Gradient-based Accelerated Adaptation of Deep Networks

Prerna Khullar
Department of Electrical Engineering
pkhullar@stanford.edu

Varun Srivastava
Department of Electrical Engineering
vsriva@stanford.edu

1 Introduction

Humans' mysteriously remarkable ability to adapt to new tasks, based on limited prior experience is based on their ability to leverage past experience with tasks that inherently share a similar "structure". For example, opening a water bottle is very similar in nature to opening a jar of jam, or even crushing pepper. Undoubtedly, deep learning has achieved significant progress in this area, but this performance is usually accompanied by obstructively large amounts of data and computation cost.

Hence, *the task of learning using few samples, i.e few-shot learning or meta learning* is appealing both from the perspective of gaining a deeper understanding of human intelligence using computational methods as well as increasing performance on tasks which have scarce data or long tails on data distributions. Achieving human performance in meta learning will also obviate the need to collect data in settings where the cost and implications are prohibitive.

The field of few-shot learning can be broadly classified into 3 paradigms from an algorithmic perspective, namely, *optimization-based*, *model-based* and *metric-based*.

Model Based (BlackBox)	**Optimization Based**	**Metric Based**
(+) Expressive, Flexible Application	(+) Embed optimization structure, model agnostic, strong (+) inductive bias, Better generalization for OOD	(+) Computationally feasible, data efficient
(-) Opaque, challenging to optimize, empirically superseded by optimization-based models	(-) Memory intensive and difficult to optimize, compounded with unstable training	(-) Hard to scale and generalize, limited to classification

Table 1: Comparison of meta learning paradigms from a computational graph perspective

Based on Table 2, the optimization-based approach offers the best trade-off in model and application flexibility, empirical performance and scalability with K, but **fixing the computational issues** still remains a concern.
These problems stem from the bi-level or dual optimization loop for meta- learning where the inner-loop adapts to a specific task using a single (or few) gradient updates, and the outer-loop achieves the *meta*-training objective of finding parameters which can generalise to many tasks. However, this requires backpropgation through the inner loop which requires computation of a Hessian. The inner-level learning is ideally rapid, while the outer optimization occurs at a gradual pace based on the feedback and performance of the inner-loop.

The main ambit of this project is to leverage the optimization structure and the well-developed techniques of Convex Optimization to suggest (potentially novel) solutions that could improve training stability (convergence, evaluation criteria at convergence), as well as test the generality of suggested improvements on standard datasets.

2 Datasets

Since, the tasks used for meta-learning must share structure the ideal dataset involves the same underlying domain (images of characters/natural phenomena etc), while differing in the output label. Consequently, the standard datasets used in the setting and used for the experiments in this project are:

1. **MiniImageNet Dataset** Vinyals et al. [2016]: It consists of 600 instances of 100 classes from the classic ImageNet dataset downsampled to 84x84 images (see Figure 1a)
2. **Omniglot dataset** Lake et al. [2015]: It consists of 1623 characters from 50 languages/character sets and every class has 20 examples (see Figure 1b).

Since the datasets are balanced, this alleviates many of the problems stemming from an imbalanced dataset, and thus test accuracy is used as the metric of choice for experiments/baselines.

3 Method

The following section is a review of the general meta-learning problem formulation, while the subsequent sections setup the mathematical formulation for specific gradient based meta learning techniques.

(a) MiniImagenet Dataset

(b) Omniglot Dataset

Figure 1: Datasets used for N-way, K-shot classification (i.e. few shot) experiments

3.1 Optimization Based Meta Learning

In this setting, one defines a collection of meta-training tasks $\{\mathcal{T}_i\}_{i=1}^{M}$ such that each task $\mathcal{T}_i$ is associated with a dataset $\mathcal{D}_i$, from which one can sample two disjoint sets: $\mathcal{D}_i^{\text{tr}}$ and $\mathcal{D}_i^{\text{test}}$ with K input-output pairs each. The datasets take the form $\mathcal{D}_i^{\text{tr}} = \{(\mathbf{x}_i^k, \mathbf{y}_i^k)\}_{k=1}^{K}$, and similarly for $\mathcal{D}_i^{\text{test}}$, where $\mathbf{x} \in \mathcal{X}$ and $\mathbf{y} \in \mathcal{Y}$ denote inputs and outputs, respectively.

