Université de Montréal
Faculté des arts et des sciences

Ce mémoire intitulé:

Towards Understanding Generalization in Gradient-Based Meta-Learning

présenté par:

Simon Guiroy

a été évalué par un jury composé des personnes suivantes:

Emma Frejinger,	président-rapporteur
Christopher Pal,	directeur de recherche
Guillaume Rabusseau,	membre du jury

Mémoire accepté le: 30 août 2019

Résumé

Dans ce mémoire, nous étudions la généralisation des réseaux de neurones dans le contexte du méta-apprentissage, en analysant divers priopriétés des surface leurs fonctions objectifs. La recherche en apprentissage automatique portant sur les surfaces de fonctions objectifs des réseaux de neurones ayant aidé à comprendre leur généralisation en apprentissage supervisé standard, nous proposons l'étude de telles surfaces dans le but d'approfondir nos connaissances sur la généralisation en méta-apprentissage. Nous introduisons d'abord la littérature sur les fonctions objectifs des réseaux de neurones à la Section 1.2, puis celle portant sur le méta-apprentissage à la Section 1.3, pour enfin terminer notre introduction avec le méta-apprentissage par descente de gradient, très similaire à l'entraînement des réseaux de neurones par descente de gradient stochastique et pour une tâche unique. Nous présentons par la suite notre travail sur les fonctions objectifs en méta-apprentissage au Chapitre 2, lequel nous avons soumis à la conférence NeurIPS 2019 en tant qu'article scientifique. Au moment d'écrire ce mémoire, et au meilleur de notre connaissance, ce travail est le premier à étudier empiriquement les surfaces des fonctions objectifs en méta-apprentissage dans le contexte de l'apprentissage profond, mettant en lumière certaines propriétés de ces surfaces qui apparaissent liées à la généralisation des réseaux de neurones à de nouvelles tâches. Nous démontrons qu'alors que progresse le méta-entraînement, pour les solutions aux nouvelles tâches obtenues via quelques itérations de descente de gradient, la courbure de la fonction objective décroit monotoniquement, la valeur de la fonction objective diminue, tandis que la distance euclidienne avec la solution "méta-entraînement" augmente. Nous observons que la courbure des minima continue de décroître même lorsque le sur-apprentissage devient apparent et que la généralisation commence à se dégrader, indiquant que la courbure des minima semble peu corélée à la généralisation en méta-apprentissage par descente de gradient. De plus, nous montrons que la généralisation aux nouvelles tâches semble plutôt liée à la cohérence de leurs trajectoires d'adaptation dans l'espace des paramètres, mesurée par la similarité cosinus moyenne entre les trajectoires. Nous montrons également que la cohérence des gradients "meta-test", mesurée par le produit scalaire moyen entre les vecteurs de gradients spécifiques aux nouvelles tâches, evalué à solution meta-entraînement, est également corélée à la généralisation. Nous basant sur ces observations, nous proposons un nouveau terme de régularisation pour l'algorithme de méta-apprentissage Model Agnostic Meta-Learning (MAML).

Mots-clés: apprentissage profond, méta-apprentissage, fonction objectif, généralisation, apprentissage via peu d'exemples

Summary

In this master's thesis, we study the generalization of neural networks in gradient-based meta-learning by analyzing various properties of the objective landscapes. Meta-learning, a challenging paradigm where models not only have to learn a task but beyond that, are trained for "learning to learn" as they must adapt to new tasks and environments with very limited data about them. With research on the objective landscapes of neural networks in classical supervised having provided some answers regarding their ability to generalize for new data points, we propose similar analyses aimed at understanding generalization in meta-learning. We first introduce the literature on objective landscapes of neural networks in Section 1.2. We then introduce the literature of meta-learning in Section 2, concluding our introduction with the approach of gradient-based meta-learning, a meta-learning setup that bears strong similarities to the traditional supervised learning setup through stochastic gradient-based optimization. At the time of writing of this thesis, and to the best of our knowledge, this is the first work to empirically study the objective landscapes in gradient-based meta-learning, especially in the context of deep learning. We notably provide some insights on some properties of those landscapes that appear correlated to the generalization to new tasks.

We experimentally demonstrate that as meta-training progresses, the meta-test solutions, obtained after adapting the meta-train solution of the model, to new tasks via few steps of gradient-based fine-tuning, become flatter, lower in loss, and further away from the meta-train solution. We also show that those meta-test solutions become flatter even as generalization starts to degrade, thus providing experimental evidence against the correlation between generalization and flat minima in the paradigm of gradient-based meta-leaning. Furthermore, we provide empirical evidence that generalization to new tasks is correlated with the coherence between their adaptation trajectories in parameter space, measured by the average cosine similarity between task-specific trajectory directions, starting from a same meta-train solution. We also show that coherence of meta-test gradients, measured by the average inner product between the task-specific gradient vectors evaluated at meta-train solution, is also correlated with generalization. Based on these observations, we propose a novel regularizer for the Model Agnostic Meta-Learning (MAML) algorithm and provide experimental evidence for its effectiveness.

Keywords: deep learning, meta-learning, objective landscapes, generalization, few-shot learning

Table des matières

Table des figures

List of Abbreviations

CNN	Convolutional Neural Network
DNN	Deep Neural Network
MAML	Model Agnostic Meta-Learning
ML	Machine Learning
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent

Acknowledgments

I want to express my profound gratitude to my advisor and co-author Christopher Pal, who encouraged me in pursuing my research interests, as well as my passion, who trusted me. I also want to thank him for his supervision and precious advice. I want to thank Yoshua Bengio for his vision and determination in helping flourish, in Montréal, our vibrant research community. I want to thank as well my co-author Vikas Verma, who from early on has believed in the endeavor of the work presented in this thesis.

I would like to thank my mother Johanne Vermette and father Alain Guiroy, for their support, coaching, encouragement and their love. I would like to thank my friends Olivier Rousseau and Charles-Émile Trudel, my little brothers Antoine and Mathieu, for all their support. I would also like to thank the following colleagues, friends, and professors with whom I enjoyed very insightful discussions and precious feedback: Tristan Deleu, Valentin Thomas, Brady Neal, Jie Fu, Hugo Larochelle, Xavier Bouthiller.

Finally, the work reported in this thesis would not have been possible without the financial support from the following institutions: Ubisoft, Google, Samsung, IBM, NSERC, Calcul Quebec, Compute Canada, the Canada Research Chairs and CIFAR

1 Introduction

1.1 Overview

The work that we present in Chapter 2 of this thesis proposes to study the generalization of neural networks in gradient-based meta-learning, by empirically analyzing the properties of the objective landscapes involved in the optimization of those neural networks. At the time of writing this thesis and to the best of our knowledge, there had been no prior work empirically analyzing the objective landscapes involved in gradient-based meta-learning. In this chapter, we first introduce in Section 1.2 the literature that studied the objective landscapes in standard, supervised machine learning. This rich body of literature, with the various topics that were studied and with the results that it provides, constituted a valuable reference for crafting our questions and inquiries about the objective landscapes of gradient-based meta-learning. We then introduce Meta-Learning in Section 1.3, a subfield of machine learning where the goal is not simply to design a model that can learn some task but rather, to design algorithms for models to "learn how to learn". We summarize the different approaches to meta-learning, ending with the approach of gradient-based meta-learning, which is part of the central focus of this work. With these two research areas introduced, we then present our work in Chapter 2.

1.2 Objective Landscapes of Neural Networks

To understand the phenomenons involved in the optimization of neural networks to their learning problems, but also in a hope to understand the underlying factors behind their generalization, researchers in the machine learning community have studied the training objective landscapes of neural networks, giving rise to a rich body of literature.

Traditionally, the goal of machine learning is to train a function f that learns a task, for example, an image classification task and f is parametrized by its parameter vector θ where $\theta \in \mathbb{R}^d$. The model f takes as input a sample $\mathbf{x} \in \mathbb{R}^P$ and outputs a class prediction vector among m classes. In supervised learning, we compare this vector and the ground truth vector $\mathbf{y} \in \mathbb{R}^m$, which tells us if the model succeeded in classifying the sample, but also allows us to compute a *loss* for this classification via a loss function, which is usually the cross-entropy between $f(\mathbf{x};\theta)$ and $\mathbf{y}$. To learn a function f that has a high expected accuracy for the task, we usually want to minimize the expectation of some loss function $\mathcal{L}(f(\mathbf{x};\theta),\mathbf{y})$ with respect to the model parameters θ. This expectation $\mathbb{E}_{\mathbf{x},\mathbf{y}}\mathcal{L}(f(\mathbf{x}_i;\theta),\mathbf{y}_i)$ is the population risk, but when training f we don't have access to the whole population nor do we know the true data generating distribution. We thus train the model by minimizing the empirical risk over the data from a training set $\mathcal{D}^{train}$:

$$\text{Empirical Risk} := \frac{1}{|\mathcal{D}^{train}|} \sum_{(\mathbf{x}_i,\mathbf{y}_i)\in\mathcal{D}^{train}} \mathcal{L}(f(\mathbf{x}_i;\theta),\mathbf{y}_i) \tag{1.1}$$

obtaining a solution $\tilde{\theta}$. Generalization is measured by the ability of the trained model $f(\cdot;\tilde{\theta})$ to reach a high accuracy on test samples from $\mathcal{D}^{test}$ that weren't seen during training but that belong to the same task (generated by the same data generating distribution as for $\mathcal{D}^{train}$), and we thus care about maximizing this test accuracy:

$$Acc(f(\mathcal{D}^{test};\tilde{\theta})) := \frac{1}{|\mathcal{D}^{test}|} \sum_{(\mathbf{x}_i,\mathbf{y}_i)\in\mathcal{D}^{test}} \mathbb{1}_{\{\arg\max f(\mathbf{x}_i;\tilde{\theta}) \,=\, \arg\max \mathbf{y}_i\}} \tag{1.2}$$

We note that good generalization is often defined as having a small gap between the training and testing accuracies, but ultimately we want the model to reach a high testing accuracy (for example, a small gap between training and testing accuracy both at random chance, shouldn't be considered as good generalization). Since the loss landscape of neural networks (the empirical risk) on real world problems being of high dimension and, more importantly, non-convex, optimization is normally performed by a *gradient descent* method where the parameters are iteratively updated:

$$\theta^{s+1} = \theta^{s} - \alpha \frac{1}{|\mathcal{D}^{train}|} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{D}^{train}} \nabla_{\theta} \mathcal{L}(f(\mathbf{x}_i; \theta^{s}), \mathbf{y}_i) \tag{1.3}$$

In practice, neural networks are trained via *stochastic gradient descent* (SGD), where at each time step s we minimize the loss over a randomly sampled subset of $\mathcal{D}^{train}$ of size n, called a mini-batch:

$$\theta^{s+1} = \theta^{s} - \alpha \frac{1}{n} \sum_{i=1}^{n} \nabla_{\theta} \mathcal{L}(f(\mathbf{x}_i; \theta^{s}), \mathbf{y}_i) \tag{1.4}$$

where α is the optimization step-size, also called the learning rate. Optimization continues until reaching a critical point $\nabla_{\theta} \mathcal{L}(f(\mathbf{x}; \tilde{\theta}), \mathbf{y}) \leq \epsilon$ (or solution) where we hope the loss to be minimal $\mathcal{L}(f(\mathbf{x}_i; \tilde{\theta}), \mathbf{y}_i) \approx 0$. Most of the literature that deals with the objective landscapes of neural networks has focused on objective landscapes obtained from optimization via SGD. The theoretical contributions for deep neural networks are still relatively modest, with many works dealing with linear or shallow networks, but some were able to provide theoretical results for deeper, nonlinear neural architectures. Experimentally, most works use either standard fully-connected neural networks (MLP) or convolutional neural networks (CNN or ConvNet) [Lecun et al., 1998], often as variants of a popular deep CNN architecture, the VGG network [Simonyan and Zisserman, 2014]. Most of these works deal with the objective landscape for a classification task, and thus use cross-entropy as their loss function, and often use MNIST or Cifar10 [Krizhevsky, 2009] as their dataset, but some dealt with regression and thus Mean Square Error as their loss function. These experimental design choices in the literature, as well as the mathematical objects of interest (minima and their curvature, optimization trajectories, etc), served as a reference for the different research and experimental decisions we made for our work that we present in Chapter 2.

In this section we review a good portion of the results that stemmed from that literature, and divide it into three subsections: 1) *Properties of solutions*, where the optimization minima or points of convergence are being analyzed ; 2) *Properties of optimization trajectories*, where the optimization trajectory between network initialization and solution is analyzed ; and 3) *General properties of objective landscapes*, which concerns properties of the landscapes more generally across the parameter space, and not necessarily only where lies the point of convergence nor the optimization trajectory.

Throughout those subsections, we cover multiple questions that have generated great interest and debate among the machine learning community. This review of literature helped orient our approach of conducting our analyses of the objective landscapes in gradient-based meta-learning that we present in Chapter 2, where we focus on the local properties of solutions as well as the optimization trajectories involved in meta-learning.

1.2.1 Properties of solutions

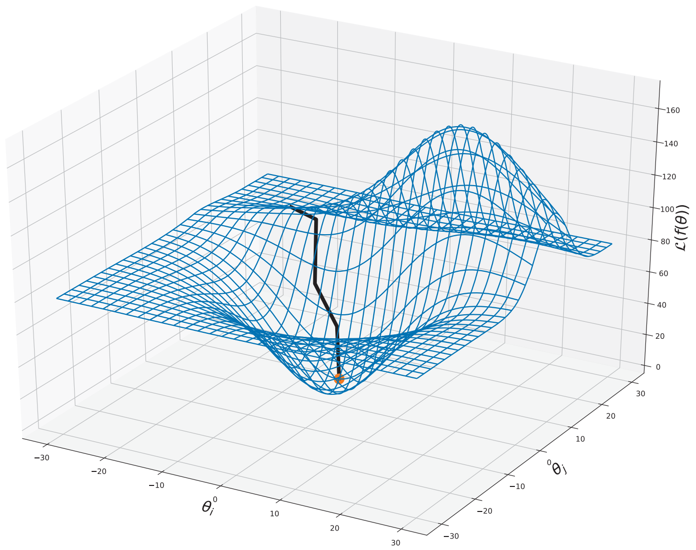


Figure 1.1 – Solutions found at convergence of optimization

In this subsection about the objective landscape of minima or solutions found by SGD, where $\frac{\partial}{\partial \theta_j}\mathcal{L}(f(\mathbf{x};\tilde{\theta}),\mathbf{y}) \leq \epsilon, \ \ \forall j \in [1,..,p]$, we cover two major questions regarding local properties of the landscapes around those points, the first being the curvature of such minima and whether their flatness is indicative of better generalization, and secondly, whether those minima found by SGD are actually minima or if they are in fact, most often, saddle points.

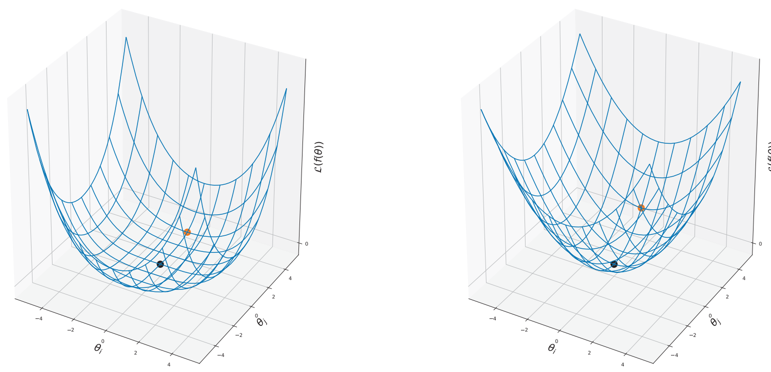


Figure 1.2 – Curvature of minimam: (left) Flatter minimum ; (right) Sharper minimum. For a same small displacement $\delta\theta$ in parameter space, the increase in level of loss $\delta\mathcal{L}$ is greater for a sharper minimum than for a flatter minimum.

Curvature of minima The notion of *curvature of minima* usually refers to how much the loss may vary in the vicinity of the minima, with flatter minima demonstrating less variation of their loss around them. For flatter minima, a small displacement in parameter space $\Delta\theta$ would thus result in a smaller increase in loss $\Delta\mathcal{L}$, compared to a minimum with a sharper curvature. Thus if we consider the training loss landscape to be an approximation of the testing loss landscape, where the test landscape has the same shape as its training counterpart but shifted by a small displacement $\Delta\theta$ in parameter space, then a flatter training minimum would lead to a smaller gab $\mathcal{L}(f(\mathcal{D}^{test};\tilde{\theta})) - \mathcal{L}(f(\mathcal{D}^{train};\tilde{\theta}))$ between train and test loss, compared to a sharper minimum [Keskar et al., 2016], and would also lead to a lower test loss. The notion of flatness of minima obtained by training neural networks via stochastic gradient descent, and its hypothesized relation to generalization, was first proposed by [Hochreiter and Schmidhuber, 1997a]. In their work, the authors define the region of flatness of a minimum through a sensibility measure, limited by a hypervolume in parameter space where the loss variation is bounded by a small quantity ϵ, and show a correlation between generalization and their measure of flatness.

