Learning Accurate and Robust Deep Visual Models

Yandong Li

University of Central Florida

Part of the Theory and Algorithms Commons

Find similar works at: https://stars.library.ucf.edu/etd2020

University of Central Florida Libraries http://library.ucf.edu

This Doctoral Dissertation (Open Access) is brought to you for free and open access by STARS. It has been accepted for inclusion in Electronic Theses and Dissertations, 2020- by an authorized administrator of STARS. For more information, please contact STARS@ucf.edu.

STARS Citation


https://stars.library.ucf.edu/etd2020/521
LEARNING ACCURATE AND ROBUST DEEP VISUAL MODELS

by

YANDONG LI
B.S. Southeast University, 2016

A dissertation submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy
in the Department of Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Spring Term
2021

Major Professors: Liqiang Wang and Boqing Gong
ABSTRACT

Over the last decade, we have witnessed the renaissance of deep neural networks (DNNs) and their successful applications in computer vision. There is still a long way to build intelligent and reliable machine vision systems, but DNNs provide a promising direction. The goal of this thesis is to present a few small steps along this road. We mainly focus on two questions: How to design label-efficient learning algorithms for computer vision tasks? How to improve the robustness of DNN based visual models? Concerning label-efficiency, we investigate a reinforced sequential model for video summarization, a background hallucination strategy for high-resolution image generation, and a selective module integrated into self-supervised self-training for improving object detection with noisy Web images. Besides, we study how to rank many pre-trained deep neural checkpoints for the transfer learning to a downstream task. Considering robustness, we propose a powerful blackbox adversarial attack to facilitate the research toward robust DNNs, and we also explore a new threat model that the adversaries can distill the knowledge from a blackbox teacher model to harvest a student model for imitating the characteristics of the teacher. In each chapter, we introduce the problem and present our solutions using machine learning and deep neural architectures, followed by comparisons with existing baselines and discussions on future research.

Keywords: neural networks, deep learning, supervised learning, self-supervised learning, adversarial attack, object detection, knowledge distillation
ACKNOWLEDGMENTS

This thesis concludes a fantastic 4-year journey of my PhD. career at UCF. I am fortunate to have had wonderful advisors, collaborators, family, and friends who offered me invaluable guidance and support. It is impossible to mention all of them here, but I will try my best.

First, I would like to thank my advisers, Dr. Boqing Gong and Dr. Liqiang Wang for guiding me through the finish line. They are dedicated to the success of my Ph.D. training. On the one hand, they are strict to bring up high-quality research. Sometimes, they stay up very late to help me correct the typos and polish the draft of my conference paper until the last minute of the submission deadline. I can not forget that we have spent one week iterating 6 versions of slides for a 5-minute talk at ICML 2019. On the other hand, they give me the freedom to explore some research topics according to my own interests, and they are very supportive and patient when I spent a lot of time building the codebase.

I would also like to thank my knowledgeable mentors, Chuang and Rogerio at IBM Research, Yu, Zhe, Licheng, and Jingjing at Microsoft, Di, Danfeng, and Xuhui at Google, for hosting me as a research intern in their groups. They offered me great research positions in these prestigious industry research labs. I am so lucky to meet all those remarkable guys, they not only guided me to come up with interesting ideas for my research, but also hung out with me to experience different city lives (New York, Seattle, and Bay Area) in the US. They also provided me references and suggestions when I reached out to the job market. I want to express my special gratitude to Xuhui for his huge efforts in the process of my Intern-to-Full-Time conversion at Google.

I give special thanks to the amazing collaborators and lab mates: Yuhui, Xianfeng, Huaxiu, Dongdong, Siyang, Zixia, Bingbing, Zihang, Jamal, Lijun, Yifan, and Yang. I enjoy every moment when we work together, play together, and talk with each other. The professional discussions with them
provided me fresh thoughts and inspirations when I was stuck in the saddle points of my research. Besides, the joyful random chats with them made me happy and cheered me up when I had been through hard times.

I am deeply grateful to my parents and my elder sister. This thesis would not be finished without the love and support from them. My parents never doubted my abilities and always supported me. They stand behind me no matter what I decide to do and have given me the freedom to pursue my dream. I dedicate this thesis to them. Finally, my heartfelt appreciation goes to my beloved girl, Yuting. She shared every happy and sad moment with me during this trip. She is my today and all of my tomorrows. Thank you Yuting, for lighting up my life and making every day better.
# TABLE OF CONTENTS

LIST OF FIGURES ................................................................. xii

LIST OF TABLES ................................................................. xvi

CHAPTER 1: INTRODUCTION ...................................................... 1

CHAPTER 2: LITERATURE REVIEW .............................................. 6

2.1 Video summarization ......................................................... 6

2.2 Image Generation ............................................................ 7

2.3 Weakly- and Semi-Supervised learning ............................. 9

2.4 Unsupervised/Self-supervised Representation Learning ........ 11

2.5 Transferability of DNNs .................................................... 12

2.6 Adversarial Attack .......................................................... 13

2.7 Blackbox Knowledge Distillation ..................................... 14

2.8 Active Learning ............................................................. 15

2.9 Mixup .......................................................................... 15

CHAPTER 3: REINFORCING SEQUENTIAL DETERMINANTAL POINT PROCESSES 17
4.3.1 Self-training for object detection (SOD) . . . . . . . . . . . . . . . . . . . 45
4.3.2 Selective self-training for object detection ($S^2$OD) . . . . . . . . . . . . 47
4.3.3 Selective self-supervised self-training object detection ($S^4$OD) . . . . 50
4.4 Experiments . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 51
  4.4.1 Augmenting COCO-backpack(chair) with Web-backpack(chair) . . . . . . 51
  4.4.2 Semi-supervised object detection on PASCAL VOC . . . . . . . . . . . . 56
4.5 Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 57

CHAPTER 5: RANKING NEURAL CHECKPOINTS . . . . . . . . . . . . . . . . . . . . . . . . . . 58
5.1 Problem Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 58
5.2 The Neural Checkpoint Ranking Benchmark (NeuCRaB) . . . . . . . . . . . . 61
  5.2.1 Downstream Tasks $\mathcal{T}$ . . . . . . . . . . . . . . . . . . . . . . . . . . . 63
  5.2.2 Neural Checkpoints $\mathcal{C}$ . . . . . . . . . . . . . . . . . . . . . . . . . . . 63
  5.2.3 Evaluation Metrics $\mathcal{M}$ . . . . . . . . . . . . . . . . . . . . . . . . . . . 65
5.3 Checkpoint Ranking Methods . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 65
  5.3.1 Fine-tuning with Early Stopping . . . . . . . . . . . . . . . . . . . . . . . . . . . 66
  5.3.2 Linear Classifiers . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 66
  5.3.3 Mutual Information . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 66
5.3.4 LEEP for the Checkpoints with Classification Heads ............... 67

5.3.5 NALEEP .............................................. 68

5.4 Experiments on NeuCRaB ........................................... 69

5.4.1 Comparison Results ............................................. 70

5.4.2 Main Findings ................................................. 70

5.5 Summary .......................................................... 75

CHAPTER 6: IMAGE SYNTHESIS FROM SALIENT OBJECT LAYOUT .... 76

6.1 Problem Introduction ............................................... 76

6.2 BachGAN .......................................................... 78

6.2.1 Problem Formulation and Preliminaries ......................... 79

6.2.2 Background Retrieval Module ................................. 80

6.2.3 Background Fusion Module .................................... 83

6.3 Experiments ........................................................ 85

6.3.1 Experimental Setup ............................................. 85

6.3.2 Quantitative Evaluation ......................................... 88

6.3.3 Qualitative Analysis ............................................ 89

6.3.4 Human Evaluation .............................................. 90
6.3.5 Ablation Study ................................................................. 91

6.4 Summary ............................................................................. 92

CHAPTER 7: LEARNING THE DISTRIBUTIONS OF ADVERSARIAL EXAMPLES ........................................ 93

7.1 Problem Introduction .......................................................... 93

7.2 Approach ............................................................................ 96

7.2.1 A Black-box Adversarial Attack by NES ......................... 96

7.2.2 N ATTACK ................................................................. 97

7.2.3 Initializing N ATTACK by Regression ............................... 101

7.3 Experiments ..................................................................... 102

7.3.1 Attacking 13 Most Recent Defense Techniques ............... 103

7.3.2 Attack Success Rate vs. Attack Iteration ...................... 106

7.3.3 Transferability .............................................................. 108

7.4 Summary ........................................................................... 110

CHAPTER 8: DATA-EFFICIENT BLACKBOX KNOWLEDGE DISTILLATION .................................. 111

8.1 Problem Introduction .......................................................... 111

8.2 Approach .......................................................................... 114

8.2.1 Constructing a Candidate Pool .................................... 114
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.2.2 Actively Choosing a Subset to Query the Teacher Model</td>
<td>115</td>
</tr>
<tr>
<td>8.2.3 Training the Student Network</td>
<td>116</td>
</tr>
<tr>
<td>8.2.4 Overall Algorithm</td>
<td>116</td>
</tr>
<tr>
<td>8.3 Experiments</td>
<td>118</td>
</tr>
<tr>
<td>8.3.1 Comparison Experiments</td>
<td>118</td>
</tr>
<tr>
<td>8.3.2 Ablation Study</td>
<td>123</td>
</tr>
<tr>
<td>8.3.3 Active Mixup with Out-of-Domain Data for Blackbox Knowledge Distillation</td>
<td>127</td>
</tr>
<tr>
<td>8.4 Summary</td>
<td>128</td>
</tr>
<tr>
<td>CHAPTER 9: CONCLUSION</td>
<td>130</td>
</tr>
<tr>
<td>9.1 Overall Summary</td>
<td>130</td>
</tr>
<tr>
<td>9.2 Future Work</td>
<td>130</td>
</tr>
<tr>
<td>9.2.1 Large-scale Unsupervised/Weakly Supervised Pre-training</td>
<td>130</td>
</tr>
<tr>
<td>9.2.2 Defense on Large-scale Datasets</td>
<td>131</td>
</tr>
<tr>
<td>LIST OF REFERENCES</td>
<td>133</td>
</tr>
</tbody>
</table>
### LIST OF FIGURES

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Dynamic Sequential DPP (DySeqDPP) for Video Summarization</td>
<td>18</td>
</tr>
<tr>
<td>3.2</td>
<td>From left to right: Determinantal point process (DPP) [124], Markov DPP [3], and sequential DPP (SeqDPP) [69]. The ground sets are denoted by the shaded nodes.</td>
<td>23</td>
</tr>
<tr>
<td>3.3</td>
<td>Comparison results in terms of average F1-score (the higher, the better) for 4 videos in UTE dataset where horizon axis means different $K$ used in local bipartite matching.</td>
<td>32</td>
</tr>
<tr>
<td>3.4</td>
<td>Generated video summary examples with SeqDPP and DySeqDPP</td>
<td>35</td>
</tr>
<tr>
<td>4.1</td>
<td>Chairs (top) and backpacks (bottom) are difficult to detect in daily scenes. (Images and groundtruth boxes (red) are from COCO [151], and the blue boxes are predicted by the best detectors in our experiments.)</td>
<td>40</td>
</tr>
<tr>
<td>4.2</td>
<td>Examples of the top three retrieved images from image-to-image search.</td>
<td>43</td>
</tr>
<tr>
<td>4.3</td>
<td>Labeling errors in COCO-backpack. Two images in one group mean that they contain conflict annotations.</td>
<td>44</td>
</tr>
<tr>
<td>4.4</td>
<td>Overview of the proposed approach. Top: using a small curated dataset to retrieve relevant Web images and to train a teacher detector. Middle: training a selective net to group pseudo boxes predicted by the teacher into positive, negative, and ambiguous groups. Bottom: learning a student detector from the Web-augmented training set with a self-supervised loss.</td>
<td>46</td>
</tr>
</tbody>
</table>
4.5 Top: images in the COCO training set with groundtruth boxes. Bottom: boxes, their IoUs with the groundtruth, and confidence scores predicted by the teacher detector. The correlation between IoU and confidence is low. . . . 47

4.6 The two panels on the left: Positive (negative) boxes chosen by the selective net yet with low (high) confidence scores predicted by the teacher detector. Rightmost: Ambiguous boxes according to the selective net. . . . . . . . . . 53

4.7 Example Web images for augmenting COCO-backpack. . . . . . . . . . . . . 54

5.1 Fine-tuning the checkpoints in Group I on four downstream tasks. We keep the best fine-tuning accuracy for each checkpoint-task pair after hyper-parameter sweeping. For better visualization, the values are offset by their mean (cf. Table 4 in Appendix for the absolute values). (Best viewed in color. Red: generative models. Black: From-Scratch. Green: self-supervised models. Blue: semi-supervised models. Yellow, Pink, and Orange: supervised models trained on ImageNet, Inaturalist, and Places365, respectively. Cyan: Hybridly-supervised model.) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 62

5.2 LEEP’ checkpoint ranking performance, evaluated by Kendall’s $\tau$, on Groups I and II in NeuCRaB. We vary the PCA feature dimension and the number of Gaussian components in GMM. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 69

6.1 Top row: images synthesized from semantic segmentation maps. Bottom row: high-resolution images synthesized from salient object layouts, which allows users to create an image by drawing only a few bounding boxes. . . . . 77
6.2 Overview of the proposed BachGAN for image synthesis from salient object layout. ............................................................... 79

6.3 Examples of image synthesis results from different models on the Cityscapes dataset. Results from Layout2im are not included due to low resolution of the generated images (provided in Appendix). .................................................... 82

6.4 Examples of image synthesis results from different models on the ADE20K dataset. .................................................................................. 85

6.5 Generated images by adding bounding boxes sequentially to previous layout (Cityscapes). ................................................................. 87

6.6 Generated images by adding bounding boxes sequentially to previous layout (ADE20k). ................................................................. 88

6.7 Top row: synthesized images based on salient object layouts from the test set. Bottom row: synthesized images based on salient layouts with flipping objects, modified from the top-row layouts. ............................................ 88

6.8 Examples of top-3 retrieved images from the memory bank of Cityscapes. . 89

7.1 (a) Success rate versus run steps of N\textsuperscript{\textregistered}ATTACK. (b) Comparison results with QL measured by the log of average number of queries per successful image. The solid lines denote N\textsuperscript{\textregistered}ATTACK and the dashed lines illustrate QL. .................. 107

7.2 Transferabilities of BPDA [10] (left) and N\textsuperscript{\textregistered}ATTACK (right). Each entry shows the attack success rate of attacking the column-wise defense by the adversarial examples that are originally generated for the row-wise DNN. . . . 109

xiv
8.1 Data-efficient blackbox knowledge distillation. Given a blackbox teacher model and a small set of unlabeled images, we propose to employ mixup [307] and active learning [138] to train a high-performing student neural network in a data-efficient manner (b) so that we do not need to re-do the heavy and expensive data curation used to train the teacher model (a).

8.2 Mixup images whose confidence scores (cf. eq. (8.3)) are the lowest among all candidates in the third iteration. For each mixup image, we show the top three labels and probabilities returned by the blackbox teacher model.

8.3 Different mixup images from the same pair of the original images by varying the mixup coefficient $\lambda$. We show the top three labels and probabilities predicted by the teacher model for each of them. It is interesting to see how the top-1 label changes from Hockey Arena, to Baseball Field, and to Golf Course.

8.4 Test accuracy of student networks vs. number of queries into the blackbox teacher model on CIFAR-10 (left) and Places365-Standard (right). We use 500 and 20K natural images for the two datasets, respectively. The plot for CIFAR-10 starts from first active learning stage ($t = 1$ in Algorithm 1) and the one for Places365 starts from the initial student network training by natural images. The initial student network for CIFAR-10 trained by using natural images only yields 43.67% accuracy.
# LIST OF TABLES

3.1 Comparison results on UTE evaluated by the bipartite matching F1-score
\((K = 12)\) ........................................................................................................ 33

3.2 Comparison results on video summarization on SumMe and TVsum dataset.
The results are evaluated by F1-score, the higher the better. ....................... 37

4.1 Statistics of the Web images in this work and their counterparts in COCO . . 44

4.2 Comparison results for detecting backpacks and chairs. ......................... 51

4.3 Results of S^4OD with various Web data augmentations to COCO-backpack. . 54

4.4 Comparison results on the relabeled COCO-backpack. ......................... 55

4.5 Comparison results on VOC2007 (* the numbers reported in [102]) .......... 56

5.1 Checkpoint ranking results on Group I, the checkpoints of mixed supervision (GFLOPS excludes a forward pass on training data, which takes 3.04E5 GFLOPS shared by all methods) ......................................................... 72

5.2 Checkpoint ranking results on Group II, the checkpoints at different pre-training stages (GFLOPS excludes a forward pass on training data, which takes 3.04E5 GFLOPS shared by all) ........................................... 72
5.3 Checkpoint ranking results on Group III, the checkpoints of heterogeneous architectures (GFLOPS excludes a forward pass on training data, which takes 2.73E5 GFLOPS shared by all) ........................................... 73

6.1 Results on Cityscapes and ADE20K w.r.t. FID and the pixel accuracy (Acc). Results with (†) are reported in [198], serving as the upper bound of our model performance. ......................................................... 87

6.2 User preference study. Win/lose/tie indicates the percentage of images generated by BachGAN are better/worse/equal to the compared model. .............. 90

6.3 FID scores of BachGAN with different numbers of retrieved segmentation maps (Cityscapes). ................................................................. 91

6.4 FID scores of BachGAN and BachGAN-r trained using memory bank of different sizes (Cityscapes). ................................................................. 91

7.1 Adversarial attack on 13 recently published defense methods. (* the number reported in [10]. For all the other numbers, we obtain them by running the code released by the authors or implemented ourselves with the help of the authors. For D-based and ADV-TRAIN, we respectively report the results on 100 and 1000 images only because they incur expensive computation costs.) 104
8.1 Comparison results on Places365-Standard, CIFAR-10, MNIST, and Fashion-MNIST. The “Teacher” column reports the teacher model’s accuracy on the test sets, “KD Accuracy” is the student network’s test accuracy, “Success” stands for the distillation success rates, “Black/White” indicates whether or not the teacher model is blackbox, “Queries” lists the numbers of queries into the teacher models, and “Unlabeled Data” shows the numbers of original training images used in the experiments. (* results reported in the original paper) .......................................................... 119

8.2 Classification accuracy on CIFAR-10 with different numbers of real images and selected synthetic images. ...................................................... 123

8.3 Classification accuracy on Places365-Standard with different numbers of real images and selected synthetic images. .......................... 124

8.4 CIFAR-10 classification accuracy by the student neural networks which are distilled by using out-of-domain data. ............................. 127

8.5 CIFAR-10 classification accuracy by the student neural networks which are distilled by using out-of-domain data. We set the number of selected synthetic images to 40K and vary the numbers of real images. .................. 128
CHAPTER 1: INTRODUCTION

We embrace the big data era, as people produce and consume enormous amounts of visual data every day by taking pictures, watching movies, and sharing their photos and videos on social media. Driven by the astounding amounts of visual data and the availability of computational power (GPU/TPU) [107, 312, 311, 163, 308, 161, 237, 310, 309], DNNs rise into prominence again and come to dominate the field of computer vision [77, 38, 78, 42, 141, 100, 265, 269]. Consequently, deep learning is able to achieve human-level recognition performance on real-world benchmarks like ImageNet [122], which significantly outperforms the traditional algorithms. However, deep learning is far from perfect, and it incurs many problems when we apply it to real-world computer vision applications. In this thesis, we mainly focus on tackling three challenges existing in deep learning based computer vision algorithms:

- DNNs can not perform well on specific vision tasks, especially when they include non-differentiable components and can not be end-to-end optimized (cf. Chapter 3).

- The superiority of deep learning relies on high-quality human-curated datasets, involving a lot of human efforts and limiting the usage in particular domains like medical images (cf. Chapter 4,5,6).

- DNNs are highly vulnerable against adversarial examples [266, 145, 41, 110], which may be imperceptible to humans, but lead the deep neural networks to make completely incorrect decisions (cf. Chapter 7,8).

We try to address above issues in two directions. On the one hand, we strive to design label-efficient algorithms to improve the accuracy of deep visual models. We first present a reinforcing probabilistic model to capture ‘local diversity’ for supervised video summarization [147]. In order
to deal with the lack of high-volume labeled training data on a target task, we seek solutions from unsupervised learning with massive unlabelled Web images [143] and transfer learning with checkpoints pretrained on large-scale human-curated data [144]. In addition, we conduct extensive study on a specific task – photo-realistic image-to-image translation [141], and find that it demands fine-grained semantic segmentation maps. We propose a new task – image generation from salient object layouts to alleviate the requirements of expensive pixel-level segmentation annotations. On the other hand, we explore the robustness of deep neural networks by proposing a strong and universal blackbox attack [145] and a new threat model [266] for DNN based Web APIs. Please see detailed discussions below.

Most of the deep neural networks are coupled with a straightforward and somewhat brute-force technique – end-to-end learning, by applying gradient-based learning to the system as a whole. The power of end-to-end deep learning has been demonstrated on image classification [122], where AlexNet increased the accuracy by a factor of one point two, riveting attention even outside of the community. However, it can not be directly integrated into all tasks since all the modules in the learning systems should be differentiable, and video summarization is one of those tasks due to the non-differentiable evaluation metrics.

In Chapter 3, we investigate how to incorporate end-to-end deep learning into video summarization, where we propose a novel probabilistic model, built upon SeqDPP [69], to dynamically control the time span of a video segment upon which the local diversity is imposed. In particular, we enable SeqDPP to learn to automatically infer how local the local diversity is supposed to be from the input video. The resulting model is extremely involved to train by the hallmark maximum likelihood estimation (MLE), which further suffers from the exposure bias and non-differentiable evaluation metrics. To tackle these problems, we instead devise a reinforcement learning algorithm for training the proposed model. Extensive experiments verify the advantages of our model and the new learning algorithm over MLE-based methods.
Recent works [171, 81, 285] demonstrate excellent results on image recognition by introducing additional unlabeled/weakly-labeled training images. In Chapter 4, we study how to leverage Web images to augment human-curated object detection datasets. Our approach is two-pronged. On the one hand, we retrieve Web images by image-to-image search, which incurs less domain shift from the curated data than other search methods. The Web images are diverse, supplying a wide variety of object poses, appearances, their interactions with the context, etc. On the other hand, we propose a novel learning method motivated by two parallel lines of work that explore unlabeled data for image classification: self-training and self-supervised learning. They fail to improve object detectors in their vanilla forms due to the domain gap between the Web images and curated datasets. To tackle this challenge, we propose a selective net to rectify the supervision signals in Web images. It not only identifies positive bounding boxes but also creates a safe zone for mining hard negative boxes. We report state-of-the-art results on detecting backpacks and chairs from everyday scenes, along with other challenging object classes.

In Chapter 5, we explore how to rank many pre-trained deep neural networks (DNNs), called checkpoints, for the transfer learning to a downstream task. Thanks to the broad use of DNNs, we may easily collect hundreds of checkpoints from various sources. Which of them transfers the best to our downstream task of interest? Striving to answer this question thoroughly, we establish a neural checkpoint ranking benchmark (NeuCRaB) and study some intuitive ranking measures. These measures are generic, applying to the checkpoints of different output types without knowing how the checkpoints are pre-trained on which dataset. They also incur low computation cost, making them practically meaningful. Our results suggest that the linear separability of the features extracted by the checkpoints is a strong indicator of transferability. We also arrive at a new ranking measure, NLEEP, which gives rise to the best performance in the experiments.

With the rapid growth of deep generative adversarial networks (GANs), photo-realistic images can be synthesised conditioned on fine-grained semantic segmentation map. However, collecting
the segmentation annotations is extremely labor-intensive and time-consuming. In Chapter 6, we explore a new task towards more practical application for image generation - high-quality image synthesis from salient object layout. This new setting allows users to provide the layout of salient objects only (i.e., foreground bounding boxes and categories), and lets the model complete the drawing with an invented background and a matching foreground. Two main challenges spring from this new task: (i) how to generate fine-grained details and realistic textures without segmentation map input; and (ii) how to create a background and weave it seamlessly into standalone objects. To tackle this, we propose Background Hallucination Generative Adversarial Network (BachGAN), which first selects a set of segmentation maps from a large candidate pool via a background retrieval module, then encodes these candidate layouts via a background fusion module to hallucinate a suitable background for the given objects.

By generating the hallucinated background representation dynamically, our model can synthesize high-resolution images with both photo-realistic foreground and integral background. Experiments on Cityscapes and ADE20K datasets demonstrate the advantage of BachGAN over existing methods, measured on both visual fidelity of generated images and visual alignment between output images and input layouts.

As described above, DNNs have been widely used in many tasks, including some security-sensitive ones like face recognition, autonomous driving systems, and financial data analysis. However, recent works reveal the increasing concern about the robustness of DNNs – they are vulnerable against adversarial examples whose changes from the benign ones are imperceptible. In Chapter 7 and 8, we explore the robustness of DNNs.

In Chapter 7, we provide a strong adversarial attack method that can universally defeat a variety of DNNs and associated defense techniques. Progress on powerful adversarial attack algorithms will significantly facilitate the research toward more robust DNNs that are deployed in uncertain or even
adversarial environments. Instead of searching for an "optimal" adversarial example for a benign input to a targeted DNN, our algorithm finds a probability density distribution over a small region centered around the input, such that a sample drawn from this distribution is likely an adversarial example, without the need of accessing the DNN’s internal layers or weights. Our approach is universal as it can successfully attack different neural networks by a single algorithm. It is also strong; according to the testing against 2 vanilla DNNs and 13 defended ones, it outperforms state-of-the-art black-box or white-box attack methods for most test cases. Additionally, our results reveal that adversarial training remains one of the best defense techniques, and the adversarial examples are not as transferable across defended DNNs as them across vanilla DNNs.

In the end, we explore a new threat model for DNNs in Chapter 8— the adversaries are capable of distilling knowledge from a blackbox teacher model in a data-efficient manner. There are two major challenges. One is that the number of queries into the teacher model should be minimized to save computational and/or financial costs. The other is that the number of images used for the knowledge distillation should be small; otherwise, it violates our expectation of reducing the dependence on large-scale datasets. To tackle these challenges, we propose an approach that blends mixup and active learning. The former effectively augments the few unlabeled images by a big pool of synthetic images sampled from the convex hull of the original images, and the latter actively chooses from the pool hard examples for the student neural network and query their labels from the teacher model. We validate our approach with extensive experiments\(^1\).

\(^1\) Note that the mathematical notations are consistent in each individual chapter, but the same symbol may refer to different concepts in different chapters.
CHAPTER 2: LITERATURE REVIEW

2.1 Video summarization

Different algorithms for automatic video summarization are generally designed by the same principles. Those informative guidelines contain three main factors: (1) individual interestingness or relevance [136, 166], which means selecting frames/shots that are important in the video; (2) representativeness [90, 114, 184], which means the summary should contain the main event of the videos; (3) collective diversity or coverage [154, 314], which is to reduce redundant frames/shots without losing much information. These factors are used in most of the existing works. Next, we review the representative approaches in two common classes, unsupervised and supervised video summarization.

**Unsupervised video summarization:** A variety of prior works is designed based on basic visual quality like low-level appearance and motion cues [314, 136, 184, 166, 90, 114, 108, 169, 128, 154]. Graph models are utilized for event detection in some approaches [128, 184]. In general, the criteria applied in those methods for making decisions about including or excluding shots are devised by the system developers empirically. Besides, some approaches leverage Web images for video summarization based on the assumption that the static Web pictures tend to contain information of interest to people, so the Web images reveal user-oriented importance selecting video shots/frames [114, 116, 286, 29].

**Supervised video summarization:** Recently, several explorations on supervised video summarization have been exerted for various goals [69, 80, 315, 79, 21, 166, 136, 152, 223, 224, 316]. They achieve superior performance over the traditional unsupervised clustering algorithms. Among them, Gygli et al. try to add some supervised flavor to optimize mixture objectives with learning
each criterion’s weight [79, 80]. A hierarchical model has been proposed to learn with few labels, and it is optimized to generate video summary containing interesting objects [152]. Egocentric videos [33] can be compacted with importance of people and objects [136]; on the other hand, Zheng et al. explicitly consider how one sub-event leads to another in order to provide a better sense of story for those kinds of videos [166]. Meanwhile, Yao et al. propose a pairwise deep ranking model to highlight video segments of first-person videos [298]. In conclusion, supervised methods are capable of utilizing the intentions of users about what a qualified video summary is rather than designing the systems only relying on the experts’ own perspective.

Besides, as a powerful diverse subset selection model, the determinantal point process (DPP) has been widely used for video summarization. For instance, Gong et al. propose the first supervised video summarization method [69] (SeqDPP) as far as we know, it models local diversity to capture the temporal information of videos rather than modeling global diversity. Combining long short-term memory (LSTM) with DPPs has been studied in [316] to model the variable-range temporal dependency and diversity among video frames at the same time. Effort has been spent to study transferring summary structures from annotated videos to unseen test videos in [315]. Sharghi et al. explore the query-focused video summarization in [223, 224]. Large margin separation principle has been leveraged for DPPs to estimate parameters in [21].

We will provide more details of DPPs and SeqDPP in Sections 3.2.1 and 3.2.2.

2.2 Image Generation

**Conditional Image Generation** Conditional image synthesis tasks can facilitate diverse inputs, such as source image [99, 153, 200, 324, 325], sketch [219, 325, 279], scene graph [106, 8], text [172, 313, 289, 139, 142], video clip [133, 54, 228], and dialogue [226, 27]. These approaches
fall into three main categories: Generative Adversarial Networks (GANs) [70, 179], Variational Autoencoders (VAEs) [119], and autoregressive models [260, 190].

