Is Support Set Diversity Necessary for Meta-Learning?

Anonymous Author(s)
Affiliation
Address
email

Abstract

Meta-learning is a popular framework for learning with limited data in which a model is produced by training over multiple few-shot learning tasks. For classification problems, these tasks are typically constructed by sampling a small number of support and query examples from a subset of the classes. While conventional wisdom is that task diversity should improve the performance of meta-learning, in this work we find evidence to the contrary: we propose a modification to traditional meta-learning approaches in which we keep the support sets fixed across tasks, thus reducing task diversity. Surprisingly, we find that not only does this modification *not* result in adverse effects, it almost always improves the performance for a variety of datasets and meta-learning methods. We also provide several initial analyses to understand this phenomenon. Our work serves to: (i) more closely investigate the effect of support set construction for the problem of meta-learning, and (ii) suggest a simple, general, and competitive baseline for few-shot learning.

1 Introduction

The ability to learn from limited experience is a defining aspect of human intelligence. The domain of few-shot learning aims to evaluate this criterion within machine learning [9]. For few-shot learning problems, meta-learning [29] techniques have attracted increasing attention. Meta-learning methods typically construct few-shot classification tasks in an episodic manner by sampling *support* (S) and *query* (Q) examples from a fixed number of classes. The objective is then to deliver an algorithm that can perform well on the query points of a task by training on only a few support samples.

Conventional wisdom is that the performance of meta-learning methods will improve as we train on more diverse tasks. Therefore, during meta-training, we typically allow a task to be constructed from any possible pair of support and query sets. In this work, we question this notion, specifically investigating the effect that support set diversity has on meta-learning. Surprisingly, we find that reducing the total number of unique support sets not only does *not* result in adverse effects—in most cases, it in fact yields significant performance improvements.

The main contributions of this work are as follows: (1) We propose a simple modification to the meta-learning objective that applies to a broad set of methods, in which we aim to optimize a biased objective by restricting the number of support sets used to construct tasks. (2) On multiple datasets and model architectures we empirically demonstrate the performance improvement of the learned algorithm's when trained with our proposed objective. (3) Finally, we explore a framework to understand this surprising phenomenon and attempt to provide some insights into the observed gains. Our work delivers a simple, competitive baseline for few-shot learning and, more generally, investigates the role that task construction plays in meta-learning.

Submitted to 34th Conference on Neural Information Processing Systems (NeurIPS 2020). Do not distribute.

2 Background & Related Work

Meta-learning formulations typically rely on episodic training, wherein an algorithm is learned to adapt to a task, given its support set, so as to minimize the loss incurred on the query set. Meta-learning methods differ in terms of the algorithms they learn: *Gradient-based meta-learning methods* [10, 11, 17, 22, 26] consider learning gradient-based algorithms, which are typically parameterized by an initialization (possibly together with pre-conditioning matrix). Recently, *last-layer meta-learning methods* [3, 17, 21, 25] have gained popularity. These methods don't learn an optimal initialization for the task distribution but instead a feature extractor on the covariates in the support and query sets. The extracted support features along with their labels are used to learn the parameters of the last layer, e.g., in a non-parametric fashion [25] or by finding the unique solution for convex problems [2, 17].

Several recent works [6, 7, 20] on few-shot learning have proposed methods that don't directly optimize the original meta-learning objective. Instead they chose to minimize a *transfer-learning* based loss where a deep network is trained in a supervised manner on all the (x, y) pairs present in the dataset. In contrast to episodic training, they show that a model trained without any notion of a task can be used to construct an algorithm whose performance matches or improves upon meta-learning methods across many benchmarks. Similar to these works, we propose to optimize an alternate objective for meta-learning; however, our work differs by retaining the episodic structure from the meta-learning framework but restricting the support set (and by consequence task) diversity.

