UNIVERSITY OF CALIFORNIA SAN DIEGO

Trustworthy Neural Architecture Search

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy

in

Electrical Engineering (Intelligent Systems, Robotics, and Control)

by

Ramtin Hosseini

Committee in charge:

Professor Pengtao Xie, Chair
Professor Manmohan Chandraker
Professor Pamela Cosman
Professor Bhaskar Rao

2024

The Dissertation of Ramtin Hosseini is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2024

DEDICATION

To my parents for their endless love and support

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

First and foremost, I would like to extend my deepest appreciation to my advisor and mentor, Prof. Pengtao Xie, for granting me the opportunity to pursue research and for his unwavering support and guidance throughout this journey. Prof. Xie has not only exemplified professionalism and integrity but has also been a beacon of kindness and compassion towards others. His role as a mentor has profoundly influenced me, and I am deeply thankful for his generosity, empathy, and friendship.

My heartfelt thanks go to my family for their unwavering support and sacrifices over these years. Their presence and encouragement have been indispensable to me on this journey.

I am grateful to my committee members, Professor Pamela Cosman, Professor Bhaskar Rao, and Professor Manmohan Chandraker, for their insightful feedback and guidance during my research. My gratitude also extends to my co-authors and lab mates—Li Zhang, Han Guo, Youwei Liang, Sai Ashish Somayajula, Bhanu Garg, Shubham Chitnis, and Xingyi Yang—for their invaluable assistance and camaraderie throughout this research process.

Chapter 1, in part, has been published in the Proceedings of the 2021 Computer Vision and Pattern Recognition (CVPR) Conference under the title "DSRNA: Differentiable Search of Robust Neural Architectures" by Ramtin Hosseini, Xingyi Yang, and Pengtao Xie. The dissertation author was the primary author of this material.

Chapter 2, in part, has been published in the Proceedings of the 2023 International Conference on Machine Learning (ICML) under the title "Fair and Accurate Decision Making through Group-Aware Learning" by Ramtin Hosseini, Bhanu Garg, Li Zhang, and Pengtao Xie. The dissertation author was the primary author of this material.

Chapter 3, in part, has been published in the 2022 ACM Multimedia (ACM MM) Conference under the title "Image Understanding by Captioning with Differentiable Architecture Search" by Ramtin Hosseini, and Pengtao Xie. The dissertation author was the primary author of this material.

Chapter 4, in part, has been published in the Proceedings of the 2022 Neural Information

Processing Systems (NeurIPS) Conference under the title ”Saliency-Aware Neural Architecture Search” by Ramtin Hosseini and Pengtao Xie. Additionally, a portion of this chapter has been published in the 2022 Nature Scientific Reports as ”Brain Tumor Classification Based on Neural Architecture Search” by Ramtin Hosseini, Shubham Chitnis, and Pengtao Xie. The dissertation author served as the primary author for these publications.

VITA

2018 Bachelor of Science in Mechanical Engineering, University of California, Berkeley

2022 Master of Science in Electrical Engineering (Intelligent Systems, Robotics, and Control), University of California San Diego

2024 Doctor of Philosophy in Electrical Engineering (Intelligent Systems, Robotics, and Control), University of California San Diego

PUBLICATIONS

- **Ramtin Hosseini**, Xingyi Yang, Pengtao Xie. "DSRNA: Differentiable Search of Robust Neural Architectures". Accepted at *CVPR 2021*.
- Bhanu Garg, Li Zhang, Pradyumna Sridhara, **Ramtin Hosseini**, Eric Xing, Pengtao Xie. "Learning from Mistakes - with Application to Neural Architecture Search". Accepted at *AAAI 2022*.
- **Ramtin Hosseini**, Pengtao Xie. "Image Understanding by Captioning with Differentiable Architecture Search". Accepted at *ACM MM 2022*.
- **Ramtin Hosseini**, Pengtao Xie. "Sailency-Aware Neural Architecture Search". Accepted at *NeurlPS 2022*.
- **Ramtin Hosseini**, Li Zhang, Bhanu Garg, Pengtao Xie. "Fair and Accurate Decision Making using Group-Aware Learning". Accepted at *ICML 2023*.
- **Ramtin Hosseini***, Han Guo *, Ruiyi Zhang, Sai Ashish Somayajula, Ranak Roy Chowdhury, Rajesh K. Gupta, Pengtao Xie. "Downstream Task Guided Masking Learning in Masked Autoencoders Using Multi-Level Optimization". Under review at *ICML 2024*.
- **Ramtin Hosseini**, Shubham Chitnis, Pengtao Xie. "Brain tumor classification based on neural architecture search". Published at *Scientific Reports Nature 2022*.
- Sai Ashish Somayajula, Onkar Litake, Youwei Liang, **Ramtin Hosseini**, Shamim Nemati, David O. Wilson, Robert N. Weinreb, Atul Malhotra, Pengtao Xie. "Improving Long COVID-Related Text Classification: A Novel End-to-End Domain-Adaptive Paraphrasing Framework". Published at *Scientific Reports Nature 2024*.
- Han Guo, **Ramtin Hosseini**, Sai Ashish Somayajula, Pengtao Xie. "Improving Image Classification of Gastrointestinal Endoscopy Using Curriculum Self-Supervised Learning". Published at *Scientific Reports Nature 2024*.

FIELDS OF STUDY

Major Field: Electrical Engineering (Intelligent Systems, Robotics, and Control)
Focus: Machine Learning; Multi-Level Optimization; Trustworthy AI; Healthcare Applications

ABSTRACT OF THE DISSERTATION

Trustworthy Neural Architecture Search

by

Ramtin Hosseini

Doctor of Philosophy in Electrical Engineering (Intelligent Systems, Robotics, and Control)

University of California San Diego, 2024

Professor Pengtao Xie, Chair

In the dynamically progressing domain of Artificial Intelligence (AI), Machine Learning (ML) stands out as a pivotal innovation, acclaimed for its proficiency in enabling autonomous learning from data without direct human supervision. This advancement has positioned AI systems at the vanguard of technological progress, endeavoring to amplify human capabilities across diverse tasks through heightened efficiency, precision, and the reduction of manual intervention. The drive towards automation within these systems has given rise to Automated Machine Learning (AutoML), an advanced, AI-facilitated methodology that automates the process of model selection, dataset optimization, weight parameterization, and hyperparameter tuning, significantly reducing the need for extensive human expertise and intervention.

Central to AutoML research is the area of Neural Architecture Search (NAS), which has shown exceptional performance, rivaling human capabilities across various applications. However, the deployment of ML solutions, especially in critical areas such as healthcare diagnostics and autonomous vehicle guidance, is hampered by concerns about their trustworthiness. To alleviate these apprehensions, it is imperative to rigorously investigate and incorporate principles of trustworthy AI, such as interpretability, fairness, robustness, and reliability, thereby encouraging their wider acceptance and utilization.

This dissertation investigates the creation and evaluation of multi-level optimization (MLO) frameworks aimed at augmenting the trustworthiness and effectiveness of NAS methodologies. By weaving together elements of interpretability, fairness, robustness, and generalization, this research endeavors to devise NAS frameworks that are distinguished not only by their performance but also by their ethical and dependable operation. We systematically apply our innovative frameworks to diverse tasks, including image classification, natural language processing, and image captioning, to empirically verify their impact and efficacy.

Expanding on this foundation, the study examines the implementation of these trustworthy NAS frameworks in crucial healthcare contexts, undertaking thorough experiments to assess their accuracy and reliability. Additionally, this dissertation thoughtfully explores potential future research directions and the limitations of the proposed approaches. Through this examination we highlight the ongoing necessity for research to address the complexities and ethical issues surrounding the broad implementation of AI specially NAS.

Introduction

Thesis Introduction and Scope

Artificial Intelligence (AI) profoundly influences our daily lives, from the Face ID on our smartphones and voice assistants to transforming our work and lifestyle. The World Economic Forum's Future of Jobs Report 2020 forecasts that AI is set to create 58 million new jobs in the next five years, underscoring its significant economic potential. It's projected that AI could boost global economic output by an additional 13 trillion US dollars by 2030, equating to an annual GDP growth rate of 1.2% worldwide. AI holds the promise of reshaping our lives for the better by enhancing efficiency, reducing costs, and fostering new avenues for economic growth. Nonetheless, it's crucial to acknowledge AI's potential risks, such as the possibility of job displacement and the exacerbation of economic disparities if left unmanaged. Thus, establishing a responsible, ethical, and beneficial AI ecosystem that mitigates these risks is imperative.

In recent times, Machine Learning (ML)—a pivotal AI subset—has achieved remarkable progress in areas like natural language processing and computer vision, giving rise to Automated Machine Learning (AutoML). AutoML leverages ML algorithms to refine model performances and hyperparameters autonomously, minimizing the need for extensive labor and time. Among the forefronts of AutoML is Neural Architecture Search (NAS), renowned for achieving or even surpassing human-level benchmarks. NAS automates the design of optimal neural networks for specific tasks, moving away from manual architecture design. However, despite its impressive advancements, NAS methods are prone to inaccuracies that could have dire consequences in healthcare, economics, and security sectors. Consequently, ensuring the reliability of NAS systems has become a critical challenge in their broader acceptance and application. Building

trustworthy and dependable systems is crucial for their integration into mission-critical and safety-sensitive sectors, as user confidence significantly influences their industry adoption.

In this thesis, our goal is to develop reliable Neural Architecture Search (NAS) methods suitable for mission-critical and safety-critical applications, such as healthcare and autonomous driving. To achieve this, we aim to delineate various dimensions of trustworthiness and devise improved optimization frameworks and search algorithms for NAS that adhere to these dimensions. Central to creating trustworthy and explainable AI are criteria like (i) robustness and reliability, (ii) fairness, (iii) out-of-domain generalization, and (iv) interpretability. Accordingly, we propose optimization frameworks and algorithms tailored to find optimal architectures that are not only task-specific and data-driven but also meet these trustworthiness criteria. In the process, we encounter several challenges: (1) Identifying robust architectures without compromising accuracy on standard models, (2) Mitigating overfitting and bolstering fairness in NAS, (3) Utilizing knowledge transfer to boost generalization, and (4) Enhancing the interpretability of NAS and assessing the impact of our method and understanding the trade-offs involved. These challenges are addressed in subsequent chapters. In **Chapter 1**, we delve into adversarial robustness within NAS techniques, focusing on how to design architectures that are resilient to adversarial attacks. This involves directly assessing and enhancing the robustness of architectures during the search process, paving the way for more secure neural networks across various applications. **Chapter 2** examines the fairness aspect of NAS, exploring strategies to mitigate biases that lead to unequal resource distribution and, consequently, subpar performance for certain model classes. Through reviewing existing methods and developing novel approaches, this chapter aims to rectify current inequalities, enhance fairness, and prevent overfitting in NAS processes. In **Chapter 3**, we tackle the challenge of out-of-domain generalization in NAS algorithms, proposing solutions to enhance NAS models' performance on unseen data. The effectiveness of these solutions is validated through comprehensive experiments across multiple datasets. Finally, **Chapter 4** addresses the issue of interpretability in NAS. Here, we review the state of NAS interpretability, discuss the application of existing interpretability techniques to NAS, and explore the benefits

and challenges of incorporating interpretability into NAS. This chapter also outlines future research directions in NAS and interpretability, highlighting the application of these methods in healthcare to demonstrate their real-world efficacy.

Backgrounds

In this section, we discuss about the concept of Trustworthy AI and we summarize the related works in this area. Then we review some of the existing works of neural architecture search (NAS) and we study their advantages and disadvantages.

Trustworthy AI

The notion of *trustworthiness*, as delineated in lexical references such as Diction, is fundamentally described as the capacity to be deemed reliable or *"able to be trusted"*. This idea comes from "trust," which is all about believing in someone or something's reliability. Trust helps us navigate risks in our environment, even though we can't control everything. By trusting, we're okay with taking some risks because we believe in the reliability of others or technology. Building trust is essential for strong relationships, and this is particularly true when it comes to technology. Despite technological advances, people often worry about how much they can trust technology. This concern is crucial because without trust, people might hesitate to use technology, missing out on its benefits. For instance, since the term "Artificial Intelligence" (AI) was first used in 1956, AI has made incredible progress. AI systems, which can make decisions like humans, have become a big part of various fields like economics, education, healthcare, and transportation, significantly changing these areas. Because AI plays such an important role in our lives, it's vital to make AI systems trustworthy. This will help reduce worries about the possible dangers of AI. To make AI trustworthy, we need to focus on four main things: (1) **Robustness:** AI systems should be reliable and perform well even when conditions change or when they're faced with unexpected challenges, like cyberattacks. (2) **Fairness:** AI should

make decisions fairly, without bias. This means treating all users equally, regardless of their background. (3) **Generalization:** AI should work well not just on the data it was trained on but also on new, unseen data. This is important because the real world is full of diverse situations and data. (4) **Interpretability:** This is about making AI's decisions understandable. AI should not be a "black box"; users should be able to understand how AI makes its decisions. This helps users trust and feel comfortable using AI systems. In simple terms, making AI trustworthy is not just about technology; it's about ethics and responsibility. It's about ensuring AI can truly benefit us without causing harm or unfairness. By focusing on interpretability, fairness, robustness, and generalization, we can build AI systems that are not only smart but also deserving of our trust.

Neural Architecture Search (NAS)

The landscape of Neural Architecture Search (NAS) has evolved rapidly, with a diverse array of methods being proposed that have demonstrated remarkable success in automating the identification of high-performance neural network architectures, thereby diminishing the dependence on human expertise. This surge in NAS methodologies has garnered significant interest across a spectrum of deep learning applications, encompassing domains such as computer vision and natural language processing. Specifically, NAS has been instrumental in advancing tasks like image classification, object detection, and language modeling through the automated design of optimal neural network architectures. Early NAS endeavors primarily leveraged reinforcement learning (RL) strategies, exemplified by seminal works [218, 144, 219]. These approaches employ a policy network to generate and subsequently evaluate architectures against a validation set, utilizing the validation loss as a feedback signal to refine and optimize the policy network. This iterative process aims to train the policy network to generate increasingly effective architectures. Despite achieving initial successes, RL-based NAS methods are notably resource-intensive, necessitating substantial computational investment for architecture evaluation. This requirement often renders RL-based approaches impractical for users with limited computational capabilities. In response to these limitations, the advent of differentiable search techniques

marked a significant milestone in NAS research [122]. Differentiable NAS methods conceptualize architectures as differentiable entities, facilitating the use of efficient gradient-based optimization techniques. This paradigm introduces a search space constituted by an extensive collection of modular blocks, each associated with a continuous variable denoting its relative importance. The search process then evolves into an optimization problem, aiming to identify a configuration of these variables that minimizes the validation loss. Pioneered by DARTS [122], this approach has been further refined and expanded by subsequent methodologies, including P-DARTS [195], PC-DARTS [194], and DATA [20], among others. These advancements have contributed to the progressive enhancement of architecture depth, reduction in search redundancy, and optimization of operation weights, thereby narrowing the performance discrepancy between search and validation phases. Moreover, recent developments have introduced topology optimization into the realm of differentiable NAS, underscoring the critical role of network topology in optimizing neural network performance [70]. Complementing the RL-based and differentiable paradigms, evolutionary algorithms represent another innovative approach within NAS research [121, 147]. By treating architectures as individuals within a generational population model, this strategy fosters the evolution of architectures through a natural selection-like process, predicated on fitness scores. Although evolutionary algorithms share the computational intensity characteristic of RL-based methods, they offer a distinct and valuable perspective on the architecture search challenge.

Chapter 1

Robust NAS Against Adversarial Attacks

1.1 Introduction

In deep learning applications, the architectures of neural models play a crucial role in improving performance. For example, on the ImageNet [46] benchmark, the image classification error is reduced from 16.4% to 3.57%, when the architecture is evolved from AlexNet [109] to ResNet [78]. Previously, neural architectures are mostly designed by humans, which is time-consuming to obtain a highly-performant architecture. Recently, automated neural architecture search [218, 219, 148, 149, 187, 188] which develops algorithms to find out the optimal architecture that yields the best performance on the validation datasets, has raised much attention and achieved promising results. For example, on the CIFAR-10 dataset, an automatically searched architecture [122] achieves an image classification error rate of 2.76% while the error achieved by state-of-the-art human-designed architecture is 3.46%.

As we will show in the experiments, the architectures searched by existing methods are prone to adversarial attacks. A small perturbation (which is not perceivable by humans) of the input data can render the architecture to change prediction outcomes significantly. Many approaches [65, 17, 132, 38, 112] have been proposed to improve the robustness of DNNs. In these approaches, the architecture of a DNN is provided by humans, and the defense method focuses on training the weights in this architecture in a robust way. However, the robustness of a DNN is not only relevant to its weight parameters, but also determined by the architecture. It is

important to search for architectures that are robust to adversarial attacks as well.

In this chapter, we develop a novel approach for robust NAS. We define two differentiable metrics to measure the robustness of architectures and formulate robust NAS as an optimization problem that aims to find out an optimal architecture by maximizing the robustness metrics. The first metric is defined based on certified lower bound [15]. Linear bounding methods are applied to individual building blocks in the differentiable architecture search space and these individual bounds are composed to obtain global bounds for the entire neural architecture. The second metric is based on the Jacobian norm bound [80], where the robustness is measured by how much the output shifts as the input is perturbed. The shift is upper bounded by the norms of row vectors in the Jacobian matrix of the neural architecture. Our approach is applicable to various forms of differentiable architecture search methods (e.g., DARTS [122], PC-DARTS [194], P-DARTS [29], etc. and is robust against adversarial attacks in various norm choices. Previously, robust NAS has been investigated in [71, 26], based on adversarial training of randomly sampled sub-architectures [71] and differentiable architecture variables [26]. Unlike these methods that achieve robustness implicitly via adversarial training, our method explicitly defines robustness metrics and directly optimizes these metrics to obtain robust architectures.

On CIFAR-10, ImageNet, and MNIST, we perform game-based and verification-based evaluations on the robustness of our methods. The experimental results show that our methods 1) are more robust to various norm-bound attacks than several robust NAS baselines; 2) are more accurate than baselines when there are no attacks; 3) have significantly higher certified lower bounds than baselines.

1.2 Related Works

1.2.1 Adversarial Attacks and Defenses

Adversarial attacks aim to perturb input data examples by adding imperceptible noises so that the prediction results are altered significantly. In white-box attack [173, 22, 31, 220], the

adversary has full access to the target model, while in the black-box attack [24, 91, 177, 30], the target model is unknown to the adversary. In targeted attacks, the adversary aims to change the prediction outcome in certain classes, while untargeted attacks are not class-specific. Arguably, the most popular and effective white-box untargeted attacks with various norm-bounds are: fast gradient sign method (FGSM) [65], projected gradient descent (PGD) [132], and Carlini & Wagner (C&W) [17]. FGSM is a single step attack algorithm that aims to increase the adversarial loss by updating its gradient sign. PGD is a more general version of FGSM that runs over several iterations to increase the adversarial loss. The attacks of FGSM and PGD are based on l_∞-norm bound, while those in C&W are based on l_0, l_2, and l_∞ norms. C&W is particularly effective for l_2-norm attacks. Additionally, a recent work AutoAttack [42] proposes a reliable and robust attack method using an ensemble of stepsize-free versions of PGD attacks, a white-box attack – Fast Adaptive Boundary (FAB) [41], and a black-box attack – Square Attack [7] to create parameter-free attacks. To improve the robustness of neural networks against adversarial attacks, many adversarial defense methods have been proposed, such as random smoothing [112, 38], adversarial training [65, 132, 17], and Jacobian regularization [94, 80, 19]. Jacobian regularization aims to minimize the change of network outputs when inputs are perturbed. Mathematically, this amounts to minimizing the Frobenius norm of a Jacobian matrix.

Most of these defense methods assume the neural architectures are manually designed by humans and focus on improving the robustness of network weights. Automatically searching for robust architectures is largely under-explored. In [48], experiments show that architectures searched by existing NAS methods such as DARTS, PC-DARTS, and P-DARTS are vulnerable to various forms of adversarial attacks. To address this issue, studies have been conducted to robustify NAS methods. RobNet [71] used one-shot NAS to obtain a large number of networks and then studied the patterns of architectures that are robust against adversarial attacks. They discovered that using dense connectivity and adding convolution operations to direct connection edges help to improve robustness. Chen et al. [26] proposed performing adversarial training and random smoothing on architecture variables, which can improve the robustness of DARTS-

based methods. Our work takes a different approach for robustifying architectures, where we explicitly define differentiable metrics to measure architectures' robustness and search for robust architectures by maximizing these metrics.

1.2.2 Robustness Verification of Neural Networks

Robustness verification aims to provide certified defense against any possible attacks under a threat model. A robustness certificate ϵ means the prediction outcome cannot be changed if the strength of the attack is smaller than ϵ. Many verification approaches [190, 186, 208, 55] have been proposed, which focus on achieving tighter lower bounds of the robustness certificate, computing bounds for various complex building blocks in neural networks, and improving the efficiency in computing the bounds. Dvijotham et al. [55] formulate verification as an optimization problem and seek bounds of the certificate by solving a Lagrangian relaxation of the optimization problem. Weng et al. [186] propose methods to verify the robustness of Rectified Linear Unit (ReLU) networks by bounding the ReLU units with linear functions or local Lipschitz constant. CNN-Cert [15] applies linear bounding techniques to provide certified lower bounds for various operations including convolution, pooling, batch normalization, residual blocks, activation functions, etc.

1.3 Methods

We begin with defining differentiable metrics to measure the robustness of neural architectures. Then we propose a robust NAS framework that performs optimization in the architecture search space to maximize the robustness metrics. The objective function explores a tradeoff between predictive accuracy and robustness and can be efficiently optimized using gradient-based methods.

1.3.1 Defining Differentiable Robustness Metrics

In this section, we define two differentiable metrics to measure the robustness of neural architectures. The first one is based on robustness certification methods [15]. Specifically, given an architecture, we seek to obtain a certified lower bound of this architecture and use the bound to measure robustness. The architecture with a larger lower bound is more robust against different attacks. The second metric is based on upper-bounding the shift of the model's prediction when the inputs are perturbed, and the bound is based on the norm of the Jacobian matrix [80] of the architecture. The smaller the upper bound is, the more robust the network is. Previous works [15, 80] have utilized certified bounds and Jacobian regularization to measure or improve the robustness of neural networks that have human-designed and fixed architectures. Different from these works, our work defines certified bounds and Jacobian regularizers on neural architecture variables and leverage them to search for robust architectures.

Measuring Robustness Based on Certified Bound

One way to measure the robustness of a neural network is to use the verified robustness certificate. A certificate with value $\varepsilon(\mathbf{x})$ means that model prediction on the input data $\mathbf{x}$ cannot be changed if the attack strength is smaller than $\varepsilon(\mathbf{x})$. A larger $\varepsilon(\mathbf{x})$ indicates more robustness. In practice, it is infeasible to obtain the exact robustness certificate of a model. Instead, one can derive lower bounds of $\varepsilon(\mathbf{x})$ and use these lower bounds as surrogates for measuring robustness. Given an architecture search space comprised of various building blocks such as ReLU-Conv-BN, (dilated) separable convolutions, pooling operations, etc., we perform linear bounding [15] on these building blocks and compose the individual bounds to obtain a certified lower bound for each architecture in the search space. These bounds are differentiable functions of architecture variables and are amenable for gradient-based optimization. In the sequel, we discuss how to derive the certified upper and lower bounds for each type of building blocks.

ReLU-Conv-BN Block

The ReLU-Conv-BN building block consists of three consecutive operations including rectified linear unit (ReLU) as a nonlinear activation operation, convolution, and batch normalization (BN). Let Φ^r and Φ^{r-2} be the output and input of an ReLU-Conv-BN block r, then we have

$$\Phi^{r-1} = W^{r-1} * \sigma(\Phi^{r-2}) + b^{r-1} \tag{1.1}$$

$$\Phi^r = \gamma_{bn} \frac{\Phi^{r-1} - \mu_{bn}}{\sqrt{\sigma_{bn}^2 + \varepsilon_{bn}}} + \beta_{bn} \tag{1.2}$$

where $\sigma(\cdot)$ is the ReLU function. W^{r-1} and b^{r-1} are the weight parameters and bias parameters in the convolution operation. μ_{bn} and σ_{bn}^2 are the mean and variance of a batch of Φ^{r-1} in batch normalization. γ_{bn}, ε_{bn}, and β_{bn} are hyperparameters in BN.

By applying linear bounds to these equations, we get these upper and lower bounds:

$$A_{L,bn}^r * \Phi^{r-1} + B_{L,bn}^r \leq \Phi^r \leq A_{U,bn}^r * \Phi^{r-1} + B_{U,bn}^r \tag{1.3}$$

$$A_{L,bn}^r \Phi^{r-1} + B_{L,bn}^r \geq A_{L,bn}^r (A_{L,conv}^{r-1} \Phi^{r-2} + B_{L,conv}^{r-1}) + B_{L,bn}^r \tag{1.4}$$

$$A_{U,bn}^r \Phi^{r-1} + B_{U,bn}^r \leq A_{U,bn}^r (A_{U,conv}^{r-1} \Phi^{r-2} + B_{U,conv}^{r-1}) + B_{U,bn}^r \tag{1.5}$$

where $A_{L,bn}$, $A_{U,bn}$, $B_{L,bn}$, and $B_{U,bn}$ are constants that can be computed as in [15]:

$$A_{L,bn}^r = A_{U,bn}^r = \frac{\gamma_{bn}}{\sqrt{\sigma_{bn}^2 + \varepsilon_{bn}}} \tag{1.6}$$

$$B_{L,bn}^r = B_{U,bn}^r = \frac{-\gamma_{bn}\mu_{bn}}{\sqrt{\sigma_{bn}^2 + \varepsilon_{bn}}} + \beta_{bn} \tag{1.7}$$

and $A_{L,conv}$, $A_{U,conv}$, $B_{L,conv}$, $B_{U,conv}$ are constant tensors.

(Dilated) Separable Convolutions

Another two types of building blocks in our search space are separable convolutions and dilated separable convolutions. Dilated separable convolutions consist of four consecutive operations: ReLU, convolution, convolution, and batch normalization (BN). Separable convolutions consist of two consecutive dilated separable convolutions. Let Φ^{r-3} and Φ^{r} denote the input and output of a dilated separable convolution, then:

$$\Phi^{r-1} = W^{r-1} * (W^{r-2} * \sigma(\Phi^{r-3}) + b^{r-2}) + b^{r-1} \tag{1.8}$$

where W^{r-1} and W^{r-2} are weights of convolutions; b^{r-2} and b^{r-1} are bias parameters in convolutions. The calculation of Φ^{r} is the same as that in Eq.(1.2). We can again use Eq.(1.3) to find the upper and lower bound of Φ^{r}, which are:

$$\begin{aligned} &A^{r}_{L,bn} * \Phi^{r-1} + B^{r}_{L,bn} \geq A^{r}_{L,bn} * (A^{r-1}_{L,conv} * (W^{r-2} \\ &\Phi^{r-3} + b^{r-2}) + B^{r-1}_{L,conv}) + B^{r}_{L,bn} \end{aligned} \tag{1.9}$$

$$\begin{aligned} &A^{r}_{U,bn} * \Phi^{r-1} + B^{r}_{U,bn} \leq A^{r}_{U,bn} * (A^{r-1}_{U,conv} * (W^{r-2} \\ &\Phi^{r-3} + b^{r-2}) + B^{r-1}_{U,conv}) + B^{r}_{U,bn} \end{aligned} \tag{1.10}$$

The upper and lower bound for separable convolution operations can be derived in a similar way.

Pooling Operations

Let Φ^{r-1} and Φ^{r} denote the input and output of a pooling operation r. We have the following lower and upper bound of Φ^{r}:

$$A^{r}_{L,pool} * \Phi^{r-1} + B^{r}_{L,pool} \leq \Phi^{r} \leq A^{r}_{U,pool} * \Phi^{r-1} + B^{r}_{U,pool} \tag{1.11}$$

Robustness Metric

Given the lower and upper bounds of individual building blocks, we are ready to derive a certified lower bound for the entire network as a measure of the robustness of its architecture. In differentiable architecture search [122], the neural network is overparameterized with many building blocks that are organized into a directed acyclic graph (DAG). The output of each block is multiplied with a positive scalar. The larger the scalar is, the more critical the block is. After learning, a subset of blocks with the largest scalars are selected to form the final architecture of this network. Therefore, these scalars (called architecture variables) represent the architecture. Given a block with lower bound L and upper bound U, after multiplying with an architecture variable α, this block has a lower bound of αL and αU. Following the topological order of blocks in the DAG, we recursively compose the lower and upper bounds (multiplied with architecture variables) of blocks and get a global lower and upper bound for the entire network. These two bounds are functions of architecture variables and the input data example. The lower bound is used as the robustness metric.

Measuring Robustness with Jacobian Regularization

When the architecture search space is large, computing gradients of the certified lower bound with respect to architecture variables is time-consuming. To address this problem, we investigate another measure of robustness, which is computationally efficient. Let $f(\mathbf{x})$ denote the neural network which takes a data example $\mathbf{x} \in \mathbb{R}^D$ as input and outputs a K-dimensional vector. Similar to the robustness metric defined in Section 1.3.1, the architecture search space is differentiable, where continuous architecture variables are multiplied to the outputs of building blocks. Therefore, $f(\mathbf{x})$ is a continuous function of the architecture variables. Let $\mathbf{x}+\varepsilon$ be an adversarial example where ε is a small perturbation vector. We assume the p-norm of ε is less equal to a small scalar δ: $\|\varepsilon\|_p \leq \delta$. The robustness of the network can be measured using the

following quantity [80]:

$$S = -\mathbb{E}_{\mathbf{x}}\mathbb{E}_{\varepsilon}\left[\frac{1}{K}\sum_{k=1}^{K}|f_k(\mathbf{x}+\varepsilon)-f_k(\mathbf{x})|\right] \tag{1.12}$$

where $a = 1/K\sum_{k=1}^{K}|f_k(\mathbf{x}+\varepsilon)-f_k(\mathbf{x})|$ is the average change of the output across all dimensions when $\mathbf{x}$ is perturbed with ε and S is the expectation of a defined with respect to the distributions of $\mathbf{x}$ and ε. The smaller this quantity is, the more robust the network is: intuitively, a network is robust if for every input data example, no matter how it is perturbed, the change of network output is small. According to Taylor expansion, we have:

$$f_k(\mathbf{x}+\varepsilon)-f_k(\mathbf{x}) \approx \left[\frac{\partial f_k(\mathbf{x})}{\partial \mathbf{x}}\right]^{\top}\varepsilon \tag{1.13}$$

Let $\mathbf{J}(\mathbf{x})$ denote the Jacobian matrix at $\mathbf{x}$ where $J_{kj} = \partial f_k(\mathbf{x})/\partial x_j$. Then $\partial f_k(\mathbf{x})/\partial \mathbf{x} = \mathbf{J}_k(\mathbf{x})$ where $\mathbf{J}_k(\mathbf{x})$ is the k-th row vector of $\mathbf{J}(\mathbf{x})$. According to Hölder's inequality, we have:

$$\left|\mathbf{J}_k(\mathbf{x})^{\top}\varepsilon\right| \leq \|\mathbf{J}_k(\mathbf{x})\|_q \|\varepsilon\|_p \leq \|\mathbf{J}_k(\mathbf{x})\|_q \delta \tag{1.14}$$

where $\frac{1}{p}+\frac{1}{q}=1$.

