Title

The Design of Dynamic Neural Networks for Efficient Learning and Inference

Author

Wang, Xin

Publication Date

2020

Peer reviewed|Thesis/dissertation

The Design of Dynamic Neural Networks for Efficient Learning and Inference

by

Xin Wang

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Joseph E. Gonzalez, Co-chair
Professor Trevor Darrell, Co-chair
Professor Joshua Bloom

Fall 2020

The Design of Dynamic Neural Networks for Efficient Learning and Inference

Abstract

The Design of Dynamic Neural Networks for Efficient Learning and Inference

by

Xin Wang

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Joseph E. Gonzalez, Co-chair

Professor Trevor Darrell, Co-chair

Intelligent systems with computer vision capabilities are poised to revolutionize social interaction, entertainment, healthcare, and work automation. Recent progress in computer vision research, especially with deep learning methods, is largely driven by immense datasets, complex models, and billions of parameters. However, real-world applications often neither have access to large scale datasets nor can afford expensive data labeling and computation overhead. Furthermore, we live in a dynamic world, which requires machine learning models to handle changing data distributions and novel prediction tasks. When using learning techniques in actual systems, we still face significant hurdles coming from the scarcity of data and labels, limited computation, and never-ending varying prediction scenarios.

The focus of this thesis is the design of dynamic neural networks and its application to efficient learning (e.g., few-shot learning, semi-supervised learning) and inference. Dynamic networks adapt the model parameters and structures as a function of the input to leverage test-time information. For example, we introduce the design of dynamic neural architectures that can scale their computational complexity as a function of the input. To achieve small improvements in prediction accuracy, deep neural networks are rapidly increasing in depth and computational complexity. Also, the same amount of computation is applied regardless of the input. However, many inputs can reduce the level of computation required to render an accurate prediction. In Part I, we describe a range of dynamic models such as SkipNet and DeepMoE, which adjust the network depths on a per-input basis without reducing the prediction accuracy. In Part II, we describe the usage of dynamic neural networks for sample efficient learning. We propose dynamic weight generation using a task-aware meta learner and its application to a few-shot learning setting. We find that adapting the network parameters at prediction time improves model generalization.

To my family

Contents

List of Figures

List of Tables

Acknowledgments

There are no words to adequately express my gratitude and appreciation to my advisors Prof. Joseph E. Gonzalez and Prof. Trevor Darrell. It is their tremendous support and nurturing that makes the completion of my dissertation possible. I first got to know Joey and his work was at ICML 2014 in Beijing, when he was giving a tutorial on large scale machine learning. It was also the very first time that I attended an international academic conference, and at that time, I didn't expect I would become Joey's first Ph.D. student at Berkeley one year later. During my Ph.D., Joey spent countless hours discussing the research agenda, giving feedback on projects, sharing his wisdom to conduct high-impact work, polishing my paper drafts, and so on. Joey has influenced my academic career deeply, and I'm just so fortunate to have him by my side along my Ph.D. journey. I would also extend my deepest gratitude to Trevor, who has given me a lot of guidance since I entered the Computer Vision area. Trevor is a person with superior wisdom and omniscient insight. I can always learn something when meeting with Trevor. He has shaped my academic career deeply and taught me how to be a great researcher and academic. I'm grateful for the time he spent reviewing my career objectives and recommending strategies for achieving them. I also want to thank Prof. Kurt Keutzer and Professor Joshua Bloom for being on my Qual exam committee. Their feedback is very valuable for the completion of this dissertation.

I would like to thank the faculty members in RISE Lab and BAIR Lab for their feedback on my work. Prof. Ion Stoica had provided a lot of guidance when I was working on the Clipper project. I can always learn something new when chatting with Prof. Alyosha Efros and Prof. Jitendra Malik. I'm fortunate to have the opportunity to learn from these pioneers in my area.

I'm grateful to my collaborators, whose support is the source of force to my research achievements. My first research paper was published, working together with Daniel Crankshaw. For the first two years of my Ph.D., Dan and I have worked on a range of research projects around model serving and fast prediction. During the course, I have also worked with Alexey Tumanov and learned many system knowledge from him. I also want to extend my gratitude to Fisher Yu, who first introduced me to the area of Computer Vision and inspired me to pursue a rewarding path of studying vision systems research. Fisher and I have worked closely on numerous projects, and he has always been a good mentor and collaborator. I also want to thank Xiaolong Wang, Azalia Mirhoseini, Thomas Huang, Bingyi Kang, Zhuang Liu, Zuxuan Wu, and Yanzhao Zhou for their successful collaboration, which composes the large portion of this dissertation.

I would like to thank my colleagues and friends at BAIR and RISELab: Sayna Ebrahimi, Anna Rohrbach, Seth Park, Devin Guillory, Samaneh Azadi, Mike Laielli, Yang Gao, Dequan Wang, Shizhan Zhu, Rudy Corona, Huazhe Xu, Huijuan Xu, Roi Herzig, Amir Bar, Tete Xiao, Medhini Narasimhan, Suzie Petryk in BAIR and Wenting Zheng, Neeraja Yadwadkar, Shishir Patil, Tianjun Zhang, Vikram Sreekanti, Chenggang Wu, Daniel Rothchild, Anurag khandelwal, Zongheng Yang, Stephanie Wang, Robert Nishihar, Philipp Moritz from RISE Lab. I also want to thank the admins in RISE and BAIR for their help: Kattt Atchley, Jon Kuroda, Boban Zarkovich, Shane Knapp, David Schonenberg, and Angie Abbatecola.

Ph.D. life won't be easy without the companion and support from my friends. I would like to thank Jenny Huang, Junyu Cao, Mengqiao Yu, Xuaner Zhang, Biye Jiang, Tianhao Zhang, Haoran

Tang, Rocky Duan, Hezheng Yin, Yichao Zhou, Xu Rao, Zhewei Yao at Berkeley, Yuxi Zhang, Ruotian Luo, Xiaowen Li, Yanjing Chen, Tianyuan Liu, Licheng Yu, Xinchen Yan, Lin Li from SJTU and Bo Feng, Yue Ming from my high school. I also want to thank Jiangmiao Pang, Zhichao Yin, Haofeng Chen, Linlu Qiu, and Ruth Wang to have a great time together when they visited Berkeley.

Finally, this thesis is dedicated to my parents for their selfless love and support over the years.

Chapter 1

Introduction

1.1 Motivation

Intelligent systems with computer vision capabilities are poised to revolutionize social interaction, entertainment, healthcare, and work automation, reshaping every aspect of our life. Self-driving vehicles empowered by advanced visual perception and control technologies are reshaping the future transportation, providing a more accessible and efficient means of traveling. Intelligent medical diagnosis and treatment offers new promises to advance our healthcare through assisted disease discovery, surgical robots, and so on. Smart home appliances bring much convenience to our daily life, releasing people from repetitive housework.

Recent progress in computer vision research, especially with deep learning methods, is largely driven by *immense datasets*, *complex models*, and *billions of parameters*. However, real-world applications often neither have access to large scale datasets nor can afford expensive data labeling and computation overhead. Furthermore, we live in a *dynamic* world, which requires machine learning models to handle changing data distributions and novel prediction tasks. When using learning techniques in actual systems, we still face significant hurdles coming from the scarcity of data and labels, limited computation, and never-ending varying prediction scenarios.

Learning with limited data and labels is critical in the area of machine learning and computer vision, which has been studied in various learning settings, such as few-shot learning, semi-supervised learning, and unsupervised/self-supervised learning. For example, few-shot learning aims to classify new concepts, having seen only a few training examples. There might only be a single example of each class (one-shot learning) in the extreme. Few-shot learning is useful, especially for learning the rare concepts where the training data is difficult to obtain. In the past decades, researchers have designed various approaches, such as approaches based on meta-learning and metric learning, conquering fundamental machine learning tasks (e.g., few-shot image classification), and have expanded the study into emerging structural prediction tasks in 2D and 3D computer vision. Semi-supervised learning leverages unlabeled training data, and unsupervised/self-supervised learning solely learns feature representations from unlabeled data, which can be used to improve the data efficiency of downstream tasks. Improving the data and label efficiency is crucial

to applying machine learning models to real-world applications and a key focus of this thesis work.

Learning with limited computation is also critical for applying machine learning model to real-world applications, which aims to reduce the computational cost when rending predictions. Researchers have developed various techniques, such as model pruning and quantization to compress models, designing compact models. More recently, researchers have also developed the co-design of machine learning models and hardware to optimize the inference cost. However, efficient inference still requires a customized solution for each application, which is challenging given the large volume of the types of edge devices and application tasks.

This thesis aims to study the design of a special class of neural networks, dynamic neural networks for efficient learning and inference, which improves the efficiency of learning and inference in the unified framework.

1.2 Review on Dynamic Neural Networks

In this section, we review the recent progress of dynamic neural networks. The concept of dynamic neural networks are inspired by human brains, which activate a different part of the brain for various tasks. The *Thinking, Fast and Slow* book by Daniel Kahneman, a Nobel prize laureate, suggests that human brains have two thinking systems. The slow system is adopted to make more delicate and complicated decisions. In contrast, the fast system is used for making more intuitive and straightforward decisions. Dynamic neural networks share a similar philosophy, which activates different part of the neural networks based on the input.

Dynamic neural networks for fast prediction can trace back to Bengio *et al.* [9], which adopts a conditional computation approach to select neurons to activate through a gradient estimator. However, the original work doesn't show how the resulting neural network's sparsity reduces the inference cost on real-world edge devices. More recently, many other works [52, 63, 173] have proposed to adjust computation by examining early termination adaptively. Graves *et al.* [63] proposes to predict halting in recurrent networks to save computational cost. Figurnov *et al.* [52] and Teerapittayanon *et al.* [173] propose the use of early termination in convolutional networks.

Another line of work in dynamic neural networks have focused on designing compositional representations to improve data efficiency. For example, researchers [134] have proposed the compositional zero-shot learning (CZSL) of attribute-object categories, a special case of zero-shot learning. In CZSL, each visual concept (category) is represented by a attribute-object pair (e.g., `red elephant`, `modern city`, etc.) or a subject-predicate-object (SPO) triplet (e.g., `person ride horse`, `dog on grass`, etc. SPO triplet is used in the StanfordVRD dataset). Like ZSL, images of only a subset of compositions are labeled during training, and the model is learned to recognize unseen object-attribute compositions during inference. The compositionality and the contextual nature of the task make it interesting for compositional reasoning and sample efficient learning. Many [134, 137, 145] propose various approaches based on metric-learning and meta-learning to learn the compositional feature representation. Tokmakov *et al.* [174] learns compositional representation for few-shot recognition.

1.3 Thesis Overview and Contributions

In this thesis, we study various designs of dynamic neural networks and their application to efficient inference in Part I (chapter 2 to 4) and efficient learning (e.g., few-shot learning, semi-supervised learning, etc.) in Part II (chapter 5 to 8).

In Part I, we will introduce the design of dynamic neural architectures that can scale their computational complexity as a function of the input. To achieve small improvements in prediction accuracy, deep neural networks are rapidly increasing in depth and computational complexity. Also, the same amount of computation is applied regardless of the input. However, many inputs can reduce the level of computation required to render an accurate prediction. We propose SkipNet [188] in Chapter 2 and introduce gating networks and latent embeddings to activate and deactivate layers as a function of the input image. We also introduce a hybrid reinforcement learning approach to learn the gating policy. On various image classification benchmark, SkipNet reduces computation by 30-90% while preserving the accuracy and outperformed state-of-the-art static compression methods.

In Chapter 3, we introduce DeepMoE [184], which learns a channel-level dynamic routing scheme based on the shallow embeddings of the input image. In DeepMoE, the per-input selection decision is decided before executing the prediction network, while SkipNet makes skipping decisions during the forward passing of the prediction network. This strategy allows batching of examples with the same inference path to improve the network throughput on computing units with a massive parallel structure such as GPUs. DeepMoE also provides finer grain selection policies, which further improves the computation efficiency.

Chapter 4 introduces an IDK cascade model, where we stack a sequence of models with various complexity levels. We introduce an additional "I don't know" (IDK) class, and only when the cheap model in the cascade predicts IDK, the prediction task will proceed to a more expensive model. Based on this calibrated uncertainty score, the IDK cascade model adjusts the prediction cost based on the input image's complexity. IDK cascade can be used for pre-trained models from the user and requires no knowledge about the model's architecture, which can be readily used in low latency model serving systems, such as Clipper [27].

Starting in Chapter 5, we will study a line of research to improve learning efficiency. Humans' capability to learn new concepts from a few examples or even only descriptions is envied by generations of machine learning researchers. This data-efficient learning is vital for learning systems in the age of deep learning, which is notoriously hungry for supervision. Motivated by the dynamic nature of human brains, In Chapter 5, we discuss TAFE-Net [189], which generates the new network weights based on the novel concepts to construct task-aware feature embeddings, achieving fast knowledge adaptation without accessing any labeled data (a.k.a zero-shot learning). Instead of generating the full weight matrices, we adopt a factorization scheme to generate lightweight task-specific weights and the heavy shared weights across all tasks once. TAFE-Net established the state-of-the-art on the various low-shot image classification benchmarks.

Chapter 6 discusses how to apply dynamic weight generation to structural prediction task, few-shot object detection. Similar to TAFE-Net, the proposed few-shot re-weighting (FSRW) approach uses as input a meta learner, which takes the supporting images and bounding boxes from each

category to generate class-aware weights that are used to re-weight the generic features. We were the first to establish the few-shot object task on two large scale object detection datasets, and the later work has widely used the formulation in the area.

In Chapter 7, we further study an alternative approach using fine-tuning for few-shot object detection. Meta-learning prevails in few-shot learning, given its premise on fast adaptation. However, meta training often requires large memory consumption. The meta learner design varies from case to case, making it hard to deploy in real-world applications, such as adapting an object detector for a specific user on their mobile phone. Therefore, We revisited the simple fine-tuning methods and conducted a systematic study about the proper training schemes and model capacity of fine-tuning methods for few-shot object detection. In this work [185], we found that fine-tuning the object detector in a controlled manner, that is, only the last layer of the detector is fine-tuned given examples from the new categories, can outperform the previous meta learning-based methods by a large margin. Besides, this controlled fine-tuning approach naturally breaks the memory barrier of updating deep networks on edge devices. In the recent few-shot object detection challenge on the latest LVIS benchmark co-located with ECCV'20, most of the participants adopted our fine-tuning scheme as part of their algorithms.

In Chapter 8, we look into the application of semi-supervised instance segmentation. We propose ShapeProp, which learns to activate the salient regions within the object detection and propagate the areas to the whole instance through an iterative learnable message passing module. ShapeProp can benefit from more bounding box supervision to locate the instances more accurately and utilize the feature activations from the larger number of instances to achieve more accurate segmentation. We extensively evaluate ShapeProp on three datasets (MS COCO, PASCAL VOC, and BDD100k) with different supervision setups based on both two-stage (Mask R-CNN) and single-stage (RetinaMask) models. ShapeProp established a state of the art for semi-supervised instance segmentation.

This thesis focuses on various dynamic neural networks where the network architecture, parameter, and execution paths are adjusted at inference time. We discuss the applications of dynamic neural networks to efficient learning and inference. And we hope this in-depth discussion on dynamic neural networks can foster more future study in this line of work and unveil more interesting applications.

Part I

Efficient Inference

Chapter 2

Block-Level Dynamic Routing

2.1 Overview

Deep convolutional neural networks are the enabling technology behind the recent rapid progress in computer vision. A growing body of research in convolutional network design [73, 103, 165, 172] reveals a clear trend: *deeper networks are more accurate*. Consequently, the best-performing image recognition networks have tens of millions of parameters and hundreds of layers. While commodity GPUs are able to substantially accelerate training, the high computation cost of deep networks hinders their deployment in latency sensitive end-user applications and on low-power devices. Moreover, the depth of these networks results in fundamental and significant increases in prediction latency.

The continuous improvements in accuracy, while significant, are small relative to the growth in model complexity. A network that doubles in depth may only improve by a few percentage points on key benchmarks. These small improvements are critical to the adoption of these models in real-world applications; however, they imply that only a small fraction of images require very deep representations and thus the vast majority of images could be accurately processed using shallower architectures.

In this work, we study the design of *dynamically executed networks* (SkipNets), neural networks that determine which layers of a convolutional neural network should be skipped when processing a given image at inference, illustrated in Fig. 2.1. We frame the dynamic skipping problem as a sequential decision problem in which the outputs of previous layers are used to decide whether to bypass the subsequent layer. The objective in the dynamic skipping problem is then to skip as many layers as possible while retaining the accuracy of the full network. Not only can skipping policies significantly reduce the average cost of model inference they also provide insight into the diminishing return and role of individual layers.

While conceptually simple, learning an efficient skipping policy is challenging. To achieve a reduction in computation, we need to bypass the correct layers in the network. This inherently discrete decision is not differentiable, and therefore precludes the application of established supervised learning methods based on gradient based optimization. Although one could introduce a soft

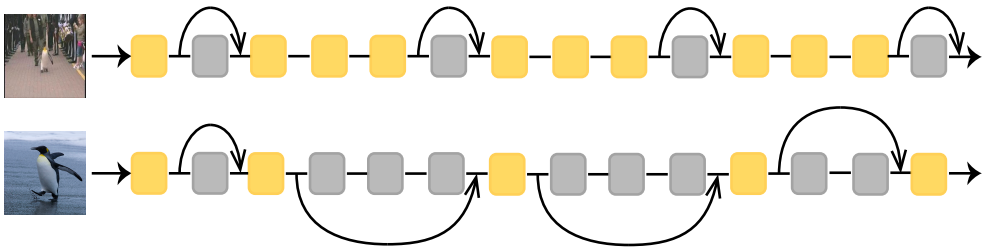


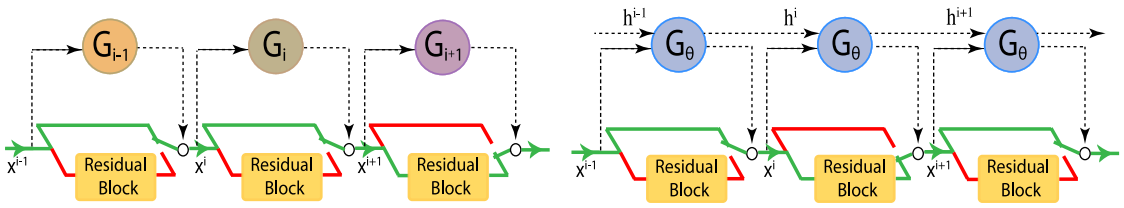
Figure 2.1: SkipNet illustration.

approximation similar to soft-attention techniques [14, 177, 179], we show that the subsequent hard thresholding required to achieve a reduction in the cost of computation results in low prediction accuracy.

Similar to our goals, recent research has made progress in applying reinforcement learning (RL) techniques to address hard attention in recurrent models [37, 136]. While these techniques are promising, in our experiments we show that these RL based techniques are brittle, often getting stuck in poor local minima and producing networks that are not competitive with the state-of-the-art. One can also apply the reparametrization techniques [90, 130], however, these approaches often find suboptimal policies partly due to the approximation error introduced by the relaxation (detailed in later sections).

In this work we explore several SkipNet designs and introduce a hybrid learning algorithm which combines supervised learning with reinforcement learning to address the challenges of non-differentiable skipping decisions. We explicitly assign a gating module to each group of layers. The gating module consumes the output of the previous layers and then decides if the subsequent group of layers should be skipped. We adopt two stages to learn the gating modules. First, we relax integer skipping decisions to continuous values with soft-max, a reparameterization trick similar to [90, 130], and train the layers and gates jointly with supervised outputs. Then, we treat the probabilistic gate outputs as an initial skipping policy and use REINFORCE [195] to refine the policy without relaxation. In the latter stage, we jointly optimize the skipping policy and prediction error to stabilize the exploration process.

We evaluate SkipNets, using ResNets [73] as the base models, on the CIFAR-10, CIFAR-100, SVHN and ImageNet datasets. We show that, with the hybrid learning procedure, SkipNets learn skipping policies that significantly reduce model inference costs (50% on the CIFAR-10 dataset, 37% on the CIFAR-100 dataset, 86% on the SVHN dataset and 30% on the ImageNet dataset) while preserving accuracy. We compare SkipNet with several state-of-the-art models and techniques on both CIFAR-10 and ImageNet datasets and find that SkipNet consistently outperforms the previous methods on both benchmarks. By manipulating the computational cost hyper-parameter, we show how SkipNets can be tuned for different computation constraints. Finally, we study the skipping behavior of the learned skipping policy and reveal that it learns to identify more challenging images and route those images through more layers of the network.



(a) Feed-forward Gate (b) Recurrent Gate

Figure 2.2: Gate designs in SkipNets.

2.2 SkipNet: Skipping Layers on a Per Input Basis

SkipNets are convolutional networks in which individual layers are selectively included or excluded for a given input. The per-input selection of layers is accomplished using small gating networks that are interposed between layers. The gating networks map the output of the previous layer or group of layers to a binary decision to execute or bypass the subsequent layer or group of layers as illustrated in Figure 2.2.

More precisely, let $\mathbf{x}^i$ be the input and $F^i(\mathbf{x}^i)$ be the output of the i^{th} layer or group of layers, then we define the output of the gated layer (or group of layers) as:

$$\mathbf{x}^{i+1} = G^i(\mathbf{x}^i)F^i(\mathbf{x}^i) + (1 - G^i(\mathbf{x}^i))\mathbf{x}^i, \tag{2.1}$$

where $G^i(\mathbf{x}^i) \in \{0, 1\}$ is the gating function for layer i. In order for Eq. 2.1 to be well defined, we require $F^i(\mathbf{x}^i)$ and $\mathbf{x}^i$ to have the same dimensions. This requirement is satisfied by commonly used residual network architectures where

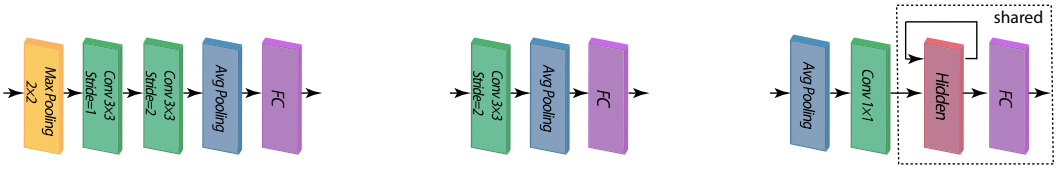
$$\mathbf{x}^{i+1}_{\text{ResNet}} = F^i(\mathbf{x}^i_{\text{ResNet}}) + \mathbf{x}^i_{\text{ResNet}}, \tag{2.2}$$

and can be easily addressed by pooling or up-sampling $\mathbf{x}^i$ so its dimensions match that of $F^i(\mathbf{x}^i)$.

When developing a dynamically executed network we must address several key design challenges. First, the gating networks need to be sufficiently expressive to effectively decide whether to execute or skip individual layers for a given input without degrading prediction accuracy. Second, the gating network must also be computationally light when compared to the execution of the layer. To address the trade-off between expressivity and computational cost we explore a range of gating network designs (Sec. 2.2) spanning basic feed-forward convolutional architectures to recurrent networks with varying degrees of parameter sharing.

Finally, learning the gating network parameters as well as the base network parameters is complicated by the presence of hard decisions at the gate level in conjunction with the need to learn gating parameters that preserve accuracy while minimizing computational costs. To address this challenge we introduce a two stage training algorithm that combines supervised pre-training (Sec. 2.2) with RL based policy optimization (Sec. 2.2) for the hard gated decisions using a hybrid reward function that combines prediction accuracy with the computational cost.

Gating Network Design



(a) FFGate-I (b) FFGate-II (c) RNNGate

Figure 2.3: FFGate and RNNGate designs.

The feed-forward gate design (Fig. 2.2a) depends only on the output of the previous layer and applies a small number of convolution and pooling operations. In this paper, we evaluate two feed-forward gate designs. *FFGate-I* (Fig. 2.3a) is composed of two 3×3 convolutional layers with stride of 1 and 2 respectively followed by a global average pooling layer and a fully connected layer to output a single dimension vector. To reduce the gate computation, we add a 2×2 max pooling layer prior to the first convolutional layer. The overall computational cost of FFGate-I is roughly 19% of the residual blocks [73] used in this paper.

The *FFGate-I* design is still relatively expensive and is not well suited for deeper networks. We therefore introduce the *FFGate-II* (Fig. 2.3b), consisting of one 3×3 stride 2 convolutional layer followed by the same global average pooling and fully connected layers as *FFGate-I* to further reduce computation of gates (roughly 12.5% of the residual block). We use the computationally less expensive *FFGate-II* for extremely deep networks (greater than 100 layers) in our experiments and *FFGate-I* for shallow networks.

Even with a limited number of convolutions, the feed-forward gate design is still relatively costly to compute and does not leverage the decisions from previous gates. Therefore, we introduce a *recurrent gate* (RNNGate) design (Fig. 2.3c) which enables parameter sharing and allows gates to re-use computation across stages. We first apply global average pooling on the input feature map of the gates and then linearly project the feature to the input size of 10. We adopt a single layer *Long Short Term Memory* [80] (LSTM) with hidden unit size of 10. At each gate, we project the LSTM output to a one-dimensional vector to compute the final gate decision. Compared to the cost of computing residual blocks, the cost of this recurrent gate design is negligible (roughly 0.04% of the computation of residual blocks).

In our later experiments, we find that the recurrent gate is actually more superior than feed-forward gates in terms of prediction accuracy and computation cost. We also try to remove the convolutional layers in the feed-forward gate to match the computation cost of recurrent gates but the resulting gate doesn't work well in the experiments. One reason may be that the recurrent gate design better captures the cross-layer correlation by passing the hidden vectors through different layers.

Skipping Policy Learning with Hybrid RL

During the process of rendering a prediction (inference), the most likely action is taken from the probability distribution encoded by each gate: *the layer is skipped or executed*. This inherently *discrete* and therefore non-differentiable decision process creates unique challenges for how we train SkipNets. A natural approximation, similar to that used in Highway Networks [170], would be to use differentiable soft-max decisions during training and then revert to hard decisions during inference. While this approach enables gradient based training, it results in poor prediction performance (Sec. 2.3) as the network parameters are not optimized for the subsequent hard-gating during inference. We therefore explore the use of reinforcement learning to learn the model parameters for the non-differentiable decision process.

Because SkipNets make a sequence of discrete decisions, one at each gated layer, we frame the task of estimating the gating function in the context of policy optimization through reinforcement learning. We define the skipping policy:

$$\pi(\mathbf{x}^i, i) = \mathbb{P}(G^i(\mathbf{x}^i) = g_i) \tag{2.3}$$

as a function from the input $\mathbf{x}^i$ to the probability distribution over the gate action g_i to execute ($g_i = 1$) or skip ($g_i = 0$) layer i. We define a sample sequence of gating decisions drawn from the skipping policy starting with input $\mathbf{x}$ as:

$$\mathbf{g} = [g_1, \ldots, g_N] \sim \pi_{F_\theta}(\mathbf{x}), \tag{2.4}$$

where $F_\theta = \left[F_\theta^1, \ldots, F_\theta^N\right]$ is the sequence of network layers (including the gating modules) parameterized by θ and $\mathbf{g} \in \{0, 1\}^N$. The overall objective is defined as

$$\begin{aligned} \min \mathcal{J}(\theta) &= \min \mathbb{E}_{\mathbf{x}} \mathbb{E}_{\mathbf{g}} L_\theta(\mathbf{g}, \mathbf{x}) \\ &= \min \mathbb{E}_{\mathbf{x}} \mathbb{E}_{\mathbf{g}} \left[\mathcal{L}(\hat{y}(\mathbf{x}, F_\theta, \mathbf{g}), y) - \frac{\alpha}{N} \sum_{i=1}^{N} R_i \right], \end{aligned} \tag{2.5}$$

where $R_i = (1 - g_i)C_i$ is the reward of each gating module. The constant C_i is the cost of executing F^i and the term $(1 - g_i)C_i$ reflects the reward associated with *skipping* F^i. In our experiments, all F^i have the same cost and so we set $C_i = 1$. Finally α is a tuning parameter that allows us to trade-off the competing goals of minimizing the prediction loss and maximizing the gate rewards.

To optimize this objective, we can derive the gradients with respect to θ as follows. We define $\pi_{F_\theta}(\mathbf{x}) = p_\theta(\mathbf{g}|\mathbf{x})$, $\mathcal{L} = \mathcal{L}(\hat{y}(\mathbf{x}, F_\theta, \mathbf{g}), y)$ and $r_i = -[\mathcal{L} - \frac{\alpha}{N} \sum_{j=i}^{N} R_j]$.

$$\begin{aligned} \nabla_\theta \mathcal{J}(\theta) &= \mathbb{E}_{\mathbf{x}} \nabla_\theta \sum_{\mathbf{g}} p_\theta(\mathbf{g}|\mathbf{x}) L_\theta(\mathbf{g}, \mathbf{x}) \\ &= \mathbb{E}_{\mathbf{x}} \sum_{\mathbf{g}} p_\theta(\mathbf{g}|\mathbf{x}) \nabla_\theta \mathcal{L} + \mathbb{E}_{\mathbf{x}} \sum_{\mathbf{g}} p_\theta(\mathbf{g}|\mathbf{x}) \nabla_\theta \log p_\theta(\mathbf{g}|\mathbf{x}) L_\theta(\mathbf{g}, \mathbf{x}) \\ &= \mathbb{E}_{\mathbf{x}} \mathbb{E}_{\mathbf{g}} \nabla_\theta \mathcal{L} - \mathbb{E}_{\mathbf{x}} \mathbb{E}_{\mathbf{g}} \sum_{i=1}^{N} \nabla_\theta \log p_\theta(g_i|\mathbf{x}) r_i. \end{aligned} \tag{2.6}$$

Algorithm 1: Hybrid Learning Algorithm (HRL+SP)

Input: A set of images $\mathbf{x}$ and labels $\mathbf{y}$
Output: Trained SkipNet
1. Supervised pre-training (Sec. 2.2)
$\theta_{SP} \leftarrow$ SGD($L_{\text{Cross-Entropy}}$, SkipNet-$G_{\text{relax}}(\mathbf{x})$)
2. Hybrid reinforcement learning (Sec. 2.2)
Initialize θ_{HRL+SP} with θ_{SP}
$\theta_{HRL+SP} \leftarrow$ REINFORCE($\mathcal{J}$, SkipNet-$G(\mathbf{x})$)

As we can see from Eq. 2.6, the first component can be obtained naturally by supervised learning while the second part has the same form as the REINFORCE [195] where r_i is the cumulative future rewards associated the gating modules. Since our optimization procedure is composed of both reinforcement learning and supervised learning, we refer this procedure as *hybrid reinforcement learning*. In practice, we relax the reward $\hat{r}_i = -\left[\beta\mathcal{L} - \frac{\alpha}{N}\sum_{j=i}^{N} R_j\right]$ to scale down the influence of the prediction loss as this hybrid reinforcement learning is followed by the supervised pre-training that will be discussed in the next section. We set $\beta = \frac{\alpha}{N}$ in our experiments.

Supervised Pre-training

Optimizing Eq. 2.5 starting from random parameters also consistently produces models with poor prediction accuracy (Sec. 2.3). We conjecture that the degraded ability to learn an accurate model is due to interference between policy learning and image representation learning. The skipping policy can over-fit to premature features which prevents the network from exploring different gating patterns and results in less effective feature learning.

To provide an effective supervised initialization procedure we introduce a form of supervised pre-training that combines hard-gating during the forward pass with *soft-gating* during backpropagation. We relax the gate outputs $G(\mathbf{x})$ in Eq. 2.1 to continuous values, i.e. approximating $G(\mathbf{x})$ by $S(\mathbf{x}) \in [0, 1]$. We round the output gating probability of the skipping modules to 0 or 1 in the forward pass. During backpropagation we use the soft-max approximation [90, 130]. That is, the gate is restored to soft outputs and the gradients to the soft-max outputs can be calculated accordingly. The relaxation procedure is summarized by:

$$G_{\text{relax}}(\mathbf{x}) = \begin{cases} \mathbb{I}(S(\mathbf{x}) \geq 0.5), & \text{forward pass} \\ S(\mathbf{x}), & \text{backward pass} \end{cases}, \tag{2.7}$$

where $\mathbb{I}(\cdot)$ is the indicator function. This hybrid form of supervised pre-training is able to effectively leverage labeled data to initialize model parameters for both the primary network and the gating networks. After supervised pre-training we then apply the REINFORCE algorithm to refine the model and gate parameters improving prediction accuracy and further reducing prediction cost. A summary of our two stage hybrid algorithm is given in Alg. 1.

2.3 Experiments

We evaluate a range of SkipNet architectures and our proposed training procedure on four image classification benchmarks: CIFAR-10 [100], CIFAR-100 [100], SVHN [138] and ImageNet 2012 [157]. We construct SkipNet models from ResNet models [73] by introducing hard gates between residual blocks. In Section 2.3, we evaluate the performance of SkipNets with both gate designs and compare SkipNets with the state-of-the-art models including dynamic networks (i.e. SACT and ACT [52]) and static compression networks (i.e. PEFC [113], LCCL [42] and EP [125]) which are also complementary approaches to our methods. We also compare our approach with naive baselines with random block dropping to demonstrate the effectiveness of the learned skipping policy. Furthermore, We study the trade-off between computation cost and accuracy. In Section 2.3, we analyze the design choices and visualize the skipping patterns of SkipNets.

Setup

Datasets. Table 2.1 summarizes the statistics of the datasets used in this paper. We follow the common data augmentation scheme (mirroring/shifting) that is adopted for CIFAR and ImageNet datasets [62, 111, 116, 183]. For the SVHN dataset, we used both the training dataset and provided extra dataset for training and did not perform data augmentation. For preprocessing, we normalize the data with the channel means and standard deviations for all the datasets used in this paper.

Table 2.1: Dataset statistics.

Dataset	Train Size	Test Size	# Classes
CIFAR-10	50,000	10,000	10
CIFAR-100	50,000	10,000	100
SVHN	604,388 [1]	26,032	10
ImageNet-12	1.28 million	50,000	1000

Models. For the CIFAR and SVHN datasets, we use the ResNet [73] architecture with $6n + 2$ stacked weighted layers for our base models and choose $n = \{6, 12, 18, 25\}$ to construct network instances with depth of $\{38, 74, 110, 152\}$. For the ImageNet dataset, we evaluate ResNet-34, ResNet-50 and ResNet-101 as described in [73]. We denote our model at various depth by SkipNet-x (x is the depth of the base model). We show the accuracy numbers of the base models in Table 2.2. In later sections, we demonstrate SkipNets can preserve the same accuracy (within a variance of 0.5%).

Training. Our two-stage training procedure combines both supervised pre-training and policy refinement with hybrid reinforcement learning. In the supervised training stage, for the CIFAR and ImageNet datasets, we adopt the same hyper-parameters used in [73]. For the SVHN dataset, we follow the hyper-parameters used in [84].

[1] 531,131 of the images are extra images of SVHN for additional training.

Table 2.2: Top-1 accuracy of the original ResNets

Model	CIFAR-10	CIFAR-100	SVHN	Model	ImageNet
ResNet-38	92.50%	68.54%	97.94%	ResNet-34	73.30%
ResNet-74	92.95%	70.64%	97.92%	ResNet-50	76.15%
ResNet-110	93.6%	71.21%	98.09%	ResNet-101	77.37%

For the policy refining stage, we use the trained models as initialization and optimize them with the same optimizer with decreased learning rate of 0.0001 for all datasets. We train a fixed number of iterations (10k iterations for the CIFAR datasets, 50 epochs for the SVHN dataset and 40 epochs for the ImageNet dataset) and report the test accuracy evaluated at termination. The training time of the supervised pre-training stage is roughly the same as training the original models without gating. Our overall training time is slightly longer with an increase of about 30-40%.

SkipNet Performance Evaluation

In this subsection, we evaluate the computation reduction of SkipNets with both feed-forward and recurrent gates on various datasets. We study the trade-off between computation cost and accuracy governed by the hyper-parameter α in Eq. 2.5. Larger values of α favor decreased prediction cost potentially leading to decreased accuracy.

Computation reduction while preserving full network accuracy

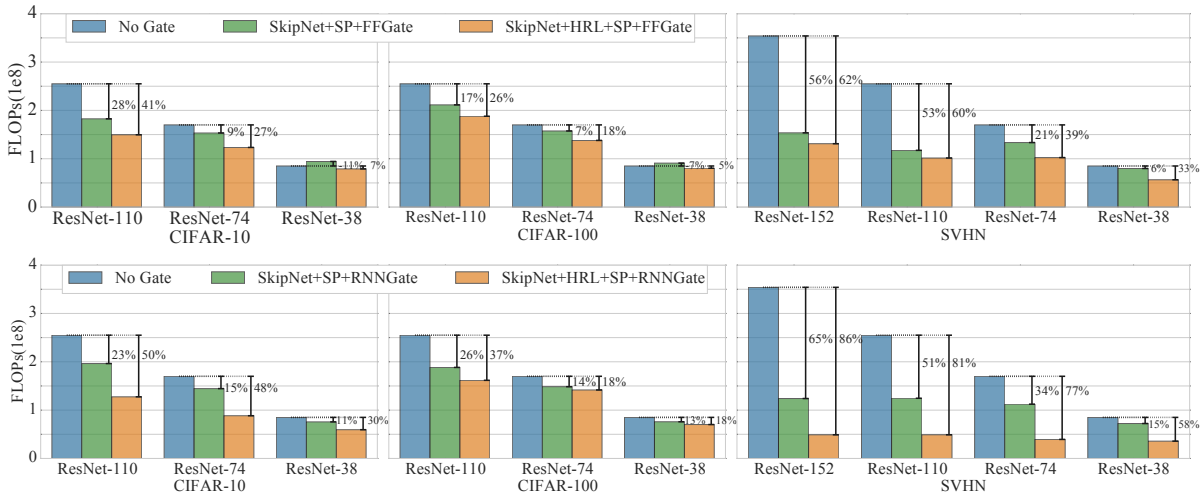


Figure 2.4: Computation reduction of SkipNets on CIFAR-10, 100 and SVHN.

We first demonstrate that SkipNets can substantially reduce computation without affecting accuracy. Fig. 2.4 and Fig. 2.5 show the computation cost (including the computation of the gate networks), measured in *floating point operations* (FLOPs), of the original ResNets and SkipNets

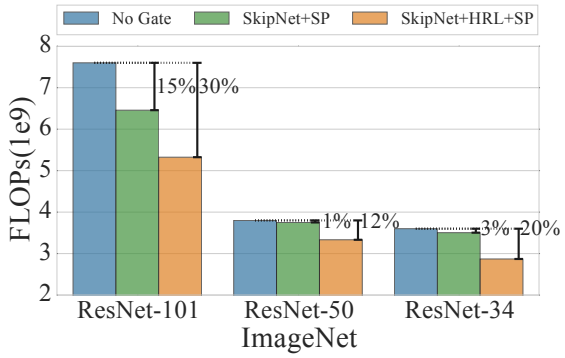


Figure 2.5: Computation reduction of SkipNets with RNNGate on ImageNet.

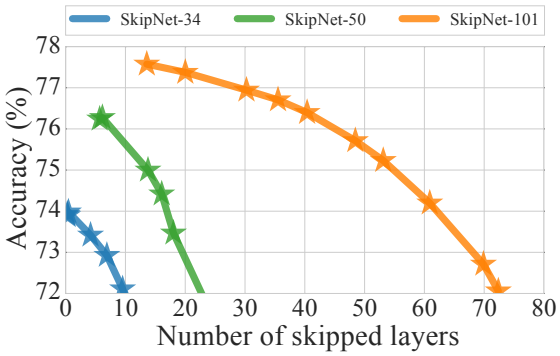


Figure 2.6: Trade-off between computation and accuracy on ImageNet.

with both feed-forward and recurrent gate designs with α tuned to match the same accuracy (variance less than 0.5%). The trade-off between accuracy and computational cost will be discussed later. Following [73], we only consider the multiply-adds associated with convolution operations as others have negligible impact on cost.

We observe that the hybrid reinforcement learning (HRL) with supervised pre-training (SkipNet+HRL+SP) is able to substantially reduce the cost of computation. Overall, for the deepest model on each dataset, SkipNet-110+HRL+SP with recurrent gates reduces computation on the CIFAR-10 and CIFAR-100 datasets by 50% and 37% respectively. The largest SkipNet-152+HRL+SP model with recurrent gates reduces computation on the SVHN dataset by 86%. On the ImageNet data, the SkipNet-101+HRL+SP using recurrent gates is able to reduce computation by 30%. Interestingly, as noted earlier, even in the absence of the cost regularization in the objective, the supervised pre-training of the SkipNet architecture consistently results in reduced prediction costs. One way to explain it is that the shallower network is easier to train and thus more favorable. We also observe that deeper networks tend to experience greater cost reductions which supports our conjecture that only a small fraction of inputs require relatively deep networks.

