmentations we propose improve on zero-shot (ZS) as well as all-shot recall.

5.2 Related work

Scene Graph Generation. SGG (Xu et al., 2017) extended an earlier visual relationship detection (VRD) task (Lu et al., 2016; Sadeghi and Farhadi, 2011), enabling generation of a complete scene graph (SG) for an image. This spurred more research at the intersection of vision and language, where a SG can facilitate high-level visual reasoning tasks such as VQA (Zhang et al., 2019a; Hudson and Manning, 2019a; Shi et al., 2019) and others (Agarwal et al., 2020; Xu et al., 2020; Raboh et al., 2020). Follow-up SGG works (Li et al., 2017; Yang et al., 2018a; Zellers et al., 2018; Zhang et al., 2019e; Gu et al., 2019b; Tang et al., 2019; Lu et al., 2019, 2021) have significantly improved the performance in terms of all-shot recall (Fig. 5.3). While the problem of zero-shot (ZS) generalization was already actively explored in the VRD task (Zhang et al., 2017; Yang et al., 2018b; Wang et al., 2019b), in a more challenging SGG task and on a realistic dataset, such as Visual Genome (Krishna et al., 2017a), this problem has been addressed only recently in (Tang et al., 2020) by proposing Total Direct Effect (TDE), in (Knyazev et al., 2020) by normalizing the graph loss, and in (Suhail et al., 2021) by the energy-based loss. Previous SGG works have not addressed the compositional generalization issue by synthesizing rare SGs. The closest work that also considers a generative approach is (Wang et al., 2019b) solving the VRD task. Compared to it, our model follows a standard SGG pipeline and evaluation (Xu et al., 2017; Zellers et al.,

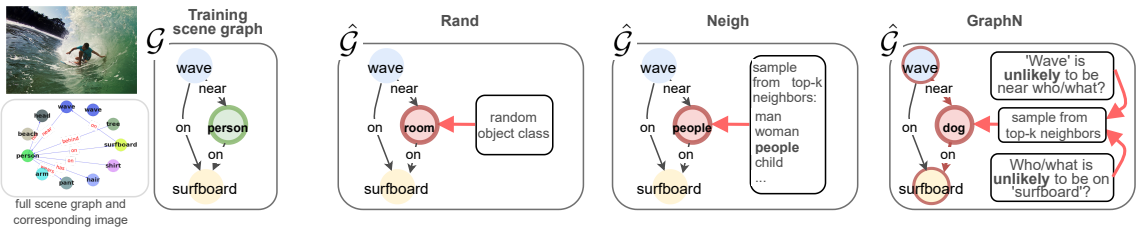


Figure 5.4: Illustrative examples of different perturbation schemes we consider. Only the subgraph is shown for clarity.

2018) including object and predicate classification, instead of classifying only the predicate. We also condition a GAN on SGs rather than triplets, which combinatorially increases the number of possible augmentations. To improve SG's likelihood, we leverage both the language model and dataset statistics as opposed to random compositions as in (Wang et al., 2019b).

Predicate imbalance and mean recall. Recent SGG works have focused on the predicate imbalance problem (Chen et al., 2019a; Dornadula et al., 2019; Tang et al., 2019; Zhang et al., 2019e; Tang et al., 2020; Chen et al., 2019b; Zareian et al., 2020a; Lin et al., 2020; Zareian et al., 2020b; Yan et al., 2020) and mean (over predicates) recall as a metric not sensitive to the dominance of frequent predicates. However, as we discussed in § 5.1, the challenge of compositional generalization does not directly stem from the imbalance of predicates, since frequent predicates (*e.g.* 'on') still dominate in unseen/rare triplets (Fig. 5.2). Moreover, (Tang et al., 2020) showed mean recall is relatively easy to improve by standard Reweight/Resample methods, while ZS recall is not.

Data augmentation with GANs. Data augmentation is a standard method for improving machine learning models (Ratner et al., 2017). Typically these methods rely on domain specific knowledge such as applying known geometric transformations to images (DeVries and Taylor, 2017b; Cubuk et al., 2018). In the case of SGG we require more general augmentation methods, so here we explore a GAN-based approach as one of them. GANs (Goodfellow et al., 2014) have been significantly improved w.r.t. stability of training and the quality of generated samples (Brock et al., 2018; Karras et al., 2020), with recent works considering their usage for data augmentation (Ravuri and Vinyals, 2019; Shin et al., 2018; Sandfort et al., 2019). Furthermore, recent work has shown that it is possible to produce plausible out-of-distribution (OOD) examples conditioned on unseen label combinations, by intervening on the underlying graph (Kocaoglu et al., 2017; Casanova et al., 2020; Sun and Wu, 2020; Deng et al., 2021; Greff et al., 2019). In this work, we have direct access to the underlying graphs of images in the form of SGs, which allows us to condition on OOD compositions as in (Casanova et al., 2020; Deng et al., 2021).

5.3 Methods

We consider a dataset of N tuples $\mathcal{D} = \{(I, \mathcal{G}, B)\}^N$, where I is an image with a corresponding *scene graph* $\mathcal{G}$ (Johnson et al., 2015) and bounding boxes B. A scene graph $\mathcal{G} = (O, R)$ consists of n objects $O = \{o_1, ..., o_n\}$, and m relationships between them $R = \{r_1, ..., r_m\}$. For each object o_i there is an associated bounding box $b_i \in \mathbb{R}^4$, $B = \{b_1, ..., b_n\}$. Each object o_i is labeled with a particular category $o_i \in C$, while each relationship $r_k = (i, e_k, j)$ is a triplet with a subject (start node) i, an object (end node) j and a predicate $e_k \in \mathcal{R}$, where $\mathcal{R}$ is a set of all predicate classes. For further convenience, we define a categorical triplet (*composition*) $\tilde{r}_k = (o_i, e_k, o_j)$ consisting of object and predicate categories, $\tilde{R} = \{\tilde{r}_1, ..., \tilde{r}_m\}$. An example of a scene graph is presented in Figure 5.4 with objects $O = \{\texttt{person}, \texttt{surfboard}, \texttt{wave}\}$ and relationships $R = \{(3, \texttt{near}, 1), (1, \texttt{on}, 2)\}$ and categorical relationships $\tilde{R} = \{(\texttt{wave}, \texttt{near}, \texttt{person}), (\texttt{person}, \texttt{on}, \texttt{surfboard})\}$.

5.3.1 Generative compositional augmentations

In a given dataset $\mathcal{D}$, such as Visual Genome (Krishna et al., 2017a), the distribution of triplets is extremely long-tailed with a small fraction of dominating triplets (Fig. 5.1). To address the long-tail issue, we consider a GAN-based approach to augment $\mathcal{D}$ and artificially upsample rare compositions. Our model is based on the high-level idea of generating an additional set $\hat{\mathcal{D}} = \{(\hat{I}, \hat{\mathcal{G}}, \hat{B})\}^{\hat{N}}$. A typical scene-graph-to-image generation pipeline is (Johnson et al., 2018) $\hat{\mathcal{G}} \rightarrow \hat{B} \rightarrow \hat{I}$. We describe our model accordingly by beginning with constructing $\hat{\mathcal{G}}$ and $\hat{B}$ (§ 5.3.1) followed by the generation of $\hat{I}$ (in our case, features) (§ 5.3.1). See Figure 5.5 for the overall pipeline.

Scene Graph Perturbations

We propose three methods to synthetically upsample underrepresented triplets in the dataset (Fig. 5.4). Our goal is to construct diverse compositions avoiding both very likely (already abundant in the dataset) and very unlikely ("implausible") combinations of objects and predicates, so that the distribution of synthetic $\hat{\mathcal{G}}$ will resemble the tail of the real distribution of $\mathcal{G}$. To construct $\hat{\mathcal{G}}$, we perturb existing $\mathcal{G}$ available in $\mathcal{D}$, since constructing graphs from scratch is more difficult: $\mathcal{G} \rightarrow \hat{\mathcal{G}}$. We focus on perturbing nodes only as it allows the creation of highly diverse compositions, so $\hat{\mathcal{G}} = (\hat{O}, R)$, where $\hat{O} = \{\hat{o}_1, ..., \hat{o}_n\}$ are the replacement object categories. We perturb only $L \cdot n$ nodes, where $L \in \mathbb{R}^{[0,1]}$, so $\hat{o}_i = o_i$ for $n(1 - L)$ nodes. We sample $L \cdot n$ nodes for perturbation based on their sum of in and out degrees. Each scene graph typically has a few "hub" nodes densely connected to other nodes. So, by perturbing the hubs, we introduce more novel compositions with fewer perturbations.

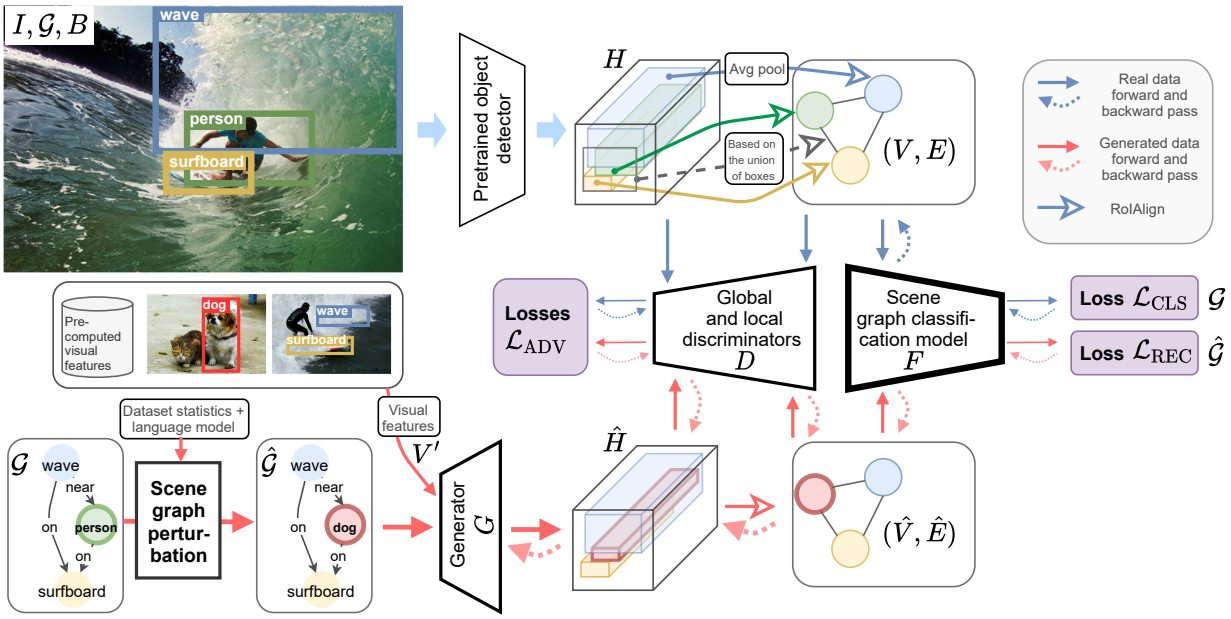


Figure 5.5: Our generative scene graph augmentation pipeline with its main components: discriminators D, a generator G and a scene graph classification model F. See § 5.3 for a detailed description of our pipeline and model architectures.

RAND (random) is the simplest strategy, where for a node i we uniformly sample a category $\hat{o}$ from C, so that $o_i = \hat{o}$.

NEIGH (semantic neighbors) leverages pretrained GloVe word embeddings (Pennington et al., 2014) available for each of the object categories C. Thus, given node i of category o_i we retrieve the top-k neighbors of o_i in the embedding space using cosine similarity. We then uniformly sample $\hat{o}$ from the top-k neighbors replacing o_i with $\hat{o}$.

GRAPHN (graph-structured semantic neighbors). RAND and NEIGH do not take into account the graph structure or dataset statistics leading to unlikely or not diverse enough compositions. To alleviate that, we propose the GRAPHN method. Given node i of category o_i in the graph $\mathcal{G}$, we consider all triplets $\tilde{R}_i = \{\tilde{r}_{k,i}\}$ in $\mathcal{G}$ that contain i as the start or end node, i.e. $\tilde{r}_{k,i} = (o_i, e_k, o_j)$ or (o_j, e_k, o_i). For example in Figure 5.4, if o_i is 'person', then $\tilde{R}_i = \{(\texttt{person}, \texttt{on}, \texttt{surfboard}), (\texttt{wave}, \texttt{near}, \texttt{person})\}$. For each $\tilde{r}_{k,i}$ we find all triplets $\tilde{R}_c$ in the dataset $\mathcal{D}$ matching (o_c, e_k, o_j) or (o_j, e_k, o_c), where $o_c \neq o_i$ is a candidate replacement for o_i. For each candidate o_c, we count matched triplets $n_c = |\tilde{R}_c|$ and define unnormalized probabilities $\hat{p}_c$ based on the inverse of n_c, namely $\hat{p}_c = 1/n_c$. This way we define a set of possible replacements $\{o_c, \hat{p}_c\}$ for node i.

One of our key observations is that depending on the evaluation metric and amount of noise in the dataset, we might want to avoid sampling candidates with very high $\hat{p}_c$ (low n_c). Therefore, to control for that, we introduce an additional hyperparameter α that allows to filter out candidates with

$n_c < \alpha$ by setting their $\hat{p}_c$ to 0. This way we can trade-off between upsampling rare and frequent triplets. We then normalize p_c to ensure $\sum p_c = 1$ and sample $o' \sim p_c$. To further increase the diversity, the final $\hat{o}$ is chosen from the top-k semantic neighbors of o' as in NEIGH, including o' itself. GRAPHN is a sequential perturbation procedure, where for each node the perturbation is conditioned on the current graph state. In contrast, RAND and NEIGH perturb all $L \cdot n$ nodes in parallel.

Bounding boxes. Since we perturb only a few nodes, for simplicity we assume that the perturbed graph has the same bounding boxes B: $\hat{B} = B$. While one can reasonably argue that object sizes and positions vary a lot depending on the category, i.e. "elephant" is much larger than "dog", we can often find instances disproving that, *e.g.* if a toy "elephant" or a drawing of an elephant is present. Empirically we found this approach to work well.

Scene Graph to Visual Features

Given perturbed $(\hat{\mathcal{G}}, \hat{B})$, the next step in our GAN-based pipeline is to generate visual features (Figure 5.5). To train such a model, we first need to extract real features from the dataset $\mathcal{D} = \{(I, \mathcal{G}, B)\}^N$. Following (Xu et al., 2017; Zellers et al., 2018), we use a pretrained and frozen object detector (Ren et al., 2015) to extract global visual features H from input images. Then, given B and H, we use RoIAlign (He et al., 2017) to extract visual features (V, E) of nodes and edges, respectively. To extract edge features between a pair of nodes, the union of their bounding boxes is used (Zellers et al., 2018). Since we do not update the detector, we do not need to generate images as in scene-graph-to-image models (Johnson et al., 2018), just intermediate features $\hat{H}, \hat{V}, \hat{E}$.

Main scene graph classification model F. Given extracted (V, E), the main model F predicts a scene graph $\mathcal{G} = (O, R)$, i.e. it needs to correctly assign object labels O to node features V and predicate classes R to edge features E. Our pipeline is not constrained to the choice of F.

Generator G. Our scene-graph-to-features generator G follows the architecture of (Johnson et al., 2018). First, a scene graph $\hat{\mathcal{G}}$ is processed by a graph convolutional network (GCN) to exchange information between nodes and edges. We found it beneficial to concatenate output GCN features of all nodes with visual features V', where V' are sampled from the set $\{V_{o_i}\}$ precomputed at the previous stage and o_i is the category of node i. By conditioning the generator on visual features, the main task of G becomes simply to align and smooth the features appropriately, which we believe is easier than generating visual features from the categorical distribution. In addition, the randomness of this sampling step injects noise improving the diversity of generated features. The generated node features and the bounding boxes $\hat{B}$ are used to construct the layout followed by feature refinement (Johnson et al., 2018) to generate $\hat{H}$. Afterwards, $(\hat{V}, \hat{E})$ are extracted from $\hat{H}$ the same way as (V, E).

Discriminators D. We have independent discriminators for nodes and edges, D_{node} and D_{edge},

that discriminate real features (V, E) from fake ones ($\hat{V}$, $\hat{E}$) conditioned on their class as per the CGAN (Mirza and Osindero, 2014; Radford et al., 2015). We add a global discriminator D_{global} acting on feature maps H, which encourages global consistency between nodes and edges. Thus, D_{node} and D_{edge} are trained to match marginal distributions, while D_{global} is trained to match the joint distribution. The right balance between these discriminators should enable the generation of realistic visual features conditioned on OOD scene graphs. Please see our source code for the detailed architectures of D and G.

Losses. To train our generative model, we define several losses. These include the baseline SG classification loss (5.1) and ones specific to our generative pipeline (5.2)-(5.5). The latter are motivated by a CycleGAN (Zhu et al., 2017) and, similarly, consist of the reconstruction and adversarial losses (5.2)-(5.5).

We use an improved **scene graph classification loss** from (Knyazev et al., 2020), which is a sum of the node cross-entropy loss $\mathcal{L}^O$ and graph density-normalized edge cross-entropy loss $\mathcal{L}^R$:

$$\mathcal{L}_{\text{CLS}} = \mathcal{L}(F(V,E),\mathcal{G}) = \mathcal{L}^O(F(V,E),O) + \mathcal{L}^R(F(V,E),R). \tag{5.1}$$

$\mathcal{L}^R$ is computed based on the ratio of foreground (annotated) to background (not annotated) edges in a batch of scene graphs (Knyazev et al., 2020). To improve F by training it on augmented features ($\hat{V}$, $\hat{E}$), we define the **reconstruction (cycle-consistency) loss** analogous to (5.1):

$$\mathcal{L}_{\text{REC}} = \mathcal{L}(F(G(\hat{\mathcal{G}},\hat{B},V')),\hat{\mathcal{G}}) = \mathcal{L}^O(F(\hat{V},\hat{E}),\hat{O}) + \mathcal{L}^R(F(\hat{V},\hat{E}),R). \tag{5.2}$$

We do not update G on this loss to prevent its potential undesirable collaboration with F. Instead, to train G as well as D, we optimize **conditional adversarial losses** (Mirza and Osindero, 2014). We first write these separately for D and G in a general form. So, for some features $\boldsymbol{x}$ and their corresponding class $\boldsymbol{y}$:

$$\mathcal{L}^D_{\text{ADV}}(\boldsymbol{x},\boldsymbol{y}) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x}|\boldsymbol{y})] + \mathbb{E}_{\hat{\mathcal{G}}\sim p_{\hat{\mathcal{G}}}(\hat{\mathcal{G}})}[\log(1 - D(G(\hat{\mathcal{G}})|\boldsymbol{y})] \tag{5.3}$$

$$\mathcal{L}^G_{\text{ADV}}(\boldsymbol{y}) = \mathbb{E}_{\hat{\mathcal{G}}\sim p_{\hat{\mathcal{G}}}(\hat{\mathcal{G}})}[\log D(G(\hat{\mathcal{G}})|\boldsymbol{y})]. \tag{5.4}$$

We compute these losses for object and edge visual features by using the discriminators D_{node} and D_{edge}. This loss is also computed for global features H using D_{global}, so that the total discriminator and generator losses are:

$$\begin{aligned}\mathcal{L}^D_{\text{ADV}} &= \mathcal{L}^D_{\text{ADV}}(V,O) + \mathcal{L}^D_{\text{ADV}}(E,R) + \mathcal{L}^D_{\text{ADV}}(H,\emptyset)\\ \mathcal{L}^G_{\text{ADV}} &= \mathcal{L}^G_{\text{ADV}}(O) + \mathcal{L}^G_{\text{ADV}}(R) + \mathcal{L}^G_{\text{ADV}}(\emptyset),\end{aligned} \tag{5.5}$$

where $\emptyset$ denotes that our global discriminator is unconditional for simplicity. Thus, the total loss to minimize is:

$$\mathcal{L} = \underbrace{\mathcal{L}_{\text{CLS}} + \mathcal{L}_{\text{REC}}}_{\text{update } F} - \gamma (\underbrace{\mathcal{L}^{D}_{\text{ADV}}}_{\text{update } D} + \underbrace{\mathcal{L}^{G}_{\text{ADV}}}_{\text{update } G}), \tag{5.6}$$

where the loss weight $\gamma = 5$ worked well in our experiments. Compared to a similar work of (Wang et al., 2019b), in our model all of its components (F, D, G) are learned jointly end-to-end.

5.3.2 Semantic plausibility of scene graphs

Language model. To directly evaluate the quality of perturbations, it is desirable to have some quantitative measure other than downstream SGG performance. We found that a cheap (relative to human evaluation) and effective way to achieve this goal is to use a language model. In particular, we use a pretrained BERT (Devlin et al., 2018) model and estimate the "semantic plausibility" of both ground truth and perturbed scene graphs in the following way. We create a textual query from a scene graph by concatenating all triplets (in a random order). We then mask out one of the perturbed nodes (in case of $\hat{\mathcal{G}}$) or a random node (in case of $\mathcal{G}$) in the triplet, so that BERT can return (unnormalized) likelihood scores for the object category of the masked out token. We have also considered using this strategy to create SG perturbations as an alternative to GraphN. However, we did not find it effective for obtaining rare scene graphs, since BERT is not grounded to visual concepts and not aware of what is considered "rare" in a particular SG dataset. For qualitative evaluation and when BERT scores are averaged over many samples, we found them still useful as a rough measure of SG quality.