Putting these pieces together, we have:

$$\begin{aligned}
&-\mathbb{E}_{\mathbf{x}}\mathbb{E}_{\varepsilon}\left[\tfrac{1}{K}\textstyle\sum_{k=1}^{K}|f_k(\mathbf{x}+\varepsilon)-f_k(\mathbf{x})|\right] \\
&\approx -\mathbb{E}_{\mathbf{x}}\mathbb{E}_{\varepsilon}\left[\tfrac{1}{K}\textstyle\sum_{k=1}^{K}|\mathbf{J}_k(\mathbf{x})^{\top}\varepsilon|\right] \\
&\geq -\mathbb{E}_{\mathbf{x}}\mathbb{E}_{\varepsilon}\left[\tfrac{1}{K}\textstyle\sum_{k=1}^{K}\|\mathbf{J}_k(\mathbf{x})\|_q \delta\right] \\
&= -\delta\mathbb{E}_{\mathbf{x}}\left[\tfrac{1}{K}\textstyle\sum_{k=1}^{K}\|\mathbf{J}_k(\mathbf{x})\|_q\right] \\
&\approx -\tfrac{\delta}{N}\textstyle\sum_{i=1}^{N}\left[\tfrac{1}{K}\sum_{k=1}^{K}\|\mathbf{J}_k(\mathbf{x}_i)\|_q\right]
\end{aligned} \tag{1.15}$$

where in the last step, the expectation is approximated by the mean on a set of data examples $\{\mathbf{x}_i\}_{i=1}^{N}$. To maximize S for achieving robustness, we can maximize its approximated lower

Table 1.1. Accuracy (%) (mean and standard deviation) of different methods under various norm-bound attacks on CIFAR-10. *Average of five runs. † Using early stopping. The best method is boldfaced and the second best is underlined.

Method	PGD (10)	PGD (20)	PGD (100)	FGSM	C&W	AutoAttack (l_∞)	AutoAttack (l_2)
RobNet-large [71]	49.49	49.44	49.24	54.98	47.19	48.93	46.38
RobNet-free [71]	52.80	52.74	52.57	58.38	46.95	50.13	46.33
SDARTS-ADV [26]*	56.94 ± 0.02	56.90 ± 0.04	56.77 ± 0.17	63.84 ± 0.02	42.67 ± 0.09	55.04 ± 0.07	40.98± 0.19
PC-DARTS-ADV [26]*	57.15 ± 0.02	57.11 ± 0.05	56.83 ± 0.21	65.29 ± 0.03	42.58 ± 0.04	55.29± 0.05	40.57± 0.21
DSRNA-CB (ours)*	60.31 ± 0.07	60.22 ± 0.11	59.93 ± 0.24	69.88 ± 0.09	63.01 ± 0.07	59.24± 0.04	61.87 ± 0.15
DSRNA-Jacobian (Ours)*	59.81 ± 0.02	59.77 ± 0.04	59.47 ± 0.14	68.92 ± 0.02	62.87 ± 0.04	59.11 ± 0.04	62.09 ± 0.10
DSRNA-Combined (Ours)* †	**61.12 ± 0.03**	**61.06 ± 0.04**	**60.71 ± 0.15**	**70.32 ± 0.04**	**64.76± 0.06**	**59.83 ± 0.05**	**64.51 ± 0.12**

bound $-\delta/N\sum_{i=1}^{N}\left[1/K\sum_{k=1}^{K}\|\mathbf{J}_k(\mathbf{x}_i)\|_q\right]$. This bound is referred to as the Jacobian norm bound. It is a function of the architecture variables. For l_2 and l_∞ norm bound attacks, $\sum_{k=1}^{K}\|\mathbf{J}_k(\mathbf{x})\|_q$ is the Frobenius norm and l_1 norm of the Jacobian matrix, respectively. We use the method in [80] to compute the Jacobian matrix efficiently based on random projection.

1.3.2 Differentiable Search of Robust Neural Architectures

Given the robustness metrics defined based on certified lower bound and Jacobian norm bound, which are increasing functions of the architecture variables (i.e., larger values of the metrics indicate that the architecture is more robust), we search for robust architectures by maximizing these robustness metrics. The formulation is as follows:

$$\begin{aligned} \min_{\alpha} \quad & \sum_{i=1}^{M} L(w^*(\alpha),\alpha,x_i^{(\text{val})}) - \gamma R(w^*(\alpha),\alpha,x_i^{(\text{val})}) \\ s.t. \quad & w^*(\alpha) = \operatorname*{argmin}_{w} \sum_{i=1}^{N} L(w,\alpha,x_i^{(\text{tr})}) \end{aligned} \tag{1.16}$$

where α denotes the set of architecture variables, and w denotes the weight parameters of blocks. R denotes the robustness metric (either based on certified lower bound or Jacobian norm bound). M is the number of validation examples, and N is the number of training examples. On each validation example $x_i^{(\text{val})}$, we measure the robustness R and predictive loss L of the architecture α and aim to search for an optimal architecture that yields the largest robustness and smallest predictive loss on the validation set. γ is a tradeoff parameter balancing these two

Table 1.2. Accuracy (%) (mean and standard deviation) of different NAS methods when there are no attacks. ‡Average of five runs. †Training without cutout augmentation. ⋆Using early stopping.

Method	Test Acc. (%)	Params (M)	Search Cost (GPU days)	Search Method
NASNet-A [218]	97.35	3.3	1800	RL
AmoebaNet-B [148]	97.45	2.8	3150	evolution
PNAS [119]†	96.59	3.2	255	SMBO
ENAS [144]	97.11	4.6	0.5	RL
DARTS (1st) [122]	97.00 ± 0.14	3.3	1.5	gradient
DARTS (2nd) [122]	97.26 ± 0.09	3.3	4.0	gradient
SNAS (moderate) [193]	97.15	2.8	1.5	gradient
ProxylessNAS [16]*	97.92	–	4.0	gradient
R-DARTS (L2) [206]	97.05 ± 0.21	–	1.6	gradient
DARTS+ [116]	97.68	3.7	0.4	gradient
P-DARTS [29]	97.50	3.4	0.3	gradient
PC-DARTS [194]	97.43 ± 0.07	3.6	0.1	gradient
RobNet-large [71]	78.57	6.9	–	one shot
RobNet-free [71]	82.79	5.5	–	one shot
SDARTS-RS [26]	97.33 ± 0.03	3.4	0.4	gradient
SDARTS-ADV [26]	97.39 ± 0.02	3.3	1.3	gradient
PC-DARTS-ADV [26]	97.51 ± 0.04	3.5	0.4	gradient
DSRNA-CB (ours)‡	97.42 ± 0.07	3.5	4.0	gradient
DSRNA-Jacobian (ours)‡	97.50 ± 0.03	3.5	0.4	gradient
DSRNA-Combined (Ours)‡ ⋆	97.51 ± 0.04	3.5	0.6	gradient

objectives. Similar to [122], this is a bi-level optimization problem. In the inner optimization problem, given an architecture configuration α, an optimal set of weights $w^*(\alpha)$ is learned by minimizing the training loss $\sum_{i=1}^{N} L(w, \alpha, x_i^{(\text{tr})})$. Note that $w^*(\alpha)$ is a function of α: each architecture configuration α corresponds to a set of optimal weights $w^*(\alpha)$. $w^*(\alpha)$ and α are both used to measure the robustness and predictive loss on the validation set. In the outer optimization problem, we learn the architecture variables by minimizing the validation loss and maximizing the robustness metric, i.e., searching for an architecture that is accurate and robust. When R is the metric based on certified bound (CB), our method is denoted as DSRNA-CB; when

Table 1.3. Accuracy (%) (mean and standard deviation) of different methods on ImageNet under various attacks and without attack. *Average of five runs. These architectures were searched on CIFAR-10. The best method is boldfaced.

Method	Without attack	PGD (100)	FGSM	C&W	AutoAttack (l_∞)	AutoAttack (l_2)	Params (M)
RobNet-large [71]	61.26	37.14	39.74	25.73	32.96	23.90	11.6
SDARTS-ADV [26] *	74.85 ± 0.06	46.54 ± 0.13	48.09 ± 0.07	36.86 ± 0.10	41.58± 0.07	35.71± 0.15	4.7
PC-DARTS-ADV [26] *	75.73 ± 0.07	**46.59 ± 0.15**	48.25 ± 0.08	36.69 ± 0.09	41.79±0.06	35.86±0.11	5.3
DSRNA-CB (ours)*	75.84 ± 0.11	45.39 ± 0.18	**50.89 ± 0.07**	**43.64 ± 0.19**	**44.05±0.09**	**42.98±0.16**	5.4
DSRNA-Jacobian (ours)*	**75.88 ± 0.07**	43.79 ± 0.11	48.69 ± 0.04	43.17 ± 0.08	43.81±0.03	42.56± 0.11	5.3

Table 1.4. Accuracy (%) (mean and standard deviation) of different methods on MNIST under various attacks and without attack. *Average of five runs. The best method is boldfaced and the second best is underlined.

Method	Without attack	PGD (100)	FGSM	C&W	AutoAttack (l_∞)	AutoAttack (l_2)
RobNet-large [71]	90.73	87.28	89.43	69.38	86.85	65.07
SDARTS-ADV [26] *	99.19 ± 0.01	97.31 ± 0.02	98.67 ± 0.02	78.94 ± 0.05	95.29±0.02	77.73±0.06
PC-DARTS-ADV [26] *	99.21 ± 0.01	97.33 ± 0.04	98.75 ± 0.01	78.93 ± 0.03	95.86±0.03	77.83±0.07
DSRNA-CB (ours)*	99.21 ± 0.03	<u>97.34 ± 0.06</u>	**98.85 ± 0.03**	94.02 ± 0.08	**97.01±0.06**	94.31±0.14
DSRNA-Jacobian (ours)*	<u>99.36 ± 0.01</u>	96.82 ± 0.02	98.79 ± 0.01	<u>95.37 ± 0.02</u>	96.28±0.04	<u>94.91±0.08</u>
DSRNA-Combined (ours)*	**99.40 ± 0.02**	**97.36 ± 0.04**	<u>98.83 ± 0.04</u>	**96.72 ± 0.02**	<u>96.31 ± 0.03</u>	**95.47± 0.09**

R is the metric based on Jacobian norm bound, our method is denoted as DSRNA-Jacobian. The two metrics can be summed together as a single metric, leading to a DSRNA-Combined method. The algorithm for solving the optimization problem in Eq.(16) can be derived in a similar way to that in DARTS [122]. We approximate $w^*(\alpha)$ using one step gradient descent update of w with respect to the training loss. Then we plug in this approximation into the validation loss and robustness metric, and perform gradient descent update of α with respect to the approximated objective in the first line in Eq.(1.16). The detailed algorithm is deferred to the supplements.

1.4 Experiments

1.4.1 Dataset

We used three datasets in the experiments: CIFAR-10 [106], ImageNet [46], and MNIST [111]. CIFAR-10 contains 60K images with a size of 32×32. The train, valida-

tion, and test sets in CIFAR-10 contain 25K, 25K, 10K images, respectively. ImageNet has 1.3M training images and 50K validation images. MNIST has a training set of 60,000 examples and a test set of 10,000 examples, which are 28×28 gray-scale images of handwritten single digits between 0 and 9.

1.4.2 Experimental Settings

Baselines

We compare our proposed methods with the following baselines: 1) RobNet [71] which searches robust architectures based on adversarial training in one-shot NAS; 2) SDARTS-ADV and PC-DARTS-ADV [26], which performs adversarial training on architecture variables in DARTS-based NAS. During architecture evaluation, DSRNA-CB, DSRNA-Jacobian, DSRNA-Combined, SDARTS-ADV, and PC-DARTS-ADV are trained with Jacobian regularization, while RobNet-Free and RobNet-large are trained with adversarial training. We select four popular adversarial attack methods to evaluate the robustness of our methods: fast gradient sign method (FGSM) [65] , projected gradient descent (PGD) [132], Carlini & Wagner (C&W) [17], and AutoAttack [42].

Hyperparameter Settings

The search space of our methods is the same as that of PC-DARTS, which is composed of 3×3 and 5×5 separable convolutions, 3×3 and 5×5 dilated separable convolutions, 3×3 max pooling, 3×3 average pooling, identity, and zero. The convolutional cell consists of 6 nodes, which has 2 input nodes, 3 intermediate nodes, and 1 output node. For CIFAR-10 and MNIST, our methods search the architectures from scratch. In the searching phase, a small network of 8 cells was trained for 50 epochs with an initial number of channels of 16.

In DSRNA-CB, we used SGD for optimizing the network weights w with a learning rate of 0.1, a batch size of 256, a momentum of 0.9, and a weight decay of $3e-4$. We used the Adam optimizer [104] for optimizing architecture variables α, with a fixed learning rate of $6e-4$,

$\beta_1 = 0.5$, $\beta_2 = 0.999$, and a weight decay of $3e-4$. In DSRNA-Jacobian, the network weights w were optimized via SGD with a learning rate of 0.025, a batch size of 128, a momentum of 0.9, and a weight decay of $3e-4$. The architecture variables α were optimized using Adam [104] with a learning rate of $3e-4$, $\beta_1 = 0.5$, $\beta_2 = 0.999$, and a weight decay of $1e-3$.

Given the searched cell, we stack 20 copies of them into a larger network and train this network from scratch on CIFAR-10 or MNIST. The network was trained for 600 epochs from scratch with a batch size of 128, an initial learning rate of 0.025, norm gradient clipping of 5, drop-path with a rate of 0.3, and an initial number of channels of 36. For ImageNet, the architecture is transferred from CIFAR-10: given the optimal cell searched on CIFAR-10, we stack 14 copies of them into a larger network with 48 initial channels and train this network on ImageNet. The training was performed for 250 epochs using an SGD optimizer with an annealing learning rate of 0.5, a momentum of 0.9, and a weight decay of $3e-5$. The tradeoff parameter γ in both DSRNA-CB and DSRNA-Jacobian was set to 0.01. In DSRNA-CB, we initialized ε as 0.03, and then linearly increased or decreased it based on the global difference between the certified upper bound and lower bound. The hyperparameters of baseline methods are deferred to the supplements. A single NVIDIA GTX 1080Ti GPU was used to perform the search.

Table 1.5. Comparison of averaged l_∞-norm certified lower bounds of architectures searched by various methods. Larger is better.

Dataset	**RobNet-large** [71]	**SDARTS-ADV** [26]	**PC-DARTS-ADV** [26]	**DSRNA-CB (ours)**	**DSRNA-Jacobian (ours)**
MNIST	0.0325	0.0471	0.0474	**0.0526**	0.0514
CIFAR-10	0.0024	0.0039	0.0040	**0.0049**	0.0048

Table 1.6. Comparison of averaged l_2-norm certified lower bounds of architectures searched by various methods. Larger is better.

Dataset	**RobNet-large** [71]	**SDARTS-ADV** [26]	**PC-DARTS-ADV** [26]	**DSRNA-CB (ours)**	**DSRNA-Jacobian (ours)**
MNIST	0.1340	0.1767	0.1765	**0.4288**	0.4285
CIFAR-10	0.0167	0.0337	0.0336	**0.0412**	0.0409

1.4.3 Results

In this section, we perform game-based and verification-based evaluations of the adversarial robustness of our proposed methods and compare with state-of-the-art baselines.

Game-based Evaluation

Game-based evaluation estimates the success rate of defending against adversarial attacks with various forms of norm-bounds, such as l_2, l_∞, etc. FGSM [65, 191] and PGD [132] are two effective l_∞ attack methods. C&W [17] is an effective l_2 attack method. On CIFAR-10, ImageNet, and MNIST, we evaluate our proposed methods against 1) PGD attack with $\varepsilon = 8/255$ on CIFAR-10, $\varepsilon = 2/255$ on ImageNet, and $\varepsilon = 0.3$ on MNIST, attack iterations of 10, 20, and 100, and a step size of $2/255$, 2) FGSM attack with $\varepsilon = 2/255$, 3) C&W with 60 attack iterations, 4) AutoAttack (l_∞) with $\varepsilon = 8/255$ on CIFAR-10, $\varepsilon = 2/255$ on ImageNet, and $\varepsilon = 0.3$ on MNIST, and 5) AutoAttack (l_2) with $\varepsilon = 1$.

Table 1.1 shows the accuracy of different methods under various norm-bound attacks on CIFAR-10. PGD (n) denotes the PGD attack with n iterations. From this table, we make the following observations. **First**, the accuracy of our proposed methods, including DSRNA-CB and DSRNA-Jacobian is much higher than that of other robust NAS baselines including RobNet-large, RobNet-free, SDARTS-ADV, and PC-DARTS-ADV, under PGD, FGSM, C&W attacks, AutoAttack (l_∞), and AutoAttack (l_2). This demonstrates that our methods are more robust against various attacks than these baselines. One major reason is that our methods search for robust architectures by explicitly and directly maximizing differentiable robustness metrics and therefore are guaranteed to obtain robust architectures. In contrast, the baseline methods try to improve the robustness of searched architectures implicitly and indirectly: performing adversarial training and injecting random noise. The implicitness and indirectness of these methods do not guarantee robustness. **Second**, among the baselines, there is no consistent winner: SDARTS-ADV and PC-DARTS-ADV perform better than the other baselines under PGD attack, FGSM attack, and AutoAttack (l_∞); RobNet-large and RobNet-free perform better than the other baselines on C&W

attack and AutoAttack (l_2). None of these baselines consistently outperforms others across all these types of attacks. In contrast, our proposed methods are consistently more robust than these baselines under all types of attacks. **Third**, between our two proposed methods DSRNA-CB and DSRNA-Jacobian, DSRNA-CB is slightly more robust than DSRNA-Jacobian. This is probably because the first-order Taylor approximation in DSRNA-Jacobian incurs larger inexactness. However, DSRNA-Jacobian is much faster to train and more memory efficient than DSRNA-CB, as we will show later. **Fourth**, DSRNA-Combined, which utilizes CB and Jacobian norm bound simultaneously for regularization, performs better than DSRNA-CB and DSRNA-Jacobian. This shows that when used together, these two regularizers bring in a synergistic effect.

While our methods are robust against different attacks, we also would like them to be accurate when there are no attacks. To verify this, we compare the accuracy of our methods with state-of-the-art baselines under the attack-free setting. Table 1.2 shows the accuracy achieved by different methods on CIFAR-10 when there are no attacks. From this table, we make the following observations. **First**, the accuracy achieved by our methods is very close to the best accuracy achieved by ASAP. This demonstrates that not only being robust, our methods are also highly accurate when there are no attacks. **Second**, the accuracy of RobNet is much lower than that of ours. This shows that while our methods are not only more robust than RobNet when there are attacks, but also are much more accurate than RobNet when there are no attacks. **Third**, in general, the search cost of our methods is similar to that of other gradient-based baselines. This demonstrates that our methods gain robustness without significantly increasing search cost. Note that the search cost of DSRNA-CB is higher than SDARTS-RS, SDARTS-ADV, and PC-DARTS-ADV. One may wonder whether DSRNA-CB achieves higher robustness than the three baselines because it performs search for a longer time. To check this, in DSRNA-CB, we decrease the batch-size to 64 and use early stopping to reduce the search cost to 0.5 GPU days. The corresponding accuracy on CIFAR-10 is: 57.82% under PGD(100), 65.94% under FGSM, 62.35% under C&W, and 97.37% under no attack. Comparing these results with those in Table 1.1, we can see that our DSRNA-CB method is still more robust than SDARTS-RS,

SDARTS-ADV, and PC-DARTS-ADV when their search costs are about the same. **Fourth**, while SDARTS-ADV and PC-DARTS-ADV can achieve high performance when there are no attacks, they are not as robust as our methods in the presence of attacks, as shown in Table 1.1.

To investigate our methods' transferability, we use the best cell structure searched on CIFAR-10 to compose a larger network and train it on ImageNet. Table 1.3 shows the accuracy of different methods achieved on ImageNet under various norm-bound attacks and without attack. From this table, we make the following observations. **First**, under all the attacks, our methods achieve much higher accuracy than RobNet. Under C&W attack and AutoAttack (l_2), our methods achieve substantially higher accuracy than SDARTS-ADV and PC-DARTS-ADV. Under PGD attack, FGSM attack, and AutoAttack (l_∞), our methods are on par with SDARTS-ADV and PC-DARTS-ADV: our methods are slightly better than SDARTS-ADV and PC-DARTS-ADV under FGSM attacks and AutoAttack (l_∞); SDARTS-ADV and PC-DARTS-ADV are slightly better than our methods under PGD attacks. These results further demonstrate that our methods are more robust against various types of attacks than the baselines. **Second**, when there are no attacks, the accuracy of our methods is much higher than that of RobNet. In addition to being more robust, our methods are also more accurate than RobNet under the attack-free setting. **Third**, DSRNA-CB is slightly more robust than DSRNA-Jacobian. Note that the search costs of methods in Table 1.3 are the same as those in Table 1.2 since the architectures were searched on CIFAR-10 and evaluated on ImageNet.

Table 1.4 shows the results on MNIST. Similarly, our methods are substantially more robust than RobNet-large under all types of attacks, and are substantially more robust than SDARTS-ADV and PC-DARTS-ADV under C&W attacks, AutoAttack (l_2), and AutoAttack (l_∞). Our methods are on par with SDARTS-ADV and PC-DARTS-ADV under PGD and FGSM attacks. When there is no attack, our methods achieve much higher accuracy than RobNet-large and are on par with SDARTS-ADV and PC-DARTS-ADV.

Runtime

With a single GTX 1080Ti GPU, the runtime on CIFAR-10 for the search phase of DSRNA-CB is 4 GPU days, while that of DSRNA-Jacobian is 0.4 GPU days. On MNIST, DSRNA-CB takes 1 GPU day for architecture search while DSRNA-Jacobian takes 0.2 GPU days. DSRNA-Jacobian is more efficient than DSRNA-CB, but is less robust than DSRNA-CB as shown previously.

Verification-based Evaluation

In this section, we use the certification method developed in Section 1.3.1 to find the certified lower bounds of the architectures searched by different methods. Larger lower bound indicates more robustness. Table 1.5 and Table 1.6 compare the averaged certified lower bounds of architectures searched by different methods on MNIST and CIFAR-10 under l_2 and l_∞ norms. As can be seen, the lower bounds achieved by our methods under various norms are larger than those achieved by baselines. This further demonstrates that our methods are more robust than these baseline methods.

1.5 Conclusion

To address the problem that existing neural architecture search (NAS) methods are vulnerable to adversarial attacks, we propose methods for differentiable search of robust architectures. We define two differentiable measures of architectures' robustness, based on certified robustness lower bound and Jacobian norm bound. Then we search for robust architectures by performing optimization in the architecture space with an objective of maximizing the robustness metrics. On various datasets, we demonstrate that our methods 1) are more robust to various norm-bound attacks than several robust NAS baselines; 2) are more accurate than baselines when there are no attacks; 3) have significantly higher certified lower bounds than baselines.

1.6 Appendix

Baselines

DARTS [122]:

DARTS is differentiable NAS method that uses one-shot shared parameter technique based on the continuous relaxation of the discrete architecture spaces, where gradient descent is applied to optimize the architecture.

PC-DARTS [194]:

Partially-Connected DARTS [122] uses edge normalization method and only a random subset of the channels in each step on DARTS to improve the accuracy and stability of the searching. PC-DARTS also, is much faster than DARTS with less memory cost.

SmoothDARTS [26]:

SmoothDARTS(SDARTS) uses either random smoothing or adversarial attack on DARTS-based methods to improve the stability in different search spaces, such as the search spaces that were introduced in R-DARTS [206]. This method not only can improve the accuracy and stability of the DARTS-based methods, but also it makes them more robust against adversarial attacks. Based on the results applying adversarial attack to the DARTS-based methods performs better with clean models and the models under the adversarial attacks. Thus, in our paper we compare our results mostly with SDARTS-ADV and PC-DARTS-ADV, which represent DARTS with adversarial training and PC-DARTS with adversarial training, respectively.

RobNet [71]:

RobNet uses one-shot NAS to obtain a large number of networks and then studies the patterns of architectures that are robust against adversarial attacks. This method suffers from large memory cost. RobNet proposes using dense connectivity and adding convolution operations to direct connection edges to improve robustness , and utilize a feature flow guided search (FSP) strategy to compute FSP matrix's distance between the clean data and the adversarial example to

see how robust is the network.

Derivation of Conv Block and Computing Constant Tensors

In this section, we use the same procedure as CNN-Cert [15] to derive the lower/upper bound constant tensors $A^r_{L,conv}$, $A^r_{U,conv}$, $B^r_{L,conv}$, and $B^r_{U,conv}$, which are functions of weights W^r, bias b^r, and linear bound parameters α_L, α_U, β_L, and β_U of bounded output Φ^r. (i,j,k) indicates the location in the weight filter. Thus, for (x,y,z)-th output $\Phi^r_{(x,y,z)}$ we get the following lower/upper bound constant tensors $A^r_{L,conv}$, $A^r_{U,conv}$, $B^r_{L,conv}$, and $B^r_{U,conv}$ in the convolutional block:

$$A^r_{L,conv} * \Phi^{r-1} + B^r_{L,conv} \leq \Phi^r \leq A^r_{U,conv} * \Phi^{r-1} + B^r_{U,conv} \tag{1.17}$$

Upper bound:

By applying the linear upper bounds on the activation functions $\sigma(.)$ we will get the following:

$$\Phi^r_{(x,y,z)} = W^r_{(x,y,z)} * \sigma(\Phi^{r-1}) + b^r_{(x,y,z)} \tag{1.18}$$

$$= \sum_{i,j,k} W^r_{(x,y,z),(i,j,k)} \cdot [\sigma(\Phi^{r-1})]_{(x+i,y+j,k)} + b^r_{(x,y,z)} \tag{1.19}$$

$$\begin{aligned} \leq \sum_{i,j,k} & W^{r+}_{(x,y,z),(i,j,k)} \alpha_{U,(x+i,y+j,k)} (\Phi^{r-1}_{(x+i,y+j,k)} + \beta_{U,(x+i,y+j,k)}) \\ & + W^{r-}_{(x,y,z),(i,j,k)} \alpha_{L,(x+i,y+j,k)} (\Phi^{r-1}_{(x+i,y+j,k)} + \beta_{L,(x+i,y+j,k)}) + b^r_{(x,y,z)} \end{aligned} \tag{1.20}$$

$$= A^r_{U,(x,y,z)} * \Phi^{r-1} + B^r_{U,(x,y,z)} \tag{1.21}$$

Therefore, upper bounds constant tensors can be computed as:

$$A^r_{U,conv(x,y,z)} = W^{r+}_{(x,y,z),(i,j,k)} \alpha_{U,(x+i,y+j,k)} + W^{r-}_{(x,y,z),(i,j,k)} \alpha_{L,(x+i,y+j,k)} \tag{1.22}$$

$$B^r_{U,conv(x,y,z)} = W^{r+}_{(x,y,z)} * (\alpha_U \bigodot \beta_U) + W^{r-}_{(x,y,z)} * (\alpha_L \bigodot \beta_L) \tag{1.23}$$

Lower bound:

By applying the linear lower bounds on the activation functions $\sigma(.)$ we will get the following:

$$\Phi^r_{(x,y,z)} = W^r_{(x,y,z)} * \sigma(\Phi^{r-1}) + b^r_{(x,y,z)} \tag{1.24}$$

$$= \sum_{i,j,k} W^r_{(x,y,z),(i,j,k)} \cdot [\sigma(\Phi^{r-1})]_{(x+i,y+j,k)} + b^r_{(x,y,z)} \tag{1.25}$$

$$\begin{aligned} \geq \sum_{i,j,k} & W^{r+}_{(x,y,z),(i,j,k)} \alpha_{L,(x+i,y+j,k)} (\Phi^{r-1}_{(x+i,y+j,k)} + \beta_{L,(x+i,y+j,k)}) \\ & + W^{r-}_{(x,y,z),(i,j,k)} \alpha_{U,(x+i,y+j,k)} (\Phi^{r-1}_{(x+i,y+j,k)} + \beta_{U,(x+i,y+j,k)}) + b^r_{(x,y,z)} \end{aligned} \tag{1.26}$$

$$= A^r_{L,(x,y,z)} * \Phi^{r-1} + B^r_{L,(x,y,z)} \tag{1.27}$$

Therefore, lower bounds constant tensors can be computed as:

$$A^r_{L,conv(x,y,z)} = W^{r+}_{(x,y,z),(i,j,k)} \alpha_{L,(x+i,y+j,k)} + W^{r-}_{(x,y,z),(i,j,k)} \alpha_{U,(x+i,y+j,k)} \tag{1.28}$$

$$B^r_{L,conv(x,y,z)} = W^{r+}_{(x,y,z)} * (\alpha_L \bigodot \beta_L) + W^{r-}_{(x,y,z)} * (\alpha_U \bigodot \beta_U) \tag{1.29}$$

Pooling Operations.

Similarly, by doing the same procedure, we compute the bounds of the pooling operations, as well:

$$\Phi^r_n = \max_{S_n} \Phi^{r-1}_{S_n} \tag{1.30}$$

where S_n, Φ^{r-1}, and Φ^r indicate the pooled input index, input of the pooling layer, and output of the pooling layer, respectively. And employing linear bounds to obtain the lower/upper bound of the pooling operation will leads us to the following formulation:

$$\begin{aligned} A^r_{L,pool} * \Phi^{r-1} + B^r_{L,pool} &\leq \Phi^r \\ &\leq A^r_{U,pool} * \Phi^{r-1} + B^r_{U,pool} \end{aligned} \tag{1.31}$$

where $A^r_{L,pool}$, $A^r_{U,pool}$, $B^r_{L,pool}$, and $B^r_{U,pool}$ can be computed as [15] proposes:

Let define:

$$\gamma_0 = \frac{\sum_i \frac{u_i}{u_i - l_i} - 1}{u_i - l_i} \tag{1.32}$$

$$\gamma = min\left\{max\left\{\gamma_0, max\{l_1..l_\infty\}\right\}, min\left\{u_1..u_\infty\right\}\right\} \tag{1.33}$$

Therefore, upper bounds tensors are:

$$A^r_{U,pool,(x,y,z),(i,j,k)} = \frac{u_{(x+i,y+j,z)} - \gamma}{u_{(x+i,y+j,z)} - l_{(x+i,y+j,z)}} \tag{1.34}$$

$$B^r_{U,pool,(x,y,z),(i,j,k)} = \sum_{i,j} \frac{(\gamma - u_{(x+i,y+j,z)}) l_{(x+i,y+j,z)}}{u_{(x+i,y+j,z)} - l_{(x+i,y+j,z)}} + \gamma \tag{1.35}$$

where u_i are the upper bounds and l_i are the lower bounds. Now let define $G = \sum_i \frac{u_i - \gamma}{u_i - l_i}$ and then using G we can compute η as following:

$$\eta = \begin{cases} min\{l_1, ..., l_\infty\} & ,if\ G < 1 \\ max\{u_1, ..., u_\infty\} & ,if\ G > 1 \\ \gamma & ,if\ G = 1 \end{cases} \tag{1.36}$$

And using that based on [15] we can compute the lower bounds tensors as:

$$A^r_{L,pool,(x,y,z),(i,j,k)} = \frac{u_{(x+i,y+j,z)} - \gamma}{u_{(x+i,y+j,z)} - l_{(x+i,y+j,z)}} \tag{1.37}$$

$$B^r_{L,pool,(x,y,z),(i,j,k)} = \sum_{i,j} \frac{(\gamma - u_{(x+i,y+j,z)})\eta}{u_{(x+i,y+j,z)} - l_{(x+i,y+j,z)}} + \eta \tag{1.38}$$

Certified Lower Bound.

Then, we compute the global bounds $\eta_{j,U}$ and $\eta_{j,L}$ of network output $\Phi^m(x)$ by considering the whole neural network as one building block. By employing the same procedure as we did to the ReLU-Conv-BN building block [15], we can compute global upper bound and lower bound using the following formulations:

$$\eta_{j,U} = \varepsilon \left\| vec(A^0_U) \right\|_q + A^0_U * x_0 + B^0_U \tag{1.39}$$

$$\eta_{j,L} = -\varepsilon \left\| vec(A^0_L) \right\|_q + A^0_L * x_0 + B^0_L \tag{1.40}$$

where $\frac{p+q}{pq}$ and $p, q \geq 1$.

Finally, we obtain the largest certified lower bound by increasing ε, if the global lower bound of the predicted class, with the given ε, is larger than the global upper bound of the targeted class; otherwise, we decrease it. Once we get the certified lower bound of the network,

we can compare our results to see which approach was more successful in searching for robust architecture against adversarial attacks.