Trade-off computational cost and accuracy

Eq. 2.5 introduces the hyper-parameter α to balance the computational cost and classification error. In Fig. 2.6 and Fig. 2.7 we plot the accuracy against the average number of skipped layers for different values of α from 0.0 to 4.0. One observation is that both feed-forward and recurrent gates skip a similar fraction of layers. However, because of the large computational overhead of feed-forward gates, recurrent gates enjoy a greater overall reduction in FLOPs. We also observe that for small $\alpha \leq 1.0$ the trade-off curve is relatively flat indicating that a large decrease in computation is accompanied by a negligible decrease in accuracy. However, for larger $\alpha > 1.0$, both computation and accuracy decrease rapidly. By adjusting α, one can trade-off computation and accuracy to meet various computation or accuracy requirements (e.g. deployment on mobile devices).

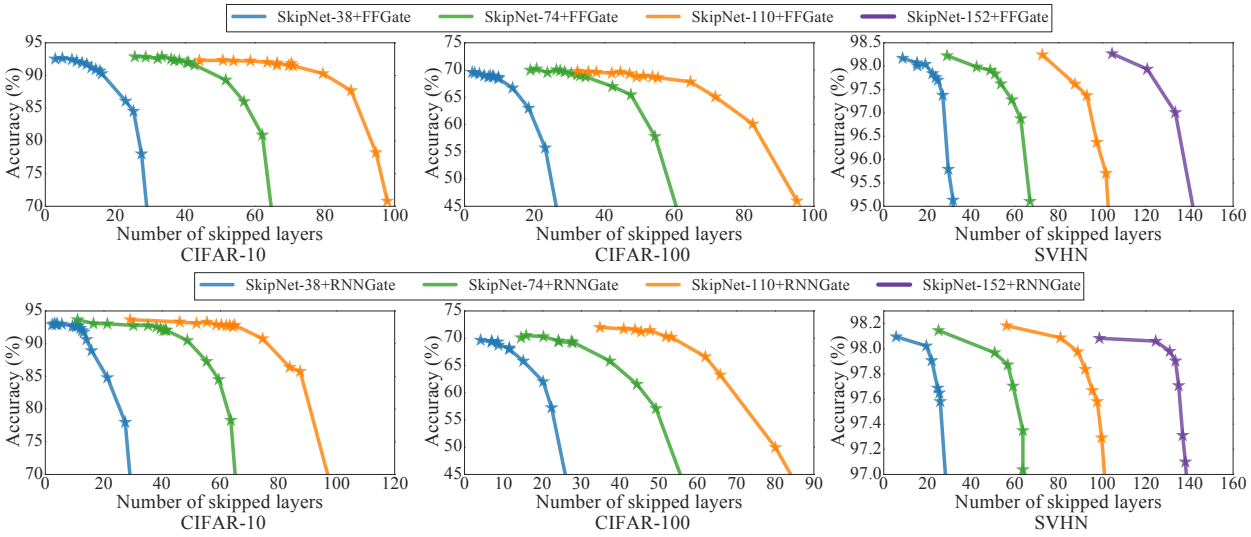
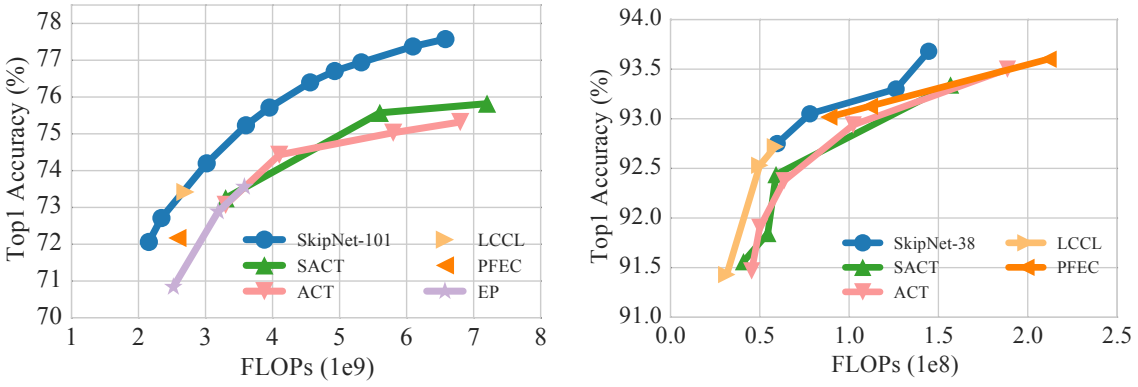


Figure 2.7: Trade-off between computation and accuracy on CIFAR-10, 100 and SVHN.

Comparison with State-of-the-art Models



(a) ImageNet (b) CIFAR-10

Figure 2.8: Comparison of SkipNet with the state-of-the-art models on ImageNet and CIFAR-10.

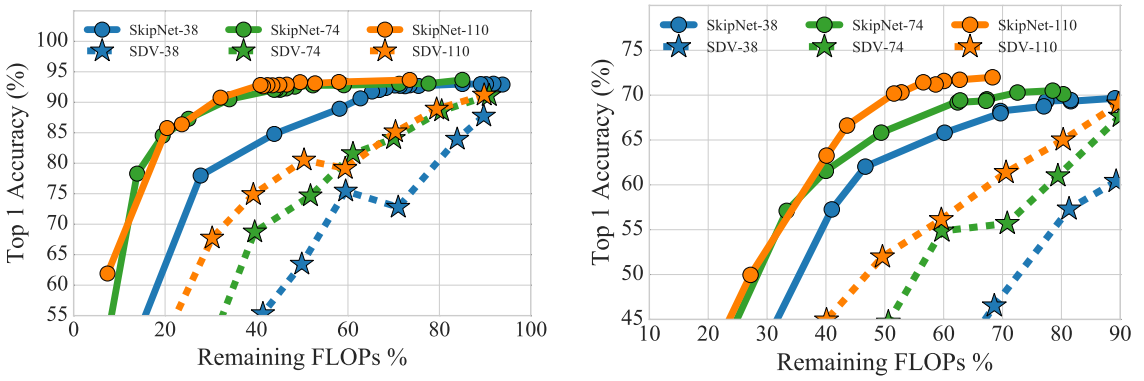
We compare SkipNet with existing state-of-the-art models including both dynamic networks and static compression models on both ImageNet (Fig. 2.8a) and CIFAR-10 (Fig. 2.8b). The SACT and ACT models proposed by [52] are adaptive computation time models that attempt to terminate computation early in each group of blocks of ResNets (Sec. 2.4). In addition, we compare SkipNet with static compression techniques: PEFC [113], LCCL [42] and EP [125] which are also complementary approaches to our method.

As shown in Fig. 2.8a, SkipNet-101 outperforms SACT and ACT models by a large margin on the ImageNet benchmark despite they are using the recent more accurate pre-activation [74] ResNet-101 as the base model. We hypothesize that increased flexibility afforded by the skipping

model formulation enables the SkipNet design to outperform SACT and ACT. Similar patterns can be observed on CIFAR-10 in Fig. 2.8b.[2]

For the comparison with the static compression techniques, we plot the computation FLOPs and the accuracy of the compressed residual networks (may have different depths from what we used in this paper) in Fig. 2.8. Though the static compression techniques are complementary approaches, SkipNet performs better or similar than these techniques. To note that, though LCCL [42] uses shallower and cheaper ResNets (34 layers on ImageNet and 20, 32, 44 layers on CIFAR-10) than our approach, our approach could still obtain comparable results.

Comparison with stochastic depth model variant



(a) CIFAR-10 (b) CIFAR-100

Figure 2.9: Comparison with a variant of the stochastic depth model (SDV).

Huang *et al.* [84] proposes deep networks with stochastic depth which randomly drop a subset of layers and bypass them with the identity function for a given mini-batch during training while using the network with full depth at inference. The original goal of the stochastic depth model is to avoid gradient vanishing and speed up training. A natural variant of this model in order to reduce inference computation cost is to skip blocks randomly with a chosen ratio in both training and inference phases referred as *SDV*. We compare SkipNet to SDV on both CIFAR-10 and CIFAR-100 datasets shown in Fig. 2.9. SkipNet outperforms SDV by a large margin under networks with different depths.

Skipping Behavior Analysis and Visualization

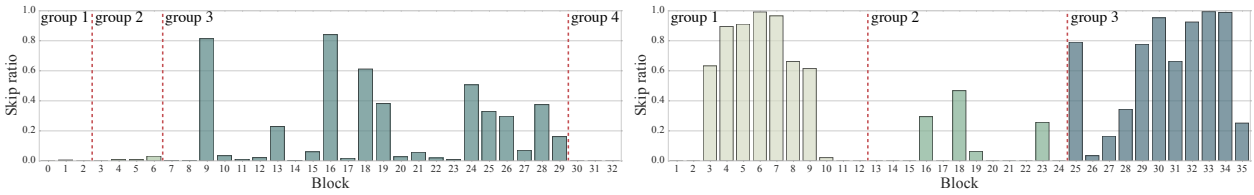
In this subsection, we study the skipping pattern of SkipNet and then investigate the key factors associated with the dynamic skipping behaviors. We study the correlation between block skipping and the input images in the following aspects: (1) the scale of the inputs (2) qualitative difference between images (3) prediction accuracy per category. We find that SkipNet skips more aggressively

[2]We obtain the CIFAR-10 results by running the code provided by the authors

on inputs with smaller scales and also brighter and clearer. Moreover, more blocks are skipped for classes with high accuracy.

Skip ratio of different blocks

We first visualize the skip ratio of different blocks in SkipNet in Fig. 2.10 on the CIFAR-10 and ImageNet datasets. The ResNet model can be divided into 4 groups for ImageNet and 3 groups for CIFAR-10 where blocks in the same group have the same feature map size and tend to have similar functionality. We observe that SkipNet-101 on ImageNet skips less in the first, second and the last groups partly due to those few blocks in the first two groups are capturing low level features which are critical for all inputs and layers in the last group are important for the final predictions. For SkipNet-74 on CIFAR-10, the model tends to skip less in the first several blocks of each group which could be considered as the representative blocks. Also, group 2 has less skipping compared to group 1 and group 3 which indicates that blocks in group 2 may be more critical in the feature extraction.



(a) SkipNet-101 on ImageNet

(b) SkipNet-74 on CIFAR-10

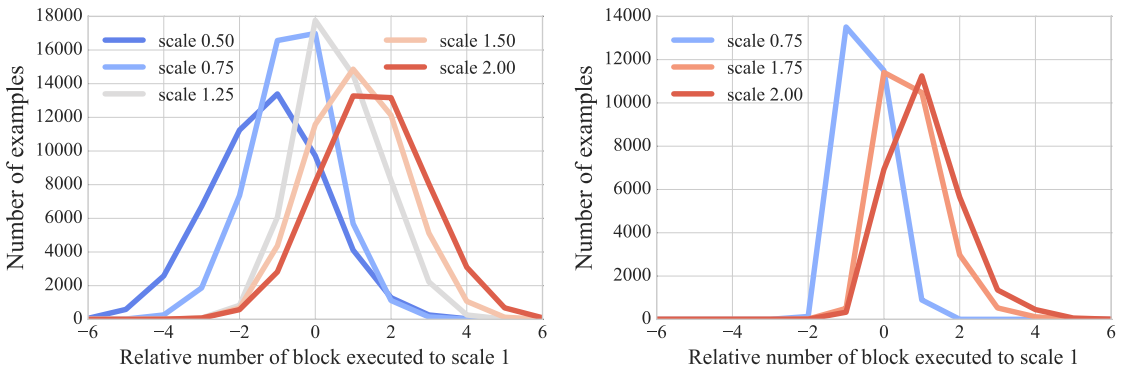
Figure 2.10: Skip ratio of different blocks of SkipNet.

Input scales

We conjecture that the scale of the inputs is a factor that affects the skipping decisions of the gates. To verify this hypothesis, we conduct multi-scale testing of trained models on the ImageNet and SVHN datasets. We plot the distribution of the number of blocks executed of different input scales relative to the original scale 1 used in other experiments. We observe on both datasets that the distributions of smaller scales are skewed left (executing less blocks than the model with input scale 1) while the distributions of larger scales are skewed right (more block executed). This observation indicates that inputs with larger scale requires larger receptive field and thus need to execute more blocks. Another way to interpret this observation is that the gating modules in SkipNet may learn to find the appropriate receptive field size for the given input.

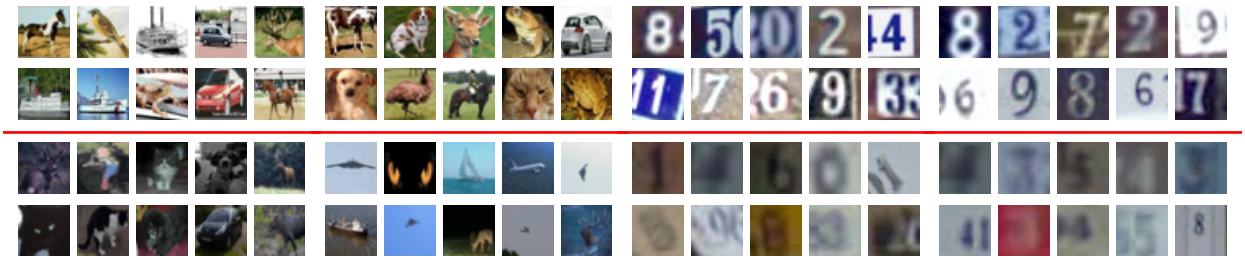
Visual difference between inputs

To better understand the learned skipping patterns, we cluster the images that SkipNets skip many layers (treated as *easy* examples) and keep many layers (treated as *hard* examples) in Fig. 2.12 for both the CIFAR-10 and SVHN dataset. Interestingly, we find that images within each cluster share



(a) ResNet-101 on ImageNet (b) ResNet-152 on SVHN

Figure 2.11: Distribution of number of blocks executed with multi-scale inputs.



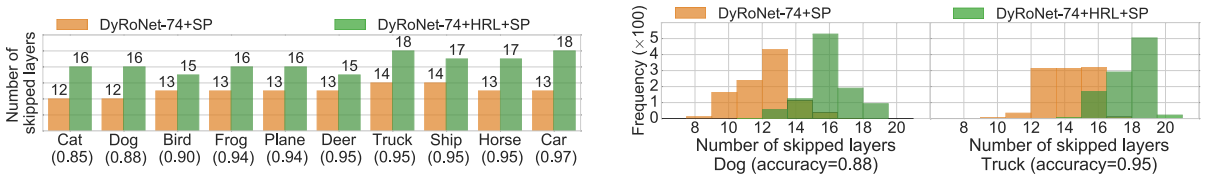
(a) SkipNet+FFGate CIFAR-10 (b) SkipNet+RNNGate CIFAR-10 (c) SkipNet+FFGate SVHN (d) SkipNet+RNNGate SVHN

Figure 2.12: Visualization of *easy* and *hard* images in the CIFAR-10 and SVHN with SkipNet-74.

similar characteristics with respect to saliency and clarity. In Fig. 2.12a and Fig. 2.12b, we see that the easier examples (those which bypass many layers of the network) are more salient while the hard examples (those that require most layers in the network) tend to be dark and blurry. We further visualize SkipNets on the SVHN dataset in Fig. 2.12c and Fig. 2.12d. We find that the hard examples are blurry and difficult to distinguish even for humans while the easy examples are clear and straightforward. These findings reveal that the visual characteristics of the input images also affect the block skipping.

Prediction accuracy per category

We further study the correlation of skipping behaviors and the level of difficulty of different classes on the CIFAR-10 dataset. The conjecture is that the SkipNet skips more on easy classes (class with high accuracy, e.g., truck class) while skipping less on hard classes (class with low accuracy, e.g., cat and dog classes). We plot the median of number of skipped layers of examples in each of the ten classes for SkipNet+SP and SkipNet+HRL+SP in Fig. 2.13a. It shows that while all classes tend to skip more aggressively after applying HRL, SkipNets tend to skip more layers on easy classes. Fig. 2.13 indicates that the distribution of hard classes (e.g. dog class) are skewed left, whereas



(a) Median of number of skipped layers of each class

(b) Distribution of number of skipped layers

Figure 2.13: Correlation of number of skipped layers and the level of easiness of different classes.

easier classes (e.g. truck class) are skewed right as SkipNet tends to skip more layers on easier classes.

SkipNet Design and Algorithm Analysis

In this subsection, we analyze the design choices of SkipNet: hybrid objective function, supervised pre-training and "hard" gating design.

Effectiveness of hybrid learning algorithm

To investigate the effectiveness of supervised pre-training and the hybrid reward function for RL, we compare the performances of SkipNet-38 trained using basic RL, hybrid RL from scratch (HRL+S), and hybrid RL plus supervised pre-training (HRL+SP). We plot the results on CIFAR-10 in Fig. 2.14. For SkipNet+HRL+S and SkipNet+RL, we train both networks for 80k iterations to match the total training steps of the two-stage training of SkipNet+HRL+SP.

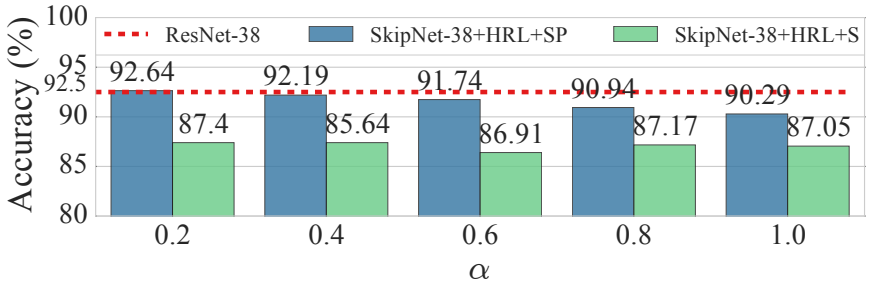


Figure 2.14: Ablation study of different α.

First, we were unable to train the model using the pure RL approach (SkipNet-38+RL accuracy was roughly 10%). This provides strong evidence for the importance of supervision in complex vision tasks. Second, SkipNet-38+HRL+SP consistently achieves higher accuracy than SkipNet-38+HRL+S. The accuracy of SkipNet-38+HRL+S is lower than the accuracy of the original ResNet-38 model even with very small α. This suggests that supervised pre-training can provide a better initialization for SkipNets which improves the prediction accuracy.

Table 2.3: "Hard" gating *vs* "soft" gating.

Data	Model	Accuracy	FLOPs (1e8)
CIFAR-10	SkipNet-38-Hd	90.83	0.58
	SkipNet-38-St	66.67	0.61
	SkipNet-74-Hd	92.38	0.92
	SkipNet-74-St	52.29	1.03
	SkipNet-110-Hd	88.11	0.18
	SkipNet-110-St	23.44	0.05

Data	Model	Accuracy	FLOPs (1e8)
CIFAR-100	SkipNet-38-Hd	67.68	0.50
	SkipNet-38-St	21.70	0.62
	SkipNet-74-Hd	67.79	0.61
	SkipNet-74-St	25.47	0.89
	SkipNet-110-Hd	63.66	0.96
	SkipNet-110-St	9.84	1.00

"Hard" gating and "Soft" gating design

During supervised pre-training, we can either treat gate outputs as "hard" (Sec. 2.2) or "soft" (Sec. 2.2). In the case of "soft" gating, to achieve the desired reduction in computation hard gating must be applied during inference. In Tab. 2.3, we compare the classification accuracy of SkipNet with "hard" (SkipNet-Hd) and "soft" gating (SkipNet-St) under similar computation cost[3]. We find that SkipNet-Hd achieves much higher accuracy than SkipNet-St. We conjecture the inconsistency between training and inference when using soft gating results in the gap in accuracy.

2.4 Related Work

Accelerating existing convolutional networks has been a central problem in real-world deployments and several complementary approaches have been proposed. Much of this work focuses on model compression [60, 68] and distillation [79]. [60] proposes using vector quantization and [68] introduces trained quantization and Huffman coding to reduce the network sizes by orders of magnitude and reduce the prediction cost several times. [79] introduces distillation to transfer knowledge from a very deep network to a shallower one. These methods are applied after training the initial networks and they are usually used as post-processing. Also, these optimized networks do not dynamically adjust the model complexity in response to the input. Though these approaches are considered as complementary approaches to ours, we show our dynamic network design actually outpeforms the existing static compression techniques in the experiment section.

There are also related works [52, 63, 173] that adaptively adjust computation by examining early termination. [63] proposes to predict halting in recurrent networks to save computational cost. [52] and [173] propose the use of early termination in convolutional networks. [52], the closest work to ours, studies early termination in each group of blocks of ResNets. In contrast, SkipNet does not exit early but instead conditionally bypasses individual layers based on the output of the proceeding layers which we show in experiments enables greater computation savings.

Another line of work [12, 65, 132, 181, 186] explores cascaded model composition. Close to our proposed algorithm, [65] and [132] study reinforcement learning algorithms to address the decision problem of the cascaded model composition. We extend reinforcement learning with supervised loss to achieve faster convergence and improved accuracy and show the pure reinforcement learning

[3]We tune SkipNet-Hd to match the computation of SkipNet-St.

algorithm is not stable and hard to tune to match the accuracy of the original models. Moreover, like the work on early termination, this work follows a sequential progression through a sequence of models. In contrast, SkipNets selectively include and exclude individual model layers on a per-input basis.

The gating modules in SkipNets act as regulating gates for groups of layers and are related to the gating designs in recurrent neural networks (RNN) [30, 80, 163, 170]. [80] proposes to add gates to an RNN so that the network can keep important memory in network states, while [170] introduces similar techniques to convolutional networks in order to learn very deep image representation. [30] and [163] further apply gates to other image recognition problems. These proposed gates are "soft" in the sense that the gate outputs are continuous values, while our gates are "hard", either executing or bypassing any given layer. We show in our experiment section that "hard" gating is more preferable than "soft" gating in this problem.

2.5 Chapter Summary

We observe that the recent steady progress in computer vision models has been accompanied by rapid increases in computational cost. This suggests that only a small fraction of images require the full depth of large convolutional networks and thus the vast majority of images could be effectively processed by shallower architectures.

To dynamically scale the depth of network on a per image basis we introduced SkipNet architecture. The SkipNet architecture learns to dynamically skipping unnecessary layers on a per-input basis, without sacrificing prediction accuracy. We framed the dynamic execution problem as a sequential decision problem. To address the inherent non-differentiability of dynamic execution, we proposed a novel hybrid learning algorithm which combines the merits of both supervised learning and reinforcement learning.

We evaluated the proposed approach on four benchmark datasets, showing that SkipNets reduce computation substantially while preserving the same accuracy as the original models on all datasets. Compared to both state-of-the-art dynamic models (SACT and ACT models) and static compression techniques, SkipNets obtain better accuracy with lower computation on both the ImageNet and CIFAR-10 benchmarks. We hypothesize that framing the skipping problem as a sequential decision problem instead of a halting problem used in SACT and ACT models empowers flexibility of the decision making and thus leads to better skipping policies. Moreover, we visualize the skipping patterns to show that SkipNets can identify *hard* and *easy* examples and route them effectively.

We believe this work could be generalized beyond sequential graphs to allow cycles of layers. This more complex skipping behavior could allow greater parameter sharing and dynamic architecture design on a per image basis.

Chapter 3

Channel-Level Dynamic Routing

3.1 Overview

Increasing network depth has been a dominant trend [73] in the design of deep neural networks for computer vision. However, increased network depth comes at the expense of computational overhead and increased training time. To reduce the computational cost of machine translation models, Shazeer et al. [160] recently explored the design of "outrageously" wide sparsely-gated mixture of experts models, which employs a combination of simple networks, called experts, to determine the output of the overall network. They demonstrated that these relatively shallow models can reduce computational costs and improve prediction accuracy. However, their resulting models needed to be many times larger than existing translation models to recover state-of-the-art translation accuracy. Expanding on this, preliminary work by Eigen et al. [45] demonstrated the advantages of stacking *two* layers of mixture of experts models for MNIST digits classification. With these results, a natural question arises: *can we stack and train many layers of mixture of experts models to improve accuracy and reduce prediction cost without radically increasing the network width?*

In this paper, we explore the design of *deep mixture of experts models (DeepMoEs)* that compose hundreds of mixture of experts layers. DeepMoEs combines the improved accuracy of deep models with the computational efficiency of sparsely-gated mixture of expert models. However, constructing and training DeepMoEs has several key challenges. First, mixture decisions interact across layers in the network requiring joint reasoning and optimization. Second, the discrete expert selection process is non-differentiable, complicating gradient-based training. Finally, the composition of multiple mixture of experts models increases the chance of degenerate (i.e., singular) combinations of experts at each layer.

To address these challenges we propose a general DeepMoE architecture that combines a deep convolutional network with a shallow embedding network and a multi-headed sparse gating network (see Fig. 3.1). The shallow embedding network terminates in a soft-max output layer that computes *latent mixture weights* over a fixed set of latent experts. These latent mixture weights are then fed into the multi-headed sparse gating networks (with ReLU outputs) to *select* and *re-weight* the channels in each layer of the base convolutional network. We then jointly train the base model,

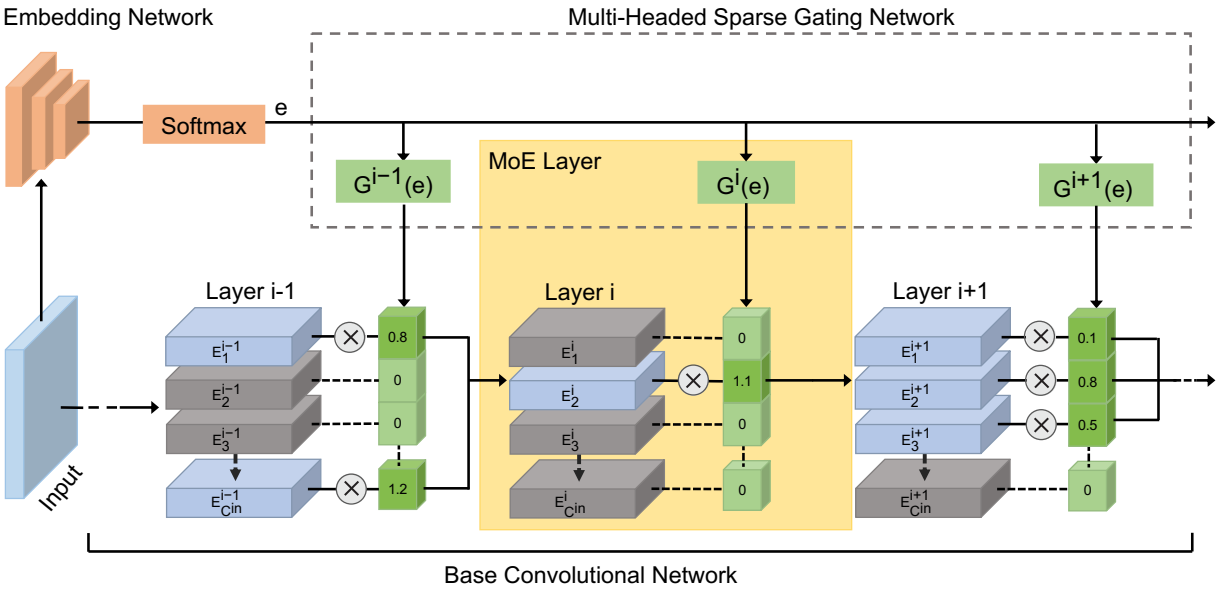


Figure 3.1: DeepMoE architecture.

shallow embedding network, and multi-headed gating network with an auxiliary classification loss function over the shallow embedding network and sparse regularization on the gating network outputs to encourage diversity in the latent mixture weights and sparsity in the layer selection. This helps balance expert utilization and keep computation costs low.

Recent work [23] proves that the expressive power of a deep neural network increases super-exponentially with its depth, based on the width. By stacking multiple mixture of expert layers and dynamically generating the sparse channel weights, we analyze in Sec. 3.3 that DeepMoEs reserve the expressive power of the unsparsified deep networks.

Based on this theoretical analysis, we further propose two variants *wide-DeepMoE* and *narrow-DeepMoE* to improve prediction accuracy while reducing the computational cost compared to standard convolutional networks. For wide-DeepMoEs, we first double the number of channels in the standard convolutional networks and then replace the widened convolutional layers with MoE layers. We examine in experiments that if only half of channels in the widened layers are selected at inference, wide-DeepMoE is able to achieve higher prediction accuracy due to the increase of model capacity while maintaining the same computational cost as the unwidened network. For narrow-DeepMoEs, we directly replace the convolutional layers with MoE layers in the standard convolution networks which generalizes the existing work [115] on dynamic channel pruning and produces models that are more accurate and more efficient than the existing channel pruning literature.

We empirically evaluate the DeepMoE architecture on both image classification and semantic segmentation tasks using four benchmark datasets (i.e.,CIFAR-10, CIFAR-100, ImageNet2012, CityScapes) and conduct extensive ablation study on the gating behavior and the network design in Sec. 3.4. We find that DeepMoEs achieves the goal of improving the prediction accuracy with

reduced computational cost on various benchmarks.

Our contributions can be summarized as: (1) We first propose a novel DeepMoE design which allows the network to dynamically select and execute part of the network at inference. (2) We theoretically analyze that the proposed DeepMoE design preserves the expressive power of a standard convolutional network with reduced computational cost. (3) We further introduce two DeepMoE variants that are more accurate and efficient than the prior methods on different benchmarks.

3.2 Deep Mixture of Expert: Channel-Level Dynamics

In this section, we first describe the DeepMoE formulation and then introduce the detailed architecture design and loss function formulation.

Mixture of Experts

The original mixture of experts [89] formulation combines a set of experts (classifiers), $E_1, ..., E_C$, using a mixture (gating) function G that returns a distribution over the experts given the input $\mathbf{x}$:

$$y = \sum_{i=1}^{C} G(\mathbf{x})_i E_i(\mathbf{x}). \tag{3.1}$$

Here $G(\mathbf{x})_i$ is the weight assigned to the i^{th} expert E_i. Later work [25] generalized this mixture of experts formulation to a non-probabilistic setting where the gating function G outputs arbitrary weights for the experts instead of probabilities. We adopt this non-probabilistic view since it provides increased flexibility in re-scaling and composing the expert outputs.

DeepMoE Formulation

In this work, we propose the DeepMoE architecture which extends the standard single-layer MoE model to multiple layers within a single convolutional network. While traditional MoE frameworks focus on the model level combinations of experts, DeepMoE operates within a single model and treats each channel as an expert. The experts in each MoE layer consist of the output channels of the previous convolution operation. In this section, we derive the equivalence between gated channels in a convolution layer and the classic mixture of experts formulation.

A convolution layer with tensor input $\mathbf{x}$ having spatial resolution $W \times H$ and C^{in} input channels, and $C^{\text{in}} \times k \times k \times C^{\text{out}}$ convolutional kernel $\mathbf{K}$ of dimension $k \times k$ can be written as:

$$\mathbf{z}_{o,s,t} = \sum_{i=1}^{C^{\text{in}}} \sum_{u=0}^{k-1} \sum_{v=0}^{k-1} \mathbf{K}_{i,u,v,o} \mathbf{x}_{i,s+u,t+v}, \tag{3.2}$$

where $\mathbf{z}$ is the $C^{\text{out}} \times W \times H$ output tensor. To construct an MoE convolutional layer we scale the input channels by the gate values $\mathbf{g} \in \mathbb{R}^{C^{\text{in}}}$ for that layer and rearrange terms:

$$\mathbf{z}_{o,s,t} = \sum_{i=1}^{C^{\text{in}}} \sum_{u=0}^{k-1} \sum_{v=0}^{k-1} \mathbf{g}_i \mathbf{K}_{i,u,v,o} \mathbf{x}_{i,s+u,t+v} \tag{3.3}$$

$$= \sum_{i=1}^{C^{\text{in}}} \mathbf{g}_i \left(\sum_{u=0}^{k-1} \sum_{v=0}^{k-1} \mathbf{K}_{i,u,v,o} \mathbf{x}_{i,s+u,t+v} \right), \tag{3.4}$$

Defining convolution operator $*$, we can eliminate the summations and subscripts in (3.4) to obtain:

$$\mathbf{z} = \sum_{i=1}^{C^{\text{in}}} \mathbf{g}_i \mathbf{K}_i * \mathbf{x}_i = \sum_{i=1}^{C^{\text{in}}} \mathbf{g}_i E_i \left(\mathbf{x} \right). \tag{3.5}$$

Thus, we have shown that gating the input channels to a convolutional network is equivalent to constructing a mixture of experts for each output channel. In the following sections, we describe how the gate values $\mathbf{g}$ are obtained for each layer and then present how individual mixture of experts layers can be efficiently composed and trained in the DeepMoE architecture.

DeepMoE Architecture

DeepMoE is composed of three components: a base convolutional network, a shallow embedding network, and a multi-headed sparse gating network.

The **base convolutional network** is a deep network where each convolution layer is replaced with an MoE convolution layer as described in the previous section. In our experiments we use ResNet [73] and VGG [165] as the base convolutional networks.

The **shallow embedding network** maps the raw input image into a latent mixture weights to be fed into the multi-headed sparse gating network. To reduce the computational overhead of the embedding network, we use a 4-layer (for CIFAR) or 5-layer (for ImageNet) convolutional network with 3-by-3 filters with stride 2 (roughly 2% of the computation of the base models).

The **multi-headed sparse gating network** transforms the latent mixture weights produced by the shallow embedding network into sparse mixture weights for each layer in the convolutional network. The gate for layer l is defined as:

$$G^l(\mathbf{e}) = \texttt{ReLU}(W_g^l \cdot \mathbf{e}), \tag{3.6}$$

where $\mathbf{e}$ is the output of the shared embedding network $\mathbf{M}$ and W_g^l are the learned parameters which, using the ReLU operation, project the latent mixture weights into *sparse* layer specific gates.

We refer to this gating design as *on demand gating*. The number of experts chosen at each level is data-dependent and the expert selection across different layers can be optimized jointly. Unlike the "noisy Top-K" design in [160], it is not necessary to determine the number of experts at each layer and indeed each layer can learn to use a different number of experts.

DeepMoE Training

As with standard convolutional neural networks, DeepMoE models can be trained end-to-end using gradient based methods. The overall goals of the DeepMoE are threefold: (1) achieve high prediction accuracy, (2) lower computation costs, and (3) keep the network highly expressive. Thus, DeepMoE must learn a gating policy that selects a diverse, low-cost mixture of experts for each input. To this end, given the input $\mathbf{x}$ and the target y, we define the learning objective as

$$\mathcal{J}(\mathbf{x}; y) = \mathcal{L}_b(\mathbf{x}; y) + \lambda \mathcal{L}_g(\mathbf{x}) + \mu \mathcal{L}_e(\mathbf{x}; y), \tag{3.7}$$

$\mathcal{L}_b$ is the cross entropy loss for the base convolutional model, which encourages a high prediction accuracy.

The $\mathcal{L}_g$ term defined:

$$\mathcal{L}_g(\mathbf{x}) = \sum_{l=1}^{L} ||G^l(M(\mathbf{x}))||_1, \tag{3.8}$$

is used to control the computational cost (via the λ parameter) by encouraging sparsity in the gating network.

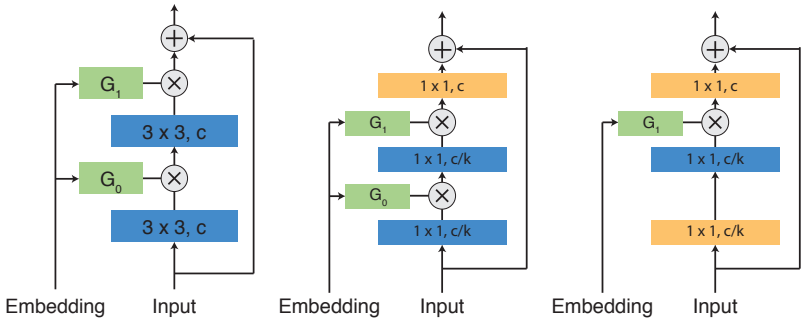
Finally, we introduce an additional embedding classification loss $\mathcal{L}_e$, which is the cross-entropy classification loss. This encourages the embedding or some transformation of the embedding to be predictive of the class label, preventing the phenomenon of gating networks converging to an imbalanced utilization of experts [160]. The intutition behind this loss construction is that examples from the same class should have similar embeddings and thus similar subsequent gate decisions, while examples from different classes should have divergent embeddings, which would in turn discourage the network from over-using a certain subset of channels.

Because the DeepMoE loss is differentiable we train all three sub-networks jointly using stochastic gradient descent. Once trained, we then set λ and μ to 0 and continue to train a few more epochs to refine the base convolutional network.

3.3 Theoretical Analysis on Expressive Power

The expressive power of deep neural networks is associated with both the width and the depth of the network. Intuitively, the wider the network is, the more expressive power the network has. Cohen *et al.* [23] proves that the expressive power of a deep neural network increases super-exponentially with respect to the network depth, based on the network width. In this section, we demonstrate that due to the dynamic execution nature and the multi-layer stacking design, DeepMoE preserves the expressive power of a standard unsparsified neural network with reduced runtime computational cost.

We define the expressive power of a convolutional neural network as the ability to construct labeling to differentiate input values. Following Cohen *et al.* [23], we view a neural network as a mapping from a particular example to a cost function (e.g., negative log probability) over labels. The mapping can be represented by a tensor $\mathcal{A}^y$ operated on the combination of the representation functions.



(a) Basic Block (b) Bottleneck-A (c) Bottleneck-B

Figure 3.2: Gated residual block designs.

More concretely, the rank of $\mathcal{A}^y$, which scales as n^{2^L} with measure 1 over the space of all possible network parameters (n is the number of channels of a convolutional layer, a.k.a, network width; and L is the network depth), is a measure of the expressive power of a neural network as established in [23]. In static channel pruning, if m channels are kept, then the expressive power of the pruned network becomes m^{2^L} which is a strict subspace of n^{2^L} as $m < n$.

What makes our DeepMoE prevail is that (the sparsity pattern of) our mapping $\mathcal{A}^y$ depends on the data. We provide theoretical analysis that DeepMoE has an expressive power of n^{2^L} with probability $1 - \binom{m}{n}^{-L}$ when stacking multiple MoE layers, indicating DeepMoE preserves the expressive power of the unsparsified network.

Motivated by the theoretical analysis, we propose two variants of DeepMoE: *wide-DeepMoE* and *narrow-DeepMoE*. In the former one, we first increase the number of channels in the convolutional networks to increase the expressive power of the network and then replace the widened layers with MoE layers. By controlling the number of channels selected at runtime, we can improve the prediction accuracy with the same amount of the computation as the unwidened network. This design has the potential to be applied to the real-world deployment on the new hardware architecture supporting dynamic routing, e.g., TPU, where we can place a wide network on it and only execute part of the network at runtime instead of placing a static thin network with the same amount of computation. Narrow-DeepMoE is closer to the dynamic channel pruning setting in [113] and comparable to the traditional static channel pruning.

3.4 Experiments

In this section, we first evaluate the performance of both wide-DeepMoE and narrow-DeepMoE on the image classification (Sec. 3.4 and 3.4) and semantic segmentation tasks (Sec. 3.4). We observe that DeepMoEs can achieve lower prediction error rate with reduced computational cost. We also analyze the behavior of our gating network, DeepMoEs regularization effect, and other strategies for widening the network in Sec. 3.4.

Datasets. For the image classification task, we use the CIFAR-10 [101], CIFAR-100 [101] and

Table 3.1: Wide-DeepMoE evaluation on ImageNet.

Model	Top-1 Error Rate (%)
ResNet-18	30.24
Hard MoE [64]	30.43
Wide-DeepMoE-18	**29.05**
ResNet-34	26.70
Wide-DeepMoE-34	**25.87**
ResNet-50	23.85
Wide-DeepMoE-50	**22.88**

ImageNet 2012 datasets [157]. For the semantic segmentation task, we use the CityScapes [26] dataset, which provides pixel-level annotations in the images with a resolution of 2048×1024. We apply standard data augmentation using basic mirroring and shifting [183] for CIFAR datasets and scale and aspect ratio augmentation with color perturbation [201] for ImageNet. We follow [206] to enable random cropping and basic mirroring and shifting augmentation for the CityScapes dataset.