Hit rate. For perturbed SGs, we compute an additional qualitative metric, which we call the 'Hit rate'. Assuming we perturbed M triplets in total for all training SGs, this metric computes the percentage of the triplets matching an actual annotation in an evaluation test subset (zero-, few- or all-shot).

5.4 Experiments

5.4.1 Dataset, models and hyperparameters

We use a publicly available SGG codebase[1] for evaluation and baseline model implementations. For the model F, we use Iterative Message Passing (IMP+) (Xu et al., 2017; Zellers et al., 2018) and Neural Motifs (NM) (Zellers et al., 2018). IMP+ shows strong compositional generalization capabilities (Knyazev et al., 2020) and, therefore is more explored in this work. We use an improved loss for (5.1) from (Knyazev et al., 2020), so we denote our baselines as IMP++ and NM++. We use

[1] https://github.com/rowanz/neural-motifs

the default hyperparameters and identical setups for the baseline models without a GAN and our models with a GAN. We borrow the detector Faster-RCNN with the VGG16 backbone pretrained on Visual Genome (VG) from (Zellers et al., 2018) and use it in all our experiments. We evaluate the models on a standard split of VG (Krishna et al., 2017a), with the 150 most frequent object classes and 50 predicate classes, introduced in (Xu et al., 2017). The training set has 57723 and the test set has 26446 images. Similarly to (Knyazev et al., 2020; Wang et al., 2019b; Tang et al., 2020; Suhail et al., 2021), in addition to the all-shot (all test scene graphs) case, we define zero-shot, 10-shot and 100-shot test subsets. For each such subset we keep only those triplets in a scene graph that occur 0, 1-10 or 11-100 times during training and remove samples without such triplets, which results in 4519, 9602 and 16528 test scene graphs (and images) respectively. We use a held-out validation set of 5000 images for tuning the hyperparameters.

Table 5.1: Results on Visual Genome (Krishna et al., 2017a) using models based on IMP++ (Knyazev et al., 2020). The top-1 result in each column is **bolded** (ignoring ORACLE-Zs). ORACLE-Zs results are an upper bound estimate of ZS recall obtained by directly using ZS test triplets for perturbations.

MODEL	ZERO-SHOT RECALL		10-SHOT RECALL		100-SHOT RECALL		ALL-SHOT RECALL		
	SGCls	PredCls	SGCls	PredCls	SGCls	PredCls	SGCls	PredCls	SGCls-mR
Baseline (IMP++)	9.27±0.10	28.14±0.05	21.80±0.19	42.78±0.32	40.42±0.02	67.78±0.07	48.70±0.08	77.48±0.09	27.78±0.10
GAN+GRAPHN, $\alpha = 2$	**9.89**±0.15	28.90±0.14	21.96±0.30	**43.79**±0.27	41.22±0.33	69.17±0.24	50.06±0.29	78.98±0.09	27.79±0.48
GAN+GRAPHN, $\alpha = 5$	9.62±0.29	**29.18**±0.33	**22.24**±0.11	43.74±0.10	41.39±0.26	69.11±0.05	50.14±0.21	78.94±0.03	27.98±0.23
GAN+GRAPHN, $\alpha = 10$	9.84±0.17	28.90±0.46	22.04±0.33	43.54±0.36	41.46±0.15	69.13±0.24	50.10±0.23	79.00±0.09	27.68±0.37
GAN+GRAPHN, $\alpha = 20$	9.65±0.15	28.68±0.28	21.97±0.30	43.64±0.20	41.24±0.08	**69.31**±0.17	49.89±0.28	78.95±0.04	27.42±0.36
Ablated models									
GAN (no perturb.)	9.25±0.20	28.66±0.35	22.15±0.21	43.66±0.29	**41.58**±0.20	69.16±0.16	**50.38**±0.28	**79.05**±0.08	28.17±0.08
GAN+RAND	9.71±0.09	28.71±0.40	21.89±0.21	43.33±0.18	41.01±0.32	68.88±0.23	49.83±0.32	78.84±0.10	27.45±0.48
GAN+NEIGH	9.65±0.04	28.68±0.40	21.86±0.23	43.77±0.15	41.25±0.35	69.07±0.09	50.00±0.36	78.94±0.10	27.41±0.51
Other baselines									
REWEIGHT	9.58±0.14	28.27±0.22	22.19±0.09	42.98±0.17	40.00±0.01	65.27±0.13	48.13±0.10	74.68±0.13	**30.95**±0.05
RESAMPLE-predicates	9.13±0.06	27.77±0.10	21.35±0.05	42.14±0.16	39.69±0.06	66.74±0.01	48.23±0.10	76.59±0.05	28.44±0.38
RESAMPLE-triplets	8.94±0.16	27.66±0.14	21.65±0.10	42.60±0.17	39.39±0.08	66.44±0.06	47.77±0.10	76.38±0.14	27.56±0.10
TDE	9.21±0.21	27.91±0.09	21.20±0.16	41.61±0.32	39.72±0.10	65.40±0.21	48.35±0.08	76.22±0.17	28.25±0.21
ORACLE perturbations $\hat{\mathcal{G}}$									
GAN+ORACLE-Zs $\hat{\mathcal{G}}$	10.11±0.34	29.27±0.10	22.05±0.38	43.78±0.09	41.38±0.50	69.06±0.16	50.19±0.36	79.00±0.08	27.91±0.56
GAN+ORACLE-Zs $\hat{\mathcal{G}} + \hat{B}$	10.52±0.31	29.43±0.42	21.98±0.39	43.03±0.13	41.12±0.19	68.73±0.17	50.05±0.35	78.65±0.09	27.52±0.46

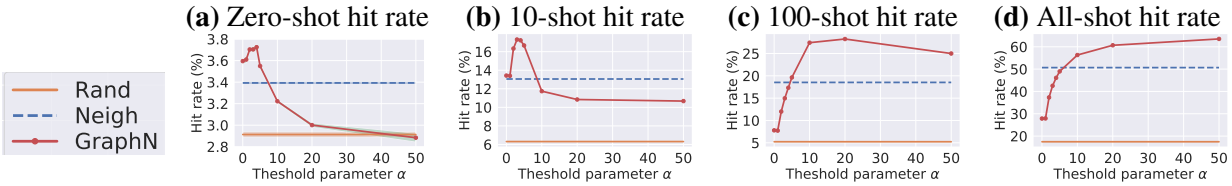


Figure 5.6: Triplet hit rates (§ 5.3.2) versus the threshold α on four different VG test subsets using our perturbation methods.

Baselines. In addition to the IMP++ and NM++ baselines, we evaluate RESAMPLE, REWEIGHT

and TDE (Tang et al., 2020) when combined with IMP++. RESAMPLE samples training images based on the inverse frequency of predicates/triplets (Tang et al., 2020). REWEIGHT increases the softmax scores of rare predicate classes. TDE debiases contextual edge features of a SGG model. We use the Total Effect (TE) variant according to Eq. 6 in (Tang et al., 2020), since applying TDE to IMP++ is not straightforward due to the absence of conditioning on node labels when making predictions for edges in IMP++. REWEIGHT and TDE/TE do not require retraining IMP++.

GAN. To train the generator G and discriminators D of a GAN, we generally follow hyper-parameters suggested by SPADE (Park et al., 2019). In particular, we use Spectral Norm (Miyato et al., 2018) for D, Batch Norm (Ioffe and Szegedy, 2015) for G, and TTUR (Heusel et al., 2017) with learning rates of 1e-4 and 2e-4 for G and D respectively.

Perturbation methods (§ 5.3.1). We found that perturbing $L = 20\%$ nodes works well across the methods, which we use in all our experiments. For NEIGH we use top-k=10 as a compromise between too limited diversity and plausibility. For GRAPHN, we set top-k=5, as the method enables larger diversity even with very small top-k. To train the GAN-based models with GRAPHN, we use frequency threshold $\alpha = [2, 5, 10, 20]$. In addition to the proposed perturbation methods, we also consider so called ORACLE-Zs perturbations. These are created by directly using ZS triplets from the test set (all obtained triplets are the same as ZS triplets, so that zero-shot hit rate is 100%). We also evaluate ORACLE-Zs $+\hat{B}$, which in addition to exploiting test ZS triplets, uses bounding boxes from the test samples corresponding to the resulted ZS triplets. ORACLE-Zs-based results are an upper bound estimate of ZS recall, highlighting the challenging nature of the task.

Evaluation. Following prior work (Xu et al., 2017; Zellers et al., 2018; Knyazev et al., 2020; Tang et al., 2020), we focus our evaluation on two standard SGG tasks: scene graph classification (**SGCls**) and predicate classification (**PredCls**), using recall (R@K) metrics. Unless otherwise stated, we report results with K=100 for SGCls and K=50 for PredCls, since the latter is an easier task with saturated results for K=100. We compute recall *without* the graph constraint in Table 5.1, since it is a less noisy metric (Knyazev et al., 2020). We emphasize performance metrics that focus on the ability to recognize rare and novel visual relationship compositions (Knyazev et al., 2020; Tang et al., 2020; Suhail et al., 2021): **zero-shot** and **10-shot** recalls. In Tables 5.1 and 5.2, the mean and standard deviations of 3 runs (random seeds) are reported.

5.4.2 Results

Main SGG results (Table 5.1). First, we compare the baseline IMP++ to our GAN-based model trained *without* and *with* perturbation methods. Even without any perturbations, the GAN-based model significantly outperforms IMP++, especially on the 100-shot and all-shot recalls. GANs with simple perturbation strategies, RAND (as in (Wang et al., 2019b)) and NEIGH, improve on

Table 5.2: ZS recall results on VG using the graph constraint evaluation. †The results are obtained with a more advanced feature extractor and, thus, are not directly comparable.

Model	SGCls		PredCls	
	zsR@50	zsR@100	zsR@50	zsR@100
Freq (Zellers et al., 2018)	0.0	0.0	0.1	0.1
KERN (Chen et al., 2019a)	–	1.5	3.9	–
VCTree† (Tang et al., 2020)	1.9	2.6	10.8	14.3
NM (Zellers et al., 2018)	1.1	1.7	6.5	9.5
NM† (Tang et al., 2020)	2.2	3.0	10.9	14.5
NM, TDE† (Tang et al., 2020)	3.4	**4.5**	14.4	18.2
NM, EBM† (Suhail et al., 2021)	1.3	–	4.9	–
NM++ (Knyazev et al., 2020)	1.8±0.1	2.3±0.1	10.2±0.1	13.4±0.3
NM++, GAN+GraphN	2.5±0.1	3.1±0.1	14.2±0.0	17.4±0.3
IMP+ (Xu et al., 2017; Zellers et al., 2018)	2.5	3.2	14.5	17.2
IMP+, EBM† (Suhail et al., 2021)	3.7	–	18.6	–
IMP++ (Knyazev et al., 2020)	3.5±0.1	4.2±0.2	18.3±0.4	21.2±0.5
IMP++, TDE	3.5±0.1	4.3±0.1	18.5±0.3	21.5±0.3
IMP++, GAN+GraphN	3.7±0.1	4.4±0.1	19.1±0.3	21.8±0.4
IMP++, GAN+GraphN (max)	**3.8**	**4.5**	**19.5**	**22.4**

zero-shots, but at a drop in the 100-shot and all-shot recalls. GANs with GraphN further improve ZS and 10-shot recalls, but compared to Rand and Neigh, also show high recalls on the 100-shots and all-shots.

For GraphN, there is a connection between the SGG recall results (Table 5.1) and triplet hit rates (Fig. 5.6) for different values of the threshold α. Specifically, GraphN with lower α values upsamples more of the rare compositions leading to higher ZS and 10-shot *hit rate* (Fig. 5.6 a,b) and, as a result, higher ZS and 10-shot *recalls* (Table 5.1). GraphN with higher α values upsamples more of the frequent compositions leading to higher 100-shot and all-shot *hit rates* (Fig. 5.6 c,d) and, as a result, higher 100-shot and all-shot *recalls*. Compared to Rand and Neigh, the compositions obtained using GraphN have higher triplet hit rates due to better respecting the graph structure and dataset statistics. As a result, GraphN shows overall better recalls in SGG, even approaching the Oracle-Zs model (Table 5.1). Devising a perturbation strategy universally strong across all metrics is challenging. Neigh can be viewed as such an attempt, which shows average hit rates for all test subsets, but lower performance in all SGG metrics.

Among the alternatives to our GAN approach, Reweight improves on zero-shots, 10-shots and mean recall (SGCls-mR) (Table 5.1). However it downweights the class scores of frequent predicates, which directly degrades 100-shot and all-shot recalls. Resample underperforms on all metrics except for SGCls-mR. The main limitation of Resample is that when we resample images with rare predicates/triplets, those images are likely to contain annotations of frequent predicates/triplets. Another method, TDE (Tang et al., 2020), only debiases the predicates similarly to Reweight and Resample-predicates. So, it may benefit little in recognizing ZS triplets

Table 5.3: Evaluation of generated (fake) node feature using the metrics of "similarity" between two distributions X and Y (Kynkäänniemi et al., 2019; Naeem et al., 2020). The same held-out set of real test features ($Y \sim V$) is used as the reference distribution in all cases. The percentage in the superscripts denotes a relative drop of the average metric when switching from test to test-zs conditioning. For all metrics, higher is better.

Distribution X	Fidelity (realism)		Diversity		Avg
	Precision	Density	Recall	Coverage	
Real test	0.74	1.02	0.75	0.97	0.87
Real test-zs	0.66	0.99	0.70	0.94	$0.82^{-6\%}$
GAN: Fake test	0.55	0.77	0.42	0.82	0.64
GAN: Fake test-zs	0.47	0.60	0.41	0.75	$0.56^{-13\%}$

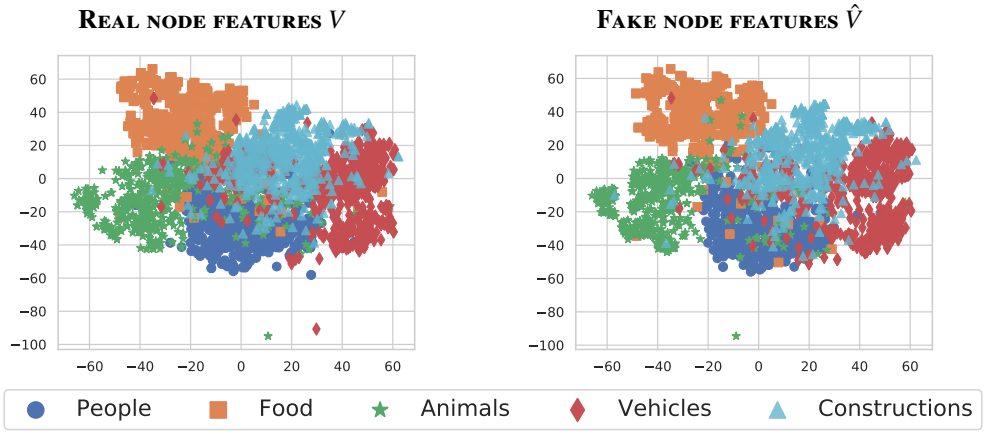


Figure 5.7: Real *vs* generated node features plotted using t-SNE.

such as (`cup`, `on`, `surfboard`), because the predicate 'on' is the frequent one. ZS compositions with such frequent predicates are abundant in VG (Fig. 5.1). Thus, debiasing only the predicates fundamentally limits TDE's performance. In contrast, our GAN method does not suffer from this limitation, since we perturb scene graphs aiming to increase *compositional diversity*, not merely the frequency of rare predicates. As a result, our GAN method improves on all metrics, *especially* on ZS (in relative terms).

Comparison to other SGG works (Table 5.2). Our GAN approach also improves ZS recall (zsR) of other SGG models, namely NM++. For example in PredCls, GAN+GraphN improves zsR of NM++ by 4 percentage points. Compared to the other previous methods presented in Table 5.2, we obtain competitive ZS results on par or better with TDE (Tang et al., 2020) and recent EBM (Suhail et al., 2021). However, it is hard to directly compare to the results reported in (Tang et al., 2020; Suhail et al., 2021) due to the different object detectors and potential implementation discrepancies.

Evaluation of generated visual features. We evaluate the quality of generated features of our GAN trained with GraphN by comparing the generated (fake) features to the real ones. To obtain fake node features $\hat{V}$, we condition our GAN on test SGs. To obtain real node features

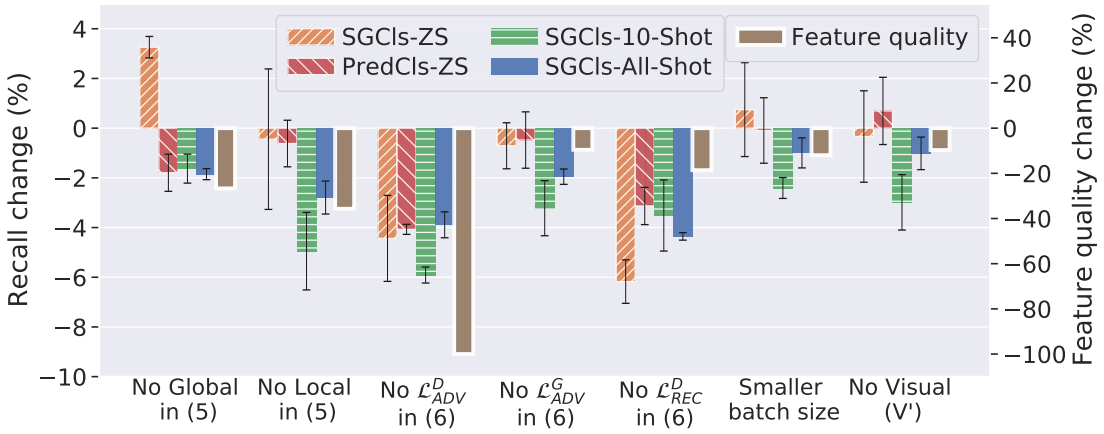


Figure 5.8: Ablations of our GAN model on SGG and feature quality metrics. Error bars denote standard deviation. For feature quality the average metric on the test-zs SGs from Table 5.3 is used.

V, we apply the pretrained object detector to test images as described in § 5.3.1. First, for the qualitative evaluation of node features, we group features based on the object category's super-type, *e.g.* 'people' includes all features of 'man', 'woman', 'person', etc. When projected on a 2D space using t-SNE (Van der Maaten and Hinton, 2008), the fake features $\hat{V}$ generated using our GAN are clustered similarly to the real features V (Fig. 5.7). Therefore, qualitatively our GAN generates realistic and diverse features given a scene graph.

Second, we evaluate GAN features quantitatively. For that purpose, we follow (DeVries et al., 2020) and use Precision, Recall (Kynkäänniemi et al., 2019) and Density, Coverage (Naeem et al., 2020) metrics. These metrics compare the manifolds spanned by real and fake features and do not require any labels. We consider two cases: conditioning our GAN on test SGs and test zero-shot (test-zs) SGs. The motivation is similar to (Casanova et al., 2020): understand if novel compositions confuse the GAN and lead to poor features, that in our context may result in poor training of the main model F. Indeed, the features generated conditioned on test-zs SGs significantly degrade in quality compared to test SGs, especially in terms of fidelity (Table 5.3). This result suggests that it is more challenging to produce realistic features for more rare compositions limiting our approach (see § 5.4.3). The same qualitative and quantitative experiments for edge features $(E, \hat{E})$ and global features $(H, \hat{H})$ confirm our results: (1) when conditioned on test SGs, the generated features are realistic and diverse; (2) conditioning on more rare compositions degrades feature quality.

Ablations (Figure 5.8). We also performed ablations to determine the effect of the proposed GAN losses (5.6) and other design choices on the (i) SGG performance and (ii) quality of generated features. As a reference model, we use our GAN model without any perturbations. In general, all ablated GANs degrade both in (i) and (ii) with correlated drops between (i) and (ii). So by improving generative models in future work, we can expect larger SGG gains. One exception is

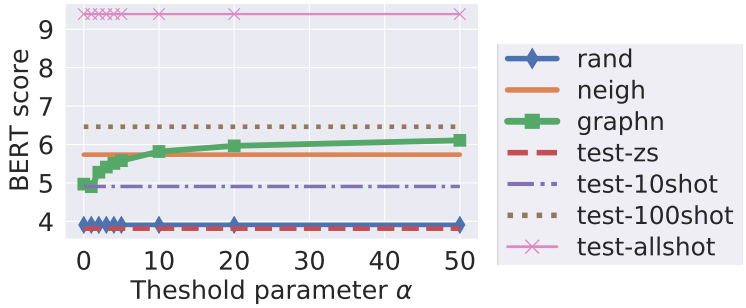


Figure 5.9: Semantic plausibility (as per BERT) depending on α. These results should be interpreted with caution, because: (1) the variance of scores is very high (not shown); (2) in the zero- and few-shot test subsets the graphs are significantly smaller, which affects the amount of contextual information available to BERT.

the GAN without the global terms in (5.5), which performed better on zero-shots despite having lower feature quality. This might be explained as some regularization effect. We also found that this model did not combine well with perturbations.