Search Spaces

RobustDARTS (R-DARTS) [206] paper proposes four other search spaces beside the original DARTS space, which DARTS fails in them. To show the stability of our methods in other search spaces we will study our methods performances on these other four search spaces, which are as following:

- **S1:** uses a different set of two operators for each edge.
- **S2:** uses only 3×3 separable convolutions and skip connections as the candidate operations for each edge.
- **S3:** uses only 3×3 separable convolutions, skip connections, and Zero operation as the candidate operations for each edge.
- **S4:** uses only 3×3 separable convolutions and Noise operation $\varepsilon \sim \mathcal{N}(0,1)$ as the candidate operations for each edge.

Adversarial Attacks

We test our methods against three different attacks with different effective norms. These attacks are PGD, FGSM, and C&W. The RobNet and DARTS-based results are obtained from [71] and [48], respectively, where [71] tests RobNet against the PGD attack with total perturbation $\varepsilon = 0.03$ and [48] evaluates the DARTS-based methods (i.e., DARTS, P-DARTS, and PC-DARTS) against the PGD attack with total perturbation $\varepsilon = 0.3$ on CIFAR-10.

Algorithms

Based on Algorithm 1, we aim to search for an optimal architecture that yields the largest robustness and smallest predictive loss on the validation set by using one of the two robustness

metrics R in norm-bound l_p (i.e., Jacobian Regularization or Certified Bound) to compute the R and use it in validation loss with the trade-off parameter γ to maximize the robustness in the PC-DARTS search process. (Note: we use PC-DARTS since it is faster than other DARTS-based methods. Certified Bound technique for measuring the robustness metric operates slowly in the search process.)

Algorithm 1: Training DSRNA

Create a mixed operation for selected channels $f^{PC}_{i,j}(x_i; S_{i,j})$ for every edge (i, j)

while *not converged* **do**

1. Compute $\sum_{i=1}^{M} R(w^*(\alpha), \alpha, x_i^{(\text{val})})$ by using either Jacobian Regularization or Certified Bound techniques

2. Update architecture α by descending $\sum_{i=1}^{M} L(w^*(\alpha), \alpha, x_i^{(\text{val})}) - \gamma R(w^*(\alpha), \alpha, x_i^{(\text{val})})$

end

Additional Experiments

Table 1.7 shows the clean models test errors of our proposed methods compared to the previous approaches in 4 search spaces studied in section 1.6 on CIFAR-10 dataset. These results indicate that our methods have more stability and outperform the others in different search spaces.

Table 1.7. Comparison our methods to other DARTS-based methods in four various search spaces, from section 1.6, to compare the stability of the methods on clean models using CIFAR-10.

Search Spaces	DARTS	PC-DARTS	DARTS-ES	R-DARTS(L2)	SDARTS-RS	SDARTS-ADV	DSRNA-CB(ours)	DSRNA-Jacob(ours)
S1	3.84	3.11	3.01	2.78	2.78	2.73	2.69	2.70
S2	4.85	3.02	3.26	3.31	2.75	2.65	2.62	2.55
S3	3.34	2.51	2.74	2.51	2.53	2.49	2.57	2.50
S4	7.20	3.02	3.71	3.56	2.93	2.87	2.95	2.79

We perform analysis of variance test, which is a parametric statistical significance test, between the results of our methods and the results of the baselines under adversarial attacks to

confirm that our tests are statistically significant. Based on the Table 2 of the paper, the probability value (p-value) between our test results and baselines' on CIFAR-10 is $p-value = 0.008$ and $Statistics = 4.120$, which is less than 0.05 and rejects the null hypothesis. On ImageNet and MNIST from the results, we get the $p-value = p = 0.001$ and $Statistics = 11.262$, and $p-value = 0.002$ and $Statistics = 25.325$, respectively. To reproduce these values run the ANOVA.ipynb code in the Supplementary Materials zip file.

Figure 1.1 demonstrates the accuracy of our methods and the baselines vs PGD attack iterations with the total perturbation $\varepsilon = 0.03$. As it is shown, DSRNA-CB (blue line) is the most robust method among these approaches that have been tested in the figure 1.1. However, DSRNA-Jacob (orange line) reaches approximately the same accuracy and results as DSRNA-CB, while it is more memory efficient and faster.

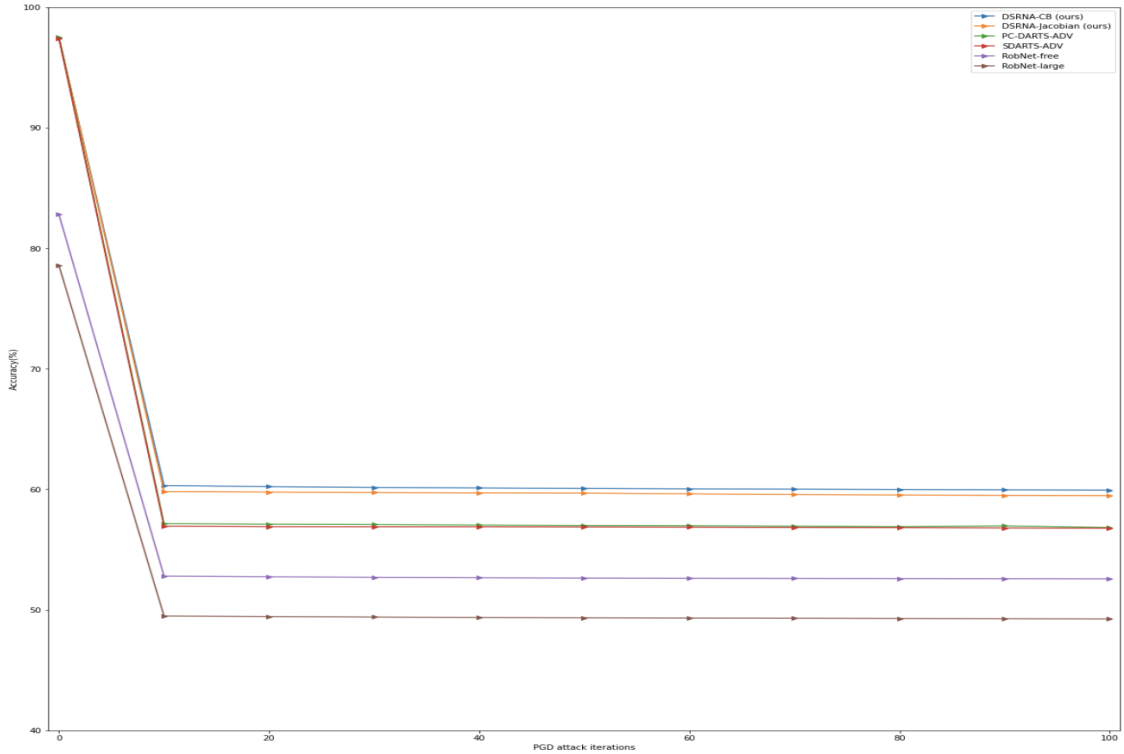


Figure 1.1. Accuracy comparison of our methods versus others against PGD attack with $\varepsilon = 8/255$ = (0.03) on CIFAR-10.

1.7 Acknowledgements

Chapter 1, in part, has been published in the Proceedings of the 2021 Computer Vision and Pattern Recognition (CVPR) Conference under the title “DSRNA: Differentiable Search of Robust Neural Architectures” by Ramtin Hosseini, Xingyi Yang, and Pengtao Xie. The dissertation author was the primary author of this material.

Chapter 2

Bias Mitigation in NAS for Fairness

2.1 Introduction

Learning by grouping is an outstanding human learning skill aiming to organize a set of given problems into different subgroups and domains where each subgroup contains similar problems that can be solved independently and efficiently. In this chapter, we formulate *Learning by Grouping* (LBG) as an optimization problem and investigate its effectiveness in ML. Our proposed framework contains two types of model: 1) Group Assignment Model (GAM); and 2) Group-Specific Classification Models (GSCM). The GAM model takes a data example as input and predicts the subgroup it belongs to – a K-way classification problem, where K is the number of GSCM models (i.e., experts). For each subgroup k, a GSCM model performs the supervised learning on the target task. We then apply the GAM and the K GSCM models to improve the existing machine learning models' fairness and accuracy. Additionally, we extend our LBG formulation to the neural architecture search to obtain the most suitable task-specific GSCM models. We depict the high-level learning process in Fig 2.1.

We formulate LBG as a three-stage optimization problem. First, we learn the Group-Assignment Model (GAM); then, we train Group-Specific Classification Models (GSCMs); finally, we apply the GAM and the GSCMs to the validation set to learn the subgroups for subgroup assignment and the learnable architecture. We develop a gradient-based method to solve this three-level optimization problem. In previous related works, mixture-of-expert

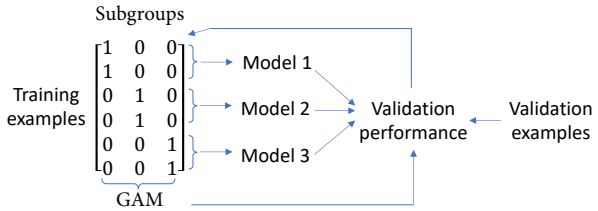


Figure 2.1. Illustration of Learning by Grouping (LBG) with three subgroups.

methods learn the experts – analogous to GSCMs; and the gating network – similar to GAM. The mixture-of-experts (MoE) methods learn the gating network and the experts jointly on the training data, which has a high risk of overfitting the gating network to the training data. We address this problem of overfitting by MoE methods by formulating a three-stage optimization framework that learns the subgroups for the subgroup assignment tasks on the validation set instead of the training examples.

Currently, the majority of state-of-the-art neural network performance is achieved through architectures that are manually designed by humans. However, this process of designing and evaluating neural network architectures by human experts is both time-consuming and may not end with the most suitable task-specific architecture. In recent years, there has been a growing interest in automating this manual process, referred to as neural architecture search (NAS). On the other hand, humans possess powerful learning skills that have been developed through evolution. This study also examines the potential of using a human-based learning technique, known as learning by grouping, in differentiable NAS approaches.

2.2 Related Works

2.2.1 Mixture of Experts

Lately, a wide variety of works [161, 210, 183] have proposed applying the mixture-of-experts (MoE) approach, which was initially proposed by [93], to varied deep learning tasks. Generally, deep learning MoE frameworks consist of expert networks and a gating function, where the gating function assigns each expert a subset of training data. The methods assume

a set of latent experts where each expert performs a classification or regression task. A gating function assigns the given data example to an expert. Then this example is classified using the classification model specific to this expert. The MoE has been an active research area aiming to improve the vanilla ML approaches, such as [161, 210, 183]. [161] introduces a trainable gating function to assign the experts' sparse combinations for the given data. DeepMOE [183] proposes a deep convolutional network including a shallow embedding network and a multi-headed sparse gating network, where the multi-headed sparse gating network uses the mixture weights computed by the shallow embedding network to select and re-weight gates in each layer. In MGE-CNN[210], experts are learned with the extra knowledge of their previous experts along with a Kullback-Leibler (KL) divergence constraint to improve the diversity of the experts. Recently, [155] proposed the Vision Transformer MoE (V-MoE) that can successfully reach state-of-the-art on ImageNet with approximately half of the required resources.

In the existing MoE methods, which are based on single-level optimization, the gating function and expert-specific Classification Models are learned jointly by minimizing the training loss. Hence, there is a high risk of the gating function overfitting the training data, which can lead to unfair and inaccurate decision-making. In our method, we address this issue via learning the group assignments of training examples by minimizing the validation loss instead and developing a multi-stage optimization problem rather than joint training. The results show the efficacy of our method.

2.2.2 Domain Adaptation

Domain adaptation (DA) is a technique in machine learning that aims to enhance the performance of models trained on one domain, known as the source domain, on a different yet related domain, referred to as the target domain. The objective is to transfer the knowledge acquired from the source domain to the target domain, where the input features and/or output labels may vary. This approach is particularly valuable in scenarios where the amount of labeled data in the target domain is scarce, but a large amount of labeled data is available in

the source domain. Different methods for domain adaptation [69, 66, 138, 96] can be classified into three main categories: instance-based, feature-based, and adversarial-based approaches. These methods mostly focus on measuring and minimizing the distance between the source and target domains. Some well-established distance measuring approaches include Maximum Mean Discrepancy (MMD) [127, 68], Correlation Alignment (CORAL) [169], Kullback-Leibler (KL) divergence [110], and Contrastive Domain Discrepancy (CDD) [101].

2.2.3 Multi-Level Optimization

In the past few years, Bi-Level Optimization (BLO) and Multi-Level Optimization (MLO) [179] techniques have been applied to Meta-Learning [58, 59], and Automated Machine Learning (AutoML) tasks such as neural architecture search [16, 122, 193, 196, 116, 85] and hyperparameter optimization [58, 12] to learn the meta parameters automatically and reduce the required resources and reliance on humans for designing such methods. Lately, inspired by humans' learning skills [192], several existing works [83, 32, 82, 61, 54, 84, 162, 53, 215] have borrowed these skills from humans and extended them to ML problems in MLO frameworks to study whether these techniques can assist the ML models in learning better.

2.3 Methods

Our method consists of a Group-Assignment Model (GAM) and K Group-Specific Classification Models (GSCMs). The GAM model predicts and assigns the training samples to their corresponding GSCM expert model. Then the GSCM models predict the classes of the inputs. Lastly, we apply the GAM and the GSCMs to the validation set and minimize the validation loss to learn the assignments of training samples. The illustration of our proposed method is shown in Fig 2.2. In Section 2.3.1, we first begin with defining the three-level optimization framework to formulate LBG (Section 2.3.1), and then we integrate domain adaptation techniques to our proposed LBG to mitigate the risk of overfitting (Section 2.3.1). Afterward, we extend the LBG to the Neural Architecture Search problem in Section 2.3.1. Finally, in Section 2.3.2, we develop

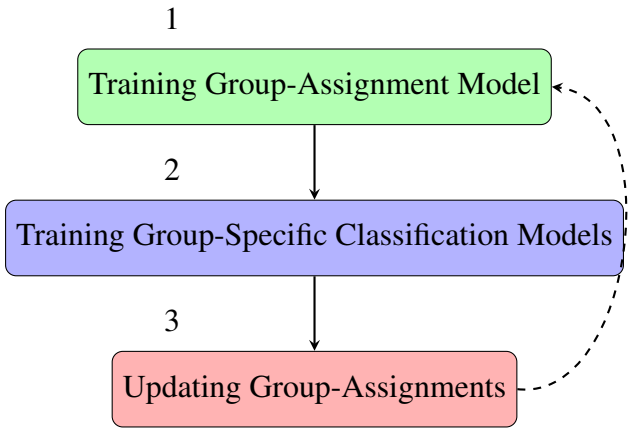


Figure 2.2. Overview of our proposed three-level optimization framework (Learning by Grouping).

an efficient optimization algorithm to address the three-level optimization problem.

2.3.1 Three-Level Optimization Framework

Our framework is composed of two types of models: the Group-Assignment Model (GAM) and the Group-Specific Classification Models (GSCM). The GAM model takes a data example as input and assigns it to one of the subgroups, which is a K-way classification problem, where K is the number of GSCM models (i.e., experts). We propose an end-to-end three-stage optimization problem where: First, the Group-Assignment Model (GAM) is learned; then, the Group-Specific Classification Models (GSCMs) are trained; finally, the GAM and the GSCMs are applied to the validation set to determine the group assignments and the learnable architecture. As shown in Fig 2.1, the Group-Assignment Model (GAM) is updated for training examples by validating the performance on the validation set, which distinguishes LBG from existing MoE approaches. As discussed in Section 2.3.1, the group assignments from GAM are continuous values $C_{nk} \in [0,1]$. Therefore, to convert these probability distributions to one-hot encoded format (similar to Fig 2.1) we can compute the top-k and obtain the k-hot encoded matrix, where k is one in this case.

Learning by Grouping (LBG)

We assume there are K latent subgroups. Let C be a matrix denoting the learnable 'ground-truth' grouping of the training samples. The size of C is $N \times K$ where N is the number of training examples - row n represents the grouping of the n-th training example. We relax the values in each row from a one-hot encoding to continuous values in order to perform gradient descent, so that $C_{nk} \in [0,1]$ denotes the probability that the n-th training example belongs to the k-th latent subgroup. Subgroups C are initialized randomly. The latent subgroup labels for subgroups are permutation-invariant. We then assign the n-th training example to subgroup j_n such that $j_n = \arg\max_e C_{ne}$, and let $G_n = C_{nj_n}$ be the probability of grouping the sample x_n to subgroup j_n. Let the GAM be represented by $f(x_n;T)$ with SoftMax output, which takes a data example x_n as input and predicts which subgroup x_n should be assigned to. T is the weights parameter of this network. The output of $f(x_n;T)$ is a K-dimensional vector, where the k-th element $f_k(x_n;T)$ denotes the probability that x_n should be assigned to the k-th subgroup. The sum of elements in $f(x_n;T)$ is one. Let $\hat{j}_n = \arg\max_e f_e(x_n;T)$, and let $E_n = f_{\hat{j}_n}(x_n;T)$ be the confidence of the GAM in assigning x_n the subgroup $\hat{j}_n$. We then have a GSCM classifier $f(x_n;S_{\hat{j}_n})$ for each latent subgroup $\hat{j}_n \in \{1 \dots K\}$, which predicts the class label for a data example x_n that has been assigned $\hat{j}_n$ as its GSCM by the GAM: $f(x_n;T)$. $S_{\hat{j}_n}$ are the network weights of this GSCM classifier.

Stage I.

In the first stage, we optimize the GAM: $f(x;T)$ given C by solving the following 'relaxed' negative log-likelihood optimization problem:

$$T^*(C) = \underset{T}{\operatorname{argmin}} \sum_{n=1}^{N} -G_n(C) \log f_{j_n}(x_n;T) \tag{2.1}$$

Note that we do not update the 'ground-truth' subgroups C in this stage.

Stage II.

In the second stage, we learn the K GSCM models. For each latent subgroup k, there is a GSCM classifier $f(x;S_k)$ with parameters S_k, which predicts the class label for a data example x assigned k as its subgroup by the GAM. Let $D_n = \{x_n\}$ denote the subset of training data examples assigned subgroup k by GAM. We want to learn S_k by minimizing the following loss:

$$\sum_{x_n \in D_n} \ell(f(x_n;S_k),y_n) \tag{2.2}$$

where y_n is the class label of x_n . $l(\cdot,\cdot)$ is the cross-entropy loss. In addition, we take into account the confidence of the GAM in assigning the training example x_n to its corresponding GSCM $\hat{j}_n$. So we relax the above equation, and summarize the total loss of all GSCM models and objective of this stage as:

$$\{S_k^*(T^*(C))\}_{k=1}^K = \underset{\{S_k\}_{k=1}^K}{\operatorname{argmin}} \sum_{n=1}^N E_n(x_n;T^*(C))\,\ell(f(x_n;S_{\hat{j}_n}),y_n) \tag{2.3}$$

where $S_k^*(T^*(C))$ for $k \in \{1\ldots K\}$ denotes the optimal solution set for the K GSCM classifiers.

Stage III.

Given $T^*(C)$ and $S_k^*(T^*(C))$, we apply them to make predictions on the validation examples and update the 'ground-truth' matrix C. The validation loss is:

$$\min_C \sum_{i=1}^M \ell\left(E_{\hat{j}_i}(x_i;T^*(C))\,f\left(x_i;S_{\hat{j}_i}^*(T^*(C))\right),y_i\right) \tag{2.4}$$

where $\hat{j}_i = \arg\max_e f_e(x_i;T^*(C))$ and M is the number of validation examples. y_i is the class label of x_i. We update C by minimizing this validation loss.

Putting these pieces together, we have the following optimization problem:

$$\min_C \sum_{i=1}^M \ell\left(E_{\hat{j}_i}(x_i;T^*(C))\,f\left(x_i;S_{\hat{j}_i}^*(T^*(C))\right),y_i\right)$$

$$s.t.\{S_k^*(T^*(C))\}_{k=1}^K = \underset{\{S_k\}_{k=1}^K}{\operatorname{argmin}} \sum_{n=1}^{N} E_n\left(x_n; T^*(C)\right) \ell(f(x_n; S_{\hat{j}_n}), y_n) \tag{2.5}$$

$$T^*(C) = \underset{T}{\operatorname{argmin}} \sum_{n=1}^{N} -G_n(C) \log f_{j_n}(x_n; T)$$

Domain Adaptive LBG

In our proposed *Learing by Grouping* (LBG) from Section 2.3.1, the N training examples are divided into K subgroups. As a result, each subgroup has approximately N/K training examples. The reduced number of training examples in small datasets can potentially lead to higher risk of overfitting for each subgroup. To address this problem, we propose domain-adaptive LBG (DALBG) where we treat each subgroup as a domain. During the second stage of our framework, when we are training a group-specific classifier for a subgroup k, we perform domain adaptation to adapt examples from other subgroups into subgroup k and use these adapted examples as additional training data for subgroup k. For the sake of simplicity, our proposed framework employs the MMD-based [127] domain adaptation approach. However, it should be noted that other domain adaptation techniques can also be incorporated within our framework. For a specific subgroup, k, let $\{x_i^k\}_{i=k}^{N_k}$ represent the examples assigned to this subgroup and $\{x_j^{-k}\}_{j=1}^{N-N_k}$ represent the examples not assigned to this subgroup. In order to adapt $\{x_j^{-k}\}_{j=1}^{N-N_k}$ into subgroup k, we minimize the Maximum Mean Discrepancy (MMD) loss as follows:

$$M_k = \left\| \frac{1}{N_k} \sum_{i=1}^{N_k} \phi\left(x_i^k; S_k\right) - \frac{1}{N-N_k} \sum_{j=1}^{N-N_k} \phi\left(x_j^{-k}; S_k\right) \right\|_2^2 \tag{2.6}$$

where $\phi\left(x_i^k; S_k\right)$ denotes the embedding of x_i^k extracted by S_k. This loss can be relaxed to:

$$M_k = \left\| \frac{1}{N} \sum_{n=1}^{N} f_k\left(x_n; T^*(C)\right) \phi\left(x_n; S_k\right) - \frac{1}{N} \sum_{n=1}^{N} \left(1 - f_k\left(x_n; T^*(C)\right)\right) \phi\left(x_n; S_k\right) \right\|_2^2 \tag{2.7}$$

Thus, by adding M_k to our second stage Eq. (2.3) we define the following *domain*

adaptive LBG (DALBG) problem:

$$\min_{C} \sum_{i=1}^{M} \ell\left(E_{\hat{j}_i}(x_i; T^*(C))\, f\left(x_i; S^*_{\hat{j}_i}(T^*(C))\right), y_i\right)$$

$$s.t. \{S^*_k(T^*(C))\}_{k=1}^{K} = \underset{\{S_k\}_{k=1}^{K}}{\operatorname{argmin}} \sum_{n=1}^{N} E_n(x_n; T^*(C))\, \ell(f(x_n; S_{\hat{j}_n}), y_n) + \lambda M_k \tag{2.8}$$

$$T^*(C) = \underset{T}{\operatorname{argmin}} \sum_{n=1}^{N} -G_n(C) \log f_{j_n}(x_n; T)$$

where λ is a tradeoff parameter. Note that our proposed LBG in Eq. (2.5) method is a special case of DALBG in Eq. (2.8) with $\lambda = 0$. For the sake of simplicity, we refer to both Learning by Grouping ***with/without*** domain adaptation as (DA)LBG.

Neural Architecture Search Application

In this section, we extend the formulation in Eq. (2.5) to be applicable to neural architecture search. Similar to [122], the k-th GSCM has a differentiable architecture A_k. The search space of A_k is composed of large number of building blocks, where the output of each block is associated with a weight a indicating the importance of the block. After learning, the block whose weight a is among the largest are retained to form the final architecture. To this end, architecture search amounts to optimizing the set of architecture weights $A_k = \{a\}$.

Stage I and **Stage II** have the same procedure as Eq. (2.1) and Eq. (2.7). In the second stage, the network weights S_k of the expert model are a function of its architecture A_k. We keep the architecture fixed at this stage, and learn the weights $S_k(A_k)$. However, **Stage III** does not precisely follow Eq. (2.4). Given $T^*(C)$ and $S^*_k(A_k, T^*(C))$, we apply them to make predictions on the validation examples and update the 'ground-truth' matrix C, as well as the architectures A_k based on the validation loss, where $k \in \{1 \dots K\}$. Hence, we update Eq. (2.4) as follows:

$$\min_{C, \{A_k\}_{k=1}^{K}} \sum_{i=1}^{M} \ell\left(E_{\hat{j}_i}(x_i; T^*(C))\, f\left(x_i; S^*_{\hat{j}_i}(A_{\hat{j}_i}, T^*(C))\right), y_i\right) \tag{2.9}$$

Thus, the overall optimization problem with learnable architecture is as follows:

$$\min_{C,\{A_k\}_{k=1}^{K}} \sum_{i=1}^{M} \ell\left(E_{\hat{j}_i}(x_i;T^*(C))\, f\left(x_i;S^*_{\hat{j}_i}(A_{\hat{j}_i},T^*(C))\right),y_i\right)$$

$$s.t.\{S^*_k(A_k,T^*(C))\}_{k=1}^{K} = \underset{\{S_k\}_{k=1}^{K}}{\operatorname{argmin}} \sum_{n=1}^{N} E_n(x_n;T^*(C))\,\ell(f(x_n;S_{\hat{j}_n}(A_{\hat{j}_n}),y_n)+\lambda M_k \tag{2.10}$$

$$T^*(C) = \underset{T}{\operatorname{argmin}} \sum_{n=1}^{N} -G_n(C)\log f_{j_n}(x_n;T)$$

Our framework is orthogonal to existing differentiable NAS methods, and hence can be applied on top of any like DARTS [122], P-DARTS [29], PC-DARTS [196], and DARTS− [35] among the others.

2.3.2 Optimization Algorithm

We promote an efficient algorithm to solve the LBG, DALBG, and LBG-NAS problems described in Eq. (2.5), Eq. (2.8), and Eq. (2.10), respectively. We utilize a fairly similar procedure as [122] to calculate the gradient of Eq. (2.1) w.r.t T and approximately update $T^*(C)$ via one-step gradient descent. Then since DALBG in Eq. (2.8) is the generalized version of LBG in Eq. (2.5), we plug the approximation $T^{'}(C)$ into the Eq. (2.7) to get an O_{S_k}, which denotes the approximated objective of S_k. Similarly to the previous step, we approximate $S^*_k(T^{'}(C))$ using a one-step gradient descent update of S_k based on the gradient of the approximated objective. Note that in LBG-NAS, we approximate $S^*_k(A_k,T^*(C))$, which is also a function of architecture A_k. Finally, we plug the approximations $T^{'}(C)$ and $S^{'}_k(T^{'}(C))$ into the third stage equations to get the third approximate objective denoted by O_C. C can be updated using gradient descent on O_C. In LBG-NAS, we update the architectures $\{A_k\}_{k=1}^{K}$, as well. Thus, use the same approach to find the approximate objective of the architectures $\{A_k\} : O_{\{A_k\}}$ for each $k \in \{1 \dots K\}$, and we update it using gradient descent. We do these steps until convergence.

2.4 Experiments

In this section, we investigate the effectiveness of our proposed (DA)LBG framework with both fixed human-designed GSCMs and searchable GSCMs. The differentiable NAS approach consists of architecture search and evaluation stages, where the optimal cell obtained from the search stage is stacked several times into a larger composite network. We then train the resultant composite network from scratch in the evaluation stage.

2.4.1 Datasets

Various experiments are conducted on four datasets: ISIC-18, CelebA, CIFAR-10, CIFAR-100, and ImageNet [47] for image classification. The CelebA dataset, consisting of 200k images of human faces with 40 features per image [126], is used in this study. From the dataset, we select a sample of 10,000 images, with 70% allocated for training, 15% for validation, and 15% for testing. The Skin ISIC 2018 dataset [37, 176] consists of a total of 11,720 dermatological images, specifically curated for the purpose of 7-class skin cancer classification. In this research paper, we have identified gender (male and female) as the sensitive attribute that may introduce bias. To mitigate this potential bias, we have performed a partitioning of the dataset into training, validation, and testing sets. The training set comprises 10,015 images, the validation set contains 1,512 images, and the testing set consists of 193 images, collectively representing the entirety of the dataset. The CIFAR-10 dataset contains of 10 distinct classes, while the CIFAR-100 dataset encompasses 100 classes. Each dataset holds 60K images. For each of the datasets, during grouping and architecture search processes, we use 25K images as the training set, 25K images as the validation set, and the rest of the 10K images as the test set. During grouping and architecture evaluations, the combination of the above training and validation set is used as the training set of size 50k images. ImageNet carries 1.2M training images and 50K test images with 1000 classes. Due to extensive amount of images in ImageNet, the architecture search can be pretty costly. Thus, following [196], we randomly choose 10%, and 2.5% of the 1.2M images

Table 2.1. Results on CelebA with the target label of "attractive" and sensitive attribute of "gender".

Methods	Error (%)	DP	DEO	Architecture
ResNet18	17.57	0.5023	0.5683	Manual
LBG-ResNet18 (ours)	<u>17.02</u>	<u>0.2173</u>	**0.0596**	Manual
DALBG-ResNet18 (ours)	**16.84**	**0.2116**	<u>0.0835</u>	Manual
DARTS	16.39	0.4571	0.3606	NAS
LBG-DARTS (ours)	<u>15.91</u>	**0.2149**	**0.0535**	NAS
DALBG-DARTS (ours)	**15.22**	<u>0.2185</u>	<u>0.0891</u>	NAS

to create a new training set and validation set, respectively, for the architecture search phase. Then, we utilize all the 1.2M images through the evaluation.

2.4.2 Experimental Settings

We compare the (DA)LBG image classification tasks with fixed architectures to the following MoE baselines: ResNet [78], Swin-T [125], T2T-ViT [204], DeepMOE [183], and MGE-CNN [210]. Next, we compare LBG-NAS on image classification with DARTS-based methods including DARTS [122], P-DARTS [29], and PC-DARTS [196]. To ensure the training costs of our methods with K GSCM models are similar to those of baselines, we reduce the parameter number of each expert to $1/K$ of the parameter number of the baseline models by reducing the number of layers in each GSCM model. In this way, the total size of our methods are comparable to the baselines. In addition, we train each group-specific sub-model only using examples assigned to its corresponding subgroup, rather than using all training examples. So the computation cost is $O(N)$ rather than $O(NK)$, where N is the number of training examples and K is the number of latent subgroups. In each iteration of the algorithm, we use minibatches of training examples to update sub-models, which further reduces computation cost. We utilize the Betty library [34] for the implementation of our multilevel optimization tasks.

Table 2.2. Results of ISIC when the sensitive attribute is "gender".

Methods	Error(%)	SPD	EOD	AOD
ResNet18	14.3	0.114	0.143	0.170
LBG-ResNet18 (ours)	**12.8**	**0.051**	**0.088**	**0.074**
DARTS	10.2	0.121	0.139	0.154
LBG-DARTS (ours)	**8.4**	**0.048**	**0.065**	**0.069**

Human-Designed GSCMs.

For experiments on CIFAR-10/100 and ImageNet datasets, we use ResNet [78], Swin-T [125], and T2T-ViT [204] models as our base GSCM models in the conducted experiments. For consistency and a fair comparison, we apply $K = 2$ latent subgroups to our four image classification datasets. To train our models, first we apply our proposed LBG training, where we use half of training images as the training set and the other half as the validation set, for 100 epochs with early stopping technique to obtain the optimal subgroups. Then, we use the obtained subgroups to fine-tune our GSCM models using the standard training settings with SGD optimizer for 200 epochs on the entire training examples. The initial learning rate is set to 0.1 with momentum 0.9 and will be reduced using a cosine decay scheduler with the weight decay of 3e-4. The batch size for CIFAR-10 and CIFAR-100 is set to 128, while for ImageNet we use the batch size of 1024. The rest of hyperparameter settings follows as [73]. In all DALBG experiments we use $\lambda = 0.1$. In this study, the Adam optimizer has been employed to train all models on the CelebA dataset, utilizing a learning rate of 5e-4, and implementing a batch size of 64. On the other hand, for the ISIC-18 experiments, we have set the learning rate to 1e-3, and the batch size to 32. For the experiments involving CelebA and ISIC-18, we leverage models that have been pretrained on ImageNet. Our models are trained for a range of 30 to 50 epochs, incorporating early stopping techniques to enhance efficiency.