In previous studies, the layout is typically treated as an intermediate representation between the input source (e.g., text [91, 139] or scene graph [106]) and the output image. Instead of learning a direct mapping from text/scene graph to an image, the model constructs a semantic layout (including bounding boxes and object shapes), based on which the target image is generated. Well-labeled instance segmentation maps are required to train the object shape generator. There is also prior work that aims to synthesize photo-realistic images directly from semantic segmentation maps [272, 99]. However, obtaining detailed segmentation maps for large-scale datasets is time consuming and labor intensive. In [132], to avoid relying on instance segmentation mask as the key input, additional background layout and object layout are used as the input. [318] proposed the task of image synthesis from object layout; however, both foreground and background object layouts are required, and only low-resolution images are generated.

**High-Resolution Image Synthesis** Adversarial learning has been applied to image-to-image translation [99, 271], to convert an input image from one domain to another using image pairs as training data. $L_1$ loss [105] and adversarial loss [70] are popular choices for many image-to-image translation tasks.

Recently, Chen and Koltun [24] suggest that it might be difficult for conditional GANs to generate high-resolution images due to training instability and optimization issues. To circumvent this, they use a direct regression objective based on a perceptual loss [46] and produce the first model that can synthesize high-quality images. Motivated by this, pix2pix-HD [272] uses a robust adversarial learning objective together with a new multi-scale generator-discriminator architecture to improve high-resolution generation performance. In [271], high-resolution video-to-video synthesis are explored to model temporal dynamics. Park et al. [198] shows that spatially-adaptive normalization
(SPADE), a conditional normalization layer that modulates the activations using input semantic layouts, can synthesize images significantly better than state-of-the-art methods.

2.3 Weakly- and Semi-Supervised learning

**Weakly supervised object detection.** The great success of the state-of-the-art object detectors [323, 67, 214, 149, 155, 213, 83] heavily relies on large volume of human annotated data. For instance, the flagship COCO dataset [151] contains about 10k boxes per class. Acquiring more curated data is challenging due to the high financial and time costs. Accordingly, weakly supervised object detection (WSOD) [7, 16, 295, 247, 246, 304] — learning to localize objects with image-level annotations only - has become an active research topic, since image-level annotations are easier to obtain than bounding box annotations. Representative works [297, 207, 232, 125] utilize motion cues in videos to delineate objects and refine object proposals. In addition, Tao et al. [249] incorporate web images to learn a good feature representation for WSOD. Fine-grained segmentation [65, 140] can also be used to guide WSOD. Although great progress has been made in WSOD, there is still quite a gap for them to catch up its supervised counterpart. The performance of fully supervised methods [277, 214] are about 25 points better in terms of mean average precision compared to the weakly supervised ones [304].

**Semi-supervised Object Detection.** The semi-supervised object detection approaches can be divided into two categories: weakly semi-supervised detectors [293, 248, 64, 294, 210] and complete semi-supervised detectors [102, 270]. Weakly semi-supervised object detection method uses fully annotated data with box-level annotations as well as weakly labeled data with only image-level annotations. Tang et al. [248] propose an LSDA-based method that can handle disjoint sets in semi-supervised detection. Note-RCNN [64] proposes a mining and training scheme using a few seedbox-level annotations and a large scale of image-level annotations. Recently, Yang et
al. [294] propose a fine-grained detection method that requires only bounding box annotations of a smaller number of coarse-grained classes and image-level labels on a large number of fine-grained classes.

Compared with weakly semi-supervised detectors, the complete semi-supervised detectors are more general by using unlabeled data in combination with the box-level labeled data. Our approach can be technically categorized into complete semi-supervised detectors. There are only a few research works in the complete semi-supervised detection field. Wang et al. [270] present a principled Self-supervised Sample Mining (SSM) process in active learning to get reliable region proposals for enhancing the object detector. But they need additional human labeling effort to annotate the low-consistency samples. Jeong et al. [102] introduce a consistency loss based method in semi-supervised object detection. They propose to add a simple consistency loss between original box and flipped box for the classification network. It is consistently effective for one-stage detectors but have limited performance improvement for two-stage detectors as box selection is not incorporated for unlabeled data. This pioneer work inspires us to dig deeper into two-stage detectors to build a more robust learning system with components like box selection. Besides, we use the crawled Web images as the unlabeled data where the data distribution is unknown, while [270, 102] use COCO [151] and PASCAL VOC 2012 [53] as unlabeled data where the data/class distributions is similar to their labeled dataset – PASCAL VOC 2007 [53].

Semi-supervised Classification. The majority of data samples in the real-world lacks annotations. Hence, semi-supervised learning [11, 212, 129, 250, 175, 167, 15, 284] exploits the potential of unlabeled data to gain more understanding of the population structure in general. Most of the works [167, 15, 284, 175] are based on consistency training, which constrains model predictions to be invariant to the noise injected to the input, hidden states, or model parameters. Consistency regularization [129, 250, 175] has shown state-of-the-art performance in semi-supervised classification [189]. Besides, pseudo-label based approaches have improved the per-
formance of semi-supervised learning by utilizing high-confident samples with pseudo-labels in training [98, 229, 134, 5].

Xie et al. [285] argue that consistency training in the early phase regularizes models towards high-entropy predictions and pseudo label based approaches rely on a model being trained rather than a high-accuracy converged model, which all prevent those methods from achieving better performance. They [285] instead utilize self-training [221, 292, 215, 299] and aggressively inject noise to make the student better. They report state-of-the-art results on ImageNet with 300M unlabeled images. Hence, we build our approach upon self-training in my work.

2.4 Unsupervised/Self-supervised Representation Learning.

Unsupervised/self-supervised representation [43, 81, 62, 61, 63, 320, 317] learning on unlabeled data has attracted a great deal of attention nowadays. Some of them define a wide range of pretext tasks like recovering the input under some corruption [263, 200], predicting rotation [66] or patch orderings [43, 188] of an exemplar image, and tracking [273] or segmenting objects [199] in videos. Others utilize contrastive learning [191, 45, 81, 278, 25] by maximizing agreement between differently augmented views of the same data example. Good visual representations can help object detection [171, 66], and self-supervised learning has been applied to replace the supervised ImageNet pretraining [199, 101] for object detection. In addition, Lee et al. [135] propose a set of auxiliary tasks to make better use of given limited labels. However, [135] requires box-level annotations to serve auxiliary task learning.
2.5 Transferability of DNNs

Task transferability. A task usually refers to a joint distribution over input and label. Task transferability [40, 258, 303] aims to predict how well a deep neural network pre-trained on a source task transfers to the target task. One may estimate the task transferability by data similarities regardless of models being used. Some work in this line includes conditional entropy [258], data set distance as optimal transport [4], $F$-relatedness [14], $A$-distance [115], and discrepancy distance [173]. Besides, Poole et al. [204] derived information theoretic bounds. These methods are generally hard to compute in practice and rely on the availability of the source data. Some recent task transferability estimators involve both data and the models. Taskonomy [303] is a fully computation method, where task similarity scores are obtained by transfer learning experiments. Dwivedi et al. [50] analyzed the representation similarities to construct a task taxonomy. More recently, Puigcerver et al. [208] proposed a novel approach for transfer learning by firstly training experts on large-scale datasets and then selecting the relevant expert by simple heuristics for downstream tasks. Besides the models trained on source tasks, all these methods also require a fine-tuned or independently trained model from the target task.

Recent works demonstrated that using pre-trained checkpoints that have similar feature representations as the target task’s representations can improve transfer learning [50, 234, 235]. Song et al. [234, 235] employed attribution maps to compare two models and then quantified transferabilities by the similarity of two models. More recently, Dwivedi et al. [49] proposed a duality diagram similarity (DDS) based approach to select a model initialization for transfer learning. Those approaches all require a converged model on target datasets, incurring intensive computation. However, we want to design a lightweight method for ranking checkpoints [144], ideally without any training procedures.

Predicting neural networks’ generation gap. The difference between a model’s performance
on the training data versus its performance on test data is known as the generalization gap. It is practically useful and theoretically impactful to predict a neural network’s generalization gap. Most recent work does so by finding a set of features that is predictive of the generalization, e.g., by estimating data margins [12, 52, 233]. Jiang et al. [103] and Yak et al. [291] demonstrate how the margin signatures of a neural network can predict the generalization gap with small errors. Besides, the network complexity and noise stability are also useful cues [183, 109, 12, 6].

2.6 Adversarial Attack

There is a vast literature of adversarial attacks on and defenses for DNNs. We focus on the most related works in this section rather than a thorough survey.

White-Box Attacks. The adversary has full access to the target DNN in the white-box attack. Szegedy et al. [244] first find that DNNs are fragile to the adversarial examples by using box-constrained L-BFGS. Goodfellow et al. [71] propose a fast gradient sign (FGS) method, which is featured by efficiency and high performance for generating the $\ell_\infty$ bounded adversarial examples. Papernot et al. and Moosavi-Dezfooli et al. [196, 178] instead formulate the problems with the $l_0$ and $\ell_2$ metrics, respectively. Carlini et al. [20] have proposed a powerful iterative optimization based attack. Similarly, a projected gradient descent has been shown strong in attacking DNNs [170]. Most the white-box attacks rely on the gradients of the DNNs. When the gradients are “obfuscated” (e.g., by randomization), Athalye et al. [10] derive various methods to approximate the gradients, while we use a single algorithm to attack a variety of defended DNNs.

Black-Box Attacks. As the name suggests, some parts of the DNNs are treated as black boxes in the black-box attack. Thanks to the adversarial examples’ transferabilities [244], Papernot et al. [195] train a substitute DNN to imitate the target black-box DNN, produce adversarial examples
of the substitute model, and then use them to attack the target DNN. Chen et al. [23] instead use the zero-th order optimization to find adversarial examples. [95] use the evolution strategy [218] to approximate the gradients. Brendel et al. [17] introduce a decision-based attack by reading the hard labels predicted by a DNN, rather than the soft probabilistic output. Similarly, Cheng et al. [26] also provide a formulation to explore the hard labels. Most of the existing black-box methods are tested against vanilla DNNs. In my work, we test them on defended ones along with our N\textsc{Attack}.

2.7 Blackbox Knowledge Distillation.

**Knowledge distillation.** Knowledge distillation is proposed in [89] to solve model compression problems, thus relieving the burden of ensemble learning. This work suggests that class probabilities, as “dark knowledge”, are very useful to retain the performance of original network, and thus, light-weight substitute model could be trained to distill this knowledge. This approach is very useful and has been justified to solve a variety of complex application problems, such as pose estimation [220, 264, 186], lane detection [92], real-time streaming [180], object detection [36], video representation [251, 59, 63], and so forth. Furthermore, this approach is able to boost the performance of deep neural network with improvement on efficiency [203] and accuracy [126]. Accordingly, lots of research is conducted to enhance its performance from the perspective of training strategy [259, 104], distillation scheme [86, 28], or network properties [202], etc.

However, there is an important issue. Traditional knowledge distillation requires lots of original training data which are very difficult to be obtained. To alleviate this data demand, few-shot knowledge distillation is proposed to retain teacher model performance with pseudo samplers which are generated in adversarial manner [117]. Another approach called data free knowledge distillation leverages extra activation records from teacher model to reconstruct original datasets, thus recov-
ering teacher model [162]. Recently, a zero-knowledge distillation method is developed by synthesizing data with gradient information of teacher network [182]. Nevertheless, these approaches require the gradient information of teacher network, which enables them intractable in the real world.

**Blackbox Optimization.** Blackbox optimization is developed based on zero knowledge in the gradient information of queried models and widely used to solve practical problems. Recently, this work is widely used in deep learning, especially model attack. A rich line of blackbox attacking approaches [23, 95, 218, 17, 145] are explored by accessing the input-output pairs of classifiers, most of which are focusing on attacks resulting from accessing the data. [57] instead investigates that the adversaries are capable of recovering sensitive data by model inversion. However, there is no work for blackbox knowledge distillation.

### 2.8 Active Learning.

Active learning is a learning process by interaction between oracle and learner agents. This strategy is widely used to solve learning problems which exhibit costly data labelling since it could exploit existing data information to efficiently improve obtained model, thus reducing the number of queries. Lots of effective approaches are proposed to optimize this process, such as uncertainty-based [138, 296, 58] and margin-based methods [48, 222]. Form the review by [68], uncertainty-based methods, despite simple, are able to obtain good performance.

### 2.9 Mixup.

Zhang *et al.* first proposed mixup to improve the generalization of deep neural network [307]. Between-Class learning [252] (BC learning) was proposed for deep sound recognition, and then,
they extended this approach to image classification [253]. Following them, Pairing Samples [96] was proposed as a data augmentation approach by taking an average of two images for each pixel. More recently, an approach called AutoAugment [31], explores improving data augmentation policies by automatically searching.
CHAPTER 3: REINFORCING SEQUENTIAL DETERMINANTAL POINT PROCESSES

3.1 Problem Introduction

The Internet age has come to such a new phase that high-definition videos are both ubiquitous and dominant in the IP traffic featured by the boom of video sharing websites, online movies and television shows, and the emerging live video streaming services. Some statistics indicate that about 300 hours of video are uploaded to YouTube per minute and more than 500 million hours of video are watched on YouTube daily. Such a large volume of video content and high viewing frequency demand automatic video summarization algorithms. By distilling important events from the original video and condensing them to a short video clip (or a story board, text description, etc.), video summarization has a great potential in many real-world applications.

Video summarization has been one of the basic research areas in the fields of computer vision and multimedia for decades [176]. A variety of techniques have been proposed for different scenarios of video summarization. In general, a good video summary is supposed to describe main events [90, 114, 184] happened in the video and meanwhile remove the video shots that are redundant [154, 314] and/or unimportant [136, 166].

We consider video summarization as a diverse subset selection problem: given a video that can be

This chapter contains previously published materials from “How local is the local diversity? reinforcing sequential determinantal point processes with dynamic ground sets for supervised video summarization.” by Yandong Li, Liqiang Wang, Tianbao Yang, and Boqing Gong, published in Proceedings of the European Conference on Computer Vision (ECCV), pp. 151-167. 2018 [147].
seen a collection of shots, the goal is to select a subset from the collection to summarize the whole video. This view opens the door for supervised learning approaches to video summarization [69, 80, 315, 79, 21] that fit subset selection models to the video summaries annotated by users. Unlike the conventional unsupervised video summarization methods [314, 136, 184, 166, 90, 114, 108, 169], the supervised ones implicitly infer users’ intentions and summarization criteria as opposed to domain experts’ handcrafting.

In the supervised video summarization models, a key factor they are supposed to encompass is the diversity of the selected subset of video shots. This is often imposed by submodularity [80, 287] and determinant [69, 315, 223]. When a video sequence is short, **global diversity** over the whole sequence seems like a natural choice [315, 80].

However, if the videos are lengthy like the egocentric videos that are often hours long, it is necessary to track the temporal structures of the videos and enforce **local diversity** instead [69, 224].
The local diversity refers to that the shots selected from a short time duration are diverse but visually similar shots are allowed to co-exist in the summary if they appear far apart in the video. Consider a video sequence that is about “leaving home for shopping in the morning and then coming back home to have lunch”. Although the video shots of the “home” scene in the morning may be similar to those at noon, the summary should contain some shots of both in order to make the summary a complete story carried by the video.

In this work, we are mainly interested in summarizing extremely lengthy (e.g., egocentric) videos and, accordingly, models that are capable of observing the local diversity. Among the existing works, sequential determinantal point process (SeqDPP) [69] and dppLSTM [316] both account for the temporal dynamics of the videos. However, neither of them explores “how local” the local diversity should be. Take the SeqDPP for instance, it requires users to manually partition the video into disjoint segments of the same length and then impose diversity both within each of them and between adjacent segments, locally. There is no guiding principle about how to best partition a video sequence into such segments. Besides, it could be sub-optimal to make the segments of the same length because different types of events often unroll at distinct frame rates. The same snags exist in dppLSTM.

We propose to improve the SeqDPP model [69] by a latent variable that dynamically controls the time span of a segment upon which the local diversity is then defined in the form of a conditional DPP. In other words, we enable SeqDPP to learn to automatically infer how local the local diversity is in the input video. Figure 3.1 illustrates our main idea. Given an input video shown on the top panel, our dynamic SeqDPP seeks the appropriate and possibly different lengths of the segments (cf. the middle panel) from which it selects video shots (the bottom panel) and places them on a story board or links them into a short video clip as the summary of the video.

Another contribution of this work is a novel reinforcement learning algorithm for the proposed
dynamic SeqDPP (DySeqDPP). While DySeqDPP seems like a straightforward extension to the vanilla SeqDPP, it is less obvious how to efficiently train the model. The DPPs [123] and its variants (e.g., SeqDPP [69], dppLSTM [316], and SH-DPP [223]) are almost all trained by the hallmark maximum likelihood estimation (MLE) except for the large-margin DPP [21] and Bayesian DPP [2]. However, it is often difficult to maximize the likelihood of a sequential model with latent variables; gradient ascent fails to track the statistical structure, and the EM algorithm [34] becomes involved and inefficient unless one assumes special compositions of a sequential model [275].

In light of these challenges, we instead provide a reinforcement learning perspective for understanding SeqDPPs. The proposed DySeqDPP is used as a policy by an agent to interact with the environment — the input video. Accordingly, we train this DySeqDPP model by policy gradient descent [239]. Not only we do not have to explicitly deal with the latent variables, but also we benefit from the flexible reward functions in policy gradient descent — we can bridge the training and validation phases of the summarizer by defining the reward function as some evaluation metric(s).

We evaluate this dynamic SeqDPP model on standard video summarization datasets. Extensive results show that it significantly outperforms competing baselines especially the vanilla SeqDPP, verifying the necessity of dynamically determining how local the local diversity is. The rest of the paper is organized as follows. After that, we describe our dynamic SeqDPP and the reinforcement learning algorithm in Section 3.3. We report empirical results in Section 3.4.

3.2 Background: DPP and SeqDPP

We briefly review the determinantal point process (DPP) and the sequential DPP (SeqDPP) in this section. It will become clear soon how the former promotes diversity in the selected subsets and the latter enables local diversity.
3.2.1 DPPs

A discrete DPP defines a distribution over the subsets of a ground set and assigns high probability to a subset if its items are diverse from each other. The notion of diversity is induced by a kernel matrix whose entries can be understood as pairwise similarities between the items. The more similar two items are, the less likely they co-occur in a subset sampled from the DPP.

More concretely, given a ground set \( \mathcal{Y} = \{1, 2, \ldots, N\} \) of \( N \) items, let \( \mathbf{K} \in \mathbb{R}^{N \times N} \) be a symmetric positive semidefinite matrix, called the kernel of DPP. It measures pairwise similarities between the \( N \) items. A distribution over a random subset \( \mathcal{Y} \subseteq \mathcal{Y} \) is a DPP, if for every \( \mathbf{y} \subseteq \mathcal{Y} \) we have

\[
P_{\text{dpp}}(\mathbf{y} \subseteq \mathcal{Y}; \mathbf{K}) = \det(\mathbf{K}_y),
\]

where \( P_{\text{dpp}}(\cdot) \) is the probability of an event, \( \mathbf{K}_y \) denotes a squared submatrix of \( \mathbf{K} \) with rows and columns indexed by \( \mathbf{y} \), and \( \det(\cdot) \) is the determinant of a matrix. All the eigenvalues of the kernel matrix \( \mathbf{K} \) are between 0 and 1. Since \( P(i, j \in \mathcal{Y}; \mathbf{K}) = K_{ii}K_{jj} - K_{ij}^2 \), i.e., the probability of any two items \( i, j \) co-existing in the random subset \( \mathcal{Y} \) is discounted by their similarity \( K_{ij} \). In other words, the subsets whose items are less similar to each other are assigned higher probabilities than the other subsets.

### 3.2.1.1 L-ensemble.

In practice, it is often more convenient to use the so-called L-ensemble DPP that directly assigns atomic probabilities to all the possible subsets of the ground set. Let \( \mathbf{L} \) denote a symmetric positive semidefinite matrix in \( \mathbb{R}^{N \times N} \). The L-ensemble DPP draws a subset \( \mathbf{y} \subseteq \mathcal{Y} \) with probability

\[
P_L(\mathcal{Y} = \mathbf{y}; \mathbf{L}) = \frac{\det(\mathbf{L}_y)}{\det(\mathbf{L} + \mathbf{I})},
\]

(3.2)
where \( I \) is an identity matrix. The corresponding marginal kernel that defines the marginal probability in (3.1) is given by \( K = (L + I)^{-1} \).

### 3.2.1.2 Conditional DPP

One of the appealing properties of DPP is that there exists an analytic form of its conditional distribution. For any \( y_1 \subseteq \mathcal{Y} \) and \( y_0 \subseteq \mathcal{Y} \), \( y_1 \cap y_0 = \emptyset \),

\[
P_L(Y = y_1 \cup y_0 | y_0 \subseteq Y; L) = \frac{\det(L_{y_1 \cup y_0})}{\det(L + I_{\mathcal{Y}\setminus y_0})},
\]

(3.3)

where \( I_{\mathcal{Y}\setminus y_0} \) is a matrix with ones in the diagonal entries indexed by \( \mathcal{Y}\setminus y_0 \) and zeros everywhere else. Kulesza and Taskar have written an excellent tutorial about DPPs [124].

### 3.2.2 Sequential DPPs

A sequential DPP (SeqDPP) [69] was proposed for supervised video summarization. It adheres to the inherent temporal structure in video sequences, thus overcoming the deficiency of DPPs which treat video frames/shots as randomly permutable items. The main technique is to use the conditional DPPs to construct a Markov chain.

Given a long video sequence \( \mathcal{Y} \), we partition it into \( T \) disjoint yet consecutive short segments \( \bigcup_{t=1}^{T} \mathcal{V}_t = \mathcal{Y} \). At the \( t \)-th time step, SeqDPP selects a diverse subset of items (e.g., frames or shots), by a variable \( X_t \subseteq \mathcal{V}_t \), from the corresponding segment conditioning on the items \( x_{t-1} \subseteq \mathcal{V}_{t-1} \) selected from the immediate past segment. This subset selection variable \( X_t \) follows a distribution.
given by the conditional DPP,

\[
P_{seq}(X_t = x_t|X_{t-1} = x_{t-1}, V_t) := P_L(Y_t = x_t \cup x_{t-1}|x_{t-1} \subseteq Y_t; L^t) = \frac{\det(L^t_{x_t \cup x_{t-1}})}{\det(L^t + I_{V_t})},
\]

(3.5)

where \( P_L(Y_t; L^t) \) is an L-ensemble with the ground set \( x_{t-1} \cup V_t \). Denote by \( x_0 = \emptyset \). The SeqDPP over all the subset selection variables is factorized as

\[
P_{seq}({X_t = x_t}_{t=1}^T, V) = \prod_{t=1}^T P_{seq}(X_t = x_t|X_{t-1} = x_{t-1}, V_t).
\]

(3.6)

Figure 3.2 illustrates SeqDPP and compares it to the vanilla DPP and Markov DPP [3]. Unlike the vanilla or Markov DPPs which considers the video frames/shots as orderless items, SeqDPP maintains the temporal order among the segments and yet ignores it among the frames/shots within an individual segment, locally. Furthermore, it retains the diversity property for adjacent video segments but not for those that are far apart. Indeed, users may want to keep visually similar video clips in the summary if they are far apart in a lengthy video in order to tell a complete story of the video.
3.2.3 Reinforcement learning

Consider an agent that takes actions according to some policy to interact with the environment. Following the popular Markov decision process (MDP) formalism, we describe the problem by $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ are the state ($s$) space and action ($a$) space, respectively, $P(s_{t+1}|s_t, a_t)$ is a state transition distribution, $R(s_{t+1}; s_t, a_t)$ is a reward the agent receives if it takes action $a_t$ at state $s_t$ and results in state $s_{t+1}$, and $\gamma \in (0, 1)$ is a discount factor. A policy is denoted by $\pi : \mathcal{S} \mapsto \mathcal{A}$, which is essentially a conditional distribution $\pi(a_t|s_t)$ over the actions given any state. Reinforcement learning aims to find the agent a policy that maximizes the expected total discounted reward $E_\pi \sum_{i=0}^{\infty} \gamma^i R_{t+i}$ starting from time step $t$.

3.3 Reinforcing dynamic SeqDPPs

We are now ready to present our dynamic SeqDPP (DySeqDPP) along with a reinforcement learning algorithm for estimating the model parameters.

3.3.1 DySeqDPP

We describe the DySeqDPP model using the MDP formalism $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$ so that the corresponding learning algorithm follows naturally. We note that, in addition to the new DySeqDPP, another contribution of this section is the reinforcement learning perspective for understanding SeqDPPs. Under this framework, SeqDPP and DySeqDPP can be seen as two types of stochastic policies.

**State $s_t$ at time step $t$:** An information state is about the history of an agent’s observations (and rewards) about the environment. It is used to determine what happens next upon an action.
taken by the agent. In our context, the state \( s_t = \{ \bigcup_{t'=1}^{t-1} x_{t'}, \mathcal{V}_t \} \) comprises the dynamic partition of the video \( \mathcal{V}_t \) at time step \( t \) and the generated video summary \( \bigcup_{t'=1}^{t-1} x_{t'} \) right before the current step \( t \). One may wonder to alternatively treat all the video segments \( \mathcal{V}_1, \ldots, \mathcal{V}_t \) until step \( t \) as the state. We contend that it is oppressive and unnecessary to carry them along over time. Instead, the summary of the past conveys similar amount of information by design.

**Action \( a_t \) at time step \( t \):** In DySeqDPP, the agent takes actions 1) to select a subset \( X_t \) from the video segment \( \mathcal{V}_t \) and 2) to propose the length \( L_t \) of the next segment \( \mathcal{V}_{t+1} \). The subset selection variable \( X_t \subseteq \mathcal{V}_t \) and the partition proposal variable \( L_t \in \mathcal{L} \) jointly define the action space. In other words, an action takes the form of \( A_t = (X_t, L_t) \) whose realization is denote by \( a_t = (x_t, l_t) \). We limit the search of the segment’s length to the range of \( \mathcal{L} = \{5, 6, \ldots, 15\} \) shots.

**Policy \( \pi \):** We let the agent take a stochastic policy in the following manner,

\[
\pi(a_t|s_t) = P(x_t, l_t|s_t) = P(x_t|s_t)P(l_t|x_t, s_t),
\]

where \( P(x_t|s_t) \) is a conditional DPP used to build SeqDPP [69], *i.e.*,

\[
P(x_t|s_t) = P(x_t|\bigcup_{t'=1}^{t-1} x_{t'}, \mathcal{V}_t) := P_L(Y_t = x_t \cup x_{t-1} \subseteq \mathcal{Y}_t; \mathcal{L}^t)
\]

and \( P(l_t|x_t, s_t) \) is defined as a softmax function,

\[
P(l_t|x_t, s_t) = P(l_t|\bigcup_{t'=1}^{t-1} x_{t'}, \mathcal{V}_t) := \text{softmax}(w_t^T \phi(\bigcup_{t'=1}^{t-1} x_{t'}, \mathcal{V}_t)).
\]

There are several points in the above worth clarifying and discussing. First of all, equations (3.7–3.9) describe the main body of our DySeqDPP model. It improves SeqDPP by
the partition proposal variable $L_t$. It is a latent variable because users annotate summaries of videos without explicitly knowing the boundaries of the local diversities they have in their minds. Secondly, we condition the DPP in eq. (3.8) on its immediate past time step $(x_{t-1})$ only instead of the whole history of summaries included in the state $s_t$. This is due to the same modeling intuition as SeqDPP, i.e., in order to maintain local diversity in the summaries. Thirdly, $\phi(\cdot)$ in eq. (3.9) extracts features by max-pooling the representations of all the video shots in the current state $s_t$ as well as the new summary $x_t$ selected according to eq. (3.8). This ensures that sufficient information about both the whole past history and the current of the video is supplied to the softmax for the agent to predict the appropriate length of the next segment. Last but not the least, $\{w_l, l \in L\}$ are the model parameters to be learned from the user-annotated summaries. It is important to note that the parameters are not bound to any particular environments/videos at all, so the policy can be generalized to unseen videos, too. We postpone the parameterization of the L-ensemble DPP’s kernel $L$ to Section 3.3.2.

**State-action value function:** Our goal is to learn a policy to maximize the expected total discounted reward the agent receives, called the state-action value function,

$$Q^\pi(s_0, a_0) := \mathbb{E}_x \left[ \sum_{t=0}^{T} g(\gamma, t) R_t | S_0 = s_0, A_0 = a_0 \right],$$  

(3.10)

where $g(\gamma, t) \in [0, 1]$ is a discount function and the reward $R_t = R(s_{t+1}; s_t, a_t)$ is a function of the state and action. For video summarization, the reward can be evaluation metrics like precision, recall, or F-score computed between the video shots $\cup_{t'=1}^{t} x_{t'}$ selected by the agent and the user summaries of the video (until the current segment $V_t$). The total number of time steps the agent can take is $T$, which satisfies $\sum_{t=0}^{T-1} l_t < |V|$ and $\sum_{t=0}^{T} l_t \geq |V|$.

It is important to note that our goal is to maximize the state-action value function at the initial state.
and action \((s_0, a_0)\) which are fixed to \(s_0 = \emptyset\) and \(a_0 = (x_0 = \emptyset, l_0 = 10)\) in our experiments. In contrast to conventional setups in reinforcement learning, we do not care about the state-action values at other states because only the initial state gives rise to a whole summary of the video, which is our interest. This insight also suggests a special design of the discount function \(g(\gamma, t)\). Instead of using the common practice \(\gamma^t\), we let it be \(g(\gamma, t) = \gamma^{|V| - t} ; \gamma \in (0, 1)\), monotonically increasing with respect to \(t\) in order to weigh the reward of the whole summary more than the incomplete summaries at any other time steps.