In more recent years, many authors have since revisited this notion of flat minima, in light of the progresses made by deep neural networks. For most authors, the flatness of minima is directly measured by the curvature of the loss surface at that particular point, through the second-order derivatives of the loss, with respect to the model parameters. First defining the average loss over some subset $\mathcal{D}$ of the training dataset for f evaluated at a solution $\tilde{\theta}$, as $\mathcal{L}(f(\mathcal{D};\tilde{\theta}))$, we define the hessian matrix of that loss as $H_\theta(\mathcal{D};\tilde{\theta}) \doteq \nabla^2_\theta \mathcal{L}(f(\mathcal{D};\tilde{\theta}))$. Most authors use, as an estimate of the curvature, the spectral norm of the hessian matrix of the loss, at a given point in parameter space:

$$\left\| H_\theta(\mathcal{D};\tilde{\theta}) \right\|_\sigma = \sqrt{\lambda_{max}\left(H_\theta(\mathcal{D};\tilde{\theta})^{\mathrm{H}} H_\theta(\mathcal{D};\tilde{\theta}) \right)} = \lambda_{max}(H_\theta(\mathcal{D};\tilde{\theta})) \tag{1.5}$$

where the spectral norm of the hessian being equal to its first eigenvalue, since it is a real and symmetric matrix. Some authors rely instead on the determinant of the hessian. Most work in the literature found flatter minima to be correlated with better generalization [Hochreiter and Schmidhuber, 1997a], [Keskar et al., 2016], [Xing et al., 2018], [Li et al., 2017], [Smith and Le, 2017], [Jastrzebski et al., 2017].

Moreover, it had been observed empirically that when training neural nets with SGD, using a larger batch size usually leads to poorer generalization. [Keskar et al., 2016] showed experimentally that training with larger batches leads to sharper minima, which in turn led to poorer generalization. In contrast, their results show that small-batch methods lead to flatter minima associated with better generalization.

In a similar fashion, [Jastrzebski et al., 2017] identify three factors influencing minima found by SGD, namely the learning rate, the batch-size and the inherent variance of the loss gradients, all factors which control the trade-off between the depth and width of the minima found by SGD, with flatter minima favoured by a higher ratio of learning rate to batch size α/n, demonstrate better generalization compared to sharper minima.

However, the work of [Dinh et al., 2017] provided a theoretical argument against the classical notion of flatness of minima and its correlation to generalization. They showed that because of the symmetries within the architecture of neural networks, where neurons can be permuted and thus so can be the weights, it is possible, when having a flat minimum that generalize well, to find a reparametrization of the model that defines the exact same function (mapping inputs to outputs), with an arbitrarily sharper minimum, yet this reparametrization will generalize just as well, as it defines the same function as the one defined by the original flat minimum.

While most research points to the correlation between flat minima and generalization, this topic is still in debate, which also motivated our decision to analyze this property in the objective landscapes of gradient-based meta-learning, as the first part of our work presented in Chapter 2.

Local minima vs. saddle points Let's recall that a minimum is a critical point on the loss landscape where the curvature is positive in all directions, i.e.

$$\lambda_j \left(\frac{\partial^2}{\partial \theta^2} \mathcal{L}(f(\mathcal{D}; \tilde{\theta})) \right) > 0, \quad \forall j \in [1, .., p] \tag{1.6}$$

where directions are defined by the eigenvectors of the hessian matrix of the loss. A saddle-point is a point where there is a change of sign of curvature in at least one direction, in this case, a negative curvature. It had long been believed that what hindered neural network training, as optimization occurs in a highly non-convex loss landscape, was the presence of local minima, where SGD gets trapped and is therefore prevented from reaching solutions of lower loss value. However, the works of [Pascanu et al., 2014] [Dauphin et al., 2014] highlight that there are exponentially more saddle points than local minima in high dimensions, emphasizing that saddle points are much more important to focus on than local minima.

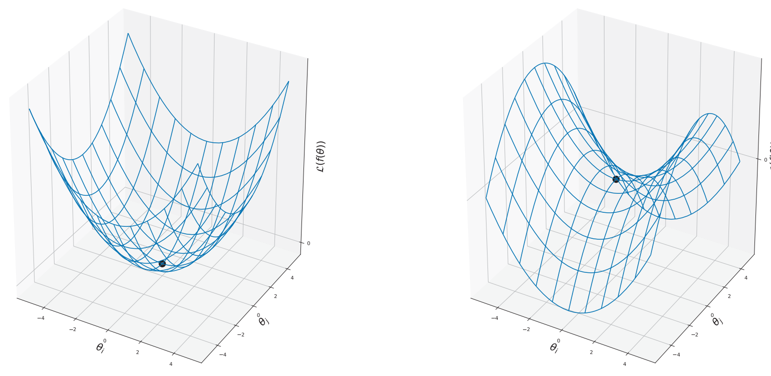


Figure 1.3 – Critical points of optimization: (left) Minimum ; (right) Saddle-point

1.2.2 Properties of optimization trajectories

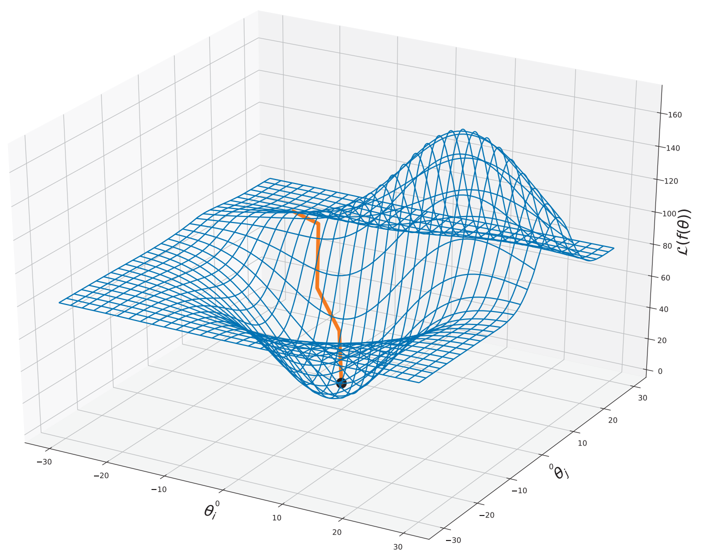


Figure 1.4 – Optimization trajectories from initialization to point of convergence

Apart from studying properties of the objective landscapes around solutions found by SGD, researchers have also extensively studied phenomenons happening during the course of optimization, where they investigated the properties of the optimization trajectories $(\theta^0, \theta^1, \theta^2, ..., \tilde{\theta})$ between initialization to the final solution. Here we first address the works that studied the distance $\|\tilde{\theta} - \theta_0\|$ travelled by those trajectories, usually measured by Euclidean distance $l_2(\cdot) = \|\cdot\|_2$ in parameter space. We thus cover those works that, in part at least, attempt to answer the question *where does SGD end up, with respect to the initialization*. We then address the works that deal with the smoothness of the optimization trajectory, or more generally, how does the landscape varies during the course of optimization, may it be in terms of variation of the loss, its gradients, even its curvature, as well as the variation of direction that those landscapes give to the optimization trajectory. Finally, we address the works dealing with the dimensionality of the optimization trajectory, often pointing out that learning actually happens in a subspace of the parameter space, often of much lower dimensionality.

Norm of trajectories Some authors have studied the norm of solutions attained by training neural networks, often in relation to the notion of model complexity, to explain the generalization abilities of neural networks. Notably, weight decay is a popular regularization technique that constraints the l_2 norm of θ during optimization, to control for model complexity, and thus reducing overfitting.

[Zhang et al., 2016] studied the norm of solutions, in the context of model complexity and regularization, and their effect on generalization. One common metric for measuring model complexity is the Rademacher complexity, which measures the richness of a family of functions with respect to a probability distribution. Given a dataset $S = (x_1, x_2, ..., x_n) \in Z^n$, a family of real-valued functions $\mathcal{F}$ defined over the domain space Z, the Rademacher random variables $\sigma_1, \sigma_2, ..., \sigma_n$ where $\sigma_i \in \{\pm 1\}$, $\sigma_i \overset{\text{iid}}{\sim} Pr(\sigma)$ and $Pr(\sigma_i = -1) = Pr(\sigma_i = +1) = \frac{1}{2}$, we obtain the empirical Rademacher complexity for this family of functions:

$$\hat{\Re}_S(\mathcal{F}) = \frac{1}{n}\mathbb{E}_\sigma \left[\sup_{f \in \mathcal{F}} \sum_{i=n}^{n} \sigma_i f(x_i) \right] \tag{1.7}$$

The Rademacher complexity basically measures the ability of a family of function to fit random data. Here we would define $\mathcal{F}$ as the family of functions achievable by training deep neural networks of a given architecture via SGD. The basic idea is that, although the original, unregularized family of function is too complex to generalize well (and thus may lead to functions that overfit on the training data), adding explicit regularization, for example, restricting the norm of the optimal solution $\tilde{\theta}$ (weight decay, also called l_2 regularization), the effective Rademacher complexity is dramatically reduced, and this should lead to solutions that generalize better. However, even when optimizing with weight decay, models are able to perfectly fit randomly labeled data, thus exhibiting Rademacher complexity approaching the highest possible value such that $\hat{\Re}_S(\mathcal{F}) \approx 1$, yet those same models showed small gaps between training and testing error (high generalization). The importance of their result posed a conceptual challenge to statistical learning theory as traditional measures of model complexity struggle to explain the generalization ability in the case of deep neural networks. Moreover, even when turning off explicit regularization, models were still able to generalize well. This suggests that explicit generalization relying on constraining the norm of the network parameters, while helpful, probably aren't fundamental factors responsible for generalization. Investigating the possible causes of generalization, the authors showed that, for linear models, SGD always converges to a solution with small norm, therefore acting as an implicit regularizer, compared to full-batch gradient descent.

Additionally, [Xing et al., 2018] show empirically that training neural networks until the training loss saturates, achieves solutions with smaller norms when training with SGD, than when training with Gradient Descent (GD), i.e., when $n = |\mathcal{D}^{train}|$. When measuring $\|\theta_t - \theta_0\|_2$, for only the initial 40 training iterations, they observe a higher distance traveled for SGD than for GD, where SGD "quickly moves away from initialization", to later on evolve in flatter loss plateaus, corresponding to better generalization. They also show that training with a smaller batch-size n allows SGD to travel further away from initialization (we will cover their explanation in the next paragraph). Those solutions of higher l_2 norm obtained from training with smaller batch size n end up in flatter regions that authors show to correlate with better generalization.

Variance of trajectories Here we address the works that cover the different types of variance occurring in the optimization trajectories of neural networks, namely the variance in magnitude of the loss $\mathcal{L}(f(\mathcal{D}_t^{train}; \theta_t))$ (where $\mathcal{D}_t^{train}$ is the mini-batch at time t), the gradient of the loss and the curvature of the loss, which are related to the smoothness of the landscape along optimization trajectory, as well as variance in gradient direction.

As we mentioned above, [Xing et al., 2018] showed different results regarding the optimization trajectory norm, in different scenarios and taking into account stochasticity of the gradients, batch size and learning rate. As another metric, they also tracked the cosine similarity between consecutive gradient vectors $g_t = \nabla_\theta \mathcal{L}(f(\mathcal{D}_t^{train}; \theta_t))$ as training progresses for the first 40 iterations. We'll define this cosine similarity as:

$$\cos(g_{t+1}, g_t) = \frac{{g_{t+1}}^T g_t}{\|g_{t+1}\|_2 \|g_t\|_2} \tag{1.8}$$

They show that for gradient descent (GD), the value of $\cos(g_{t+1}, g_t)$ gradually decreases to eventually be ~ -1, implying that optimization bounces off between "walls" of high loss. For SGD, cosine similarity decreases but stays above 0, suggesting that instead of oscillating in the same region, SGD is quickly moving away from its previous position. They also show that when training with SGD, $\cos(g_{t+1}, g_t)$ is more negative when using a larger batch size n, and suggest that the noise from a small mini-batch size facilitates exploration that may lead to better minima. Additionally, they also interpolate the loss between iterates of SGD to see if any significant loss barriers are crossed, i.e., interpolating the loss between θ_t and θ_{t+1} and observing a loss $\mathcal{L}(f(\cdot;\theta))$ higher than both $\mathcal{L}(f(\cdot;\theta_t))$ and $\mathcal{L}(f(\cdot;\theta_{t+1}))$. They observe that SGD never encounters any significant barrier, if the learning rate α is high enough. [Goodfellow and Vinyals, 2014] also observed that the exploration of SGD isn't hindered by the crossing of high-loss barriers.

Regarding the smoothness of the optimization trajectory, [Santurkar et al., 2018] notably showed that Batch Normalization [Ioffe and Szegedy, 2015], perhaps the most widely used normalization technique when training deep neural networks, has the effect of smoothing the loss landscape, as well as its gradients, along the SGD trajectory. More precisely, they show that the introduction of Batch Normalization increases the Lipschitz constant for the loss along the optimization trajectory, as well as for the gradient of the loss along the optimization trajectory. A function f is said to be *L-Lipschitz* if there is a positive real constant L such that

$$|f(\theta_t) - f(\theta_{t+1})| \leq L\|\theta_t - \theta_{t+1}\| \tag{1.9}$$

The advantages of Batch Norm were earlier believed to be caused by a control over the distributional shift in the layer's inputs, something that was referred to as *Internat Covariate Shift*, but that [Ioffe and Szegedy, 2015] showed not only wasn't affected by Batch Norm, but that there is actually no clear indication that this phenomenon is actually happening when training deep neural nets. They also showed that other normalization schemes, such as l_1 and l_2 regularization, cause this landscape smoothing.

Regarding optimization by full-batch gradient descent, [Lee et al., 2016] proved, using the theoretical framework of dynamical systems, that Gradient Descent converges to a local minimum, almost surely, when using random initialization. [Du et al., 2017] show that while GD almost always escapes saddle points asymptotically, it can be significantly slowed down by saddle points, taking exponential time to escape, even when using fairly natural random initialization schemes and dealing with non-pathological loss functions. However, they show that gradient descent with perturbations ([Jin et al., 2017]), isn't slowed down by saddle points, and can find an approximate local minimum in polynomial time, thus justifying the importance of adding perturbations when using gradient descent in non-convex optimization.

Dimensionality of trajectories With their very high number of trainable parameters, analyzing properties and patterns of the optimization trajectories of neural networks can be a very tedious task. Partly for this reason, techniques have been developed to help analyze and visualize the objective landscapes, along optimization trajectory, in a subspace of lower dimension. Moreover, it has been observed that deep neural networks are overparametrized for their learning problems, but that this overparametrization helps generalization [Freeman and Bruna, 2016], [Venturi et al., 2018], [Arora et al., 2018], [Zhang et al., 2016], [Safran and Shamir, 2017], [Garipov et al., 2018], [Draxler et al., 2018]. By overparametrization we mean that there are more parameters than required for obtaining a training minimum/solution. This fueled further inquiry into the actual dimensionality of the optimization process, and whether neural networks actually explore a smaller subspace of the parameter space when they are trained. See Figure 1.5 for an illustrative example of the projection of a parameter update $\Delta\theta$ from a 3-dimensional parameter space, onto a 2-dimensional subspace Ψ.

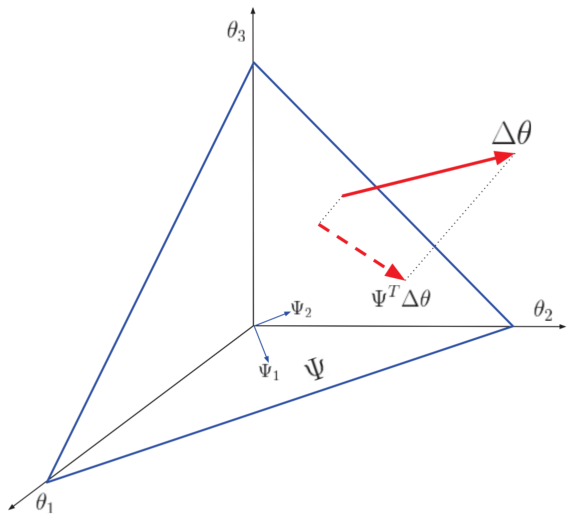


Figure 1.5 – Illustrative example of the projection of a parameter update $\Delta\theta$ from a 3-dimensional parameter space, onto a 2-dimensional subspace Ψ.

One popular technique to visualize the optimization trajectory uses Principal Component Analysis (PCA) and project the trajectory in the subspace spanned by its first few principal components. In this technique, the trajectory iterates are saved in a data matrix, on which is PCA is then performed, obtaining a set of principal components made of their eigenvectors and corresponding eigenvalues. The trajectory is then projected into the subspace spanned by the lowest PCA components.