Evaluating the quality of SG perturbations. We show examples of SG perturbations in Fig. 5.10. In case of RAND, most of the created triplets are implausible as a result of random perturbations. NEIGH leads to very likely compositions, but less often provides rare plausible compositions. In contrast, GRAPHN can create plausible compositions that are rare or more frequent depending on α.

We also analyzed the quality of real and perturbed SGs using the BERT-based metric (§ 5.3.2). We found that the overall test set has on average the highest BERT scores, while lower-shot subsets gradually decrease in "semantic plausibility", which aligns with our intuition. We then perturbed all nodes of all test SGs using our perturbation strategies. Surprisingly, real test-zs SGs have very low plausibility close to RAND-based SGs. NEIGH produces SGs of plausibility between real 10-shot and 100-shot SGs. In contrast, with GRAPHN we can gradually slide between low and high plausibility, which enabled better SGG results. The BERT scores, however, are not tied to the VG dataset. So, semantic plausibility per BERT may be different from the likelihood per VG.

5.4.3 Limitations

Our method is limited in three main aspects. **First**, we rely on a pretrained object detector to extract visual features. Without generating augmentations all the way to the images — in order to update the detector on rare compositions — it is hard to obtain significantly stronger performance. While augmentations in the feature space can be effective (DeVries and Taylor, 2017a; Verma et al., 2019), their adoption for large-scale out-of-distribution generalization is underexplored. **Second**, by making a simplification and keeping GT bounding boxes for perturbed scene graphs, we limit (1)

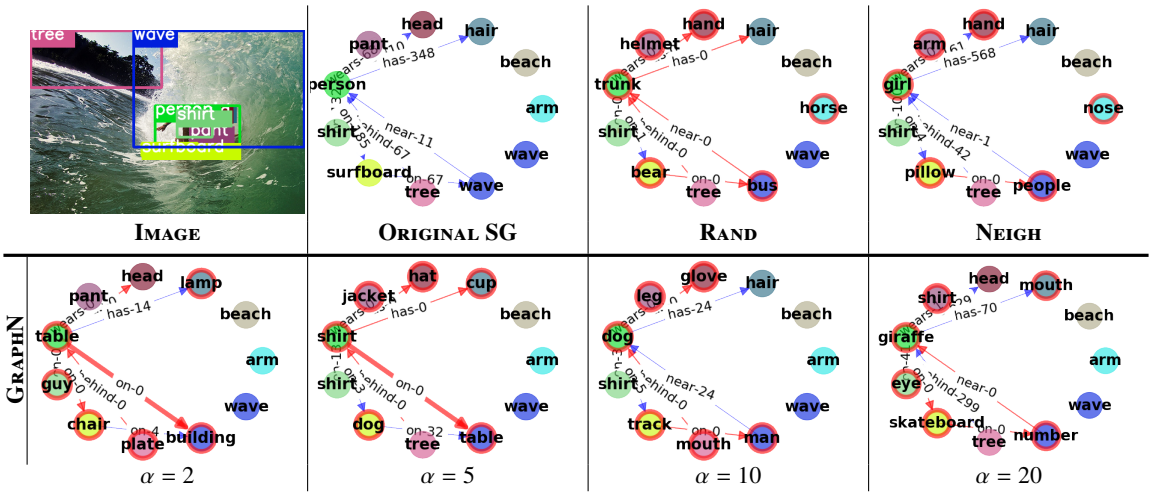


Figure 5.10: Examples of perturbations (nodes in red) applied to a scene graph. The numbers on edges denote the count of triplets in the training set and a thick red arrow denotes matching a ZS triplet.

the amount of perturbations we can make (if we permit many nodes to be perturbed, then it is hard to expect the same layout), and (2) the diversity of spatial compositions, which might be an important aspect of compositional generalization. We attempted to verify that using ORACLE-ZS perturbations, which are created by directly using ZS triplets from the test set. Using ORACLE-ZS with GT bounding boxes (our default setting) surprisingly does not result in large improvements. However, when we replace GT boxes with the ones taken from the corresponding samples of the test set, the results improve significantly. This demonstrates that: (1) our GAN model may benefit from reliable bounding box prediction (e.g. (Hong et al., 2018)); (2) GRAPHN perturbations are already effective (close to ORACLE-ZS) and improving the results further by relying solely on perturbations is challenging. **Third**, the quality of generated features, especially, for novel and rare compositions is currently limited, which is also carefully analyzed in (Casanova et al., 2020). Addressing this challenge can further improve results both of ORACLE-ZS and non-ORACLE-ZS models.

5.5 Conclusion

We focus on the compositional generalization problem within the scene graph generation task. Our GAN-based augmentation approach can be used with different SGG models and can improve their zero-, few- and all-shot SGG results. To obtain better SGG results using our augmentations, it is important to rely on the structure of scene graphs and tune the augmentation parameters towards a specific SGG metric. Our evaluation confirmed that our augmentations provide plausible compositions and the generator generally produces high-fidelity and diverse features enabling gains in SGG.

6 Parameter Prediction for Unseen Deep Architectures

Prologue

Context. Before deep learning it was often necessary to manually design features (*e.g.* SIFT (Lowe, 2004)). Now, we can simply learn features using gradient descent algorithms like SGD. However, SGD itself is manually designed and thus has its own limitations similar to manual feature engineering. In particular, SGD is computationally expensive and requires expertise to tune it. In addition, SGD does not accumulate the knowledge of previous optimizations, rather it starts from scratch. While replacing manually designed features with the learnable ones is extremely successful (Krizhevsky et al., 2012), replacing SGD appears to be more challenging. So far, learnable methods based on recurrent neural networks have been explored to tackle this task (Andrychowicz et al., 2016). However, these methods produce optimizers that are inefficient as SGD, motivating research into alternative approaches.

Contributions. We build on Graph HyperNetworks (GHNs) (Zhang et al., 2018a) that take a neural network architecture and output its trained parameters in a single forward pass. To train and evaluate GHNs, we release the DeepNets-1M dataset of neural architectures for vision tasks: CIFAR-10 and ImageNet. We significantly improve on GHNs in terms of design and generalization ability. For example, we can take a common ResNet-50 neural network and predict all its parameters using our GHN-2 in less than a second even on a CPU. This ResNet-50 achieves 58.6% on CIFAR-10 without any training, a remarkable performance since our GHN-2 has never observed models at the same scale and connectivity.

Recent works. Cai et al. (2019b) provide a method to obtain performant ImageNet parameters for diverse networks. However, all their networks are derived from a shared "supernet" based on MobileNet-v3 (Howard et al., 2019). As a result, this method cannot be applied to arbitrary networks. Other works are based on meta-optimizers (Ravi and Larochelle, 2016; Metz et al., 2020; Wichrowska et al., 2017) and, as inherent to iterative optimizers, are computationally inefficient. HyperTransformers can predict parameters for small-scale networks and can generalize across tasks (Anonymous, 2021). Scaling up HyperTransformers and connecting them with GHNs may enable parameter prediction across tasks.

6.1 Introduction

Consider the problem of training deep neural networks on large annotated datasets, such as ImageNet (Russakovsky et al., 2015). This problem can be formalized as finding optimal parameters for a given neural network a, parameterized by $\mathbf{w}$, w.r.t. a loss function $\mathcal{L}$ on the dataset

$\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ of inputs $\mathbf{x}_i$ and targets y_i:

$$\arg\min_{\mathbf{w}} \sum_{i=1}^{N} \mathcal{L}(f(\mathbf{x}_i; a, \mathbf{w}), y_i), \tag{6.1}$$

where $f(\mathbf{x}_i; a, \mathbf{w})$ represents a forward pass. (6.1) is usually minimized by iterative optimization algorithms – e.g. SGD (Ruder, 2016) and Adam (Kingma and Ba, 2015) – that converge to performant parameters $\mathbf{w}_p$ of the architecture a. Despite the progress in improving the training speed and convergence (Huang et al., 2016; Brock et al., 2017a; Choi et al., 2019; Ioffe and Szegedy, 2015), obtaining $\mathbf{w}_p$ remains a bottleneck in large-scale machine learning pipelines. For example, training a ResNet-50 (He et al., 2016) on ImageNet can take many GPU hours (NVIDIA, 2021). With the ever growing size of networks (Brown et al., 2020) and necessity of training the networks repeatedly (e.g. for hyperparameter or architecture search), the classical process of obtaining $\mathbf{w}_p$ is becoming computationally unsustainable (Strubell et al., 2019; Cai et al., 2019a; Thompson et al., 2020).

A new parameter prediction task. When optimizing the parameters for a *new* architecture a, typical optimizers disregard past experience gained by optimizing other nets. However, leveraging this past experience can be the key to reduce the reliance on iterative optimization and, hence the high computational demands. To progress in that direction, we propose a new task where iterative optimization is replaced with a *single forward pass* of a hypernetwork (Ha et al., 2016) $H_{\mathcal{D}}$. To tackle the task, $H_{\mathcal{D}}$ is expected to leverage the knowledge of how to optimize *other* networks $\mathcal{F}$. Formally, the task is to predict the parameters of an *unseen* architecture $a \notin \mathcal{F}$ using $H_{\mathcal{D}}$, parameterized by θ_p: $\hat{\mathbf{w}}_p = H_{\mathcal{D}}(a; \theta_p)$. The task is constrained to a dataset $\mathcal{D}$, so $\hat{\mathbf{w}}_p$ are the predicted parameters for which the test set performance of $f(\mathbf{x}; a, \hat{\mathbf{w}}_p)$ is similar to the one of $f(\mathbf{x}; a, \mathbf{w}_p)$. For example, we consider CIFAR-10 (Krizhevsky, 2009) and ImageNet image classification datasets $\mathcal{D}$, where the test set performance is classification accuracy on test images.

Approaching our task. A straightforward approach to expose $H_{\mathcal{D}}$ to the knowledge of how to optimize other networks is to train it on a large training set of $\{(a_i, \mathbf{w}_{p,i})\}$ pairs, however, that is prohibitive[1]. Instead, we follow the bi-level optimization paradigm common in meta-learning (Hospedales et al., 2020; Andrychowicz et al., 2016; Ravi and Larochelle, 2016), but rather than iterating over M tasks, we iterate over M training architectures $\mathcal{F} = \{a_i\}_{i=1}^M$:

$$\arg\min_{\theta} \sum_{j=1}^{N} \sum_{i=1}^{M} \mathcal{L}\Big(f\Big(\mathbf{x}_j; a_i, H_{\mathcal{D}}(a_i; \theta)\Big), y_j\Big). \tag{6.2}$$

By optimizing (6.2), the hypernetwork $H_{\mathcal{D}}$ gradually gains knowledge of how to predict performant parameters for training architectures. It can then leverage this knowledge at test time – when

[1]Training a single network a_i can take several GPU days and thousands of trained networks may be required.

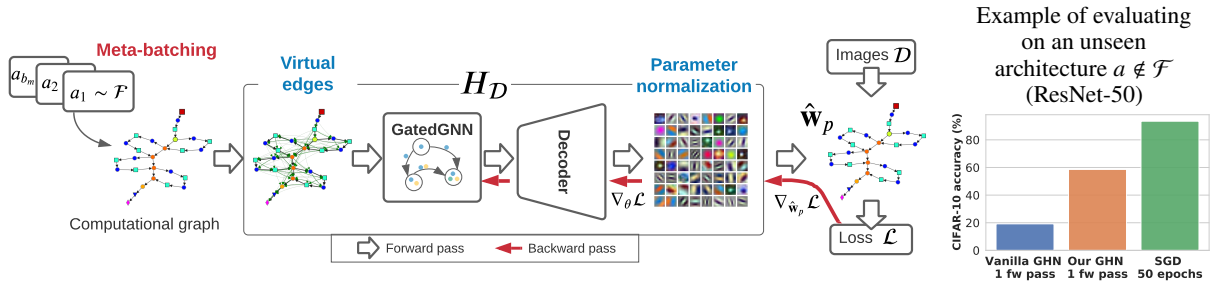
predicting parameters for *unseen* architectures. To approach the problem in (6.2), we need to design the network space $\mathcal{F}$ and $H_{\mathcal{D}}$. For $\mathcal{F}$, we rely on the previous design spaces for neural architectures (Liu et al., 2018b) that we extend in two ways: the ability to sample distinct architectures and an expanded design space that includes diverse architectures, such as ResNets and Visual Transformers (Dosovitskiy et al., 2020). Such architectures can be fully described in the form of computational graphs (Fig. 6.1). So, to design the hypernetwork $H_{\mathcal{D}}$, we rely on recent advances in machine learning on graph-structured data (Kipf and Welling, 2017; Velickovic et al., 2018; Dwivedi et al., 2020; Zhang et al., 2018a). In particular, we build on the Graph HyperNetworks method (GHNs) (Zhang et al., 2018a) that also optimizes (6.2). However, GHNs do not aim to predict large-scale performant parameters as we do in this work, which motivates us to improve on their approach.

By designing our diverse space $\mathcal{F}$ and improving on GHNs, we boost the accuracy achieved by the predicted parameters on *unseen* architectures to 77% (top-1) and 48% (top-5) on CIFAR-10 (Krizhevsky, 2009) and ImageNet (Russakovsky et al., 2015), respectively. Surprisingly, our GHN shows good out-of-distribution generalization and predicts good parameters for architectures that are much larger and deeper compared to the ones seen in training. For example, we can predict all 24 million parameters of ResNet-50 in less than a second either on a GPU or CPU achieving ~60% on CIFAR-10 without any gradient updates (Fig 6.1, (b)).

Overall, our framework and results pave the road toward a new and significantly more efficient paradigm for training networks. Our **contributions** are as follows: (**a**) we introduce the novel task of predicting performant parameters for diverse feedforward neural networks with a single hypernetwork forward pass; (**b**) we introduce DeepNets-1M – a standardized benchmark with in-distribution and out-of-distribution architectures to track progress on the task (§ 6.3); (**c**) we define several baselines and propose a GHN model (§ 6.4) that performs surprisingly well on CIFAR-10 and ImageNet (§ 6.5.1); (**d**) we show that our model learns a strong representation of neural network architectures (§ 6.5.2), and our model is useful for initializing neural networks (§ 6.5.3). Our DeepNets-1M dataset, trained GHNs and code is available at https://github.com/facebookresearch/ppuda.

6.2 Background

We start by providing a brief background about the network design spaces leveraged in the creation of our DeepNets-1M dataset of neural architectures described in § 6.3. We then cover elements of graph hypernetworks that we leverage when designing our specific GHN $H_{\mathcal{D}}$ in § 6.4.



(a) (b)

Figure 6.1: **(a)** Overview of our GHN model (§ 6.4) trained by backpropagation through the predicted parameters ($\hat{\mathbf{w}}_p$) on a given image dataset and our DeepNets-1M dataset of architectures. Colored captions show our key improvements to vanilla GHNs (§ 6.2.2). The red one is used only during training GHNs, while the blue ones are used both at training and testing time. The computational graph of a_1 is visualized as described in Table 6.1. **(b)** Comparing classification accuracies when all the parameters of a ResNet-50 are predicted by GHNs versus when its parameters are trained with SGD (see full results in § 6.5).

6.2.1 Network design space of DARTS

DARTS (Liu et al., 2018b) is a differentiable NAS framework. For image classification tasks such as those considered in this work, its networks are defined by four types of building blocks: *stems*, *normal cells*, *reduction cells*, and *classification heads*. Stems are fixed blocks of convolutional operations that process input images. The normal and reduction cells are the main blocks of architectures and are composed of: 3×3 and 5×5 separable convolutions, 3×3 and 5×5 dilated separable convolutions, 3×3 max pooling, 3×3 average pooling, identity and zero (to indicate the absence of connectivity between two operations). Finally, the classification head defines the network output and is built with a global pooling followed by a single fully connected layer.

Typically, DARTS networks have one stem block, 14-20 cells, and one classification head, altogether forming a deep computational graph. The reduction cells, placed only at 1/3 and 2/3 of the total depth, decrease the spatial resolution and increase the channel dimensionality by a factor of 2. Summation and concatenation are used to aggregate outputs from multiple operations within each cell. To make the channel dimensionalities match, 1×1 convolutions are used as needed. All convolutional operations use the ReLU-Conv-Batch Norm (BN) (Ioffe and Szegedy, 2015) order. Overall, DARTS enables defining strong architectures that combine many principles of manual (Simonyan and Zisserman, 2014b; He et al., 2016; Xie et al., 2017; Huang et al., 2017) and automatic (Zhang et al., 2018a; Zoph and Le, 2016; Zoph et al., 2018; Liu et al., 2018a; Real et al., 2019; Chen et al., 2019c; Howard et al., 2019) design of neural architectures. While DARTS learns the optimal task-specific cells, the framework can be modified to permit sampling randomly-structured cells. We leverage this possibility for the DeepNets-1M construction in § 6.3.

6.2.2 Graph hypernetwork: GHN-1

Representation of architectures. GHNs (Zhang et al., 2018a) directly operate on the computational graph of a neural architecture a. Specifically, a is a directed acyclic graph (DAG), where nodes $V = \{v_i\}_{i=1}^{|V|}$ are operations (e.g. convolutions, fully-connected layers, summations, etc.) and their connectivity is described by a binary adjacency matrix $\mathbf{A} \in \{0,1\}^{|V|\times|V|}$. Nodes are further characterized by a matrix of initial node features $\mathbf{H}^0 = [\mathbf{h}_1^0, \mathbf{h}_2^0, ..., \mathbf{h}_{|V|}^0]$, where each $\mathbf{h}_v^0$ is a one-hot vector representing the operation performed by the node. We also use such a one-hot representation for $\mathbf{H}^0$, but in addition encode the shape of parameters associated with nodes.

Design of the graph hypernetwork. In (Zhang et al., 2018a), the graph hypernetwork $H_{\mathcal{D}}$ consists of three key modules. The first module takes the input node features $\mathbf{H}^0$ and transforms them into d-dimensional node features $\mathbf{H}^1 \in \mathbb{R}^{|V|\times d}$ through an embedding layer. The second module takes $\mathbf{H}^1$ together with $\mathbf{A}$ and feeds them into a specific variant of the gated graph neural network (GatedGNN) (Li et al., 2015). In particular, their GatedGNN mimics the canonical order π of node execution in the forward (fw) and backward (bw) passes through a computational graph. To do so, it sequentially traverses the graph and performs iterative message passing operations and node feature updates as follows:

$$\forall t \in [1,...,T] : \Big[\forall \pi \in [\text{fw}, \text{bw}] : \Big(\forall v \in \pi : \mathbf{m}_v^t = \sum_{u \in \mathcal{N}_v^\pi} \text{MLP}(\mathbf{h}_u^t), \;\; \mathbf{h}_v^t = \text{GRU}(\mathbf{h}_v^t, \mathbf{m}_v^t)\Big)\Big], \quad (6.3)$$

where T denotes the total number of forward-backward passes; $\mathbf{h}_v^t$ corresponds to the features of node v in the t-th graph traversal; $\text{MLP}(\cdot)$ is a multi-layer perceptron; and $\text{GRU}(\cdot)$ is the update function of the Gated Recurrent Unit (Cho et al., 2014). In the forward propagation (π = fw), $\mathcal{N}_v^\pi$ corresponds to the incoming neighbors of the node defined by $\mathbf{A}$, then in the backward propagation (π = bw) it similarly corresponds to the outgoing neighbors of the node. The last module uses the GatedGNN output hidden states $\mathbf{h}_v^T$ to condition a decoder that produces the parameters $\hat{\mathbf{w}}_p^v$ (e.g. convolutional weights) associated with each node. In practice, to handle different parameter dimensionalities per operation type, the output of the hypernetwork is reshaped and sliced according to the shape of parameters in each node. We refer to the model described above as GHN-1 (Fig. 6.1). Further subtleties of implementing this model in the context of our task can be found in our source code.

How do graph hypernetworks learn? It may be not obvious why GHNs allow us to optimize such a difficult objective of learning to predict performant parameters (6.2). In particular, perhaps the most surprising working principle behind training GHNs is that we sample a new architecture

for each training iteration. This comes as a striking contrast to training a standard machine learning objective (6.1) that requires thousands or millions optimization steps just for a single architecture. How is it possible that GHNs learn from just a single optimization step on each architecture?

