Universiteit Leiden

Master Computer Science

Understanding MAML Through Its Loss Landscape

Name:	Bosong Ding
Student ID:	s3078930
Date:	13/07/2023
Specialisation:	Data Science: Computer Science
0th supervisor:	Mike Huisman
1st supervisor:	Jan N. van Rijn
2nd supervisor:	Aske Plaat

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1_2
2333 CA Leiden

Understanding MAML Through Its Loss Landscape

Bosong Ding

July 14, 2023

Abstract

Deep learning methods have achieved remarkable successes in various domains, but the need for large amounts of labeled data poses a significant challenge, thus limiting their practical applicability. Meta-learning is one approach to overcome this limitation by endowing these methods with the ability to perform few-shot learning (rapidly learning to perform new tasks using minimal examples) by exploiting prior knowledge. Whilst many meta-learning methods are being proposed, few studies have concentrated on increasing our understanding of how these methods work and what aspects influence their effectiveness. In this thesis, we focus specifically on arguably the most popular Meta-learning technique Model-Agnostic Meta-Learning (MAML) and examine the role and behavior of the loss landscape in shaping MAML's ability to quickly learn to perform new tasks and provide a novel view of the optimization process through loss landscape visualizations. More specifically, we investigate the effect of the used backbone and the use of data augmentation on the loss landscape as well as the relationship between the loss landscape and the few-shot learning ability of MAML. Our results support the finding that MAML predominantly reuses its learned features rather than quickly adapting them. We observe that a lower loss and sensitivity to perturbations in the training loss landscape often signify better test performance in MAML, which provides a new perspective in understanding the relationship between the loss landscape and model accuracy. Moreover, our research reveals the enhancing role of data augmentation across various architectures and backbones. These discoveries pave the way for a better understanding and improvement of MAML and other similar few-shot learning algorithms.

Contents

1 Introduction **4**

2 Related Work **6**

3 Background **8**

3.1 Few-shot Learning Algorithms 8

3.1.1 Few-shot Learning Problem 8

3.1.2 Transfer Learning 9

3.1.3 Model-Agonistic Meta-Learning 10

3.2 Visualization Method 11

3.3 Sharpness 13

4 Experiment **14**

4.1 Dataset 15

4.2 Comparative Analysis of Few-shot Learning Algorithms 16

4.2.1 Reproduced Results 16

4.2.2 Loss-Accuracy Analysis 17

4.3 Visualization 21

4.3.1 Inner Loop 21

4.3.2 Outer Loop 27

4.4 Quantitative Assessment of Loss Landscape Visualizations 32

5 Discussion **36**

6 Conclusion **38**

A Hyperparameters **42**

B Complementary Visualizations **43**

C Dual Norm of Sharpness **46**

D Sharpness & Gradient **47**

E Sharpness Results **49**

F Overfitting **49**

1 Introduction

Over the past few years, the advent of deep learning [14][10] has ushered in remarkable advancements across various fields, including image recognition [18][23][12], natural language processing [26][6]. Deep learning, a subset of machine learning, employs complex neural networks to learn from a dataset and make predictions about future data. Despite its vast potential and capabilities, deep learning is notorious for its insatiable data demands, often requiring massive amounts of labeled data to perform well. This requirement poses significant challenges, constrains its applicability in resource-limited settings, and makes the learning process both expensive and time-consuming.

This limitation has sparked attention in the field of few-shot learning, which seeks to develop deep learning models that can quickly adapt to new tasks, such as image classification or text translation, after observing only a small number of examples. Transfer learning [29] is often introduced as a baseline for few-shot learning algorithms. It involves a two-step process: first, it trains a model on a large, rich dataset, then it fine-tunes the learned model on new tasks with much less data available.

An alternative approach to transfer learning and a solution to the few-shot learning problem is meta-learning [15][24][27][22]. The general aim of meta-learning is to acquire knowledge across a variety of tasks instead of learning from a large dataset. This approach has attracted increasing attention in recent years as it enables models not merely to perform specific tasks but also to understand the broader structure of these tasks. This broad understanding facilitates a more efficient learning process when encountering new tasks. Among the numerous meta-learning techniques, First-Order Model-Agnostic Meta-Learning (FOMAML), a first-order approximation of the original Model-Agnostic Meta-Learning (MAML) proposed by Finn et al. [8], has made a significant impact. For simplicity and readability, we will refer to FOMAML as MAML throughout the remainder of this paper. MAML introduces a unique training process. This procedure is characterized by a two-step optimization: the outer loop first updates the model's initial parameters based on the performance across tasks, steering the model towards a parameter space where it can swiftly adapt to new tasks. Following this, the inner loop adapts the model to individual tasks using a few gradient steps, effectively learning a task-specific model.

The success of MAML has sparked a variety of studies aiming to delve deeper into and enhance its optimization process. These efforts include evaluations of MAML's performance in comparison with other few-shot learning algorithms under an array of experimental conditions. Chen et al.'s research [3], for instance, investigates MAML's performance under various backbones and datasets. Their findings suggest that these factors can significantly influence the effectiveness of MAML. Furthermore, it has been observed that MAML's optimization process mostly relies on feature reuse rather than rapidly adapting to new tasks [21]. However, despite

these efforts, there remains a gap in understanding the intuitive process of MAML's optimization. Previous research has primarily concentrated on the accuracy performance of MAML in different experimental conditions [3], or convergence properties for non-convex functions of MAML [7]. Surprisingly, few studies have focused on examining the loss, a crucial component when examining the optimization process. As a result, it is not well understood what the optimization loss landscape of MAML looks like and how they are related to its few-shot learning abilities. Studying the loss landscape [19] can provide us with a new perspective on MAML's behavior, which, in turn, can lead to algorithmic improvements and better few-shot learning capabilities.

This thesis aims to fill this research gap as we primarily aim to understand the interplay between the loss landscape of MAML and its few-shot learning ability. We propose the following research questions to guide our investigation:

1. How is the loss landscape of MAML affected by changes in the used backbone and the use of data augmentation?

2. What is the relationship between the loss landscape and the few-shot learning performance of MAML?

To address these research questions, we first conduct a qualitative analysis of the loss landscape in both the inner and outer loops of MAML, utilizing various backbone structures and datasets. This facilitates a more intuitive understanding of its optimization process. Following that, we delve into a quantitative exploration of the loss landscape's characteristics. We implemented a quantifiable metric called 'sharpness', which represents the sensitivity of the loss landscape to perturbations. Our investigation extends to understanding the correlation between this metric and the model's performance in few-shot learning tasks[1]. Our key findings of this thesis can be summarized as follows:

- Our visualizations of the inner loop loss landscapes provide insights into MAML's optimization process, indicating a significant reliance on feature reuse when adapting to new tasks, corroborating findings by Ragu et al. [21].

- We observed that a noisy gradient in the outer loop could destabilize MAML's training process.

- Our study demonstrates that data augmentation consistently enhances the performance of both MAML and transfer learning across various backbone architectures and datasets. By analyzing the loss landscape, we observed that its role in the optimization process could fluctuate between acting as a regularizer and providing additional information.

[1]The source code used in this research is publicly accessible and can be found at the following URL: https://github.com/dingdingding60/Understanding-MAML-Through-Its-Loss-Landscape

- Our study reveals that lower loss and sharpness of the training loss landscape often indicate better test performance in MAML, highlighting the relationship between these metrics and accuracy.

The rest of the thesis is structured as follows. In Section 2, we provide an overview of the studies that either investigate the same topic or have had a significant influence on our work. Section 3 provides an in-depth introduction to the key techniques and methods utilized in this thesis. This includes a detailed discussion of few-shot learning algorithms such as MAML, visualization methods for loss landscapes, and quantifiable measurements for loss and model performance. In Section 4, we present our training results on MAML and transfer learning on different backbones and datasets. We provide a comprehensive analysis of the loss and accuracy results to identify problems in MAML and transfer learning that are often overlooked. Following this, we offer a thorough visualization of MAML along with a quantitative assessment. In Section 5, we summarize and discuss the experiment results, addressing limitations and potential directions for future research. Finally, in Section 6, we summarize the thesis's key contributions to the understanding of MAML and few-shot learning.

2 Related Work

The field of meta-learning has been rigorously studied, leading to the development of numerous advanced techniques. Recurrent Neural Networks (RNNs), such as LSTM [15], have been widely utilized in this domain. Other significant contributions include Matching Networks by Vinyals et al. [27], which utilize attention mechanisms to classify unseen images, and Prototypical Networks by Snell et al. [24], which introduce the concept of class prototypes represented by the mean feature vector of samples from the same class.

Model-Agnostic Meta-Learning [8], a seminal work in the realm of meta-learning, introduces a two-loop optimization structure. Moreover, MAML's unique two-loop optimization process provides an excellent opportunity for visualization and further understanding of the meta-learning optimization process. The separate but linked processes of task-specific adaptation (inner loop) and meta-level update (outer loop) encapsulate the central challenges of meta-learning, making MAML a representative case for study. Visualizing this optimization process can offer insights into how meta-learning algorithms navigate the trade-off between learning across tasks and adapting to specific tasks. These insights could, in turn, shed light on the general behavior of other meta-learning algorithms, as many share similar structural traits or challenges with MAML. This makes MAML not only an important algorithm in its own right but also a valuable lens through which to study the broader field of meta-learning.

Efforts to understand MAML's optimization process are evident in several studies. Some studies focus on analyzing the performance of MAML in different experimental settings. Chen et al. [3] offer an empirical evaluation of MAML and other prevalent few-shot learning algorithms by

exploring various datasets, different backbone structures, and cross-domain scenarios. On the other hand, Bai et al. [2] investigate the impact of the train-validation split on the performance of meta-learning algorithms, emphasizing the significance of this aspect in the meta-learning setup. These studies, while providing valuable insights, focus more on the performance of MAML, whereas our thesis is interested in understanding the actual optimization process and its intricacies.

Pivotal studies focus on the nature of feature learning in MAML. For instance, Raghu et al. [21] challenge the traditional understanding of MAML's optimization process, revealing that the feature layer remains largely unchanged during the adaptation to unseen tasks. This implies that MAML's success can be primarily attributed to feature reuse rather than its capacity for rapid learning. Similarly, Collins et al. [4] provide theoretical insights into the learning process of MAML and its variant, ANIL (Almost No Inner Loop) [21], demonstrating that these methods can provably learn task-invariant representations. Despite these inspiring findings, a deeper understanding of the feature learning process in MAML, particularly how the loss landscape changes during training, remains elusive. Our work aims to contribute to this gap of study. Huisman et al. [16] investigates the learning behavior of meta-learning techniques, namely MAML and Reptile, and compares them to transfer learning. They found that the pre-trained features of transfer learning are more diverse and discriminative than those learned by MAML and Reptile, which specialize in adaptation in low-data regimes of similar data distributions as the one used for training, and this specialization may hamper their ability to generalize to out-of-distribution tasks.

Taking a broader view, the importance of feature representation in few-shot learning is emphasized. Wang et al. [28] introduces a simple yet effective approach for the meta-learning object detection problem setting. They demonstrate that fine-tuning only the last layer of existing detectors on rare classes can yield noticeable performance gains over traditional meta-learning approaches. Similarly, the study by Tian et al. [25] challenges conventional wisdom in meta-learning, arguing that a well-trained embedding model can outperform sophisticated meta-learning algorithms, highlighting the significance of good feature representation. These studies emphasize the importance of feature representation in meta-learning, providing a backdrop to our investigation of MAML's optimization process.

Despite attempts to clarify the nature of MAML, the optimization process, especially the intuitive explanation, remains an area ripe for investigation. The research gap that we are filling lies in providing a comprehensive visualization of MAML's optimization process, a perspective not yet fully explored. To our best knowledge, Sharp-MAML [1] is the first work to apply loss landscape visualization [19] to MAML. Although their analysis was limited, it sparked our interest in exploring MAML through the lens of the loss landscape. Building on their inspiration, we aim to provide a more comprehensive and deeper understanding of the loss landscape in MAML,

examining how it evolves over training and affects the model's performance. By doing so, we hope to shed new light on the intricacies of the MAML optimization process, paving the way for more efficient and effective few-shot learning models.

3 Background

This section provides the background of the methods used in this thesis. First, we explain the concept of few-shot learning. Next, we describe two popular few-shot learning algorithms: transfer learning and Model-Agonistic Meta-Learning (MAML). Then, we introduce a visualization method for understanding the optimization process in neural networks. Finally, we describe the sharpness measurement that can quantify the loss landscape.

3.1 Few-shot Learning Algorithms

In this section, we first explain the concept of few-shot learning problems. Then we will introduce the two popular few-shot algorithms we investigate in this thesis, transfer learning and FOMAML [8].

3.1.1 Few-shot Learning Problem

In the context of image classification, few-shot learning aims to build models that can accurately classify images into distinct categories, even when trained on a limited number of examples per category. This is different from traditional deep learning settings, where neural networks are often trained on massive amounts of labeled data to achieve high performance. The few-shot learning setting can be formalized as follows:

Firstly, like other image classification methods, few-shot learning employs training, validation, and test sets. To distinguish these from the data in the few-shot learning task, they are termed the meta-training set, meta-validation set, and meta-test set, respectively. Each set comprises different image categories sampled from the original dataset.

Let us denote a few-shot learning task as $\mathcal{T}_i$ sampled from a distribution over tasks $p(\mathcal{T})$ that we want our model to be able to adapt to. The associated task dataset is $\mathcal{D}_i$. The task dataset $\mathcal{D}_i$ contains two subsets: the support set $\mathcal{D}_i^{support}$ and the query set $\mathcal{D}_i^{query}$. The support set $\mathcal{D}_i^{support}$ serves as the training set for the task $\mathcal{T}_i$, while the query set $\mathcal{D}_i^{query}$ serves as the test set for the task $\mathcal{T}_i$. In an n-way, k-shot learning setup with q queries, the classifier is initially trained on the support set, which includes n total classes, each having k images. Subsequently, the trained model is evaluated on the query set, where an extra q images for each class from the support set are utilized to assess the model's performance.

The few-shot learning problem addressed in this thesis aims to find an image classifier, de-

noted as f_θ, that trains on the data from the meta-training set. The goal is to maximize its performance for all new tasks derived outside the meta-training set after supplementary training on the support set of the new task. A significant challenge in few-shot learning lies in training a classifier that can learn informative representations and generalize effectively, despite the scarcity of training data. The model must capture the fundamental features of the data and make accurate predictions based on these features.

In the following sections, the two most common algorithms for few-shot learning image classification will be introduced. These approaches will aim to address the challenges associated with learning from limited data and demonstrate their effectiveness in various few-shot learning settings.

Table 1: Common notations used throughout the thesis

Notation	Meaning
$\mathcal{T}_i \leftrightarrow \mathcal{D}_i$	A few-shot learning task drawn from a task distribution $p(\mathcal{T})$. Each task $\mathcal{T}_i$ is associated with a specific dataset $\mathcal{D}_i$.
$\mathcal{D}_i^{\text{support}}, \mathcal{D}_i^{\text{query}} \in \mathcal{D}_i$	The training set and the test set **inside** each few-shot learning task. $\mathcal{D}_i^{\text{support}}$ is also called support set, $\mathcal{D}_i^{\text{query}}$ is also called query set.
$\mathbf{x_i}$	the input image vector in task $\mathcal{T}_i$
$\mathbf{y_i}$	the corresponding one-hot label for input vector $\mathbf{x_i}$ in task $\mathcal{T}_i$
n	The count of distinct image categories in a single task, consistent across both support and query sets.
k	The number of images per distinct category within the support set of a single task.
θ/ϕ	Neural network weights
$(f/g)_\circ$	Neural Network function with parameters $\circ$
$\mathcal{L}(\phi, \mathcal{D})$	The loss function of a neural network with parameters ϕ on a given dataset $\mathcal{D}$
$\nabla_\cdot = \nabla_\cdot \mathcal{L}(\cdot, \mathcal{D})$	the gradient of the loss function $\mathcal{L}$ with respect to the model parameters $\cdot$ for dataset $\mathcal{D}$.

3.1.2 Transfer Learning

Deep learning models typically require a substantial quantity of labeled data to optimize. However, in a few-shot learning setting, the amount of labeled data is naturally scarce. Transfer learning often acts as a common baseline approach in few-shot learning contexts. It directly connects the gap between traditional supervised learning with few-shot learning. The underlying assumption of transfer learning is that a model, when trained on a large dataset, can learn effective features that can then be utilized for an unseen task $\mathcal{T}_i$ with limited data available, consequently enhancing the performance. Transfer learning generally has two stages, pre-training and fine-tuning.