In addition to using other objectives for few-shot learning, recent work aims to improve meta-learning by explicitly looking at the task structure and their relationships. Among these, Yin et al. [31] propose to handle the lack of mutual exclusiveness among different tasks through an information-theoretic regularized objective. Zhang et al. [33] propose a minimax objective to make the learned algorithm task-robust, which improves the performance of the algorithm on the most difficult task in the face of a task pool with diverse difficulty. Liu et al. [19] propose to augment the set of possible tasks by augmenting the pre-defined set of classes that generate the tasks with varying degrees of rotated inputs as new classes. In addition, several popular meta-learning methods [17, 25], in order to improve the meta-test performance, change the number of ways or shots of the sampled meta-training tasks, thus increasing the complexity and diversity of the tasks. In contrast to these works, we look at the structure and diversity of tasks specifically through the lens of support set diversity, and show that, surprisingly, reducing diversity (by fixing support set) not only maintains—but in many cases significantly improves—the performance of meta-learning. Our results indicate that this simple modification to meta-learning is an effective method worthy of future study.

3 Meta-Learning with a Fixed Support Pool

We begin with notation we will use to re-formulate the meta-learning objective. This will allow us to introduce the concept of a *support pool* in a manner amenable with current meta-learning methods.

Notation. In a standard offline meta supervised learning problem, we are given a dataset $\mathcal{D} \subseteq \mathcal{X} \times [N]$ with examples from $[N] := 1, \ldots, N$ different classes. A task (C, S, Q) is constructed by first choosing a subset of classes $C \subseteq [N]$, and conditioning on C, a support set S and a query set Q are sampled i.i.d. from $\mathcal{D}^C := \{(x, y) \in \mathcal{D} : y \in C\}$. The goal is to learn the parameters of an algorithm $w \in \mathbb{R}^d$ so that when given a support set S, the algorithm will output a model that achieves a low loss value on the query set. We denote this loss value as a function of the algorithm parameter, support, and query set: $\ell : \mathbb{R}^d \times \{S\} \times \{Q\} \to \mathbb{R}$. To identify the optimal parameter of the algorithm w, we optimize the objective in [A] below:

$$\text{[A]: } \min_w \mathbb{E}_C \, \mathbb{E}_{(S,Q) \mid C} \, \ell(w, S, Q) \quad \equiv \quad \text{[B]: } \min_w \mathbb{E}_{S_p \sim \text{Unif}(\mathcal{P})} \, \mathbb{E}_C \, \mathbb{E}_{Q \mid C} \, \ell(w; S_p^C, Q) \tag{1}$$

Reformulating the meta-learning objective. We now give another view of this objective, which will motivate our proposed modified objective. Instead of sampling from the set of classes C when constructing a task, we consider first sampling uniformly from $\mathcal{P}$, a collection of support pools. A ***support pool*** $S_p \subseteq \mathcal{P}$ is a smaller subset of the entire dataset D that contains some examples from every class in D, i.e. $\{y : (x, y) \in S_p\} = [N]$. With S_p, we sample a subset of classes $C \subseteq [N]$ as before and take all the examples from S_p that are of the classes in C. This set of examples is now our support set which we denote by $S_p^C = \{(x, y) \in S_p : y \in C\}$. With the classes C determined, we can then sample the query set Q of examples from class C the same way as described previously. In this view, the marginal distribution of the support query set pair (S_p^C, Q) is exactly the same as that of (S, Q) discussed before. As a result, we can rewrite our meta-learning objective as in (1)[B].

This reformulation prompts us to first check how big the collection of support pools $\mathcal{P}$ typically is. In the most unconstrained form, $\mathcal{P}$ can contain all possible subsets of a dataset where we have at least one sample from every class ($\forall c \in [N]$). However, most meta-learning methods only train on episodic tasks where the support S is of a specific configuration, i.e., the support of each task is comprised of exactly n *ways* (classes) with each class having k *shots* (examples). For instance, when training on *miniImagenet* [30] 5-way 5-shot ($n = 5, k = 5$) tasks, all tasks that are sampled consist of exactly 5 shots from each of the 5 classes. Hence, it also makes sense to limit $\mathcal{P}$ to only support pools that obey a certain structure. In this work, we consider $\mathcal{P}$ to consist only of subsets of $\mathcal{D}$ that have exactly k samples from each class i.e., $\mathcal{P} = \{S_p : \sum_{(x,y)\in S_p} \mathbb{1}(y = c) = k,\ \forall c \in [N]\}$. The meta-training set of miniImagenet [23] has 64 classes with 600 examples each, resulting in $|\mathcal{P}| = \binom{600}{5}^{64} \approx 3 \times 10^{755}$ support pools. This naturally leads us to ask the question: *Can we use a much smaller number of support pools and still maintain reasonable meta-test performance?*