There is no clear answer to this question in the literature so far. To provide a high-level answer we can draw an analogy between the *distribution of images* used to train a neural network in (6.1) and the *distribution of architectures* used to train GHNs in (6.2). In (6.1), when we train the parameters of a single network using SGD, we sample new images for each training iteration. While we can run optimization for the same images for more than one iteration, this is considered to be poor practice that will likely lead to overfitting. So typically we sample new images for each training iteration, however it is critical that at each iteration the images are drawn from the same distribution. This way the neural network gradually captures the regularities in the distribution of images to make better predictions in the subsequent steps. If we sample drastically different images for each training iteration (*e.g.* natural images in the first iteration, medical images in the second iteration, then some binary QR codes, etc.), then (6.1) would be hard or impossible to optimize. The same principle may be the key to enable training GHNs. At each training iteration, a GHN slightly improves its parameters (by gradient descent) w.r.t. a single architecture sampled from some distribution. If there are regularities in this distribution and the GHN can capture them, then the improvements of GHN parameters in the previous steps can result in improvements for the new architectures in the next steps as long as all the architectures are sampled from the same distribution. In terms of gradient descent, the direction of parameter updates of the GHN computed for a single architecture can be useful for the entire distribution of architectures. By following this direction using gradient descent, the GHN can gradually improve over time on the entire distribution. These improvements depend heavily on the shape of the training distribution. We need to make sure that this distribution has strong regularities to enable training of GHNs, but at the same time has diverse enough samples to enable generalization (prediction of performant parameter for unseen architectures). The design of such a distribution is described in the next section.

6.3 DeepNets-1M

The network design space of DARTS is limited by the number of unique operations that compose cells, and the low variety of stems and classification heads. Thus, many architectures are not realizable within this design space, including: VGG (Simonyan and Zisserman, 2014b), ResNets (He et al., 2016), MobileNet (Howard et al., 2019) or more recent ones such as Visual Transformer (ViT) (Dosovitskiy et al., 2020) and Normalization-free networks (Brock et al., 2021a,b). Furthermore, DARTS does not define a procedure to sample random architectures. By addressing these two limitations we aim to expose our hypernetwork to diverse training architectures and permit

its evaluation on common architectures, such as ResNet-50. We hypothesize that increased training diversity can improve hypernetworks' generalization to unseen architectures making it more competitive to iterative optimizers.

Extending the network design space. We extend the set of possible operations with non-separable 2D convolutions[2], Squeeze&Excite (SE[3]) (Hu et al., 2018) and Transformer-based operations (Vaswani et al., 2017; Dosovitskiy et al., 2020): multihead self-attention (MSA), positional encoding and layer norm (LN) (Ba et al., 2016). Each node (operation) in our graphs has two attributes: *primitive type* (e.g. convolution) and *shape* (e.g. 3×3×512×512). Overall, our extended set consists of 15 primitive types (Table 6.1). We also extend the diversity of the generated architectures by introducing VGG-style classification heads and ViT stems. Finally, to further increase architectural diversity, we allow the operations to not include batch norm (BN) (Ioffe and Szegedy, 2015) and permit networks without channel width expansion (e.g. as in (Dosovitskiy et al., 2020)).

Architecture generation process. We generate different subsets of architectures (see the description of each subset in the next two paragraphs and in Table 6.1). For each subset depending on its purpose, we predefine a range of possible model depths (number of cells), widths and number of nodes per cell. Then, we sample a stem, a normal and reduction cell and a classification head. The internal structure of the normal and reduction cells is defined by uniformly sampling from all available operations. Due to a diverse design space it is extremely unlikely to sample the same architecture multiple times, but we ran a sanity check using the Hungarian algorithm (Kuhn, 1955) to confirm that.

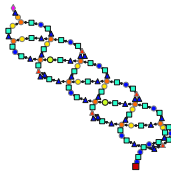
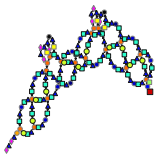
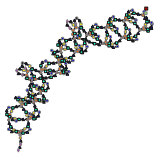
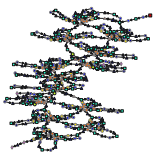
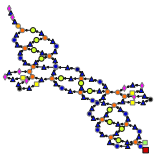
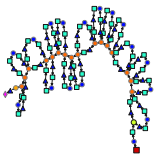
In-distribution (ID) architectures. We generate a training set of $|\mathcal{F}| = 10^6$ architectures and validation/test sets of 500/500 architectures that follow the same generation rules and are considered to be ID samples. However, training on large architectures can be prohibitive, e.g. in terms of GPU memory. Thus, in the training set we allow the number of channels and, hence the total number of parameters, to be stochastically defined given computational resources. For example, to train our models we upper bound the number of parameters in the training architectures to around 3M by sampling fewer channels if necessary. In the evaluation sets, the number of channels is fixed. Therefore, this pre-processing step prior to training results in some distribution shift between the training and the validation/test sets. However, the shift is not imposed by our dataset.

Out-of-distribution (OOD) architectures. We generate five OOD test sets that follow

[2]Non-separable convolutions have weights of e.g. shape 3×3×512×512 as in ResNet-50. NAS works, such as DARTS and GHN, avoid such convolutions, since the separable ones (Sifre and Mallat, 2014) are more efficient. Non-separable convolutions are nevertheless common in practice and can often boost the downstream performance.

[3]SE is common in many efficient networks (Howard et al., 2017; Cai et al., 2019a).

Table 6.1: Examples of computational graphs (visualized using NetworkX (Hagberg et al., 2008)) in each split and their key statistics, to which we add the average degree and average shortest path length often used to measure local and global graph properties respectively (Barrat et al., 2004; You et al., 2020a). In the visualized graphs, a node is one of the 15 primitives coded with markers shown at the bottom, where they are sorted by the frequency in the training set. For visualization purposes, a blue triangle marker differentiates a 1×1 convolution (equivalent to a fully-connected layer over channels) from other convolutions, but its primitive type is still just convolution. *Computed based on CIFAR-10.

	IN-DISTRIBUTION		OUT-OF-DISTRIBUTION				
							
	TRAIN	VAL/TEST	WIDE	DEEP	DENSE	BN-FREE	RESNET/VIT
#graphs	10^6	500/500	100	100	100	100	1/1
#cells	4-18	4-18	4-18	**10-36**	4-18	4-18	16/12
#channels	16-128	32-128	**128-1216**	32-208	32-240	32-336	64/128
#nodes ($\|V\|$)	21-827	33-579	33-579	**74-1017**	**57-993**	33-503	161/114
% w/o BN	3.5%	4.1%	4.1%	2.0%	5.0%	**100%**	0%/**100%**
#params(M)*	0.01-3.1	2.5-35	**39-101**	2.5-15.3	2.5-8.8	2.5-7.7	**23.5**/1.0
avg degree	2.3±0.1	2.3±0.1	2.3±0.1	2.3±0.1	**2.4**±0.1	**2.4**±0.1	2.2/2.3
avg path	14.5±4.8	14.5±4.9	14.7±4.9	**26.2**±9.3	15.1±4.1	10.0±2.8	11.2/10.7

marker / primitive	conv	BN	sum	bias	group conv	con-cat	dilat. gr. conv	LN	max pool	avg pool	MSA	SE	input	glob avg	pos enc
fraction in TRAIN (%)	36.3	25.5	11.1	6.5	5.1	3.8	2.5	2.5	1.8	1.7	1.2	1.0	0.5	0.5	0.2

different generation rules. In particular, we define WIDE and DEEP sets that are of interest due the stronger downstream performance of such nets in large-scale tasks (Golubeva et al., 2020; Zagoruyko and Komodakis, 2016; Brown et al., 2020). These nets are often more challenging to train for fundamental (Nguyen and Hein, 2017; Srivastava et al., 2015) or computational (Hooker, 2020) reasons, so predicting their parameters might ease their subsequent optimization. We also define the DENSE set, since networks with many operations per cell and complex connectivity are underexplored in the literature despite their potential (Huang et al., 2017). Next, we define the BN-FREE set that is of interest due to BN's potential negative side-effects (Galloway et al., 2019; Hendrycks and Dietterich, 2019) and the difficulty or unnecessity of using it in some cases (Wu and He, 2018; Qiao et al., 2019; Zhang et al., 2019b; Brock et al., 2021a,b). We finally add the RESNET/VIT set with two predefined image classification architectures: commonly-used ResNet-50 (He et al., 2016) and a smaller 12-layer version of the Visual Transformer (ViT) (Dosovitskiy et al., 2020) that has recently received a lot of attention in the vision community.

6.4 Improved graph hypernetworks: GHN-2

In this section, we introduce our three key improvements to the baseline GHN-1 described in § 6.2.2 (Fig. 6.1). These components are essential to predict stronger parameters on our task. For the empirical validation of the effectiveness of these components see ablation studies in § 6.5.1.

6.4.1 Differentiable normalization of predicted parameters

When training the parameters of a given network from scratch using iterative optimization methods, the initialization of parameters is crucial. A common approach is to use He (He et al., 2015) or Glorot (Glorot and Bengio, 2010) initialization to stabilize the variance of activations across layers of the network. Chang et al. (2019) showed that when the parameters of the network are instead predicted by a hypernetwork, the activations in the network tend to explode or vanish. To address the issue of unstable network activations especially for the case of predicting parameters of diverse architectures, we apply *operation-dependent normalizations* (Table 6.2). We normalize convolutional and fully-connected weights by following the *fan-in* scheme of (He et al., 2015): $\hat{\mathbf{w}}_p^v \sqrt{\beta/(C_{in}\mathcal{H}\mathcal{W})}$, where $C_{in}, \mathcal{H}, \mathcal{W}$ are the number of input channels and spatial dimensions of weights $\hat{\mathbf{w}}_p^v$, respectively; and β is a nonlinearity specific constant following the analysis in (He et al., 2015). The parameters of normalization layers such as BN and LN, as well as biases typically initialized with constants, are normalized by applying a squashing function with temperature T to imitate the empirical distributions of models trained with SGD (see Table 6.2). These are differentiable normalizations, so that they are applied at training (and testing) time.

Table 6.2: Parameter normalizations.

Type of node v	Normalization
Convolutional/fully-connected	$\hat{\mathbf{w}}_p^v \sqrt{\beta/(C_{in}\mathcal{H}\mathcal{W})}$
Normalization weights	$2 \times \text{sigmoid}(\hat{\mathbf{w}}_p^v/T)$
Biases	$\tanh(\hat{\mathbf{w}}_p^v/T)$

6.4.2 Enhancing long-range message propagation

Computational graphs often take the form of long chains (Table 6.1) with only a few incoming/outcoming edges per node. This structure might hinder long-range propagation of information between nodes (Alon and Yahav, 2020). Different approaches to alleviate the long-range propagation problem exist (El Hihi and Bengio, 1996; Liu et al., 2020; Pei et al., 2020), including stacking GHNs in (Zhang et al., 2018a). Instead we adopt simple graph-based heuristics in line with recent

works (You et al., 2019; Yang et al., 2021). In particular, we add *virtual edges* between two nodes v and u and weight them based on the shortest path s_{vu} between them (Fig. 6.2). To avoid interference with the *real* edges in the computational graph, we introduce a separate MLP_{sp} to transform the features of the nodes connected through these virtual edges, and redefine the message passing of (6.3) as:

$$\mathbf{m}_v^t = \sum_{u \in \mathcal{N}_v^\pi} \text{MLP}(\mathbf{h}_u^t) + \sum_{u \in \mathcal{N}_v^{(\text{sp})}} \frac{1}{s_{vu}} \text{MLP}_{\text{sp}}(\mathbf{h}_u^t), \tag{6.4}$$

where $\mathcal{N}_v^{(sp)}$ are neighbors satisfying $1 < s_{vu} \leq s^{(\max)}$, and $s^{(\max)}$ is a hyperparameter. To maintain the same number of trainable parameters as in GHN-1, we decrease MLPs' sizes appropriately.

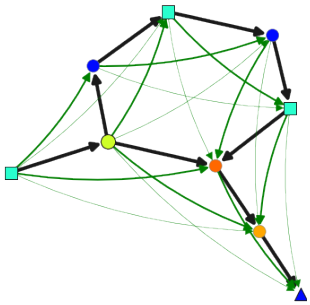


Figure 6.2: Virtual edges (in green) allow for better capture of global context.

6.4.3 Meta-batching architectures during training

GHN-1 updates its parameters θ based on a single architecture sampled for each batch of images (6.2). In vanilla SGD training, larger batches of images often speed up convergence by reducing gradient noise and improve model's performance (Radiuk, 2017). Therefore, we define a meta-batch b_m as the number of architectures sampled per batch of images. Both the parameter prediction and the forward/backward passes through the architectures in a meta-batch can be done in parallel. We then average the gradients across b_m to update the parameters θ of $H_{\mathcal{D}}$: $\nabla_\theta \mathcal{L} = 1/b_m \sum_{i=1}^{b_m} \nabla_\theta \mathcal{L}_i$.

6.5 Experiments

We focus the evaluation of GHN-2 on our parameter prediction task (§ 6.5.1). In addition, we show beneficial side-effects of i) learning a stronger neural architecture representation using GHN-2 in analyzing networks (§ 6.5.2) and ii) predicting parameters for fine-tuning (§ 6.5.3).

Datasets. We use the DeepNets-1M dataset of architectures (§ 6.3) as well as two image classification datasets $\mathcal{D}_1$ (CIFAR-10 (Krizhevsky, 2009)) and $\mathcal{D}_2$ (ImageNet (Russakovsky et al., 2015)). CIFAR-10 consists of 50k training and 10k test images of size 32×32×3 and 10 object categories. ImageNet is a larger scale dataset with 1.28M training and 50k test images of variable

size and 1000 fine-grained object categories. We resize ImageNet images to 224×224×3 following (Liu et al., 2018b; Zhang et al., 2018a). We use 5k/50k training images as a validation set in CIFAR-10/ImageNet and 500 validation architectures of DeepNets-1M for hyperparameter tuning.

Baselines. Our baselines include GHN-1 and a simple MLP that only has access to operations, but not to the connections between them. This MLP baseline is obtained by replacing the GatedGNN with an MLP in our GHN-2. Since GHNs were originally introduced for small architectures of ~ 50 nodes and only trained on CIFAR-10, we reimplement[4] them and scale them up by introducing minor modifications to their decoder that enable their training on ImageNet and on larger architectures of up to 1000 nodes. We use the same hyperparameters to train the baselines and GHN-2.

Iterative optimizers. In the parameter prediction experiments, we also compare our model to standard optimization methods: SGD and Adam (Kingma and Ba, 2015). We use off-the-shelf hyperparameters common in the literature (Zhang et al., 2018a; Liu et al., 2018b; Chen et al., 2019c; Yang et al., 2020; He et al., 2020; Li et al., 2020a). On CIFAR-10, we train evaluation architectures with SGD/Adam, initial learning rate $\eta = 0.025$ / $\eta = 0.001$, batch size $b = 96$ and up to 50 epochs. With Adam, we train only 300 evaluation architectures as a rough estimation of an average performance. On ImageNet, we train them with SGD, $\eta = 0.1$ and $b = 128$, and, for computational reasons (given 1402 evaluation architectures in total), we limit training with SGD to 1 epoch. We have also considered meta-optimizers, such as (Andrychowicz et al., 2016; Ravi and Larochelle, 2016). However, we were unable to scale them to diverse and large architectures of our DeepNets-1M, since their LSTM requires a separate hidden state for every trainable parameter in the architecture. The scalable variants exist (Wichrowska et al., 2017; Metz et al., 2020), but are hard to reproduce without open source code.

Additional experimental details. We follow (Zhang et al., 2018a) and train GHNs with Adam, $\eta = 0.001$ and batch size of 64 images for CIFAR-10 and 256 for ImageNet. We train for up to 300 epochs, except for one experiment in the ablation studies, where we train one GHN with $b_m = 1$ eight times longer, i.e. for 2400 epochs. All GHNs in our experiments use $T = 1$ propagation (6.3), as we found the original $T = 5$ of (Zhang et al., 2018a) to be inefficient and it did not improve the accuracies in our task. GHN-2 uses $s^{(\max)} = 50$ and $b_m = 8$ and additionally uses LN that slightly further improves results. Model selection is performed on the validation sets, but the results in our paper are reported on the test sets to enable their direct comparison.

[4]While source code for GHNs (Zhang et al., 2018a) is unavailable, we appreciate the authors' help in implementing some steps.

6.5.1 Parameter prediction

Experimental setup. We trained our GHN-2 and baselines on the training architectures and training images, i.e. a separate model is trained for CIFAR-10 and ImageNet. According to our DEEPNETS-1M benchmark, we assess whether these models can generalize to unseen in-distribution (ID) and out-of-distribution (OOD) test architectures from our DEEPNETS-1M. We measure this generalization by predicting parameters for the test architectures and computing their classification accuracies on the test images of CIFAR-10 (Table 6.3) and ImageNet (Table 6.4). The evaluation architectures with batch norm (BN) have running statistics, which are not learned by gradient descent (Ioffe and Szegedy, 2015), and hence are not predicted by our GHNs. To alleviate that, we follow (Zhang et al., 2018a) and evaluate the networks with BN by computing per batch statistics with batch size of 64 images.

Results. Despite GHN-2 never observed the test architectures, GHN-2 predicts good parameters for them making the test networks perform surprisingly well on both image datasets (Tables 6.3 and 6.4). Our results are especially strong on CIFAR-10, where some architectures with predicted

Table 6.3: CIFAR-10 results of predicted parameters for unseen ID and OOD architectures of DEEPNETS-1M. Mean (±standard error of the mean) accuracies are reported (random chance ≈10%). †The number of parameter updates.

METHOD	#upd†	ID-TEST		OOD-TEST				
		avg	max	WIDE	DEEP	DENSE	BN-FREE	RESNET/VIT
MLP	1	42.2±0.6	60.2	22.3±0.9	37.9±1.2	44.8±1.1	23.9±0.7	17.7/10.0
GHN-1	1	51.4±0.4	59.9	43.1±1.7	48.3±0.8	51.8±0.9	13.7±0.3	19.2/**18.2**
GHN-2	1	**66.9**±0.3	**77.1**	**64.0**±1.1	**60.5**±1.2	**65.8**±0.7	**36.8**±1.5	**58.6**/11.4
Iterative optimizers (all architectures are ID in this case)								
SGD (1 epoch)	0.5×10^3	46.1±0.4	66.5	47.2±1.1	34.2±1.1	45.3±0.7	18.0±1.1	61.8/34.5
SGD (5 epochs)	2.5×10^3	69.2±0.4	82.4	71.2±0.3	56.7±1.6	67.8±0.9	29.0±2.0	78.2/52.5
SGD (50 epochs)	25×10^3	88.5±0.3	93.1	88.9±1.2	84.5±1.2	87.3±0.8	45.6±3.6	93.5/75.7
Adam (50 epochs)	25×10^3	84.0±0.8	89.5	82.0±1.6	76.2±2.6	84.8±0.4	38.8±4.8	91.5/79.4

Table 6.4: ImageNet results on DEEPNETS-1M. Mean (±standard error of the mean) top-5 accuracies are reported (random chance ≈0.5%). *Estimated on ResNet-50 with batch size 128.

METHOD	#upd	GPU sec.	CPU sec.	ID-TEST		OOD-TEST				
		avg	avg	avg	max	WIDE	DEEP	DENSE	BN-FREE	RESNET/VIT
GHN-1	1	0.3	0.5	17.2±0.4	32.1	15.8±0.9	15.9±0.8	15.1±0.7	0.5±0.0	**6.9**/0.9
GHN-2	1	0.3	0.7	**27.2**±0.6	**48.3**	**19.4**±1.4	**24.7**±1.4	**26.4**±1.2	**7.2**±0.6	5.3/**4.4**
Iterative optimizers (all architectures are ID in this case)										
SGD (1 step)	1	0.4	6.0	0.5±0.0	0.7	0.5±0.0	0.5±0.0	0.5±0.0	0.5±0.0	0.5/0.5
SGD (5000 steps)	5k	2×10^3	3×10^4	25.6±0.3	50.7	26.2±1.4	13.2±1.1	25.4±1.1	4.8±0.8	34.8/24.3
SGD (10000 steps)	10k	4×10^3	6×10^4	37.7±0.6	62.0	38.7±1.6	22.1±1.4	36.3±1.2	8.0±1.2	49.0/33.4
SGD (100 epochs)	1000k	$6\times10^{5*}$	$6\times10^{7*}$	–		–	–	–	–	92.9/72.2

parameters achieve up to 77.1%, while the best accuracy of training with SGD for 50 epochs is around 15% more. We even show good results on ImageNet, where for some architectures we achieve a top-5 accuracy of up to 48.3%. While these results are low for direct downstream applications, they are remarkable for three main reasons. First, to train GHNs by optimizing (6.2), we do not rely on the prohibitively expensive procedure of training the architectures $\mathcal{F}$ by SGD. Second, GHNs rely on a single forward pass to predict all parameters. Third, these results are obtained for unseen architectures, including the OOD ones. Even in the case of severe distribution shifts (e.g. ResNet-50[5]) and underrepresented networks (e.g. ViT[6]), our model still predicts parameters that perform better than random ones. On CIFAR-10, generalization of GHN-2 is particularly strong with a 58.6% accuracy on ResNet-50.

