Contextualizing Enhances Gradient Based Meta Learning for Few Shot Image Classification

Evan Vogelbaum* *MIT EECS* evanv@mit.edu

Rumen Dangovski* *MIT EECS* rumenrd@mit.edu

Li Jing *MIT Physics* ljing@mit.edu

Marin Soljačić *MIT Physics* soljacic@mit.edu

***Abstract*—Meta learning methods have found success when applied to few shot classification problems, in which they quickly adapt to a small number of labeled examples. Prototypical representations, each representing a particular class, have been of particular importance in this setting, as they provide a compact form to convey information learned from the labeled examples. However, these prototypes are just one method of representing this information, and they are narrow in their scope and ability to classify unseen examples. We propose the implementation of *contextualizers*, which are generalizable prototypes that adapt to given examples and play a larger role in classification for gradient-based models. We demonstrate how to equip meta learning methods with contextualizers and show that their use can significantly boost performance on a range of few shot learning datasets. We also present figures of merit demonstrating the potential benefits of contextualizers, along with analysis of how models make use of them. Our approach is particularly apt for low-data environments where it is difficult to update parameters without overfitting. Our implementation and instructions to reproduce the experiments, available at https://github.com/naveace/proto-context/, are thoroughly tested on MIT SuperCloud, and scalable to other state-of-the-art HPC systems.**

***Index Terms*—meta-learning, few-shot learning, attention mechanism**

I. Introduction

With the rise of deep learning, models have become remarkably successful at mastering challenging, large scale image classification tasks by training on enormous amounts of data [1, 24, 13, 2, 12]. However, this reliance on "big data" remains a key flaw in most deep learning models, limiting their use cases and making them expensive to train [61, 42, 57]. To address this problem, many have turned to few shot learning methods: algorithms and architectures designed to perform tasks such as image classification with very small amounts of task-specific data. Much progress has been made in this area [66, 62, 52, 59], however many of these models nonetheless remain specific to the tasks for which they are designed to solve [48, 36, 28, 46]. Coming from another line of research, self-attention mechanisms [65] have enabled the ongoing revolution of large universal self-supervised models trained on a large quantities of data, which are extremely capable in numerous tasks [44, 6, 10, 69, 7]. Instead, when data is scarce, in a preliminary study [67] we have explored how the attention mechanism can help with efficient task adaptation.

Another method of addressing the "big data" problem with more generalizability has been found in *meta learning* [25], a field very closely related to few shot learning, in which models and algorithms are designed to quickly fine-tune on new tasks either through transferring knowledge from previously seen data samples or optimizing themselves for fast adaptation in just a few gradient updates [64]. An example of the latter is *model agnostic meta learning* (MAML), an algorithm designed by Finn et al. [16] that excels at training neural networks for meta learning problems, and which was improved upon by Nichol et al. [40], Antoniou et al. [3], Behl et al. [5], Song et al. [56] and many more.

Metric learning, such as that proposed by Snell et al. [52], has also been shown to be successful in the area of few shot learning. *Prototypes*—averages of examples that share the same class—were used by Snell et al. [52] to make classification decisions by measuring how close unclassified samples are to each prototype. Triantafillou et al. [63] helped marry prototypes with MAML through *ProtoMAML*, a version of MAML that initializes the *head* of the network (the final layer that maps features to output classes) with scaled versions of the prototypes; thus it is end-to-end trainable with MAML.

However, current models that combine few shot learning with meta learning fail to produce *task specific* initializations for heads. In addition, although the ideas of task-specific feature spaces have been explored [41, 43], these have yet to be combined with the advances in head initialization proposed in [63]. Finally, most meta-learning models rely on several gradient steps to adapt the model to a particular task and can be prone to overfitting, a problem that has gained attention before [3, 75, 36, 45]. We address these problems through the use of "contextualizers:" a generalization of prototypes that produce (*i*) a task-specific initialization (*ii*) a task-specific feature space and (*iii*) achieve near peak accuracy in as little as one inner loop gradient step.

In this paper, we focus on applying contextualization to the problem of few shot classification. This setup consists of training on separate *tasks*, where each task consists of a *support* set $\mathbf{S} = (\mathbf{X}, \mathbf{y}) \equiv \{(\mathbf{x}_j, y_j)\}_{j=1}^{|\mathbf{S}|}$ and a *query* set $\mathbf{Q} =$

* Equal contribution.