3.1.2.1 Pretraining

The initial step in transfer learning involves training a model on a dataset with abundant

labeled images. This process, known as pre-training, is designed to learn a set of neural network weights θ that can effectively capture shared features and representations across different tasks. Formally, given a source dataset $\mathcal{D}^{\text{train}} = (\mathbf{x}, \mathbf{y})$, the supervised pretraining process aims to learn a function $f_\theta(\mathbf{x})$ that minimizes a loss function $\mathcal{L}(\theta, \mathcal{D}^{\text{train}})$:

$$\arg\min_{\theta} \mathcal{L}\left(\theta, \mathcal{D}^{\text{train}}\right) \tag{1}$$

Where $\mathcal{L}(\theta, \mathcal{D}^{\text{train}})$ denotes the loss between the model's output and the ground truth label. The choice of the loss function and neural network architecture depends on the specific problem domain and dataset.

3.1.2.2 Fine-tuning

Following the supervised pretraining phase, the subsequent step in transfer learning is to fine-tune the model on the target task $\mathcal{T}_i$ with limited labeled data. Fine-tuning involves freezing the feature layers and updating the pre-trained neural network's head's weights θ_{head} using a few-shot learning task's support set $\mathcal{D}_i^{\text{support}}$.

$$\arg\min_{\theta_{head}} \mathcal{L}(\theta_{head}, \mathcal{D}_i^{\text{support}}) \tag{2}$$

where $\mathcal{L}(\theta_{head}, \mathcal{D}_i^{\text{support}})$ is the loss function for the target task. In practice, the fine-tuning process usually involves a smaller learning rate than pretraining to avoid overriding the previously learned representations. This approach is motivated by the observation that earlier layers in deep neural networks tend to capture low-level features that are more generic and transferable across tasks, while the later layers capture higher-order features that are more specific to the task.

3.1.3 Model-Agonistic Meta-Learning

MAML provides a solution to the problem of few-shot learning – how to construct a model that can adapt to new tasks quickly, given only a small number of examples from each task. The key idea behind MAML is to find an initialization of a learning model that is not only good in terms of its performance on a range of tasks but also enables rapid adaptation to new tasks.

3.1.3.1 Inner Loop

The inner loop in MAML corresponds to a task-specific learning process. For each task $\mathcal{T}_i$ drawn from the task distribution $p(\mathcal{T})$, we aim to learn a set of optimal parameters θ that minimizes the loss function $\mathcal{L}$ on the support set $\mathcal{D}_i^{\text{support}}$. This is achieved by taking a few steps of gradient descent:

$$\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}(f_\theta, \mathcal{D}_i^{\text{support}}) \tag{3}$$

Where α is the learning rate, θ_i' denotes the new parameters after the gradient step. This can be considered as a form of fast adaptation to each individual task.

3.1.3.2 Outer Loop

The outer loop of MAML corresponds to meta-optimization. Here, we aim to improve the model's ability to adapt to new tasks by updating the initial parameters θ using information from all tasks. This is done by minimizing the meta-loss function on the query set $\mathcal{D}_i^{\text{query}}$ averaged across all tasks, using the parameters θ_i' obtained from the inner loop for each task:

$$\theta = \theta - \beta \nabla_\theta \mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T})}[\mathcal{L}(f_{\theta_i'}, \mathcal{D}_i^{\text{query}})] \tag{4}$$

where β is the meta-learning rate. The expectation is to take over all tasks at one update. This aims to find the optimal initialization of parameters such that we can minimize the distance to all the tasks and adapt quickly to new tasks.

3.1.3.3 First-Order MAML

The standard MAML algorithm involves second-order derivatives during the meta-optimization phase, which can be computationally intensive, especially when the model is large, or the task is complex. To mitigate this, a variant called First-Order MAML (FOMAML) is commonly used. FOMAML approximates the MAML algorithm by considering only first-order derivatives in the outer loop, without having an obvious impact on performance. The parameter update in the outer loop of FOMAML is then changed to:

$$\theta = \theta - \beta \nabla_\theta \mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T})}[\mathcal{L}(f_{\theta}, \mathcal{D}_i^{\text{query}})] \tag{5}$$

The major difference lies in the fact that unlike MAML, which optimizes the parameters based on the updated parameters θ_i' from the inner loop, FOMAML optimizes based on the original parameters θ but uses θ_i' to calculate the meta-loss, thereby only considering first-order information. This makes FOMAML more computationally efficient compared to traditional MAML. Note that in this thesis, we solely utilize First-Order MAML (FO-MAML). For simplicity, we will refer to FO-MAML as MAML in the subsequent sections.

3.2 Visualization Method

The training process for neural networks depends on our ability to locate effective minimizers for highly non-convex loss functions. It is common knowledge that specific network design elements, such as skip connections, lead to loss functions that are simpler to train. By employing the right visualization method for the loss landscape, we can gain a deeper, more intuitive understanding of the optimization process. In this section, we will first explore the concept of loss landscape visualization, following the approach proposed by Li et al. [19]. Subsequently, we will examine a comprehensive example to demonstrate how this visualization works.

In typical supervised learning, the goal of training a neural network is to minimize the loss $\mathcal{L}(\theta) = \frac{1}{m}\sum_{i=1}^{m} \ell\left(x_i, y_i; \theta\right)$, where x_i and y_i denotes the input and label, m is the number of

data samples, θ is the network parameters, and $\ell\left(x_i, y_i; \theta\right)$ is the loss function that reflects the performance of the network. Neural networks usually contain a huge amount of parameters, resulting in loss functions that lie in an exceedingly high-dimensional space. However, the visualization of loss functions is constrained to low-dimensional formats, such as one-dimensional (line) or two-dimensional (surface) plots. A variety of techniques have been developed to solve this dimensionality disparity.

Goodfellow et al. [11] first use a 1-Dimensional linear interpolation to visualize the linear loss landscape of neural network optimization. They state that the objective of optimizing a neural network is to optimize a randomly initialized parameter θ to a global optimal parameter $\theta^\star$ so that the loss function $J(\theta')$ is minimized. 1-D visualization scans the parameters between θ and $\theta^\star$ with a varying $\alpha \in [-1, 1]$:

$$\theta(\alpha) = (1 - \alpha)\theta + \alpha\theta^\star \tag{6}$$

The visualization is then created by plotting the loss function of different parameters along the way:

$$f(\alpha) = \mathcal{L}(\theta(\alpha)) \tag{7}$$

However, 1-D visualization suffers from two drawbacks. Firstly, it is difficult to reflect the true details of local minimum, i.e., non-convexity. According to Goodfellow et al. [11], the 1-D visualization of cost function lacks non-convexity even with a high-resolution sampling with small α. This is because, in a 1D visualization, we are essentially plotting the loss function along a line between two points in the parameter space. This line is a very narrow slice of the entire parameter space, and it's quite possible that this slice bypasses many of the non-convex features of the loss function. Therefore, the 1D visualization may give the false impression that the loss function is convex or nearly convex when in fact, it has multiple local minima. Secondly, this method does not account for batch normalization and invariance symmetries, leading to a potentially misleading perception of the sharpness of the 1-D loss landscape. Batch normalization, a technique employed to enhance the speed and stability of neural network training, normalizes each layer's activations to have zero mean and unit variance. This normalization introduces a degree of scale invariance into the network, implying that if the weights in one layer are multiplied by a certain scalar, and the weights in the subsequent layer are divided by the same scalar, the network's behavior remains largely unchanged. This is often not a problem, except when we are visualizing the loss landscape. If this scale invariance is not considered during loss landscape visualization, and we use randomly generated scalars of the same scale to perturb the weights with different scales, the variance of each layer may be inconsistent. This can create an illusion that the loss function is more or less sensitive to weight changes than it truly is. Therefore, the failure of 1-D visualization methods to fully capture the effects of batch normalization and invariance symmetries can lead to an overestimation of the complexity and dimensionality of the loss landscape, potentially resulting in incorrect conclusions about the network's behavior.

In order to overcome the no-detail problem in 1D visualization, a common way is to use a 2D visualization to contain geographic non-convexness. Within the 2D landscape visualization technique, an initial step involves selecting a central point (e.g., optimal parameters), denoted as θ^*. Subsequently, two orthogonal vectors, δ, and η, having identical shapes to the network parameters, are chosen, along with their respective scalar parameters, $\alpha, \beta \in [-1, 1]$.

$$f(\alpha, \beta) = L\left(\theta^* + \alpha\delta + \beta\eta\right) \tag{8}$$

In addressing the second issue, which is the misleading intuition of the sharpness of the 1-D loss landscape. Li et al. [19] introduced a Filter-Wise Normalization approach for 2D landscape visualization.

$$d_{i,j} \leftarrow \frac{d_{i,j}}{\|d_{i,j}\|} \|\theta_{i,j}\| \tag{9}$$

where $d_{i,j}$ and $\theta_{i,j}$ are the j-th filter of the i-th layer in the direction and parameter tensors, respectively. The normalization technique employed here ensures that the direction vectors maintain the same scale as the parameter vectors. This is very important, as it allows for the"step size" of network perturbations to remain consistent with the network size, enabling a fair comparison of network backbones with varying depths.

3.3 Sharpness

Although the loss landscape visualization is an intuitive way to see the optimization problem in MAML, due to the limit of the severe dimension reduction where we only perturb the network in two directions, we can only do qualitative analysis on the plots.

In this context, we are trying to measure the sensitivity of the loss function to perturbations in the weights of the network, i.e., how "sharp" is the loss landscape. The sharpness of a target weight $\boldsymbol{w}$ is defined as the difference in the loss between a perturbed weight $\boldsymbol{w}+\boldsymbol{\epsilon}$ and $\boldsymbol{w}$, subject to the Lp-norm perturbation $||\epsilon||_p$ is smaller than a hyperparameter ρ. Here, ϵ represents the perturbation to the weights, and its value signifies the magnitude of the perturbation.

$$sharpness = \left[\max_{\|\boldsymbol{\epsilon}\|_2 \leq \rho} \mathcal{L}(\boldsymbol{w} + \boldsymbol{\epsilon}) - \mathcal{L}(\boldsymbol{w})\right] \tag{10}$$

The value of ϵ is set to maximize the loss within the constraint of the hyperparameter ρ. Basically, we want to find a small perturbation ϵ^* that can lead to the highest loss with the restriction of ρ. Since $\mathcal{L}(\boldsymbol{w})$ do not have contribution to the problem of finding ϵ^*, we can simplify the problem as:

$$\boldsymbol{\epsilon}^*(\boldsymbol{w}) \triangleq \arg\max_{\|\boldsymbol{\epsilon}\|_p \leq \rho} \mathcal{L}(\boldsymbol{w} + \boldsymbol{\epsilon}) \tag{11}$$

Usually, ρ is a small value, so we can use the first order Taylor expansion to estimate the value of $\mathcal{L}(\boldsymbol{w}+\boldsymbol{\epsilon})$. with the Taylor expansion, $\mathcal{L}(\boldsymbol{w})$ again is not relevant to the optimization problem.

we can rewrite the equation like this:

$$\boldsymbol{\epsilon}^*(\boldsymbol{w}) \triangleq \underset{\|\boldsymbol{\epsilon}\|_p \leq \rho}{\arg\max} \mathcal{L}(\boldsymbol{w}+\boldsymbol{\epsilon}) \tag{12}$$

$$\approx \underset{\|\boldsymbol{\epsilon}\|_p \leq \rho}{\arg\max} \mathcal{L}(\boldsymbol{w}) + \boldsymbol{\epsilon}^T \nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}) \tag{13}$$

$$= \underset{\|\boldsymbol{\epsilon}\|_p \leq \rho}{\arg\max} \boldsymbol{\epsilon}^T \nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}) \tag{14}$$

the value $\boldsymbol{\epsilon}^*(\boldsymbol{w})$ that solves this approximation is given by the solution to a classical dual norm problem(the dual norm problem is explained in detail in Appendix C):

$$\boldsymbol{\epsilon}^*(\boldsymbol{w}) = \rho \operatorname{sign}\left(\nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w})\right) \left|\nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w})\right| / \left(\|\nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w})\|_2^2\right)^{1/2} \tag{15}$$

In this thesis, we set $\rho = 0.05$ as the default value, following the default setting in SAM [9] and SharpMAML [1]. It is worth noting that, at first glance, using a constant range ρ across all depths of the backbone models might seem unreasonable. This is because different networks have different numbers of hyperparameters and may require different scales of constraint. However, this choice becomes justifiable when we consider the right-hand side of the sharpness equation as the norm of the gradient.

$$\boldsymbol{\epsilon}^*(\boldsymbol{w}) = \rho \frac{\nabla \mathcal{L}(\boldsymbol{w}; \mathcal{D})}{\|\nabla \mathcal{L}(\boldsymbol{w}; \mathcal{D})\|_2} \tag{16}$$

As the network becomes deeper and contains more parameters. Although the norm does not evolve in the scale of the individual gradient, it changes according to the number of parameters having gradients. Consequently, despite employing a constant value of ρ, the comparison of sharpness across various backbone architectures remains equitable and robust. This approach allows us to make valid comparisons between different network backbones, despite their varying complexities and parameter counts. Finally, with $\boldsymbol{\epsilon}^*(\boldsymbol{w})$, we can get the sharpness of a trained network weight as:

$$sharpness = [\mathcal{L}(\boldsymbol{w} + \boldsymbol{\epsilon}^*(\boldsymbol{w})) - \mathcal{L}(\boldsymbol{w})] \tag{17}$$

4 Experiment

In this section, we present and analyze the experimental results. First, we introduce the dataset we use and its diverse characteristics. Then, we conduct a comprehensive and in-depth analysis of the accuracy performance of MAML and transfer learning with different backbones. We further investigate the optimization of MAML by visualizing the loss landscape of the inner and outer loops. Finally, we provide the sharpness measurement results and explore their relationship with corresponding loss and accuracy. Through our visualizations and analysis, we address the research questions that we initially proposed in the Introduction.

4.1 Dataset

In this thesis, we use two popular benchmark datasets often used in few-shot learning: MiniImageNet and CUB.

MiniImageNet [27] is a popular dataset in the field of computer vision, particularly in the area of few-shot learning. It is a subset of the ImageNet [5] dataset, which is a large-scale dataset that contains millions of images and is commonly used for training deep neural networks. The MiniImageNet dataset consists of $60,000$ images with a resolution of 84x84 pixels, divided into 100 classes, with 600 images per class. This dataset is designed to fit the "task" scenario in few-shot learning. In a 5-way 1-shot task, the goal is to classify 15 test images (following Finn et al. [8]) into five classes, with only 1 example per class available for training. In a 5-way 5-shot task, the goal is to classify 15 test images into five classes, with five examples per class available for training. The primary challenge presented by this dataset stems from the huge variations among the classes. This requires the classifier to be capable of identifying essential yet diverse characteristics across a broad range of objects. An example is shown in Figure1

Figure 1: MiniImageNet in few-shot learning setting [17]

The Caltech-UCSD Birds-200-2011 (CUB) dataset contains 11,788 images of 200 different bird species, with each species class containing approximately 60 images. The fine-grained nature of the CUB dataset makes it a challenging dataset for few-shot learning algorithms, as the subtle differences among the bird species require models to discern fine-grained visual features. This is particularly difficult when only a small number of examples per class is available. This makes it particularly useful for fine-grained image classification research. Researchers have used

the CUB dataset to evaluate the performance of various few-shot learning approaches, such as meta-learning, memory-augmented neural networks, and data augmentation techniques. The dataset has become a standard benchmark in the field and has contributed significantly to the development of more effective few-shot learning methods. An example is shown in Figure2

Figure 2: CUB dataset [13]

4.2 Comparative Analysis of Few-shot Learning Algorithms

In this section, we first present our reproduced results to confirm that our MAML and transfer learning methods are correctly implemented. Next, we carry out a combined study of both the loss function and accuracy. This helps us better understand the strengths and weaknesses of MAML compared to transfer learning. From our findings, we propose several research questions. These questions are often neglected in earlier studies on MAML but can be addressed through the visualization of the loss landscape. The main code for our implementation comes from the work of Chen et al. [3], as acknowledged in their paper.

4.2.1 Reproduced Results

To ensure a robust foundation for further analysis, it's crucial that our reproduced results align consistently with other implementations. We validate the accuracy of our implementation by comparing it to the results in Chen et al.'s [3] work, thereby confirming its consistency and reliability. In the original implementation conducted by Chen et al. [3], they utilized data augmentation techniques to improve the accuracies reported in their study. During the meta-training phase, this data augmentation involved processes such as random cropping, left-right flipping, and color jittering. To further investigate the impact of data augmentation on these algorithms' performance, we provide two sets of results for each algorithm: one with data augmentation applied and one without. This allows us to assess the relative importance of data augmentation in the context of few-shot learning. To evaluate the impact of network depth on these algorithms, four distinct neural network backbones are employed: Conv-4, Conv-6, ResNet-10, and ResNet-18.

