

Towards Data-Efficient Neural Networks

Chaitanya Devaguptapu

A Thesis Submitted to
Indian Institute of Technology Hyderabad
In Partial Fulfillment of the Requirements for
The Degree of Master of Technology



Department of Computer Science and Engineering

June 20, 2022

© 2022 by Chaitanya Devaguptapu

All rights reserved

Declaration

I declare that this written submission represents my ideas in my own words, and where ideas or words of others have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the Institute and can also evoke penal action from the sources that have thus not been properly cited, or from whom proper permission has not been taken when needed.

Chaitanya

(Signature)

Chaitanya Devaguptapu

(Name)

CS19MTECH11025

(Roll No.)

Approval Sheet

This Thesis entitled **Towards Data-Efficient Neural Networks** by Chaitanya Devaguptapu is approved for the degree of Master of Technology from IIT Hyderabad.

C. Krishna Mohan

(Dr. C Krishna Mohan) Examiner
Dept. of Computer Science and Engineering
Indian Institute of Technology Hyderabad

Srijith

(Dr. Srijith P.K) Examiner
Dept. of Computer Science and Engineering
Indian Institute of Technology Hyderabad

N.B. Vineeth

(Dr. Vineeth N Balasubramanian) Adviser
Dept. of Computer Science and Engineering
Indian Institute of Technology Hyderabad

ch sobhan babu

(Dr. Sobhan Babu) Chairman
Dept. of Computer Science and Engineering
Indian Institute of Technology Hyderabad

Acknowledgements

Firstly, I thank my supervisor Dr. Vineeth N Balasubramanian for believing in me that I am capable of doing research. He's provided great support during my M.Tech and I couldn't thank him enough for all the things that I have learnt from him, both academically and non-academically. I would also take this opportunity to thank all my friends in IITH without whom this journey would have been really boring. Furthermore, I would like to thank Dr Animesh Garg from the PAIR Lab, University of Toronto, for their continuous support and valuable guidance. I also would love to thank all the faculty members of our department who taught us valuable things and always supported us. Thank you everyone. This has been quite a fantastic journey that I will remember forever.

Dedication

To my parents for their support in my decision to pursue higher studies

Abstract

Deep learning systems have seen wide adoption in the last couple of years for solving various large-scale, real-world problems. These systems have succeeded in solving various long-standing difficult problems like protein folding (AlphaFold), automatic image and text generation (GPT-3, DALLE-2, ImageGen), etc. While these results are impressive, most of the current deep learning systems require vast amounts of data and compute to give a reasonably good performance, and they are also very brittle and suffer from various problems like algorithmic fairness and vulnerability to adversarial examples. Considering the scale at which these systems are being adopted for various real-world applications, it is not only important but essential to make them more efficient in terms of the amount of data and compute that is required to train, along with making them robust to common attacks like adversarial attacks. This thesis provides a multi-facet view of efficiency and paves a way to make the current deep learning systems more data-efficient.

Contents

Declaration	ii
Approval Sheet	iii
Acknowledgements	iv
Abstract	vi
List of Tables	x
List of Figures	xii
List of Algorithms	xv
List of Publications	1
1 Introduction and Background	2
1.1 The Need for Data-Efficient Neural Networks	2
1.2 A Multi-Faceted view of Efficiency	2
1.2.1 Efficiency: Modality Perspective	3
1.2.2 Efficiency: Data Annotation Perspective	3
1.2.3 Efficiency: Architecture and Robustness Perspective	4
1.2.4 Efficiency: Transfer Learning Perspective	5
2 Improving Object Detection in Thermal imagery	6
2.1 Borrowing from Rich Domains like RGB	6
2.2 Object Detection methods for Thermal Imagery	6
2.3 Multi-Modal Thermal Object Detector (MMTOD)	8
2.4 Experiments and Analysis	11
2.4.1 Datasets and Experimental Setup	11
2.4.2 Results	14
2.5 Discussion and Ablation Studies	17
2.6 Conclusion	20
3 On Initial Pools for Deep Active Learning	21
3.1 Understanding the role of Initial Pool in Deep AL	21

3.2	Related Work	22
3.3	Methods and Experimental Protocol	24
3.3.1	Pool-based Active Learning Setting	24
3.3.2	Proposed Initial Pool Sampling Strategies	24
3.3.3	Active Learning Query Methods	27
3.4	Implementation Details	27
3.4.1	Additional Experiments	28
3.5	Experimental Results	28
3.5.1	Initial Pool Sampling Details	29
3.5.2	Results	30
3.6	More Training Details	36
3.6.1	Slightly Modified ResNet18 Model	36
3.6.2	Hyperparameters for AL Training	36
3.6.3	SimCLR, SCAN and VAE Training	36
3.7	Conclusion	37
4	On Adversarial Robustness: A Neural Architecture Search perspective	38
4.1	Understanding Adversarial Robustness from an Architecture Perspective	38
4.2	Adversarial Robustness and Neural Architecture Search	40
4.3	Robustness of NAS models: A Study	42
4.4	Analysis and Results	44
4.4.1	How Robust is existing SoTA Image Classification Architecture without any form of Adversarial Training?	45
4.4.2	How do NAS based models compare with Hand-crafted models in terms of Architectural Robustness?	46
4.4.3	Does an increase in the number of parameters of Architecture help improve Robustness?	49
4.4.4	Where does the source of adversarial vulnerability lie for NAS? Is it in the search space or in the way the current methods are performing the search?	51
4.5	Conclusion	53
5	On the Use of Skip Connections for Transfer Learning	54
5.1	Need for Input-Adaptive Skip Connections	54
5.2	Transfer Learning and Input-Conditioned Architectures	56
5.3	AdaSkips for Transfer Learning	58

5.4	Experiments and Results	62
5.5	Analysis and Ablation Studies	64
5.5.1	Does Routing alone help?	64
5.5.2	Flop and Parameter Count Statistics	66
5.5.3	Fine-tuning in Limited Data settings	67
5.5.4	Study of different components involved in AdaSkips	68
5.5.5	AdaSkips for Standard Image Classification	70
5.6	Conclusion	70
6	Conclusion and Future Directions	71
References		73

List of Tables

2.1	Performance comparison of proposed methodology against baseline on FLIR [1]	15
2.2	Performance comparison of proposed methodology against baseline on KAIST [2]	16
2.3	Statistics of the datasets we used for our experiments.	18
2.4	Performance comparison of proposed methodology against baseline on FLIR (1/2)	18
2.5	Performance comparison of proposed methodology against baseline on FLIR (1/4)	19
2.6	Performance comparison of proposed methodology against baseline on FLIR 400×400 images	19
3.1	Hyper-parameters of AL Cycles	36
3.2	Final Model Performances after Self-Labeling (SimCLR + SCAN + Self-Label)	37
4.1	EfficientNet Architecture comparison on ImageNet dataset with fixed image size of 224×224 (Top-5 Accuracy)	45
4.2	Comparison of clean accuracy and adversarial robustness on CIFAR-10 dataset (Top-1 Accuracy)	48
4.3	Comparison of clean accuracy and adversarial robustness on CIFAR-100 dataset (Top-1 Accuracy)	48
4.4	Comparison of clean accuracy and adversarial robustness on ImageNet dataset (Top-5 Accuracy)	48
4.5	Comparison of clean accuracy and adversarial robustness on Flowers-102 dataset (Top-1 Accuracy)	49
4.6	Comparison of parameter count vs Adversarial accuracy for five different family of architectures on ImageNet dataset	50

4.7	Adversarial accuracy comparison of DARTS based architectures on CIFAR-10 dataset	52
5.1	Comparison of performance of AdaSkips with existing methods on standard TL benchmark [3], including in low-data regimes with limited target data. (E.g, 50% denotes availability of 50% of target dataset for finetuning). Note that AdaSkips can not only outperform existing TL approaches, but can also improve performance of existing TL approaches (see row for Bi-Tuning + AdaSkips)	63
5.2	AdaSkips improves performance of existing ZSDA methods. Comparison of introducing AdaSkips for Zero-Shot domain adaptation using a base ResNet-18 network	64
5.3	Comparison of different architectures on five fine-grained classification datasets. AdaSkips improve performance significantly when the semantic relationship between source and target datasets is minimal	66
5.4	Qualitative comparison of Parameters and Flops for standard models and models with AdaSkips on FGVC-Aircraft	67
5.5	HyperNetwork and Batchnorm both play a key role in improving the performance. Quantitative ablation analysis of marginal contribution of each component in AdaSkips.	69
5.6	AdaSkips can help improve performance on common image classification benchmark datasets without pre-training. Comparison of different architectures when trained from scratch without any pre-training on standard image classification datasets	70

List of Figures

2.1	Adaptation of proposed Mutli-modal framework for Faster-RCNN	9
2.2	<i>Row 1 & Row 2:</i> Example images from FLIR [1] ADAS dataset, <i>Row 3:</i> Example Images from KAIST [2] dataset	12
2.3	<i>Row 1:</i> Thermal images from FLIR ADAS[1] dataset; <i>Row 2:</i> Translations generated using UNIT[4]; <i>Row 3:</i> Translations generated using CycleGAN[5].	15
2.4	Qualitative results of detections on the FLIR ADAS dataset. <i>Row 1:</i> Baseline <i>Row 2:</i> MMTOD	16
2.5	Qualitative results of detections on the KAIST. <i>Row 1:</i> Baseline. <i>Row 2:</i> MMTOD	17
2.6	Qualitative comparison: <i>Left:</i> Detection with single mode Faster-RCNN; <i>Middle:</i> Detection using the proposed method; <i>Right:</i> Annotated Ground Truth as provided in FLIR dataset [1].	17
2.7	Some examples of missed detections, <i>Red:</i> Predictions using MMTOD, <i>Green:</i> Ground Truth	20
3.1	Illustration of query strategies in traditional pool-based AL: (a) A toy dataset of 500 instances, evenly sampled from two class Gaussians; (b) Decision boundary of a logistic regression model trained on the dataset; (c) Trained model’s <i>CrossEntropy</i> loss on training instances; Decision boundary of logistic regression models trained on 35 instances chosen (d) <i>randomly</i> (e) using Least Confidence method [6] (f) using Max-Entropy method [7]; Best viewed in color. Labeled instances are emphasized for clarity.	25

3.2	Performance of each active learning query method with different initial pool sampling strategies on CIFAR-10. There are 8 graphs shown, one for each active querying method as mentioned in the graph titles. Each colored line in the graph corresponds to an initial pool sampling method, as shown in the legend.	29
3.3	AL Performance of each active learning query method with different initial pool sampling strategy on CIFAR-100.	30
3.4	Tiny ImageNet: Our initial pools perform no better than random initial pools across all AL configurations.	31
3.5	MNIST.	33
3.6	CIFAR-100: Class distribution of initial pools picked by various methods. Note the apparent class imbalance in the initial pool picked by VAE. Is this the reason for the performance gain?	33
3.7	CIFAR-10: Initial pools visualized using t-SNE.	34
3.8	CIFAR-10: In low budget AL setting, only VAE initial pools show marginal performance gains over random initial pools.	35
3.9	Long-Tail CIFAR-10: Our unsupervised sampling methods (SCAN and K-Means), motivated by this imbalance setting, did not improve LC, ME, DBAL query method performances. In the long run, VAE-based initial pools show marginal performance gains over random initial pools.	35
4.1	Comparison of test-set accuracy and PGD accuracy of NAS and hand-crafted architectures on CIFAR-10 dataset. Bubble size represents the number of parameters	39
4.2	<i>Left:</i> Standard procedure for building architectures from DARTS search space; <i>Right:</i> Procedure for building ensembles using DARTS search space. 12, 6, 2 can be replaced with any values that sum to 20.	43
4.3	Comparison of robustness and clean accuracy of different architectures; As the difficult of the task or the scale of the dataset increases hand-crafted architectures are more robust; (best performance is indicated by diamond symbol)	47
4.4	Comparison of PGD accuracy and Parameter count across different family of architectures	50

5.1	Illustration of AdaSkips for two images of same class from Stanford-Dogs [8] dataset. AdaSkips determine which skip connections to use and assigns a weight for each pre-trained feature map via skip connections.	55
5.2	Models with AdaSkips start better and improve with training; Qualitative comparison of validation accuracy of WideResNet-50 on datasets that have minimal class overlap with ImageNet	66
5.3	Models with AdaSkips are better even when the labelled data is limited. Qualitative visualisation to show the effectiveness of AdaSkips with varying amounts of labelled data	68
5.4	Despite not imposing a specific sparsity schema, the hyper-network learns to select input-conditioned skip connections. Qualitative visualisation of the weights (average) learned by the hypernetwork for all the test samples of FGVC-Aircraft and Stanford Cars datasets, we use ResNeXt-50 as the base network.	69

List of Algorithms

List of Publications

1. **Chaitanya Devaguptapu**, Ninad Akolekar, Manuj Sharma, V. Balasubramanian. "Borrow from Anywhere: Pseudo Multi-modal Object Detection in Thermal Imagery," The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, June 2019.
2. **Chaitanya Devaguptapu**, Ninad Akolekar, Manuj Sharma, V. Balasubramanian., A Methodology for Transfer of Knowledge from Data-Rich Domains for Thermal Image Processing, Indian Patent Application No. 202011032663 (filed in Aug 2020)
3. Akshay Chandra L*, Sai Vikas Desai*, **Chaitanya Devaguptapu***, V. Balasubramanian. "On Initial Pools for Deep Active Learning", Preregistration Workshop at NeurIPS 2020 (PMLR Volume 148) (* – denotes equal contribution)
4. **Chaitanya Devaguptapu**, Devansh Agarwal, Gaurav Mittal, Pulkit Gopalani, V. Balasubramanian; Balasubramanian, On Adversarial Robustness: A Neural Architecture Search Perspective, Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2021
5. **Chaitanya Devaguptapu**, Samarth Sinha, V. Balasubramanian, Animesh Garg, On the Use of Skip Connections for Transfer Learning (under review at ECCV-2022)

Chapter 1

Introduction and Background

1.1 The Need for Data-Efficient Neural Networks

Deep Neural networks, often abbreviated as DNNs, have helped achieve state-of-the-art (SOTA) performance on many standard benchmark tasks in computer vision [9], natural language processing [10], and speech recognition [11]. With the advent of Deep Convolutional Neural Networks (CNNs), [12], the performance of various computer vision tasks such as image classification, object detection, and object segmentation has significantly increased. DNNs are good at capturing patterns that solve complex problems and have a wide range of interesting applications like autonomous driving, medical diagnosis, precision agriculture, and robotic vision, to name a few. However, these models require large-scale, well-annotated datasets along with a reasonable amount of compute to perform reasonably well. This work presents the directions we have explored to make DNNs more data-efficient. Our work not only focuses on decreasing the amount of data and compute that is required to train DNNs but also focuses on making large DNNs more robust and efficient.

1.2 A Multi-Faceted view of Efficiency

In the following sections, we present a multi-faceted view of efficiency. We view data efficiency from perspectives of modality, data annotation, adversarial robustness, and ease of transfer.

1.2.1 Efficiency: Modality Perspective

Object Detection in Thermal Imagery: As indicated by the recent fatalities [13], the current sensors in self-driving vehicles with level 2 and level 3 autonomy (lacking thermal imaging) do not adequately detect vehicles and pedestrians. Pedestrians are especially at risk after dark when 75% of the 5,987 U.S. pedestrian fatalities occurred in 2016 [14]. Thermal sensors perform well in such conditions where autonomy level 2 and level 3 sensor-suite technologies are challenged. As is well-known, thermal IR cameras are relatively more robust to illumination changes and can thus be useful for deployment both during the day and night. In addition, they are low-cost, non-intrusive, and small in size. Consequently, thermal IR cameras have become increasingly popular in applications such as autonomous driving recently, as well as in other mainstream applications such as security and military surveillance operations. Detection and classification of objects in thermal imagery is thus an important problem to be addressed and invested in to achieve successes that can be translated to the deployment of such models in real-world environments.

Although object detection has always remained an important problem in computer vision, most of the efforts have focused on detecting humans and objects in standard RGB imagery. Object detection performance in the RGB domain has been significantly improved using region-based methods, such as the R-CNN [15], Fast R-CNN [16], and Faster R-CNN [17], and regression-based methods such as YOLO [18]. These object detection methods rely, however, on architectures and models that have been trained on large-scale RGB datasets such as ImageNet [19]†, PASCAL-VOC[20], and MS-COCO[21]. A relative dearth of such publicly available large-scale datasets in the thermal domain restricts the achievement of an equivalent level of success of such frameworks on thermal images.

1.2.2 Efficiency: Data Annotation Perspective

With the success of convolutional neural networks (CNNs) in supervised learning on a wide range of tasks, several large and high-quality datasets have been developed. However, data annotation remains a key bottleneck for deep learning practitioners. Depending on the task, data annotation cost may vary from a few seconds to a few hours per sample and, in many real-world scenarios, supervision of domain experts is necessary ([22]).

Active Learning: Active learning (AL) methods aim to alleviate the data annotation bottleneck by labeling only a subset of the most informative samples from a large pool of unlabeled data. Various query methods ([23, 24, 25, 26, 27, 28]) have been recently proposed for AL in the context DNN models. Since DNN models require large amounts of labeled data to learn, Deep AL is a key and important area of research. AL has been well-explored in the context of traditional (shallow) machine learning methods ([29]). Generally, before starting the AL cycles, a small randomly chosen subset of a dataset (with size around 1-10% of the entire dataset), typically called the *initial pool*, is labeled first to train an initial model. after which various sampling strategies are used to pick which samples to label from the unlabelled pool of data, then the samples are sent to an oracle for acquiring the labels and the model is again trained with the entire pool of labeled samples. The cycle of *sample-label-train* is repeated multiple times depending on the annotation budget.

1.2.3 Efficiency: Architecture and Robustness Perspective

The choice of neural network architecture and its complex topology plays a crucial role in improving the performance of several deep learning based applications. However, in most of the cases, these architectures are typically designed by experts in an ad-hoc, trial-and-error fashion. Neural Architecture Search (NAS) automates the design of neural network architectures for a given task.

Neural Architecture Search: Neural Architecture Search (NAS) [30] alleviates the pain of hand designing these architectures by partially automating the process of finding the right topology that can result in best-performing architectures. Since the work by [30], there has been much interest in this space. Many researchers have come up with unique approaches [31, 32, 33] to improve the performance besides decreasing the computational cost. Current SoTA(state-of-the-art) on image classification and object detection [34, 35], are developed using NAS, which shows how important a role NAS plays in solving standard learning tasks, especially in computer vision.

Adversarial Examples and Robustness: Adversarial examples, in general, refer to samples that are imperceptible to the human eye but can fool a deep classifier to predict a non-true class with high confidence. Adversarial examples can result in degraded performance even in the presence of perturbations too subtle to be perceived by humans. Adversarial robustness is defined as the accuracy of a model when adversarial examples (images perturbed with some imperceptible noise) are provided

as input. Adversarial examples have the potential to be dangerous. [36] discusses an example where attackers could target autonomous vehicles by using stickers or paint to create an adversarial stop sign that the vehicle could interpret as a yield or other sign.

1.2.4 Efficiency: Transfer Learning Perspective

Deep neural network models are trained typically on large-scale datasets which are annotated for a specific task at hand. However, all real-world application settings do not have similar scales of annotated data available and hence need mechanisms for efficient feature transfer and reuse.

Transfer Learning: Transfer learning (TL) provides a solution for training deep neural networks on datasets that are relatively much smaller than the dataset on which the models were pre-trained. While TL is widely adopted across several real-world applications and domains where it is difficult to construct large-scale well-annotated datasets, it is often unclear how to best transfer the knowledge from the source dataset to a target dataset.

Skip Connections: Skip connections introduced in ResNets [37] and Highway Networks [38] allow a path to pass information between intermediate layers of a neural network and help in avoiding problems such as vanishing gradients by turning the model into an implicit ensemble with different paths [39].

Chapter 2

Improving Object Detection in Thermal imagery

2.1 Borrowing from Rich Domains like RGB

This chapter presents a *pseudo multi-modal* framework we propose for object detection in the thermal domain. The proposed framework helps borrow complex high-level features from the RGB domain to improve object detection in the thermal domain. It does not need paired examples from two modalities and can borrow from any large-scale RGB dataset available for object detection and does not need the collection of a synchronized multi-modal dataset. This setting makes this problem challenging and more useful for an application perspective. The proposed framework also overcomes the problem of inadequacy of training examples in the thermal domain. At a high-level, the proposed framework consists of two branches, one branch is pre-trained on large-scale RGB datasets (such as PASCAL-VOC or MS-COCO) and finetuned using a visual RGB input that is obtained using an image-to-image (I2I) translation framework from a given thermal image (and hence the name ‘pseudo multi-modal’). The second branch follows the standard training process on a relatively smaller thermal dataset.

2.2 Object Detection methods for Thermal Imagery

Detection and classification of objects in the thermal imagery has been an active area of research in computer vision [40, 41, 42, 2], especially in the context of military and surveillance[43]. There has been a significant amount of work on classifying and

detecting people and objects in thermal imagery using standard computer vision and machine learning models, even before deep learning became popular. Bertozzi *et al.* [44] proposed a probabilistic template-based approach for pedestrian detection in far infrared (IR) images. They divided their algorithm into three parts: candidate generation, candidate filtering and validation of candidates. One main weakness of this approach is that it assumes the human is hotter than the background which may not be the case in many real-world scenarios. Davis *et al.* [45] proposed a two-stage template-based method to detect people in widely varying thermal imagery. To locate the potential person locations, a fast screening procedure is used with a generalized template and then an AdaBoost ensemble classifier is used to test the hypothesized person locations. Kai *et al.* [46] proposed a local feature-based pedestrian detector on thermal data. They used a combination of multiple cues to find interest points in the images and used SURF [47] as features to describe these points. A codebook is then constructed to locate the object center. The challenge of this detector is whether a high performance can be achieved when local features are not obvious.

While these efforts have shown good performance for IR image classification and detection tasks over a small number of objects, they have been outperformed in recent years by deep learning models that enable more descriptive features to be learned. With the increase in popularity of deep neural networks, several methods have been proposed for applying deep learning methods to thermal images. Peng *et al.* [48] proposed a Convolutional Neural Network (CNN) for face identification in near IR images. Their CNN is a modification of GoogLeNet but has a more compact structure. Lee *et al.* [49] designed a lightweight CNN consisting of two convolutional layers and two subsampling layers for recognizing unsafe behaviors of pedestrians using thermal images captured from moving vehicles at night. They combined their lightweight CNN with a boosted random forest classifier. Chevalier *et al.* [50] proposed LR-CNN for automatic target recognition which is a deep architecture designed for classification of low-resolution images with strong semantic content. Rodger *et al.* [51] developed a CNN trained on short-to-midrange high resolution IR images containing six object classes (person, land vehicle, helicopter, aeroplane, unmanned aerial vehicle and false alarm) using an LWIR sensor. This network was successful at classifying other short to mid-range objects in unseen images, although it struggled to generalize to long range targets. Abbott *et al.* [52] used a transfer learning approach with the YOLO [18] framework to train a network on high-resolution thermal imagery for classification of pedestrians and vehicles in low-resolution thermal images. Berg *et al.* [53][54] proposed an anomaly-based obstacle detection method using a train-mounted thermal

camera. Leykin *et al.* [55] proposed a fusion tracker and pedestrian classifier for multispectral pedestrian detection. Proposals for performing detection are generated using background subtraction and evaluated using periodic gait analysis.

Among efforts that use a multimodal approach, Wagner *et al.* [56] applied Aggregated Channel Features (ACF) and Boosted Decision trees (BDT) for proposal generation and classified these proposals with a CNN, which fuses Visual and IR information. Choi *et al.* [57] uses two separate region proposal networks for both Visual and IR images and evaluates the proposals generated by both the networks with Support Vector Regression on fused deep features. The efforts closest to our work are that of Konig *et al.* [58] and Liu *et al.* [59], both of which propose a multi-modal framework that combines RGB and thermal information in a Faster-RCNN architecture by posing it as a convolutional network fusion problem. However, all of these multimodal efforts assume the availability of a dataset with paired training examples from the visual and thermal domain. On the other hand, our work assumes only the presence of thermal imagery and seeks to leverage the use of publicly available RGB datasets (which may not be paired with the thermal dataset) to obtain significant improvement in thermal object detection performance.