$(\mathbf{X}', \mathbf{y}') \equiv \left\{(\mathbf{x}'_j, y'_j)\right\}_{j=1}^{|\mathbf{Q}|}$ of images and labels [52]. Each task consists of *n* different classes with *k* samples per class in the support set. This type of learning is referred to as *n-way k-shot*, and we will use this terminology throughout [66]. We fine-tune the model on the samples in $\mathbf{S}$ and then test the fine-tuned model on the samples in $\mathbf{Q}$. The performance on $\mathbf{Q}$ is used to *meta train* the model, which in the *second order* version of MAML consists of propagating the loss through the update made from the gradient on $\mathbf{S}$ to the base parameters, and thus requires the calculation of Hessians (second order gradients). This method of updating can be impractical due to size and time constraints; hence, we focus on the *first order* setting of MAML [16]. In this setting, we begin each task with a model at a set of *base parameters* $\boldsymbol{\theta}$. Next, we fine-tune the model on $\mathbf{S}$ and then evaluate it on $\mathbf{Q}$. Finally, we apply the gradient of the loss on $\mathbf{Q}$ with respect to the tuned parameters directly to the base parameters as follows:

$$
\begin{aligned}
L_{\mathbf{S}}\left(\boldsymbol{\theta}^{(0)}\right) &= L\left(\boldsymbol{\theta}^{(0)}, \mathbf{S}\right) && \text{loss on the support } \mathbf{S} \\
\boldsymbol{\theta}^{(i+1)} &= \boldsymbol{\theta}^{(i)} - \alpha \cdot \nabla_{\boldsymbol{\theta}^{(i)}} L_{\mathbf{S}}\left(\boldsymbol{\theta}^{(i)}\right) \\
& && \text{inner loop steps } i = 1, \ldots, M \\
L_{\mathbf{Q}}\left(\boldsymbol{\theta}^{(M)}\right) &= L\left(\boldsymbol{\theta}^{(M)}, \mathbf{Q}\right) && \text{loss on the query } \mathbf{Q} \\
\boldsymbol{\theta}^{(0)} &= \boldsymbol{\theta}^{(0)} - \beta \cdot \nabla_{\boldsymbol{\theta}^{(M)}} L_{\mathbf{Q}}\left(\boldsymbol{\theta}^{(M)}\right), \\
& && \text{outer loop update of base parameters } \boldsymbol{\theta}^{(0)}
\end{aligned}
$$

where L is our loss function and α and β are our *inner loop* and base update (*outer loop*) learning rates. We modify the MAML algorithm to allow for the creation of task-specific intializations for the *head* of the model as well as the creation of task-specific feature spaces. Our main contributions are:

1) We propose a new method of inference in meta learning that combines task-specific feature spaces with task-specific head initializations and demonstrate its ability to outperform conventional gradient methods on a range of few shot learning datasets.
2) We present empirical and theoretical analysis of our contextualization mechanism demonstrating that our contextualizers play a major role in the model.
3) We introduce a new figure of merit, *intra-class similarity*, to measure upstream benefits our contextualizers may have in the training process.

The rest of the paper is organized as follows. In Section II we discuss related work. In Section III we introduce the concept of contextualization. In Section IV we introduce and motivate our method. In Section V we describe the datasets we use. In Section VI we present our experiments. In Section VIII we present an empirical analysis of our experiments. In Section A (in the appendix) we present a theoretical analysis of our method. In Section IX we conclude and discuss future work.

II. Related Work

A. Few shot learning

Few shot learning is, broadly, the task of making inference with use of few labeled examples [35, 37]. To solve this fairly formidable problem, many fields have been explored including meta learning [58, 36, 48, 16], metric learning [53, 66, 9, 29], as well as more broad approaches [14, 62, 60, 33]. Closest to our work are methods that approach few shot learning through forms of contextualization [8, 41, 20]. We were particularly encouraged by the results of Ye et al. [72], who found that attention mechanisms are a very powerful set-to-set function for few shot learning. However, no approach that we are aware of makes use of attention not only to construct task-specific initializations, but also to modify features in a meta-learning model.