Fixing the support pool. Motivated the question above, we take the most extreme reduction approach: before training begins, we randomly construct one support pool $S_{p,0}$ and let $\mathcal{P} = \{S_{p,0}\}$ (just this single support pool) and solve for the corresponding optimization problem:

$$\min_w \mathbb{E}_C\, \mathbb{E}_{Q \mid C}\, \ell(w; S_{p,0}^C, Q). \tag{2}$$

Note that the expectation over S_p is removed because there is only a single possible realization of the random support pool. In terms of the total number of different *possible support sets* that the algorithm being learned can see, we have also achieved a reduction factor of $\binom{600}{5}^5$ times for the aforementioned miniImagenet example. In the rest of the paper, we first show the performance obtained by optimizing this reduced objective (2) for some of the best-performing meta-learning methods on multiple meta-learning problems and/or with different model architectures. We then make an initial attempt at understanding the effectiveness of using this fixed support pool objective.

4 Experiments

We now aim to understand the differences in generalization performance between the algorithm learned by optimizing (2), i.e. our proposed objective (which we denote FIX-ML), and the algorithm obtained by optimizing the original meta-learning objective (1) (which we denote ML).

For every FIX-ML experiment, we randomly sample k samples (shots) from each class in the dataset to form the support pool $S_{p,0}$, which is then fixed throughout training. Here, different runs of FIX-ML on the exact same problem and meta-learning method can use different (but fixed) support pools. Later in this section, we will discuss how this randomness affects meta-test performance over multiple FIX-ML runs. For an unbiased estimation and a fair comparison, when we evaluate a FIX-ML learned algorithm on the meta-validation or meta-test set, we compute the loss and accuracy using the objective given in (1) (i.e., an average over all possible support pools), just as we would for the ML learned algorithm.

To fully evaluate the performance differences, we compare FIX-ML and ML along three axes:

1. **Meta-learning methods:** We focus our initial experiments on *protoyypical-networks* (PN) [25] which uses a non-parametric last layer solver, as *Protonets* is a highly competitive ML approach, and also has the benefit of training faster and with less memory than competitors such as *SVM* [17], *Ridge Regression* (RR) [3] However, we also experiment with SVM and RR-based solvers to confirm that the trends observed with Protonets generalize to these methods. As initialization-based methods are harder to optimize for highly over-parameterized models, we defer such evaluations to future work.

2. **Datasets:** Our study uses two of the most widely-used benchmarks in few-shot learning: (i) *miniImagenet* (*mini*) [30], which consists of 100 classes of 84×84 images, split into 64 train, 16 validation and 20 test, and (ii) *CIFAR-FS* (*cifar*) [3] with an identical split but with 32×32 images. Additionally for the SVM method we elect to compare on the more difficult *FC-100* (FC) dataset, as the SVM-based solver has achieved significantly superior performance on this benchmark [17].

3. **Architectures:** We conduct our main experiments using *Resnet-12*, as many meta-learning methods [2, 21] have relied on this architecture to achieve good performance. Additionally, we experiment with another wider but shallower over-parameterized architecture, *WideResNet-16-10* [32] (also used by [24]), and *Conv64*, which was used by some initial works in meta-learning [10, 25].