2.3 Multi-Modal Thermal Object Detector (MM-TOD)

Our overall proposed methodology for ‘pseudo multi-modal’ object detection for thermal images is summarized in Figure 2.1. The key idea of our methodology is to borrow knowledge from data-rich domains such as visual (RGB) without the explicit need for a paired multimodal dataset. We achieve this objective by leveraging the success of recent image-to-image translation methods [5, 4] to automatically generate a pseudo-RGB image from a given thermal image, and then propose a multimodal Faster R-CNN architecture to achieve our objective. Image-to-Image translation models aim to learn the visual mapping between a source domain and target domain. Learning this mapping becomes challenging when there are no paired images in source and target domains. Recently, there have been noteworthy efforts on addressing this problem using unpaired images [5, 60, 61, 4, 62, 63, 64]. While one could use any unsupervised image-to-image translation framework in our overall methodology, we use CycleGAN[5] and UNIT[4] as I2I frameworks of choice in our work, owing to their wide use and popularity. We begin our discussion with the I2I translation frameworks

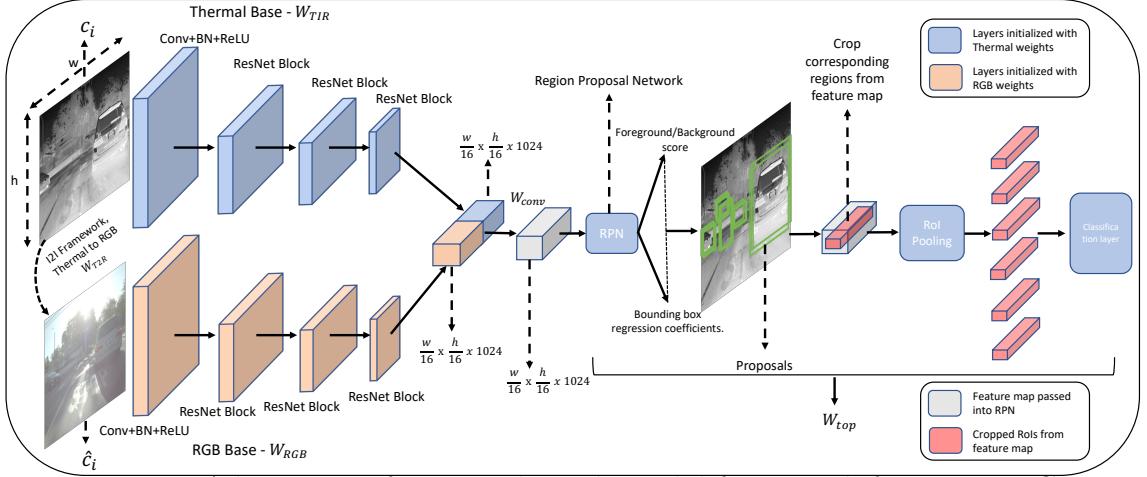


Figure 2.1: Adaptation of proposed Multi-modal framework for Faster-RCNN

used in this work.

Unpaired Image-to-Image Translation: CycleGAN [5] is a popular unpaired image-to-image translation framework that aims to learn the mapping functions $F : \mathcal{X} \rightarrow \mathcal{Y}$ and $G : \mathcal{Y} \rightarrow \mathcal{X}$ where \mathcal{X} and \mathcal{Y} are source and target domains respectively. It maps the images onto two separate latent spaces and employs two generators $\mathcal{G}_{\mathcal{X} \rightarrow \mathcal{Y}}, \mathcal{G}_{\mathcal{Y} \rightarrow \mathcal{X}}$ and two discriminators $\mathcal{D}_{\mathcal{X}}, \mathcal{D}_{\mathcal{Y}}$. The generator $\mathcal{G}_{\mathcal{X} \rightarrow \mathcal{Y}}$ attempts to generate images \hat{y}_i that look similar to images from domain \mathcal{Y} , while $\mathcal{D}_{\mathcal{Y}}$ aims to distinguish between the translated samples \hat{y}_i and real samples y_i . This condition is enforced using an adversarial loss. To reduce the space of possible mapping functions, a cycle-consistency constraint is also enforced, such that a source-domain image x_i when transformed into target domain (\hat{y}_i) and re-transformed back to source domain (\hat{x}_i) will ensure \hat{x}_i and x_i will belong to the same distribution. For more details, please see [5].

Unlike CycleGAN [5], UNIT [4] tackles the unpaired image-to-image translation problem assuming a shared latent space between both the domains. It learns the joint distribution of images in different domains using the marginal distribution in individual domains. The framework is based on variational autoencoders VAE_1, VAE_2 and generative adversarial networks GAN_1, GAN_2 with a total of six sub-networks including two image encoders $\mathcal{E}_1, \mathcal{E}_2$, two image generators $\mathcal{G}_1, \mathcal{G}_2$ and two adversarial discriminators $\mathcal{D}_1, \mathcal{D}_2$. Since they assume a shared latent space between the two domains, a weight sharing constraint is enforced to relate the two VAEs. Specifically, weight sharing is done between the last few layers of encoders $\mathcal{E}_1, \mathcal{E}_2$ that are responsible for higher level representations of the input images in the two domains

and the first few layers of the image generators $\mathcal{G}_1, \mathcal{G}_2$ responsible for decoding the high-level representations for reconstructing the input images. The learning problems of $\text{VAE}_1, \text{VAE}_2, \text{GAN}_1, \text{GAN}_2$ for image reconstruction, image translation and cyclic reconstruction are jointly solved. For more information, please see [4].

In case of both CycleGAN and UNIT, the trained model provides two generators which perform the translation between source and target domains. In our case, we use the generator which performs the Thermal-to-RGB translation, which is given by $G : \mathcal{X} \rightarrow \mathcal{Y}$ in case of a CycleGAN and G_1 in case of UNIT (we used Thermal as the source domain, and RGB as the target domain while training these models). We refer to the parameters of these generators as W_{T2R} in our methodology.

Pseudo Multi-modal Object Detection: As shown in Figure 2.1, our object detection framework is a multi-modal architecture consisting of two branches, one for the thermal image input and the other for the RGB input. Each branch is initialized with a model pre-trained on images from that domain (specific details of implementation are discussed in Section 2.4). To avoid the need for paired training examples from two modalities but yet use a multi-modal approach, we use an image-to-image (I2I) translation network in our framework. During the course of training, for every thermal image input, we generate a pseudo-RGB using W_{T2R} and pass the pseudo-RGB and Thermal to the input branches (parametrized by W_{RGB} and W_{TIR} respectively). Outputs from these branches are stacked and passed through a 1×1 convolution (W_{conv}) to learn to combine these features appropriately for the given task. The output of this 1×1 convolution is directly passed into the rest of the Faster-RCNN network (denoted by W_{top}). We use the same Region Proposal Network (RPN) loss as used in Faster-RCNN, given as follows:

$$L(\{p_i\}, \{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*)$$

where i is the index of an anchor, p_i is the predicted probability of anchor i being an object, p_i^* is the ground truth, t_i represents the coordinates of the predicted bounding box, t_i^* represents the ground truth bounding box coordinates, L is log loss, R is the robust loss function (smooth L_1) as defined in [16], and λ is a hyperparameter. We use the same multi-task classification and regression loss as used in Fast-RCNN [16] at the end of the network.

While the use of existing I2I models allow easy adoption of the proposed methodology, the images generated from such I2I frameworks for thermal-to-RGB transla-

tion are perceptually far from natural RGB domain images (like MS-COCO[21] and PASCAL-VOC [20]), as shown in Figure 2.3. Therefore, during the training phase of our multi-modal framework, in order to learn to combine the RGB and thermal features in a way that helps improve detection, we also update the weights of the I2I generator W_{T2R} . This helps learn a better representation of the pseudo-RGB image for borrowing relevant features from the RGB-domain, which we found to be key in improving detection in the thermal domain. The proposed methodology provides a fairly simple strategy to improve object detection in the thermal domain. We refer to the proposed methodology as MMTOD (Multimodal Thermal Object Detection) hereafter. Our algorithm for training is summarized in Algorithm 1. More details on the implementation of our methodology are provided in Section 2.4.

Algorithm 1: MMTOD: Multi-modal Thermal Object Detection Methodology

```

1 Input: Thermal image training data:  $\{(c_i, y_i)\}_{i=1}^m$ ; Generator of I2I
   framework:  $W_{T2R}$ ; Pre-trained RGB base network:  $W_{RGB}$ ; Pre-trained
   thermal base network:  $W_{TIR}$ , Pre-trained thermal top network  $W_{top}$ ;
   Randomly initialised 1x1 conv weights:  $W_{conv}$ ; Number of epochs:
    $num\_epochs$ ; Loss function:  $\mathcal{L}(.)$ 
2 Output: Trained MMTOD model,  $\mathcal{F}(.)$ 
3 for  $num\_epochs$  do
4   for  $c_i, i = 1, \dots, m$  do
5     Generate a pseudo-RGB image  $\hat{c}_i$  using  $W_{T2R}$ .
6     Generate feature maps by passing  $c_i$  and  $\hat{c}_i$  to base networks  $W_{TIR}$ 
      and  $W_{RGB}$  respectively
7     Stack the feature maps and use  $W_{conv}$  to get  $1 \times 1$  conv output
8     Pass the  $1 \times 1$  conv output to  $W_{top}$ 
9     Update weights:  $W_{RGB}, W_{TIR}, W_{top}, W_{conv}, W_{T2R}$  by minimizing  $\mathcal{L}$  of
      the object detection framework.
10    end
11 end

```

2.4 Experiments and Analysis

2.4.1 Datasets and Experimental Setup

Datasets: We use the FLIR ADAS [1] dataset and the KAIST Multispectral Pedestrian dataset [2] for our experimental studies. FLIR ADAS [1] consists of a total of 9,214 images with bounding box annotations, where each image is of 640×512 res-

olution and is captured using a FLIR Tau2 camera. 60% of the images are collected during daytime and the remaining 40% are collected during night. While the dataset provides both RGB and thermal domain images (not paired though), we use only the thermal images from the dataset in our experiments (as required by our method). For all the experiments, we use the training and test splits as provided in the dataset benchmark, which contains the person (22,372 instances), car (41,260 instances), and bicycle (3,986 instances) categories.

The KAIST Multispectral pedestrian benchmark dataset [2] contains around 95,000 8-bit day and night images (consisting of only the Person class). These images are collected using a FLIR A35 microbolometer LWIR camera with a resolution of 320×256 pixels. The images are then upsampled to 640×512 in the dataset. Sample images from the dataset are shown in Figure 2.2. Though the KAIST dataset comes with fully aligned RGB and Thermal, we choose not to use the RGB images as our goal to improve the detection in the absence of paired training data. Some example images from both the datasets are shown in Figure 2.2.



Figure 2.2: *Row 1 & Row 2*: Example images from FLIR [1] ADAS dataset, *Row 3*: Example Images from KAIST [2] dataset

Our methodology relies on using publicly available large-scale RGB datasets to improve thermal object detection performance. For this purpose, we use RGB datasets with the same classes as in the aforementioned thermal image datasets. In particular, we perform experiments using two popular RGB datasets namely, PASCAL VOC [20] and MS-COCO [21]. In each experiment, we pre-train an object detector on either of these datasets and use these parameters to initialise the RGB branch of our multimodal framework. We also compare the performance of these two initializations in

our experiments. In case of thermal image datasets, an end-to-end object detector is first trained on the dataset and used to initialize the thermal branch of our framework. We use mean Average Precision (mAP) as the performance metric, as is common for the object detection task.

Baseline: A Faster-RCNN trained in a fully supervised manner on the thermal images from the training set is used as the baseline method for the respective experiments in our studies. We followed the original paper [17] for all the hyperparameters, unless specified otherwise. The FLIR ADAS dataset [1] also provides a benchmark test mAP (at IoU of 0.5) of 0.58 using the more recent RefineDetect-512 [65] model. We show that we beat this benchmark using our improved multi-modal Faster-RCNN model.

Image-to-Image Translation (IR-to-RGB): For our experiments, we train two CycleGAN models: one for $\text{FLIR} \leftrightarrow \text{RGB}$ which uses thermal images from FLIR [1] and RGB images from PASCAL VOC [20], and another for $\text{KAIST} \leftrightarrow \text{RGB}$ which uses thermal images from KAIST [2] and RGB images from PASCAL VOC [20]. We use an initial learning rate of 1e-5 for the first 20 epochs, which is decayed to zero over the next 20 epochs. The identity mapping is set to zero, i.e., the identity loss and the reconstruction loss are given equal weightage. The other hyperparameters of training are as described in [5]. For training of the UNIT framework, all the hyperparameters are used as stated in the original paper, without any alterations. Since UNIT takes a long time to train (7 to 8 days on an NVIDIA P-100 GPU), we trained it only for $\text{FLIR} \leftrightarrow \text{RGB}$, so the experiments on KAIST are performed using CycleGAN only. Our variants are hence referred to as MMTOD-CG (when I2I is CycleGAN) and MMTOD-UNIT (when I2I is UNIT) in the remainder of the text.

We use the same metrics as mentioned in CycleGAN [5] and UNIT [4] papers for evaluating the quality of translation. In an attempt to improve the quality of generated images in CycleGAN [5], we tried adding feature losses in addition to cycle consistency loss and adversarial loss. However, this did not improve the thermal to visual RGB translation performance. We hence chose to finally use the same loss as mentioned in [5].

Training our Multi-modal Faster-RCNN: Our overall architecture (as in Figure 2.1) is initialized with pre-trained RGB and Thermal detectors as described in Section 2.3. Since our objective is to improve detection in thermal domain, the region

proposal network (RPN) is initialized with weights pre-trained on thermal images. The model is then trained on the same set of images on which the thermal detector was previously pre-trained. The I2I framework generates a pseudo-RGB image corresponding to the input thermal image. The thermal image and the corresponding pseudo-RGB image are passed through the branches of the multi-modal framework to obtain two feature maps of 1024 dimension each, as shown in figure 2.1. These two feature maps are stacked back-to-back and passed through a 1×1 convolution, which is then passed as input to the Region Proposal Network (RPN). RPN produces the promising Regions of Interest (RoIs) that are likely to contain a foreground object. These regions are then cropped out of the feature map and passed into a classification layer which learns to classify the objects in each ROI. Note that as mentioned in Section 2.3, during the training of the MMTOD framework, the weights of the I2I framework are also updated which allows it to learn a better representation of the translated image for improved object detection in thermal domain. We adapted the Faster-RCNN code provided at [66] for our purpose. The code for the CycleGAN and UNIT was taken from their respective official code releases[5, 67, 4]. Our code will be made publicly available for further clarifications.

Experimental Setup: To evaluate the performance of the proposed multi-modal framework, the following experiments are carried out:

- MMTOD-CG with RGB branch initialized by PASCAL-VOC pre-trained detector, thermal branch initialized by FLIR ADAS pre-trained detector
- MMTOD-UNIT with RGB branch initialized by PASCAL-VOC pre-trained detector, thermal branch initialized by FLIR ADAS pre-trained detector
- MMTOD-CG with RGB branch initialized by MS-COCO pre-trained detector, thermal branch initialized by FLIR ADAS pre-trained detector
- MMTOD-UNIT with RGB branch initialized by MS-COCO pre-trained detector, thermal branch initialized by FLIR ADAS pre-trained detector
- MMTOD-CG with RGB branch initialized by PASCAL-VOC pre-trained detector, thermal branch initialized by KAIST pre-trained detector
- MMTOD-CG with RGB branch initialized by COCO pre-trained detector, thermal branch initialized by KAIST pre-trained detector

2.4.2 Results

IR-to-RGB Translation Results: Figure 2.3 shows the results of CycleGAN and UNIT trained for Thermal \leftrightarrow RGB translation. As mentioned in Section 2.3, the gen-

erated pseudo-RGB images are perceptually far from natural domain images. This can be attributed to the fact that the domain shift between RGB and Thermal domains is relatively high compared to other domains. In addition, RGB images have both chrominance and luminance information, while thermal images just have the luminance part which makes estimating the chrominance for RGB images a difficult task. However, we show that using our method, these generated images add value to the detection methodology.



Figure 2.3: *Row 1*: Thermal images from FLIR ADAS[1] dataset; *Row 2*: Translations generated using UNIT[4]; *Row 3*: Translations generated using CycleGAN[5].

Thermal Object Detection Results: Tables 2.1 and 2.2 show the comparison of AP for each class and the mAP of our framework against the baseline detector when trained on FLIR ADAS and KAIST datasets respectively. (Note that the KAIST dataset has only one class, the Person.) We observe that in all the experiments, our framework outperforms the baseline network across all the classes.

Method	AP across each class			
	Bicycle	Person	Car	mAP
Baseline	39.66	54.69	67.57	53.97
Framework		RGB Branch		
MMTOD-UNIT	MSCOCO	49.43	64.47	70.72 61.54
	Pascal VOC	45.81	59.45	70.42 58.56
MMTOD-CG	MSCOCO	50.26	63.31	70.63 61.40
	Pascal VOC	43.96	57.51	69.85 57.11

Table 2.1: Performance comparison of proposed methodology against baseline on FLIR [1]

<i>Method</i>	<i>mAP</i>
Baseline	49.39
Framework	RGB Branch
MMTOD-CG	MS-COCO 53.56
	Pascal VOC 52.26

Table 2.2: Performance comparison of proposed methodology against baseline on KAIST [2]

In case of FLIR, we observe that initializing the RGB branch with MS-COCO obtains better results than those with PASCAL-VOC. This can be attributed to the fact that MS-COCO has more instances of car, bicycle, and person as compared to PASCAL VOC. Also, experimental results show that employing UNIT as the I2I framework achieves better performance than CycleGAN. Our framework with MS-COCO initialization and UNIT for I2I translation results in an increase in mAP by at least 7 points. In particular, as mentioned earlier, the FLIR ADAS dataset provides a benchmark test mAP (at IoU of 0.5) of 0.58 using the more recent RefineDetect-512 [65] model. Our method outperforms the benchmark despite using a relatively older object detection model such as the Faster-RCNN.



Figure 2.4: Qualitative results of detections on the FLIR ADAS dataset. *Row 1:* Baseline *Row 2:* MMTOD

As shown in Table 2.2, our improved performance on the KAIST dataset shows that although this dataset has more examples of the 'Person' category than the RGB dataset used such as PASCAL-VOC, our framework still improves upon the performance of the baseline method. This allows us to infer that the proposed framework can be used in tandem with any region-CNN based object detection method to improve the performance of object detection in thermal images. On average our frame-

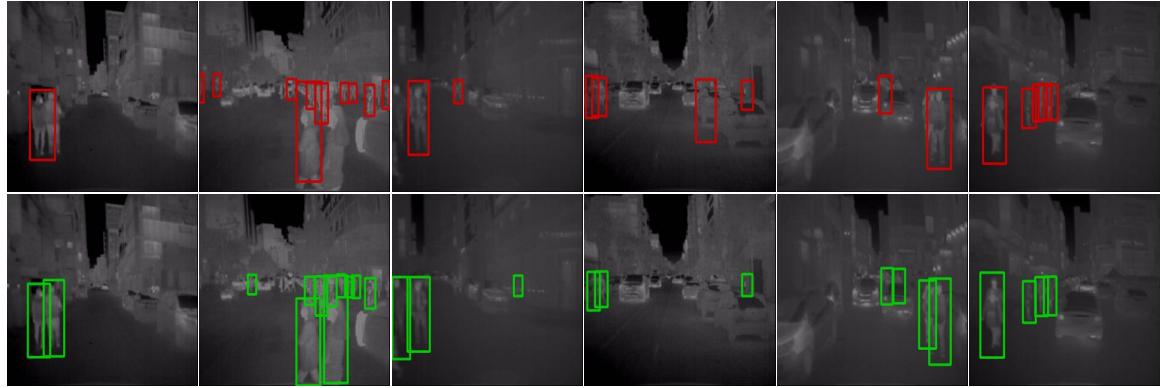


Figure 2.5: Qualitative results of detections on the KAIST. *Row 1*: Baseline. *Row 2*: MMTOD

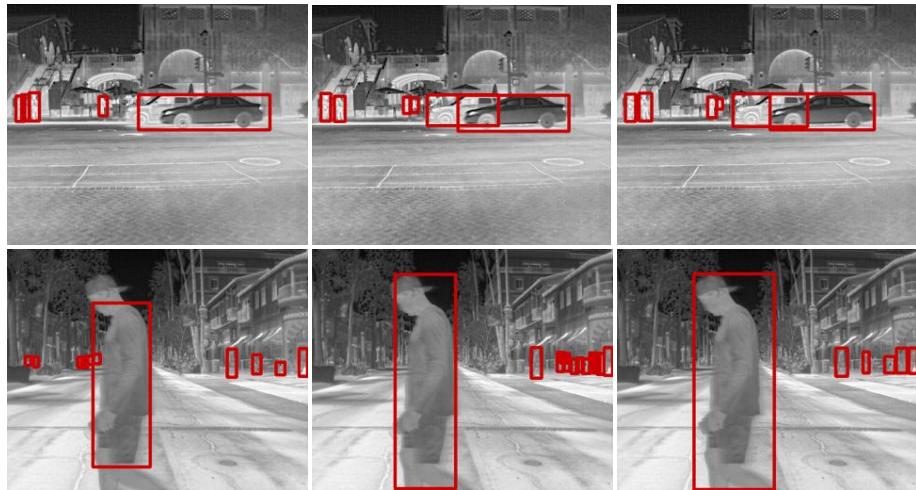


Figure 2.6: Qualitative comparison: *Left*: Detection with single mode Faster-RCNN; *Middle*: Detection using the proposed method; *Right*: Annotated Ground Truth as provided in FLIR dataset [1].

work takes 0.11s to make detections on a single image, while the baseline framework takes 0.08s. Our future directions of work include improving the efficiency of our framework while extending the methodology to other object detection pipelines such as YOLO and SSD. Figures 2.4, 2.5, 4.1 show qualitative results on FLIR and KAIST datasets.

2.5 Discussion and Ablation Studies

Learning with limited examples: We also conducted studies to understand the capability of our methodology when there are limited samples in the thermal domain.

Dataset	Number of Instances		
	Car	Person	Bicycle
FLIR	41,260	22,372	3,986
FLIR (1/2)	20,708	11,365	2,709
FLIR (1/4)	10,448	5,863	974

Table 2.3: Statistics of the datasets we used for our experiments.

Our experiments on the FLIR ADAS dataset showed that our framework outperforms the current state-of-the-art detection performance using only half the training examples. Moreover, our experiments show that using only a quarter of the training examples, our framework outperforms the baseline on the full training set. Table 2.3 presents the statistics of the dataset used for this experiment. Note that the test set used in these experiments is still the same as originally provided in the dataset.

We perform the same set of experiments (as discussed in Section 2.4) on FLIR(1/2) and FLIR(1/4) datasets. Tables 2.4 and 2.5 present the results.

Method	AP across each class			
	Bicycle	Person	Car	mAP
Baseline (FLIR)	39.66	54.69	67.57	53.97
Baseline (FLIR-1/2)	34.41	51.88	65.04	50.44
Framework	RGB Branch			
MMTOD-UNIT	MSCOCO	49.84	59.76	70.14
	Pascal VOC	45.53	57.77	69.86
MMTOD-CG	MSCOCO	50.19	58.08	69.77
	Pascal VOC	40.17	54.67	67.62
				59.35
				54.15

Table 2.4: Performance comparison of proposed methodology against baseline on FLIR (1/2)

Table 2.4 shows the baselines for training the Faster-RCNN on the complete FLIR training dataset as well as FLIR (1/2). We observe that both MMTOD-UNIT and MMTOD-CG trained on FLIR(1/2) outperform both the baselines, even when Faster-RCNN is trained on the entire training set.

Similarly, Table 2.5 shows the baselines for training the Faster-RCNN on the complete FLIR training dataset as well as FLIR (1/4). Once again, we observe that both MMTOD-UNIT and MMTOD-CG trained on FLIR(1/4) outperform both the baselines, even when Faster-RCNN is trained on the entire training set. In other words, the MMTOD framework requires only a quarter of the thermal training set to surpass the baseline accuracy achieved using the full training set. The results

Method	<i>AP across each class</i>				<i>mAP</i>
	<i>Bicycle</i>	<i>Person</i>	<i>Car</i>		
Baseline(FLIR)	39.66	54.69	67.57	53.97	
Baseline(FLIR-1/4)	33.35	49.18	60.84	47.79	
Framework	RGB Branch				
MMTOD-UNIT	MSCOCO	44.24	57.76	69.77	57.26
	Pascal VOC	35.23	54.71	67.83	52.59
MMTOD-CG	MSCOCO	41.29	57.08	69.10	55.82
	Pascal VOC	35.02	51.62	66.09	50.91

Table 2.5: Performance comparison of proposed methodology against baseline on FLIR (1/4)

clearly demonstrate the proposed framework’s ability to learn from fewer examples. This shows that our framework effectively borrows features from the RGB domain that help improve detection in the thermal domain. This is especially useful in the context of thermal and IR images, where there is a dearth of publicly available large-scale datasets.

Effect of Image Resolution: To understand the effect of image resolution on object detection performance, we repeated the above experiments were conducted using subsampled images of the FLIR ADAS dataset. Table 2.6 presents these results for 400×400 input images. We observe that our multi-modal framework improves the object detection performance significantly even in this case. Our future work will involve extending our work to images of even lower resolutions.

Dataset	Method	<i>AP across each class</i>				<i>mAP</i>
		<i>Bicycle</i>	<i>Person</i>	<i>Car</i>		
FLIR	Baseline	29.25	43.13	58.83	43.74	
	P-VOC + CycleGAN	39.42	52.75	62.05	51.41	
FLIR (1/2)	Baseline	23.31	40.82	56.25	40.13	
	P-VOC + CycleGAN	33.32	48.32	60.87	47.50	
FLIR (1/4)	Baseline	18.81	35.42	52.82	35.68	
	P-VOC + CycleGAN	30.63	45.45	60.32	45.47	

Table 2.6: Performance comparison of proposed methodology against baseline on FLIR 400×400 images

Missed Detections: We tried to analyze the failure cases of the proposed methodology, by studying the missed detections. Some examples of these missed detections are shown in figure 2.7. We infer that MMTOD finds object detection challenging when: (i) the objects are very small and located far from the camera; (ii) two objects are close to each other, and are detected as a single object; and (iii) there is heavy occlusion and crowd. Our future efforts will focus on addressing these challenges.