B. Meta learning

Meta learning algorithms create models that quickly adapt to new tasks [73]. The goals of meta learning fit those of few shot learning very well, and many meta learning ideas have been successfully applied to few shot learning [27, 74, 23]. Approaches vary, but can broadly be split into those that focus on using architectures or learned gradient updates [34, 71, 70, 41, 17] and those that make use of learned initializations which can quickly adapt to new tasks through gradient descent [16, 40, 50, 11, 15, 18]. The latter approach is much closer to our work. We take this approach one step further and construct *task-specific* initializations before fine-tuning occurs along the lines of Triantafillou et al. [63]. In addition, we produce a task-specific feature space through our contextualizers that is somewhat similiar to the ideas presented in [75, 43]. Our mechanism is very different, however, using self-attention to learn our feature space. Furthermore, none of these approaches explore both empirically and theoretically the benefits provided by contextualization in very challenging environments, such as allowing only 1 gradient step for adaptation.

C. Metric learning

Deep metric learning produces algorithms and models which are able to construct "metrics" by which images can be compared or retrieved. Much work focuses on loss functions in metric learning [19, 55, 54, 39, 68], and some of this work has led to methods useful for few shot learning [52, 52]. Metric learning has also been used in one-shot learning [31, 66]. This work is useful in comparison to our gradient-based methods; however, we seek to extend it by using strong metrics for initialization of gradient learners. More recenlty, in state of the art self-supervised learning, Gidaris et al. [21] use a dynamic predictor module to learn useful representations without labels, which can also be adapted in few-shot settings. The dynamic predictor is contextualized at every step to an online bag-of-visual-words, which resembles our contextualziation methods. Inspired by this, for future work we will pursue our contextualization methods for improving self-supervised learning with specific emphasis on few-shot learning.

III. The Concept of Contextualization

Concretely, Contextualization is a modification to the output features of a Feature Extraction Model (such as a Deep Convolutional Neural Network) using a set of *contextualizer* vectors which develops a task-specific feature space.

A. Model

We denote our feature extractor by $f_{\boldsymbol{\theta}}$, parameterized by the weights $\boldsymbol{\theta}$, the contextualization mechanism by $s_{\boldsymbol{\phi}}$ (we use a self-attention mechanism [65] in our model), parametrized by the weights $\boldsymbol{\phi}$, the predictor function (head) by $h_{\boldsymbol{w},b}$, parametrized by the weights $\boldsymbol{w}$ and bias b, and the loss as L. For notation, we note that while the above $\boldsymbol{\theta}$ referred to any set of parameters being trained via MAML, for the rest of the paper it refers specifically to the parameters of the feature extractor. In addition, because we sometimes discard parts of the output of the self-attention mechanism that implements contextualization, we write its output as $b_1, b_2 = s_{\boldsymbol{\phi}}(a_1, a_2)$ where b_1 is the attended version of a_1 and likewise for b_2 and a_2. If we discard an output, we write it as $b_1, \_ = s_{\boldsymbol{\phi}}(a_1, a_2)$ where _ is the discarded output.

B. Contextualization of features

Previous work has theorized that the $f_{\boldsymbol{\theta}}$ trained by MAML produces highly generalizable features [45]. While this is an appealing property, we believe that performance can be boosted by adapting these generalized features to be specific to the task at hand. We do this in conjunction with initialization of the head, producing a highly task-specific model even before updates through gradient descent.

We present the general mechanism of contextualization for a model's features in Figure 1. Samples go through $f_{\boldsymbol{\theta}}$. We concatenate the extracted features with a set of contextualizers in the same feature space and feed them through $s_{\boldsymbol{\phi}}$. This mechanism provides updates to the features, creating a task-dependent feature space. We add these updates to our feature representations via a skip-connection. Finally $h_{\boldsymbol{w},b}$ produces a classification using the updated representations. In the following section we specify further details.

IV. Head Initializations

In this section, we (*i*) conduct a simple study to demonstrate the importance of intitializing the head properly for each task, (*ii*) introduce the use of prototypical representations for head initialization, and (*iii*) in light of (*i, ii*) we marry our concept of contextualization with that of prototypes.

A. Head initialization is important

In Table I we present the *meta test* results for a simple study that aims to understand how important the initialization for the head is. The regular experiment is MAML. In the second experiment, we train with MAML and randomize the head before the inner loop during meta testing. In the last experiment, during both meta training and testing we start the inner loop with a random initialization for the head to see if the network can learn to overcome the random initialization.