In our implementation, **MAML** is trained with $60,000$ updates on both MiniImageNet and CUB datasets. Each update contained four randomly chosen 5-way 1-shot tasks. The best

initialization was determined based on the accuracy of the validation set. **Transfer learning**, on the other hand, utilizes a traditional supervised learning training method. In this thesis, two discrepancies from Chen et al.'s [3] implementation are notable. Firstly, to fairly compare MAML and Transfer learning, we also implemented $60,000$ updates during the training phase with a common batch size of 128, equivalent to $1,000$ epochs on the CUB dataset and 200 epochs on the MiniImageNet dataset. Secondly, Chen et al. [3] apply a fixed epoch in the pre-training phase across all network backbones, yet this was contrasted with a flexible best epoch of MAML. This comparison does not provide equal conditions for different backbones. To rectify this, we pre-trained our network and selected the best weight by evaluating it on the validation set for every 5 epochs and observed that this change brought about a 5% increase in accuracy for all the backbones with or without data augmentation. Detailed settings for other hyperparameters of MAML and Transfer learning are consistent with Chen et al.'s [3] implementation and can be found in Appendix A.

The results of our experiment are displayed in Table 2, which provides a thorough comparison by outlining the average accuracy and standard deviation for each algorithm and backbone, both with and without data augmentation during the training process. The reproduced results obtained in our experiments have similar performance to those from Chen et al.'s [3] implementations. Note that each experiment is trained 3 times with 3 random seeds {5,6,7}, and the reported accuracy is the average accuracy of the three experiments in $1,000$ tasks sampled from the meta-test set.

The results demonstrate a significant performance difference concerning the backbone architecture used. Generally, more intricate structures like ResNet10 and ResNet18 outperform simpler ones, such as Conv4 and Conv6, across both MAML and transfer learning techniques. This suggests that deeper architectures can extract more nuanced and useful features, leading to superior performance. Moreover, the application of data augmentation notably enhances performance across almost all scenarios. This highlights the effectiveness of data augmentation in strengthening model generalization. When comparing performances, it becomes apparent that MAML significantly outperforms transfer learning across all backbone architectures when tested on the CUB dataset. While the performance gap narrows on the MiniImageNet dataset, MAML only maintains a relatively close lead.

4.2.2 Loss-Accuracy Analysis

When assessing few-shot learning algorithms, relying solely on average accuracy may yield an incomplete representation of the model's performance. The loss function provides a more nuanced perspective, revealing the magnitude of prediction errors that might be neglected if accuracy is the only metric used. Evaluating both aspects allows for a more comprehensive understanding of the model's strengths and weaknesses.

MiniImageNet						
	MAML			**Transfer learning**		
	no aug	aug	reported(aug)	no aug	data aug	reported(aug)
Conv4	45.62 ± 0.79	45.29 ± 0.79	46.47 ± 0.82	36.46 ± 0.58	45.83 ± 0.69	42.11 ± 0.71
Conv6	47.23 ± 0.82	51.31 ± 0.87	50.96 ± 0.92	41.66 ± 0.63	50.13 ± 0.74	45.82 ± 0.74
ResNet10	49.38 ± 0.83	54.17 ± 0.87	54.69 ± 0.89	51.30 ± 0.63	54.76 ± 0.75	52.37 ± 0.79
ResNet18	48.78 ± 0.92	51.97 ± 0.95	49.61 ± 0.92	48.56 ± 0.78	55.26 ± 0.77	51.75 ± 0.80
CUB						
	MAML			**Transfer learning**		
	no aug	aug	reported(aug)	no aug	aug	reported(aug)
Conv4	53.24 ± 0.51	56.36 ± 0.55	54.73 ± 0.97	35.83 ± 0.17	46.69 ± 0.47	47.12 ± 0.74
Conv6	56.40 ± 0.59	66.51 ± 0.60	66.26 ± 1.05	40.13 ± 0.23	53.16 ± 0.39	55.77 ± 0.86
ResNet10	56.26 ± 0.61	69.80 ± 0.56	70.32 ± 0.99	58.86 ± 0.47	60.6 ± 0.43	63.34 ± 0.91
ResNet18	55.50 ± 0.57	70.42 ± 0.64	68.42 ± 1.07	59.90 ± 0.47	63.79 ± 0.45	65.51 ± 0.87

Table 2: Performance comparison table showing the mean accuracy along with standard deviation (mean accuracy $\pm$ 95% confidence intervals) of Meta-learning with MAML and transfer learning methods applied on the MiniImageNet and CUB datasets. Both methods are compared with and without the use of data augmentation techniques. Various neural network architectures, namely Conv4, Conv6, ResNet10, and ResNet18, are utilized to provide a detailed view of how the performance varies across different model architectures and learning algorithms. We also include the reported results of these methods with data augmentation from the original paper for a fair and comprehensive comparison.

As a result, we introduce a scatter plot of loss versus accuracy in Figure 3. By revealing the different patterns of transfer learning and MAML in these plots and also highlighting the inconsistency between accuracy and loss, we can draw conclusions and identify research questions that require further exploration through the visualization of the loss landscape. Each point on the plot represents the average query set loss (x-axis) and average accuracy (y-axis) over $1,000$ test tasks. The results from three distinct random seeds are plotted separately for clarity.

Transfer learning and MAML. Through the scatter plot across two datasets, we identified consistent patterns and critical insights on the performance of different backbone architectures, the impact of data augmentation, and the difference between MAML and transfer learning.

In terms of backbone architecture, ResNet10, and ResNet18 generally surpass Conv6 and Conv4 models in performance. This pattern is consistent across both datasets and learning methods, with ResNet18 slightly outperforming the rest in most scenarios. However, an interesting anomaly we noticed is that the additional depth of ResNet18 does not always result in superior

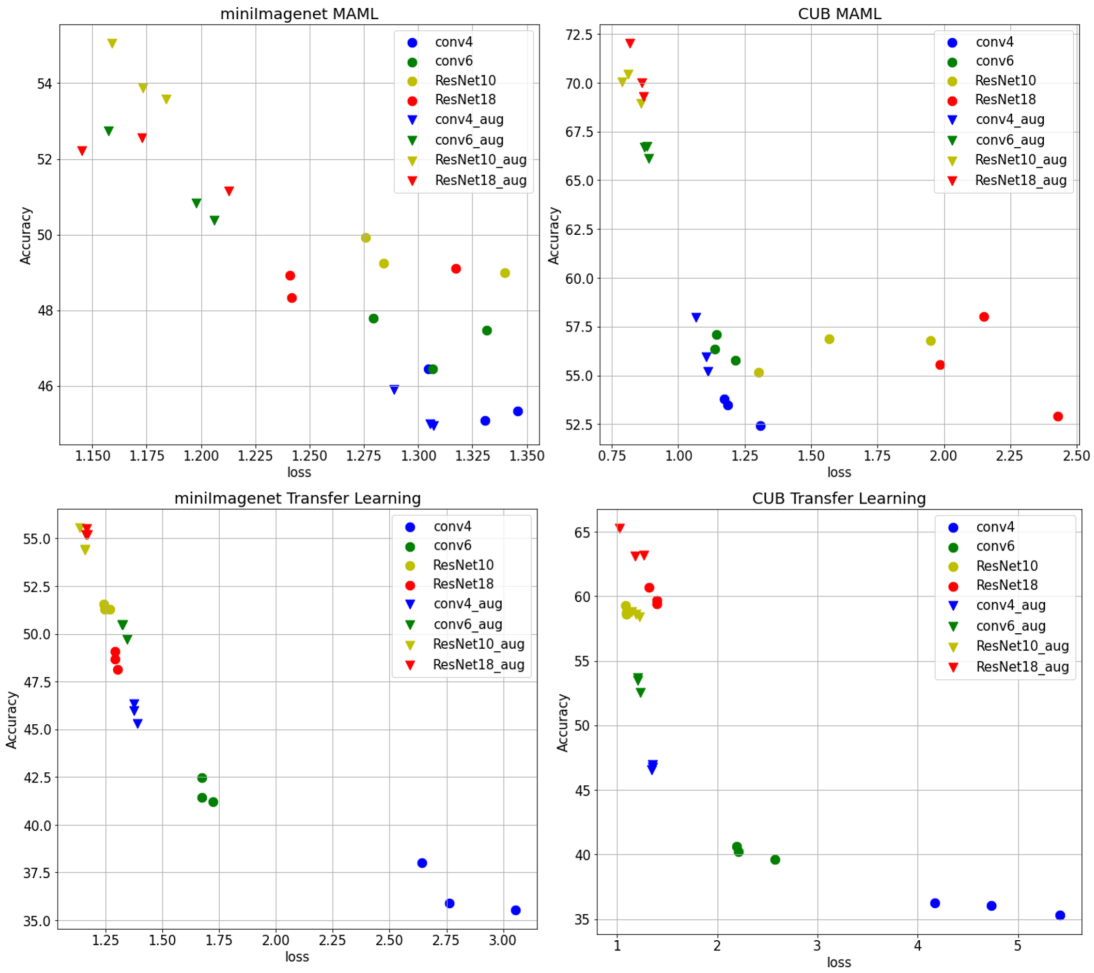


Figure 3: Scatter plot showing the results of two learning methods, MAML and Transfer learning. Each point on the plot shows the average loss and accuracy from 1,000 tests. Different colors and shapes show different types of experiments. We used three different random seeds, and are shown separately.

performance compared to ResNet10. This raises questions about the efficiency of deeper architectures in certain tasks and warrants further investigation.

Data augmentation emerges as a significant factor in our analysis, consistently improving model performance across various metrics, architectures, and datasets. The benefit of data augmentation is most noticeable in terms of accuracy and loss improvement across all models. Notably, while Conv4 also gains from data augmentation, the extent of improvement is marginal, suggesting capacity limitations in fully leveraging the augmented data. In contrast, ResNet10, ResNet18, and Conv6 show significant improvements from data augmentation. These models effectively utilize the augmented data to enhance their learning capabilities, reflected in their lower loss values and higher accuracy rates.

When evaluating the performance of MAML and transfer learning, an intriguing pattern becomes apparent. MAML tends to outperform transfer learning when utilizing simpler backbones, such as Conv4 and Conv6, particularly in the absence of data augmentation. This observation may highlight MAML's capacity to exploit simpler architectures effectively. This finding is also supported by the studies of Huisman et al. [16] and Chen et al. [3]

Interestingly, the relative effectiveness of MAML and transfer learning also seems to be contingent on the dataset in use. For instance, in the CUB dataset, MAML consistently outperforms transfer learning, irrespective of the backbone architecture employed. However, when it comes to the MiniImageNet dataset, the performance competition appears more balanced. Transfer learning marginally outperforms MAML in ResNet architectures, while MAML proves more effective in Conv architectures.

Notably, some abnormal patterns are evident in the results. In the MiniImageNet dataset, both MAML and transfer learning interestingly demonstrate that ResNet18 underperforms ResNet10. As this pattern was not observed in the CUB dataset, it may be attributable to a dataset-specific characteristic. MiniImageNet, as a subset of a much larger dataset, might not present the complexity that requires a deeper model like ResNet18. It is possible that ResNet10 is already sufficient to capture the dataset's complexity, and the additional capacity of ResNet18 does not translate into better performance.

When no data augmentation is applied to the MiniImageNet dataset, all backbones report a similar loss. However, ResNets deliver higher accuracy, indicating that loss is not the sole determinant of accuracy. In contrast, for the CUB dataset, the loss values of Conv4 and Conv6 as equal and lower than ResNet10 and ResNet18. This may suggest a pattern of overfitting in ResNets. Even though the models were evaluated at the epoch of best validation performance during training, an overly complex backbone relative to the dataset can still fit the training data exceptionally well. We further provide proof that the overfitting is not because of too many training epochs in Appendix F. This"over-complex" could result in the model capturing not only the underlying patterns but also the noise and outliers.

Our analysis has led to several key findings that can steer future research:

- Data augmentation consistently boosts model performance across various backbone architectures, datasets, and learning methods. This raises a pivotal question: *How does data augmentation improve model performance?*
- Without data augmentation, all backbones in the MiniImageNet dataset register similar losses, but ResNets display higher accuracy. On the CUB dataset, all backbones attain similar accuracy, but ConvNets demonstrate lower loss. This phenomenon prompts the

question: *What causes this discrepancy? What factors influence accuracy other than loss?*

The answers to these questions hold the potential to significantly advance our understanding and utilization of MAML and data augmentation techniques.

4.3 Visualization

Visualizing the loss landscape provides significant insight into the optimization problem inherent in MAML. MAML's optimization problem can be peeled layer by layer. The outermost layer corresponds to the query loss of the test tasks in MAML. This layer represents the model's output and is a critical evaluation of model performance. The next layer of analysis involves the training process on the support set of a task. This focuses on how the model learns from limited data, which represents the ability to adapt and predict unseen tasks effectively. At the core of MAML lies the initialization weight. This is designed to rapidly adapt to new tasks, and it is determined by optimizing the training loss. Interestingly, the training loss within MAML is not determined in isolation. Instead, it's influenced by both the query and support loss of the meta-training tasks, which creates a fascinating cycle.

In this section, we divide the visualization into two parts: Inner loop and Outer loop. In the inner loop (Section 4.3.1), we randomly choose different tasks from the training and test set and visualize the support loss landscape. In the outer loop (Section 4.3.2), we offer a visualization of the outer loop loss landscape along with a comprehensive discussion. It's vital to note that due to the computational power limit, our analysis only focuses on the 5-way 1-shot setting, and the weights used were the best weights trained using a single random seed 5.

4.3.1 Inner Loop

The concept of "Task" is the cornerstone of the few-shot learning setting. For each task, MAML initially updates the parameters on the support set and, subsequently, assesses them on the query set. During MAML's training process, the meta-gradient comes from the query loss after inner loop optimization. The learned initialization is subsequently utilized on test tasks through the same inner loop optimization process. Therefore, understanding inner loop optimization is crucial.

This led us to our primary question of this section: **what does the optimization problem look like for a single task?** In the original MAML paper [8], the inner loop optimization process in a 5-way 1-shot setting contains training the initial weight on five images from five different categories, undertaking five updates during training and ten updates during testing. Following the same setting, we have depicted the loss landscape of inner loop optimization on both the CUB and miniImageNet datasets and on the meta-training and meta-test sets. The results are shown in Figure 4 and Figure 16. Each point on the surface represents the training loss on the support set (z-axis) at the current update, with the weight disturbed in both the x

and y directions. For each loss landscape, the center point $(x = 0, y = 0)$ represents the weight the algorithm lands on without our perturbations. The displayed visualizations are created based on single 5-way 1-shot tasks. This task is randomly chosen from either the meta-training or meta-test set of the miniImagenet or CUB datasets. Due to the randomness of the tasks, this selection is not cherry-picked. To avoid any bias associated with specific tasks, we have included a second set of visualizations from another randomly sampled task in Appendix B. The conclusions we draw in this section remain valid for both tasks. We also did not notice any significant differences in the visualizations that could lead us to draw any general conclusions when comparing the scenarios of using data augmentation versus not using it. Considering that data augmentation is often, by default, used in few-shot learning papers, all our visualized landscapes employ weights trained with data augmentation. We provide visualizations without data augmentation in Appendix B for comparison. The effect of data augmentation will be later discussed in Section 4.3.2 using the outer loop visualizations.
In these figures, instead of only visualizing the loss landscape around the optimum weight after training on the support set, we additionally include the 0 update loss landscapes. This means that there are no updates have been performed yet. This serves as a starting point for understanding the optimization process and how the landscape changes as the network updates take place. The rows below them illustrate the loss landscape after the network has completed five or ten updates on the support set of the task. This provides an intuition about the loss landscape after completing all the updates, where the network should have adapted to the new task. By comparing the initial and final loss landscapes, we can observe the changes in the optimization process and evaluate how well the network has learned to adapt to the given task.

The pattern we observed in both figures indicates that, at 0 updates, the initialization weight of Conv6 and ResNets locates on a steep slope area, suggesting a clear and fast optimization path toward the minimum. Furthermore, all the backbones have converged to a local minimum within the target update numbers, which, as expected, confirms that MAML rapidly adapts to the new task. In contrast, the initialization of the Conv4 backbone tends to be located in a locally flat area. We speculate this might be because Conv4 is a simpler model compared to Conv6, ResNet10, and ResNet18. The training process of MAML aims to find a minimum loss for all tasks. Although this minimum might not be the optimal solution for every task, given the model's ability, it may be challenging for Conv4 to find a deeper loss and thus remain stuck in the same area. If we examine the loss variance in the color bar near the visualizations, we can see that Conv4 has the smallest difference before and after the updates, further validating our hypothesis. This could also explain why Conv4 exhibits the poorest performance among the four models.