For the PCA, each parameter iterate θ_t is stored as a row in the data matrix X, such that

$$X = \begin{bmatrix} {\theta_0}^T \\ {\theta_1}^T \\ \vdots \\ \tilde{\theta}^T \end{bmatrix}$$

for a trajectory of T steps ($\tilde{\theta} = \theta_T$). We then subtract the mean vector θ_μ (computed colomn-wise) from each row of X, obtaining the matrix $\hat{X}$. We then compute the covariance matrix $C = \hat{X}^T \hat{X}$, then compute the eigenvectors and eigenvalues of C to obtain the principal components of the optimization trajectory. Those component are often approximated with power iteration.

Exploiting this technique, [Li et al., 2017] observed that most of the variance of the SGD optimization trajectory is in the first two PCA components, meaning that the optimization trajectory actually lives in a subspace of very low dimension.

Considering the theory behind this visualization technique, [Antognini and Sohl-Dickstein, 2018] show that when projecting the trajectory of SGD for deep neural networks on its first few principal components, we qualitatively obtain the same projections as for an discrete Ornstein-Uhlenbeck process in high dimension, namely a random walk in high dimension, in a quadratic potential (but instead of being mean-reverting, here for the case of high dimensions and a long trajectory, the authors proved that the mean of the walk eventually gets trapped at a fixed distance from the origin of the walk).

[Li et al., 2018] take a different approach towards the dimensionality of the training of neural networks. Rather than measuring a so-called intrinsic dimension on the optimization trajectory a posteriori, they show that for deep neural networks, this optimization can be projected to a random subspace of the parameter space, of much lower dimensionality, thus constrained in its exploration, while attaining generalization performances similar to the unconstrained optimization. With $\theta \in \mathbb{R}^d$, they initialize the network to $\theta_0^{(d)}$ in the original parameter space, then generate a random projection matrix P to project in subspace of dimension D where $D < d$, such that $P \in \mathbb{R}^{d \times D}$. They thus train f in the random subspace, and obtain the equivalent parametrization in $\mathbb{R}^d$ such that:

$$\theta^{(d)} = \theta_0^{(d)} + P\theta^{(D)} \tag{1.10}$$

Note that after initialization, $\theta_0^{(d)}$ and P are fixed, and the columns of P are normalized so that steps of unit length of $\theta^{(D)}$ correspond to steps of unit length of $\theta^{(d)}$. One important fact that they also report is that, as they want to achieve generalization, for the optimization with intrinsic dimensionality, that approaches that of the standard optimization, this intrinsic dimensionality grows to eventually reach that of the whole parameter space.

1.2.3 General properties of the objective landscapes

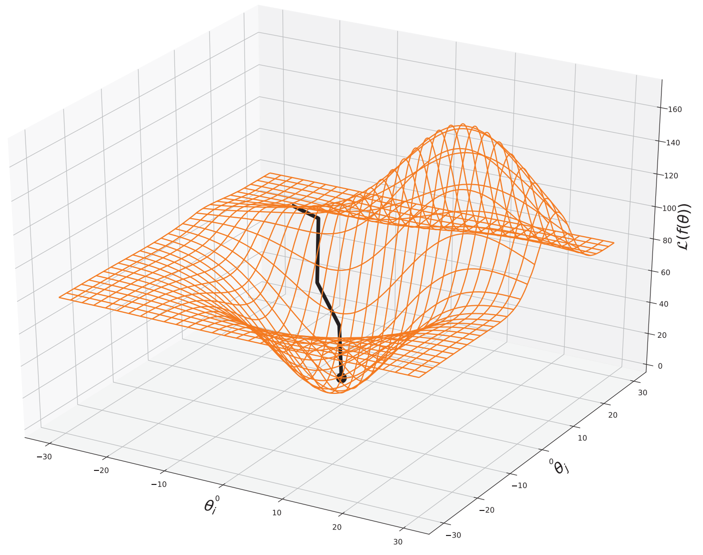


Figure 1.6 – General properties of the objective landscapes

In this subsection, we finally review the body of work which is concerned with properties of the landscapes more generally across the parameter space, and not necessarily only where lies the point of convergence nor the optimization trajectory.

We first review the works dealing with the non-convexities of the objective landscapes globally and what makes them smoother. Second, we address the works that are concerned with properties of the basins of attraction, or the regions of parameter space where solutions lie, with topics including the loss barriers between different solutions. Finally, we cover the works that deal with the question of whether minima present in the objective landscapes are local or global minima.

Non-convexities of the objective landscape Training neural networks involves optimizing high dimensional non-convex objective loss functions. These non-convexities may influence the quality of solutions found by the optimization and their generalization properties, but also the characteristics of the optimization trajectories between initialization and final solution.

However, [Goodfellow and Vinyals, 2014] empirically showed that when interpolating the loss landscape between the initialization θ_0 and final solution $\tilde{\theta}$, the loss decreases monotonically and is roughly convex, meaning that if this initial direction was known, the final solution could be found via a coarse line search.

Introducing their technique for visualizing objective landscapes of deep neural networks, [Li et al., 2017] observed the effect of different hyperparameter and architecture choices, on landscape convexity. They observe that as network depth increases, the loss surface transitions from nearly convex to chaotic, having many non-convexities. When analyzing the effect of network width, by varying the number of filters per convolution layer, they observe that wider networks, hence more overparametrized, have loss landscapes with significantly less chaotic behavior, resulting in flat minima and wide regions of apparent convexity. They further analyze the effect of skip connection [He et al., 2015], which are intended to alleviate the problems of vanishing gradient and exploding gradient in deep architectures and thus facilitate their optimization. They show that the introduction of skip connections resulted in significantly smoother loss surfaces for deep networks.

In a theoretical work where they study Gaussian random fields, [Bray and Dean, 2007] show that the eigenvalues of the hessian at a critical point are distributed according to a Wigner semicircle distribution, except that the spectrum is shifted by an amount dependent on the loss. In one dimension, the Wigner probability distribution is supported on the interval $[-R, R]$ and the probability density function f is a semicircle of radius R centered at the origin, and suitably normalized, such that:

$$f(x) = \frac{2}{\pi R^2}\sqrt{R^2 - x^2} \ .$$

The authors show that for the global minimum, the spectrum of eigenvalues is shifted so that there are no negative eigenvalues. As the loss increases, the spectrum shifts resulting in more null eigenvalues and negative eigenvalues (i.e., as the loss increases, the index increases). In a later work, [Dauphin et al., 2014] showed that this theory qualitatively holds for the loss of neural networks, where the spectrum of eigenvalues of the hessian of the loss shifts to the left when the loss increases.

Studying theoretically the geometry of the sublevel sets in the objective landscapes of deep neural networks, [Freeman and Bruna, 2016] shed light on fundamental topological differences of those landscapes, between linear and nonlinear networks. For deep linear networks, the sublevel sets are connected while for deep nonlinear networks, they are not connected. Connected sublevel sets imply that optimizers converge to a global minimum. However, the maximum energy of the barrier (uphill loss) decreases with increasing overparameterization. Their finding provided some theoretic argument supporting the emerging realization that overparametrization helps generalization.

Properties of basins of attraction Here we cover some of the different works that analyzed the properties of the objective landscapes in their basins of attractions, or regions of the landscapes where solutions lie.

Simultaneously, and independently, [Garipov et al., 2018] and [Draxler et al., 2018] discovered that for minima found by SGD, despite the loss landscape of deep neural networks being complex high-dimensional non-convex functions, their minima are in fact connected by simple paths along which the loss remains nearly constant, thus having no significant loss barriers along those paths. They show this to be true for the training loss, as well as for the test loss. Their work extends the work of [Freeman and Bruna, 2016], which had built their theory for shallow networks of one hidden layer with ReLU activations, and showed their result to hold experimentally for a CNN architecture on the MNIST dataset, and an LSTM architecture [Hochreiter and Schmidhuber, 1997b] on the Penn Treebank dataset (PTB, a widely used dataset in Natural Language Processing, or NLP) [Marcus et al., 1993]. However, [Freeman and Bruna, 2016] didn't deal with more practical architectures. Here [Garipov et al., 2018] and [Draxler et al., 2018] showed their results to hold experimentally with practical neural architectures, such as ResNet [He et al., 2015], on the more challenging problem of classifying the Cifar10 dataset.

[Sagun et al., 2017] study the curvature of minima residing in the low loss basins of attraction that SGD leads to. More precisely, they inspect the spectrum of eigenvalues of the hessian matrix of the loss. They notably show that the spectrum is essentially made of two parts: 1) the bulk of the spectrum is made of eigenvalues around zero, and 2) a few eigenvalues of high magnitude (named outliers), with minima being essentially flat in almost all directions, except for a very few directions of relatively high curvature. They notably show that overparametrization scales the bulk of low eigenvalues, while increasing the complexity of the data (making it less separable) increases the number of high eigenvalues or outliers (shifts the spectrum to the left). They also show that small batch and large batch methods (between online stochastic gradient descent where $n = 1$ to full batch gradient descent) both lead to the same number of outliers, but those outliers have larger magnitudes for large batch methods. Finally, their results motivate rethinking our interpretations of basins of attraction, as they observe that small and large batch gradient descent appear to converge to different basins of attraction, yet they appear in fact connected through their flat region, thus belonging the same basin.

Minima: local vs global Finally, one important question that has been of interest to the machine learning community is the existence of local minima versus global minima, in the optimization landscapes of neural networks.

Developing theory for deep linear networks of arbitrary depth and arbitrary width, trained to minimize a squared loss function, [Kawaguchi, 2016] proved that the loss function is non-convex and non-concave, that every local minimum is a global minimum, and every critical point that is not a global minimum is a saddle point. They also prove that there exist ”bad” saddle points, when models are deep enough (more than 3 layers), in the sense that their hessian has no negative eigenvalue, and thus optimization gets trapped, even when using second-order methods, whereas, for shallower networks, there are no such bad saddle points.

Studying theoretically the non-convex loss landscape of fully-connected ReLU networks (nonlinear), [Safran and Shamir, 2015] identify ”favorable conditions for optimization”, namely those 1) favoring the probability of (randomly) initializing at a point θ_0 from which there is a monotonically decreasing path to a global minimum

$\tilde{\theta}$; and 2) high probability of initializing at a basin of low loss. Their results converge to the fact that overparametrization is beneficial for those properties.

[Safran and Shamir, 2017] consider the optimization of a simple yet non-trivial nonlinear network, a two-layered ReLU network trained with a squared loss for a regression problem, in the form:

$$\mathbf{x} \mapsto \sum_{i=1}^{k} \max\{0, {\mathbf{w}_i}^T \mathbf{x}\}$$

Proposing a computer-assisted proof, they show, for an arbitrary network width k, that even if the input is distributed by a standard Gaussian, and even if the targets are generated with a network of the same architecture and dimension (that maps $\mathbf{x}$ to targets, so that the two-layered ReLU network can be the same function as the data generating function), the objective landscape of such network can still have spurious local minima, when the network width is $6 \leq k \leq 20$ (both for the discriminator and the network generating the data). Moreover, they show that the probability of hitting such local minimum increases with network width. On the other hand, they show that overparametrization (namely when k is greater for the discriminator network, i.e. the one being trained, than for the generator network), reduces drastically the presence of such local minima.

In another theoretical contribution, [Zhou and Feng, 2018] study the convergence behavior of the empirical risk to its population counterpart (the loss coming from the unknown, true data generating distribution). They study this convergence rate for the stationary points of the loss, the gradients of loss, and the loss landscape itself, for deep linear neural networks minimizing a squared loss and trained with SGD. They then show similar results for deep nonlinear networks with sigmoid activations. The authors show that the depth of the architecture, the number of nonzero weights (their sparsity), the norm of the parameter vector, the total number of weights, and the width of a network are critical to the convergence rates. One of their main result is that, as long as the training dataset size $|\mathcal{D}^{train}|$ is sufficiently large, any stationary point of the empirical risk, i.e., where $\nabla_\theta \mathcal{L}(f(\mathcal{D}^{train}; \theta)) \leq \epsilon$, is also a stationary point of the population risk, a one-to-one correspondence which they then prove.

As another work supporting overparametrization as a critical factor helping generalization, [Nguyen and Hein, 2017] analyze a special case of overparametrized deep neural networks, for a fully connected network with squared loss, with analytical activation functions. In their approach, they show that if there exists a very wide layer where the number of hidden units is larger than the number of samples in the training dataset, then almost all local minima are globally optimal.

1.3 Meta-Learning

1.3.1 Introduction

Most of the recent successes of deep learning, whether in supervised learning, unsupervised learning, or reinforcement, implied a system that we require to master only a single task. However, real-world situations would require an AI system to being able to adapt what it has previously learned, to its changing environment. Moreover, adaptation and versatility are considered by some to be the next necessary step, for machine learning algorithms, towards achieving general artificial intelligence. These notions call for machine learning algorithms to not only having the faculty to learn but also, that of "learning to learn", which we equivalently refer to as "meta-learning".

1.3.2 Early Works

For machine learning researchers, the topic of meta-learning has been of interest since the late 1980s, with Jürgen Schmidhuber's master's thesis being among the first works addressing it, within the framework of genetic programming [Schmidhuber, 1987]. Subsequent works explored using networks that learn how to modify their own weights ([Schmidhuber, 1992], [Schmidhuber, 1993]). The updating of the weights is defined in a parametric form that allows the prediction and weight-change process to be differentiable end-to-end.

The work of [Bengio et al., 1991] and [Bengio et al., 1995] and [Bengio, 1993] considered learning update rules for neural networks that are biologically plausible. This property is enforced by allowing the parametric form of the update to only have, as input, local information at each hidden unit to determine the weight change. Different optimization methods, such as genetic programming or simulated annealing, are used to train the learning rule.

1.3.3 Generalizing from few-examples: the next step for deep learning towards general artificial intelligence

While the topic of meta-learning remained of interest for the ML research community, the numerous successes that occurred over the recent years from the deep learning approaches, mostly in the first half the 2010s, primarily concerned models learning a single task. And while deep learning approaches achieved impressive generalization performances, the issue of sample efficiency, namely the need for vast amounts of labeled data to train deep neural networks, became an apparent drawback of deep learning.

The work of [Lake et al., 2015] later challenged modern machine learning methods to be able to learn new concepts from one or a few instances of that concept, with the example of models being able to correctly recognize and classify a new class of images, after having learned that class with only one example image. The authors pointed out the fact that people, when learning new concepts, can often generalize successfully from just a single example, while machine learning algorithms typically require tens or hundreds of examples to perform with similar accuracy. This work introduced the Omniglot dataset, which consists of 1623 handwritten characters classes from 50 alphabets, with each class comprising 20 examples. The Omniglot dataset thus became the de facto equivalent of MNIST for few-shot learning, and remained one of the two most widely used benchmark datasets of meta-learning, especially in the image classification domain.

The authors proposed to tackle this few-shot learning problem using Bayesian program learning, which is able to learn a hierarchical, non-parametric generative model of handwritten characters. Their approach incorporates specific knowledge of how strokes are formed and how they are combined to produce characters of different types. The model represents concepts as simple programs that best explain observed examples under a Bayesian criterion. On a challenging one-shot classification task, the model achieves human-level performance while outperforming recent deep learning approaches.

Subsequently, the work of [Rezende et al., 2016] proposed deep generative models for one-shot generalization, that combine the representational power of deep neural networks, with Bayesian reasoning for inference.

1.3.4 Definition of few-shot learning

In few-shot learning for classification, we have a distribution over classification tasks $p(\mathcal{T})$, and a model f (parametrized by θ), that must learn to adapt to tasks $\mathcal{T}_i$ sampled from $p(\mathcal{T})$. The model is trained on a set of training tasks $\{\mathcal{T}_i\}^{train}$ and evaluated on a set of testing tasks $\{\mathcal{T}_i\}^{test}$, all drawn from $p(\mathcal{T})$, and $\{\mathcal{T}_i\}^{train}$ and $\{\mathcal{T}_i\}^{test}$ using disjoint sets of classes to constitute their tasks. In the setting of k-shot learning, when f adapts to a task $\mathcal{T}_i^{test}$, it only has access to a set of few support samples $\mathcal{D}_i = \{(\mathbf{x}_i^{(1)}, \mathbf{y}_i^{(1)}), ..., (\mathbf{x}_i^{(k)}, \mathbf{y}_i^{(k)})\}$ drawn from $\mathcal{T}_i^{test}$. We then evaluate the model's performance on $\mathcal{T}_i^{test}$ using a new set of target samples $\mathcal{D}'_i$.

In this context, we define generalization as the model's ability to reach high accuracy on a testing task $\mathcal{T}_i^{test}$, evaluated with a set of target samples $\mathcal{D}'_i$, for several testing tasks. In most cases, after training is complete, the model f will adapt to the test tasks $\mathcal{T}_i^{test}$ using their support sets $\mathcal{D}_i$ before we evaluate the model on their target sets $\mathcal{D}_i$. We thus care about the average accuracy

$$\mathbb{E}_{\mathcal{T}_i^{test} \sim p(\mathcal{T})}[Acc(f(\mathcal{D}'_i; \tilde{\theta}_i^{test})] \tag{1.11}$$

The different approaches to few-shot learning can vary in how they perform the training phase, but also in how they adapt to the new tasks at meta-test time.