We want to learn models of the form $h_{\boldsymbol{\theta}}(\mathbf{x}) : \mathcal{X} \to \mathcal{Y}$, parameterized by $\boldsymbol{\theta} \in \Theta \equiv \mathbb{R}^d$. Performance on a task is specified by a loss function denoted as $\mathcal{L}(\boldsymbol{\theta}, \mathcal{D})$, as a function of a parameter vector and dataset. The goal for task $\mathcal{T}_i$ is to learn task-specific parameters $\boldsymbol{\theta}_i$ using $\mathcal{D}_i^{\text{tr}}$ such that we can minimize the test loss of the task, $\mathcal{L}(\boldsymbol{\theta}_i, \mathcal{D}_i^{\text{test}})$. We chose $\mathcal{L}$ to be the cross-entropy loss for all our experiments.

The goal of optimization base meta-learning is to learn meta-parameters that produce good task specific parameters using the following procedure:

$$\overbrace{\boldsymbol{\theta}^*_{\mathrm{ML}} := \operatorname*{arg\,min}_{\boldsymbol{\theta}\in\Theta} F(\boldsymbol{\theta})}^{\text{outer}-\text{level optimization}}, \text{ where } F(\boldsymbol{\theta}) = \frac{1}{M}\sum_{i=1}^{M} \mathcal{L}\Bigg(\overbrace{\mathcal{A}lg\big(\boldsymbol{\theta}, \mathcal{D}_i^{\text{tr}}\big)}^{\text{inner}-\text{level optimization}}, \mathcal{D}_i^{\text{test}}\Bigg). \tag{1}$$

This is a bi-level optimization problem since $\mathcal{A}lg\big(\boldsymbol{\theta}, \mathcal{D}_i^{\text{tr}}\big)$ is either explicitly or implicitly solving an underlying optimization problem.

At meta-test time, when presented with a dataset $\mathcal{D}_j^{\text{tr}}$ corresponding to a new task $\mathcal{T}_j$, low test error is achieved by using the optimization procedure with the meta-learned parameters as $\boldsymbol{\theta}_j = \mathcal{A}lg(\boldsymbol{\theta}^*_{\mathrm{ML}}, \mathcal{D}_j^{\text{tr}})$.

3.2 MAML

In the case of MAML Finn et al. [2017], $\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D})$ corresponds to one or multiple steps of gradient descent initialized at $\boldsymbol{\theta}$. For example, if one step of gradient descent is used, we have:

$$\boldsymbol{\theta}_i \equiv \mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^{\text{tr}}) = \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \mathcal{D}_i^{\text{tr}}). \qquad \text{(inner-level of MAML)} \tag{2}$$

Typically, α is a scalar hyperparameter. Hence, for MAML, the meta-learned parameter ($\boldsymbol{\theta}^*_{\mathrm{ML}}$) is essentially learning a good initialization for to solve various tasks using gradient descent. To solve the outer-level problem with gradient-based methods, one needs to differentiate through $\mathcal{A}lg$.

$$\frac{dAlg(\boldsymbol{\theta}, \mathcal{D}_i^{\text{tr}})}{d\boldsymbol{\theta}} = \boldsymbol{I} - \alpha \nabla^2_{\boldsymbol{\theta}} \hat{\mathcal{L}}_i(\boldsymbol{\theta}, \mathcal{D}_i^{\text{tr}}) \equiv \mathbb{R}^{d\times d} \tag{3}$$

Hence, the practical performance (as shown in Section 4) is **crippled by the backpropagation through $\mathcal{A}lg$ which requires memory and time proportional to the number of inner steps** (which are consequently forced to be low) and requires the computation of a Hessian matrix making this method infeasible for practical learning.