Interestingly, we noticed that in the miniImageNet dataset, the loss landscapes of ResNet10 and ResNet18 at 0 updates exhibit a tilted plane pattern, suggesting that the training loss is likely to be much more sensitive to a certain direction, either one of δ and η or a linear com-

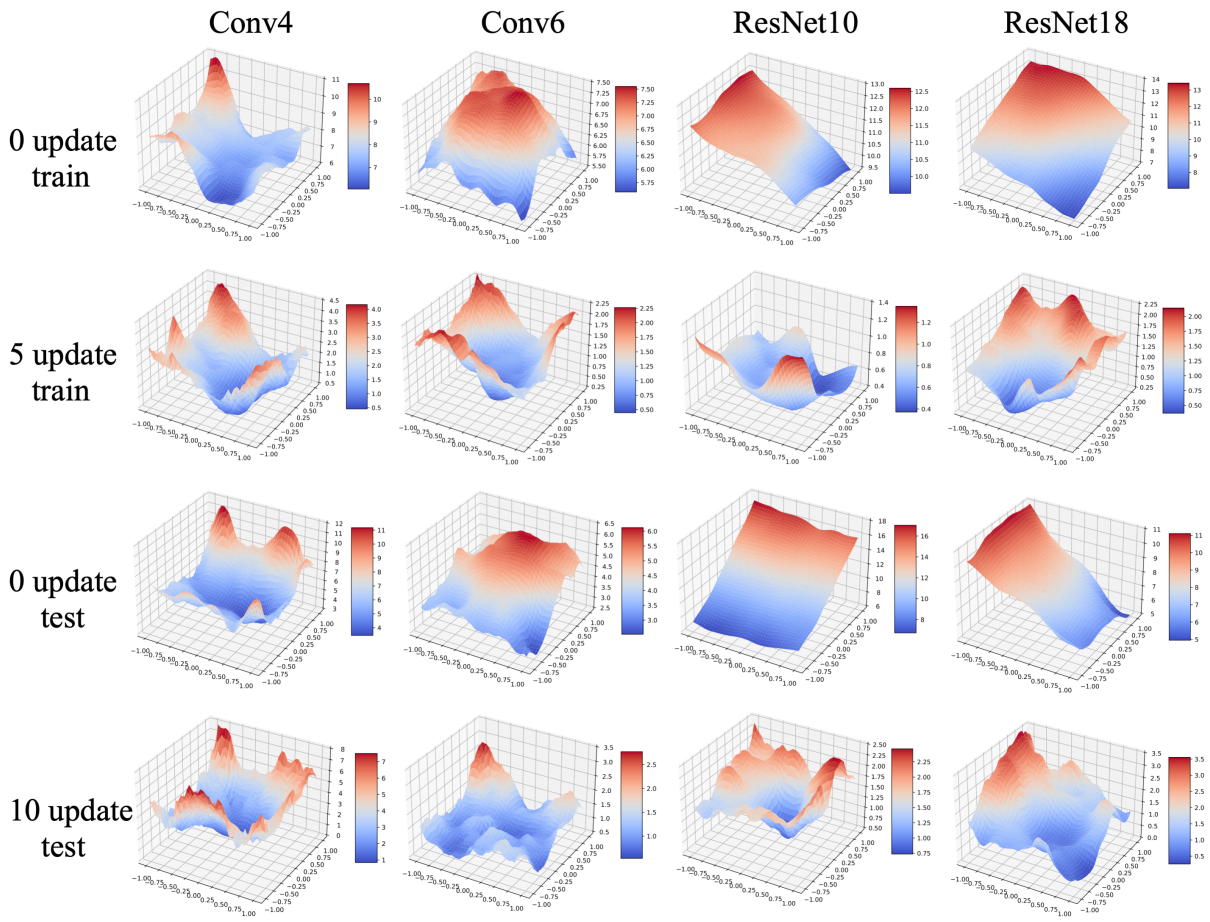


Figure 4: Loss landscapes of various network architectures on **MiniImagenet** dataset before and after inner loop optimization of meta-training or meta-testing.
0 update means no updates have been performed on the trained initialization weight. 5 and 10 updates are the update times of meta training and meta-testing phase. The center point $(x = 0, y = 0)$ of each plot corresponds to the weight the algorithm lands on without our perturbations.

bination of δ and η. This observation may initially appear peculiar for non-linear models like neural networks. However, the primary factor that can result in a tilted plane loss surface is a linear model, which corresponds to the 1-layer network head of these backbones.

We conducted further experiments to investigate the relationship between the network's classifier head and the feature-learning architectures for different backbones. We decomposed the loss landscape into "feature" loss landscapes and "head" loss landscapes. In "feature" loss landscapes, the perturbation weight of the head layer in both directions δ or η is set to zero, while in "head" loss landscapes, all perturbation weights are set to zero except the head layer. Note that although we set the perturbation weight to zero, we do not actually freeze the weight during training. Both the head and the feature layers are trained together in an original way. To better compare the influence of the head and the feature layers, we make the scale of their loss

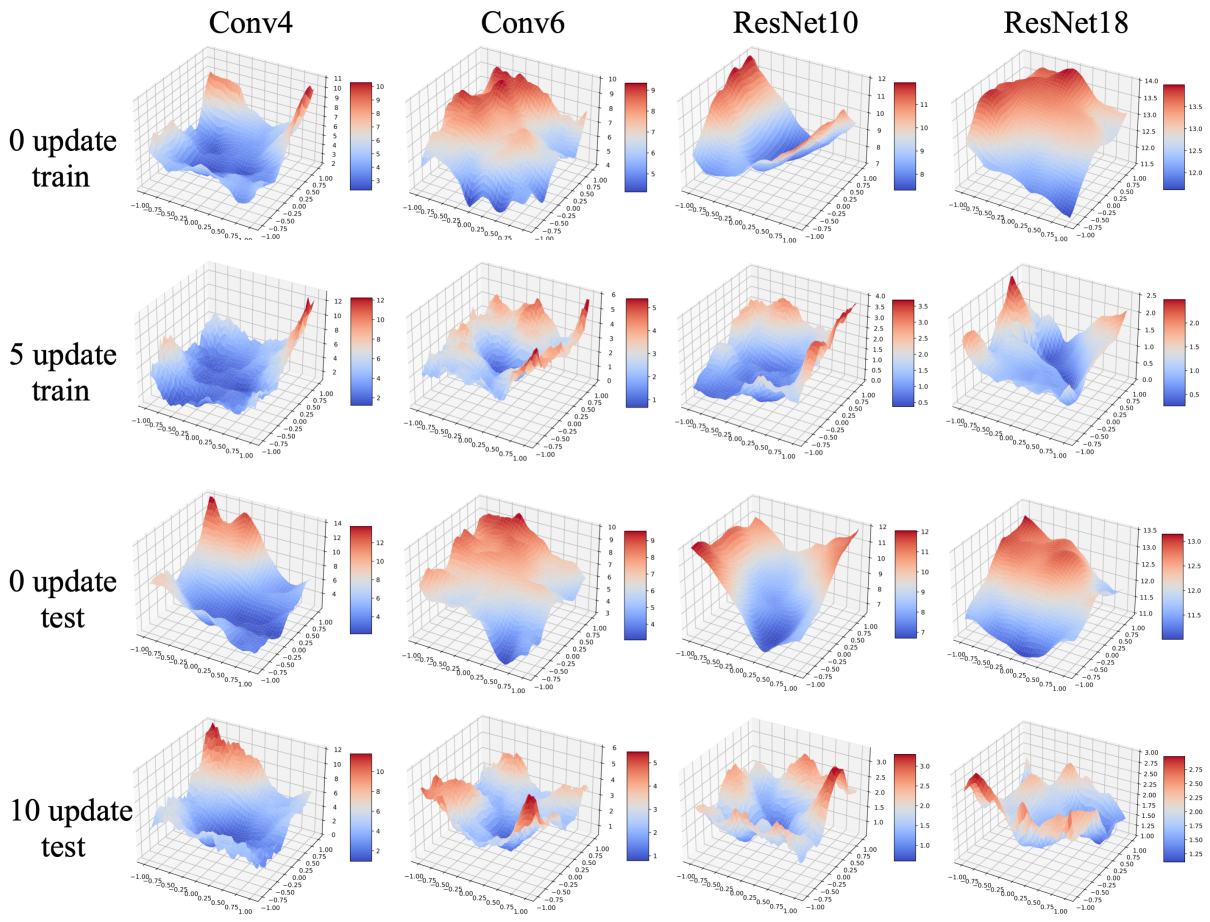


Figure 5: Loss landscapes of various network architectures on **CUB** dataset before and after inner loop optimization. 0 update means no updates have been performed on the trained initialization weight. 5 and 10 updates are the update times of meta training and meta-testing phase. The center point $(x = 0, y = 0)$ of each plot corresponds to the weight the algorithm lands on without our perturbations.

landscape consistent with the original loss landscape. As we did not see an obvious difference between the meta-train and meta-test tasks, we used the meta-test tasks as default for the rest of this section. We show the decomposed loss landscape of MiniImagenet and CUB in Figure 17 and Figure18.

From the figures, we find that all the backbones reveal a strong correlation between head loss and the total loss landscape, and most of the non-convexity arises from the feature layers, though they only slightly influence the loss value. This holds true for both datasets. This observation suggests that the network heads significantly shape the total loss landscape. They also imply that during MAML's adaptation to a new task, it is more crucial to identify an appropriate weight for the head classifier than to adjust the weights of the feature layers. This finding aligns with Raghu et al.'s work [21], where the authors noted that the similarity of feature layers approaches 1 after inner loop training, and the greatest variance exists in the network head.

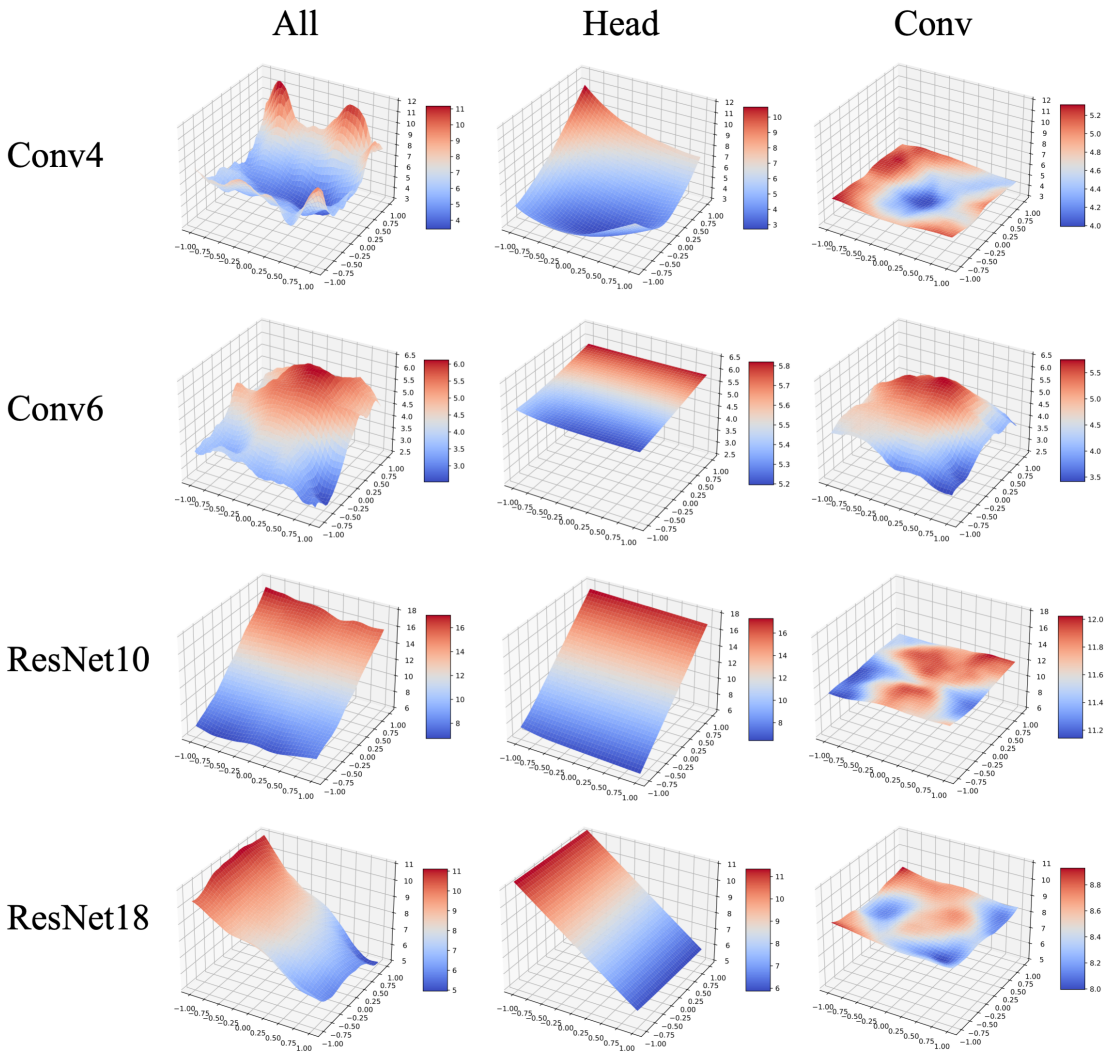


Figure 6: The decomposition of the Loss landscapes of different backbones on **MiniImageNet** at the 0 step of inner loop optimization. The left column is the loss landscape of perturbing all the network weights, The middle column is the loss landscape of perturbing only the head of the network, and the right column is the loss landscape of perturbing the feature(Conv) layers. Note that We only fix the perturbation of the loss landscape. Both the network head and feature layers are trained together during the updates.

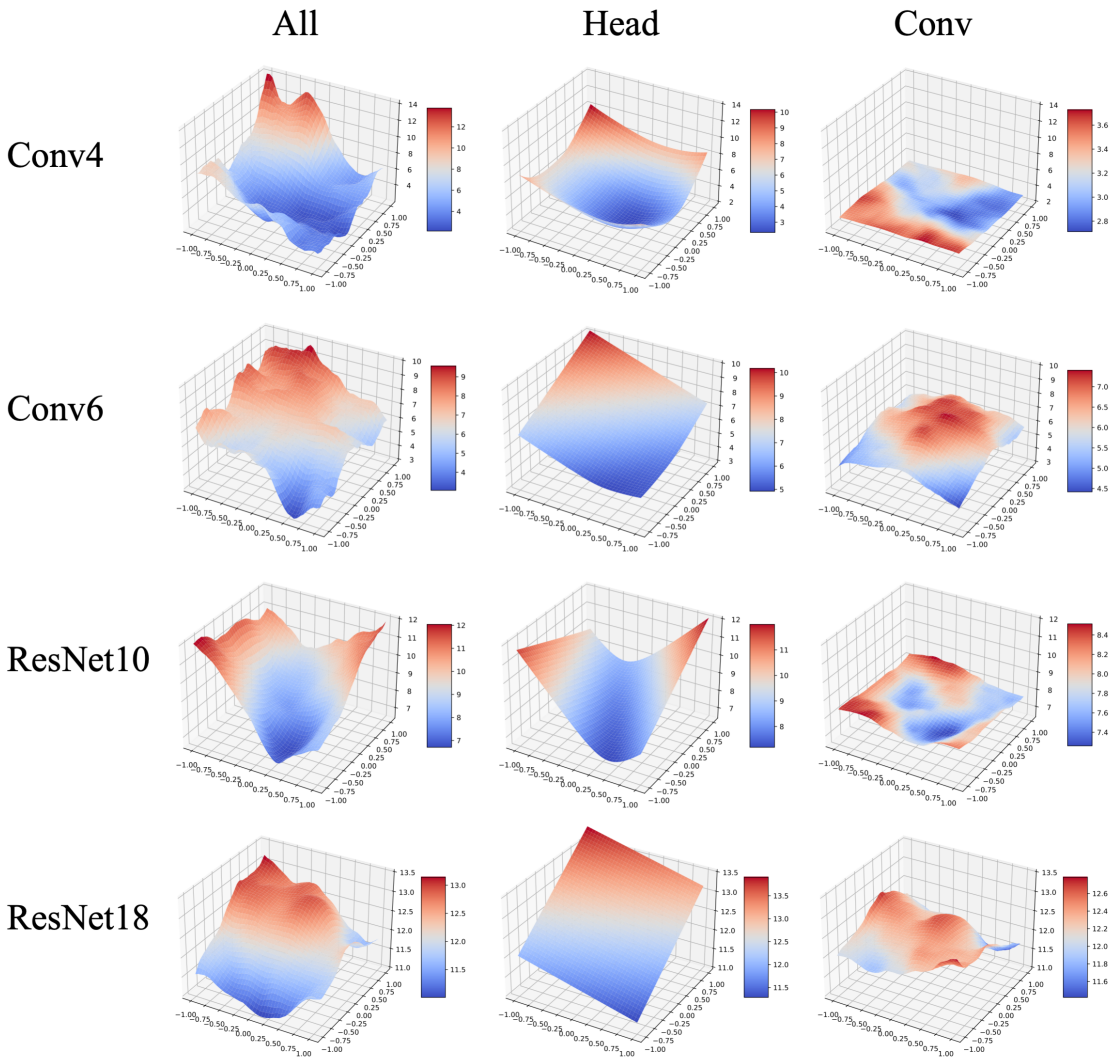


Figure 7: The decomposition of the Loss landscapes of different backbones on **CUB** at the 0 step of inner loop optimization. The left column is the loss landscape of perturbing all the network weights, The middle column is the loss landscape of perturbing only the head of the network, and the right column is the loss landscape of perturbing the feature(Conv) layers. Note that we only fix the perturbation of the loss landscape. Both the network head and feature layers are trained together during the updates.