1.3.5 Learning a metric space

With the meta-learning application of few-shot classification, discriminating among samples from new classes requires to quickly learn what differentiates those new classes in order to generalize well. One may thus wonder, instead of simply learning a fixed classification dataset, could we train the models to *learn to classify*? One approach to this is to learn a metric space in which classification or discrimination among samples from different classes, when embedded in that space, can be easily computed by means of a distance, or metric.

One good example of this approach is the Prototypical Network proposed by [Snell et al., 2017]. The central idea is that there exists an embedding space in which points cluster around a single prototype representation of their class. A neural network learns a non-linear mapping, from the input space to that embedding space:

$$f_\phi : \mathbb{R}^P \to \mathbb{R}^M \tag{1.12}$$

with input samples originally being $x \in \mathbb{R}^P$. In the space obtained by $f_\phi(\mathbf{x})$, we can then learn the prototype of a class m as the mean representation (cluster mean) c_m, having access of a set of samples $\mathcal{D}_m$ all belonging to class m:

$$\mathbf{c}_m = \frac{1}{|\mathcal{D}_m|} \sum_{(\mathbf{x}_i,\mathbf{y}_i)\in\mathcal{D}_m} f_\phi(\mathbf{x}_i) \tag{1.13}$$

Given a distance function $d : \mathbb{R}^M \times \mathbb{R}^M \to [0, +\infty)$, Prototypical Networks produce a distribution over classes for a query point $\mathbf{x}$ based on a softmax over distances to the prototypes in the embedding space:

$$p_\phi(y = m|\mathbf{x}) = \frac{\exp(-d(f_\phi(\mathbf{x}, \mathbf{c}_m)))}{\sum_{m'} \exp(-d(f_\phi(\mathbf{x}, \mathbf{c}_{m'})))} \tag{1.14}$$

In this work, the squared Euclidean distance $d(\mathbf{z}, \mathbf{z}') = \|\mathbf{z} - \mathbf{z}'\|^2$ is primarily used. Learning is performed by minimizing the negative log-probability $J(\phi) = -\log p_\phi(y = m|\mathbf{x})$ of the true class m via SGD. Training episodes are formed by randomly selecting a subset of classes from the training set, then choosing a subset of examples within each class to act as the support set and a subset of the remainder to serve as query points. At meta-test time, we form test tasks $\mathcal{T}_i^{test}$ each made of m new classes, with say k support samples per class, the m prototypes c_m are obtained by computing the class means in $f_\phi(\mathbf{x})$. To evaluate generalization (accuracy on target samples of the test tasks), class probabilities are computed with $p_\phi(y = m|\mathbf{x})$.

The training procedure, where tasks $\mathcal{T}_i^{train}$ are sampled with their few-shot support set $\mathcal{D}_i$ and their target set $\mathcal{D}_i'$, was based on the prior work of [Vinyals et al., 2016]. This training procedure has since been prevalent in few-shot classification literature. Their authors made two other significant contributions. First, they introduced the MiniImagenet dataset, a subset of Imagenet [Russakovsky et al., 2014] tailored for few-shot classification, having a total of 100 real image classes with 600 instances each. Mini-Imagenet and Omniglot have since been the two most widely used datasets in meta-learning for few-shot classification. The authors also proposed the Matching Network, which is also based on the idea of learning a metric space for few-shot classification and which inspired the approach for Prototypical Networks.

Another prominent approach to learning a metric space for few-shot classification was the work of [Koch et al., 2015], where the authors propose to use deep Siamese Networks, which train a convolutional network to embed examples so that items in the same class are close while items in different classes are far away, according to some distance metric. Siamese networks were first introduced in the early 1990s to solve signature verification as an image matching problem [Bromley et al., 1993]. A siamese neural network consists of twin networks $f_1(\mathbf{x};\theta_1)$ and $f_2(\mathbf{x};\theta_2)$, which accept distinct inputs $\mathbf{x}_1$ and $\mathbf{x}_2$ but are joined by an distance function $d(\mathbf{z}_1,\mathbf{z}_2)$ at the top. This function computes some metric between the highest level feature representations on each of the twin networks. The twin networks f_1 and f_2 have the same architecture and parameters are shared between them ($\theta_1 = \theta_2$), the whole model thus being symmetric and the two twins having the same parameter values at all time and are trained conjointly. With both networks computing the same function that maps inputs to outputs, two extremely similar images could not possibly be mapped by their respective networks to very different locations $\mathbf{z}_1$ and $\mathbf{z}_2$ in the feature space, thus allowing discrimination based on a distance computed in that feature space.

1.3.6 Learning to optimize

Neural networks are in practice, for the most part, trained through some stochastic optimization process. Thus, since the learning process is, in essence, an optimization process, one natural approach to *learning to learn* is to cast it as *learning to optimize*. [Hochreiter et al.] proposed the approach of *Learning to Learn Using Gradient Descent*, where a meta-learner model, parametrized by a recurrent neural (more precisely, a Long-Short Term Memory, or LSTM [Hochreiter and Schmidhuber, 1997b]), learns to derive learning algorithms that are able to approximate quadratic functions after seeing a relatively small amount of examples. More recently, and in a similar spirit, [Andrychowicz et al., 2016] show that the design of an optimization algorithm can be itself cast as a learning problem. The goal of training a machine learning model often being of minimizing a differentiable function $f(\theta)$ through gradient descent, with the resulting parameter update rule:

$$\theta_{t+1} = \theta_t - \alpha_t \nabla_\theta f(\theta_t) \tag{1.15}$$

the authors recast this parameter update as:

$$\theta_{t+1} = \theta_t + o_t(\nabla_\theta f(\theta_t), \phi) \tag{1.16}$$

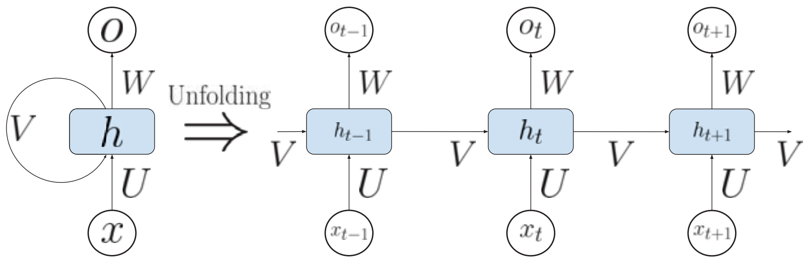


Figure 1.7 – Recurrent Neural Network

where o_t is the output of an *optimizer* model m, parametrized by ϕ, which learns the parameter update rule, and f is referred as the *optimizee* function, parametrized by θ, that we which to optimize. Here again, the optimizer, or meta-learner model, is a recurrent neural network, the approach building upon [Hochreiter et al.] by modifying the network architecture of the optimizer in order to scale this approach to larger neural network optimization problems. A recurrent network at time-step t takes an input vector x_t and outputs a vector o_t, just like a feed-forward network, but also has an internal state h_t which is a function of the current input x_t and of the previous internal state h_t, as illustrated in Figure 1.7.

The goal is thus to train the optimizer $m(\cdot;\phi)$ such that on expectation, for a given function f to be optimized, it leads to a final parameter vector $\tilde{\theta}$ that minimizes this function f. We can therefore see the loss of the optimizer as:

$$\mathcal{L}(m(\cdot\,;\phi)) = \mathbb{E}_f[f(\tilde{\theta})] \tag{1.17}$$

At every time step t, the recurrent network m has an internal state h_t that is a function of the current input $x_t = \nabla_\theta f(\theta_t)$, which is the current gradient of f, and of its previous internal state h_{t-1}, and m also outputs a vector o_t, which is the current parameter update, leading to the next parameter vector $\theta_{t+1} = \theta_t + o_t$. With an optimization trajectory of fixed horizon T (fixed number of time steps), the final parameter vector of f is thus:

$$\tilde{\theta} = \theta_0 + \sum_{t=1}^{T} o_t \tag{1.18}$$

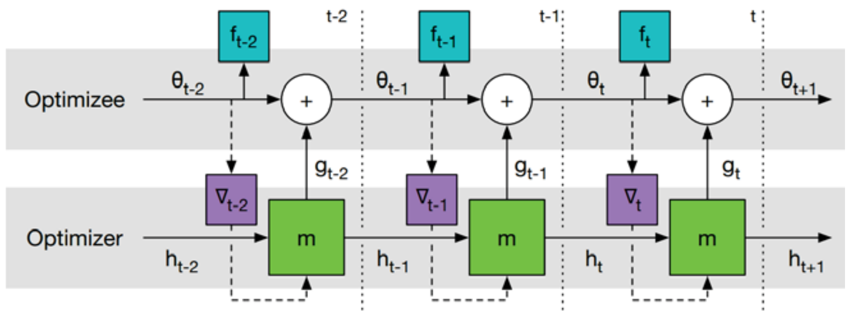


Figure 1.8 – Learning to Learn by Gradient Descent by Gradient Descent: Gradient flows backward through the solid arrows. Image from [Andrychowicz et al., 2016].

Instead of having the loss of the optimizer depend solely on the final parameters $\tilde{\theta}$, the authors make it depend on the parameters of the entire optimization trajectory, such that:

$$\mathcal{L}(\phi) = \mathbb{E}_f \left[\sum_{t=1}^{T} f(\theta_t) \right] \tag{1.19}$$

Thus for a given parametrization ϕ and an optimization trajectory of T steps of a randomly sampled function f, the optimizer model $m(\cdot;\phi)$ can thus be trained by performing gradient descent on the loss $\mathcal{L}(m(\cdot\,;\phi))$, where the gradient estimate $\frac{\partial \mathcal{L}(\phi)}{\partial \phi}$ is computed by Backpropagation Through Time (see Figure 1.8).

Building upon [Andrychowicz et al., 2016], who were interested in learning a general optimization algorithm to train neural networks for large-scale classification rather than in few-shot learning, the work [Ravi and Larochelle, 2017] revisits this approach in the context of few-shot classification. Here again, taking the parameter update rule of a model, parametrized by θ, such as the minimization of its empirical loss $\mathcal{L}$ through gradient descent:

$$\theta_t = \theta_{t-1} - \alpha_t \nabla_{\theta_{t-1}} \mathcal{L}_t \tag{1.20}$$

the authors make the key observation is that this update resembles the update for the cell state in an LSTM network:

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{1.21}$$

if $f_t = 1$, $c_{t-1} = \theta_{t-1}$, $i_t = \alpha_t$ and $\tilde{c}_t = -\nabla_{\theta_{t-1}} \mathcal{L}_t$, and where i_t is the input gate, f_t is the forget gate, c_{t-1} is the previous cell state of the LSTM cell, $\tilde{c}_t$ is the candidate

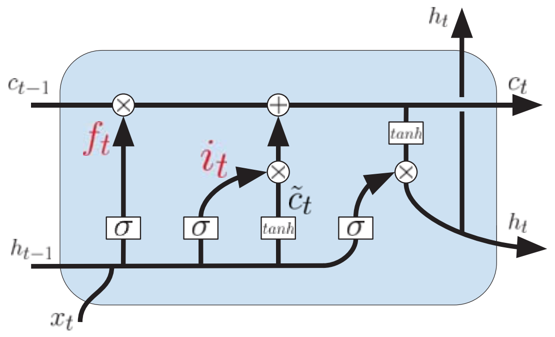


Figure 1.9 – Long-Short Term Memory (LSTM) cell

state, and c_t is the output and current cell state of the LSTM (see Figure 1.9 for illustration of the LSTM cell). They then define parametric forms for the input and forget gates i_t and f_t and make them more general, so that they are function of the previous parameters θ_{t-1}, the current loss and its gradient, as well as their respective previous values i_{t-1} and f_{t-1}, such that:

$$i_t = \sigma(\mathbf{W}_I \cdot [\nabla_{\theta_{t-1}} \mathcal{L}_t, \mathcal{L}_t, \theta_{t-1}, i_{t-1}] + \mathbf{b}_I) \tag{1.22}$$

$$f_t = \sigma(\mathbf{W}_F \cdot [\nabla_{\theta_{t-1}} \mathcal{L}_t, \mathcal{L}_t, \theta_{t-1}, f_{t-1}] + \mathbf{b}_F) \tag{1.23}$$

where $\mathbf{W}_I$ and $\mathbf{W}_F$ are the weight matrices of the input and forget gate respectively, while $\mathbf{b}_I$ and $\mathbf{b}_I$ are the bias vectors of the input and forget gate respectively.

1.3.7 Learning the hyperparameters

Another approach to meta-learning is to automate the optimization of hyperparameters and other design choices for the model, that are normally handmade by the researcher. In [Maclaurin et al., 2015], the authors propose to automatically learn hyperparameters of neural networks by computing exact gradients of cross-validation performance with respect to all hyperparameters. They achieve this by chaining the derivatives backwards through the entire training procedure, allowing to optimize for thousands of hyperparameters of networks trained by SGD with momentum [Nesterov, 1983], including the learning step-size α, scheduling of the momentum ρ, weight initialization distributions, as well as some aspects of architecture configuration. Those gradients are computing by exactly reversing the dynamics of SGD with momentum.

Pushing this approach further, [Zoph and Le, 2016] use a recurrent neural network to generate the description of the neural network architecture, and train this meta-learner RNN with reinforcement learning to maximize the expected validation accuracy of the generated models, on a set of validation tasks.

1.3.8 External memory, attention and temporal convolution

Other approaches to meta-learning making use of novel developments in the field of deep learning, including the use of external memory, attention mechanism, and temporal convolution, which have been recently proposed.

[Santoro et al., 2016] use a memory-augmented neural network that is trained to learn how to store and retrieve memories to use for each classification task. This approach builds upon the proposed Neural Turing Machine (NTM) of [Graves et al., 2014]. An NTM consists of a controller (modeled either as an MLP or LSTM) interacting with an external memory module via read and write heads. The encoding and retrieval of memory, stored as vector representation, is rapid, making an NTM a viable candidate for meta-learning in the few-shot regime, thanks to its ability for long term retention of memory via the slow updates of its weights, and the short term storage of new memories using its external memory module.

[Mishra et al., 2017] propose a meta-learning architecture called Simple Neural Attentive Meta-Learner (SNAIL), that use a combination of temporal convolutions [van den Oord et al., 2016] and soft attention [Vaswani et al., 2017]. The temporal convolution serves to aggregate information from past experience, to produce what is called a context. The soft attention mechanism pinpoints specific pieces of information, from a potentially infinitely-large context, which is treated as an unordered key-value store, which it can query based on the content of each element. The SNAIL architecture interleaves temporal convolution layers with soft attention layers, and being trained end-to-end, it learns how to pick up specific pieces of information from the experience it has accumulated and also learns a feature representation that is well suited for this selection of past experiences.

1.3.9 Gradient-Based Meta-Learning

The final approach that we cover is perhaps the closest analog to the standard regime of supervised learning of neural network with stochastic, gradient-based optimization. In *gradient-based meta-learning*, the aim is to conceive a training procedure for a model f that will favor generalization to new tasks, by finding a good parametrization θ (sometimes called a "good initialization"). Generally this approach also aims to be independent of any architecture choice or from learning a meta-learner that updates a learner model, but really views generalization to new tasks as the learning problem itself for which we train a model end-to-end, akin to supervised learning where the goal is to train the model to favor generalization to new data points.

One of the most natural ways to approach gradient-based meta-learning is to pretrain a model f on a task or dataset that shares similarities with the distribution of the target/test tasks, which we ultimately wish to learn with few examples and generalize well. For image classification, pre-training models on Imagenet (a large and diversified dataset of real images, comprising one thousand classes and more than five hundred samples per class) before finetuning the pre-trained model on the target tasks. [Donahue et al., 2013] have analyzed this approach for a multi-task framework with deep convolutional networks. They showed that the learned features, obtained from supervised image classification, have good representational power and can be adapted to vision tasks from other domains (e.g. scene recognition) while demonstrating good generalization.