Those differences highlight the fact that video summarization actually lacks some characteristics of reinforcement learning (e.g., delayed feedback). Hence, we have to customize the MDP formalism in order to match it with the goal of interest. Nonetheless, by casting DySeqDPP as a policy, we can conveniently learn its model parameters by algorithms in reinforcement learning — we employ gradient descent in this work.

### 3.3.2 Policy gradient descent for learning DySeqDPP

We review the model parameters in DySeqDPP before deriving the learning algorithm. We parameterize two conditional distributions in DySeqDPP for the purpose of out-of-sample extension, so that one can readily apply the learned model to unseen test videos. The first is the partition proposal distribution (eq. (3.9)) and the second is the conditional DPP (eq. (3.8)) at each time step \(t\), whose L-ensemble kernel is constructed as follows,

\[
[L']_{ij} = z_i^T W^T W z_j, \quad z_i = \text{ReLU}(U \text{ReLU}(V f_i))
\]

where \(f_i\) is the feature representation of video shot \(i\) in the ground set \(x_{t-1} \cup V_t\) of the time step \(t\). This feature vector goes through a feedforward network with ReLU activations. Denote by \(\theta\)
the union of the weights of the network \((W, U, V)\) and the unknowns \(\{w_l, l \in L\}\) in eq. (3.8).

We next derive a learning algorithm using the policy gradient descent [238] to estimate the model parameters \(\theta\).

Recall that our goal is to maximize the state-action value function at the initial state and action. Denoting by \(J \triangleq -Q^\pi(s_0, a_0)\), we can minimize it by gradient descent,

\[
\nabla_\theta J|_{\theta = \theta_{old}} = -\mathbb{E}_{\tau \sim \pi(\theta_{old})} \left[ \sum_{t=1}^{T} g(\gamma, t) R_t \nabla_\theta \log p(\tau; \theta)|_{\theta = \theta_{old}} \right] \quad (3.12)
\]

\[
\approx -\frac{1}{K} \sum_{k=1}^{K} \left[ \sum_{t=1}^{T_k} g(\gamma, t) r_{tk} \nabla_\theta \log p(\tau_k; \theta)|_{\theta = \theta_{old}} \right] \quad (3.13)
\]

where the last equation is obtained by sampling \(K\) trajectories \(\{\tau_k\}\) from the policy instantiated by the old parameter \(\theta_{old}\), \(r_{tk}\) is the reward that the agent receives at time step \(t\) of the \(k\)-th trajectory, and the first equation is due to the following fact,

\[
\nabla_\theta \mathbb{E}_{x \sim \theta} [f(x)]|_{\theta = \theta_{old}} = \mathbb{E}_{x \sim \theta_{old}} \left[ \nabla_\theta \log p(x; \theta)|_{\theta = \theta_{old}} f(x) \right]. \quad (3.14)
\]

We still need to work out \(\nabla_\theta \log p(\tau; \theta)\) in eq. (3.13). The key is that the state-transition distribution \(p(s_{t+1}|s_t, a_t)\) is actually deterministic under our context laid out in Sec. 3.3 (because the action \(a_t\) fully determines the summary \(x_t\) and the next segment \(V_{t+1}\), and hence the next state). Therefore, for a trajectory \(s_0, a_0, s_1, a_1, \cdots\), we have

\[
\nabla_\theta \log p(\tau; \theta) = \nabla_\theta \log \left[ p(s_0, a_0) \prod_{t=1}^{T} p(s_t|s_{t-1}, a_{t-1}) \pi(a_t|s_t; \theta) \right] \quad (3.15)
\]

\[
= \nabla_\theta \sum_{t=1}^{T} \log \pi(a_t|s_t; \theta) = \sum_{t=1}^{T} \left[ \nabla_\theta \log P(x_t|s_t) + \nabla_\theta \log P(l_t|x_t, s_t) \right] \quad (3.16)
\]
where the first summand of the last equation is the gradient with respect to the parameters of conditional DPP and the second is of the softmax (eq. (3.9)).

**Implementation:** Instead of computing the gradients explicitly, one may use the “autodiff” feature of many existing deep learning tools to obtain the gradients. Take PYTORCH (http://pytorch.org) for instance. We may program the following for a trajectory,

\[
J(\tau; \theta) = -\sum_{t=1}^{T} g(\gamma, t) r_t \left[ \log P(x_t|s_t; \theta) + \log P(l_t|x_t, s_t; \theta) \right],
\]

and then use the `backward()` function to automatically compute the gradients followed by calling the `step()` function to do a one-step gradient descent. After that, we sample another trajectory and repeat the procedure until the termination condition.

### 3.4 Experiments

We run experiments on three datasets, UTE [136], SumMe [79], and TVSum [236], and compare our approach to several competing baselines.

#### 3.4.1 The UT egocentric (UTE) dataset

**Data and features.**

UTE [136] contains four egocentric videos, each of which lasts between three and five hours long. It captures daily activities such as shopping in a grocery store, having lunch, working, chatting with friends, meeting with colleagues, etc. In addition to the big variety of content, the videos are also quite challenging due to ego motions — as a result, the views change frequently. The motion blur
is more frequent and severe than “third-person” videos. In general, the video shots of an activity are placed in between of blurred frames and nuisance views. Following the experiment protocol of [224], we run four rounds of experiments. In each round, we use two videos for training, one for validation, and the last for testing. We uniformly divide the videos to 5-second shots. From each video frame, we extract 4,096D deep CNN features as the activation of the last fully connected layer of the VGG19 network [231] pretrained on ImageNet [35]. After that, we use PCA to reduce the feature dimension from 4,096D to 512D, followed by max-pooling within each shot in order to have a shot-level feature representation (i.e., $f_i$ in eq. (3.11)).

3.4.1.2 Competing methods.

We mainly compare our approach (DySeqDPP) to the following methods and their variations which, like ours, locally promote diversity in video summaries: SeqDPP [69, 166], dppLSTM [316], and uniform sampling (Uniform). We let the methods automatically work out the lengths of the summaries except for the uniform sampling, to which we supply the lengths of the oracles. For SeqDPP, however, the length of each segment has to be manually set. In addition to the 10-shot segments suggested in the original work [69], we also include the results of segments of 5 shots and 12 shots. Finally, we include another comparison by improving the original SeqDPP with our reinforcement learning algorithm. This is implemented by fixing the partition proposal distribution $P(L_t|x_t, s_t)$ as a Dirac delta function $\delta(L_t = l)$, where $l = 10$ is independent of the time steps. Besides, we learn using the reward of the whole summary by setting $g(\gamma, t) = 0$ for $t < T$ and $g(\gamma, T) = 1$, unless specified otherwise.
3.4.1.3 Evaluation.

In the literature, system-generated summaries have been evaluated in a variety of manners including but not limited to user studies [137], percentage of frames overlapped with user summaries [316], bipartite matching based on distances of low-level visual features [224], etc. Arguably, user study is the “gold” standard, but it is extremely time-consuming. In this work, we instead use the bipartite matching based on a “semantic distance” — pairwise Hamming distance between video shots computed upon the concepts annotated for each shot. This imitates user studies in the sense that the “semantic distance” is strongly correlated with users’ perceptions about the difference between a system-generated summary and an actual user’s summary. The concepts per video shot are borrowed from an earlier work by Sharghi et al. [224], in which the authors asked users to choose from 54 concepts the ones relevant to a given video shot.

Given two summaries (i.e., a system-generated one and a user summary), we construct a bipartite graph between them with the shots as nodes. A node in one part is connected to all the nodes in the other part with edge weights as the (negative) Hamming distance computed from the per-shot concepts [224]. After that, we find the size of the maximum bipartite matching and divide it by the length of the user (system) summary to obtain the recall (precision). Additionally, we improve this metric by removing the edges between the video shots that are more than $K$ time steps away from each other. In other words, if two shots are far away from each other for more than $5K$ seconds, there is no edge between them in the improved evaluation metric.

3.4.1.4 Comparison results.

Figure 3.3 reports the results using the above evaluation scheme at $K = 8, 12, 16, \infty$. Each system-generated summary is compared against three user summaries and the corresponding precision,
recall, and F-measure scores are averaged to reduce user bias. We can see that the proposed DySeqDPP outperforms the competing methods by a large margin. The SeqDPP trained by our reinforcement learning algorithm ranks the second. These results verify the benefit of understanding the SeqDPPs from the novel reinforcement learning perspective. Moreover, the latent variable for dynamically partitioning the videos into segments also helps. It not only removes the need of handcrafting the segments but also gives rise to superior performance over the equally paced segments.

Another intriguing observation is that there is no significant difference among the results of SeqDPP when we change the sizes of the segments (i.e., 5, 10, and 12 shots). It indicates that one can hardly find an “optimal” length for the equally placing segments of SeqDPP, signifying the need
### Table 3.1: Comparison results on UTE evaluated by the bipartite matching F1-score ($K = 12$)

<table>
<thead>
<tr>
<th>Method</th>
<th>$\gamma = 1e^{-20}$</th>
<th>Full $\gamma = 0.2$</th>
<th>Full $\gamma = 0.5$</th>
<th>Full $\gamma = 0.9$</th>
<th>Partial $\gamma = 0.2$</th>
<th>Partial $\gamma = 0.5$</th>
<th>Partial $\gamma = 0.9$</th>
<th>Greedy Sample</th>
<th>Pool Seg</th>
<th>Pool Video</th>
</tr>
</thead>
<tbody>
<tr>
<td>video 1</td>
<td>29.53</td>
<td>28.96</td>
<td>28.03</td>
<td>29.27</td>
<td>28.83</td>
<td>28.33</td>
<td>28.23</td>
<td>27.76</td>
<td>29.19</td>
<td>30.33</td>
</tr>
<tr>
<td>video 2</td>
<td>31.17</td>
<td>30.67</td>
<td>31.61</td>
<td>30.80</td>
<td>32.53</td>
<td>32.07</td>
<td>30.91</td>
<td>29.24</td>
<td>31.20</td>
<td>31.90</td>
</tr>
<tr>
<td>video 3</td>
<td>46.38</td>
<td>45.79</td>
<td>45.88</td>
<td>42.04</td>
<td>45.20</td>
<td>45.23</td>
<td>44.42</td>
<td>43.56</td>
<td>40.40</td>
<td>43.68</td>
</tr>
<tr>
<td>Avg.</td>
<td>33.45</td>
<td>33.08</td>
<td>32.96</td>
<td>32.16</td>
<td>33.15</td>
<td>33.01</td>
<td>32.77</td>
<td>31.09</td>
<td>31.33</td>
<td>32.71</td>
</tr>
</tbody>
</table>

of dynamically partitioning the videos to segments of variable lengths as our DySeqDPP does.

It is a little surprising to see that dppLSTM underperforms uniform sampling. Upon examining the existing works [224, 223] carefully, we find that uniform sampling is actually a very competitive baseline partially because it receives unfair information at inference — length of the oracle summary. Another possible reason is that we did not pre-train the dppLSTM using any additional datasets as done in [316].

#### 3.4.1.5 Ablation study.

Besides, we run some ablation studies to test several variations to our approach and illustrate the quantitative results in Table 3.1. First, instead of sampling $K$ trajectories $\{\tau_k\}$ based on the old policy, we sample the trajectory $\tau$ in a greedy manner, which chooses the subsets with the maximum probability at each step during training. The “Greedy Sample” column in Table 3.1 indicates that greedy sampling produces worse video summarization results. The reason is that the system can not explore the real environment (video) thoroughly under the greedy sampling strategy.

We also study how the hyper-parameter $\gamma$ ($\gamma = 1e^{-20}, 0.2, 0.5, 0.9$) influences the model. Specifically, larger $\gamma$ means we give higher weight to the incomplete summaries at early time steps.
Meanwhile $\gamma = 1e^{-20}$ means we just consider the whole video summary at the final time step. The experimental results in Table 3.1 verify our intuitive assumption that weighing more on the reward of the whole summary is better than on the incomplete summaries at other time steps. In addition, we notice a problem that it is unreasonable to calculate the reward of each time step by comparing the incomplete summary up to the current step with the full user summary (shown in the columns titled “Full $\gamma = 0.2/0.5/0.9$”). To address this problem, we compute the reward by comparing the current system summary with the user summary until this time step, as shown in the column titled “Partial $\gamma = 0.2/0.5/0.9$”. The experimental results verify that the latter kind of reward calculation is more reasonable.

Finally, we also study what features work better for predicting $l_t$. Recall that, for $\phi(\bigcup_{t'=1}^{t} x_{t'}, V_t)$, we concatenate the features of the generated video summary until the current time step and the features of the current segment. We test two alternatives. One is pooling the features of this segment only (PoolSeg) and the other is pooling the features of the whole video sequence up to the current segment (PoolVideo). PoolSeg gives rise to worse results than PoolVideo since it lacks the larger context than the current segment only. PoolVideo is a little worse than and certainly incurs more computation cost than $\phi(\bigcup_{t'=1}^{t} x_{t'}, V_t)$ because pooling over the video encounters redundant information.

\subsection*{3.4.2 SumMe and TVSum dataset}

\subsubsection*{3.4.2.1 Experiment setup.}

In addition to the egocentric videos, we also test our approach on two other popular datasets for video summarization: SumMe \cite{summe} and TVSum \cite{tvsum}. They are both “third-person” video datasets. SumMe consists of 25 consumer videos covering holidays, events, and sports. The
lengths of the videos range from about one to six minutes. TVSum contains 50 videos of 10 categories downloaded from YouTube. The videos are one to five minutes in length.

We follow the same experimental setup as dppLSTM [316] in this work. We extract the output (1,024D) of the penultimate layer (pool 5) of GoogLeNet [242] for each video frame. Followed by max-pooling within each shot (15 frames), we get the shot-level feature representation. In our experiments, we train the model with 60% videos of SumMe (TVSum), validate on 20% of the dataset, and test on the remaining 20% videos. We run 10 rounds of experiments with different
random splits of the dataset and report both the mean F1-scores and standard errors.

3.4.2.2 Evaluation.

We evaluate the results again by F1-score. However, the precisions and recalls for computing the F1-score are calculated in a different way from the bipartite graph matching earlier. Following the practice in dppLSTM [316], we first split a video into a set of disjoint temporal scenes (which are usually longer and contain more visual information than the segments and shots used in the UTE dataset) using the KTS approach [205]. We train the model with shot-level feature representations and then use it to obtain shot-level importance scores. Specifically, the importance score of each frame is equal to the score of shots they belong to. We compute the scene-level scores by averaging the scores of frames within each scene and then rank the scenes in the descending order by their scores. In order to generate a video summary, we select the scenes with a duration below a certain threshold (e.g., using the knapsack algorithm as in [236]). Finally, we calculate the precision, recall, and F1-score according to the temporal overlap between the generated summary and the user summaries.

In order to account for the above evaluation scheme, we make some changes to our reinforcement learning algorithm on these two datasets. For training process, firstly we sample the partition proposal \( l_t \) with oracle summary based on the old policy on each time step. Thus we can utilize the diagonal values of \( L' \) as shot-level scores and then generate the video summary using the approach described above. Consequently, we can get the reward (F1-score) with the generated video summary. Note that the trajectory \( \tau \) here is the oracle summary. Therefore, we can optimize the dynamic SeqDPP with reinforcement learning.
Table 3.2: Comparison results on video summarization on SumMe and TVsum dataset. The results are evaluated by F1-score, the higher the better.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Method</th>
<th>Unsupervised</th>
<th>Canonical</th>
</tr>
</thead>
<tbody>
<tr>
<td>SumMe</td>
<td>Video-MMR [146]</td>
<td>26.6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Gygli et al. [79]</td>
<td>39.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Gygli et al. [80]</td>
<td>39.7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zhang et al. [315]</td>
<td>40.9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>vsLSTM [316]</td>
<td>37.6 ± 0.8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>dppLSTM [316]</td>
<td>38.6 ± 0.8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SeqDPP [69]</td>
<td>40.8 ± 4.8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DySeqDPP</td>
<td>44.3 ± 2.8</td>
<td></td>
</tr>
<tr>
<td>TVSum</td>
<td>LiveLight [319]</td>
<td>46.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Khosla et al. [114]</td>
<td>36.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Song et al. [236]</td>
<td>50.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>vsLSTM [316]</td>
<td>54.2 ± 0.7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>dppLSTM [316]</td>
<td>54.7 ± 0.7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SeqDPP [69]</td>
<td>57.4 ± 2.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DySeqDPP</td>
<td>58.4 ± 2.5</td>
<td></td>
</tr>
</tbody>
</table>

3.4.2.3 Comparison results.

Table 3.2 shows the comparison results between our DySeqDPP and several baselines. Note that some of the baseline methods are unsupervised so they are tuned to achieve the best results on the test set. Nonetheless, the supervised ones in general perform better than them. Both SeqDPP and DySeqDPP significantly outperform the others and DySeqDPP ranks to the first by a big margin on SumMe.

3.4.2.4 Qualitative results.

Figure 3.4 demonstrates some exemplar video summaries generated by SeqDPP and our DySeqDPP, respectively. It is interesting to see that DySeqDPP captures some shots that are key for the
story flow and are yet missed by SeqDPP. Take the first video for instance. The sky diver shows up only at the end of SeqDPP’s summary while s/he is kept at both the beginning and the end of DySeqDPP’s summary. The second is an amusing video recording how a bird saves a ball from a dog’s mouth. However, SeqDPP fails to select the key shot in which the dog bites the ball.

3.5 Summary

In this chapter, we study “how local” the local diversity should be for video summarization and utilize it as a guideline to devise a sequential model to tackle the dynamic diverse subset selection problem. Furthermore, we apply reinforcement inference [239] in the dynamic seqDPP model to solve the problem of exposure bias [211] as well as the issue of non-differentiable metrics existing in SeqDPP [69]. The proposed DySeqDPP can not only seek the appropriate and possibly different lengths of segments dynamically, but also bridge the training and validation phases. Experimental results on video summarization demonstrate the effectiveness of our approach.
CHAPTER 4: IMPROVING OBJECT DETECTION WITH WEB IMAGES

4.1 Problem Introduction

Object detection is a fundamental task in computer vision. It has achieved unprecedented performance for many objects, partially thanks to the recently developed deep neural detectors [214, 149, 213, 245]. Some detectors have made their way into real-world applications, such as smart mobile phones and self-driving cars.

However, upon a careful investigation into the top three teams’ class-wise detection results on 80 common objects in context (COCO) [151], we find that they still fall short of detecting backpacks, handbags, and chairs, among other functional objects. As of the dissertation submission, they report an average precision [151] of less than 0.30 on detecting backpacks and less than 0.40 on chairs.

These man-made objects are defined by their functionalities more than visual appearances, leading to high intra-class variation. Minsky [174] wrote that “there’s little we can find in common to all chairs – except for their intended use.” Grabner et al. [72] quantitatively evaluated the challenges of detecting functional objects like chairs. Another potential reason that contributes to the low performance in detecting backpacks and chairs is that they are too common to draw photographers’ attention. Consequently, they often sit out of the camera focus, appear small, and become occluded in context (cf. Figure 4.1).

How to improve the detection of backpacks, chairs, and other common, “less eye-catching” functional objects? We believe the answer resides on both the quality of training data and the inductive bias of advanced detectors. In this work, we focus on the data aspect and mainly study the potential of unlabeled Web images for improving the detection results of backpacks and chairs, without heavily taxing human raters.

In other words, we study how to leverage the unlabeled Web images to augment human-curated object detection datasets. Web images are diverse and massive, supplying a wide variety of object poses, appearances, their interactions with the context, etc., which may lack in the curated object detection datasets. However, the Web images are out of the distribution of the curated datasets. The domain gap between them calls for a careful design of methods to effectively take advantage of the signals in the Web data.

Our approach is two-pronged. On the one hand, we retrieve a big pool of candidate Web images
via Google Image (https://images.google.com) by using its image-to-image search. The
query set consists of all training images in the original human-curated dataset. Compared with
text-based search, which mainly returns iconic photos, the image-based search gives rise to more
natural images with diverse scenes, schematically reducing the domain mismatch between the
retrieved images and the original datasets.

On the other hand, we propose a novel learning method to utilize the Web images for object de-
tection by drawing inspiration from self-training [221, 285] and self-supervised learning [191, 45,
81, 278, 25, 66], both of which are popular in semi-supervised learning. Our problem is similar to
semi-supervised learning, but there exists a domain gap between the Web images and the curated
datasets. We find that the domain gap fails both self-training and self-supervised learning in their
vanilla forms because the out-of-domain Web images give rise to many inaccurate candidate boxes
and uncalibrated box classification scores. To tackle these challenges, we propose a selective net to
identify high-quality positive boxes and a safe zone for mining hard negative boxes [150, 214]. It
rectifies the supervision signals in Web images, enabling self-training and self-supervised learning
to improve neural object detectors by leveraging the Web images.

The main contributions in this work are as follows. First, we customize self-training for the ob-
ject detection task by a selective net, which identifies positive bounding boxes and assigns some
negative boxes to a safe zone to avoid messing up the hard negative mining in the training of
object detectors. Second, we improve the consistency-based [129, 274, 39] semi-supervised ob-
ject detection [102] by the selective net under our self-training framework. Third, to the best of
our knowledge, this work is the first to explore unlabeled, out-of-domain Web images to augment
curated object detection datasets. We report state-of-the-art results for detecting backpacks and
chairs, along with other challenging objects.
4.2 Augmenting COCO detection with Web images

We augment the training set of COCO detection [151] by retrieving relevant Web images through Google Image (https://images.google.com). We focus on the backpack and chair classes in this work. They represent non-rigid and rigid man-made objects, respectively, and the existing results of detecting them are still unsatisfactory (less than 0.40 \( AP \) on COCO as of March 5th, 2020).

4.2.1 \textit{COCO-backpack, COCO-chair, Web-backpack, and Web-chair.}

COCO is a widely used dataset for object detection, which contains 118k training images and 5k validation images [151]. Out of them, there are 8,714 backpacks in 5,528 training images. We name these images the COCO-backpack query set. Similarly, we have a COCO-chair query set that contains 12,774 images with 38,073 chair instances. Using the images in COCO-backpack and COCO-chair to query Google Image, we obtain 70,438 and 186,192 unlabeled Web images named Web-backpack and Web-chair, respectively. We have removed the Web images that are nearly duplicate with any image in the COCO training and validation sets. Figure 4.2 shows two query images and the retrieved Web images.

\textbf{Labeling Web-backpack.} To facilitate the evaluation of our approach and future research, we label a subset of the Web-backpack images. This subset contains 16,128 images and is selected as follows. We apply a pre-trained R101-FPN object detector [277, 149] to all Web-backpack images and then keep the ones that contain at least one backpack box detected with the confidence score higher than 0.7. We ask three raters to label each survived images. One rater draws bounding boxes over all backpacks in an image. The other two examine the results sequentially. They modify the boxes if they find any problem with the previous rater’s annotation. Please see the supplementary
materials for more details of the annotation procedure, including a full annotation instruction we provided to the raters.

**Relabeling COCO-backpack**\(^1\). Using the same annotation procedure above, we also relabel the backpacks in COCO training and validation sets. The main reason for the relabeling is to mitigate the annotation mismatch between Web-backpack and COCO-backpack caused by different annotation protocols. Another reason is that we observe inconsistent bounding boxes in the existing COCO detection dataset. As Figure 4.3 shows, some raters label the sling bags, while others do not, and some raters enclose the straps in the bounding boxes and others do not. We still ask three raters to label the COCO-backpack training images. For the validation set, we tighten the quality control and ask five raters to screen each image.

**Statistics.** Table 4.1 shows the statistics of the datasets used in this work. We augment the COCO-backpack (chair) by Web-backpack (chair), whose size is about 15 times as the former. “Web-backpack labeled” is for evaluation only.
Figure 4.3: Labeling errors in COCO-backpack. Two images in one group mean that they contain conflict annotations.

Table 4.1: Statistics of the Web images in this work and their counterparts in COCO

<table>
<thead>
<tr>
<th>Dataset</th>
<th># Images</th>
<th># Boxes by COCO</th>
<th># Boxes by us</th>
<th># Raters</th>
</tr>
</thead>
<tbody>
<tr>
<td>COCO-backpack</td>
<td>5,528</td>
<td>8,714</td>
<td>7,170</td>
<td>3</td>
</tr>
<tr>
<td>COCO-backpack validation</td>
<td>5,000</td>
<td>371</td>
<td>436</td>
<td>5</td>
</tr>
<tr>
<td>Web-backpack</td>
<td>70,438</td>
<td>–</td>
<td>23,683</td>
<td>–</td>
</tr>
<tr>
<td>Web-backpack labeled</td>
<td>16,128</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>COCO-chair</td>
<td>12,774</td>
<td>38,073</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>COCO-chair validation</td>
<td>5,000</td>
<td>1,771</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Web-chair</td>
<td>186,192</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

4.3 Selective self-supervised self-training

In this section, we describe our learning method named Selective Self-Supervised Self-training for Object Detection (S\textsuperscript{4}OD). Without loss of generality, we consider detecting only one class of objects from an input image. Denote by \( \mathcal{D} = \{(I_1, T_1), (I_2, T_2), \ldots, (I_n, T_n)\} \) the labeled image set
and $\mathcal{U} = \{\widetilde{I}_1, \widetilde{I}_2, ..., \widetilde{I}_m\}$ the crawled Web images, where $t^i_j = \{x^i_j, y^i_j, w^i_j, h^i_j\} \in \mathcal{T}_i$ contains the top-left coordinate, width, and height of the $j$-th ground-truth bounding box in the $i$-th image $I_i$. The labeled image set $\mathcal{D}$ is significantly smaller than the set of Web images $\mathcal{U}$. Besides, there exists a domain shift between the two sets, although we have tried to mitigate the mismatch by using the image-to-image search. Finally, some Web images could contain zero objects of the class being considered. In the following, we first customize vanilla self-training for object detection, discuss its limitations and fixations by a selective net, and then arrive at the full $\text{S}^4\text{OD}$ algorithm. Figure 4.4 illustrates a diagram of our approach.

### 4.3.1 Self-training for object detection (SOD)

Given the labeled set $\mathcal{D}$ and unlabeled set $\mathcal{U}$, it is natural to test how self-training performs, especially given that it has recently achieved remarkable results [285] on ImageNet [35]. Following the procedure in [285], we first train a teacher object detector $f(I, \theta^*_t)$ from the labeled images, where $\theta^*_t$ stands for the network weights. We then produce pseudo boxes for each unlabeled Web image $\widetilde{I}_i \in \mathcal{U}$:

$$\widetilde{T}_i, \widetilde{S}_i \leftarrow f(\widetilde{I}_i, \theta^*_t), \quad i = 1, ..., m$$

where each pseudo box $\widetilde{t}^i_j \in \widetilde{T}_i$ is also associated with a confidence score $\widetilde{s}^i_j \in \widetilde{S}_i$. We obtain the confidence score from the detector’s classification head. Finally, we train a student detector in a pre-training-and-fine-tuning manner. The idea is to pre-train the student detector using the Web images $\mathcal{U}$ along with the pseudo bounding boxes, followed by fine-tuning it on the curated set $\mathcal{D}$.

Modern object detectors generate hundreds of object candidates per image to ensure high recall even after non-maximum suppression [214], implying that many of the predicted pseudo boxes
Figure 4.4: Overview of the proposed approach. Top: using a small curated dataset to retrieve relevant Web images and to train a teacher detector. Middle: training a selective net to group pseudo boxes predicted by the teacher into positive, negative, and ambiguous groups. Bottom: learning a student detector from the Web-augmented training set with a self-supervised loss.

are incorrect. Traditional self-training used in image classification [285, 221] disregards low-confidence labels when they train the student model. In the same spirit, we only keep the pseudo boxes whose confidence scores are higher than 0.7.
In SOD described above, a crucial step is the selection of pseudo boxes by thresholding the confidence scores. The effectiveness of SOD largely depends on the quality of the selected boxes, which, unfortunately, poorly correlates with the confidence score. Figure 4.5 shows some examples where the pseudo boxes tightly bound the backpacks, but the teacher detector assigns them very low confidence scores. As a result, those boxes would be removed before SOD trains the student detector, under-utilizing the Web images $\mathcal{U}$. What’s worse is that the mistakenly removed boxes could be discovered as false hard negatives during training.

To tackle the challenges, we propose a selective net $g$ to calibrate the confidence scores of the pseudo boxes in the Web images $\mathcal{U}$. The main idea is to automatically group the boxes into three categories: positive, negative, and ambiguity. Denote by $g \circ \tilde{T}_i$ the grouping results of the pseudo boxes in $\tilde{T}_i$ for the $i$-th Web image $\tilde{I}_i$. We pre-train the student detector $f(I, \theta^*_s)$ by the following
(and then fine-tune it on the curated training set $D$),

$$
\theta^*_s \leftarrow \arg \min_{\theta_s} \frac{1}{m} \sum_{i=1}^{m} \ell(g \circ \tilde{T}_i, f(\tilde{I}_i, \theta_s)),
$$

(4.2)

where $\ell(\cdot, \cdot)$ is the conventional loss for training object detectors. For Faster-RCNN [214], the loss consists of regression, classification, objectiveness, etc. All the positive boxes predicted by our selective net $g$ are used to activate those loss terms. In contrast, the ambiguous boxes, which could be correct but missed by the selective net, create a safe zone and do not contribute to any of the loss terms. This safe zone is especially useful when the learning algorithm has a hard negative mining scheme built in because it excludes potentially false “hard negatives” that fall in this ambiguity group.