Figure 2.7: Some examples of missed detections, *Red*: Predictions using MMTOD, *Green*: Ground Truth

2.6 Conclusion

We propose a novel multi-modal framework to extend and improve upon any Region-CNN-based object detector in the thermal domain by borrowing features from the RGB domain, without the need of paired training examples. We evaluate the performance of our framework applied to a Faster-RCNN architecture in various settings including the FLIR ADAS and KAIST datasets. We demonstrate that our framework achieves better performance than the baseline, even when trained only on quarter of the thermal dataset. The results suggest that our framework provides a simple and straightforward strategy to improve the performance of object detection in thermal images.

Chapter 3

On Initial Pools for Deep Active Learning

3.1 Understanding the role of Initial Pool in Deep AL

Across all Active Learning efforts so far, to the best of our knowledge, the initial pool is always sampled randomly and labeled ([23, 24, 25, 26, 27, 6]). This initial pool design strategy has generally worked well for AL in traditional/shallow ML models. However, the success of AL in DNNs has not been convincing yet, especially when such models are trained on large-scale datasets. On one hand, while there have been several encouraging newly proposed deep AL methods, deeper analysis of those methods in ([68, 69, 70]) show that AL struggles to outperform random sampling baselines when slight changes are made to either datasets (class-imbalance) or training procedures (data augmentation, regularization, etc.). Interestingly, to the best of our knowledge, the design of better initial labeled pools received no attention by the deep AL community. Considering the tremendous success of self-supervised learning methods in recent years ([71, 72, 73, 74, 75, 76, 77]), we ask the question if choosing an initial labeled pool intelligently can improve AL performance.

This chapter focuses on discussing an empirical study we perform on existing deep AL methods while using initial labeled pools, sampled using methods other than random sampling. To investigate the effect of *intelligently* sampled initial labeled pools on deep AL methods, we propose two sampling techniques, leveraging state-of-the-art self-supervised learning methods and well-known clustering methods. In particular, we propose the following ways of choosing the initial pool:

- Sample datapoints that a state-of-the-art self-supervised model finds *challenging*, as observed using the trained model’s loss on the data.
- Cluster the unlabeled pool first and then perform sampling across each cluster. Equal proportions of datapoints are sampled from each cluster to make sure the chosen samples span the entire dataset.

We hypothesize that AL methods (we focus our efforts on deep AL methods) can benefit from more intelligently chosen initial pools, thus eventually reducing annotation cost in creation of datasets. Our empirical study will seek to address the following specific questions:

- Can pool-based deep AL methods leverage design of intelligently sampled initial pools to improve AL performance?
- Can we exploit latest advancements in self-supervised learning to boost deep AL performance with no additional labeling cost?
- Are some initial pools better than others? What makes an initial pool *good*?

In a realistic training setting with measures to avoid overfitting (*i.e.* regularization, batch norm, early stopping), we hypothesize that the generalization error of AL models starting with our initial pools will be lower than those of AL models starting with random initial pools, across AL cycles. However, as AL cycles increase, we expect to see shorter margins of error difference as the effect of our initial pools on the model performance could diminish with increase in labeled pool size. Studying the use of unsupervised/self-supervised learning in later epochs could be an interesting direction of future work. If initial pools do contribute to better model performances, our work could make a positive contribution to: (i) boosting AL performance with no additional annotation; (ii) developing datasets with lesser annotation cost in general; and (iii) promoting further research in the use of unsupervised learning methods for AL. On the other hand, if random initial pools perform better than our initial pools, the community will still have useful insights about this rather unexplored part of AL through this study.

3.2 Related Work

Analysis of Active Learning: In recent years, previous works have evaluated the robustness and effectiveness of deep AL methods for various tasks. [70] first reported

some obstacles of deploying AL in practice by empirically evaluating consistency of AL gains over random sampling and transferability of active samples across models. Along the same lines, [69] evaluated the performance of deep AL methods under data augmentation, low-budget regime and a label-intensive task of semantic segmentation. More recently, [68] comprehensively tested the performance variance of deep AL methods across 25 runs of experiments. They considered various settings such as regularization, noisy oracles, varying annotation and validation set size, heavy data augmentation and class imbalance. However, none of these efforts have considered varying the sampling strategy for the initial pool.

Exploiting Unlabeled Data: Our focus is on finding out if initial pools with certain *desirable* qualities can bolster AL performance. We exploit self-supervised pretext tasks to sample the initial pool more intelligently. Previous works have successfully managed to integrate unlabeled data into AL using self-supervised learning and semi-supervised learning. [78] showed that initializing the target model with the features obtained from self-supervised pretraining gives AL a kick-start in performance. Contemporaneously, [28] also used this technique in combination with a GAN based AL method and reported SOTA results on SVHN, CIFAR-10, ImageNet, CelebA datasets. This is enough evidence that exploiting self-supervised learning methods can boost AL performance, but the cited works operate in the model weight space. The importance of good initialization in weight space ([79, 80, 81]) is well understood by the deep learning community; To the best of our knowledge, there have been no efforts in understanding the importance of good initialization in data space for deep AL methods. In case of traditional AL (before deep learning’s popularity), there have been handful of encouraging works that support our hypothesis ([82, 83]). Both these works use k -means clustering to initialize the initial label pool, k -nearest neighbor algorithm for training and report better AL performance on small scale text classification tasks.

Model Loss for AL: In our work, we use a trained model’s loss to identify the most *informative* unlabeled samples. Existing AL methods largely rely on using the target model’s loss for active sampling. [84] first proposed an AL framework by calculating Expected Gradient Length (EGL) where the learner queries an unlabeled instance which, if labeled and added to the labeled pool, would result in the new training gradient of the largest magnitude. More recently, [26] proposed a loss prediction module which is attached to the target network to predict the loss value of unlabeled samples. In contrast to these methods, we strictly rely on a self-supervised model’s loss, instead of the target model, since the initial pool needs to be selected/sampled,

before any model is trained on the target data..

3.3 Methods and Experimental Protocol

In this section, we describe: (i) the notations and setting for pool-based AL cycles; (ii) our strategies for sampling the initial pool; and (iii) the AL methods that are subsequently used to build on top of the initial labeled pool. Implementation details and other considered additional experiments and ablation studies are mentioned at the end of this section.

3.3.1 Pool-based Active Learning Setting

Given a dataset \mathcal{D} , we split it into train (T_r), validation (V), and test (T_s) sets. At the beginning, the train set is also treated as an unlabeled (U) set, from which samples are moved to a labeled set (L) after every AL cycle. Pool-based AL cycles operate on a set of labeled data $L_0=\{(x_i, y_i)\}_{i=1}^{N_L}$ and a large pool of unlabeled data $U_0=\{x_i\}_{i=1}^{N_U}$, and model Φ_0 is trained on L_0 in every AL cycle. In our setting, given $L_0 = \emptyset$ to start with, a sampling function $\Psi(L_0, U_0, \Phi_0)$ parses $x_i \in U_0$, and selects k (budget size) **samples**. These samples are then labeled by an **oracle** and added to L_0 , resulting in a new, extended L_1 labeled set, which is then used to **retrain** Φ . This cycle of **sample-label-train** repeats until the sampling budget is exhausted or a satisfying performance metric is achieved. In our case, we populate L_0 using our proposed methods, discussed in Section 3.3.2. Sec 3.3.3 describes the query methods we use to perform the traditional AL cycles after this initial pool selection. We can confirm that there exists a good initial pool if the generalization error of models starting with our initial pools is lower than those of models starting with random initial pools across the AL cycles.

3.3.2 Proposed Initial Pool Sampling Strategies

We now describe our initial pool sampling strategies, which were briefly stated in Section 3.1.

Our methods are fundamentally motivated by the hypothesis that samples considered *challenging* for an unsupervised/self-supervised setting can help bootstrap AL methods through a more intelligent, well-guided choice of an initial labeled pool.

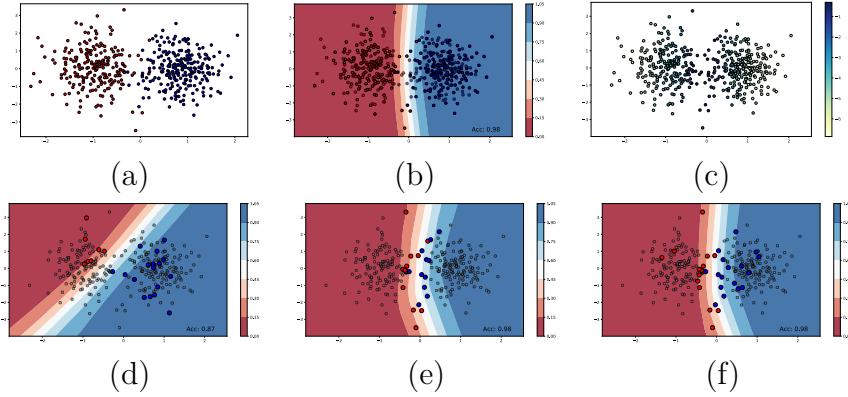


Figure 3.1: Illustration of query strategies in traditional pool-based AL: (a) A toy dataset of 500 instances, evenly sampled from two class Gaussians; (b) Decision boundary of a logistic regression model trained on the dataset; (c) Trained model’s *CrossEntropy* loss on training instances; Decision boundary of logistic regression models trained on 35 instances chosen (d) *randomly* (e) using Least Confidence method [6] (f) using Max-Entropy method [7]; Best viewed in color. Labeled instances are emphasized for clarity.

Self-Supervision Methods

As shown in Figure 3.1, well-known pool-based AL methods rely on choosing samples with a high uncertainty of the target model trained on an initial labeled pool.

Since our focus is on choosing an initial labeled pool where there is no target model, we cannot use these AL query methods since we do not have access to labels to calculate the supervised model’s loss on training data. We hence train a self-supervised model on the entire unlabeled pool and identify samples as "challenging", where the self-supervised model’s loss is relatively higher than that of others. The recent success of self-supervised learning in learning useful data representations ([77, 76, 75, 72, 73, 71]) motivates us to hypothesize that such a model could help us sample the most *informative* datapoints without any supervision.

Let τ be any self-supervised task with an objective to minimize the loss function \mathcal{L} . Let θ be trained weights obtained by solving τ on the unlabeled data pool U . We want the oracle to label and populate the initial pool L_0 with datapoints sampled by solving:

$$\arg \max_i \mathcal{L}^\tau(x_i; \theta) \quad \forall x_i \in U \quad (3.1)$$

Since we are working with a learned model’s loss, any self-supervised task can be used, making our proposed method task-agnostic. We have chosen tasks that are simple and easy to interpret, such as image inpainting ([74]) and image rotation

prediction ([72]). For example, in case of the rotation prediction task, our strategy can be summarized as: *if a trained rotation predictor struggles to rightly predict the rotation of a sample, even after looking at it while training, then it is a hard sample - thus human labeling is needed.* In addition to the above tasks, we will also train a Variational Autoencoder (VAE) ([85]) as one of our tasks where datapoints with highest loss *i.e.* hard to reconstruct images are sampled for the initial pool. We want to do this to understand how complexity of self supervised tasks (*e.g.* image inpainting task is more complex than VAEs) relates to efficiency of the sampled initial pool using those tasks.

Unsupervised Learning (Clustering) Methods

Sampling bias is the most fundamental challenge posed by AL especially in case of uncertainty based AL methods ([86]). Assume AL is performed on a dataset with data distribution \mathcal{D} . But as AL cycles proceed, and datapoints are sampled and labeled based on increasingly confident assessments of their informativeness, the labeled set starts to look less like \mathcal{D} . This problem is further exacerbated by highly imbalanced real-world datasets where random initial samples, with high probability, may not span the entire data distribution \mathcal{D} . To overcome this, several works proposed diversity based methods ([24, 27]) whose fundamental goal is to sample unlabeled datapoints from a non-sampled area of \mathcal{D} such that all areas of \mathcal{D} are seen by the target model. Motivated by these methods and their success, we propose a clustering-based sampling method for choosing the initial pool such that the sampled points spans all area of \mathcal{D} (*i.e.*, all clusters) even before AL starts. In a way, this is analogous to exploration in AL ([87]), albeit in an unsupervised way.

We assume that number of classes to be labeled (K) in the dataset \mathcal{D} is known apriori. If a clustering algorithm is applied on the unlabeled data $U=\{x_i\}_{i=1}^{N_U}$ to obtain K clusters and every datapoint x_i is assigned only one cluster C_j , we get K disjoint sample sets $C = \{C_1, C_2, \dots, C_K\}$. If the initial pool budget is B samples, we sample $\frac{B}{K}$ datapoints from each cluster. Equal weight is given to each cluster to make sure initial pool is populated with datapoints that span the entire \mathcal{D} . As another variant to this method, we will also experiment by sampling $\frac{|C_j| * B}{N_U}$ datapoints from each cluster, keeping the original cluster proportions intact. We will use DeepCluster ([73]) and k -means as clustering methods in our experiments.

3.3.3 Active Learning Query Methods

In order to study the usefulness of the choice of the initial labeled pool across AL methods, we need to study different AL query methods in later cycles of model updation. Modern pool-based AL methods may be broadly classified into three categories. We will evaluate the effectiveness of our sampling methods on AL methods from all three categories:

- **Uncertainty Sampling:** Least Confidence (LC) ([6]), Max-Entropy (ME) ([7]), Min-Margin (MM) ([88]) & Deep Bayesian AL (DBAL) ([23])
- **Diversity Sampling:** Coreset (greedy) ([24]) & Variational Adversarial AL (VAAL) ([27])
- **Query-by-Committee Sampling:** Ensemble with Variation Ratio (ENS-varR) ([25]) (3 ResNet18 models) & ensemble variants of Least Confidence (ENS-LC), Max-Entropy (ENS-ME) and Margin Sampling (ENS-MM)

All the above methods are already implemented in the AL toolkit offered by [68], and we will leverage it to study the methods.

3.4 Implementation Details

Following recent deep AL efforts, we will use MNIST, CIFAR-10, CIFAR-100 and Tiny ImageNet-200 ([89]) datasets in our experiments. We use the AL methods, model architectures, data augmentation schemes and implementation details from [68] for our experiments.

For all datasets, we plan to tune hyperparameters using grid search. However, going by previous works, we expect to use an Adam optimizer ([90]) across the datasets. For datasets CIFAR-10 and CIFAR-100, we expect to use learning rate (lr), weight decay (wd) from [68] - ($lr = 5e^{-4}$, $wd = 5e^{-4}$) and ($lr = 5e^{-4}$ and $wd = 0$) respectively. For all datasets, we augment the data with random horizontal flips ($p = 0.5$) and normalize them using statistics provided in ¹, ². We will use ResNet18 ([91]) for all our experiments.

AL Details: As usually done in most AL work, we will initialize L_0 with 10% of the unlabeled set U and in every AL cycle 10% of the original unlabeled set U will be sampled, labeled and moved to labeled set L_i . However, we expect some changes

¹<https://github.com/pytorch/examples/>

²<https://github.com/kuangliu/pytorch-cifar>

in AL cycle details due to irregularities between datasets (*e.g.* MNIST is easier to learn compared to Tiny ImageNet) and those changes will be reported appropriately post-experiments.

Performance Metrics: We will measure accuracy on the test set after every AL cycle (including after the choice of the initial labeled pool). Our initial pool sampling strategies will be compared against a random selection of the initial pool (the default option used today), and all our results will be reported (as mean and std) over 5 trials to avoid any randomness bias in the results.

We also plan to visualize the chosen initial labeled pool using t-SNE embeddings in case this provides any understanding of sampling strategies that work best. We will also examine overlap between every labeled pool acquired during all AL cycles when our initial pool sampling strategy is used against a random choice. This would allow us to know if initial pool played any role in altering the labeled pools (for better or worse).

3.4.1 Additional Experiments

In practice, populating the initial pool only with *challenging* datapoints may not be fully conducive for learning. Hence, we plan to follow [92] and split the sorted list obtained by solving Eqn (3.1) into n equal-sized bins. If the initial pool budget size is B , we query $\frac{B}{n}$ highest scored images from the top $(n - 1)$ bins (hard samples) and $\frac{B}{n}$ lowest scored images from the last bin (easy samples). So the resultant batch contains images from different regions of the score space. In the experiments, we will use 2, 5 and 10 as the values of n .

Additionally, we will test the usefulness of our sampling methods on AL for imbalanced data. For this, we will follow [93] to simulate a long-tailed distribution of classes on CIFAR-10, by following power law.

3.5 Experimental Results

In this section, we first document the modifications to the original experimental protocol. Then, we present the experimental results on MNIST, CIFAR-10, CIFAR-100 and Tiny ImageNet datasets. Then, we discuss our experimental findings and evaluate the extent to which intelligently sampled initial labeled pools help boost AL performance³. Finally, we provide more training details needed for reproducing our

³Our code is available at <https://github.com/acl21/init-pools-dal>

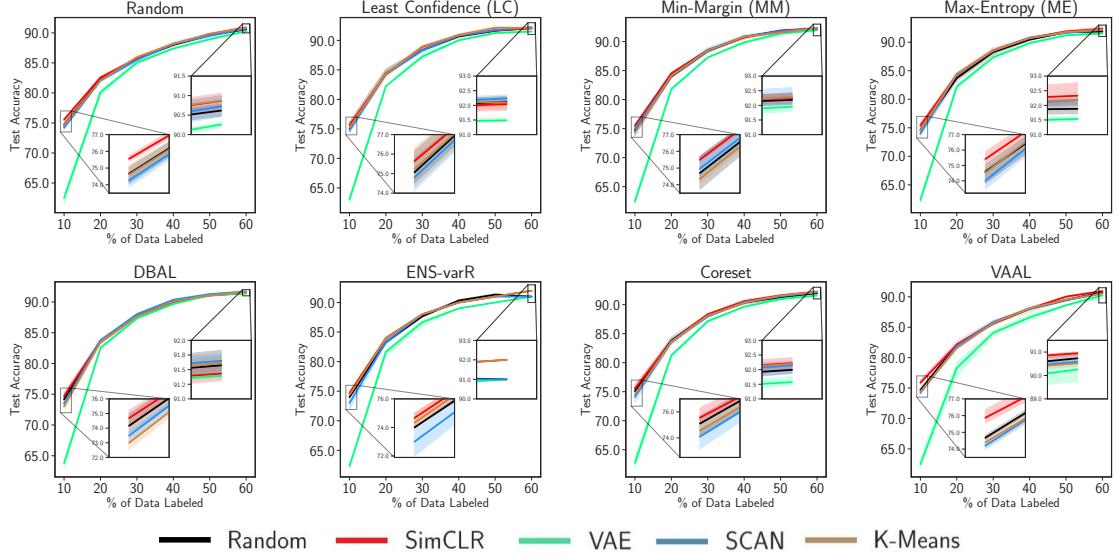


Figure 3.2: Performance of each active learning query method with different initial pool sampling strategies on CIFAR-10. There are 8 graphs shown, one for each active querying method as mentioned in the graph titles. Each colored line in the graph corresponds to an initial pool sampling method, as shown in the legend.

results.

Using grid search, we obtained hyperparameters which are better than the ones mentioned in the proposal. We report the final hyperparameter choices in section 3.6.

3.5.1 Initial Pool Sampling Details

For completeness, we briefly describe the methods used for our experiments below:

SimCLR: Contrastive learning methods, such as SimCLR ([71]), learn representations by contrasting positive pairs against negative pairs. Positive pairs include input images and their augmented variants. Ideally, a trained SimCLR model should have comparatively low contrastive loss for positive pairs of a given image taken from the unlabeled set. We design our sampling method on this fact. Firstly, we train a ResNet-18 SimCLR model with the recommended augmentations: image horizontal flipping, Gaussian blur, color jitter, and image gray-scale. After training the model, we assign each image in the unlabeled pool a score - model’s average contrastive loss between an input image and four of its augmented variants⁴. The higher the average contrastive loss, the *harder* it was for the trained SimCLR model to learn that input, so we sample such images first.

⁴MNIST dataset has gray-scale images so we average the contrastive loss over the other three augmentations.

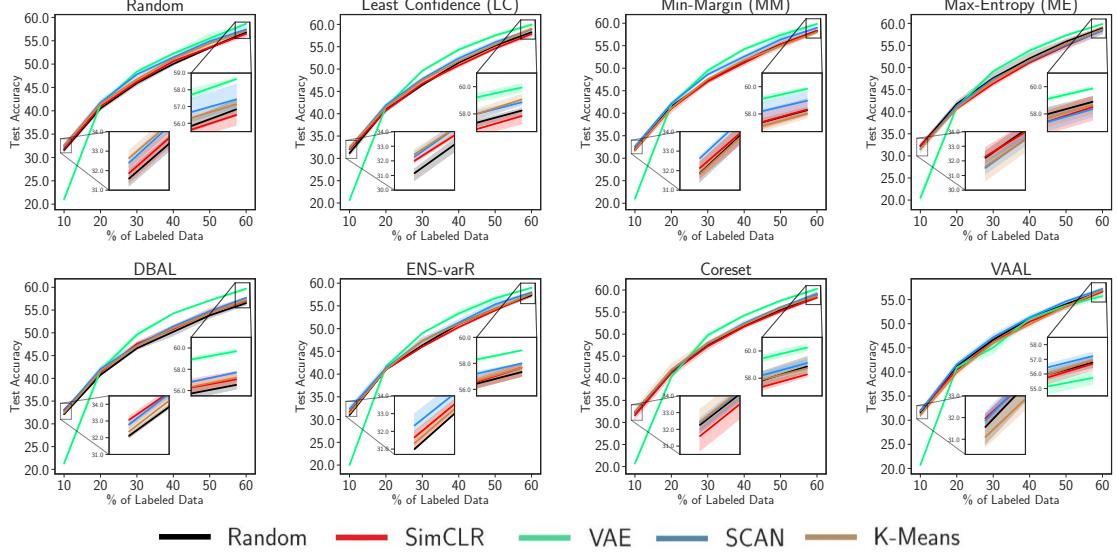


Figure 3.3: AL Performance of each active learning query method with different initial pool sampling strategy on CIFAR-100.

VAE: We train a vanilla VAE model on the entire training data till convergence. We then sample those data points from the training set whose reconstruction error was high post-training. The higher the reconstruction error, the *harder* it was for the model to learn such images, hence we sample them first. We chose VAE in particular to understand how task complexity (the VAE reconstruction task is simpler than SimCLR’s) contributes to initial pool efficiency.

SCAN and K-Means: SCAN ([94]) is a state-of-the-art clustering method where feature learning and feature clustering are decoupled. SCAN builds on top of features learned by any self-supervision task (in our case, we used SimCLR). At the end of training, the SCAN model assigns a single cluster to each data point. In case of K-Means, we apply K-Means algorithm to SimCLR-learned feature representations to get cluster assignments. Once again, we chose these two clustering methods (one very simple, K-Means, and one more sophisticated, SCAN) to understand the role of model complexity w.r.t. the effectiveness of the initial pool.

3.5.2 Results

Main Experiments

Figures 3.2-3.5 depict the main results of our experimental study on finding good initial pools for AL. We first describe how the results are illustrated in Figures 3.2-3.5. We have one figure for each of the following four datasets: CIFAR-10, CIFAR-100,

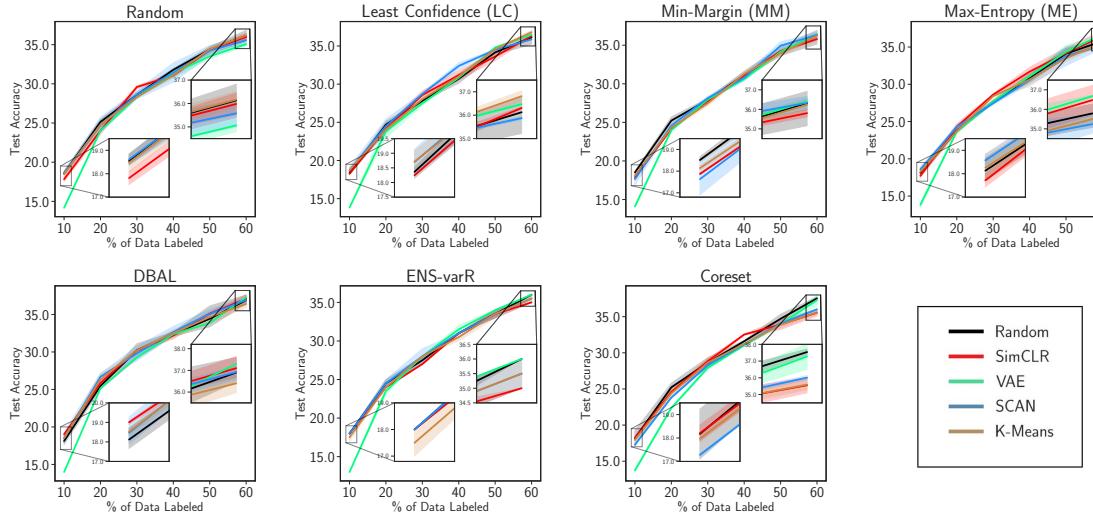


Figure 3.4: Tiny ImageNet: Our initial pools perform no better than random initial pools across all AL configurations.

Tiny ImageNet and MNIST. Each plot inside the figures depicts the performance of one AL method with various initial pool techniques. For instance, the plot titled VAAL (second row, right most) in Figure 3.2 shows the performance of models trained on data sampled by VAAL’s query method in each episode, however initiated with different initial pool strategies (indicated by different colored lines). For example, the red lines show that AL methods were started using the SimCLR-based initial pool strategy. The plots are conventional AL plots where the x -axis represents the percentage of labeled data used to train the model, and y -axis represents the model’s performance on the test set.

We now briefly analyze the results of each initial pool sampling strategy.