More recently, [Finn et al., 2017] introduced the Model Agnostic Meta-Learning (MAML) algorithm. Inspired by the training procedure of [Vinyals et al., 2016], they directly train a model f to generalize on few-shot classification tasks $\mathcal{T}_i^{train}$. MAML learns a set of parameters θ such that on average, given a new task $\mathcal{T}_i^{test}$, only a few samples are required for f to learn and generalize well to that task. During a meta-training iteration s, where the current parametrization of f is θ^s, a batch of n training tasks is sampled from $p(\mathcal{T})$. For each task $\mathcal{T}_i$, a set of support samples $\mathcal{D}_i$ is drawn and f adapts to $\mathcal{T}_i$ by performing T steps of full batch gradient descent on $\mathcal{L}(f(\mathcal{D}_i;\theta))$ w.r.t. θ, obtaining the adapted solution $\tilde{\theta}_i$:

$$\tilde{\theta}_i = \theta^s - \alpha \sum_{t=0}^{T-1} \nabla_\theta \mathcal{L}(f(\mathcal{D}_i;\theta_i^{(t)})) \tag{1.24}$$

where $\theta_i^{(t)} = \theta_i^{(t-1)} - \alpha \nabla_\theta \mathcal{L}(f(\mathcal{D}_i; \theta_i^{(t-1)}))$ and all adaptations are independent and start from θ^s, i.e. $\theta_i^{(0)} = \theta^s, \forall i$. Then from each $\mathcal{T}_i$, a set of target samples $\mathcal{D}'_i$ is drawn, and the adapted meta-training solution θ^{s+1} is obtained by averaging the target gradients, such that:

$$\theta^{s+1} = \theta^s - \beta \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta \mathcal{L}(f(\mathcal{D}'_i; \tilde{\theta}_i)) \ . \tag{1.25}$$

As one can see in Eq.1.24 and Eq.1.25, deriving the meta-gradients implies computing second-order derivatives, which can come at a significant computational expense. The authors introduced a first-order approximation of MAML, where these second-order derivatives are omitted:

$$\begin{aligned} \nabla_\theta \mathcal{L}(f(\mathcal{D}'_i; \tilde{\theta}_i)) &= \frac{\partial}{\partial \tilde{\theta}_i} \mathcal{L}(f(\mathcal{D}'_i; \tilde{\theta}_i)) \frac{\partial}{\partial \theta} \tilde{\theta}_i \\ &= \left(\frac{\partial}{\partial \tilde{\theta}_i} \mathcal{L}(f(\mathcal{D}'_i; \tilde{\theta}_i)) \right)^T \left(\mathbb{1} - \alpha \sum_{t=0}^{T-1} \cancelto{0}{\nabla_\theta^2} \mathcal{L}(f(\mathcal{D}_i, \theta_i^{(t)})) \right) \\ &= \frac{\partial}{\partial \tilde{\theta}_i} \mathcal{L}(f(\mathcal{D}'_i; \tilde{\theta}_i)) \end{aligned} \tag{1.26}$$

and we refer to that other algorithm as First-Order MAML. As another first-order approximation of MAML, [Nichol et al., 2018] proposed Reptile, where instead of computing losses on target samples before computing the meta-gradient, the meta-gradient is simply the average of the individual adaptation trajectory displacement vectors:

$$\theta^{s+1} = \theta^s + \beta \frac{1}{n} \sum_{i=1}^{n} (\tilde{\theta}_i - \theta^s) \tag{1.27}$$

where $\tilde{\theta}_i$ is obtained just like in equation 2.1, and here we don't sample any set of target samples $\mathcal{D}'_i$.

Also building on top of MAML, [Zintgraf et al., 2018] proposed CAML, a meta-learning method for few-shot adaptation to new tasks. In this approach, the model parameters are partitioned into two parts. The first part consists of the *shared parameters* θ which are meta-trained using the meta-training tasks $\mathcal{T}_i^{train}$ and shared across tasks, and ϕ can be viewed as the trunk of the network. The second part of the network consists of the *context parameters* ϕ, which are adapted on individual tasks, and at test time, they are updated by performing a few gradient descent steps on a task-specific loss that is backpropagated through the shared part of the network. In a meta-training iteration, we first learn in the inner loop, task-specific context parameters, for each task in a batch:

$$\tilde{\phi}_i = \phi_0 - \alpha \sum_{t=0}^{T-1} \nabla_\phi \mathcal{L}(f(\mathcal{D}_i; \theta^s, \phi_i^{(t)})) \tag{1.28}$$

where ϕ_0 is usually chosen to be $\mathbf{0}$. We then compute the meta-update by updating the shared parameters:

$$\theta^{s+1} = \theta^s - \beta \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta \mathcal{L}(f(\mathcal{D}_i'; \theta^s, \tilde{\phi}_i)) \tag{1.29}$$

The work [Finn et al., 2017] has also been reformulated as a method for probabilistic inference in a hierarchical Bayesian model, called LLAMA [Grant et al., 2018]. MAML has also been recasted as a probabilistic meta-learning algorithm for few-shot classification tasks [Finn et al., 2018]. In addition, various recent work have built on top of MAML for other meta-learning paradigms, with notable examples in topics such as meta-reinforcement learning [Rothfuss et al., 2018] and continual learning [Finn et al., 2019].

Because of the simplicity of the general approach of gradient-based meta-learning and of its similarity to supervised learning through stochastic gradient-based optimization, we selected this approach to meta-learning in the work that we present in Chapter 2. We selected Model Agnostic Meta-Learning as our base meta-learning algorithm, because it also bears strong resemblance to traditional supervised learning. Moreover, we also selected this algorithm because of the major impact it has had on the field, its generalization performance for few-shot learning, as well of its independence of the underlying architecture. We selected the original MAML formulation since many subsequently proposed algorithms were built on it. Along with MAML we also analyze its first-order variant, as well as the finetuning approach (where the model is pre-trained in a supervised way, which was covered at the beginning of this section) since it is the most natural baseline algorithm to compare with the other meta-learning algorithms, and to analyze the fundamental differences in properties of objective landscapes.

2 On the Properties of the Objective Landscapes and Generalization of Gradient-Based Meta-Learning

Authors: Simon Guiroy, Vikas Verma, Christopher Pal

This chapter presents a joint work with Vikas Verma and Christopher Pal. It is submitted to Neural Information Processing Systems (NeurIPS 2019) - Conference Track.

Contribution: Simon Guiroy: First author. Work done primarily on an individual basis. Elaborated the research question, literature review, analyses, and experiments to run, wrote the code, ran the experiments, wrote the paper. Vikas Verma: Helped with the writing of the paper, participated in discussions, provided advice, ran some experiments. Christopher Pal: Helped with the writing of the paper, participated in discussions and provided advice regarding directions for the project and its research questions and analyses.

Affiliation

— Simon Guiroy, Mila, University of Montreal

— Vikas Verma, Mila, Aalto University Finland

— Christopher Pal, Mila, University of Montreal, Polytechnique Montreal, ElementAI Montreal

2.1 Introduction

To address the problem of the few-shot learning, many meta-learning approaches have been proposed recently [Finn et al., 2017], [Ravi and Larochelle, 2017], [Rothfuss et al., 2018], [Oreshkin et al., 2018] and [Snell et al., 2017] among others. In this work, we take steps towards understanding the characteristics of the objective landscapes of the loss functions, and their relation to generalization, in the context of gradient-based few-shot meta-learning. While we are interested in understanding the properties of optimization landscapes that are linked to generalization in gradient-based meta-learning in general, we focus our experimental work here within a setup that follows the recently proposed Model Agnostic Meta-Learning (MAML) algorithm [Finn et al., 2017]. The MAML algorithm is a good candidate for studying gradient-based meta-learning because of its independence from the underlying network architecture.

Our main insights and contributions can be summarized as follows:

1. As gradient-based meta-training progresses:
   — the adapted meta-test solutions become flatter on average, while the opposite occurs when using a finetuning baseline.
   — the adapted final solutions reach lower average support loss values, which never increases, while the opposite occurs when using a finetuning baseline.

2. When generalization starts to degrade due to overtraining, meta-test solutions keep getting flatter, implying that, in the context of gradient-based meta-learning, flatness of minima is not correlated with generalization to new tasks.

3. We empirically show that generalization to new tasks is correlated with the coherence between their adaptation trajectories, measured by the average cosine similarity between trajectory directions. Also correlated with generalization is the coherence between meta-test gradients, measured by the average inner product between meta-test gradient vectors evaluated at meta-train solution.

4. Based on this observation on coherence of adaptation trajectories , we propose a novel regularizer for gradient-based meta-learning and experimentally demonstrate its effectiveness in regularizing MAML.

2.2 Related work

There has been extensive research efforts on studying the optimization landscapes of neural networks in the standard supervised learning setup. Such work has focused on the presence of saddle points versus local minima in high dimensional landscapes [Pascanu et al., 2014],[Dauphin et al., 2014], the role of overparametrization in generalization [Freeman and Bruna, 2016], loss barriers between minima and their connectivity along low loss paths, [Garipov et al., 2018] ; [Draxler et al., 2018], to name a few examples.

One hypothesis that has gained popularity is that the flatness of minima of the loss function found by stochastic gradient-based methods results in good generalization, [Hochreiter and Schmidhuber, 1997a] ; [Keskar et al., 2016]. [Xing et al., 2018] and [Li et al., 2017] measure the flatness by the spectral norm of the hessian of the loss, with respect to the parameters, at a given point in the parameter space. Both [Smith and Le, 2017] and [Jastrzebski et al., 2017] consider the determinant of the hessian of the loss, with respect to the parameters, for the measure of flatness. For all of the work on flatness of minima cited above, authors have found that flatter minima correlate with better generalization.

In contrast to previous work on understanding the objective landscapes of neural networks in the classical supervised learning paradigm, in our work, we explore the properties of objective landscapes in the setting of gradient-based meta-learning.

2.3 Gradient-based meta-learning

We consider the meta-learning scenario where we have a distribution over tasks $p(\mathcal{T})$, and a model f parametrized by θ, that must learn to adapt to tasks $\mathcal{T}_i$ sampled from $p(\mathcal{T})$. The model is trained on a set of training tasks $\{\mathcal{T}_i\}^{train}$ and evaluated on a set of testing tasks $\{\mathcal{T}_i\}^{test}$, all drawn from $p(\mathcal{T})$. In this work we only consider classification tasks, with $\{\mathcal{T}_i\}^{train}$ and $\{\mathcal{T}_i\}^{test}$ using disjoint sets of classes to constitute their tasks. Here we consider the setting of k-shot learning, that is, when f adapts to a task $\mathcal{T}_i^{test}$, it only has access to a set of few support samples $\mathcal{D}_i = \{(\mathbf{x}_i^{(1)}, \mathbf{y}_i^{(1)}), ..., (\mathbf{x}_i^{(k)}, \mathbf{y}_i^{(k)})\}$ drawn from $\mathcal{T}_i^{test}$. We then evaluate the model's performance on $\mathcal{T}_i^{test}$ using a new set of target samples $\mathcal{D}'_i$. By gradient-based meta-learning, we imply that f is trained using information about the gradient of a certain loss function $\mathcal{L}(f(\mathcal{D}_i; \theta))$ on the tasks. Throughout this work the loss function is the cross-entropy between the predicted and true class.

2.3.1 Model-Agnostic Meta-Learning (MAML)

MAML learns an initial set of parameters θ such that on average, given a new task $\mathcal{T}_i^{test}$, only a few samples are required for f to learn and generalize well to that task. During a meta-training iteration s, where the current parametrization of f is θ^s, a batch of n training tasks is sampled from $p(\mathcal{T})$. For each task $\mathcal{T}_i$, a set of support samples $\mathcal{D}_i$ is drawn and f adapts to $\mathcal{T}_i$ by performing T steps of full batch gradient descent on $\mathcal{L}(f(\mathcal{D}_i; \theta))$ w.r.t. θ, obtaining the adapted solution $\tilde{\theta}_i$:

$$\tilde{\theta}_i = \theta^s - \alpha \sum_{t=0}^{T-1} \nabla_\theta \mathcal{L}(f(\mathcal{D}_i; \theta_i^{(t)})) \tag{2.1}$$

where $\theta_i^{(t)} = \theta_i^{(t-1)} - \alpha \nabla_\theta \mathcal{L}(f(\mathcal{D}_i; \theta_i^{(t-1)}))$ and all adaptations are independent and start from θ^s, i.e., $\theta_i^{(0)} = \theta^s, \forall i$. Then from each $\mathcal{T}_i$, a set of target samples $\mathcal{D}'_i$ is drawn, and the adapted meta-training solution θ^{s+1} is obtained by averaging the target gradients, such that:

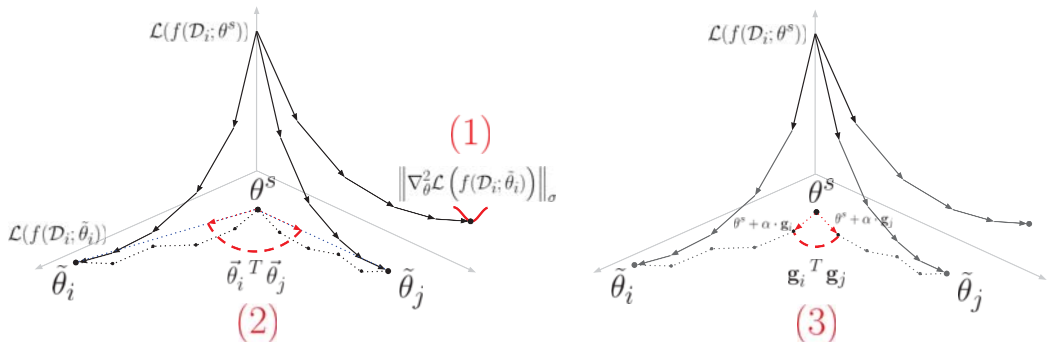
$$\theta^{s+1} = \theta^s - \beta \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta \mathcal{L}(f(\mathcal{D}'_i; \tilde{\theta}_i)) \tag{2.2}$$

As one can see in Eq.2.1 and Eq.2.2, deriving the meta-gradients implies computing second-order derivatives, which can come at a significant computational expense. The authors introduced a first-order approximation of MAML, where these second-order derivatives are ommited, and we refer to that other algorithm as First-Order MAML.

2.3.2 Finetuning baseline

For the finetuning baseline, the model is trained in a standard supervised learning setup: the model is trained to classify all the classes from the training split using a stochastic gradient-based optimization algorithm, its output layer size being equal to the number of meta-train classes. During evaluation on meta-test tasks, the model's final layer (fully-connected) is replaced by a layer with the appropriate size for the given meta-test task (e.g. if 5-way classification, the output layer has five logits), with its parameter values initialized to random values or with another initialization algorithm, then all the model parameters are optimized to the meta-test task, just like for the other meta-learning algorithms.

2.4 Analyzing the objective landscapes



$\mathcal{L}(f(\mathcal{D}_i;\theta^s))$
(1)
$\left\|\nabla^2_\theta \mathcal{L}\left(f(\mathcal{D}_i;\tilde{\theta}_i)\right)\right\|_\sigma$
θ^s
$\mathcal{L}(f(\mathcal{D}_i;\tilde{\theta}_i))$
$\tilde{\theta}_i$
$\vec{\tilde{\theta}}_i^{\,T}\vec{\tilde{\theta}}_j$
$\tilde{\theta}_j$
(2)
$\mathcal{L}(f(\mathcal{D}_i;\theta^s))$
θ^s
$\theta^s+\alpha\cdot\mathbf{g}_i$
$\theta^s+\alpha\cdot\mathbf{g}_j$
$\tilde{\theta}_i$
$\mathbf{g}_i{}^T\mathbf{g}_j$
$\tilde{\theta}_j$
(3)

Figure 2.1 – Visualizations of metrics measuring properties of objective loss landscapes. The black arrows represent the descent on the support loss and the dotted lines represent the corresponding displacement in the parameter space. (1): Curvature of the loss for an adapted meta-test solution $\tilde{\theta}_i$ (for a task $\mathcal{T}_i$), is measured as the spectral norm of the hessian matrix of the loss. (2): Coherence of adaptation trajectories to different meta-test tasks is measured as the average cosine similarity for pairs of trajectory directions. A direction vector is obtained by dividing a trajectory displacement vector (from meta-train solution θ^s to meta-test solution $\tilde{\theta}_i$) by its Euclidean norm, i.e. $\vec{\tilde{\theta}}_i = (\tilde{\theta}_i - \theta^s)/\|\tilde{\theta}_i - \theta^s\|_2$. (3): Characterizing a meta-train solution by the coherence of the meta-test gradients, measured by the average inner product for pairs of meta-test gradient vectors $\mathbf{g}_i = -\nabla_\theta \mathcal{L}(f(\mathcal{D}_i;\theta^s))$.

In the context of gradient-based meta-learning, we define generalization as the model's ability to reach a high accuracy on a testing task $\mathcal{T}_i^{test}$, evaluated with a set of target samples $\mathcal{D}'_i$, for several testing tasks. This accuracy is computed after f, starting from a given meta-training parametrization θ^s, has optimized its parameters to the task $\mathcal{T}_i^{test}$ using only a small set of support samples $\mathcal{D}_i$, resulting in the adapted solution $\tilde{\theta}_i^{test}$ (minima). We thus care about the average accuracy $\mathbb{E}_{\mathcal{T}_i^{test}\sim p(\mathcal{T})}[Acc(f(\mathcal{D}'_i;\tilde{\theta}_i^{test})]$. With these definitions in mind, for many meta-test tasks $\mathcal{T}_i^{test}$, we consider the optimization landscapes $\mathcal{L}(f(\mathcal{D}_i;\theta))$, and the properties of these loss landscapes evaluated at the solutions $\tilde{\theta}_i^{test}$; the adaptation trajectories when f, starting from θ^s, adapts to those solutions ; as well as properties of those landscapes evaluated at the meta-train solutions θ^s. See Figure 2.1 for a visualization of our different metrics. We follow the evolution of the metrics as meta-training progresses: after each epoch, which results in a different parametrization θ^s, we adapt f to several meta-test tasks, compute the metrics averaged over those tasks, and compare with $\mathbb{E}[Acc(f(\mathcal{D}_i'; \tilde{\theta}_i^{test})]$. We do not deal with the objective landscapes involved during meta-training, as this is beyond the scope of this work. From here on, we drop the superscript *test* from our notation, as we exclusively deal with objective landscapes involving meta-test tasks $\mathcal{T}_i$, unless specified otherwise.