GSCMs with Searchable Architectures.

We apply LBG to various DARTS-based approaches: DARTS [122], P-DARTS [29], and PC-DARTS [196]. The search spaces of these methods are the combination of (dilated) separable

Table 2.3. Test errors comparison of vanilla (base) models, baselines and LBG on CIFAR-10, CIFAR-100, and ImageNet.

Dataset	Model	Error(%)
CIFAR-10	ResNet56 (vanilla)	6.55
CIFAR-10	DeepMOE-ResNet56	6.03
CIFAR-10	MGE-CNN-ResNet56	5.91
CIFAR-10	LBG-ResNet56 (ours)	5.53
CIFAR-10	DALBG-ResNet56 (ours)	**5.47**
CIFAR-100	ResNet56 (vanilla)	31.46
CIFAR-100	DeepMOE-ResNet56	29.77
CIFAR-100	MGE-CNN-ResNet56	29.82
CIFAR-100	LBG-ResNet56 (ours)	27.96
CIFAR-100	DALBG-ResNet56 (ours)	**27.95**
ImageNet	ResNet18 (vanilla)	30.24
ImageNet	DeepMOE-ResNet18	29.05
ImageNet	MGE-CNN-ResNet18	29.30
ImageNet	LBG-ResNet18 (ours)	28.21
ImageNet	DALBG-ResNet18 (ours)	**28.08**
ImageNet	T2T-ViT-14 (vanilla)	17.16
ImageNet	LBG-T2T-ViT-14 (ours)	15.50
ImageNet	DALBG-T2T-ViT-14 (ours)	**15.47**
ImageNet	Swin-T (vanilla)	18.70
ImageNet	LBG-Swin-T (ours)	16.81
ImageNet	DALBG-Swin-T (ours)	**16.64**

Table 2.4. Test errors, number of model parameters (in millions), and search costs (GPU days on a Tesla v100) on CIFAR-100 and CIFAR-10. (DA)LBG-DARTS represents (DA)LBG applied to DARTS. Similar meanings hold for other notations in such a format.

Method	CIFAR-100			CIFAR-10		
	Error(%)	Param(M)	Cost	Error(%)	Param(M)	Cost
DARTS [122]	20.58±0.44	3.4	1.5	2.76±0.09	3.3	1.5
LBG-DARTS (ours)	18.02±0.36	3.6	1.7	**2.62±0.08**	3.5	1.6
DALBG-DARTS (ours)	**17.97±0.43**	3.7	2.0	2.64±0.12	3.6	2.0
PC-DARTS [196]	17.96±0.15	3.9	0.1	2.57±0.07	3.6	0.1
LBG-PCDARTS (ours)	16.21±0.19	4.1	0.3	2.51±0.11	3.7	0.3
DALBG-PCDARTS (ours)	**16.18±0.21**	4.2	0.4	**2.48±0.15**	3.8	0.4
P-DARTS [29]	17.49	3.6	0.3	2.50	3.4	0.3
LBG-PDARTS (ours)	16.46±0.54	3.7	0.6	2.48±0.16	3.5	0.5
DALBG-PDARTS (ours)	**16.39±0.48**	3.9	0.6	**2.47±0.19**	3.7	0.6

convolutions with two different sizes of 3×3 and 5×5, max pooling with the size of 3×3, average pooling with the size of 3×3, identity, and zero operations. Each LBG experiment was repeated five times with different random seeds. The mean and standard deviation of classification errors obtained from the experiments are reported.

In the architecture search stage, for CIFAR-10 and CIFAR-100, the architecture of each group-specific classification model contains 5 cells – reduced from 8 cells to 5 cells to match the parameter numbers of our baseline models – and each cell consists of 7 nodes. We use two group-specific sub-models (i.e., two subgroups $K = 2$) in the search process with the initial channels of 16. The search algorithm was based on SGD with a batch size of 64, the initial learning rate of 0.025 (reduced in later epochs using a cosine decay scheduler), epoch number of 50, weight decay of 3e-4, and momentum of 0.9. The rest of hyperparameters mostly follow the original settings in DARTS, P-DARTS, and PC-DARTS. For a fair comparison, in all the DALBG-NAS experiments $\lambda = 0.1$. For ISIC-18 and CelebA experiments, we utilize the same setting as described in the previous part.

During architecture evaluation, each GSCM sub-model is formed by stacking 11 copies (reduced from 20 layers to align with the baselines' sizes) of the corresponding optimally searched cell for CIFAR-10 and CIFAR-100 experiments. The initial channel number is set to 36. We train the networks with a batch size of 96 and 600 epochs on a single Tesla V100 GPU. For

evaluation of ImageNet, we use the searched architectures on CIFAR-10 and we stack 8 copies (similarly reduced from 14 layers to match the baselines' sizes) of obtained cells are stacked into each GSCM larger network, which was trained using four Tesla V100 GPUs on the 1.2M training images, with the batch size of 1024 and initial channel number of 48 for 250 epochs. Finally, for the evaluation of architecture in ISIC-18 and CelebA, we follow the same settings as those described for fixed human-designed GSCMs. However, as we don't have models pretrained on ImageNet available, we supplement our training with additional data for both CelebA and ISIC-18.

2.4.3 Results

First, we evaluate and compare the fairness of our proposed methods with the our baselines on CelebA dataset. In line with the methodology of [185], we use "attractive" as the binary class label for prediction, and as bias-sensitive attribute, we consider "gender" (male and female) in relation to the predicted labels. For evaluation we use Demographic Parity (DP) and Difference in Equalized Odds (DEO) metrices similar to [185]. The results of our experiments, as shown in Table 2.1, demonstrate that our proposed methods can improve accuracy while simultaneously mitigating unfair decision-making on minority groups. This is achieved through the use of group-specific models, which are trained on individual groups. We can also observe that DALBG improves accuracy more than LBG, but LBG achieves better fairness results on the DEO metric. This could be due to the fact that domain adaptation incorporated in DALBG may perpetuate or even amplify any existing biases present in the source domain, which may not be fully removed if the target domain is significantly different.

Table 2.2 demonstrates the results of fairness experiments on the ISIC-18 dataset where gender is the sensitive attribute and we use Statistical Parity Difference (SPD), Equal Opportunity Difference (EOD), and Average Odds Difference (AOD) as metrics to evaluate a model's fairness. This table shows that our method not only boosts accuracy performance, but also improves fairness by effectively mitigating bias in both fixed human-designed neural networks and NAS.

This performance improvement is attributable to our group-aware approach, which effectively groups similar samples with respect to unprotected sensitive attributes. This proves the advantage of our methods in addressing imbalanced attributes in the data.

Furthermore, in Table 2.3, we compare our proposed method with ResNet, Vision Transformers (Swin-T and T2T-ViT) ,and our MoE baselines (i.e., MGE-CNN and DeepMOE). The results in this table verify that our proposed method performs better than the baselines on all CIFAR-10, CIFAR-100, and ImageNet datasets considerably. This empirically verifies our claim that (DA)LBG reduces the overfitting risk found in MoE methods since the group assignments are learned by minimizing the validation loss during a multi-stage optimization.

Table 2.4 shows the comparison of our proposed methods and the existing works, which includes the classification errors with error bars, the number of model parameters, and search costs on CIFAR-10 and CIFAR-100 test sets. By comparing different methods, we make the following observation. Applying (DA)LBG to different NAS methods, including DARTS, P-DARTS, and PC-DARTS, the classification errors of these methods are greatly reduced. For instance, the original error of DARTS on CIFAR-100 is 20.58%; when DALBG is applied, this error is significantly reduced to 17.97%. As another example, after applying LBG to PC-DARTS and P-DARTS, the errors of CIFAR-100 experiments are decreased from 17.96% to 16.21% and 17.49% to 16.46%, respectively. Similarly for CIFAR-10, utilizing (DA)LBG in DARTS-based methods manages to reduce the errors and overfittings. These results strongly indicate the broad effectiveness of our framework in searching better neural architectures.

In Table 2.5, we compare different methods on ImageNet, in terms of top-1 and top-5 errors on the test set and number of model parameters, where the search costs are the same as the ones reported in Table 2.4. In these experiments, the architectures are searched on CIFAR-10 and evaluated on ImageNet similar to original DARTS [122]. DALBG-DARTS-CIFAR10 denotes that DALBG is applied to DARTS and performs search on CIFAR-10. Similar meanings hold for other notations in such a format. The observations made from these results are consistent with those made from Table 2.4. The architectures searched using our methods are consistently

Table 2.5. Results of ImageNet with gradient-based NAS methods.

Method	Top-1 Error (%)	Top-5 Error (%)	Param (M)
DARTS-CIFAR10 [122]	26.7	8.7	4.7
DALBG-DARTS-CIFAR10 (ours)	**24.9**	8.1	4.9
P-DARTS (CIFAR10) [29]	24.4	7.4	4.9
DALBG-PDARTS-CIFAR10 (ours)	**23.9**	6.9	5.0
PC-DARTS-CIFAR10 [196]	24.8	7.3	5.3
DALBG-PCDARTS-CIFAR10 (ours)	**23.1**	6.3	5.7

better than those searched by corresponding baselines. For example, DALBG-DARTS-CIFAR10 achieves 1.8% lower top-1 error than DARTS-CIFAR10. To the best of our knowledge, DALBG-PCDARTS-CIFAR10 is the new SOTA on mobile setting of Imagenet.

2.4.4 Ablation Studies

In this section, we conduct ablation studies to analyze the impact of individual components in our proposed frameworks.

Ablation on tradeoff parameter λ:

We study the effectiveness of tradeoff parameter λ in Eq. (2.8) on accuracy and fairness. We apply DALBG on CelebA dataset with two searchable GSCMs (i.e., $K = 2$) with the same setting as described in Section 2.4.2. In Table 2.6, we illustrate how the accuracy and fairness of DALBG on the test sets of CelebA are affected by increasing the tradeoff parameter λ. It can be observed that increasing λ from 0 to 0.1 leads to a decrease in fairness but an increase in accuracy, as a result of the MMD loss feedback. However, continuing to increase λ leads to a drop in accuracy as well. This is because placing too much emphasis on domain shift can result in less focus on in-domain performance ability.

Table 2.6. Ablation results on tradeoff parameter λ.

Methods	Error (%)	DEO
LBG-DARTS with $\lambda = 0$	15.91	**0.0535**
DALBG-DARTS with $\lambda = 0.01$	15.73	0.0754
DALBG-DARTS with $\lambda = 0.1$	**15.22**	0.0891
DALBG-DARTS with $\lambda = 1$	15.38	0.0917

Ablation on number of subgroups:

Next, we examine how different numbers of GSCM models with different number of subgroups $K \in \{1,2,3,4\}$ in Eq. (2.10) impact both accuracy and fairness performances. We apply (DA)LBG to DARTS. Table 2.7 indicates that for CelebA larger number of subgroups can decrease the classification error and improve the fairness. However, in our experiments number of subgroups $K = 3$ and $K = 4$ seem to achieve on par results, while $K = 3$ is more computationally efficient. The improved performance with a larger number of subgroups can be due to the fact that, in real life, many unprotected attributes may not be considered, but their combinations can still be used as proxies and affect decision-making processes. Thus, depending on the data and task, we can choose the most suitable number of subgroups, which can be different in various scenarios. Also, additional experiments and comparisons of (DA)LBG with bagging-based model ensemble can be found in the Supplements.

Table 2.7. Ablation results on number of subgroups.

Methods	Error (%)	DEO
LBG-ResNet18 with $K = 1$	17.59	0.5427
LBG-ResNet18 with $K = 2$	17.02	0.0596
LBG-ResNet18 with $K = 3$	16.88	0.0541
LBG-ResNet18 with $K = 4$	**16.85**	**0.0533**

Ablation on different domain adaptation techniques:

In this study, we aim to investigate the efficacy of different distance measuring approaches, namely Maximum Mean Discrepancy (MMD), Correlation Alignment (CORAL), Kullback-Leibler (KL) divergence, and Contrastive Domain Discrepancy (CDD), by incorporating them

into our framework. We conduct our experiments on DARTS with a similar experimental setup to Table 2.1. The results, presented in Table 2.8, indicate that MMD loss is the most effective approach in achieving both high accuracy and fairness compared to the other three methods. The superior performance of MMD in our framework can be attributed to its non-parametric nature and ability to capture non-linear relationships due to its kernel-based approach. In contrast, KL divergence relies on the assumption that both distributions are well-defined probability distributions and, along with CDD, may struggle to capture non-linear relationships in the data. Furthermore, while CORAL aligns the second-order statistics (i.e., covariance matrices) of the feature distributions, MMD maps the data into a Reproducing Kernel Hilbert Space (RKHS) using kernel functions. This capability enables MMD to capture more intricate relationships between data points, potentially resulting in improved performance within our framework. However, it is worth noting that the effectiveness of a domain adaptation technique may vary depending on the specific task and the degree of domain shift between the source and target domains. Thus, the choice of an appropriate technique should be based on the unique characteristics of the data and the task at hand.

Table 2.8. Ablation results on different domain adaptation techniques.

Methods	Error (%)	DEO
DALBG-DARTS-MMD	**15.22**	**0.0891**
DALBG-DARTS-CORAL	15.71	0.1137
DALBG-DARTS-KL	15.36	0.0945
DALBG-DARTS-CDD	15.80	0.1142

2.5 Conclusions and Discussion

In this chapter, we propose a novel MLO approach, called *Learning by Grouping* (LBG), drawing from humans' grouping-driven methodology of solving problems. Our approach learns to group a diverse set of problems into distinct subgroups where problems in the same subgroups

are similar; a group-specific solution is developed to solve problems in the same subgroups. We formulate our LBG as a multi-level optimization problem which is solved end-to-end. An efficient gradient-based optimization algorithm is developed to solve the LBG problem. We further incorporate domain adaptation in our framework to reduce the risk of overfitting. In our experiments on various datasets, we demonstrate that the proposed framework not only helps to mitigate overfitting and improve fairness, but also consistently outperforms baseline methods. The main limitation of LBG is that it cannot be applied to non-differentiable NAS approaches up to a point. In our future works, we will extend learning by grouping to reinforcement learning and evolutionary algorithms.

2.6 Appendix

2.6.1 Optimization Algorithm

We develop an efficient optimization algorithm to solve our proposed LBG problem. Notations are given in Table 2.9. We define group j_n such that $j_n = \arg\max_k C_{nk}$, and let $G_n = C_{nj_n}$ be the ground truth assignments of the sample x_n to group j_n. Let $\hat{j}_n = \arg\max_k f_k(x_n;T)$, and let $E_n = f_{\hat{j}_n}(x_n;T)$ be the confidence of the GAM in assigning x_n to the group $\hat{j}_n$.

Table 2.9. Notations used in LBG

Notation	Meaning
A_k	Architecture of the Group-Specific Classification Model (GSCM) of group k (Only in NAS applications)
S_k	Network weights of the Group-Specific Classification Models $f(x;S_k)$ of group k
T	Network weights of the Group Assignment Model (GAM) $f(x;T)$
C	Learnable 'ground-truth' categorization matrix
D_n	Training data
D_i	Validation data

We approximate $T^*(C)$ using one step gradient descent w.r.t $\sum_{n=1}^{N} -G_n(C)\log f_{j_n}(x_n;T)$:

$$T^*(C) \approx T' = T - \eta_t \nabla_T \sum_{n=1}^{N} -G_n(C)\log f_{j_n}(x_n;T) \tag{2.11}$$

Then we plug T' into $\sum_{n=1}^{N} E_n(x_n;T^*(C))\,\ell(f(x_n;S_{\hat{j}_n}(A_{\hat{j}_n}),y_n)$ and get an approximated objective.

And we approximate $S^*_{\hat{j}_n}(T^*(C))$ using one step gradient descent w.r.t the approximated objective:

$$S^*_{\hat{j}_n}(T^*(C)) \approx S'_{\hat{j}_n} = S_{\hat{j}_n} - \eta_s \nabla_{S_{\hat{j}_n}} \sum_{n=1}^{N} E_n\left(x_n; T'(C)\right) \ell(f(x_n; S_{\hat{j}_n}(A_{\hat{j}_n})), y_n) \tag{2.12}$$

Finally, we plug T' and $S'_{\hat{j}_n}$ into $\sum_{i=1}^{M} \ell\left(E_{\hat{j}_i}(x_i; T^*(C)) f\left(x_i; S^*_{\hat{j}_i}(A_{\hat{j}_i}, T^*(C))\right), y_i\right)$ and get an approximated objective. Then we update A by gradient descent:

$$C \leftarrow C - \eta_c \nabla_C \sum_{i=1}^{M} \ell\left(E_{\hat{j}_i}\left(x_i; T'(C)\right) f\left(x_i; S'_{\hat{j}_i}(A_{\hat{j}_i}, T'(C))\right), y_i\right), \tag{2.13}$$

where by applying chain rule it yields:

$$\begin{aligned} &\nabla_C \sum_{i=1}^{M} \ell\left(E_{\hat{j}_i}(x_i; T^*(C)) f\left(x_i; S^*_{\hat{j}_i}(A_{\hat{j}_i}, T^*(C))\right), y_i\right) = \\ &\frac{\partial T'}{\partial C} \sum_{i=1}^{M} \frac{\partial S'_{\hat{j}_i}}{\partial T'} \nabla_{S'_{\hat{j}_i}} \ell\left(E_{\hat{j}_i}\left(x_i; T'(C)\right) f\left(x_i; S'_{\hat{j}_i}(A_{\hat{j}_i}, T'(C))\right), y_i\right) + \\ &\frac{\partial T'}{\partial C} \nabla_{T'} \sum_{i=1}^{M} \ell\left(E_{\hat{j}_i}\left(x_i; T'(C)\right) f\left(x_i; S'_{\hat{j}_i}(A_{\hat{j}_i}, T'(C))\right), y_i\right) \end{aligned} \tag{2.14}$$

and $\frac{\partial T'}{\partial C}$ and $\frac{\partial S'_{\hat{j}_i}}{\partial T'}$ are computed as follows:

$$\frac{\partial T'}{\partial C} = -\eta_t \nabla^2_{C,T} \sum_{n=1}^{N} -G_n(C) \log f_{j_n}(x_n; T) \tag{2.15}$$

$$\frac{\partial S'_{\hat{j}_i}}{\partial T'} = -\eta_s \nabla^2_{T', S_{\hat{j}_i}} \sum_{n=1}^{N} E_n\left(x_n; T'(C)\right) \ell(f(x_n; S_{\hat{j}_n}(A_{\hat{j}_n})), y_n) \tag{2.16}$$

For the NAS applications we also update architectures $A_{\hat{j}_i}$, where $\hat{j}_i \in K$:

$$A \leftarrow A - \eta_a \nabla_A \sum_{i=1}^{M} \ell\left(E_{\hat{j}_i}\left(x_i; T'(C)\right) f\left(x_i; S'_{\hat{j}_i}(A_{\hat{j}_i}, T'(C))\right), y_i\right), \tag{2.17}$$

similar to group updating in Eq. 2.14, we apply chain rule as follows:

$$\nabla_A \sum_{i=1}^{M} \ell\left(E_{\hat{j}_i}\left(x_i; T^*(C)\right) f\left(x_i; S^*_{\hat{j}_i}(A_{\hat{j}_i}, T^*(C))\right), y_i\right) = \tag{2.18}$$

$$\sum_{i=1}^{M} \frac{\partial S'_{\hat{j}_i}}{\partial A_{\hat{j}_i}} \nabla_{S'_{\hat{j}_i}} \ell\left(E_{\hat{j}_i}\left(x_i; T'(C)\right) f\left(x_i; S'_{\hat{j}_i}(A_{\hat{j}_i}, T'(C))\right), y_i\right)$$

and $\frac{\partial S'_{\hat{j}_i}}{\partial A_{\hat{j}_i}}$ can be computed using the following equation:

$$\frac{\partial S'_{\hat{j}_i}}{\partial A_{\hat{j}_i}} = -\eta_s \nabla^2_{A_{\hat{j}_i}, S_{\hat{j}_i}} \sum_{n=1}^{N} E_n\left(x_n; T'(C)\right) \ell(f(x_n; S_{\hat{j}_n}(A_{\hat{j}_n})), y_n) \tag{2.19}$$

This algorithm is summarized in Algorithm 2.

Algorithm 2: Optimization algorithm for Learning by Grouping

while *not converged* **do**

1. Update the group assignment model's weights T using Eq. 2.11.
2. Update the group-specific classification models' weights $\{S_k\}_{k=1}^{K}$ using Eq. 2.12.

if *NAS application* **then**

3. Update the group-assignment matrix C and the group-specific classification models' architectures $\{A_k\}_{k=1}^{K}$ using Eq. 2.13 and Eq. 2.17.

else

3. Only update the group-assignment matrix C using Eq. 2.13.

end

end

2.6.2 Additional Experiments

2.6.3 Comparison with Bagging-based Model Ensemble

In this section we compare our proposed method (LBG) with bagging-based model ensemble, which uses three models (the same as our method). Table 2.10 demonstrates the results. Our method works better than model ensemble because it uses a divide-and-conquer

strategy. It divides data examples into groups where examples in the same group are similar; then for each group, an expert model is learned. Divide-and-conquer makes model training easier, because it is easier to train a highly-performant model for a group of similar examples than for a mixture of dissimilar examples from different groups. In ensemble learning, each model is trained on a mixture of dissimilar examples from different groups, which is a harder problem to solve. Additionally, in our method, the expert for each group can capture the unique data patterns in that group. Capturing group-specific data patterns can help to make more accurate predictions. In contrast, each model in ensemble learning is trained on all examples from different groups, which does not take group-specific data patterns into account.

Table 2.10. Comparison of our work with existing bagging-based model ensemble on CIFAR-100.

Methods	**Test error (%)**
Ensemble+DARTS-2nd	19.66±0.34
LBG-DARTS-2nd (ours)	**18.02**±0.36
Ensemble+P-DARTS	17.32±0.27
LBG-PDARTS (ours)	**16.46**±0.54

Language Tasks

In this section, we apply LBG with fixed human-designed architectures to language understanding tasks. We conducted experiments on the various tasks of the General Language Understanding Evaluation (GLUE) benchmark [181]. GLUE contains nine tasks, which are two single-sentence tasks (CoLA and SST-2), three similarity and paraphrase tasks (MRPC, STS-B, and QQP), and four inference tasks (MNLI, QNLI, RTE, WNLI). We test the performance of LBG in language understanding by submitting our inference results to the GLUE evaluation server. GLUE offers training and development data splits, that are used as training and validation data. For the test dataset, and GLUE organisers provide a submission server that reports the performance on the private held out test dataset.

Table 2.11. Comparison of BERT-based and RoBERTa-based experiments on GLUE sets.

Corpus	**BERT**	**LBG-BERT**	**RoBERTa**	**LBG-RoBERTa**
CoLA (Matthews Corr.)	60.5	**62.8**	68.0	**69.5**
SST-2 (Accuracy)	94.9	**96.5**	96.4	**96.8**
MRPC (Accuracy/F1)	85.4/89.3	**86.2/89.5**	**90.9**/92.3	90.2/**92.4**
STS-B (Pearson/Spearman Corr.)	87.6/86.5	**88.4/87.9**	92.4/92.0	**92.5/92.3**
QQP (Accuracy/F1)	89.3/72.1	**89.6/72.3**	92.2/-	**92.6/77.0**
MNLI (Matched/Mismatched Accuracy)	**86.7/85.9**	86.5/**85.9**	90.2/90.2	**91.1/91.1**
QNLI (Accuracy)	92.7	**93.5**	94.7	**94.9**
RTE (Accuracy)	70.1	**72.4**	86.6	**86.7**
WNLI (Accuracy)	65.1	**66.3**	**91.3**	86.3

Experimental Settings

We examine our proposed method by conducting varied experiments on several different tasks and datasets. We compare LBG on language understanding tasks with fixed architectures using BERT [50] and RoBERTa [124]. BERT [50] and RoBERTa [124] initialize the Transformer encoder with pre-trained BERT and RoBERTa, respectively, with the intentions of masked language modeling and next sentence prediction. Then, they utilize the pre-trained encoder and a classification head to build a text classification model. This text classification model latter will be fine-tuned on a target classification task.

To examine our method in language understanding, we employ BERT and RoBERTa as the group-specific sub-models with $K = 4$ latent subgroups on the GLUE tasks. LBG-BERT and LBG-RoBERTa are optimized using Adam optimizer [141]. The maximum length of text was set to 512 tokens. Our hyperparameter settings for BERT and RoBERTa experiments are the same as in [73]. Each GLUE task has a different batch size, learning rate, and number of epochs, where they are within the batch sizes $\in \{16, 32\}$, learning rates $\in \{1e^{-5}, 2e^{-5}, 3e^{-5}, 4e^{-5}\}$, and number of epochs $\in \{3, 4, 5, 6, 10\}$.

Results

Table 2.11 demonstrates the comparison of our methods with BERT and RoBERTa methods on nine different GLUE tasks. It is shown in this table that LBG can efficiently enhance the performance of existing base models in various language understanding tasks. In most of the

tasks, LBG-BERT and LBG-RoBERTa outperform BERT and RoBERTa, respectively. In MNLI and MRPC, the results of our methods are on par with the baselines, while RoBERTa achieves a slightly better result than our methods on the WNLI task.

2.7 Acknowledgements

Chapter 2, in part, has been published in the Proceedings of the 2023 International Conference on Machine Learning (ICML) under the title “Fair and Accurate Decision Making through Group-Aware Learning” by Ramtin Hosseini, Bhanu Garg, Li Zhang, and Pengtao Xie. The dissertation author was the primary author of this material.

Chapter 3

Advancing Generalizability in NAS with Self-Training

3.1 Introduction

The rapid advancement of deep learning techniques is assisting with resolving social difficulties in various fields. One of these fields that has recently attracted the researchers' attention is image understanding (e.g., image captioning). *Image Captioning* is a multi-model task that evaluates the computer's ability to understand images by generating the language descriptions of the input images, as shown in Figure 3.1. Solving this visual-language problem can be challenging, considering the complexity of understanding the relationships of recognized critical objects in the images. Since the architecture design of deep neural networks plays a critical role in the performance enhancement of the model, researchers have proposed a significant number of techniques in order to enhance the performance of their models by designing proper architectures for the encoder and the decoder modules for various tasks. However, obtaining high-performance human-designed architectures for each dataset with different distributions is exhausting and time-consuming.

Recently, neural architecture search (NAS) has achieved remarkable progress in obtaining the optimal architectures automatically, which helps attain better performances in computer vision and natural language processing applications. Nevertheless, most researchers in this area have focused on applying NAS methods to language modeling [216], image classification[218,

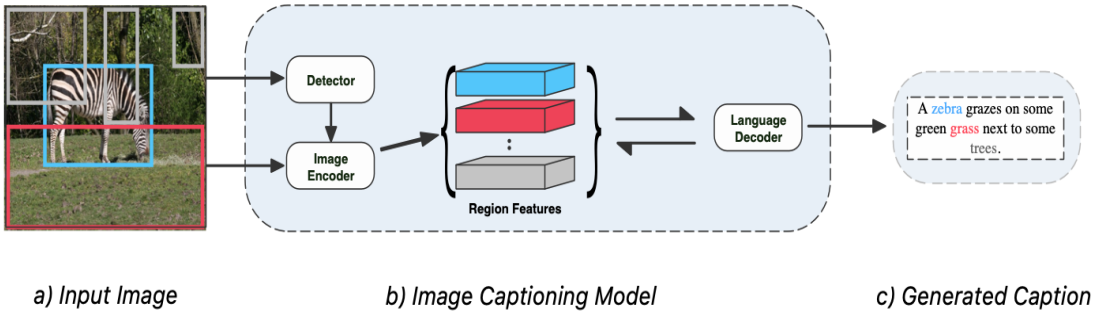


Figure 3.1. Overview of an *Image Captioning* task.

219, 195, 144, 60, 82], and adversarial training[85], while image captioning study with NAS is still largely underexplored. Several works have been investigated in employing NAS methods on image captioning, such as [216], where they focus only on searching for the architecture of the decoder module (i.e., language generation module) by using Reinforcement Learning [216].

In this chapter, we propose a three-stage optimization problem, called Image Understanding by Captioning (IUC), that applies differentiable architecture search [122] on image captioning tasks. Unlike the previous related works [216] that apply Reinforcement Learning based (RL-based) NAS methods only on decoder modules, we utilize differentiable architecture search based (DARTS-based) approaches on both encoder and decoder modules to improve our model's performance and, also, study the effectiveness and importance of each module. Extensive experiments on COCO datasets [118] explicate that our proposed methods outperform the existing strategies and can achieve state-of-the-art performances in image captioning.

3.2 Related Works

3.2.1 Image Captioning

Image Captioning is a multi-modal task that generates textual descriptions of input images. In order to perform this vision-language process, we need an encoder-decoder framework, where the encoders take the input images and create embeddings to feed them into the decoders to generate captions. Various techniques have been introduced in the past few years to enhance image captioning. Early works [133] in this area mostly use CNN as the image encoder, and LSTM [180] and RNN [203] as the language decoder to generate the captions of the input images. Later on, various attempts [33] showed performance improvements by applying attention mechanisms for more information exchange between the encoder and decoder modules. In several recent works, significant progress has been made with transformers [39] architectures. On the other hand, various approaches have been made to enhance the object detection task part of the image captioning, such as employing grid feature, region feature, and relation-aware visual feature. In the most recent works [115], researchers exhibit that vision-language pre-training on large image-text datasets can improve image captioning performances significantly. Moreover, several other recent works have been studying the importance of the image-captioning encoder-decoder architecture [8] by investigating different encoder-decoder architectures' performances versus their model sizes. Despite all the progress that has been made in image captioning tasks over the past decade, most of the existing image captioning models suffer from the design of their encoder and decoder architectures, which are fixed human-designed. Recently, AutoCaption [216] has proposed applying reinforcement learning-based neural architecture search (NAS) to image captioning tasks in order to design a better language decoder on the X-LAN [139]. Since reinforcement learning-based neural architecture search methods are mainly expensive for computer vision tasks (e.g., image classification), the existing image captioning methods that utilize NAS are inefficient in searching for the image encoder architecture. To address this problem, we propose a novel method that uses differentiable architecture search techniques to

obtain the optimal task-specific image encoder and language decoder architectures for image captioning tasks.