SimCLR: In Figure 3.2 (CIFAR-10), before the first episode, models trained with SimCLR-sampled initial pools show better performance than models trained on a randomly generated initial pool across all eight configurations, including passive learning. However, this performance gain in the beginning of the AL cycles did not contribute to the model in picking better active samples. We can see that models starting with SimCLR-based initial pools performed similar to the models which started with random initial pools at each episode of the AL cycles. Similarly, on CIFAR-100, we see in Figure 3.3 that the models starting with SimCLR sampled initial pools perform either same or worse than the models starting on random initial pools at both ends of the AL cycles across all eight configurations. On Tiny ImageNet (Figure 3.4) and MNIST (Figure 3.5), we see the same trend as that of CIFAR-100’s.

SCAN and K-Means: Across all datasets, none of the two clustering methods:

SCAN and K-Means, show signs of contributing to better model performances compared to random initial pools.

VAE: Perhaps the most surprising behaviour we noticed among all the methods was how VAE-sampled initial pools worked. Models trained with VAE sampled initial pools consistently underperformed in the first episode of the AL cycles across all four datasets. On CIFAR-10, in the first episode, the average test accuracy difference between models trained with VAE initial pools and other initial pools was 12%. Similarly, there was a 11% difference in case of CIFAR-100, 4.5% in case of Tiny Imagenet and 18% in case of MNIST. We suspect this is due to the difference in models used for initial pool sampling and active learning. It has been empirically shown that data points actively sampled by one model, say VGG16, do not transfer well to another model, say ResNet-18 ([70, 68]). To allow for smooth transfer of samples, we used the same ResNet18 model for training SimCLR, SCAN and active learning episodes. However, following general trends of use of VAEs, we use a simple VAE model with 4 convolutional layers each in the encoder and the decoder, which may have resulted in this significant difference.

At the end of all AL cycles, on CIFAR-10, Tiny Imagenet and MNIST, we see that all initial pools converge to largely similar test accuracy, suggesting no significant improvement in AL performance. But we see a different outcome in the case of CIFAR-100 (Figure 3.3). On CIFAR-100, models starting with VAE-sampled initial pools, despite the bad start, ultimately outperform the models starting with the other four initial pools in six out of the seven configurations. In the passive learning configuration (control experiment), we notice that VAE appears to be outperforming others but that happens only in the final episode of the AL cycle, suggesting that this performance gain in six configurations was not a mere coincidence. The models starting with VAE start to outperform their random counterparts right after the second episode (20%), and we notice the VAE curve starting to diverge from others. However, this behavior was only seen on CIFAR-100.

To summarize the findings, our proposed methods could not conclusively prove the existence of *good* initial pools that help AL methods in the long run, although the use of VAE-based initial pool strategy showed some interesting trends.

What explains the odd behavior of VAE sampled initial pools on CIFAR-100?

To investigate the reason behind the odd behavior of models started with VAE-based initial pools, we study the class distribution of initial pools obtained by 4 methods - VAE, SCAN, SimCLR and K-Means. To this end, we picked initial pools from the

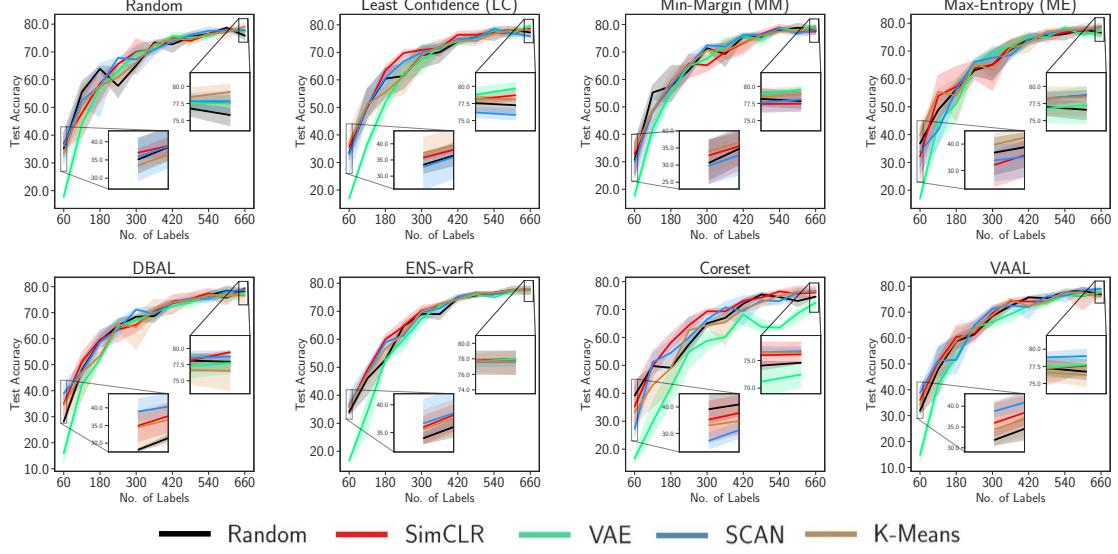


Figure 3.5: MNIST.

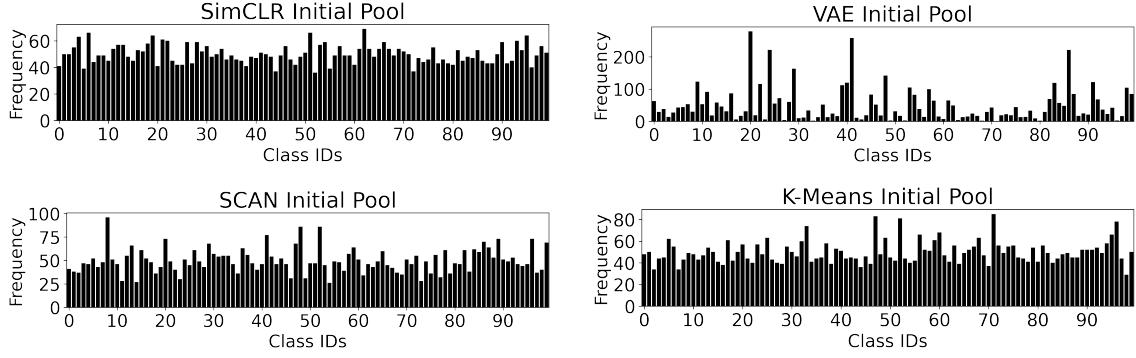


Figure 3.6: CIFAR-100: Class distribution of initial pools picked by various methods. Note the apparent class imbalance in the initial pool picked by VAE. Is this the reason for the performance gain?

DBAL experiment.

Looking at the class frequencies of all CIFAR-100 initial pools in Figure 3.6, we notice a clear difference between the VAE-sampled initial pool and the others. VAE-sampled initial pool has more class imbalance and is particularly emphasizing on images from specific classes. To verify if the VAE-based initial pool sampling technique is biased towards difficult classes, we created two sets of CIFAR-100 classes: (1) Top 10 classes sampled by VAE, (2) 10 classes with least per-class test accuracy w.r.t. the model in the final AL episode. We use (2) as a proxy for “difficult” classes. We observed that both these sets have 4 classes in common. While this overlap is not high enough to conclude that VAE sampling is biased towards difficult classes, it nevertheless is an interesting future direction to pursue, and merits more study.

Ablation Experiments

Comparing the Initial Pools: We used SimCLR representations to obtain t-SNE embeddings on all initial pools of a randomly chosen Max-Entropy (ME) experiment on CIFAR-10. The t-SNE plots of 5000 data points are shown in Figure 3.7. Unsurprisingly, we see no apparent inconsistencies or anomalies in either class distribution or inter-class relationships across the four initial pools except for the VAE-sampled initial pool whose class distribution is noticeably different than the others.

The other four initial pools are nearly identical. A confusion matrix with their overlap statistics, shown in Figure 3.7, shows that all five initial pools roughly shared approximately 10% of the data points among themselves. Even with nearly 90% of unique data points, all four initial pools contributed to largely similar model generalization error (as seen in Max-Entropy graph of Figure 3.2).

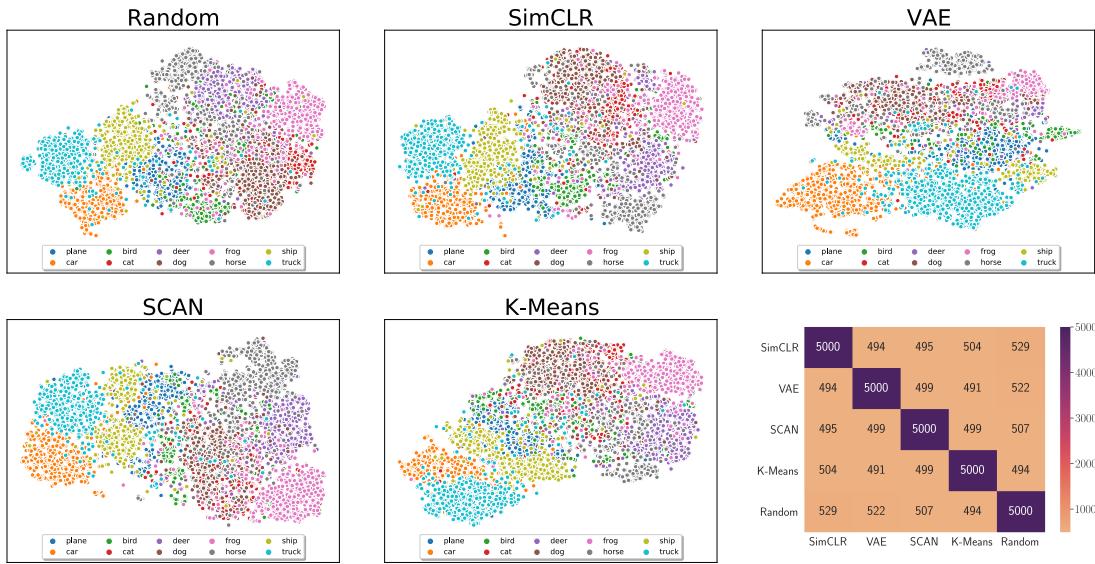


Figure 3.7: CIFAR-10: Initial pools visualized using t-SNE.

Low-Budget AL: Is 10% of data points too many for the model? Is that why we are unable to spot any potential performance differences between the four mostly unique initial pools? To find out if a low query budget can help spot performance differences, we repeated our experiments on CIFAR-10 for Max-Entropy (ME), Least Confidence (LC) and Deep Bayesian (DBAL) AL query methods but with just 1000 samples (2% of the overall dataset size) in the initial pool. We set the AL budget to 1000 and allowed the AL cycles to run up to 10 episodes (22% of the overall dataset size). The results of these experiments (averaged over 2 runs) are shown in Figure 3.8. All three AL methods benefit from VAE-sampled initial pools, albeit marginally, while other

initial pools do not contribute to any performance gain compared to random initial pools.

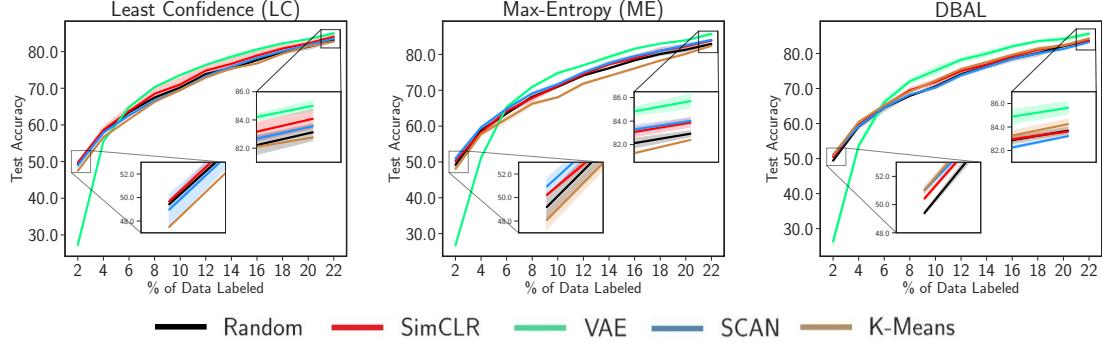


Figure 3.8: CIFAR-10: In low budget AL setting, only VAE initial pools show marginal performance gains over random initial pools.

Long-Tail CIFAR-10: One of the motivations behind our proposed unsupervised method (Section 3.3.2) was to allow AL cycles to start with a balanced initial pool, which spans the entire data distribution, when dealing with imbalanced datasets. To that end, we created a Long-Tail CIFAR-10 with an imbalance factor of 50 ([93]). We report the experiment results on three AL methods (ME, LC, DBAL) averaged over 2 runs in Figure 3.9. Surprisingly, our unsupervised initial pool sampling methods did not help the three AL methods. In fact, models trained on SCAN-based initial pools did consistently worse than models trained on random initial pools. Once again, VAE-based initial pools positively contribute to three AL methods albeit the performance gain is quite marginal.

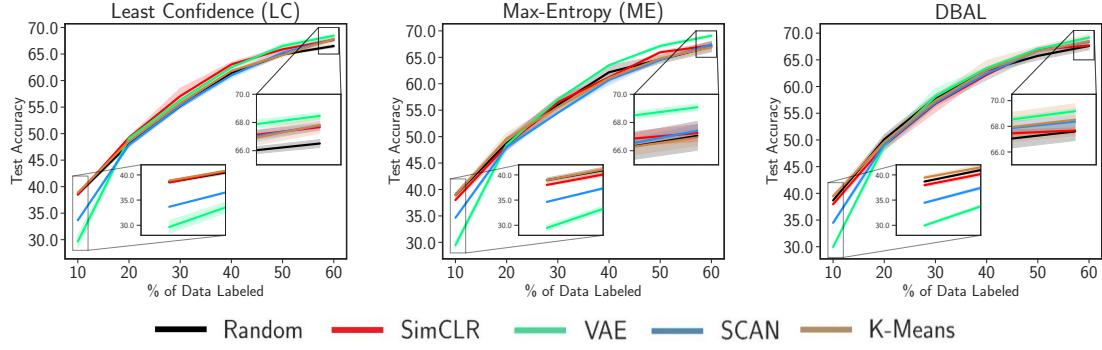


Figure 3.9: Long-Tail CIFAR-10: Our unsupervised sampling methods (SCAN and K-Means), motivated by this imbalance setting, did not improve LC, ME, DBAL query method performances. In the long run, VAE-based initial pools show marginal performance gains over random initial pools.

3.6 More Training Details

In this section we mention more training details necessary for reproducing our experiments.

3.6.1 Slightly Modified ResNet18 Model

To add an extra projection layer as the penultimate layer of the model is a convention in self-supervised learning methods ([71, 95]). To be consistant with the ResNet18 model used for SimCLR and SCAN training, we added a projection layer to the model just before the final fully connected layer. Projection dimension was set to 128 for MNIST, CIFAR-10/100, 512 for Tiny ImageNet experiments. Also, the official ResNet18 implementation doesn't include any dropout layers in it. To allow for DBAL method to run Monte Carlo simulations, we added a dropout layer with $p=0.5$ after the flattening layer. We only did this for DBAL experiments.

3.6.2 Hyperparameters for AL Training

For all experiments on all datasets, we set momentum = 0.9, $wd = 3e^{-4}$, gamma = 0.1. Other hyperparameter choices are as follows:

Dataset	Epochs	Optimizer	Learning Rate	Scheduler	Batch Size
CIFAR-10/100	200	SGD	0.025	Cosine (0.1)	96
MNIST	100	Adam	0.005	None	64
TinyImageNet	100	Adam	0.001	None	200

Table 3.1: Hyper-parameters of AL Cycles

3.6.3 SimCLR, SCAN and VAE Training

For all four datasets we train SimCLR and SCAN with largely similar hyperparameters. We use the official implementation of SCAN⁵ (includes SimCLR implementation as well) for training and use their recommended hyperparameters across all experiments. In case of CIFAR-100, we follow the standard practice and group the 100 classes into 20 super classes before training SimCLR (the grouping details can be found in the official SCAN repository). Evaluation metrics of our final SimCLR + SCAN + Self Labeling models are as follows:

⁵<https://github.com/wvangansbeke/Unsupervised-Classification/>

Dataset	ACC	NMI	ARI
CIFAR-10	0.70	0.46	0.38
CIFAR-100	0.44	0.41	0.25
MNIST	0.86	0.72	0.72
Tiny ImageNet	0.10	0.07	0.05

Table 3.2: Final Model Performances after Self-Labeling (SimCLR + SCAN + Self-Label)

In case of VAE training, we trained a Vanilla VAE⁶ on the entire training data with hyperparameters as follows: $\text{optimizer} = \text{Adam}$, $lr = 0.001$, $\text{epochs} = 100$, $\text{momentum} = 0.9$, $wd = 5e^{-4}$, batch size = 200, for all four datasets. We used 5% of the training data as the validation set. The model weights at epoch with best loss are saved for initial pool sampling.

3.7 Conclusion

In this work, we proposed two kinds of strategies – self-supervision based and clustering based – for intelligently sampling initial pools before the use of active learning (AL) methods for deep neural network models. Our motivation was to study if there exist good initial pools that contribute to better model generalization and better deep AL performance in the long run. Our proposed methods and experiments conducted on four image classification datasets couldn’t conclusively prove the existence of such good initial pools. However, a surprising outcome of this study was how initial pools sampled with a simple VAE task contributed to improved AL performance, better than more complex SimCLR and SCAN tasks. Even though VAE-based initial pools worked better than random initial pools only on one dataset (CIFAR-100), ablation studies on low budget CIFAR-10 settings as well as on Long-Tail CIFAR-10 point towards potential in VAE-sampled initial pools. Are images that are hard to reconstruct for VAEs good for generalization? Can better generative models like GANs do better than VAEs? We leave this for future work. While our methods and findings could not conclusively prove our hypothesis that AL methods can benefit from more intelligently chosen initial pools, we are optimistic about the potential this research direction holds.

⁶<https://github.com/AntixK/PyTorch-VAE>

Chapter 4

On Adversarial Robustness: A Neural Architecture Search perspective

4.1 Understanding Adversarial Robustness from an Architecture Perspective

One of the two popular ways in which adversarial examples are created is known as white-box attacks. In this setting, the attacker has access to the model architecture and the model parameters. The adversarial examples are typically calculated by taking the gradient of the loss function, which is a function of the network topology. This raises an important question that forms the primary motivation of this work, *Can the complex topology of a neural network architecture provide adversarial robustness without any form of adversarial training?*

In an attempt to understand adversarial robustness purely from an architectural perspective, in this chapter, we seek to answer the following questions,

abc

- How robust are existing SoTA image classification architectures without any form of adversarial training?
- How do NAS-based architectures compare with hand-crafted architectures (like ResNets, DenseNets, etclet@tokeneonedot) in terms of architectural robustness?
- Does an increase in the number of parameters of the architecture help improve robustness?

- Where does the source of adversarial vulnerability lie for NAS? Is it in the search space or in the way the current methods are performing the search?

To the best of our knowledge, our work is the first attempt at understanding adversarial robustness purely from an architectural perspective. We show that the complex topology of neural network architectures can be leveraged to achieve robustness without adversarial training. Additionally, we introduce two simple metrics, *Harmonic Robustness Score (HRS)* and *Per-parameter HRS (PP-HRS)* that combine (1) the total number of parameters in a model, (2) accuracy on both clean and perturbed samples, to convey how robust and deployment-ready a given model is when no adversarial training is performed.

We examine the adversarial robustness of different hand-crafted and NAS-based architectures in a wide range of scenarios and find that for large-scale datasets, complex tasks and stronger attacks (like PGD) the traditional hand-crafted architectures like ResNet and DenseNets are more robust than NAS-based architectures (Figure 4.1) i.e., the adversarial robustness of a model heavily depends on the network topology. Results of our study can be used to design network architectures that can give adversarial robustness as a free add-on along with SoTA performance on un-perturbed samples. Finally, we show that the popular way of increasing parameters to increase robustness [96, 97] holds up only to an extent; after a certain threshold increasing parameters alone hurts both the adversarial robustness and accuracy of clean samples.

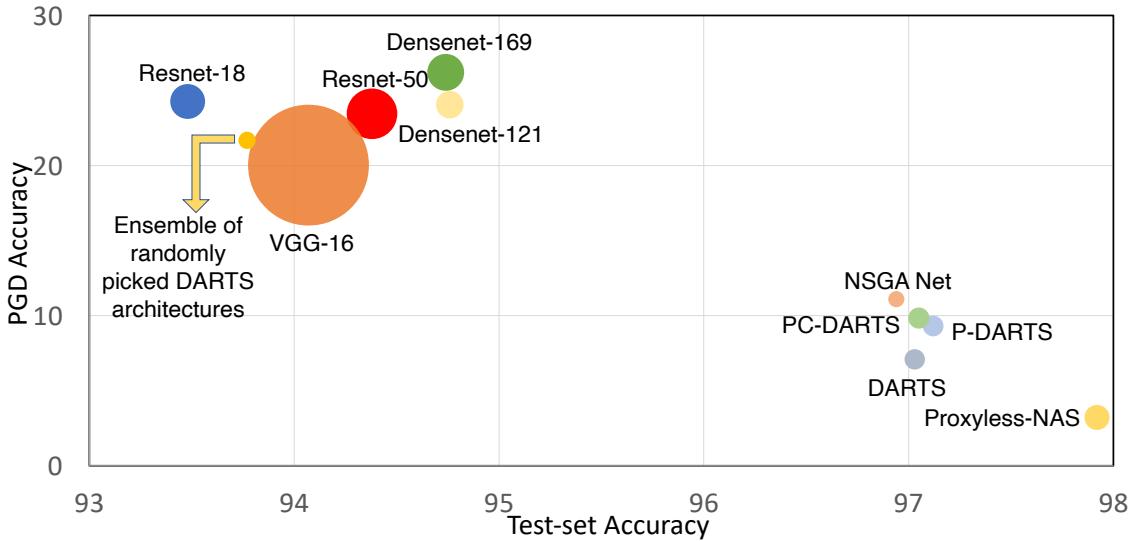


Figure 4.1: Comparison of test-set accuracy and PGD accuracy of NAS and hand-crafted architectures on CIFAR-10 dataset. Bubble size represents the number of parameters

4.2 Adversarial Robustness and Neural Architecture Search

Adversarial Attacks Categorisation: Existing adversarial attacks can be broadly classified into white-box and black-box attacks. The difference between these lies in the knowledge of the adversaries. In white-box attacks, the adversaries have the full knowledge of the target model, including the model architecture and parameters. In a black-box setting, the adversaries can only resort to the query access to generate adversarial samples. In the frameworks of these threat models several effective adversarial attacks have been proposed over the years such as L-BFGS [98], FGSM [99], BIM [100], C&W attacks [101] JSMA [102], Deep-Fool [103], R-FGSM [104], StepLL [105], PGD [96] and most recently SparseFool [106], F-FGSM [107]. For more information on adversarial attacks and defenses, please see [108, 109]. White-box is a stronger setting as the attackers can access the model parameters and architecture. It is also closely related to the network topology aspect of our study. So we mainly focus on the white-box setting for our work.

One popular way to improve the adversarial robustness of deep learning models is adversarial training (AT) [110]. The basic idea of AT is to create and incorporate adversarial samples during the training phase. A critical downside of AT is that it is time-consuming[111]; in addition to the gradient computation needed to update the network parameters, each stochastic gradient descent (SGD) iteration requires multiple gradients computations to produce adversarial images.

Categorisation of existing NAS approaches Over the years, several approaches have emerged to search architectures using methods ranging from Reinforcement Learning (RL) [30], Neuro-evolutionary approaches [112], Sequential Decision Processes [113], One-shot methods [33] and fully differentiable Gradient-based methods [114]. While most of these algorithms attempt to search a cell architecture (micro search) due to the computational cost involved and repeat the cell a fixed number of times, few recent approaches have also demonstrated searching the full architecture (macro search).

Most of the early approaches are based on RL and neuro-evolutionary algorithms, making the search process computationally intensive. Recently these have been replaced by one-shot fully-differentiable gradient-based NAS methods, such as DARTS [114], which are orders of magnitude faster than non-differentiable techniques and have gained much traction recently. P-DARTS [115] bridges the gap between search and evaluation by progressively increasing search depth. Partially-Connected

DARTS [116], a SoTA approach in NAS, significantly improves the efficiency of one-shot NAS by sampling parts of the super-network and adding edge normalization to reduce redundancy and uncertainty in search. DenseNAS [117], a more recent method, attempts to improve search space design by further searching block counts and block widths in a densely connected search space. Despite a plethora of these methods and their applications, there has been minimal effort to understand the adversarial robustness of final learned architectures.

Adversarial Robustness of Architectures: [96] is one of the early works to talk about adversarial robustness of network architectures. It shows that when training with un-perturbed samples, increasing the network’s capacity in terms of width, depth, and the number of parameters can alone help improve the robustness for datasets like MNIST and CIFAR-10. Recently, [97] echoes this observation by showing the depth of the networks helps improve the adversarial robustness during adversarial training. Both [96, 97] talk about robustness mainly in the context of adversarial training. However, our results show that when no adversarial training is performed, increasing parameters alone after a certain point will hurt adversarial robustness rather than helping.

Very recently, there have been limited efforts to improve adversarial robustness using architecture search [118, 119]. [118] proposes a robust architecture search framework by leveraging one-shot NAS. However, the proposed method adversarially trains the entire NAS search space before starting the search process, making it harder to assess the contribution of just the architecture to the adversarial robustness. [119] uses black-box attacks to generate a fixed set of adversarial examples on CIFAR-10 and used these examples to search for a robust architecture using NAS. The experimental setting is constrained and does not reflect the model’s true robustness as the adversarial examples are fixed a priori. No study is done on white-box attacks. Both [118] and [119] do not make any comparisons with existing NAS methods, which, as per our study, are already robust to an extent.