Models. We examine DeepMoE with VGG [165] and ResNet [73] network designs as the base convolutional network (a.k.a, backbone network). VGG is a typical feed-forward network without skip connections and feature aggregation while ResNet, which is composed of many residual blocks, has more complicated connections. To construct DeepMoE, we add a gating header after each convolutional layer in VGG and modify the residual blocks in ResNet (Fig. 3.2).

In wide-DeepMoE, we increase the number of channels in each convolutional layer by a factor of two unless stated otherwise. In narrow-DeepMoE, we retain the same channel configuration as the original base convolutional model.

Training. To train DeepMoE we follow common training practices [73, 208]. For the CIFAR datasets, we start training with learning rate 0.1 for ResNet and 0.01 for VGG16, which is reduced by $10\times$ at 150 and 250 epochs with total 350 epochs for the baselines and 270 epochs for DeepMoE joint optimization stage and another 80 epochs for fine-tuning with fixed gating networks.

For ImageNet, we train the network with initial learning rate 0.1 for 100 epochs and reduce it by $10\times$ every 30 epochs. We do not further fine-tune the base convolutional network on ImageNet as we find the improvement from fine-tuning is marginal compared to that on CIFAR datasets.

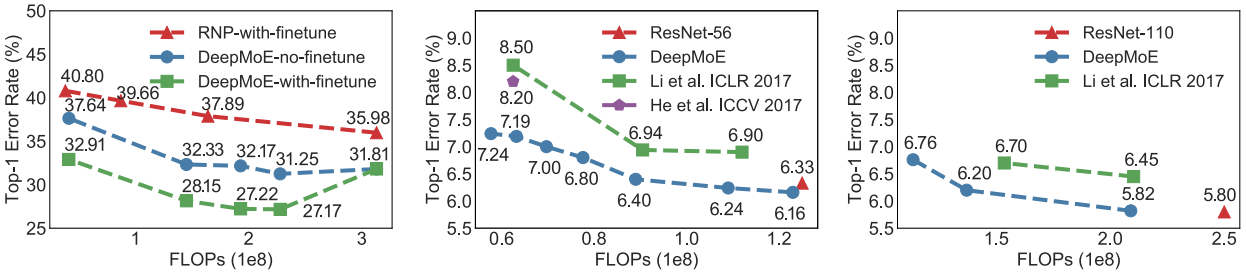
We set the computational cost parameter λ in the DeepMoE loss function (Eq. 4.12) between [0.001, 8] (larger values reduce computation) and $\mu = 1$ for the CIFAR datasets to match the scale of the cross entropy loss on the base model. For ImageNet we set $\mu = 0$ to improve base model feature extraction. The training schedule for semantic segmentation is detailed in Sec. 3.4.

Wide-DeepMoE

In this section, we evaluate the performance of wide-DeepMoE as well as its memory usage.

Table 3.2: Wide-DeepMoE with ResNet-56 and ResNet-100 on CIFAR.

Dataset	Model	Top-1 Error Rate (%)
CIFAR-10	ResNet-56	6.55
	Wide-DeepMoE-56	**6.03**
CIFAR-100	ResNet-56	31.46
	Wide-DeepMoE-56	**29.77**
	ResNet-110	29.45
	Wide-DeepMoE-110	**26.14**



(a) DeepMoE *vs* RNP (b) DeepMoE *vs* Static Pruning (c) DeepMoE *vs* Static Pruning

Figure 3.3: Benmark comparison on CIFAR-10 and 100.

Improved Accuracy with Reduced Computation

To conduct the evaluation, we first increase the number of channels in the residual networks by a factor of 2 and then control the sparsification so that on average half of the convolutional channels are selected at the inference time. Through our evaluations we find that wide-DeepMoE has lower prediction error rate than the standard ResNets on ImageNet (Tab. 3.1), CIFAR-10 and CIFAR-100 (Tab. 3.2).

We evaluate ResNet-56 and ResNet-110 on the CIFAR-10 and CIFAR-100 datasets and ResNet-18, ResNet-34, ResNet-50 on ImageNet, using the basic block (Fig. 3.2a) for 18 and 34 and bottleneck-A(Fig. 3.2b) for 50. The more memory efficient bottleneck-B (Fig. 3.2c) is also adopted on ImageNet.

As expressed in Tab. 3.1, wide-DeepMoE is able to reduce the error rate of the ImageNet benchmark without increasing the computational cost (measured by FLOPs) with networks of different depths. In particular, wide-DeepMoE with ResNet-18 and 34 reduce $\sim 1\%$ Top-1 error on ImageNet on which the previous work [64] fails to show any improvement. Similar results can be observed on CIFAR datasets as shown in Tab. 3.2, where wide-DeepMoE improves the prediction accuracy of the baseline ResNet by $3\sim4\%$ on CIFAR-100 and 0.5% on CIFAR-10.

Memory Usage

Another aspect to consider about DeepMoE is its memory footprint (proportional to the number of parameters). We examine the memory usage of wide-DeepMoE with widened ResNet-50 as

the backbone network and compare it to the standard ResNet-101 which has a similar prediction accuracy. We find that wide-DeepMoE using Bottleneck-A (Fig. 3.2b) achieves a 22.88% Top-1 error rate, which compared to ResNet-110 with an error rate of 22.63%, is only 0.2% lower in error but requires 20% less computation. Moreover, wide-DeepMoE using Bottleneck-B (Fig. 3.2c), which is more memory efficient than Bottleneck-A (Fig. 3.2b), achieves 22.84% top-1 error with 6% less parameters and 18% less FLOPs than the standard ResNet-101 indicating that wide-DeepMoE is competitive on the memory usage.

Narrow DeepMoE

In this section we compare DeepMoE to current static and dynamic channel pruning techniques. We show that DeepMoE is able to out-preform both dynamic and static channel pruning techniques in prediction accuracy while maintaining or reducing computational costs.

Narrow-DeepMoE *vs* Dynamic Channel Pruning

DeepMoE generalizes existing channel pruning work since it both dynamically prunes and *re-scales* channels to reduce the computational cost and improve accuracy. In previous dynamic channel pruning work [115], channels are pruned based on the outputs of previous layers. In contrast, the gate decisions in DeepMoEs are determined in advance based on the shared embedding (latent mixture weights) which enables improved batch parallelism at inference.

We compare DeepMoE to the latest dynamic channel pruning work RNP [115] with VGG-16[1] as the base model on CIFAR-100. As we can see from Fig. 3.3a, without fine-tuning, the prediction error and computation trade-off curve (dotted blue line) of DeepMoE is much flatter than RNP (dotted red line) which indicates DeepMoE has a greater reduction in computation without loss of accuracy. Moreover, when fine-tuning DeepMoE for only 10 epochs (dotted green line in Fig. 3.3a), DeepMoE improves the prediction accuracy by a large margin by 4% which is a $\sim 13\%$ improvement over the baseline VGG model due to the regularization effect of DeepMoE (Sec. 3.4).

Narrow-DeepMoE *vs* Static Channel Pruning

Similarly, DeepMoE outperforms the state-of-the-art static channel pruning results [126, 73, 113, 86] on both ImageNet shown in Tab. 3.3 and the CIFAR-10 dataset in Fig. 3.3b and 3.3c. DeepMoE with ResNet-50 reduces 56.8% of the computation of the standard ResNet-50 with a top-1 error rate of 26.21%, approximately 2% better than He *et al.* [78], which currently has the best accuracy for an equivalent amount of computation among previous work on ImageNet. Fig. 3.3b and 3.3c show that DeepMoE achieves a higher accuracy less computation times than current techniques.

[1] Our baseline accuracy is higher than RNP since we use a version with batch normalization in contrast to the published method.

Table 3.3: Pruned ResNet-50 on ImageNet.

Model	Top-1	Top-5	FLOPs(x10^9)	Reduct.(%)
SSS [86]	26.8	-	3.0	20.3
Li et al. [113]	27.0	8.9	3.0	19.0
He et al. [78]	-	9.2	1.9	50.0
ThiNet [126]	29.0	10.0	1.7	55.8
DeepMoE	**26.2**	**8.4**	**1.6**	**56.8**

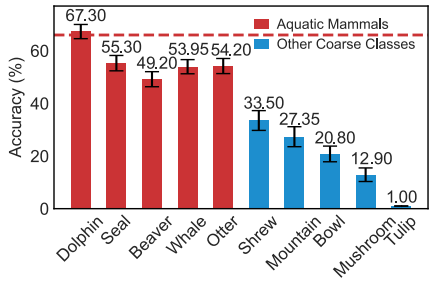


Figure 3.4: Gate embedding shuffling.

Analysis

In this section, we first analyze the effectiveness of the gating behavior in generating embeddings that are predictive of the class label and thus its ability balance expert utilization. We then study the regularization effect of DeepMoEs sparsification of the channel outputs. Lastly, we explore the effects of widening certain combinations of layers in a network as opposed to widening all convolutional layers as we do in DeepMoE.

Gating Behavior Analysis

To analyze the gating behavior of DeepMoE, we evaluate the trained DeepMoE with VGG-16 as follows: for a given fine-grained class A (e.g., *dolphin*), we re-assign the gate embedding for each input in class A with a randomly chosen gate embedding from other classes either within the same coarse category (referred to as *in-group shuffling*) or different categories (referred to as *out-of-group shuffling*).

In Fig. 3.4, we plot the test accuracy of class *dolphin*, belonging to the coarse category *aquatic mammals* with randomly selected gate embeddings (repeated 20 times for each input) from 5 classes in the same coarse category (in red) and 5 classes for other coarse categories (in blue). Fig. 3.4 shows that the test accuracy with in-group embeddings is 20-60% higher than with out-of-group shuffling. Especially when applying the gate embeddings from the tulip category, the test accuracy drops to 1% while the accuracy with in-group shuffling is mostly above 50%. This indicates that the latent mixture of weights are similar for semantically related image categories, and since DeepMoE is never given this coarse class structure, our results are significant.

Table 3.4: Different widening strategies for VGG16 on CIFAR-100.

Control	Model	Params	FLOPs (x10^8)	Acc. (%)
Params	W1-High	24.15M	3.51	71.96
	W1-Mid	24.16M	9.18	72.02
	W4-Low	24.18M	43.16	72.51
	W13-All	24.18M	8.15	**73.91**
Params & FLOPs	W1-High	24.15M	2.98	73.28
	W1-Mid	24.16M	2.74	72.68
	W4-Low	24.18M	2.45	73.33
	W13-All	24.18M	2.29	**73.39**

Regularization Effect of DeepMoE

Since DeepMoE sparsifies the channel outputs during training and testing, we study the regularization effect of such sparsification. We increase the number of channels of a modified ResNet-18 with bottleck-B (in Fig. 3.2c) by 2-8$\times$ on CIFAR-100. In Fig. 3.5, we plot the accuracy and computation FLOPs of the baseline widened ResNet-18 models (in blue) and wide-DeepMoE (in orange) with $\lambda = 2$.

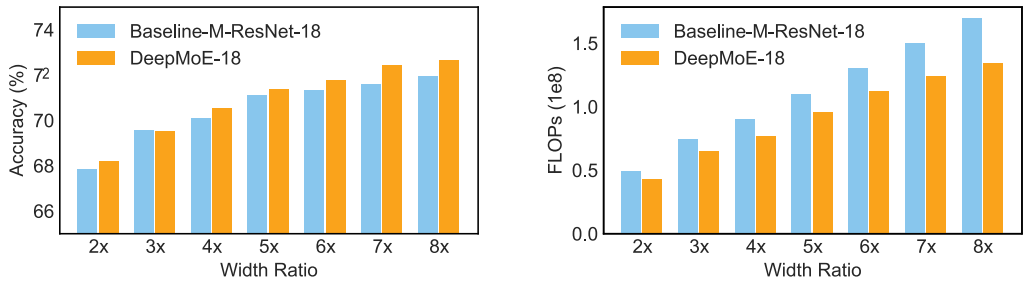


Figure 3.5: Regularization Effect of DeepMoE on Widened ResNet.

Fig. 3.5 suggests that DeepMoE has a lower computation cost and higher accuracy than the baseline widened ResNet, and the advantages of DeepMoE increase with the width of the base convolutional network. This indicates a potential regularization effect to the DeepMoE design.

DeepMoE *vs* Single-Layer MoE

So far in our experiments, we have widened the network by increasing the number of experts/channels for all convolutional layers. Here, we study other strategies for widening the network. We try to widen the VGG-16 model in four different kinds of layers: the top layer (W1-High), the middle layer (W1-Mid), the lower 4 layers (W4-Low), and finally all the 13 convolutional layers (W13-All) as used in all the other experiments (details in Sec. A.2).

As shown in Tab. 3.4, the prediction accuracy of W13-All is strictly better than that of a single-layer MoE, even though they have the same number of parameters. Adding MoE to the bottom

Table 3.5: Segmentation Results on CityScapes.

	Road	Sidewalk	Building	Wall	Fence	Pole	Light	Sign	Vegetation	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motorcycle	Bicycle	mIoU	FLOPs($\times 10^9$)
DRN-A-50	96.9	77.4	90.3	35.8	42.8	59.0	66.8	74.5	91.6	57.0	93.4	78.7	55.3	92.1	43.2	59.5	36.2	52.0	75.2	67.3	**703**
wide-DeepMoE-50-A	97.2	78.9	90.3	45.6	48.4	56.2	61.6	72.9	91.6	60.7	94.2	77.4	50.6	92.5	48.7	68.7	44.1	52.7	74.2	**68.8**	804
wide-DRN-A-50	97.4	80.6	90.6	38.5	49.0	58.7	65.1	73.4	91.8	59.5	93.9	78.2	51.1	92.9	49.1	68.7	51.3	52.2	74.5	69.3	2173
wide-DeepMoE-50-B	97.5	80.4	91.0	48.9	50.6	58.5	65.7	75.3	92.0	60.1	94.7	79.2	54.7	93.2	53.8	73.2	53.2	54.8	75.6	**71.2**	**1738**

or top layers is more effective than adding it to the middle layer. Alternatively, if we control both the number of parameters and the computation FLOPs, the accuracy differences between different strategies are reduced but W13-All is still favorable to other widening strategies.

Semantic Segmentation

Semantic image segmentation requires predictions for each pixel, instead of one label for the whole image in classification. We evaluate DeepMoE on the segmentation task to understand its generalizability. In specific, we apply DeepMoE to DRN-A [206], which adopts ResNet architecture as the backbone, and evaluate the results on the popular segmentation dataset CityScapes [26]. We follow the same training procedure as Yu *et al.* [206] for fair comparison. The optimizer is SGD with momentum 0.9 and crop size 832. The starting learning rate is set to 5e-4 and divided by 10 after 200 epochs. The intersection-over-union (IoU) scores and computation costs in FLOPs of DeepMoE are presented in Tab. 3.5.

The hyper-parameter λ can adjust the trade-offs between computer efficiency and prediction accuracy. Our efficient model `wide-DeepMoE-50-A` beats the baseline by 1.5% of mIoU with a slight increase in FLOPs, while our accurate model `wide-DeepMoE-50-B` outperforms the wide baseline by almost 2% mIoU with lower FLOPs. These results indicate DeepMoE is effective on pixel-level prediction such as semantic segmentation as well as image classification.

3.5 Related Work

Mixture of experts. Jacobs et al. [89] introduced the original formulation of mixture of experts (MoE) models. In this early work, they describe a learning procedure for systems composed of many separate neural networks each devoted to subsets of the training data. Later work [24, 25, 91] applied the MoE idea to classic machine learning algorithms such as support vector machines. More recently, several [160, 64, 1] have proposed MoE variants for deep learning in language modeling and image recognition domains. These more recent efforts to combine deep learning and mixtures of experts have focused on mixtures of deep sub-networks rather than stacking many mixture of expert models. While preliminary work by Eigen et al. [45] explored stacked MoE models, they only successfully demonstrated networks up to depth two and only evaluated their design on MNIST Digits. In contrast, we construct deep models with hundreds of MoE layers

based on a shared shallow embedding rather than the layer outputs [45] which makes DeepMoE more suitable to parallel hardware with batch parallelism as the gate decisions are pre-determined. We also address several of the key challenges around the design and training of multi-layer MoE models. More recently, the mixture of experts design has been applied in different applications, e.g., video captioning [187], multi-task learning [128], etc.

Conditional computation. Related to mixture of experts, recent works by Bengio et al. [8, 9, 22] explored conditional computation in the context of neural networks which selectively executes part of the network based on the input. They use reinforcement learning (RL) for the discrete selection decisions which are delicate to train while our sparsely-gated DeepMoE design can be embedded into standard convolutional networks and optimized with stochastic gradient descent.

Dynamic channel pruning. To reduce storage and computation overhead, many [113, 78, 126] have explored channel level pruning which removes entire channels at each layer in the network and thus leads to structured sparsity. However, permanently dropping channels limits the network capacity. Bridging conditional computation and channel pruning, recent works [115, 188, 197] have explored dynamic pruning, which use per-layer gating networks to dynamically drop individual channels or entire layers based on the output of previous layers. Therefore the channels to be dropped are dependent on the input, resulting in a more expressive network than one that applies static pruning techniques. Like the work on conditional computation [186], dynamic pruning relies on sample inefficient reinforcement learning techniques to train many convolutional gates. In this work, we generalize the earlier work on dynamic channel pruning by introducing a more efficient shared convolutional embedding and simple ReLU based gates to enable sparsification and feature re-calibration and allowing end-to-end training using stochastic gradient descent.

3.6 Chapter Summary

In this work we introduced our design of deep mixture of experts models, which produces a more accurate and computationally inexpensive model for computer vision applications. Our DeepMoE architecture leverages a shallow embedding network to construct latent mixture weights, which is then used by sparse multi-headed gating networks to select and re-weight individual channels at each layer in the deep convolutional network. This design in conjunction with a novel sparsifying and diversifying loss enabled joint differentiable training, addressing the key limitations of existing mixture of experts approaches in deep learning. We provided theoretical analysis on the expressive power of DeepMoE and proposed two design variants. The extensive experimental evaluation indicated that DeepMoE can reduce computation and surpass accuracy over baseline convolutional networks, as well as improving upon the residual network result on the challenging ImageNet benchmark by a full 1%. Through our analysis we were also able to prove that our embedding and gating network is able to resolve coarse grain class structure in the underlying problem. This work shows promising results when applied to semantic segmentation tasks, and could be incredibly useful for various other problems.

Chapter 4

Uncertainty-Aware Model Cascades

4.1 Overview

Advances in deep learning have enabled substantial recent progress on challenging machine learning benchmarks. As a consequence, deep learning is being deployed in real-world applications, ranging from automated video surveillance, to voice powered personal assistants, to self-driving cars. In these applications, accurate predictions must be delivered in *real-time* (e.g, under 200ms) under *heavy query load* (e.g., processing millions of streams) with *limited resources* (e.g., limited GPUs and power).

The need for accurate, low-latency, high-throughput, and low-cost predictions has forced the machine learning community to explore a complex trade-off space spanning model and system design. For example, several researchers have investigated techniques for performing deep learning model compression [205, 35, 33]. However, model compression primarily reduces model memory requirements so as to fit on mobile devices or in other energy-bounded settings. There is a limit to how far compression-based techniques can be pushed to reduce latency at inference time while retaining state-of-the-art accuracy across all inputs.

We conjecture that in the pursuit of improved classification accuracy the machine learning community has developed models that effectively *"overthink"* on an increasing fraction of queries. To support this conjecture we show that while the cost of computing predictions has increased by an order of magnitude over the past 5 years, the accuracy of predictions on a large fraction of the ImageNet 2012 validation images has remained constant (see Fig. 4.2). This observation suggests that *if* we could distinguish between easy and challenging inputs (e.g., images) and only apply more advanced models when necessary, we could reduce computational costs without impacting accuracy. In this paper we study the design of *prediction cascades* as a mechanism to *exploit this conjecture* by combining fast models with accurate models to increase throughput and reduce mean latency without a loss in accuracy.

Though prediction cascades are well established in the machine learning literature [156, 182, 4, 13], the classic approaches focused on detection tasks and developed cascades for early rejection of negative object or region proposals – leveraging the class asymmetry of detection tasks. In this

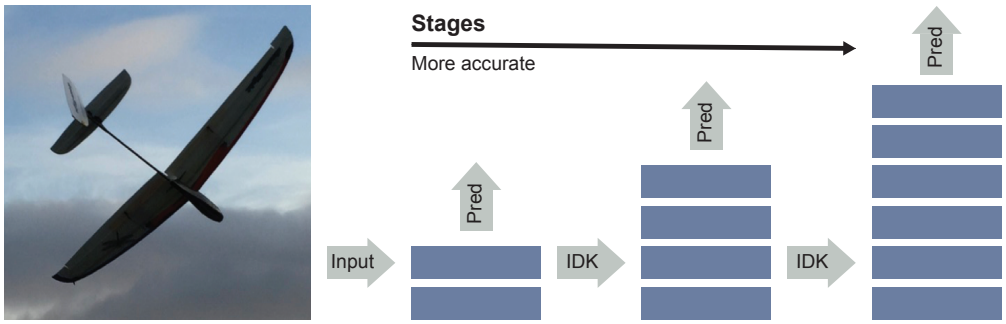


Figure 4.1: IDK cascade illustration.

paper, we revisit the question of how to effectively build model cascades with little training overhead to trade off between prediction accuracy and cost, extending to any multi-class classification task.

We introduce *IDK prediction cascades*, a general framework to accelerate inference without reducing prediction accuracy by composing pre-trained models. IDK prediction cascades (see Fig. 4.1) are composed of IDK classifiers which are constructed by attaching an augmenting classifier to the existing classifiers, *base models*, enabling the IDK classifiers to predict an auxiliary *"I don't know"* (IDK) class besides the original prediction classes. The augmenting classifiers, which are *light-weighted* (the computational cost is negligible compared to the cost of the base models) and *independent* from the base model architectures, measure the uncertainty of the base model predictions. When an IDK classifier predicts the IDK class the subsequent model in the cascade is invoked. The process is repeated until either a model in the cascade predicts a real class or the end of the cascade is reached at which point the last model must render a prediction. Furthermore, we can introduce a human expert as the last model in an IDK cascade to achieve nearly perfect accuracy while minimizing *the cost* of human intervention.

The base models in the model cascades are treated as black-boxes and thus the proposed framework can be applied to any existing model serving systems [27, 140] with little modification. In addition, the proposed IDK cascade framework model naturally fits the edge-cloud scenario where the fast models can be deployed on edge devices (e.g. Nvidia Drive PX2, Jetson TX2, etc.) while the expensive models are stored in the cloud and are only triggered when the fast model is not certain about a prediction.

To build such model cascade, we need to address the following problems: (1) without retraining the base models or obtaining the base model architecture, what is the best measure available to distinguish the easy and hard examples in the workload? (2) to construct the IDK classifiers that can effectively decide the execution path for the given input while not introducing additional computational overhead, what is the proper objective to balance the computational cost and the overall prediction accuracy of the model cascades?

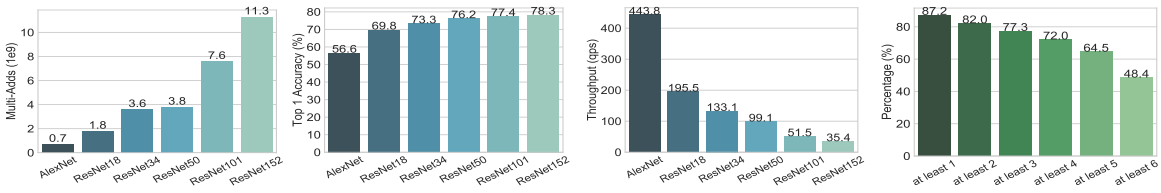
In this work, we propose two search-based methods for constructing IDK classifiers: *cascading by probability* and *cascading by entropy*. Cascade by probability examines the confidence scores of a model directly to estimate uncertainty. Cascade by entropy leverages well-calibrated class conditional probabilities to estimate model uncertainty. Both techniques then search to find the

optimal uncertainty threshold at which to predict the IDK class for each model in the cascade. When the uncertainty in a predicted class exceeds this threshold, the model predicts the IDK class instead. While both search-based methods produce reasonable prediction cascades, neither leverages the cascade design when *training* the augmenting classifier.

As a third approach to constructing IDK classifiers we cast the IDK cascade problem in the context of empirical risk minimization with an additional computational cost term and describe how the objective can be easily incorporated into gradient based learning procedures. The empirical risk minimization based approach allows the IDK classifier to trade-off between *cascade accuracy* and *computational cost* when building the prediction cascade.

We apply all three techniques to the image classification task on ImageNet2012 and CIFAR10 to demonstrate we can reduce computation by 37%, resulting in a 1.6x increase in throughput, while maintaining state-of-the-art accuracy. We conduct a detailed study of the impact of adding the computational-cost term to the objective and show that it is critical for training the augmenting classifiers. Compared to the cascades built with cost-oblivious objectives which cannot usually achieve the desired accuracy, the proposed cost-aware objective better serves the goal of model cascading. Furthermore, we demonstrate that in a real autonomous vehicle setting the IDK cascades framework can be applied in conjunction with human experts to achieve 95% accuracy on driving motion prediction task while requiring human intervention less than 30% of the time.

4.2 Motivating Example



(a) Multi-Adds (b) Accuracy (c) Throughput (d) Easy Examples

Figure 4.2: ImageNet model statistics.

In the pursuit of improved accuracy, deep learning models are becoming increasingly expensive to evaluate. To illustrate this trend, in Fig. 4.2c we plot the throughput for six benchmark models on the ImageNet 2012 datasets. We observe that prediction throughput has decreased by more than an order of magnitude. We expect the trend towards more costly models to continue with improvements in model design and increased adoption of ensemble methods [79]. In contrast, as Fig. 4.2b shows, the gains in prediction accuracy have increased much more slowly. A result of this trend is that even the cheaper and less advanced models can correctly classify many of the examples.

Easy Samples: For many prediction tasks, less accurate models are adequate *most of the time*. For example, a security camera may observe an empty street most of the time and require a more sophisticated model only in the infrequent events that people or objects enter the scene. Even in the standard benchmarks, many of the examples can be correctly classified by older, less advanced

models. In Fig. 4.2d we plot the percentage of images that were correctly classified by an increasing fraction of models. We observe that a large fraction ($\approx 48\%$) of the images are correctly classified by all six of the models, suggesting that these images are perhaps *inherently easier* and may not require the recent substantial increases in model complexity and computational cost.

4.3 IDK Prediction Cascades

We start describing the IDK prediction cascade framework by examining simple two model cascades and then extend these techniques to deeper cascades at the end of this section. We start by formalizing two element cascades for the multiclass prediction problem.

We consider the k class multiclass prediction problem in which we are given two pre-trained models: (1) a fast but less accurate model m^{fast} and (2) an accurate but more costly model m^{acc}. In addition, we assume that the fast model estimates the class conditional probability:

$$m^{\text{fast}}(x) = \hat{\mathbf{P}}(\text{class label} \mid x). \tag{4.1}$$

Many multi-class estimators (e.g., DNNs trained using cross entropy) provide class conditional probabilities. In addition, we are given a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ consisting of n labeled data points.

To develop IDK prediction cascades we introduce an additional **augmenting classifier**:

$$h_\alpha(m^{\text{fast}}(x)) \to [0, 1], \tag{4.2}$$

which evaluates the *distributional output* of $m^{\text{fast}}(x)$ and returns a number between 0 and 1 encoding how *uncertain* the fast model $m^{\text{fast}}(x)$ is about a given prediction. The **IDK classifier** is composed of the base model and the augmenting classifier. For simplicity, we will refer to training the augmenting classifier as training the IDK classifier. In this paper we consider several designs for the IDK classifier:

$$h_\alpha^{\text{prb}}(m^{\text{fast}}(x)) = \mathbb{I}\left[\max_j m^{\text{fast}}(x)_j < \alpha\right] \tag{4.3}$$

$$h_\alpha^{\text{ent}}(m^{\text{fast}}(x)) = \mathbb{I}\left[\mathbf{H}\left[m^{\text{fast}}(x)\right] > \alpha\right] \tag{4.4}$$

$$h_\alpha^{\text{cst}}(m^{\text{fast}}(x)) = \sigma\left(\alpha_1 f_{\alpha_2}\left(m^{\text{fast}}(x)\right) + \alpha_0\right), \tag{4.5}$$

where $\mathbb{I}$ is the indicator function, $\mathbf{H}$ is the *entropy* function:

$$\mathbf{H}[m^{\text{fast}}(x)] = -\sum_{j=1}^{k} m^{\text{fast}}(x)_j \cdot \log m^{\text{fast}}(x)_j, \tag{4.6}$$

and f is a feature representation of $m^{\text{fast}}(x)$.

While f can be any featurization of the prediction $m^{\text{fast}}(x)$, in this work we focus on the entropy featurization $f = \mathbf{H}$ as this is a natural measure of uncertainty. When using the entropy featurization,

the IDK classifier h^{cst} becomes a differentiable approximation of h^{ent} enabling direct cost based optimization. In our experimental evaluation, we also evaluate $f_{\alpha_2}(m^{\text{fast}}(x)) = \mathbf{NN}_{\alpha_2}(m^{\text{fast}}(x))$ which recovers a neural feature encoding of x and allows us to assess a neural network based IDK classifier in the context of the differentiable cost based optimization.

Given an IDK classifier $h_\alpha(\cdot)$ we can define a two element IDK prediction cascade as:

$$m^{\text{casc}}(x) = \begin{cases} m^{\text{fast}}(x) & \text{if } h_\alpha(m^{\text{fast}}(x)) <= 0.5 \\ m^{\text{acc}}(x) & \text{otherwise.} \end{cases} \tag{4.7}$$

Thus, for a given choice of IDK classifier h_α we only need to determine the optimal value for parameter α to ensure maximum accuracy while minimizing the fraction of examples for which the more expensive model is required. In the following, we formalize this objective and describe a set of techniques for choosing the optimal value of α.

Given the above definition of an IDK prediction cascade we can define two quantities of interest. We define the accuracy $\mathbf{Acc}(m)$ of a model m as the zero-one prediction accuracy evaluated on our training data $\mathcal{D}$:

$$\mathbf{Acc}(m) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}\left[y_i = \arg\max_j m(x_i)_j\right]. \tag{4.8}$$

We define the IDK rate $\mathbf{IDKRate}(h)$ of an IDK classifier h as the fraction of training examples that are evaluated by the next model in the cascade:

$$\mathbf{IDKRate}(h) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}\left[h_\alpha(m^{\text{fast}}(x_i)) > 0.5\right]. \tag{4.9}$$

Our goal in designing a prediction cascade is then to maintain the accuracy of the more accurate model while minimizing the IDK rate (i.e., the fraction of examples that require the more costly m^{acc} model). We formalize this goal as:

$$\min_\alpha \ \mathbf{IDKRate}(h_\alpha) \tag{4.10}$$

$$\text{s.t.:} \quad \mathbf{Acc}(m^{\text{casc}}) \geq (1-\epsilon)\mathbf{Acc}(m^{\text{acc}}). \tag{4.11}$$

In the following we describe a set of search procedures for achieving this goal for each of the IDK classifier designs.

Baseline Uncertainty Cascades

Similar to [93], as a baseline, we propose using the *confidence scores* (i.e. probability over the predicted class) of $m^{\text{fast}}(x)$ and follow the IDK classifier design (Eq. 4.3). The intuition is that if the prediction of m^{fast} is insufficiently confident then the more accurate classifier is invoked.

A more rigorous measure of prediction uncertainty is the entropy of the class conditional probability. We therefore propose an entropy based IDK classifier in Eq. 4.4. The entropy based IDK classifier captures the overall uncertainty as well as the certainty within the dominant class.

Due to the indicator functions neither the *cascade by probability* (Eq. 4.3) nor the *cascade by entropy* functions (Eq. 4.4) are differentiable. However, because the parameter α is a single scalar, we can apply a simple grid to search procedure to find the optimal value for the threshold α.

Regularizing for Prediction Cost

The uncertainty based cascades described above adopt a relatively simple IDK classifier formulation and rely on grid search to select the optimal parameters. However, by reframing the cascade objective in the context of regularized empirical risk minimization and defining a differentiable regularized loss we can admit more complex IDK classifiers.

In the framework of empirical risk minimization, we define the objective as the sum of the loss $\mathbf{L}(\cdot, \cdot)$ plus the *computational cost* $\mathbf{C}(\cdot)$ of invoking the cascaded model:

$$J(\alpha) = \frac{1}{n} \sum_{i=1}^{n} [\mathbf{L}(y_i, m_\alpha^{\text{casc}}(x_i)) + \lambda \cdot \mathbf{C}(m_\alpha^{\text{casc}}(x_i))] \tag{4.12}$$

where λ is a hyper parameter which determines the trade-off between the cascade accuracy and the computational cost. Because we are not directly optimizing the fast or accurate models we can adopt the zero-one loss which is compatible with our earlier accuracy goal:

$$\begin{aligned} \mathbf{L}(y_i, m_\alpha^{\text{casc}}(x_i)) = &\big(1 - h_\alpha(m^{\text{fast}}(x))\big) \cdot \mathbb{I}[y_i, m^{\text{fast}}(x_i)] \\ &+ h_\alpha(m^{\text{fast}}(x)) \cdot \mathbb{I}[y_i, m^{\text{acc}}(x_i)], \end{aligned} \tag{4.13}$$

where $\mathbb{I}[y_i, m(x_i)]$ is 1 if $\arg\max_j m(x_i)_j \neq y_i$ and 0 otherwise. The IDK classifier $h_\alpha(m^{\text{fast}}(x))$ governs which loss is incurred. It is worth noting that alternative loss formulations could be used to further fine-tune the underlying fast and accurate model parameters in the cascaded setting.

The cascaded prediction cost $\mathbf{C}(\cdot)$ is defined as:

$$\mathbf{C}(m_\alpha^{\text{casc}}(x_i)) = c^{\text{fast}} + h_\alpha(m^{\text{fast}}(x)) \cdot c^{\text{acc}}, \tag{4.14}$$

where the computational cost of m^{fast} and m^{acc}, are denoted by c^{fast} and c^{acc} respectively. In practice, the cost of model m^{fast} and m^{acc} could be measured in terms of multi-adds, latency, or number of parameters. The formulation of Eq. 4.14 captures the cascade formulation in which the fast model is always evaluated and the accurate model is evaluated conditioned on the IDK classifier decision.

Combining both prediction loss and computational cost of the IDK cascade, we can now use this *regularized loss* objective function to optimize the IDK prediction cascade with stochastic gradient descent based algorithms. This objective allows us to optimize both prediction precision and overall computational cost in one pass and support more complex parametric IDK classifiers.

Beyond Two Element Cascades

We can extend the two element cascade to construct deeper cascades by introducing additional IDK classifiers between each model and then either optimizing the IDK classifier parameters in

a stage-wise fashion or by jointly optimizing the IDK classifiers using the extended loss function. More precisely, for an N-model cascade where m^j is the j-th model in the cascade, we define $N-1$ IDK classifiers $h_{\alpha_j}(m^j(x))$. For non-differentiable IDK classifiers with scalar parameters we can apply the grid search procedure in a stage wise fashion starting with the least accurate model. For more complex differentiable IDK classifiers we can define an extended loss:

$$\mathbf{L}(y_i, m_\alpha^{\text{casc}}(x_i)) = \sum_{j=1}^{N} \prod_{q=0}^{j-1} p_q (1 - p_j) \mathbb{I}[y_i, m^j(x_i)] \tag{4.15}$$

where $p_j = h_{\alpha_j}(m^j(x))$ and $p_0 = p_N = 1$. The computational cost function $\mathbf{C}(\cdot)$ is then generalized as

$$\mathbf{C}(m_\alpha^{\text{casc}}(x_i)) = \sum_{j=1}^{N} \prod_{q=0}^{j-1} p_q c^j \tag{4.16}$$

4.4 Experiments

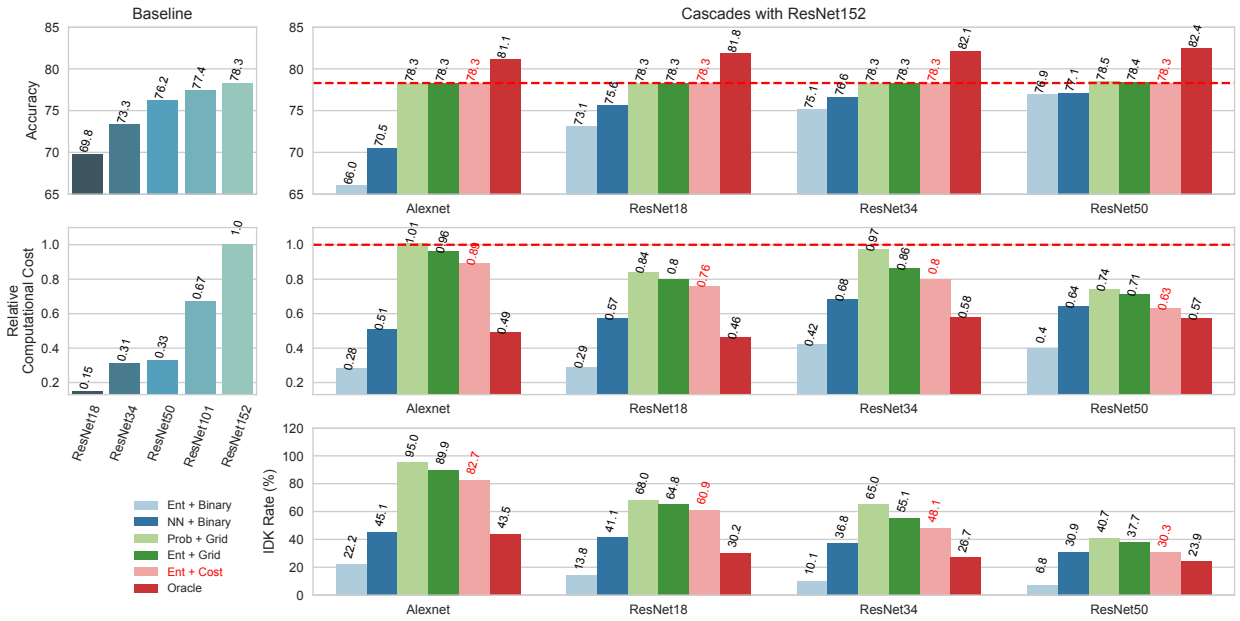


Figure 4.3: ImageNet Two Model Cascades.

In this section, we evaluate the proposed cascading methods in two scenarios: cascading machine learning models with different computation budgets and collaboration between algorithms and human.

To study the prediction accuracy and cost trade-off under each cascade design, we use standard image classification benchmark tasks and models. We evaluate the cascaded models on ImageNet 2012 [103] and CIFAR-10 [102]. We assess whether the proposed IDK cascade approaches can

match the state-of-the-art accuracy while significantly reducing the cost of rendering predictions. We also evaluate the robustness of the proposed framework on CIFAR-10.

To assess how cascades can be used to augment models with human intervention, we evaluate a motion prediction task, a representative of autonomous vehicle workloads [202]. In this human-in-the-loop prediction task, the human serves as the *accurate* model to further improve the accuracy and safety of autonomous driving. The cascade design is used to determine when the fast model can no longer be trusted and human intervention is required (e.g., by taking over steering).

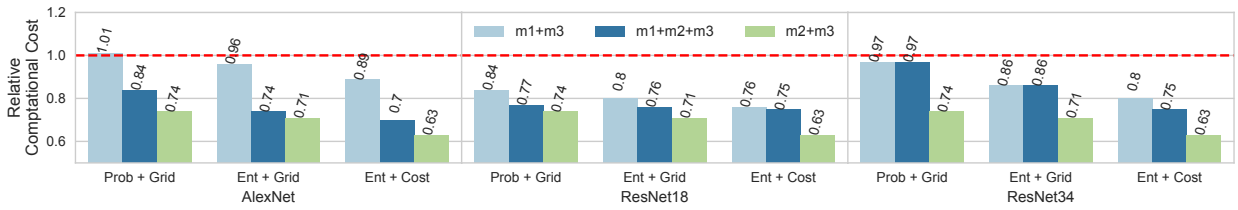


Figure 4.4: Three Model Cascade Results.

We evaluate each cascade using a range of different metrics. The **accuracy** and **IDK rate** correspond to the accuracy and IDK rate defined in Eq. 4.8 and Eq. 4.9 respectively. In the multi-model cascade setting, we measure **IDK Rate** at each level in the cascade. As a measure of computational cost we compute the **average flops** which is the average floating point arithmetic operations required by the model cascade. Finally, as a relative measure of runtime we compute the **relative computational cost** which is computed as $\text{FLOP}_{\text{casc}}/\text{FLOP}_{\text{acc}}$.