3.3 iMAML

Implicit MAML provides a critical breakthrough for the gradient based class of architectures by making the task adaption step independent of the optimization path (under some convergence assumptions) Rajeswaran et al. [2019] where one can see that Eq. 5 is **independent of** $\mathcal{A}lg$ unlike MAML (in Eq. 2).

$$\boldsymbol{\phi}_i = Alg_i^*(\boldsymbol{\theta}, \mathcal{D}_i^{\text{tr}})) := \operatorname*{arg\,min}_{\boldsymbol{\phi}' \in \boldsymbol{\Phi}} \mathcal{L}(\boldsymbol{\phi}', \mathcal{D}_i^{tr}) + \frac{\lambda}{2}\|\boldsymbol{\phi}' - \boldsymbol{\theta}\|^2 \tag{4}$$

$$\frac{dAlg_i^*(\boldsymbol{\theta}, \mathcal{D}_i^{\text{tr}}))}{d\boldsymbol{\theta}} = \left(\boldsymbol{I} + \frac{1}{\lambda}\nabla^2_{\boldsymbol{\theta}} \hat{\mathcal{L}}_i(\boldsymbol{\phi}_i)\right)^{-1} \quad \text{Proved in Appendix A.1} \tag{5}$$

This represents *a leap forward* in the memory consumption of MAML of MAML which is crippled both by the computationally expensive backpropgation through $\mathcal{A}lg$ and the memory requirements of storing the entire computational history of $\mathcal{A}lg$ all of which are obviated by iMAML. Note however the computation is still quite expensive due to the computation of the hessian $\nabla^2_{\boldsymbol{\theta}} \hat{\mathcal{L}}_i(\boldsymbol{\phi}_i)$

One can interpret the additional L2 norm penalty on the parameters as regularization encouraging the adapated parameters to not stray too far from the initialization θ.

3.4 Convex (i)MAML

The two primary issues plaguing MAML (and in general the optimization based techniques) is the computational time and memory complexity. We present a simple yet elegant method to solve both by leveraging convex optimization techniques. iMAML proposes to solve by computing an exact solution to the inner optimization (making it path independent). However, computing exact solutions is practically infeasible due to the incumbent computational requirements and mathematically impossible since the loss function $\mathcal{L}(\boldsymbol{\theta}, \mathcal{D}_i^{tr})$ is nonconvex in the parameters $\boldsymbol{\theta}$. However, consider the parameter split given by $\theta = \begin{bmatrix} \theta_i \\ \theta_o \end{bmatrix}$, where θ_i are the **final layer** parameters to be tuned in the inner loop and θ_o are the outer loop parameters consisting of the pre-final layer weights. This simple modification achieves two critical functions:

1. It makes the inner optimization a convex optimization problem (with non-linear input features) over the final layer weights. Hence, the problem can now be solved (under mild smoothness assumptions) exactly (under the margin of numerical error)
2. It allows one to leverage *arbitrary* convex optimization routines for optimization since iMAMLs critical contribution allows one to use path independence of gradients and thus substitute in arbitrary convex optimization routines in the place of $\mathcal{A}lg$.

We can clearly see the reduction in backpropgation memory and time requirements by examining the hessian matrix given by

$$\text{ConvexMAML:} \quad \begin{bmatrix} \overbrace{\begin{matrix} \mathbb{I}_{d_1} - \alpha \nabla^2_{\theta_i} \hat{\mathcal{L}}(\theta, \mathcal{D}^{tr}) \\ -\alpha \nabla_{\theta_o} \nabla_{\theta_i} \hat{\mathcal{L}}(\theta, \mathcal{D}^{tr}) \end{matrix}}^{\mathbb{R}^{d_1 \times d_1}} \kern-3em \underbrace{\phantom{\begin{matrix} \mathbb{I} \end{matrix}}}_{\mathbb{R}^{d_2 \times d_1}} & \begin{matrix} 0 \\ \mathbb{I}_{d_2} \end{matrix} \end{bmatrix}_{n \times n}, \qquad d_1 + d_2 = d \tag{6}$$

$$\theta = \begin{bmatrix} \theta_i \\ \theta_o \end{bmatrix} \quad \theta_i \in \mathbb{R}^{d_1} \ \theta_o \in \mathbb{R}^{d_2} \qquad d_1 << d_2 \tag{7}$$