In this work, we mainly focus on evaluating the robustness of SoTA NAS methods on white-box attacks across datasets of different sizes, including large-scale datasets such as ImageNet [19] and compare them with hand-crafted models like ResNets and DenseNets. As a part of our study, we introduce metrics that can be used to estimate the trade-off between clean accuracy and adversarial robustness when comparing architectures within and across different families.

4.3 Robustness of NAS models: A Study

We have carefully designed our experimental setting to answer the questions stated in Section 4.1. We begin by describing the design of our experiments, providing details about datasets, models, attacks, and metrics.

Datasets: Since we want to compare the robustness of architectures across different dataset scales and complexities, we choose four different image classification datasets. In addition to the standard CIFAR-10 [120] data set, which consists of 60K images of 32×32 resolution, we also choose CIFAR-100 [121] to test if the same robustness trends hold when the labels turn from coarse to more fine-grained and the number of classes increase by a factor of 10. To study the robustness trend for tougher tasks like fine-grained image classification where the classes are semantically and perceptually more similar, we choose 102 Flowers dataset [122], which consists of 8189 flowers images split across 102 categories with each category having 40-258 images.

Since most real-world applications deal with large-scale datasets, we also test robustness on ImageNet [19] dataset, consisting of $\sim 1.3M$ images from 1000 classes. This makes our study more complete when compared to earlier works.

Architectures: We select most commonly used NAS methods including DARTS [114], P-DARTS [115], ProxylessNAS [123], NSGA-Net [124], along with recent methods like PC-DARTS [116] and DenseNAS [117]. We evaluate five well-known handcrafted architectures and at least four NAS architectures on each dataset mentioned above for a fair comparison. For all experiments, we either use pre-trained models made available by the respective authors or train the models from scratch until we obtain the performance reported in the respective papers. For the results on Flowers-102 dataset, we explicitly search for an architecture using the code provided by [125]. The results for NSGA-Net are only available for CIFAR-10/100 because its implementation does not support Imagenet. Similarly, the implementation of DenseNAS does not support CIFAR-10/100, so the results are shown only for ImageNet. ProxylessNAS provides pre-trained models for both CIFAR-10 and ImageNet, so we show the results only on those two datasets.

Ensemble of Architectures: When compared with single architecture, an ensemble of architectures are known to be adversarially more robust [126]. To understand the effectiveness of ensembling, in Section 4.4.4, we random sample cells from the DARTS search space using the code provided by [125]; and stack these cells to create small architectures, since this is randomly sampling, the search cost associated with building

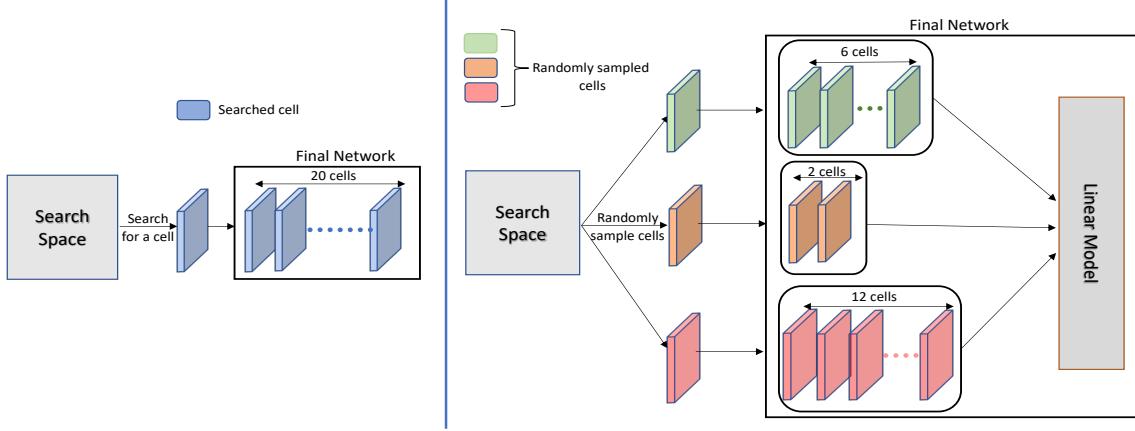


Figure 4.2: *Left:* Standard procedure for building architectures from DARTS search space; *Right:* Procedure for building ensembles using DARTS search space. 12, 6, 2 can be replaced with any values that sum to 20.

these architectures is zero. After sampling, we follow the standard DARTS training protocol to train these architectures. In general, for the CIFAR-10 dataset, DARTS architectures having 20 cells are trained for 600 epochs. Following this, the number of epochs for training each network in the ensemble is determined based on the number of cells in that network. Effectively the ensemble as a whole is trained for 600 epochs to ensure we make a fair comparison with existing approaches. After training each of these networks separately; We train a simple linear model to combine the individual model outputs; this linear model is trained only for two epochs. The difference between standard DARTS and ensembling by sampling from DARTS search space is visually shown in Figure 4.2. More details on the structure of the linear model are discussed in Section 4.4.4

Adversarial Attacks: For adversarial robustness, we test against the standard attacks like FGSM [99], PGD [96] and additionally report results on recently introduced F-FGSM[107] and AutoPGD[127]. For all these attacks, we use a perturbation value of 8/255 (3e-2) this denotes the maximum noise added to each pixel in the input image for perturbing it and step size of 2/255 (7e-3), with the attack iterations as 10. Additionally, we run the AutoPGD attack with 10 random restarts. All these parameter choices are standard and are widely used in the community [128, 129, 107], architectures are trained using standard training protocols, and no adversarial training is performed. We use the library provided by [130] for all the adversarial attacks in our experiments.

Metrics: We use Clean Accuracy and Adversarial Accuracy as our performance metrics. Clean accuracy refers to the accuracy on the undisturbed test set as provided

in the dataset. For each attack, we measure Adversarial accuracy by perturbing the test set examples using various attacks in the methods listed in 4.3. These adversarial accuracies are reported in the tables as FGSM, F-FGSM, PGD, depending on the attack.

One of the main problems with adversarially trained models is that their clean accuracy is less than standard non-adversarially trained models. Adversarial vulnerability is a side-effect of overfitting to the training set [131]. While this overfitting gives good performance on the test set, it makes the model vulnerable to adversarial examples. If a model accuracy on clean samples is not good, it is not useful when deployed in a situation where un-perturbed samples are often more often. On the other hand, if the model has SoTA performance on a clean test-set, it becomes vulnerable to adversarial examples. There is no proper metric to capture this clean accuracy vs. adversarial accuracy trade-off. We introduce a metric, which we call *Harmonic Robustness Score (HRS)*, which is the harmonic mean of clean accuracy and adversarial accuracy. HRS captures the trade-off between the clean accuracy and robustness to the commonly used PGD attack. By doing so, it captures the real-world usefulness of a model along with its resilience when no adversarial training is performed. Consider a model with clean accuracy C and PGD accuracy P (both in percentage), HRS for that model is calculated as follows:

$$\text{HRS} = \frac{2CP}{C + P} \quad (4.1)$$

When comparing performances of architectures belonging to the same family, number of parameters play an important role. So we further define per-parameter harmonic robustness score (PP-HRS) to measure the accuracy vs. robustness trade-off within a family of architectures. PP-HRS compares the parameters of the model with the parameters in the baseline model of that family. In a family of architectures (\mathcal{F}), consider a baseline model m_b having p_b million parameters, now for a model $m_i \in \mathcal{F}$ with p_i million parameters, PP-HRS is calculated as follows:

$$\text{PP-HRS} = \text{HRS} * \frac{p_b}{p_i} \quad (4.2)$$

4.4 Analysis and Results

In this section, we compare and contrast the robustness of different architectures in a wide-range of scenarios and answer questions listed in Section 4.1

4.4.1 How Robust is existing SoTA Image Classification Architecture without any form of Adversarial Training?

To gauge the adversarial robustness of current SoTA architecture used for image classification, we have chosen EfficientNet [34], a widely popular and SoTA architecture for image classification. In Table 4.1, we compare the clean accuracy, adversarial accuracy and HRS for three different variants, B0, B4, B7 of EfficientNet (choice based on parameter count; small, medium, high) on the ImageNet dataset with a fixed input image size of 224×224 . Results for all the other variants are discussed in Section 4.4.3. EfficientNet cannot be distinctly categorized as NAS or hand-crafted; they’re a mix of NAS and some heuristic-based hand-engineering. For this reason, we omit EfficientNet from our analysis in Section 4.4.2

[34] proposed EfficientNets, a family of models developed using NAS and compound scaling. The baseline network is developed using NAS, after which the optimal depth, width, and input image resolution are determined using compound scaling. Compound Scaling uses a compound coefficient (ϕ) to scale width, depth, and resolution in a principled way. In general, the compound coefficient ϕ is specified by the user; it controls how many more resources are available for model scaling. α , β , and γ shown in Eq ?? specify how to assign the resources to network width (w), depth(d), and image resolution(r), respectively.

In comparison to the best performing NAS and hand-crafted architectures in Table 4.4 (discussed in Section 4.4.2), EfficientNets are significantly better in terms of robustness to adversarial attacks. The difference in HRS score of EfficientNet from the best performing architecture is nearly 10%. In the case of PGD, a difference of 6% is quite significant for an ImageNet scale dataset. This significant difference in robustness raises the following question, What makes EfficientNets more robust than other architectures?

Variant	Clean %	FGSM	F-FGSM	PGD	HRS
B0	91.35	48.94	42.80	8.11	14.89
B4	92.73	66.73	59.26	16.99	28.71
B7	91.57	60.13	48.48	11.20	19.96

Table 4.1: EfficientNet Architecture comparison on ImageNet dataset with fixed image size of 224×224 (Top-5 Accuracy)

One significant difference between EfficientNet and existing NAS and hand-crafted models is the scaling factor. Most of the hand-crafted and NAS-based architectures are developed in a micro-style, *i.e* let@tokeneonedot, a small cell (like the ResNet block

or DARTS cell) is developed/searched, and it is stacked to build the full architectures of varying depths and parameter sizes. In the case of EfficientNet, this scaling is done systematically using the compound scaling method described in Eqn. ???. The difference of 6% to PGD attack can be purely attributed to the use of compound scaling, [132] echoes our claim; in the context of adversarial training, they empirically show that compound scaling can help achieve adversarial robustness when used for scaling any architectures. However, there is one crucial down-side of compound scaling. In general, the compound coefficient ϕ is specified by the user, and determining the optimal values for α, β, γ requires a grid-search to be performed. This is one of the main drawbacks of compound scaling as performing a grid search and choosing a good initial compound coefficient ϕ requires a good amount of computing power and brings back the trial-and-error mode in which most of the early deep learning architectures are developed. Letting NAS figure out the optimal way to scale a neural network would alleviate the compute required for grid-search and makes the complete process of finding an adversarially robust architecture end-to-end.

While 16% accuracy for a standard PGD attack might seem less compared to adversarially trained counterparts, our primary goal is to show that adversarial accuracy can be easily achieved as an *add-on* without using any form of adversarial training. Since NAS is already the de-facto choice for achieving SoTA clean accuracy on standard datasets, making adversarial robustness an add-on to these architectures would be a convenient way to make deep learning architectures resilient to simple adversarial attacks.

4.4.2 How do NAS based models compare with Hand-crafted models in terms of Architectural Robustness?

The HRS score and robustness of different hand-crafted and NAS based architectures on CIFAR-10, CIFAR-100, ImageNet and Flowers-102 datasets are shown in Tables 4.2, 4.3, 4.5, 4.4 respectively.

In the case of CIFAR-10 and CIFAR-100, NAS-based architectures outperform hand-crafted architectures in terms of architectural robustness for attacks like FGSM and F-FGSM by a significant margin. However, for stronger and most commonly used attacks like PGD, NAS based architectures fail significantly compared to hand-crafted models. In terms of HRS score, the difference in the best-performing NAS and hand-crafted models is 21%.

This trend seen in CIFAR-10/100 for attacks like FGSM, F-FGSM did not hold

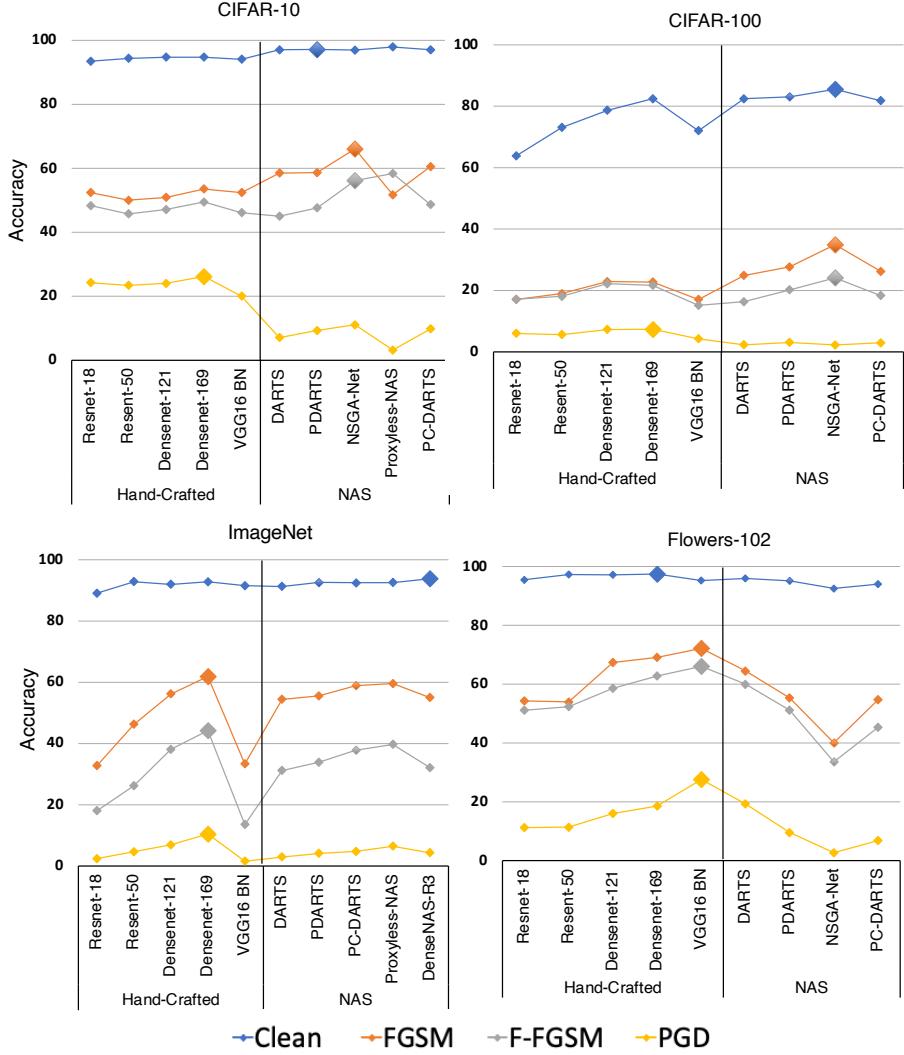


Figure 4.3: Comparison of robustness and clean accuracy of different architectures; As the difficult of the task or the scale of the dataset increases hand-crafted architectures are more robust; (best performance is indicated by diamond symbol)

for large scale datasets like ImageNet and relatively complex tasks like fine-grained classification. In the case of Imagenet, handcrafted models are more robust than NAS-based architectures for all the attacks. Similarly, for the task of fine-grained classification on Flowers-102 dataset, handcrafted models like DenseNet-169 and VGG-16 beat NAS based architectures by a significant margin. Even in terms of clean accuracy, for which the NAS-based models are generally known to be better than handcrafted models, NAS architectures fail by a margin of $\sim 1.5\%$ for the Flowers-102 dataset.

This trend of robustness for all four datasets is clearly shown in Figure 4.4. As the dataset size or the task complexity increases, hand-crafted models start to be better for all the three adversarial attacks. For stronger attacks like PGD, handcrafted

Model	Clean %	FGSM	F-FGSM	PGD	AutoPGD	HRS
ResNet-18	93.48	52.43	48.33	24.27	23.13	38.53
ResNet-50	94.38	50.05	45.78	23.45	22.35	37.57
DenseNet-121	94.76	50.94	47.14	24.06	22.66	38.38
DenseNet-169	94.74	53.53	49.47	26.21	24.35	41.06
VGG16 BN	94.07	52.42	46.16	20.03	18.63	33.03
DARTS [114]	97.03	58.53	45.03	7.09	6.10	13.21
PDARTS [115]	97.12	58.67	47.62	9.31	7.98	16.99
NSGA Net [124]	96.94	66.08	56.16	11.1	9.82	19.92
Proxyless-NAS [123]	97.92	51.73	58.38	3.22	4.24	6.23
PC-DARTS [116]	97.05	60.55	48.65	9.84	8.36	17.87

Table 4.2: Comparison of clean accuracy and adversarial robustness on CIFAR-10 dataset (Top-1 Accuracy)

Model	Clean %	FGSM	F-FGSM	PGD	AutoPGD	HRS
ResNet-18	63.87	17.08	17.12	6.05	5.39	11.05
ResNet-50	73.09	19	18.12	5.63	5.16	10.45
DenseNet-121	78.71	22.9	22.22	7.28	6.68	13.33
DenseNet-169	82.44	22.73	21.66	7.37	6.90	13.53
VGG16 BN	72.05	17.09	15.15	4.27	3.81	8.06
DARTS [114]	82.43	24.91	16.34	2.32	1.89	4.51
PDARTS [115]	83.07	27.69	20.23	3.09	2.66	5.96
NSGA Net [124]	85.44	34.93	24.1	2.26	1.94	4.40
PC-DARTS [116]	81.83	26.22	18.35	2.93	2.51	5.66

Table 4.3: Comparison of clean accuracy and adversarial robustness on CIFAR-100 dataset (Top-1 Accuracy)

models are more robust when compared to NAS based architectures at any given dataset scale. While NAS based architectures achieve SoTA clean accuracy in general, the robustness of these architectures is very erratic.

Model	Clean %	FGSM	F-FGSM	PGD	AutoPGD	HRS
ResNet18	89.08	32.75	18.03	2.41	21.65	4.70
ResNet50	92.86	46.28	26.22	4.68	20.93	8.90
DenseNet121	91.97	56.20	38.11	6.932	24.20	12.89
DenseNet169	92.81	61.89	44.22	10.46	27.15	18.80
VGG16	91.52	33.34	13.54	1.55	19.57	3.05
DARTS	91.26	54.41	31.18	2.94	20.81	5.70
P-DARTS	92.61	55.53	33.87	4.11	20.67	7.86
PC-DARTS	92.49	58.90	37.86	4.75	21.52	9.04
Proxyless-NAS	92.54	59.56	39.69	6.48	22.28	12.11
DenseNAS-Large	92.80	47.91	27.25	2.97	19.62	5.76
DenseNAS-R3	93.81	54.99	32.11	4.32	19.94	8.25

Table 4.4: Comparison of clean accuracy and adversarial robustness on ImageNet dataset (Top-5 Accuracy)

In summary, as the dataset size (in terms of the number of classes) or the task’s complexity increases, NAS-based architectures are more vulnerable to adversarial attacks than hand-crafted models if no adversarial training is performed.

Model	Clean %	FGSM	F-FGSM	PGD	AutoPGD	HRS
ResNet-18	95.48	54.33	51.16	11.23	10.38	20.10
ResNet-50	97.31	53.97	52.38	11.36	10.01	20.34
DenseNet-121	97.19	67.4	58.61	16	13.80	27.48
DenseNet-169	97.44	69.11	62.76	18.56	16.48	31.18
VGG16 BN	95.24	72.16	66.06	27.59	26.74	42.78
DARTS [114]	95.97	64.47	59.95	19.29	18.19	32.12
PDARTS [115]	95.12	55.31	51.16	9.52	8.55	17.31
NSGA Net [124]	92.55	40.05	33.58	2.69	2.08	5.23
PC-DARTS [116]	94.02	54.7	45.3	6.84	6.23	12.75

Table 4.5: Comparison of clean accuracy and adversarial robustness on Flowers-102 dataset (Top-1 Accuracy)

4.4.3 Does an increase in the number of parameters of Architecture help improve Robustness?

[133] and [96] observed that within the same family of architectures, increasing the number of network parameters helps improve robustness. We hypothesize that thus increasing model capacity benefits network robustness. To study this claim, we compare the robustness of five families of architectures on the ImageNet dataset with respect to the parameter count. For comparing the trends, we use PGD accuracy along with the Per-parameter Harmonic Robustness Score (PP-HRS). The five different families of architectures we considered for this study are mentioned below.

Firstly, we choose all the eight different variants of the EfficientNet family introduced in Section 4.4.1, followed by a recent SoTA NAS-based approach DenseNAS. DenseNAS architectures are developed using two different search spaces. DenseNAS-A/B/C and Large are developed using a MobileNetV2-based search space, and DenseNAS-R1, R2, R3 are developed using a ResNet based search space. These networks are listed in the increasing order of their parameters. Lastly, to also understand the trend in hand-crafted models, we study the robustness of standard DenseNet and ResNet models.

All the results of this comparison are shown in Table 4.6 and Figure 4.4. In 4/5 families considered for this study, an increase in parameters increases both clean and adversarial accuracy. The maximum value of the parameter count in these four families is nearly 26 million. This trend of increase in robustness with parameter count is also seen in the fifth family (EfficientNet) but only up to a parameter count of 20 million. Increasing the parameters "alone" beyond 20 million results in a decrease of both clean and adversarial accuracy; this is probably why EfficientNet considers different image sizes for each of the eight networks. After a certain point, increasing the parameters alone will not help improve robustness, and Efficient, which has the

Family	Variant	Params (M)	Clean %	PGD	PP-HRS
Efficient-Net	B0	5.29	91.36	8.11	14.90
	B1	7.79	88.89	5.47	7.00
	B2	9.11	92.77	11.40	11.79
	B3	12.23	93.04	13.37	10.11
	B4	19.34	92.73	16.99	7.86
	B5	30.39	90.95	9.37	2.96
	B6	43.04	91.86	11.71	2.55
	B7	66.35	91.57	11.20	1.59
DenseNAS	A	4.77	90.94	1.84	3.61
	B	5.58	91.89	2.13	3.56
	C	6.13	92.31	2.29	3.48
	Large	6.48	92.80	2.97	4.24
	R1	11.09	91.33	2.01	3.93
	R2	19.47	92.47	3.19	3.51
	R3	24.66	93.81	4.32	3.71
ResNet	18	11.69	89.08	2.41	4.69
	50	25.56	92.86	4.68	4.08
DenseNet	121	7.98	91.97	6.93	12.89
	169	14.15	92.81	10.46	10.60

Table 4.6: Comparison of parameter count vs Adversarial accuracy for five different family of architectures on ImageNet dataset

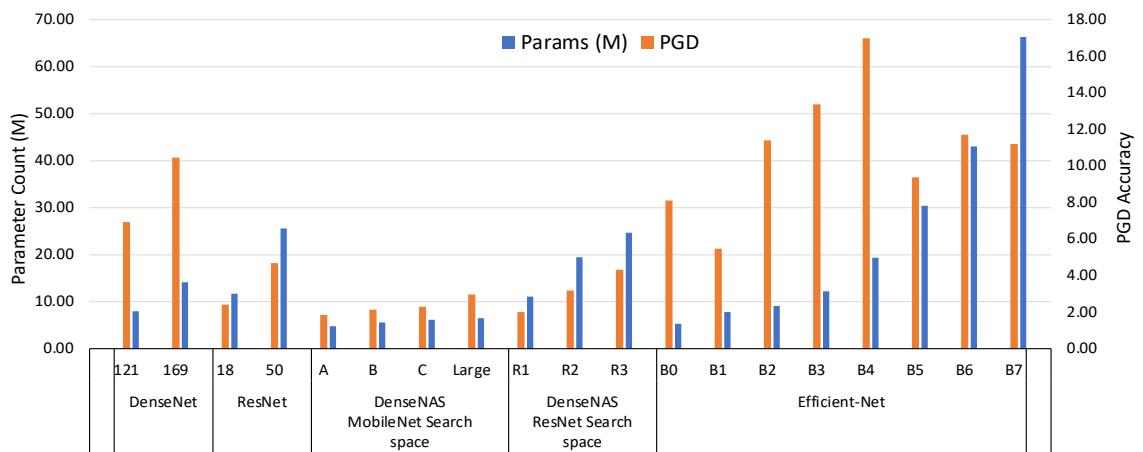


Figure 4.4: Comparison of PGD accuracy and Parameter count across different family of architectures

best adversarial accuracy in the case of ImageNet dataset, conveys this.

"In what family of architectures, the increase in parameter count is helping the performance?", to better understand this, we report PP-HRS in Table 4.6. In the case of DenseNAS models developed using MobileNet-V2 search space, an increase in parameters from DenseNAS-A to DenseNAS-Large is improving both clean accuracy and adversarial robustness; which as a result led to improved PP-HRS score. For all the other families, the increased parameter count does not give a significant and sufficient improvement in the PP-HRS score and adversarial robustness.

In summary, adversarial robustness can be improved by increasing the number of parameters, but this holds only to an extent. Beyond a certain point (approximately 20-25 million as per our results), increasing parameters alone cannot improve adversarial robustness.

4.4.4 Where does the source of adversarial vulnerability lie for NAS? Is it in the search space or in the way the current methods are performing the search?