Combining these insights, it is plausible to claim that MAML, instead of rapidly adapting all layers to a new task, reuses features learned during training and adjusts the head layer when facing a new task.

4.3.2 Outer Loop

Section 4.3.1 delves into the inner loop optimization of MAML. **This section presents a visualization of MAML's meta-loss landscape.** Consistent with the original experimental setup, the meta-loss is computed by averaging the 15-shot query loss across 4 randomly selected tasks. Note that at each point in the plot, the tasks are randomly chosen to simulate the actual optimization process. The results can be seen in Figure 8 (4 task train/test). A common feature across all the loss landscapes is the presence of small spikes, which can be attributed to the randomness of task sampling. When only four tasks are sampled for each update, the optimization process is influenced by the noisy gradients generated by the random task selection. This also explains the high variance in the performance of different random seeds. To further highlight the noise within the optimization process, we've visualized the training loss landscape of transfer learning on the MiniImagenet dataset, as shown in Figure 9. With traditional supervised learning, we observe a smooth, convex loss landscape, which tends to be more straightforward to converge and train. However, it's important to note that we can only make qualitative comparisons between the two loss landscapes, thereby highlighting the complexity involved in the noisy optimization of MAML. A detailed comparison is not justified as the loss landscapes of MAML and transfer learning are evaluated on different bases—the former is assessed on few-shot learning tasks, while the latter is not.

To obtain a less noisy loss landscape to focus on the shape, we average the query loss over 100 tasks at each point instead of sampling only 4 tasks. The results are displayed in Figure 8. The loss landscape retains its shape but appears significantly smoother, reconfirming that the noisy gradient originates from the randomness of task sampling.

Another pattern common to all backbones is the strictly convex shape. MAML is known for its difficulty in optimization due to its two-level optimization structure. We have already demonstrated that the inner loop optimization exhibits apparent non-convexity, making the convex loss landscape counterintuitive. However, the query loss in the query set is calculated using 15 images per class, which means for 5 classes, 75 images in total are used to evaluate the query loss. So compared with the inner loop optimization, which has only 5 images, the non-convexity will offset each other when we are evaluating 75 images. To corroborate this, we plot the loss surfaces with increasing numbers of query shots: 1, 2, 4, and 15 shots. The results are displayed in Figure 10. To better observe the differences in the number of query shots, we present the 2D contour of the surface with a contour interval of 0.2, where each contour line represents a constant meta-loss value rather than using the 3D loss surface.

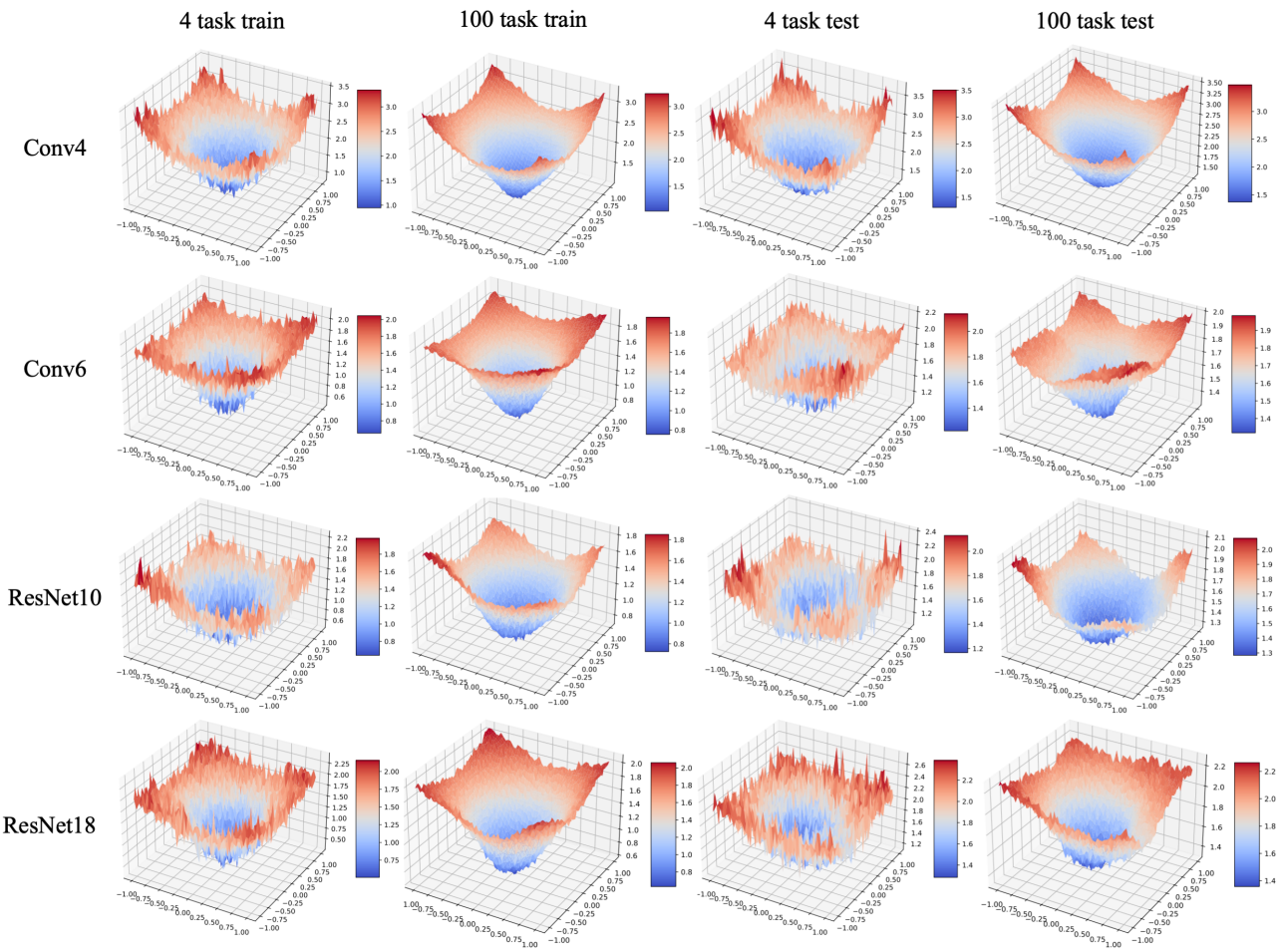
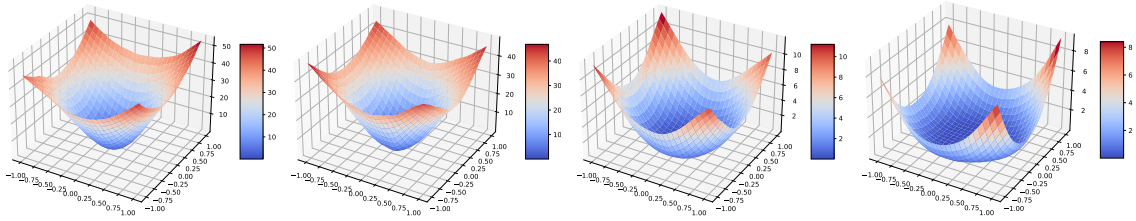


Figure 8: Visualization of meta-loss landscapes for various neural network architectures on the meta train and test sets. The meta-loss is computed by averaging the 15-shot query loss across different numbers(4/100) of randomly selected tasks.

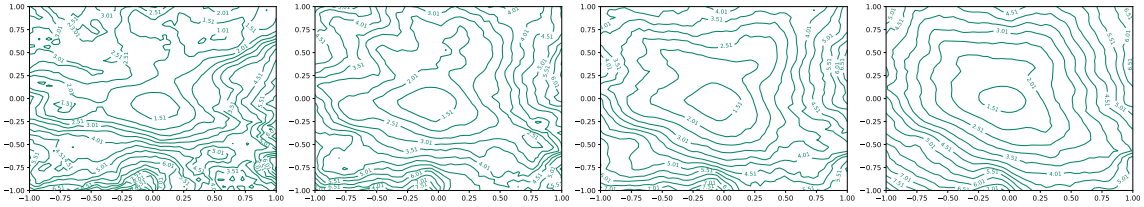
When there is only one query shot, the loss landscape exhibits clear non-convexity. This is because the query set in the 1-shot setting is similar to the inner loop support set, and the non-convexity of the 1-shot inner loop optimization problem has already been discussed in Section 4.3.1. As seen in Figure 10, the convexity becomes more pronounced as the number of query shots increases from left to right. Furthermore, recall that MAML aims to find an initialization that, after training with support sets, minimizes the average 15-shot query loss across all tasks. So the convexity is enhanced even more by averaging multiple tasks with 15 queries each, as shown in Figure 8.

After examining the common features shared by all the backbones, we now compare the differences between them. The sharpness of the minimum is a crucial aspect when evaluating the loss landscape. In fact, Li et al. [19] demonstrated that by using filter-normalized direction vectors,



(a) Conv4 (b) Conv6 (c) ResNet10 (d) ResNet18

Figure 9: The training loss landscape of Transfer learning on MiniImagenet dataset.The training process follows a traditional supervised learning method. Each point on the plot represents the training loss evaluated on $1,000$ images. The center point represents the optimum weight throughout the training.



(a) 1shot (b) 2shot (c) 4shot (d) 15shot

Figure 10: Comparison of meta-loss landscapes for varying numbers of query shots (1, 2, 4, and 15 shots) in the query set of MiniImagenet using Conv4 backbone. The 2D contour representation with a contour interval of 0.2 is used to better visualize the differences in the number of query shots, with each contour line representing a constant meta-loss value.

we could compare the sharpness of loss landscapes for neural networks with different sizes. They stated that as the sharpness increases, generalization ability decreases, resulting in higher test errors.

To get a clear image of the sharpness of the meta-loss landscape, we provide a colored contour version of Figure 8. The colored figure is presented in Figure 11. We colored the minima of the loss landscape and use different colors to get an intuitive way to see the sharpness of the loss landscape. In the two left columns (meta-training set), the green area represents the area that has meta loss ≤ 0.81, and the yellow area represents the area with meta loss ≤ 0.61. In the two right columns (meta-test set), the blue area represents the area with meta loss ≤ 1.41. The red area represents the area with meta loss ≤ 1.21.

Without data augmentation, it is clear that as the network depth increases, the loss areas in the training set become larger, indicating that the network finds a flatter optimum. The size of the area in the test set correlates well with MAML's performance as reported in Table 2, with ResNet 10 having the best performance. It is noteworthy that ResNet10 is the only backbone

that finds a smaller loss in the training set. Conversely, Conv4 fails to reach the loss level of other backbones and has the smallest area of loss in the test set. This reaffirms that its capacity might be inadequate for the MiniImageNet dataset.

When data augmentation is introduced, the Conv4 backbone cannot find a lower loss on the meta-training set, and the minima even become sharper, which again indicates it cannot make full use of the data augmentation because of its low model capacity. Data augmentation acts like a regularizer in Conv6 and ResNet18. It reduces the size of the minimum loss area in the meta-training set but significantly improves the performance in the meta-test set, where it finds a larger loss area and deeper loss. ResNet10 benefits the most from data augmentation. In both meta-training and meta-test sets, ResNet10 finds a larger loss area and a deeper loss. This is also reflected in the accuracy results. It further reinforces the idea that ResNet10 is the most capable of capturing the complexity of the MiniImageNet dataset.

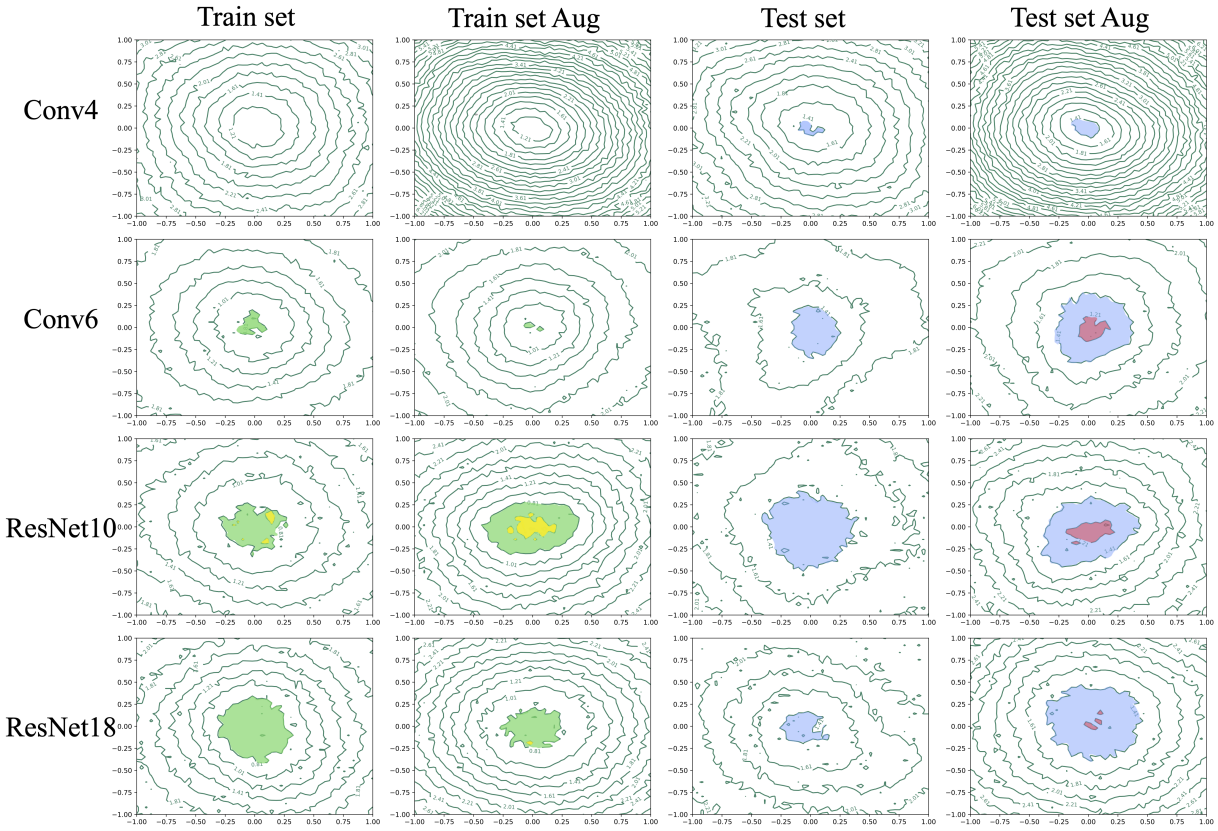


Figure 11: Colored visualization of meta-loss landscapes on the MiniImageNet meta-training and meta-test sets. The minima of the loss landscape are represented in different colors to intuitively depict the sharpness of the landscape. On the meta-training set (left two columns), yellow areas have meta-loss ≤ 0.61, and green areas have meta-loss ≤ 0.81. On the meta-test set (right two columns), red areas indicate meta-loss ≤ 1.21 and blue areas indicate meta-loss ≤ 1.41

Turning to the CUB dataset, it presents a significantly more complex meta-loss landscape, as

illustrated in Figure 12. In the absence of data augmentation, the loss found by Conv Nets is relatively uniform, with Conv6 showing a flatter minimum. In contrast, the ResNets locate a much lower loss and a flatter minimum. Observing the test set, Conv4 and ResNet10 locate a similar minimum loss, with ResNet10 appearing much flatter. Conv6 locates the lowest loss, which also translates into the highest accuracy, as detailed in Table 2. ResNet18 identifies the flattest minima, but this corresponds to the highest loss. This aligns with our preceding analysis, reinforcing the theory of overfitting in ResNets as inferred from the comparative analysis.

The introduction of data augmentation regulates the flatness across all the networks, validating the efficacy of data augmentation. Simultaneously, all backbones except Conv4 locate a lower loss in the test set, with an increased flatness corresponding to the depth of the network. Given the limited network representational ability of Conv4, it fails to effectively utilize data augmentation.