D / Alg	Train Task	Test Task	ML	FIX-ML
mini / PN	5w5s15q	5w5s15q	76.28%	**77.30%**
mini / PN	20w5s15q	5w5s15q	77.43%	**77.80%**
mini / PN	64w5s5q	5w5s15q	76.25%	**78.19%**
mini / PN	5w1s15q	5w1s15q	60.74%	**61.27%**
mini / PN	64w5s5q	5w1s15q	59.89%	**61.01%**
cifar / PN	5w5s15q	5w5s15q	83.72%	**84.04%**
cifar / PN	20w5s15q	5w5s15q	83.97%	**84.43%**
cifar / PN	64w5s5q	5w5s15q	83.82%	**84.63%**
cifar / PN	5w1s15q	5w1s15q	71.53%	**72.20%**
cifar / PN	20w1s15q	5w1s15q	68.99%	**69.83%**
cifar / RR	5w5s6q	5w5s15q	84.69%	**84.96%**
FC / SVM	5w15s6q	5w5s15q	54.26%	**55.11%**

(a)

Architecture	ML	FIX-ML
Conv-64	**69.65%**	67.33%
Resnet-12	76.28%	**77.30%**
WRN-16-10	74.86%	**76.33%**

(b)

	cifar 5w5s Test		*cifar* 5w1s Test	
	ML	FIX-ML	ML	FIX-ML
	84.41%	84.37%	71.80%	72.75%
	83.49%	84.96%	70.80%	72.20%
	84.34%	84.34%	71.37%	71.76%
	84.02%	84.39%	71.75%	72.41%
	83.59%	84.09%	71.96%	71.90%
avg:	83.97%	84.43%	71.53%	72.20%
std:	0.37%	**0.28%**	0.41%	**0.35%**

(c)

Figure 1: Performance gains observed upon reducing support set diversity by fixing a single support pool. In (a) we compare FIX-ML and ML across datasets, algorithms and train/test configurations (XwYsZq := X-ways, Y-shots and Z-query), in (b) we evaluate the architecture's role in a comparison between ML and FIX-ML on *mini*-5w5s tasks and in (c) we show how the meta-test performance varies across runs for both FIX-ML and ML.

Main Results. Figure 1 consolidates the results of our experiments on different datasets with a myriad of training task configurations. Surprisingly, contrary to our intuition about task diversity, we can clearly see that optimizing the FIX-ML objective performs similarly or in many cases better with respect to its ML counterpart. Specifically, we see that on all of the *mini* and *cifar* 5w1s and 5w5s evaluations, Protonets trained with FIX-ML achieve better performance. In addition, for each of these evaluations, we consider training with different number of ways and shots as proposed in [17, 25]. Our results also confirm that FIX-ML is better for a range of meta-training specifics. Besides, for other meta-learning methods like SVM and RR, fixing the support pool can also improve performance. For all the experiments in Figure 1(a) we use the Resnet-12 backbone. In order to further validate our findings, we show results on two other architectures in Figure 1(b). WRN-16-10 has roughly 50% more parameters than Resnet-12 and we see that FIX-ML continues to perform better than ML in the over-parameterized regime. On the other hand, it seems that fixing the support pool for shallower backbones like Conv64 hinders the meta-test performance. For this we hypothesize that FIX-ML is effective for over-parameterized models and algorithms but less so for shallower models with fewer parameters. Further analysis on how model architecture and over-parameterization affects FIX-ML is a future research direction.

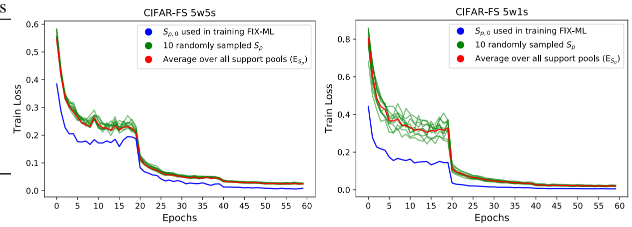
Recall that earlier in this section, we suspected that fixing the support pool randomly prior to the start of training may induce higher variance in the final meta-test performance. From the observations in Figure 1(c), we confirm that this is **not** true. We compare FIX-ML and ML on *cifar*-5w5s and *cifar*-5w1s over five different runs each, where for every run of FIX-ML a different support pool was fixed. We can clearly see that the performance standard deviation is not only **not worse** but actually **better** than ML. Finally, in Figure 2(a) we see that Protonets trained using FIX-ML objective achieves competitive results on *mini*-5w5s when compared with other other meta-learning methods and with [6], which proposes a simple transfer learning based objective devoid of any self-supervised learning [20] tricks or a transductive setting [13].