In Section 4.4.2, we have seen that NAS based architectures are more robust than hand-crafted architectures for small scale datasets and simpler attacks. However, when it comes to standard attacks like PGD, NAS-based architectures are not robust even at the scale of CIFAR-10. Most of the existing NAS methods perform the search on CIFAR-10 or a subset of ImageNet, and the discovered cell is stacked and trained for other datasets. To understand whether the problem lies in the search space or in the way search is being performed by the existing methods, we performed two simple experiments.

Our first experiment is motivated by [125]; [125] shows that a randomly sampled cell in the DARTS search space gives as good a clean accuracy as a searched cell. To test if this fact also holds for the case of adversarial robustness, we sampled random cells from the DARTS search space, stacked and trained them using the standard procedure, and tested their robustness on the CIFAR-10 dataset. Due to the randomness involved, we report the value over four different runs; results of this experiment are shown in Table 4.7. Randomly sampled cells have a better PGD accuracy than the searched architecture, but the variance is very high, which shows relying on randomly sampled architectures for adversarial robustness is not a good idea; this led us to our second experiment.

For the second experiment, we randomly sample cells from the DARTS search

space to build small models (please refer to Figure 4.2 for a diagrammatic representation of this procedure). After training these models independently, we ensemble the outputs of all these models using a simple linear network. This linear model consists of 2 linear layers with Batch-norm and one fully connected layer towards the end for outputting logits based on the number of classes in the dataset. This linear model is just fine-tuned for two epochs. Entire ensemble is treated as one single-network when generating the adversarial examples. To make a fair comparison, we ensure that the ensemble as a whole has the same number of cells as the standard DARTS networks. Since the procedure uses randomly sampled architectures, we run the entire sample-train-ensemble procedure four times and report the mean value in Table 4.7. Since this is an expensive procedure computationally, due to the randomness involved, we restrict our experiments to the CIFAR-10 dataset and DARTS search space.

Table 4.7: Adversarial accuracy comparison of DARTS based architectures on CIFAR-10 dataset

Model	# cells	Params (M)	Clean %	PGD
DARTS [114]	20	3.35	97.03	7.09
P-DARTS [115]	20	3.43	97.12	9.31
PC-DARTS [116]	20	3.63	97.05	9.84
RANDOM ¹	20	2.73 ± 0.49	95.57 ± 0.40	14.47 ± 4.70
ENSEMBLE ²	20	2.74 ± 0.41	93.77 ± 0.39	21.68 ± 0.35

¹ Randomly picked architectures from DARTS search-space. Value reported over four runs

² Ensemble of small, randomly picked architectures from DARTS search space. Value reported over four runs

Surprisingly this simple ensemble of randomly sampled architectures can improve the PGD accuracy of DARTS based models by nearly 12% and can decrease the variance by $\sim 10\%$. Now, this leads to two interesting conclusions; (1) Learning to build a simple network to combine the outputs of randomly sampled architectures can give clean accuracy with adversarial robustness as an add-on. In this case, we used a simple linear model; replacing this with a searched NAS based architecture can improve the results further. (2) Using NAS to search for an ensemble of architectures can be a potential way to achieve adversarial robustness as an add-on to SoTA clean accuracy. In this case, the NAS objective should be modified to find small models that can complement each other. We plan to explore this in our future work.

4.5 Conclusion

In this chapter, we presented a detailed analysis of the adversarial robustness of NAS and hand-crafted models and show how the complex topology of neural networks can be leveraged to achieve adversarial robustness without any form of adversarial training. We also introduce a metric that can be used to calculate the trade-off between clean and adversarial accuracy within and across different families of architectures. Finally, we show that using NAS to find an ensemble of architectures can be one potential way to build robust and reliable models without any form of adversarial training.

Chapter 5

On the Use of Skip Connections for Transfer Learning

5.1 Need for Input-Adaptive Skip Connections

Existing efforts in transfer learning can be broadly divided into two categories: mechanisms for fine-tuning of pre-trained weights in a task-specific manner [134, 135, 136, 137, 138] and regularisation schemes [139, 140, 141, 142, 143] to make the pre-trained features more suitable to the current task. While our method falls in the former category, unlike previous methods, we primarily focus on improving the *interaction* between pre-trained features, as defined by the skip connections. We develop a mechanism that can be easily integrated and used in existing architectures.

We view skip connections as a handle that can control pre-trained feature interactions at various granularities. Though skip connections are widely used in modern-day architectures, their primary usage has focused on training of deeper models. In this work, we introduce an architectural modification using skip connections that are weighted and input-conditioned. Our modified version, called **Adaptive Skip Connections** (AdaSkips), allows us to transfer a model pre-trained on a source dataset to a newer target dataset by simply learning interactions between pre-trained features using such adaptive skip connections.

While the standard practice of one skip connection for a block of layers addresses the problem of training very deep neural networks, revisiting skip connections as a mechanism to control the routing of feature information across layers in a neural network can allow modeling complex interactions between the features. To this end, we consider all possible skip connections between layers of a pre-trained neural network

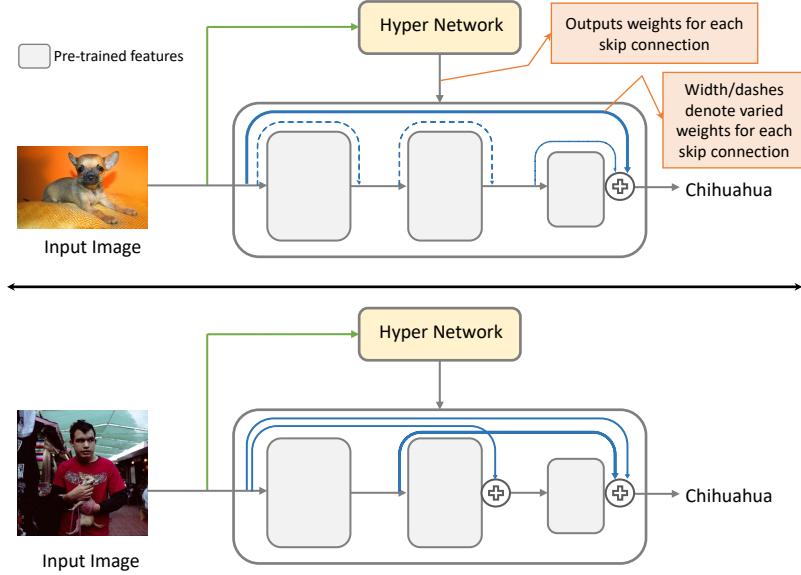


Figure 5.1: Illustration of AdaSkips for two images of same class from Stanford-Dogs [8] dataset. AdaSkips determine which skip connections to use and assigns a weight for each pre-trained feature map via skip connections.

as possible routes for a given input. The target input is fed to an auxiliary (small and lightweight) HyperNetwork [144] to obtain weights for each skip connection in the above model.

Such input-adaptive skip connections (AdaSkips) provide for learning input conditioned models, allowing better reuse of the learned features from a pre-trained model. We show that this reasonably simple strategy outperforms existing methods on standard TL benchmarks with negligible training time overhead and almost the same time as base models at test time. In particular, we show that this strategy works well even in low-data regimes when there is limited data available from the target dataset. Our overall approach is illustrated in Fig 5.1. Our key contributions can be summarized as follows:

1. We propose AdaSkips, a simple and effective use of skip connections in existing architectures to facilitate better feature reuse for transfer learning (TL). AdaSkips uses a hypernetwork to learn input-conditioned skip connection weights on target data, thus making the process of adaptively selecting and weighing skip connections end-to-end differentiable and optimizing directly for task loss.
2. AdaSkips can not only be used independently on pre-trained models to perform TL, but can also be easily added to existing methods to further improve performance.
3. We perform a comprehensive suite of experiments on standard TL benchmarks, and show that our method consistently outperforms existing methods, including

in low-data regimes when target domain data is limited. We also study the generalizability of our method by demonstrating its use for more complex tasks such as zero-shot domain adaptation, object detection and semantic segmentation. Our thorough analysis and ablations show that careful routing and re-weighting of pre-trained features is a promising avenue for better TL performance.

5.2 Transfer Learning and Input-Conditioned Architectures

Transfer Learning: Transfer learning is widely adopted in domains where it is difficult to construct large-scale well-annotated datasets. Transfer learning settings can be broadly classified into two categories, 1. Transductive transfer learning (no labeled data available in target domain) 2. Inductive transfer learning (labeled data available in target domain). In both of these settings, fine-tuning of the pre-trained weights is common along with the introduction of a FC layer having neurons equal to the number of classes in the target dataset. Finetuning can be done in different ways; one can choose to finetune all the features of the pre-trained model or finetune only the last few layers. Recently there has been much work on improving transfer learning [145, 134, 141, 139, 136, 142, 135, 143, 140, 137, 138] and all of these methods fall in one of the two categories mentioned above. In contrast to most of these methods, which introduce explicit regularisation or a learned policy to select which layers or filters to use, we primarily make use of a commonly used inductive bias, the skip connections to develop a flexible finetuning mechanism. While most of existing works mainly focus on either Inductive or Transductive transfer learning, our method is suitable for both the settings and can be integrated easily with any existing transfer learning methods.

Skip Connections: Skip connections can help preserve low-level features and avoid performance degradation when adding more layers. Since the introduction of skip connections, several works have been proposed to use skip connections selectively. Rather than using skip connections in a specific manner (one for a set of two or three layers), [146] has introduced DenseNets that use all possible skip connections in a feed-forward manner and replaced the fully-differentiable addition operation in ResNets with concatenation. Due to the absence of any input conditioning in DenseNets, all the skip connections are used for every data point, introducing overhead on the number of addition operations required. [147] introduced SparseNet, a sparse variant of

DenseNet which drops connections in a fixed manner without any input-conditioning i.e., it drops connections from middle layers preserving only the farthest and nearest connections. Unlike these approaches, we select which skip connections to use and assign a weight to each skip connections in a input-specific manner.

[148] proposed weighted skip connections for super-resolution; while the high-level idea of their work is close to ours, there are significant differences in the way we approach the design of the skip connection, primarily in how we assign the weight to the skip connection. To determine each skip connection weight in the network, they use a separate self-attention mechanism, which is acceptable computationally for their method because the maximum number of skip connections in their network architecture is 15. In our case, we use only four hypernetworks for any variant of ResNet [37]. Additionally, their method and results can directly benefit from the variance reduction approach we discuss in Section ???. Lastly, our method is specifically designed for transfer learning, keeping in mind the size of large networks that are commonly used.

Input-conditioned architectures: Best performing deep neural networks have tens of millions of parameters and hundreds of layers. This increase in depth and parameters comes with increased prediction accuracy. However, the relationship between accuracy and the number of parameters, layers is not linear. Very few images in a given dataset might require all the layers. So to make the weights input conditioned [136] proposed SpotTune, it determines which layers to fine-tune in an input-conditioned manner. In contrast to their approach, our method can work in a setting where all the layers except the fully-connected layers are frozen, and we only weight the skip connections of each layer.

Another body of work primarily focuses on using input-conditioning to decrease the cost of inference; many methods [149, 150, 151, 152, 153] have been proposed to reduce the cost of inference by choosing layers in an input conditioned manner. While we condition a small part of the network on the input, our goal is entirely different from this line of work. Our goal is to perform better when fine-tuning, and unlike most of the methods listed above, we only condition a small part of the network architecture (skip connection) rather than the entire network. While all these works focus on conditioning the weights, we focus on conditioning the input image on the path to be followed, keeping the pre-trained weights untouched.

Hypernetworks: [144] introduced hypernetworks, an approach of using one network to generate weights of another network. While Hypernetworks were initially introduced to generate weights of an entire network, we only use them to generate weights

for skip connections in a network. The primary reason for the usage of hypernetwork is to enable adaptive selection of pre-trained features.

5.3 AdaSkips for Transfer Learning

Preliminaries: We begin by briefly describing the standard transfer learning setting as well as skip connections in most commonly used architectures, before describing our method on how these skip connections can be leveraged and re-designed to allow controlling interactions between pre-trained features.

The standard transfer learning setup consists of a source dataset \mathcal{D}_S and target dataset \mathcal{D}_T . Generally, the target dataset has lesser number of samples when compared to the source dataset $|\mathcal{D}_T| << |\mathcal{D}_S|$ thus necessitating transfer and ensuring that a large-model can be easily pre-trained on \mathcal{D}_S without problems of overfitting. The primary goal of a transfer learning approach is to learn a good representation \mathcal{F}_S on \mathcal{D}_S and leverage that representation along with \mathcal{D}_T to obtain a newer representation that helps in solving the task on the target dataset \mathcal{D}_T accurately without any overfitting on \mathcal{D}_T . Let the features learned on \mathcal{D}_S be denoted by \mathcal{F}_S , where \mathcal{F}_S^k denotes a single layer or a block of layers, say with convolutional, max-pooling operations and activation functions. Notably, \mathcal{F}_S , the final representation learned from \mathcal{D}_S , is a composition of features learned at each layer, and can be written as below for a given input sample \mathbf{x}_s :

$$\mathcal{F}_S = \mathcal{F}_S^k(\mathcal{F}_S^{k-1}(\dots(\mathcal{F}_S^1(\mathbf{x}_s)))) \quad (5.1)$$

Now, for transferring \mathcal{F}_S to the target dataset, the last fully-connected layer, \mathcal{F}_S^k is typically replaced with \mathcal{F}_T^k which contains as many neurons as the classes in the target dataset. The other pre-trained weights and the weights of the new FC layer are updated with the help of samples from \mathcal{D}_T . Let the final representation learned on the target dataset then be given by:

$$\mathcal{F}_T = \mathcal{F}_T^k(\mathcal{F}_S^{k-1}(\dots(\mathcal{F}_S^1(\mathbf{x}_t)))) \quad (5.2)$$

where \mathbf{x}_t is an input sample from \mathcal{D}_T .

As mentioned previously, \mathcal{F}^k denotes a single layer (or a block of layers comprising convolutional, max-pooling layers along with the commonly used activation functions). For example, in the case of ResNet [37], \mathcal{F}^k could be a residual block with a skip connection. Most modern-day architectures design such blocks and stack

blocks together along with skip connections to build larger architectures. Given Eqns 5.1 and 5.2, the output representation for any \mathcal{F}^j which represents a block of layers with a skip connection can be denoted as:

$$\mathcal{F}^j = \mathcal{F}^j(\mathcal{F}^{j-1}(\dots(\mathcal{F}^1(\mathbf{x}))); \mathbf{W}_j) + \underbrace{\mathcal{F}^{j-1}(\mathbf{x}; S_w)}_{\text{Skip connection}} \quad (5.3)$$

where \mathbf{W}_j denotes the weights corresponding to the j th block and the highlighted term in Eq 5.3 refers to the representation learned by the previous block, which is obtained with the help of a simple identity mapping i.e, $S_w = I$. While existing transfer learning methods [141, 139, 136, 142, 135, 143, 140, 137, 138] primarily focus on modifying or fine-tuning \mathbf{W}_j , we instead focus on learning an input-conditioned skip connection and thus a weighted combination of features which are adaptive to a given input image without modifying \mathbf{W}_j .

AdaSkips: As discussed above, the addition of features obtained from previous layers, $\mathcal{F}^{j-1}(\mathbf{x})$ using skip connections (Eqn 5.3) helps propagate information in deep neural network models with many layers during backpropagation. We leverage this simple and elegant design of a skip connection to transfer pre-trained features to other datasets more effectively. In other words, our fundamental objective is to develop a mechanism that allows us to control the interactions among pre-trained features in a way to achieve effective transfer the features to a distinct target dataset. To this end, we introduce AdaSkips, which help maneuver the learned low-level and high-level features of a pre-trained model in an input-adaptive manner for a target dataset. In order to allow such an adaptive approach, we first introduce all skip connections possible between each block in a module (a set of blocks) of the network which is trained on the source dataset. When learning to combine pre-trained features, a feature combination that is optimal for one image might be suboptimal for another image, even from the same dataset or class label. Keeping this in mind, we learn feature combinations in an input-conditioned manner using a separate, small and lightweight hypernetwork (\mathcal{H}) parametrized by weights θ . Thus, for a given input \mathbf{x} , the skip connections of the model are weighted in an input-conditioned manner using a hypernetwork \mathcal{H} . Following Eqn 5.3, the output of a block \mathcal{F}_j now changes

as follows when AdaSkips are introduced:

$$\lambda_i = \mathcal{H}(\mathbf{x}; \theta)$$

$$\mathcal{F}^j = \mathcal{F}^j(\mathcal{F}^{j-1}(\dots(\mathcal{F}^1(\mathbf{x})); \mathbf{W}_j) + \sum_{i=1}^{j-1} \mathcal{F}^i(\mathbf{x}; \lambda_i)) \quad (5.4)$$

Intuitively, we input the image to the hypernetwork to obtain mixing coefficients $\lambda_{\{1, \dots, (j-1)\}}$ for each of the skip connections in the network. Since the hypernetwork is a parametrized neural network and therefore fully differentiable, it can be trained directly on the task loss via gradient descent. Unlike standard skip connections with $\lambda_i = 1$, adaptive skip connections have input-conditioned weights on them instead. We note that since the skip connections are introduced after training, our model can be directly used with existing pre-trained models. An end-to-end algorithm for transfer learning using AdaSkips is shown in Algorithm 2.

Algorithm 2: AdaSkips for Transfer Learning

```

1 Input: Pre-trained model  $\mathcal{F}_{\mathcal{S}}^{1\dots k}$  with  $k$  modules; Samples  $\mathbf{x}_d^{m=1\dots n} \in \mathcal{D}_{\mathcal{T}}$  in
    $n$  batches; Randomly initialised hypernetworks  $\mathcal{H}^{1\dots k}$ ; Loss function used for
   classification  $\mathcal{L}$ 
2 Training: Initialize  $\mathcal{F}_{\mathcal{D}}$  using  $\mathcal{F}_{\mathcal{S}}$  and freeze initial layers; Introduce all
   possible skip connections ( $p$ ) in each block of  $\mathcal{F}_{\mathcal{D}}^{1\dots k}$ 
3 for  $num\_epochs$  do
4   for  $\mathbf{x}_d^{m=1\dots n}$  do
5     for  $j = 1 \dots k$  do
6        $\lambda_{1\dots p} = \mathcal{H}^j(\mathbf{x}_d^m; \theta_j)$ 
7        $\mathcal{F}_{\mathcal{D}}^j = \mathcal{F}_{\mathcal{D}}^j(\mathbf{x}_d^m; \mathbf{W}_j) + \sum_{i=1}^{j-1} \mathcal{F}_{\mathcal{D}}^i(\mathbf{x}_d^m; \lambda_i)$ 
8     end
9     Calculate  $\mathcal{L}$      $\triangleright$  task loss
10    Update  $\mathcal{F}_{\mathcal{D}}^k$      $\triangleright$  update non-frozen layers
11    Update  $\mathcal{H}^k$        $\triangleright$  update hypernetworks
12  end
13 end
14 Output:  $\mathcal{F}_{\mathcal{D}}$ - model fine-tuned on the target dataset

```

Addressing variance introduced by AdaSkips: Random initialization of weights in a multi-layer network can result in large variance in outputs potentially resulting in unstable training [154]. Several weight initialization methods [154, 155, 156] have been proposed to address this problem. Kaiming initialization [156] is a popular strategy

typically used to maintain the variance between the input and output of every block. However, when skip connections are present in an architecture, this initialization introduces a new problem of increased variance [157]. The output of any given j th block, \mathcal{F}^j , with one skip connection is given by $\mathcal{F}^j = \mathcal{F}^j(\mathcal{F}^{j-1}(\mathbf{x})) + \mathcal{F}^{j-1}(\mathbf{x})$. The variance of the output representation can hence be written as:

$$\begin{aligned}\text{Var}(\mathcal{F}^j) &= \text{Var}(\mathcal{F}^j(\mathcal{F}^{j-1}(\mathbf{x})) + \mathcal{F}^{j-1}(\mathbf{x})) \\ &= \text{Var}(\mathcal{F}^j(\mathcal{F}^{j-1}(\mathbf{x}))) + \text{Var}(\mathcal{F}^{j-1}(\mathbf{x})) \\ &\quad + \text{Cov}(\mathcal{F}^j(\mathcal{F}^{j-1}(\mathbf{x})), \mathcal{F}^{j-1}(\mathbf{x}))\end{aligned}\tag{5.5}$$

It is known that when a network is initialized with standard initialization strategies such as Kaiming initialization [156], the variance of input and output of the block remains the same (we include proof for Kaiming initialization in the Appendix). Besides, as shown in [157], the correlation term in Eqn 5.5 turns out to be negligible, and the first two variance terms will be approximately the same (due to the initialization strategy such as Kaiming's). Hence, when skip connections are present, the overall variance of the block output representation would approximately equal two times the variance of the input representation:

$$\text{Var}(\mathcal{F}^j) \approx 2 \text{Var}(\mathcal{F}^{j-1}(\mathbf{x}))\tag{5.6}$$

The variance thus doubles at each block and exponentially increases with an increasing number of blocks. A general solution to address this problem is to introduce normalization layers like BatchNorm [158] inside the block. However, when AdaSkips are introduced, the variance increases even more rapidly since we add features from all preceding blocks, linearly increasing the variance with each additional connection. To address this variance, we add a small number of batch-normalization layers as discussed below.

Hypernetworks and Batch-normalization for AdaSkips: We now describe the design of the hypernetwork along with the procedure followed to introduce additional batch-norm layers. Any standard contemporary DNN model developed by stacking blocks of layers can be segregated into a set of modules, which we denote as \mathcal{F}^j , with each module containing varying number of blocks. The input \mathbf{x} is passed through these modules in a feedforward manner. When AdaSkips are introduced, before passing the input \mathbf{x} to module \mathcal{F}^j , we first pass it through the hypernetwork \mathcal{H} and obtain the weights λ_i for each of the skip connections in that module. Once the skip connection weights λ_i are obtained, we pass the input \mathbf{x} to the module \mathcal{F}^j , along with

the mixing coefficients i.e. skip connection weights λ_i . If a module contains k blocks, we introduce a total $k(k + 1)/2$ skip connections in the module and consequently the hypernetwork outputs the weight for each of the $k(k + 1)/2$ coefficients. As discussed earlier, in order to curtail the variance introduced by adding new skip connections, at the end of each module, we add one batch-normalization layer which helps alleviate the variance introduced by adding outputs from all possible skip connections. Considering that we need only weights of skip connections (and not all parameters of the network), our hypernetwork consists of a simple two-layer CNN with ReLU activations, normalization layers, and a final fully-connected layer to determine the weights for skip connections. The output of the hypernetwork is constrained between $[0, 1]$ using a sigmoid activation.

5.4 Experiments and Results

In this section, we compare our methodology with six different transfer learning methods, including state-of-the-art approaches: LWF [141], BSS [142], DELTA [143], StochNorm [140], Co-Tuning [137], and Bi-Tuning [138]. We also compare with a vanilla baseline, where the backbone architecture is finetuned end-to-end on the target dataset. We study our empirical results on the standard TL benchmark provided by the library in [3], comprising FGVC-Aircraft [159] and Stanford-Cars [160] datasets, which are commonly reported across all the above baselines. To test the effectiveness of our approach in the low-data regime, we train the model on 15%, 30%, 50% and 100% of the training data for both the above datasets. Going beyond, we also study the effectiveness of our approach on more complex tasks including zero-shot domain adaptation (later in this section), as well as object detection and segmentation (in the Appendix).

Results of TL Performance with Existing Methods. Table 5.1 presents the results of our method against the abovementioned baseline methods. In order to show that AdaSkips can be easily integrated with existing TL approaches, we also add AdaSkips to the state-of-the-art Bi-Tuning [138] method and show its results. We follow the standard benchmark [3] training protocol with ResNet-50 as our backbone architecture, similar to all recent methods in this direction [140, 137, 138]. We emphasize that no modifications are made to the training protocols of the benchmark for fair comparison. As shown in Table 5.1, AdaSkips outperforms existing approaches on almost all experiments consistently, even with varying amounts of training data in the target domain. Bi-Tuning [138] uses contrastive learning and

Method	FGVC-Aircraft					Stanford-Cars				
	15%	30%	50%	100%	Avg	15%	30%	50%	100%	Avg
Vanilla Baseline	41.6	57.8	68.7	80.2	62.1	41.1	65.9	78.4	87.8	68.3
+ LWF	44.1	60.6	68.7	82.4	64.0	44.9	67.0	77.6	87.5	69.3
+ BSS	43.6	59.5	69.6	81.2	63.5	43.3	67.6	79.6	88.0	69.6
+ DELTA	44.4	61.9	71.4	82.7	65.1	45.0	68.4	79.6	88.4	70.4
+ Co-Tuning	45.9	61.2	71.3	82.2	65.2	49.0	70.6	81.9	89.1	72.7
+ StochNorm	44.3	60.6	70.1	81.5	64.1	44.4	68.1	79.1	87.9	69.9
+ Bi-Tuning	47.2	64.3	73.7	84.3	67.4	48.3	72.8	83.3	90.2	73.7
+ AdaSkips	46.9	64.8	74.3	85.3	67.8	48.5	73.8	82.5	90.4	73.8
+ Bi-Tuning + AdaSkips	47.7	65.1	76.5	87.1	69.1	49.2	73.9	84.3	91.1	74.6

Table 5.1: Comparison of performance of AdaSkips with existing methods on standard TL benchmark [3], including in low-data regimes with limited target data. (E.g, 50% denotes availability of 50% of target dataset for finetuning). **Note that AdaSkips can not only outperform existing TL approaches, but can also improve performance of existing TL approaches (see row for Bi-Tuning + AdaSkips)**

leverages both supervised and unsupervised pre-trained representations, which perhaps makes it stronger compared to other baselines. AdaSkips instead uses a simple strategy to outperform all baselines including Bi-Tuning on most settings. Moreover, as shown in the last row, adding AdaSkips to state-of-the-art methods like Bi-Tuning improves the performance significantly.