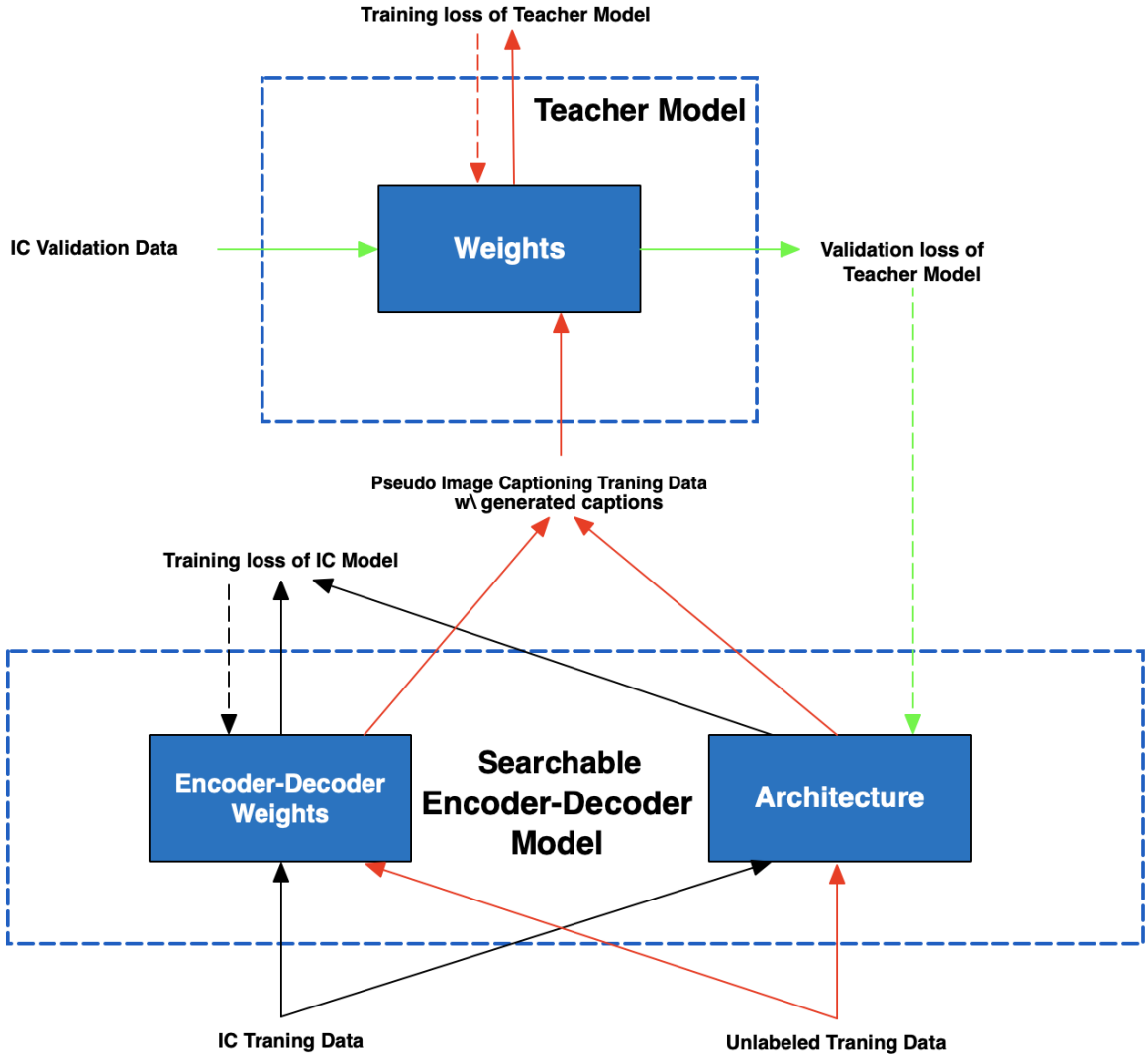


Figure 3.2. Overview of Image Understanding by Captioning (IUC) optimization framework.

3.3 Methods

In this section, we propose a novel method called Image Understanding by Captioning (IUC), where we apply differentiable architecture search techniques to obtain the optimal architectures for the encoder-decoder model. Inspired by [122], the architecture cells are directed

acyclic graphs (DAG) with N nodes (i.e., latent representations) and directed edges, representing the operation between the corresponding nodes. Our three-level optimization framework contains an encoder-decoder image captioning model with searchable architecture and a predictive image captioning (IC) model with fixed human-designed architecture. The searchable encoder-decoder model learns to take an image and generate text descriptions of the given image. Then, using the learned encoder-decoder model, we generate a pseudo image captioning dataset from the unlabeled dataset, and we train our predictive model on the new pseudo-IC dataset. Lastly, the trained predictive model validates its performance on the validation set of the IC dataset and minimizes its validation loss. To independently investigate the effectiveness of the image encoder module and language decoder module, we perform image encoder and language decoder architecture searches individually. Figure 3.2 illustrates our three-level optimization framework (IUC), where the solid arrows represent that the predictions are made and training/validation losses are determined; and the dotted arrows denote that the gradient updates of network weights and architecture variables are determined and weights/architecture are updated.

3.3.1 Image Understanding by Captioning

In our framework, there are three learning stages. In the **first stage**, a model learns to create image captions. The model has an encoder-decoder architecture. The encoder takes an input image and produces an embedding. The embedding is fed into the decoder, which decodes a textual description. We use the image captioning datasets to train the encoder and decoder by solving the following problem:

$$E^*(A), F^*(A) = \underset{E,F}{\operatorname{argmin}} \; L(E, A, F, D^{(tr)}) \tag{3.1}$$

where F and E denote the network weights of the decoder and the encoder, respectively, and A represents the architectures of the encoder and the decoder. $D^{(tr)}$ is an image captioning dataset.

Algorithm 3: Optimization algorithm for image understanding by captioning

while *not converged* **do**

1. Update encoder weights:
   $E \leftarrow E - \eta_e \nabla_E L(E, A, F, D^{(tr)})$
2. Update decoder weights:
   $F \leftarrow F - \eta_f \nabla_F L(E, A, F, D^{(tr)})$
3. Update predictive model weights:
   $W \leftarrow W - \eta_w \nabla_W L(W, U, E', F')$
4. Update the encoder-decoder architecture:
   $A \leftarrow A - \eta_a \nabla_A L(W', D^{(val)}))$

end

In the **second stage**, we use the trained encoder $E^*(A)$ and decoder $F^*(A)$ with searchable architectures from the first stage to generate a pseudo image captioning dataset using unlabeled images U. Then, using this new dataset, we train the network weights W of the predictive model with a fixed architecture by minimizing the following training loss:

$$W^*(E^*(A), F^*(A)) = \underset{W}{\operatorname{argmin}} \; L(W, U, E^*(A), F^*(A)) \tag{3.2}$$

Finally, we evaluate $W^*(E^*(A), F^*(A))$ in the **third stage** on the image captioning validation set and update A by minimizing the validation loss:

$$\min_{A} \; L(W^*(E^*(A), F^*(A)), D^{(val)}) \tag{3.3}$$

The predictive model in our framework assists the initial image captioning model to obtain the optimal architecture by testing its caption generating performance. Putting these pieces together, we get the following optimization problem:

$$\begin{aligned} &\min_{A} \; L(W^*(E^*(A), F^*(A)), D^{(val)}) \\ &s.t. \; W^*(E^*(A), F^*(A)) = \underset{W}{\operatorname{argmin}} \; L(W, U, E^*(A), F^*(A)) \\ &\quad E^*(A), F^*(A) = \underset{E,F}{\operatorname{argmin}} \; L(E, A, F, D^{(tr)}) \end{aligned} \tag{3.4}$$

The architecture searches for the image encoder and the language decoder modules are similar to Conventional Cell Search and Recurrent Cell Search as proposed in DARTS [122], respectively. The encoder during the architecture search contains 8 optimal cells, and the decoder is a single cell. Our proposed IUC framework is orthogonal to the various differentiable NAS methods, and it can be employed on any DARTS-based method, including DARTS [122], P-DARTS [195], PC-DARTS [194], and DATA [20].

Image Encoder Architecture Search.

In order to obtain the optimal image encoder architecture for image captioning on a particular dataset, we search for convolutional cells similar to DARTS [122] to optimize the encoder module architecture A using Eq.3.4. Inspired by [122], the encoder module is a convolution network built by stacking the learned cells together. The search spaces for the encoder architectures include (dilated) separable convolutions with sizes of 3×3 and 5×5, max pooling with the size of 3×3, average pooling with the size of 3×3, identity, and zero.

Table 3.1. Notations in Image Understanding by Captioning (IUC).

Notation	Meaning
E	Encoder weights
F	Decoder weights
W	Network weights of predictive model
A	Encoder-Decoder architecture
$D^{(tr)}$	Image captioning training set
$D^{(val)}$	Image captioning validation set
U	Unlabeled image dataset
η_e	Learning rate of E
η_f	Learning rate of F
η_w	Learning rate of W
η_a	Learning rate of A

Language Decoder Architecture Search.

To design the language decoder module, we perform the architecture searching for recurrent cells analogous to DARTS [122]. In this case, The architecture A in Eq.3.4 represents the decoder architecture, where the learned cells are recursively connected in order to build

the recurrent network of the language decoder. Our defined primitive operations in the search space for the recurrent cell of the language decoder include linear transformations followed by activation functions, the identity mapping, and the zero operation. The activation functions are chosen from one of the following: *relu*, *tanh*, *sigmoid*, *elu*, *celu*, or *gelu*.

3.3.2 Optimization Algorithm

In this section, we develop an optimization algorithm to solve the problem in Eq.(3.4) with our defined notations from Table 3.1. We approximate $E^*(A)$ and $F^*(A)$ using one-step gradient descent w.r.t $L(E,A,F,D^{(tr)})$:

$$E^*(A) \approx E' = E - \eta_e \nabla_E L(E,A,F,D^{(tr)}) \tag{3.5}$$

$$F^*(A) \approx F' = F - \eta_f \nabla_F L(E,A,F,D^{(tr)}) \tag{3.6}$$

We plug E' and F' into $L(W,U,E^*(A),F^*(A))$ and get an approximated objective. We approximate $W^*(E^*(A),F^*(A))$ using one-step gradient descent w.r.t the approximated objective:

$$W^*(E^*(A),F^*(A)) \approx W' = W - \eta_w \nabla_W L(W,U,E',F') \tag{3.7}$$

We plug W' into $L(W^*(E^*(A),F^*(A)),D^{(val)})$ and get an approximated objective. Thus, we update A using gradient descent:

$$A \leftarrow A - \eta_a \nabla_A L(W',D^{(val)}), \tag{3.8}$$

where by applying chain rule to the approximate architecture gradient Eq. 3.8, we get:

$$\nabla_A L(W',D^{(val)}) = (\frac{\partial E'}{\partial A}\frac{\partial W'}{\partial E'} + \frac{\partial F'}{\partial A}\frac{\partial W'}{\partial F'})\nabla_{W'} L(W',D^{(val)}) =$$

$$(\eta_e \eta_w \nabla^2_{A,E} L(E,A,F,D^{(tr)})\nabla^2_{E',W} L(W,U,E',F') + \eta_f \eta_w \nabla^2_{A,F} \tag{3.9}$$

$$L(E,A,F,D^{(tr)}))\nabla^2_{F',W}L(W,U,E',F'))\nabla_{W'}L(W',D^{(val)})$$

The overall algorithm of IUC is shown in Algorithm 3.

Table 3.2. Comparison of our methods and the state-of-the-art image captioning models on the COCO "Karpathy" test split (single-model). Methods with † are using NAS methods.

	Cross-Entropy Loss						**Encoder-Decoder**
	BLEU-1	BLEU-4	METEOR	ROUGE	CIDEr	SPICE	**Architecture**
LSTM [180]	-	29.6	25.2	52.6	94.0	-	manual
SCST [152]	-	30.0	25.9	53.4	99.4	-	manual
LSTM-A [202]	75.4	35.2	26.9	55.8	108.8	20.0	manual
RFNet [98]	76.4	35.8	27.4	56.5	112.5	20.5	manual
Up-Down [6]	77.2	36.2	27.0	56.4	113.5	20.3	manual
GCN-LSTM [201]	77.3	36.8	27.9	57.0	116.3	20.9	manual
LBPF [146]	77.8	37.4	28.1	57.5	116.4	21.2	manual
SGAE [198]	77.6	36.9	27.7	57.2	116.7	20.9	manual
AoANet [90]	77.4	37.2	28.4	57.5	119.8	21.3	manual
X-LAN [139]	78.0	38.2	28.8	58.0	122.0	21.9	manual
X-Transformer[139]	77.3	37.0	28.7	57.5	120.0	21.8	manual
$OSCAR_L$ [115]	-	37.4	30.7	-	127.8	23.5	manual
$OSCAR+_L$ w/ VINVL [211]	-	38.5	30.4	-	130.8	23.4	manual
AutoCaption [216]†	79.4	39.2	29.0	58.6	125.2	22.4	RL
IUC-D (ours) †	79.6	39.5	30.8	58.9	130.6	**23.8**	gradient-based
IUC-E (ours) †	**79.9**	**40.0**	**30.9**	**59.3**	**131.1**	23.7	gradient-based
	CIDEr Score Optimization						**Encoder-Decoder**
	BLEU-1	BLEU-4	METEOR	ROUGE	CIDEr	SPICE	**Architecture**
LSTM [180]	-	31.9	25.5	54.3	106.3	-	manual
SCST [152]	-	34.2	26.7	55.7	114.0	-	manual
LSTM-A [202]	78.6	35.5	27.3	56.8	118.3	20.8	manual
RFNet [98]	79.1	36.5	27.7	57.3	121.9	21.2	manual
Up-Down [6]	79.8	36.3	27.7	56.9	120.1	21.4	manual
GCN-LSTM [201]	80.5	38.2	28.5	58.3	127.6	22.0	manual
LBPF [146]	80.5	38.3	28.5	58.4	127.6	22.0	manual
SGAE [198]	80.8	38.4	28.4	58.6	127.8	22.1	manual
AoANet [90]	80.2	38.9	29.2	58.8	129.8	22.4	manual
X-LAN [139]	80.8	39.5	29.5	59.2	132.0	23.4	manual
X-Transformer[139]	80.9	39.7	29.5	59.1	132.8	23.4	manual
Meshed-Memory Transformer [39]	80.8	39.1	29.2	58.6	131.2	22.6	manual
X-Transformer+PPO [209]	81.1	39.7	29.6	59.2	133.3	23.4	manual
$OSCAR_L$ [115]	-	41.7	30.6	-	140.0	24.5	manual
$OSCAR+_L$ w/ VINVL [211]	-	41.0	31.1	-	140.9	25.2	manual
AutoCaption [216] †	81.5	40.2	29.9	59.5	135.8	23.8	RL
IUC-D (ours) †	81.8	40.9	31.0	59.5	140.6	25.3	gradient-based
IUC-E (ours) †	**82.3**	**42.1**	**31.4**	**60.1**	**141.9**	**25.8**	gradient-based

3.4 Experiments

In this section, we apply our proposed IUC methods to perform image encoder and language decoder architecture searches. Each experiment consists of two steps: architecture search and architecture evaluation. The optimal cell is obtained in the search process, and it will

be evaluated in the evaluation stage based on the formed large network from the optimal cell. The large network will be retrained from scratch for the architecture evaluation.

3.4.1 Datasets

We perform experiments on the COCO captions dataset [118] to evaluate and compare our proposed methods. The COCO captions dataset contains $82,783$ and $40,504$ images in the training and validation sets, respectively. We conduct thorough experiments by analyzing our models on the offline and the online evaluations. Each image in the dataset holds five captions, which were annotated by humans. During the offline evaluation, we utilize the 'Karpathy' splits setting [102], which has $113,287$ images and 5000 images in the training set and test set, respectively. At the second stage of the architecture search, we use the 123K unlabeled images of the COCO dataset, which has a similar class distribution as the labeled images, and we use our trained encoder-decoder model from the first stage to generate a textual description of the images. Then the pre-trained predictive IC model will be trained based on the generated pseudo-dataset. Finally, we update the architecture of the encoder-decoder by minimizing the validation loss of the predictive model on the validation set.

3.4.2 Experimental Settings

Recall that our proposed framework IUC is a comprehensive differentiable method, which can be applied with any differentiable architecture search approach. With that being said, we employ DARTS-2nd [122] in the conducted experiments that are exhibited in Table 3.2, 3.4, and 3.5, and DARTS [122], PC-DARTS [194], and DATA [20] in Table 3.6 of the Ablation 3.4.4, which is shown in Table 3.6. We introduce two variants of our IUC method: 1) IUC with image encoder architecture search (IUC-E); and 2) IUC with language decoder architecture search (IUC-D). Moreover, we simultaneously apply IUC architecture searching on both the image encoder and the language decoder in Ablation 3.4.4. Note that the common operations in differentiable architecture searches usually do not contain some practical operations such as:

Table 3.3. Comparison of our methods and the state-of-the-art image captioning models on the COCO "Karpathy" test split with multiple models (Ensemble/Fusion). Methods with † are using NAS methods.

	Cross-Entropy Loss						**Encoder-Decoder Architecture**
	BLEU-1	BLEU-4	METEOR	ROUGE	CIDEr	SPICE	
SCST$^{\Sigma}$ [152]	-	32.8	26.7	55.1	106.5	-	manual
RFNet$^{\Sigma}$ [98]	77.4	37.0	27.9	57.3	116.3	20.8	manual
GCN-LSTM$^{\Sigma}$ [201]	77.4	37.1	28.1	57.2	117.1	21.1	manual
SGAE$^{\Sigma}$ [198]	-	-	-	-	-	-	manual
AoANet$^{\Sigma}$ [90]	78.7	38.1	28.5	58.2	122.7	21.7	manual
X-LAN$^{\Sigma}$ [139]	78.8	39.1	29.1	58.5	124.5	22.2	manual
X-Transformer$^{\Sigma}$[139]	77.8	37.7	29.0	58.0	122.1	21.9	manual
AutoCaption$^{\Sigma}$ [216] †	79.8	40.3	29.6	59.2	128.5	22.8	RL
IUC (ours) †	**80.0**	**40.6**	**31.2**	**59.4**	**132.8**	**23.9**	gradient-based
	CIDEr Score Optimization						**Encoder-Decoder Architecture**
	BLEU-1	BLEU-4	METEOR	ROUGE	CIDEr	SPICE	
SCST$^{\Sigma}$ [152]	-	35.4	27.1	56.6	117.5	-	manual
RFNet$^{\Sigma}$ [98]	80.4	37.9	28.3	58.3	125.7	21.7	manual
GCN-LSTM$^{\Sigma}$ [201]	80.9	38.3	28.6	58.5	128.7	22.1	manual
SGAE [198]	81.0	39.0	28.4	58.9	129.1	22.2	manual
AoANet$^{\Sigma}$ [90]	81.6	40.2	29.3	59.4	132.0	22.8	manual
X-LAN$^{\Sigma}$ [139]	81.6	40.3	29.8	59.6	133.7	23.6	manual
X-Transformer$^{\Sigma}$[139]	81.7	40.7	29.9	59.7	135.3	23.8	manual
AutoCaption$^{\Sigma}$ [216] †	82.9	42.1	30.4	60.4	139.5	24.3	RL
IUC (ours) †	**82.4**	**42.5**	**31.8**	**60.4**	**142.7**	**25.9**	gradient-based

attention, top-down, or RoI pooling. To address this issue, we modify our image encoder and language decoder inspired by some SoTA methods such as Faster R-CNN [150] and X-LAN [139] techniques.

Architecture Search Details.

During the search stage, we use Faster R-CNN [150] and X-LAN [139] as our image encoder and sentence decoder of the image captioning model, respectively. In IUC-E, we switch their human-designed architecture network with 8 convolutional cells (each holding 7 nodes), while during the IUC-D search we replace their LSTMs with our recurrent cell, which consists of 12 nodes. In IUC-D, ResNeXt-152 is adopted as the CNN in the Faster R-CNN. In addition, the initial convolutional architectures of the image encoder in IUC-E are pre-trained on ImageNet [47], and Visual Genome [105] datasets prior to the search for improving the object feature extractions. Following the settings in DARTS [122], we use SGD for the IUC-E architecture searching with a batch size of 128, an initial learning rate of 0.025, weight decay of 3e-4, and a momentum of 0.9 for 50 epochs. More detailed hyperparameter settings are exhibited in the

Appendix. Due to high-performance achievements of *OSCAR* [115], we use pre-trained $OSCAR_L$ [115] as our predictive model to help us obtain the optimal encoder-decoder architecture. At the second stage of architecture search, the trained encoder-decoder from the first stage generates a textual description of the input image. Then, the predictive model learns from the generated pseudo dataset and validates its performance on the validation set to update the architecture of our encoder-decoder. In this chapter, we constructed our experiments with similar settings as $OSCAR_L$ [115] for a fair comparison.

Architecture Evaluation Details.

For a fair comparison, we adopt X-LAN [139] method as our language decoder and Faster R-CNN [150] as the image decoder. In IUC-E, we replace the convolution neural network architecture of Faster R-CNN [150] with our obtained image encoder architecture, designed by stacking 14 searched optimal convolution cells. Later, we pre-train the model on ImageNet [47] and Visual Genome [105] to extract the object tags and image region features. Similarly, in IUC-D, we use Faster R-CNN [150] with ResNeXt-152 pre-trained on ImageNet and Visual Genome dataset datasets, and we change the LSTMs of the X-LAN with our optimal architecture that was obtained during the architecture search. Then, we train our constructed large network with cross-entropy loss for 100 epochs and 10000 warmup steps. Next, we choose the model that achieves the highest CIDEr score, and we CIDEr score optimization with the learning rate of 0.00001 for another 100 epochs. During the validation, we utilize the beam search with the beam size of 3 and our models are trained with Adam optimizer [104].

Metrics.

We adopt the official evaluation metrics - including BLEU-N[140], METEOR [10], ROUGE[117], CIDEr [178], and SPICE[5] - to analyze and compare our proposed methods' performances with the other existing approaches in image captioning tasks.

3.4.3 Results

We evaluate the image captioning results of our proposed methods on the COCO "Karpathy" test split [102] to compare with the recent proposals in this area, which have achieved noteworthy performances. Our primary baselines include: LSTM [180] and LSTM-A[202], which are non-attention based; SCST[152] that proposes employing attention over the grid of features; RFNet[98] merges CNN features by adopting recurrent fusion networks; Up-Down[6] uses attention over regions; GCN-LSTM[201] uses visual relations between image regions; SGAE[198] utilizes auto-encoding scene graphs for sentence generation; AoANet[90] applies attention on attention and LSTM as the image encoder and language decoder, respectively; Meshed-Memory Transformer[39] constructs mesh-like transformer connectivity between the encoder and the decoder; X-LAN[139] plugs unified attention blocks, called X-Linear attention blocks, into the encoder-decoder architecture, and further uses such blocks in the Transformer-based encoder-decoder architecture, which is called X-Transformer[139]; OSCAR[115] adopts object tags as anchor points to enhance the learning of the image-text semantic alignments; OSCAR+ with VINVL[211] shows that visual features are crucial in image understanding tasks by improving object detection model of OSCAR+; X-Transformer+PPO[209] applies proximal policy optimization to X-Transformer; and AutoCaption [216] that applies neural architecture search on the language decoder with a similar structure as X-LAN.

Table 3.2 reports the performance comparisons of our proposed methods (IUC-D and IUC-E) and the state-of-the-art models on the offline COCO "Karpathy" test split for both cross-entropy loss and CIDEr score optimization. It is shown that our method IUC-E outperforms the baselines and elevates the state-of-the-art in most of the metrics, while IUC-D frequently exhibits the second-best performances among the other methods. IUC-D performs slightly better than IUC-E in SPICE. Our proposed IUC-E model can achieve 141.9 on CIDEr using CIDEr score optimization, which indicates an improvement of 1 CIDEr point compared to OSCAR+ with VINVL, and 6.1 points improvement in comparison to AutoCaption. This

performance enhancement verifies the critical advantage of employing architecture search to design the encoder and the decoder architectures. Additionally, the better performances of IUC-E compared to IUC-D in the evaluation results demonstrate the urgency of the architecture design of the image encoder, which has not been investigated relatively. Finally, we ensemble our IUC-E and IUC-D models, called IUC, for further improvement. In Table 3.3, we evaluate and compare the performance of IUC and the existing works by utilizing ensemble models. Our extensive experiments show that IUC can outperform the existing image captioning models in single and ensemble model settings with both Cross-Entropy loss optimization and CIDEr Score optimization.

Figure 3.3 showcases different exemplar of generated captions by our IUC and our baseline (AutoCaption$^{\Sigma}$ [216]) along with the human-annotated ground truth (GT) captions. As it is shown, our model generates more accurate, clear, and detailed textual descriptions in challenging image cases, since our encoder-decoder architecture in IUC is task-specific designed, while AutoCaption [216] aims only to design the language decoder using NAS. This implies that the architecture design of the image encoder has more impact on the model's performance than the language decoder's architecture design.

3.4.4 Ablation Studies

Ablation 1.

We are interested in verifying the critical advantage of employing architecture search in image captioning tasks. In this study, we employ architecture search and random sampling for the architecture designs of the image encoder and the language decoder.

Table 3.4. Comparison of searched (S) and randomly sampled (R) encoder and decoder architectures on COCO "Karpathy" test split (single-model with Cross-Entropy Loss).

Encoder	Decoder	BLEU-4	METEOR	CIDEr	SPICE
R	R	37.7	28.0	117.9	21.7
R	S	37.9	28.7	122.1	21.9
S	R	**38.3**	**29.4**	**127.3**	**22.6**

AutoCaption: Some food is on a white plate.
IUC: A white plate of fish and broccoli is sitting on a table
GT: The meal of fish has a side of broccoli.

AutoCaption: A street sign next to a tree.
IUC: A red do not enter sign with two green street signs above it.
GT: A red do not enter sign under a green street sign.

AutoCaption: A young man is standing next to a bed.
IUC: A man with a leopard robe is standing next to a white bed.
GT: A man dressed in leopard robe next to a bed.

AutoCaption: A white refrigerator in the patio.
IUC: A dirty refrigerator and some garbage on the floor next to a building.
GT: An abandoned refrigerator next to a building with a window.

Figure 3.3. Exemplar captions generated by IUC and AutoCaption as well as their corresponding ground truth sentences generated by humans.

Similar to Section 3.4.2; first, we perform an architecture search or random sampling to obtain the optimal cells, then we construct the large network by using the optimal cells, and finally, we train and evaluate the large network. To get a deeper understanding of the impact that each module's architecture (i.e., image encoder and language decoder) has on the image captioning performance, we do not use the OSCAR pre-training during this study - unlike the experiments in Table 3.2 - to reduce the constraints and dependencies of our investigation. Similar to DARTS [122], image encoder and language decoder architectures are searched or randomly sampled from convolutional or recurrent cells, respectively. Table 3.4 shows that architecture search can significantly enhance the image captioning models' performance, and the architecture design of the image encoder is more crucial for achieving higher performance than the architecture design of the language decoder.

Ablation 2.

In this setting, we investigate how the IUC-E model's performance varies as the tradeoff parameters λ and γ change in Eq.(3.10).

$$\min_{A} \ \lambda L(W^*(E^*(A),F^*(A)),D^{(val)}) + \gamma L(E^*(A),F^*(A),D^{(val)})$$

$$s.t. \ W^*(E^*(A),F^*(A)) = \underset{W}{\operatorname{argmin}} \ L(W,U,E^*(A),F^*(A)) \tag{3.10}$$

$$E^*(A),F^*(A) = \underset{E,F}{\operatorname{argmin}} \ L(E,A,F,D^{(tr)})$$

Table 3.5. Image captioning evaluation with different tradeoff parameters (λ and γ) on COCO "Karpathy" test split (single-model with Cross-Entropy Loss).

Lambda λ	Gamma γ	BLEU-4	METEOR	CIDEr	SPICE
1	0	**39.7**	**30.9**	**131.1**	**23.8**
0	1	39.1	29.7	129.9	22.9
1	1	39.5	30.4	130.6	23.1

Table 3.5 demonstrates the performance of IUC-E in three different cases: **1)** $\lambda = 1$ and $\gamma = 0$, the encoder-decoder model updates its architecture by minimizing the validation loss of the predictive model only, without considering the validation loss of itself - similar to Eq.(3.4) - and this model achieves the best performance on all four metrics. **2)** $\lambda = 0$ and $\gamma = 1$. Unlike the first case, we have a bi-level optimization problem since the encoder-decoder model updates its architecture by minimizing its validation loss without going through the second stage. This model exhibits the lowest performance. **3)** When $\lambda = 1$ and $\gamma = 1$, we are combining the validation loss of the predictive model and encoder-decoder model. In this scenario, we can achieve higher performance than in the second case since the predictive model provides more useful feedback, which assists in better learning. The achievement of these three cases implies the significant impact of $L(W^*(E^*(A),F^*(A)),D^{(val)})$ in the validation loss, which helps the model to improve its understanding of the images.

Ablation 3.

In spite of the high achievements of DARTS [122], some of the newer variations of differentiable NAS methods were able to enhance the performance of DARTS and reduce its memory cost by utilizing different techniques on DARTS.

Table 3.6. Comparison of utilizing various differentiable architecture search based methods with IUC on the COCO "Karpathy" test split (single-model with Cross-Entropy Loss).

Methods	BLEU-4	METEOR	CIDEr	SPICE
IUC-E + DARTS	39.7	30.9	131.1	23.8
IUC-E + PC-DARTS	39.8	31.2	131.6	**24.1**
IUC-E + DATA	**40.1**	**31.3**	**131.9**	23.9

To study some of these methods for additional enhancements, we evaluate our proposed models in image captioning by applying various DARTS-based methods, including DARTS, PC-DARTS, and DATA to search for the image encoder architecture of the IUC-E model. Evaluation results in Table 3.6 show that utilizing more advanced DARTS-based methods, such as DATA, can be applied for further improvements on top of IUC.

3.5 Conclusion

In mission-critical applications (e.g., disease diagnosis), if the textual descriptions generated by image captioning models are incorrect, they may mislead human decision-makers and have potential negative social impacts. Thus, it is crucial to minimize the prediction error in such tasks. To tackle this issue, we presented a novel approach for image captioning tasks by utilizing differentiable NAS techniques to obtain the high-performance encoder-decoder model architectures. We introduced a three-level optimization problem by formulating IUC and provided efficient solutions to this specific framework. Furthermore, our investigations show the effectiveness of the encoder and decoder modules individually in image understanding. We applied our proposed methods on COCO image captions dataset to verify IUC can outperform the existing state-of-the-art methods in the image captioning tasks.

3.6 Acknowledgements

Chapter 3, in part, has been published in the 2022 ACM Multimedia (ACM MM) Conference under the title "Image Understanding by Captioning with Differentiable Architecture

Search" by Ramtin Hosseini, and Pengtao Xie. The dissertation author was the primary author of this material.

Chapter 4

Interpretable NAS via Saliency Learning

4.1 Introduction

Neural architecture search (NAS) [218, 122, 148], which aims to automatically identify highly performant neural architectures, has received much attention recently. Existing NAS methods treat all elements in an input data example (such as pixels in an image, tokens in a sentence, etc.) as being equally important, without considering the different saliency of individual elements, which leads to less-optimal performance. In machine learning applications, an input data example typically consists of many data elements. For instance, an image example consists of a grid of pixel elements and a sentence example consists of a sequence of token elements. When used to make a prediction, different data elements have different importance (or saliency). For example, when predicting which object category an image belongs to, pixels in foreground object regions are more important than those in background regions. Such saliency information is not leveraged by existing NAS methods.

In this chapter, we aim to bridge this gap, by proposing a saliency-aware NAS method which automatically identifies the saliency of data elements and leverages that to search for better architectures. Our framework is formulated as a four-level optimization problem. At the first level, the model tentatively fixes its architecture and trains its first set of network weights. At the second level, the trained model generates saliency maps using an adversarial attack-based method [56]. At the third level, input data is reweighted using saliency maps and the second

set of model weights are trained using reweighted data. At the fourth level, the two sets of trained model weights are evaluated on a human-provided validation set and the architecture is optimized by minimizing the validation losses.

4.2 Related works

Saliency detection.

Many methods [205, 217, 166, 136, 156, 154, 145] have been proposed for saliency detection, based on perturbing inputs [205, 217], propagating gradients [9, 164, 166], attention [113, 200, 136], model approximation [153, 4], etc. Several works [156, 154, 145] show that leveraging saliency of input data can enhance model's predictive power. Rieger et al. [154] leverage domain-specific rules or knowledge to provide "groundtruth" saliency. Such rules/knowledge are difficult to obtain in many applications. Pillai and Pirsiavash [145] encourage a prediction model to produce saliency maps that are consistent with those generated by GradCAM [159]. GradCAM is an unsupervised approach; without any supervision from humans, its generated saliency maps may not be reliable. For example, it is shown in [167] that GradCAM cannot highlight adversarial image patches that cause wrong predictions. In [156], saliency maps are either labeled by humans or auto-generated based on gradient magnitude with no human supervision, which suffers the same problems as [159, 154]. Different from existing works, our method generates saliency maps with weak supervision such as human-provided class labels. Such weak supervision is much easier to obtain than human annotations of saliency maps and can yield more reliable saliency maps than using no supervision at all.

Bi-level optimization.