On both image datasets, our GHN-2 significantly outperforms GHN-1 on all test subsets of DeepNets-1M with more than a 20% absolute gain in certain cases, e.g. 36.8% vs 13.7% on the BN-free networks (Table 6.3). Exploiting the structure of computational graphs is a critical property of GHNs with the accuracy dropping from 66.9% to 42.2% on ID (and even more on OOD) architectures when we replace the GatedGNN of GHN-2 with an MLP. Compared to iterative optimization methods, GHN-2 predicts parameters achieving an accuracy similar to ~2500 and ~5000 iterations of SGD on CIFAR-10 and ImageNet respectively. In contrast, GHN-1 performs similarly to only ~500 and ~2000 (not shown in Table 6.4) iterations respectively. Comparing SGD to Adam, the latter performs worse in general except for the ViT architectures similar to (Zhang et al., 2019d; Dosovitskiy et al., 2020).

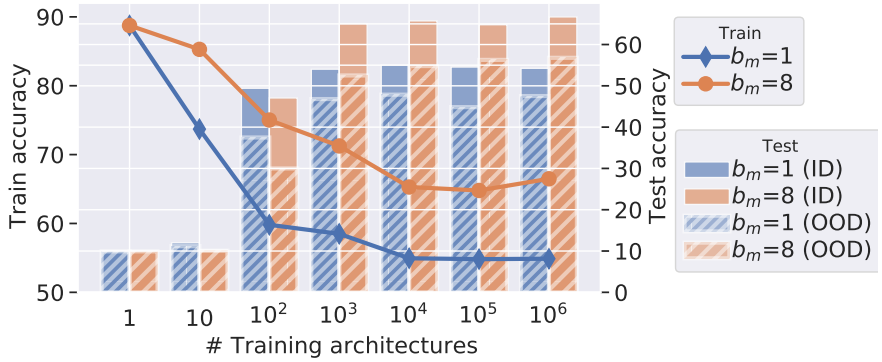


Figure 6.3: GHN-2 with meta batch $b_m = 8$ versus $b_m = 1$ for different numbers of training architectures on CIFAR-10.

To report speeds on ImageNet in Table 6.4, we use a dedicated machine with a single NVIDIA V100-32GB and Intel Xeon CPU E5-1620 v4@ 3.50GHz. So for SGD these numbers can

[5]Large architectures with bottleneck layers such as ResNet-50 do not appear during training.

[6]Architectures such as ViT do not include BN and, except for the first layer, convolutions – the two most frequent operations in the training set.

Table 6.5: Ablating GHN-2 on CIFAR-10. An average rank of the model is computed across all ID and OOD test architectures.

Model	ID-test	OOD-test	Avg. rank
GHN-2	**66.9**±0.3	**56.8**±0.8	**1.9**
1000 training architectures	65.1±0.5	52.5±1.0	2.6
No normalization (§ 6.4.1)	62.6±0.6	47.1±1.2	3.9
No virtual edges (§ 6.4.2)	61.5±0.4	53.9±0.6	4.1
No meta-batch ($b_m = 1$, § 6.4.3)	54.3±0.3	47.5±0.6	5.5
$b_m = 1$, train 8× longer	62.4±0.5	51.9±1.0	3.7
No GatedGNN (MLP)	42.2±0.6	32.2±0.7	7.4
GHN-1	51.4±0.4	39.2±0.9	6.8

be reduced by using faster computing infrastructure and more optimal hyperparameters (Goyal et al., 2017). Using our setup, SGD requires on average $10^4\times$ more time on a GPU ($10^5\times$ on a CPU) to obtain parameters that yield performance similar to GHN-2. As a concrete example, AlexNet (Krizhevsky et al., 2012) requires around 50 GPU hours (on our setup) to achieve a 81.8% top-5 accuracy, while on some architectures we achieve ≥48.0% in just 0.3 GPU seconds.

Ablations (Table 6.5) show that all three components proposed in § 6.4 are important. Normalization is particularly important for OOD generalization with the largest drops on the Wide and BN-free networks. Using meta-batching ($b_m = 8$) is also essential and helps stabilize training and accelerate convergence. We also confirm that the performance gap between $b_m = 1$ and $b_m = 8$ is not primarily due to the observation of more architectures, since the ablated GHN-2 with $b_m = 1$ trained eight times longer is still inferior. The gap between $b_m = 8$ and $b_m = 1$ becomes pronounced with *at least* 1k training architectures (Fig. 6.3). When training with fewer architectures (e.g. 100), the GHN with meta-batching starts to overfit to the training architectures. Given our challenging setup with unseen evaluation architectures, it is surprising that using 1k training architectures already gives strong results. However, OOD generalization degrades in this case compared to using all 1M architectures, especially on the BN-free networks. When training GHNs on just a few architectures, the training accuracy soars to the level of training them with SGD. With more architectures, it generally decreases indicating classic overfitting and underfitting cases.

6.5.2 Property prediction

Representing computational graphs of neural architectures is a challenging problem (Li et al., 2020b; Wen et al., 2019; Jin et al., 2019; Kriege et al., 2020; Makarov et al., 2021). We verify if GHNs are capable of doing that out-of-the-box in the property prediction experiments. Our hypothesis is that by better solving our parameter prediction task, GHNs should also better solve

graph representation tasks.

Experimental setup. We predict the properties of architectures given their graph embeddings obtained by averaging node features[7]. We consider four such properties:

- Accuracy on the "clean" (original) validation set of images;
- Accuracy on a corrupted set (obtained by adding the Gaussian noise to images following (Hendrycks and Dietterich, 2019));
- Inference speed (latency or GPU seconds per a batch of images);
- Convergence speed (the number of SGD iterations to achieve a certain training accuracy).

Estimating these properties accurately can have direct practical benefits. Clean and corrupted accuracies can be used to search for the best performing architectures (e.g. for the NAS task); inference speed can be used to choose the fastest network, so by estimating these properties we can trade-off accurate, robust and fast networks (Cai et al., 2019a). Convergence speed can be used to find networks that are easier to optimize. These properties correlate poorly with each other and between CIFAR-10 and ImageNet, so they require the model to capture different regularities of graphs. While specialized methods to estimate some of these properties exist, often as a NAS task (Wen et al., 2020; Lukasik et al., 2020; Baker et al., 2017; Liu et al., 2018a; Li et al., 2020b), our GHNs provide a generic representation that can be easily used for many such properties. For each property, we train a simple regression model using graph embeddings and ground truth property values. We use 500 validation architectures of DeepNets-1M for training the regression model and tuning its hyperparameters. We then use 500 testing architectures of DeepNets-1M to measure Kendall's Tau rank correlation between the predicted and ground truth property values similar to (Wen et al., 2020).

Additional baseline. We compare to the Neural Predictor (NeuPred) (Wen et al., 2020). NeuPred is based on directed graph convolution and is developed for accuracy prediction achieving strong NAS results. We train a separate such NeuPred for each property from scratch following their hyperparameters.

Results. GHN-2 consistently outperforms the GHN-1 and MLP baselines as well as NeuPred (Fig. 6.4). We also verify if higher correlations translate to downstream gains. For example, on CIFAR-10 by choosing the most accurate architecture according to the regression model and training it from scratch following (Liu et al., 2018b; Zhang et al., 2018a), we obtained a 97.26%($\pm$0.09) accuracy, which is competitive with leading NAS approaches, e.g. (Liu et al., 2018b; Zhang et al., 2018a; Chen et al., 2019c; Yang et al., 2020; He et al., 2020; Li et al., 2020a). In contrast, the network chosen by the regression model trained on the GHN-1 embeddings achieves 95.90%($\pm$0.08).

[7] A fixed size graph embedding for the architecture a can be computed by averaging the output node features: $\mathbf{h}_a = \frac{1}{|V|}\sum_{v\in V}\mathbf{h}_v^T$, where $\mathbf{h}_a \in \mathbb{R}^d$ and d is the dimensionality of node features.

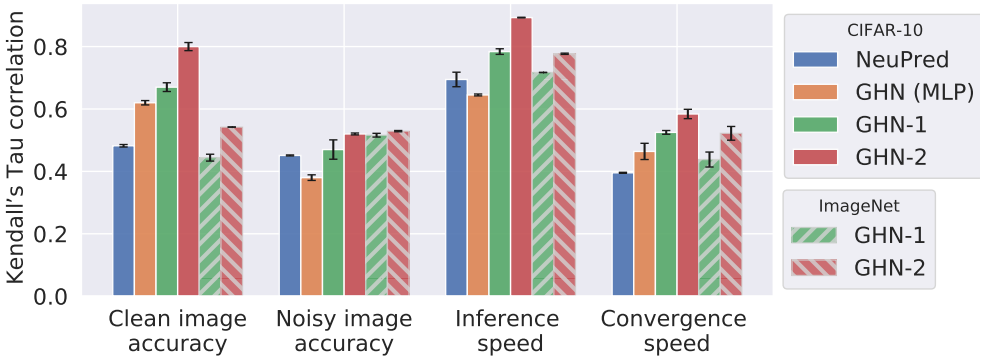


Figure 6.4: Property prediction of neural networks in terms of correlation (higher is better). Error bars denote the standard deviation across 5 runs.

6.5.3 Fine-tuning predicted parameters

Neural networks trained on ImageNet and other large datasets have proven useful in diverse visual tasks in the transfer learning setup (Kornblith et al., 2019; Huh et al., 2016; Neyshabur et al., 2020; Raghu et al., 2019; Zhai et al., 2019; Dosovitskiy et al., 2020). Therefore, we explore how predicting parameters on ImageNet with GHNs compares to pretraining them on ImageNet with SGD in such a setup. We consider low-data tasks as they often benefit more from transfer learning (Raghu et al., 2019; Zhai et al., 2019).

Experimental setup. We perform two transfer-learning experiments. The first experiment is fine-tuning the predicted parameters on 1,000 training samples (100 labels per class) of CIFAR-10. We fine-tune ResNet-50, Visual Transformer (ViT) and a 14-cell architecture based on the DARTS best cell (Liu et al., 2018b). The hyperparameters of fine-tuning (initial learning rate and weight decay) are tuned on 200 validation samples held-out of the 1,000 training samples. The number of epochs is fixed to 50 as in § 6.5.1 for simplicity. In the second experiment, we fine-tune the predicted parameters on the object detection task. We closely follow the experimental protocol and hyperparameters from (PyTorch, 2021) and train the networks on the Penn-Fudan dataset (Wang et al., 2007). The dataset contains only 170 images and the task is to detect pedestrians. Therefore this task is also well suited for transfer learning. Following (PyTorch, 2021), we replace the backbone of a Faster R-CNN with one of the three architectures. To perform transfer learning with GHNs, in both experiments we predict the parameters of a given architecture using GHNs trained on ImageNet. We then replace the ImageNet classification layer with the target task-specific layers and fine-tune the entire network on the target task. We compare the results of GHNs to He's initialization (He et al., 2015) and the initialization based on pretraining the parameters on ImageNet with SGD.

Results. The CIFAR-10 image classification results of fine-tuning the parameters predicted

Table 6.6: CIFAR-10 test set accuracies and Penn-Fudan object detection average precision (at IoU=0.50) after fine-tuning the networks using SGD initialized with different methods. Average results and standard deviations for 3 runs with different random seeds are shown. For each architecture, similar GHN-2-based and ImageNet-based results are bolded. *Estimated on ResNet-50.

Initialization Method	GPU sec. to init.*	100-Shot CIFAR-10			Penn-Fudan Object Detection		
		ResNet-50	ViT	DARTS	ResNet-50	ViT	DARTS
He's (He et al., 2015)	0.003	41.0±0.4	33.2±0.3	45.4±0.4	0.197±0.042	0.144±0.010	0.486±0.035
GHN-1 (trained on ImageNet)	0.6	46.6±0.0	23.3±0.1	49.2±0.1	0.433±0.013	0.0±0.0	0.468±0.024
GHN-2 (trained on ImageNet)	0.7	**56.4**±0.1	**41.4**±0.6	**60.7**±0.3	**0.560**±0.019	**0.436**±0.032	**0.785**±0.032
ImageNet (1k pretraining steps)	6×10^2	45.4±0.3	**44.3**±0.1	**62.4**±0.3	0.302±0.022	0.182±0.046	**0.814**±0.033
ImageNet (2.5k pretraining steps)	1.5×10^3	**55.4**±0.2	50.4±0.3	70.4±0.2	**0.571**±0.056	0.322±0.073	0.823±0.022
ImageNet (5 pretraining epochs)	3×10^4	84.6±0.2	70.2±0.5	83.9±0.1	0.723±0.045	0.391±0.024	0.827±0.053
ImageNet (final epoch)	6×10^5	89.2±0.2	74.5±0.2	85.6±0.2	0.876±0.011	**0.468**±0.023	0.881±0.023

by our GHN-2 are ≥10 percentage points better (in absolute terms) than fine-tuning the parameters predicted by GHN-1 or training the parameters initialized using He's method (Table 6.6). Similarly, the object detection results of GHN-2-based initialization are consistently better than both GHN-1 and He's initializations. The GHN-2 results are a factor of 1.5-3 improvement over He's for all the three architectures. Overall, the two experiments clearly demonstrate the practical value of predicting parameters using our GHN-2. Using GHN-1 for initialization provides relatively small gains or hurts convergence (for ViT). Compared to pretraining on ImageNet with SGD, initialization using GHN-2 leads to performance similar to 1k-2.5k steps of pretraining on ImageNet depending on the architecture in the case of CIFAR-10. In the case of Penn-Fudan, GHN-2's performance is similar to ≥1k steps of pretraining with SGD. In both experiments, pretraining on ImageNet for just 5 epochs provides strong transfer learning performance and the final ImageNet checkpoints are only slightly better, which aligns with previous works (Neyshabur et al., 2020). Therefore, further improvements in the parameter prediction models appear promising.

6.6 Related work

Our proposed parameter prediction task, objective in (6.2) and improved GHN are related to a wide range of machine learning frameworks, in particular meta-learning and neural architecture search (NAS). Meta-learning is a general framework (Hospedales et al., 2020; Schmidhuber, 2020) that includes meta-optimizers and meta-models, among others. Related NAS works include differentiable (Liu et al., 2018b) and one-shot methods (Cai et al., 2019a).

Meta-optimizers. Meta-optimizers (Andrychowicz et al., 2016; Ravi and Larochelle, 2016; Metz et al., 2020; Kirsch and Schmidhuber, 2020; Gomes et al., 2021) define a problem similar to our task, but where $H_{\mathcal{D}}$ is an RNN-based model predicting the gradients $\nabla\mathbf{w}$, mimicking the behavior of iterative optimizers. Therefore, the objective of meta-optimizers may be phrased as

learning to optimize as opposed to our *learning to predict* parameters. Such meta-optimizers can have their own hyperparameters that need to be tuned for a given architecture a and need to be run expensively (on the GPU) for many iterations following (6.1).

Meta-models. Meta-models include methods based on MAML (Finn et al., 2017), ProtoNets (Snell et al., 2017) and auxiliary nets predicting task-specific parameters (Romero et al., 2016; Requeima et al., 2019; Li et al., 2019; Bertinetto et al., 2016). These methods are tied to a particular architecture and need to be trained from scratch if it is changed. Several recent methods attempt to relax the choice of architecture in meta-learning. T-NAS (Lian et al., 2020) combines MAML with DARTS (Liu et al., 2018b) to learn both the optimal architecture and its parameters for a given task. However, the best network, a, needs to be trained using MAML from scratch. Meta-NAS (Elsken et al., 2020) takes a step further and only requires fine-tuning of a on a given task. However, the a is obtained from a single meta-architecture and so its choice is limited, preventing parameter prediction for arbitrary a. CATCH (Chen et al., 2020b) follows a similar idea, but uses reinforcement learning to quickly search for the best a on the specific task. Overall meta-learning mainly aims at generalization *across tasks*, often motivated by the few-shot learning problem. In contrast, our parameter prediction problem assumes a single task (here an image dataset), but aims at generalization *across architectures* a with the ability to predict parameters in a single forward pass.

One-shot NAS. One-shot NAS aims to learn a single "supernet" (Yu et al., 2020; Cai et al., 2019a; He et al., 2021) that can be used to estimate the performance of smaller nets (subnets) obtained by some kind of pruning the supernet, followed by training the best chosen a from scratch with SGD. Recent models, in particular BigNAS (Cai et al., 2019a) and OnceForAll (OFA) (Yu et al., 2020), eliminate the need to train subnets. However, the fundamental limitation of one-shot NAS is poor scaling with the number of possible computational operations (Zhang et al., 2018a). This limits the diversity of architectures for which parameters can be obtained. For example, all subnets in OFA are based on MobileNet-v3 (Howard et al., 2019), which does not allow to solve our more general parameter prediction task. To mitigate this, SMASH (Brock et al., 2017b) proposed to predict some of the parameters using hypernetworks (Ha et al., 2016) by encoding architectures as a 3D tensor. Graph HyperNetworks (GHNs) (Zhang et al., 2018a) further generalized this approach to "arbitrary" computational graphs (DAGs), which allowed them to improve NAS results. GHNs focused on obtaining reliable subnetwork rankings for NAS and did not aim to predict large-scale performant parameters. We show that the vanilla GHNs perform poorly on our parameter prediction task mainly due to the inappropriate scale of predicted parameters, lack of long-range interactions in the graphs, gradient noise and slow convergence when optimizing (6.2). Conventionally to NAS, GHNs were also trained in a quite constrained architecture space (Bender et al., 2018). We expand the architecture space adopting GHNs for a more general problem.

Our work is also loosely related to other parameter prediction methods (Denil et al., 2013; Bertinetto et al., 2016; Ratzlaff and Fuxin, 2019), analysis of graph structure of neural networks (You et al., 2020a), knowledge distillation from multiple teachers (Liu et al., 2019), compression methods (Cheng et al., 2017) and optimization-based initialization (Dauphin and Schoenholz, 2019; Zhu et al., 2021; Das et al., 2021). Denil et al. (2013) train a model that can predict a fraction of network parameters given other parameters requiring to retrain the model for each new architecture. Bertinetto et al. (2016) train a model that predicts parameters given a new few-shot task similarly to (Ravi and Larochelle, 2016; Requeima et al., 2019), and the model is also tied to a particular architecture. The HyperGAN (Ratzlaff and Fuxin, 2019) allows to generate an ensemble of trained parameters in a computationally efficient way, but as the aforementioned works is constrained to a particular architecture. Finally, MetaInit (Dauphin and Schoenholz, 2019), GradInit (Zhu et al., 2021) and Sylvester-based initialization (Das et al., 2021) can initialize arbitrary networks by carefully optimizing their initial parameters, but due to the optimization loop they are generally more computationally expensive compared to predicting parameters using GHNs. Overall, these prior works did not formulate the task nor proposed the methods of predicting performant parameters for diverse and large-scale architectures as ours.

Finally, the construction of our DeepNets-1M is related to the works on network design spaces. Generating arbitrary architectures using a graph generative model, e.g. (Yu et al., 2019; Guo and Zhao, 2020; You et al., 2020a), can be one way to create the training dataset $\mathcal{F}$. Instead, we leverage and extend an existing DARTS framework (Liu et al., 2018b) specializing on neural architectures to generate $\mathcal{F}$. More recent works (Radosavovic et al., 2020) or other domains (You et al., 2020b) can be considered in future work.

6.7 Conclusion

We propose a novel framework and benchmark to learn and evaluate neural parameter prediction models. Our model (GHN-2) is able to predict parameters for very diverse and large-scale architectures in a single forward pass in a fraction of a second. The networks with predicted parameters yield surprisingly high image classification accuracy given the extremely challenging nature of our parameter prediction task. However, the accuracy is still far from networks trained with handcrafted optimization methods. Bridging the gap is a promising future direction. As a beneficial side-effect, GHN-2 learns a strong representation of neural architectures as evidenced by our property prediction evaluation. Finally, parameters predicted using GHN-2 trained on ImageNet benefit transfer learning in the low-data regime. This motivates further research towards solving our task.

7 Concluding Remarks

This thesis explores generalization in graph reasoning tasks such as graph classification, compositional visual reasoning using scene graphs and reasoning about neural networks (NNs). Approaching these tasks using models that learn from data instead of engineering features is a de facto standard. Nevertheless, one of the fundamental challenges of such models, in particular NNs, is poor generalization. This issue is perhaps due to models' reliance on spurious correlations ("shortcuts") that are often abundant in training data (Shen et al., 2021; Zhou et al., 2021; Schölkopf et al., 2021). The issue is particularly noticeable when NNs are evaluated on test data that are sampled from a slightly different distribution as compared to the training data. While humans often show strong generalization to various distribution shifts, machine learning models are much weaker in that respect. This thesis makes several contributions towards understanding and improving generalization.