2.4.1 Flatness of minima

We start our analysis of the objective loss landscapes by measuring properties of the landscapes at the adapted meta-test solutions $\tilde{\theta}_i$. More concretely, we measure the curvature of the loss at those minima, and whether flatter minima are indicative of better generalization for the meta-test tasks.

After s meta-training iterations, we have a model f parametrized by θ^s. During the meta-test, f must adapt to several meta-test tasks $\mathcal{T}_i$ independently. For a given $\mathcal{T}_i$, f adapts by performing a few steps of full-batch gradient descent on the objective landscape $\mathcal{L}(f(\mathcal{D}_i; \theta))$, using the set of support samples $\mathcal{D}_i$, and reaches an adapted solution $\tilde{\theta}_i$. Here we are interested in the curvature of $\mathcal{L}(f(\mathcal{D}_i; \tilde{\theta}_i))$, that is, the objective landscape when evaluated at such solution, and whether on average, flatter solutions favour better generalization. Considering the hessian matrix of this loss w.r.t the model parameters, defined as $H_\theta(\mathcal{D}_i; \tilde{\theta}_i) \doteq \nabla_\theta^2 \mathcal{L}(f(\mathcal{D}_i; \tilde{\theta}_i))$, we measure the curvature of the loss surface around $\tilde{\theta}_i$ using the spectral norm $\|\cdot\|_\sigma$ of this hessian matrix:

$$\left\| H_\theta(\mathcal{D}_i; \tilde{\theta}_i) \right\|_\sigma = \sqrt{\lambda_{max}\left(H_\theta(\mathcal{D}_i; \tilde{\theta}_i)^{\mathrm{H}} H_\theta(\mathcal{D}_i; \tilde{\theta}_i) \right)} = \lambda_{max}(H_\theta(\mathcal{D}_i; \tilde{\theta}_i)) \qquad (2.3)$$

as illustrated in Figure 2.1 (1). (We get $\|H_\theta(\mathcal{D}_i; \tilde{\theta}_i)\|_\sigma = \lambda_{max}(H_\theta(\mathcal{D}_i; \tilde{\theta}_i))$ since $H_\theta(\mathcal{D}_i; \tilde{\theta}_i)$ is real and symmetric.)

We define the average loss curvature for meta-test solutions $\tilde{\theta}_i$, obtained from a *meta-train solution θ^s, as:*

$$\mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T})}[\|H_\theta(\mathcal{D}_i;\tilde{\theta}_i)\|_\sigma] \tag{2.4}$$

Note that we do not measure curvature of the loss at θ^s, since θ^s is not a point of convergence of f for the meta-test tasks. In fact, at θ^s, since the model has not been adapted to the unseen meta-test classes, the target accuracy for the meta-test tasks is random chance on average. Thus, measuring the curvature of the meta-test support loss at θ^s does not relate to the notion of flatness of minima. Instead, in this work we characterize the meta-train solution θ^s by measuring the average inner product between the meta-test gradients, as explained later in Section 2.4.3.

2.4.2 Coherence of adaptation trajectories

Other than analyzing the objective landscapes at the different minima reached when f adapts to new tasks, we also analyze the adaptation trajectories to those new tasks, and whether some similarity between them can be indicative of good generalization. Let's consider a model f adapting to a task $\mathcal{T}_i$ by starting from θ^s, moving in parameter space by performing T steps of full-batch gradient descent with $\nabla_\theta\mathcal{L}(f(\mathcal{D}_i;\theta))$ until reaching $\tilde{\theta}_i$. We define the adaptation trajectory to a task $\mathcal{T}_i$ starting from θ^s as the sequence of iterates $(\theta^s, \theta_i^{(1)}, \theta_i^{(2)}, ..., \tilde{\theta}_i)$. To simplify the analyses and alleviate some of the challenges in dealing with trajectories of multiple steps in a parameter space of very high dimension, we define the trajectory displacement vector $(\tilde{\theta}_i - \theta^s)$. We define a trajectory direction vector $\vec{\theta}_i$ as the unit vector: $\vec{\theta}_i \doteq (\tilde{\theta}_i - \theta^s)/\|\tilde{\theta}_i - \theta^s\|_2$.

We define a metric for the coherence of adaptation trajectories to meta-test tasks $\mathcal{T}_i$, starting from a meta-train solution θ^s, as the average inner product between their direction vectors:

$$\mathbb{E}_{\mathcal{T}_i,\mathcal{T}_j \sim p(\mathcal{T})}[\vec{\theta}_i^{\,T}\vec{\theta}_j] \tag{2.5}$$

The inner product between two meta-test trajectory direction vectors is illustrated in Figure 2.1 (2).

2.4.3 Characterizing meta-train solutions by the average inner product between meta-test gradients

In addition to characterizing the adaptation trajectories at meta-test time, we characterize the objective landscapes at the meta-train solutions θ^s. More concretely, we measure the coherence of the meta-test gradients $\nabla_\theta \mathcal{L}(f(\mathcal{D}_i; \theta^s))$ evaluated at θ^s.

The coherence between the meta-test gradients can be viewed in relation to the metric for coherence of adaptation trajectories of Eq. 2.5 from Section 2.4.2. Even after simplifying an adaptation trajectory by its displacement vector, measuring distances between trajectories of multiple steps in the parameter space can be problematic: because of the symmetries within the architectures of neural networks, where neurons can be permuted, different parameterizations θ can represent identically the same function f that maps inputs to outputs. This problem is even more prevalent for networks with higher number of parameters. Since here we ultimately care about the functional differences that f undergoes in the adaptation trajectories, measuring distances between functions in the parameter space, either using Euclidean norm or cosine similarity between direction vectors, can be problematic [Benjamin et al., 2018].

Thus to further simplify the analyses on adaptation trajectories, we can measure coherence between trajectories of only one step ($T = 1$). Since we are interested in the relation between such trajectories and the generalization performance of the models, we measure the target accuracy at those meta-test solutions obtained after only one step of gradient descent. We define those solutions as: $\theta^s + \alpha \cdot \mathbf{g}_i$, with meta-test gradient $\mathbf{g}_i = -\nabla_\theta \mathcal{L}(f(\mathcal{D}_i; \theta^s))$. To make meta-training consistent with meta-testing, for the meta-learning algorithms we also use $T = 1$ for the inner loop updates of Eq. 2.1.

We thus measure coherence between the meta-test gradient vectors $\mathbf{g}_i$ that lead to those solutions. Note that the learning rate α is constant and is the same for all experiments on a same dataset. In contrast to Section 2.4.2, here we observed in practice that the average inner product between meta-test gradient vectors, and not just their direction vectors, is more correlated to the average target accuracy. The resulting metric is thus the average inner product between meta-test gradients evaluated at θ^s.

We define the average inner product between meta-test gradient vectors $\mathbf{g}_i$*,* *evaluated at a meta-train solution θ^s, as:*

$$\mathbb{E}_{\mathcal{T}_i,\mathcal{T}_j \sim p(\mathcal{T})}[\ \mathbf{g}_i{}^T \mathbf{g}_j\ ] \tag{2.6}$$

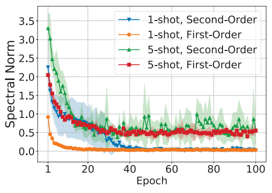
The inner product between two meta-test gradients, evaluated at θ^s, is illustrated in Figure 2.1 (3). We show in the experimental results in Section 2.5.2 and 2.5.3 that the coherence of the adaptation trajectories, as well as of the meta-test gradients, correlate with generalization on the meta-test tasks.

2.5 Experiments

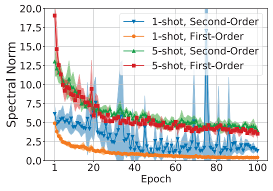
We apply our analyses to the two most widely used benchmark datasets for few-shot classification problems: Omniglot and MiniImagenet datasets. We use the standardized CNN architecture used by [Vinyals et al., 2016] and [Finn et al., 2017]. We perform our experiments using three different gradient-based meta-learning algorithms: MAML, First-Order MAML and a Finetuning baseline. For more details on the meta-learning datasets, architecture and meta-learning hyperparameters, see Appendix 2.7

We closely follow the experimental setup of [Finn et al., 2017]. Except for the Finetune baseline, the meta-learning algorithms use during meta-training the same number of ways and shots as during meta-testing. For our experiments, we follow the setting of [Vinyals et al., 2016]: for MiniImagenet, training and testing our models on 5-way classification 1-shot learning, as well as 5-way 5-shot, and for Omniglot, 5-way 1-shot ; 5-way 5-shot ; 20-way 1-shot ; 20-way 5-shot. Each experiment was repeated for five independent runs. For the meta-learning algorithms, the choice of hyperparameters closely follows [Finn et al., 2017]. For our finetuning baseline, most of the original MAML hyperparameters were left unchanged, as we want to compare the effect of the pre-training procedure, thus are kept fixed the architecture and meta-test procedures. We kept the same optimizer as for the meta-update of MAML (ADAM), and performed hyperparameter search on the mini-batch size to use, for each setting that we present. (For our reproduction results on the meta-train and meta-test accuracy, see Figure 2.9a and 2.9b in 2.8.1.)

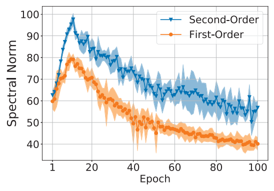
2.5.1 Flatness of meta-test solutions



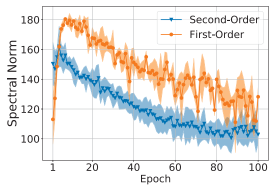
(a) Omniglot 5-way



(b) Omniglot 20-way



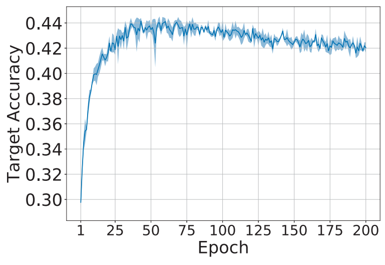
(c) MiniImagenet 5-way, 1-shot



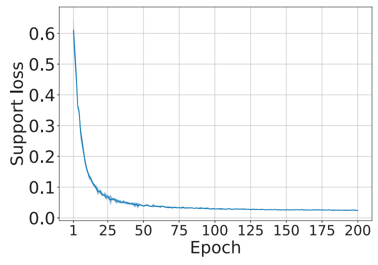
(d) MiniImagenet 5-way, 5-shot

Figure 2.2 – Flatness of meta-test solutions for MAML and First-Order MAML, on Omniglot and MiniImagenet

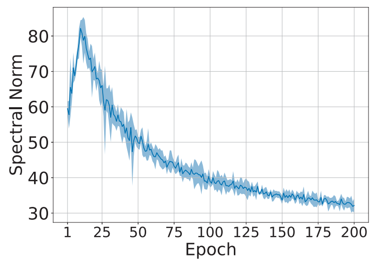
After each training epoch, we compute $\mathbb{E}[\|H_\theta(\mathcal{D}_i;\tilde{\theta}_i)\|_\sigma]$ using a fixed set of 60 randomly sampled meta-test tasks $\mathcal{T}_i$. Across all settings, we observe that MAML first finds sharper solutions $\tilde{\theta}_i$ until reaching a peak, then as the number of epoch grows, those solutions become flatter, as seen in Figure 2.2. To verify the correlation between $\mathbb{E}[\|H_\theta(\mathcal{D}_i;\tilde{\theta}_i)\|_\sigma]$ and $\mathbb{E}[Acc(f(\mathcal{D}'_i;\tilde{\theta}_i))]$, we trained for an extra 100 epochs, the model that appeared the most likely to overfit in a noticeable way, that is, First-Order MAML, with 5-way 1-shot learning on MiniImagenet, hoping that its decrease in $\mathbb{E}[Acc(f(\mathcal{D}'_i;\tilde{\theta}_i))]$ would be reflected by an increase in $\mathbb{E}[\|H_\theta(\mathcal{D}_i;\tilde{\theta}_i)\|_\sigma]$ after a certain point. On the contrary, and remarkably, even as f starts to show poorer generalization (see Figure 2.3a), the solutions keep getting flatter, as shown in Figure 2.3c. Thus for the case of gradient-based meta-learning, our finding directly contradicts the argument that flatter minima favour better generalization. We performed the same analysis for our finetuning baseline (Figures 2.4a, 2.4c), with results suggesting that flatness of solutions might be more linked with $\mathbb{E}[\mathcal{L}(f(\mathcal{D}_i;\tilde{\theta}_i))]$, the average level of support loss attained by the solutions $\tilde{\theta}_i$ (see Figures 2.4b and 2.3b), which is not an indicator for generalization. We also noted that across all settings involving MAML and First-Order MAML, this average meta-test support loss $\mathbb{E}[\mathcal{L}(f(\mathcal{D}_i;\tilde{\theta}_i))]$ decreases monotonically as meta-training progresses.



(a) Target Accuracy

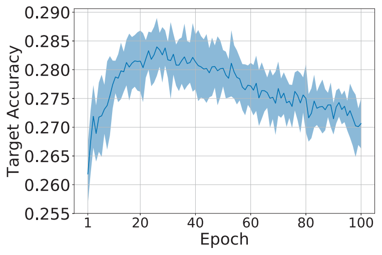


(b) Support loss

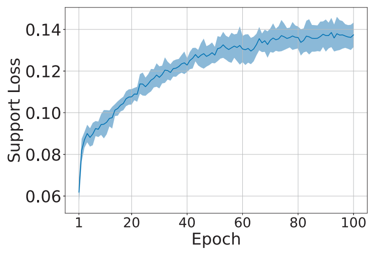


(c) Curvature of solutions

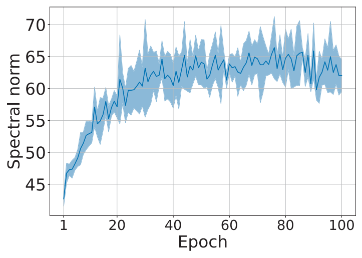
Figure 2.3 – MAML: Characterization of meta-test solutions



(a) Target accuracy



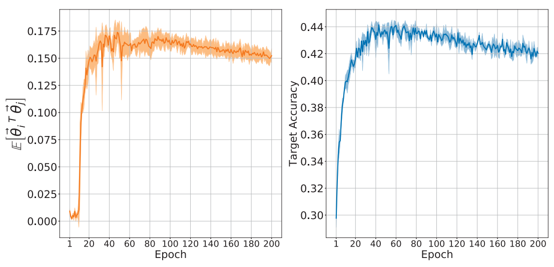
(b) Support loss



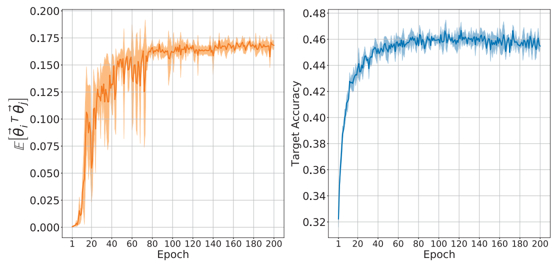
(c) Curvature of solutions

Figure 2.4 – Finetune baseline : Characterization of meta-test solutions

2.5.2 Coherence of adaptation trajectories



(a) MiniImagenet, 5-way, 1-shot, First-Order

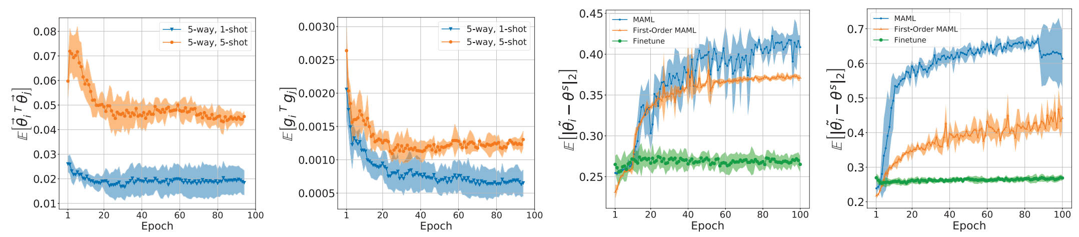


(b) MiniImagenet, 5-way, 1-shot, Second-Order

Figure 2.5 – Comparison between average inner product between meta-test trajectory direction vectors (orange), and average target accuracy on meta-test tasks (blue), MAML First-Order and Second-Order, MiniImagenet 5-way 1-shot. See Figure 2.10 in Appendix 2.8.2 for full set of experiments.