**Preparing training data for the selective net $g$.** How do we learn the selective net $g$ without knowing any groundtruth labels of the Web images? We seek answers by revisiting the labeled training set $D$ instead. After we learn the teacher detector, we apply it to the training images in $D$ and obtain a large pool of pseudo boxes. We assign each pseudo box $\tilde{I}_i^j$ in the $i$-th image to one of the three groups by comparing it to the groundtruth boxes $T_i$,

$$
g \circ \tilde{I}_i^j = \begin{cases} 
\text{Negative,} & \max_{t \in T_i} \text{IoU}(\tilde{I}_i^j, t) \leq \gamma_l, \\
\text{Positive,} & \max_{t \in T_i} \text{IoU}(\tilde{I}_i^j, t) \geq \gamma_h, \\
\text{Ambiguity,} & \text{otherwise},
\end{cases}
$$

(4.3)

where $\text{IoU}$ is the intersection-over-union, a common evaluation metric in object detection and semantic segmentation [151, 53], and $\gamma_h = 0.6$ and $\gamma_l = 0.05$ are two thresholds for IoU as opposed for the confidence scores. Interestingly, we can choose $\gamma_h$ by using the COCO evaluation protocol [151] as follows. Considering all the boxes in the positive group as the teacher detector’s final output, we can compute their mean average precision (mAP) over the labeled images $D$. We
choose the threshold of $\gamma_h = 0.6$ that maximizes the mAP.

**Preparing features for the selective net** $g$. We accumulate all potentially useful features to represent a pseudo box so that the selective net can have enough information to group the boxes. The feature vector for a box $\tilde{t}_i^j$ is

\[
\begin{pmatrix}
    f_{RoI}(\tilde{t}_i^j, I_i, \theta^*_i), \tilde{s}_i^j, \tilde{x}_i^j/W_i, \tilde{y}_i^j/H_i, \tilde{w}_i^j/W_i, \tilde{h}_i^j/H_i, W_i, H_i
\end{pmatrix}
\] (4.4)

where $f_{RoI}(\tilde{t}_i^j, I_i, \theta^*_i)$ is the RoI-pooled features [214] from the teacher detector, $\tilde{s}_i^j$ is the confidence score, $W_i$ and $H_i$ are respectively the width and height of the $i$-th image, and the others are normalized box coordinate and size.

**Training the selective net** $g$. With the training data (eq. (4.3)) and features (eq. (4.4)) of the pseudo boxes, we learn the selective net by a three-way cross-entropy loss. We employ a straightforward architecture for the selective net. It comprises two towers. One is to process the normalized RoI-pooled features, and the other is to encode the remaining box features. They are both one-layer perceptrons with 512 and 128 hidden units, respectively. We then concatenate and feed their outputs into a three-way classifier.

One may concern that applying the teacher detector to the original training set $D$ may not give rise to informative training data for the selective net because the detector could have “overfitted” the training set. Somehow surprisingly, we find that it is extremely difficult to overfit a detector to the training set, probably due to inconsistent human annotations of the bounding boxes. At best, the detector plays the role of an “average rater” who still cannot reach 100% mAP on the training set whose bounding boxes are provided by different users.
Finally, we boost S\textsuperscript{2}OD by a self-supervised loss based on two considerations. One is that Xie et al. demonstrate that it is beneficial to enforce the student network to learn more knowledge than what the teacher provides (e.g., robustness to artificial noise) [285]. The other is that Jeong et al. show the effectiveness of adding a consistency regularization to semi-supervised object detection [102].

More concretely, we add the following loss to eq. (4.2) for each Web image \( \tilde{I} \),

\[
\ell_i := \sum_{\tilde{t}_j^i \in \tilde{T}_i} \left\| f_{\text{RoI}}(\tilde{t}_j^i, \tilde{I}_i; \theta_s) - f_{\text{RoI}}(\tilde{t}_j^i, \tilde{I}_i; \theta_s) \right\|_2 + \ell(g \circ \tilde{T}_i, f(\tilde{I}_i; \theta_s)), \tag{4.5}
\]

which consists of a consistency term borrowed from [102] and the same detection loss as eq. (4.2) yet over a transformed Web image \( \tilde{I} \) — we explain the details below. They are additional cues to the pseudo boxes provided by the teacher. By learning harder than the teacher from all the supervision using the extra Web data, we expect the student detector to outperform the teacher.

Given a noisy Web image \( \tilde{I} \in \mathcal{U} \), we use the selective net \( g \) to pick up positive boxes \( \tilde{T}(\text{Positive}) \) and limit the consistency loss over them. This small change from [102], which is feasible due to the selective net, turns out vital to the final performance. We transform a Web image \( \tilde{I} \) to \( \tilde{I} \) by randomly choosing an operation from \{rotation by 90, 180, or 270 degrees\} × \{horizontal flip or not\} × \{random crop\}. We use the crop ratio 0.9 and always avoid cutting through positive boxes. Accordingly, we can also obtain the transformed pseudo boxes \( \tilde{v}_j^i \in \tilde{T} \). The bottom panel of Figure 4.4 exemplifies this transformation procedure.
Table 4.2: Comparison results for detecting backpacks and chairs.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Method</th>
<th>Web</th>
<th>$AP_{@[.5,.95]}$</th>
<th>$AP_{@.5}$</th>
<th>$AP_{@.75}$</th>
<th>$AP_{S}$</th>
<th>$AP_{M}$</th>
<th>$AP_{L}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>COCO-backpack</td>
<td>R101-FPN [149]</td>
<td>X</td>
<td>16.45</td>
<td>33.63</td>
<td>15.26</td>
<td>18.29</td>
<td>18.23</td>
<td>17.61</td>
</tr>
<tr>
<td></td>
<td>BD [149]</td>
<td>X</td>
<td>17.34</td>
<td>34.71</td>
<td>15.13</td>
<td>19.03</td>
<td>19.46</td>
<td>21.31</td>
</tr>
<tr>
<td></td>
<td>SOD [221]</td>
<td>✓</td>
<td>17.62</td>
<td>33.83</td>
<td>16.71</td>
<td>18.98</td>
<td>21.88</td>
<td>15.93</td>
</tr>
<tr>
<td></td>
<td>CSD [102]</td>
<td>✓</td>
<td>17.87</td>
<td>32.97</td>
<td>16.55</td>
<td>20.42</td>
<td>22.05</td>
<td>20.45</td>
</tr>
<tr>
<td></td>
<td>CSD-Selective (ours)</td>
<td>✓</td>
<td>18.32</td>
<td>33.35</td>
<td>17.05</td>
<td>20.84</td>
<td>21.08</td>
<td>21.11</td>
</tr>
<tr>
<td></td>
<td>$S^2$OD (ours)</td>
<td>✓</td>
<td>18.28</td>
<td>35.35</td>
<td>18.46</td>
<td>19.23</td>
<td>21.76</td>
<td>23.94</td>
</tr>
<tr>
<td></td>
<td>$S^4$OD (ours)</td>
<td>✓</td>
<td><strong>19.48</strong></td>
<td><strong>36.25</strong></td>
<td><strong>19.01</strong></td>
<td>20.31</td>
<td><strong>23.47</strong></td>
<td>18.53</td>
</tr>
<tr>
<td>COCO-chair</td>
<td>R101-FPN [149]</td>
<td>X</td>
<td>28.28</td>
<td>48.93</td>
<td>28.57</td>
<td>19.14</td>
<td>33.21</td>
<td>42.06</td>
</tr>
<tr>
<td></td>
<td>BD [149]</td>
<td>X</td>
<td>29.56</td>
<td>49.15</td>
<td>30.70</td>
<td>20.16</td>
<td>36.39</td>
<td>41.99</td>
</tr>
<tr>
<td></td>
<td>SOD [221]</td>
<td>✓</td>
<td>30.01</td>
<td>49.05</td>
<td>31.53</td>
<td>20.32</td>
<td>37.10</td>
<td>43.12</td>
</tr>
<tr>
<td></td>
<td>CSD [102]</td>
<td>✓</td>
<td>30.19</td>
<td>48.58</td>
<td>31.92</td>
<td>20.77</td>
<td>37.19</td>
<td>43.74</td>
</tr>
<tr>
<td></td>
<td>CSD-Selective (ours)</td>
<td>✓</td>
<td>30.95</td>
<td>49.75</td>
<td>32.11</td>
<td>21.03</td>
<td>38.02</td>
<td>44.96</td>
</tr>
<tr>
<td></td>
<td>$S^2$OD (ours)</td>
<td>✓</td>
<td>30.95</td>
<td>50.25</td>
<td>31.92</td>
<td><strong>21.19</strong></td>
<td>37.82</td>
<td>45.48</td>
</tr>
<tr>
<td></td>
<td>$S^4$OD (ours)</td>
<td>✓</td>
<td><strong>31.74</strong></td>
<td><strong>51.15</strong></td>
<td><strong>33.29</strong></td>
<td>20.57</td>
<td><strong>38.66</strong></td>
<td><strong>46.93</strong></td>
</tr>
</tbody>
</table>

4.4 Experiments

We augment the COCO-backpack and COCO-chair training images with unlabeled Web images and run extensive experiments to test our approach on them. We note that the Web images are out of the distribution of COCO [151]. Some of them may contain no backpack or chair at all, and we rely on our selective net to identify the useful pseudo boxes in them produced by a teacher detector.

Besides, since $S^4$OD is readily applicable to vanilla semi-supervised object detection, whose labeled and unlabeled sets follow the same distribution, we also test it following the experiment protocol of [102].

4.4.1 Augmenting COCO-backpack(chair) with Web-backpack(chair)

We compare $S^4$OD to the following competing methods.
**R101-FPN [149]:** We use R101-FPN implemented in Detectron2 [277] as our object detector. Its $AP$ on COCO-validation is 42.03, which is on par with the state of the arts. However, this detector still performs unsatisfactorily on the backpack and chair classes. As shown in Table 4.2, it only achieves 16.45 $AP$ on detecting backpacks and 28.28 $AP$ on chairs.

**BD [149]:** We finetune R101-FPN on COCO-backpack and COCO-chair, respectively, by changing the 80-class detector to a binary backpack or chair detector. We observe that the binary detector outperforms its 80-class counterpart by about 1% (0.9% $AP$ for backpacks and 1.3% $AP$ for chairs). We denote by BD the binary backpack (chair) detector and use it as the teacher detector for the remaining experiments.

**CSD [102]:** We include the recently published consistency-based semi-supervised object detection (CSD) in the experiments. We carefully re-implement it by using the same backbone detector (R101-FPN) as ours. CSD imposes the consistency loss over all candidate boxes.

**CSD-Selective:** We also report the results of applying our selective net to CSD. We remove the loss terms in CSD over the boxes of negative and ambiguity groups predicted by the selective net. Table 4.2 shows that it increases the performance of the original CSD by 0.45 $AP$ on detecting backpacks and 0.76 $AP$ on chairs.

**SOD [221]:** We presented a vanilla self-training procedure for object detection (SOD) in Section 4.3.1. Unlike self-training in image classification [285], we find that SOD is sensitive to the threshold of confidence scores probably because the out-of-distribution Web images make the scores highly uncalibrated. We test all thresholds between 0.5 and 0.9 (with an interval of 0.2) and find that SOD can only beat BD at the threshold of 0.7 (reported in Table 4.2).

**S$^2$OD (ours):** This is an improved method over SOD by employing the selective net (cf. Section 4.3.2). It is also an ablated version of our S$^4$OD by removing the self-supervised loss.
Table 4.2 presents the comparison results on the validation sets of both COCO-backpack and COCO-chair. We can see that $S^2$OD performs consistently better than its teacher detector (BD), the vanilla self-training (SOD), and the consistency-based self-supervised learning (CSD). Our full approach ($S^4$OD) brings additional gains and gives rise to 19.48 $AP$ on detecting COCO backpacks and 31.74 $AP$ on detecting chairs — about 2% better than its original teacher (BD). Besides, the improvements of $S^2$OD over SOD and CSD-selective over CSD both attribute to the selective net. Finally, Figure 4.6 shows some cases where the selective net correctly groups low(high)-confidence boxes into the positive (negative) group.

In addition to the overall comparison results in Table 4.2, we next ablate our approach and examine some key components. We also report the “upper-bound” results for augmenting COCO-backpack with Web-backpack by using the bounding box labels we collected for a subset of Web-backpack.

**Web-backpack vs. text-to-image search.** The image-to-image search for Web images is the very first step of our approach, and it is superior over the text-to-image search in various aspects. As the leftmost panel in Figure 4.7 shows, most top-ranked Web images are iconic with salient objects sitting in clean backgrounds if we search using class names. For COCO [151], the images are collected from Flickr by complex object-object and object-scene queries. Using the same technique,
we can retrieve more natural images. However, they are mostly recent and come from diverse sources, exhibiting a clear domain shift from the COCO dataset which is about six years old. In contrast, the image-to-image search well balances between the number of the retrieved Web images and their domain similarity to the query images (cf. the right panel in Figure 4.7).

Table 4.3 compares image-to-image search with text-to-image search by their effects on the final results. The S^4OD-text row is the results obtained using 230k Web images crawled by text-to-image. While they are slightly better than BD’s results, they are significantly worse than what S^4OD achieves with 70k Web images retrieved by image-to-image search (cf. Table 4.1).

**Web-backpack: the size matters.** We study how the number of the unlabeled Web images influences the proposed S^4OD by training with 1/3 and 2/3 of the crawled Web-backpack. As shown

<table>
<thead>
<tr>
<th>Method</th>
<th>AP@[.5, .95]</th>
<th>AP@.5</th>
<th>AP@.75</th>
<th>AP_S</th>
<th>AP_M</th>
<th>AP_L</th>
</tr>
</thead>
<tbody>
<tr>
<td>BD [149]</td>
<td>17.34</td>
<td>34.71</td>
<td>15.13</td>
<td>19.03</td>
<td>19.46</td>
<td>21.31</td>
</tr>
<tr>
<td>S^4OD-text</td>
<td>17.37</td>
<td>34.26</td>
<td>14.46</td>
<td>18.77</td>
<td>20.10</td>
<td>17.66</td>
</tr>
<tr>
<td>S^4OD w/ 1/3 Web-backpack</td>
<td>18.40</td>
<td>34.12</td>
<td>17.91</td>
<td>19.25</td>
<td>21.20</td>
<td>21.56</td>
</tr>
<tr>
<td>S^4OD w/ 2/3 Web-backpack</td>
<td>19.08</td>
<td>35.35</td>
<td>17.96</td>
<td>21.04</td>
<td>20.15</td>
<td>21.27</td>
</tr>
<tr>
<td>S^4OD w/ full Web-backpack</td>
<td>19.48</td>
<td>36.25</td>
<td>19.01</td>
<td>20.31</td>
<td><strong>23.47</strong></td>
<td>18.53</td>
</tr>
<tr>
<td>S^4OD-2nd Iteration</td>
<td><strong>19.52</strong></td>
<td><strong>36.44</strong></td>
<td><strong>19.07</strong></td>
<td><strong>21.70</strong></td>
<td>20.90</td>
<td><strong>24.90</strong></td>
</tr>
</tbody>
</table>
Table 4.4: Comparison results on the relabeled COCO-backpack.

<table>
<thead>
<tr>
<th>Method</th>
<th>$AP_{@[.5,.95]}$</th>
<th>$AP_{@.5}$</th>
<th>$AP_{@.75}$</th>
<th>$AP_{S}$</th>
<th>$AP_{M}$</th>
<th>$AP_{L}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>BD</td>
<td>18.75</td>
<td>36.03</td>
<td>16.98</td>
<td>11.90</td>
<td>21.62</td>
<td>31.58</td>
</tr>
<tr>
<td>Upper bound</td>
<td>22.63</td>
<td>40.89</td>
<td>21.15</td>
<td>15.97</td>
<td>26.53</td>
<td>32.11</td>
</tr>
<tr>
<td>$S^4$OD</td>
<td>20.27</td>
<td>37.55</td>
<td>20.49</td>
<td>13.51</td>
<td>24.04</td>
<td>31.17</td>
</tr>
</tbody>
</table>

in table 4.3, $S^4$OD-1/3 can improve over BD by 1% $AP$, and $S^4$OD-2/3 is better than BD by 1.7% $AP$. In contrast, $S^4$OD with the full Web-backpack leads to about 2.1% $AP$ improvement. Overall, we see that the more Web images, the larger boost in performance, implying that it is worth studying the data pipeline and expanding its coverage in future work.

Iterating $S^4$OD. What happens if we use the detector trained by $S^4$OD as the teacher and train another student detector using $S^4$OD again? $S^4$OD-2nd Iteration in table 4.3 outperforms the single-iteration version only by a very small margin. It is probably because, in the second iteration, the student detector does not receive more supervision than what the teacher BD provides. We will explore some stochastic self-supervised losses in future work to enforce the student to learn more than what the teacher provides at every iteration.

An “upper bound” of $S^4$OD. We further investigate the effectiveness of $S^4$OD by comparing it to an “upper bound”. We run the experiments using the labeled subset of Web-backpack and the relabeled COCO-backpack to have consistent annotations across the two datasets. Recall that we have fixed some inconsistent bounding boxes in the original COCO-backpack in the relabeling process. As a result, if we train BD [214] on the relabeled COCO-backpack and evaluate on the relabeled validation set, the $AP$ is 18.75 (cf. Table 4.4), in contrast to 17.34 $AP$ (cf. Table 4.2) by BD trained and evaluated using the original COCO-backpack. Using this BD as the teacher, we train a student detector using $S^4$OD. By the “upper bound”, we pool the labeled subset of Web-backpack and the relabeled COCO-backpack together and then train BD. The results in Table 4.4
Table 4.5: Comparison results on VOC2007 (* the numbers reported in [102])

<table>
<thead>
<tr>
<th>Method</th>
<th>Labeled data</th>
<th>Unlabeled data</th>
<th>(AP)</th>
<th>Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supervised [32, 214]</td>
<td>VOC07</td>
<td>–</td>
<td>*73.9/74.1</td>
<td>–</td>
</tr>
<tr>
<td>Supervised [32, 214]</td>
<td>VOC07&amp;12</td>
<td>–</td>
<td>*79.4/80.3</td>
<td>–</td>
</tr>
<tr>
<td>CSD [102]</td>
<td>VOC07</td>
<td>VOC12</td>
<td>*74.7</td>
<td>*0.8</td>
</tr>
<tr>
<td>SOD</td>
<td>VOC07</td>
<td>VOC12</td>
<td>74.8</td>
<td>0.7</td>
</tr>
<tr>
<td>(S^2)OD</td>
<td>VOC07</td>
<td>VOC12</td>
<td>75.8</td>
<td>1.7</td>
</tr>
<tr>
<td>(S^4)OD</td>
<td>VOC07</td>
<td>VOC12</td>
<td>\textbf{76.4}</td>
<td>\textbf{2.3}</td>
</tr>
</tbody>
</table>

indicate that \(S^4\)OD is almost right in the middle of the lower-bound (BD) and the upper bound. The gap between \(S^4\)OD and the upper bound is small. We will study better learning methods to close the gap and how the Web data volume could impact the performance in future work.

4.4.2 Semi-supervised object detection on PASCAL VOC

It is straightforward to extend \(S^4\)OD to multiple object detection tasks. Following the experiment protocol in [102], we further validate it in the setting of semi-supervised object detection, whose labeled and unlabeled sets are drawn from the same distribution. We use PASCAL VOC2007 trainval (5,011 images) as the labeled set and PASCAL VOC2012 trainval (11,540 images) as the unlabeled set. There are 20 classes of objects to detect. We test all detectors on the test set of PASCAL VOC2007 (4,952 images). In order to make our results comparable to what are reported in [102], we switch to Faster-RCNN [214] with ResNet-50 [84] as the base detector.

Table 4.5 shows the comparison results. We include both our results and those reported in [101] and mark the latter by *. Considering the object detector trained on VOC2007 as a baseline and the one trained on both datasets with labels as an upper bound, our \(S^4\)OD is right in the middle, outperforming CSD by 1.5 \(AP\). Although we propose the selective net mainly to handle noisy Web images, it is delightful that the resulting method also works well with the clean, in-domain
VOC2012 images. It is probably because the consistency loss in CSD drives the detector toward high-entropy, inaccurate predictions (cf. more discussions in [285]). Our selective net avoids this caveat by supplying high-quality boxes to the consistency loss.

4.5 Summary

In this chapter, we propose a novel approach to improving object detection with massive unlabeled Web images. We collect the Web images with image-to-image search, leading to smaller domain mismatch between the retrieved Web images and the curated dataset than text-to-image search does. Besides, we incorporate a principled selective net into self-training rather than threshold confidence scores as a simple heuristic for selecting bounding boxes. Moreover, we impose a self-supervised training loss over the high-quality boxes chosen by the selective net to make better use of the Web images. The improvement in detecting challenging objects is significant over the competing methods, Our approach works consistently well on not only the Web-augmented object detection but also the traditional semi-supervised object detection.
CHAPTER 5: RANKING NEURAL CHECKPOINTS

5.1 Problem Introduction

There is an increasing number of pre-trained deep neural networks (DNNs), which we call checkpoints. We may produce hundreds of intermediate checkpoints when we sweep through various learning rates, optimizers, and losses to train a DNN. Furthermore, semi-supervised [22, 11, 212, 129, 250, 175, 167, 15] and self-supervised [43, 81, 25, 262, 188] learning make it feasible to harvest DNN checkpoints with scarce or no labels. Fine-tuning [300, 193] has become a de facto standard to adapt the pre-trained checkpoints to target tasks. It leads to faster convergence [44, 82, 225] and better performance [120] on the downstream tasks.

However, not all checkpoints are equally useful for a target task, and some could even underperform a randomly initialized checkpoint (cf. Section 5.2.2). This work is concerned with ranking neural checkpoints, which aims to measure how effectively fine-tuning can transfer knowledge from the pre-trained checkpoints to the target task. The measurement should be generic enough for all the neural checkpoints, meaning that it works without knowing any pre-training details (e.g., pre-training examples, hyper-parameters, losses, early stopping stages, etc.) of the checkpoints. It also should be lightweight, ideally without training on the downstream task, to make it practically useful. We may use the measurement to choose the top few checkpoints before running fine-tuning, which is computationally more expensive than calculating the measurements.

Ranking neural checkpoints is crucial. Some domains or applications lack large-scale human-curated data, like medical images [209], raising a pressing need for high-quality pre-trained checkpoints as a warm start for fine-tuning. Fortunately, there exist hundreds of thousands of checkpoints of popular neural network architectures. For instance, many computer vision models are built upon ResNet [84], Inception-ResNet [241], and VGG [231]. As a result, we can construct a candidate pool by collecting the checkpoints released by different groups, for various tasks, and over distinct datasets.

It is nontrivial to rank the checkpoints for a downstream task. We explain this point by drawing insights from the related, yet arguably easier, task transferability problem [1, 51, 303, 185], which aims to provide high-level guidance about how well a neural network pre-trained in one task might transfer to another. However, not all checkpoints pre-trained in the same source task transfer equally well to the target task [326, 120]. The pre-training strategy also matters. Zhai et al. [306] find that combining supervision with self-supervision improves a network’s transfer results on downstream tasks. He et al. [81] also show that self-supervised pre-training benefits object detection more than its supervised counterpart under the same fine-tuning setup.

We may also appreciate the challenge in ranking neural checkpoints by comparing it with another related line of work: predicting DNNs’ generalization gaps [183, 109, 12]. Jiang et al. [103] use a linear regressor to predict a DNN’s generalization gap, i.e., the discrepancy between its training and test accuracies, by exploring the training data’s margin distributions. Other signals studied in the literature include network complexity and noise stability. Ranking neural checkpoints is more challenging than predicting a DNN’s generalization gap. Unlike the training and test sets that share the same underlying distribution, the downstream task may be arbitrarily distant from the source task over which a checkpoint is pre-trained. Moreover, we do not have access to the pre-training data at all. Finally, instead of keeping the networks static, fine-tuning dramatically changes all weights of the checkpoints.
We establish a neural checkpoint ranking benchmark (NeuCRaB) to study the problem systematically. NeuCRaB covers various checkpoints pre-trained on widely used, large-scale datasets by different training strategies and architectures at a range of early stopping stages. It also contains diverse downstream tasks, whose training sets are medium-sized, making it practically meaningful to rank and fine-tune existing checkpoints. Pairing up all the checkpoints and downstream tasks, we conduct careful fine-tuning with thorough hyper-parameter sweeping to obtain the best transfer accuracy for each checkpoint-downstream-task pair. Hence, we know the groundtruth ranking of the checkpoints for each downstream task according to the final accuracies (over the test/validation set).

A functional checkpoint ranking measurement should be highly correlated with the groundtruth ranking and, equally importantly, incurs as less computation cost as possible. We study several intuitive methods for ranking the neural checkpoints. One is to freeze the checkpoints as feature extractors and use a linear classifier to evaluate the features’ separability on the target task. Another is to run fine-tuning for only a few epochs (to avoid heavy computation) and then evaluate the resulting networks on the target task’s validation set. We also estimate the mutual information between labels and the features extracted from a checkpoint.

Finally, we propose a lightweight measure, named Gaussian LEEP (ΛLEEP), to rank checkpoints based on the recently proposed log expected empirical prediction (LEEP) [185]. LEEP was originally designed to measure between-task transferabilities. It cannot handle the checkpoints pre-trained by unsupervised or self-supervised learning since it requires all checkpoints to have a classification head. Its computation cost could blow up when the classification head corresponds to a large output space. Moreover, it depends on the classification head’s probabilistic output, which, unfortunately, is often overly confident [74].

To tackle the above problems, we replace the checkpoints’ output layer with a Gaussian mixture
model (GMM). This simple change kills two birds with one stone. On the one hand, GMM’s soft
assignment of input to clusters seamlessly applies to LEEP, resulting in the lightweight, effective
LEEP measure that works regardless of the checkpoints’ output types. On the other hand, since
we fit GMM to the target task’s data, instead of the pre-training data of a different source task, the
cluster assignment probabilities are likely more calibrated than the classification probabilities (if
there exist classification heads).

5.2 The Neural Checkpoint Ranking Benchmark (NeuCRaB)

We formalize ranking neural checkpoints as follows. Suppose we have \( m \) pre-trained neural net-
works, called checkpoints, \( C := \{\theta_i\}_{i=1}^m \). Denote by \( T \) a distribution of tasks. Without loss of
generality, we study classification downstream tasks, each of which, \( t \sim T \), contains a training set
and a test set. An evaluation procedure, \( G : C \times T \mapsto \mathbb{R} \), replaces the output layer of a checkpoint
\( \theta_i \) with a linear classifier for a downstream task \( t \), followed by fine-tuning using the task’s training
set. It employs hyper-parameter sweeping and returns the best accuracy on the test set. We apply
this evaluation procedure to all the checkpoints for the task \( t \) and obtain their test accuracies:

\[
G_t := \{G(\theta_i, t)\}_{i=1}^m \in \mathbb{R}^m, \quad (5.1)
\]

which defines the groundtruth ranking list for the task \( t \).

Denote by \( \mathcal{R} \) all measures that return a ranking score for any checkpoint-task pair under a compu-
tation budget \( b \). A measure \( R \in \mathcal{R} \) gives rise to the following ranking scores for a task \( t \),

\[
R_t := \{R(\theta_i, t; b)\}_{i=1}^m \in \mathbb{R}^m, \quad (5.2)
\]

where we underscore the computation budget \( b \) in the measure.
Figure 5.1: Fine-tuning the checkpoints in Group I on four downstream tasks. We keep the best fine-tuning accuracy for each checkpoint-task pair after hyper-parameter sweeping. For better visualization, the values are offset by their mean (cf. Table 4 in Appendix for the absolute values). (Best viewed in color. Red: generative models. Black: From-Scratch. Green: self-supervised models. Blue: semi-supervised models. Yellow, Pink, and Orange: supervised models trained on ImageNet, Inatualist, and Places365, respectively. Cyan: Hybridly-supervised model.)

Our objective in ranking neural checkpoints is to find the best ranking measure in expectation,

$$ R^* \leftarrow \arg \max_{R \in \mathcal{R}} E_{t \sim \mathcal{T}} \mathcal{M}(R_t, G_t) $$

(5.3)

where $\mathcal{M}$ is a metric evaluating the ranking scores $R_t$ against the test accuracies $G_t$. Section 5.2.3 details the evaluation methods used in this work. Equipped with such a ranking measure $R^*$, we can identify the checkpoints that potentially transfer to a downstream task better than the others without resorting to heavy computation.


5.2.1 Downstream Tasks $\mathcal{T}$

Following the design principle of [306], we study diverse downstream tasks including Caltech101 [56], Flowers102 [187], Sun397 [281], and Patch Camelyon [262]. These tasks are representative of general object recognition, fine-grained object recognition, scenery image classification, and medical image classification, respectively. Table 1 in Appendix A.1 provides more details of these tasks. A common theme is that their training sets are all medium-sized, making it especially beneficial to leverage pre-trained checkpoints to avoid overfitting.

5.2.2 Neural Checkpoints $C$

Thanks to the broad use of DNNs, one may collect neural checkpoints of various types from multiple sources. To simulate this situation, we construct a rich set of checkpoints and separate them into three groups according to the pre-training strategies and network architectures.