Image Classification on ImageNet

We first demonstrate that the proposed IDK model cascade framework can preserve the accuracy of the expensive models without a loss while reducing the overall computational cost.

Experimental Details

On the ImageNet 2012 dataset we study cascades assembled from pre-trained models including AlexNet [104] and residual networks of various depths including ResNet-18, ResNet-34, ResNet-50, and ResNet-152 [73]. Detailed statistics such as top-1 accuracy, FLOPs, etc of the models are shown in Fig. 4.2. To train the IDK classifiers, we sample 25.6K training images randomly from the ImageNet 2012 training data and report the cascade accuracy on the entire ImageNet 2012 validation data. In grid search for cascade by probability and entropy, we evaluate 100 different settings of α and select the cascade which has the lowest IDK rate while reaching the desired accuracy. Because 1% reduction in accuracy can translate to a nearly 30% reduction in computational cost on ImageNet (as measured in flops), we set the desired accuracy to be the same as the ResNet-152 (i.e., setting $\epsilon = 0$ in Eq. 4.11).

For cascade by entropy via cost-aware objective, we set the hyper-parameter $\lambda = 0.04$ across different model combinations and use the actual FLOPs number of each model as the model cost in

the objective. We also compare against a cascade constructed using an oracle IDK classifier as a cascade accuracy upper bound which optimally selects between the fast and accurate models. In addition to proposed cascade designs, we include two more IDK cascades constructed by supervised training an IDK classifier of the form in Eq. 4.5 using the oracle labels with cost-oblivious objectives. We discuss these alternative baselines in more detail in the next section.

Computation Reduction

Detailed results are shown in Fig. 4.3. We find the best cascade design employs the Entropy features and regularized cost formulation to combine the ResNet-50 and ResNet-152 models. This cascade is able to reduce prediction costs by 37% while achieving the accuracy of the most computationally expensive model. This is also close to the oracle performance, though it assumes a perfect IDK classifier (i.e. the IDK classifier can distinguish the correctness predictions of the fast model with 100% accuracy and only passing the incorrect predictions to the accurate model). In general we find that our regularized cost based formulation outperforms the other baseline techniques.

Effectiveness of Cost-Aware Objective

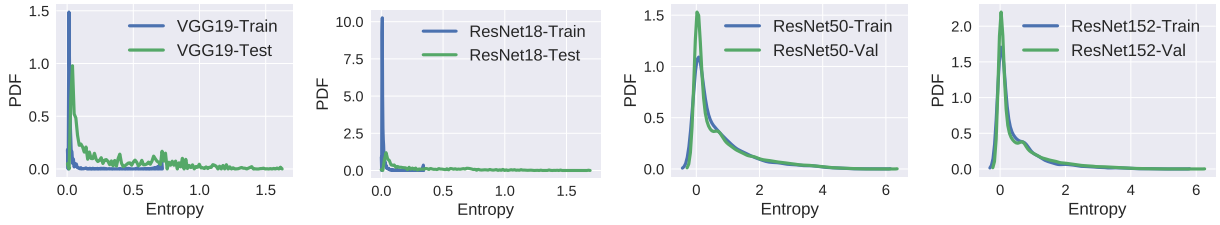
In Fig. 4.3 we also compare the proposed cost based IDK cascade design with two IDK classifiers following the form of Eq. 4.5 with cost-oblivious cross entropy loss. The training labels are the correctness of the predictions of the fast model evaluated on the ImageNet2012 dataset. We consider two forms of the feature function f: entropy based features identical to the cost based cascade and neural network features. The neural network feature function $f_{\alpha_2}(m^{\text{fast}}(x))$ consists of a 7-layer fully connected network with 1024, 1024, 512, 512, 128, and 64 hidden units, ReLU activation functions, and trained using stochastic gradient descent with momentum and batch normalization. In general, we find that these sophisticated baselines are unable to accurately predict the success of the fast model and as a consequence are unable to match the accuracy of the cost based cascade formulation. With the cost-aware objective, the IDK cascades can meet the desired accuracy which shows that the proposed objective is more suitable for building model cascades.

Three Model Cascades

We also investigate three model cascades and the results are shown in Fig. 4.4. Compared to the two models ResNet-50 + ResNet-152 cascade, adding a faster model like AlexNet, ResNet-18 or ResNet-34 actually *increases* computational cost, because a reasonable fraction of examples will need to pass through all three models in the cascade. However, the three-model cascade tends to reduce the computational cost relative to a two-model cascade including a less accurate model than ResNet-50. Moreover, adding more accurate models within a cascade consistently improved overall cascade performance.

Table 4.1: CIFAR Model Details

Model	% Train Acc	% Test Acc	Flops (10^7)
VGG19	99.996	93.66	39.8
ResNet18	100.00	95.26	3.7
DLA-48-B-s	99.204	89.06	1.7



(a) VGG19 on CIFAR (b) ResNet-18 on CIFAR (c) ResNet-50 on ImageNet (d) ResNet-152 on ImageNet

Figure 4.5: Entropy distribution.

Table 4.2: CIFAR Model DLA-48-B-s + ResNet-18 Cascade Results.

Type	Desired Acc (%)	Acc (%)	IDK Rate (%)	Avg Flops ($\times 10^7$)	Relative Computational Cost
Prob + Grid	95.26	95.23	67.5	4.198	1.135
Entropy + Grid	95.26	95.23	51.7	3.613	0.977
Entropy + Cost	95.26	95.22	**48.9**	**3.509**	**0.949**
Entropy + Grid	95.16	95.16	40.1	3.184	0.861
Entropy + Cost	95.16	95.16	**40.0**	**3.179**	**0.859**
Prob + Grid	95.05	95.07	36.5	3.051	0.824
Entropy + Grid	95.05	95.06	33.7	2.947	0.796
Entropy + Cost	95.05	95.05	**32.8**	**2.915**	**0.788**
Prob + Grid	94.90	94.89	29.9	2.806	0.759
Entropy + Grid	94.90	94.91	30.6	2.832	0.766
Entropy + Cost	94.90	94.91	**30.5**	**2.827**	**0.764**

Robustness Analysis on CIFAR-10

To further analyze the robustness of the IDK prediction cascades, we conduct a set of experiments on the CIFAR-10 datasets. We consider three models: ResNet-18 [73], VGG19 [164] and a recently proposed compact model DLA-48-B-s [207]. Tab. 4.1 shows details of the models. As we can see from the table, all models overfit the training data with high accuracy close to 100%. We want to study the robustness of the IDK prediction cascades under such extreme case. In this experiment, since VGG19 is less accurate and more costly than ResNet-18, we focus on cascades constructed with DLA-48-B-s and ResNet-18.

We evaluate the model cascades with four different cascade accuracy goals shown in Tab. 4.2. We observe cascade by entropy via the cost-aware objective consistently outperforms grid search

methods. Also, by admitting a small 0.03% reduction in accuracy, the IDK rate drops substantially from 48.9% to 30.5%. Compared to the single expensive model, the best model cascade reduces computational costs by 24%.

Robustness analysis

The proposed IDK classifiers rely on various measures of uncertainty in the class conditional probability distribution and are therefore sensitive to over confidence often as a result of over-fitting. To assess this effect, we evaluate the entropy distribution of the VGG19 and ResNet-18 models which have been trained to near perfect training accuracy (see Tab. 4.1). We plot the entropy distribution of these models in Figure 4.5a and 4.5b on both training and held-out test data and observe that both models substantially over-estimate their confidence on training data when compared with test data. In contrast, the ResNet-50 and ResNet-152 models are much better estimators of prediction uncertainty as seen in Figures 4.5c and 4.5d. As a consequence, in settings where the fast model is likely to over-fit it is important to use separate held-out data when training the IDK classifier.

Figure 4.6: Sample frames from the Berkeley Deep Drive Dataset

Driving Control Prediction

We evaluate IDK cascades for autonomous driving and demonstrate that we can achieve nearly perfect accuracy with less than 30% human intervention. In this experiment, we apply the IDK model cascade framework on Berkeley Deep Drive dataset, a large scale real driving video dataset [202] which contains the 2.6 million frames in the training video and 384,599 frames for testing. The driving dataset contains 4 discrete motion states: left turn, right turn, forward and stop. The task is to predict the next vehicle motion given previous video frames. Fig. 4.6 shows some sample frames in the dataset. We use the *Long-term Recurrent Convolutional Networks* (LRCN) [40] model as m^{fast} and experiment on a large scale driving video dataset [202]. We consider a human-in-the-loop setting where human serves as the m^{acc} with 100% accuracy.

We use the same training setting for the proposed cascade approaches as the image classification task and report the results in Tab. 4.3. The LRCN model has an accuracy of 84.5% [1] and we set the desired accuracy to 100%, 99% and 95%. We find that with only 28.88% human intervention, the cascade model can achieve 95% accuracy which is about 10% more accurate than the base LRCN model. This experiment demonstrate that the model cascade can be easily applied to real applications which are in high demand of low latency and high accuracy.

Table 4.3: Driving Model LRCN + Human Expert Cascades

Type	Desired Acc (%)	Acc (%)	IDK rate (%)
Prob + Grid	100.0	99.9	83.91
Ent + Grid	100.0	99.9	83.50
Ent + Cost	100.0	99.9	**80.70**
Prob + Grid	99.0	99.2	61.72
Ent + Grid	99.0	99.2	61.36
Ent + Cost	99.0	99.1	**59.70**
Prob + Grid	95.0	95.4	30.08
Ent + Grid	95.0	95.3	30.02
Ent + Cost	95.0	95.1	**28.88**

4.5 Related Work

Compression & Distillation. Much of the existing work to accelerate predictions from deep neural networks has focused on model compression [33, 35, 205, 69] and distillation [79]. Denton et al. [35] applied low-rank approximations to exploit redundancy in convolution parameters to achieve a factor of two speedup with only 1% reduction in accuracy. Han et al. [69] introduced quantization and Huffman encoding methods to reduce network sizes by orders of magnitude and reduce prediction cost by a factor of 3 to 4. Our work on IDK prediction cascades is complementary to the work on model compression and focuses exclusively on decreasing computation costs. In fact, by coupling model compression with IDK cascades it may be possible to support more aggressive lossy compression techniques. Alternatively, Hinton et al. [79] proposed using soft-targets to transfer knowledge from a costly ensemble to a single model while largely preserving prediction accuracy. Our approach does not require retraining base models and instead focuses on accelerating inference by using more complex models only when necessary. The existing model compression and distillation techniques can be used to construct the fast base models while our framework serves as a bridge to connect models with different levels of complexity and accuracy.

Cascaded Predictions. Prediction cascades are a well established method to improve prediction performance. Much of the early work on prediction cascades was developed in the context of face detection. While Viola and Jones [182] are credited with introducing the terminology of

[1] A slightly reduced accuracy early version was used.

prediction cascades, prior work by Rowley et al. [156] explored cascading neural networks by combining coarse candidate region detection with high accuracy face detection. More recently, Angelova et al. [4] proposed using deep network cascades and achieved real-time performances on pedestrian detection tasks. Cai et al. [13] also examined cascades for pedestrian detection, proposing a complexity aware term to regularize the cascade objective. While this approach has similarities to the loss function we propose, Cai et al. leverage the cost aware risk to choose an optimal ordering of cascade elements rather than to train a specific classifier. These papers all focus on using cascades for detection tasks, and only use the earlier models in the cascade to reject negative region or object proposals more cheaply. Positive detection (e.g. object identification) can only be made by the final model in the cascade, which is the only model that can predict the full set of classes in the prediction task. Recently, Huang et al. [85] applied the cascading concept by allowing early exiting within the model. Instead of cascading features of a single model, we aim to cascade the trained models (one may not know the model structure or not be able to retrain) in a practical scenario.

Uncertainty Classes. The introduction of an IDK class to capture prediction uncertainty has also been studied under other settings. [175, 94]. Trappenberg [175] introduced an *"I don't know"* (IDK) class to learn to identify input spaces with high uncertainty. Khani et al. [94] introduced a *"don't know"* class to enable classifiers to achieve perfect precision when learning semantic mappings. In both cases, the addition of an auxiliary uncertainty class is used to improve prediction accuracy rather than performance. We build on this work by using the IDK class in the construction of cascades to improve performance.

4.6 Chapter Summary

In this paper we revisited the classic idea of prediction cascades to reduce prediction costs. We extended the classic cascade framework focused on binary classifications to multi-class classification setting. We argue that the current deep learning models are "over-thinking" simple inputs in the majority of the real-world applications. Therefore, we aim to learn prediction cascades within the framework of empirical risk minimization and propose a new cost aware loss function, to leverage the accuracy and reduced cost of the IDK cascades.

We focused on build simple cascade with the the pre-trained base models with little training and negligible computation overhead. We tried to answer two questions in this paper: (1) what is a good measure to distinguish the easy and hard examples in the workload without querying much information about the mode itself? We found that the entropy value of the model prediction distribution is a good measure than the vanilla confidence score and can be used as input data to train a light-weighted but effective IDK classifier. (2) How to design the objective function that balances the prediction accuracy and the computation cost? We proposed in this work to use the cost regularized objective which utilizes the actual FLOPs of the base models as the cost measures. Incorporating the cost factor in the objective, we found the model cascade works more effectively than the model cascades with cost-oblivious function.

We also proposed two search based methods *cascade by probability* and *cascade by entropy*, which obtain reasonable performance and require no additional training. We evaluated these

techniques on both benchmarks and real-world datasets to show that our approach can successfully identify hard examples in the problem, and substantially reduce the number of invocations of the accurate model with negligible loss in accuracy. We also found that the cost based cascade formulation outperforms uncertainty based techniques.

We believe this work is a first step towards learning to compose models to reduce computational costs. Though not studied in this work, the proposed framework can be easily applied to the existing model serving systems and fit the edge-cloud scenario naturally with little modification. In the future, we would like to explore feature reusing and joint training of the cascade models so that different models can specialize in either easy or hard examples of the given workload.

Part II

Efficient Learning

Chapter 5

Task-Aware Weight Generation

5.1 Overview

Feature embeddings are central to computer vision. By mapping images into semantically rich vector spaces, feature embeddings extract key information that can be used for a wide range of prediction tasks. However, learning good feature embeddings typically requires substantial amounts of training data and computation. As a consequence, a common practice [39, 59, 211] is to re-use existing feature embeddings from convolutional networks (e.g., ResNet [73], VGG [165]) trained on large-scale labeled training datasets (e.g., ImageNet [157]); to achieve maximum accuracy, these general feature embedding are often fine-tuned [39, 59, 211] or transformed [81] using additional task specific training data.

In many settings, the training data are insufficient to learn or even adapt general feature embeddings to a given task. For example, in zero-shot and few-shot prediction tasks, the scarcity of training data forces the use of generic feature embeddings [107, 199, 213]. As a consequence, in these situations much of the research instead focuses on the design of joint task and data embeddings [15, 55, 213] that can be generalized to unseen tasks or tasks with fewer examples. Some have proposed treating the task embedding as linear separators and learning to generate them for new tasks [180, 124]. Others have proposed hallucinating additional training data [198, 70, 192]. However, in all cases, a common image embedding is shared across tasks. Therefore, the common image embedding may be out of the domain or sub-optimal for any individual prediction task and may be even worse for completely new tasks. This problem is exacerbated in settings where the number and diversity of training tasks is relatively small [54].

In this work, we explore the idea of dynamic feature representation by introducing the task-aware feature embedding network (TAFENet) with a meta-learning based design to transform generic image features to task-aware feature embeddings (TAFEs). As illustrated in Figure 5.1, the representation of TAFEs is adaptive to the given semantic task description, and thus able to accommodate the need of new tasks at testing time. The feature transformation is realized with a task-aware meta learner, which generates the parameters of feature embedding layers within the classification subnetwork shown in Figure 5.2. Through the use of TAFEs, we are able to adopt a

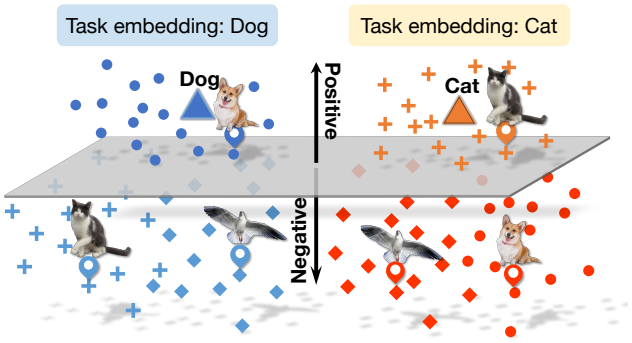


Figure 5.1: A cartoon illustration of Task-aware Feature Embeddings (TAFEs).

simple binary classifier to learn a task-independent linear boundary that can separate the positive and negative examples and generalize to new tasks.

We further propose two design innovations to address the challenges due to the limited number of training tasks [54] and the complexity of the parameter generation [10]. Dealing with the limited tasks, we couple the task embedding to the task aware feature embeddings with a novel embedding loss based on metric learning. The resulting coupling improves generalization across tasks by jointly clustering both images and tasks. Moreover, the parameter generation requires predicting a large number of weights from a low dimensional task embedding (e.g., a 300-dimensional vector extracted with GloVe [141]), which can be complicated and even infeasible to train in practice, we therefore introduce a novel decomposition to factorize the weights into a small set of task-specific weights needed for generation on the fly and a large set of static weights shared across all tasks.

We conduct an extensive experimental evaluation in Section 5.3. The proposed TAFENet exceeds the state-of-the-art zero-shot learning approaches on four out of five standard benchmarks (Section 5.3) without the need of additional data generation, a complementary approach that has shown boosted performance compared to mere discriminative models by the recent work [198]. On the newly proposed unseen attribute-object composition recognition task [134], we are able to achieve an improvement of 4 to 15 points over the state-of-the-art (Section 5.3). Furthermore, the proposed architecture can be naturally applied to few-shot learning (Section 5.3), achieving competitive results on the ImageNet based benchmark introduced by Hariharan *et al.* [70]. The code is available at https://github.com/ucbdrive/tafe-net.

5.2 Task-Aware Feature Embedding

As already widely recognized, feature embeddings are the fundamental building blocks for many applications [103, 123, 58] in computer vision. In this work, we focus on the construction of task-aware feature embeddings (TAFEs), a type of dynamic image feature representation that is adaptive to the given task. We observe that such dynamic feature representation can find its applications in the zero-shot learning, few-shot learning and unseen attribute-object pair recognition tasks to improve model generalization.

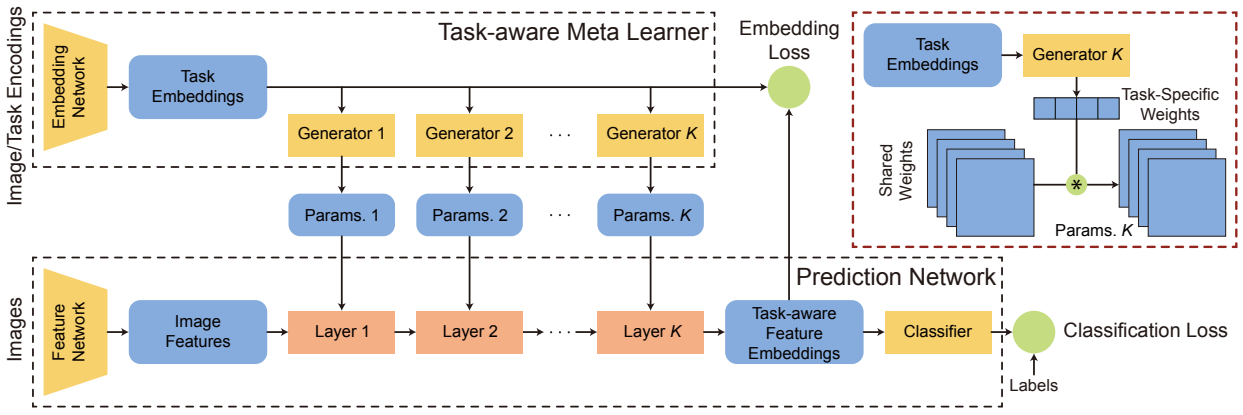


Figure 5.2: TAFENet architecture design.

We start with the TAFENet model design in Section 5.2 and then introduce the weight factorization (Section 5.2) and the embedding loss (Section 5.2) to address the challenges with the weight generation and the limited number of training tasks. We delay the specifications of different task descriptions and the setup of various applications to Section 5.2.

TAFENet Model

There are two sub-networks in TAFENet as shown in Figure 5.2: a task-aware meta leaner $\mathcal{G}$ and a prediction network $\mathcal{F}$. The task-aware meta learner takes a task description $\mathbf{t} \in \mathcal{T}$ (e.g., word2vec [133] encoding or example images, detailed in Section 5.2) and generates the weights of the feature layers in the prediction network.

For an input image $\mathbf{x} \in \mathcal{X}$, the prediction network:

$$\mathcal{F}(\mathbf{x}; \theta_t) = \mathbf{y}, \tag{5.1}$$

predicts a binary label $\mathbf{y} \in \mathcal{Y}$ indicating whether or not the input image x is compatible with the task description t. More specifically, we adopt a pre-trained feature extractor on ImageNet (e.g., ResNet [73], VGG [165] whose parameters are frozen during training) to produce generic features of the input images and then feed the generic features to a sequence of *dynamic* feature layers whose parameters denoted by θ_t are generated by $\mathcal{G}(\mathbf{t})$. The output of the dynamic feature layers is named as *task-aware feature embedding* (TAFE) in the sense that the feature embedding of the same image can be different under different task descriptions. Though not directly used as the input to $\mathcal{F}$, the task description $\mathbf{t}$ controls the parameters of the feature layers in $\mathcal{F}$ and further injects the task information to the image feature embeddings.

We are now able to introduce a simple binary classifier in $\mathcal{F}$, which takes TAFEs as inputs, to learn a task-independent decision boundary. In the case where multi-class predictions are needed, we can leverage the predictions of $\mathcal{F}(\mathbf{x})$ under different tasks descriptions and use them as probability scores. The objective formulation is presented in Section 5.2.

The task-aware meta learner $\mathcal{G}$ paramterized by η is composed of an embedding network $\mathcal{T}(\mathbf{t})$ to generate a task embedding $\mathbf{e}_t$ and a set of weight generators $\mathbf{g}^i, i = \{1...K\}$ that generate parameters for K dynamic feature layers in $\mathcal{F}$ conditioned on the same task embedding $\mathbf{e}_t$.

Weight Generation via Factorization

We now present the weight generation scheme for the feature layers in $\mathcal{F}$. The feature layers that produce the task aware feature embeddings (TAFE) can either be convolutional layers or fully-connected (FC) layers. To generate the full weight of the feature layer, we will need the output dimension of $\mathbf{g}^i$ (usually a FC layer) matches the weight size of the i-th feature layer in $\mathcal{F}$. As noted by Bertinetto *et al.* [10], the number of weights that must be estimated by the meta-learner is often much larger than the task descriptions and can therefore be difficult to learn from a small number of example tasks. Moreover, the parametrization of the parameter generators $\mathbf{g}$ can consume large amount of memory and make the training costly and even infeasible.

To ensure meta learner generalizes effectively, we propose a weight factorization scheme along the output dimension of each FC layer and the output channel dimension of a convolutional layer. This is distinct form the low-rank decomposition used in prior meta-learning works [10]. The channel-wise factorization builds on the intuition that channels of a convolutional layer may have different or even orthogonal functionality.

Weight factorization for convolutions. Given an input tensor $\mathbf{x}^i \in \mathbb{R}^{w \times h \times c_{\text{in}}}$ for the i-th feature layer in $\mathcal{F}$ whose weight is $\mathbf{W}^i \in \mathbb{R}^{k \times k \times c_{\text{in}} \times c_{\text{out}}}$ (k is the filter support size and c_{in} and c_{out} are the number of input and output channels) and bias is $\mathbf{b}^i \in \mathbb{R}^{c_{\text{out}}}$, the output $\mathbf{x}^{i+1} \in \mathbb{R}^{w' \times h' \times c_{\text{out}}}$ of the convolutional layer is given by

$$\mathbf{x}^{i+1} = \mathbf{W}^i * \mathbf{x}^i + \mathbf{b}^i, \tag{5.2}$$

where $*$ denotes convolution. Without loss of generality, we remove the bias term of the convolutional layer as it is often followed by the batch normalization [87]. $\mathbf{W}^i = \mathbf{g}^i(\mathbf{t})$ is the output of the i-th weight generator in $\mathcal{G}$ in the full weight generation setting. We now decompose the weight $\mathbf{W}^i$ into

$$\mathbf{W}^i = \mathbf{W}^i_s *_{c_{\text{out}}} \mathbf{W}^i_t, \tag{5.3}$$

where $\mathbf{W}^i_s \in \mathbb{R}^{k \times k \times c_{\text{in}} \times c_{\text{out}}}$ is a shared parameter aggregating all tasks $\{\mathbf{t}_1, ...\mathbf{t}_N\}$ and $\mathbf{W}_t \in \mathbb{R}^{1 \times 1 \times c_{\text{out}}}$ is a task-specific parameter depending on the current task input. The parameter generator $\mathbf{g}^i$ only needs to generate $\mathbf{W}^i_t$ which reduces the output dimension of $\mathbf{g}^i$ from $k \times k \times c_{\text{in}} \times c_{\text{out}}$ to c_{out}. $*_{c_{\text{out}}}$ denotes the grouped convolution along the output channel dimension, i.e. each channel of $x *_{c_{\text{out}}} y$ is simply the convolution of the corresponding channels in x and y.

Weight factorization for FCs. Similar to the factorization of the convolution weights, the FC layer weights $\mathbf{W}^i \in \mathbb{R}^{m \times n}$ can be decomposed into

$$\mathbf{W}^i = \mathbf{W}^i_s \cdot \text{diag}(\mathbf{W}^i_t), \tag{5.4}$$

where $\mathbf{W}^i_s \in \mathbb{R}^{m \times n}$ is the shared parameters for all tasks and $\mathbf{W}^i_t \in \mathbb{R}^n$ is the task-specific parameter. Note that this factorization is equivalent to the feature activation modulation, that is, for an input

$\mathbf{x} \in \mathbb{R}^{1\times m}$,

$$\mathbf{x} \cdot (\mathbf{W}_s^i \cdot \text{diag}(\mathbf{W}_t^i)) = (\mathbf{x} \cdot \mathbf{W}_s^i) \odot \mathbf{W}_t^i, \tag{5.5}$$

where $\odot$ denotes element-wise multiplication.

With such factorization, the weight generators only need to generate the task-specific parameters for each task in lower dimension and learn one set of parameters in high dimension shared across all tasks.

Embedding Loss for Meta Learner

The number of task descriptions used for training the task-aware meta learner is usually much smaller than the number of images available for training the prediction network. The data scarcity issue may lead to a corrupted meta learner. We, therefore, propose to add a secondary *embedding loss* $\mathcal{L}_{\text{emb}}$ for the meta learner alongside the classification loss $\mathcal{L}_{\text{cls}}$ used for the prediction network. Recall that we adopt a shared binary classifier in $\mathcal{F}$ to predict the compatibility of the task description and the input image. To be able to distinguish which task (i.e., class) the image belong to, instead of using a binary cross-entropy loss directly, we adopt a calibrated multi-class cross-entropy loss [204] defined as

$$\mathcal{L}_{\text{cls}} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\log\left[\frac{\exp(\mathcal{F}(\mathbf{x}_i;\theta_t))\cdot y_t^i}{\sum_{j=1}^{T}\exp(\mathcal{F}(\mathbf{x}_i;\theta_j))}\right], \tag{5.6}$$

where x_i is the i-th sample in the dataset with size N and $\mathbf{y}_i \in \{0,1\}^T$ is the one-hot encoding of the ground-truth labels. T is the number of tasks either in the whole dataset or in the minibatch during training.

For the embedding loss, the idea is to project the latent task embedding $\mathcal{T}(\mathbf{t})$ into a joint embedding space with the task-aware feature embedding (TAFE). We adopt a metric learning approach that for positive inputs of a given task, the corresponding TAFE is closer to the task embedding $\mathbf{e_t}$ while for negative inputs, the corresponding TAFE is far from the task embedding as illustrated in Figure 5.1. We use hinged cosine similarity as the distance measurement (i.e. $\phi(p,q) = \max(\texttt{cosine\_sim}(p,q), 0)$) and the embedding loss is defined as

$$\mathcal{L}_{\text{emb}} = \frac{1}{NT}\sum_{i}^{N}\sum_{t}^{T}||\phi(\text{TAFE}(\mathbf{x}_i;\theta_t), \mathbf{e}_t) - y_t^i||_2^2. \tag{5.7}$$

We find in experiments this additional supervision helps training the meta learner especially under the case where the number of training tasks is extremely limited. So far, we can define the overall objective as

$$\min_{\theta,\eta}\mathcal{L} = \min_{\theta,\eta}\mathcal{L}_{\text{cls}} + \beta\cdot\mathcal{L}_{\text{emb}}, \tag{5.8}$$

where β is the hyper-parameter to balance the two terms. We use β as 0.1 in our experiments if not specified.

Applications

We now describe how TAFENet design can be utilized in various applications (e.g., zero-shot learning, unseen attribute-object recognition and few shot learning) and specify the task descriptions adopted in this work.

Zero-shot learning. In the zero-shot learning (ZSL) setting, the set of classes seen during training and evaluated during testing are disjoint [107, 3]. Specifically, let the training set be $\mathcal{D}_s = \{(x,t,y)|x \in \mathcal{X}, t \in \mathcal{T}, y \in \mathcal{Y}\}$, and the testing set be $\mathcal{D}_u = \{(x,u,z)|x \in \mathcal{X}, u \in \mathcal{U}, z \in \mathcal{Z}\}$, where $\mathcal{T} \cap \mathcal{U} = \phi$, $|T| = |Y|$ and $|U| = |Z|$. In benchmark datasets (e.g., CUB [193], AWA [108]), each image category is associate with an attribute vector, which can be used as the task description in our work. The goal is to learn a classifier $f_{\text{zsl}} : \mathcal{X} \rightarrow \mathcal{Z}$. More recently, the generalized zero-shot learning (GZSL) setting proposed by Xian *et al.* [199] is more realistic compared to ZSL which involves classifying test examples from both seen and unseen classes, with no prior distinction between them. The classifier in GZSL maps $\mathcal{X}$ to $\mathcal{Y} \cup \mathcal{Z}$. We consider both settings in our work.

Unseen attribute-object pair recognition. Motivated by the human capability to compose and recognize novel visual concepts, Misra *et al.* [134] recently propose a new recognition task to predict unseen compositions of a given set of attributes (e.g., red, modern, ancient, etc) and objects (e.g., banana, city, car, etc) during testing and only a subset of attribute-object pairs are seen during training. This can be viewed as a zero-shot learning problem but requires more understanding of the contextuality of the attributes. In our work, the attribute-object pairs are used as the task descriptions.

Few-shot Learning. In few-shot learning, there are one or a few examples from the novel classes and plenty of examples in the base classes [70]. The goal is to learn a classifier that can classify examples from both the novel and base classes. The sample image features from different categories can be used as the task descriptions for TAFENetplural.

5.3 Experiments

We evaluate our TAFENet on three tasks: zero-shot learning (Section 5.3), unseen attribute-object composition (Section 5.3 and few-shot learning (Section 5.3). We observe that TAFE-Net is highly effective in generalizing to new tasks or concepts and is able to exceed or match the state-of-the-art on all the tasks.

Model configurations. We first describe the network configurations. The task embedding network $\mathcal{T}$ is a three-layer FC network with the hidden unit size of 2048 except for the aPY dataset [51] where we choose $\mathcal{T}$ as a 2-layer FC network with the hidden size of 2048 to avoid overfitting. The weight generator $\mathbf{g}^i$ is a single FC layer with the output dimension same as the output dimension of the corresponding feature layer in $\mathcal{F}$. For the prediction networ $\mathcal{F}$, the TAFE is generated through a 3-layer FC network with the hidden size of 2048 with input image features extracted from different pre-trained backbones (e.g., ResNet-18, ResNet-50, ResNet-101, VGG-16, VGG-19, etc.)

Table 5.1: Datasets used in GZSL

Dataset	SUN	CUB	AWA1	AWA2	aPY
No. of Images	14,340	11,788	30,475	37,322	15,339
Attributes Dim.	102	312	85	85	64
$\mathcal{Y}$	717	200	50	50	32
$\mathcal{Y}^{seen}$	645	150	40	40	20
$\mathcal{Y}^{unseen}$	72	50	10	10	12
Granularity	fine	fine	coarse	coarse	coarse

Zero-shot Learning

Datasets and evaluation metrics. We conduct our experiments on 5 benchmark datasets: SUN [200], CUB [194], AWA1 [108], AWA2 [199] and aPY [51], which have different numbers of categories and granularity. In particular, there are only 20 classes (i.e. tasks) available in the aPY dataset while 645 classes are available for training in the SUN dataset. The dataset statistics are shown in Table 5.1.

We consider both the generalized zero-shot learning (GZSL) and the conventional ZSL proposed by Xian *et al.* [199]. For GZSL, we follow the evaluation metrics proposed by Xian *et al.* [199] to report the average per class top-1 accuracy of both unseen acc_u and seen classes acc_s and the harmonic mean $H = 2 \times (acc_u \times acc_s)/(acc_u + acc_s)$. For the conventional ZSL, we report the average per-class top-1 accuracy of the unseen classes and adopt the new split provided by Xian *et al.* [199].

Training details. We set the batch size to 32 and use Adam [96] as the optimizer with the initial learning rate of 10^{-4} for the prediction network and weight generators, and 10^{-5} for the task embedding network. We reduce the learning rate by $10\times$ at epoch 30 and 45, and train the network for 60 epochs. For AWA1, we train the network for 10 epochs and reduce the learning rate by $10\times$ at epoch 5.

Baselines. We compare our model with two lines of prior works in our experiments. (1) Discriminative baselines which focus on mapping the images into a rich semantic embedding space. We include the recent competitive baselines: LATEM [213], ALE [3], DeViSE [55], SJE [2], SYNC [15], DEM [212] and the newly proposed RelationNet [204]. (2) Generative models that tackle the data scarcity problem by generating synthetic images for the unseen classes using a GAN [61, 215] based approach. The generative models can combine different discriminative models as base networks [198, 192]. We conduct comparison with f-CLSWGAN [198], SE [178], SP-AEN [19] in this category. Our model falls into the discriminative model category requiring no additional synthetic data.

Quantitative results. We first report the performance of our TAFENet in Table 5.2 compared to the prior works. Overall, our model outperforms the existing approaches including the generative models on the AWA1, AWA2 and aPY datasets under ZSL setting and on the AWA1 and aPY datasets under GZSL setting. Compared to the prior arts of the discriminative models (denoted

Table 5.2: TAFENet evaluation for ZSL and GZSL.

Method	Zero-shot Learning					Generalized Zero-shot Learning														
	SUN	CUB	AWA1	AWA2	aPY	SUN			CUB			AWA1			AWA2			aPY		
	T1	T1	T1	T1	T1	u	s	H	u	s	H	u	s	H	u	s	H	u	s	H
LATEM [213]	55.3	49.3	55.1	55.8	35.2	14.7	28.8	19.5	15.2	57.3	24.0	7.3	71.7	13.3	11.5	77.3	20.0	0.1	73.0	0.2
ALE [3]	58.1	54.9	59.9	62.5	39.7	21.8	33.1	**26.3**	23.7	62.8	34.4	16.8	76.1	27.5	14.0	81.8	23.9	4.6	73.7	8.7
DeViSE[55]	56.5	52	54.2	59.7	**39.8**	16.9	27.4	20.9	23.8	53.0	32.8	13.4	68.7	22.4	17.1	74.7	27.8	4.9	76.9	9.2
SJE [2]	53.7	53.9	65.6	61.9	32.9	14.7	80.5	19.8	23.5	59.2	33.6	11.3	74.6	19.6	8.0	73.9	14.4	3.7	55.7	6.9
ESZSL [154]	54.5	53.9	58.2	58.6	38.3	11.0	27.9	15.8	12.6	63.8	21.0	6.6	75.6	12.1	5.9	77.8	11.0	2.4	70.1	4.6
SYNC [15]	56.3	**55.6**	54.0	46.6	23.9	7.9	43.3	13.4	11.5	70.9	19.8	8.9	87.3	16.2	10.0	90.5	18.0	7.4	66.3	13.3
RelationNet [204]	-	**55.6**	68.2	64.2	-	-	-	-	38.1	61.1	**47.0**	31.4	91.3	46.7	30.0	93.4	**45.3**	-	-	-
DEM [212]	**61.9**	51.7	**68.4**	**67.1**	35.0	20.5	34.3	25.6	19.6	57.9	29.2	32.8	84.7	**47.3**	30.5	86.4	45.1	11.1	75.1	**19.4**
f-CLSWGAN† [198]	60.8	57.3	68.2	-	-	42.6	36.6	**39.4**	57.7	43.7	**49.7**	61.4	57.9	59.6	-	-	-	-	-	-
SE† [178]	**63.4**	**59.6**	69.5	69.2	-	40.9	30.5	34.9	53.3	41.5	46.7	67.8	56.3	61.5	58.3	68.1	**62.8**	-	-	-
SP-AEN† [19]	59.2	55.4	-	58.5	24.1	24.9	38.6	30.3	34.7	70.6	46.6	-	-	-	23.3	90.9	37.1	13.7	63.4	22.6
TAFENet	60.9	56.9	**70.8**	**69.3**	**42.2**	27.9	40.2	33.0	41.0	61.4	49.2	50.5	84.4	**63.2**	36.7	90.6	52.2	24.3	75.4	**36.8**

in **blue** in Table 5.2), TAFENet is able to surpass them by a large margin (e.g., roughly 16 points improvement ion AWA1 and 17 points on aPY) in GZSL. For the more challenging fine-grained SUN and CUB datasets, we are able to improve the results by 7 and 2 points. The results indicate that better embedding learning can largely help the model generalization.

Embedding loss ablation. We provide the harmonic mean of our models with and without the embedding loss under the GZSL setting on five benchmark datasets in Table 5.3. In general, models with the embedding loss have stronger performance than those without the embedding loss except for the SUN dataset whose number of categories is about $3-22\times$ larger than the other datasets. This observation matches our assumption that the additional supervision on the joint embedding better addresses the data scarcity issue (i.e. fewer class descriptions than the visual inputs) of training the controller model.

Table 5.3: Ablation of the embedding loss.

Method	SUN	CUB	AWA1	AWA2	aPY
TAFENet w/o EmbLoss	**33.1**	45.4	58.8	47.2	30.5
TAFENet	33.0	**49.2**	**63.2**	**52.2**	**36.8**

Embedding visualization. In Figure 5.3, we visualize the task-aware feature embeddings of images from the aPY dataset under different task descriptions. As we can see, image embeddings of the same image are projected into different clusters conditioned on the task descriptions.

Unseen Visual-attribute Composition

Besides the standard zero-shot learning benchmarks, we evaluate our model in the visual-attribute composition task proposed by Misra *et al.* [134]. The goal is to compose a set of visual concept primitives like attributes and objects (e.g. `large elephant`, `old building`, etc.) to

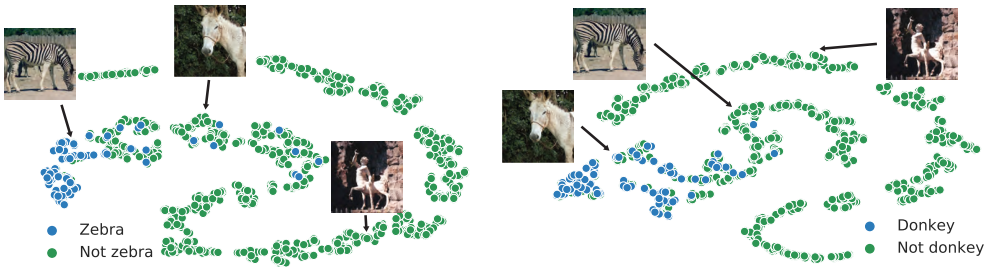


Figure 5.3: TAFE visualization.

Table 5.4: Evaluation on MITStates and StanfordVRD.