In this section, we use the same experimental setup as in Section 2.5.1, except here we measure $\mathbb{E}[\vec{\theta}_i{}^T\vec{\theta}_j]$. To reduce the variance on our results, we sample 500 tasks after each meta-training epoch. Also for experiments on Omniglot, we drop the analyses with First-Order MAML, since it yields performance very similar to that of the Second-Order MAML. We start our analyses with the setting of ”MiniImagenet, First-Order MAML, 5-way 1-shot”, as it allowed us to test and invalidate the correlation between flatness of solutions and generalization, earlier in Section 2.5.1.

We clearly observe a correlation between the coherence of adaptation trajectories and generalization to new tasks, with higher average inner product between trajectory directions, thus smaller angles, being linked to higher average target accuracy on those new tasks, as shown in Figure 2.5a. We then performed the analysis on the other settings, with the same observations (see Figure 2.5b and Figure 2.10 in Appendix 2.8.2 for full set of experiments). We also perform the analysis on the Finetuning baselines, which reach much lower target accuracies, and where we see that $\mathbb{E}[\vec{\theta}_i^{\,T}\vec{\theta}_j]$ remains much closer to zero, meaning that trajectory directions are roughly orthogonal to each other, akin to random vectors in high dimension (see Figure 2.6a).

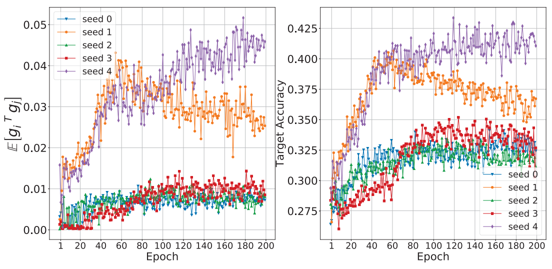


(a) Trajectories coherence **(b)** Gradients coherence **(c)** l_2 norm of trajectories (1-shot) **(d)** l_2 norm of trajectories (5-shot)

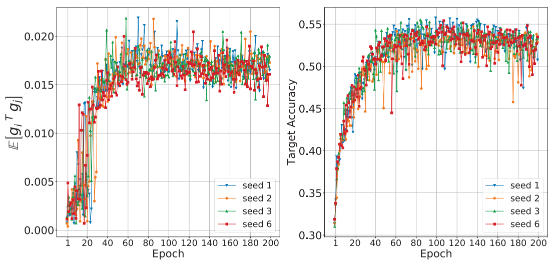
Figure 2.6 – (a): Average inner product between meta-test adaptation direction vectors, for Finetuning baseline on MiniImagenet. (b): Average inner product between meta-test gradients, for Finetuning baseline on MiniImagenet. Average l_2 norm of meta-test adaptation trajectories, all algorithms on MiniImagenet, (c): 1-shot learning, (d): 5-shot learning.

2.5.3 Characterizing meta-train solutions by the average inner product between meta-test gradients

Despite the clear correlation between $\mathbb{E}[\vec{\theta}_i^{\,T}\vec{\theta}_j]$ and generalization for the settings that we show in Figure 2.5 and 2.10, we observed that for some other settings, this relationship appears less linear. We conjecture that such behavior might arise from the difficulties of measuring distances between networks in the parameter space, as explained in Section 2.4.3. Here we present our results on the characterization of



(a) MiniImagenet, 5-way, 5-shot, First-Order



(b) MiniImagenet, 5-way, 5-shot, Second-Order

Figure 2.7 – Comparison between average inner product between meta-test gradient vectors, evaluated at meta-train solution, and average target accuracy on meta-test tasks, with higher average inner product being linked to better generalization. See Figure 2.11 in Appendix 2.8.3 for full set of experiments.

the objective landscapes at the meta-train solutions θ^s, by measuring the average inner product between meta-test gradient vectors $\mathbf{g}_i$.

We observe that coherence between meta-test gradients is correlated to generalization, which is consistent with the observations on the coherence of adaptation trajectories from Section 2.5.2. In Figure 2.7, we compare $\mathbb{E}[\ \mathbf{g}_i{}^T \mathbf{g}_j\ ]$ to the target accuracy (here we show results for individual model runs rather than the averages over the runs). See Figure 2.11 in Appendix 2.8.3 for the full set of experiments. This metric consistently correlates with generalization across the different settings. Similarly as in Section 2.5.2, for our finetuning baselines we observe very low coherence between meta-test gradients (see Figure 2.6b).

Based on the observations we make in Section 2.5.2 and 2.5.3, we propose to regularize gradient-based meta-learning as described in Section 2.6. As an added observation, here we include our experimental results on the average meta-test trajectory norm $\mathbb{E}[\|\tilde{\theta}_i - \theta^s\|_2]$ (where we used $T = 5$), in Figure 2.6c and 2.6d, where $\mathbb{E}[\|\tilde{\theta}_i - \theta^s\|_2]$ grows as meta-training progresses when f is meta-trained with MAML, as opposed to the Finetune baseline, and note that this norm does not reflect generalization.

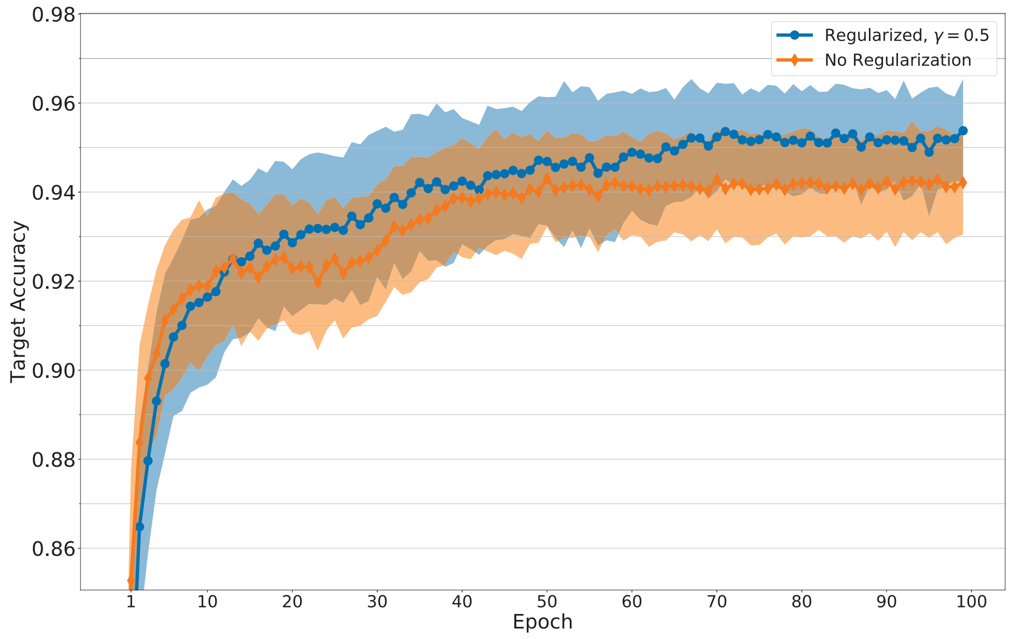


Figure 2.8 – Average target accuracy on meta-test tasks using our proposed regularizer on MAML, for Omniglot 20-way 1-shot learning, with regularization coefficient $\gamma = 0.5$

2.6 Regularizing MAML

Based on our observations on the coherence of adaptation trajectories, we propose a modification of the MAML algorithm by adding a regularization term based on $\mathbb{E}[\vec{\theta}_i^{\,T}\vec{\theta}_j]$. Within a meta-training iteration, we first let f adapt to the n training tasks $\mathcal{T}_i$ following Eq 2.1. We then compute the average direction vector $\vec{\theta}_\mu = \frac{1}{n}\sum_{i=1}^{n}\vec{\theta}_i$. From this point, we consider $\vec{\theta}_\mu$ to be fixed, such that $\nabla_\theta\vec{\theta}_\mu = 0$. For each task, we want to reduce the angle defined by $\vec{\theta}_i^{\,T}\vec{\theta}_\mu$, and thus introduce the penalty on $\Omega(\theta) = -\vec{\theta}_i^{\,T}\vec{\theta}_\mu$, obtaining the regularized solutions $\hat{\theta}_i$. The outer loop gradients are then computed, just like in MAML following Eq 2.2, but using these regularized solutions $\hat{\theta}_i$ instead of $\tilde{\theta}_i$. Note that after adding the regularizer, we consider it constant to avoid additional gradient computation overhead. We obtain the variant of MAML with regularized inner loop updates, as detailed in Algorithm 1.

We used this regularizer with MAML (Second-Order), for "Omniglot 20-way 1-shot", thereby tackling the most challenging few-shot classification setting for Omniglot. As shown in Figure 2.8, we observed an increase in meta-test target

Algorithm 1 Regularized MAML: Added penalty on angles between inner loop updates

```
During a meta-training iteration:
1: Sample a batch of $n$ tasks $\mathcal{T}_i \sim p(\mathcal{T})$
2: for all $\mathcal{T}_i$ do
3:     Perform the inner loop adaptation as in Eq. 2.1: $\tilde{\theta}_i = \theta^s - \alpha \sum_{t=0}^{T-1} \nabla_\theta \mathcal{L}(f(\mathcal{D}_i; \theta_i^{(t)}))$
4: end for
5: Compute the average direction vector: $\vec{\theta}_\mu = \frac{1}{n} \sum_{i=1}^{n} \vec{\theta}_i$
   Compute the corrected inner loop updates:
6: for all $\mathcal{T}_i$ do
7:     $\hat{\theta}_i = \tilde{\theta}_i - \gamma \nabla_\theta \Omega(\theta)$, where $\Omega(\theta) = -\vec{\theta}_i^{\,T} \vec{\theta}_\mu$
8: end for
9: Perform the meta-update as in Eq. 2.2, but using the corrected solutions:
   $\theta^{s+1} = \theta^s - \beta \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta \mathcal{L}(f(\mathcal{D}_i'; \hat{\theta}_i))$
```

accuracy: the performance increases from 94.05% to 95.38% (average over five trials, 600 test tasks each), providing $\sim$ 23% relative reduction in meta-test target error.

2.7 Additional Experimental Details

2.7.1 Model Architectures

We use the architecture proposed by [Vinyals et al., 2016] which is used by [Finn et al., 2017], consisting of 4 modules stacked on each other, each being composed of 64 filters of of 3×3 convolution, followed by a batch normalization layer, a ReLU activation layer, and a 2×2 max-pooling layer. With Omniglot, strided convolution is used instead of max-pooling, and images are downsampled to 28×28. With MiniImagenet, we used fewer filters to reduce overfitting, but used 48 while MAML used 32. As a loss function to minimize, we use cross-entropy between the predicted classes and the target classes.

2.7.2 Meta-Learning datasets

The Omniglot dataset consists of a total of 1623 classes, each comprising 20 instances. The classes correspond to distinct characters, taken from 50 different datasets, but the taxonomy among characters isn't used. The MiniImagenet dataset comprises 64 training classes, 12 validation classes and 24 test classes. Each of those classes was randomly sampled from the original Imagenet dataset, and each contains 600 instances with a reduced size of 84×84.

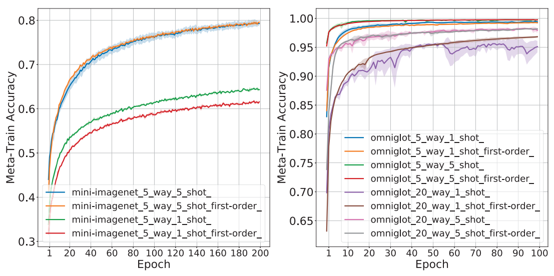
2.7.3 Hyperparameters used in meta-training and meta-testing

We follow the same experimental setup as [Finn et al., 2017] for training and testing the models using MAML and First-Order MAML. During meta-training, the inner loop updates are performed via five steps of full batch gradient descent (except for Section 2.5.3 where $T = 1$), with a fixed learning rate α of 0.1 for Omniglot and 0.01 for MiniImagenet, while ADAM is used as the optimizer for the meta-update, without any learning rate scheduling, using a meta-learning rate β of 0.001. At meta-test time, adaptation to meta-test task is always performed by performing the same number of steps as for the meta-training inner loop updates. We use a mini-batch of 16 and 8 tasks for the 1-shot and 5-shot settings respectively, while for the MiniImagenet experiments, we use batches of 4 and 2 tasks for the 1-shot and 5-shots settings respectively. Let's also precise that, in *k-shot* learning for an *m-way* classification task $\mathcal{T}_i$, the set of support samples $\mathcal{D}_i$ comprises $k \times m$ samples. Each meta-training epoch comprises 500 meta-training iterations.

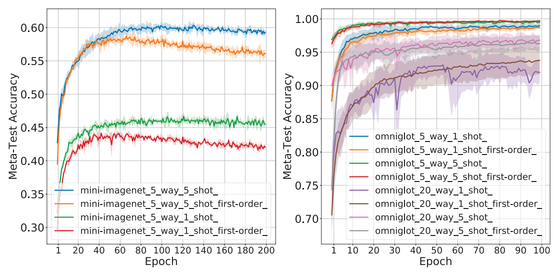
For the finetuning baseline, we kept the same hyperparameters for the ADAM optimizer during meta-training, and for the adaptation during meta-test. We searched the training hyperparameter values for the mini-batch size and the number of iterations per epoch. Experiments are run for a 100 epochs each. In order to limit meta-overfitting and maximize the highest average meta-test target accuracy, the finetuning models see roughly 100 times less training data per epoch compared to a MAML training epoch. In order to evaluate the baseline on the 1-shot and 5-shot meta-test tasks, during training we used mini-batches of 64 images with 25 iterations per epoch for 1-shot learning, and mini-batches of 128 images with 12 iterations per epoch, for 5-shot learning. At meta-test time, we use Xavier initialization [Glorot and Bengio, 2010] to initialize the weights of the final layer.

2.8 Additional Experimental Results

2.8.1 Performance of models trained with MAML and First-Order MAML, on the few-shot learning settings



(a) Meta-Train Accuracy



(b) Meta-Test Accuracy

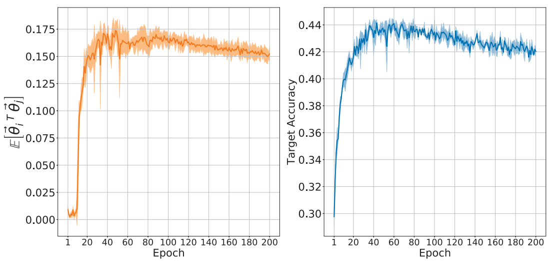
Figure 2.9 – MAML: Accuracies on training and testing tasks

2.8.2 Coherence of adaptation trajectories

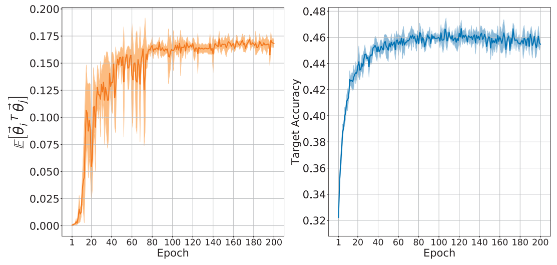
The relation between target accuracy on meta-test tasks, and angles between trajectory directions is presented in Figure 2.10.

2.8.3 Average inner product between meta-test gradients

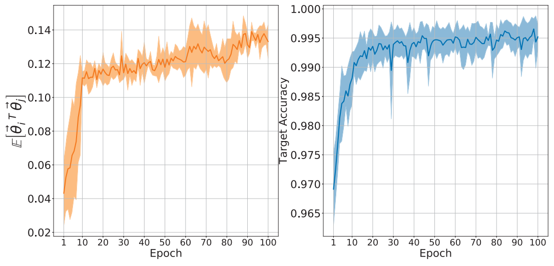
The relation between target accuracy on meta-test tasks, and average inner product between meta-test gradients evaluated at meta-train solution, is presented in Figure 2.11.