In Chapter 3, we address generalization in classic graph tasks. Machine learning on graphs and developing stronger and more expressive graph neural networks (GNNs) to solve them have become increasingly popular topics in artificial intelligence. Generalization of GNNs to different distribution shifts remains a fundamental issue. Some distribution shifts, such as increased graph size at test time, are studied more in detail than others. In particular, Yehudai et al. (2021) show the existence of a fundamental issue in GNNs as they converge to a "bad" optimum under certain assumptions preventing generalization to larger graphs. Along with our work, additional supervision improved their generalization results. Bevilacqua et al. (2021) address the size generalization problem by explicitly modeling a size-invariant representation. In practice, different distribution shifts might arise, such as noisy node features in our work. Modeling each distribution shift explicitly is a time consuming process. Instead, it is desirable to learn to generalize to diverse distribution shifts. However, this is perhaps an ill-posed task. Therefore, a possible direction to improve generalization without explicitly hardcoding it is to follow the meta-learning idea of "learning to generalize" (Finn et al., 2017). Another direction is to learn more generic models that have access to more diverse knowledge similar to humans (Brown et al., 2020; Cobbe et al., 2021). These models can be fine-tuned to diverse tasks and potentially can have better generalization by having a strong inductive prior. In general, the approaches of improving generalization of other kinds of NNs (especially CNNs) are possible to be applied to GNNs and vice versa due the inherent similarity between different kinds of NNs (§ 2.1). Therefore, transferring generalization approaches from one domain to another should be studied more.

In Chapters 4 and 5, we address the compositional generalization problem in visual reasoning, specifically in the scene graph generation (SGG) task. SGG models typically have two stages: (i) object detection and (ii) compositional reasoning that predicts scene graphs based on features from the detector. Our works only improve the second stage. Meanwhile, it is possible that the composi-

tional generalization issue primarily arises from the first stage due to the "bad" bias learned by the detectors (Michaelis et al., 2019), which is hard to fix in the second stage. Therefore, one approach to further improve generalization in SGG tasks is to improve generalization of object detectors (Shen et al., 2021). Object detectors are in turn built on image classification networks, so improving generalization in image classification can indirectly lead to improved compositional generalization in SGG (Li et al., 2020c; Chen et al., 2021). Another approach to improve compositional generalization in SGG can be based on learning a more generic language (Brown et al., 2020; Cobbe et al., 2021) or multimodal (Radford et al., 2021) model along the lines of the discussion in the previous paragraph. In particular, the SGG task is closely related to natural language processing, so by leveraging a strong prior from language models better compositional generalization in vision can be achieved.

In Chapter 6, we explore a novel graph reasoning task, in which models must generalize to novel graphs in a non-trivial way. In particular, we introduced a parameter prediction task, in which a single GNN (more specifically, GHN) must predict parameters for unseen deep neural networks represented as graphs. Generalization to unseen networks is extremely challenging even if the networks are *i.i.d.* sampled (see our work (Knyazev et al., 2021) as well as works on meta-optimizers (Wichrowska et al., 2017; Metz et al., 2020)). The challenge is due to a complex interplay between thousands or millions of neurons in a NN. To study generalization of GHNs more systematically, we evaluate generalization to specific distribution shifts, including NN's capacity, graph size and complexity of connections. Despite surprisingly good results of our GHN in certain cases, scaling our GHN to larger datasets is an important future direction to increase the practical value of our GHN. Another important direction to pursue is to enable generalization of GHNs across tasks along with meta-optimizers (Metz et al., 2020) and other hypernetwork works (Anonymous, 2021). This way, GHNs has a potential to become more practical for end users to solve their tasks with little computational resources available. Thus, our work makes an important step towards democratization of machine learning (Ahmed and Wahed, 2020; Abdalla and Abdalla, 2021).

7 References

Abdalla, M. and Abdalla, M. (2021). The grey hoodie project: Big tobacco, big tech, and the threat on academic integrity. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, pages 287–297. 90

Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., and Süsstrunk, S. (2012). Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2274–2282. 27

Agarwal, A., Mangal, A., et al. (2020). Visual relationship detection using scene graphs: A survey. *arXiv preprint arXiv:2005.08045*. 54, 56

Agrawal, A., Batra, D., Parikh, D., and Kembhavi, A. (2018). Don't just assume; look and answer: Overcoming priors for visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4971–4980. 16

Agrawal, A., Kembhavi, A., Batra, D., and Parikh, D. (2017). C-vqa: A compositional split of the visual question answering (vqa) v1. 0 dataset. *arXiv preprint arXiv:1704.08243*. 16

Ahmed, N. and Wahed, M. (2020). The de-democratization of ai: Deep learning and the compute divide in artificial intelligence research. *arXiv preprint arXiv:2010.15581*. 90

Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., and Marchand, M. (2014). Domain-adversarial neural networks. *arXiv preprint arXiv:1412.4446*. 18

Alon, U. and Yahav, E. (2020). On the bottleneck of graph neural networks and its practical implications. *arXiv preprint arXiv:2006.05205*. 78

Anand, A., Belilovsky, E., Kastner, K., Larochelle, H., and Courville, A. (2018). Blindfold baselines for embodied QA. *arXiv preprint arXiv:1811.05013*. 38

Andreas, J., Rohrbach, M., Darrell, T., and Klein, D. (2016). Neural module networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 39–48. 18

Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., Shillingford, B., and De Freitas, N. (2016). Learning to learn by gradient descent by gradient descent. In *Advances in neural information processing systems*, pages 3981–3989. 4, 70, 71, 80, 86

Anonymous (2021). Hypertransformer: Attention-based cnn model generation from few samples. *ICLR 2020 submission, https://openreview.net/forum?id=E9z2A1-O7e*. 70, 90

Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Lawrence Zitnick, C., and Parikh, D. (2015). VQA: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433. 8, 13, 14, 37

Arbelaez, P., Maire, M., Fowlkes, C., and Malik, J. (2010). Contour detection and hierarchical image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):898–916. 19

Arjovsky, M., Bottou, L., Gulrajani, I., and Lopez-Paz, D. (2019). Invariant risk minimization. *arXiv preprint arXiv:1907.02893*. 16, 54

Atzmon, Y., Berant, J., Kezami, V., Globerson, A., and Chechik, G. (2016). Learning to generalize to new compositions in image understanding. *arXiv preprint arXiv:1608.07639*. 16, 17, 54

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*. 76

Bahdanau, D., de Vries, H., O'Donnell, T. J., Murty, S., Beaudoin, P., Bengio, Y., and Courville, A. (2019). Closure: Assessing systematic generalization of clevr models. *arXiv preprint arXiv:1912.05783*. 16

Bahdanau, D., Murty, S., Noukhovitch, M., Nguyen, T. H., de Vries, H., and Courville, A. (2018). Systematic generalization: what is required and can it be learned? *arXiv preprint arXiv:1811.12889*. 16, 38, 54

Baker, B., Gupta, O., Raskar, R., and Naik, N. (2017). Accelerating neural architecture search using performance prediction. *arXiv preprint arXiv:1705.10823*. 84

Bansal, N., Chen, X., and Wang, Z. (2018). Can we gain more from orthogonality regularizations in training deep cnns? *arXiv preprint arXiv:1810.09102*. 20

Bapst, V., Sanchez-Gonzalez, A., Doersch, C., Stachenfeld, K. L., Kohli, P., Battaglia, P. W., and Hamrick, J. B. (2019). Structured agents for physical construction. *arXiv preprint arXiv:1904.03177*. 12

Barrat, A., Barthelemy, M., Pastor-Satorras, R., and Vespignani, A. (2004). The architecture of complex weighted networks. *Proceedings of the national academy of sciences*, 101(11):3747–3752. xi, 77

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. (2018). Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*. xii, 12, 15

Belilovsky, E., Blaschko, M., Kiros, J., Urtasun, R., and Zemel, R. (2017). Joint embeddings of scene graphs and images. 15, 39, 54

Belkin, M. and Niyogi, P. (2001). Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Nips*, volume 14, pages 585–591. 9

Bender, G., Kindermans, P.-J., Zoph, B., Vasudevan, V., and Le, Q. (2018). Understanding and simplifying one-shot architecture search. In *International Conference on Machine Learning*, pages 550–559. 87

Bertinetto, L., Henriques, J. F., Valmadre, J., Torr, P. H., and Vedaldi, A. (2016). Learning feed-forward one-shot learners. *arXiv preprint arXiv:1606.05233*. 87, 88

Bevilacqua, B., Zhou, Y., and Ribeiro, B. (2021). Size-invariant graph representations for graph classification extrapolations. *arXiv preprint arXiv:2103.05045*. 89

Borgwardt, K. M., Ong, C. S., Schönauer, S., Vishwanathan, S., Smola, A. J., and Kriegel, H.-P. (2005). Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl_1):i47–i56. 24, 27

Brock, A., De, S., and Smith, S. L. (2021a). Characterizing signal propagation to close the performance gap in unnormalized resnets. *arXiv preprint arXiv:2101.08692*. 75, 77

Brock, A., De, S., Smith, S. L., and Simonyan, K. (2021b). High-performance large-scale image recognition without normalization. *arXiv preprint arXiv:2102.06171*. 75, 77

Brock, A., Donahue, J., and Simonyan, K. (2018). Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*. 57

Brock, A., Lim, T., Ritchie, J. M., and Weston, N. (2017a). Freezeout: Accelerate training by progressively freezing layers. *arXiv preprint arXiv:1706.04983*. 71

Brock, A., Lim, T., Ritchie, J. M., and Weston, N. (2017b). Smash: one-shot model architecture search through hypernetworks. *arXiv preprint arXiv:1708.05344*. 87

Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. (2017). Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42. 2, 7, 9, 22

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*. 2, 71, 77, 89, 90

Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. (2014). Spectral networks and locally connected networks on graphs. In *International Conference on Learning Representations (ICLR)*. 9

Burgess, C. P., Matthey, L., Watters, N., Kabra, R., Higgins, I., Botvinick, M., and Lerchner, A. (2019). Monet: Unsupervised scene decomposition and representation. *arXiv preprint arXiv:1901.11390*. 15, 19

Cai, H., Gan, C., Wang, T., Zhang, Z., and Han, S. (2019a). Once-for-all: Train one network and specialize it for efficient deployment. 2, 9, 71, 76, 84, 86, 87

Cai, H., Gan, C., Wang, T., Zhang, Z., and Han, S. (2019b). Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*. 70

Cangea, C., Belilovsky, E., Liò, P., and Courville, A. (2019). VideoNavQA: Bridging the Gap between Visual and Embodied Question Answering. *arXiv preprint arXiv:1908.04950*. 37, 54

Casanova, A., Drozdzal, M., and Romero-Soriano, A. (2020). Generating unseen complex scenes: are we there yet? *arXiv preprint arXiv:2012.04027*. 57, 67, 69

Chang, O., Flokas, L., and Lipson, H. (2019). Principled weight initialization for hypernetworks. In *International Conference on Learning Representations*. 78

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020a). A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR. 19

Chen, T., Yu, W., Chen, R., and Lin, L. (2019a). Knowledge-embedded routing network for scene graph generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6163–6171. x, 17, 40, 44, 47, 50, 55, 57, 65

Chen, V. S., Varma, P., Krishna, R., Bernstein, M., Re, C., and Fei-Fei, L. (2019b). Scene graph prediction with limited labels. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2580–2590. 17, 40, 55, 57

Chen, X., Duan, Y., Chen, Z., Xu, H., Chen, Z., Liang, X., Zhang, T., and Li, Z. (2020b). Catch: Context-based meta reinforcement learning for transferrable architecture search. In *European Conference on Computer Vision*, pages 185–202. Springer. 87

Chen, X., Xie, C., Tan, M., Zhang, L., Hsieh, C.-J., and Gong, B. (2021). Robust and accurate object detection via adversarial learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16622–16631. 90

Chen, X., Xie, L., Wu, J., and Tian, Q. (2019c). Progressive darts: Bridging the optimization gap for nas in the wild. *arXiv preprint arXiv:1912.10952*. 73, 80, 84

Cheng, Y., Wang, D., Zhou, P., and Zhang, T. (2017). A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*. 9, 88

Chetlur, S., Woolley, C., Vandermersch, P., Cohen, J., Tran, J., Catanzaro, B., and Shelhamer, E. (2014). cudnn: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759*. 9

Cho, K., Van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*. 74

Choi, D., Passos, A., Shallue, C. J., and Dahl, G. E. (2019). Faster neural network training with data echoing. *arXiv preprint arXiv:1907.05550*. 71

Choromanska, A., Henaff, M., Mathieu, M., Arous, G. B., and LeCun, Y. (2015). The loss surfaces of multilayer networks. In *Artificial intelligence and statistics*, pages 192–204. PMLR. 5

Chung, F. R. and Graham, F. C. (1997). *Spectral graph theory*. Number 92. American Mathematical Soc. 9

Ciregan, D., Meier, U., and Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3642–3649. IEEE. 7

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*. 6

Cobbe, K., Kosaraju, V., Bavarian, M., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. (2021). Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*. 89, 90

Cogswell, M., Ahmed, F., Girshick, R., Zitnick, L., and Batra, D. (2015). Reducing overfitting in deep networks by decorrelating representations. *arXiv preprint arXiv:1511.06068*. 20

Cogswell, M., Lu, J., Lee, S., Parikh, D., and Batra, D. (2019). Emergence of compositional language with deep generational transmission. *arXiv preprint arXiv:1904.09067*. 16

Cong, W., Wang, W., and Lee, W.-C. (2018). Scene graph generation via conditional random fields. *arXiv preprint arXiv:1811.08075*. 17, 40

Corso, G., Cavalleri, L., Beaini, D., Liò, P., and Velivcković, P. (2020). Principal neighbourhood aggregation for graph nets. *arXiv preprint arXiv:2004.05718*. 12

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. V. (2018). Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*. 57

Dai, B., Zhang, Y., and Lin, D. (2017). Detecting visual relationships with deep relational networks. In *Proceedings of the IEEE conference on computer vision and Pattern recognition*, pages 3076–3086. 44

Damodaran, V., Chakravarthy, S., Kumar, A., Umapathy, A., Mitamura, T., Nakashima, Y., Garcia, N., and Chu, C. (2021). Understanding the role of scene graphs in visual question answering. *arXiv preprint arXiv:2101.05479*. 54

Das, D., Bhalgat, Y., and Porikli, F. (2021). Data-driven weight initialization with sylvester solvers. *arXiv preprint arXiv:2105.10335*. 88

Dauphin, Y. and Schoenholz, S. S. (2019). Metainit: Initializing learning by learning to initialize. 88

Deac, A., VelivCković, P., and Sormanni, P. (2018). Attentive cross-modal paratope prediction. *Journal of Computational Biology*. 22

Defferrard, M., Bresson, X., and Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. *arXiv preprint arXiv:1606.09375*. x, 10, 11, 12, 23, 26, 27, 35

Demirel, B., Gokberk Cinbis, R., and Ikizler-Cinbis, N. (2017). Attributes2classname: A discriminative model for attribute-based unsupervised zero-shot learning. In *Proceedings of the IEEE international conference on computer vision*, pages 1232–1241. xii, 14, 16

Deng, F., Zhi, Z., Lee, D., and Ahn, S. (2021). Generative scene graph networks. In *International Conference on Learning Representations*. 57

Denil, M., Shakibi, B., Dinh, L., Ranzato, M., and de Freitas, N. (2013). Predicting parameters in deep learning. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 26*, pages 2148–2156. Curran Associates, Inc. 88

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*. 62

DeVries, T., Drozdzal, M., and Taylor, G. W. (2020). Instance selection for gans. *arXiv preprint arXiv:2007.15255*. 67

DeVries, T. and Taylor, G. W. (2017a). Dataset augmentation in feature space. *arXiv preprint arXiv:1702.05538*. 68

DeVries, T. and Taylor, G. W. (2017b). Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*. 57

Dittadi, A., Papa, S., De Vita, M., Schölkopf, B., Winther, O., and Locatello, F. (2021). Generalization and robustness implications in object-centric learning. *arXiv preprint arXiv:2107.00637*. 19, 37

Dobson, P. D. and Doig, A. J. (2003). Distinguishing enzyme structures from non-enzymes without alignments. *Journal of molecular biology*, 330(4):771–783. 24, 27

Dodge, S. and Karam, L. (2017). A study and comparison of human and deep learning recognition performance under visual distortions. In *2017 26th international conference on computer communication and networks (ICCCN)*, pages 1–7. IEEE. 28

Dornadula, A., Narcomey, A., Krishna, R., Bernstein, M., and Fei-Fei, L. (2019). Visual relationships as functions: Enabling few-shot scene graph prediction. In *ArXiv*. 17, 40, 55, 57

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*. 7, 12, 72, 75, 76, 77, 82, 85

Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., Van Der Smagt, P., Cremers, D., and Brox, T. (2015). Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766. 8

Dvornik, N., Mairal, J., and Schmid, C. (2018). Modeling visual context is key to augmenting object detection datasets. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 364–380. 53

Dwibedi, D., Misra, I., and Hebert, M. (2017). Cut, paste and learn: Surprisingly easy synthesis for instance detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1301–1310. 53

Dwivedi, V. P. and Bresson, X. (2020). A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*. 12

Dwivedi, V. P., Joshi, C. K., Laurent, T., Bengio, Y., and Bresson, X. (2020). Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*. 12, 22, 72

El Hihi, S. and Bengio, Y. (1996). Hierarchical recurrent neural networks for long-term dependencies. In *Advances in neural information processing systems*, pages 493–499. 78

Elsken, T., Staffler, B., Metzen, J. H., and Hutter, F. (2020). Meta-learning of neural architectures for few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12365–12375. 87

Engelcke, M., Kosiorek, A. R., Jones, O. P., and Posner, I. (2019). Genesis: Generative scene inference and sampling with object-centric latent representations. *arXiv preprint arXiv:1907.13052*. 19

Everingham, M., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338. 5

Fey, M., Lenssen, J. E., Weichert, F., and Müller, H. (2018). Splinecnn: Fast geometric deep learning with continuous b-spline kernels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 869–877. 27

Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org. 18, 87, 89

Frome, A., Corrado, G. S., Shlens, J., Bengio, S., Dean, J., Ranzato, M., and Mikolov, T. (2013). Devise: A deep visual-semantic embedding model. In *Advances in neural information processing systems*, pages 2121–2129. 40

Fukushima, K. and Miyake, S. (1982). Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer. 7

Galloway, A., Golubeva, A., Tanay, T., Moussa, M., and Taylor, G. W. (2019). Batch normalization is a cause of adversarial vulnerability. *arXiv preprint arXiv:1905.02161*. 2, 77

Ganin, Y. and Lempitsky, V. (2015). Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, pages 1180–1189. PMLR. 18

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., March, M., and Lempitsky, V. (2016). Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(59):1–35. 18

Gao, H. and Ji, S. (2019). Graph U-Net. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*. 13, 22, 23, 25, 29, 33

Garg, V., Jegelka, S., and Jaakkola, T. (2020). Generalization and representational limits of graph neural networks. In *International Conference on Machine Learning*, pages 3419–3430. PMLR. 3, 22

Ghiasi, G., Cui, Y., Srinivas, A., Qian, R., Lin, T.-Y., Cubuk, E. D., Le, Q. V., and Zoph, B. (2021). Simple copy-paste is a strong data augmentation method for instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2918–2928. 53

Ghiasi, G., Lin, T.-Y., and Le, Q. V. (2018). Dropblock: A regularization method for convolutional networks. *arXiv preprint arXiv:1810.12890*. 20

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. (2017). Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 1263–1272. xii, 12, 15

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. 78

Golubeva, A., Neyshabur, B., and Gur-Ari, G. (2020). Are wider nets better given the same number of parameters? *arXiv preprint arXiv:2010.14495*. 77

Gomes, H. S., Leger, B., and Gagne, C. (2021). Meta learning black-box population-based optimizers. *arXiv preprint arXiv:2103.03526*. 86

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680. 8, 55, 57

Gori, M., Monfardini, G., and Scarselli, F. (2005). A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734. IEEE. 2, 11, 12

Goyal, P., Dollar, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., and He, K. (2017). Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*. 83

Greff, K., Kaufman, R. L., Kabra, R., Watters, N., Burgess, C., Zoran, D., Matthey, L., Botvinick, M., and Lerchner, A. (2019). Multi-object representation learning with iterative variational inference. In *International Conference on Machine Learning*, pages 2424–2433. PMLR. 19, 57

Greff, K., van Steenkiste, S., and Schmidhuber, J. (2020). On the binding problem in artificial neural networks. *arXiv preprint arXiv:2012.05208*. 15, 19, 53

Gu, J., Joty, S., Cai, J., Zhao, H., Yang, X., and Wang, G. (2019a). Unpaired image captioning via scene graph alignments. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 10323–10332. 15, 37, 38, 54

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J., et al. (2018). Recent advances in convolutional neural networks. *Pattern Recognition*, 77:354–377. 8

Gu, J., Zhao, H., Lin, Z., Li, S., Cai, J., and Ling, M. (2019b). Scene graph generation with external knowledge and image reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1969–1978. 56

Gunning, D. and Aha, D. (2019). Darpas explainable artificial intelligence (xai) program. *AI Magazine*, 40(2):44–58. 15