TABLE I

MAML with different initialization schemes for the head of the network. Accuracy in %. Experiments performed with five inner loop steps. Mean and standard deviation from three trials on the full test set. The dataset is Mini-ImageNet. All experiments are 5-way.

Initialization	1-shot	5-shot
Regular	49.16 ± 0.27	66.61 ± 0.55
Random Head (test only)	34.89 ± 1.00	45.13 ± 0.65
Random Head (train & test)	34.96 ± 0.95	59.60 ± 0.34

We observe significant degradation in accuracy as the meta initialization is lost via randomization. This suggests that fine-tuning through the inner loop is not enough to produce a strong task-specific head, and thus that initialization matters. We use these insights to motivate our exploration of new ways to initialize the head for each task.

B. Prototypes as initializations

ProtoMAML [63] builds on MAML by bringing the concept of prototypes into the MAML training process. Prototypes are vectors meant to be representative of a particular class, and are computed by averaging together the feature representations of support samples for a given class a

$$\mathbf{X}_{(a)} = \left\{ \mathbf{x}_j^{(a)} | 1 \leq j \leq k, y_j = a \right\} \subseteq \mathbf{S} \tag{1}$$

$$\mathbf{p}^{(a)} = \frac{1}{k} \sum_{\mathbf{x} \in \mathbf{X}_{(a)}} f_{\boldsymbol{\theta}}(\mathbf{x}). \tag{2}$$

ProtoMAML then initializes the head of the network with weights equal to $2\mathbf{p}^{(a)}$ and biases equal to $-\left\|\mathbf{p}^{(a)}\right\|_2^2$ for $a = 1, \ldots n$ as follows [63, 52]:

$$h_{\boldsymbol{w}} = \left[2\mathbf{p}^{(1)}, \ldots, 2\mathbf{p}^{(n)} \right]^\top, \tag{3}$$

$$h_b = \left[-\left\|\mathbf{p}^{(1)}\right\|_2^2, \ldots, -\left\|\mathbf{p}^{(n)}\right\|_2^2 \right]^\top. \tag{4}$$

C. ProtoContext

In Figure 2 we introduce *ProtoContext*: our method of generalizing prototypes which can leverage ProtoMAML's initialization to produce even stronger, task-specific head initializations. ProtoContext collects prototypes through Equation (1) and Equation (2), contextualizing them via self-attention *using the support examples* as follows:

$$\mathbf{C}, \_ = s_{\boldsymbol{\phi}} \left(\mathbf{P}, f_{\boldsymbol{\theta}} \left(\mathbf{S} \right) \right),$$

where $\mathbf{C}$ is a matrix with the contextualized prototypes for each class $\mathbf{c}^{(a)}$ and $\mathbf{P}$ is a matrix of the prototypes $\mathbf{p}^{(a)}$. Feeding the prototypes through the contextualization mechanism *before* initialization ensures that the head is initialized with information about the task at large, and not just one particular class. We initialize the head h in the manner detailed in Equation (4), using $\mathbf{c}^{(a)}$ instead of $\mathbf{p}^{(a)}$.

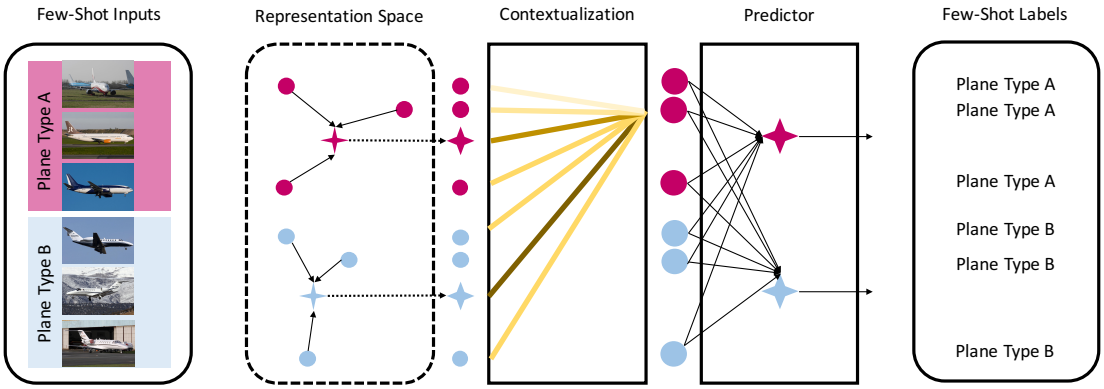

Fig. 1. Rough sketch of our ProtoContext method. Increased size of the shapes indicates updated representations. Dotted horizontal lines in the Representation Space indicate a parametrized transformation of the prototypes. Darker shades of yellow color indicates stronger attention connection for a particular representation. The predictor compares the contextualized input representations to the contextualized prototypes to make a prediction. Bias in the predictor is ignored for simplicity. By making the forward pass of the network task specific we can distinguish between Planes A and B.