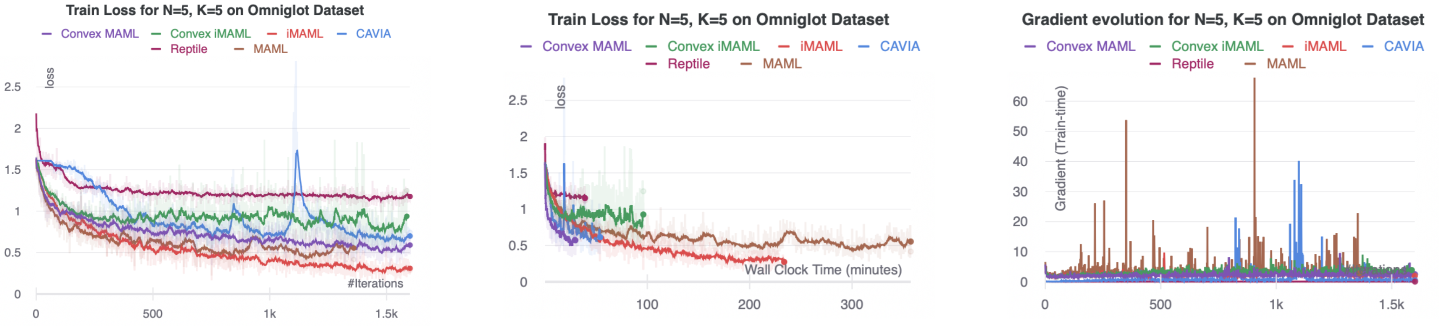
Note that unlike the case of MAML where the hessian was a $\mathbb{R}^{d \times d}$, this is a significantly sparser matrix ($\in \mathbb{R}^{d \times d_1}$) since the final layer weights d_1 are usually the smallest layer in a network (and usually insignificant in comparison to the total parameters in big networks.)

We demonstrate the clear computational superiority of our method using the Experiments in Section 4.

4 Experiments

The task chosen for all experiments is that of N way, K shot learning which is learning to distinguish between N (usually a small number between 2 - 10) classes given K **labelled** samples (usually a small number between 1 - 20, as opposed to thousands and millions of samples) **for each class**. Prediction then occurs by classifying input data belonging to one of the N classes, using only the extremely limited data (K samples provided). We use cross entropy loss in all our experiments, and use classification accuracy on a balanced dataset to evaluate our models. We use *train loss* for our convergence analysis since that is the principal quantity dictating the convergence of an algorithm in practice (note that one can equivalently use train accuracy which will on most occasions follow identical trends).