Implementation. We follow the learning rate schedule and data augmentation scheme of [17]. The test set evaluation for all methods was done by sampling 2000 random tasks with different (S, Q) pairs from the test (novel) classes of the respective datasets. We use a task batch size of 4-8 when training with lower number of ways (e.g., 5) due to observed improvements in convergence and generalization. Additionally, in some cases varying the number of query points per class (in a single task) further improves FIX-ML performance (but has no impact on ML).

5 Discussion

To understand the improved meta-test performance of the FIX-ML-learned algorithms, we first further analyze their performance **on the meta-training set**. Evaluating the FIX-ML-learned algorithm's performance averaged over all possible pools (equivalent to the standard ML training loss) is important because, according with generalization theory using uniform convergence bounds (e.g. VC dimension,

Method	Backbone	*mini* 5w5s
MatchingNet [30]	Conv64	63.48%
RelationNet [26]	Conv64	66.60%
DCO [2]	Conv64	68.40%
LEO [24]	WRN-28-10	77.59%
Chen et al. [6]	Resnet-10	75.90%
TADAM [21]	Resnet-12	76.70%
MetaOptNet-SVM [17]	Resnet-12	77.40%
Protonet	Resnet-12	77.43%
Protonet FIX-ML	Resnet-12	**78.19%**



(a) (b) (c)

Figure 2: In (a) we compare FIX-ML learned Protonets with other popular methods[1] on *mini*-5w5s. On *cifar*-5w5s (b) and *cifar*-5w1s (c) we evaluate the FIX-ML-learned algorithm after each training epoch using three criterions; *blue*: FIX-ML's loss on the specific support pool ($S_{p,0}$) used in its objective (2); *red*: the meta-learning objective in (1)[B] and *green*: the objective in (2) but on 10 randomly sampled support pools different from $S_{p,0}$.

Rademacher complexity [1]), the test loss tracks training loss more closely with more training samples. In the case of meta-learning, the "samples" are tasks in the form of (S, Q) pairs. Therefore, we see that an algorithm's ML-training loss which considers all possible (S, Q) pairs, should track the algorithm's loss on the meta-test set more reliably than the FIX-ML training loss which uses only a single support pool (inducing much fewer (S, Q) pairs).

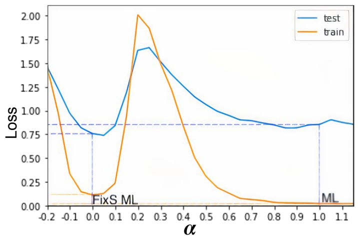
Naively, one might think that because FIX-ML is optimized on a **single** fixed support pool, the training loss on any other support pool would be much worse. To verify this idea, we re-evaluate the FIX-ML-learned algorithms **1)** over 10 other fixed support pools by replacing $S_{p,0}$ in (2) with $S_p \neq S_{p,0}$; **2)** on the original meta-learning objective (1)[B] by averaging over all possible support pools S_p. We conduct this evaluation for Protonet algorithms trained on *cifar*-5w5s and *cifar*-5w1s tasks, for the algorithm snapshots across the entire 60 training epochs. Here, the random class set C in (1)[B] is set to always contain exactly 5 classes ($|C| = 5$) in order to match the meta-test scenario. The results are shown in Figure 2(b),(c). As we see in both plots, despite optimizing for one fixed support pool (blue curve), the training losses of other support pools (green curves) and the ML training loss (red curve) are all decreasing consistently in the same trend as the blue curve throughout the entire optimization trajectory. To understand this consistent decrease, we note that the gradient of FIX-ML objective is a biased gradient of the original ML objective. When the gradient bias is not too large, stochastic gradient descent using a biased gradient can still lead to a reduction in objective loss and convergence [4], which is what we observe for the ML objective when using biased stochastic FIX-ML gradients.