Results on Zero-shot Domain Adaptation: To further evaluate the utility of AdaSkips for learning transfer learning, we test AdaSkips on Zero-shot domain adaptation (ZSDA) for digit classification. In ZSDA, the models are trained on a given source dataset and evaluated on their ability to transfer to a different target dataset that has the same classes but is from a different data distribution. We follow the same experimental setting as Uniform Prior [161], and evaluate AdaSkips using a ResNet-18 architecture. We consider three datasets, MNIST [162], SVHN [163], USPS [164] all of which have the same classes (0-9 digits). Additionally, we also experiment with adding both the Uniform Prior [161] and AdaSkips to investigate the ability of AdaSkips to work synergistically with other transfer learning methods. The results for ZSDA are shown in Table 5.2.

We continue to observe that by simply adding AdaSkips, we significantly improve the model’s performance on the target dataset. Moreover, AdaSkips also works well with existing ZSDA methods such as Uniform Prior [161], and Adversarial Discriminative Domain Adaptation [165], showing how it is a general solution that can be applied alongside existing techniques.

Table 5.2: **AdaSkips improves performance of existing ZSDA methods.**
 Comparison of introducing AdaSkips for Zero-Shot domain adaptation using a base ResNet-18 network

ResNet-18	MNIST → USPS	USPS → MNIST	SVHN→MNIST
Vanilla	49.0 ± 0.2	42.8 ± 0.1	69.7 ± 0.1
+ AdaSkips	56.5 ± 0.7	47.0 ± 0.2	75.3 ± 0.2
+ Uniform prior [161]	67.2 ± 0.1	56.2 ± 0.1	71.3 ± 0.1
+ Uniform prior + AdaSkips	71.3 ± 0.3	61.0 ± 0.2	77.9 ± 0.2
ADDA [165]	88.2 ± 0.1	89.0 ± 0.1	73.4 ± 0.2
+ AdaSkips	90.1 ± 0.3	90.7 ± 0.2	81.1 ± 0.5
+ Uniform Prior	91.6 ± 0.1	92.7 ± 0.3	79.4 ± 0.1
+ Uniform Prior + AdaSkips	92.4 ± 0.4	94.1 ± 0.1	83.6 ± 0.2
Target only	98.1 ± 0.2	99.8 ± 0.1	99.8 ± 0.1

Time and Storage Complexity: In terms of parameters, models with AdaSkips have slightly more parameters because of the hyper-network used for input conditioning (a quantitative overview of flop and parameter counts is provided in Section 5.5.2). This translates to a marginal increase in training time of the models (models with AdaSkips take approximately 1.1-1.2x more training time than the vanilla baseline). There is negligible difference between baseline models and models with AdaSkips in terms of inference time though. To study if providing this additional storage to the baseline model would counter the improvement in performance, we point to a result in Table 5.3 in Sec 5.5, where we show ResNet-18+AdapSkips outperforms ResNet-34 by a margin of 10%. This highlights the usefulness of the proposed method and that the minimal cost overhead may be valuable.

5.5 Analysis and Ablation Studies

The proposed methodology includes various components like input-conditioned routing, using all the skip connections, introducing a batch normalization layer at the end of each module, using a hypernetwork to learn the weights for skip connections. In this section, we understand the contribution of each component and analyze the learned lambda values.

5.5.1 Does Routing alone help?

To understand if input-conditioned routing can work alone in a standalone manner, we test the performance of architectures with AdaSkips in the standard and most

commonly used [166, 167, 168] transfer learning setting of fine-tuning just the last fully connected layer with the backbone network frozen. We use four architectures and five different image classification data sets.

First, we consider datasets that have minimal semantic relationship, i.e., the source and target datasets have minimal class overlap, which is a practical transfer learning setting. Second, we consider datasets that have a high semantic relationship (similar classes in the source and target datasets). Dividing the commonly used transfer learning datasets into two bins and analyzing the performance separately will give us a better picture of where AdaSkips are helpful. For all the experiments in this section we use pre-trained weights of models trained on the ImageNet [169] dataset, we directly use the weights provided by PyTorch[170]. Please note that the pre-trained backbone and the experimental setting used in the current section and Table 5.1, Sec. 5.4 are completely different, this is the primary reason for significant difference in numbers.

Target and Source Datasets with Minimal Semantic Relationship:

In this section, we study the effect of AdaSkips when there is minimum or no class overlap between the source and target datasets. The transfer learning datasets we consider in this section are, FGVC-Aircraft [159], Food-101 [171] and Stanford-Cars [160], these datasets have very minimum or no class overlap with ImageNet dataset. This setting is challenging and practically useful since there is very minimal or zero overlap in the classes of source and target datasets. Moreover, all the datasets considered in this section are fine-grained classification datasets, which are comparatively difficult to standard classification datasets.

Results for this section are qualitatively and quantitatively shown in Fig 5.2 and first three columns of Table 5.3, respectively. For models with AdaSkips, we introduce all the possible skip connections and train the hypernetwork, which learns how to combine the pre-trained features in an input conditioned manner and compare directly to finetuning. None of the pre-trained weights are modified when training the hypernetwork. We observe a significant improvement in classification accuracy of up to 26% using AdaSkips, compared to the baseline architecture.

Target and Source Datasets with High Semantic Relationship: Similarly, we extend the experiments to using target datasets that have high class overlap with the source dataset i.e., ImageNet. Stanford-Dogs [8] and Oxford-Flowers [172] are classic examples as ImageNet contains a wide variety of flower and dog species. To understand if better feature interaction using AdaSkips will be helpful in cases like these where there is significant overlap in the classes, we test AdaSkips on these two

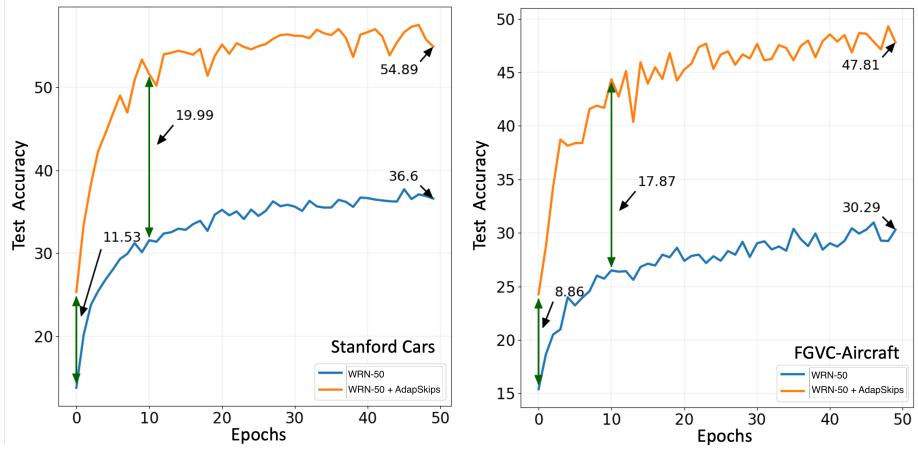


Figure 5.2: **Models with AdaSkips start better and improve with training;** Qualitative comparison of validation accuracy of WideResNet-50 on datasets that have minimal class overlap with ImageNet

datasets. We follow the same experimental procedure as before; the results are shown in the fourth, fifth columns of Table 5.3. As one may expect, the relative improvement of using AdaSkips is smaller as pre-trained features already have enough information about the classes/objects in the target dataset since there is a high semantic relationship between the source and target datasets.

Table 5.3: Comparison of different architectures on five fine-grained classification datasets. **AdaSkips improve performance significantly when the semantic relationship between source and target datasets is minimal**

Source Dataset: ImageNet	Minimal semantic relation between source and target datasets			High semantic relation between source and target datasets	
Architecture	FGVC-Aircraft	Food-101	Stanford-Cars	Flowers-102	Stanford-Dogs
ResNet-18	32.79 ± 1.1	41.43 ± 1.5	39.07 ± 0.4	92.27 ± 0.6	74.02 ± 0.6
+ AdaSkips	43.12 ± 0.6	62.16 ± 0.5	48.77 ± 0.2	94.47 ± 0.4	79.01 ± 0.1
ResNet-34	33.92 ± 0.1	40.69 ± 2.8	39.49 ± 0.9	93.00 ± 0.7	80.68 ± 0.8
+ AdaSkips	46.19 ± 0.4	66.76 ± 0.0	53.67 ± 2.0	96.17 ± 0.1	82.85 ± 0.4
ResNeXt-50	32.66 ± 1.6	55.37 ± 0.3	41.20 ± 0.6	93.61 ± 0.8	87.30 ± 0.9
+ AdaSkips	50.22 ± 0.8	68.71 ± 3.0	62.47 ± 0.2	96.26 ± 0.7	85.94 ± 1.8
WideResNet50	30.29 ± 0.9	52.12 ± 0.7	36.60 ± 0.3	91.53 ± 0.2	87.81 ± 0.4
+ AdaSkips	47.81 ± 0.7	69.07 ± 1.3	54.89 ± 0.6	96.05 ± 0.1	88.11 ± 0.3

5.5.2 Flop and Parameter Count Statistics

Table 5.4 shows a comparison of parameters and flops between standard models and models with AdaSkips. Since models with AdaSkips use four 2-layer hyper-networks

and 4 additional batch-norm layers, they have more parameters. Due to the use of all possible skip connections, the flops are higher. Our current implementation does not leverage sparsity in outputs of the hyper-network, which can decrease the number of flops significantly. We believe that this cost overhead may not be significant when compared to the improvement in results. Regarding the increase in parameter count, the input and output dimensions of the layers in the hyper-network depend on the base network. As the input dimension of the base network increases, the number of parameters increases. This is the primary reason for significant increase in parameters for ResNeXt-50 and WideResNet-50 models.

Table 5.4: Qualitative comparison of Parameters and Flops for standard models and models with AdaSkips on FGVC-Aircraft

Architecture	Flops (G)	Parameters (M)
ResNet-18	1.82	11.23
+ AdaSkips	4.61	16.10
ResNet-34	3.68	21.34
+ AdaSkips	6.47	26.22
ResNeXt-50	4.30	23.18
+ AdaSkips	38.33	97.79
WRN-50	11.50	67.10
+ AdaSkips	45.49	141.6

We would like to re-emphasise that the improvement in performance by using AdaSkips cannot be attributed to parameters alone. For example, there is a significant increase in the number of parameters from **ResNet-18** to **ResNet-34** but that does not result by itself in performance improvement in Tables 5.3, 5.5. Carefully selecting hyper-network layer dimensions in manner specific to the base network and leveraging the hyper-network’s sparsity can decrease the parameters and flops reasonably.

5.5.3 Fine-tuning in Limited Data settings

In many real-world applications of transfer learning, the amount of labeled data available is limited, which makes it important to test the effectiveness of AdaSkips in such settings. Towards this, we create a subset of FGVC-Aircraft [159] and Stanford-Cars [160] datasets with 50% and 25% of the dataset available while training. We randomly sample the subset for each dataset, maintaining the same class distribution. The experimental results for this section are shown in Fig 5.3, where we see

that the model continues to outperform vanilla finetuning over two standard network architectures.

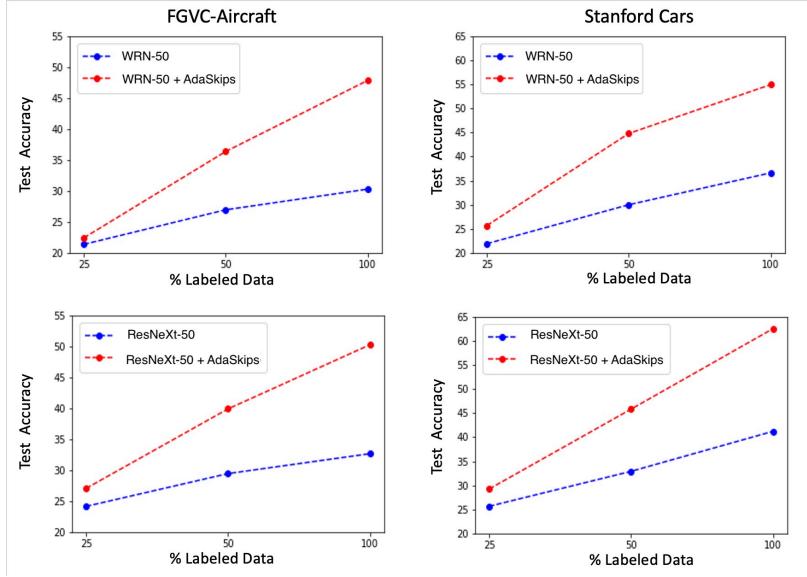


Figure 5.3: Models with AdaSkips are better even when the labelled data is limited. Qualitative visualisation to show the effectiveness of AdaSkips with varying amounts of labelled data

5.5.4 Study of different components involved in AdaSkips

In this section, we understand the contribution of each component involved in AdaSkips and analyze the learned lambda values. Results for all the analyses in the sub-section are shown in Table 5.5.

Effect of having all skip connections: In contrast to the idea of one skip connection per a block of a layers, we introduce all the possible skip connections in a feed-forward manner and sum the output of all skip connections towards the end of each block. Introducing skip connections in this manner may help in long-range interaction between the features. As shown in Table 5.5, introducing these skip connections in a naive way makes the models perform poorly due to the high variance in the output.

Effect of BatchNorm: In order to control the variance introduced by the use of all possible skip connections, we add a batch-normalisation layer at the end of each module. From the results shown in Table 5.5, it is clear that batch-norm significantly helps in dealing with the variance, but the hypernetwork is still required for input conditioned routing.

Effect of HyperNetwork: Finally, as shown in Table 5.5, the use of a hypernetwork increases performance by finding skip connection paths based on the input itself.

Table 5.5: **HyperNetwork and Batchnorm both play a key role in improving the performance.** Quantitative ablation analysis of marginal contribution of each component in AdaSkips.

Architecture	FGVC-Aircraft	Stanford-Cars
ResNet-18	32.79 ± 1.1	39.07 ± 0.4
+ All Skips	29.71 ± 1.2	34.26 ± 0.1
+ All Skips + BN	42.63 ± 1.1	46.60 ± 0.3
+ All Skips + BN + Random	10.32 ± 4.1	11.01 ± 0.5
+ All Skips + BN + HyperNW	43.12 ± 0.6	48.77 ± 0.2
ResNet-34	33.92 ± 0.1	39.49 ± 0.9
+ All Skips	24.02 ± 0.1	22.84 ± 0.3
+ All Skips + BN	38.36 ± 0.7	38.80 ± 0.2
+ All Skips + BN + Random	2.90 ± 1.5	2.00 ± 0.3
+ All Skips + BN + HyperNW	46.19 ± 0.4	53.67 ± 2.0
ResNeXt-50	32.66 ± 1.6	41.20 ± 0.6
+ All Skips	26.72 ± 1.7	27.32 ± 0.8
+ All Skips + BN	44.66 ± 0.7	50.43 ± 0.4
+ All Skips + BN + Random	6.93 ± 1.7	7.30 ± 2.1
+ All Skips + BN + HyperNW	50.22 ± 0.8	62.47 ± 0.2
WRN-50	30.29 ± 0.9	36.60 ± 0.3
+ All Skips	25.81 ± 0.7	25.93 ± 0.7
+ All Skips + BN	43.91 ± 0.6	49.80 ± 0.1
+ All Skips + BN + Random	4.07 ± 0.7	4.16 ± 0.7
+ All Skips + BN + HyperNW	47.81 ± 0.7	54.89 ± 0.6

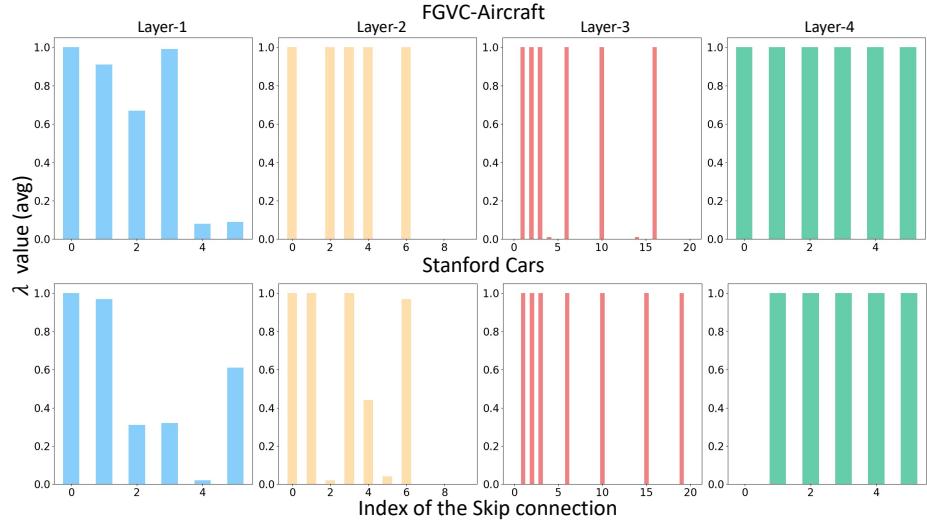


Figure 5.4: **Despite not imposing a specific sparsity schema, the hypernetwork learns to select input-conditioned skip connections.** Qualitative visualisation of the weights (average) learned by the hypernetwork for all the test samples of FGVC-Aircraft and Stanford Cars datasets, we use ResNeXt-50 as the base network.

Replacing the hypernetwork with randomly predicted values between $[0, 1]$ reduces the performance drastically. To see if the hypernetwork is using all skip connections,

we visualize the weights of these skip connections in Fig 5.4. We take the average value of weights for all the test samples in FGVC-aircraft and Stanford-Dogs datasets. Even without imposing any specific sparsity schemes such as L1-penalty, some of the skip connection weights are zeros, which shows how the hypernetwork learns to combine features better than finetuning algorithms for feature reuse.

5.5.5 AdaSkips for Standard Image Classification

To test the effectiveness of AdaSkips on commonly used image classification benchmarks, we ran experiments on CIFAR-10, CIFAR-100 and ImageNet datasets. Results are shown in Table 5.6. AdaSkips once again shows improvement on these benchmarks with no pre-training (without significant hyperparameter tuning). (For the Imagenet dataset, we show results only with ResNet-18 architecture due to the significant time and computation requirements.)

Table 5.6: AdaSkips can help improve performance on common image classification benchmark datasets without pre-training. Comparison of different architectures when trained from scratch without any pre-training on standard image classification datasets

Architecture	CIFAR-10	CIFAR-100	ImageNet (Top-5)
ResNet-18	85.21	50.43	85.87
+ AdaSkips	85.47	52.17	86.16
ResNet-34	81.25	48.05	-
+ AdaSkips	84.76	51.77	-
ResNeXt-50	75.27	40.92	-
+ AdaSkips	80.42	41.90	-

5.6 Conclusion

In this chapter, we have shown how skip connections can be effectively leveraged to transfer pre-trained features across completely different data sets. We introduce AdaSkips, that can be directly used with any existing pre-trained models as a plug-and-play without requiring any retraining on the source dataset. We demonstrated the effectiveness of our method on a wide range of tasks and our through ablation and analysis shows that input conditioned routing and weighing of pre-trained features can be a very useful and potential direction for improving transfer learning. Future follow-up work may extend this work to compositional representation learning by studying how modularity can be imposed as an inductive bias to the model that will further enhance the re-usability of the learned features.

Chapter 6

Conclusion and Future Directions

The problem of making deep learning models more data-efficient is studied in this thesis. We primarily present the data efficiency of neural networks from the perspectives of modality, data annotation, adversarial robustness, and transfer learning. Chapter 2 provided a brief background on multi-modal object detection and proposes a pseudo-multi-modal framework that leverages the pre-trained RGB models to improve object detection in thermal images. Just with 1/4th of the training data, the proposed method beats the SoTA performance on a standard large-scale thermal imagery dataset. Chapter 3 primarily focuses on understanding the role of the initial pool in the commonly used active learning framework. It explores and answers a very interesting and important question of whether the active learning cycle can benefit from an intelligently sampled initial pool; we conclude the chapter with our finding that a randomly picked initial pool generally outperforms many existing methods and VAE-based sampling slightly performs better than random sampling. Chapter 4 discusses adversarial robustness from an architecture perspective. It answers four important questions that help us understand the role of network topology in improving adversarial robustness. We compare and contrast the adversarial robustness of various hand-crafted and NAS based architectures. While NAS-based architectures achieve slightly better accuracy than hand-crafted models on un-perturbed test sets, they are significantly more vulnerable than hand-crafted models for adversarial attacks. Chapter 5 discuss how a simple and most commonly used inductive bias like skip connection can be leveraged to improve transfer learning. It proposes AdaSkips, an input-conditioned skip connection that changes the way data is routed inside a neural network; it performs better than existing transfer learning approaches and shows promising results on tasks like object detection, segmentation, and zero-shot domain adaptation. In terms of future work, we plan to extend the current work

and present a holistic view of data efficiency by exploring data efficiency from other perspectives.

References

- [1] F. A. Group. FLIR Thermal Dataset for Algorithm Training. <https://www.flir.in/oem/adas/adas-dataset-form/> . x, xii, 11, 12, 13, 15, 17
- [2] S. Hwang, J. Park, N. Kim, Y. Choi, and I. S. Kweon. Multispectral pedestrian detection: Benchmark dataset and baseline. In 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2015 1037–1045. x, xii, 6, 11, 12, 13, 16
- [3] J. Junguang, B. Chen, F. Bo, and M. Long. Transfer-Learning-library. <https://github.com/thuml/Transfer-Learning-Library> 2020. xi, 62, 63
- [4] M. Liu, T. Breuel, and J. Kautz. Unsupervised Image-to-Image Translation Networks. *CoRR* abs/1703.00848. xii, 8, 9, 10, 13, 14, 15
- [5] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. In Computer Vision (ICCV), 2017 IEEE International Conference on. 2017 . xii, 8, 9, 13, 14, 15
- [6] D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. In SIGIR’94. Springer, 1994 3–12. xii, 21, 25, 27
- [7] C. E. Shannon and W. Weaver. A Mathematical Theory of Communication. University of Illinois Press, USA, 1963. xii, 25, 27
- [8] A. Khosla, N. Jayadevaprakash, B. Yao, and L. Fei-Fei. Novel Dataset for Fine-Grained Image Categorization. In First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition. Colorado Springs, CO, 2011 . xiv, 55, 65
- [9] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In Proceedings of the 25th International

Conference on Neural Information Processing Systems - Volume 1, NIPS'12.
Curran Associates Inc., Red Hook, NY, USA, 2012 1097–1105. [2](#)

- [10] R. Socher, Y. Bengio, and C. D. Manning. Deep Learning for NLP (without Magic). In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts. Association for Computational Linguistics, Jeju Island, Korea, 2012 5. [2](#)
- [11] D. Amodei, S. Ananthanarayanan, Anubhai et al. Deep Speech 2 : End-to-End Speech Recognition in English and Mandarin. In M. F. Balcan and K. Q. Weinberger, eds., Proceedings of The 33rd International Conference on Machine Learning, volume 48 of *Proceedings of Machine Learning Research*. PMLR, New York, New York, USA, 2016 173–182. [2](#)
- [12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds., Advances in Neural Information Processing Systems 25, 1097–1105. Curran Associates, Inc., 2012. [2](#)
- [13] NTSB. PRELIMINARY REPORT HIGHWAY HWY18MH010 2018. [3](#)
- [14] R. Retting and S. Schwatz. Governors Highway Safety Association Pedestrian Traffic Fatalities by State (2017 Preliminary Data). <https://www.ghsa.org/sites/default/files/2018-02/pedestrians18.pdf> . [3](#)
- [15] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR* abs/1311.2524. [3](#)
- [16] R. B. Girshick. Fast R-CNN. *CoRR* abs/1504.08083. [3, 10](#)
- [17] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *CoRR* abs/1506.01497. [3, 13](#)
- [18] J. Redmon and A. Farhadi. YOLOv3: An Incremental Improvement. *CoRR* abs/1804.02767. [3, 7](#)
- [19] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In CVPR09. 2009 . [3, 41, 42](#)

- [20] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. <http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html>. 3, 11, 12, 13
- [21] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common Objects in Context. *CoRR* abs/1405.0312. 3, 11, 12
- [22] A. L. Bearman, O. Russakovsky, V. Ferrari, and L. Fei-Fei. What’s the Point: Semantic Segmentation with Point Supervision. In ECCV. 2016 . 3
- [23] Y. Gal, R. Islam, and Z. Ghahramani. Deep bayesian active learning with image data. In Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR. org, 2017 1183–1192. 4, 21, 27
- [24] O. Sener and S. Savarese. Active Learning for Convolutional Neural Networks: A Core-Set Approach. In International Conference on Learning Representations. 2018 . 4, 21, 26, 27
- [25] W. H. Beluch, T. Genewein, A. Nürnberg, and J. M. Köhler. The Power of Ensembles for Active Learning in Image Classification. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* 9368–9377. 4, 21, 27
- [26] D. Yoo and I. S. Kweon. Learning Loss for Active Learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019 93–102. 4, 21, 23
- [27] S. Sinha, S. Ebrahimi, and T. Darrell. Variational Adversarial Active Learning. *arXiv preprint arXiv:1904.00370* . 4, 21, 26, 27
- [28] A. Mottaghi and S. Yeung. Adversarial Representation Active Learning. *ArXiv* abs/1912.09720. 4, 23
- [29] B. Settles. Active learning literature survey. Technical Report, University of Wisconsin-Madison Department of Computer Sciences 2009. 4
- [30] B. Zoph and Q. V. Le. Neural Architecture Search with Reinforcement Learning. *arXiv e-prints* arXiv:1611.01578. 4, 40