With this head initialization we then initalize our contextualization vectors $\mathbf{C}$ using one of two methods we desctibe below.

Algorithm 1 (in the appendix) presents the first method that uses the weights of the head of the network as contextualization, with the understanding that samples whose features are similar to a particular layer of the head are likely to belong to the class that layer of the head connects to. In this case, changes in the contextualization are due to gradient updates applied to the head.

Algorithm 2 (in the appendix) is as an alternative method of contextualization that allows for the context to be updated in a non-gradient manner completely independent of the head. This contextualization begins with a set of contextualized prototypes and continuously updates them through self-attention by setting them equal to the output of a self-attention mechanism during each forward pass. Our hypothesis is that in this form of contextualization, the context will become a non-gradient method of transferring information useful for inference on the task at hand from the support set to the query set.

D. Xavier Initialization

Although it is not proposed in [63], we find empirically that scaling the last layer initialization used in ProtoMAML, ProtoContext, and PCABaseline to the magnitudes of *Xavier initialization* proposed in Glorot and Bengio [22] bolstered convergence and in some cases made convergence possible for our models. Specifically, if the initializations for the weights and bias are respectively $h_{\mathbf{w}}$ and h_b, the feature size is d_{feature} and the output classes are n, we compute as follows

$$f = \frac{1}{\max(h_b)} \sqrt{\frac{6}{d_{\text{feature}} + n}}$$

and then rescale elementwise the intializations as $f \cdot h_{\mathbf{w}}$ and $f \cdot h_b$ respectively.

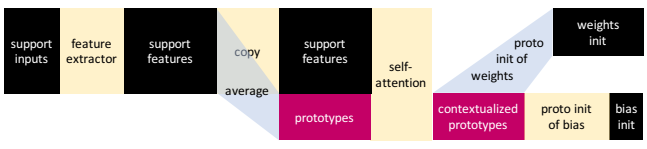

Fig. 2. Contextualized prototypes form the predictor $h_{\boldsymbol{w},b}$'s initialization.

E. Why marry contextualization and the head?

This last fact, that our initial contextualizers are proportional to the contextualized prototypes, is very useful as it provides a mechanism by which our model can leverage self-attention to boost prediction. In the beginning of the inner loop we obtain $\widetilde{\mathbf{X}}, \_ = s_{\boldsymbol{\phi}}(f_{\boldsymbol{\theta}}(\mathbf{X}), \mathbf{C})$ where $\widetilde{\mathbf{X}} = \{\widetilde{\mathbf{x}}_j\}$ is the collection of contextualized features. The contextualization by the self-attention constructs any contextualized feature $\widetilde{\mathbf{x}}$ (we removed the indexing here) as a linear combination of features and contextualized prototypes:

$$\widetilde{\mathbf{x}} \propto \sum_j u_j f_{\boldsymbol{\theta}}(\mathbf{x}_j) + \sum_a v_a \mathbf{c}^{(a)} + f_{\boldsymbol{\theta}}(\mathbf{x}),$$

where u_j and v_a are the weights given by the self-attention. Using Equation (4) for our head initialization the predicted logit for class a' is given by the a'-th component of the output logits as follows:

$$\begin{aligned}(h_{\boldsymbol{w}}\widetilde{\mathbf{x}} + h_b)_{a'} \propto & f_{\boldsymbol{\theta}}(\mathbf{x}) \cdot 2\mathbf{c}^{(a')} + \sum_j 2u_j \mathbf{c}^{(a')} \cdot f_{\boldsymbol{\theta}}(\mathbf{x}_j) \\ & + \boxed{\sum_a 2v_a \mathbf{c}^{(a')} \cdot \mathbf{c}^{(a)}} - \left\|\mathbf{c}^{(a')}\right\|_2^2 .\end{aligned} \tag{5}$$

The boxed term demonstrates why contextualization through self attention can be effective when combined with Equation (4). Each coefficient v_a is determined by the self-attention key of $\mathbf{x}$ and the self-attention query of $\mathbf{c}^{(a')}$. Thus, if $\mathbf{x}$ is of class a', the activation for class a' can be strengthed by our model learning a self-attention key that activates very highly with the self-attention query of the context vector for class

a'. In Section VIII we demonstrate empirical validation of this technique in action by analysing the attention patterns produced by our experiments.