**Group I: Checkpoints of mixed supervision.** The first group of checkpoints are pre-trained with mixed supervision till convergence, including supervised learning, self-supervised learning, semi-supervised learning, and the discriminators or encoders in deep generative models. It consists of 16 ResNet-50s [84]. We borrow 14 models pre-trained on ImageNet [35] from [306]. Among them, four are pre-trained by self-supervised learning (Jigsaw [188], Relative Patch Location [43], Exemplar [47], and Rotation [66]), six are the discriminators of generative models (WAE-UKL [217], WAE-GAN, WAE-MMD [254], Cond-BigGAN, Uncond-BigGAN [18], and VAE [119]), two are based on semi-supervised learning (Semi-Rotation-10% and Semi-Exemplar-10% [305]), one is by fully supervised learning (Sup-100%-Img [84]), and one is trained with a hybrid supervised loss (Sup-Exemplar-100% [305]). We also add two supervised checkpoints pre-trained on iNaturalist (Sup-100%-Inat) [261] and Places365 (Sup-100%-Pla) [321], respectively. Using the evaluation
procedure $G(\theta_i, t)$ (cf. equation (5.1)), we obtain their final accuracies on the downstream tasks described in Section 5.2.1.

Figure 5.1 shows the best fine-tuning accuracies offset by their mean, for better visualization, and Table 4 (in Appendix) contains the absolute accuracy values. We include the training from scratch (From-Scratch) for comparison. Most of the checkpoints yield significantly better fine-tuning results than From-Scratch. Some of the discriminators in generative models, however, under-perform From-Scratch. The highest-performance checkpoints change from one downstream task to another.

**Group II: Checkpoints at different pre-training stages.** This group comprises 12 ResNet-50s pre-trained by fully supervised learning on ImageNet, iNaturalist, and Places-365. We save a checkpoint right after each learning rate decay, resulting in four checkpoints per dataset. Figure 2 and Table 5 in Appendix show the best fine-tuning accuracies over the four downstream tasks, where Img-90k refers to the checkpoint trained on ImageNet for 90k iterations. Interestingly, the downstream tasks favor different pre-training sources, indicating the necessity of studying between-task transferabilities [306, 303]. However, the source task information may be not known for all checkpoints. Moreover, the converged model over a source task does not always transfer the best to a downstream task (cf. Img-270k vs. Img-300k on Camelyon, Inat-270k vs. Inet-300k on Flowers102, etc.). We hence construct this NeuCRaB for studying the ranking of neural checkpoints without accessing how one pre-trained the checkpoints over which dataset.

**Group III: Checkpoints of heterogeneous architectures.** Kornblith et al. [120] show that better network architectures can learn better features that can be transferred across vision-based tasks. Therefore, we construct the third group of checkpoints by using different neural architectures. Four of them belong to the Inception family [242], one is Inception-ResNet-v2 [241], six come from the MobileNet family [93], and two are from ResNet-v1 family [84]. We train them on ImageNet till convergence. Figure 3 and Table 6 in Appendix visualize their fine-tuning accuracies on the four
downstream tasks.

5.2.3 Evaluation Metrics \( M \)

We use multiple metrics (cf. \( M \) in eq. (5.3)) to evaluate the checkpoint ranking measures.

**Recall@\( k \):** A practitioner may have resources to test up to \( k \) checkpoints for their task of interest. We consider it a success if a measure ranks the highest-performance checkpoint into the top \( k \). The measure’s Recall@\( k \) is the ratio between the number of downstream tasks on which it succeeds and the total number of tasks. We employ \( k = 1 \) and \( k = 3 \) in the experiments.

**Top-\( k \) relative accuracy (Rel@\( k \)):** Given a task, a ranking measure returns an ordered list of the checkpoints. If the measure selects a high-performing checkpoint to the top \( k \) despite that it misses the highest-performance one, we do not want to overly penalize it. This Rel@\( k \) is the ratio between the best fine-tuning accuracy on the downstream task with the top \( k \) checkpoints and the the best fine-tuning accuracy with all the checkpoints.

**Pearson correlation:** We incorporate Pearson’s \( r \) [201] to compute the linear correlation between a measure’ ranking scores \( R_t \) and the evaluation procedure’s final accuracies \( G_t \).

**Kendall ranking correlation:** We also include Kendall’s \( \tau \) [113] to measure the ordinal association between a ranking measure \( R \) and the evaluation procedure \( G \) for each task. After all, what matter is the order of the checkpoints rather than the precise ranking scores.

5.3 Checkpoint Ranking Methods

In this section, we describe some intuitive neural checkpoint ranking methods. These methods strive to achieve high correlation with the checkpoint evaluation procedure \( G \) at low computation.
5.3.1 Fine-tuning with Early Stopping

If there is no constraint over computing, the evaluation procedure $G$ itself becomes the gold ranking measure. Hence, a natural ranking method is the fine-tuning with early stopping, by which the model is far from convergence. The premature models’ test accuracies are the ranking scores. Experiments reveal that it is hard to forecast from the premature models.

5.3.2 Linear Classifiers

We derive the second ranking method also from the evaluation procedure $G$, which replaces a checkpoint’s output layer by a linear classifier tailored for the downstream task. We train the linear classifier while freezing the other layers. The ranking score equals the classifier’s test accuracy. It is worth mentioning that self-supervised learning [25, 81, 73] often adopts this practice as well to evaluate the learned feature representations. We shall see that the linear separability of the features extracted from a checkpoint is a strong indicator of the performance of fine-tuning the full checkpoint.

5.3.3 Mutual Information

Suppose the extracted features’ quality well correlates with a checkpoint’s final accuracy on a downstream task. Besides the linear separability above, we can rank the checkpoints by their mutual information between the high-dimensional features and discrete labels of the downstream task. We employ the state-of-the-art $I_\alpha$ mutual information estimator [204], where $\alpha$ controls the trade-off between variance and bias. It is a variational lower bound parameterized by a neural
network. Belghazi et al. [13] report that the neural estimators generally outperform prior mutual information estimations, especially when the variables are high-dimensional. We use the code released by the authors to calculate $I_\alpha$ [204].

### 5.3.4 LEEP for the Checkpoints with Classification Heads

To rank the checkpoints pre-trained over classification source tasks, the recently proposed LEEP [185] measure is directly applicable despite that it was originally designed for between-task transfer. Denote by $\mathcal{Z}$ the classification space of a checkpoint $\theta$. We can interpret $\theta(x)_z$, the $z$-th (softmax) output element, as the probability of classifying the input $x$ into the class $z \in \mathcal{Z}$. Given a downstream task $t \sim \mathcal{T}$ and its test set $\{(x_j, y_j)\}_{j=1}^n$, the LEEP ranking score for the checkpoint $\theta$ is calculated by

$$R_{\text{LEEP}}(\theta, t) := \frac{1}{n} \sum_{j=1}^{n} \log P(y_j|x_j, \theta, t)$$

$$P(y|x, \theta, t) := \sum_{z \in \mathcal{Z}} \hat{P}(y|z)\theta(x)_z$$

where $\hat{P}(y|z)$ is the empirical conditional distribution of the downstream task’s label $y$ given the source label $z \in \mathcal{Z}$, and $P(y|x, \theta, t)$ is a “dummy” classifier, which firstly draws a label $z$ from the checkpoint $\theta(x)$ and then draws a class $y$ from the conditional distribution $\hat{P}(y|z)$.

Denote by $\{(x_j, y_j)\}_{j=1}^\tilde{n}$, $y \in \mathcal{Y}$, the downstream task’s training set. LEEP computes the conditional distribution $\hat{P}(y|z)$ by “counting”. The joint distribution $\hat{P}(y, z)$ due to the checkpoint $\theta$ is

$$\hat{P}(y, z) = \frac{1}{\tilde{n}} \sum_{j: y_j = y} \theta(x_j)_z,$$

which gives rise to the conditional distribution $\hat{P}(y|z) = \hat{P}(y, z)/\hat{P}(z) = \hat{P}(y, z)/\sum_{y \in \mathcal{Y}} \hat{P}(y, z)$. 67
In the experiments, LEEP and the linear classifier are the second best ranking methods for the checkpoints pre-trained for classification. However, LEEP’s computation cost is high when a checkpoint’s classification output is high-dimensional (e.g., iNaturalist contains more than 8000 classes). Besides, its softmax estimation of the classification probability \( \theta(x)_z \) is often poorly calibrated [74]. Finally, it does not apply to the checkpoints with no classification heads.

5.3.5 \( N \)LEEP

We propose a variation to LEEP that applies to all types of checkpoints including those obtained from unsupervised learning and self-supervised learning. It can also avoids the overly confident softmax.

Feeding the training data of a downstream task into a checkpoint, we obtain their feature representations. The representations are thousands of dimensions, depending on the checkpoint’s neural architecture. We reduce their dimension by using the principal component analysis (PCA). Denote by \( s \) the resultant low-dimensional representation of the input \( x \).

We then fit a Gaussian mixture model (GMM), \( P(s) = \sum_{v \in \mathcal{V}} \pi_v \mathcal{N}(s \mid \mu_v, \Sigma_v) \), to the training set \( \{s_j\}_{j=1}^n \), where \( \mathcal{V} \) is a collection of all the Gaussian components, and \( \pi_v, v \in \mathcal{V}, \) are the mixture weights. It is convenient to compute the posterior distribution:

\[
P(v \mid x) = P(v \mid s) \propto \pi_v \mathcal{N}(s \mid \mu_v, \Sigma_v),
\]

which is arguably more reliable than the class assignment probability \( \theta(x)_z \) output by the softmax classifier because we fit GMM to the downstream task’s training data, whereas the softmax classifier is learned from a different source task.
Figure 5.2: NLEEP' checkpoint ranking performance, evaluated by Kendall’s τ, on Groups I and II in NeuCRaB. We vary the PCA feature dimension and the number of Gaussian components in GMM.

Hence, we arrive at an improved ranking measure, named NLEEP, by replacing $\theta(x)_z$, the probability of classifying an input $x$ to the class $z$, in equations (5.4–5.5) by the posterior distribution $P(v|x)$.

5.4 Experiments on NeuCRaB

There are free parameters in each of the ranking methods. Before presenting the main results, we study how the free parameters in NLEEP affect its checkpoint ranking performance. Figure 5.2 illustrates NLEEP’s Kendall’s τ values over Groups I and II with different PCA feature dimensions and the numbers of Gaussian components. Each Kendall’s τ is averaged across all the downstream tasks; the higher, the better. Along the vertical axes, we change the feature dimensions by keeping different percentages of the PCA energies; PCA50 means the percentage is 50%. Along the horizontal axes, we adopt different numbers of Gaussian components in GMM; $2\times$ means the number is twice the class number of the downstream task. Notably, the Kendall’s τ values remain relatively
stable. In the remaining experiments with NLEEP, we fix the PCA energy to 80% and the number of Gaussian components five times the class number of a downstream task.

5.4.1 Comparison Results

Tables 5.1, 5.2, and 5.3 show the checkpoint ranking methods’ performance on Groups I (checkpoints of mixed supervision), II (different pre-training stages), and III (heterogeneous architectures), respectively. We also union the three groups and present the corresponding ranking performance in Table 2 in Appendix. The numbers in the tables are the average over all downstream tasks. In addition to the evaluation metrics detailed in Section 5.2.3, the GFLOPS column measures the ranking methods’ computing performance; the lower, the better.

We report multiple variations of the ranking methods in the tables. Fine-tuning is computationally expensive, so we stop it after one or five epochs. The linear classifiers are less so as we save the feature representations of downstream tasks’ after one forward pass to the checkpoints. We report the linear classifiers’ ranking results after training them for one epoch, five epochs, and convergence. We test $\alpha = 0.01$ and $\alpha = 0.50$ in the $I_\alpha$ mutual information estimator. Additionally, we experiment with $I_\alpha$ after reducing the feature dimensions by using PCA.

5.4.2 Main Findings

In each column of Tables 5.1, 5.2, 5.3, and Table 2 in Appendix, we highlight the best and second best by the bold font and underscore, respectively. The mutual information fails to rank high-performing checkpoints to the top and even produces negative Pearson and Kendall correlations, probably because of the features’ high dimensions. Reducing the feature dimensions by PCA significantly improves the mutual information’s ranking performance; MI w/ PCA ($\alpha=0.01$) performs
the second best Rel@1, Recall@3 and Rel@3 among the ranking methods in Group III, the checkpoints of heterogeneous neural architectures. Varying $\alpha$ in $I_\alpha$ mutual information estimator [204] can control the trade-off between variance and bias. MI w/ and w/o PCA ($\alpha=0.01$) perform better than MI w/ and w/o PCA ($\alpha=0.50$), respectively. It indicates that neural checkpoint ranking requires low-bias MI estimator since smaller $\alpha$ means low-bias but high-variance estimation.

Fine-tuning up to some epochs turns out the worst ranking methods because it leads to low correlation with the groundtruth ranking and yet incurs heavy computation. Similarly, training the linear classifier up to one or five epochs does not perform well except in Group II. These results indicate that it is difficult to forecast the checkpoints’ final performance from early-stage premature models. Fine-tuning (5 epochs) and Linear (5 epochs) perform better than Fine-tuning (1 epoch) and Linear (1 epoch) in terms of Person and Kendall correlation, respectively. However, they all fail to select the top checkpoint in Group I and Group III since they produce lower Recall@1 and Recall@3 than others. One possible reason is that the evaluation accuracies of checkpoints in the early stage tend to have large variance and they are not stable indicators for ranking neural checkpoints.

If we train the linear classifiers till convergence, they become the best in Group II, and the second best checkpoint ranking method in Groups I and III in terms of Pearson and Kendall correlations. It can also produce better Recall@1 and Recall@3 than Linear (1 epoch) and Linear (5 epoch) in Groups I, II and III since the evaluation accuracies of converged models are more stable than models in the early training stage. Note that the linear classifiers’ accuracies, i.e., the ranking scores, imply the linear separability of the features extracted by the checkpoints. Recall that the mutual information with PCA feature dimension reduction is among the second best (Rel@1, Recall@3 and Rel@3) in Group III. Since both methods measure the feature representations’ quality by the downstream tasks’ labels, we conjecture that the quality of the features is a strong indicator of the checkpoints’ final fine-tuning performance on the downstream tasks. It would be interesting to study other feature quality measures beyond the linear separability and mutual information in
future work.

Table 5.1: Checkpoint ranking results on Group I, the checkpoints of mixed supervision (GFLOPS excludes a forward pass on training data, which takes 3.04E5 GFLOPS shared by all methods)

<table>
<thead>
<tr>
<th>Method</th>
<th>Recall@1</th>
<th>Rel@1</th>
<th>Recall@3</th>
<th>Rel@3</th>
<th>Pearson</th>
<th>Kendall</th>
<th>GFLOPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear (1 epoch)</td>
<td>0.00</td>
<td>96.97</td>
<td>25.00</td>
<td>98.79</td>
<td>23.56</td>
<td>18.44</td>
<td>4.95E4</td>
</tr>
<tr>
<td>Linear (5 epoch)</td>
<td>25.00</td>
<td>98.79</td>
<td>50.00</td>
<td>98.94</td>
<td>49.77</td>
<td>32.33</td>
<td>4.97E4</td>
</tr>
<tr>
<td>Linear (converged)</td>
<td>50.00</td>
<td>99.63</td>
<td><strong>75.00</strong></td>
<td><strong>99.65</strong></td>
<td>68.97</td>
<td>53.43</td>
<td>5.33E4</td>
</tr>
<tr>
<td>Fine-tune (1 epoch)</td>
<td>25.00</td>
<td>97.45</td>
<td>25.00</td>
<td>97.66</td>
<td>30.25</td>
<td>22.15</td>
<td>6.51E5</td>
</tr>
<tr>
<td>Fine-tune (5 epoch)</td>
<td>0.00</td>
<td>91.09</td>
<td>25.00</td>
<td>98.61</td>
<td>48.19</td>
<td>36.78</td>
<td>4.28E6</td>
</tr>
<tr>
<td>MI ($\alpha=0.01$) [204]</td>
<td>0.00</td>
<td>64.67</td>
<td>0.00</td>
<td>87.96</td>
<td>2.39</td>
<td>-0.31</td>
<td>1.62E5</td>
</tr>
<tr>
<td>MI ($\alpha=0.50$)</td>
<td>0.00</td>
<td>66.71</td>
<td>25.00</td>
<td>90.31</td>
<td>-4.91</td>
<td>-13.05</td>
<td>1.62E5</td>
</tr>
<tr>
<td>MI w/ PCA ($\alpha=0.01$)</td>
<td>0.00</td>
<td>89.45</td>
<td>50.00</td>
<td>99.27</td>
<td>16.16</td>
<td>20.67</td>
<td>5.58E4</td>
</tr>
<tr>
<td>MI w/ PCA ($\alpha=0.50$)</td>
<td>0.00</td>
<td>86.49</td>
<td>25.00</td>
<td>94.28</td>
<td>-24.72</td>
<td>-16.06</td>
<td>5.58E4</td>
</tr>
<tr>
<td>LEEP [185]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$\lambda$LEEP</td>
<td><strong>75.00</strong></td>
<td><strong>99.65</strong></td>
<td><strong>75.00</strong></td>
<td><strong>99.65</strong></td>
<td><strong>84.30</strong></td>
<td><strong>76.00</strong></td>
<td><strong>12.85</strong></td>
</tr>
</tbody>
</table>

Table 5.2: Checkpoint ranking results on Group II, the checkpoints at different pre-training stages (GFLOPS excludes a forward pass on training data, which takes 3.04E5 GFLOPS shared by all)

<table>
<thead>
<tr>
<th>Method</th>
<th>Recall@1</th>
<th>Rel@1</th>
<th>Recall@3</th>
<th>Rel@3</th>
<th>Pearson</th>
<th>Kendall</th>
<th>GFLOPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear (1 epoch)</td>
<td>0.00</td>
<td>96.46</td>
<td>25.00</td>
<td>98.79</td>
<td>27.01</td>
<td>24.24</td>
<td>4.95E4</td>
</tr>
<tr>
<td>Linear (5 epochs)</td>
<td>50.00</td>
<td>99.57</td>
<td><strong>100.00</strong></td>
<td><strong>100.00</strong></td>
<td>55.07</td>
<td>51.28</td>
<td>4.97E4</td>
</tr>
<tr>
<td>Linear (converged)</td>
<td><strong>75.00</strong></td>
<td><strong>99.95</strong></td>
<td><strong>100.00</strong></td>
<td><strong>100.00</strong></td>
<td><strong>79.30</strong></td>
<td><strong>68.60</strong></td>
<td><strong>5.33E4</strong></td>
</tr>
<tr>
<td>Fine-tune (1 epoch)</td>
<td>25.00</td>
<td>99.05</td>
<td>25.00</td>
<td>99.47</td>
<td>19.61</td>
<td>15.52</td>
<td>6.51E5</td>
</tr>
<tr>
<td>Fine-tune (5 epochs)</td>
<td>25.00</td>
<td>99.55</td>
<td><strong>100.00</strong></td>
<td><strong>100.00</strong></td>
<td>68.47</td>
<td>58.33</td>
<td>4.28E6</td>
</tr>
<tr>
<td>MI ($\alpha=0.01$) [204]</td>
<td>0.00</td>
<td>94.84</td>
<td>25.00</td>
<td>97.43</td>
<td>-29.41</td>
<td>-17.81</td>
<td>1.62E5</td>
</tr>
<tr>
<td>MI ($\alpha=0.50$)</td>
<td>0.00</td>
<td>96.66</td>
<td>0.00</td>
<td>97.03</td>
<td>-11.36</td>
<td>-10.21</td>
<td>1.62E5</td>
</tr>
<tr>
<td>MI w/ PCA ($\alpha=0.01$)</td>
<td>50.00</td>
<td>99.60</td>
<td>75.00</td>
<td>99.85</td>
<td>52.14</td>
<td>51.34</td>
<td>5.58E4</td>
</tr>
<tr>
<td>MI w/ PCA ($\alpha=0.50$)</td>
<td>0.00</td>
<td>96.68</td>
<td>50.00</td>
<td>99.52</td>
<td>23.73</td>
<td>17.09</td>
<td>5.58E4</td>
</tr>
<tr>
<td>LEEP [185]</td>
<td>75.00</td>
<td>99.44</td>
<td>75.00</td>
<td>99.90</td>
<td>50.36</td>
<td>55.49</td>
<td>378.31</td>
</tr>
<tr>
<td>$\lambda$LEEP</td>
<td><strong>100.00</strong></td>
<td><strong>100.00</strong></td>
<td><strong>100.00</strong></td>
<td><strong>100.00</strong></td>
<td><strong>72.84</strong></td>
<td><strong>67.49</strong></td>
<td><strong>12.95</strong></td>
</tr>
</tbody>
</table>

$\lambda$LEEP performs consistently well in all the groups of checkpoints over all the evaluation metrics with the lowest computation cost. In contrast, the original LEEP measure is not applicable to Group I, the checkpoints of mixed supervision, because it requires that the checkpoints have a
Table 5.3: Checkpoint ranking results on Group III, the checkpoints of heterogeneous architectures (GFLOPS excludes a forward pass on training data, which takes 2.73E5 GFLOPS shared by all)

<table>
<thead>
<tr>
<th>Method</th>
<th>Recall@1</th>
<th>Rel@1</th>
<th>Recall@3</th>
<th>Rel@3</th>
<th>Pearson</th>
<th>Kendall</th>
<th>GFLOPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear (1 epoch)</td>
<td>25.00</td>
<td>98.17</td>
<td>25.00</td>
<td>99.35</td>
<td>30.14</td>
<td>13.80</td>
<td>3.37E4</td>
</tr>
<tr>
<td>Linear (5 epoch)</td>
<td>25.00</td>
<td>98.98</td>
<td>25.00</td>
<td>99.63</td>
<td>33.45</td>
<td>18.95</td>
<td>3.38E4</td>
</tr>
<tr>
<td>Linear (converged)</td>
<td>25.00</td>
<td>99.66</td>
<td>25.00</td>
<td>99.72</td>
<td>63.55</td>
<td>36.91</td>
<td>3.62E4</td>
</tr>
<tr>
<td>Fine-tune (1 epoch)</td>
<td>0.00</td>
<td>98.28</td>
<td>25.00</td>
<td>99.80</td>
<td>17.61</td>
<td>11.59</td>
<td>4.43E5</td>
</tr>
<tr>
<td>Fine-tune (5 epoch)</td>
<td>25.00</td>
<td>98.62</td>
<td>25.00</td>
<td>99.68</td>
<td>25.72</td>
<td>15.72</td>
<td>2.91E6</td>
</tr>
<tr>
<td>MI ($\alpha=0.01$)</td>
<td>25.00</td>
<td>98.29</td>
<td>25.00</td>
<td>99.34</td>
<td>4.42</td>
<td>2.94</td>
<td>1.30E5</td>
</tr>
<tr>
<td>MI ($\alpha=0.50$)</td>
<td>25.00</td>
<td>98.36</td>
<td>25.00</td>
<td>99.37</td>
<td>-9.79</td>
<td>-6.81</td>
<td>1.30E5</td>
</tr>
<tr>
<td>MI w/ PCA ($\alpha=0.01$)</td>
<td>0.00</td>
<td>99.18</td>
<td>50.00</td>
<td>99.82</td>
<td>61.94</td>
<td>38.83</td>
<td>5.56E4</td>
</tr>
<tr>
<td>MI w/ PCA ($\alpha=0.50$)</td>
<td>0.00</td>
<td>96.34</td>
<td>0.00</td>
<td>98.47</td>
<td>33.17</td>
<td>21.26</td>
<td>5.56E4</td>
</tr>
<tr>
<td>LEEP [185]</td>
<td>25.00</td>
<td>97.36</td>
<td><strong>75.00</strong></td>
<td><strong>99.90</strong></td>
<td>42.99</td>
<td>45.06</td>
<td>247.56</td>
</tr>
<tr>
<td>VLEEP</td>
<td>25.00</td>
<td><strong>99.66</strong></td>
<td>25.00</td>
<td>99.70</td>
<td><strong>66.94</strong></td>
<td><strong>51.14</strong></td>
<td><strong>12.68</strong></td>
</tr>
</tbody>
</table>

classification output layer. Overall, LEEP is the second best over all evaluation metrics among the ranking methods in Groups II and III, whose checkpoints all have a classification output layer. Specifically, LEEP can produce second best Recall@1, Recall@3 and Rel@3 in Group II, and best Recall@3 and Rel@3, second best Kendall correlation in Group III. It is a more consistent indicator than fine-tuning, linear classifier and MI based ranking methods. However, LEEP can not produce better results than VLEEP, and it requires slightly larger GFLOPS due to the extra computation cost from the classification head.

We conjecture that VLEEP outperforms LEEP mainly because GMMs calibrate the posterior probabilities better than the checkpoints’ softmax classifiers. The checkpoint ranking quality of LEEP score hinges on the performance of the ‘dummy classifier’ — $P(y|x, \theta, t)$, and $\theta(x)$ is the key element to calculate it. However, $\theta(x)$ can be poorly calibrated [185] and it can not represent a true probability. In contrast, $P(v|x)$ used in VLEEP is indeed the probability that the sample belongs to one cluster from a mixture of Gaussian distributions and it can remedy the poor-calibrated problem in LEEP.
In particular, we highlight the GFLOPS column in the tables. NLEEP and LEEP exhibit a clear advantage over the other checkpoint ranking methods in terms of computing. The main reason is that NLEEP and LEEP can avoid intensive computation from neural network training, and they only require one forward pass through the training data.

Checkpoint ranking on different groups of checkpoints varies in degrees of difficulty. The most challenging group is Group III, the checkpoints of heterogeneous neural architectures. All the ranking methods produce lower correlations with the groundtruth ranking, and they can barely select the top checkpoints in this group. The main reason is that neural architecture matters for transfer learning [120]. Besides, heterogeneous neural architectures can demonstrate various performance even if we train them from scratch on downstream tasks. Ranking neural checkpoints by the feature representations of last layer is not sufficient for those checkpoints. We may explore more advanced ranking methods considering the structures of the deep neural networks in the future.

Checkpoint ranking on Group II is easier than on Group I since all the ranking methods can achieve relatively better results over all evaluation metrics in Group II. The results indicate that checkpoints with various training strategies (Group I) can bring more complex knowledge from source domains, comparing with checkpoints with different early stopping stages (Group II). In addition, fine-tuning the entire models and training linear classifiers up to one or five epochs perform significantly better on Group II since those ranking methods are based on early stopping as well.

To simulate a sufficiently large pool of checkpoints in the real applications, we finally combine the checkpoints in Group I, II, and III into one large group and conduct checkpoint ranking experiments on it. We also add one more group of checkpoints with ResNet-101s [84] to evaluate checkpoint ranking on deeper models. Please see more details in Appendix A.3 and A.4. Although the benchmark can be easily extended to many downstream tasks in other modalities, e.g., voice,
text, and cross-modal modalities, we steer our attention into comparing several intuitive ranking measures on variants of checkpoints, covering different training strategies, source domains, and architectures at a range of early stopping stages. We initial the checkpoint ranking idea, demonstrate the existence of an effective yet lightweight measure, NLEEP, and hope it can shed light on more efficient ranking methods and practical applications.

5.5 Summary

Deep learning has triumphed over many fields in both research and real-world applications. There must exist hundreds of thousands of DNNs trained and released by various groups. To this end, it is natural to select an existing, promising DNN checkpoint as a warm start to a training procedure when solving a new task. How to identify useful checkpoints from a large pool for the target task? Towards answering this question, we present NeuCRaB, a thorough benchmark covering diverse downstream tasks and pre-trained DNN checkpoints, along with NLEEP, a lightweight, effective checkpoint ranking measure.

The experiments with linear classifiers and mutual information (after PCA) reveal that the features extracted from the checkpoints are good indicators of the checkpoints’ potential in transfer learning. It is worth exploring other ways of evaluating the features’ quality in future work. It is also interesting to investigate the checkpoints’ inherent signatures, such as topology and stability to noise, which might be informative of their transferabilities. Finally, some learning-based methods in predicting networks’ generalization gaps are also promising for the checkpoint ranking problem.
CHAPTER 6: IMAGE SYNTHESIS FROM SALIENT OBJECT LAYOUT

6.1 Problem Introduction

Pablo Picasso once said “Every child is an artist. The problem is how to remain an artist once grown up.” Now with the help of smart image editing assistant, our creative and imaginative nature can well flourish. Recent years have witnessed a wide variety of image generation works conditioned on diverse inputs, such as text [313, 289], scene graph [106], semantic segmentation map [99, 272], and holistic layout [318]. Among them, text-to-image generation provides a flexible interface for users to describe visual concepts via natural language descriptions [313, 289]. The limitation is that one single sentence may not be adequate for describing the details of every object in the intended image.

Scene graph [106], with rich structural representation, can potentially reveal more visual relations of objects in an image. However, pairwise object relation labels are difficult to obtain in real-life applications. The lack of object size, location and background information also limits the quality of synthesized images.

Another line of research is image synthesis conditioned on semantic segmentation map. While previous work [99, 272, 198] has shown promising results, collecting annotations for semantic segmentation maps is time consuming and labor intensive. To save annotation effort, Zhao et al. [318] proposed to take as the input a holistic layout including both foreground objects (e.g., “cat”, “per-
Figure 6.1: Top row: images synthesized from semantic segmentation maps. Bottom row: high-resolution images synthesized from salient object layouts, which allows users to create an image by drawing only a few bounding boxes.

son”) and background (e.g., “sky”, “grass”). In this work, we push this direction to a further step and explore image synthesis given salient object layout only, with just coarse foreground object bounding boxes and categories. Figure 6.1 provides a comparison between segmentation-map-based image synthesis (top row) and our setting (bottom row). Our task takes foreground objects as the only input, without any background layout or pixel-wise segmentation map.