However, this does not give us a complete picture for meta-training. Even though ML training loss is considerably reduced for the FIX-ML-learned algorithm, we need to compare this value against ML-training loss of the corresponding ML-learned algorithm to understand the meta-test performance difference. In Figure 3(c), we compare the end-of-training FIX-ML solutions with the corresponding ML solutions using the ML training objective (1)[B] on 4 different dataset/task configurations. We see that FIX-ML still consistently achieves higher ML-training losses than their ML counterparts. With a relatively lower ML test loss but a much higher ML training loss, the FIX-ML solution must have a smaller generalization gap than that of the ML solution. Since there isn't a single best way to understand neural networks' generalization gap, we look at some of the recently proposed metrics:

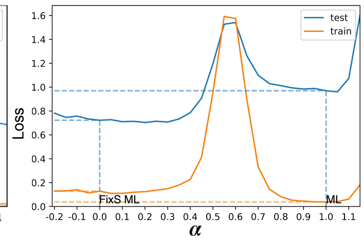
1) Sharpness of the training loss landscape: A popular hypothesis (dating back to [12]) is that the flatness of the training loss landscape is correlated with (and possibly leads to) a smaller generalization gap [5, 16]. Several sharpness metrics based on this hypothesis were called into question by [8], showing that a functionally equivalent neural network with differently scaled parameters could have arbitrarily different sharpness according to these metrics. However, it was argued in [15] that even though sharp minima with similar test performance as flat minima do exist, stochastic gradient descent will not converge to them. While further understanding sharpness's relationship with generalization gap is still an ongoing research topic, we still consider two sharpness analyses here.

1a) Sharpness visualization using 1d interpolation plots was first proposed by [16] and also used by [14] to compare the flatness of two different solutions' training loss landscape when explaining the two solutions' generalization gap differences. For two models with the same neural architecture but different weights, training and test loss are evaluated at model parameters that are different linear interpolations of the two models' weights. In our case, we interpolate between the parameters of

[1]For MetaOptNet-SVM we report the performance observed by [17] without label smoothing.



(a)



(b)

end-of-training ML train loss

Data/Task	ML	FIX-ML
cifar 5w5s	0.001	0.016
cifar 5w1s	0.012	0.013
mini 5w5s	0.03	0.11
mini 5w1s	0.123	0.295

(c)

Dataset	ML		FIX-ML	
	$\frac{\mathrm{Tr}(\boldsymbol{C})}{\mathrm{Tr}(\boldsymbol{F})}$	Gen. Gap	$\frac{\mathrm{Tr}(\boldsymbol{C})}{\mathrm{Tr}(\boldsymbol{F})}$	Gen. Gap
cifar	2.58	0.46	1.14	0.42
mini	12.37	0.86	9.22	0.63
FC	6.87	0.37	4.49	0.30

(d)

Figure 3: In (a),(b) (two different runs) we plot the train and test loss (averaging over all support sets) for network weights obtained by linearly interpolating between the FIX-ML (w_{fml} at 0.) and ML (w_{ml} at 1.) Protonet solutions for *mini*-5w5s tasks. We evaluate (1)[B] with $w = (1-\alpha)w_{fml} + \alpha w_{ml},\ \alpha \in [-0.2, 1.2]$. In (c) we compare the ML train losses for FIX-ML and ML learned Protonets at the end of training. In (d) we show the correlation between the generalization gap and $\frac{\mathrm{tr}(\boldsymbol{C})}{\mathrm{tr}(\boldsymbol{F})}$ on 5w5s tasks for *cifar*, *mini* (Protonet) and FC (SVM).

end-of-training FIX-ML-learned algorithm and ML-learned algorithm from Protonet for miniImagenet 5w5s. This result is shown in Figure 3(a),(b). In 3(a), we see that FIX-ML is located at a relatively "sharper" minima than ML despite having a smaller generalization gap. This is inconsistent with observations in [14, 16] possibly because in our case, the loss landscape between meta-train and meta-test are **not** necessarily shifted versions of each other (confirmed also in Figure 3(b)). Instead, we see that the trends of these two graphs align well with each other. Therefore, we cannot solely rely on 1d visualization-based sharpness analysis to explain the generalization gap.