Method	MITStates AP	MITStates Top-k Accuracy 1	2	3	StanfordVRD AP	StanfordVRD Top-k Accuracy 1	2	3
Visual Product [134]	8.8	9.8	16.1	20.6	4.9	3.2	5.6	7.6
Label Embed (LE) [134]	7.9	11.2	17.6	22.4	4.3	4.1	7.2	10.6
LEOR [134]	4.1	4.5	6.2	11.8	0.9	1.1	1.3	1.3
LE + R [134]	6.7	9.3	16.3	20.8	3.9	3.9	7.1	10.4
Red Wine [134]	10.4	13.1	21.2	27.6	5.7	6.3	9.2	12.7
TAFENet	**16.3**	**16.4**	**26.4**	**33.0**	**12.2**	**12.3**	**19.7**	**27.5**

obtain new visual concepts for a given image. We see this task as a more challenging "zero-shot" learning task which requires the model not only to predict unseen visual concept compositions but model the contextuality of the concepts.

Datasets and evaluation metrics. We conduct the experiments on two datasets: MITStates [88] (image samples in Figure 5.5) and the modified StanfordVRD [124] (image samples in Figure 5.4). The setup is the same as Misra *et al.* [134]. Each image in the MITStates dataset is assigned a pair of (attribute, object) as its label. The model is trained on 34K images with 1,292 label pairs and tested on 19K images with 700 unseen pairs. The second dataset is constructed based on the bounding boxes annotations of the StanfordVRD dataset. Each sample has an SPO (subject, predicate, object) tuple as the ground truth label. The dataset has 7,701 SPO tuples and 1,029 of them are seen only in the test split. We evaluate our models only on examples with unseen labels. We extract the image features with pre-trained models on ImageNet. We use ResNet-101 [73] as our main feature extractor and also test features extracted with ResNet-18 [73], VGG-16/19 [165] for ablation. For the task descriptions, we concatenate the word embeddings of the attributes and objects with word2vec [133] trained with GoogleNews. We also consider one-hot encoding for the task id in the ablation.

For evaluation metrics, we report the mean Average Precision (mAP) of images with unseen labels in the test set together with the top-k accuracy where $k = 1, 2, 3$. We follow the same training schedule as that used in the zero shot learning experiments.

Quantitative results. We compare our model with several baselines provided by Misra *et al.* [134] and summarize the results in Table 5.4 on both the MITStates and StanfordVRD datasets. Our

Figure 5.4: Samples in StandfordVRD.

Table 5.5: Ablation study with different task encoding and base network features.

Task Encoding	Features	AP	Top-k Accuracy		
			1	2	3
Word2vec	ResNet-101	16.2	17.2	27.8	35.7
Onehot	ResNet-101	16.1	16.1	26.8	33.8
Word2vec	VGG16	16.3	16.4	26.4	33.0
Onehot	VGG16	16.3	16.4	25.9	32.5
Word2vec	VGG19	15.6	16.2	26.0	32.4
Onehot	VGG19	16.3	16.4	26.0	33.1

model surpasses the state-of-the-art models with an improvement of more than 6 points in mAP and 4 to 15 points in top-k accuracy. Nagarajan and Grauman [137] recently propose an embedding learning framework for visual-attribute composition. They report the top-1 accuracy of 12.0% on the MITStates dataset with ResNet-18 features. For fair comparison, we use the same ResNet-18 features and obtain the top-1 accuracy of 15.1%.

Ablation on the feature extractor and task specification. We consider different feature extractors (ResNet-101, VGG-16 and 19) and task encodings (word2vec and one-hot encoding) for ablation and summarize the results in Table 5.5. The average precision difference between different feature

Figure 5.5: Top retrievals on the unseen pairs of MITStates.

extractors are very minimal (within 0.1%) and the largest gap in Top-3 accuracy is within 2%. This indicates that TAFENet is robust in transforming the generic features into task-aware feature embeddings. For the task encoding, the one-hot encoding is comparable to the word2vec encoding and even stronger when using VGG-19 features. This shows that the task transformer network $\mathcal{T}$ is very expressive to extract rich semantic information simply from the task ids.

Visualization. In Figure 5.5, we show the top retrievals of unseen attribute-object pairs from the MITStates dataset. Our model can learn to compose new concepts from the existing attributes and objects while respecting their context.

Few-shot Image Classification

Our model naturally fits the few-shot learning setting where one or few images of a certain category are used as the task descriptions. Unlike prior work on meta-learning which experiments with few classes and low resolution images [180, 167, 53], we evaluate our model on the challenging benchmark proposed by Hariharan and Girshick [70]. The benchmark is based on the ImageNet images and contains hundreds of classes that are divided into base classes and novel classes. At inference time, the model is provided with one or a few examples from the novel classes and hundreds of examples from the base classes. The goal is to obtain high accuracy on the novel classes without sacrificing the performance on the base classes.

Baselines. In our experiments, the baselines we consider are the state-of-the-art meta learning models Matching Network (MN) [180] and Prototypical Network (PN) [167]. We also compare the logistic regression (LogReg) baseline provided by Hariharan and Girshick [70]. Another line of research [192, 70] for few-shot learning is to combine the meta-learner with a "hallucinator" to generate additional training data. We regard these works as complementary approaches to our meta-learning model.

Experiment details. We follow the prior works [70, 192] to run five trials for each setting of n (the

Table 5.6: Few-shot ImageNet Classification on ImageNet.

Method	Novel Top-5 Acc		All Top-5 Acc	
	n=1	n=2	n=1	n=2
LogReg [70]	38.4	51.1	40.8	49.9
PN [167]	39.3	54.4	49.5	61.0
MN [180]	43.6	54.0	54.4	61.0
TAFENet	43.0	53.9	**55.7**	**61.9**
LogReg w/ Analogies [70]	40.7	50.8	52.2	59.4
PN w/ G [192]	45.0	55.9	56.9	63.2

number of examples per novel class, n = 1 and 2 in our experiments) on the five different data splits and report the average top-5 accuracy of both the novel and all classes. We use the features trained with ResNet-10 using SGM loss provided by Hariharan and Girshick [70] as inputs. For training, we sample 100 classes in each iteration and use SGD with momentum of 0.9 as the optimizer. The initial learning rate is set to 0.1 except for the task embedding network (set to 0.01) and the learning rate is reduced by $10\times$ every 8k iterations. The model is trained for 30k iterations in total. Other hyper-paramters are set to the same as Hariharan and Girshick [70] if not mentioned.

Quantitative results. As shown in Table 5.6, our model is on par with the state-of-the-art meta learning models in the novel classes while outperforming them in all categories. Attaching a "hallucinator" to the meta learning model improves performance in general. Our model can be easily attached with a hallucinator and we leave the detailed study as future work due to the time constraint.

5.4 Related Work

Our work is related to several lines of research in zero-shot learning as well as parameter generation, dynamic neural network designs and feature modulation. Built on top of the rich prior works, to the best of our knowledge, we are the first to study dynamic image feature representation for zero-shot and few-shot learning.

Zero-shot learning falls into the multimodal learning regime which requires a proper leverage of multiple sources (e.g., image features and semantic embeddings of the tasks). Many [98, 204, 180, 213, 15, 55] have studied metric learning based objectives to jointly learn the task embeddings and image embeddings, resulting in a similarity or compatibility score that can later be used for classification [134, 180, 107, 3, 2, 55, 168]. Conceptually, our approach shares the *matching* spirit with the introduction of a binary classifier which predicts whether or not the input image matches the task description. In contrast to prior works, we transform the image features according to the task and thus we only need to learn a task-independent decision boundary to separate the positive and negative examples similar to the classic supervised learning. The proposed embedding loss in our work also adopts metric learning for joint embedding learning but with the main goal to address the

limited number of training tasks in meta learning [54]. More recently, data hallucination has been used in the zero-shot [198, 218] and few-shot [70, 192] learning which indicate that the additional synthetic data of the unseen tasks are useful to learn the classifier and can be augmented with the discriminative models [198, 192]. Our (discriminative) model does not utilize additional data points and we show in experiments that our model can outperform or match the generative models on a wide range of benchmarks. We believe the approaches requiring additional data generation can benefit from a stronger base discriminative model.

TAFENet uses a task-aware meta learner to generate parameters of the feature layers. Several efforts [10, 67, 34] have studied the idea of adopting one meta network to generate weights of another network. Our task-aware meta learner serves a similar role for the weight generation but in a more structured and constrained manner. We study different mechanisms to decompose the weights of the prediction network so that it can generate weights for multiple layers at once where Bertinetton *et al.* [10] focuses on generating weights for a single layer and Denil *et al.* [34] can generate up to 95% parameters of a single layer due to the quadratic size of the output space.

The TAFENet design is also related to works on dynamic neural networks [188, 197, 186, 115] which focus on dynamic execution at runtime. SkipNet [188] proposed by Wang *et al.*introduces a recurrent gate to dynamically control the activation of the network based on the input. In contrast, TAFENet dynamically re-configures the network parameters rather than the network structure as in the prior works [188, 197] aiming to learn adaptive image features for the given task.

In the domain of visual question answering, previous works [142, 32] explore the use of question embedding network to modulate the features of the primary convolutional network. Our factorized weight generation scheme for convolutional layers can also be viewed as channel-wise feature modulation. However, the proposed parameter generation framework is more general than feature modulation which can host different factorization strategies [10].

5.5 Chapter Summary

In this work, we explored a meta learning based approach to generate task aware feature embeddings for settings with little or no training data. We proposed TAFE-Net, a network that generates task aware feature embeddings (TAFE) conditioned on the given task descriptions. TAFENet is composed of a task-aware meta learner that generates weights for the feature embedding layers in a standard prediction network. To address the challenges in training the meta learner, we introduced two key innovations: (1) adding an additional embedding loss to improve the generalization of the meta learner; (2) a novel weight factorization scheme to generate parameters of the prediction network more effectively. We demonstrated the general applicability of the proposed network design on a range of benchmarks in zero/few shot learning, and matched or exceeded the state-of-the-art.

Chapter 6

Few-Shot Feature Re-weighting

6.1 Overview

The recent success of deep convolutional neural networks (CNNs) in object detection [153, 59, 150, 151] relies heavily on a huge amount of training data with accurate bounding box annotations. When the labeled data are scarce, CNNs can severely overfit and fail to generalize. In contrast, humans exhibit strong performance in such tasks: children can learn to detect a novel object quickly from very few given examples. Such ability of learning to detect from few examples is also desired for computer vision systems, since some object categories naturally have scarce examples or their annotations are hard to obtain, e.g., California firetrucks, endangered animals or certain medical data [161].

In this work, we target at the challenging *few-shot object detection* problem, as shown in Fig. 6.1. Specifically, given some base classes with sufficient examples and some novel classes with only a few samples, we aim to obtain a model that can detect both base and novel objects at test time. Obtaining such a few-shot detection model would be useful for many applications. Yet, effective methods are still absent. Recently, meta learning [180, 167, 53] offers promising solutions to a similar problem, i.e., few-shot classification. However, object detection is by nature much more difficult as it involves not only class predictions but also localization of the objects, thus off-the-shelf few-shot classification methods cannot be directly applied on the few-shot detection problem. Taking Matching Networks [180] and Prototypical Networks [167] as examples, it is unclear how to build object prototypes for matching and localization, because there may be distracting objects of irrelevant classes within the image or no targeted objects at all.

We propose a novel detection model that offers few-shot learning ability through fully exploiting detection training data from some base classes and quickly adapting the detection prediction network to predict novel classes according to a few support examples. The proposed model first learns meta features from base classes that are generalizable to the detection of different object classes. Then it effectively utilizes a few support examples to identify the meta features that are important and discriminative for detecting novel classes, and adapts accordingly to transfer detection knowledge from the base classes to the novel ones.

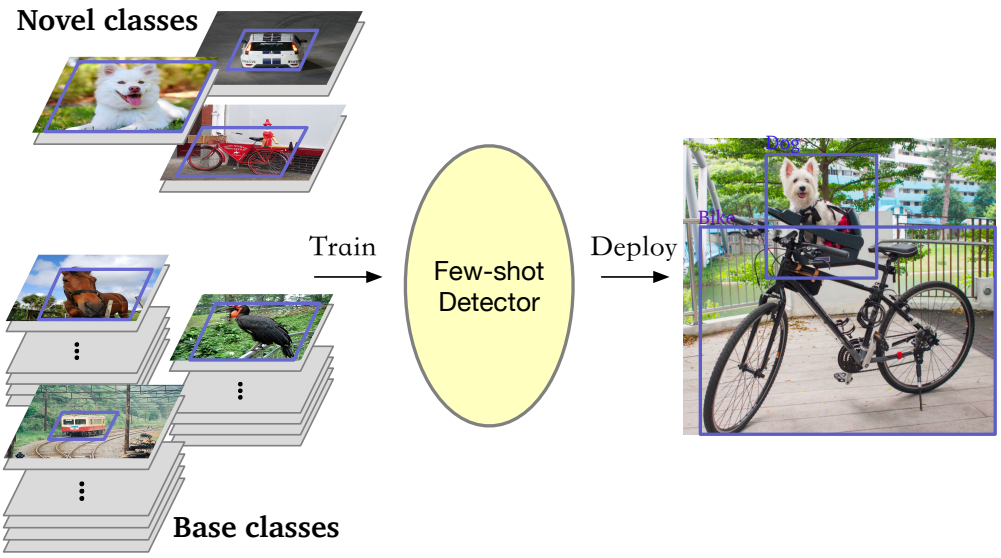


Figure 6.1: Few-shot object detector illustration.

Our proposed model thus introduces a novel detection framework containing two modules, i.e., a meta feature learner and a light-weight feature reweighting module. Given a query image and a few support images for novel classes, the feature learner extracts meta features from the query image. The reweighting module learns to capture global features of the support images and embeds them into reweighting coefficients to modulate the query image meta features. As such, the query meta features effectively receive the support information and are adapted to be suitable for novel object detection. Then the adapted meta features are fed into a detection prediction module to predict classes and bounding boxes for novel objects in the query (Fig. 6.2). In particular, if there are N novel classes to detect, the reweighting module would take in N classes of support examples and transform them into N reweighting vectors, each responsible for detecting novel objects from the corresponding class. With such class-specific reweighting vectors, some important and discriminative meta features for a novel class would be identified and contribute more to the detection decision, and the whole detection framework can learn to detect novel classes efficiently.

The meta feature learner and the reweighting module are trained together with the detection prediction module end-to-end. To ensure few-shot generalization ability, the whole few-shot detection model is trained using an two-phase learning scheme: first learn meta features and good reweighting module from base classes; then fine-tune the detection model to adapt to novel classes. For handling difficulties in detection learning (e.g., existence of distracting objects), it introduces a carefully designed loss function.

Our proposed few-shot detector outperforms competitive baseline methods on multiple datasets and in various settings. Besides, it also demonstrates good transferability from one dataset to another different one. Our contributions can be summarized as follows:

- We are among the first to study the problem of few-shot object detection, which is of great practical values but a less explored task than image classification in the few-shot learning

literature.

- We design a novel few-shot detection model that 1) learns generalizable meta features; and 2) automatically reweights the features for novel class detection by producing class-specific activating coefficients from a few support samples.
- We experimentally show that our model outperforms baseline methods by a large margin, especially when the number of labels is extremely low. Our model adapts to novel classes significantly faster.

6.2 Few-Shot Feature Re-weighting

In this work, we define a novel and realistic setting for few-shot object detection, in which there are two kinds of data available for training, i.e., the *base classes* and the *novel classes*. For the base classes, abundant annotated data are available, while only a few labeled samples are given to the novel classes [70]. We aim to obtain a few-shot detection model that can learn to detect novel object when there are both base and novel classes in testing by leveraging knowledge from the base classes.

This setting is worth exploring since it aligns well with a practical situation—one may expect to deploy a pre-trained detector for new classes with only a few labeled samples. More specifically, large-scale object detection datasets (e.g., PSACAL VOC, MSCOCO) are available to pre-train a detection model. However, the number of object categories therein is quite limited, especially compared to the vast object categories in real world. Thus, solving this few-shot object detection problem is heavily desired.

Feature Reweighting for Detection

Our proposed few-shot detection model introduces a meta feature learner $\mathcal{D}$ and a reweighting module $\mathcal{M}$ into a one-stage detection framework. In this work, we adopt the proposal-free detection framework YOLOv2 [150]. It directly regresses features for each anchor to detection relevant outputs including classification score and object bounding box coordinates through a detection prediction module $\mathcal{P}$. As shown in Fig. 6.2, we adopt the backbone of YOLOv2 (i.e., DarkNet-19) to implement the meta feature extractor $\mathcal{D}$, and follow the same anchor setting as YOLOv2. As for the reweighting module $\mathcal{M}$, we carefully design it to be a light-weight CNN for both enhancing efficiency and easing its learning. Its architecture details are deferred to the supplementary due to space limit.

tab:coco

The meta feature learner $\mathcal{D}$ learns how to extract meta features for the input query images to detect their novel objects. The reweighting module $\mathcal{M}$, taking the support examples as input, learns to embed support information into reweighting vectors and adjust contribution of each meta feature of the query image accordingly for following detection prediction module $\mathcal{P}$. With the reweighting module , some meta features informative for detecting novel objects would be excited and thus assist detection prediction.

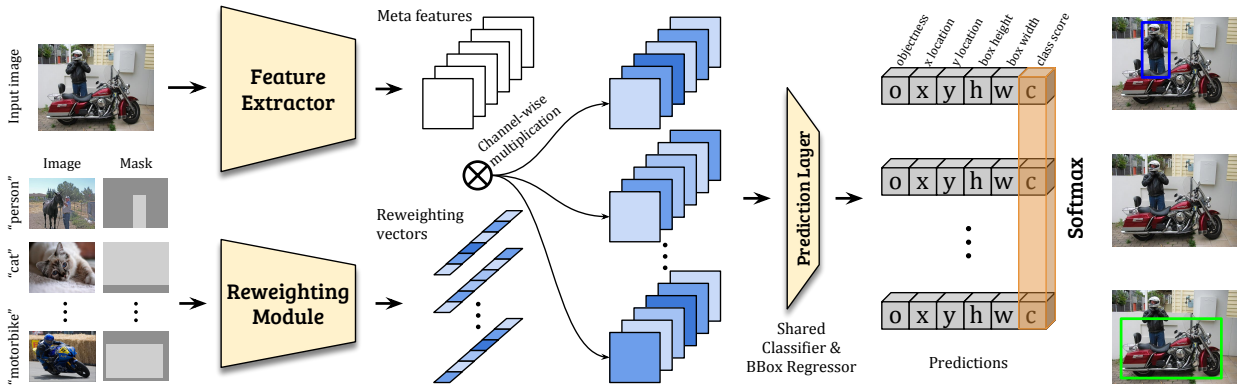


Figure 6.2: The architecture of our proposed few-shot detection model.

Formally, let I denote an input query image. Its corresponding meta features $F \in \mathbb{R}^{w \times h \times m}$ are generated by $\mathcal{D}$: $F = \mathcal{D}(I)$. The produced meta feature has m feature maps. We denote the support images and their associated bounding box annotation, indicating the target class to detect, as I_i and M_i respectively, for class $i, i = 1, \ldots, N$. The reweighting module $\mathcal{M}$ takes one support image (I_i, M_i) as input and embed it into a class-specific representation $w_i \in \mathbb{R}^m$ with $w_i = \mathcal{M}(I_i, M_i)$. Such embedding captures global representation of the target object w.r.t. the m meta features. It will be responsible for reweighting the meta features and highlighting more important and relevant ones to detect the target object from class i. More specifically, after obtaining the class-specific reweighting coefficients w_i, our model applies it to obtain the class-specific feature F_i for novel class i by:

$$F_i = F \otimes w_i, \quad i = 1, \ldots, N, \tag{6.1}$$

where $\otimes$ denotes channel-wise multiplication. We implement it through 1×1 depth-wise convolution.

After acquiring class-specific features F_i, we feed them into the prediction module $\mathcal{P}$ to regress the objectness score o, bounding box location offsets (x, y, h, w), and classification score c_i for each of a set of predefined anchors:

$$\{o_i, x_i, y_i, h_i, w_i, c_i\} = \mathcal{P}(F_i), \quad i = 1, \ldots, N, \tag{6.2}$$

where c_i is one-versus-all classification score indicating the probability of the corresponding object belongs to class i.

Learning Scheme

It is not straightforward to learn a good meta feature learner $\mathcal{D}$ and reweighting module $\mathcal{M}$ from the base classes such that they can produce generalizable meta features and rweighting coefficients. To ensure the model generalization performance from few examples, we develop a new two-phase learning scheme that is different from the conventional ones for detection model training.

We reorganize the training images with annotations from the base classes into multiple few-shot detection learning tasks $\mathcal{T}_j$. Each task $\mathcal{T}_j = \mathcal{S}_j \cup \mathcal{Q}_j = \{(I_1^j, M_1^j), \ldots, (I_N^j, M_N^j)\} \cup \{(I_j^q, M_j^q)\}$ contains a support set S_j (consisting of N support images each of which is from a different base class) and a query set $\mathcal{Q}_j$ (offering query images with annotations for performance evaluation).

Let θ_D, θ_M and θ_P denote the parameters of meta feature learner $\mathcal{D}$, the reweighting module $\mathcal{M}$ and prediction module $\mathcal{P}$ respectively. We optimize them jointly through minimizing the following loss:

$$\begin{aligned}\min_{\theta_D,\theta_M,\theta_P} \mathcal{L} &:= \sum_j \mathcal{L}(\mathcal{T}_j) \\ &= \sum_j \mathcal{L}_{\text{det}}(\mathcal{P}_{\theta_P}(\mathcal{D}_{\theta_D}(I_q^j) \otimes \mathcal{M}_{\theta_M}(\mathcal{S}_j)), M_j^q).\end{aligned}$$

Here $\mathcal{L}_{\text{det}}$ is the detection loss function and we explain its details later. The above optimization ensures the model to learn good meta features for the query and reweighting coefficients for the support.

The overall learning procedure consists of two phases. The first phase is the *base training* phase. In this phase, despite abundant labels are available for each base class, we still jointly train the feature learner, detection prediction together with the reweighting module . This is to make them coordinate in a desired way: the model needs to learn to detect objects of interest by referring to a good reweighting vector. The second phase is *few-shot fine-tuning*. In this phase, we train the model on both base and novel classes. As only k labeled bounding boxes are available for the novel classes, to balance between samples from the base and novel classes, we also include k boxes for each base class. The training procedure is the same as the first phase, except that it takes significantly fewer iterations for the model to converge.

In both training phases, the reweighting coefficients depend on the input pairs of (support image, bounding box) that are randomly sampled from the available data per iteration. After few-shot fine-tuning, we would like to obtain a detection model that can directly perform detection without requiring any support input. This is achieved by setting the reweighting vector for a target class to the average one predicted by the model after taking the k-shot samples as input. After this, the reweighting module can be completely removed during inference. Therefore, our model adds negligible extra model parameters to the original detector

Detection loss function. To train the few-shot detection model, we need to carefully choose the loss functions in particular for the class prediction branch, as the sample number is very few. Given that the predictions are made class-wisely, it seems natural to use binary cross-entropy loss, regressing 1 if the object is the target class and 0 otherwise. However, we found using this loss function gave a model prone to outputting redundant detection results (e.g., detecting a train as a bus and a car). This is due to that for a specific region of interest, only one out of N classes is truly positive. However, the binary loss strives to produce balanced positive and negative predictions. Non-maximum suppression could not help remove such false positives as it only operates on predictions within each class.

To resolve this issue, our proposed model adopts a softmax layer for calibrating the classification scores among different classes, and adaptively lower detection scores for the wrong classes.

Therefore, the actual classification score for the i-th class is given by $\hat{c}_i = \frac{e^{c_i}}{\sum_{j=1}^{N} e^{c_j}}$. Then to better align training procedure and few-shot detection, the cross-entropy loss over the calibrated scores $\hat{c}_i$ is adopted:

$$\mathcal{L}_c = -\sum_{i=1}^{N} \mathbb{1}(\cdot, i) \log(\hat{c}_i), \tag{6.3}$$

where $\mathbb{1}(\cdot, i)$ is an indicator function for whether current anchor box really belongs to class i or not. After introducing softmax, the summation of classification scores for a specific anchor is equal to 1, and less probable class predictions will be suppressed. This softmax loss will be shown to be superior to binary loss in the following experiments. For bounding box and objectiveness regression, we adopt the similar loss function $\mathcal{L}_{bbx}$ and $\mathcal{L}_{obj}$ as YOLOv2 [150] but we balance the positive and negative by not computing some loss from negatives samples for the objectiveness scores. Thus, the overall detection loss function is $\mathcal{L}_{\text{det}} = \mathcal{L}_c + \mathcal{L}_{bbx} + \mathcal{L}_{obj}$.

Reweighting module input. The input of the reweighting module should be the object of interest. However, in object detection task, one image may contain multiple objects from different classes. To let the reweighting module know what the target class is, in additional to three RGB channels, we include an additional "mask" channel (M_i) that has only binary values: on the position within the bounding box of an object of interest, the value is 1, otherwise it is 0 (see left-bottom of Fig. 6.2). If multiple target objects are present on the image, only one object is used. This additional mask channel gives the reweighting module the knowledge of what part of the image's information it should use, and what part should be considered as "background". Combining mask and image as input not only provides class information of the object of interest but also the location information (indicated by the mask) useful for detection. In the experiments, we also investigate other input forms.

6.3 Experiments

In this section, we evaluate our model and compare it with various baselines, to show our model can learn to detect novel objects significantly faster and more accurately. We use YOLOv2 [150] as the base detector. Due to space limit, we defer all the model architecture and implementation details to the supplementary material. The code to reproduce the results will be released at `https://github.com/bingykang/Fewshot_Detection`.

Experimental Setup

Datasets. We evaluate our model for few-shot detection on the widely-used object detection benchmarks, i.e., VOC 2007 [48], VOC 2012 [50], and MS-COCO [118]. We follow the common practice [150, 153, 162, 29] and use VOC 07 test set for testing while use VOC 07 and 12 train/val sets for training. Out of its 20 object categories, we randomly select 5 classes as the novel ones, while keep the remaining 15 ones as the base. We evaluate with 3 different base/novel splits. During base training, only annotations of the base classes are given. For few-shot fine-tuning, we use a

very small set of training images to ensure that each class of objects only has k annotated bounding boxes, where k equals 1, 2, 3, 5 and 10. Similarly, on the MS-COCO dataset, we use 5000 images from the validation set for evaluation, and the rest images in train/val sets for training. Out of its 80 object classes, we select 20 classes overlapped with VOC as novel classes, and the remaining 60 classes as the base classes. We also consider learning the model on the 60 base classes from COCO and applying it to detect the 20 novel objects in PASCAL VOC. This setting features a cross-dataset learning problem that we denote as *COCO to PASCAL*.

Note the testing images may contain distracting base classes (which are not targeted classes to detect) and some images do not contain objects of the targeted novel class. This makes the few-shot detection further challenging.

Table 6.1: Few-shot detection performance (mAP) on PASCAL VOC.

	Novel Set 1					Novel Set 2					Novel Set 3				
Method / Shot	1	2	3	5	10	1	2	3	5	10	1	2	3	5	10
\	0.0	0.0	1.8	1.8	1.8	0.0	0.1	0.0	1.8	0.0	1.8	1.8	1.8	3.6	3.9
YOLO-ft	3.2	6.5	6.4	7.5	12.3	8.2	3.8	3.5	3.5	7.8	8.1	7.4	7.6	9.5	10.5
YOLO-ft-full	6.6	10.7	12.5	24.8	38.6	12.5	4.2	11.6	16.1	33.9	13.0	15.9	15.0	32.2	38.4
LSTD(YOLO)	6.9	9.2	7.4	12.2	11.6	9.9	5.4	3.3	5.7	19.2	10.9	7.6	9.5	15.3	16.9
LSTD(YOLO)-full	8.2	11.0	12.4	29.1	38.5	11.4	3.8	5.0	15.7	31.0	12.6	8.5	15.0	27.3	36.3
Ours	**14.8**	**15.5**	**26.7**	**33.9**	**47.2**	**15.7**	**15.3**	**22.7**	**30.1**	**40.5**	**21.3**	**25.6**	**28.4**	**42.8**	**45.9**

Table 6.2: Few-shot detection performance on COCO.

		Average Precision						Average Recall					
# Shots		0.5:0.95	0.5	0.75	S	M	L	1	10	100	S	M	L
10	YOLO-ft	0.4	1.1	0.1	0.3	0.7	0.6	5.8	8.0	8.0	0.6	5.1	15.5
	YOLO-ft-full	3.1	7.9	1.7	0.7	2.0	6.3	7.8	10.5	10.5	1.1	5.5	20
	LSTD(YOLO)	0.4	1.1	0.2	0.2	0.7	0.6	5.8	7.9	7.9	0.6	5.0	15.3
	LSTD(YOLO)-full	3.2	8.1	2.1	**0.9**	2.0	6.5	7.8	10.4	10.4	1.1	5.6	19.6
	Ours	**5.6**	**12.3**	**4.6**	**0.9**	**3.5**	**10.5**	**10.1**	**14.3**	**14.4**	**1.5**	**8.4**	**28.2**
30	YOLO-ft	0.6	1.5	0.3	0.2	0.7	1.0	7.4	9.4	9.4	0.4	3.9	19.3
	YOLO-ft-full	7.7	16.7	6.4	0.4	3.3	14.4	11.7	15.3	15.3	1.0	7.7	29.2
	LSTD(YOLO)	0.6	1.4	0.3	0.2	0.8	1.0	7.1	9.1	9.2	0.4	3.9	18.7
	LSTD(YOLO)-full	6.7	15.8	5.1	0.4	2.9	12.3	10.9	14.3	14.3	0.9	7.1	27.0
	Ours	**9.1**	**19.0**	**7.6**	**0.8**	**4.9**	**16.8**	**13.2**	**17.7**	**17.8**	**1.5**	**10.4**	**33.5**

Baselines. We compare our model with five competitive baselines. Three of them are built upon the vanilla YOLOv2 detector with straightforward few-shot learning strategies. The first one is to train the detector on images from the base and novel classes together. In this way, it can learn good features from the base classes that are applicable for detecting novel classes. We term this baseline as *YOLO-joint*. We train this baseline model with the same total iterations as ours. The other two

YOLO-based baselines also use two training phases as ours. In particular, they train the original YOLOv2 model with the same base training phase as ours; for the few-shot fine-tuning phase, one fine-tunes the model with the same iterations as ours, giving the *YOLO-ft* baseline; and one trains the model to fully converge, giving *YOLO-ft-full*. Comparing with these baselines can help understand the few-shot learning advantage of our models brought by the proposed feature reweighting method. The last two baselines are from a recent few-shot detection method, i.e., Low-Shot Transfer Detector (LSTD) [16]. LSTD relies on background depression (BD) and transfer knowledge (TK) to obtain a few-shot detection model on the novel classes. For fair comparison, we re-implement BD and TK based on YOLOV2, train it for the same iterations as ours, obtaining *LSTD(YOLO)*; and train it to convergence to obtain the last baseline, *LSTD(YOLO)-full*.

Comparison with Baselines

PASCAL VOC. We present our main results on novel classes in Table 6.1. First we note that our model significantly outperforms the baselines, especially when the labels are extremely scarce (1-3 shot). The improvements are also consistent for different base/novel class splits and number of shots. In contrast, LSTD(YOLO) can boost performance in some cases, but might harm the detection in other cases. Take 5-shot detection as an example, LSTD(YOLO)-full brings 4.3 mAP improvement compared to YOLO-ft-full on novel set 1, but it is worse than YOLO-ft-full by 5.1 mAP on novel set 2. Second, we note that YOLO-ft/YOLO-ft-full also performs significantly better than \. This demonstrates the necessity of the two training phases employed in our model: it is better to first train a good knowledge representation on base classes and then fine-tune with few-shot data, otherwise joint training with let the detector bias towards base classes and learn nearly nothing about novel classes. More detailed results about each class is available at supplementary material.

COCO. The results for COCO dataset is shown in Table 6.2. We evaluate for $k = 10$ and $k = 30$ shots per class. In both cases, our model outperforms all the baselines. In particular, when the YOLO baseline is trained with same iterations with our model, it achieves an AP of less than 1%. We also observe that there is much room to improve the results obtained in the few-shot scenario. This is possibly due to the complexity and large amount of data in COCO so that few-shot detection over it is quite challenging.

COCO to PASCAL. We evaluate our model using 10-shot image per class from PASCAL. The mAP of YOLO-ft, YOLO-ft-full, LSTD(YOLO), LSTD(YOLO)-full are 11.24%, 28.29%, 10.99% 28.95% respectively, while our method achieves 32.29%. The performance on PASCAL novel classes is worse than that when we use base classes in PASCAL dataset (which has mAP around 40%). This might be explained by the different numbers of novel classes, i.e., 20 v.s. 5.

Performance Analysis

Learning speed. Here we analyze learning speed of our models. The results show that despite the fact that our few-shot detection model does not consider adaptation speed explicitly in the optimization process, it still exhibits surprisingly fast adaptation ability. Note that in experiments of

Table 6.1, YOLO-ft-full and LSTD(YOLO)-full requires 25,000 iterations for it to fully converge, while our model only require 1200 iterations to converge to a higher accuracy. When the baseline YOLO-ft and LSTD(YOLO) are trained for the same iterations as ours, their performance is far worse. In this section, we compare the full convergence behavior of \, YOLO-ft-full and our method in Fig. 6.3. The AP value are normalized by the maximum value during the training of our method and the baseline together. This experiment is conducted on PASCAL VOC base/novel split 1, with 10-shot bounding box labels on novel classes.

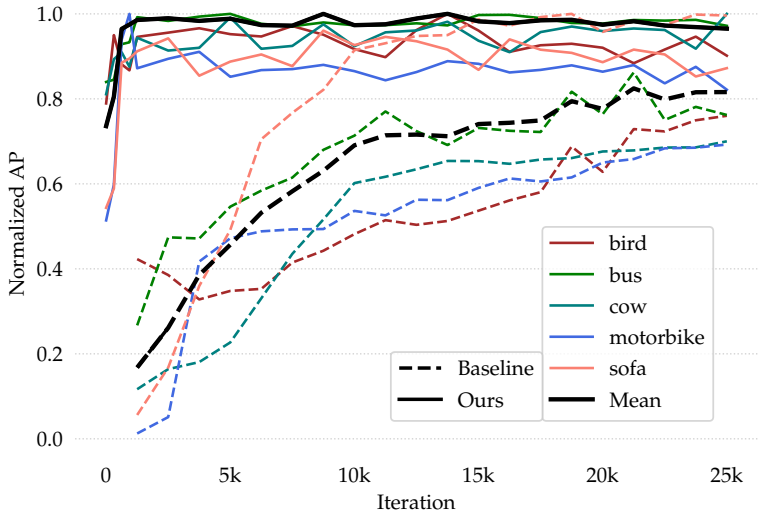
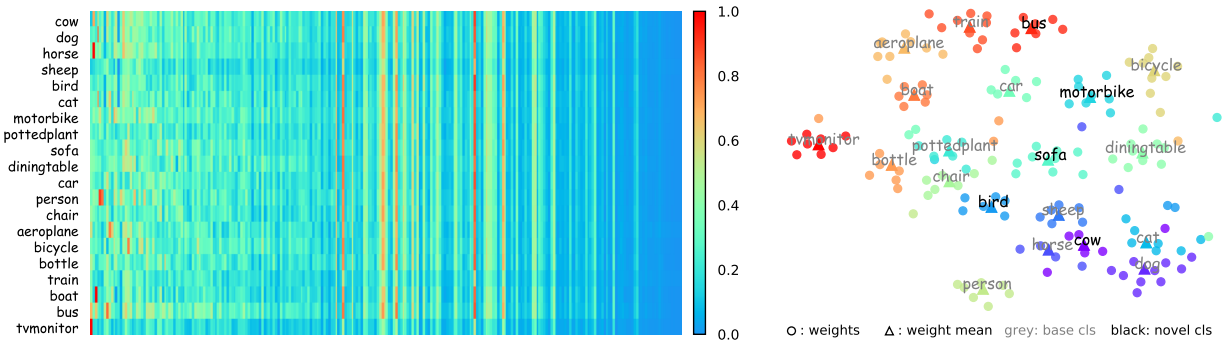


Figure 6.3: Learning speed comparison.

From Fig. 6.3, our method (solid lines) converges significantly faster than the baseline YOLO detector (dashed lines), for each novel class as well as on average. For the class Sofa (orange line), despite the baseline YOLO detector eventually slightly outperforms our method, it takes a great amount of training iterations to catch up with the latter. This behavior makes our model a good few-shot detector in practice, where scarcely labeled novel classes may come in any time and short adaptation time is desired to put the system in real usage fast. This also opens up our model's potential in a life-long learning setting [21], where the model accumulates the knowledge learned from past and uses/adapts it for future prediction. We also observe similar convergence advantage of our model over YOLO-ft-full and LSTD(YOLO)-full.

Learned reweighting coefficients.

The reweighting coefficient is important for the meta-feature usage and detection performance. To see this, we first plot the 1024-d reweighting vectors for each class in Fig. 6.4a. In the figure, each row corresponds to a class and each column corresponds to a feature. The features are ranked by variance among 20 classes from left to right. We observe that roughly half of the features (columns) have notable variance among different classes (multiple colors in a column), while the other half are insensitive to classes (roughly the same color in a column). This suggests that indeed only a portion of features are used differently when detecting different classes, while the remaining ones are shared across different classes.



(a) (b)

Figure 6.4: Visualization of the reweighting coefficients.

We further visualize the reweighting vectors by t-SNE [129] in Fig. 6.4b learned from 10 shots/class on base/novel split 1. In this figure, we plot the reweighting vector generated by each support input, along with their average for each class. We observe that not only vectors of the same classes tend to form clusters, the ones of visually similar classes also tend to be close. For instance, the classes Cow, Horse, Sheep, Cat and Dog are all around the right-bottom corner, and they are all animals. Classes of transportation tools are at the top of the figure. Person and Bird are more visually different from the mentioned animals, but are still closer to them than the transportation tools.

Learned meta features. Here we analyze the learned meta features from the base classes in the first training stage. Ideally, a desirable few-shot detection model should preferably perform as well when data are abundant. We compare the mAP on base classes for models obtained after the first-stage base training, between our model and the vanilla YOLO detector (used in latter two baselines). The results are shown in Table 6.3. Despite our detector is designed for a few-shot scenario, it also has strong representation power and offers good meta features to reach comparable performance with the original YOLOv2 detector trained on a lot of samples. This lays a basis for solving the few-shot object detection problem.

Table 6.3: Detection performance (mAP) on base categories.

	Base Set 1	Base Set 2	Base Set 3
YOLO Baseline	**70.3**	**72.2**	70.6
Our model	69.7	72.0	**70.8**

Ablation Studies

We analyze the effects of various components in our system, by comparing the performance on both base classes and novel classes. The experiments are on PASCAL VOC base/novel split 1, using 10-shot data on novel classes.

Table 6.4: Ablation study on re-weighted layers.

	Layer 13	Layer 20	Layer 21	Layer 21(half)
Base	69.6	69.2	**69.7**	69.2
Novel	40.7	43.6	**47.2**	46.9

Which layer output features to reweight. In our experiments, we apply the reweighting module to moderate the output of the second last layer (layer 21). This is the highest level of intermediate features we could use. However, other options could be considered as well. We experiment with applying the reweighting vectors to feature maps output from layer 20 and 13, while also considering only half of features in layer 21. The results are shown in Table 6.4. We can see that the it is more suitable to implement feature reweighting

at deeper layers, as using earlier layers gives worse performance. Moreover, moderating only half of the features does not hurt the performance much, which demonstrates that a significant portion of features can be shared among classes, as we analyzed in Sec. 6.3.

Loss functions. As we mentioned in Sec. 6.2, there are several options for defining the classification loss. Among them the binary loss is the most straightforward one: if the inputs to the reweighting module and the detector are from the same class, the model predicts 1 and otherwise 0. This binary loss can be defined in following two ways. The single-binary loss refers to that in each iteration the reweighting module only takes one class of input, and the detector regresses 0 or 1; and the multi-binary loss refers to that per iteration the reweighting module takes N examples from N classes, and compute N binary loss in total. Prior works on Siamese Network [98] and Learnet [10] use the single-binary loss. Instead, our model uses the softmax loss for calibrating the classification scores of N classes. To investigate the effects of using different loss functions, we compare model performance trained with the single-binary, multi-binary loss and with our softmax loss in Table 6.5. We observe that using softmax loss significantly outperforms binary loss. This is likely due to its effect in suppressing redundant detection results.

Table 6.5: Ablation of loss functions.