The proposed new task presents new challenges for image synthesis: (i) how to generate fine-grained details and realistic textures with only a few foreground object bounding boxes and categories; and (ii) how to invent a realistic background and weave it into the standalone foreground objects seamlessly. Note that no knowledge about the background is provided; while in [318], a holistic layout is provided and only low-resolution (64 × 64) images are generated. In our task, the goal is to synthesize high-resolution (512 × 256) images given very limited information (salient layout only).

To tackle these challenges, we propose Background Hallucination Generative Adversarial Network (BachGAN). Given a salient object layout, BachGAN generates an image via two steps: (i) a

\(^1\) Salient and foreground are used interchangeably in this work.
background retrieval module selects from a large candidate pool a set of segmentation maps most relevant to the given object layout; \((ii)\) these candidate layouts are then encoded via a background fusion module to hallucinate a best-matching background. With this retrieval-and-hallucination approach, BachGAN can dynamically provide detailed and realistic background that aligns well with any given foreground layout. In addition, by feeding both foreground objects and background representation into a conditional GAN (via a SPADE normalization layer \([198]\)), BachGAN can generate high-resolution images with visually consistent foreground and background.

Our contributions are summarized as follows:

- We propose a new task - image synthesis from salient object layout, which allows users to draw an image by providing just a few object bounding boxes.

- We present BachGAN, the key components of which are a retrieval module and a fusion module, which can hallucinate a visually consistent background on-the-fly for any foreground object layout.

- Experiments on Cityscapes \([30]\) and ADE20K \([322]\) datasets demonstrate our model’s ability to generate high-quality images, outperforming baselines measured on both visual quality and consistency metrics.

6.2 BachGAN

We first define the problem formulation and introduce preliminaries in Sec. 6.2.1, before presenting the proposed Background Hallucination Generative Adversarial Network (BachGAN). As illustrated in Figure 6.2, BachGAN consists of three components: \((i)\) Background Retrieval Module (Sec. 6.2.2), which selects a set of segmentation maps from a large candidate pool given a fore-
Figure 6.2: Overview of the proposed BachGAN for image synthesis from salient object layout.

(ii) Background Fusion Module (Sec. 6.2.3), which fuses the salient object layout and the selected candidate into a feature map for background hallucination; and (iii) Image Generator, which adopts a SPADE layer [198] to generate an image based on the fused representation. Discriminators are omitted in Figure 6.2 for simplicity.

6.2.1 Problem Formulation and Preliminaries

Problem Definition Assume we have a set of images $\mathcal{I}$ and their corresponding salient object layouts $\mathcal{L}$. The goal is to train a model that learns a mapping from layouts to images, i.e., $\mathcal{L} \rightarrow \mathcal{I}$. Specifically, given a ground-truth image $\mathbf{I} \in \mathcal{I}$ and its corresponding layout $\mathbf{L} \in \mathcal{L}$, where $\mathbf{L}_i = (x_i, y_i, h_i, w_i)$ denotes the top-left coordinates plus the height and width of the $i$-th bounding
box. Following [198, 272], we first convert $L$ into a label map $M \in \{0, 1\}^{H \times W \times C_o}$, where $C_o$ denotes the number of categories, and $H, W$ are the height and width of the label map, respectively. Different from the semantic segmentation map used in [272], some pixels $M(i, j)$ can be assigned to $n$ object instances, i.e., $M(i, j) \in \{0, 1\}^{C_o}$ s.t. $\sum_p M(i, j, p) = n$.

**A Naive Solution** To draw in the motivation of our framework, we first consider a simple conditional GAN model and discuss its limitations. By considering the label map $M$ as the input image, image-to-image translation models can be directly applied with the following objective:

$$
\min_G \max_D \mathbb{E}_{M,I}[\log(D(M, I))] + \mathbb{E}_M[\log(1 - D(M, G(M)))],
$$

(6.1)

where $G$ and $D$ denote the generator and the discriminator, respectively. The generator $G(\cdot)$ takes a label map $M$ as input to generate a fake image.

State-of-the-art conditional GANs, such as pix2pix-HD [272], can be directly applied here. However, some caveats can be readily noticed, as only a coarse foreground layout is provided in our setting, making the generation task much more challenging than when a semantic segmentation map is provided. Thus, we introduce Background Hallucination to address this issue in the following sub-section.

### 6.2.2 Background Retrieval Module

The main challenge in this new task is how to generate a proper background to fit the foreground objects. Given an object layout $L$ that contains $k$ instances: $L_0^{C_0}, \ldots, L_k^{C_k}$, where $C_i$ is the category of instance $L_i$, assume we have a memory bank $B$ containing pairs of image $I$ and its fine-grained semantic segmentation map $S$ with $l$ instances: $S_0^{C_0}, \ldots, S_l^{C_l}$. We first retrieve a pair of $I$ and $S$ that
contain the most similar layout to \( L \), by using a layout-similarity score, a variant of the Intersect over Union (IoU) metric, to measure the distance between a salient object layout and a fine-grained semantic segmentation map:

\[
\text{IoU}_r = \frac{\sum_{j=1}^{C} S_j^i \cap L_j^i}{\sum_{j=1}^{C} S_j^i \cup L_j^i},
\]

(6.2)

where \( C \) is the total number of object categories, \( S_j^i = \bigcup_i S_j^i \) and \( L_j^i = \bigcup_i L_j^i \). \( \bigcup \) and \( \bigcap \) denote union and intersect, respectively. The proposed metric can preserve the overall location and category information of each object, since the standard IoU score is designed for measuring the quality of object detection. However, instead of calculating the mean IoU scores across all the classes, we use Eqn. (6.2) to prevent the weights of small objects from growing too high.

Given a salient object layout \( L_q \) as the query, we rank the pairs of image and semantic segmentation map in the memory bank by the aforementioned layout-similarity score. As a result, we can obtain a retrieved image \( I_r \) with semantic segmentation map \( S_r \), which has a salient object layout most similar to the query \( L_q \). The assumption is that images with a similar foreground composition may share similar background as well. Therefore, we treat the retrieved semantic segmentation map, \( S_r \), as the potential background for \( L_q \). Formally, we first convert the background of \( S_r \) into a label map \( M_b \): \( M_b(i, j) \in \{0, 1\}^{C_b} \) s.t. \( \sum_p M_b(i, j, p) = 1 \), where \( C_b \) denotes the number of categories in the background. Then, we produce a new label map, encoding both the foreground object layout and the fine-grained background segmentation map, by concatenating \( M_b \) and the foreground label map \( M_q \) of \( L_q \):

\[
\hat{M} = [M_b; M_q],
\]

(6.3)

where \([;]\) denotes concatenation, and the resulting label map is represented as \( \hat{M} \in \{0, 1\}^{H \times W \times (C_o + C_b)} \). Note that the memory bank \( B \) can be much smaller than the scale of training data. Therefore, this
A Simple Baseline

Based on the obtained new label map $\hat{M}$, now we describe a simple baseline that motivates our proposed model. We consider the new label map as an input image, and adopt an image-to-image translation model. The following conditional GAN loss can be used for training:

$$\min_G \max_D \mathbb{E}_{\hat{M},I_r}[\log(D(\hat{M}, I_r))] + \mathbb{E}_{\hat{M},I_q}[\log(D(\hat{M}, I_q))]$$
$$+ \mathbb{E}_{\hat{M}}[\log(1 - D(\hat{M}, G(\hat{M})))],$$

(6.4)

where $I_q$ is the ground-truth image corresponding to the query $L_q$, and $I_r$ is the retrieved image. $^2$

We name this baseline method “BachGAN-r”. Though the ground-truth background cannot be obtained, the use of the retrieved background injects useful information that helps the generator synthesize better images, compared to the objective in Eqn. (6.1).

$^2$Empirically, we observe that adding the retrieved image to the GAN loss improves the performance.
6.2.3 Background Fusion Module

Though the retrieval-based baseline approach can hallucinate a background given a foreground object layout, the relevance of one retrieved semantic segmentation map to the input foreground layout is not guaranteed. One possible solution is to use multiple retrieved segmentation maps in Eqn. (6.4). However, this renders training unstable as the discriminator becomes unbalanced when several retrieved images are included in the loss function. More importantly, the dimension of the input label map is too high. In order to utilize multiple retrieved segmentation maps for a fuzzy hallucination of the background, we further introduce a Background Fusion Module to encode Top-$m$ retrieved segmentation maps to hallucinate a smoother background.

Assume we obtain $m$ retrieved segmentation maps $S_{r,0}, ..., S_{r,m}$ with their corresponding background label maps $M_{b,0}, ..., M_{b,m}$. The query salient object layout $L_q$ has a corresponding label map $M_q$, where $M_{b,i} \in \{0, 1\}^{H \times W \times C_b}$ and $M_q \in \{0, 1\}^{H \times W \times C_o}$. As illustrated in Figure 6.2, we first obtain $\hat{M}_{r,0}, ..., \hat{M}_{r,m}$ with Eqn. (6.3), where $\hat{M}_{r,i} \in \{0, 1\}^{H \times W \times (C_o + C_b)}$. $M_q$ is padded with 0 to obtain a query label map $\hat{M}_q$ with the same shape as $\hat{M}_{r,i}$. $\hat{M}_{r,0}, ..., \hat{M}_{r,m}$ are then concatenated into $\hat{M}_r \in \{0, 1\}^{m \times H \times W \times (C_o + C_b)}$. A convolutional network $F$ is then used to encode the label maps into feature maps:

$$ m_0 = F(\hat{M}_q) \oplus \text{Pool}(F(\hat{M}_r)), $$

where Pool represents average pooling, $\oplus$ denotes element-wise addition, and $m_0 \in \mathbb{R}^{H \times W \times h}$ ($h$ is the number of feature maps). We then use another convolutional network $M$ to obtain updated feature maps:

$$ m_t = m_{t-1} \oplus M(m_{t-1}). $$

83
After $T$ steps, we obtain the final feature map $\hat{m} = m_T$, which contains information from both salient object layout and hallucinated background.

**Training Objective** Based on the feature map $\hat{m}$, BachGAN uses the following conditional GAN loss for training:

$$
\min_G \max_D \mathbb{E}_{m, I_q}[\log(D(\hat{m}, I_q))] + \mathbb{E}_m[\log(1 - D(\hat{m}, G(\hat{m})))], \tag{6.7}
$$

where $I_q$ is the ground-truth image corresponding to the query $L_q$. Compared with Eqn. (6.4), multiple retrieved segmentation maps are used to hallucinate the background, which leads to better performance in practice.

**Image Generator** Now, we describe how the generator $G(\cdot)$ takes $\hat{m}$ as input to generate a high-quality image. In order to generate photo-realistic images, we utilize the spatially-adaptive normalization (SPADE) layer [198] in our generator. Let $h_i$ denote the activation feature map of the $i$-th layer of the generator $G$. Similar to batch normalization [97], SPADE [198] first normalizes $h_i$, then produces the modulation parameters $\gamma$ and $\beta$ to denormalize it, both of which are functions of $\hat{m}$:

$$
\hat{h}_i = \text{norm}(h_i) \otimes \gamma(\hat{m}) \oplus \beta(\hat{m}), \tag{6.8}
$$

where $\hat{h}_i$ denotes the output of a SPADE layer, norm $(\cdot)$ is a normalization operation, and $\otimes$ and $\oplus$ are element-wise production and addition, respectively. An illustration of the generator is provided in the bottom part of Figure 6.2. More details about SPADE can be found in [198].
6.3 Experiments

In this section, we describe experiments comparing BachGAN with state-of-the-art approaches on the new task, as well as detailed analysis that validates the effectiveness of our proposed model.

6.3.1 Experimental Setup

Datasets  We conduct experiments on two public datasets: Cityscapes [30] and ADE20K [322]. Cityscapes contains images with street scene in cities. The size of training and the validation set is 3,000 and 500, respectively. We exclude 23 background classes and use the remaining 10 foreground objects in the salient object layout. With provided instance-level annotations, we can readily transform a semantic segmentation instance to its bounding box, by taking the max and min of the coordinates of each pixel in an instance. ADE20K consists of 20,210 training and 2,000 validation images. The dataset contains challenging scenes with 150 semantic classes. We
exclude the 35 background classes and utilize the remaining 115 foreground objects. There are no instance-level annotations for ADE20k, thus, we use a simple approach to find contours [240] from a semantic segmentation map and then obtain the bounding box for each contour. A separate memory bank is used for each dataset. We train all the image synthesis methods on the same training set and report their results on the same validation set.

**Baselines**  We include several strong baselines that can generate images with object layout as input:

- **SPADE**: We adopt SPADE [198] as our first baseline, taking as input the salient object layout instead of semantic segmentation map used in the original paper.

- **SPADE with Segmentation (SPADE-SEG)**: We obtain the second baseline by exploiting the pairs of segmentation mask and image from the memory bank. Besides GAN loss, the model is trained with an additional loss. It minimizes the segmentation loss between the real image and the output from the generator based on the memory bank.

- **Layout2im**: We use the code from Layout2im [318], which generates images from holistic layouts and supports the generation of $64 \times 64$ images only.

**Performance metrics**  Following [24, 272], we run a semantic segmentation model on the synthesized images and measure the segmentation accuracy. We use state-of-the-art segmentation networks: DRN-D-105 [301] for Cityscapes, and UperNet101 [282] for ADE20K. Pixel accuracy (Acc) is compared across different models. This is done using real objects cropped and resized from ground-truth images in the training set of each dataset. In addition to classification accuracy, we use the Frechet Inception Distance (FID) [87] to measure the distance between the distribution of synthesized results and the distribution of real images.
Table 6.1: Results on Cityscapes and ADE20K w.r.t. FID and the pixel accuracy (Acc). Results with (†) are reported in [198], serving as the upper bound of our model performance.

<table>
<thead>
<tr>
<th>Model</th>
<th>Cityscapes Acc</th>
<th>Cityscapes FID</th>
<th>ADE20K Acc</th>
<th>ADE20K FID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layout2im [318]</td>
<td>-</td>
<td>99.1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SPADE</td>
<td>57.6</td>
<td>86.7</td>
<td>55.3</td>
<td>59.4</td>
</tr>
<tr>
<td>SPADE-SEG</td>
<td>60.2</td>
<td>81.2</td>
<td>60.9</td>
<td>57.2</td>
</tr>
<tr>
<td>BachGAN-r</td>
<td>67.3</td>
<td>74.4</td>
<td>64.5</td>
<td>53.2</td>
</tr>
<tr>
<td>BachGAN</td>
<td><strong>70.4</strong></td>
<td><strong>73.3</strong></td>
<td><strong>66.8</strong></td>
<td><strong>49.8</strong></td>
</tr>
<tr>
<td>SPADE-v [198]</td>
<td><strong>81.9†</strong></td>
<td>71.8†</td>
<td><strong>79.9†</strong></td>
<td><strong>33.9†</strong></td>
</tr>
</tbody>
</table>

Figure 6.5: Generated images by adding bounding boxes sequentially to previous layout (Cityscapes).

**Implementation Details**  All the experiments are conducted on an NVIDIA DGX1 with 8 V100 GPUs. We use Adam [118] as the optimizer, and learning rates for the generator and discriminator are both set to 0.0002. For Cityscapes, we train 60 epoches to obtain a good generator, and ADE20k needs 150 epoches to converge. $m$ is set to 3 for both datasets.
Figure 6.6: Generated images by adding bounding boxes sequentially to previous layout (ADE20k).

Figure 6.7: Top row: synthesized images based on salient object layouts from the test set. Bottom row: synthesized images based on salient layouts with flipping objects, modified from the top-row layouts.

### 6.3.2 Quantitative Evaluation

Table 6.1 summarizes the results of all the models w.r.t. the FID score and classification accuracy. We also report the scores on images generated from vanilla SPADE (SPADE-v) using segmentation map as input (upper bound). Measured by FID, BachGAN outperforms all the baselines in both datasets with a relatively large margin. For Cityscapes, BackGAN achieves a FID score of 73.3, which is close to the upper bound. In ADE20K, the improved gain over baselines is not significant.

88
This is because most images in ADE20K are dominated by salient foregrounds, with relatively less space for background, which limits the effect of our hallucination module. The pixel accuracy of our method is also higher than other baselines. BachGAN-r also achieves reasonable performance on both datasets.

### 6.3.3 Qualitative Analysis

In Figure 6.3 and 6.4, we provide qualitative comparison of all the methods. Our model produces images with much higher visual quality compared to the baselines. Particularly, in Cityscapes, our method can generate images with detailed/sharp backgrounds while the other approaches fail to. In ADE20K, though the background region is relatively smaller than Cityscapes, BachGAN still produces synthesized images with better visual quality.

Figure 6.5 and 6.6 demonstrate that BachGAN is able to manipulate a series of complex images progressively, by starting from a simple layout and adding new bounding boxes sequentially. The generated samples are visually appealing, with new objects depicted at the desired locations in the images, and existing objects remain consistent to the layout in previous rounds. These examples demonstrate our model’s ability to perform controllable image synthesis based on layout.
Table 6.2: User preference study. Win/lose/tie indicates the percentage of images generated by BachGAN are better/worse/equal to the compared model.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>BachGAN vs. SPADE</th>
<th>BachGAN vs. SPADE-Seg</th>
<th>BachGAN vs. BachGAN-r</th>
<th>BachGAN vs. Layout2im</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>win</td>
<td>loss</td>
<td>tie</td>
<td>win</td>
</tr>
<tr>
<td>Cityscapes</td>
<td>85.5</td>
<td>3.4</td>
<td>11.1</td>
<td>71.7</td>
</tr>
<tr>
<td>ADE20K</td>
<td>75.9</td>
<td>12.8</td>
<td>11.3</td>
<td>66.8</td>
</tr>
</tbody>
</table>

Figure 6.7 further illustrates that BachGAN also works well when objects are positioned in an unconventional way. In the bottom row, we flip some objects in the object layouts (e.g., windowpane in the top-left image), and generate images with the manipulated layouts. BachGAN is still able to generate high-quality images with a reasonable background, proving the robustness of BachGAN.

In Figure 6.8, we sample some retrieved images from Cityscapes. The top-3 results are consistent and similar with the original background. More synthesized and retrieval results (for ADE20K) are provided in Appendix.

### 6.3.4 Human Evaluation

We use Amazon Mechanical Turk (AMT) to evaluate the generation quality of all the approaches. AMT turkers are provided with one input layout and two synthesized outputs from different methods, and are asked to choose the image that looks more realistic and more consistent with the input layout. The user interface of the evaluation tool also provides a neutral option, which can be selected if the turker thinks both outputs are equally good. We randomly sampled 300 image pairs, each pair judged by a different group of three people. Only workers with a task approval rate greater than 98% can participate in the study.
Table 6.3: FID scores of BachGAN with different numbers of retrieved segmentation maps (Cityscapes).

<table>
<thead>
<tr>
<th>Method</th>
<th>BachGAN-3</th>
<th>BachGAN-4</th>
<th>BachGAN-5</th>
</tr>
</thead>
<tbody>
<tr>
<td>FID</td>
<td>73.31</td>
<td>73.03</td>
<td>72.95</td>
</tr>
</tbody>
</table>

Table 6.4: FID scores of BachGAN and BachGAN-r trained using memory bank of different sizes (Cityscapes).

<table>
<thead>
<tr>
<th>Bank size</th>
<th>BachGAN</th>
<th>BachGAN-r</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>73.31</td>
<td>74.44</td>
</tr>
<tr>
<td>2 ×</td>
<td>72.50</td>
<td>73.95</td>
</tr>
</tbody>
</table>

Table 6.2 reports the pairwise comparison between our method and the other four baselines. Based on human judgment, the quality of images generated by BachGAN is significantly higher than SPADE. Comparing with two strong baselines (SPADE-SEG and BachGAN-r), BachGAN achieves the best performance. As expected, Layout2im receives the lowest acceptance by human judges, due to its low resolution.

6.3.5 Ablation Study

Effect of segmentation map retrieval First, we train three BachGANs with different numbers of retrieved segmentation maps, setting \( m \) to 3, 4 and 5, and evaluate them on Cityscapes. The FID scores of different models are summarized in Table 6.3. The model using Top-5 retrieved segmentation maps (BachGAN-5) achieves the best performance, compared to models with Top-3 and Top-4. This analysis demonstrates that increasing the number of selected segmentation maps can slightly improve the scores. Due to the small performance gain, we keep \( m = 3 \) in our experiments.
Effect of memory bank  We also compare models trained with memory banks of different sizes. Specifically, we compare the performance of BachGAN and BachGAN-r with memory bank size $|B|$ (used in our experiments) and $2 \times |B|$. Results are summarized in Table 6.4. With a larger memory bank, both models are able to improve the evaluation scores. Interestingly, the gain of BachGAN is larger than that of BachGAN-r, showing that BachGAN enjoys more benefit from the memory bank. More analysis about the size of memory bank is provided in Appendix.

6.4 Summary

In this chapter, we introduce a novel framework, BachGAN, to generate high-quality images conditioned on salient object layout. By hallucinating the background based on given object layout, the proposed model can generate high-resolution images with photo-realistic foreground and integral background. Comprehensive experiments on both Cityscapes and ADE20K datasets demonstrate the effectiveness of our proposed model, which can also perform controllable image synthesis by progressively adding salient objects in the layout. For future work, we will investigate the generation of more complicated objects such as people, cars and animals [192]. Disentangling the learned representations for foreground and background is another direction.
CHAPTER 7: LEARNING THE DISTRIBUTIONS OF ADVERSARIAL EXAMPLES

7.1 Problem Introduction

Szegedy et al. [244] found that DNNs are vulnerable to adversarial examples whose changes from the benign ones are imperceptible and yet can mislead DNNs to make wrong predictions. A rich line of work furthering their finding reveals more worrisome results. Notably, adversarial examples are transferable, meaning that one can design adversarial examples for one DNN and then use them to fail others [194, 244, 257]. Moreover, adversarial perturbation could be universal in the sense that a single perturbation pattern may convert many images to adversarial ones [177].

The adversarial examples raise a serious security issue as DNNs become increasingly popular [230, 122, 88, 147, 60, 164, 160, 159, 267, 94, 112, 165, 111]. Unfortunately, the cause of the adversarial examples remains unclear. Goodfellow et al. [71] conjectured that DNNs behave linearly in the high dimensional input space, amplifying small perturbations when their signs follow the DNNs’ intrinsic linear weights. Fawzi et al. [55] experimentally studied the topology and geometry of adversarial examples and Xu et al. [288] provide the image-level interpretability of adversarial examples. Ma et al. [168] characterized the subspace of adversarial examples. Nonetheless, defense methods [197, 256, 216, 170] motivated by them were broken in a short amount of time [85, 10, 290, 227], indicating that better defense techniques are yet to be developed, and there

may be unknown alternative factors that play a role in the DNNs’ sensitivity.

Powerful adversarial attack methods are key to better understanding of the adversarial examples and for thorough testing of defense techniques.

In this work, we propose a black-box adversarial attack algorithm that can generate adversarial examples to defeat both vanilla DNNs and those recently defended by various techniques. Given an arbitrary input to a DNN, our algorithm finds a probability density over a small region centered around the input such that a sample drawn from this density distribution is likely an adversarial example, without the need of accessing the DNN’s internal layers or weights — thus, our method falls into the realm of black-box adversarial attack [195, 17, 23, 95].

Our approach is strong; tested against two vanilla DNNs and 13 defended ones, it outperforms state-of-the-art black-box or white-box attack methods for most cases, and it is on par with them for the remaining cases. It is also universal as it attacks various DNNs by a single algorithm. We hope it can effectively benchmark new defense methods in the future — code is available at https://github.com/Cold-Winter/Nattack. Additionally, our study reveals that adversarial training remains one of the best defenses [170], and the adversarial examples are not as transferable across defended DNNs as them across vanilla ones. The latter somehow weakens the practical significance of white-box methods which otherwise could fail a black-box DNN by attacking a substitute.

Our optimization criterion is motivated by the natural evolution strategy (NES) [276]. NES has been previously employed by [95] to estimate the gradients in the projected gradient search for adversarial examples. However, their algorithm leads to inferior performance to what we proposed (cf. Table 7.1). This is probably because, in their approach, the gradients have to be estimated relatively accurately for the projected gradient method to be effective. However, some of the neural networks $F(x)$ are not smooth, so that the NES estimation of the gradient $\nabla F(x)$ is not
reliable enough.

In this work, we opt for a different methodology using a constrained NES formulation as the objective function instead of using NES to estimate gradients as in [95]. The main idea is to smooth the loss function by a probability density distribution defined over the $\ell_p$-ball centered around a benign input to the neural network. All adversarial examples of this input belong to this ball\(^1\). In this frame, assuming that we can find a distribution such that the loss is small, then a sample drawn from the distribution is likely adversarial. Notably, this formulation does not depend on estimating the gradient $\nabla F(x)$ any more, so it is not impeded by the non-smoothness of DNNs.

We adopt parametric distributions in this work. The initialization to the distribution parameters plays a key role in the run time of our algorithm. In order to swiftly find a good initial distribution to start from, we train a regression neural network such that it takes as input the input to the target DNN to be attacked and its output parameterizes a probability density as the initialization to our main algorithm.

Our approach is advantageous over existing ones in multiple folds. First, we can designate the distribution in a low-dimensional parameter space while the adversarial examples are often high-dimensional. Second, instead of questing an “optimal” adversarial example, we can virtually draw an infinite number of adversarial examples from the distribution. Finally, the distribution may speed up the adversarial training for improving DNNs’ robustness because it is more efficient to sample many adversarial examples from a distribution than to find them using gradient based optimization.

\(^1\)It is straightforward to extend our method to other constraints bounding the offsets between inputs and adversarial examples.
7.2 Approach

Consider a DNN classifier \( C(x) = \arg \max_i F(x)_i \), where \( x \in [0,1]^{\text{dim}(x)} \) is an input to the neural network \( F(\cdot) \). We assume softmax is employed for the output layer of the network and let \( F(\cdot)_i \) denote the \( i \)-th dimension of the softmax output. When this DNN correctly classifies the input, i.e., \( C(x) = y \), where \( y \) is the groundtruth label of the input \( x \), our objective is to find an adversarial example \( x_{\text{adv}} \) for \( x \) such that they are imperceptibly close and yet the DNN classifier labels them distinctly; in other words, \( C(x_{\text{adv}}) \neq y \). We exclude the inputs for which the DNN classifier predicts wrong labels in this work, following the convention of previous work [20].

We bound the \( \ell_p \) distance between an input \( x \) and its adversarial counterparts: \( x_{\text{adv}} \in S_p(x) := \{x' : \|x - x'\|_p \leq \tau_p\}, p = 2 \) or \( \infty \). We omit from \( S_p(x) \) the argument \( (x) \) and the subscript \( p \) when it does not cause ambiguity. Let \( \text{proj}_S(x') \) denote the projection of \( x' \in \mathbb{R}^{\text{dim}(x)} \) onto \( S \).

We first review the NES based black-box adversarial attack method [95]. We show that its performance is impeded by unstable estimation of the gradients of certain DNNs, followed by our approach which does not depend at all on the gradients of the DNNs.

7.2.1 A Black-box Adversarial Attack by NES

Ilyas et al. [95] proposed to search for an optimal adversarial example in the following sense,

\[
x_{\text{adv}} \leftarrow \arg \min_{x' \in S} f(x'),
\]

(7.1)

given a benign input \( x \) and its label \( y \) correctly predicted by the neural network \( F(\cdot) \), where \( S \) is a small region containing \( x \) defined above, and \( f(x') \) is a loss function defined as \( f(x') := -F(x')_y \).
In [95], this loss is minimized by the projected gradient method,

\[ x_{t+1} \leftarrow \text{proj}_S(x_t - \eta \text{sign}(\nabla f(x_t))) \tag{7.2} \]

where \( \text{sign}(\cdot) \) is a sign function. The main challenge here is how to estimate the gradient \( \nabla f(x_t) \) with derivative-free methods, as the network’s internal architecture and weights are unknown in the black-box adversarial attack. One technique for doing so is by NES [276]:

\[ \nabla f(x_t) \approx \nabla_x \mathbb{E}_{\mathcal{N}(z|x_t,\sigma^2)} f(z) \tag{7.3} \]
\[ = \mathbb{E}_{\mathcal{N}(z|x_t,\sigma^2)} f(z) \nabla_x \log \mathcal{N}(z|x_t,\sigma^2), \tag{7.4} \]

where \( \mathcal{N}(z|x_t,\sigma^2) \) is an isometric normal distribution with mean \( x_t \) and variance \( \sigma^2 \). Therefore, the stochastic gradient descent (SGD) version of eq. (7.2) becomes:

\[ x_{t+1} \leftarrow \text{proj}_S(x_t - \eta \text{sign}(\frac{1}{b} \sum_{i=1}^{b} f(z_i) \nabla \log \mathcal{N}(z_i|x_t,\sigma^2))) \tag{7.2} \]

where \( b \) is the size of a mini-batch and \( z_i \) is sampled from the normal distribution. The performance of this approach hinges on the quality of the estimated gradient. Our experiments show that its performance varies on attacking different DNNs probably because non-smooth DNNs lead to unstable NES estimation of the gradients (cf. eq. (7.3)).