1b) In addition to visualizations, **quantitative measures of the training loss landscape sharpness** has also been used. Specifically, one notion of the loss landscape sharpness is in the curvature of the loss function. This is typically measured by the maximum eigenvalue [18] or the trace $(\mathrm{tr}(\mathbf{H}))$ of the Hessian matrix $(\mathbf{H})$ [15] for the parameters at the end of training . However, directly computing $\mathbf{H}$ accurately is both computationally expensive and memory-wise "impossible" for over-parameterized models like Resnet-12. Thus, $\mathbf{H}$ is typically approximated with the Fisher information matrix $(\mathbf{F})$ at the same parameter [28]. $\mathrm{tr}(\mathbf{F})$ can be easily estimated without constructing the entire matrix $\mathbf{F}$. However, $\mathbf{F}$ is only equal to $\mathbf{H}$ when the parameters are the true parameters of the training set (in practice, achieve training loss extremely close to 0). On the other hand, this approximation is very crude when the training loss is much larger than 0 in the case of FIX-ML (Figure 3(c)). Hence, $\mathrm{tr}(\boldsymbol{H})$ for FIX-ML's parameters cannot be accurately estimated using $\mathrm{tr}(F)$. As a result, for our analysis, this approach is not readily applicable.

2) Another quantitative measure which approximates the Takeuchi Information Criterion (TIC) [27] was empirically shown to correlate more with the generalization gap as compared to measures of the loss landscape's flatness [28]. Specifically, Thomas et al. (2020) proposes the quantity $\frac{\mathrm{tr}(\boldsymbol{C})}{\mathrm{tr}(\boldsymbol{F})}$, where $\boldsymbol{C}$ is the uncentered covariance matrix of the gradient with respect to the training data distribution. To evaluate how well this measure tracks the generalization gap difference between FIX-ML and ML, we evaluate this approximate TIC value for three pairs of ML and FIX-ML-learned algorithms in Figure 3(d). We see that for all three datasets, ML has both higher $\frac{\mathrm{tr}(\boldsymbol{C})}{\mathrm{tr}(\boldsymbol{F})}$ and larger generalization gap than its FIX-ML counterpart, indicating some possible correlation between these two. Thus it would be interesting to understand what intuitive property of the solutions is captured by this quantity and why optimizing the FIX-ML objective leads to solutions with lower $\frac{\mathrm{tr}(\boldsymbol{C})}{\mathrm{tr}(\boldsymbol{F})}$.

6 Conclusion

In this paper, we have proposed a simple yet effective modification of the meta-learning objective. We begin by re-formulating the traditional meta-learning objective, which motivates our approach of reducing the support set diversity. Then, we experimentally show that for a variety of meta-learning methods, datasets, and model architectures, the algorithms learned using our objective result in superior generalization performance relative to the original objective. We analyze this improvement by understanding the optimization loss and the generalization gap separately. While the optimization loss trade-off is easier to reason about using our framework, the generalization gap is still not fully understood based on several initial analyses, which paves the way for future work in this area.

References

[1] P. L. Bartlett and S. Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002.

[2] L. Bertinetto, J. F. Henriques, P. H. Torr, and A. Vedaldi. Meta-learning with differentiable closed-form solvers. *arXiv preprint arXiv:1805.08136*, 2018.

[3] L. Bertinetto, J. F. Henriques, P. H. S. Torr, and A. Vedaldi. Meta-learning with differentiable closed-form solvers. *arXiv preprint arXiv:1805.08136*, 2018.

[4] L. Bottou, F. E. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311, 2018.