Guo, X. and Zhao, L. (2020). A systematic survey on deep generative models for graph generation. *arXiv preprint arXiv:2007.06686*. 88

Ha, D., Dai, A., and Le, Q. V. (2016). Hypernetworks. *arXiv preprint arXiv:1609.09106*. 71, 87

Hagberg, A., Swart, P., and S Chult, D. (2008). Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States). xi, 77

Hamilton, W. L., Bajaj, P., Zitnik, M., Jurafsky, D., and Leskovec, J. (2018). Embedding logical queries on knowledge graphs. *arXiv preprint arXiv:1806.01445*. 13

Hamilton, W. L., Ying, R., and Leskovec, J. (2017). Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*. 12, 22

Hammond, D. K., Vandergheynst, P., and Gribonval, R. (2011). Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150. 10

He, C., Ye, H., Shen, L., and Zhang, T. (2020). Milenas: Efficient neural architecture search via mixed-level reformulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11993–12002. 80, 84

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969. xi, 41, 46, 51, 60

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034. xiii, 2, 8, 34, 78, 85, 86

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778. 2, 8, 71, 73, 75, 77

He, X., Zhao, K., and Chu, X. (2021). Automl: A survey of the state-of-the-art. *Knowledge-Based Systems*, 212:106622. 87

Hendrycks, D. and Dietterich, T. (2019). Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations (ICLR)*. 2, 3, 28, 77, 84

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in neural information processing systems*, pages 6626–6637. 64

Hildebrandt, M., Li, H., Koner, R., Tresp, V., and Günnemann, S. (2020). Scene graph reasoning for visual question answering. *arXiv preprint arXiv:2007.01072*. 54

Hill, F., Lampinen, A., Schneider, R., Clark, S., Botvinick, M., McClelland, J. L., and Santoro, A. (2019). Environmental drivers of systematicity and generalization in a situated agent. 40, 48, 53, 55

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*. 8, 20

Hjelm, R. D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A., and Bengio, Y. (2019). Learning deep representations by mutual information estimation and maximization. In *International Conference on Learning Representations*. 18, 19

Hong, S., Yang, D., Choi, J., and Lee, H. (2018). Inferring semantic layout for hierarchical text-to-image synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7986–7994. 69

Hooker, S. (2020). The hardware lottery. 77

Hospedales, T., Antoniou, A., Micaelli, P., and Storkey, A. (2020). Meta-learning in neural networks: A survey. *arXiv preprint arXiv:2004.05439*. 18, 71, 86

Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., et al. (2019). Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1314–1324. 70, 73, 75, 87

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*. 9, 76

Hu, J., Shen, L., and Sun, G. (2018). Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141. 76

Hu, R., Andreas, J., Rohrbach, M., Darrell, T., and Saenko, K. (2017). Learning to reason: End-to-end module networks for visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 804–813. 18

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. (2020). Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*. 12

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708. 73, 77

Huang, G., Sun, Y., Liu, Z., Sedra, D., and Weinberger, K. Q. (2016). Deep networks with stochastic depth. In *European conference on computer vision*, pages 646–661. Springer. 71

Huang, L., Yang, D., Lang, B., and Deng, J. (2018). Decorrelated batch normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 791–800. 20

Hudson, D. and Manning, C. D. (2019a). Learning by abstraction: The neural state machine. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 5903–5916. Curran Associates, Inc. 14, 38, 45, 54, 56

Hudson, D. A. and Manning, C. D. (2019b). Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6700–6709. x, xi, 37, 39, 45, 47, 51

Huh, M., Agrawal, P., and Efros, A. A. (2016). What makes imagenet good for transfer learning? *arXiv preprint arXiv:1608.08614*. 9, 85

Hupkes, D., Dankers, V., Mul, M., and Bruni, E. (2019). The compositionality of neural networks: integrating symbolism and connectionism. *arXiv preprint arXiv:1908.08351*. 3

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*. 20, 64, 71, 73, 76, 81

Jabri, A., Joulin, A., and Van Der Maaten, L. (2016). Revisiting visual question answering baselines. In *European conference on computer vision*, pages 727–739. Springer. 38

Jain, P. and Kar, P. (2017). Non-convex optimization for machine learning. *arXiv preprint arXiv:1712.07897*. 4

Ji, C., Wang, R., Zhu, R., Cai, Y., and Wu, H. (2020). Hopgat: Hop-aware supervision graph attention networks for sparsely labeled graphs. *arXiv preprint arXiv:2004.04333*. 22

Jiang, Y., Gu, S., Murphy, K., and Finn, C. (2019). Language as an abstraction for hierarchical deep reinforcement learning. *arXiv preprint arXiv:1906.07343*. 16

Jin, H., Song, Q., and Hu, X. (2019). Auto-keras: An efficient neural architecture search system. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1946–1956. 83

Johnson, J., Gupta, A., and Fei-Fei, L. (2018). Image generation from scene graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1219–1228. 58, 60

Johnson, J., Hariharan, B., van der Maaten, L., Fei-Fei, L., Lawrence Zitnick, C., and Girshick, R. (2017a). Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2901–2910. 14, 16, 54

Johnson, J., Hariharan, B., Van Der Maaten, L., Hoffman, J., Fei-Fei, L., Lawrence Zitnick, C., and Girshick, R. (2017b). Inferring and executing programs for visual reasoning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2989–2998. 18

Johnson, J., Krishna, R., Stark, M., Li, L.-J., Shamma, D., Bernstein, M., and Fei-Fei, L. (2015). Image retrieval using scene graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3668–3678. 14, 15, 38, 54, 58

Joshi, N. and He, H. (2021). An investigation of the (in) effectiveness of counterfactually augmented data. *arXiv preprint arXiv:2107.00753*. 53

Karras, T., Aittala, M., Hellsten, J., Laine, S., Lehtinen, J., and Aila, T. (2020). Training generative adversarial networks with limited data. *arXiv preprint arXiv:2006.06676*. 57

Kaushik, D., Hovy, E., and Lipton, Z. C. (2019). Learning the difference that makes a difference with counterfactually-augmented data. *arXiv preprint arXiv:1909.12434*. 53

Kazemi, S. M., Goel, R., Jain, K., Kobyzev, I., Sethi, A., Forsyth, P., and Poupart, P. (2019). Relational representation learning for dynamic (knowledge) graphs: A survey. *arXiv preprint arXiv:1905.11485*. 12

Keysers, D., Schärli, N., Scales, N., Buisman, H., Furrer, D., Kashubin, S., Momchev, N., Sinopalnikov, D., Stafiniak, L., Tihon, T., et al. (2019). Measuring compositional generalization: A comprehensive method on realistic data. *arXiv preprint arXiv:1912.09713*. 16, 54

Khandelwal, S., Suhail, M., and Sigal, L. (2021). Segmentation-grounded scene graph generation. *arXiv preprint arXiv:2104.14207*. 37

Kim, D. and Oh, A. (2020). How to find your friendly neighborhood: Graph attention design with self-supervision. In *International Conference on Learning Representations*. 22

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*. 5, 29, 71, 80

Kipf, T., Fetaya, E., Wang, K.-C., Welling, M., and Zemel, R. (2018). Neural relational inference for interacting systems. In *International Conference on Machine Learning*, pages 2688–2697. PMLR. 2, 12

Kipf, T., Li, Y., Dai, H., Zambaldi, V., Sanchez-Gonzalez, A., Grefenstette, E., Kohli, P., and Battaglia, P. (2019). Compile: Compositional imitation learning and execution. In *International Conference on Machine Learning*, pages 3418–3428. PMLR. 16

Kipf, T. N. and Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*. x, 7, 10, 11, 24, 26, 35, 72

Kirsch, L. and Schmidhuber, J. (2020). Meta learning backpropagation and improving it. *arXiv preprint arXiv:2012.14905*. 86

Knyazev, B., de Vries, H., Cangea, C., Taylor, G. W., Courville, A., and Belilovsky, E. (2020). Graph density-aware losses for novel compositions in scene graph generation. *British Machine Vision Conference (BMVC)*. xi, xii, 3, 16, 17, 54, 55, 56, 61, 62, 63, 64, 65

Knyazev, B., Drozdzal, M., Taylor, G. W., and Romero-Soriano, A. (2021). Parameter prediction for unseen deep architectures. In *Advances in Neural Information Processing Systems (NeurIPS)*. 90

Knyazev, B., Lin, X., Amer, M. R., and Taylor, G. W. (2018). Spectral multigraph networks for discovering and fusing relationships in molecules. In *NeurIPS Workshop on Machine Learning for Molecules and Materials*. 10, 12

Knyazev, B., Lin, X., Amer, M. R., and Taylor, G. W. (2019a). Image classification with hierarchical multigraph networks. In *British Machine Vision Conference (BMVC)*. 10, 11, 12

Knyazev, B., Taylor, G. W., and Amer, M. (2019b). Understanding attention and generalization in graph neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4204–4214. 10

Kocaoglu, M., Snyder, C., Dimakis, A. G., and Vishwanath, S. (2017). CausalGAN: Learning causal implicit generative models with adversarial training. *arXiv preprint arXiv:1709.02023*. 57

Kornblith, S., Shlens, J., and Le, Q. V. (2019). Do better imagenet models transfer better? In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2661–2671. 9, 85

Kriege, N. M., Johansson, F. D., and Morris, C. (2020). A survey on graph kernels. *Applied Network Science*, 5(1):1–42. 83

Krishna, R., Zhu, Y., Groth, O., Johnson, J., Hata, K., Kravitz, J., Chen, S., Kalantidis, Y., Li, L.-J., Shamma, D. A., et al. (2017a). Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123(1):32–73. xi, xv, 16, 54, 55, 56, 58, 63

Krishna, R., Zhu, Y., Groth, O., Johnson, J., Hata, K., Kravitz, J., Chen, S., Kalantidis, Y., Li, L.-J., Shamma, D. A., and et al. (2017b). Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123(1):3273. xiv, 38, 39, 45

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. Technical report, Citeseer. 71, 72, 79

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105. 2, 4, 8, 70, 83

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97. 76

Kynkäänniemi, T., Karras, T., Laine, S., Lehtinen, J., and Aila, T. (2019). Improved precision and recall metric for assessing generative models. *arXiv preprint arXiv:1904.06991*. xi, 66, 67

Lake, B. and Baroni, M. (2018). Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International conference on machine learning*, pages 2873–2882. PMLR. 13

Lake, B. M. (2019). Compositional generalization through meta sequence-to-sequence learning. In *Advances in Neural Information Processing Systems*, pages 9788–9798. 16, 19, 54

Lampert, C. H., Nickisch, H., and Harmeling, S. (2013). Attribute-based classification for zero-shot visual object categorization. *IEEE transactions on pattern analysis and machine intelligence*, 36(3):453–465. 14, 16, 39

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444. 1

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324. 5, 7, 8, 24, 26, 27

Lee, J., Lee, I., and Kang, J. (2019a). Self-attention graph pooling. In *International Conference on Machine Learning*, pages 3734–3743. PMLR. 13

Lee, J. B., Rossi, R., and Kong, X. (2018a). Graph classification using structural attention. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1666–1674. ACM. 23

Lee, J. B., Rossi, R. A., Kim, S., Ahmed, N. K., and Koh, E. (2018b). Attention models in graphs: A survey. *arXiv preprint arXiv:1807.07984*. 23

Lee, S., Kim, J.-W., Oh, Y., and Jeon, J. H. (2019b). Visual question answering over scene graph. In *2019 First International Conference on Graph Computing (GC)*, pages 45–50. IEEE. 54

Leskovec, J., Kleinberg, J., and Faloutsos, C. (2007). Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):2. 24, 27

Li, G., Qian, G., Delgadillo, I. C., Muller, M., Thabet, A., and Ghanem, B. (2020a). Sgas: Sequential greedy architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1620–1630. 80, 84

Li, H., Dong, W., Mei, X., Ma, C., Huang, F., and Hu, B.-G. (2019). Lgm-net: Learning to generate matching networks for few-shot learning. In *International conference on machine learning*, pages 3825–3834. PMLR. 87

Li, W., Gong, S., and Zhu, X. (2020b). Neural graph embedding for neural architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4707–4714. 83, 84

Li, X. and Jiang, S. (2019). Know more say less: Image captioning based on scene graphs. *IEEE Transactions on Multimedia*, 21(8):2117–2130. 54

Li, Y. and Gupta, A. (2018). Beyond grids: Learning graph representations for visual recognition. *Advances in Neural Information Processing Systems*, 31:9225–9235. 12, 13

Li, Y., Ouyang, W., Zhou, B., Wang, K., and Wang, X. (2017). Scene graph generation from objects, phrases and region captions. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1261–1270. 56

Li, Y., Tarlow, D., Brockschmidt, M., and Zemel, R. (2015). Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*. 12, 74

Li, Y., Yu, Q., Tan, M., Mei, J., Tang, P., Shen, W., Yuille, A., and Xie, C. (2020c). Shape-texture debiased neural network training. *arXiv preprint arXiv:2010.05981*. 90

Lian, D., Zheng, Y., Xu, Y., Lu, Y., Lin, L., Zhao, P., Huang, J., and Gao, S. (2020). Towards fast adaptation of neural architectures with meta learning. In *ICLR*. JMLR. org. 87

Liao, R., Li, Y., Song, Y., Wang, S., Nash, C., Hamilton, W. L., Duvenaud, D., Urtasun, R., and Zemel, R. S. (2019). Efficient graph generation with graph recurrent attention networks. *arXiv preprint arXiv:1910.00760*. 12

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017). Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988. 40

Lin, X., Ding, C., Zeng, J., and Tao, D. (2020). Gps-net: Graph property sensing network for scene graph generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3746–3753. 55, 57

Liu, C., Mao, J., Sha, F., and Yuille, A. L. (2017). Attention correctness in neural image captioning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. 29

Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L.-J., Fei-Fei, L., Yuille, A., Huang, J., and Murphy, K. (2018a). Progressive neural architecture search. In *Proceedings of the European conference on computer vision (ECCV)*, pages 19–34. 73, 84

Liu, H., Simonyan, K., and Yang, Y. (2018b). Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*. 4, 72, 73, 80, 84, 85, 86, 87, 88

Liu, H., Yan, N., Mortazavi, M., and Bhanu, B. (2021). Fully convolutional scene graph generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11546–11556. 37

Liu, I.-J., Peng, J., and Schwing, A. G. (2019). Knowledge flow: Improve upon your teachers. *arXiv preprint arXiv:1904.05878*. 88

Liu, M., Wang, Z., and Ji, S. (2020). Non-local graph neural networks. *arXiv preprint arXiv:2005.14612*. 13, 78

Locatello, F., Weissenborn, D., Unterthiner, T., Mahendran, A., Heigold, G., Uszkoreit, J., Dosovitskiy, A., and Kipf, T. (2020). Object-centric learning with slot attention. *arXiv preprint arXiv:2006.15055*. 15, 19, 37

Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440. 8

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110. 4, 8, 70

Lu, C., Krishna, R., Bernstein, M., and Fei-Fei, L. (2016). Visual relationship detection with language priors. In *European conference on computer vision*, pages 852–869. Springer. 17, 40, 41, 44, 54, 56

Lu, Y., Chang, C., Rai, H., Yu, G., and Volkovs, M. (2019). Learning effective visual relationship detector on 1 gpu. *arXiv preprint arXiv:1912.06185*. 56

Lu, Y., Chang, C., Rai, H., Yu, G., and Volkovs, M. (2021). Multi-view scene graph generation in videos. *International Challenge on Activity Recognition (ActivityNet) CVPR 2021 Workshop*. 56

Lukasik, J., Friede, D., Stuckenschmidt, H., and Keuper, M. (2020). Neural architecture performance prediction using graph neural networks. *arXiv preprint arXiv:2010.10024*. 84

Lust, J. and Condurache, A. P. (2020). A survey on assessing the generalization envelope of deep neural networks at inference time for image classification. *arXiv preprint arXiv:2008.09381*. 3

Ma, N., Bu, J., Yang, J., Zhang, Z., Yao, C., Yu, Z., Zhou, S., and Yan, X. (2020). Adaptive-step graph meta-learner for few-shot graph classification. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1055–1064. 22

Maas, A. L., Hannun, A. Y., Ng, A. Y., et al. (2013). Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3. Citeseer. 6

Makarov, I., Kiselev, D., Nikitinsky, N., and Subelj, L. (2021). Survey on graph embeddings and their applications to machine learning problems on graphs. *PeerJ Computer Science*, 7. 83

Mascharka, D., Tran, P., Soklaski, R., and Majumdar, A. (2018). Transparency by design: Closing the gap between performance and interpretability in visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4942–4950. 18

Metz, L., Maheswaranathan, N., Freeman, C. D., Poole, B., and Sohl-Dickstein, J. (2020). Tasks, stability, architecture, and compute: Training more effective learned optimizers, and using them to train themselves. *arXiv preprint arXiv:2009.11243*. 4, 70, 80, 86, 90

Meyer-Bäse, A., Foo, S., Tahmassebi, A., Meyer-Bäse, U., Amani, A. M., Götz, T., Leithner, D., Stadlbauer, A., and Pinker, K. (2020). Large-scale graph networks and ai applied to medical image data processing. In *Computational Imaging V*, volume 11396, page 1139605. International Society for Optics and Photonics. 12

Michaelis, C., Mitzkus, B., Geirhos, R., Rusak, E., Bringmann, O., Ecker, A. S., Bethge, M., and Brendel, W. (2019). Benchmarking robustness in object detection: Autonomous driving when winter is coming. *arXiv preprint arXiv:1907.07484*. 90

Milewski, V., Moens, M.-F., and Calixto, I. (2020). Are scene graphs good enough to improve image captioning? *arXiv preprint arXiv:2009.12313*. 54

Mirza, M. and Osindero, S. (2014). Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*. 55, 61

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. (2018). Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*. 64

Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J., and Bronstein, M. M. (2017). Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proc. CVPR*, volume 1, page 3. 27

Naeem, M. F., Oh, S. J., Uh, Y., Choi, Y., and Yoo, J. (2020). Reliable fidelity and diversity metrics for generative models. In *International Conference on Machine Learning*, pages 7176–7185. PMLR. xi, 66, 67

Naeem, M. F., Xian, Y., Tombari, F., and Akata, Z. (2021). Learning graph embeddings for compositional zero-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 953–962. 14, 16, 18

Newell, A. and Deng, J. (2017). Pixels to graphs by associative embedding. In *Advances in neural information processing systems*, pages 2171–2180. 41, 44

Neyshabur, B., Sedghi, H., and Zhang, C. (2020). What is being transferred in transfer learning? *arXiv preprint arXiv:2008.11687*. 85, 86

Nguyen, Q. and Hein, M. (2017). The loss surface of deep and wide neural networks. In *International conference on machine learning*, pages 2603–2612. PMLR. 77

Nilsson, A. and Bresson, X. (2020). An experimental study of the transferability of spectral graph networks. *arXiv preprint arXiv:2012.10258*. 10

Niu, Y., Tang, K., Zhang, H., Lu, Z., Hua, X.-S., and Wen, J.-R. (2020). Counterfactual vqa: A cause-effect look at language bias. *arXiv preprint arXiv:2006.04315*. 16, 54

Norcliffe-Brown, W., Vafeias, S., and Parisot, S. (2018). Learning conditioned graph structures for interpretable visual question answering. In *Advances in Neural Information Processing Systems*, pages 8334–8343. 15, 38

NT, H. and Maehara, T. (2019). Revisiting graph neural networks: All we have is low-pass filters. *arXiv preprint arXiv:1905.09550*. 11

NVIDIA (2021). Nvidia data center deep learning product performance. *nvidia.com*. 71

Park, D. H., Hendricks, L. A., Akata, Z., Schiele, B., Darrell, T., and Rohrbach, M. (2016). Attentive explanations: Justifying decisions and pointing to the evidence. *arXiv preprint arXiv:1612.04757*. 22, 25

Park, T., Liu, M.-Y., Wang, T.-C., and Zhu, J.-Y. (2019). Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2337–2346. 53, 64

Pei, H., Wei, B., Chang, K. C.-C., Lei, Y., and Yang, B. (2020). Geom-gcn: Geometric graph convolutional networks. *arXiv preprint arXiv:2002.05287*. 78

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543. 59

Peyre, J., Sivic, J., Laptev, I., and Schmid, C. (2017). Weakly-supervised learning of visual relations. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5179–5188. 17, 40

PyTorch (2021). Pytorch object detection finetuning tutorial. *PyTorch.org*. 85

Qiao, S., Wang, H., Liu, C., Shen, W., and Yuille, A. (2019). Micro-batch training with batch-channel normalization and weight standardization. *arXiv preprint arXiv:1903.10520*. 77

Raboh, M., Herzig, R., Berant, J., Chechik, G., and Globerson, A. (2020). Differentiable scene graphs. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1488–1497. 56

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. (2021). Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*. 90

Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*. 19, 61

Radiuk, P. M. (2017). Impact of training set batch size on the performance of convolutional neural networks for diverse datasets. *Information Technology and Management Science*, 20(1):20–24. 79