### 7.2.2 \( \mathcal{N} \)Attack

We propose a different formulation albeit still motivated by NES. Given an input \( x \) and a small region \( S \) that contains \( x \) (i.e., \( S = S_\rho(x) \) defined earlier), the key idea is to consider a smoothed
objective as our optimization criterion:

\[
\min_\theta J(\theta) := \int f(x')\pi_S(x'|\theta)dx'
\]

(7.5)

where \( \pi_S(x'|\theta) \) is a probability density with support defined on \( S \). Compared with problem (7.1), this frame assumes that we can find a distribution over \( S \) such that the loss \( f(x') \) is small in expectation. Hence, a sample drawn from this distribution is likely adversarial. Furthermore, with appropriate \( \pi_S(\cdot|\theta) \), the objective \( J(\theta) \) is a smooth function of \( \theta \), and the optimization process of this formulation does not depend on any estimation of the gradient \( \nabla f(x_t) \). Therefore, it is not impeded by the non-smoothness of neural networks. Finally, the distribution over \( S \) can be parameterized in a much lower dimensional space \((\dim(\theta) \ll \dim(x))\), giving rise to more efficient algorithms than eq. (7.2) which directly works in the high-dimensional input space.

7.2.2.1 The distribution on \( S \)

In order to define a distribution \( \pi_S(x'|\theta) \) on \( S \), we take the following transformation of variable approach:

\[
x' = \text{proj}_S(g(z)), \quad z \sim \mathcal{N}(z|\mu,\sigma^2)
\]

(7.6)

where \( \mathcal{N}(z|\mu,\sigma^2) \) is an isometric normal distribution whose mean \( \mu \) and variance \( \sigma^2 \) are to be learned and the function \( g : \mathbb{R}^{\dim(\mu)} \mapsto \mathbb{R}^{\dim(x)} \) maps a normal instance to the space of the neural network input. We leave it to future work to explore the other types of distributions.

In this work, we implement the transformation of the normal variable by the following steps:
1. draw $z \sim \mathcal{N}(\mu, \sigma^2)$, compute $g(z)$ as

$$g(z) = \frac{1}{2} (\tanh(g_0(z)) + 1),$$

2. clip $\delta' = \text{clip}_p(g(z) - x)$, $p = 2$ or $\infty$, and

3. return $\text{proj}_S(g(z))$ as $x' = x + \delta'$

Step 1 draws a “seed” $z$ and then maps it by $g_0(z)$ to the space of the same dimension as the input $x$. In our experiments, we let $z$ lie in the space of the CIFAR10 images [121] (i.e., $\mathbb{R}^{32 \times 32 \times 3}$), so the function $g_0(\cdot)$ is an identity mapping for the experiments on CIFAR10 and a bilinear interpolation for the ImageNet images [35]. We further transform $g_0(z)$ to the same range as the input by $g(z) = \frac{1}{2} (\tanh(g_0(z)) + 1) \in [0, 1]^{\dim(x)}$ and then compute the offset $\delta = g(z) - x$ between the transformed vector and the input. Steps 2 and 3 detail how to project $g(z)$ onto the set $S$, where the clip functions are respectively

$$\text{clip}_2(\delta) = \begin{cases} 
\frac{\delta \tau_2}{\|\delta\|_2} & \text{if } \|\delta\|_2 > \tau_2 \\
\delta & \text{else}
\end{cases} \quad (7.7)$$

$$\text{clip}_\infty(\delta) = \min(\delta, \tau_\infty) \quad (7.8)$$

with the thresholds $\tau_2$ and $\tau_\infty$ given by users.

Thus far, we have fully specified our problem formulation (eq. (7.5)). Before discussing how to solve this problem, we recall that the set $S$ is the $\ell_p$-ball centered at $x$: $S = S_p(x)$. Since problem (7.5) is formulated for a particular input to the targeted DNN, the input $x$ also determines the distribution $\pi_S(z|\theta)$ via the dependency of $S$ on $x$. In other words, we will learn personalized distributions for different inputs.
7.2.2.2 Optimization

Let \( \text{proj}_S(g(z)) \) be steps 1–3 in the above variable transformation procedure. We can rewrite the objective function \( J(\theta) \) in problem (7.5) as

\[
J(\theta) = \mathbb{E}_{\mathcal{N}(z|\mu,\sigma)} f(\text{proj}(g(z))),
\]

where \( \theta = (\mu, \sigma^2) \) are the unknowns. We use grid search to find a proper bandwidth \( \sigma \) for the normal distribution and NES to find its mean \( \mu \):

\[
\mu_{t+1} \leftarrow \mu_t - \eta \nabla_{\mu} J(\theta)|_{\mu_t},
\]

whose SGD version over a mini-batch of size \( b \) is

\[
\mu_{t+1} \leftarrow \mu_t - \frac{\eta}{b} \sum_{i=1}^{b} f(\text{proj}_S(g(z_i))) \nabla_{\mu} \log \mathcal{N}(z_i|\mu_t, \sigma^2).
\]

In practice, we sample \( \epsilon_i \) from a standard normal distribution and then use a linear transformation \( z_i = \mu + \epsilon_i \sigma \) to make it follow the distribution \( \mathcal{N}(z|\mu, \sigma^2) \). With this notion, we can simplify \( \nabla_{\mu} \log \mathcal{N}(z_i|\mu_t, \sigma^2) \propto \sigma^{-1} \epsilon_i \).

Algorithm 1 summarizes the full algorithm, called \( \mathcal{NA}TTACK \), for optimizing our smoothed formulation in eq. (7.5). In line 6 of Algorithm 1, the z-score operation is to subtract from each loss quantity \( f_i \) the mean of the losses \( f_1, \cdots, f_b \) and divide it by the standard deviation of all the loss quantities. We find it stablizes \( \mathcal{NA}TTACK \); the algorithm converges well with a constant learning rate \( \eta \). Otherwise, one would have to schedule more sophisticated learning rates as reported in [95]. Regarding the loss function in line 5, we employ the C&W loss [20] in the experiments:

\[
f(x') := \max \left( 0, \log F(x')_y - \max_{c \neq y} \log F(x')_c \right).
\]
Algorithm 1: Black-box adversarial $\mathcal{N}$ATTACK

1. **Input:** DNN $F(\cdot)$, input $x$ and its label $y$, initial mean $\mu_0$, standard deviation $\sigma$, learning rate $\eta$, sample size $b$, and the maximum number of iterations $T$

2. **Output:** $\mu_T$, mean of the normal distribution

1: for $t = 0, 1, \ldots, T - 1$ do
2: Sample $\epsilon_1, \ldots, \epsilon_b \sim \mathcal{N}(0, I)$
3: Compute $g_i = g(\mu_t + \epsilon_i \sigma)$ by Step 1 $\forall i \in \{1, \ldots, b\}$
4: Obtain proj$(g_i)$ by steps 2–3, $\forall i$
5: Compute losses $f_i := f(\text{proj}(g_i)), \forall i$
6: Z-score $\hat{f}_i = (f_i - \text{mean}(f)) / \text{std}(f), \forall i$
7: Set $\mu_{t+1} \leftarrow \mu_t - \frac{\eta \sigma}{\sum_{i=1}^{b} \hat{f}_i \epsilon_i}$
8: end for

In order to generate an adversarial example for an input $x$ to the neural network classifier $C(\cdot)$, we use the $\mathcal{N}$ATTACK algorithm to find a probability density distribution over $S_p(x)$ and then sample from this distribution until arriving at an adversarial instance $x'$ such that $C(x') \neq C(x)$.

Note that our method differs from that of [95] in that we allow an arbitrary data transformation $g(\cdot)$ which is more flexible than directly seeking the adversarial example in the input space, and we absorb the computation of proj$_x(\cdot)$ into the function evaluation before the update of $\mu$ (line 7 of Algorithm 1). On the contrary, the projection of [95] is after the computation of the estimated gradient (which is similar to line 7 in Algorithm 1) because it is an estimation of the projected gradient. The difference in the computational order of projection is conceptually important because, in our case, the projection is treated as part of the function evaluation, which is more stable than treating it as an estimation of the projected gradient. Practically, this also makes a major difference, which can be seen from our experimental comparisons of the two approaches.

7.2.3 Initializing $\mathcal{N}$ATTACK by Regression

The initialization to the mean $\mu_0$ in Algorithm 1 plays a key role in terms of run time. When a good initialization is given, we often successfully find adversarial examples in less than 100
iterations. Hence, we propose to boost the \( \mathcal{N} \)\( \text{ATTACK} \) algorithm by using a regression neural network. It takes a benign example \( x \) as the input and outputs \( \mu_0 \) to initialize \( \mathcal{N} \)\( \text{ATTACK} \). In order to train this regressor, we generate many (input, adversarial example) pairs \( \{(x, x_{adv})\} \) by running \( \mathcal{N} \)\( \text{ATTACK} \) on the training set of benchmark datasets. The regression network’s weights are then set by minimizing the \( \ell_2 \) loss between the network’s output and \( g_0^{-1}(\arctan(2x_{adv} - 1)) - g_0^{-1}(\arctan(2x - 1)) \); in other words, we regress for the offset between the adversarial example \( x_{adv} \) and the input \( x \) in the space \( \mathbb{R}^{\dim(\mu)} \) of the distribution parameters. The supplementary materials present more details about this regression network.

7.3 Experiments

We use the proposed \( \mathcal{N} \)\( \text{ATTACK} \) to attack 13 defense methods for DNNs published in 2018 or 2019 and two representative vanilla DNNs. For each defense method, we run experiments using the same protocol as reported in the original paper, including the datasets and \( \ell_p \) distance (along with the threshold) to bound the differences between adversarial examples and inputs — this experiment protocol favors the defense method. In particular, CIFAR10 [121] is employed in the attack on nine defense methods and ImageNet [35] is used for the remaining four. We examine all the test images of CIFAR10 and randomly choose 1,000 images from the test set of ImageNet. 12 of the defenses concern the \( \ell_\infty \) distance between the adversarial examples and the benign ones and one works with the \( \ell_2 \) distance. We threshold the \( l_\infty \) distance in the normalized \([0, 1]^{\dim(x)} \) input space. The \( l_2 \) distance is normalized by the number of pixels.

In addition to the main comparison results, we also investigate the defense methods’ robustness versus the varying strengths of \( \mathcal{N} \)\( \text{ATTACK} \) (cf. Section 7.3.2). Specifically, we plot the attack success rate versus the attack iteration. The curves provide a complementary metric to the overall attack success rate, uncovering the dynamic traits of the competition between a defense and an
attack.

Finally, we examine the adversarial examples’ transferabilities between some of the defended neural networks (cf. Section 7.3.3). Results show that, unlike the finding that many adversarial examples are transferable across different vanilla neural networks, a majority of the adversarial examples that fail one defended DNN cannot defeat the others. In some sense, this weakens the practical significance of white-box attack methods which are often thought applicable to unknown DNN classifiers by attacking a substitute neural network instead [195].

### 7.3.1 Attacking 13 Most Recent Defense Techniques

We consider 13 defenses recently developed: adversarial training (ADV-TRAIN) [170], adversarial training of Bayesian DNNs (ADV-BNN) [157], Thermometer encoding (THERM) [19], THERM-ADV [10, 170], ADV-GAN [268], local intrinsic dimensionality (LID) [168], stochastic activation pruning (SAP) [37], random self-ensemble (RSE) [156], cascade adversarial training (CAS-ADV) [181], randomization [283], input transformation (INPUT-TRANS) [75], pixel deflection [206], and guided denoiser [148]. We describe them in detail in the supplementary materials. Additionally, we also include Wide Resnet-32 (WRESNET-32) [302] and INCEPTION V3 [243], two vanilla neural networks for CIFAR10 and ImageNet, respectively.

**Implementation Details.** In our experiments, the defended DNNs of SAP, LID, RANDOMIZATION, INPUT-TRANS, THERM, and THERM-ADV come from [10], the defended models of GUIDED DENOISER and PIXEL DEFLECTION are based on [9], and the models defended by RSE, CAS-ADV, ADV-TRAIN, and ADV-GAN are respectively from the original papers. For ADV-BNN, we attack an ensemble of ten BNN models. In all our experiments, we set $T = 600$ as the maximum number of optimization iterations, $b = 300$ for the sample size, variance of the isotropic Gaussian $\sigma^2 = 0.01$, and learning rate $\eta = 0.008$. NATTACK is able to defeat most of the defenses
under this setting and about 90% inputs for other cases. We then fine-tune the learning rate $\eta$ and sample size $b$ for the hard leftovers.

7.3.1.1 Attack success rates

We report in Table 7.1 the main comparison results evaluated by the attack success rate, the higher the better. Our $\mathcal{N}$ATTACK achieves 100% success on six out of the 13 defenses and more than 90% on five of the rest. As a single black-box adversarial algorithm, $\mathcal{N}$ATTACK is better than or on par with the set of powerful white-box attack methods of various forms [10], especially on the defended DNNs. It also significantly outperforms three state-of-the-art black-box attack methods: ZOO [23], which adopts the zero-th order gradients to find adversarial examples; QL [95], a query-limited attack based on an evolution strategy; and a decision-based (D-based) attack method [17] mainly generating $\ell_2$-bounded adversarial examples.

<table>
<thead>
<tr>
<th>Defense Technique</th>
<th>Dataset</th>
<th>Classification Accuracy %</th>
<th>Threshold &amp; Distance</th>
<th>Attack Success Rate %</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADV-TRAIN [170]</td>
<td>CIFAR10</td>
<td>87.3</td>
<td>0.031 ($L_\infty$)</td>
<td>46.9 16.9 40.3 – 47.9</td>
</tr>
<tr>
<td>ADV-BNN [157]</td>
<td>CIFAR10</td>
<td>79.7</td>
<td>0.035 ($L_\infty$)</td>
<td>48.3 – – – 75.3</td>
</tr>
<tr>
<td>THERM-ADV [10]</td>
<td>CIFAR10</td>
<td>88.5</td>
<td>0.031 ($L_\infty$)</td>
<td>76.1 0.0 42.3 – 91.2</td>
</tr>
<tr>
<td>CAS-ADV [181]</td>
<td>CIFAR10</td>
<td>75.6</td>
<td>0.015 ($L_\infty$)</td>
<td>85.0* 96.1 68.4 – 97.7</td>
</tr>
<tr>
<td>ADV-GAN [268]</td>
<td>CIFAR10</td>
<td>90.9</td>
<td>0.031 ($L_\infty$)</td>
<td>48.9 76.4 53.7 – 98.3</td>
</tr>
<tr>
<td>LID [168]</td>
<td>CIFAR10</td>
<td>66.9</td>
<td>0.031 ($L_\infty$)</td>
<td>95.0 92.9 95.7 – 100.0</td>
</tr>
<tr>
<td>THERM [19]</td>
<td>CIFAR10</td>
<td>92.8</td>
<td>0.031 ($L_\infty$)</td>
<td>100.0 0.0 96.5 – 100.0</td>
</tr>
<tr>
<td>SAP [37]</td>
<td>CIFAR10</td>
<td>93.3</td>
<td>0.031 ($L_\infty$)</td>
<td>100.0 5.9 96.2 – 100.0</td>
</tr>
<tr>
<td>RSE [156]</td>
<td>CIFAR10</td>
<td>91.4</td>
<td>0.031 ($L_\infty$)</td>
<td>– – – – 100.0</td>
</tr>
<tr>
<td>VANILLA WRESNET-32 [302]</td>
<td>CIFAR10</td>
<td>95.0</td>
<td>0.031 ($L_\infty$)</td>
<td>100.0 99.3 96.8 – 100.0</td>
</tr>
<tr>
<td>GUIDED denoiser [148]</td>
<td>ImageNet</td>
<td>79.1</td>
<td>0.031 ($L_\infty$)</td>
<td>100.0 – – – 95.5</td>
</tr>
<tr>
<td>RANDOMIZATION [283]</td>
<td>ImageNet</td>
<td>77.8</td>
<td>0.031 ($L_\infty$)</td>
<td>100.0 6.7 45.9 – 96.5</td>
</tr>
<tr>
<td>INPUT-TRANS [75]</td>
<td>ImageNet</td>
<td>77.6</td>
<td>0.05 ($L_2$)</td>
<td>100.0 38.3 66.5 66.0 100.0</td>
</tr>
<tr>
<td>PIXEL deflection [206]</td>
<td>ImageNet</td>
<td>69.1</td>
<td>0.015 ($L_\infty$)</td>
<td>97.0 – 8.5 – 100.0</td>
</tr>
<tr>
<td>VANILLA INCEPTION V3 [243]</td>
<td>ImageNet</td>
<td>78.0</td>
<td>0.031 ($L_\infty$)</td>
<td>100.0 62.1 100.0 – 100.0</td>
</tr>
</tbody>
</table>

Table 7.1: Adversarial attack on 13 recently published defense methods. (* the number reported in [10]. For all the other numbers, we obtain them by running the code released by the authors or implemented ourselves with the help of the authors. For D-based and ADV-TRAIN, we respectively report the results on 100 and 1000 images only because they incur expensive computation costs.)
Notably, \texttt{ADV-TRAIN} is still among the best defense methods, so is its extension to the Bayesian DNNs (i.e., \texttt{ADV-BNN}). However, along with \texttt{CAS-ADV} and \texttt{THERM-ADV} which are also equipped with the adversarial training, their strengths come at the price that they give worse classification performances than the others on the clean inputs (cf. the third column of Table 7.1). Moreover, \texttt{ADV-TRAIN} incurs extremely high computation cost. When the image resolutions are high, Kurakin \textit{et al.} [127] found that it is difficult to run the adversarial training at the ImageNet scale. Since our \texttt{NATTACK} enables efficient generation of adversarial examples once we learn the distribution, we can potentially scale up the adversarial training with \texttt{NATTACK} and will explore it in the future work.

We have tuned the main free parameters of the competing methods (e.g., batch size and bandwidth in QL). ZOO runs extremely slow with high-resolution images, so we instead use the hierarchical trick the authors described [23] for the experiments on ImageNet. In particular, we run ZOO starting from the attack space of $32 \times 32 \times 3$, lift the resolution to $64 \times 64 \times 3$ after 2,000 iterations and then to $128 \times 128 \times 3$ after 10,000 iterations, and finally up-sample the result to the same size as the DNN input with bilinear interpolation.

7.3.1.2 Ablation study and run-time comparison

\texttt{NATTACK} vs. \texttt{QL}. We have discussed the conceptual differences between \texttt{NATTACK} and QL [95] in Section 7.2 (e.g., \texttt{NATTACK} formulates a smooth optimization criterion and offers a probability density on the $\ell_p$-ball of an input). Moreover, the comparison results in Table 7.1 verify the advantage of \texttt{NATTACK} over QL in terms of the overall attack strengths. Additionally, we here conduct an ablation study to investigate two major algorithmic differences between them: \texttt{NATTACK} absorbs the projection (\texttt{proj$_S$}) into the objective function and allows an arbitrary change of variable transformation $g(\cdot)$. Our study concerns THERM-ADV and SAP, two defended DNNs on which
QL respectively reaches 42.3% and 96.2% attack success rates. After we instead absorb the projection in QL into the objective, the results are improved to 54.7% and 97.7%, respectively. If we further apply $g(\cdot)$, the change of variable procedure (cf. Steps 1–3), the success rates become 83.3% and 98.9%, respectively. Finally, with the z-score operation (line 6 of Algorithm 1), the results are boosted to 90.9%/100%, approaching $\mathcal{N}$ATTACK’s 91.2%/100%. Therefore, we say that $\mathcal{N}$ATTACK boosts QL’s performance, thanks to both the smoothed objective and the transformation $g(\cdot)$.

$\mathcal{N}$ATTACK vs. the White-Box BPDA Attack. While BPDA achieves high attack success rates by different variants for handling the diverse defense techniques, $\mathcal{N}$ATTACK gives rise to better or comparable results by a single universal algorithm. Additionally, we compare them in terms of the run time in the supplementary materials; the main observations are the following. On CIFAR10, BPDA and $\mathcal{N}$ATTACK can both find an adversarial example in about 30s. To defeat an ImageNet image, it takes $\mathcal{N}$ATTACK about 71s without the regression network and 48s when it is equipped with the regression net; in contrast, BPDA only needs 4s. It is surprising to see that BPDA is almost 7 times faster at attacking a DNN for ImageNet than a DNN for CIFAR10. It is probably because the gradients of the former are not “obfuscated” as well as the latter due to the higher resolution of the ImageNet input.

7.3.2 Attack Success Rate vs. Attack Iteration

The $\mathcal{N}$ATTACK algorithm has an appealing property as follows. In expectation, the loss (eq. (7.5)) decreases at every iteration and hence a sample drawn from the distribution $\pi_S(x|\theta)$ is adversarial with higher chance. Though there could be oscillations, we find that the attack strengths do grow monotonically with respect to the evolution iterations in our experiments. Hence, we propose a new curve shown in Figure 7.1a featuring the attack success rate versus number of evolution
Figure 7.1: (a) Success rate versus run steps of $\mathcal{N}\text{Attack}$. (b) Comparison results with QL measured by the log of average number of queries per successful image. The solid lines denote $\mathcal{N}\text{Attack}$ and the dashed lines illustrate QL.

iterations — strength of attack. For the experiment here, the Gaussian mean $\mu_0$ is initialized by $\mu_0 \sim \mathcal{N}(g_0^{-1}(\arctan(2x - 1)), \sigma^2)$ for any input $x$ to maintain about the same starting points for all the curves.

Figure 7.1a plots eight defense methods on CIFAR10 along with a vanilla DNN. It is clear that $\text{ADV-TRAIN}$, $\text{ADV-BNN}$, $\text{THERM-ADV}$, and $\text{CAS-ADV}$, which all employ the adversarial training strategy, are more difficult to attack than the others. What’s more interesting is with the other five DNNs. Although $\mathcal{N}\text{Attack}$ completely defeats them all by the end, the curve of the vanilla DNN is the steepest while the SAP curve rises much slower. If there are constraints on the computation time or the number of queries to the DNN classifiers, SAP is advantageous over the vanilla DNN, RSE, THERM, and LID.

Note that the ranking of the defenses in Table 7.1 (evaluation by the success rate) is different from the ordering on the left half of Figure 7.1a, signifying the attack success rate and the curve mutually complement. The curve reveals more characteristics of the defense methods especially when there
are constraints on the computation time or number of queries to the DNN classifier.

Figure 7.1b shows $\mathcal{N}$ATTACK (solid lines) is more query efficient than the QL attack [95] (dashed lines) on 6 defenses under most attack success rates and the difference is even amplified for higher success rates. For SAP, $\mathcal{N}$ATTACK performs better when the desired attack success rate is bigger than 80%.

### 7.3.3 Transferability

We also study the transferability of adversarial examples across different defended DNNs. This study differs from the earlier ones on vanilla DNNs [244, 158]. We investigate both the white-box attack BPDA and our black-box $\mathcal{N}$ATTACK.

Following the experiment setup in [127], we randomly select 1000 images for each targeted DNN such that they are classified correctly, and yet the adversarial images of them are classified incorrectly. We then use the adversarial examples of the 1000 images to attack the other DNNs. In addition to the defended DNNs, we also include two vanilla DNNs for reference: VANILLA-1 and VANILLA-2. VANILLA-1 is a light-weight DNN classifier built by [20] with 80% accuracy on CIFAR10. VANILLA-2 is the Wide-ResNet-28 [302] which gives rise to 92.3% classification accuracy on CIFAR10. For fair comparison, we change the threshold $\tau_\infty$ to 0.031 for CAS-ADV. We exclude RSE and CAS-ADV from BPDA’s confusion table because it is not obviously clear how to attack RSE using BPDA and the released BPDA code lacks the piece for attacking CAS-ADV.

The confusion tables of BPDA and $\mathcal{N}$ATTACK are shown in Figure 7.2, respectively, where each entry indicates the success rate of using the adversarial examples originally targeting the row-wise defense model to attack the column-wise defense. Both confusion tables are asymmetric; it is easier to transfer from defended models to the vanilla DNNs than vice versa. Besides, the overall
transferrability is lower than that across the DNNs without any defenses [158]. We highlight some additional observations below.

Firstly, the transferability of our black-box $\mathcal{N}$ATTACK is not as good as the black-box BPDA attack. This is probably because BPDA is able to explore the intrinsically common part of the DNN classifiers — it has the privilege of accessing the true or estimated gradients that observe the DNNs’ architectures and weights.

Secondly, both the network architecture and defense methods can influence the transferability. VANILLA-2 is the underlying classifier of SAP, THERM-Adv, and THERM. The adversarial examples originally attacking VANILLA-2 do transfer better to SAP and THERM than to the others probably because they share the same DNN architecture, but the examples achieve very low success rate on THERM-Adv due to the defense technique.

Finally, the transfer success rates are low no matter from THERM-Adv to the other defenses or
vice versa, and \textsc{Adv-Train} and \textsc{Adv-BNN} lead to fairly good results of transfer attacks on the other defenses and yet themselves are robust against the adversarial examples of the other defended DNNs. The unique result of \textsc{THERM-Adv} probably attributes to its use of double defense techniques, i.e., Thermometer encoding and adversarial training.

7.4 Summary

In this chapter, we present a black-box adversarial attack method which learns a probability density on the $\ell_p$-ball of a clean input to the targeted neural network. One of the major advantages of our approach is that it allows an arbitrary transformation of variable $g(\cdot)$, converting the adversarial attack to a space of much lower dimensional than the input space. Experiments show that our algorithm defeats 13 defended DNNs, better than or on par with state-of-the-art white-box attack methods. Additionally, our experiments on the transferability of the adversarial examples across the defended DNNs show different results reported in the literature: unlike the high transferability across vanilla DNNs, it is difficult to transfer the attacks on the defended DNNs.
8.1 Problem Introduction

Data curation is one of the most important steps for learning high-performing visual recognition models. However, it is often tedious and sometimes daunting to collect large-scale relevant data that have sufficient coverage of the inference-time scenarios. Additionally, labeling the collected data is time-consuming and costly.

Given a new task, how can we learn a high-quality machine learning model in a more data-efficient manner? We believe the answer varies depending on specific application scenarios. In this work, we focus on the case that there exists a blackbox teacher model whose capability covers our task of interest. Indeed, there are many high-performing generic visual recognition models available as Web-based APIs, in our smart devices, or even as an obsolete model built by ourselves some while ago. The challenge is, however, we often have limited knowledge about their specifics, e.g., not knowing the exact network architecture or weights. Moreover, it could be computationally and/or financially expensive to query the models and read out their outputs for a large-scale dataset.

To this end, we study how to distill a blackbox teacher model for visual recognition into a student neural network in a data-efficient manner. Our objective is three-fold. First of all, we would like
Figure 8.1: Data-efficient blackbox knowledge distillation. Given a blackbox teacher model and a small set of unlabeled images, we propose to employ mixup [307] and active learning [138] to train a high-performing student neural network in a data-efficient manner (b) so that we do not need to re-do the heavy and expensive data curation used to train the teacher model (a).

The distilled student network to perform well as the teacher model as possible at the inference time. Besides, we try to minimize the number of queries to the blackbox teacher model to save costs. Finally, we also shall use as a small number of examples as possible to save data collection efforts. It is hard to collect abundant data for rare classes or privacy-critical applications.

We propose to blend active learning [255, 138] and image mixup [307] to tackle the data-efficient knowledge distillation from a blackbox teacher model. The main idea is to synthesize a big pool of images from the few training examples by mixup and then use active learning to select from the pool the most helpful subset to query the teacher model. After reading out the teacher model’s outputs, we simply treat them as the “groundtruth labels” of the query images and train the student
neural network with them.

Image mixup [307, 76, 15] was originally proposed for data augmentation to improve the generalization performance of a neural recognition network. It synthesizes a virtual image by a convex combination of two training images. While the resultant image may become cluttered and semantically meaningless, it resides near the manifold of the natural images — unlike white-noise images. Given 1000 images, we can construct $O(10^5)$ pairs, each of which can further generate tens to thousands of virtual images depending on the coefficients in the convex combination. We conjecture that the big pool of mixup images provides good coverage of the manifold of natural images. Hence, we expect that a student network that imitates the blackbox teacher on the mixup images can give rise to similar predictions over the test images as the teacher model does.

Instead of querying the blackbox teacher model by all the mixup images, we resort to active learning to improve the querying efficiency. We first acquire the labels of the small number of original images from the blackbox teacher model and use them for training the student network. We then apply the student network to all the mixup images to identify the subset with which the current student network is the most uncertain. Notably, if two mixup images are synthesized from the same pair of original images, we keep only the one with higher uncertainty. We query labels for this subset, merge it into the previously labeled data, and then re-train the student network. We iterate this procedure of subset selection, querying the blackbox teacher model, and training the student neural network multiple times until reaching a stopping criterion.

To the best of our knowledge, we are the first to distill knowledge from a blackbox teacher model while underscoring the need for data-efficiency and query-efficiency. We empirically validate our approach by contrasting it to both vanilla and few/zero-shot knowledge distillation methods. Experiments show that, despite the blackbox teacher in our work, our approach performs on par or better than the competing methods that learn from whitebox teachers.
Note that the mixup images are often semantically meaningless, making them almost impossible for human raters to label. However, the blackbox teacher model returns predictions for them regardless, and the student network still gains from such fake image-label pairs. In this sense, we say that the blackbox teacher model is more productive than human raters in teaching the student network.

8.2 Approach

We present our approach to the data-efficient knowledge distillation from a blackbox teacher model in detail in this section. Given a blackbox teacher model and a small number of unlabeled images, the approach iterates over the following three steps: 1) constructing a big candidate pool of synthesized images from the small number of unlabeled images, 2) actively choosing a subset from the pool with which the current student network is the most uncertain, 3) querying the blackbox teacher model to acquire labels for this subset and to re-train the student network.

8.2.1 Constructing a Candidate Pool

In real-world applications, data collection could consume a huge amount of time due to various reasons, such as privacy concerns, rare classes, data quality, etc. Instead of relying on a big dataset of real images, we begin with a small number of unlabeled images and use the recently proposed mixup [307] to augment this initial image pool.