Many ML methods [57, 11, 59, 122, 163, 212] have been formulated as bi-level optimization (BLO) problems. In these methods, network weights are learned by solving an inner optimization problem defined on a training set while meta parameters are learned by solving an outer optimization problem defined on a validation set. BLO-based methods have been applied

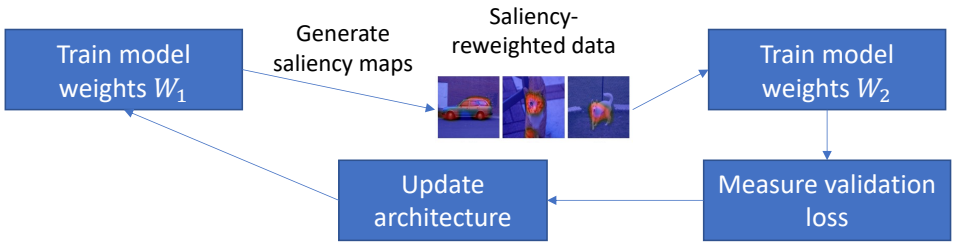


Figure 4.1. Overview of our framework.

for neural architecture search [122], hyperparameter tuning [57], learning rate adaptation [11], data selection [163, 151, 184], meta learning [59], label correction [212], etc., where meta parameters are neural architectures, hyperparameters, importance weights of data examples, meta network weights, etc. Many optimization algorithms [40, 62, 67, 97, 123, 197] have been developed for solving BLO problems where convergence analysis is provided.

4.3 Methods

In this section, we propose a four-level optimization based framework to perform saliency-aware neural architecture search. For the ease of presentation, we assume the task is image classification. In the experiments, we show that our method can be applied for other tasks as well.

4.3.1 A four-level optimization framework

In our framework (Figure 4.1), a model has a learnable architecture A and two sets of learnable network weights W_1 and W_2. W_2 is a tensor that has the same dimensions as W_1. The weight values in W_2 and W_1 are different. It consists of four stages performed end-to-end. At the first stage, the model trains its network weights W_1 with the architecture A tentatively fixed. At the second stage, the trained W_1 generates saliency maps for input images: a saliency score is calculated for each pixel. At the third stage, images are reweighted using saliency maps and saliency-reweighted images are used to train model weights W_2. At the fourth stage, the trained W_2 is evaluated on a human-labeled validation set and the architecture A is updated by

minimizing the validation loss.

Stage I.

At the first stage, the model trains its first set of network weights W_1 by minimizing a loss L on training dataset $D^{(\text{tr})}$, with the architecture A tentatively fixed:

$$W_1^*(A) = \text{argmin}_{W_1} L(W_1, A, D^{(\text{tr})}). \tag{4.1}$$

To define the training loss, we need to use both the architecture parameters A and network weights W_1. However, A cannot be updated by minimizing the training loss. Otherwise, a trivial solution of A will be yielded: A can perfectly overfit the training data but will make incorrect predictions on unseen data examples. $W_1^*(A)$ denotes that the optimal weights W_1^* depends on A. This is because $L(W_1, A, D^{(\text{tr})})$ is a function of A, and W_1^* depends on $L(W_1, A, D^{(\text{tr})})$.

Stage II.

At the second stage, the trained $W_1^*(A)$ is used to generate saliency maps. Specifically, given an input image x, we first use $W_1^*(A)$ and A to predict the class label (denoted by $f(x; W_1^*(A), A)$) of x. Then an adversarial attack based approach [65, 56] is leveraged to calculate saliency scores of pixels. Adversarial attack adds small random perturbations δ to pixels in x so that the prediction outcome on the perturbed image $x+\delta$ is no longer $f(x; W_1^*(A), A)$. Pixels perturbed more are more correlated with the prediction outcome $f(x; W_1^*(A), A)$ and are considered to be more salient. This process amounts to solving the following optimization problem:

$$\{\delta_i^*(W_1^*(A), A)\}_{i=1}^N = \text{argmax}_{\{\|\delta_i\|_\infty \leq \varepsilon\}_{i=1}^N} \sum_{i=1}^N \ell(f(x_i+\delta_i; W_1^*(A), A), f(x_i; W_1^*(A), A)) \tag{4.2}$$

where δ_i is the perturbation added to image x_i and ε is a small norm-bound. N is the number of images. $f(x_i+\delta_i; W_1^*(A), A)$ and $f(x_i; W_1^*(A), A)$ are predictions made by $W_1^*(A)$ and A on $x_i+\delta_i$ and x_i. Assume the number of classes is K. $f(x_i+\delta_i; W_1^*(A), A)$ and $f(x_i; W_1^*(A), A)$ are

K-dimensional vectors containing prediction probabilities on individual classes. $\ell(\cdot,\cdot)$ is the cross-entropy loss with $\ell(\mathbf{a},\mathbf{b}) = -\sum_{k=1}^{K} b_i \log a_i$. In this optimization problem, we aim to find perturbations for each image so that the predicted outcome on the perturbed image is largely different from that on the original image. The learned optimal perturbations are considered as saliency scores of pixels: larger perturbations indicate that the corresponding pixels are more correlated with the prediction outcome and therefore are more salient. δ_i^* depends on $W_1^*(A)$ and A since δ_i^* depends on the loss in Eq.(4.2), and the loss is a function of $W_1^*(A)$ and A.

Stage III.

At the third stage, given the saliency scores $\{\delta_i^*(W_1^*(A),A)\}_{i=1}^N$, the second set of model weights W_2 are trained. We use the saliency scores to reweight the pixels: $x \odot \delta$, where $\odot$ denotes element-wise multiplication (we compare with other reweighting mechanisms in Table 4.4). Pixels that are more salient are given more weights. Then W_2 is trained on these weighted pixels:

$$W_2^*(\{\delta_i^*(W_1^*(A),A)\}_{i=1}^N,A) = \text{argmin}_{W_2} \ \sum_{i=1}^N \ell(f(\delta_i^*(W_1^*(A),A) \odot x_i; W_2, A), t_i), \tag{4.3}$$

where $f(\delta_i^*(W_1^*(A),A) \odot x_i; W_2, A)$ is the prediction made by W_2 and A on the weighted image $\delta_i^*(W_1^*(A),A) \odot x_i$, and t_i is the class label. W^* depends on $\{\delta_i^*(W_1^*(A),A\}_{i=1}^N$ and A since W^* depends on the loss in Eq.(4.3), and the loss is a function of $\{\delta_i^*(W_1^*(A),A)\}_{i=1}^N$ and A.

Stage IV.

At the fourth stage, $W_2^*(\{\delta_i^*(W_1^*(A),A)\}_{i=1}^N,A)$ and $W_1^*(A)$ are evaluated on a human-labeled validation set $D^{(\text{val})}$. The architecture A is updated by minimizing the validation losses:

$$\min_A L(W_2^*(\{\delta_i^*(W_1^*(A),A)\}_{i=1}^N,A),A,D^{(\text{val})}) + \gamma L(W_1^*(A),A,D^{(\text{val})}), \tag{4.4}$$

where γ is a tradeoff parameter.

Four-level optimization framework.

We integrate the four stages into a unified four-level optimization framework and obtain the following formulation:

$$
\begin{aligned}
&\min_A L(W_2^*(\{\delta_i^*(W_1^*(A),A)\}_{i=1}^N,A),A,D^{(\text{val})}) + \gamma L(W_1^*(A),A,D^{(\text{val})}) \\
&s.t.\ \ W_2^*(\{\delta_i^*(W_1^*(A),A)\}_{i=1}^N,A) = \text{argmin}_{W_2} \sum_{i=1}^N \ell(f(\delta_i^*(W_1^*(A),A)\odot x_i;W_2,A),t_i) \\
&\quad \{\delta_i^*(W_1^*(A),A)\}_{i=1}^N = \text{argmax}_{\{\|\delta_i\|_\infty\leq\varepsilon\}_{i=1}^N} \sum_{i=1}^N \ell(f(x_i+\delta_i;W_1^*(A),A),f(x_i;W_1^*(A),A)) \\
&\quad W_1^*(A) = \text{argmin}_{W_1} L(W_1,A,D^{(\text{tr})}).
\end{aligned}
\tag{4.5}
$$

In this framework, there are four optimization problems, each corresponding to a learning stage. From bottom to up, the optimization problems correspond to learning stage I to IV respectively. The first three optimization problems are nested on the constraint of the fourth optimization problem. These four stages are conducted end-to-end in this unified framework. The solution $W_1^*(A)$ obtained at the first stage is used to generate explanations at the second stage. The saliency maps $\{\delta_i^*(W_1^*(A),A)\}_{i=1}^N$ obtained at the second stage are used to train W_2 at the third stage. The solutions obtained at the first and third stage are used to calculate validation losses at the fourth stage. The architecture A updated at the fourth stage changes the training loss at the first stage and consequently changes the solution $W_1^*(A)$, which subsequently changes $\{\delta_i^*(W_1^*(A),A)\}_{i=1}^N$ and $W_2^*(\{\delta_i^*(W_1^*(A),A)\}_{i=1}^N,A)$.

Optimization algorithm.

To solve the problem in Eq.(4.5), we used a standard algorithm developed in [122], which is broadly used in many previous works and demonstrated to be effective in the literature. The convergence analysis of this algorithm has been given in many works [62, 67, 97, 123, 197]. The optimization algorithm is not the focus or contribution of our work. In this algorithm, we calculate the

gradient of $L(W_1, A, D^{(\text{tr})})$ w.r.t W_1 and approximate $W_1^*(A)$ using a one-step gradient descent update of W_1. We plug the approximation W_1' of $W_1^*(A)$ into $\ell(f(x_i+\delta_i; W_1^*(A), A), f(x_i; W_1^*(A), A))$ and obtain an approximated objective denoted by O_{δ_i}. Then we approximate $\delta_i^*(W_1^*(A), A)$ using a one-step gradient ascent update of δ_i based on the gradient of O_{δ_i}. Next, we plug the approximation δ_i' of $\delta_i^*(W_1^*(A), A)$ into $\ell(f(\delta_i^*(W_1^*(A), A) \odot x_i; W_2, A), t_i)$ and get another approximated objective denoted by O_{W_2}. Then we approximate $W_2^*(\{\delta_i^*(W_1^*(A), A)\}_{i=1}^N, A)$ using a one-step gradient descent update of W_2 based on the gradient of O_{W_2}. Finally, we plug the approximation W_1' of $W_1^*(A)$ and the approximation W_2' of $W_2^*(\{\delta_i^*(W_1^*(A), A)\}_{i=1}^N, A)$ into the validation loss and get the third approximate objective O_A. A is updated by descending the gradient of O_A. These steps iterate until convergence.

4.4 Experiments

In this section, we present experimental results.

4.4.1 Experiments on image classification

Following [122] the architecture A is parameterized in a differentiable way. A contains a set of importance weights, each multiplied to the output of a candidate architecture block. Architecture search amounts to learning these weights using gradient methods. After learning, blocks with top weights compose an architecture. Each experiment consists of two phrases: 1) architecture search where an optimal cell is identified, and 2) architecture evaluation where multiple copies of the optimal cell are stacked into a larger network, which is retrained from scratch.

Datasets

We used three datasets: CIFAR-10 [107], CIFAR-100 [108], and ImageNet [47]. Both CIFAR-10 and CIFAR-100 contain 60K images from 10 classes. For each of them, we split it into a train, validation, and test set with 25K, 25K, and 10K images respectively. ImageNet contains

1.2M training images and 50K test images from 1000 objective classes. Following [196], we randomly sample 10% of the 1.2M images to form a new training set and another 2.5% to form a validation set, then perform a search on them.

Experimental settings

We experimented with the search spaces in DARTS [122], P-DARTS [29], and PC-DARTS [196]. The tradeoff parameter γ is set to 2. The norm bound ϵ of perturbations is set to 0.03. During architecture search, for CIFAR-10 and CIFAR-100, the architecture of the target is a stack of 8 cells. Each cell consists of 7 nodes. Initial channel number is 16. The search algorithm was based on SGD, with a batch size of 64, an initial learning rate of 0.025 with cosine scheduling, an epoch number of 50, a weight decay of 3e-4, and a momentum of 0.9. We update the architecture A every 5 mini-batches (iterations), update model weights W_2 and perturbations δ every 3 mini-batches, and update W_1 on every mini-batch. The rest of hyperparameters mostly follow those in DARTS, P-DARTS, and PC-DARTS. We compare with the following baselines: 1) explanation constraints (EC) [156]; 2) contextual decomposition explanation penalization (CDEP) [154]; 3) global max pooling + GradCAM (GMPGC) [145]. The mean and standard deviation of 10 random runs are reported.

Results and analysis on CIFAR-100 and CIFAR-10

Table 4.1 shows results on CIFAR-100 and CIFAR-10. Applying our framework to DARTS, P-DARTS, and PC-DARTS, test errors are greatly reduced, which shows that by end-to-end detecting and leveraging saliency of pixels, the quality of searched architectures can be improved. Another observation from Table 4.1 is: the performance gain of our method does not come at the cost of substantially increasing model size (number of parameters) or search cost.

Table 4.1 shows that our methods outperform EC, CDEP, and GMPGC. These methods use GradCAM, gradient magnitude, and pretrained semantic segmentation models to calculate saliency scores, which are not very reliable. In contrast, the calculation of saliency in our method is weakly supervised by the validation loss of the second model calculated on human-provided

Table 4.1. Test errors on CIFAR-100 (C100) and CIFAR-10 (C10), number of model parameters (in millions), and search cost (GPU days on a Nvidia 1080Ti).

Method	Error-C100	Error-C10	Param.	Cost
ResNet [78]	22.10	6.43	1.7	-
DenseNet [88]	17.18	3.46	25.6	-
PNAS [120]	19.53	3.41±0.09	3.2	150
ENAS [144]	19.43	2.89	4.6	0.5
AmoebaNet [148]	18.93	2.55±0.05	3.1	3150
GDAS [52]	18.38	2.93	3.4	0.2
R-DARTS [207]	18.01±0.26	2.95±0.21	-	1.6
DARTS$^-$ [35]	17.51±0.25	2.59±0.08	3.3	0.4
AutoFormer [23]	17.42±0.17	2.72±0.09	3.7	2.8
Sampling [72]	17.30±0.10	2.75±0.11	3.8	2.0
DropNAS [81]	16.95±0.41	2.58±0.14	4.4	0.7
DrNAS [28]	-	2.54±0.03	4.0	0.4
ISTA-NAS [199]	-	2.54±0.05	3.3	0.1
MiLeNAS [76]	-	2.51±0.11	3.9	0.3
GAEA [114]	-	2.50±0.06	-	0.1
PDARTS-ADV [27]	-	2.48±0.02	3.4	1.1
Darts2nd [122]	20.58±0.44	2.76±0.09	3.1	4.0
EC-darts2nd [156]	20.05±0.31	2.83±0.12	3.3	5.7
CDEP-darts2nd [154]	19.53±0.46	2.75±0.05	3.2	5.3
GMPGC-darts2nd [145]	19.08±0.36	2.81±0.07	3.2	5.6
Ours-darts2nd	**16.42**±0.09	**2.54**±0.05	3.2	4.6
Pcdarts [196]	17.96±0.15	2.57±0.07	3.9	0.1
EC-pcdarts [156]	17.83±0.28	2.63±0.11	4.1	0.9
CDEP-pcdarts [154]	17.88±0.13	2.75±0.08	4.0	1.1
GMPGC-pcdarts [145]	17.73±0.09	2.64±0.05	4.0	1.0
Ours-pcdarts	**16.19**±0.04	**2.49**±0.03	3.9	0.8
Prdarts [213]	16.48±0.06	2.37±0.03	3.4	0.2
EC-prdarts [156]	17.32±0.14	2.58±0.08	3.5	1.1
CDEP-prdarts [154]	16.86±0.07	2.54±0.05	3.4	1.3
GMPGC-prdarts [145]	16.37±0.10	2.46±0.06	3.6	1.1
Ours-prdarts	**16.01**±0.03	**2.30**±0.04	3.6	0.8
Pdarts [29]	17.52±0.06	2.54±0.04	3.6	0.3
EC-pdarts [156]	17.25±0.11	2.68±0.07	3.8	1.3
CDEP-pdarts [154]	17.49±0.08	2.63±0.10	3.6	0.9
GMPGC-pdarts [145]	17.33±0.10	2.59±0.07	3.7	1.1
Ours-pdarts	**15.16**±0.09	**2.45**±0.03	3.6	1.1

Table 4.2. Top-1 and top-5 test errors on ImageNet in the mobile setting.

Method	Top-1	Top-5
Inception-v1 [172]	30.2	10.1
MobileNet [86]	29.4	10.5
ShuffleNet 2× (v2) [130]	25.1	7.6
NASNet-A [219]	26.0	8.4
PNAS [120]	25.8	8.1
MnasNet-92 [174]	25.2	8.0
AmoebaNet-C [148]	24.3	7.6
PARSEC-CIFAR10 [18]	26.0	8.4
GDAS-CIFAR10 [52]	26.0	8.5
DSNAS-ImageNet [87]	25.7	8.1
AutoFormer [23]	25.3	7.4
Sampling [72]	25.3	-
SDARTS-ADV-CIFAR10 [27]	25.2	7.8
PC-DARTS-CIFAR10 [196]	25.1	7.8
ProxylessNAS-ImageNet [16]	24.9	7.5
FairDARTS-ImageNet [36]	24.4	7.4
PR-DARTS [213]	24.1	7.3
DARTS$^-$-ImageNet [35]	23.8	7.0
Darts2nd-cifar10 [122]	26.7	8.7
EC-darts2nd-cifar10 [156]	26.4	8.5
CDEP-darts2nd-cifar10 [154]	26.5	8.5
GMPGC-darts2nd-cifar10 [145]	26.3	8.2
SANAS-darts2nd-cifar10 (ours)	**24.8**	**8.3**
Pdarts-cifar100 [29]	24.7	7.5
EC-pdarts-cifar100 [156]	24.5	7.3
CDEP-pdarts-cifar100 [154]	24.6	7.4
GMPGC-pdarts-cifar100 [145]	24.5	7.4
SANAS-pdarts-cifar100 (ours)	**23.8**	**6.6**
Pcdarts-ImageNet [196]	24.2	7.3
EC-pcdarts-ImageNet [156]	24.0	7.2
CDEP-pcdarts-ImageNet [154]	23.9	7.3
GMPGC-pcdarts-ImageNet [145]	24.0	7.3
SANAS-pcdarts-ImageNet (ours)	**22.2**	**6.1**

class labels, which have higher fidelity. As analyzed earlier, saliency with higher fidelity can result in higher-quality architectures.

Results on ImageNet

In Table 4.2, we compare different methods on ImageNet. In experiments based on PC-DARTS, architectures are searched on a subset of ImageNet. In other experiments, architectures are searched on CIFAR-10 and CIFAR-100. Ours-darts2nd-cifar10 denotes that our method is applied to DARTS-2nd and performs search on CIFAR10. Similar meanings hold for other notations in such a format. The observations made from these results are consistent with those made from Table 4.1. The architectures searched using our methods are consistently better than those searched by corresponding baselines. These results again show that by end-to-end detecting and leveraging saliency can improve architecture search.

Sanity check of saliency maps

We evaluate saliency maps generated by the adversarial saliency method using model parameter cascading randomization tests [2]. The model architecture is searched by SANAS on ImageNet. Figure 4.2(left) shows that saliency maps change considerably as more layers are randomized, on multiple ImageNet examples. Figure 4.2(right) shows the Spearman rank correlation (with absolute values) between original saliency maps and randomized saliency maps, on ImageNet. The rank correlation consistently decreases as more layers are randomized. These results demonstrate that saliency maps generated by the adversarial saliency method are sensitive to model parameters and pass the sanity check.

Human evaluation of saliency

We randomly sampled 100 images from the test set of ImageNet and generated saliency maps for them using different methods. Then we asked three undergraduates to judge whether the saliency maps are sensible. The ratings are from 1-5 (higher is better). Table 4.3 summarizes the mean and standard deviation of ratings. Our methods achieve significantly higher ratings (significance test results are in the supplements), which demonstrates that our methods can

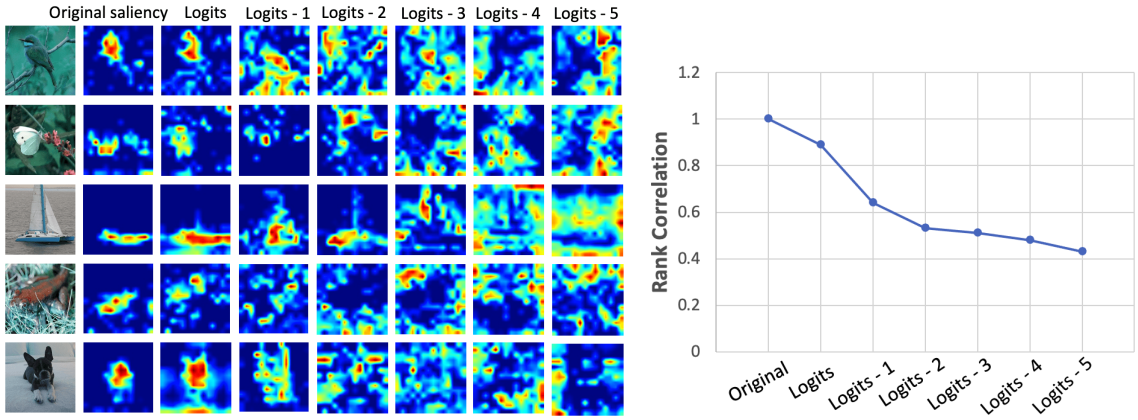


Figure 4.2. Sanity check of saliency maps. Logits$-n$ is the n-th layer below the logits layer.

Table 4.3. Human evaluation of saliency.

	Ratings
Darts2nd	3.30±0.24
EC-darts2nd [156]	3.42±0.16
CDEP-darts2nd [154]	3.58±0.11
GMPGC-darts2nd [145]	3.51±0.12
SANAS-darts2nd (ours)	**3.93**±0.06
Pdarts	3.26±0.14
EC-pdarts [156]	3.59±0.21
CDEP-pdarts [154]	3.46±0.19
GMPGC-pdarts [145]	3.50±0.09
SANAS-pdarts (ours)	**4.07**±0.11

generate more accurate saliency maps than the baselines. Inter-annotator Kappa score is 0.71, which shows strong agreements among annotators.

Visualization of saliency

In Figure 4.3, we visualize the saliency maps generated for some randomly-sampled ImageNet images. These saliency maps are very sensible. Warmer color (representing higher saliency) regions correspond to objects. Colder color regions correspond to background. These results show that our method is effective in generating correct saliency maps. In contrast, the saliency maps generated by baselines are less sensible. For example, in the schipperke, hay,

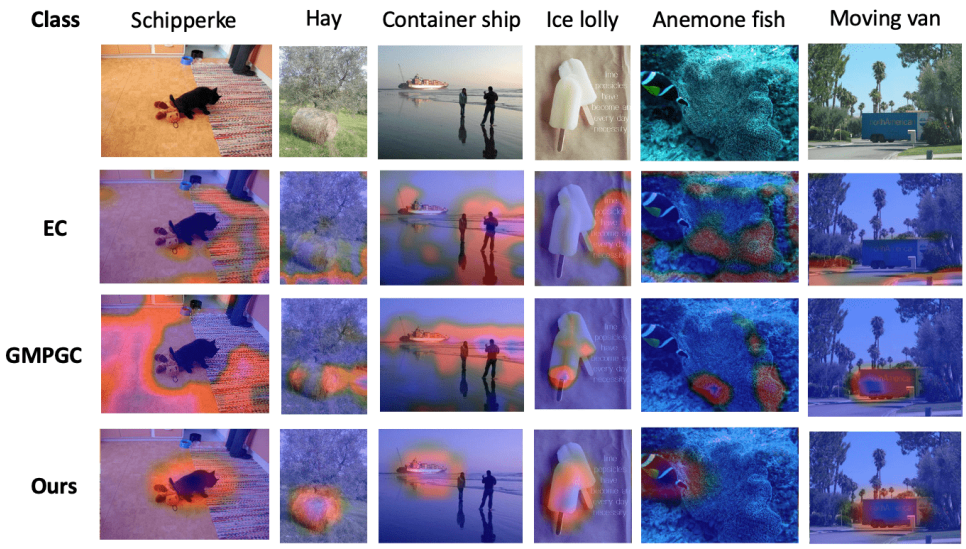


Figure 4.3. Visualization of saliency maps.

container ship, ice lolly images, higher saliency regions of baselines are in the background instead of on objects.

Ablation studies

In terms of how to use saliency scores to reweight pixels, we compare the element-wise product between pixels and saliency scores in Eq.(4.3) with 1) element-wise addition between pixels and saliency scores; 2) element-wise product between pixels and the absolute values of saliency scores; 3) concatenate saliency map with input image and feed the concatenation into the second model W_2; 4) no reweighting. Table 4.4 shows results where our method is applied to DARTS2nd and P-DARTS. From this table, we make the following observations. **First**, product works better than addition. The reason is: magnitude of saliency scores (perturbations) is very small; adding them to pixels does not render a significant change of pixel values, and consequently cannot distinguish important pixels from unimportant ones. In contrast, the relative difference between saliency scores is significantly large; multiplying them to pixels can better distinguish important and unimportant pixels. **Second**, reweighting pixels using signed saliency score and absolute saliency scores does not have a significant difference. This shows that signs of saliency scores do not matter too much. **Third**, product works better than concatenation. The

Table 4.4. Test errors of different reweighting mechanisms.

	CIFAR-100		CIFAR-10	
	Ours+darts	Ours+pdarts	Ours+darts	Ours+pdarts
Product	**16.4**±0.09	**15.2**±0.09	**2.54**±0.05	**2.45**±0.03
Addition	20.3±0.27	17.7±0.03	2.71±0.11	2.53±0.06
Absolute	16.8±0.11	15.4±0.12	2.55±0.08	2.48±0.07
Concatenate	17.9±0.11	16.4±0.06	2.74±0.07	2.61±0.08
No reweight	20.6±0.44	17.5±0.06	2.76±0.09	2.54±0.04

possible reason is: compared with concatenation, product can better differentiate important and unimportant pixels using the saliency scores. **Fourth**, reweighting pixels works better than no reweighting. This demonstrates that multiplying saliency scores to inputs is indeed helpful in identifying important pixels, which helps to improve classification performance.

In the next ablation study, we compare the adversarial attack (AA) based saliency detection method with another two saliency detection methods, including integrated gradients (IG) [171] and SmoothGrad (SG) [166] by plugging them into our framework. These studies are performed on Darts2nd and Pdarts. Table 4.5 shows the results, where we make two observations. First, our framework with IG and SG still outperforms vanilla Darts2nd and Pdarts. This demonstrates that our framework is a general one that generalizes beyond a single saliency detection method. Second, IG and SG perform worse than AA. A possible reason is: IG and SG restrict the definition of saliency to be gradient-based. In contrast, AA treats saliency scores as optimization variables and automatically learns them by solving an optimization problem, which is more flexible.

To better understand the effectiveness of the proposed four stages performed end-to-end, we compare with the following two ablation settings, which are performed on Darts-2nd and Pdarts.

- Perform the four stages separately (denoted as **Separate**) instead of end-to-end.
- Perform stages I, II, III by optimizing the weighted sum of their objective functions with

Table 4.5. Ablation results on saliency detection methods.

	CIFAR-100	CIFAR-10
Darts [122]	20.58±0.44	2.76±0.09
IG+darts	16.92±0.08	2.62±0.06
SG+darts	17.05±0.11	2.59±0.03
AA+darts	**16.42**±0.09	**2.54**±0.05
Pdarts [29]	17.52±0.06	2.54±0.04
IG+pdarts	15.83±0.08	2.47±0.03
SG+pdarts	15.81±0.05	2.48±0.04
AA+pdarts	**15.16**±0.09	**2.45**±0.03

weights 1, 0.5, 1, in a multi-task learning (MTL) manner (denoted as **MTL**).

Table 4.7 shows the results on Separate and MTL. We make two observations. First, our end-to-end method works better than Separate which conducts the four stages separately. Conducting the four stages end-to-end can enable them to mutually influence each other to achieve the overall best performance. In contrast, when conducted separately, earlier stages cannot be influenced by later stages (e.g., stage I cannot be influenced by stage IV), which leads to worse performance. Second, our method performs better than MTL. The tasks in stages I-III have an inherent order: before detecting saliency maps using a model, we first need to train this model; before training the second model on saliency-reweighted data, we need to detect the saliency maps first. MTL performs these three tasks simultaneously by minimizing a single objective, which breaks their inherent order and therefore leads to worse performance. In contrast, our method preserves this order using multi-level optimization.

Figure 4.4(right) shows how the test error of SANAS-darts2nd on CIFAR100 varies with γ. When $\gamma = 0$, the validation loss of W_1 is not used for architecture search and the performance is inferior (compared with $\gamma = 2$). A γ in the middle ground which properly balances the two validation losses yields the optimal performance. Using the validation loss of W_1 only is equivalent to vanilla DARTS-2nd (results are in the supplements).

Table 4.6. Results on the GLUE benchmark.

Method	# Param.	Infer	SST-2	MRPC	QQP	MNLI	QNLI	RTE	Average
$BERT_{12}$	109M	1x	93.5	88.9	71.2	84.6	90.5	66.4	82.5
$BERT_{12}$-T	109M	1x	93.3	88.7	71.1	84.8	90.4	66.1	82.4
$BERT_6$-PKD	67.0M	1.9x	92.0	85.0	70.7	81.5	89.0	65.5	80.6
$BERT_3$-PKD	45.7M	3.7x	87.5	80.7	68.1	76.7	84.7	58.2	76.0
$DistilBERT_4$	52.2M	3.0x	91.4	82.4	68.5	78.9	85.2	54.1	76.8
$TinyBert_4$	14.5M	9.4x	92.6	86.4	71.3	82.5	87.7	62.9	80.6
$BiLSTM_{SOFT}$	10.1M	7.6x	90.7	-	68.2	73.0	-	-	-
AdaBERT	8.3M	16.1x	91.9	85.3	70.2	81.9	86.9	64.8	80.2
EC-AdaBERT	8.7M	15.8x	91.9	85.6	70.7	81.8	86.9	65.0	80.3
CDEP-AdaBERT	8.8M	16.4x	92.5	85.9	70.6	82.2	87.4	65.2	80.6
GMPGC-AdaBERT	8.1M	15.5x	92.0	85.6	71.4	82.8	87.1	65.0	80.7
Ours-AdaBERT	8.2M	16.3x	93.4	87.0	71.8	83.7	88.5	66.6	81.8

4.4.2 Experiments on text classification

In this section, we apply the proposed framework for text classification. The Gumbel softmax trick [95] is leveraged to deal with non differentiability of texts.

Datasets

We conduct experiments on six datasets in the GLUE benchmark [182]: SST-2, MRPC, QQP, MNLI, QNLI and RTE. SST-2 is a sentiment classification dataset where the input text is movie review and the output label is whether the review is positive or negative. In MRPC and QQP, the input is a pair of sentences and the output is whether they are semantically equivalent. MNLI, QNLI, and RTE are textual entailment recognition datasets.

Baselines

We compare with the following baselines: 1) BERT [49], 2) BERT-PKD [170], 3) Distil-BERT [158], 4) TinyBERT [99], 5) $BiLSTM_{SOFT}$ [175], 6) AdaBERT [21], 7) EC-AdaBERT [156], 8) CDEP-AdaBERT [154], and 9) GMPGC-AdaBERT [145].

Table 4.7. Ablation results on Separate and MTL.

	CIFAR-100	CIFAR-10
Separate+darts	18.05±0.27	2.68±0.06
MTL+darts	18.26±0.12	2.70±0.05
Ours+darts	**16.42**±0.09	**2.54**±0.05
Separate+pdarts	16.49±0.07	2.51±0.03
MTL+pdarts	16.83±0.10	2.52±0.04
Ours+pdarts	**15.16**±0.09	**2.45**±0.03

Hyperparameter settings

Candidate operations are commonly used operations in convolutional networks, including 1D convolution, dilated convolution, pooling, identify, and zero. In dilated convolution, the kernel size includes 3, 5, and 7. Each convolution operation consists of an Relu-Conv-BatchNorm sequence. For pooling, we used average pooling and max pooling, where the kernel size is set to 3. The "SAME" padding is utilized for convolution and pooling. We optimize weight parameters using SGD. The initial learning rate is set to $2e-2$. It is annealed using a cosine scheduler. The momentum is set to 0.9. We use Adam [103] to optimize the architecture variables. The learning rate is set to $3e-4$ and weight decay is set to $1e-3$.