[5] P. Chaudhari, A. Choromanska, S. Soatto, Y. LeCun, C. Baldassi, C. Borgs, J. Chayes, L. Sagun, and R. Zecchina. Entropy-sgd: Biasing gradient descent into wide valleys. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124018, 2019.

[6] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C. F. Wang, and J.-B. Huang. A closer look at few-shot classification. *arXiv preprint arXiv:1904.04232*, 2019.

[7] Y. Chen, X. Wang, Z. Liu, H. Xu, and T. Darrell. A new meta-baseline for few-shot learning. *arXiv preprint arXiv:2003.04390*, 2020.

[8] L. Dinh, R. Pascanu, S. Bengio, and Y. Bengio. Sharp minima can generalize for deep nets. *arXiv preprint arXiv:1703.04933*, 2017.

[9] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):594–611, 2006.

[10] C. Finn and S. Levine. Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm. *arXiv preprint arXiv:1710.11622*, 2017.

[11] C. Finn, K. Xu, and S. Levine. Probabilistic model-agnostic meta-learning. In *Advances in Neural Information Processing Systems*, pages 9516–9527, 2018.

[12] S. Hochreiter and J. Schmidhuber. Flat minima. *Neural Computation*, 9(1):1–42, 1997.

[13] Y. Hu, V. Gripon, and S. Pateux. Leveraging the feature distribution in transfer-based few-shot learning. *arXiv preprint arXiv:2006.03806*, 2020.

[14] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson. Averaging weights leads to wider optima and better generalization. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, pages 876–885. Association For Uncertainty in Artificial Intelligence (AUAI), 2018.

[15] S. Jastrzębski, Z. Kenton, D. Arpit, N. Ballas, A. Fischer, Y. Bengio, and A. Storkey. Three factors influencing minima in sgd. *arXiv preprint arXiv:1711.04623*, 2017.

[16] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

[17] K. Lee, S. Maji, A. Ravichandran, and S. Soatto. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10657–10665, 2019.

[18] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein. Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems*, pages 6389–6399, 2018.

[19] J. Liu, F. Chao, and C.-M. Lin. Task augmentation by rotating for meta-learning. *arXiv preprint arXiv:2003.00804*, 2020.

[20] P. Mangla, N. Kumari, A. Sinha, M. Singh, B. Krishnamurthy, and V. N. Balasubramanian. Charting the right manifold: Manifold mixup for few-shot learning. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 2218–2227, 2020.

[21] B. N. Oreshkin, P. Rodriguez, and A. Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. *arXiv preprint arXiv:1805.10123*, 2018.

[22] 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.

[23] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. 2016.

[24] A. A. Rusu, D. Rao, J. Sygnowski, O. Vinyals, R. Pascanu, S. Osindero, and R. Hadsell. Meta-learning with latent embedding optimization. *arXiv preprint arXiv:1807.05960*, 2018.

[25] J. Snell, K. Swersky, and R. Zemel. Prototypical networks for few-shot learning. In *Advances in neural information processing systems*, pages 4077–4087, 2017.

[26] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1199–1208, 2018.

[27] K. Takeuchi. The distribution of information statistics and the criterion of goodness of fit of models. *Mathematical Science*, 153:12–18, 1976.

[28] V. Thomas, F. Pedregosa, B. Merriënboer, P.-A. Manzagol, Y. Bengio, and N. Le Roux. On the interplay between noise and curvature and its effect on optimization and generalization. In *International Conference on Artificial Intelligence and Statistics*, pages 3503–3513. PMLR, 2020.

[29] S. Thrun and L. Pratt. *Learning to Learn*. Springer, 1998.

[30] 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.

[31] M. Yin, G. Tucker, M. Zhou, S. Levine, and C. Finn. Meta-learning without memorization. *arXiv preprint arXiv:1912.03820*, 2019.

[32] S. Zagoruyko and N. Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[33] C. Zhang, A. Khan, S. Paternain, V. Kumar, and A. Ribeiro. Learning task agnostic sufficiently accurate models. *arXiv preprint arXiv:1902.06862*, 2019.