4.1 Omniglot Dataset

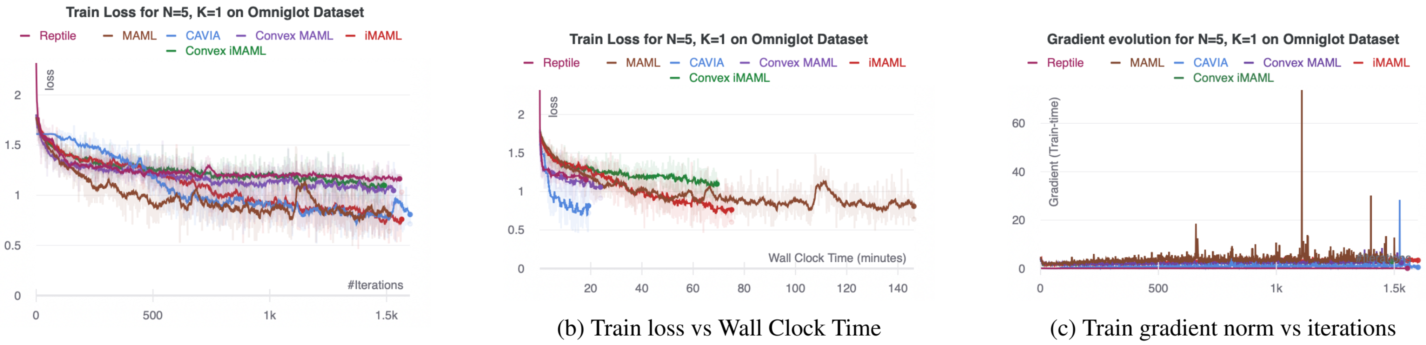


(a) Train loss vs Iterations

(b) Train loss vs Wall Clock Time

(c) Train gradient norm vs iterations

Figure 2: 5-way, 5-shot classification on Omniglot Dataset

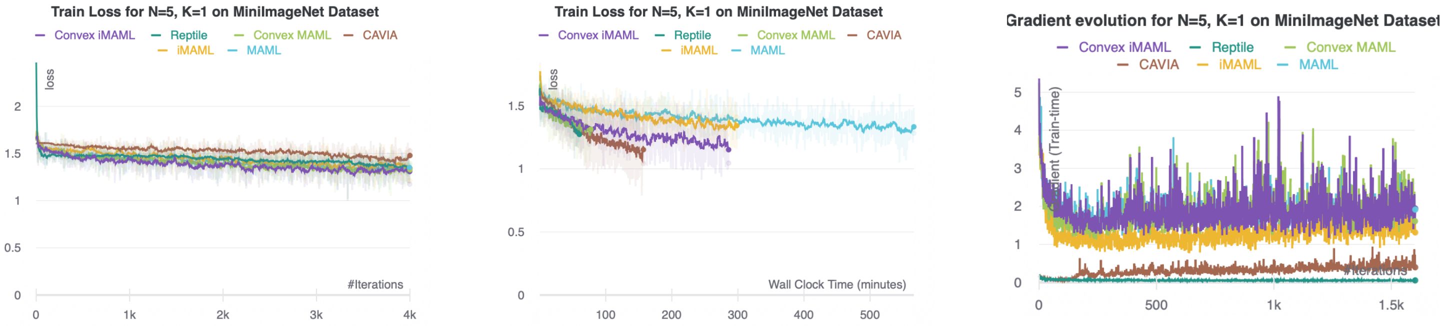


(a) Train loss vs Iterations

(b) Train loss vs Wall Clock Time

(c) Train gradient norm vs iterations

Figure 3: 5-way, 1-shot classification on Omniglot Dataset

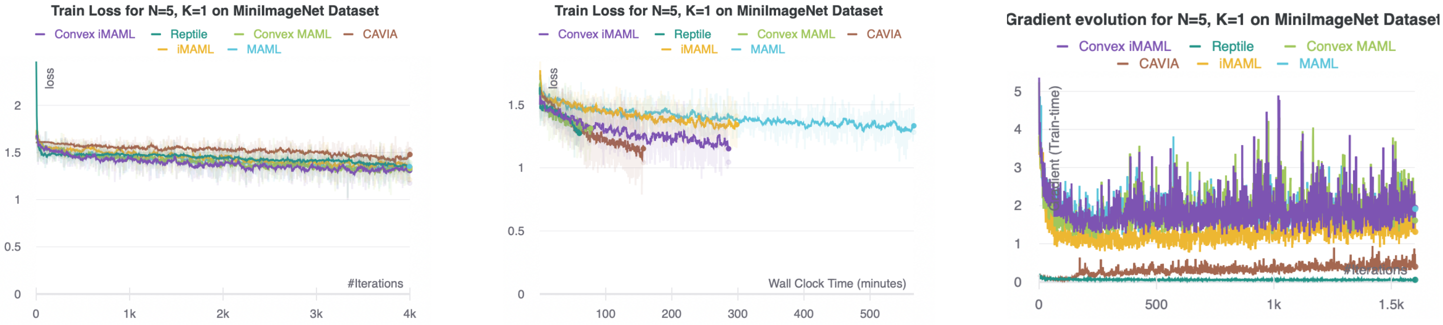


(a) Train loss vs Iterations

(b) Train loss vs Wall Clock Time

(c) Train gradient norm vs iterations

Figure 4: 5-way, 1-shot classification on MiniImageNet Dataset



(a) Train loss vs Iterations

(b) Train loss vs Wall Clock Time

(c) Train gradient norm vs iterations

Figure 5: 5-way, 5-shot classification on MiniImageNet Dataset

4.2 MiniImagenet Dataset

For all Figures 2, 3, 5, 4 we see that Convex(i)MAML matches all baseline algorithms and outperforms CAVIA and Reptile. However, one can note significant increase in computational efficiency in Figure 2b, where the Convex approaches utterly outperform MAML and iMAML converging in a few minutes instead of the hours required for MAML.