Given two natural images $x_i$ and $x_j$, mixup generates multiple synthetic images by a convex combination of the two with different coefficients,

$$\hat{x}_{ij}(\lambda) = \lambda x_i + (1 - \lambda)x_j,$$

(8.1)
where the coefficient $\lambda \in [0, 1]$. Note that this notation also includes the original unlabeled data $x_i$ and $x_j$ when $\lambda = 1$ and $\lambda = 0$, respectively.

This technique comes handy and effective for our work. It can exponentially expand the size of the initial image pool. Suppose we have collected 1000 natural images, and we generate 10 mixup images for each image pair by varying the coefficient $\lambda$. We then arrive at a pool of about $10^6$ images in total. Besides, this pool of synthetic images also provides good coverage of the manifold of natural images. Indeed, this pool can be viewed as a dense sampling of the convex hull of the natural images we have collected. The test images likely fall into or close to this convex hull if the collected images are diverse and representative. Hence, we expect the student neural network to generalize well to the inference-time data by enforcing it to imitate the blackbox teacher model on the mixup images.

8.2.2 Actively Choosing a Subset to Query the Teacher Model

Let $\{\hat{x}_{ij}(\lambda), \lambda \in [0, 1], i \neq j\}$ denote the augmented pool of images. It is straightforward to query the teacher model to obtain the (soft) labels for these synthetic images and then train the student network with them. However, this brute-force strategy incurs high computational and financial costs. Instead, we employ active learning to reduce the cost.

We define the student neural network’s confidence over an input $x$ as

$$C_1(x) := \max_y P_S(y|x),$$

(8.2)

where $P_S(y|x)$ is the probability of the input image $x$ belonging to the class $y$ predicted by the current student network. Intuitively, the less confidence the student network has over the input $x$, the more the student network can gain from the teacher model’s label for the input.
Therefore, we could rank all the synthetic images in the candidate pool according to the student network’s confidences on them, and then choose the top ones as the query subset. However, this simple strategy results in near-duplicated images, for example $\hat{x}_{ij}(\lambda = 0.5)$ and $\hat{x}_{ij}(\lambda = 0.55)$. We avoid this situation by choosing at most one image from any pair of images.

In particular, instead of ranking the synthetic images, we rank image pairs in the candidate pool. We define the confidence of the student network over an image pair $x_i$ and $x_j$ as the following,

$$C_2(x_i, x_j) := \min_{\lambda} C_1(\hat{x}_{ij}(\lambda)), \quad \lambda \in [0, 1],$$

which depends on a coefficient $\lambda^*$ for the image pair. Hence, we obtain a confidence score and its corresponding coefficient for any pair of the original images. The synthetic image $\hat{x}_{ij}(\lambda^*)$ is selected into the query set if the confidence score $C_2(x_i, x_j)$ is among the lowest $k$ ones. We study the size of the query set in the experiments.

**8.2.3 Training the Student Network**

With the actively selected query set of images, we query the blackbox teacher model and read out its soft predictions as the labels for the images. We then merge them with the previous training set, if there is, to train the student network using a cross-entropy loss. The soft probabilistic labels returned by the teacher model give rise to slightly better results than the hard labels, so we shall use the soft labels in the experiments below.

**8.2.4 Overall Algorithm**

Algorithm 2 presents the overall procedure of our approach to the data-efficient blackbox knowl-
Algorithm 2: Data-efficient blackbox knowledge distillation

1 INPUT: Pre-trained teacher model $M^T$
2 INPUT: A small set of unlabeled images $X = \{x_i\}_{i=1}^n$
3 INPUT: Hyper-parameters (learning rate, subset size, etc.)
4 OUTPUT: Student network $M^S$

1: Query $M^T$ and acquire labels $Y_0$ for all images in $X$
2: Train an initial student network $M_0^S$ with $(X, Y_0)$
3: Construct a synthetic image pool $P = \{\hat{x}_{ij}(\lambda)\}$ by using the unlabeled images $X$ with eq. (8.1)
4: Initialize $P_1^s = X$, $Y_1 = Y_0$
5: for $t = 1, 2, ..., T$ do
6: Select a subset $\Delta P_t^s$ from $P$ with lowest confidence scores $\{C_2(x_i, x_j)\}$ returned by student $M_{t-1}^S$
7: Query $M^T$, acquire labels $\Delta Y_t$ for all images $\Delta P_t^s$
8: $P_t^s \leftarrow P_t^s \cup \Delta P_t^s$, $Y_t \leftarrow Y_t \cup \Delta Y_t$
9: Train a new student network $M_t^S$ with $(P_t^s, Y_t)$
10: Update $P \leftarrow P - \Delta P_t^s$
11: end for

edge distillation. Beginning with a teacher model $M^T$ and a few unlabeled images $X = \{x_1, x_2, ..., x_n\}$, we firstly train an initial student network $M_0^S$ with $(X, Y_0)$, where $Y_0$ contains the labels for the images in $X$ and is obtained by querying the teacher model. We then construct a big pool of synthetic images $P$ with mixup [307] (eq. (8.1)) to facilitate the active learning stage. We iterate the following steps until the accuracy of the student network converges. 1) Actively select a subset $\Delta P_t^s$ of the synthetic images $P$ with the lowest confidence scores, $C_2(x_i, x_j)$, as predicted by the current student network so that the resulting subset $\Delta P_t^s$ contains hard samples for the current student network $M_{t-1}^S$. 2) Acquire labels $\Delta Y_t$ of the selected subset of synthetic images $\Delta P_t^s$ by querying the teacher model. 3) Train a new student network $M_t^S$ with all the labeled images thus far, $(P_t^s, Y_t)$. Note that, in Line 6 of Algorithm 2, we only keep one synthetic image for any pair $(x_i, x_j)$ of the original images to reduce redundancy.
8.3 Experiments

We design various experiments to test our approach, including both comparison experiments with state-of-the-art knowledge distillation methods and ablation studies. Additionally, we also challenge our approach when the available data is out of the distribution of the main task of interest. In practice, across all experiments, we select $\lambda \in \{0.3, 0.7\}$ (with an interval of 0.04) to generate synthetic images to produce more diverse mixup images.

8.3.1 Comparison Experiments

Since our main objective is to explore how to train a high-performing student neural network from a blackbox teacher model in a data-efficient manner, it is worth comparing our approach with existing knowledge distillation methods although they were developed for other setups. The comparison can help review how data-efficient our approach is given the blackbox teacher model.

8.3.1.1 Experiment Setting

Datasets. We run experiments on MNIST [130], Fashion-MNIST [280], CIFAR-10 [121], and Places365-Standard [321], which are popular benchmark datasets for image classification. The MNIST dataset contains 60K training images and 10K testing images about ten handwritten digits. The image resolution is 28×28. Fashion-MNIST is composed of 60K training and 10K testing fashion product images of the size 28×28. CIFAR-10 consists of 60K (50K training images and 10K test images) 32×32 RGB images in 10 classes, with 6K images per class. In addition to evaluating the proposed approach on the above described low-resolution images, we also test our approach on Places365-Standard, which is a challenging dataset for natural scene recognition. It has 1.8M training images and 18,250 validation images in 365 classes. We use the resolution of
Table 8.1: Comparison results on Places365-Standard, CIFAR-10, MNIST, and Fashion-MNIST. The “Teacher” column reports the teacher model’s accuracy on the test sets, “KD Accuracy” is the student network’s test accuracy, “Success” stands for the distillation success rates, “Black/White” indicates whether or not the teacher model is blackbox, “Queries” lists the numbers of queries into the teacher models, and “Unlabeled Data” shows the numbers of original training images used in the experiments. (* results reported in the original paper)

<table>
<thead>
<tr>
<th>Task (Model)</th>
<th>Teacher</th>
<th>KD Accuracy</th>
<th>Success</th>
<th>Black/White</th>
<th>Queries</th>
<th>Unlabeled Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Places365-Standard (ZSKD) [182]</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>0</td>
</tr>
<tr>
<td>Places365-Standard (FSKD [117])</td>
<td>53.69</td>
<td>38.18</td>
<td>71.11</td>
<td>White</td>
<td>480,000</td>
<td>80,000</td>
</tr>
<tr>
<td>Places365-Standard (KD)</td>
<td>53.69</td>
<td>49.01</td>
<td>90.35</td>
<td>Black</td>
<td>1,800,000</td>
<td>1,800,000</td>
</tr>
<tr>
<td>Places365-Standard (Ours)</td>
<td>53.69</td>
<td><strong>45.71</strong></td>
<td><strong>85.14</strong></td>
<td>Black</td>
<td>480,000</td>
<td>80,000</td>
</tr>
<tr>
<td>CIFAR-10 (ZSKD) [182]</td>
<td>83.03*</td>
<td>69.56*</td>
<td>83.78</td>
<td>White</td>
<td>&gt;2,000,000</td>
<td>0</td>
</tr>
<tr>
<td>CIFAR-10 (FSKD [117])</td>
<td>83.07</td>
<td>40.58</td>
<td>48.85</td>
<td>White</td>
<td>40,000</td>
<td>2,000</td>
</tr>
<tr>
<td>CIFAR-10 (KD)</td>
<td>83.07</td>
<td>80.01</td>
<td>96.31</td>
<td>Black</td>
<td>50,000</td>
<td>50,000</td>
</tr>
<tr>
<td>CIFAR-10 (Ours)</td>
<td>83.07</td>
<td><strong>74.60</strong></td>
<td><strong>89.87</strong></td>
<td>Black</td>
<td>40,000</td>
<td>2,000</td>
</tr>
<tr>
<td>MNIST (ZSKD) [182]</td>
<td>99.34*</td>
<td><strong>98.77</strong></td>
<td>99.42</td>
<td>White</td>
<td>&gt;1,200,000</td>
<td>0</td>
</tr>
<tr>
<td>MNIST (FSKD [117])</td>
<td>99.29</td>
<td>80.43</td>
<td>81.01</td>
<td>White</td>
<td>24,000</td>
<td>2,000</td>
</tr>
<tr>
<td>MNIST (KD)</td>
<td>99.29</td>
<td>99.05</td>
<td>99.76</td>
<td>Black</td>
<td>60,000</td>
<td>60,000</td>
</tr>
<tr>
<td>MNIST (Ours)</td>
<td>99.29</td>
<td>98.74</td>
<td><strong>99.45</strong></td>
<td>Black</td>
<td>24,000</td>
<td>2,000</td>
</tr>
<tr>
<td>Fashion-MNIST (ZSKD) [182]</td>
<td>90.84*</td>
<td>79.62*</td>
<td>87.85</td>
<td>White</td>
<td>&gt;2,400,000</td>
<td>0</td>
</tr>
<tr>
<td>Fashion-MNIST (FSKD [117])</td>
<td>90.80</td>
<td>68.64</td>
<td>75.60</td>
<td>White</td>
<td>48,000</td>
<td>2,000</td>
</tr>
<tr>
<td>Fashion-MNIST (KD)</td>
<td>90.80</td>
<td>87.79</td>
<td>96.69</td>
<td>Black</td>
<td>60,000</td>
<td>60,000</td>
</tr>
<tr>
<td>Fashion-MNIST (Ours)</td>
<td>90.80</td>
<td><strong>80.90</strong></td>
<td><strong>89.10</strong></td>
<td>Black</td>
<td>48,000</td>
<td>2,000</td>
</tr>
</tbody>
</table>

256×256 for Places365-Standard in the following experiments.

**Evaluation Metric.** We mainly use the classification accuracy as the evaluation metric. Additionally, we also propose a straightforward metric to measure how much “knowledge” the student network distills from the teacher model. This metric is computed as the ratio between the student network’s classification accuracy and the teacher’s accuracy, and we call it the distillation success rate.

**Blackbox Teacher Models.** For each task except Places365-Standard, we prepare a teacher model by following the training setting provided in [182]. For Places365-Standard, there is no training setting reference for the knowledge distillation research yet, so we use a pre-trained model from
the dataset repository [321] as our teacher model. On MNIST and Fashion-MNIST, we use the LeNet-5 architecture [131] as the teacher model and optimize it to achieve 99.29% and 90.80% top-1 accuracies, respectively. On CIFAR-10, we have an AlexNet [122] as the teacher model and train it to obtain 83.07% top-1 accuracy. As shown in Table 8.1, the above teacher models are comparable to the teacher models in [182]: 83.03% vs. 83.07% on CIFAR-10, 99.34% vs. 99.29% on MNIST, and 90.84% vs. 90.87% on Fashion-MNIST. For Places365-Standard, the teacher model is a ResNet-18 [84] and yields 53.68% top-1 accuracy.

**Competing Methods.** We identify three existing relevant methods for comparison.

- One is zero-shot knowledge distillation (ZSKD) [182], which distills a student neural network with zero training example from a *whitebox* teacher model. It synthesizes data by backpropagating gradients to the input through the whitebox teacher network.

- The second method is few-shot knowledge distillation (FSKD) [117], which augments the training images by generating adversarial examples. It is the most relevant work to ours, but it depends on the computationally expensive adversarial examples [244] and has no active learning scheme to reduce the query cost at all. The original work assumes a *whitebox* teacher neural network so that it is straightforward to produce the adversarial examples, whereas there exist blackbox attack methods [145, 23].

- The third is the vanilla knowledge distillation [89], which accesses the whole training set of the teacher model and is somehow an upper bound of our method.

### 8.3.1.2 Quantitative Results

Table 8.1 shows the comparison results. For simplicity, we run the active learning stage for only one step (i.e., $T = 1$ in Algorithm 1). Section 8.3.2 presents the results of running it for multiple
Accuracy. Our approach significantly outperforms FSKD over all the datasets. On CIFAR-10, MNIST, and Fashion-MNIST, ours yields 41%, 18%, and 14% success rate improvements over FSKD, respectively. On Places365-Standard, whose images are high-resolution about natural scenes, we also outperform FSKD by 14% success rate. Compared to ZSKD, which relies on a whitebox teacher network, our approach also shows higher accuracies and success rates except on MNIST. We were not able to reproduce ZSKD on Places365-Standard because its images are all high-resolution, making it computationally infeasible to generate a large number of gradient-based inputs. Similarly, the advantage of ours over ZSKD is larger on CIFAR-10 than other MNIST or Fashion-MNIST, probably because the CIFAR-10 images have a higher resolution. In contrast, the computation cost of our active mixup approach does not depend on the input resolution. Overall, the results indicate that active mixup has a higher potential to solve the larger-scale knowledge distillation in a data-efficient manner.

Queries. Our approach saves orders of queries into the teacher model compared to ZSKD. For example, we only query the blackbox teacher model up to 40K times for CIFAR-10. In contrast, ZSKD requires more than 2M queries and yet yields lower accuracy than ours. The big difference is not surprising because the gradient-based inputs in ZSKD are less natural than or representative of the test images than our mixup images. Besides, ZSKD incurs additional queries into the whitebox teacher model every time it produces an input.

8.3.1.3 Qualitative Intermediate Results

We show some mixup images in Figures 8.2 and 8.3. These images are selected from the candidate pool constructed using the natural images in the Places365-Standard training set. Figure 8.2 shows some mixup images with low confidence scores. They can potentially benefit the student network
Figure 8.2: Mixup images whose confidence scores (cf. eq. (8.3)) are the lowest among all candidates in the third iteration. For each mixup image, we show the top three labels and probabilities returned by the blackbox teacher model.

Figure 8.3: Different mixup images from the same pair of the original images by varying the mixup coefficient $\lambda$. We show the top three labels and probabilities predicted by the teacher model for each of them. It is interesting to see how the top-1 label changes from Hockey Arena, to Baseball Field, and to Golf Course.

more than the other candidate images if we use them to query the teacher model. Figure 8.3 demonstrates some mixup images synthesized from the same pair of natural images by varying the mixup coefficient $\lambda$. It is interesting to see that the mix of “Hockey Arena” and “Golf Course” leads to a “Baseball Field” at $\lambda = 0.46$ predicted by the blackbox teacher model. This indicates that our active mixup approach can effectively augment the originally small training set by not only bringing in new synthetic images but also comprehensive coverage of classes.
8.3.2 Ablation Study

We select CIFAR-10 and Places365-Standard to study our approach in detail since they represent the small-scale and large-scale settings, respectively. For CIFAR-10, we switch to VGG-16 \cite{231} as the blackbox teacher model, which gives rise to 93.31% top-1 accuracy.

8.3.2.1 Data-Efficiency and Query-Efficiency

We investigate how the results of our active mixup approach change as we vary the total number of unlabeled real images (data-efficiency) and the number of synthetic images selected by the active learning scheme (query-efficiency). Here we run only one step of the active learning stage ($T = 1$ in Algorithm 1) to save computation cost. Tables 8.2 and 8.3 show the results on CIFAR-10 and Places365-Standard, respectively. Each entry in the tables is a classification accuracy on the test set, and it is obtained by a student network which we distill by using the corresponding number of unlabeled real images (Real images) and the number of selected synthetic images (Selected Syn.).

Table 8.2: Classification accuracy on CIFAR-10 with different numbers of real images and selected synthetic images.

<table>
<thead>
<tr>
<th>Selected Syn.</th>
<th>0.5K</th>
<th>1K</th>
<th>2K</th>
<th>4K</th>
<th>8K</th>
<th>16K</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>44.72</td>
<td>56.87</td>
<td>68.09</td>
<td>76.59</td>
<td>83.61</td>
<td>86.89</td>
</tr>
<tr>
<td>5K</td>
<td>66.97</td>
<td>71.67</td>
<td>77.76</td>
<td>81.76</td>
<td>85.76</td>
<td>87.05</td>
</tr>
<tr>
<td>10K</td>
<td>73.60</td>
<td>77.27</td>
<td>81.27</td>
<td>83.27</td>
<td>86.56</td>
<td>88.79</td>
</tr>
<tr>
<td>20K</td>
<td>77.44</td>
<td>81.18</td>
<td>84.19</td>
<td>86.29</td>
<td>88.07</td>
<td>89.01</td>
</tr>
<tr>
<td>40K</td>
<td>82.28</td>
<td>84.25</td>
<td>86.06</td>
<td>87.71</td>
<td>89.00</td>
<td>90.49</td>
</tr>
<tr>
<td>80K</td>
<td>85.18</td>
<td>86.53</td>
<td>87.89</td>
<td>88.71</td>
<td>89.61</td>
<td>90.96</td>
</tr>
<tr>
<td>160K</td>
<td>86.56</td>
<td>88.94</td>
<td>89.42</td>
<td>90.26</td>
<td>90.87</td>
<td>91.51</td>
</tr>
</tbody>
</table>

We can see that the more synthetic images we select by their confidence scores (cf. eq. (8.3)), the
Table 8.3: Classification accuracy on Places365-Standard with different numbers of real images and selected synthetic images.

<table>
<thead>
<tr>
<th>Selected Syn.</th>
<th>20K</th>
<th>40K</th>
<th>80K</th>
</tr>
</thead>
<tbody>
<tr>
<td>100K</td>
<td>40.72</td>
<td>41.95</td>
<td>43.52</td>
</tr>
<tr>
<td>200K</td>
<td>41.15</td>
<td>42.86</td>
<td>44.77</td>
</tr>
<tr>
<td>400K</td>
<td>41.94</td>
<td>43.42</td>
<td>45.71</td>
</tr>
</tbody>
</table>

higher-quality the distilled student network is. It indicates that the mixup images can effectively boost the performance of our method. Meanwhile, the higher the number of unlabeled real images we have, the higher the distillation success rate we can achieve. What’s more interesting is that, when the number of synthetic images is high (e.g., 160K), the gain is diminishing as we increase the number of real images. Hence, depending on the application scenarios, we have the flexibility to trade-off the real images and synthetic images for achieving a certain distillation success rate.

We can take a closer look at Tables 8.2 and 8.3 to obtain an understanding about the “market values” of the selected synthetic images. In Table 8.2, 10K selected synthetic images and 8K unlabeled real images yield 86.56% accuracy; 20K synthetic images and 4K real images lead to 86.29% accuracy; and 40K synthetic images with 2K real examples give rise to 86.06% accuracy. The accuracies are about the same. There is a similar trend along the off-diagonal entries in Table 8.3, implying that if we reduce the number of real images by half, we can complement it by doubling the size of synthetic images to maintain about the same distillation success rate.

8.3.2.2 Active Mixup vs. Random Search

We design another experiment to compare active mixup with the random search to understand the effectiveness of our active learning scheme. We keep 500 real images for CIFAR-10 and
20K for Places365-Standard. We then use them to construct 100K and 300K synthetic images, respectively. For a fair comparison, we let random search and active mixup share the same sets of natural images. Since our active learning scheme avoids selecting redundant images by using the improved confidence score in eq. (8.3), we also equip the random search such capability by using a single mixup coefficient of $\lambda = 0.5$ to construct the synthetic images. This guarantees that, like our approach, no two synthetic images selected by the random search are from the same pair of real images.

Figure 8.4: Test accuracy of student networks vs. number of queries into the blackbox teacher model on CIFAR-10 (left) and Places365-Standard (right). We use 500 and 20K natural images for the two datasets, respectively. The plot for CIFAR-10 starts from first active learning stage ($t = 1$ in Algorithm 1) and the one for Places365 starts from the initial student network training by natural images. The initial student network for CIFAR-10 trained by using natural images only yields 43.67% accuracy.

Figure 8.4 shows the comparison results of our active mixup and the random search. On CIFAR-10, we select 10K synthetic images every time and run the active learning stage for 10 steps ($T = 10$ in Algorithm 1). On Places365-Standard, we run it for six steps and choose 50K synthetic images.
per step. We can see that active mixup significantly outperforms random search over the whole course of knowledge distillation, verifying its effectiveness on improving the query-efficiency. More concretely, 80K actively selected synthetic images yield 86.76% accuracy, which is about the same as what 160K randomly selected synthetic images can achieve on CIFAR-10. Similarly, 40K synthetic images by active mixup lead to 84.2% accuracy, on par with the 85.18% accuracy by 80K randomly chosen synthetic images.

8.3.2.3 Active Mixup vs. Vanilla Active Learning

Our active learning scheme (eq. (8.3)) improves upon the vanilla score-based active learning (eq. (8.2)) by selecting only one synthetic image at most from any pair of real images. This change is necessary because two nearly duplicated synthetic images could both have very low scores according to eq. (8.2).

To quantitatively compare the two active learning methods, we run another experiment by replacing our active learning scheme with the vanilla version. The candidate pool is the same as ours, i.e., mixup images generated by varying $\lambda \in \{0.3, 0.7\}$ with an interval of 0.04. Figure 8.4 shows the results on both CIFAR-10 and Place365-Standard.

Generally, the vanilla active learning yields lower accuracy than our active mixup and the random search. This shows that the vanilla score-based active learning even fails to improve upon random search because it selects nearly duplicated synthetic images to query the teacher model. In contrast, our active mixup consistently performs the better than the vanilla active learning and random search. The prominent gap justifies that the constraint by $C_2$ in eq. (8.3) is crucial in our approach.
In real-world applications, it may be hard to collect real training images for some tasks, e.g., due to privacy concerns. Under such scenarios, we have to use out-of-domain data to distill the student neural network. Hence, we further challenge our approach by revealing some images that are out of the domain of the training images of the blackbox teacher model.

We conduct this experiment on CIFAR-10 by providing our approach some training images in CIFAR-100 [121]. To reduce information leak, we exclude the images that belong to the CIFAR-10 classes and keep 2K images to construct the candidate pool. Equipped with these synthetic images, we run active mixup to distill student neural networks from a blackbox teacher model for CIFAR-10. The teacher model is VGG-16, which yields 93.31% accuracy on the CIFAR-10 test set.

Table 8.4: CIFAR-10 classification accuracy by the student neural networks which are distilled by using out-of-domain data.

<table>
<thead>
<tr>
<th>Selected Syn.</th>
<th>10K</th>
<th>20K</th>
<th>40K</th>
<th>80K</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy (%)</td>
<td>64.10</td>
<td>71.39</td>
<td>77.89</td>
<td>83.03</td>
</tr>
</tbody>
</table>

Table 8.4 shows the results of different numbers of selected synthetic images. We still run only one iteration of the active learning to save computation costs. The best distillation performance is 83% top-1 accuracy and success rate is 88.9%. Comparing the result to Table 8.2, especially the entry (87.89%) of 80K selected synthetic images and 2K real images, we can see that our approach leads to about the same performance by using the out-of-domain data as the in-domain data.

To better understand how different factors influence the distillation performance, we also decouple the number of available real images from the number of selected synthetic images in Table 8.5. We
Table 8.5: CIFAR-10 classification accuracy by the student neural networks which are distilled by using out-of-domain data. We set the number of selected synthetic images to 40K and vary the numbers of real images.

<table>
<thead>
<tr>
<th>Real images</th>
<th>500</th>
<th>1000</th>
<th>1500</th>
<th>2000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy (%)</td>
<td>70.21</td>
<td>74.60</td>
<td>75.54</td>
<td>77.89</td>
</tr>
</tbody>
</table>

fix the number of selected synthetic images to 40K and vary the numbers of real images. Not surprisingly, the more real images there are, the higher distillation accuracy the active mixup achieves. Furthermore, the number of synthetic images still plays a prominent role in distillation accuracy, according to Table 8.4. Without the original training data, mixup augmentation is probably more critical to enhancing the distillation performance than otherwise.

8.4 Summary

In this chapter, we formalize a novel problem, knowledge distillation from a blackbox teacher model in a data-efficient manner, which we think is more realistic than previous knowledge distillation setups. There are two key challenges to this problem. One is that the available examples are insufficient to represent the vast variation in the original training set of the teacher model. The other is that the blackbox teacher model often implies that it is financially and computationally expensive to query.

To deal with the two challenges, we propose an approach combining mixup and active learning. Although neither of them is new by itself, combining them is probably the most organic solution to our problem setup for the following reasons. First of all, we would like to augment the few available examples. Unlike conventional data augmentations (e.g., cropping, adding noise), which only probe the regions around the available examples, mixup provides a continuous interpolation.
between any pairwise examples. As a result, mixup allows the student model to probe diverse regions of the input space. We then employ active learning to reduce the query transactions to the teacher model. Extensive experiments verify the effectiveness of our approach to the data-efficient blackbox knowledge distillation.
CHAPTER 9: CONCLUSION

9.1 Overall Summary

With the rapid development of machine learning algorithms and the explosive growing visual data, deep neural networks have shown great promise in many computer vision tasks. In this thesis, we focus on two aspects of current deep learning based algorithms: accuracy and robustness. On the one hand, we strive to push the new state-of-the-art in specific vision tasks such as video summarization (cf. Chapter 3), object detection on “less eye-catching” functional objects (cf. Chapter 4), image recognition on downstream tasks (cf. Chapter 5), and high-resolution image generation from object layout (cf. Chapter 6). On the other hand, we are dedicated to exploring the potential threats of deep learning. We propose a powerful blackbox attack that can universally defeat both vanilla DNNs and defended ones, which provides a strong opponent for new defense algorithms (cf. Chapter 7). Besides, we study a new threat model (cf. Chapter 8) in the context of blackbox knowledge distillation – the adversaries can efficiently distill knowledge from a blackbox teacher model (e.g., deep neural networks deployed as web API).

9.2 Future Work

9.2.1 Large-scale Unsupervised/Weakly Supervised Pre-training

Self-supervised pre-training has achieved remarkable success for natural language process. While fully-supervised pre-training still dominates the computer vision field. Recent work illustrates that self-supervised pre-training can obtain better feature representation for some specific downstream tasks (e.g., object detection) than its fully-supervised counterpart, which sheds light on the future
research directions. Specifically, contrastive learning based approaches show significant improvement over pretext task based methods, and some traditional approaches like self-training step onto the stage of history.

Efficient self-supervised learning on a large number of unlabeled Web images or videos will be a promising direction. Web images or videos usually tend to be diverse, noisy, and out of the distribution of the curated datasets. However, they supply a wide variety of object poses, appearances, their interactions with the context. It is challenging to remedy the negative effect of ‘domain shift’ and learn informative representation from the noisy Web images. My work on selective self-supervised learning for object detection [143] serves as my starting point to tackle those problems.

9.2.2 Defense on Large-scale Datasets

Exploring the robustness of DNNs is a fertile area of research. There is a rich line of work focusing on defending against adversarial attacks. However, DNNs’ vulnerability is just like a natural property – defenses can not make much progress while attacks are so powerful that they can achieve nearly 100% success rate even on some defended DNNs, which have shown robustness in a specific territory. Recent explorations demonstrate that adversarial training (adv-training) is the most promising approach to improve DNNs’ robustness. However, adv-training is extremely slow to apply on large-scale datasets – Projected Gradient Descent (PGD) based adv-training is approximately 20 times slower than normal training. Although some approaches try to accelerate adv-training, the robustness is still not desirable, especially on some large-scale datasets like ImageNet. It raises the pressing need to develop more effective and efficient defending approaches on more challenging and realistic benchmarks.

There are some remaining questions about defenses. For instance, Is PGD based attack the proper way to generate adv examples for adv-training? Do we have to use adv-training? How do DNNs
make decisions for too complex problems? What makes DNNs so different from humans’ sensory-perception? We will strive to tackle those problems through intensive empirical study and rigorous theoretical analysis, and we believe this is the key to improve the robustness of DNNs.
LIST OF REFERENCES


\[ \text{C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In } \text{Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2818–2826, 2016.} \]


\[ \text{P. Tang, X. Wang, A. Wang, Y. Yan, W. Liu, J. Huang, and A. Yuille. Weakly supervised region proposal network and object detection. In } \text{Proceedings of the European conference on computer vision (ECCV), pages 352–368, 2018.} \]