	Single-binary	Multi-binary	Softmax
Base	49.1	64.1	**69.7**
Novel	14.8	41.6	**47.2**

Input form of reweighting module. In our experiments, we use an image of the target class with a binary mask channel indicating position of the object as input to the meta-model. We examine the case where we only feed the image. From Table 6.6 we see that this gives lower performance especially on novel classes. An apparently reasonable alternative is to feed the cropped target object together with the image. From Table 6.6, this solution is also slightly worse. The necessity of the mask may lie in that it provides the precise information about the object location and its context.

We also analyze the input sampling scheme for testing and effect of sharing weights between feature extractor and reweighting module. See supplementary material.

Table 6.6: Performance comparison for different support input forms.

Image	Mask	Object	Base	Novel
✓			69.5	43.3
✓	✓		**69.7**	**47.2**
✓		✓	69.2	45.8
✓	✓	✓	69.4	46.8

6.4 Related Work

General object detection. Deep CNN based object detectors can be divided into two categories: proposal-based and proposal-free. RCNN series [59, 58, 153] detectors fall into the first category. RCNN [59] uses pre-trained CNNs to classify the region proposals generated by selective search [176]. SPP-Net [76] and Fast-RCNN [58] improve RCNN with an RoI pooling layer to extract regional features from the convolutional feature maps directly. Faster-RCNN [153] introduces a region-proposal-network (RPN) to improve the efficiency of generating proposals. In contrast, YOLO [152] provides a proposal-free framework, which uses a single convolutional network to directly perform class and bounding box predictions. SSD [121] improves YOLO by using default boxes (anchors) to adjust to various object shapes. YOLOv2 [150] improves YOLO with a series of techniques, e.g., multi-scale training, new network architecture (DarkNet-19). Compared with proposal-based methods, proposal-free methods do not require a per-region classifier, thus are conceptually simpler and significantly faster. Our few-shot detector is built on the YOLOv2 architecture.

Few-shot learning. Few-shot learning refers to learning from just a few training examples per class. Li *et al.* [112] use Bayesian inference to generalize knowledge from a pre-trained model to perform one-shot learning. Lake *et al.* [106] propose a Hierarchical Bayesian one-shot learning system that exploits compositionality and causality. Luo *et al.* [127] consider the problem of adapting to novel classes in a new domain. Douze *et al.* [44] assume abundant unlabeled images and adopts label propagation in a semi-supervised setting.

An increasingly popular solution for few-shot learning is meta-learning, which can further be divided into three categories: a) Metric learning based [98, 171, 180, 167]. In particular, Matching Networks [180] learn the task of finding the most similar class for the target image among a small set of labeled images. Prototypical Networks [167] extend Matching Networks by producing a linear classifier instead of weighted nearest neighbor for each class. Relation Networks [171] learn a distance metric to compare the target image to a few labeled images. b) Optimization for fast adaptation. Ravi and Larochelle [149] propose an LSTM meta-learner that is trained to quickly converge a learner classifier in new few-shot tasks. Model-Agnostic Meta-Learning (MAML) [53] optimizes a task-agnostic network so that a few gradient updates on its parameters would lead to good performance on new few-shot tasks. c) Parameter prediction. Learnet [10] dynamically learns the parameters of factorized weight layers based on a single example of each class to realize one-shot learning.

Above methods are developed to recognize novel images only, there are some other works

tried to learn a model that can classify both base and novel images. Recent works by Hariharan *et al.*[70, 192] introduce image hallucination techniques to augment the novel training data such that novel classes and base classes are balanced to some extend. Weight imprinting [146] sets weights for a new category using a scaled embedding of labeled examples. Dynamic-Net [57] learns a weight generator to classification weights for a specific category given the corresponding labeled images. These previous works only tackle image classification task, while our work focuses on object detection.

Object detection with limited labels. There are a number of prior works on detection focusing on settings with limited labels. The weakly-supervised setting [11, 38, 169] considers the problem of training object detectors with only image-level labels, but without bounding box annotations, which are more expensive to obtain. Few example object detection [135, 190, 41] assumes only a few labeled bounding boxes per class, but relies on abundant unlabeled images to generate trustworthy pseudo annotations for training. Zero-shot object detection [7, 148, 216] aims to detect previously unseen object categories, thus usually requires external information such as relations between classes. Different from these settings, our few-shot detector uses very few bounding box annotations (1-10) for each novel class, without the need for unlabeled images or external knowledge. Chen *et al.* [16] study a similar setting but only in a transfer learning context, where the target domain images only contains novel classes without base classes.

6.5 Chapter Summary

This work is among the first to explore the practical and challenging few-shot detection problems. It introduced a new model to learn to fast adjust contributions of the basic features to detect novel classes with a few example. Experiments on realistic benchmark datasets clearly demonstrate its effectiveness. This work also compared the model learning speed, analyzed predicted reweighting vectors and contributions of each design component, providing in-depth understanding of the proposed model. Few-shot detection is a challenging problem and we will further explore how to improve its performance for more complex scenes.

Chapter 7

Test-Time Fine-Tuning

7.1 Overview

Machine perception systems have witnessed significant success in the past years thanks to the emergence of convolutional neural networks [110, 73, 153]. However, compared to human visual systems, learning to generalize to novel concepts, which do not usually have abundant labeled data, is still far from satisfaction — even a toddler can easily recognize a new concept with very little instruction [109, 159, 166].

Equipping machines with a similar low-shot learning capability has long been desired by the machine learning community, with much focus on few-shot image classification. Many [180, 167, 53, 70, 57, 189] seek to transfer knowledge from the data-abundant base classes to the data-scarce novel classes through *meta-learning*, which acquires task-level meta knowledge using simulated few-shot tasks from base classes during training.

More recently, several [92, 203, 191] attempt to tackle the under-explored few-shot object detection task, where only a few labeled bounding boxes are available for novel classes, by attaching meta learners to existing object detection networks (*e.g.*, Faster R-CNN [153], YOLO [150]). Unlike image classification, object detection requires the model to not only classify the object types but also localize the target objects through millions of region proposals, which substantially increases the task complexity.

Meta-learning based approaches seem to be promising in both few-shot classification as well as detection. However, some [20] raise concerns about the reliability of the results given that a consistent comparison analysis of different approaches is missing. Some simple fine-tuning based approaches, which draw little attention in the community, turn out to be more favorable than many prior works that use meta-learning on few-shot image classification [20, 36]. As for the emerging few-shot object detection task, there is no consensus on the evaluation benchmarks nor a consistent comparison of different approaches due to the increased network complexity, various implementation details, and variances in evaluation protocols.

In this work, we take a step to consolidate the few-shot object detection benchmarking and carefully examine the fine-tuning based approaches, which are considered to be under-performing in

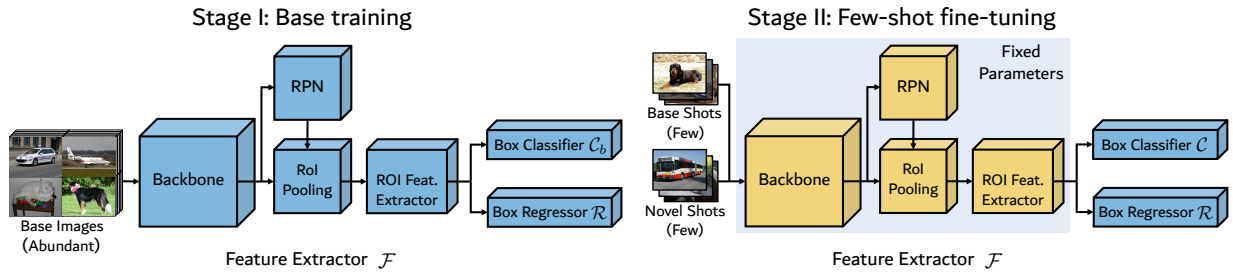


Figure 7.1: Illustration of our two-stage fine-tuning approach(TFA).

the previous literature [92, 203, 191]. We find that the performance of fine-tuning based approaches depend heavily on the training schedule as well as the instance-level feature normalization of the object detectors.

For a two-stage object detector (*e.g.*, Faster R-CNN), we first train the entire object detector on the data-abundant base classes, and then only fine-tune the last layers of the detector (*i.e.*, the fully connected layers used as the box classifier and the box regressor) on a small balanced training set consisting of both base and novel classes while freezing the parameters of the rest of the model. At the fine-tuning stage, we add instance-level feature normalization to the box classifier inspired by [57, 147, 20]. We find that such a simple two-stage fine-tuning approach (TFA) outperforms all previous meta-learning based methods by roughly 2$\sim$20 points on the existing benchmarks using Pascal VOC [47] and COCO [118] and sometimes even double the performance of the prior art.

In addition, we observe several issues with the existing evaluation protocols that prevent a consistent model comparison in the few-shot object detection task. First, the existing benchmark only reports the detection accuracy measured by average precision (AP) on the novel classes, missing the assessment over the base classes. Second, for the low shots experiments, the accuracy numbers usually have large variance which makes it comparison unreliable.

To this end, we build a new benchmark on three datasets: Pascal VOC, COCO and LVIS [66]. We revise the evaluation protocols to report the average precision on both the base classes and novel classes separately as well as the mean AP on all classes, referred to as generalized few-shot object detection, following the convention in the few-shot classification literature [70, 189]. We sample different groups of training examples repeatedly for the low-shot experiments to obtain a stable estimation of the uncertainty and quantitatively analyze the variances of different evaluation metrics.

Our approach establishes a new state of the art on all three datasets of the new benchmark. On the challenging LVIS dataset, our two-stage training scheme improves the AP of rare classes ($<$10 images) by $\sim$4 points and common classes (10-100 images) by $\sim$2 points with minimal decrease in AP ($\sim$0.5 points) of the frequent classes ($>$100 images).

7.2 Algorithms for Few-Shot Object Detection

In this section, we start with introducing the preliminaries on the few-shot object detection setting and then talk about our simple two-stage fine-tuning approach in Section 7.2. We summarize the

current meta-learning based approaches considered in this work in Section 7.2.

Few-shot object detection setup. We follow the few-shot object detection setting used in Kang *et al.* [92] and Yan *et al.* [203]. For an object detection dataset $\mathcal{D} = \{(x, y), x \in \mathcal{X}, y \in \mathcal{Y}\}$, where x is the input image and $y = \{(c^i, x_1^i, x_2^i, y_1^i, y_2^i), i = 1, ..., N\}$ denotes the categories and bounding box coordinates of the N object instances in the image x. There are a set of base classes C_b which have many instances and a set of novel classes C_n which have only K (usually less than 10) instances per category. For synthetic few-shot datasets using Pascal VOC and COCO, the novel set for training is balanced and each class has the same number of annotated objects (*i.e.*, K-shot). The recent LVIS has a natural long-tail distribution which does not have the manual K-shot split. The classes in LVIS are divided into *frequent* classes (appearing in more than 100 images), *common* classes (10-100 images), and *rare* classes (less than 10 images). We consider both synthetic and natural datasets in our work and follow the naming convention of k-shot for simplicity.

The few-shot object detector is evaluated on a test set of both the base classes and the novel classes. The goal is to optimize the detection accuracy measured by average precision (AP) of the novel classes as well as the base classes. This setting is different from the N-way-K-shot setting [53, 180, 167] commonly used in few-shot classification.

Two-stage fine-tuning approach

We describe our simple two-stage fine-tuning approach (TFA) for few-shot object detection in this section. We adopt the widely used Faster R-CNN [153], a two-stage object detector, as our base detector. As shown in Figure 7.1, the feature learning components (referred as $\mathcal{F}$) of a faster R-CNN model $\mathcal{O}$ include the backbone (*e.g.*, ResNet [73], VGG16 [165]), the region proposal network (RPN) as well as a two-layer fully-connected (FC) sub-network as a proposal-level feature extractor. There is also a box predictor composed of a box classifier $\mathcal{C}$ to classify the object categories and a box regressor $\mathcal{R}$ to predict the bounding box coordinates. Intuitively, the backbone features as well as the RPN features are class-agnostic; therefore, features learned from the base classes are likely to transfer to the novel classes without further parameter updates. Our key insight is to separate the feature representation learning and the box predictor learning into two stages.

Base training. In our first stage, we train the feature extractor and the box predictor only on the base classes C_b with the same loss function used in Ren *et al.* [153]. That is,

$$\mathcal{L} = \mathcal{L}_{\text{rpn}} + \mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{loc}}, \tag{7.1}$$

where $\mathcal{L}_{\text{rpn}}$ is applied to the output of RPN to distinguish the background/foreground and refine the pre-defined anchors. $\mathcal{L}_{\text{cls}}$ is a cross-entropy loss for the box classifier $\mathcal{C}$ and $\mathcal{L}_{\text{loc}}$ is a smoothed L_1 loss for the box regressor $\mathcal{R}$.

Few-shot fine-tuning. In the second stage, we create a small balanced dataset with K shots per class (both base and novel classes). We add randomly initialized weights of the novel classes to the trained box predictor and fine-tune only the box classifier and box regressor, namely the last layers of the detector $\mathcal{O}$, while keeping the entire feature extractor $\mathcal{F}$ fixed. We use the same loss

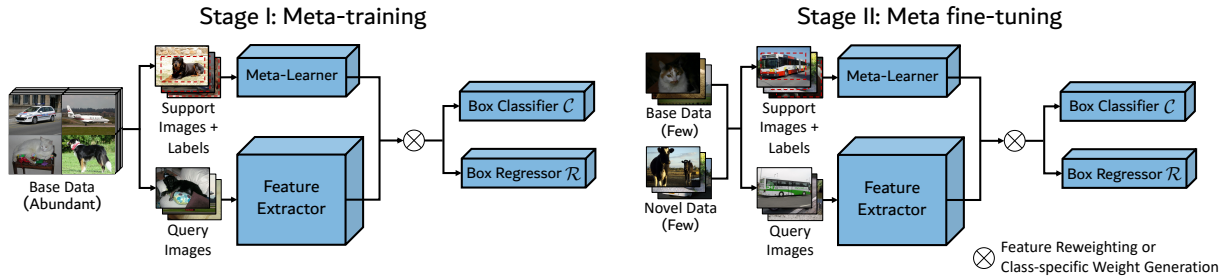


Figure 7.2: Abstraction of the meta-learning based few-shot object detectors.

objective function (Equation 7.1) and a reduced learning rate than the first stage. We reduce the learning rate by 20 in our experiments.

Cosine similarity based box classifier. Inspired by Gidaris *et al.* [57, 147, 20], we consider using a cosine similarity based classifier in the second fine-tuning stage. The weight matrix $W \in \mathbb{R}^{d \times c}$ of the box classifier $\mathcal{C}$ can be written as $[w_1, w_2, ..., w_c]$, where $w_c \in \mathbb{R}^d$ is the per-class weight vector. The output of C is scaled similarity scores S of the input feature $\mathcal{F}(x)$ and the weight vectors of different classes. That is,

$$s_{i,j} = \alpha \cdot \mathcal{F}(x)_i^\top w_j / (||\mathcal{F}(x)_i|| \cdot ||w_j||), \tag{7.2}$$

where $s_{i,j}$ is the similarity scores between the i-th object proposal of the input x and the weight vector of class j. α is the scaling factor, which we used a fixed value of 20 in our experiments. We find empirically that the instance level feature normalization used in the cosine similarity based classifier helps reduce the intra-class variance and improves the detection accuracy of novel classes with less decrease in base classes compared to a FC-based classifier especially when the number of training examples are very limited.

Meta-learning based approaches

We describe the existing meta-learning based few-shot object detection networks (FSRW [92], Meta R-CNN [203] and MetaDet [191]) in this section to draw connections with our approach. The architecture of FSRW [92] and Meta R-CNN [203] can be abstracted in Figure 7.2. In meta-learning based approaches, besides the base object detector (either single-stage or two-stage detectors), a meta-learner is introduced to acquire class-level meta knowledge and help the model generalize to novel classes through feature re-weighting (*e.g.*, FSRW and Meta R-CNN) or class-specific weight generation (*e.g.*, MetaDet). The input to the meta learner is a small set of support images as well as the bounding box annotations of the target objects.

The base object detector and the meta-learner are often jointly trained using episodic training. Each episode is composed of a supporting set of N objects and a set of query images. The support images and the binary masks of the annotated objects are used as input to the meta learner, which modulates the feature representation of the query images as adopted in FSRW and Meta R-CNN. As illustrated in Figure 7.2, the training procedure is also split into a meta-training stage where the

model is only trained on the data of the base classes and a meta fine-tuning stage, where the support set includes the few examples of the novel classes as well as a subset of examples from the base classes.

Both the meta-learning based approaches and our approach have a two-stage training. However, we find that the episodic learning used in meta-learning based approaches can be very memory inefficient as the number of classes in the supporting set increases. Our fine-tuning method only fine-tunes the last layers of the network with a normal batch training scheme which is much more memory efficient.

7.3 Experiments

In this section, we conduct extensive comparison with the previous methods on the existing few-shot object detection benchmarks using Pascal VOC and COCO, which our approach can obtain about 2$\sim$20 points on all settings (Section 7.3). We then introduce a new benchmark on three datasets (Pascal VOC, COCO and LVIS) with revised evaluation protocols to address the unreliability of the previous benchmark (Section 7.3). We also provide various ablation study and visualizations in Section 7.3.

Implementation details. All of our models are implemented on the Detectron2 framework [196]. We use Faster R-CNN [153] as our base detector and Resnet-101 [73] with a Feature Pyramid Network [77] as the backbone. For hyperparameters related to the model architecture, we use the default parameters provided by the Detectron2 framework. All models are trained using SGD with mini-batch size of 16, momentum of 0.9, and weight decay of 0.0001. A learning rate of 0.02 is used during base training and 0.001 during few-shot fine-tuning. We plan on open-sourcing our code for ease of reproduction and extension.

Existing few-shot object detection benchmark

Existing benchmarks. Followin the previous work [92, 203, 191], we first evaluate our approach on Pascal VOC 2007+2012 and MS-COCO, using the same data splits and training examples provided by Kang *et al.* [92][1]. For Few-shot Pascal VOC set, the 20 classes are randomly divided into 15 base classes and 5 novel classes, where the novel classes have $K = 1, 2, 3, 5, 10$ objects per class sampled from the combination of the trainval sets of the 2007 and 2012 versions for training. Three random split groups are considered in this work. Pascal VOC 2007 test set is used for evaluation. For MS-COCO, the non-overlapping categories (60 classes) with Pascal VOC are used as base classes while the remaining 20 classes are used as novel classes with $K = 10, 30$. For evaluation metrics, AP50 (matching threshold is 0.5) of the novel classes is used for the Pascal VOC setting and the COCO-style AP of the novel classes is used on MS-COCO.

Baselines. We compare our approach with the meta-learning based approaches `FSRW`, `Meta-RCNN` and `MetaDet` together with the fine-tuning based approaches: *jointly training* (denoted by

[1] code and data at `https://github.com/bingykang/Fewshot_Detection`

Table 7.1: Few-shot detection performance (mAP50) on PASCAL VOC.

	Backbone	Novel Set 1					Novel Set 2					Novel Set 3				
Method / Shot	-	1	2	3	5	10	1	2	3	5	10	1	2	3	5	10
\ [92]	YOLOv2	0.0	0.0	1.8	1.8	1.8	0.0	0.1	0.0	1.8	0.0	1.8	1.8	1.8	3.6	3.9
YOLO-ft [92]		3.2	6.5	6.4	7.5	12.3	8.2	3.8	3.5	3.5	7.8	8.1	7.4	7.6	9.5	10.5
YOLO-ft-full [92]		6.6	10.7	12.5	24.8	38.6	12.5	4.2	11.6	16.1	33.9	13.0	15.9	15.0	32.2	38.4
FSRW [92]		14.8	15.5	26.7	33.9	47.2	15.7	15.3	22.7	30.1	40.5	21.3	25.6	28.4	42.8	45.9
MetaDet [191]		17.1	19.1	28.9	35.0	48.8	18.2	20.6	25.9	30.6	41.5	20.1	22.3	27.9	41.9	42.9
FRCN+joint [191]	FRCN w/VGG16	0.3	0.0	1.2	0.9	1.7	0.0	0.0	1.1	1.9	1.7	0.2	0.5	1.2	1.9	2.8
FRCN+joint-ft [191]		9.1	10.9	13.7	25.0	39.5	10.9	13.2	17.6	19.5	36.5	15.0	15.1	18.3	33.1	35.9
MetaDet [191]		18.9	20.6	30.2	36.8	49.6	21.8	23.1	27.8	31.7	43.0	20.6	23.9	29.4	43.9	44.1
FRCN+joint [203]	FRCN w/R-101	2.7	3.1	4.3	11.8	29.0	1.9	2.6	8.1	9.9	12.6	5.2	7.5	6.4	6.4	6.4
FRCN+ft [203]		11.9	16.4	29.0	36.9	36.9	5.9	8.5	23.4	29.1	28.8	5.0	9.6	18.1	30.8	43.4
FRCN+ft-full [203]		13.8	19.6	32.8	41.5	45.6	7.9	15.3	26.2	31.6	39.1	9.8	11.3	19.1	35.0	45.1
Meta R-CNN [203]		19.9	25.5	35.0	45.7	51.5	10.4	19.4	29.6	34.8	**45.4**	14.3	18.2	27.5	41.2	48.1
FRCN+ft-full (Ours)		15.2	20.3	29.0	40.1	45.5	13.4	20.6	28.6	32.4	38.8	19.6	20.8	28.7	42.2	42.1
TFA w/ fc (Ours)	FRCN w/R-101	36.8	29.1	43.6	**55.7**	**57.0**	18.2	**29.0**	33.4	**35.5**	39.0	27.7	33.6	42.5	48.7	**50.2**
TFA w/ cos (Ours)		**39.8**	**34.4**	**44.7**	**55.7**	56.0	**23.5**	26.9	**34.1**	35.1	**39.1**	**30.8**	**34.8**	**42.8**	**49.5**	49.8

Table 7.2: Few-shot detection performance on Novel Set 1 of PASCAL VOC.

# shots	Method	Base AP50	Novel AP50
3	FRCN+ft-full [203]	63.6	32.8
	Meta R-CNN [203]	64.8	35.0
	Train base only	80.8	9.0
	FRCN+ft-full (Ours)	66.1	29.0
	TFA w/ cos (Ours)	79.1	44.7
10	FRCN+ft-full [203]	61.3	45.6
	Meta R-CNN [203]	67.9	51.5
	Train base only	80.8	9.0
	FRCN+ft-full (Ours)	66.0	45.5
	TFA w/ cos (Ours)	78.4	56.0

`FRCN/YOLO+joint`. FRCN is Faster R-CNN for short) where the base and novel class examples are jointly trained in the base training stage, and *fine-tuning the entire model* (denoted by `FRCN/YOLO+ft-full`) where both the feature extractor $\mathcal{F}$ and the box predictor ($\mathcal{C}$ and $\mathcal{R}$) are jointly fine-tuned until convergence in the second fine-tuning stage. Fine-tuning with less iterations (denoted by `FRCN/YOLO+ft`) are reported in Kang *et al.* [92] and Yan *et al.* [203].

Results on PASCAL VOC. We provide the average AP50 of the novel classes on Pascal VOC with three random splits in Table 7.1. Our approach uses ResNet-101 as the backbone which is comparable with Meta R-CNN. We implement the `FRCN+ft-full` in our framework which roughly matches the results reported in Yan *et al.* [203]. `MetaDet` uses a different backbone of VGG16 but the performance is similar and sometimes worse than Meta R-CNN with ResNet-101 as the backbone according to Table 7.1.

In all different data splits and different number of training shots, our approach (the last row) is

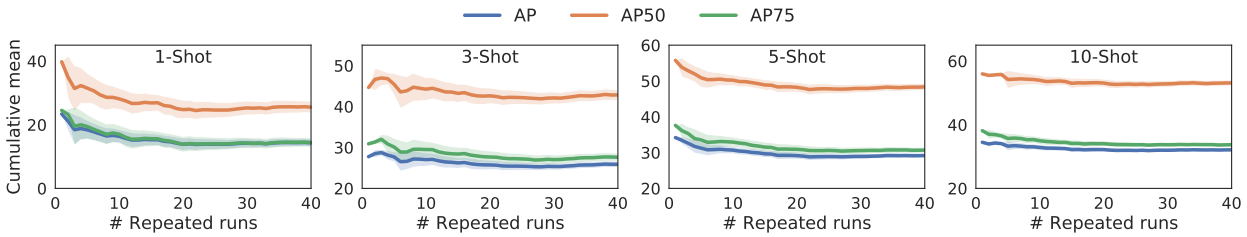


Figure 7.3: Cumulative mean with 95% confidence interval across 40 repeated runs, computed on the first split of PASCAL VOC.

able to out perform the previous methods with a large margin. It even doubles the performance of the previous approaches in the one-shot cases. The improvements (up to 20 points) is much larger than the gap among the previous meta-learning based approaches, indicating the effectiveness of our approach. We also compare the cosine similarity based box classifier (`TFA+w/cos`) with a normal FC-based classifier(`TFA+w/fc`) and find that `TFA+w/cos` is better than `TFA+w/fc` on extremely low shots (*e.g.,* 1-shot) but the two are roughly similar as there are more training shots (*e.g.*, 10-shot).

We also cite the numbers from Yan *et al.* [203] for their model performance on the base classes for more detailed comparison in Table 7.2. We find that our model also has much higher average AP on the base classes than `Meta R-CNN` with a gap of about 10 to 15 points. To eliminate the differences in implementation details, we also report our re-implementation of `FRCN+ft-full` and `training base only` (where the base class should have the highest accuracy as the model is only trained on the base classes examples). Again, we find our model has a minimal decrease (less than 2 points) on the base classes.

Table 7.3: Few-shot detection performance on COCO.

		novel AP		novel AP75	
Model	Backbone	10	30	10	30
FSRW	YOLOv2	5.6	9.1	4.6	7.6
MetaDet	FRCN w/VGG16	7.1	11.3	6.1	8.1
FRCN+ft+full	FRCN w/R-101	6.5	11.1	5.9	10.3
Meta R-CNN	FRCN w/R-101	8.7	12.4	6.6	10.8
FRCN+ft-full (Ours)	FRCN w/R-101	9.2	12.5	**9.2**	12.0
TFA w/ fc (Ours)	FRCN w/R-101	**10.0**	13.4	**9.2**	13.2
TFA w/ cos (Ours)	FRCN w/R-101	9.8	**13.7**	9.0	**13.4**

Results on MS-COCO. Similarly, we report the average AP and AP75 (matching threshold is 0.75, a more strict metric than AP50) of the 20 novel classes on MS-COCO in Table 7.3. Similar to Pascal VOC, we consistently outperform previous methods across all shots on both novel AP and

Table 7.4: Generalized object detection benchmarks on LVIS.

Method	Backbone	Repeated sampling	AP	AP50	AP75	APs	APm	APl	APr	APc	APf
Joint training	FRCN w/ R-50		19.8	33.6	20.4	17.1	25.9	33.2	2.1	18.5	28.5
TFA w/ fc (Ours)			22.3	37.8	22.2	18.5	28.2	36.6	14.3	21.1	27.0
Joint training	FRCN w/ R-50	✓	23.1	38.4	24.3	18.1	28.3	36.0	13.0	22.0	**28.4**
TFA w/ fc (Ours)			24.1	39.9	25.4	19.5	29.1	36.7	14.9	23.9	27.9
TFA w/ cos (Ours)			**24.4**	**40.0**	**26.1**	**19.9**	**29.5**	**38.2**	**16.9**	**24.3**	27.7
Joint training	FRCN w/ R-101		21.9	35.8	23.0	18.8	28.0	36.2	3.0	20.8	**30.8**
TFA w/ fc (Ours)			23.9	39.3	25.3	19.5	29.5	38.6	16.2	22.3	28.9
Joint training	FRCN w/ R-101	✓	24.7	40.5	26.0	19.0	30.3	38.0	13.4	24.0	30.1
TFA w/ fc (Ours)			25.4	**41.8**	27.0	19.8	31.1	39.2	15.5	26.0	28.6
TFA w/ cos (Ours)			**26.2**	**41.8**	**27.5**	**20.2**	**32.0**	**39.9**	**17.3**	**26.4**	29.6

novel AP75. We achieve around 1 point improvement in AP over the best performing baseline and around 2.5 points improvement in AP75.

Generalized few-shot object detection benchmark

Revised evaluation protocols. We find several issues with existing benchmarks. First, previous evaluation protocols focus only on performance on novel classes. This ignores the potential performance drop in base classes and the overall performance of the network. Second, the sample variance is large due to the few samples that are used for training. This makes it difficult to compare against other methods, as differences in performance could be insignificant.

To address the issues, we first evaluate our approach using three metrics: AP, nAP, and bAP. nAP and bAP are the AP measurements for novel and base classes, respectively. We also evaluate on both novel and base classes together, which we term as just AP. Doing so allows us to observe trends in performance of both base and novel classes, and the overall performance of the network.

Additionally, we train our models on multiple random samples of training shots to obtain averages and confidence intervals. In Figure 7.3, we show the cumulative mean and 95% confidence interval across 40 repeated runs on four different shots on the first split of PASCAL VOC. Although the performance is high on the first random sample, the average decreases significantly as more samples are used. Additionally, the confidence intervals across the first few runs are large, especially in the low-shot scenario. When we use more repeated runs, the averages stabilizes and the confidence intervals become small, which allows for better comparisons.

Results on LVIS. We evaluate our approach on the recently introduced LVIS dataset [66], which has a natural long-tail distribution. We treat the frequent and common classes as base classes, and the rare classes as novel classes. The base training is done the same as before. During few-shot fine-tuning, we artificially create a balanced-subset of the entire dataset by sampling up to 10 instances for each class and fine-tune on this subset.

We show evaluation results on generalized LVIS in Table 7.4. Compared to the methods in [66], our method is able to achieve better performance ($\sim$1-1.5 points in overall AP and $\sim$2-4 points in

Table 7.5: Generalized object detection benchmarks on Pascal VOC.

Split	# shots	Method	Overall			Base class			Novel class		
			AP	AP50	AP75	bAP	bAP50	bAP75	nAP	nAP50	nAP75
Split 1	1	TFA w/ fc	39.6±0.5	63.5±0.7	43.2±0.7	48.7±0.7	77.1±0.7	53.7±1.0	12.2±1.6	22.9±2.5	11.6±1.9
		TFA w/ cos	40.6±0.5	64.5±0.6	44.7±0.6	49.4±0.4	77.6±0.2	54.8±0.5	14.2±1.4	25.3±2.2	14.2±1.8
	2	TFA w/ fc	40.5±0.5	65.5±0.7	43.8±0.7	47.8±0.7	75.8±0.7	52.2±1.0	18.9±1.5	34.5±2.4	18.4±1.9
		TFA w/ cos	42.6±0.3	67.1±0.4	47.0±0.4	49.6±0.3	77.3±0.2	55.0±0.4	21.7±1.0	36.4±1.6	22.8±1.3
	3	TFA w/ fc	41.8±0.9	67.1±0.9	45.4±1.2	48.2±0.9	76.0±0.9	53.1±1.2	22.6±1.2	40.4±1.7	22.4±1.7
		TFA w/ cos	43.7±0.3	68.5±0.4	48.3±0.4	49.8±0.3	77.3±0.2	55.4±0.4	25.4±0.9	42.1±1.5	27.0±1.2
	5	TFA w/ fc	41.9±0.6	68.0±0.7	45.0±0.8	47.2±0.6	75.1±0.6	51.5±0.8	25.9±1.0	46.7±1.4	25.3±1.2
		TFA w/ cos	44.8±0.3	70.1±0.4	49.4±0.4	50.1±0.2	77.4±0.3	55.6±0.3	28.9±0.8	47.9±1.2	30.6±1.0
	10	TFA w/ fc	42.8±0.3	69.5±0.4	46.0±0.4	47.3±0.3	75.4±0.3	51.6±0.4	29.3±0.7	52.0±1.1	29.0±0.9
		TFA w/ cos	45.8±0.2	71.3±0.3	50.4±0.3	50.4±0.2	77.5±0.2	55.9±0.3	32.0±0.6	52.8±1.0	33.7±0.7
Split 2	1	TFA w/ fc	36.2±0.8	59.6±0.9	38.7±1.0	45.6±0.9	73.8±0.9	49.4±1.2	8.1±1.2	16.9±2.3	6.6±1.1
		TFA w/ cos	36.7±0.6	59.9±0.8	39.3±0.8	45.9±0.7	73.8±0.8	49.8±1.1	9.0±1.2	18.3±2.4	7.8±1.2
	2	TFA w/ fc	38.5±0.5	62.8±0.6	41.2±0.6	46.9±0.5	74.9±0.5	51.2±0.7	13.1±1.0	26.4±1.9	11.3±1.1
		TFA w/ cos	39.0±0.4	63.0±0.5	42.1±0.6	47.3±0.4	74.9±0.4	51.9±0.7	14.1±0.9	27.5±1.6	12.7±1.0
	3	TFA w/ fc	39.4±0.4	64.2±0.5	42.0±0.5	47.5±0.4	75.4±0.5	51.7±0.6	15.2±0.8	30.5±1.5	13.1±0.8
		TFA w/ cos	40.1±0.3	64.5±0.5	43.3±0.4	48.1±0.3	75.6±0.4	52.9±0.5	16.0±0.8	30.9±1.6	14.4±0.9
	5	TFA w/ fc	40.0±0.4	65.1±0.5	42.6±0.5	47.5±0.4	75.3±0.5	51.6±0.5	17.5±0.7	34.6±1.1	15.5±0.9
		TFA w/ cos	40.9±0.4	65.7±0.5	44.1±0.5	48.6±0.4	76.2±0.4	53.3±0.5	17.8±0.8	34.1±1.4	16.2±1.0
	10	TFA w/ fc	41.3±0.2	67.0±0.3	44.0±0.3	48.3±0.2	76.1±0.3	52.7±0.4	20.2±0.5	39.7±0.9	18.0±0.7
		TFA w/ cos	42.3±0.3	67.6±0.4	45.7±0.3	49.4±0.2	76.9±0.3	54.5±0.3	20.8±0.6	39.5±1.1	19.2±0.6
Split 3	1	TFA w/ fc	39.0±0.6	62.3±0.7	42.1±0.8	49.5±0.8	77.8±0.8	54.0±1.0	7.8±1.1	15.7±2.1	6.5±1.0
		TFA w/ cos	40.1±0.3	63.5±0.6	43.6±0.5	50.2±0.4	78.7±0.2	55.1±0.5	9.6±1.1	17.9±2.0	9.1±1.2
	2	TFA w/ fc	41.1±0.6	65.1±0.7	44.3±0.7	50.1±0.7	77.7±0.7	54.8±0.9	14.2±1.2	27.2±2.0	12.6±1.3
		TFA w/ cos	41.8±0.4	65.6±0.6	45.3±0.4	50.7±0.3	78.4±0.2	55.6±0.4	15.1±1.3	27.2±2.1	14.4±1.5
	3	TFA w/ fc	40.4±0.5	65.4±0.7	43.1±0.7	47.8±0.5	75.6±0.5	52.1±0.7	18.1±1.0	34.7±1.6	16.2±1.3
		TFA w/ cos	43.1±0.4	67.5±0.5	46.7±0.5	51.1±0.3	78.6±0.2	56.3±0.4	18.9±1.1	34.3±1.7	18.1±1.4
	5	TFA w/ fc	41.3±0.5	67.1±0.6	44.0±0.6	48.0±0.5	75.8±0.5	52.2±0.6	21.4±0.9	40.8±1.3	19.4±1.0
		TFA w/ cos	44.1±0.3	69.1±0.4	47.8±0.4	51.3±0.2	78.5±0.3	56.4±0.3	22.8±0.9	40.8±1.4	22.1±1.1
	10	TFA w/ fc	42.2±0.4	68.3±0.5	44.9±0.6	48.5±0.4	76.2±0.4	52.9±0.5	23.3±0.8	44.6±1.1	21.0±1.2
		TFA w/ cos	45.0±0.3	70.3±0.4	48.9±0.4	51.6±0.2	78.6±0.2	57.0±0.3	25.4±0.7	45.6±1.1	24.7±1.1

AP for rare and common classes). We also demonstrate results without using repeated sampling, which is a weighted sampling scheme that is used in [66] to address the data imbalance issue. Our simple approach is able to greatly outperform the corresponding baseline. Notably, the AP on rare classes increased around 12 points and the AP on common classes increased around 1.5 points. This shows that our simple two-stage fine-tuning scheme is able to address the severe data imbalance issue without needing repeated sampling.

Benchmarking results on PASCAL VOC and MS-COCO. We show evaluation results on generalized PASCAL VOC in Table 7.5 and MS-COCO in Table 7.6. On both datasets, we evaluate on the base classes and the novel classes and report AP scores for each. On PASCAL VOC, we evaluate our models over 30 repeated runs and report the average and the 95% confidence interval. On MS-COCO, we provide results on 1, 2, 3, and 5 shots in addition to the 10 and 30 shots used by

Table 7.6: Generalized object detection benchmarks on COCO.

# shots	Method	Overall						Base class			Novel class		
		AP	AP50	AP75	APs	APm	APl	bAP	bAP50	bAP75	nAP	nAP50	nAP75
1	TFA w/ fc	25.7	41.0	27.7	14.8	28.3	33.7	33.3	52.8	36.0	2.9	5.7	2.8
	TFA w/ cos	26.4	42.5	28.2	16.4	28.4	33.0	34.1	54.7	36.4	3.4	5.8	3.8
2	TFA w/ fc	26.1	41.4	28.3	14.7	29.2	34.8	33.3	52.3	36.4	4.3	8.5	4.1
	TFA w/ cos	27.1	43.4	29.4	16.8	29.5	34.0	34.7	55.1	37.6	4.6	8.3	4.8
3	TFA w/ fc	26.9	42.5	29.5	15.3	29.8	36.3	33.7	52.5	37.2	6.7	12.6	6.6
	TFA w/ cos	27.7	44.1	30.0	16.6	29.9	35.5	34.7	54.8	37.9	6.6	12.1	6.5
5	TFA w/ fc	27.5	43.6	30.0	15.6	30.5	36.8	33.9	52.8	37.2	8.4	16.0	8.4
	TFA w/ cos	28.1	44.6	30.2	16.9	30.3	36.3	34.7	54.4	37.6	8.3	15.3	8.0
10	TFA w/ fc	27.9	44.6	30.4	15.8	30.9	37.6	33.9	53.1	37.4	10.0	19.2	9.2
	TFA w/ cos	28.4	45.4	30.7	16.6	30.9	37.2	34.6	54.3	37.9	9.8	18.7	9.0
30	TFA w/ fc	29.7	46.8	32.5	16.7	32.7	40.1	35.1	54.2	38.9	13.4	24.7	13.2
	TFA w/ cos	30.3	47.9	32.9	17.8	32.6	40.2	35.8	55.5	39.4	13.7	24.9	13.4

Figure 7.4: Detection results of our approach on novel classes of PASCAL VOC

the existing benchmark for a better picture of performance trends in the low-shot regime. We defer the full experimental results of the generalized MS-COCO benchmark to the appendix.

Ablation study and Visualization

Weight initialization. We explore two different ways of initializing the weights of the novel classifier before few-shot fine-tuning: (1) random initialization and (2) fine-tuning a predictor on the novel set and using the classifier's weights as initialization. We compare both methods on 1/3/10 shots on split 3 of PASCAL VOC and MS-COCO and show the results in Table 7.7. On PASCAL VOC, simple random initialization can outperform initialization using fine-tuned novel weights. On MS-COCO, using the novel weights can improve the performance over random initialization. This is probably due to the increased complexity and number of classes of MS-COCO compared to PASCAL VOC. We use random initialization for all PASCAL VOC experiments and novel initialization for all MS-COCO and LVIS experiments.

Detection results. We visualize the success and failure cases of our 10-shot TFA w/ cos model on novel objects from the first split of PASCAL VOC in Figure 7.4. Some failure cases include

Table 7.7: Ablation of weight initialization of the novel classifier.

Dataset	Init.	Base AP 1	Base AP 3	Base AP 10	Novel AP 1	Novel AP 3	Novel AP 10
PASCAL VOC (split 3)	Random	**51.2**	52.6	**52.8**	**15.6**	**25.0**	**29.4**
	Novel	50.9	**53.1**	52.5	13.4	24.9	28.9
MS-COCO	Random	34.0	**34.7**	**34.6**	3.2	6.4	9.6
	Novel	**34.1**	**34.7**	**34.6**	**3.4**	**6.6**	**9.8**

misclassifying novel objects as similar base objects (*e.g.*, cow as horse) and identifying incorrect instance boundaries.