Main results

Table 4.6 shows the results. We make the following observations. First, our method works better than AdaBERT, which is an NAS method without saliency detection. This further demonstrates the effectiveness of saliency detection in improving NAS. Second, our method works better than EC, CDEP, and GMPGC. This further shows that performing saliency detection and NAS jointly is better than conducting them separately as the three baselines do. Third, compared with BERT_{12} and BERT_{12}-T, our method achieves similar performance while using much fewer parameters and being much faster during inference.

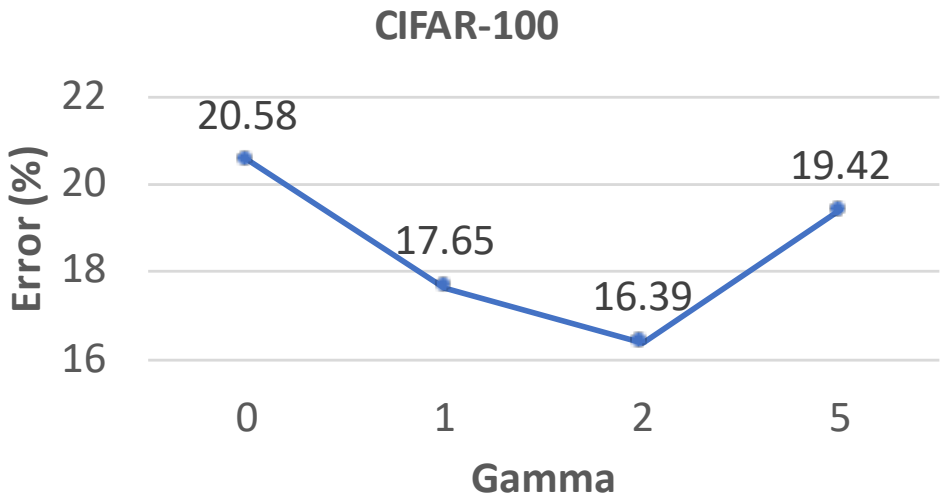


Figure 4.4. How errors change with γ.

Qualitative results

Table 4.8 shows salient words detected by different methods on a randomly sampled sentence (whose sentiment is labeled as being positive). As can be seen, our method can successfully recognize the words "entertaining" and "please" which are mostly relevant to a positive sentiment. In contrast, the two baselines fail to do that.

4.5 Conclusions and discussions

In this chapter, we propose to leverage the saliency information of input data to improve NAS. Our work makes the following contributions. First, our method can detect saliency and perform NAS end-to-end, based on a four-level optimization framework. The framework performs four stages in a unified way: train a preliminary model, generate saliency maps using the preliminary model, retrain the model on saliency-reweighted data, and update architecture by minimizing validation losses. Second, our framework is end-to-end differentiable, allowing using efficient gradient-based algorithms as solvers. Third, our method provides a mechanism to evaluate generated saliency maps by checking whether they are helpful for improving classification performance. We demonstrate the effectiveness of our method on several datasets.

Table 4.8. Top-2 salient words (marked with red color) detected by different methods.

EC	an entertaining mix of period drama and flat-out farce that should please history fans.
GMPGC	an entertaining mix of period drama and flat-out farce that should please history fans.
Ours	an entertaining mix of period drama and flat-out farce that should please history fans.
EC	a very witty take on change, risk and romance, and the film uses humor to make its points about acceptance and growth.
GMPGC	a very witty take on change, risk and romance, and the film uses humor to make its points about acceptance and growth.
Ours	a very witty take on change, risk and romance, and the film uses humor to make its points about acceptance and growth.

One major limitation of this work is that it cannot be easily applied to non-differentiable NAS methods that are based on reinforcement learning (RL) and evolutionary algorithm (EA). The reason is that our method uses a gradient-based optimization algorithm to solve the multi-level optimization problem. For non-differentiable NAS methods, their non-differentiable objective functions do not have gradients, which therefore are not compatible with the gradient-based algorithm used by our method. Please see Appendix 4.7.1 for discussion on how to extend our method to non-differentiable NAS methods. Another limitation of our method is its higher time cost than baselines, due to the extra computation needed for detecting saliency maps. Considering the benefits and limitations of our method, we recommend using our method in applications that strongly need high-performance architectures capable of generating sensible saliency maps but do not have strong efficiency requirements on architecture search time. For applications which have high restrictions on search cost but allow sacrificing some performance and ignoring saliency maps, other NAS methods might be better choices. Please see Appendix 4.7.1 for a more detailed discussion.

One potential negative societal impact of our work is: in mission-critical applications such as disease diagnosis and autonomous driving, if saliency maps generated by our method are not correct, they may mislead human decision-makers. For future works, we plan to investigate these ideas: 1) formulate saliency-based network pruning [214] as a saliency-aware NAS problem and automatically search for the optimal pruning decisions based on detected saliency; 2) extend the notion of saliency from input data to blocks in neural networks, develop multi-level optimization based frameworks to detect the saliency of blocks, and perform pruning on blocks based on detected saliency.

4.6 Healthcare Applications – Brain Tumors Classification

4.6.1 Introduction

Brain tumor is a life-threatening disease and causes about 0.25 million deaths worldwide in 2020. Deep learning methods have been developed to classify brain tumors based on magnetic resonance imaging (MRI), to assist physicians in accurately diagnosing and determining the subtypes of brain tumors. Existing methods rely heavily on human experts to design deep neural networks, which is time-consuming, labor-intensive, and sometimes sub-optimal. To address this problem, we apply SANAS for MRI-based brain tumor classification, including four classes: glioma, meningioma, pituitary tumor, and healthy. The dataset contains 3264 MRI images. Results show that our method can search for neural architectures that achieve better classification accuracy than manually-designed deep neural networks while having fewer model parameters. For example, our method achieves a test accuracy of 90.6% with 3.75M parameters while the accuracy of a human-designed network – ResNet101 – is 84.5% with 42.56M parameters.

Brain tumor, where abnormal brain cells grow in an uncontrollable way, is a life-threatening disease that causes about 0.25 million deaths worldwide in 2020[1]. The 5-year survival rate for people with brain tumors is about 36% and the 10-year survival rate is about 31%[1]. Brain tumors vary from non-cancerous benign variants to much more harmful malignant ones[14]. The World Health Organisation (WHO) has assigned grades[128] (I through IV) to tumors based on their severity and other molecular characteristics. Higher-grade tumors are more malignant, with patients having smaller survival rates[14]. Immediate diagnosis and treatment are crucial for improving the survival rate[14]. Among various tests for identifying the existence and types of brain tumors, Magnetic Resonance Imaging (MRI) is frequently used due to its noninvasive nature, being less harmful to human bodies, the ability to capture high-resolution images, and the timeliness in getting results[131]. Detecting brain tumors and determining their types from MRI is a highly challenging medical task, which requires many years of training and medical practice[131]. In medically less developed regions such as rural areas, physicians

who can accurately interpret MRI images to diagnose and assess the severity of brain tumors are highly lacking[131].

To address this problem, artificial intelligence methods (especially deep learning methods) [131, 3, 74, 63, 51] have been developed to provide physicians with decision support for brain tumor classification. In these methods, deep neural networks are manually designed by human experts, which is time-consuming and labor-intensive. To address this problem, we propose a method that can automatically search for neural architectures to perform accurate classification of brain tumors and are computationally efficient. Neural architecture search (NAS) [218, 144, 219, 122, 168] has been broadly studied previously. The performance of these methods is not stable and architectures searched by these methods oftentimes perform less well than human-designed architectures

Our proposed framework is applied for classifying brain tumors from MRI images. The dataset used in our experiments contains 3264 MRI images from four classes: glioma, meningioma, pituitary tumor, and healthy. Our method achieves better classification accuracy with fewer model parameters compared with manually-designed neural networks and previous neural architecture search methods.

4.6.2 Related Works

A variety of deep learning methods[92] have been proposed for brain tumor classification and segmentation. Menze et al.[134] developed a multi-modal brain tumor image segmentation benchmark, where 20 tumor segmentation algorithms were evaluated on 65 multi-contrast MRI images that have low-grade and high-grade glioma. Pereira et al.[143] utilized convolutional neural networks for brain tumor segmentation in MRI images. Havaei et al.[75] proposed a convolutional neural network for brain tumor classification, which exploits both local features and global contextual features. Afshar et al.[3] utilized capsule networks to perform brain tumor classification. Chen et al.[25] proposed a dual-force convolutional neural network for brain tumor segmentation, which leverages multi-level information and a dual-force training

mechanism to improve latent representations. Sajjad et al.[157] utilized deep CNN with data augmentation for multi-grade brain tumor classification. Kaldera et al.[100] utilized faster region-based convolutional neural networks for brain tumor classification and segmentation. Ghosal et al.[64] utilized a squeeze and excitation ResNet model for brain tumor classification. Mzoughi et al.[137] proposed a multi-scale three-dimensional convolutional neural network for glioma brain tumor classification based on the whole volumetric T1-Gado MRI sequence. Pei et al.[142] proposed a 3D context aware deep learning method for brain tumor segmentation, subtype classification, and survival prediction. Ghassemi et al.[63] pretrained a deep neural network as the discriminator of a generative adversarial network (GAN) for extracting robust features, which is utilized for classifying brain tumors. Shaik et al.[160] proposed a multi-level attention mechanism for brain tumor recognition, where spatial and cross-channel attention is utilized to identify tumor regions and maintain cross-channel temporal dependencies. Hao et al. [74] proposed a transfer learning based active learning method for brain tumor classification. This method aims to reduce human annotation cost and stabilize model performance. Lu et al.[129] proposed data distillation and augmentation methods for brain tumor detection. This method distills representative examples which are mixed to create augmented examples. Deepak et al.[45] leveraged siamese network and neighborhood analysis for brain tumor classification. Díaz-Pernas et al.[51] utilized a multiscale convolutional neural network for brain tumor classification and segmentation.

4.6.3 Datasets

The data used for this work is from a public dataset [13] on Kaggle. There are 3264 MRI images in total, which are from four classes: Glioma, Meningioma, Pituitary, and Healthy. Glioma is the most frequent type of malignant brain tumor[44], which typically occurs in the glial cells of the brain and spinal cord. Meningioma is a type of benign brain tumor; however, it can develop into malignant tumors without proper intervention. Meningioma is typically located in meninges, which are protective membranes enclosing the brain. Like meningioma, pituitary

tumors are benign and formed in the pituitary gland below the brain. Both meningioma and pituitary tumors are difficult to diagnose as they show very few symptoms. The correctness of class labels are verified by medical practitioners. The size of input images is 64×64. The dataset is split into a training set with 2870 images and a test set with 394 images. Table 4.9 shows data split statistics. Image augmentation is performed using AutoAugment [43].

Table 4.9. Number of training and test images per class

Tumor Type	Number of Training Images	Number of Test Images
Glioma	826	100
Meningioma	822	115
Healthy	395	105
Pituitary	827	74

4.6.4 Experimental Settings

Our framework is orthogonal to existing NAS approaches and can be applied to any differentiable NAS method. In the experiments, SANAS was applied to DARTS [122], P-DARTS [29], and PC-DARTS [196]. The search spaces of these methods are composed of (dilated) separable convolutions with sizes of 3×3 and 5×5, max pooling with size of 3×3, average pooling with size of 3×3, identity, and zero. Following Liu et al.[122], each experiment consists of two phrases: 1) architecture search where an optimal cell is identified, and 2) architecture evaluation where multiple copies of the optimal cell are stacked into a larger network, which is retrained from scratch. During architecture search, the architecture of the explainer is a stack of 8 cells. Each cell consists of 7 nodes. We set the initial channel number to 16. For the audience model, we set it to ResNet-18 [79]. We set the tradeoff parameter γ to 1. We randomly split the training set into two parts. During architecture search in SANAS, the first part is used as $D_e^{(\mathrm{tr})}$ and $D_a^{(\mathrm{tr})}$, and the second part is used as $D_e^{(\mathrm{val})}$ and $D_a^{(\mathrm{val})}$. During architecture evaluation, the composed large network is trained on the entire training set. The search algorithm was based on SGD, with a batch size of 64, an initial learning rate of 0.025 (reduced in later epochs using a cosine decay scheduler), an epoch number of 50, a weight decay of 3e-4, and

a momentum of 0.9. The rest of hyperparameters mostly follow those in DARTS, P-DARTS, and PC-DARTS. During architecture evaluation, a larger network of the explainer is formed by stacking 12 copies of the searched cell. The initial channel number was set to 36. We trained the network with a batch size of 96, an epoch number of 3000, on a single Tesla v100 GPU. We compared with manually-designed neural architectures including ResNet[77], VGGNet[165], and DenseNet[89].

4.6.5 Results and Discussion

Table 4.10. Test accuracy and number of model parameters of different methods.

Method	**Accuracy (%)**	**Parameters (M)**
DenseNet40	83.50	0.25
DenseNet101	86.80	0.95
VGGNet13	88.07	10.72
VGGNet16	88.33	16.03
ResNet50	85.79	23.54
ResNet101	84.52	42.56
DARTS	89.34	3.85
SANAS+DARTS	**90.61**	3.75
PCDARTS	88.07	3.57
SANAS+PCDARTS ($\gamma = 0.1$)	89.60	4.27
SANAS+PCDARTS ($\gamma = 0.5$)	89.09	4.05
SANAS+PCDARTS ($\gamma = 1$)	88.83	4.25
PDARTS	88.33	3.85
SANAS+PDARTS	88.83	3.77

Table 4.10 shows test accuracy and number of model parameters of different methods. From this table, we make the following observations. **First**, our SANAS+DARTS method achieves the highest test accuracy among all methods. Its accuracy is much higher than ResNet and VGGNet, while its parameter size is much smaller than ResNet and VGGNet. For instance, our method achieves a test accuracy of 90.6% with 3.75M parameters while the accuracy of a human-designed network – ResNet101 – is 84.5% with 42.56M parameters. **Second**, applying our proposed SANAS method to different NAS baselines including DARTS, PCDARTS, and PDARTS improves the performance of these baselines. For example, by applying SANAS, the

Table 4.11. Results of the ablation study where the explainer updates its architecture by minimizing the validation loss of the audience only.

Method	Error (%)
Audience only (SANAS+DARTS)	90.18
Audience + explainer (SANAS+DARTS)	**90.61**
Audience only (SANAS+PDARTS)	88.49
Audience + explainer (SANAS+PDARTS)	**88.83**

test accuracy of DARTS is improved from 89.34% to 90.61%. These results strongly demonstrate the broad effectiveness of our framework in searching for better neural architectures. The reason behind this is: in our framework, the explanations made by the explainer are used to train the audience model; the validation performance of the audience reflects how good the explanations are; to make good explanations, the explainer's model has to be trained well; driven by the goal of helping the audience to learn well, the explainer continuously improves the training of itself. Such an explanation-driven learning mechanism is lacking in baseline methods, which are hence inferior to our method.

To better understand our proposed method, we perform an ablation study where the explainer updates its architecture by minimizing the validation loss of the audience only, without considering the validation loss of itself. Table 4.11 shows the results of SANAS+DARTS and SANAS+PDARTS. As can be seen, "audience + explainer" where the validation losses of both the audience model and explainer itself are minimized to update the explainer's architecture works better than "audience only" where only the audience's validation loss is used to learn the architecture. Audience's validation loss reflects how good the explanations made by the explainer are. Explainer's validation loss reflects how strong the explainer's prediction ability is. Combining these two losses provides more useful feedback to the explainer than using one loss only, which hence can help the explainer to learn better.

We also performed an ablation study on how the choice of audience models affects test accuracy. We experimented with two architectures for the audience model: ResNet18 and VGGNet13, where ResNet18 is more expressive than VGGNet13 since it has more layers.

Table 4.12. Results on how different choices of audience models affect test accuracy.

Method	Accuracy (%)
SANAS+DARTS (VGGNet13)	90.17
SANAS+DARTS (ResNet18)	**90.61**
SANAS+PDARTS (VGGNet13)	88.56
SANAS+PDARTS (ResNet18)	**88.83**

Table 4.12 shows the results. As can be seen, in SANAS applied to DARTS and PDARTS, using ResNet18 as the audience achieves better performance than using VGGNet13. The reason is that to help a stronger audience to learn better, the explainer has to be even stronger. For a stronger audience model, it already has great capability in achieving excellent classification performance. To further improve this audience, the explanations used to train this audience need to be very sensible and informative. To generate such explanations, the explainer has to force itself to learn very well.

We investigated how test accuracy changes with the tradeoff parameter γ. The third panel in Table 4.10 shows the results of SANAS+PCDARTS. As can be seen, the test accuracy increases when we increase γ from 0 (which is equivalent to vanilla PCDARTS) to 0.1. The reason is that a larger γ enables the audience to provide stronger feedback to the explainer regarding how good the explanations are. Such feedback can guide the explainer to refine its architecture for generating better explanations. However, if γ is further increased, the error becomes worse. Under such circumstances, too much emphasis is put on evaluating how good the explanations are and less attention is paid to the predictive ability of the explainer. The architecture is biased to generating good explanations with predictive performance compromised, which leads to inferior performance.

We perform visualization of the results. Figure 4.5 shows the convergence curves of validation accuracy for different NAS methods with and without SANAS, convergence curves of validation accuracy for SANAS+PCDARTS under different γ values, and convergence curves of validation accuracy for non-NAS methods. Figure 4.6 show the architectures searched by SANAS+DARTS, SANAS+PCDARTS, and SANAS+PDARTS.

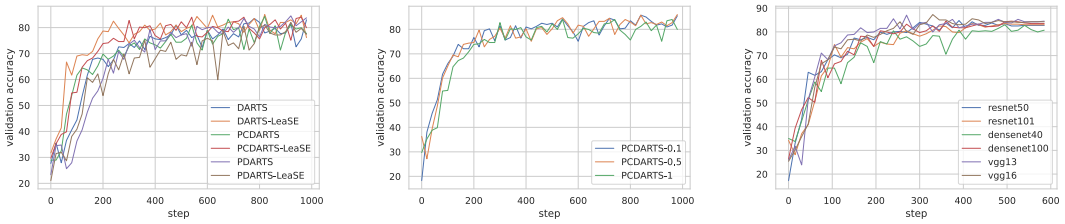
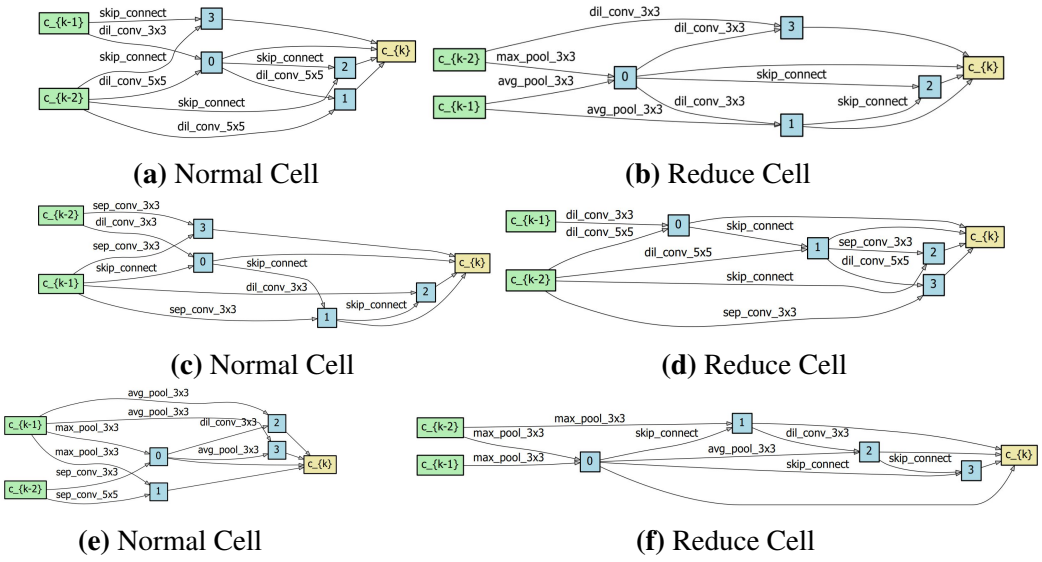


Figure 4.5. Left: Convergence curves of validation accuracy for different NAS methods with and without SANAS; **Middle**: Convergence curves of validation accuracy for SANAS+PCDARTS under different γ values; **Right**: Convergence curves of validation accuracy for non-NAS methods.



(a) Normal Cell **(b)** Reduce Cell

(c) Normal Cell **(d)** Reduce Cell

(e) Normal Cell **(f)** Reduce Cell

Figure 4.6. (a-b): the architecture searched by SANAS+DARTS; **(c-d)**: the architecture searched by SANAS+PCDARTS; **(e-f)**: The architecture searched by SANAS+PDARTS.

4.7 Appendix

4.7.1 Limitations

Apply SANAS to non-differentiable NAS methods.

To apply SANAS to non-differentiable NAS methods, we have to change the current gradient-based optimization algorithm to some other non-gradient-based optimization algorithms (such as REINFORCE [189] for reinforcement learning), which might incur higher computational costs. To apply SANAS to reinforcement learning (RL) based NAS methods, we perform the following procedures. First, we use an RL controller [218] to generate a set of candidate

architectures. Second, given a candidate architecture, we train its weight parameters on a training dataset, similar to stage I in SANAS. Third, given the trained model, we perform adversarial attacks to detect the saliency maps of the training data, similar to stage II in SANAS. Fourth, we use saliency maps to reweight training data and retrain the model on reweighted data, similar to stage III in SANAS. Fifth, we evaluate the retrained model on a validation set and use validation accuracy as a reward for this architecture. We repeat steps 2-5 for every candidate architecture, calculate the mean reward on all candidate architectures, and update the RL controller by maximizing the mean reward using policy gradient [218]. These procedures repeat until convergence. Similar procedures can be conducted to perform saliency-aware architecture search in evolutionary algorithm based NAS methods.

When to use SANAS and when not.

It is recommended to use SANAS in applications that strongly need high-performance neural architectures capable of generating sensible saliency maps but do not have strong requirements on the time spent on architecture search. For example, imaging-based disease diagnosis is a good application scenario of SANAS, for two reasons. First, disease diagnosis needs to be highly accurate and needs high-fidelity saliency maps for interpreting predicted diagnosis outcomes. Second, to use an automatically searched neural architecture in hospitals, FDA approval is needed, which usually takes several months. To successfully pass the FDA approval, it is acceptable to take some extra time to search for a high-quality neural architecture. For applications which have high restrictions on search cost but allow sacrificing some performance and ignoring saliency maps, other NAS methods that have higher search efficiency but lower performance and weaker saliency-generation capability than our method might be better choices. Examples of such applications are online learning applications which need to update architectures in real time.

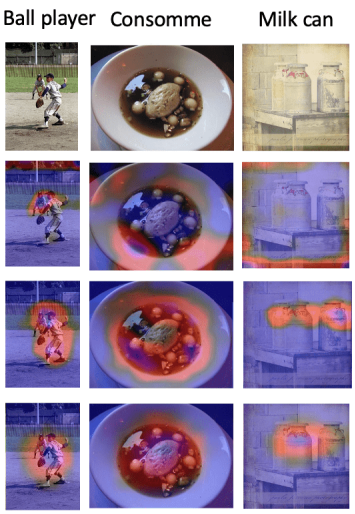


Figure 4.7. More examples of visual saliency maps.

4.7.2 Visualization of saliency maps

Figure 4.7 shows more examples of visual saliency maps. The saliency maps detected by our method are more sensible than those detected by baselines.

4.7.3 Salient word detection

Table 4.13 shows more examples of salient words detected by different methods. In each example, the top-2 words detected by our method are more salient than baselines. The prediction task corresponding to Table 4.8 and 4.13 is sentiment classification. A word is more salient if it has a stronger correlation with a sentiment (either positive or negative). For example, the word "entertaining" in Table 4.8 implies a positive sentiment, and therefore is considered to be salient. In contrast, the word "mix" is a neutral word that is irrelevant to sentiments, and therefore is not considered to be salient.

4.7.4 Improving computational efficiency

We improved the computational efficiency of our method from both the algorithm side and implementation side. On the algorithm side, we speed up computation by approximating the optimal solution at each stage using a one-step gradient descent update [122] and reducing the

Table 4.13. Top-2 salient words (marked with red color) detected by different methods.

EC	with youthful high spirits, tautou remains captivating throughout michele's religious and romantic quests, and she is backed by a likable cast.
GMPGC	with youthful high spirits, tautou remains captivating throughout michele's religious and romantic quests, and she is backed by a likable cast.
Ours	with youthful high spirits, tautou remains captivating throughout michele's religious and romantic quests, and she is backed by a likable cast.
EC	the title's lameness should clue you in on how bad the movie is.
GMPGC	the title's lameness should clue you in on how bad the movie is.
Ours	the title's lameness should clue you in on how bad the movie is.

frequencies of these updates. Specifically, we update the architecture A every 5 mini-batches. In contrast, baselines (including Darts, Parts, Pcdarts, Prdarts) update A on every mini-batch. We update model weights W_2 and perturbations δ every 3 mini-batches, and update W_1 on every mini-batch. We empirically found that reducing the update frequencies of certain parameters can significantly speed up convergence without sacrificing performance. Besides, when calculating hypergradients of A, we recursively approximate matrix-vector multiplications using finite-difference calculations [122], which reduces the computation cost from being quadratic in matrix dimensions down to linear.

On the implementation side, we speed up computation by leveraging techniques and tricks including 1) automatic mixed precision [135], 2) using multiple (4, specifically) workers and pinned memory in PyTorch DataLoader, 3) using cudNN autotuner, 4) kernel fusion, etc.

4.7.5 Hyperparameter tuning strategies

Most hyperparameters in our method follow their default values used in baseline methods. The only hyperparameter needing to be tuned is the tradeoff parameter γ. To tune γ on CIFAR-

100, we randomly sample 2.5K data from the 25K training set and sample 2.5K data from the 25K validation set. Then we use the 5K sampled data as a hyperparameter tuning set. γ is tuned in 0.1, 0.5, 1, 2, 3. For each configuration of γ, we use the remaining 22.5K training data and 22.5K validation data to perform architecture search and use their combination to perform architecture evaluation (retraining a larger stacked network from scratch). Then we measure the performance of the stacked network on the 5K sampled data. γ value yielding the best performance on the 5K sampled data is selected. For γ in CIFAR-10 and ImageNet experiments, we simply used the value tuned on CIFAR-100 and did not conduct further tuning.

4.8 Acknowledgements

Chapter 4, in part, has been published in the Proceedings of the 2022 Neural Information Processing Systems (NeurIPS) Conference under the title ”Saliency-Aware Neural Architecture Search” by Ramtin Hosseini and Pengtao Xie. Additionally, a portion of this chapter has been published in the 2022 Nature Scientific Reports as ”Brain Tumor Classification Based on Neural Architecture Search” by Ramtin Hosseini, Shubham Chitnis, and Pengtao Xie. The dissertation author served as the primary author for these publications.

Conclusion and Future Works

In conclusion, neural architecture search (NAS) has emerged as a powerful tool for automating the design of deep neural networks. However, Trustworthy Neural Architecture Search is essential for any successful deep learning system and in order to be truly useful in real-world applications, NAS must be made trustworthy in terms of interpretability, fairness, robustness, and out-of-domain generalization. Where interpretability is a crucial aspect of NAS, as it allows researchers and practitioners to understand the reasoning behind the selected architecture. This is particularly important in applications where the consequences of a mistake are severe, such as medical imaging or self-driving cars. One approach to addressing interpretability in NAS is to use architectures that are composed of modules with simple, interpretable functions. Additionally, techniques such as visualization and explainability methods can be used to gain insights into the internal workings of NAS models. Fairness is another important consideration in NAS. This is because NAS models are often used in applications where the consequences of bias can be severe, such as in criminal justice or hiring. To address fairness in NAS, one approach is to use architectures that are designed to be robust to bias in the training data. Additionally, techniques such as adversarial training and debiasing methods can be used to mitigate bias in NAS models. Robustness is another key consideration in NAS. This is because NAS models are often used in applications where the operating conditions may be uncertain or adversarial, such as in autonomous vehicles or cybersecurity. To address robustness in NAS, one approach is to use architectures that are designed to be robust to various types of perturbations, such as noise and adversarial examples. Additionally, techniques such as adversarial training and robust optimization can be used to improve the robustness of NAS models. Out-of-domain

generalization is the final key consideration in NAS. This is because NAS models are often used in applications where the test data may be different from the training data, such as in natural language processing or computer vision. To address out-of-domain generalization in NAS, one approach is to use architectures that are designed to be robust to changes in the input distribution. Additionally, techniques such as domain adaptation and meta-learning can be used to improve the out-of-domain generalization of NAS models. With the right approaches, Neural Architecture Search can be a powerful tool for creating robust and reliable deep learning systems that are reliable and trustworthy. It has shown that through the use of interpretability, fairness, robustness, and out-of-domain generalization, neural architecture search can be used to identify optimal architectures for specific tasks, while maintaining trustworthiness. The results of this research have shown that these four trustworthiness criteria are beneficial in neural architecture search, and that they can be used to improve the performance of a network. In particular, interpretability can be used to better understand the network, fairness can be used to ensure that the network does not discriminate against certain groups, robustness can be used to ensure that the network is robust to changes in the data, and out-of-domain generalization can be used to ensure that the network is generalizable to new data.

In this thesis, we addressed these four trustworthiness criteria of robustness,, fairness, out-of-domain generalization, and interpretability. In **chapter 1**, we introduced techniques for searching for robust neural architectures in a differentiable manner. Our methods define two differentiable metrics for measuring the robustness of architectures, which are based on certified lower bounds and Jacobian norm bounds. The goal of our search is to find architectures that maximize these robustness metrics. Unlike previous methods that aim to enhance the robustness of architectures through implicit means such as adversarial training or adding random noise, our techniques explicitly and directly optimize for robustness metrics to achieve more robust architectures. In **chapter 2**, we proposed a machine learning framework inspired by the human ability to learn by categorization to address the fairness issue of NAS techniques, which divides a wide range of problems into distinct categories and solves each one using

a category-specific sub-model. This framework is formulated as a three-level optimization problem, which consists of three stages of learning that are carried out in a joint manner with gradient descent. We have developed an efficient optimization algorithm to address this problem and applied it to differentiable neural architecture search methods. Experiments on GLUE, CIFAR-10, CIFAR-100, and ImageNet datasets demonstrate the effectiveness of our method, which can reduce overfitting and achieve state-of-the-art results with both fixed and searchable network architectures. In **chapter 3**, we proposed a three-level optimization framework that leverages self-training technique to tackle the generalization issue in existing NAS techniques. Finally, in **chapter 4**, we proposed an interpratible framework that uses four-level optimization to dynamically detect the salience of input data, reweight data based on salience maps, and search for architectures on the reweighted data. The first stage involves training a model with a fixed architecture. The second level generates salience maps using the trained model. The third stage retrains the model on data reweighted by the salience maps. The fourth and final stage evaluates the model on a validation set and updates the architecture to minimize the validation loss. Experiments on several datasets have proven the efficacy of our framework. Furthermore, we applied this proposed method in healthcare applications (i.e., brain tumor classification).

Future Directions in Neural Architecture Search

Enhancing the trustworthiness and efficiency of Neural Architecture Search (NAS) presents a comprehensive challenge that necessitates innovative approaches. Herein, we outline several directions for future research aimed at refining NAS processes to ensure they are interpretably fair, robust, and generalizable across domains, while also optimizing computational and data efficiency.