- [31] S. Yan, B. Fang, F. Zhang, Y. Zheng, X. Zeng, H. Xu, and M. Zhang. HM-NAS: Efficient Neural Architecture Search via Hierarchical Masking. *arXiv e-prints* arXiv:1909.00122. 4
- [32] X. Chen, L. Xie, J. Wu, and Q. Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In Proceedings of the IEEE International Conference on Computer Vision. 2019 1294–1303. 4
- [33] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean. Efficient Neural Architecture Search via Parameter Sharing. *arXiv e-prints* arXiv:1802.03268. 4, 40
- [34] M. Tan and Q. V. Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *arXiv e-prints* arXiv:1905.11946. 4, 45
- [35] M. Tan, R. Pang, and Q. V. Le. EfficientDet: Scalable and Efficient Object Detection. *arXiv e-prints* arXiv:1911.09070. 4
- [36] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. Berkay Celik, and A. Swami. Practical Black-Box Attacks against Machine Learning. *arXiv e-prints* arXiv:1602.02697. 5
- [37] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016 . 5, 57, 58
- [38] R. K. Srivastava, K. Greff, and J. Schmidhuber. Highway Networks. *arXiv e-prints* arXiv:1505.00387. 5
- [39] X. Ding, X. Zhang, N. Ma, J. Han, G. Ding, and J. Sun. Repvgg: Making vgg-style convnets great again. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021 13,733–13,742. 5
- [40] T. T. Zin, H. Takahashi, and H. Hama. Robust Person Detection using Far Infrared Camera for Image Fusion. In Second International Conference on Innovative Computing, Information and Control (ICICIC 2007). 2007 310–310. 6
- [41] T. P. Breckon, A. Gaszczak, J. Han, M. L. Eichner, and S. E. Barnes. Multi-modal target detection for autonomous wide area search and surveillance 2013. 6

- [42] S. Mangale and M. Khambete. Moving Object Detection using Visible Spectrum Imaging and Thermal Imaging. In 2015 International Conference on Industrial Instrumentation and Control (ICIC). 2015 590–593. [6](#)
- [43] S. Liu and Z. Liu. Multi-Channel CNN-based Object Detection for Enhanced Situation Awareness. *CoRR* abs/1712.00075. [6](#)
- [44] M. Bertozzi, A. Broggi, C. Hilario, R. Fedriga, G. Vezzoni, and M. Del Rose. Pedestrian Detection in Far Infrared Images based on the use of Probabilistic Templates. 2007 327 – 332. [7](#)
- [45] J. W. Davis and M. A. Keck. A Two-Stage Template Approach to Person Detection in Thermal Imagery. In 2005 Seventh IEEE Workshops on Applications of Computer Vision (WACV/MOTION'05) - Volume 1, volume 1. 2005 364–369. [7](#)
- [46] K. Jüngling and M. Arens. Feature based person detection beyond the visible spectrum. *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops* 30–37. [7](#)
- [47] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded up robust features. volume 3951. 2006 404–417. [7](#)
- [48] M. Peng, C. Wang, T. Chen, and G. Liu. NIRFaceNet: A Convolutional Neural Network for Near-Infrared Face Identification. *Information* 7. [7](#)
- [49] E. J. Lee, B. C. Ko, and J.-Y. Nam. Recognizing pedestrian's unsafe behaviors in far-infrared imagery at night. *Infrared Physics and Technology* 76, (2016) 261 – 270. [7](#)
- [50] M. CHEVALIER, N. Thome, M. Cord, J. Fournier, G. Henaff, and E. Dusch. LOW RESOLUTION CONVOLUTIONAL NEURAL NETWORK FOR AUTOMATIC TARGET RECOGNITION. In 7th International Symposium on Optronics in Defence and Security. Paris, France, 2016 . [7](#)
- [51] I. Rodger, B. Connor, and N. Robertson. Classifying objects in lwir imagery via cnns. In In Proc. SPIE: Electro-Optical and Infrared Systems: Technology and Applications XII. 2016 . [7](#)
- [52] R. Abbott, J. Del Rincon, B. Connor, and N. Robertson. Deep object classification in low resolution LWIR imagery via transfer learning. In Proceedings of the 5th IMA Conference on Mathematics in Defence. 2017 . [7](#)

- [53] A. Berg, K. Öfjäll, J. Ahlberg, and M. Felsberg. Detecting Rails and Obstacles Using a Train-Mounted Thermal Camera. In R. R. Paulsen and K. S. Pedersen, eds., *Image Analysis*. Springer International Publishing, Cham, 2015 . 7
- [54] A. Berg. Detection and Tracking in Thermal Infrared Imagery. *Linköping Studies in Science and Technology*. Thesis No. 1744, Linköping University, Sweden 2016. 7
- [55] A. Leykin, Y. Ran, and R. Hammoud. Thermal-Visible Video Fusion for Moving Target Tracking and Pedestrian Classification. 2007 . 8
- [56] J. Wagner, V. Fischer, M. Herman, and S. Behnke. Multispectral Pedestrian Detection using Deep Fusion Convolutional Neural Networks. 2016 . 8
- [57] H. Choiand S. Kim, , and K. Sohn. Multi-spectral pedestrian detection based on accumulated object proposal with fully convolutional networks. In 2016 23rd International Conference on Pattern Recognition (ICPR). 2016 621–626. 8
- [58] D. König, M. Adam, C. Jarvers, G. Layher, H. Neumann, and M. Teutsch. Fully Convolutional Region Proposal Networks for Multispectral Person Detection. In 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). 2017 243–250. 8
- [59] J. Liu, S. Zhang, S. Wang, and D. N. Metaxas. Multispectral Deep Neural Networks for Pedestrian Detection. *CoRR* abs/1611.02644. 8
- [60] Z. Yi, H. Zhang, P. Tan, and M. Gong. DualGAN: Unsupervised Dual Learning for Image-to-Image Translation. *CoRR* abs/1704.02510. 8
- [61] Y. Choi, M. Choi, M. Kim, J. Ha, S. Kim, and J. Choo. StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation. *CoRR* abs/1711.09020. 8
- [62] Y. Taigman, A. Polyak, and L. Wolf. Unsupervised Cross-Domain Image Generation. *CoRR* abs/1611.02200. 8
- [63] S. Mo, M. Cho, and J. Shin. InstaGAN: Instance-aware Image-to-Image Translation. *CoRR* abs/1812.10889. 8
- [64] H. Lee, H. Tseng, J. Huang, M. K. Singh, and M. Yang. Diverse Image-to-Image Translation via Disentangled Representations. *CoRR* abs/1808.00948. 8

- [65] S. Zhang, L. Wen, X. Bian, Z. Lei, and S. Z. Li. Single-Shot Refinement Neural Network for Object Detection. In CVPR. 2018 . [13](#), [16](#)
- [66] J. Yang, J. Lu, D. Batra, and D. Parikh. A Faster Pytorch Implementation of Faster R-CNN. <https://github.com/jwyang/faster-rcnn.pytorch> . [14](#)
- [67] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-Image Translation with Conditional Adversarial Networks. In Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on. 2017 . [14](#)
- [68] P. Munjal, N. Hayat, M. Hayat, J. Sourati, and S. Khan. Towards Robust and Reproducible Active Learning Using Neural Networks. *ArXiv* abs/2002.09564. [21](#), [23](#), [27](#), [32](#)
- [69] S. Mittal, M. Tatarchenko, Ö. Çiçek, and T. Brox. Parting with Illusions about Deep Active Learning. *ArXiv* abs/1912.05361. [21](#), [23](#)
- [70] D. Lowell, Z. C. Lipton, and B. C. Wallace. Practical Obstacles to Deploying Active Learning. In EMNLP/IJCNLP. 2019 . [21](#), [22](#), [32](#)
- [71] T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton. A Simple Framework for Contrastive Learning of Visual Representations. *ArXiv* abs/2002.05709. [21](#), [25](#), [29](#), [36](#)
- [72] S. Gidaris, P. Singh, and N. Komodakis. Unsupervised Representation Learning by Predicting Image Rotations. *ArXiv* abs/1803.07728. [21](#), [25](#), [26](#)
- [73] M. Caron, P. Bojanowski, A. Joulin, and M. Douze. Deep Clustering for Unsupervised Learning of Visual Features. In ECCV. 2018 . [21](#), [25](#), [26](#)
- [74] D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, and A. A. Efros. Context Encoders: Feature Learning by Inpainting. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* 2536–2544. [21](#), [25](#)
- [75] M. Noroozi, H. Pirsiavash, and P. Favaro. Representation Learning by Learning to Count. *2017 IEEE International Conference on Computer Vision (ICCV)* 5899–5907. [21](#), [25](#)
- [76] M. Noroozi and P. Favaro. Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles. In ECCV. 2016 . [21](#), [25](#)

- [77] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised Visual Representation Learning by Context Prediction. *2015 IEEE International Conference on Computer Vision (ICCV)* 1422–1430. [21](#), [25](#)
- [78] O. Siméoni, M. Budnik, Y. Avrithis, and G. Gravier. Rethinking deep active learning: Using unlabeled data at model training. *ArXiv* abs/1911.08177. [23](#)
- [79] D. Mishkin and J. Matas. All you need is a good init. *CoRR* abs/1511.06422. [23](#)
- [80] K. He, X. Zhang, S. Ren, and J. Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), ICCV ’15. IEEE Computer Society, USA, 2015 1026–1034. [23](#)
- [81] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In Y. W. Teh and M. Titterington, eds., Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, volume 9 of *Proceedings of Machine Learning Research*. PMLR, Chia Laguna Resort, Sardinia, Italy, 2010 249–256. [23](#)
- [82] J. Kang, K. Ryu, and H.-C. Kwon. Using Cluster-Based Sampling to Select Initial Training Set for Active Learning in Text Classification. 2004 384–388. [23](#)
- [83] R. Hu, B. M. Namee, and S. J. Delany. Off to a Good Start: Using Clustering to Select the Initial Training Set in Active Learning. In FLAIRS Conference. 2010 . [23](#)
- [84] B. Settles, M. Craven, and S. Ray. Multiple-Instance Active Learning. In Proceedings of the 20th International Conference on Neural Information Processing Systems, NIPS’07. Curran Associates Inc., Red Hook, NY, USA, 2007 1289–1296. [23](#)
- [85] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. *CoRR* abs/1312.6114. [26](#)
- [86] S. Dasgupta. Two faces of active learning. *Theor. Comput. Sci.* 412, (2011) 1767–1781. [26](#)

- [87] A. Bondu, V. Lemaire, and M. Boullé. Exploration vs. exploitation in active learning : A Bayesian approach. *The 2010 International Joint Conference on Neural Networks (IJCNN)* 1–7. [26](#)
- [88] T. Scheffer, C. Decomain, and S. Wrobel. Active Hidden Markov Models for Information Extraction 2001. [27](#)
- [89] Y. Le and X. Yang. Tiny ImageNet Visual Recognition Challenge. 2015 . [27](#)
- [90] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings. 2015 . [27](#)
- [91] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition. 2016 770–778. [27](#)
- [92] S. Roy, A. Unmesh, and V. P. Namboodiri. Deep active learning for object detection. In BMVC. 2018 . [28](#)
- [93] Y. Cui, M. Jia, T.-Y. Lin, Y. Song, and S. Belongie. Class-Balanced Loss Based on Effective Number of Samples. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* 9260–9269. [28](#), [35](#)
- [94] W. Van Gansbeke, S. Vandenhende, S. Georgoulis, M. Proesmans, and L. Van Gool. Scan: Learning to classify images without labels. In Proceedings of the European Conference on Computer Vision. 2020 . [30](#)
- [95] J.-B. Grill, F. Strub, F. Altch'e, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko. Bootstrap Your Own Latent: A New Approach to Self-Supervised Learning. *ArXiv* abs/2006.07733. [36](#)
- [96] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. *arXiv e-prints* arXiv:1706.06083. [39](#), [40](#), [41](#), [43](#), [49](#)
- [97] C. Xie and A. Yuille. Intriguing properties of adversarial training at scale. *arXiv e-prints* arXiv:1906.03787. [39](#), [41](#)

- [98] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv e-prints* arXiv:1312.6199. **40**
- [99] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and Harnessing Adversarial Examples. *arXiv e-prints* arXiv:1412.6572. **40, 43**
- [100] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial Machine Learning at Scale. *arXiv e-prints* arXiv:1611.01236. **40**
- [101] N. Carlini and D. Wagner. Towards Evaluating the Robustness of Neural Networks. *arXiv e-prints* arXiv:1608.04644. **40**
- [102] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. Berkay Celik, and A. Swami. The Limitations of Deep Learning in Adversarial Settings. *arXiv e-prints* arXiv:1511.07528. **40**
- [103] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. DeepFool: a simple and accurate method to fool deep neural networks. *arXiv e-prints* arXiv:1511.04599. **40**
- [104] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel. Ensemble Adversarial Training: Attacks and Defenses. *arXiv e-prints* arXiv:1705.07204. **40**
- [105] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial examples in the physical world. *arXiv e-prints* arXiv:1607.02533. **40**
- [106] A. Modas, S.-M. Moosavi-Dezfooli, and P. Frossard. SparseFool: A Few Pixels Make a Big Difference. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2019 . **40**
- [107] E. Wong, L. Rice, and J. Zico Kolter. Fast is better than free: Revisiting adversarial training. *arXiv e-prints* arXiv:2001.03994. **40, 43**
- [108] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay. Adversarial Attacks and Defences: A Survey. *arXiv e-prints* arXiv:1810.00069. **40**
- [109] K. Ren, T. Zheng, Z. Qin, and X. Liu. Adversarial Attacks and Defenses in Deep Learning. *Engineering* 6, (2020) 346 – 360. **40**

- [110] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and Harnessing Adversarial Examples. *arXiv e-prints* arXiv:1412.6572. [40](#)
- [111] A. Shafahi, M. Najibi, M. A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein. Adversarial training for free! In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds., Advances in Neural Information Processing Systems 32, 3358–3369. Curran Associates, Inc., 2019. [40](#)
- [112] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le. Regularized evolution for image classifier architecture search. In Proceedings of the aaai conference on artificial intelligence, volume 33. 2019 4780–4789. [40](#)
- [113] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy. Progressive neural architecture search. In Proceedings of the European Conference on Computer Vision (ECCV). 2018 19–34. [40](#)
- [114] H. Liu, K. Simonyan, and Y. Yang. DARTS: Differentiable Architecture Search. *arXiv e-prints* arXiv:1806.09055. [40](#), [42](#), [48](#), [49](#), [52](#)
- [115] X. Chen, L. Xie, J. Wu, and Q. Tian. Progressive Differentiable Architecture Search: Bridging the Depth Gap between Search and Evaluation. *arXiv e-prints* arXiv:1904.12760. [40](#), [42](#), [48](#), [49](#), [52](#)
- [116] Y. Xu, L. Xie, X. Zhang, X. Chen, G.-J. Qi, Q. Tian, and H. Xiong. PC-DARTS: Partial Channel Connections for Memory-Efficient Architecture Search. In International Conference on Learning Representations. 2020 . [41](#), [42](#), [48](#), [49](#), [52](#)
- [117] J. Fang, Y. Sun, Q. Zhang, Y. Li, W. Liu, and X. Wang. Densely Connected Search Space for More Flexible Neural Architecture Search. *arXiv e-prints* arXiv:1906.09607. [41](#), [42](#)
- [118] M. Guo, Y. Yang, R. Xu, and Z. Liu. When NAS Meets Robustness: In Search of Robust Architectures against Adversarial Attacks. *CVPR* . [41](#)
- [119] D. V. Vargas, S. Kotyan, and S. IIIT-NR. Evolving Robust Neural Architectures to Defend from Adversarial Attacks. *arXiv preprint arXiv:1906.11667* . [41](#)

- [120] A. Krizhevsky, V. Nair, and G. Hinton. CIFAR-10 (Canadian Institute for Advanced Research) . 42
- [121] A. Krizhevsky, V. Nair, and G. Hinton. CIFAR-100 (Canadian Institute for Advanced Research) . 42
- [122] M.-E. Nilsback and A. Zisserman. Automated Flower Classification over a Large Number of Classes. In Indian Conference on Computer Vision, Graphics and Image Processing. 2008 . 42
- [123] H. Cai, L. Zhu, and S. Han. ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware. In International Conference on Learning Representations. 2019 . 42, 48
- [124] Z. Lu, I. Whalen, V. Boddeti, Y. Dhebar, K. Deb, E. Goodman, and W. Banzhaf. NSGA-Net: Neural Architecture Search using Multi-Objective Genetic Algorithm. *arXiv e-prints* arXiv:1810.03522. 42, 48, 49
- [125] A. Yang, P. M. Esperança, and F. M. Carlucci. NAS evaluation is frustratingly hard. In International Conference on Learning Representations. 2020 . 42, 51
- [126] T. Pang, K. Xu, C. Du, N. Chen, and J. Zhu. Improving Adversarial Robustness via Promoting Ensemble Diversity. volume 97 of *Proceedings of Machine Learning Research*. PMLR, Long Beach, California, USA, 2019 4970–4979. 42
- [127] F. Croce and M. Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks 2020. 43
- [128] T. Pang, X. Yang, Y. Dong, H. Su, and J. Zhu. Bag of Tricks for Adversarial Training. *arXiv e-prints* arXiv:2010.00467. 43
- [129] J. Fan and W. Li. Adversarial Training and Provable Robustness: A Tale of Two Objectives. *arXiv e-prints* arXiv:2008.06081. 43
- [130] H. Kim. Torchattacks: A Pytorch Repository for Adversarial Attacks. *arXiv preprint arXiv:2010.01950* . 43
- [131] A. Sanyal, P. K. Dokania, V. Kanade, and P. H. S. Torr. How benign is benign overfitting? *arXiv e-prints* arXiv:2007.04028. 44
- [132] C. Xie, M. Tan, B. Gong, A. Yuille, and Q. V. Le. Smooth Adversarial Training. *arXiv e-prints* arXiv:2006.14536. 46

- [133] D. Su, H. Zhang, H. Chen, J. Yi, P.-Y. Chen, and Y. Gao. Is Robustness the Cost of Accuracy? – A Comprehensive Study on the Robustness of 18 Deep Image Classification Models. *arXiv e-prints* arXiv:1808.01688. 49
- [134] W. Ge and Y. Yu. Borrowing Treasures From the Wealthy: Deep Transfer Learning Through Selective Joint Fine-Tuning. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2017 . 54, 56
- [135] Y. Guo, Y. Li, L. Wang, and T. Rosing. AdaFilter: Adaptive Filter Fine-Tuning for Deep Transfer Learning. *Proceedings of the AAAI Conference on Artificial Intelligence* 34, (2020) 4060–4066. 54, 56, 59
- [136] Y. Guo, H. Shi, A. Kumar, K. Grauman, T. Rosing, and R. Feris. Spot-Tune: Transfer Learning Through Adaptive Fine-Tuning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2019 . 54, 56, 57, 59
- [137] K. You, Z. Kou, M. Long, and J. Wang. Co-Tuning for Transfer Learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, eds., Advances in Neural Information Processing Systems, volume 33. Curran Associates, Inc., 2020 17,236–17,246. 54, 56, 59, 62
- [138] J. Zhong, X. Wang, Z. Kou, J. Wang, and M. Long. Bi-tuning of Pre-trained Representations. *arXiv e-prints* arXiv:2011.06182. 54, 56, 59, 62
- [139] L. Xuhong, Y. Grandvalet, and F. Davoine. Explicit inductive bias for transfer learning with convolutional networks. In ICML. 2018 . 54, 56, 59
- [140] Z. Kou, K. You, M. Long, and J. Wang. Stochastic Normalization. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, eds., Advances in Neural Information Processing Systems, volume 33. Curran Associates, Inc., 2020 16,304–16,314. 54, 56, 59, 62
- [141] Z. Li and D. Hoiem. Learning without Forgetting. In ECCV. 2016 . 54, 56, 59, 62
- [142] X. Chen, S. Wang, B. Fu, M. Long, and J. Wang. Catastrophic Forgetting Meets Negative Transfer: Batch Spectral Shrinkage for Safe Transfer Learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, eds., Advances in Neural Information Processing Systems, volume 32. Curran Associates, Inc., 2019 . 54, 56, 59, 62

- [143] X. Li, H. Xiong, H. Wang, Y. Rao, L. Liu, and J. Huan. DELTA: DEEP LEARNING TRANSFER USING FEATURE MAP WITH ATTENTION FOR CONVOLUTIONAL NETWORKS. In International Conference on Learning Representations. 2019 . [54](#), [56](#), [59](#), [62](#)
- [144] D. Ha, A. Dai, and Q. V. Le. HyperNetworks. *arXiv e-prints* arXiv:1609.09106. [55](#), [57](#)
- [145] S. Kornblith, J. Shlens, and Q. V. Le. Do Better ImageNet Models Transfer Better? In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2019 . [56](#)
- [146] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely Connected Convolutional Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2017 . [56](#)
- [147] W. Liu and K. Zeng. SparseNet: A Sparse DenseNet for Image Classification. *arXiv e-prints* arXiv:1804.05340. [56](#)
- [148] J. Wang and Y. Yang. Self-Adaptive Weighted Skip Connections for Image Super-Resolution. In 2020 International Conference on Culture-oriented Science Technology (ICCST). 2020 192–197. [57](#)
- [149] X. Wang, F. Yu, Z.-Y. Dou, T. Darrell, and J. E. Gonzalez. SkipNet: Learning Dynamic Routing in Convolutional Networks. In Proceedings of the European Conference on Computer Vision (ECCV). 2018 . [57](#)
- [150] B. E. Bejnordi, T. Blankevoort, and M. Welling. Batch-shaping for learning conditional channel gated networks. In International Conference on Learning Representations. 2020 . [57](#)
- [151] Z. Wu, T. Nagarajan, A. Kumar, S. Rennie, L. S. Davis, K. Grauman, and R. Feris. BlockDrop: Dynamic Inference Paths in Residual Networks. *arXiv e-prints* arXiv:1711.08393. [57](#)
- [152] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger. Deep Networks with Stochastic Depth. In B. Leibe, J. Matas, N. Sebe, and M. Welling, eds., Computer Vision – ECCV 2016. Springer International Publishing, Cham, 2016 646–661. [57](#)

- [153] C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, and D. Wierstra. PathNet: Evolution Channels Gradient Descent in Super Neural Networks. *arXiv e-prints* arXiv:1701.08734. [57](#)
- [154] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In JMLR W&CP: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2010), volume 9. 2010 249–256. [60](#)
- [155] D. Mishkin and J. Matas. All you need is a good init. *arXiv e-prints* arXiv:1511.06422. [60](#)
- [156] K. He, X. Zhang, S. Ren, and J. Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *arXiv e-prints* arXiv:1502.01852. [60, 61](#)
- [157] H. Zhang, Y. N. Dauphin, and T. Ma. Residual Learning Without Normalization via Better Initialization. In International Conference on Learning Representations. 2019 . [61](#)
- [158] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In F. Bach and D. Blei, eds., Proceedings of the 32nd International Conference on Machine Learning, volume 37 of *Proceedings of Machine Learning Research*. PMLR, Lille, France, 2015 448–456. [61](#)
- [159] S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi. Fine-Grained Visual Classification of Aircraft. Technical Report 2013. [62, 65, 67](#)
- [160] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. 3D Object Representations for Fine-Grained Categorization. In 4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13). Sydney, Australia, 2013 . [62, 65, 67](#)
- [161] S. Sinha, K. Roth, A. Goyal, M. Ghassemi, H. Larochelle, and A. Garg. Uniform Priors for Data-Efficient Transfer. *arXiv e-prints* arXiv:2006.16524. [63, 64](#)
- [162] Y. LeCun, C. Cortes, and C. Burges. MNIST handwritten digit database. *ATT Labs [Online]. Available: <http://yann.lecun.com/exdb/mnist>* 2. [63](#)
- [163] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading Digits in Natural Images with Unsupervised Feature Learning . [63](#)

- [164] J. J. Hull. A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 16, (1994) 550–554. 63
- [165] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell. Adversarial Discriminative Domain Adaptation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2017 . 63, 64
- [166] F. Chollet. The typical transfer learning workflow. https://keras.io/guides/transfer_learning/ 2020. 65
- [167] S. Chilamkurthy. Transfer Learning tutorial in PyTorch. https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html 2020. 65
- [168] Tensorflow. Create the base model from the pre-trained convnets. https://www.tensorflow.org/tutorials/images/transfer_learning 2020. 65
- [169] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition. Ieee, 2009 248–255. 65
- [170] A. Paszke, S. Gross, F. Massa, A. Lerer et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *arXiv e-prints* arXiv:1912.01703. 65
- [171] L. Bossard, M. Guillaumin, and L. Van Gool. Food-101 – Mining Discriminative Components with Random Forests. In European Conference on Computer Vision. 2014 . 65
- [172] M.-E. Nilsback and A. Zisserman. Automated Flower Classification over a Large Number of Classes. In Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing. 2008 . 65