7.4 Related Work

Our work is related to the rich literature on few-shot image classification which uses various meta-learning based or metric-learning based methods. We also draw connections between our work and the existing meta-learning based few-shot object detectors. To the best of our knowledge, we are the first to conduct a systematic analysis of fine-tuning based approaches on the few-shot object detection task.

Meta-learning. The goal of meta-learning is to acquire task-level meta knowledge that can help the model quickly adapt to new tasks and environments with very few labeled examples. Some [53, 158, 139] use learning to fine-tuning which aims to obtain a good parameter initialization that can adapt to new tasks with a few scholastic gradient updates. Another popular line of research on meta-learning is to use parameter generation when adapting to novel tasks. Gidaris *et al.* [57] propose an attention based weight generator to generate the classifier weights for the novel classes. Wang *et al.* [189] construct task-aware feature embeddings by generating parameters for the feature layers. These approaches have only been used for the few-shot image classification, not on more challenging tasks like object detection.

Metric-learning. Another line of work [98, 167, 180] focuses on learning to compare or metric learning. Intuitively, if the model can construct distance metrics to estimate the similarity between two input images, it may generalize to novel categories with few labeled instances. More recently, several [20, 57, 147] adopt a cosine similarity based classifier to reduce the intra-class variance on the few-shot classification task, which has favorable performance than many meta-learning based approaches. Similarly, our work also adopts a cosine similarity classifier to classify the categories of the region proposals. However, we are focused on the instance-level distance measurement rather than on the image level.

Few-shot object detection. There are several early attempts on few-shot object detection using meta learning. Kang *et al.* [92] and Yan *et al.* [203] apply feature re-weighting schemes to a single-stage object detector (YOLOv2) and a two-stage object detector (Faster R-CNN) with the help of a meta learner which takes the support images (i.e., small number of labeled images of the

novel/base classes) as well as the bounding box annotations as inputs. Wang *et al.* [191] propose a weight prediction meta-model to predict parameters of category-specific components from few examples while learning the category-agnostic parts from base class examples.

In all these works, fine-tuning based approaches are considered as baselines with worse performances than meta-learning based approaches. They consider *jointly fine-tuning* (where base classes and novel classes are trained together) and *fine-tuning the entire model* (where the detector is first trained on the base classes only and then fine-tuned on a balanced set with both base and novel classes). In contrast, we find that when fine-tuning only the last layer of the object detector on the balanced subset and keep the rest of model fixed can substantially improve the detection accuracy, outperforming all the prior meta-learning based approaches. This indicates that feature representations learned from the base classes might be able to transfer to the novel classes and simple adjustments on the box predictor can provide strong performance in this transductive learning setting [36].

7.5 Chapter Summary

In this work, we proposed a simple two-stage fine-tuning approach for the under-explored few-shot object detection task, which outperformed the previous meta-learning based methods by a large margin on the existing few-shot object detection benchmark. In addition, we build a more reliable banchmarks with revised evaluation protocols on three datasets: Pascal VOC, COCO and LVIS. On the new benchmark, our model established new state of the art and improve the AP of rare classes by 4 points with neligible reduction on frequent classes on the new LVIS dataset.

Chapter 8

ShapeProp for Semi-Supervised Learning

8.1 Overview

Instance segmentation methods [75, 17, 119] have enjoyed great success recently thanks to deep convolutional networks and availability of new datasets [26, 209]. Those methods can be applied in a broad range of applications such as key point detection and 3D cuboid fitting. However, collecting these fine annotations requires prohibitively expensive human effort, preventing the existing frameworks from learning on larger data. This difficulty limits our further study in the instance segmentation problem.

One possible solution is to use the abundant cheaper bounding box annotations to relieve the shortage of segmentation supervision. Some works [83, 105] have explored how to generalize to unseen categories with the weak supervision. However, the segmentation modules usually only get pixel information from only fine segmentation supervision. It is still not well understood how to exploit the new pixel-level information from bounding boxes, even though more box-level supervision can improve the object localization,

In this work, we aim to combine the weekly supervised segmentation information in bounding boxes and full instance segmentation supervision for semi-supervised learning of instance segmentation. The task is a generalization of the "partially supervised instance segmentation" in previous literature [82, 105] which assumes the costly mask annotations are available for a subset of categories. We denote this as category-wise semi-supervision and also consider a realistic image-wise semi-supervision where only a subset of images has masks. The category-wise and image-wise semi-supervision focuses on inter- and intra-class generalization. Note that the considerably cheaper bounding box annotations are available for all object instances.

Semi-supervised instance segmentation is challenging to existing instance segmentation frameworks. Current single-stage (e.g., RetinaMask [56]) or two-stage (e.g., Mask R-CNN family [75, 119]) instance segmentation frameworks do not take full advantage of the existing supervision. They typically use a detection head and a segmentation head to learn from the box and mask supervision separately. So the segmentation head does not explicitly benefit from the abundant box annotations as each box is a coarse-grained representation which does not specify object shape. Also, the

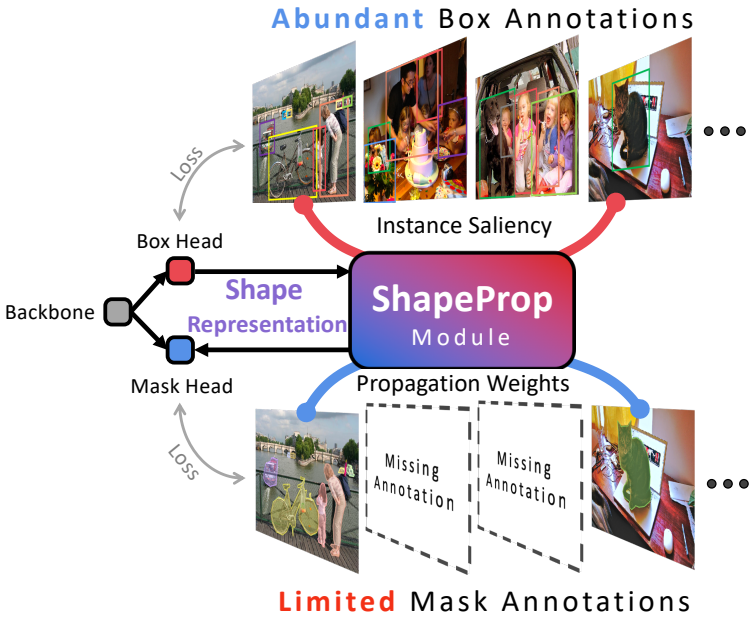


Figure 8.1: ShapeProp illustration.

segmentation head requires high-level semantics to predict pixel-wise labels, making it difficult to adequately capture data distribution when only limited masks are available.

Our goal is to learn to locate and segment objects from both box and segmentation annotations. We propose ShapeProp, a lightweight network module that can extend existing instance segmentation frameworks to utilize box-level instance supervision for the semi-supervised instance segmentation. ShapeProp exploits the shape prior information hidden inside box and mask annotations to improve accuracy and generalization of existing instance segmentation frameworks, as illustrated in Fig. 8.1. ShapeProp can learn instance-specific saliency from the abundant box supervision and model the instance-specific latent relationships between pixels from the limited segmentation annotations.

ShapeProp extracts salient regions of the instance from the box detection outputs. Then it propagates salient regions into an intermediate shape representation, referred as *Shape Activation*, which specifies fine-detailed instance shape extents, as shown in Fig. 8.2. Shape activation indicates the potential object shapes and provides shape prior. We then fuse it with the region features before making final instance segmentation predictions.

Our approach does not use preprocessing steps such as grouping masks in ShapeMask [105], and can be easily integrated and jointly trained with existing instance segmentation frameworks. In contrast to methods [83] based on transfer learning, which only aims to improve inter-class generalization, our approach improves both inter- and intra-class generalization. Furthermore, our method is lightweight and do not introduce heavy computational overhead to the model inference speed.

Extensive experiments show that ShapeProp module brings consistent and significant improvements to baselines, achieving top performance on various benchmarks. For instance, on semi-supervision setting, Mask R-CNN augmented by ShapeProp improves the baseline by 10.8 AP

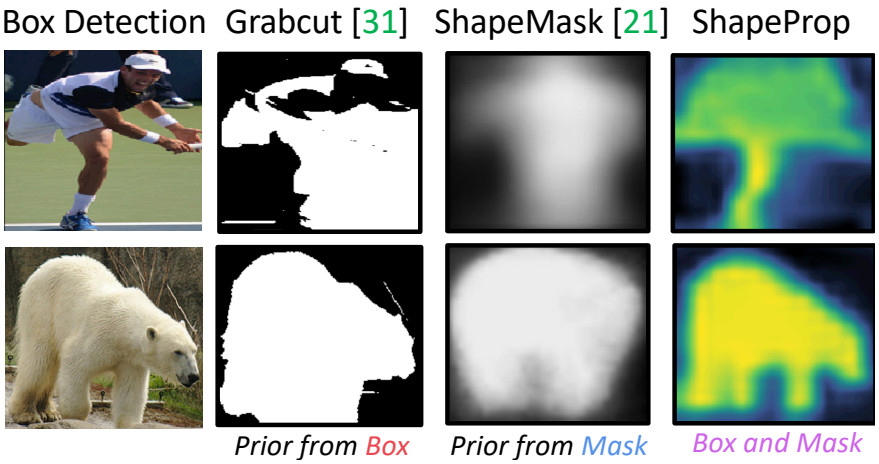


Figure 8.2: Visualization of extracted shape representation.

and outperform the state-of-the-art by 2.2 AP. Those results show strong evidence that ShapeProp effectively improves model's segmentation quality and generalization ability.

8.2 Related Work

Instance segmentation. Instance segmentation is a fundamental task in the computer vision can be roughly categorized into *detection-based* or *grouping-based* approaches. The detection-based methods [71, 28, 72, 114, 18, 75] dominates the state-of-the-art performance in commonly used benchmarks, e.g., COCO [118]. They typically follow a multi-task learning paradigm where a backbone network first extracts spatial features and generates a set of candidate regions, either with region proposal networks [153] or dense anchor boxes [56]. Then a detection head and a segmentation head which composes several convolution -relu layers predicted the accurate box and the segmentation mask inside the region cropped by the detected box. The grouping-based approaches [99, 6, 31, 122, 120, 5, 97] view the instance segmentation as a bottom-up grouping problem. Although great progress has been achieved in the instance segmentation task, most of these works require strong supervision in the form of hand-annotated instance masks for all objects, which limits their application on large-scale datasets.

Weakly supervised instance segmentation. Methods learning with weaker labels try to break this limitation by learning with a weaker form of supervision. [95] leverages the idea that given a bounding box for the target object, one can obtain a pseudo mask label from a grouping-based segmentation algorithm like GrabCut [155]. Pham *et al.* [143] study open-set instance segmentation by using a boundary detector followed by grouping. Zhou *et al.* [214, 217] tackle weakly supervised instance segmentation by exploiting the class peak response of classification networks. Although effective, these approaches do not take advantage of *existing* instance mask labels to achieve better performance.

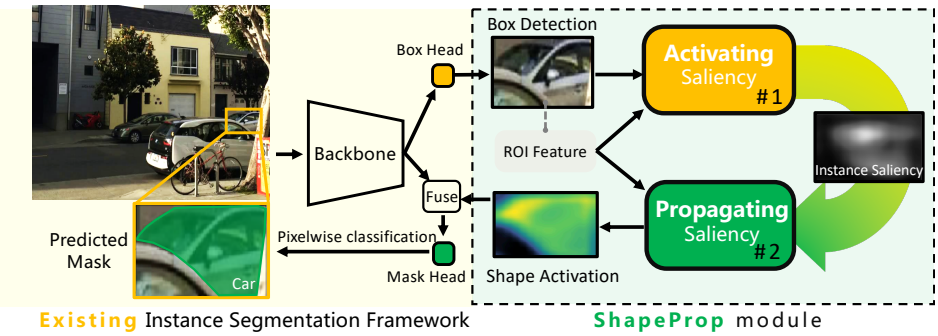


Figure 8.3: ShapeProp Framework.

Learning with limited masks. As opposed to the weakly-supervised setting [95, 214], some recent approaches tackle the partially supervised setting where only box labels (not mask labels) are available for a subset of categories at training time. The model is required to perform instance segmentation on these categories at test time, which requires strong generalization ability. Mask^X R-CNN [83] tackles the problem by learning to predict weights of mask segmentation branch from the box detection branch. This transfer learning approach shows significant improvement over class-agnostic training; however, performance gap to fully supervised methods remains significant.

Moreover, it only addresses inter-class generalization problems while ignoring intra-class generalization (i.e. novel instances from seen categories). ShapeMask [105] is based on a strong assumption that shape bases from existing mask annotations could serve as canonical shapes and generalize to unseen categories. ShapeMask uses the limited mask annotations to extract prior knowledge to help segmentation. Despite its effectiveness, this approach drew class agnostic shape prior from the limited mask annotations and neglect the abundant box annotations, which we argue could provide informative instance-specific saliency.

In this paper, we tackle semi-supervised instance segmentation problem, which includes both category-wise semi-supervision setting (i.e., partially supervised setting), and image-wise semi-supervision setting, i.e., only a small subset of images have masks. Those two settings focus on inter- and intra-class generalization, respectively. In other words, our model generalizes to both objects from unseen categories and novel objects from seen categories.

8.3 ShapeProp Model

In this section, we first revisit the detection based instance segmentation frameworks as we aim to improve their accuracy and generalization. We then introduce the proposed ShapeProp approach, starting with the process of statistically learning instance-specific saliency via multiple instance learning and followed by the design of propagating saliency to a well-generalized shape representation, referred to as Shape Activation. Finally, we discuss how to integrate Shape Activation to the instance segmentation frameworks. The overall architecture of ShapeProp is illustrated in Fig. 8.3.

Learning Saliency Propagation

Activating Saliency. Although a single box annotation does not specify the label for each pixel, they entail information on what an object looks like and provide one weak label for all pixels within the bounding box. We can still learn pixel-level label statistics from the weak supervision after looking at a large number of examples. This is also studied in the context of Multiple Instance Learning (MIL). MIL [131] is a form of weakly supervised learning where training instances are arranged in sets, called bags, and a label is provided for the entire bag instead of an individual instance. Note that in our context, we consider each box as a bag of pixels.

We first construct positive and negative bags from the box detections $\mathcal{B} = \{b_1, b_2, ..., b_N\}$ outputted by the instance segmentation framework where $b_i = (\hat{x}, \hat{y}, \hat{w}, \hat{h}, \hat{c})$ is the i-th detection and N is the number of detected objects in the image. $(\hat{x}, \hat{y}, \hat{w}, \hat{h})$ is the predicted coordinates of the bounding box and $0 \leq \hat{c} < K$ is the predicted class label from K predefined object categories, e.g., car, dog, and person. Let Y_i be the label of a bag $X_i = \{x_1, x_2, ..., x_{\hat{w}*\hat{h}}\}$, where a bag is defined as a set of pixels within the box detection b_i. Each instance (i.e., pixel) x_j corresponds to a binary label y_j which indicates whether the pixel is in the mask of object detected by b_i. We follow the standard MIL assumption that all negative bags contain only negative instances, and positive bags contain at least one positive instance:

$$Y_i = \begin{cases} +1 \; if \; \exists \, y_j \; : \; y_j = +1, \\ -1 \; if \; \forall \, y_j \; : \; y_j = -1. \end{cases} \tag{8.1}$$

For each category c, we use the box annotations G^c to partition $\mathcal{B}^c$ into positive bags $\mathcal{P}^c = \{p_1, p_2, ..., p_u\}$ and negative bags $\mathcal{N}^c = \{n_1, n_2, ..., n_v\}$ based on the Intersection over Union (IoU) and a threshold t (e.g., $t = 0.5$):

$$\begin{aligned} \mathcal{P}^c &= \{b_i \in \mathcal{B} \; if \; IoU(b_i, g) \; > \; t, \exists \, g \in G^c\}, \\ \mathcal{N}^c &= \{b_i \in \mathcal{B} \; if \; IoU(b_i, g) \; \leq \; t, \forall \, g \in G^c\}. \end{aligned} \tag{8.2}$$

We do this because positive sample must contain part of the object (i.e., positive bag) as the IoU is high. The negative sample is misaligned with the object (low IoU) thus all pixels are consider invalid (i.e., negative bag). Note that this is different from previous works that utilize MIL to discover class-specific responses in the image for semantic segmentation [144]. We extract object-specific responses for instance segmentation.

We build a weak learner $\mathcal{F}$, which is a lightweight module containing several conv-relu layers. For each sample p_i, $\mathcal{F}$ predicts a map $M \in \mathcal{R}^{h \times w}$ based on the corresponding region feature. In contrast to using pixel-level ground truth, we learn $\mathcal{F}$ using bag-level labels from boxes:

$$L = \sum_c \Big(\sum_{p_i \in \mathcal{P}^c} -\sigma(F(\omega(p_i), \theta)) + \sum_{n_j \in \mathcal{N}^c} +\sigma(F(\omega(n_j), \theta)) \Big), \tag{8.3}$$

where L is the loss, θ is the learnable parameters in $\mathcal{F}$, σ is a 2D aggregation operator, e.g., Avg2D, and ω is a region feature pooling operator, e.g., ROIAlign. The learning of $\mathcal{F}$ accumulates

several bags of instances to collect pixel-level information statistically, and the predicted M highlights salient regions in the image specific to each detected object and provides rich shape prior information to the subsequent mask head, as shown in Fig. 8.4.

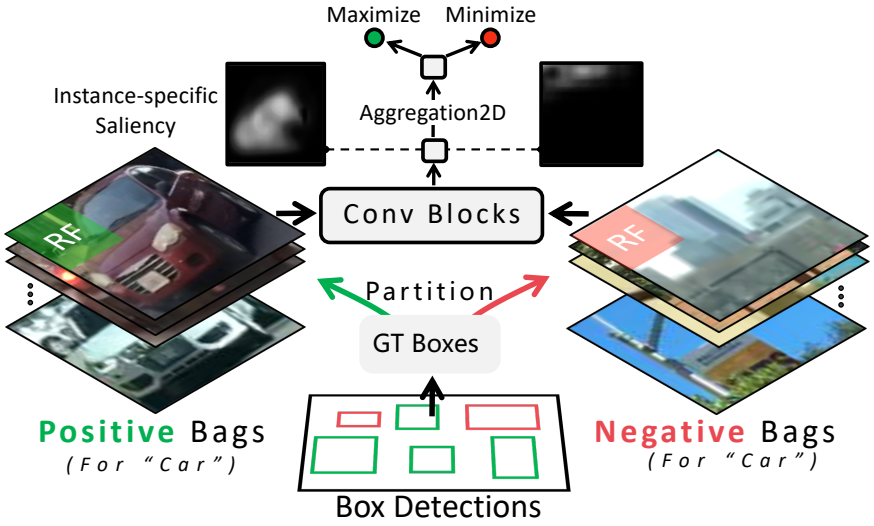


Figure 8.4: Multiple instance learning illustration.

Propagating Saliency. Object-specific region responses M obtained with MIL highlight only salient regions, we further design a propagation step to make use of the existing limited mask annotations and to recover full object extent from the incomplete object region responses. The motivation is to exploit the relation between pixels to estimate object shape from the salient regions obtained from boxes. Instead of considering it as a pixelwise classification problem, we learn how to propagate messages between deep pixels in the latent feature space. Learning to predict edges between nodes (i.e., deep pixels) for propagation instead of labels of nodes is more effective for the cases with only limited mask annotations. The reason lies in that pixelwise classification depends on high-level semantics and thus requires sufficient supervision to capture the diverse data distribution. Meanwhile, the relationship between pixels, i.e., whether two pixels belong to the same object, can be more reliably inferred with low-level semantics, e.g., similar color, and smooth texture. Intuitively speaking, a model does not need to recognize the semantic category of a banana but still can segment its extent by grouping pixels with a similar color.

The saliency propagation is implemented as a latent space message passing process. As illustrated in Fig. 8.5, we first use conv blocks (i.e., conv-relu layers) to encode the instance saliency map $M \in \mathcal{R}^{H \times W}$ to latent features $\tilde{M} \in \mathcal{R}^{C \times H \times W}$, where C is the channel dimension (e.g., 16). The encoding extracts features from M while making the subsequent message passing more robust to noise.

For each channel $\tilde{M}_i$, we consider deep pixels as nodes and learn to propagate the message between spatially adjacent nodes. Based on the appearance feature of b, we use conv blocks to predict propagation weights between nodes $W \in \mathbb{R}^{C \times (r \times r) \times (H \times W)}$, where r is a window size (e.g., 3). We then normalize and shuffle the learned propagation weights W to construct location-specific

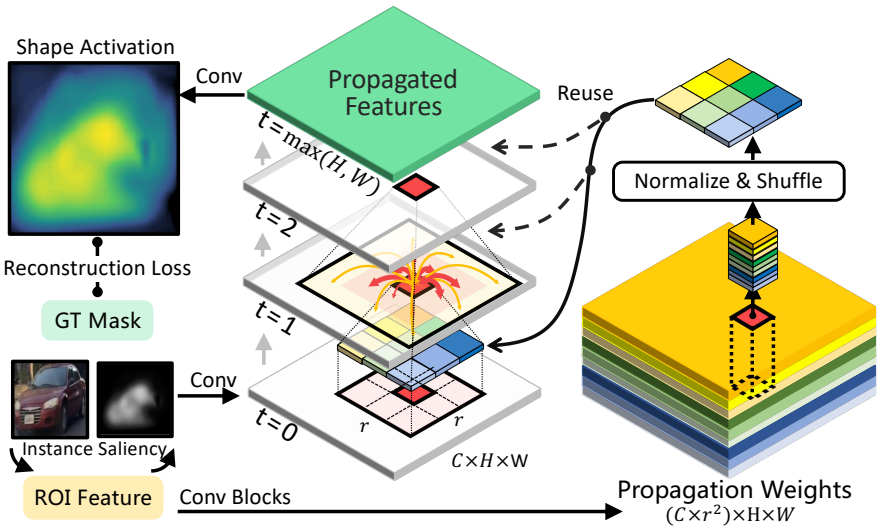


Figure 8.5: Saliency Propagation through message passing.

kernels $K_{i,u,v} \in \mathcal{R}^{r\times r}$, where i is the channel dimension and (u, v) is the spatial location and the $\sum_{(p,q)} K^{p,q}_{i,u,v} = 1$. The propagation is an iterative process. At each step, we use K to update the latent features:

$$\tilde{M}_i^{u,v}(t+1) = \sum_{(p,q)\in\mathcal{N}} \tilde{M}_i^{p,q}(t) \cdot K_{i,u,v}^{p-u+\frac{r}{2},q-v+\frac{r}{2}}, \tag{8.4}$$

where $\tilde{M}_i^{(u,v)}$ is the feature value at the (u, v) location of i-th channel, and $\mathcal{N}$ is the set of neighbor locations of (u, v). We iterate $max(H, W)$ steps to guarantee that messages can spread over all locations, and each node could absorb information from all other nodes. This iterative process does not introduce significant computation overhead as the spatial size of the region feature is typically very small (e.g., 14), and our efficient GPU implementation computes message passing for all detections simultaneously.

Finally, we use the convolution layer to decode the updated latent features back to a single channel map, referred to as Shape Activation, which combines shape information from both box and mask annotations and specify the extent of the object. Shape Activation serves as an intermediate shape representation that provides strong shape prior to subsequent mask prediction (i.e., mask head). During training, we use binary cross-entropy to compute the reconstruction loss against the ground truth mask. For the semi-supervised setting, we only calculate the loss in the small subset of images that have mask annotations. Note that learning of saliency propagation is in a class-agnostic manner that allows it to accumulate common knowledge from all existing masks and effectively generalize to novel categories.

Intergrating Shape Representation

We integrate the learned Shape Activation $\mathcal{S}$ into existing instance segmentation frameworks by concatenating it with the input region features before feeding into the mask head. The additional input channels provide strong prior information of objects' possible shapes; thus can significantly simplify the task of learning mask prediction, i.e., pixel-level classification, and allows mask head to focus on capturing fine-detailed information. Experimentally, we show that the Shape Activation guided segmentation not only significantly improves generalization ability (10.8% AP improvements on COCO's partially supervised setting), but also benefit segmentation quality even when mask annotations are sufficient (1.4% AP_{75} improvements on COCO's fully supervised setting).

8.4 Experiments

We evaluate the proposed ShapeProp method on popular instance segmentation benchmarks including COCO [118], PASCAL-VOC [49] and BDD100K [209]. We report standard metrics, that is, mask AP, AP50, AP75, and AP, for small/medium/large objects, following the evaluation protocol in previous works [75, 82, 105].

In Sec. 8.4, we test our method on the category-wise semi-supervision setting (i.e., "partially supervised" setting in previous literature [82, 105]). The significant improvements over baselines (10.8% AP) and the new state-of-the-art results indicate ShapeProp's capability to learn from limited masks and generalize to unseen categories. In Sec. 8.4, we benchmark on BDD100K, where only a subset of images have mask annotations. We augment both the single-stage and two-stages instance segmentation frameworks [56, 75] with ShapeProp and show consistent improvements over baselines. This indicates the effectiveness of ShapeProp's intra-class generalization. We further show that our approach can improve strong baselines that are trained using full mask annotations from the dataset. In Sec. 8.4, statistical analyses show that ShapeProp can learn high-quality shape representation. Finally, we perform ablation studies to investigate our model design further.

Generalization to unseen categories

Experiment setup. The experiments are set up following [83, 105]. We split the COCO17's 80 categories into VOC (20) vs. Non-VOC (60). The VOC categories are also present in PASCAL VOC [46]. At training time, models have access to the bounding boxes of all classes, but the masks only come from either VOC or Non-VOC categories. The performance upper bounds are set by the oracle models that have access to masks from all categories. In this section, our training set is COCO train2017, and the comparison with other methods is done on val2017 Non-VOC/VOC.

We build our models by plugging the ShapeProp module into the representative Mask R-CNN framework [75]. In order to evaluate across categories, we use the class-agnostic setting, which considers all object classes as the foreground class. We implement the model with two backbones, i.e., ResNet50-FPN and ResNet101-FPN [73, 117]. We use the same training parameters as the baseline.

Table 8.1: Semi-supervised performance on COCO2017

Backbone	Method	non-voc → voc: test on $B = \{\text{voc}\}$						voc → non-voc: test on $B = \{\text{non-voc}\}$					
		AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
R-50-FPN	Mask R-CNN trained w/ GrabCut [95]	19.5	46.2	14.2	4.7	15.9	32.0	19.5	39.2	17.0	6.5	20.9	34.3
	Mask^X R-CNN [83]	28.9	52.2	28.6	12.1	29.0	40.6	23.7	43.1	23.5	12.4	27.6	32.9
	Mask R-CNN (Baseline) [75]	23.9	42.9	23.5	11.6	24.3	33.7	19.2	36.4	18.4	11.5	23.3	24.4
	Mask R-CNN w/ ShapeProp (Ours)	**34.4**	**59.6**	**35.2**	**13.5**	**32.9**	**48.6**	**30.4**	**51.2**	**31.8**	**14.3**	**34.2**	**44.7**
	fully supervised (Oracle) [75]	37.5	63.1	38.9	15.1	36.0	53.1	33.0	53.7	35.0	15.1	37.0	49.9
R-101-FPN	Mask R-CNN trained w/ GrabCut [95]	19.6	46.1	14.3	5.1	16.0	32.4	19.7	39.7	17.0	6.4	21.2	35.8
	Mask^X R-CNN [82]	29.5	52.4	29.7	13.4	30.2	41.0	23.8	42.9	23.5	12.7	28.1	33.5
	ShapeMask [105]	33.3	56.9	34.3	17.1	38.1	45.4	30.2	49.3	31.5	16.1	38.2	28.4
	Mask R-CNN (Baseline) [75]	24.7	43.5	24.9	11.4	25.7	35.1	18.5	34.8	18.1	11.3	23.4	21.7
	Mask R-CNN w/ ShapeProp (Ours)	**35.5**	**60.5**	**36.7**	15.6	33.8	**50.3**	**31.9**	**52.1**	**33.7**	14.2	35.9	**46.5**
	fully supervised (Oracle) [75]	38.5	64.4	40.4	18.9	39.4	51.4	34.3	54.7	36.3	18.6	39.1	47.9

Numerical results. It can be seen in table 8.1, Mask R-CNN equipped with ShapeProp improves the baseline by a significant margin (e.g., 34.4% vs 23.9% for non-voc → voc and 30.4% vs 19.2% for voc → non-voc). Our model with ResNet50-FPN backbone outperforms the state-of-the-art ShapeMask [105] that is build on a stronger backbone (ResNet101-FPN. Our ShapeProp module improves segmentation by fully exploiting shape knowledge from the box and mask annotations; thus, it can also benefit from other advances in deep learning i.e., stronger backbone. Switching to ResNet101-FPN backbone improves yields a top result on these benchmarks (e.g., 2.2% AP higher than state of the art ShapeMask [105]).

Generalization with less data. To further validate the generalization ability of ShapeProp with less training data, we train the ResNet50-FPN based models on full categories using only 1/10, 1/50, 1/100 of the data. As can be seen in Fig. 8.6, our approach generalizes well down to 1/100 of the training data, and it consistently outperforms the baseline (Mask R-CNN without ShapeProp).

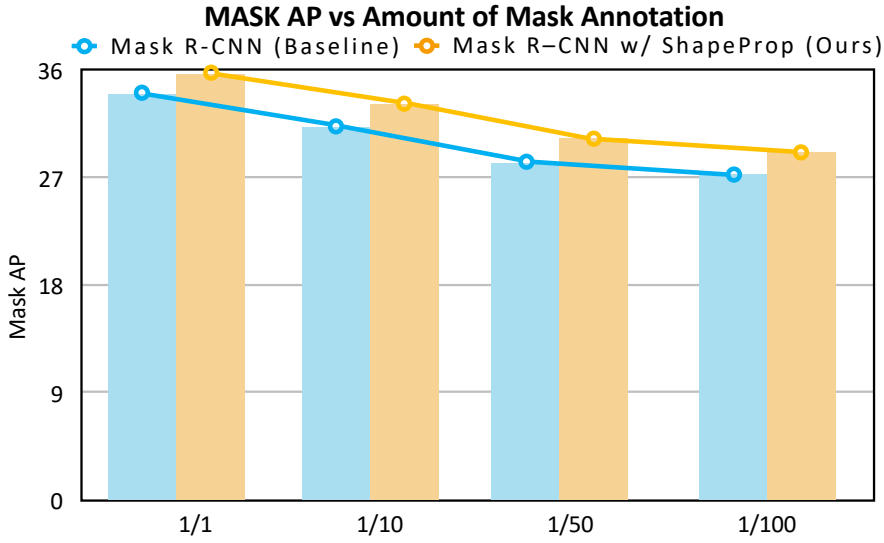


Figure 8.6: Generalization with less data.

Qualitative results. Fig. 8.7 gives qualitative examples from the non-voc → voc setting. It can be seen in the second row, the baseline method failed to segment the novel category "bicycle" as no masks for this category is available during training. However, adding ShapeProp to the baseline significantly improves the segmentation quality. It can also be seen in the bottom row that Mask R-CNN predicts a broken mask for the "cow" instance. In contrast, Mask R-CNN with ShapeProp model segment it correctly.

Figure 8.7: Visualization of Mask R-CNN (w/ or wo/ ShapeProp)'s results on novel categories.

Generalization to novel instances

Experiment setup. We further benchmark upon the BDD100K dataset [209], which is the largest and most diverse driving video dataset. Due to the extensive human efforts required for labeling detailed instance segmentation, only a subset of BDD100K provides mask annotations. The dataset fits naturally with our semi-supervised setting. We fuse the annotations of BDD100K's instance segmentation and detection to build a data contains 67k images with box annotations, among which 7k images have mask annotations. We test models on BDD100K's val. set (1k images).

We build our models by plugging the proposed ShapeProp module into two representative detection based instance segmentation frameworks, i.e., Mask R-CNN (two-stages method), and RetinaMask (single-stage method). We compare with the joint learning version of Mask R-CNN. It learns from all images and only compute the loss for segmentation head when mask annotation is available. We also compare with Grabcut Mask R-CNN [95] and Progressive Mask R-CNN, which

use Grabcut post-processing and the pretrained annotator model to obtain pseudo masks from box annotations. All models are based on the ResNet-50 FPN backbone and are trained via standard SGD optimization with LR 0.01 and batch size 12.

Numerical results. As shown in Tab. 8.2, the mask APs for both single-stage and two-stage baselines are significantly improved when more box annotations are available, i.e., Mask RCNN (24.5 vs 21.6) and RetinaMask (24.4 vs 20.0). However, the model trained with Grabcut pseudo masks performs even worse than the baseline, indicating the shape representation quality from Grabcut is not good enough. It can be seen in Tab. 8.2, equipping the proposed ShapeProp module bridges the learning of box and mask and further improves the model's segmentation ability (26.2% vs 24.5%). This shows ShapeProp can effectively exploit shape prior hidden inside the annotations to enhance the quality of segmentation. Moreover, it can be seen in Tab. 8.3, equipping ShapeProp also improves the strong baselines trained with fully supervised setting where all masks are available during training. This indicates that fully exploiting annotations can improve segmentation quality even when masks are sufficient.

Table 8.2: Semi-supervised evaluation on BDD100K.

Training data (# annotations)	Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	Comment
BDD100k-Instseg (7k w/ masks, 7k w/ boxes)	RetinaMask [56]	20.0	37.6	18.1	7.6	27.0	42.3	Single-stage baseline
	Mask R-CNN [75]	21.6	40.5	20.5	9.3	28.8	45.4	Two-stage baseline
BDD100k (7k w/ masks, 67k w/ boxes)	RetinaMask [56]	24.4	44.7	22.5	9.8	32.6	51.5	Joint training
	RetinaMask w/ ShapeProp	**26.1**	**46.8**	**24.9**	**10.7**	**34.7**	**56.3**	ShapeProp augmented (Ours)
	Mask R-CNN [75]	24.5	45.4	21.6	10.1	33.1	48.3	Joint training
	Grabcut Mask R-CNN [95]	21.0	41.0	19.5	8.3	27.0	40.7	Grabcut limited mask
	Progressive Mask R-CNN	24.8	45.4	23.0	10.0	33.0	52.7	Progressive learning
	Mask R-CNN w/ ShapeProp	**26.2**	**48.4**	**23.5**	**11.4**	**34.2**	**53.0**	ShapeProp augmented (Ours)

Table 8.3: Comparison under fully supervised setting.

Dataset	Method	AP	AP_{50}	AP_{75}
VOC12	Mask R-CNN [75]	29.7	54.1	29.6
(1k images)	**ShapeProp (Ours)**	**30.6**	**53.2**	**32.0**
BDD100K	Mask R-CNN [75]	21.6	40.5	20.5
(7k images)	**ShapeProp (Ours)**	**22.7**	**42.2**	**21.8**
COCO17	Mask R-CNN [75]	34.2	56.4	36.7
(120k images)	**ShapeProp (Ours)**	**35.7**	**56.9**	**38.2**

Inference time. The proposed ShapeProp module is a lightweight module built on top of the convolution blocks. The message passing operation for propagating saliency is efficiently implemented as matrix dot production. Therefore, overall ShapeProp module does not introduce heavy computation overhead (0.35 vs 0.39 s/img on 2080 Ti).

Qualitative results. In Fig. 8.8, we visualize examples of box detection, instance saliency, shape activation, and mask predictions from models with or without ShapeProp. It can be seen from the left sample, multiple cars are occluded in the detected region, and the baseline model failed to segment the correct one. In contrast, ShapeProp find the object-specific salient parts and further

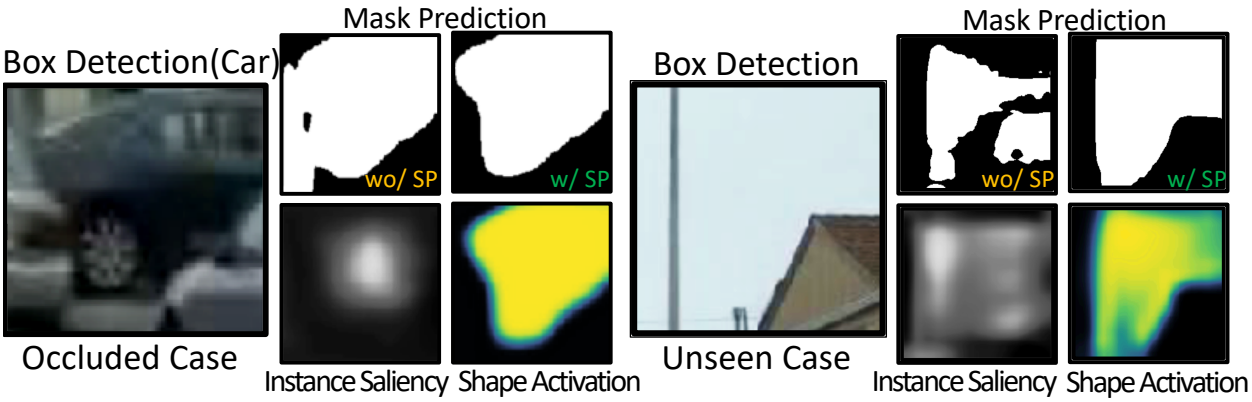


Figure 8.8: Visualization of samples in BDD100K datasets.

predict propagation weights conditional on the visual appearance to propagate salient regions into a high-quality shape activation. Based on the shape activation's strong shape prior, the mask prediction of our model is significantly improved. The right sample gave a case when detection failed, i.e., a background region is mis-detected as a car. Despite this background region is unseen from the training data, our model gives a more reasonable shape representation and mask prediction while the baseline outputs a mask with broken pieces. Note the model is designed to estimate the shape of the centered salient "object" in the box. This further demonstrates the strong generalization of our approach.

Analysis and ablation studies

Statistical Analysis. We analyze the quality of the extracted shape prior with respect to object size, demonstrating that our approach can effectively extract shape prior from the box and mask annotations. Shape activation is assigned to GT masks and judged by measuring the best overlap metric. To be considered a perfect shape activation that completely coincide with a GT mask, the IoU between the predicted shape activation M and GT masks $\mathcal{T}$ must be close to 100% as computed using the metric $\max_{\theta, T_i \in \mathcal{T}} \frac{area(f_b(M,\theta) \cap T_i)}{area(f_b(M,\theta) \cup T_i)}$, where the function $f_b(M, \theta) = M \geq \theta$ produces the best matching binary instance masks based on the probabilistic shape activation over a set of threshold values $\theta \in (0, 1)$. Note that this metric is robust to the absolute value range of prediction and is suitable for evaluating probabilistic activation maps.

We visualize the density of the best overlap for instance-specific saliency and the shape activation obtained by message passing to see whether the activation can cover object instances of different sizes.

Fig. 8.9 (left) shows that saliency samples clustered in the area where the best overlap value is around 60% and failed to cover large objects. In contrast, in Fig. 8.9 (right), most of the data points have relatively high best overlap IoUs and perform very well on both small and large objects. It can be seen that the message-passing design in our ShapeProp significantly improves the quality

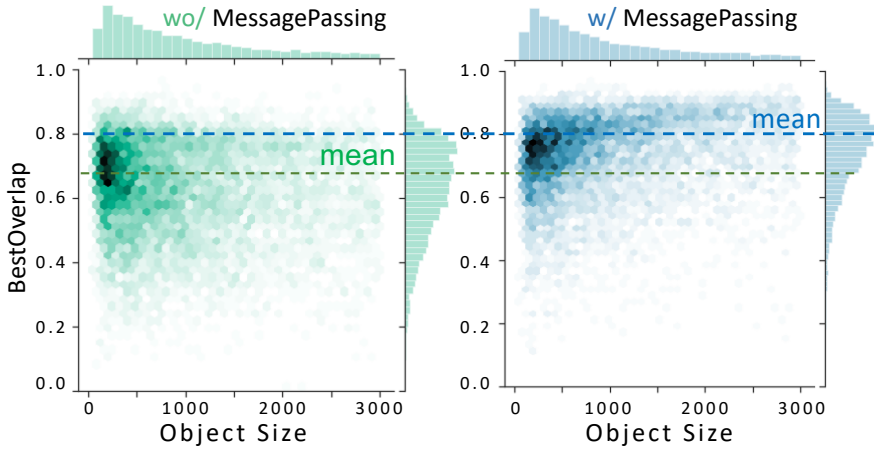


Figure 8.9: Shape activation accuracy.

Table 8.4: Ablation studies of model design on BDD100K.