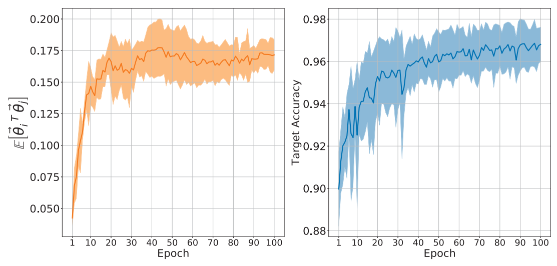
(a) MiniImagenet, 5-way, 1-shot, First-Order



(b) MiniImagenet, 5-way, 1-shot, Second-Order

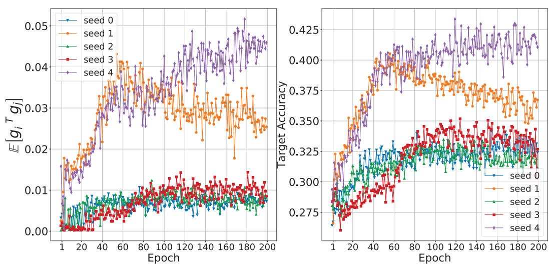


(c) Omniglot, 5-way, 5-shot, Second-Order

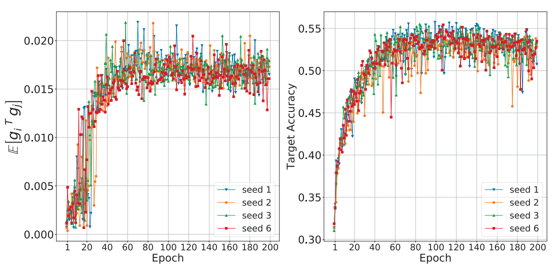


(d) Omniglot, 20-way, 5-shot, Second-Order

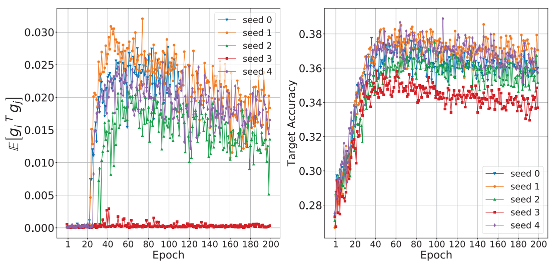
Figure 2.10 – Comparison between average inner product between trajectory directions and average target accuracy on meta-test tasks. Full set of experiments.



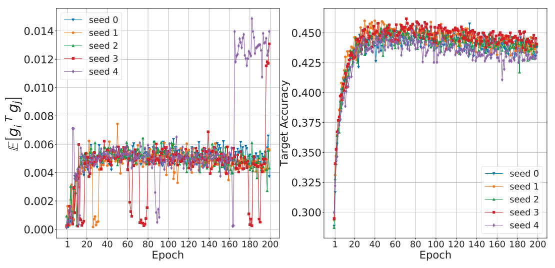
(a) MiniImagenet, 5-way, 5-shot, First-Order



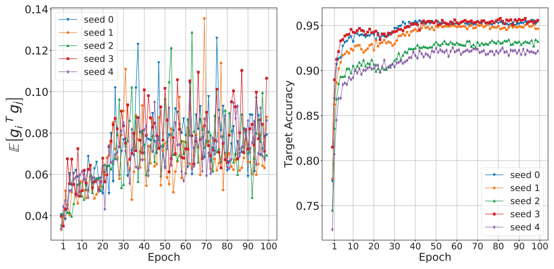
(b) MiniImagenet, 5-way, 5-shot, Second-Order



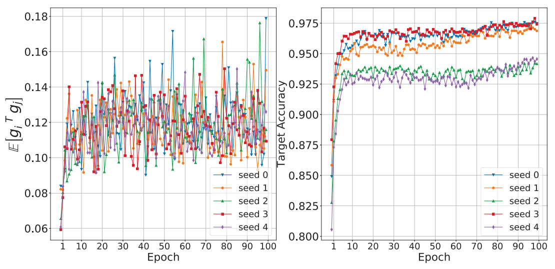
(c) MiniImagenet, 5-way, 1-shot, First-Order



(d) MiniImagenet, 5-way, 1-shot, Second-Order



(e) Omniglot, 20-way, 1-shot, Second-Order



(f) Omniglot, 20-way, 5-shot, Second-Order

Figure 2.11 – Comparison between average inner product between trajectory displacement vectors, and average target accuracy on meta-test tasks. Full set of experiments.

3 Conclusion

We experimentally demonstrate that when using gradient-based meta-learning algorithms such as MAML, meta-test solutions, obtained after adapting neural networks to new tasks via few-shot learning, become flatter, lower in loss, and further away from the meta-train solution, as meta-training progresses. We also show that those meta-test solutions keep getting flatter even when generalization starts to degrade, thus providing an experimental argument against the correlation between generalization and flat minima. More importantly, we empirically show that generalization to new tasks is correlated with the coherence between their adaptation trajectories, measured by the average cosine similarity between the adaptation trajectory directions, but also correlated with the coherence between the meta-test gradients, measured by the average inner product between meta-test gradient vectors evaluated at meta-train solution. Based on these observations, we propose a novel regularizer for MAML.

As future work, we plan to test the effectiveness of this regularizer on various datasets and meta-learning domains, architectures, and gradient-based meta-learning algorithms. Furthermore, based on our results, we believe that further inquiry into the phenomenon of coherence of meta-test updates and its relation to generalization to new tasks, ought to be pursued. Further understanding the underlying phenomena, understanding why this coherence leads to better generalization, researching more thoroughly how this insight about coherence can help in designing meta-learning algorithms that generalize well, are all questions that we believe to be relevant for the advancement of deep learning the context of ”learning to learn”.

Bibliography

Marcin Andrychowicz, Misha Denil, Sergio Gomez Colmenarejo, Matthew W. Hoffman, David Pfau, Tom Schaul, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. *CoRR*, abs/1606.04474, 2016. URL http://arxiv.org/abs/1606.04474.

Joseph M. Antognini and Jascha Sohl-Dickstein. PCA of high dimensional random walks with comparison to neural network training. *CoRR*, abs/1806.08805, 2018. URL https://arxiv.org/abs/1806.08805.

Sanjeev Arora, Nadav Cohen, and Elad Hazan. On the optimization of deep networks: Implicit acceleration by overparameterization. *CoRR*, abs/1802.06509, 2018. URL http://arxiv.org/abs/1802.06509.

Samy Bengio. *Optimisation d'une règle d'apprentissage pour réseaux de neurones artificiels*. PhD thesis, Département d'Informatique et Recherche Opérationnelle. Université de Montréal, 1993.

Samy Bengio, Y Bengio, and Jocelyn Cloutier. On the search for new learning rules for anns. *Neural Processing Letters*, 2:26–30, 07 1995. doi: 10.1007/BF02279935.

Yoshua Bengio, Samy Bengio, and Jocelyn Cloutier. Learning a synaptic learning rule. *IJCNN-91-Seattle International Joint Conference on Neural Networks*, ii: 969 vol.2–, 1991.

Ari S. Benjamin, David Rolnick, and Konrad P. Körding. Measuring and regularizing networks in function space. *CoRR*, abs/1805.08289, 2018. URL http://arxiv.org/abs/1805.08289.

Alan J. Bray and David S. Dean. Statistics of Critical Points of Gaussian Fields on Large-Dimensional Spaces. *Physical Review Letters*, 98(15):150201, April 2007. doi: 10.1103/PhysRevLett.98.150201.

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a "siamese" time delay neural network. In *Proceedings of the 6th International Conference on Neural Information Processing Systems*, NIPS'93, pages 737–744, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc. URL http://dl.acm.org/citation.cfm?id=2987189.2987282.

Yann Dauphin, Razvan Pascanu, Çaglar Gülçehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *CoRR*, abs/1406.2572, 2014. URL http://arxiv.org/abs/1406.2572.

Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. *CoRR*, abs/1703.04933, 2017. URL http://arxiv.org/abs/1703.04933.

Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *CoRR*, abs/1310.1531, 2013. URL http://arxiv.org/abs/1310.1531.

Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred A. Hamprecht. Essentially No Barriers in Neural Network Energy Landscape. *CoRR*, abs/1803.00885, 2018. URL https://arxiv.org/abs/1803.00885.

Simon S. Du, Chi Jin, Jason D. Lee, Michael I. Jordan, Barnabas Poczos, and Aarti Singh. Gradient Descent Can Take Exponential Time to Escape Saddle Points. *CoRR*, abs/1705.10412, 2017. URL https://arxiv.org/abs/1705.10412.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *CoRR*, abs/1703.03400, 2017. URL http://arxiv.org/abs/1703.03400.

Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. *CoRR*, abs/1806.02817, 2018. URL http://arxiv.org/abs/1806.02817.

Chelsea Finn, Aravind Rajeswaran, Sham M. Kakade, and Sergey Levine. Online meta-learning. *CoRR*, abs/1902.08438, 2019. URL http://arxiv.org/abs/1902.08438.

Daniel Freeman and Joan Bruna. Topology and Geometry of Half-Rectified Network Optimization. *CoRR*, abs/1611.01540, 2016. URL https://arxiv.org/abs/1611.01540.

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry Vetrov, and Andrew G. Wilson. Loss Surfaces, Mode Connectivity, and Fast Ensembling of DNNs. *CoRR*, abs/1802.10026, 2018. URL https://arxiv.org/abs/1802.10026.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL http://proceedings.mlr.press/v9/glorot10a.html.

Ian J. Goodfellow and Oriol Vinyals. Qualitatively characterizing neural network optimization problems. *CoRR*, abs/1412.6544, 2014. URL http://arxiv.org/abs/1412.6544.

Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas L. Griffiths. Recasting gradient-based meta-learning as hierarchical bayes. *CoRR*, abs/1801.08930, 2018. URL http://arxiv.org/abs/1801.08930.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *CoRR*, abs/1410.5401, 2014. URL http://arxiv.org/abs/1410.5401.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural Comput.*, 9(1): 1–42, January 1997a. ISSN 0899-7667. doi: 10.1162/neco.1997.9.1.1. URL http://dx.doi.org/10.1162/neco.1997.9.1.1.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997b. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL http://dx.doi.org/10.1162/neco.1997.9.8.1735.

Sepp Hochreiter, A. Steven Younger, and Peter R. Conwell. Learning to learn using gradient descent. In *In: Dorffner G., Bischof H., Hornik K. (eds) Artificial Neural Networks — ICANN 2001. Lecture Notes in Computer Science, vol 2130. Springer, Berlin, Heidelberg*, pages 87–94.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015. URL http://arxiv.org/abs/1502.03167.

Stanislaw Jastrzebski, Zachary Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos J. Storkey. Three factors influencing minima in SGD. *CoRR*, abs/1711.04623, 2017. URL http://arxiv.org/abs/1711.04623.

Chi Jin, Rong Ge, Praneeth Netrapalli, Sham M. Kakade, and Michael I. Jordan. How to escape saddle points efficiently. *CoRR*, abs/1703.00887, 2017. URL http://arxiv.org/abs/1703.00887.

Kenji Kawaguchi. Deep Learning without Poor Local Minima. *CoRR*, abs/1605.07110, 2016. URL https://arxiv.org/abs/1605.07110.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *CoRR*, abs/1609.04836, 2016. URL http://arxiv.org/abs/1609.04836.

Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese Neural Networks for One-Shot Image Recognition. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*. PMLR, 2015. URL https://www.cs.cmu.edu/~rsalakhu/papers/oneshot1.pdf.

Alex Krizhevsky. Learning multiple layers of features from tiny images. *Technical Report*, TR-2009 University of Toronto, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350:1332–1338, 2015.

Yann Lecun, Leon Bottou, Y Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278 – 2324, 12 1998. doi: 10.1109/5.726791.

Jason D. Lee, Max Simchowitz, Michael I. Jordan, and Benjamin Recht. Gradient Descent Converges to Minimizers. *CoRR*, abs/1602.04915, 2016. URL https://arxiv.org/abs/1602.04915.

Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. Measuring the intrinsic dimension of objective landscapes. *CoRR*, abs/1804.08838, 2018. URL http://arxiv.org/abs/1804.08838.

Hao Li, Zheng Xu, Gavin Taylor, and Tom Goldstein. Visualizing the loss landscape of neural nets. *CoRR*, abs/1712.09913, 2017. URL http://arxiv.org/abs/1712.09913.

Dougal Maclaurin, David Duvenaud, and Ryan P. Adams. Gradient-based hyperparameter optimization through reversible learning. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 2113–2122. JMLR.org, 2015. URL http://dl.acm.org/citation.cfm?id=3045118.3045343.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.*, 19 (2):313–330, June 1993. ISSN 0891-2017. URL http://dl.acm.org/citation.cfm?id=972470.972475.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. *CoRR*, abs/1707.03141, 2017. URL http://arxiv.org/abs/1707.03141.

Yurii Nesterov. A method for solving the convex programming problem with convergence rate O(1/kˆ2). *Dokl. Akad. Nauk SSSR*, 269:543–547, 1983. URL https://ci.nii.ac.jp/naid/10029946121/en/.

Quynh N. Nguyen and Matthias Hein. The loss surface of deep and wide neural networks. *CoRR*, abs/1704.08045, 2017. URL http://arxiv.org/abs/1704.08045.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *CoRR*, abs/1803.02999, 2018. URL http://arxiv.org/abs/1803.02999.

Boris N. Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. TADAM: task dependent adaptive metric for improved few-shot learning. *CoRR*, abs/1805.10123, 2018. URL http://arxiv.org/abs/1805.10123.

Razvan Pascanu, Yann N. Dauphin, Surya Ganguli, and Yoshua Bengio. On the saddle point problem for non-convex optimization. *CoRR*, abs/1405.4604, 2014. URL http://arxiv.org/abs/1405.4604.

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017. URL https://openreview.net/forum?id=rJY0-Kcll.

Danilo J. Rezende, Shakir Mohamed, Ivo Danihelka, Karol Gregor, and Daan Wierstra. One-shot generalization in deep generative models. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 1521–1529. JMLR.org, 2016. URL http://dl.acm.org/citation.cfm?id=3045390.3045551.

Jonas Rothfuss, Dennis Lee, Ignasi Clavera, Tamim Asfour, and Pieter Abbeel. Promp: Proximal meta-policy search. *CoRR*, abs/1810.06784, 2018. URL http://arxiv.org/abs/1810.06784.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. Imagenet large scale visual recognition challenge. *CoRR*, abs/1409.0575, 2014. URL http://arxiv.org/abs/1409.0575.

Itay Safran and Ohad Shamir. On the quality of the initial basin in overspecified neural networks. *CoRR*, abs/1511.04210, 2015. URL http://arxiv.org/abs/1511.04210.

Itay Safran and Ohad Shamir. Spurious local minima are common in two-layer relu neural networks. *CoRR*, abs/1712.08968, 2017. URL http://arxiv.org/abs/1712.08968.

Levent Sagun, Utku Evci, V. Ugur Güney, Yann Dauphin, and Léon Bottou. Empirical analysis of the hessian of over-parametrized neural networks. *CoRR*, abs/1706.04454, 2017. URL http://arxiv.org/abs/1706.04454.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1842–1850, New York, New York, USA, 20–22 Jun 2016. PMLR. URL http://proceedings.mlr.press/v48/santoro16.html.

Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How Does Batch Normalization Help Optimization ? (No, It Is Not About Internal Covariate Shift). *CoRR*, abs/1805.11604, 2018. URL https://arxiv.org/abs/1805.11604.

Jürgen Schmidhuber. Evolutionary principles in self-referential learning. on learning now to learn: The meta-meta-meta...-hook. Diploma thesis, Technische Universitat Munchen, Germany, 14 May 1987. URL http://www.idsia.ch/~juergen/diploma.html.

Jürgen Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, 1992. doi: 10.1162/neco.1992.4.1.131. URL https://doi.org/10.1162/neco.1992.4.1.131.

Jürgen Schmidhuber. A neural network that embeds its own meta-levels. In *In Proc. of the International Conference on Neural Networks '93*. IEEE, 1993.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. URL https://arxiv.org/abs/1409.1556.

Samuel L. Smith and Quoc V. Le. A bayesian perspective on generalization and stochastic gradient descent. *CoRR*, abs/1710.06451, 2017. URL http://arxiv.org/abs/1710.06451.

Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning. *CoRR*, abs/1703.05175, 2017. URL http://arxiv.org/abs/1703.05175.

Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *CoRR*, abs/1609.03499, 2016. URL http://arxiv.org/abs/1609.03499.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.

Luca Venturi, Afonso S. Bandeira, and Joan Bruna. Neural Networks with Finite Intrinsic Dimension have no Spurious Valleys. *CoRR*, abs/1802.06384, 2018. URL https://arxiv.org/abs/1802.06384.

Oriol Vinyals, Charles Blundell, Timothy P. Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. *CoRR*, abs/1606.04080, 2016. URL http://arxiv.org/abs/1606.04080.

Chen Xing, Devansh Arpit, Christos Tsirigotis, and Yoshua Bengio. A Walk with SGD. *CoRR*, abs/1802.08770, 2018. URL https://arxiv.org/abs/1802.08770.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *CoRR*, abs/1611.03530, 2016. URL http://arxiv.org/abs/1611.03530.

Pan Zhou and Jiashi Feng. Empirical risk landscape analysis for understanding deep neural networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=B1QgVti6Z.

Luisa M. Zintgraf, Kyriacos Shiarlis, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. CAML: fast context adaptation via meta-learning. *CoRR*, abs/1810.03642, 2018. URL http://arxiv.org/abs/1810.03642.

Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. *CoRR*, abs/1611.01578, 2016. URL http://arxiv.org/abs/1611.01578.