Radosavovic, I., Kosaraju, R. P., Girshick, R., He, K., and Dollar, P. (2020). Designing network design spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10428–10436. 88

Raghu, M., Zhang, C., Kleinberg, J., and Bengio, S. (2019). Transfusion: Understanding transfer learning for medical imaging. *arXiv preprint arXiv:1902.07208*. 85

Ratner, A. J., Ehrenberg, H., Hussain, Z., Dunnmon, J., and Ré, C. (2017). Learning to compose domain-specific transformations for data augmentation. In *Advances in neural information processing systems*, pages 3236–3246. 57

Ratzlaff, N. and Fuxin, L. (2019). Hypergan: A generative model for diverse, performant neural networks. In *International Conference on Machine Learning*, pages 5361–5369. PMLR. 88

Ravi, S. and Larochelle, H. (2016). Optimization as a model for few-shot learning. 70, 71, 80, 86, 88

Ravuri, S. and Vinyals, O. (2019). Seeing is not necessarily believing: Limitations of biggans for data augmentation. 55, 57

Real, E., Aggarwal, A., Huang, Y., and Le, Q. V. (2019). Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 4780–4789. 73

Ren, P., Xiao, Y., Chang, X., Huang, P.-Y., Li, Z., Chen, X., and Wang, X. (2020). A comprehensive survey of neural architecture search: Challenges and solutions. *arXiv preprint arXiv:2006.02903*. 4

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99. x, 8, 41, 45, 50, 60

Requeima, J., Gordon, J., Bronskill, J., Nowozin, S., and Turner, R. E. (2019). Fast and flexible multi-task classification using conditional neural adaptive processes. *arXiv preprint arXiv:1906.07697*. 87, 88

Rodríguez, P., Gonzalez, J., Cucurull, G., Gonfaus, J. M., and Roca, X. (2016). Regularizing cnns with locally constrained decorrelations. *arXiv preprint arXiv:1611.01967*. 20

Romero, A., Carrier, P. L., Erraqabi, A., Sylvain, T., Auvolat, A., Dejoie, E., Legault, M.-A., Dube, M.-P., Hussin, J. G., and Bengio, Y. (2016). Diet networks: thin parameters for fat genomics. *arXiv preprint arXiv:1611.09340*. 87

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*. 5, 71

Ruis, F., Burghours, G., and Bucur, D. (2021). Independent prototype propagation for zero-shot compositionality. *arXiv preprint arXiv:2106.00305*. 18

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252. 2, 8, 70, 72, 79

Sadeghi, M. A. and Farhadi, A. (2011). Recognition using visual phrases. In *CVPR 2011*, pages 1745–1752. IEEE. 56

Sandfort, V., Yan, K., Pickhardt, P. J., and Summers, R. M. (2019). Data augmentation using generative adversarial networks (cyclegan) to improve generalizability in ct segmentation tasks. *Scientific reports*, 9(1):1–9. 57

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2008). The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80. 2, 11, 12

Schmidhuber, J. (2020). Metalearning machines learn to learn (1987-). 4, 86

Schölkopf, B., Locatello, F., Bauer, S., Ke, N. R., Kalchbrenner, N., Goyal, A., and Bengio, Y. (2021). Toward causal representation learning. *Proceedings of the IEEE*, 109(5):612–634. 89

Schroeder, B. and Tripathi, S. (2020). Structured query-based image retrieval using scene graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 178–179. 54

Shaham, U., Stanton, K., Li, H., Nadler, B., Basri, R., and Kluger, Y. (2018). Spectralnet: Spectral clustering using deep neural networks. *arXiv preprint arXiv:1801.01587*. 12, 23

Shaw, P., Uszkoreit, J., and Vaswani, A. (2018). Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*. 12

Shelhamer, E., Rakelly, K., Hoffman, J., and Darrell, T. (2016). Clockwork convnets for video semantic segmentation. In *European Conference on Computer Vision*, pages 852–868. Springer. 8

Shen, Z., Liu, J., He, Y., Zhang, X., Xu, R., Yu, H., and Cui, P. (2021). Towards out-of-distribution generalization: A survey. *arXiv preprint arXiv:2108.13624*. 3, 89, 90

Shetty, R., Fritz, M., and Schiele, B. (2020). Towards automated testing and robustification by semantic adversarial data generation. In *European Conference on Computer Vision*, pages 489–506. Springer. 53

Shi, J., Zhang, H., and Li, J. (2019). Explainable and explicit visual reasoning over scene graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8376–8384. 15, 18, 54, 56

Shin, H.-C., Tenenholtz, N. A., Rogers, J. K., Schwarz, C. G., Senjem, M. L., Gunter, J. L., Andriole, K. P., and Michalski, M. (2018). Medical image synthesis for data augmentation and anonymization using generative adversarial networks. In *International workshop on simulation and synthesis in medical imaging*, pages 1–11. Springer. 57

Shrivastava, A. and Li, P. (2014). A new space for comparing graphs. In *Proceedings of the 2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 62–71. IEEE Press. 24, 27

Sifre, L. and Mallat, S. (2014). Rigid-motion scattering for texture classification. *arXiv preprint arXiv:1403.1687*. 76

Simonyan, K. and Zisserman, A. (2014a). Two-stream convolutional networks for action recognition in videos. *arXiv preprint arXiv:1406.2199*. 8

Simonyan, K. and Zisserman, A. (2014b). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*. 7, 73, 75

Sinha, K., Sodhani, S., Pineau, J., and Hamilton, W. L. (2020). Evaluating logical generalization in graph neural networks. *arXiv preprint arXiv:2003.06560*. 22

Snell, J., Swersky, K., and Zemel, R. S. (2017). Prototypical networks for few-shot learning. *arXiv preprint arXiv:1703.05175*. 87

Sperduti, A. and Starita, A. (1997). Supervised neural networks for the classification of structures. *IEEE Transactions on Neural Networks*, 8(3):714–735. 11

Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. (2014). Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*. 8

Srivastava, R. K., Greff, K., and Schmidhuber, J. (2015). Training very deep networks. *arXiv preprint arXiv:1507.06228*. 77

Stone, A., Wang, H., Stark, M., Liu, Y., Scott Phoenix, D., and George, D. (2017). Teaching compositionality to cnns. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5058–5067. 18

Strubell, E., Ganesh, A., and McCallum, A. (2019). Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*. 71

Su, W., Zhu, X., Cao, Y., Li, B., Lu, L., Wei, F., and Dai, J. (2019). Vl-bert: Pre-training of generic visual-linguistic representations. *arXiv preprint arXiv:1908.08530*. 37

Suhail, M., Mittal, A., Siddiquie, B., Broaddus, C., Eledath, J., Medioni, G., and Sigal, L. (2021). Energy-based learning for scene graph generation. *arXiv preprint arXiv:2103.02221*. 4, 17, 37, 55, 56, 63, 64, 65, 66

Sun, W. and Wu, T. (2020). Learning layout and style reconfigurable gans for controllable image synthesis. *arXiv preprint arXiv:2003.11571*. 57

Sylvain, T., Petrini, L., and Hjelm, D. (2019). Locality and compositionality in zero-shot learning. *arXiv preprint arXiv:1912.12179*. 18

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2013). Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*. 3, 28

Tang, K. (2020). Scene graph benchmark in pytorch. x, 47

Tang, K., Niu, Y., Huang, J., Shi, J., and Zhang, H. (2020). Unbiased scene graph generation from biased training. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. x, xv, 3, 15, 16, 17, 37, 39, 40, 41, 49, 50, 54, 55, 56, 57, 63, 64, 65, 66

Tang, K., Zhang, H., Wu, B., Luo, W., and Liu, W. (2019). Learning to compose dynamic tree structures for visual contexts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6619–6628. 17, 40, 44, 55, 56, 57

Thompson, N. C., Greenewald, K., Lee, K., and Manso, G. F. (2020). The computational limits of deep learning. *arXiv preprint arXiv:2007.05558*. 71

Tokmakov, P., Wang, Y.-X., and Hebert, M. (2019). Learning compositional representations for few-shot recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6372–6381. 14, 18

Touvron, H., Bojanowski, P., Caron, M., Cord, M., El-Nouby, A., Grave, E., Joulin, A., Synnaeve, G., Verbeek, J., and Jégou, H. (2021). Resmlp: Feedforward networks for image classification with data-efficient training. *arXiv preprint arXiv:2105.03404*. 7

Tripathi, S., Chandra, S., Agrawal, A., Tyagi, A., Rehg, J. M., and Chari, V. (2019a). Learning to generate synthetic data via compositing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 461–470. 53

Tripathi, S., Nittur Sridhar, S., Sundaresan, S., and Tang, H. (2019b). Compact scene graphs for layout composition and patch retrieval. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0. 54

Trivedi, R., Farajtabar, M., Biswal, P., and Zha, H. (2019). Dyrep: Learning representations over dynamic graphs. In *International Conference on Learning Representations*. 12

ul Hassan, M., Mulhem, P., Pellerin, D., and Quénot, G. (2019). Explaining visual classification using attributes. In *2019 International Conference on Content-Based Multimedia Indexing (CBMI)*, pages 1–6. IEEE. 14

Van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(11). 67

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008. 7, 12, 19, 22, 25, 76

Vedaldi, A. and Lenc, K. (2015). Matconvnet: Convolutional neural networks for matlab. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 689–692. 7

Vedantam, R., Desai, K., Lee, S., Rohrbach, M., Batra, D., and Parikh, D. (2019). Probabilistic neural-symbolic models for interpretable visual question answering. *arXiv preprint arXiv:1902.07864*. 15, 38

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. (2018). Graph attention networks. In *International Conference on Learning Representations (ICLR)*. 12, 23, 72

Veli*v*cković, P., Ying, R., Padovano, M., Hadsell, R., and Blundell, C. (2019). Neural execution of graph algorithms. *arXiv preprint arXiv:1910.10593*. 22

Verma, S. and Zhang, Z.-L. (2019). Stability and generalization of graph convolutional neural networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1539–1548. 22

Verma, V., Lamb, A., Beckham, C., Najafi, A., Mitliagkas, I., Lopez-Paz, D., and Bengio, Y. (2019). Manifold mixup: Better representations by interpolating hidden states. In *International Conference on Machine Learning*, pages 6438–6447. PMLR. 68

Verma, V., Luong, T., Kawaguchi, K., Pham, H., and Le, Q. (2021). Towards domain-agnostic contrastive learning. In *International Conference on Machine Learning*, pages 10530–10541. PMLR. 19

Vincent-Cuaz, C., Vayer, T., Flamary, R., Corneli, M., and Courty, N. (2021). Online graph dictionary learning. *arXiv preprint arXiv:2102.06555*. 22

Vondrick, C., Pirsiavash, H., and Torralba, A. (2016). Generating videos with scene dynamics. *Advances in neural information processing systems*, 29:613–621. 8

Wang, D., Beck, D., and Cohn, T. (2019a). On the role of scene graphs in image captioning. In *Proceedings of the Beyond Vision and LANguage: inTEgrating Real-world kNowledge (LANTERN)*, pages 29–34. 54

Wang, J., Chen, Y., Chakraborty, R., and Yu, S. X. (2020a). Orthogonal convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11505–11515. 20

Wang, L., Shi, J., Song, G., and Shen, I.-f. (2007). Object detection combining recognition and segmentation. In *Asian conference on computer vision*, pages 189–199. Springer. 85

Wang, W., Shen, J., and Shao, L. (2017). Video salient object detection via fully convolutional networks. *IEEE Transactions on Image Processing*, 27(1):38–49. 8

Wang, X., Sun, Q., ANG, M., and CHUA, T.-S. (2019b). Generating expensive relationship features from cheap objects. x, 17, 40, 44, 49, 53, 56, 57, 62, 63, 64

Wang, Y. G., Li, M., Ma, Z., Montufar, G., Zhuang, X., and Fan, Y. (2020b). Haar graph pooling. In *International conference on machine learning*, pages 9952–9962. PMLR. 22

Wen, W., Liu, H., Chen, Y., Li, H., Bender, G., and Kindermans, P.-J. (2020). Neural predictor for neural architecture search. In *European Conference on Computer Vision*, pages 660–676. Springer. 12, 84

Wen, W., Liu, H., Li, H., Chen, Y., Bender, G., and Kindermans, P.-J. (2019). Neural predictor for neural architecture search. *arXiv preprint arXiv:1912.00848*. 83

Wichrowska, O., Maheswaranathan, N., Hoffman, M. W., Colmenarejo, S. G., Denil, M., Freitas, N., and Sohl-Dickstein, J. (2017). Learned optimizers that scale and generalize. In *International Conference on Machine Learning*, pages 3751–3760. PMLR. 70, 80, 90

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. (2019). Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR. 12

Wu, L., Chen, Y., Shen, K., Guo, X., Gao, H., Li, S., Pei, J., and Long, B. (2021). Graph neural networks for natural language processing: A survey. *arXiv preprint arXiv:2106.06090*. 2

Wu, Y. and He, K. (2018). Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19. 77

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Philip, S. Y. (2020). A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*. 12, 22

Xian, Y., Akata, Z., Sharma, G., Nguyen, Q., Hein, M., and Schiele, B. (2016). Latent embeddings for zero-shot classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 69–77. 40

Xie, S., Girshick, R., Dollar, P., Tu, Z., and He, K. (2017). Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500. 73

Xu, D., Zhu, Y., Choy, C. B., and Fei-Fei, L. (2017). Scene graph generation by iterative message passing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5410–5419. x, xii, xiv, xv, 12, 15, 37, 38, 39, 41, 42, 44, 45, 46, 47, 49, 50, 51, 55, 56, 60, 62, 63, 64, 65

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2019a). How powerful are graph neural networks? In *International Conference on Learning Representations (ICLR)*. 12

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2019b). How powerful are graph neural networks? In *International Conference on Learning Representations (ICLR)*. 24, 26

Xu, P., Chang, X., Guo, L., Huang, P.-Y., Chen, X., and Hauptmann, A. G. (2020). A survey of scene graph: Generation and application. 54, 56

Yan, S., Shen, C., Jin, Z., Huang, J., Jiang, R., Chen, Y., and Hua, X.-S. (2020). Pcpl: Predicate-correlation perception learning for unbiased scene graph generation. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 265–273. 17, 55, 57

Yanardag, P. and Vishwanathan, S. (2015). Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1365–1374. ACM. 27

Yang, J., Lu, J., Lee, S., Batra, D., and Parikh, D. (2018a). Graph r-cnn for scene graph generation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 670–685. xii, 12, 15, 41, 42, 44, 56

Yang, X., Tang, K., Zhang, H., and Cai, J. (2019). Auto-encoding scene graphs for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10685–10694. 15, 37, 38, 54

Yang, X., Zhang, H., and Cai, J. (2018b). Shuffle-then-assemble: Learning object-agnostic visual relationship features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 36–52. x, 17, 40, 49, 56

Yang, Y., Wang, X., Song, M., Yuan, J., and Tao, D. (2021). Spagan: Shortest path graph attention network. *arXiv preprint arXiv:2101.03464*. 79

Yang, Z., Wang, Y., Chen, X., Shi, B., Xu, C., Xu, C., Tian, Q., and Xu, C. (2020). Cars: Continuous evolution for efficient neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1829–1838. 80, 84

Yehudai, G., Fetaya, E., Meirom, E., Chechik, G., and Maron, H. (2021). From local structures to size generalization in graph neural networks. In *International Conference on Machine Learning*, pages 11975–11986. PMLR. 3, 89

Yi, K., Wu, J., Gan, C., Torralba, A., Kohli, P., and Tenenbaum, J. B. (2018). Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. *arXiv preprint arXiv:1810.02338*. 14, 18

Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., and Leskovec, J. (2018). Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems*, pages 4805–4815. xiii, 12, 23, 25, 27, 29, 34

You, J., Leskovec, J., He, K., and Xie, S. (2020a). Graph structure of neural networks. xi, 77, 88

You, J., Ying, R., and Leskovec, J. (2019). Position-aware graph neural networks. In *International Conference on Machine Learning*, pages 7134–7143. PMLR. 79

You, J., Ying, R., Ren, X., Hamilton, W., and Leskovec, J. (2018). Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International conference on machine learning*, pages 5708–5717. PMLR. 12

You, J., Ying, Z., and Leskovec, J. (2020b). Design space for graph neural networks. *Advances in Neural Information Processing Systems*, 33. 88

Yu, J., Jin, P., Liu, H., Bender, G., Kindermans, P.-J., Tan, M., Huang, T., Song, X., Pang, R., and Le, Q. (2020). Bignas: Scaling up neural architecture search with big single-stage models. *arXiv preprint arXiv:2003.11142*. 87

Yu, Y., Chen, J., Gao, T., and Yu, M. (2019). Dag-gnn: Dag structure learning with graph neural networks. In *International Conference on Machine Learning*, pages 7154–7163. PMLR. 88

Yun, S., Jeong, M., Kim, R., Kang, J., and Kim, H. J. (2019). Graph transformer networks. *arXiv preprint arXiv:1911.06455*. 12

Zagoruyko, S. and Komodakis, N. (2016). Wide residual networks. *arXiv preprint arXiv:1605.07146*. 77

Zareian, A., Karaman, S., and Chang, S.-F. (2020a). Bridging knowledge graphs to generate scene graphs. In *European Conference on Computer Vision*, pages 606–623. Springer. 17, 55, 57

Zareian, A., You, H., Wang, Z., and Chang, S.-F. (2020b). Learning visual commonsense for robust scene graph generation. *arXiv preprint arXiv:2006.09623*. 55, 57

Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer. 30

Zellers, R., Yatskar, M., Thomson, S., and Choi, Y. (2018). Neural motifs: Scene graph parsing with global context. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5831–5840. x, 2, 15, 37, 39, 41, 42, 44, 45, 46, 47, 49, 50, 51, 56, 60, 62, 63, 64, 65

Zhai, X., Puigcerver, J., Kolesnikov, A., Ruyssen, P., Riquelme, C., Lucic, M., Djolonga, J., Pinto, A. S., Neumann, M., Dosovitskiy, A., et al. (2019). A large-scale study of representation learning with the visual task adaptation benchmark. *arXiv preprint arXiv:1910.04867*. 85

Zhang, C., Chao, W.-L., and Xuan, D. (2019a). An empirical study on leveraging scene graphs for visual question answering. *arXiv preprint arXiv:1907.12133*. 14, 38, 54, 56

Zhang, C., Ren, M., and Urtasun, R. (2018a). Graph hypernetworks for neural architecture search. *arXiv preprint arXiv:1810.05749*. 12, 70, 72, 73, 74, 78, 80, 81, 84, 87

Zhang, H., Dauphin, Y. N., and Ma, T. (2019b). Fixup initialization: Residual learning without normalization. *arXiv preprint arXiv:1901.09321*. 77

Zhang, H., Kyaw, Z., Chang, S.-F., and Chua, T.-S. (2017). Visual translation embedding network for visual relation detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5532–5540. x, 17, 40, 45, 49, 56

Zhang, J., Kalantidis, Y., Rohrbach, M., Paluri, M., Elgammal, A., and Elhoseiny, M. (2019c). Large-scale visual relationship understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9185–9194. 48

Zhang, J., Praneeth Karimireddy, S., Veit, A., Kim, S., Reddi, S. J., Kumar, S., and Sra, S. (2019d). Why adam beats sgd for attention models. *arXiv e-prints*, pages arXiv–1912. 82

Zhang, J., Shi, X., Xie, J., Ma, H., King, I., and Yeung, D.-Y. (2018b). Gaan: Gated attention networks for learning on large and spatiotemporal graphs. In *UAI*. 23

Zhang, J., Shih, K. J., Elgammal, A., Tao, A., and Catanzaro, B. (2019e). Graphical contrastive losses for scene graph parsing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11535–11543. x, 17, 40, 41, 44, 47, 50, 55, 56, 57

Zhang, L., Li, X., Arnab, A., Yang, K., Tong, Y., and Torr, P. H. (2019f). Dual graph convolutional network for semantic segmentation. *arXiv preprint arXiv:1909.06121*. 12

Zhang, L., Rosenblatt, G., Fetaya, E., Liao, R., Byrd, W. E., Might, M., Urtasun, R., and Zemel, R. (2018c). Neural guided constraint logic programming for program synthesis. *arXiv preprint arXiv:1809.02840*. 12

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M. (2020). Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81. 2

Zhou, K., Liu, Z., Qiao, Y., Xiang, T., and Loy, C. C. (2021). Domain generalization: A survey. *arXiv preprint arXiv:2103.02503*. 89

Zhou, L., Palangi, H., Zhang, L., Hu, H., Corso, J. J., and Gao, J. (2019). Unified vision-language pre-training for image captioning and VQA. *arXiv preprint arXiv:1909.11059*. 37

Zhu, C., Ni, R., Xu, Z., Kong, K., Huang, W. R., and Goldstein, T. (2021). Gradinit: Learning to initialize neural networks for stable and efficient training. *arXiv preprint arXiv:2102.08098*. 88

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232. 61

Zoph, B. and Le, Q. V. (2016). Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*. 4, 73

Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. (2018). Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710. 73