Setting	AP	AP_{50}	AP_{75}
baseline	21.6	40.5	20.5
baseline + More head params	21.4	40.3	19.5
baseline + ShapeProp(Channel-agnostic)	22.4	42.0	20.5
baseline + ShapeProp	**22.7**	**42.2**	**21.8**
baseline + More boxes	24.5	45.4	21.6
baseline + More boxes + ShapeProp	**26.2**	**48.4**	**23.5**

of the extracted shape prior. The underlying reason is that multiple instance learning typically captures the salient regions of the instance; however, it failed to recover the full extent. The message passing utilizes the limited number of ground truth masks to learn how to refine the saliency map and recover such extent in a class agnostic and well-generalized manner, thus considerably improve the quality.

Ablation Study. We perform ablation studies on the BDD100K dataset to validate some specific designs of our method, Tab. 8.4. The baseline model is a Mask R-CNN with a ResNet50-FPN backbone. The first to sixth rows correspond to models trained on the instseg split of BDD100K, which contains 7k images for training, and all instances in the training data have both box and mask annotations. The last two rows are trained with the overall BDD100K data, including its official detection and instseg split, which has 67k images in total among it; only 7k images have masks. The setting of the second row increases the number of channels used in Mask R-CNN's mask head to make the number of learnable parameters be the same as the ShapeProp-augmented counterpart. The AP of this setting is even slightly worse than the baseline, which validates that the gain of ShapeProp doesn't come from the increasing of model parameters. Comparing the fifth to the third rows, the better AP indicates that predicting propagation weights for each channel of the latent space instead of sharing the same weights for all channels can lead to performance improvements. The sixth row, which uses additional bounding box annotations from the BDD100K

detection split has 2.9% AP improvements (24.5% vs 21.6%). Meanwhile, the last row, which is augmented by the ShapeProp module, has 3.5% AP gains (26.2 % vs 22.7%) over the baseline and 4.6% AP improvements over the plain baseline of the first row. This clearly shows that the ShapeProp module can better utilize the hidden shape prior to the additional box annotations to benefit the segmentation quality.

8.5 Chapter Summary

We developed a lightweight network module, ShapeProp, which can be plugged into existing instance segmentation frameworks to tackle the challenging semi-supervised instance segmentation task. ShapeProp extracts a well-generalized shape representation from the joint learning of the abundant yet coarse-grind box supervision and the fine-detailed yet limited amount of mask annotations. Such shape representation hypothesis possible object shape and specify detailed instance boundaries that provide strong shape prior to the subsequent mask prediction, which allows the learning of strong instance segmentation model based on limited mask annotations. We extensively test ShapeProp on popular benchmarks, including COCO, PASCAL VOC, and BDD100K. The results indicate that ShapeProp-augmented frameworks consistently outperform baseline by a significant margin, establishing states-of-the-art for semi-supervised instance segmentation.

Chapter 9

The Road Ahead

Recent years have witnessed rapid progress in computer vision research. Computer vision research, especially deep learning methods, largely relies on large scale data and labels and massive computation, which are often infeasible in real-world applications. Also, distribution shifts between training and inference are inevitable when using learning techniques in actual systems. Therefore, learning with limited data, annotation, and computation has become increasingly critical in machine learning and computer vision.

In this thesis, we discussed the design of dynamic neural networks to address the aforementioned changes in intelligent systems. Dynamic neural networks leverage test-time information to adjust the network structures and parameters at prediction time. A dynamic neural network may conditionally activate parts of the network based on the input to enable efficient inference. A specialized network can be generated for novel tasks at prediction time through meta-learning the connection between the prediction task and network weights. Dynamic representations with deep neural networks are close to the dynamic and adaptive nature of human brains, where different sets of neurons are activated and composed to address various tasks.

We've shown that dynamic neural networks achieved competitive performance in both few-shot learning and fast prediction settings, improving learning and inference efficiency. Dynamic neural networks can also build a unified representation for broader learning paradigms such as multi-task learning, continual learning, etc. To move forward, there are several directions we would like to pursue, as discussed below.

9.1 Heterogeneous Multitask Learning

One of the hallmarks of human brains is the versatility of performing heterogeneous tasks simultaneously. Similarly, we hope to enable the multi-task learning capabilities for machine learning models. For example, a reliable self-driving system requires learning heterogeneous tasks in perception, planning, and control. In BDD100K [209], we have constructed a large scale driving dataset with full annotations of ten heterogeneous tasks and established a test-bed for heterogeneous multi-task learning. Dynamic neural networks are suitable for heterogeneous multi-task learning due to condi-

tional activation. Building a unified framework that can tackle various tasks simultaneously is a promising direction to pursue.

Jointly learning tasks with various structures

Recent work [209, 210] has introduced large scale heterogeneous multitask learning benchmarks for visual tasks. Taskonomy [210] introduces a large scale dataset containing 4 million synthetic images of indoor scenes from about 600 buildings; every image has an annotation for every task among 26 tasks in both 2D and 3D spaces. Taskonomy learns the transfer score between the source and target tasks to characterize the relationship between various tasks. However, Taskonomy does not provide a study on a unified framework for learning all tasks together.

BDD100K [209] provides annotations of 10 heterogeneous tasks on real-world driving data, ranging from image-level tasks (e.g., image tagging) to pixel level tasks (e.g., semantic segmentation) and object-level tasks (e.g., object detection, instance segmentation, multiple object tracking) as well as their domain adaption problem. In BDD100K, researchers have provided studies to cascade and stack four tasks: object detection, instance segmentation, multiple object tracking, and multiple object segmentation tracking. The multiple object segmentation tracking is the most challenging task among the 10 tasks, which requires the model to detect objects in adjacent frames, associate the identical objects along time and then predict the instance mask of each object. Researchers find by jointly training four companion tasks together; the resulting model performs better than learning the individual task. The study in the original BDD100K work is focused on adding tasks with cheaper annotation cost to boost the performance of complicated tasks.

We would like to study how to build a unified model framework that can tackle multiple tasks simultaneously. As shown in Taskonomy, some tasks have positive transfer scores, which can be learned together. In contrast, some other tasks may be conflicting with each other and thus should avoid architecture sharing. We could design a transfer-aware representation that avoids conflicting tasks to share the same part of the network. The network structure is dependent based on the combination of input tasks.

9.2 Continuous Adaptation in Changing Environment

We live in a dynamic world. Tackling out of distribution data and distribution shift at prediction time is crucial for robust machine learning systems. In this thesis, we discussed approaches such as *test-time fine-tuning* and *task-aware weight generation*, which allows models to adapt to novel prediction tasks inference. It is worthwhile to design test-time adaption approaches to enable continuous adaption in a changing environment along this line of work. Several problems, such as incremental learning new concepts and tasks without forgetting and uncertainty aware decision making, remain understudied and are worth further exploration.

Continuous domain adaption with self-supervision.

Imagine you are traveling from one place to another, the prediction environment (e.g., time of day, scene structure, lightning) keeps changing. This can be categorized as a continuous domain adaptation problem without labels of the changing domains. We may ask several questions here to tackle this problem.

- How to extract relevant information from the current input data to help model adaptation?
- How to utilize the test-time information from the input data?
- How to make model adaptation fast and in real-time, ideally?
- As deep learning models often suffer from catastrophic forgetting, how to avoid performance drop in the past domains?

One potential way to address this problem is to introduce auxiliary self-supervised tasks to the prediction network for the specific downstream task. The self-supervised task and the prediction network shares the re-configurable part of the prediction network. When adapting to the new environment, we could update the self-supervised task on the current input data through a few gradient descent steps to update the re-configurable part of the prediction network and improve the downstream task in the current domain. The hypothesis here is that the supervised-learning task and the downstream prediction task have a shareable part, and thus improving one task can lead to a positive effect on the other.

9.3 Systems with Conditional Computation

The prevailing computing units, such as GPUs, benefit from a massive parallel structure while remaining challenging to inference adaptive models with dynamic execution paths and changing network architecture and parameters. How do we design effective system and hardware support for conditional computation?

Algorithm and Hardware Co-designs on Embedded Devices

There is an increasing interest in new systems and chip designs for artificial intelligence technologies. We believe enabling conditional computation from both algorithmic and system perspectives is an open direction to go, which could benefit efficiency and adaptivity of intelligent systems, such as autonomous vehicles and robotics systems. Recent work [43] proposes an algorithm-hardware co-design solution for deformable convolution on FPGAs. Regular convolutions process a fixed grid of pixels across all the spatial locations in an image. In contrast, dynamic deformable convolution may access arbitrary pixels in the image, and the access pattern is input-dependent and varies per spatial location. These properties lead to inefficient memory accesses of inputs with existing hardware. In the proposed CoDeNet, researchers show that the co-designing of algorithms and architecture

optimization strategies achieves significant speedups on FPGAs with negligible accuracy. Along this line of work, we believe interdisciplinary efforts are needed to release the ultimate power of dynamic neural networks in real-world applications.

9.4 Technical Acknowledgements

This thesis is based on the following publications:

- Chapter 2 is based on a joint work with Fisher Yu, Zi-Yi Dou, Trevor Darrell and Joseph E. Gonzalez. It was published as a conference paper entitled *SkipNet: Learning Dynamic Routing in Convolutional Networks* at ECCV 2018.
- Chapter 3 is based on a joint work with Fisher Yu, Lisa Dunlap, Yi-an Ma, Azalia Mirhoseini, Trevor Darrell, Joseph E. Gonzalez. It was published as a conference paper entitled *Deep Mixture of Experts via Shallow Embedding* at UAI 2019.
- Chapter 4 is based on a joint work with Yujia Luo, Daniel Crankshaw, Alexey Tumanov, Fisher Yu, Joseph E. Gonzalez. It was published as a conference paper entitled *IDK Cascades: Fast Deep Learning by Learning not to Overthink* at UAI 2018.
- Chapter 5 is based on a joint work with Fisher Yu, Ruth Wang, Trevor Darrell, Joseph E. Gonzalez. It was published as a conference paper entitled *TAFE-Net: Task-Aware Feature Embeddings for Efficient Learning and Inference* at CVPR 2019.
- Chapter 6 is based on a joint work with Bingyi Kang, Zhuang Liu, Fisher Yu, Jiashi Feng, Trevor Darrell. It was published as a conference paper entitled *Few-shot Object Detection via Feature Reweighting* at ICCV 2019.
- Chapter 7 is based on a joint work with Thomas E. Huang, Trevor Darrell, Joseph E. Gonzalez, Fisher Yu. It was published as a conference paper entitled *Frustratingly Simple Few-Shot Object Detection* at ICML 2020.
- Chapter 8 is based on a joint work with Yanzhao Zhou, Jianbin Jiao, Trevor Darrell, Fisher Yu. It was published as a conference paper entitled *Learning Saliency Propagation for Semi-Supervised Instance Segmentation* at CVPR 2020.

This thesis work was supported by Berkeley AI Research, RISE Lab and Berkeley DeepDrive. In addition to NSF CISE Expeditions Award CCF-1730628, this research is supported by gifts from Alibaba, Amazon Web Services, Ant Financial, Arm, CapitalOne, Ericsson, Facebook, Google, Huawei, Intel, Microsoft, Nvidia, Scotiabank, Splunk and VMware.

Bibliography

[1] Karim Ahmed, Mohammad Haris Baig, and Lorenzo Torresani. "Network of experts for large-scale image categorization". In: *European Conference on Computer Vision*. Springer. 2016, pp. 516–532.

[2] Zeynep Akata et al. "Evaluation of output embeddings for fine-grained image classification". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 2927–2936.

[3] Zeynep Akata et al. "Label-embedding for image classification". In: *IEEE transactions on pattern analysis and machine intelligence* 38.7 (2016), pp. 1425–1438.

[4] Anelia Angelova et al. "Real-Time Pedestrian Detection with Deep Network Cascades." In: *BMVC*. 2015, pp. 32–1.

[5] Anurag Arnab and Philip HS Torr. "Pixelwise instance segmentation with a dynamically instantiated network". In: *CVPR*. Vol. 1. 2. 2017, p. 5.

[6] Min Bai and Raquel Urtasun. "Deep watershed transform for instance segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 5221–5229.

[7] Ankan Bansal et al. "Zero-Shot Object Detection". In: *arXiv preprint arXiv:1804.04340* (2018).

[8] Emmanuel Bengio et al. "Conditional computation in neural networks for faster models". In: *arXiv preprint arXiv:1511.06297* (2015).

[9] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. "Estimating or propagating gradients through stochastic neurons for conditional computation". In: *arXiv preprint arXiv:1308.3432* (2013).

[10] Luca Bertinetto et al. "Learning feed-forward one-shot learners". In: *Advances in Neural Information Processing Systems*. 2016, pp. 523–531.

[11] Hakan Bilen and Andrea Vedaldi. "Weakly supervised deep detection networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 2846–2854.

[12] Tolga Bolukbasi et al. "Adaptive neural networks for efficient inference". In: *Proceedings of the 34th International Conference on Machine Learning*. 2017, pp. 527–536.

[13] Zhaowei Cai, Mohammad Saberian, and Nuno Vasconcelos. “Learning Complexity-Aware Cascades for Deep Pedestrian Detection”. In: *ICCV*. 2015.

[14] William Chan et al. “Listen, attend and spell”. In: *arXiv preprint arXiv:1508.01211* (2015).

[15] Soravit Changpinyo et al. “Synthesized Classifiers for Zero-Shot Learning”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[16] Hao Chen et al. “LSTD: A Low-Shot Transfer Detector for Object Detection”. In: *arXiv preprint arXiv:1803.01529* (2018).

[17] Liang-Chieh Chen et al. “MaskLab: Instance segmentation by refining object detection with semantic and direction features”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[18] Liang-Chieh Chen et al. “Masklab: Instance segmentation by refining object detection with semantic and direction features”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 4013–4022.

[19] Long Chen et al. “Zero-shot visual recognition using semantics-preserving adversarial embedding networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 1043–1052.

[20] Wei-Yu Chen et al. “A closer look at few-shot classification”. In: *arXiv preprint arXiv:1904.04232* (2019).

[21] Zhiyuan Chen and Bing Liu. “Lifelong machine learning”. In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 12.3 (2018), pp. 1–207.

[22] KyungHyun Cho and Yoshua Bengio. “Exponentially increasing the capacity-to-computation ratio for conditional computation in deep learning”. In: *arXiv preprint arXiv:1406.7362* (2014).

[23] Nadav Cohen, Or Sharir, and Amnon Shashua. “On the expressive power of deep learning: A tensor analysis”. In: *Conference on Learning Theory*. 2016, pp. 698–728.

[24] Ronan Collobert, Samy Bengio, and Yoshua Bengio. “A parallel mixture of SVMs for very large scale problems”. In: *Advances in Neural Information Processing Systems*. 2002, pp. 633–640.

[25] Ronan Collobert, Yoshua Bengio, and Samy Bengio. “Scaling large learning problems with hard parallel mixtures”. In: *International Journal of pattern recognition and artificial intelligence* 17.03 (2003), pp. 349–365.

[26] Marius Cordts et al. “The Cityscapes Dataset for Semantic Urban Scene Understanding”. In: *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[27] Daniel Crankshaw et al. “Clipper: A Low-Latency Online Prediction Serving System.” In: *NSDI*. 2017, pp. 613–627.

[28] Jifeng Dai, Kaiming He, and Jian Sun. “Instance-Aware Semantic Segmentation via Multi-task Network Cascades”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 3150–3158.

[29] Jifeng Dai et al. “R-fcn: Object detection via region-based fully convolutional networks”. In: *Advances in neural information processing systems*. 2016, pp. 379–387.

[30] Yann N Dauphin et al. “Language modeling with gated convolutional networks”. In: *arXiv preprint arXiv:1612.08083* (2016).

[31] Bert De Brabandere, Davy Neven, and Luc Van Gool. “Semantic instance segmentation with a discriminative loss function”. In: *arXiv preprint arXiv:1708.02551* (2017).

[32] Harm De Vries et al. “Modulating early visual processing by language”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 6594–6604.

[33] Misha Denil et al. “Predicting Parameters in Deep Learning”. In: *Advances in Neural Information Processing Systems 26*. Ed. by C.j.c. Burges et al. 2013, pp. 2148–2156.

[34] Misha Denil et al. “Predicting parameters in deep learning”. In: *Advances in neural information processing systems*. 2013, pp. 2148–2156.

[35] Emily Denton et al. “Exploiting Linear Structure Within Convolutional Networks for Efficient Evaluation”. In: *Proceedings of the 27th International Conference on Neural Information Processing Systems*. NIPS’14. Montreal, Canada: MIT Press, 2014, pp. 1269–1277.

[36] Guneet S Dhillon et al. “A baseline for few-shot image classification”. In: *arXiv preprint arXiv:1909.02729* (2019).

[37] Bhuwan Dhingra et al. “Towards end-to-end reinforcement learning of dialogue agents for information access”. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Vol. 1. 2017, pp. 484–495.

[38] Ali Diba et al. “Weakly Supervised Cascaded Convolutional Networks.” In: *CVPR*. 2017.

[39] Jeff Donahue et al. “Decaf: A deep convolutional activation feature for generic visual recognition”. In: *International conference on machine learning*. 2014, pp. 647–655.

[40] Jeffrey Donahue et al. “Long-term recurrent convolutional networks for visual recognition and description”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 2625–2634.

[41] Xuanyi Dong et al. “Few-Example Object Detection with Model Communication”. In: *arXiv preprint arXiv:1706.08249* (2017).

[42] Xuanyi Dong et al. “More is less: A more complicated network with less inference complexity”. In: *arXiv preprint arXiv:1703.08651* (2017).

[43] Zhen Dong et al. “CoDeNet: Algorithm-hardware Co-design for Deformable Convolution”. In: *arXiv preprint arXiv:2006.08357* (2020).

[44] Matthijs Douze et al. “Low-shot learning with large-scale diffusion”. In: *Computer Vision and Pattern Recognition (CVPR)*. 2018.

[45] David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever. "Learning factored representations in a deep mixture of experts". In: *International Conference on Learning Representations Workshop* (2014).

[46] Dumitru Erhan et al. "Visualizing higher-layer features of a deep network". In: *University of Montreal* 1341.3 (2009), p. 1.

[47] M. Everingham et al. *The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results*. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[48] Mark Everingham et al. "The pascal visual object classes (voc) challenge". In: *International journal of computer vision* 88.2 (2010), pp. 303–338.

[49] Mark Everingham et al. "The Pascal Visual Object Classes Challenge: A Retrospective". In: *International Journal of Computer Vision (IJCV)* 111.1 (2015), pp. 98–136.

[50] Mark Everingham et al. "The pascal visual object classes challenge: A retrospective". In: *International journal of computer vision* 111.1 (2015), pp. 98–136.

[51] Ali Farhadi et al. "Describing objects by their attributes". In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE. 2009, pp. 1778–1785.

[52] Michael Figurnov et al. "Spatially Adaptive Computation Time for Residual Networks". In: *The IEEE Conference on Computer Vision and Pattern Recognition*. July 2017.

[53] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-agnostic meta-learning for fast adaptation of deep networks". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, pp. 1126–1135.

[54] Chelsea Finn et al. "One-Shot Visual Imitation Learning via Meta-Learning". In: *Conference on Robot Learning*. 2017, pp. 357–368.

[55] Andrea Frome et al. "DeViSe: A deep visual-semantic embedding model". In: *Advances in neural information processing systems*. 2013, pp. 2121–2129.

[56] Cheng-Yang Fu, Mykhailo Shvets, and Alexander C Berg. "RetinaMask: Learning to predict masks improves state-of-the-art single-shot detection for free". In: *arXiv preprint arXiv:1901.03353* (2019).

[57] Spyros Gidaris and Nikos Komodakis. "Dynamic few-shot visual learning without forgetting". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 4367–4375.

[58] Ross Girshick. "Fast r-cnn". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1440–1448.

[59] Ross Girshick et al. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

[60] Yunchao Gong et al. "Compressing deep convolutional networks using vector quantization". In: *arXiv preprint arXiv:1412.6115* (2014).

[61] Ian Goodfellow et al. “Generative adversarial nets”. In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[62] Ian J Goodfellow et al. “Maxout networks”. In: *Proceedings of the 30th International Conference on Machine Learning*. 2013, pp. III–1319.

[63] Alex Graves. “Adaptive computation time for recurrent neural networks”. In: *arXiv preprint arXiv:1603.08983* (2016).

[64] Sam Gross, Marc’Aurelio Ranzato, and Arthur Szlam. “Hard Mixtures of Experts for Large Scale Weakly Supervised Vision”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 6865–6873.

[65] Jiaqi Guan et al. “Energy-efficient Amortized Inference with Cascaded Deep Classifiers”. In: *arXiv preprint arXiv:1710.03368* (2017).

[66] Agrim Gupta, Piotr Dollar, and Ross Girshick. “LVIS: A dataset for large vocabulary instance segmentation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 5356–5364.

[67] David Ha, Andrew Dai, and Quoc V Le. “Hypernetworks”. In: *arXiv preprint arXiv:1609.09106* (2016).

[68] Song Han, Huizi Mao, and William J. Dally. “Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding”. In: *ICLR* (2015).

[69] Song Han, Huizi Mao, and William J. Dally. “Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding”. In: *ICLR* (2015).

[70] Bharath Hariharan and Ross Girshick. “Low-Shot Visual Recognition by Shrinking and Hallucinating Features”. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE. 2017, pp. 3037–3046.

[71] Bharath Hariharan et al. “Simultaneous detection and segmentation”. In: *European Conference on Computer Vision*. Springer. 2014, pp. 297–312.

[72] Zeeshan Hayder, Xuming He, and Mathieu Salzmann. “Boundary-aware instance segmentation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 5696–5704.

[73] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[74] Kaiming He et al. “Identity mappings in deep residual networks”. In: *European Conference on Computer Vision*. 2016, pp. 630–645.

[75] Kaiming He et al. “Mask r-cnn”. In: *Computer Vision (ICCV), 2017 IEEE International Conference on*. IEEE. 2017, pp. 2980–2988.

[76] Kaiming He et al. “Spatial pyramid pooling in deep convolutional networks for visual recognition”. In: *European conference on computer vision*. Springer. 2014, pp. 346–361.

[77] Kaiming He et al. "Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2117–2125.

[78] Yihui He, Xiangyu Zhang, and Jian Sun. "Channel pruning for accelerating very deep neural networks". In: *International Conference on Computer Vision (ICCV)*. Vol. 2. 2017, p. 6.

[79] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. "Distilling the knowledge in a neural network". In: *arXiv preprint arXiv:1503.02531* (2015).

[80] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[81] Judy Hoffman et al. "CyCADA: Cycle Consistent Adversarial Domain Adaptation". In: *International Conference on Machine Learning (ICML)*. 2018.

[82] Ronghang Hu et al. "Learning to Segment Every Thing". In: *CVPR*. 2018.

[83] Ronghang Hu et al. "Learning to segment every thing". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[84] Gao Huang et al. "Deep networks with stochastic depth". In: *European Conference on Computer Vision*. 2016, pp. 646–661.

[85] Gao Huang et al. "Multi-scale dense convolutional networks for efficient prediction". In: *arXiv preprint arXiv:1703.09844* (2017).

[86] Zehao Huang and Naiyan Wang. "Data-Driven Sparse Structure Selection for Deep Neural Networks". In: *arXiv preprint arXiv:1707.01213* (2017).

[87] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *ICML*. 2015.

[88] Phillip Isola, Joseph J. Lim, and Edward H. Adelson. "Discovering States and Transformations in Image Collections". In: *CVPR*. 2015.

[89] Robert A Jacobs et al. "Adaptive mixtures of local experts". In: *Neural computation* 3.1 (1991), pp. 79–87.

[90] Eric Jang, Shixiang Gu, and Ben Poole. "Categorical reparameterization with gumbel-softmax". In: *arXiv preprint arXiv:1611.01144* (2016).

[91] Michael I Jordan and Robert A Jacobs. "Hierarchical mixtures of experts and the EM algorithm". In: *Neural computation* 6.2 (1994), pp. 181–214.

[92] Bingyi Kang et al. "Few-shot Object Detection via Feature Reweighting". In: *ICCV*. 2019.

[93] Daniel Kang et al. "Optimizing Deep CNN-Based Queries over Video Streams at Scale". In: *arXiv preprint arXiv:1703.02529* (2017).

[94] Fereshte Khani, Martin C Rinard, and Percy Liang. "Unanimous Prediction for 100% Precision with Application to Learning Semantic Mappings." In: *ACL* (2016).

[95] Anna Khoreva et al. “Simple does it: Weakly supervised instance and semantic segmentation”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 1665–1674.

[96] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[97] Alexander Kirillov et al. “Instancecut: from edges to instances with multicut”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 5008–5017.

[98] Gregory Koch. “Siamese neural networks for one-shot image recognition”. In: 2015.

[99] Shu Kong and Charless C Fowlkes. “Recurrent pixel embedding for instance grouping”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 9018–9028.

[100] Alex Krizhevsky. *Learning multiple layers of features from tiny images*. Tech. rep. 2009.

[101] Alex Krizhevsky and Geoffrey Hinton. “Learning multiple layers of features from tiny images”. In: (2009).

[102] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. *The CIFAR-10 dataset*. 2014.

[103] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[104] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[105] Weicheng Kuo et al. “ShapeMask: Learning to Segment Novel Objects by Refining Shape Priors”. In: *arXiv preprint arXiv:1904.03239* (2019).

[106] Brenden M Lake, Ruslan R Salakhutdinov, and Josh Tenenbaum. “One-shot learning by inverting a compositional causal process”. In: *Advances in neural information processing systems*. 2013, pp. 2526–2534.

[107] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. “Attribute-based classification for zero-shot visual object categorization”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.3 (2014), pp. 453–465.

[108] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. “Learning to detect unseen object classes by between-class attribute transfer”. In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE. 2009, pp. 951–958.

[109] Barbara Landau, Linda B Smith, and Susan S Jones. “The importance of shape in early lexical learning”. In: *Cognitive development* 3.3 (1988), pp. 299–321.

[110] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[111] Chen-Yu Lee, Patrick W Gallagher, and Zhuowen Tu. "Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree". In: *Artificial Intelligence and Statistics*. 2016, pp. 464–472.

[112] Fei-Fei Li, Rob Fergus, and Pietro Perona. "One-shot learning of object categories". In: *IEEE transactions on pattern analysis and machine intelligence* 28.4 (2006), pp. 594–611.

[113] Hao Li et al. "Pruning filters for efficient convnets". In: *arXiv preprint arXiv:1608.08710* (2016).

[114] Yi Li et al. "Fully convolutional instance-aware semantic segmentation". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 4438–4446.

[115] Ji Lin et al. "Runtime Neural Pruning". In: *Advances in Neural Information Processing Systems*. 2017, pp. 2178–2188.

[116] Min Lin, Qiang Chen, and Shuicheng Yan. "Network in network". In: *arXiv preprint arXiv:1312.4400* (2013).

[117] Tsung-Yi Lin et al. "Feature Pyramid Networks for Object Detection." In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 1. 2. 2017, p. 3.

[118] Tsung-Yi Lin et al. "Microsoft coco: Common objects in context". In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[119] Shu Liu et al. "Path aggregation network for instance segmentation". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 8759–8768.

[120] Shu Liu et al. "Sgn: Sequential grouping networks for instance segmentation". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 3496–3504.

[121] Wei Liu et al. "Ssd: Single shot multibox detector". In: *European conference on computer vision*. Springer. 2016, pp. 21–37.

[122] Yiding Liu et al. "Affinity derivation and graph merge for instance segmentation". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 686–703.

[123] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[124] Cewu Lu et al. "Visual Relationship Detection with Language Priors". In: *European Conference on Computer Vision*. 2016.

[125] Jian-Hao Luo and Jianxin Wu. "An entropy-based pruning method for CNN compression". In: *arXiv preprint arXiv:1706.05791* (2017).

[126] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. "ThiNet: A Filter Level Pruning Method for Deep Neural Network Compression". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 5058–5066.

[127] Zelun Luo et al. "Label efficient learning of transferable representations acrosss domains and tasks". In: *Advances in Neural Information Processing Systems*. 2017, pp. 165–177.

[128] Jiaqi Ma et al. “Modeling task relationships in multi-task learning with multi-gate mixture-of-experts”. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM. 2018, pp. 1930–1939.

[129] Laurens van der Maaten and Geoffrey Hinton. “Visualizing data using t-SNE”. In: *Journal of machine learning research* 9.Nov (2008), pp. 2579–2605.

[130] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. “The concrete distribution: A continuous relaxation of discrete random variables”. In: *arXiv preprint arXiv:1611.00712* (2016).

[131] Oded Maron and Tomás Lozano-Pérez. “A framework for multiple-instance learning”. In: *Advances in neural information processing systems*. 1998, pp. 570–576.

[132] Mason McGill and Pietro Perona. “Deciding How to Decide: Dynamic Routing in Artificial Neural Networks”. In: *Proceedings of the 34th International Conference on Machine Learning*. 2017, pp. 2363–2372.

[133] Tomas Mikolov et al. “Efficient estimation of word representations in vector space”. In: *ICLR* (2013).

[134] Ishan Misra, Abhinav Gupta, and Martial Hebert. “From red wine to red tomato: Composition with context”. In: *CVPR*. Vol. 2. 2017, p. 6.

[135] Ishan Misra, Abhinav Shrivastava, and Martial Hebert. “Watch and learn: Semi-supervised learning for object detectors from video”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3593–3602.

[136] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. “Recurrent models of visual attention”. In: *Advances in neural information processing systems*. 2014, pp. 2204–2212.

[137] Tushar Nagarajan and Kristen Grauman. “Attributes as Operators”. In: *ECCV* (2018).

[138] Yuval Netzer et al. “Reading digits in natural images with unsupervised feature learning”. In: *NIPS workshop on deep learning and unsupervised feature learning*. Vol. 2011. 2. 2011, p. 5.

[139] Alex Nichol and John Schulman. “Reptile: a scalable metalearning algorithm”. In: ().

[140] Christopher Olston et al. “TensorFlow-Serving: Flexible, High-Performance ML Serving”. In: *arXiv preprint arXiv:1712.06139* (2017).

[141] Jeffrey Pennington, Richard Socher, and Christopher Manning. “Glove: Global vectors for word representation”. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543.

[142] Ethan Perez et al. “FiLM: Visual Reasoning with a General Conditioning Layer”. In: *AAAI*. 2018.

[143] Trung Pham et al. “Bayesian semantic instance segmentation in open set world”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 3–18.

[144] Pedro O Pinheiro and Ronan Collobert. "Weakly supervised semantic segmentation with convolutional networks". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 2. 2015, p. 6.

[145] Senthil Purushwalkam et al. "Task-driven modular networks for zero-shot compositional learning". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 3593–3602.

[146] Hang Qi, Matthew Brown, and David G Lowe. "Low-shot Learning with Imprinted Weights". In: *arXiv preprint arXiv:1712.07136* (2017).

[147] Hang Qi, Matthew Brown, and David G Lowe. "Low-shot learning with imprinted weights". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 5822–5830.

[148] Shafin Rahman, Salman Khan, and Fatih Porikli. "Zero-Shot Object Detection: Learning to Simultaneously Recognize and Localize Novel Concepts". In: *arXiv preprint arXiv:1803.06049* (2018).

[149] Sachin Ravi and Hugo Larochelle. "Optimization as a model for few-shot learning". In: *ICLR*. 2017.

[150] Joseph Redmon and Ali Farhadi. "YOLO9000: Better, Faster, Stronger". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2017, pp. 6517–6525.

[151] Joseph Redmon and Ali Farhadi. "Yolov3: An incremental improvement". In: *arXiv preprint arXiv:1804.02767* (2018).

[152] Joseph Redmon et al. "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[153] Shaoqing Ren et al. "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems*. 2015, pp. 91–99.

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

[155] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. "Grabcut: Interactive foreground extraction using iterated graph cuts". In: *ACM transactions on graphics (TOG)*. ACM. 2004, pp. 309–314.

[156] Henry A. Rowley, Shumeet Baluja, and Takeo Kanade. "Neural Network-Based Face Detection". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 20.1 (Jan. 1998), pp. 23–38. ISSN: 0162-8828. DOI: 10.1109/34.655647.

[157] Olga Russakovsky et al. "Imagenet large scale visual recognition challenge". In: *International Journal of Computer Vision* 115.3 (2015), pp. 211–252.

[158] Andrei A Rusu et al. "Meta-learning with latent embedding optimization". In: *arXiv preprint arXiv:1807.05960* (2018).

[159] Larissa K Samuelson and Linda B Smith. "They call it like they see it: Spontaneous naming and attention to shape". In: *Developmental Science* 8.2 (2005), pp. 182–198.

[160] Noam Shazeer et al. "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer". In: *International Conference on Learning Representations* (2017).

[161] Dinggang Shen, Guorong Wu, and Heung-Il Suk. "Deep learning in medical image analysis". In: *Annual review of biomedical engineering* 19 (2017), pp. 221–248.

[162] Zhiqiang Shen et al. "DSOD: Learning Deeply Supervised Object Detectors from Scratch". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE. 2017, pp. 1937–1945.

[163] Mennatullah Siam et al. "Convolutional Gated Recurrent Networks for Video Segmentation". In: *arXiv preprint arXiv:1611.05435* (2016).

[164] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[165] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *ICLR*. 2015.

[166] Linda B Smith et al. "Object name learning provides on-the-job training for attention". In: *Psychological science* 13.1 (2002), pp. 13–19.

[167] Jake Snell, Kevin Swersky, and Richard Zemel. "Prototypical networks for few-shot learning". In: *Advances in Neural Information Processing Systems*. 2017, pp. 4077–4087.

[168] Richard Socher et al. "Zero-shot learning through cross-modal transfer". In: *Advances in neural information processing systems*. 2013, pp. 935–943.

[169] Hyun Oh Song et al. "Weakly-supervised discovery of visual pattern configurations". In: *Advances in Neural Information Processing Systems*. 2014, pp. 1637–1645.

[170] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. "Highway networks". In: *arXiv preprint arXiv:1505.00387* (2015).

[171] Flood Sung et al. "Learning to Compare: Relation Network for Few-Shot Learning". In: *CVPR* (2018).

[172] Christian Szegedy et al. "Rethinking the inception architecture for computer vision". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 2818–2826.

[173] Surat Teerapittayanon, Bradley McDanel, and HT Kung. "Branchynet: Fast inference via early exiting from deep neural networks". In: *23rd International Conference on Pattern Recognition*. 2016, pp. 2464–2469.

[174] Pavel Tokmakov, Yu-Xiong Wang, and Martial Hebert. "Learning compositional representations for few-shot recognition". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 6372–6381.

[175] T. P. Trappenberg and A. D. Back. "A classification scheme for applications with ambiguous data". In: *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*. Vol. 6. 2000, 296–301 vol.6. DOI: 10.1109/IJCNN.2000.859412.

[176] Jasper RR Uijlings et al. "Selective search for object recognition". In: *International journal of computer vision* 104.2 (2013), pp. 154–171.

[177] Ashish Vaswani et al. "Attention Is All You Need". In: *arXiv preprint arXiv:1706.03762* (2017).

[178] Vinay Kumar Verma et al. "Generalized zero-shot learning via synthesized examples". In: ().

[179] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. "Pointer networks". In: *Advances in Neural Information Processing Systems*. 2015, pp. 2692–2700.

[180] Oriol Vinyals et al. "Matching networks for one shot learning". In: *Advances in Neural Information Processing Systems*. 2016, pp. 3630–3638.

[181] Paul Viola and Michael Jones. "Rapid object detection using a boosted cascade of simple features". In: *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*. Vol. 1. IEEE. 2001, pp. I–I.

[182] Paul Viola and Michael J. Jones. "Robust Real-Time Face Detection". In: *Int. J. Comput. Vision* 57.2 (May 2004), pp. 137–154. ISSN: 0920-5691.

[183] Li Wan et al. "Regularization of neural networks using dropconnect". In: *Proceedings of the 30th International Conference on Machine Learning*. 2013, pp. 1058–1066.

[184] Xin Wang et al. "Deep mixture of experts via shallow embedding". In: *Uncertainty in Artificial Intelligence*. PMLR. 2020, pp. 552–562.

[185] Xin Wang et al. "Frustratingly Simple Few-Shot Object Detection". In: *International Conference on Machine Learning (ICML)*. July 2020.

[186] Xin Wang et al. "IDK Cascades: Fast Deep Learning by Learning not to Overthink". In: *Conference on Uncertainty in Artificial Intelligence (UAI)* (2018).

[187] Xin Wang et al. "Learning to Compose Topic-Aware Mixture of Experts for Zero-Shot Video Captioning". In: *arXiv preprint arXiv:1811.02765* (2018).

[188] Xin Wang et al. "Skipnet: Learning dynamic routing in convolutional networks". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 409–424.

[189] Xin Wang et al. "Tafe-net: Task-aware feature embeddings for low shot learning". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 1831–1840.

[190] Yu-Xiong Wang and Martial Hebert. "Model recommendation: Generating object detectors from few samples". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 1619–1628.

[191] Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. "Meta-Learning to Detect Rare Objects". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 9925–9934.

[192] Yu-Xiong Wang et al. "Low-Shot Learning from Imaginary Data". In: *CVPR* (2018).

[193] P. Welinder et al. *Caltech-UCSD Birds 200*. Tech. rep. CNS-TR-2010-001. California Institute of Technology, 2010.

[194] Peter Welinder et al. "Caltech-UCSD birds 200". In: (2010).

[195] Ronald J Williams. "Simple statistical gradient-following algorithms for connectionist reinforcement learning". In: *Machine learning* 8.3-4 (1992), pp. 229–256.

[196] Yuxin Wu et al. *Detectron2*. `https://github.com/facebookresearch/detectron2`. 2019.

[197] Zuxuan Wu et al. "Blockdrop: Dynamic inference paths in residual networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 8817–8826.

[198] Yongqin Xian et al. "Feature generating networks for zero-shot learning". In: *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA*. 2018.

[199] Yongqin Xian et al. "Zero-shot learning-a comprehensive evaluation of the good, the bad and the ugly". In: *IEEE transactions on pattern analysis and machine intelligence* (2018).

[200] Jianxiong Xiao et al. "Sun database: Large-scale scene recognition from abbey to zoo". In: *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*. IEEE. 2010, pp. 3485–3492.

[201] Saining Xie et al. "Aggregated residual transformations for deep neural networks". In: *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. IEEE. 2017, pp. 5987–5995.

[202] Huazhe Xu et al. "End-to-end Learning of Driving Models from Large-scale Video Datasets". In: *arXiv preprint arXiv:1612.01079* (2016).

[203] Xiaopeng Yan et al. "Meta R-CNN: Towards General Solver for Instance-level Low-shot Learning". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 9577–9586.

[204] Flood Sung Yongxin Yang et al. "Learning to compare: Relation network for few-shot learning". In: *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA*. 2018.

[205] Zichao Yang et al. "Deep fried convnets". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 1476–1483.

[206] Fisher Yu, Vladlen Koltun, and Thomas Funkhouser. "Dilated residual networks". In: *Computer Vision and Pattern Recognition*. Vol. 1. 2017.

[207] Fisher Yu, Dequan Wang, and Trevor Darrell. "Deep Layer Aggregation". In: *arXiv preprint arXiv:1707.06484* (2017).

[208] Fisher Yu, Dequan Wang, and Trevor Darrell. "Deep Layer Aggregation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018).

[209] Fisher Yu et al. "BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[210] Amir R Zamir et al. "Taskonomy: Disentangling task transfer learning". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 3712–3722.

[211] Matthew D Zeiler and Rob Fergus. "Visualizing and understanding convolutional networks". In: *European conference on computer vision*. Springer. 2014, pp. 818–833.

[212] Li Zhang, Tao Xiang, and Shaogang Gong. "Learning a deep embedding model for zero-shot learning". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017.

[213] Ziming Zhang and Venkatesh Saligrama. "Zero-shot learning via joint latent similarity embedding". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 6034–6042.

[214] Yanzhao Zhou et al. "Weakly Supervised Instance Segmentation using Class Peak Response". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[215] Jun-Yan Zhu et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks". In: *arXiv preprint* (2017).

[216] Pengkai Zhu et al. "Zero-Shot Detection". In: *arXiv preprint arXiv:1803.07113* (2018).

[217] Yi Zhu et al. "Learning Instance Activation Maps for Weakly Supervised Instance Segmentation". In: *CVPR*. 2019.

[218] Yizhe Zhu et al. "A generative adversarial approach for zero-shot learning from noisy texts". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 1004–1013.