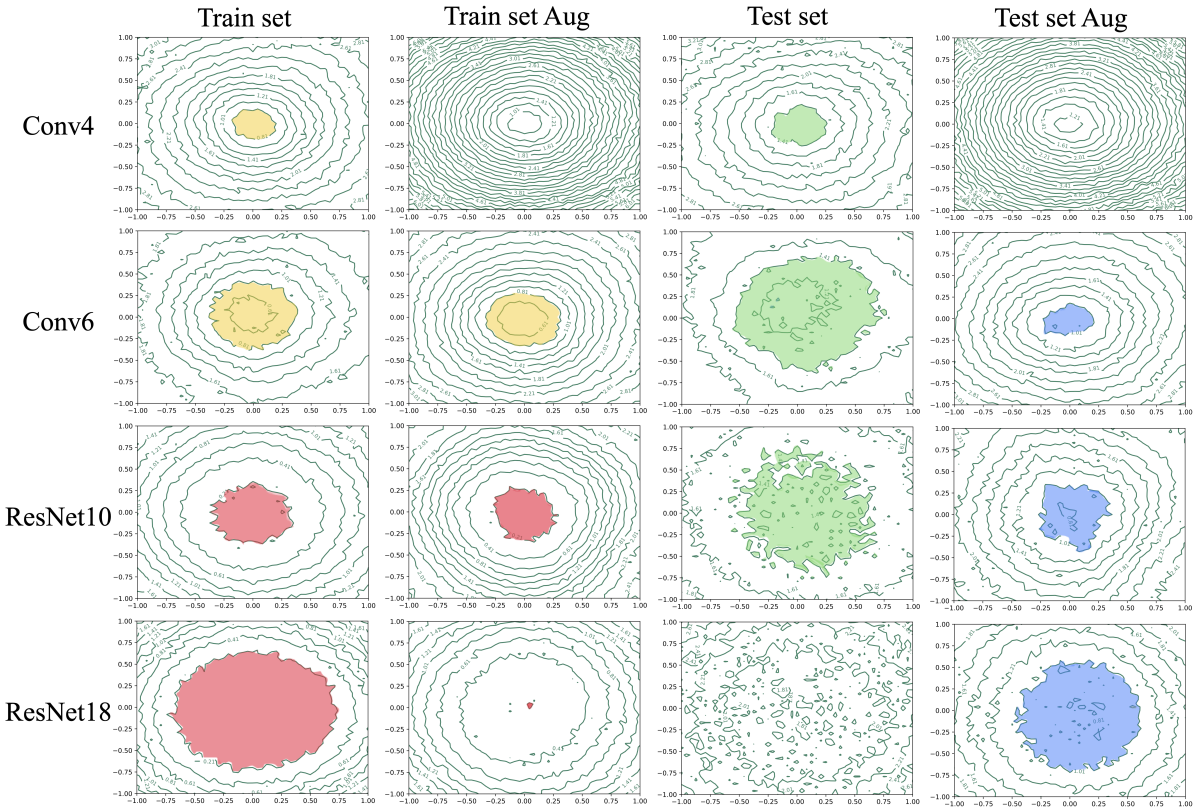


Figure 12: Colored visualization of meta-loss landscapes on the CUB meta-training and meta-test sets. The minima of the loss landscape are represented in different colors to intuitively depict the sharpness of the landscape. Red areas have meta-loss ≤ 0.21, yellow areas indicate meta-loss ≤ 0.81, blue areas indicate meta-loss ≤ 1.01, green areas indicate meta-loss ≤ 1.41.

4.4 Quantitative Assessment of Loss Landscape Visualizations

In this section, we aim to overcome the limitations of qualitative analysis of the MAML loss landscapes by employing a robust, quantifiable measurement. This will allow us to compare the minima between various backbones and datasets more effectively. The Results are shown in Figure 13.

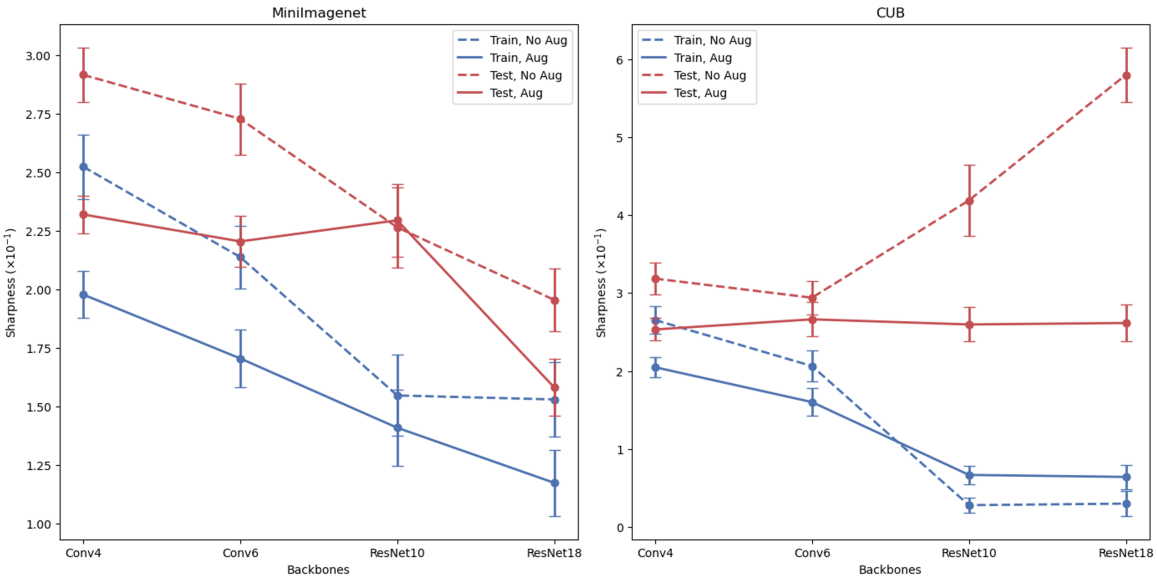


Figure 13: Sharpness trends across different network backbones (Conv4, Conv6, ResNet10, ResNet18) for both MiniImageNet and CUB datasets. The x-axis represents the network backbone, and the y-axis indicates sharpness. Four lines represent different conditions: No Augmentation (No Aug), and Augmentation (Aug) for both meta-train and meta-test classes in the training and testing phases. The trends show the impact of augmentation on the sharpness of the meta-loss landscape.

From the visualizations, we can qualitatively analyze the different minima between various backbones and datasets. However, it is challenging to draw any solid conclusions due to two biases. First, due to the severe dimension reduction, the real loss landscape remains obscured. Second, the visualizations are based on a single random seed, which reduces their reliability. **In this section, the goal is to use a robust quantifiable measurement to compare the MAML loss landscapes.** Following Equation 15, we can quantify the sharpness observed in the meta-loss landscape. However, while it is possible to calculate the average loss across 100 tasks, calculating the gradient for this number of tasks is computationally impossible(a detailed explanation is provided in Appendix D). So instead of calculating the gradient for 100 tasks, we calculate the sharpness for each task and average it over 150 tasks and 3 random seeds. This approach brings an unexpected advantage in that accuracy is calculated one task at a time and then averaged. Consequently, the way we average the sharpness provides a new perspective for

examining the correlation between the loss of landscape sharpness and the network's performance. Figure13 shows the trend of the sharpness with their variances. The exact sharpness is shown in Appendix E.

Starting with the MiniImageNet dataset. It is evident that the employment of data augmentation mostly results in diminished sharpness, suggesting that the model gains robustness in the presence of data augmentation. The exception to this general trend is shown in ResNet10 in the meta-test set, where sharpness values are comparable even if the application of data augmentation. Taking into account that the accuracy of ResNet10 increase by 5% when data augmentation is used, it is likely that the employment of data augmentation allows ResNet10 to discover a smaller loss landscape, consequently leading to an increased sharpness. Hence, the improved model accuracy achieved through data augmentation may be accompanied by an increase in sharpness, indicating a trade-off between the model's robustness and accuracy. Another trend can be observed from Conv4 to ResNet18 - the sharpness value decreases overall, suggesting that deeper architectures like ResNet10 and ResNet18 tend to yield a more robust model in terms of sharpness. This could be due to the additional layers in these architectures, which might be able to capture more complex features, thus resulting in a more robust model.

Looking at the results from the CUB dataset, the most obvious pattern is the difference between using and not using data augmentation. In the absence of data augmentation, the sharpness of Conv Nets remains relatively consistent between the meta-training and meta-test sets. However, for ResNets, there is a striking decrease in sharpness in the meta-training set, accompanied by a considerable increase in the meta-test set. This pattern suggests that while ResNets locate a highly generalizable minimum for the training data, they simultaneously find a less adaptable minimum for the test data. This once more implies that the ResNet is over-parameterized in relation to the CUB dataset. It can adapt closely to the training data.

Introducing data augmentation, For ConvNets, there's a decline in sharpness, akin to what is observed in MiniImageNet, implying that it helps in generalization. Conversely, in the case of ResNets, data augmentation serves as a regularizer. It constrains the model from overfitting the training data, thereby maintaining a closer value sharpness between the training and testing data.

As stated earlier, sharpness is not the sole determinant impacting the performance of the models. It would be oversimplified to draw a direct correlation between sharpness and performance without considering other contributing factors. To provide a more comprehensive view, we further investigate the interrelationship between sharpness, loss, and accuracy for different backbone architectures with and without the application of data augmentation. This relationship is illustrated in Figure 14. In general terms, it is observable that accuracy tends to be higher when both the loss and sharpness are lower. This pattern suggests an interplay between these three

model characteristics: sharpness, loss, and accuracy. Lower loss and sharpness typically indicate a more robust and well-performing model, as demonstrated by higher accuracy in these scenarios. This highlights the significance of examining these characteristics collectively rather than relying on sharpness as a sole performance indicator.

According to Li et al. [19], there is a negative correlation between the loss landscape sharpness of the training set and test performance. They demonstrated that a flatter minimum in the training loss landscape leads to improved test generalizability, thereby enhancing accuracy. We further provide the Pearson correlation between the test and train accuracy and sharpness/loss in the MAML model. The results are displayed in Table 3. We discovered that the correlation between test accuracy and sharpness/loss is particularly low on the CUB dataset. This finding is reasonable, considering that when data augmentation is not used, ResNets tend to overfit on CUB. Therefore, we calculated the Pearson correlation only when data augmentation was applied. The results are presented in Table 4. We discovered that both the loss and the sharpness of the training loss landscape are significantly correlated with test performance. This supports the theory that lower training sharpness can lead to improved test performance, a principle that remains applicable in the MAML model.

Table 3: Pearson Correlation coefficients for the relationships between accuracy and sharpness, accuracy and loss on MiniImageNet and CUB datasets across backbones and data augmentation.

	MiniImageNet		**CUB**	
	train accuracy	**test accuracy**	**train accuracy**	**test accuracy**
train sharpness	-0.62(p=1e-3)	-0.77(p=1e-5)	-0.93(p=3e-11)	-0.37(p=7e-2)
train loss	-0.99(p=2e-29)	-0.83(p=5e-7)	-0.99(p=2e-25)	-0.35(p=8e-2)
test sharpness	N/A	-0.54(p=5e-3)	N/A	-0.41(p=4e-2)
test loss	N/A	-0.87(p=3e-8)	N/A	-0.63(p=8e-4)

Table 4: Pearson Correlation coefficients for the relationships between accuracy and sharpness, accuracy and loss on MiniImageNet and CUB datasets across backbones with data augmentation.

	MiniImageNet		**CUB**	
	train accuracy	**test accuracy**	**train accuracy**	**test accuracy**
train sharpness	-0.80(p=1e-3)	**-0.78**(p=2e-3)	-0.95(p=1e-6)	**-0.89**(p=8e-5)
train loss	-0.99(p=5e-14)	**-0.95**(p=1e-6)	-0.99(p=4e-14)	**-0.95**(p=1e-7)
test sharpness	N/A	-0.30(p=3e-1)	N/A	0.38(p=2e-1)
test loss	N/A	-0.95(p=1e-6)	N/A	-0.98(p=1e-8)

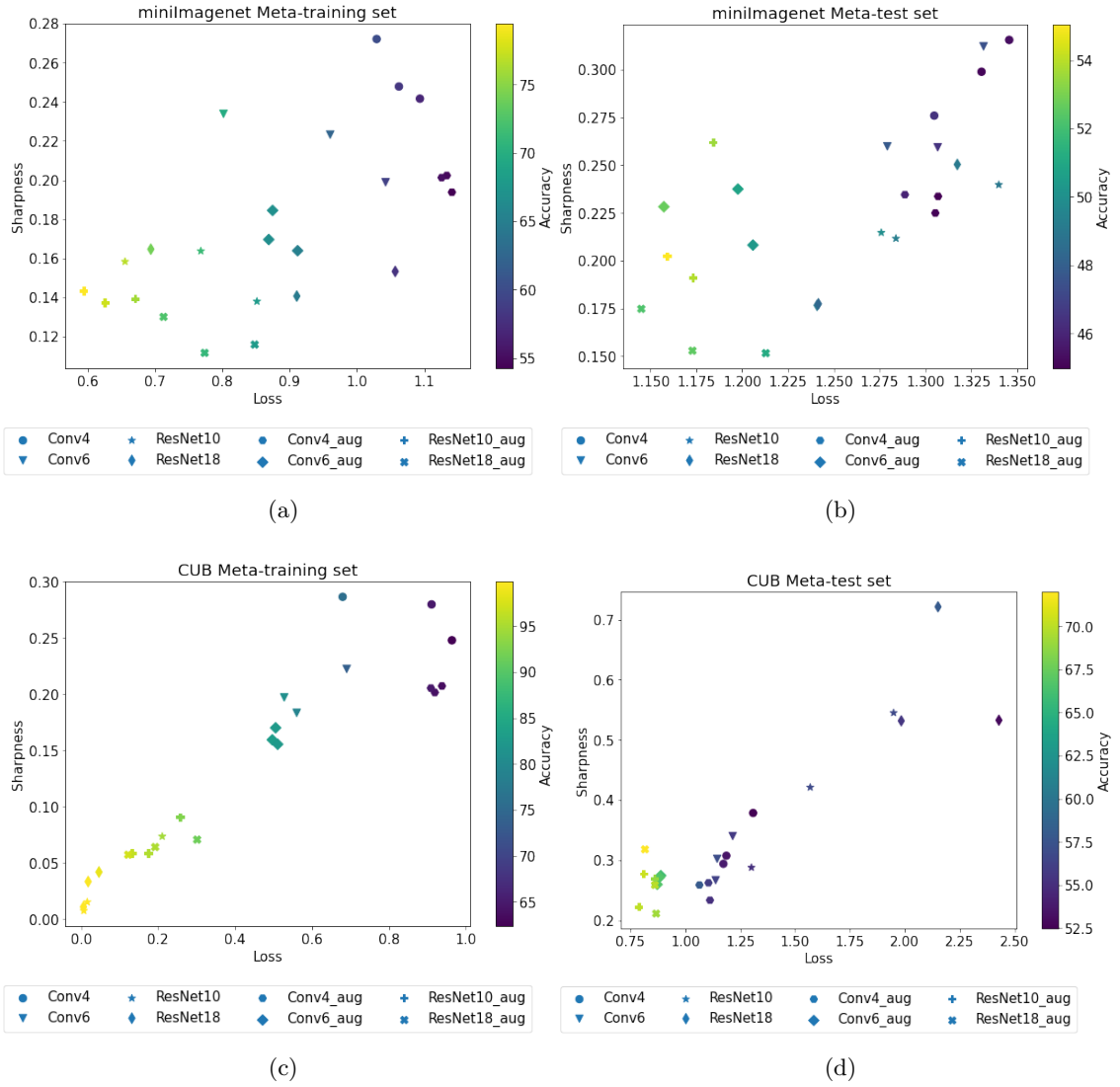


Figure 14: Comparative analysis of sharpness, loss, and accuracy for different backbone architectures with and without data augmentation. Top rows (a and b) depict results on the MiniImageNet dataset, with (a) being the training set and (b) being the test set. Bottom rows (c and d) illustrate the same for the CUBdataset, with (c) for the training set and (d) for the test set. In each graph, model accuracy(color bar) is plotted with loss (x-axis) and sharpness (y-axis) to demonstrate their interplay in the model's performance. Lower values for both sharpness and loss generally correspond with higher accuracy, indicating a robust model performance.

5 Discussion

In this thesis, we carried out a range of experiments to explore the loss landscape of Model-Agnostic Meta-Learning (MAML) across various backbone architectures and the impact of data augmentation. Our work is focused on utilizing visualizations to provide an intuitive understanding and explanation of how MAML optimizes. We aim to explain how MAML is influenced by different backbones and data augmentation techniques, aspects that are challenging to explain solely through the derivation of optimization equations [7] or by merely examining accuracies [3]. Essentially, we are attempting to address two primary research questions: How is the loss landscape of MAML affected by changes in the used backbone and the use of data augmentation? What is the relationship between the loss landscape and the few-shot learning performance of MAML?