As the primary aim is to embed different trustworthiness principles such as interpretability, fairness, robustness, generalization, and efficiency directly into the NAS framework, here are some future directions that can be studied:

- **Interpretability:** Integrating understandable components and regularization techniques to foster architectures that are transparent and comprehensible.
- **Fairness:** Including fairness metrics in the evaluation criteria and enforcing search constraints to avoid bias, ensuring solutions are equitable across diverse groups.
- **Robustness:** Employing strategies like adversarial training and regularization to enhance resilience against perturbations and attacks.
- **Generalization:** Leveraging meta-learning and transfer learning to improve performance on unseen data, extending applicability beyond the training domain.
- **Advanced Sampling Techniques:** Employing weight sharing and one-shot NAS, and leveraging architectural priors to streamline the search process.
- **Data Efficiency:** Utilizing active learning, transfer learning, and domain adaptation to optimize NAS performance with limited or skewed datasets.
- **Adaptive NAS:** Developing models capable of dynamic adjustment to evolving data and environments through online NAS or continuous learning mechanisms.
- **Domain-Specific NAS Methods:** Creating approaches tailored for challenges in fields such as graph neural networks, reinforcement learning, and generative models.
- **Correlational Studies:** Exploring the relationship between NAS-generated architectures and specific problem characteristics.
- **Optimization Techniques:** Implementing weight sharing and one-shot architectures, incorporating existing knowledge to streamline the search.
- **Data-Driven Efficiency:** Enhancing data efficiency through active learning and domain-specific adaptation techniques.

- **Customized NAS for Unique Architectures:** Designing NAS solutions for specialized structures such as recurrent and graph neural networks.
- **Environmental Adaptability:** Crafting NAS models with intrinsic capability to adapt to data and environmental shifts.
- **New Domain Applications:** Exploring NAS in areas like reinforcement learning and generative models, requiring domain-specific innovations.
- **Efficiency and Adaptability:** Refining NAS to improve its efficiency and adaptability across a broader spectrum of applications.

In summary, the future of NAS lies in addressing multidimensional challenges through dedicated research and development efforts, significantly enhancing the potential of automated architecture design.

Bibliography

[1] https://www.cancer.net/cancer-types/brain-tumor/statistics.

[2] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. *Advances in neural information processing systems*, 31, 2018.

[3] Parnian Afshar, Arash Mohammadi, and Konstantinos N. Plataniotis. Brain tumor type classification via capsule networks, 2018.

[4] David Alvarez-Melis and Tommi S Jaakkola. Towards robust interpretability with self-explaining neural networks. *arXiv preprint arXiv:1806.07538*, 2018.

[5] Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. Spice: Semantic propositional image caption evaluation. In *European conference on computer vision*, pages 382–398. Springer, 2016.

[6] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6077–6086, 2018.

[7] Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: a query-efficient black-box adversarial attack via random search. In *European Conference on Computer Vision*, pages 484–501. Springer, 2020.

[8] Viktar Atliha and Dmitrij Šešok. Image-captioning model compression. *Applied Sciences*, 12(3):1638, 2022.

[9] David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert Müller. How to explain individual classification decisions. *Journal of Machine Learning Research*, 11(Jun):1803–1831, 2010.

[10] Satanjeev Banerjee and Alon Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

[11] Atilim Gunes Baydin, Robert Cornish, David Martínez-Rubio, Mark Schmidt, and Frank D. Wood. Online learning rate adaptation with hypergradient descent. *CoRR*, abs/1703.04782, 2017.

[12] Atilim Gunes Baydin, Robert Cornish, David Martinez Rubio, Mark Schmidt, and Frank Wood. Online learning rate adaptation with hypergradient descent. *arXiv preprint arXiv:1703.04782*, 2017.

[13] Sartaj Bhuvaji, Ankita Kadam, Prajakta Bhumkar, Sameer Dedge, and Swati Kanchan. Brain tumor classification (mri), 2020.

[14] Melissa L Bondy, Michael E Scheurer, Beatrice Malmer, Jill S Barnholtz-Sloan, Faith G Davis, Dora Il'Yasova, Carol Kruchko, Bridget J McCarthy, Preetha Rajaraman, Judith A Schwartzbaum, et al. Brain tumor epidemiology: consensus from the brain tumor epidemiology consortium. *Cancer*, 113(S7):1953–1968, 2008.

[15] Akhilan Boopathy, Tsui-Wei Weng, Pin-Yu Chen, Sijia Liu, and Luca Daniel. Cnn-cert: An efficient framework for certifying robustness of convolutional neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3240–3247, 2019.

[16] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. In *ICLR*, 2019.

[17] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE, 2017.

[18] Francesco Paolo Casale, Jonathan Gordon, and Nicoló Fusi. Probabilistic neural architecture search. *CoRR*, abs/1902.05116, 2019.

[19] Alvin Chan, Yi Tay, Yew Soon Ong, and Jie Fu. Jacobian adversarially regularized networks for robustness. *arXiv preprint arXiv:1912.10185*, 2019.

[20] Jianlong Chang, Yiwen Guo, GAOFENG MENG, SHIMING XIANG, Chunhong Pan, et al. Data: Differentiable architecture approximation. *Advances in Neural Information Processing Systems*, 32:876–886, 2019.

[21] Daoyuan Chen, Yaliang Li, Minghui Qiu, Zhen Wang, Bofang Li, Bolin Ding, Hongbo Deng, Jun Huang, Wei Lin, and Jingren Zhou. Adabert: Task-adaptive bert compression with differentiable neural architecture search. *arXiv preprint arXiv:2001.04246*, 2020.

[22] Hongge Chen, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, and Cho-Jui Hsieh. Show-and-fool: Crafting adversarial examples for neural image captioning. *arXiv preprint arXiv:1712.02051*, 2017.

[23] Minghao Chen, Houwen Peng, Jianlong Fu, and Haibin Ling. Autoformer: Searching transformers for visual recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12270–12280, 2021.

[24] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 15–26, 2017.

[25] Shengcong Chen, Changxing Ding, and Minfeng Liu. Dual-force convolutional neural networks for accurate brain tumor segmentation. *Pattern Recognition*, 88:90–100, 2019.

[26] Xiangning Chen and Cho-Jui Hsieh. Stabilizing differentiable architecture search via perturbation-based regularization. In *International conference on machine learning*, pages 1554–1565. PMLR, 2020.

[27] Xiangning Chen and Cho-Jui Hsieh. Stabilizing differentiable architecture search via perturbation-based regularization. *CoRR*, abs/2002.05283, 2020.

[28] Xiangning Chen, Ruochen Wang, Minhao Cheng, Xiaocheng Tang, and Cho-Jui Hsieh. Drnas: Dirichlet neural architecture search. *CoRR*, abs/2006.10355, 2020.

[29] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *ICCV*, 2019.

[30] Minhao Cheng, Thong Le, Pin-Yu Chen, Jinfeng Yi, Huan Zhang, and Cho-Jui Hsieh. Query-efficient hard-label black-box attack: An optimization-based approach. *arXiv preprint arXiv:1807.04457*, 2018.

[31] Minhao Cheng, Jinfeng Yi, Pin-Yu Chen, Huan Zhang, and Cho-Jui Hsieh. Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples. In *AAAI*, pages 3601–3608, 2020.

[32] Shubham Chitnis, Ramtin Hosseini, and Pengtao Xie. Brain tumor classification based on neural architecture search. *Scientific Reports*, 12(1):19206, 2022.

[33] Kyunghyun Cho, Aaron Courville, and Yoshua Bengio. Describing multimedia content using attention-based encoder-decoder networks. *IEEE Transactions on Multimedia*, 17(11):1875–1886, 2015.

[34] Sang Keun Choe, Willie Neiswanger, Pengtao Xie, and Eric Xing. Betty: An automatic differentiation library for multilevel optimization. *arXiv preprint arXiv:2207.02849*, 2022.

[35] Xiangxiang Chu, Xiaoxing Wang, Bo Zhang, Shun Lu, Xiaolin Wei, and Junchi Yan. DARTS-: robustly stepping out of performance collapse without indicators. *CoRR*, abs/2009.01027, 2020.

[36] Xiangxiang Chu, Tianbao Zhou, Bo Zhang, and Jixiang Li. Fair DARTS: eliminating unfair advantages in differentiable architecture search. *CoRR*, abs/1911.12126, 2019.

[37] Noel Codella, Veronica Rotemberg, Philipp Tschandl, M Emre Celebi, Stephen Dusza, David Gutman, Brian Helba, Aadi Kalloo, Konstantinos Liopyris, Michael Marchetti, et al. Skin lesion analysis toward melanoma detection 2018: A challenge hosted by the international skin imaging collaboration (isic). *arXiv preprint arXiv:1902.03368*, 2019.

[38] Jeremy M Cohen, Elan Rosenfeld, and J Zico Kolter. Certified adversarial robustness via randomized smoothing. *arXiv preprint arXiv:1902.02918*, 2019.

[39] Marcella Cornia, Matteo Stefanini, Lorenzo Baraldi, and Rita Cucchiara. Meshed-memory transformer for image captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10578–10587, 2020.

[40] Nicolas Couellan and Wenjuan Wang. On the convergence of stochastic bi-level gradient methods. *Optimization*, 2016.

[41] Francesco Croce and Matthias Hein. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *International Conference on Machine Learning*, pages 2196–2205. PMLR, 2020.

[42] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *ICML*, 2020.

[43] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation policies from data, 2019.

[44] Mary Elizabeth Davis. Glioblastoma: Overview of disease and treatment. *Clinical journal of oncology nursing*, 20(5 Suppl):S2–S8, Oct 2016. 27668386[pmid].

[45] S Deepak and PM Ameer. Brain tumour classification using siamese neural network and neighbourhood analysis in embedded feature space. *International Journal of Imaging Systems and Technology*, 31(3):1655–1669, 2021.

[46] 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.

[47] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[48] Chaitanya Devaguptapu, Devansh Agarwal, Gaurav Mittal, and Vineeth N Balasubramanian. An empirical study on the robustness of nas based architectures. *arXiv preprint arXiv:2007.08428*, 2020.

[49] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[50] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[51] Francisco Javier Díaz-Pernas, Mario Martínez-Zarzuela, Míriam Antón-Rodríguez, and David González-Ortega. A deep learning approach for brain tumor classification and segmentation using a multiscale convolutional neural network. In *Healthcare*, volume 9, page 153. Multidisciplinary Digital Publishing Institute, 2021.

[52] Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four GPU hours. In *CVPR*, 2019.

[53] Xuefeng Du and Pengtao Xie. Small-group learning, with application to neural architecture search. *arXiv preprint arXiv:2012.12502*, 2020.

[54] Xuefeng Du, Haochen Zhang, and Pengtao Xie. Learning by passing tests, with application to neural architecture search. *arXiv preprint arXiv:2011.15102*, 2020.

[55] Krishnamurthy Dvijotham, Robert Stanforth, Sven Gowal, Timothy A Mann, and Pushmeet Kohli. A dual approach to scalable verification of deep networks. In *UAI*, volume 1, page 2, 2018.

[56] Christian Etmann, Sebastian Lunz, Peter Maass, and Carola-Bibiane Schönlieb. On the connection between adversarial robustness and saliency map interpretability. *arXiv preprint arXiv:1905.04172*, 2019.

[57] Matthias Feurer, Jost Springenberg, and Frank Hutter. Initializing bayesian hyperparameter optimization via meta-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2015.

[58] Matthias Feurer, Jost Tobias Springenberg, and Frank Hutter. Initializing bayesian hyperparameter optimization via meta-learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, page 1128–1135. AAAI Press, 2015.

[59] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[60] Bhanu Garg, Li Zhang, Pradyumna Sridhara, Ramtin Hosseini, Eric Xing, and Pengtao Xie. Learning from mistakes–a framework for neural architecture search. *arXiv preprint arXiv:2111.06353*, 2021.

[61] Bhanu Garg, Li Lyna Zhang, Pradyumna Sridhara, Ramtin Hosseini, Eric Xing, and Pengtao Xie. Learning from mistakes - a framework for neural architecture search. In *AAAI Conference on Artificial Intelligence*, 2021.

[62] Saeed Ghadimi and Mengdi Wang. Approximation methods for bilevel programming. *arXiv preprint arXiv:1802.02246*, 2018.

[63] Navid Ghassemi, Afshin Shoeibi, and Modjtaba Rouhani. Deep neural network with generative adversarial networks pre-training for brain tumor classification based on mr images. *Biomedical Signal Processing and Control*, 57:101678, 2020.

[64] Palash Ghosal, Lokesh Nandanwar, Swati Kanchan, Ashok Bhadra, Jayasree Chakraborty, and Debashis Nandi. Brain tumor classification using resnet-101 based squeeze and excitation deep neural network. In *2019 Second International Conference on Advanced Computational and Communication Paradigms (ICACCP)*, pages 1–6. IEEE, 2019.

[65] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[66] Raghuraman Gopalan, Ruonan Li, and Rama Chellappa. Domain adaptation for object recognition: An unsupervised approach. In *2011 International Conference on Computer Vision*, pages 999–1006, 2011.

[67] Riccardo Grazzi, Luca Franceschi, Massimiliano Pontil, and Saverio Salzo. On the iteration complexity of hypergradient computation. In *International Conference on Machine Learning*, pages 3748–3758. PMLR, 2020.

[68] Arthur Gretton, Karsten Borgwardt, Malte J Rasch, Bernhard Scholkopf, and Alexander J Smola. A kernel method for the two-sample problem. *arXiv preprint arXiv:0805.2368*, 2008.

[69] Arthur Gretton, Alex Smola, Jiayuan Huang, Marcel Schmittfull, Karsten M. Borgwardt, Bernhard Schölkopf, Quiñonero Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D. Lawrence. Covariate shift by kernel mean matching. In *NIPS 2009*, 2009.

[70] Yu-Chao Gu, Li-Juan Wang, Yun Liu, Yi Yang, Yu-Huan Wu, Shao-Ping Lu, and Ming-Ming Cheng. Dots: Decoupling operation and topology in differentiable architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12311–12320, 2021.

[71] Minghao Guo, Yuzhe Yang, Rui Xu, Ziwei Liu, and Dahua Lin. When nas meets robustness: In search of robust architectures against adversarial attacks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 631–640, 2020.

[72] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *European Conference on Computer Vision*, pages 544–560. Springer, 2020.

[73] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. Don't stop pretraining: Adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964*, 2020.

[74] Ruqian Hao, Khashayar Namdar, Lin Liu, and Farzad Khalvati. A transfer learning–based active learning framework for brain tumor classification. *Frontiers in Artificial Intelligence*, 4, 2021.

[75] Mohammad Havaei, Axel Davy, David Warde-Farley, Antoine Biard, Aaron Courville, Yoshua Bengio, Chris Pal, Pierre-Marc Jodoin, and Hugo Larochelle. Brain tumor segmentation with deep neural networks. *Medical image analysis*, 35:18–31, 2017.

[76] Chaoyang He, Haishan Ye, Li Shen, and Tong Zhang. Milenas: Efficient neural architecture search via mixed-level reformulation, 2020.

[77] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[78] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[79] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[80] Judy Hoffman, Daniel A Roberts, and Sho Yaida. Robust learning with jacobian regularization. *arXiv preprint arXiv:1908.02729*, 2019.

[81] Weijun Hong, Guilin Li, Weinan Zhang, Ruiming Tang, Yunhe Wang, Zhenguo Li, and Yong Yu. Dropnas: Grouped operation dropout for differentiable architecture search. In *IJCAI*, 2020.

[82] Ramtin Hosseini and Pengtao Xie. Learning by self-explanation, with application to neural architecture search. *arXiv preprint arXiv:2012.12899*, 2020.

[83] Ramtin Hosseini and Pengtao Xie. Image understanding by captioning with differentiable architecture search. In *Proceedings of the 30th ACM International Conference on Multimedia*, MM '22, page 4665–4673, New York, NY, USA, 2022. Association for Computing Machinery.

[84] Ramtin Hosseini and Pengtao Xie. Saliency-aware neural architecture search. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[85] Ramtin Hosseini, Xingyi Yang, and Pengtao Xie. Dsrna: Differentiable search of robust neural architectures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6196–6205, 2021.

[86] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017.

[87] Shoukang Hu, Sirui Xie, Hehui Zheng, Chunxiao Liu, Jianping Shi, Xunying Liu, and Dahua Lin. DSNAS: direct neural architecture search without parameter retraining. In *CVPR*, 2020.

[88] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

[89] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks, 2018.

[90] Lun Huang, Wenmin Wang, Jie Chen, and Xiao-Yong Wei. Attention on attention for image captioning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4634–4643, 2019.

[91] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. *arXiv preprint arXiv:1804.08598*, 2018.

[92] Ali Işın, Cem Direkoğlu, and Melike Şah. Review of mri-based brain tumor image segmentation using deep learning methods. *Procedia Computer Science*, 102:317–324, 2016.

[93] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 1991.

[94] Daniel Jakubovitz and Raja Giryes. Improving dnn robustness to adversarial attacks using jacobian regularization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 514–529, 2018.

[95] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[96] I-Hong Jhuo, Dong Liu, D. T. Lee, and Shih-Fu Chang. Robust visual domain adaptation with low-rank reconstruction. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2168–2175, 2012.

[97] Kaiyi Ji, Junjie Yang, and Yingbin Liang. Bilevel optimization: Convergence analysis and enhanced design. In *International Conference on Machine Learning*, pages 4882–4892. PMLR, 2021.

[98] Wenhao Jiang, Lin Ma, Yu-Gang Jiang, Wei Liu, and Tong Zhang. Recurrent fusion network for image captioning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 499–515, 2018.

[99] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*, 2019.

[100] HNTK Kaldera, Shanaka Ramesh Gunasekara, and Maheshi B Dissanayake. Brain tumor classification and segmentation using faster r-cnn. In *2019 Advances in Science and Engineering Technology International Conferences (ASET)*, pages 1–6. IEEE, 2019.

[101] Guoliang Kang, Lu Jiang, Yi Yang, and Alexander G Hauptmann. Contrastive adaptation network for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4893–4902, 2019.

[102] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3128–3137, 2015.

[103] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[104] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[105] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision*, 123(1):32–73, 2017.

[106] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[107] Alex Krizhevsky and Geoff Hinton. Convolutional deep belief networks on cifar-10. *Unpublished manuscript*, 40(7):1–9, 2010.

[108] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Citeseer*, 2009.

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

[110] S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[111] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. *public*, 2010.

[112] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 656–672. IEEE, 2019.

[113] Tao Lei, Regina Barzilay, and Tommi Jaakkola. Rationalizing neural predictions. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016.

[114] Liam Li, Mikhail Khodak, Maria-Florina Balcan, and Ameet Talwalkar. Geometry-aware gradient algorithms for neural architecture search, 2021.

[115] Xiujun Li, Xi Yin, Chunyuan Li, Pengchuan Zhang, Xiaowei Hu, Lei Zhang, Lijuan Wang, Houdong Hu, Li Dong, Furu Wei, et al. Oscar: Object-semantics aligned pre-training for vision-language tasks. In *European Conference on Computer Vision*, pages 121–137. Springer, 2020.

[116] Hanwen Liang, Shifeng Zhang, Jiacheng Sun, Xingqiu He, Weiran Huang, Kechen Zhuang, and Zhenguo Li. Darts+: Improved differentiable architecture search with early stopping. *arXiv preprint arXiv:1909.06035*, 2019.

[117] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

[118] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[119] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 19–34, 2018.

[120] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan L. Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *ECCV*, 2018.

[121] Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. Hierarchical representations for efficient architecture search. In *ICLR*, 2018.

[122] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: differentiable architecture search. In *ICLR*, 2019.

[123] Risheng Liu, Yaohua Liu, Shangzhi Zeng, and Jin Zhang. Towards gradient-based bilevel optimization with non-convex followers and beyond. *Advances in Neural Information Processing Systems*, 34, 2021.

[124] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[125] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021.

[126] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pages 3730–3738, 2015.

[127] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *International conference on machine learning*, pages 97–105. PMLR, 2015.

[128] David N. Louis, Hiroko Ohgaki, Otmar D. Wiestler, Webster K. Cavenee, Peter C. Burger, Anne Jouvet, Bernd W. Scheithauer, and Paul Kleihues. The 2007 who classification of tumours of the central nervous system. *Acta Neuropathologica*, 114(2):97–109, Aug 2007.

[129] Diyuan Lu, Nenad Polomac, Iskra Gacheva, Elke Hattingen, and Jochen Triesch. Human-expert-level brain tumor detection using deep learning with data distillation and augmentation. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3975–3979. IEEE, 2021.

[130] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet V2: practical guidelines for efficient CNN architecture design. In *ECCV*, 2018.

[131] Marc C Mabray, Ramon F Barajas, and Soonmee Cha. Modern brain tumor imaging. *Brain tumor research and treatment*, 3(1):8–23, 2015.

[132] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[133] Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, and Alan L Yuille. Explain images with multimodal recurrent neural networks. *arXiv preprint arXiv:1410.1090*, 2014.

[134] Bjoern H Menze, Andras Jakab, Stefan Bauer, Jayashree Kalpathy-Cramer, Keyvan Farahani, Justin Kirby, Yuliya Burren, Nicole Porz, Johannes Slotboom, Roland Wiest, et al. The multimodal brain tumor image segmentation benchmark (brats). *IEEE transactions on medical imaging*, 34(10):1993–2024, 2014.

[135] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2017.

[136] James Mullenbach, Sarah Wiegreffe, Jon Duke, Jimeng Sun, and Jacob Eisenstein. Explainable prediction of medical codes from clinical text. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1101–1111, 2018.

[137] Hiba Mzoughi, Ines Njeh, Ali Wali, Mohamed Ben Slima, Ahmed BenHamida, Chokri Mhiri, and Kharedine Ben Mahfoudhe. Deep multi-scale 3d convolutional neural network (cnn) for mri gliomas brain tumor classification. *Journal of Digital Imaging*, 33(4):903–915, 2020.

[138] Sinno Jialin Pan, Ivor W. Tsang, James T. Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210, 2011.

[139] Yingwei Pan, Ting Yao, Yehao Li, and Tao Mei. X-linear attention networks for image captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10971–10980, 2020.

[140] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[141] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[142] Linmin Pei, Lasitha Vidyaratne, Md Monibor Rahman, and Khan M Iftekharuddin. Context aware deep learning for brain tumor segmentation, subtype classification, and survival prediction using radiology images. *Scientific Reports*, 10(1):1–11, 2020.

[143] Sérgio Pereira, Adriano Pinto, Victor Alves, and Carlos A Silva. Brain tumor segmentation using convolutional neural networks in mri images. *IEEE transactions on medical imaging*, 35(5):1240–1251, 2016.

[144] Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. In *ICML*, 2018.

[145] Vipin Pillai and Hamed Pirsiavash. Explainable models with consistent interpretations. *UMBC Student Collection*, 2021.

[146] Yu Qin, Jiajun Du, Yonghua Zhang, and Hongtao Lu. Look back and predict forward in image captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8367–8375, 2019.

[147] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Regularized Evolution for Image Classifier Architecture Search. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):4780–4789, July 2019. Number: 01.

[148] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 4780–4789, 2019.

[149] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc Le, and Alex Kurakin. Large-scale evolution of image classifiers. *arXiv preprint arXiv:1703.01041*, 2017.

[150] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[151] Zhongzheng Ren, Raymond Yeh, and Alexander Schwing. Not all unlabeled data are equal: Learning to weight data in semi-supervised learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 21786–21797. Curran Associates, Inc., 2020.

[152] Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. Self-critical sequence training for image captioning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7008–7024, 2017.

[153] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144. ACM, 2016.

[154] Laura Rieger, Chandan Singh, William Murdoch, and Bin Yu. Interpretations are useful: penalizing explanations to align neural networks with prior knowledge. In *International Conference on Machine Learning*, pages 8116–8126. PMLR, 2020.

[155] Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. *Advances in Neural Information Processing Systems*, 34, 2021.

[156] Andrew Slavin Ross, Michael C. Hughes, and Finale Doshi-Velez. Right for the right reasons: Training differentiable models by constraining their explanations. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 2662–2670, 2017.

[157] Muhammad Sajjad, Salman Khan, Khan Muhammad, Wanqing Wu, Amin Ullah, and Sung Wook Baik. Multi-grade brain tumor classification using deep cnn with extensive data augmentation. *Journal of computational science*, 30:174–182, 2019.

[158] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[159] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[160] Nagur Shareef Shaik and Teja Krishna Cherukuri. Multi-level attention network: application to brain tumor classification. *Signal, Image and Video Processing*, pages 1–8, 2021.

[161] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

[162] Parth Sheth, Yueyu Jiang, and Pengtao Xie. Learning by teaching, with application to neural architecture search. *arXiv preprint arXiv:2103.07009*, 2021.

[163] Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng. Meta-weight-net: Learning an explicit mapping for sample weighting. In *Advances in Neural Information Processing Systems*, pages 1919–1930, 2019.

[164] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[165] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[166] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

[167] Akshayvarun Subramanya, Vipin Pillai, and Hamed Pirsiavash. Fooling network interpretation in image classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2020–2029, 2019.

[168] Felipe Petroski Such, Aditya Rawal, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. Generative teaching networks: Accelerating neural architecture search by learning to generate synthetic training data. *CoRR*, abs/1912.07768, 2019.

[169] Baochen Sun, Jiashi Feng, and Kate Saenko. Correlation alignment for unsupervised domain adaptation. *Domain adaptation in computer vision applications*, pages 153–171, 2017.

[170] Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. Patient knowledge distillation for bert model compression. *arXiv preprint arXiv:1908.09355*, 2019.

[171] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3319–3328. JMLR. org, 2017.

[172] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[173] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[174] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V. Le. Mnasnet: Platform-aware neural architecture search for mobile. In *CVPR*, 2019.

[175] Raphael Tang, Yao Lu, Linqing Liu, Lili Mou, Olga Vechtomova, and Jimmy Lin. Distilling task-specific knowledge from bert into simple neural networks. *arXiv preprint arXiv:1903.12136*, 2019.

[176] Philipp Tschandl, Cliff Rosendahl, and Harald Kittler. The ham10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. *Scientific data*, 5(1):1–9, 2018.

[177] Chun-Chen Tu, Paishun Ting, Pin-Yu Chen, Sijia Liu, Huan Zhang, Jinfeng Yi, Cho-Jui Hsieh, and Shin-Ming Cheng. Autozoom: Autoencoder-based zeroth order optimization method for attacking black-box neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 742–749, 2019.

[178] Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575, 2015.

[179] Luis N. Vicente and Paul H. Calamai. Bilevel and multilevel programming: A bibliography review, 1994.

[180] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164, 2015.

[181] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics.

[182] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

[183] Shuai Wang, Bo Kang, Jinlu Ma, Xianjun Zeng, Mingming Xiao, Jia Guo, Mengjiao Cai, Jingyi Yang, Yaodong Li, Xiangfei Meng, et al. A deep learning algorithm using ct images to screen for corona virus disease (covid-19). *medRxiv*, 2020.

[184] Yulin Wang, Jiayi Guo, Shiji Song, and Gao Huang. Meta-semi: A meta-learning approach for semi-supervised learning. *CoRR*, abs/2007.02394, 2020.

[185] Zhibo Wang, Xiaowei Dong, Henry Xue, Zhifei Zhang, Weifeng Chiu, Tao Wei, and Kui Ren. Fairness-aware adversarial perturbation towards bias mitigation for deployed deep models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10379–10388, 2022.

[186] Tsui-Wei Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Duane Boning, Inderjit S Dhillon, and Luca Daniel. Towards fast computation of certified robustness for relu networks. *arXiv preprint arXiv:1804.09699*, 2018.

[187] Yu Weng, Tianbao Zhou, Yujie Li, and Xiaoyu Qiu. Nas-unet: Neural architecture search for medical image segmentation. *IEEE Access*, 7:44247–44257, 2019.

[188] Yu Weng, Tianbao Zhou, Lei Liu, and Chunlei Xia. Automatic convolutional neural architecture search for image classification under different scenes. *IEEE Access*, 7:38495–38506, 2019.

[189] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

[190] Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, pages 5286–5295. PMLR, 2018.

[191] Eric Wong, Leslie Rice, and Zico J. Kolter. Fast is better than free: Revisiting adversarial training. *ICLR*, 2020.

[192] Pengtao Xie, Xuefeng Du, and Hao Ban. Skillearn: Machine learning inspired by humans' learning skills. *arXiv preprint arXiv:2012.04863*, 2020.

[193] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. SNAS: stochastic neural architecture search. In *ICLR*, 2019.

[194] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. Pc-darts: Partial channel connections for memory-efficient differentiable architecture search. *arXiv preprint arXiv:1907.05737*, 2019.

[195] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. Pc-darts: Partial channel connections for memory-efficient architecture search, 2020.

[196] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. PC-DARTS: partial channel connections for memory-efficient architecture search. In *ICLR*, 2020.

[197] Junjie Yang, Kaiyi Ji, and Yingbin Liang. Provably faster algorithms for bilevel optimization. *Advances in Neural Information Processing Systems*, 34, 2021.

[198] Xu Yang, Kaihua Tang, Hanwang Zhang, and Jianfei Cai. Auto-encoding scene graphs for image captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10685–10694, 2019.

[199] Yibo Yang, Hongyang Li, Shan You, Fei Wang, Chen Qian, and Zhouchen Lin. Ista-nas: Efficient and consistent neural architecture search by sparse coding, 2020.

[200] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, 2016.

[201] Ting Yao, Yingwei Pan, Yehao Li, and Tao Mei. Exploring visual relationship for image captioning. In *Proceedings of the European conference on computer vision (ECCV)*, pages 684–699, 2018.

[202] Ting Yao, Yingwei Pan, Yehao Li, Zhaofan Qiu, and Tao Mei. Boosting image captioning with attributes. In *Proceedings of the IEEE international conference on computer vision*, pages 4894–4902, 2017.

[203] Quanzeng You, Hailin Jin, Zhaowen Wang, Chen Fang, and Jiebo Luo. Image captioning with semantic attention. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4651–4659, 2016.

[204] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zi-Hang Jiang, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 558–567, 2021.

[205] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pages 818–833. Springer, 2014.

[206] Arber Zela, Thomas Elsken, Tonmoy Saikia, Yassine Marrakchi, Thomas Brox, and Frank Hutter. Understanding and robustifying differentiable architecture search. *arXiv preprint arXiv:1909.09656*, 2019.

[207] Arber Zela, Thomas Elsken, Tonmoy Saikia, Yassine Marrakchi, Thomas Brox, and Frank Hutter. Understanding and robustifying differentiable architecture search. In *ICLR*, 2020.

[208] Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. In *Advances in neural information processing systems*, pages 4939–4948, 2018.

[209] Le Zhang, Yanshuo Zhang, Xin Zhao, and Zexiao Zou. Image captioning via proximal policy optimization. *Image and Vision Computing*, 108:104126, 2021.

[210] Lianbo Zhang, Shaoli Huang, Wei Liu, and Dacheng Tao. Learning a mixture of granularity-specific experts for fine-grained categorization. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 8330–8339. IEEE, 2019.

[211] Pengchuan Zhang, Xiujun Li, Xiaowei Hu, Jianwei Yang, Lei Zhang, Lijuan Wang, Yejin Choi, and Jianfeng Gao. Vinvl: Revisiting visual representations in vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5579–5588, 2021.

[212] Guoqing Zheng, Ahmed Hassan Awadallah, and Susan T. Dumais. Meta label correction for learning with weak supervision. *CoRR*, abs/1911.03809, 2019.

[213] Pan Zhou, Caiming Xiong, Richard Socher, and Steven C. H. Hoi. Theory-inspired path-regularized differential network architecture search. *CoRR*, abs/2006.16537, 2020.

[214] Jihong Zhu and Jihong Pei. Progressive kernel pruning with saliency mapping of input-output channels. *Neurocomputing*, 467:360–378, 2022.

[215] Wenwu Zhu, Xin Wang, and Pengtao Xie. Self-directed machine learning. *arXiv preprint arXiv:2201.01289*, 2022.

[216] Xinxin Zhu, Weining Wang, Longteng Guo, and Jing Liu. Autocaption: Image captioning with neural architecture search. *arXiv preprint arXiv:2012.09742*, 2020.

[217] Luisa M Zintgraf, Taco S Cohen, Tameem Adel, and Max Welling. Visualizing deep neural network decisions: Prediction difference analysis. *International Conference on Learning Representations*, 2017.

[218] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *ICLR*, 2017.

[219] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *CVPR*, 2018.

[220] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2847–2856, 2018.