One can also observe from the evolution of gradients that inducing convexity smooths out gradients, and reduces oscillations in loss (such as the ones visible for CAVIA)

4.3 General Analysis

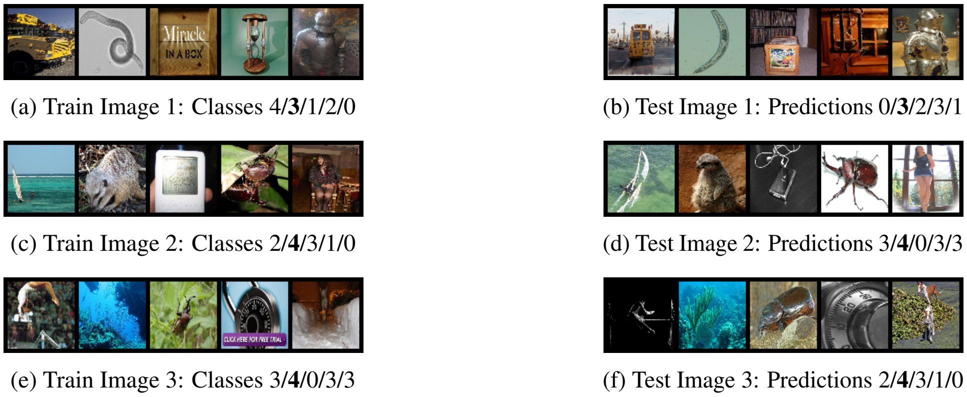
We have seen overall that ConvexMAML is leaps and bounds ahead in computational efficiency, and we hypothesize that minor hyperparameter tuning will allow it to achieve and perhaps even exceed the empirical performance of other competing algorithms on all datasets. The significant speedup in training allows it to train upt 10x faster in some cases. Reptile fails to offer competitive performance due to making the highly inaccurate approximation of taking the gradient of the computatational path to be an identity matrix. CAVIA in many cases matches the performance characteristics due to the shared similarity in modelling the architecture as task specific and task agnostic parameters.

Algorithm	**MAML**	**Convex MAML (ours)**	**iMAML**	**Convex iMAML (ours)**	**Reptile**	**CAVIA**
			K = 1			
Test Accuracy on MiniImageNet	0.42	0.42	0.42	**0.45**	0.28	0.41
Test Accuracy on Omniglot	0.76	**0.81**	0.75	**0.81**	0.59	0.72
			K = 5			
Test Accuracy on MiniImageNet	0.59	**0.62**	0.49	**0.62**	0.29	0.57
Test Accuracy on Omniglot	0.91	**0.95**	0.92	0.88	0.68	0.79