Our visualizations reveal that the optimization of MAML comprises a highly non-convex inner loop loss landscape and a noisy yet convex-shaped outer loop loss landscape. In the inner loop visualization, we noticed a linear pattern in the loss landscapes when no updates were executed. By decomposing the loss landscapes into the head and feature layers' loss landscapes, we found that the network's linear heads have a larger impact on the total loss landscape, suggesting that identifying an appropriate weight for the head classifier during MAML's adaptation to a new task is more crucial than adjusting the weights of the feature layers. Our findings are consistent with previous findings from Ragu et al. [21] that MAML reuses features learned during training and tweaks the head layer when confronted with a new task. Because in the outer loop visualization, we observed that the meta-loss landscape is full of small spikes due to the randomness of task sampling. Despite the noise, the loss landscape exhibits a strictly convex shape, which contradicts the known challenge of optimizing MAML due to its two-level optimization structure. However, taking into account that the query loss is calculated using a significant number of images (65 images across 5 classes), we have demonstrated that an increase in the number of images causes the losses from each image to offset one another, resulting in a more convex average loss landscape across all the images. Furthermore, the meta-loss of MAML, which averages across a group of tasks, boosts an even more convex-shaped landscape. Overall, we offer a novel perspective on the optimization problem of MAML, where the few-shot nonconvexity arises from the optimization within the inner loop. The meta-loss landscape, which aims to find an initial weight that can yield a minimum average loss for all sampled tasks after inner updates, exhibits a convex shape.

To answer the first research question, i.e., we examined the meta-loss landscape's 2D contour. We averaged the query loss across 100 tasks, as opposed to only 4, for a less noisy visualization. The loss landscape of MAML is notably influenced by the depth and complexity of the network backbone. Without data augmentation, for MiniImagenet, as the network depth increases, an increase of minimum loss regions within the training set happens. For CUB, ConvNets ex-

hibit a relatively consistent training loss, while ResNets identify a significantly lower and flatter loss, hinting at potential overfitting. When data augmentation is employed, Conv4 in MiniImagenet does not seem to benefit from this process, resulting in a similar minimum. Both Conv6 and ResNet18, however, apply data augmentation as a form of regularization, leading to a reduced minimum area in meta-training and, subsequently, a lower meta-test loss. Interestingly, ResNet10 displays a deeper loss in both meta-training and meta-test, suggesting that through data augmentation, it gathers more information and achieves a lower loss. As for CUB, all backbone architectures use data augmentation as a regularizer, contributing to a decreased test loss.

From our observations, we found that in addition to the test loss, the area of the minimum loss also plays an important role in determining MAML's performance. Which provides the answer to our second research question: What is the relationship between the loss landscape and the few-shot learning performance of MAML? We quantify the idea of the area of the minimum as sharpness. Our quantitative evaluation of the loss landscape visualizations confirmed our qualitative findings. We found that data augmentation generally results in less sharpness, indicating that the model becomes more robust with data augmentation. We further explore the relationship among sharpness, loss, and accuracy for different backbone architectures, both with and without data augmentation. We calculate the Pearson correlation between sharpness and accuracy and loss and accuracy. We reveal a strong negative correlation between the training loss landscape and test performance when data augmentation is applied. In other words, a lower training loss or a flatter training loss landscape both contribute positively to MAML's test performance. Similar to Abbas et al.'s [1] findings, where they found that using an optimizer to find flatter minima improves MAML's performance, we propose that a flatter minimum inherently indicates superior performance in MAML.

Although our visualization of the loss landscape offers a valuable viewpoint to understand the optimization process of MAML, it does have limitations. Firstly, the act of visualizing the loss landscape involves significant dimension reduction, which, although offering an intuitive perspective of the optimization process in MAML, may not fully represent the complexity of the high-dimensional optimization problem. Secondly, the visualizations were created based on a single random seed, meaning that the resulting loss landscape might vary if choosing different random seeds. As a consequence, our experiments may potentially have a certain degree of variability that is not captured in our visualizations. This is an unfortunate compromise, given current computational limitations.

Building upon the results of this study, we propose several exciting opportunities for future research. Firstly, a potentially fruitful direction of research would be to visualize the trajectory of the optimization process, shedding light on how MAML navigates the loss landscape. This could be achieved by employing other dimensionality reduction techniques like PHATE

[20]. Moreover, exploring methods to compare the learned initialization of MAML with that of transfer learning could provide valuable insights into the distinctive advantages of meta-learning over more traditional machine learning approaches. By quantifying the difference in learned initialization between MAML and transfer learning, we can better understand how MAML's optimization strategy contributes to its success in few-shot learning tasks.

In summary, our research paves the way for a deeper understanding of the optimization process in MAML. Further development in visualization techniques, coupled with continued advancements in computational power, will undoubtedly lead to more accurate and detailed analyses in the future.

6 Conclusion

This thesis investigated the loss landscape of First Order Model-Agnostic Meta-Learning across diverse backbone architectures and data augmentation techniques. Our findings revealed the complex nature of MAML's optimization process, characterized by a non-convex inner loop loss landscape and a noisy but convex outer loop landscape. Notably, we found MAML predominantly reuses features learned during training, adjusting mainly the head layer for new tasks. The observed outer loop convexity contradicts the common belief of MAML's optimization complexity due to its two-level structure, hinting at potential reconsideration. Our study also highlighted the consistent contribution of data augmentation to the performance of MAML and transfer learning across various architectures and datasets, providing a dual role as a regularizer and as an additional information source for different backbones. We quantified the 'sharpness' of the loss landscape, revealing a correlation between lower loss and sharpness, and better test performance in MAML. This highlights the importance of finding flatter minima in the training of MAML. In summary, our research underscores the valuable insights from studying a model's loss landscape, aiding in identifying improvement areas for more efficient learning models. Our work provides a basis for future research on understanding the optimization process of few-shot learning models, offering a novel perspective on performance evaluation and improvement.

References

[1] Momin Abbas, Quan Xiao, Lisha Chen, Pin-Yu Chen, and Tianyi Chen. Sharp-maml: Sharpness-aware model-agnostic meta learning. In *International Conference on Machine Learning*, pages 10–32. PMLR, 2022.

[2] Yu Bai, Minshuo Chen, Pan Zhou, Tuo Zhao, Jason Lee, Sham Kakade, Huan Wang, and Caiming Xiong. How important is the train-validation split in meta-learning? In *International Conference on Machine Learning*, pages 543–553. PMLR, 2021.

[3] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. *arXiv preprint arXiv:1904.04232*, 2019.

[4] Liam Collins, Aryan Mokhtari, Sewoong Oh, and Sanjay Shakkottai. Maml and anil provably learn representations. In *International Conference on Machine Learning*, pages 4238–4310. PMLR, 2022.

[5] 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, 2009.

[6] 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.

[7] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. On the convergence theory of gradient-based model-agnostic meta-learning algorithms. In *International Conference on Artificial Intelligence and Statistics*, pages 1082–1092. PMLR, 2020.

[8] 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.

[9] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.

[10] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[11] Ian J Goodfellow, Oriol Vinyals, and Andrew M Saxe. Qualitatively characterizing neural network optimization problems. *arXiv preprint arXiv:1412.6544*, 2014.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[13] Xiangteng He, Yuxin Peng, and Junjie Zhao. Fast fine-grained image classification via weakly supervised discriminative localization. *IEEE Transactions on Circuits and Systems for Video Technology*, PP, 09 2017.

[14] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

[15] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[16] Mike Huisman, Jan N van Rijn, and Aske Plaat. A preliminary study on the feature representations of transfer learning and gradient-based meta-learning techniques. In *Fifth Workshop on Meta-Learning at the Conference on Neural Information Processing Systems*, 2021.

[17] Shruti Jadon. An overview of deep learning architectures in few-shot learning domain. *arXiv preprint arXiv:2008.06365*, 2020.

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[19] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31, 2018.

[20] Kevin R Moon, David van Dijk, Zheng Wang, William Chen, Matthew J Hirn, Ronald R Coifman, Natalia B Ivanova, Guy Wolf, and Smita Krishnaswamy. Phate: a dimensionality reduction method for visualizing trajectory structures in high-dimensional biological data. *BioRxiv*, 120378, 2017.

[21] Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. Rapid learning or feature reuse? towards understanding the effectiveness of maml. *arXiv preprint arXiv:1909.09157*, 2019.

[22] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *International conference on learning representations*, 2017.

[23] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[24] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017.

[25] Yonglong Tian, Yue Wang, Dilip Krishnan, Joshua B Tenenbaum, and Phillip Isola. Rethinking few-shot image classification: a good embedding is all you need? In *Computer*

Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV 16, pages 266–282. Springer, 2020.

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[27] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29, 2016.

[28] Xin Wang, Thomas E Huang, Trevor Darrell, Joseph E Gonzalez, and Fisher Yu. Frustratingly simple few-shot object detection. *arXiv preprint arXiv:2003.06957*, 2020.

[29] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27, 2014.

A Hyperparameters

Hyperparameter	value
inner loop lr	0.01
outer loop lr	0.001
train inner loop update	5
test inner loop update	10
optimizer	Adam
meta batch size	4

Table 5: hyperparameter used in training MAML for both CUB and MiniImagenet

Hyperparameter	value
training lr	0.001
training optimizer	Adam
training batch size	128
fine-tuning update	10
fine-tuning optimizer	SGD
fine-tuning momentum	0.9
fine-tuning dampening	0.9
fine-tuning weight decay	0.001
fine-tuning lr	0.01
few shot batch size	4

Table 6: hyperparameter used in transfer learning for both CUB and MiniImagenet

B Complementary Visualizations

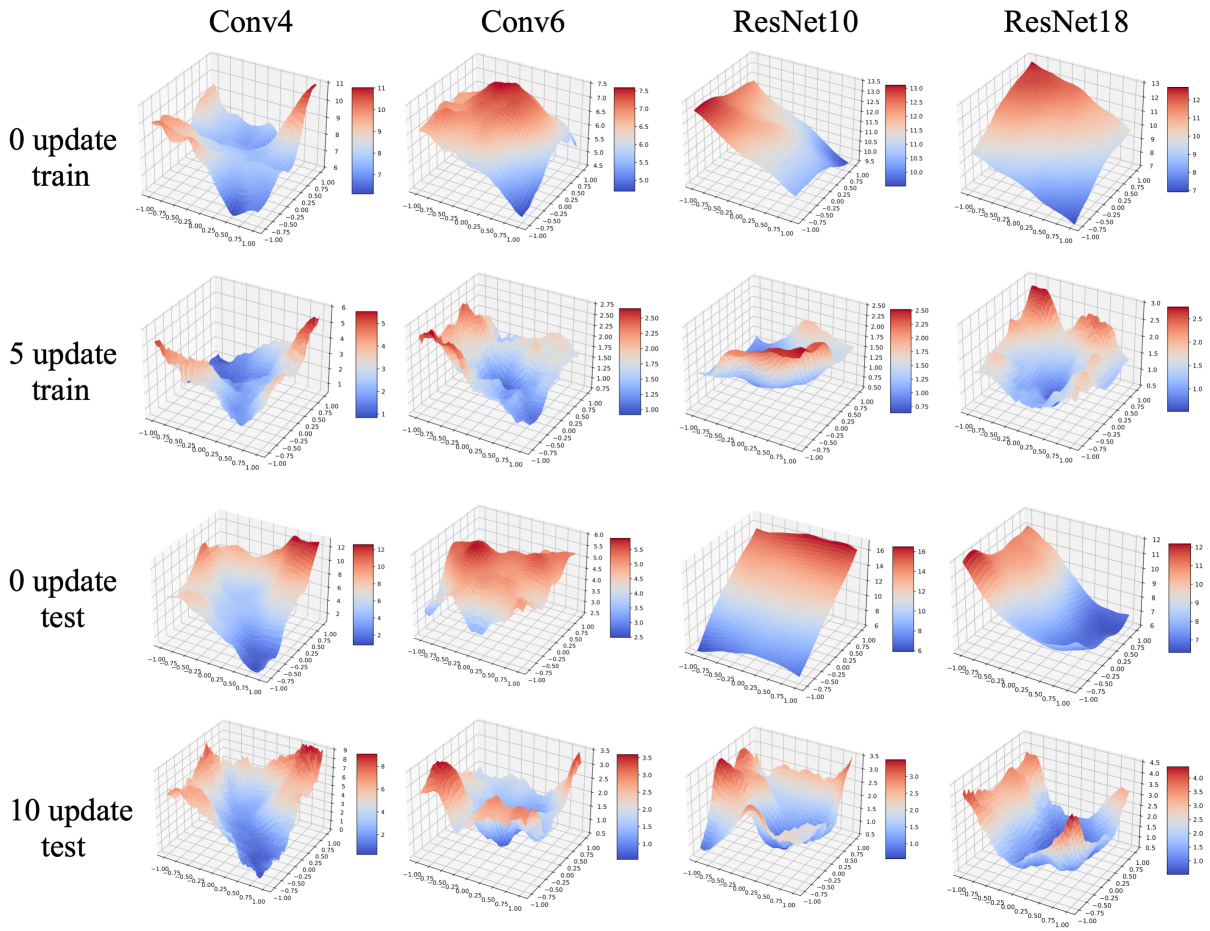


Figure 15: Loss landscapes of various network architectures on **CUB** dataset before and after inner loop optimization. 0 update means no updates have been performed on the trained initialization weight. 5 and 10 updates are the update times of meta training and meta-testing phase. The center point $(x = 0, y = 0)$ of each plot corresponds to the weight the algorithm lands on without our perturbations. (additional task)

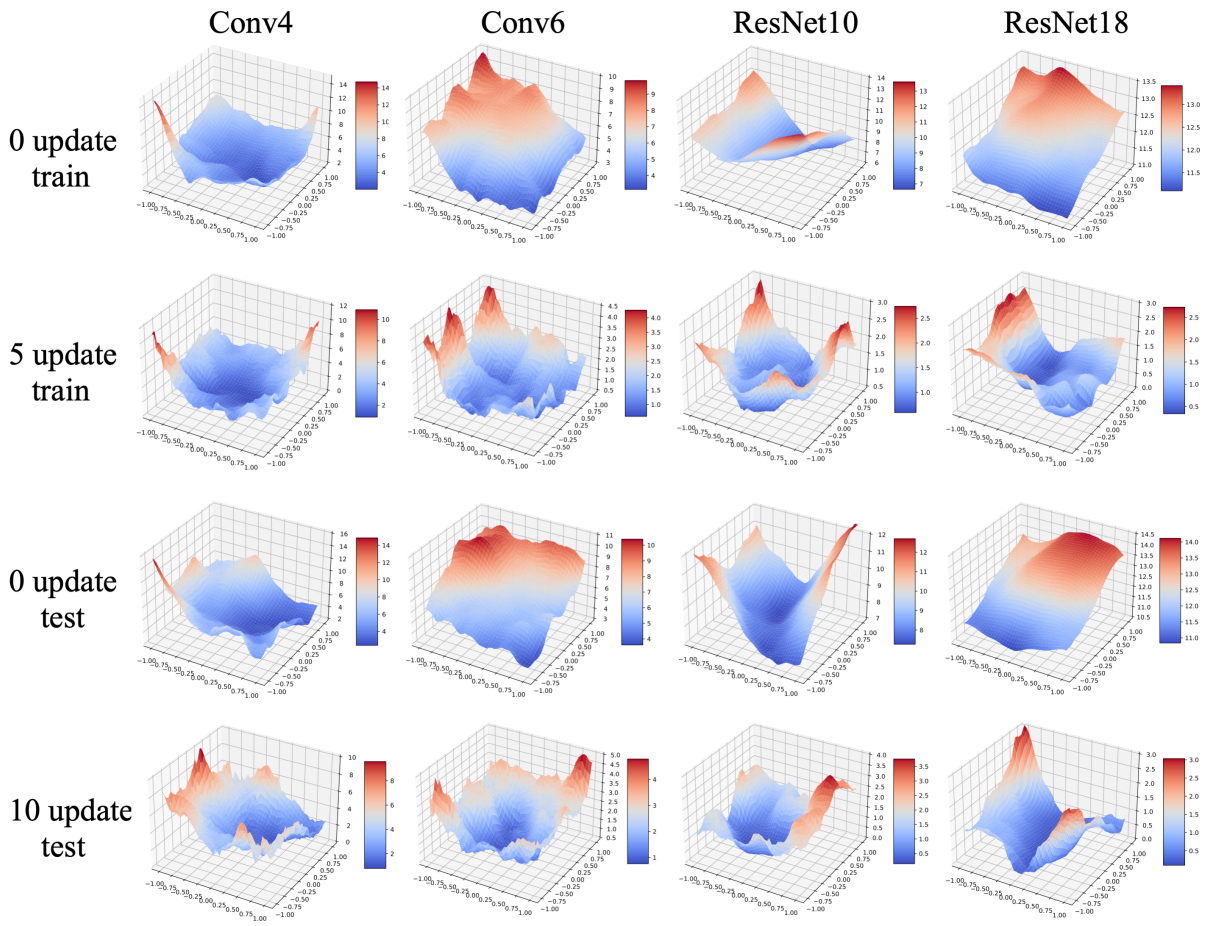


Figure 16: Loss landscapes of various network architectures on **CUB** dataset before and after inner loop optimization. 0 update means no updates have been performed on the trained initialization weight. 5 and 10 updates are the update times of meta training and meta-testing phase. The center point $(x = 0, y = 0)$ of each plot corresponds to the weight the algorithm lands on without our perturbations.(additional task)

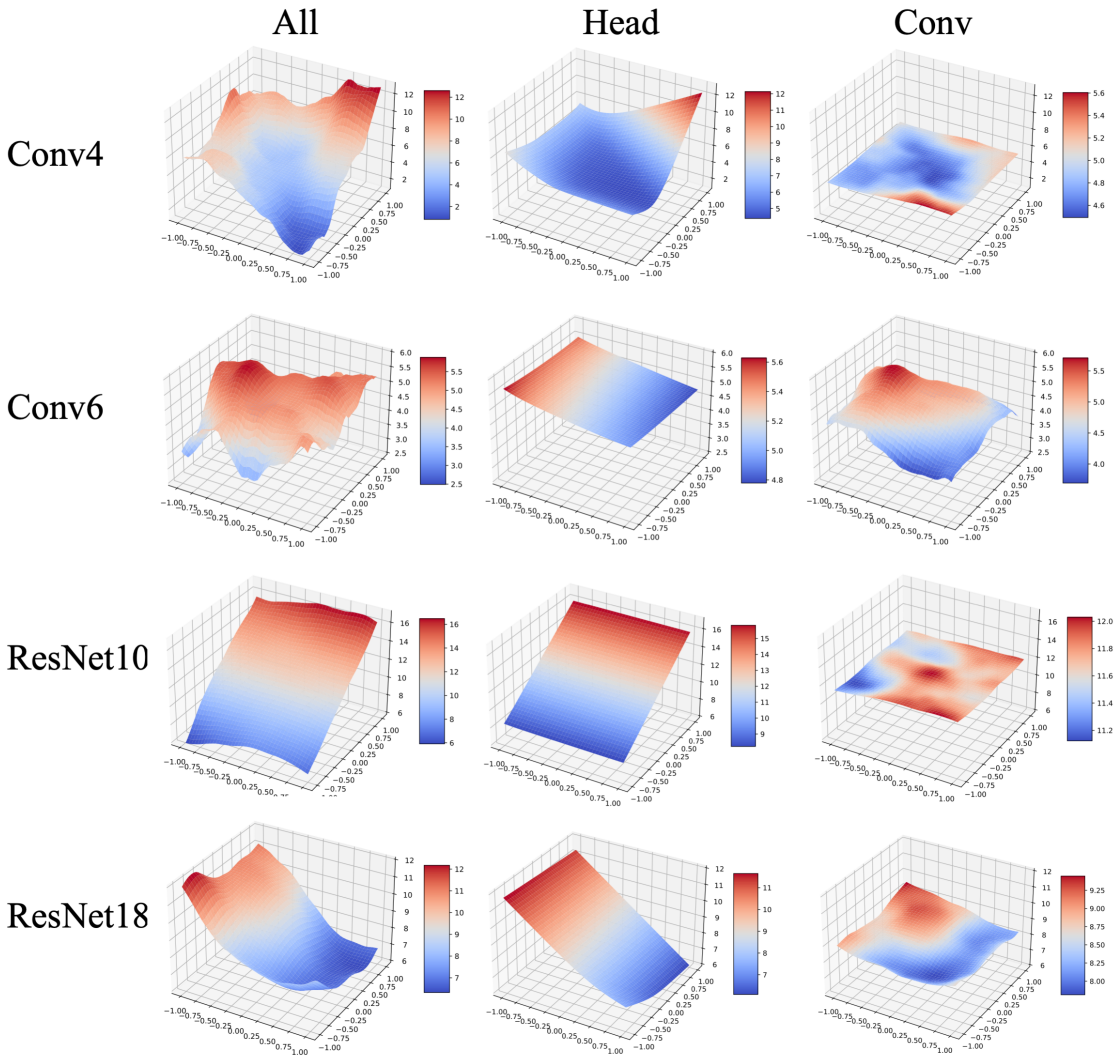


Figure 17: The decomposition of the Loss landscapes of different backbones on **MiniImageNet** at the 0 step of inner loop optimization. The left column is the loss landscape of perturbing all the network weights, The middle column is the loss landscape of perturbing only the head of the network, and the right column is the loss landscape of perturbing the feature(Conv) layers. Note that We only fix the perturbation of the loss landscape. Both the network head and feature layers are trained together during the updates. (additional task)

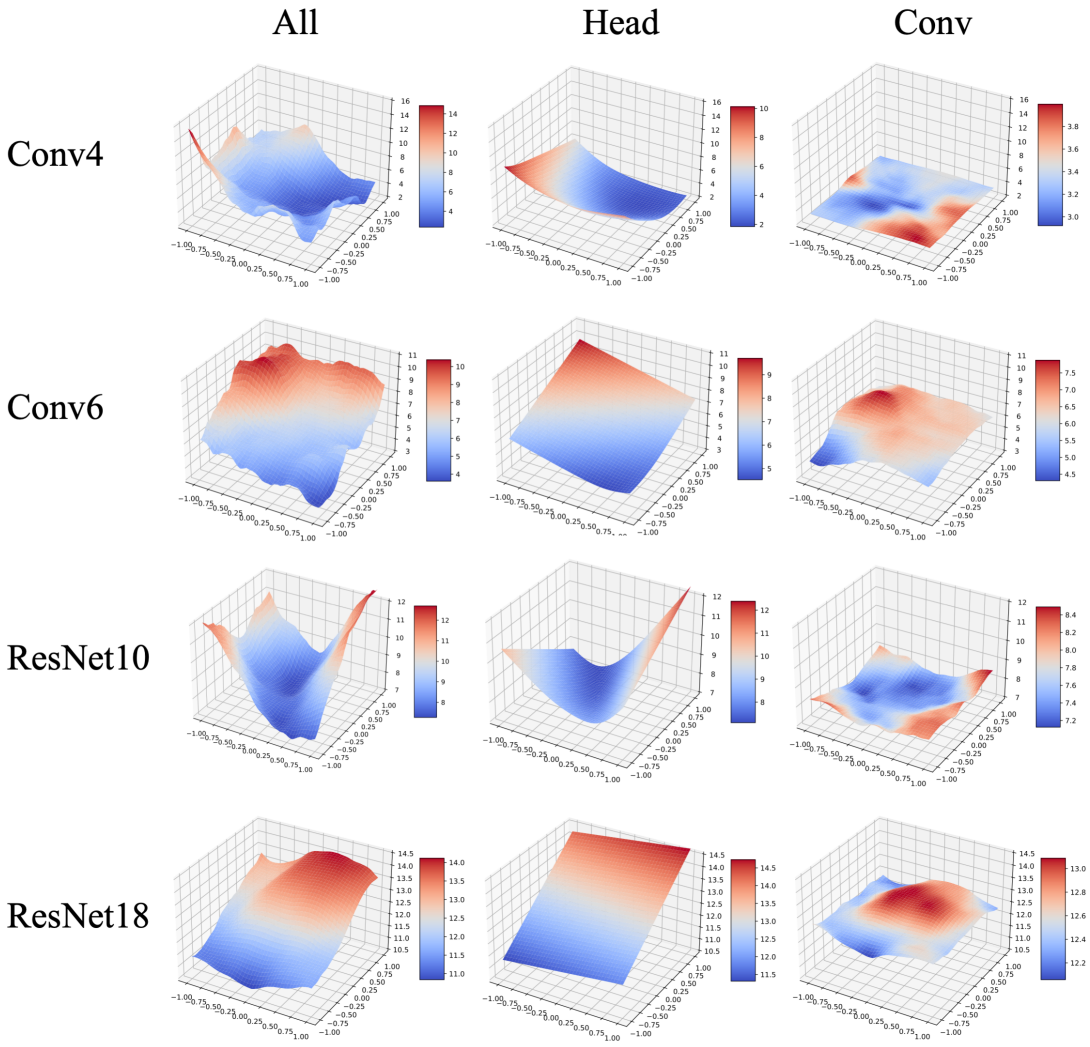


Figure 18: The decomposition of the Loss landscapes of different backbones on **CUB** at the 0 step of inner loop optimization. The left column is the loss landscape of perturbing all the network weights, The middle column is the loss landscape of perturbing only the head of the network, and the right column is the loss landscape of perturbing the feature(Conv) layers. Note that We only fix the perturbation of the loss landscape. Both the network head and feature layers are trained together during the updates.(additional task)

C Dual Norm of Sharpness

The dual norm problem arises when we want to solve the optimization problem in the third line of the equation:

$$\arg\max_{\|\boldsymbol{\epsilon}\|_p \leq \rho} \boldsymbol{\epsilon}^T \nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}) \tag{18}$$

Let's denote the dual norm of the p-norm as q-norm, where q is the conjugate exponent of p, i.e.,

$$\frac{1}{p} + \frac{1}{q} = 1 \tag{19}$$

The dual norm problem is defined as follows:

$$\max_{\boldsymbol{\epsilon}} \boldsymbol{\epsilon}^T \boldsymbol{g} \text{ subject to } \|\boldsymbol{\epsilon}\|_p \leq \rho \tag{20}$$

where $\boldsymbol{g}$ is the gradient of the loss function with respect to the weights, i.e.,

$$\boldsymbol{g} = \nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}) \tag{21}$$

To solve the dual norm problem, we can use the property of dual norms:

$$\|\boldsymbol{\epsilon}\|_p \leq \rho \Leftrightarrow \|\boldsymbol{\epsilon}\|_q \geq \frac{\boldsymbol{\epsilon}^T \boldsymbol{g}}{\rho} \tag{22}$$

Now, we want to find $\boldsymbol{\epsilon}$ that maximizes the objective function subject to the constraint. We can rewrite the constraint in terms of q-norm:

$$\hat{\boldsymbol{\epsilon}}(\boldsymbol{w}) = \rho \frac{\boldsymbol{\epsilon}}{\|\boldsymbol{\epsilon}\|_q} \tag{23}$$

The optimal $\boldsymbol{\epsilon}$ that maximizes the objective function will have the same direction as the gradient $\boldsymbol{g}$, so we can write $\boldsymbol{\epsilon}$ in terms of $\boldsymbol{g}$:

$$\hat{\boldsymbol{\epsilon}}(\boldsymbol{w}) = \rho \frac{\operatorname{sign}(\boldsymbol{g}) \, |\boldsymbol{g}|^{q-1}}{\|\boldsymbol{g}\|_q^q} \tag{24}$$

Substituting $\boldsymbol{g}$ back into the equation gives the final form:

$$\hat{\boldsymbol{\epsilon}}(\boldsymbol{w}) = \rho \operatorname{sign}\left(\nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w})\right) |\nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w})|^{q-1} / \left(|\nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w})|_q^q\right)^{1/p} \tag{25}$$

D Sharpness & Gradient

In this section of Appendix we analyze why it is possible to calculate the average loss across 100 tasks, calculating the gradient for this number of tasks is computationally impossible.

Algorithm 1 100 loss in the loss landscape

Number of tasks $T = 100$
Initialize total_loss $\leftarrow 0$
for $i \leftarrow 1$ to T **do**
    $task \leftarrow$ Sample a task
    Train on the $task$'s support set
    loss_i $\leftarrow$ Calculate loss on the $task$'s query set
    total_loss $\leftarrow$ total_loss + loss_i.item()
end for
average_loss $\leftarrow \frac{\text{total\_loss}}{T}$
return sharpness

Algorithm 2 sharpness of 100 task loss landscape

Number of tasks $T = 100$
Initialize total_loss $\leftarrow 0$
for $i \leftarrow 1$ to T **do**
    $task \leftarrow$ Sample a task
    Train on the $task$'s support set
    loss_i $\leftarrow$ Calculate loss on the $task$'s query set
    total_loss $\leftarrow$ total_loss + loss_i # we are stacking 100 networks' gradient here
end for
gradient = total_loss.backward()
sharpness = $\rho \frac{gradient}{\|gradient\|_2}$
return sharpness

Algorithm 3 average sharpness of 100 single task loss landscape (used)

Number of tasks $T = 100$
Initialize total_loss $\leftarrow 0$
for $i \leftarrow 1$ to T **do**
    $task \leftarrow$ Sample a task
    Train on the $task$'s support set
    loss_i $\leftarrow$ Calculate loss on the $task$'s query set
    gradient_i = loss_i.backward()
    sharpness_i = $\rho \frac{gradient\_i)}{\|gradient\_i\|_2}$
    total_sharpness $\leftarrow$ total_sharpness + sharpness_i
end for
return $\frac{\text{total\_sharpness}}{T}$

E Sharpness Results

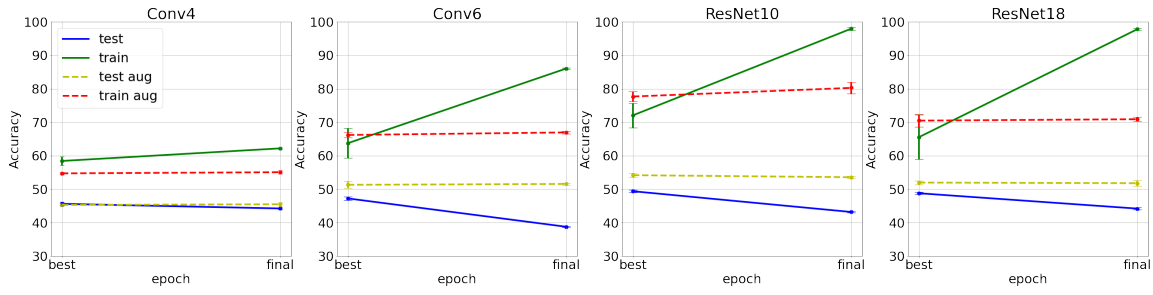
Table 7: Average Sharpness for Different Network Backbones)$\times 10^{-1}$) MiniImageNet

	Base		Novel	
Backbone	No Aug	Aug	No Aug	Aug
Conv4	2.524 ± 0.138	1.978 ± 0.099	2.916 ± 0.117	2.320 ± 0.081
Conv6	2.138 ± 0.133	1.705 ± 0.123	2.728 ± 0.152	2.205 ± 0.108
ResNet10	1.547 ± 0.173	1.409 ± 0.164	2.265 ± 0.172	2.295 ± 0.155
ResNet18	1.530 ± 0.158	1.174 ± 0.141	1.955 ± 0.134	1.581 ± 0.122

Table 8: Average Sharpness for Different Network Backbones)$\times 10^{-1}$)CUB

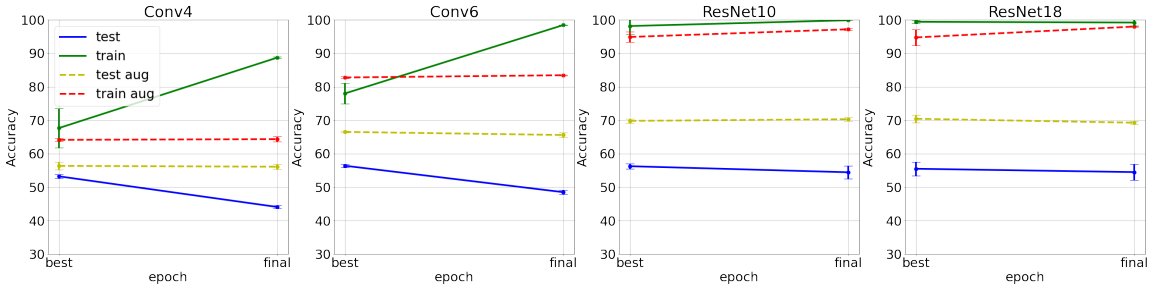
	Train		Test	
Backbone	No Aug	Aug	No Aug	Aug
Conv4	2.654 ± 0.178	2.048 ± 0.133	3.185 ± 0.203	2.533 ± 0.145
Conv6	2.063 ± 0.195	1.601 ± 0.178	2.940 ± 0.218	2.663 ± 0.218
ResNet10	0.279 ± 0.096	0.667 ± 0.116	4.189 ± 0.452	2.597 ± 0.219
ResNet18	0.299 ± 0.157	0.641 ± 0.156	5.798 ± 0.352	2.616 ± 0.233

F Overfitting



(a) Conv4 (b) Conv6 (c) ResNet10 (d) ResNet18

Figure 19: MiniImageNet accuracy



(a) Conv4 (b) Conv6 (c) ResNet10 (d) ResNet18

Figure 20: CUB accuracy