Table 2: Comparison of Convex(i)MAML on Omniglot and Miniimagenet for N=5, K=1 and N=5, K=5. One can note that since Omniglot is a smaller dataset, it is easier to fit for all algorithms, however Convex(i)MAML outperforms on the MiniImagenet database

4.4 Error Analysis

On analysing the examples from the query set, and the support set i.e. on looking at the meta-train and meta-test examples, for the points that had less 20% accuracy (equivalent to random behaviour for 5 classes), it was observed that the image was only able to classify the image correctly, based mainly on its image colour. For example, in the figure, we observe that the pictures where the entire image is blue (due to the sea), or consisted mainly of a background. This shows that the image was not able to learn the more complex features well, which would enable it to distinguish between objects. We surmise that this might be due to the fact that we are only updating the weights in the final layer, and therefore the model is unable to pick-up on the complex features.



(a) Train Image 1: Classes 4/**3**/1/2/0

(b) Test Image 1: Predictions 0/**3**/2/3/1

(c) Train Image 2: Classes 2/**4**/3/1/0

(d) Test Image 2: Predictions 3/**4**/0/3/3

(e) Train Image 3: Classes 3/**4**/0/3/3

(f) Test Image 3: Predictions 2/**4**/3/1/0

Figure 6: Examples of incorrectly labelled images, with one correctly predicted class

5 Conclusion and Future Work

In this work, we analyse a number of state of the art techniques on gradient based meta learning, focusing on their computational performance for modern workloads, where the foundational technique MAML fails to meet the benchmark of practical feasibility (despite being theoretically sound). We provide a theoretical motivation for a new modification motivated by the general good performance and guarantees of convex optimization, named ConvexMAML which shows very encouraging empirical results. Future experiments may focus on a broader class of experiments, with a larger number of datasets (and imabalanced cases). One can also analyse the algorithm from a optimization theoretic perspective to provide stronger guarantees on convergence, as well as examine scalibility under the convex optimization constraint since one is limiting to increasing the breadth of the final layer to satisfy convexity in the inner loop.

A Proofs

A.1 Path independent Gradient

Proof. Define:

$$G_i(\phi', \boldsymbol{\theta}) := \hat{\mathcal{L}}_i(\phi') + \frac{\lambda}{2} ||\phi' - \boldsymbol{\theta}||^2.$$

Since $\phi = \mathcal{A}lg^{\star}(\boldsymbol{\theta}) = \arg\min G(\phi', \boldsymbol{\theta})$ [1] is the minimizer of $G(\phi', \boldsymbol{\theta})$, it must be a critical point, hence

$$\nabla'_{\phi} G(\phi', \boldsymbol{\theta}) \mid_{\phi'=\phi} = 0 \implies \nabla \hat{\mathcal{L}}(\phi) + \lambda(\phi - \boldsymbol{\theta}) = 0 \implies \phi = \boldsymbol{\theta} - \frac{1}{\lambda} \nabla \hat{\mathcal{L}}(\phi),$$

We can differentiate the above equation to obtain:

$$\frac{d\phi}{d\boldsymbol{\theta}} = \boldsymbol{I} - \frac{1}{\lambda} \nabla^2 \hat{\mathcal{L}}(\phi) \frac{d\phi}{d\boldsymbol{\theta}} \implies \left(\boldsymbol{I} + \frac{1}{\lambda} \nabla^2 \hat{\mathcal{L}}(\phi) \right) \frac{d\phi}{d\boldsymbol{\theta}} = \boldsymbol{I}.$$

which completes the proof. □

B Contributions

Prerna: Instrumentation of code, plotting, adaptation of baselines - CAVIA, Reptile, theoretical analysis
Varun: Implementation of ConvexMAML, MAML and GCP administration, theoretical analysis
Reports made jointly.

References

C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.

B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

A. Rajeswaran, C. Finn, S. M. Kakade, and S. Levine. Meta-learning with implicit gradients. In *Advances in Neural Information Processing Systems*, pages 113–124, 2019.

O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638, 2016.

[1] Ignoring task i subscripts in the proof for convenience.