We also believe contextualization may have positive effects on the updates to the network during the inner loop. We present a step towards theoretical analysis of the network's updates with details of this theory in Section A.

V. Datasets

We experiment on four standard few shot classification datasets: Omniglot, Mini-Imagenet, Tiered-Imagenet, and FGCV Aircraft (Airplanes).

A. Omniglot

The Omniglot dataset [32] consists of 1623 handwritten characters from 50 different alphabets. Within each alphabet, every character of the alphabet corresponds to a unique class and has several examples. In some uses of the Omniglot dataset [3, 16], a task is created by sampling classes across alphabets with no regard to the structure of the dataset. This makes examples from different classes fairly distinct, allowing for simple use of prototypical representations for classification. We use a version more similar to that presented by Lake et al. [32] which increases the similarity between classes in tasks sampled from the dataset, creating harder classification problems that requires more complex class representations, such as contextualizers. To realize this form of the dataset, we construct a sampling regime that incorporates the structure of the Omniglot dataset. Instead of sampling classes across all alphabets, we first select an alphabet uniformly at random and then select classes from that alphabet. If we assume that classes from the same alphabet will be more similar to each other than those from different alphabets, this regime increases the difficulty of the tasks drawn. We analyze results on 20-way classification, as with fewer ways all models solve the dataset nearly perfectly and there are no meaningful differences to be seen.

B. Mini-Imagenet

Mini-Imagenet [66] is a frequently used benchmark for meta learning. It is a subset of the Imagenet dataset containing 100 classes from the Imagenet dataset with 600 samples per class. We use the same splits as Antoniou et al. [3].

C. Tiered-Imagenet

Tiered Imagenet [48] is also a classic benchmark in meta learning. It is made of 608 classes grouped into 34 high level sets based on the Imagenet hierarchy, 28 for training, 6 for validation, and 6 for testing. We believe that by grouping similar classes together, harder tasks are produced. We note that this is an environment particularly apt for our contextualizer-based model, as the model is able to incorporate relations between the contextualizer and the samples to evaluate samples that are highly similar.

D. Aircraft

The FGVC Airplanes benchmark [38] is, like Tiered Imagenet, a more fine-grained classification benchmark. It consists of 102 different aircraft model variants with 100 images of each. The dataset also has two coarser groupings of airplanes into "Families" and "Manufacturers," however we disregard these in favor of the more challenging fine-grain task that uses variants as classes.

VI. Experiments

In our experiments we probe the extreme scenario of allowing only one gradient step to adapt to $\mathbf{S}$ before classifying $\mathbf{Q}$. We test on four few shot learning datasets and compare our models to the gradient methods of MAML and ProtoMAML as well as a non-gradient based method in Prototypical Networks [52].

We make special note that we used a harder split of the classic "Omniglot" dataset in our experiments.

We report results for both forms of contextualization described in Algorithms 1 and 2.

On all datasets, we measure performance for 1-shot and 5-shot learning. With the exception of Omniglot, we perform all experiments with 5 ways. At test time, we evaluate our model on 600 tasks constructed from unseen classes and compute the average accuracy across all test tasks. We use 750 warmup steps for the learning rate scheduler. We use a single self-attention with a single head. The inner loop learning rate is 0.1 for Omniglot, Quickdraw and Aircraft and 0.001 for Mini-ImageNet and Tiered-ImageNet. We use 8 tasks for the MAML inner loop for Omniglot, and 2 tasks for the rest of the datasets. The initial outer loop learning rate is 0.0001. We use Adam [30] for the outer loop (for simplicity of exposition, in Algorithm 1 we have replaced the Adam update rule with the standard gradient descent rule). Following [3], all experiments are performed on three separate seeds on the full test set with the average result across seeds reported.

A. Baselines

In addition to ProtoMAML, MAML and Prototypical Networks, we also compare our model to two baselines each of which contains one aspect of the improvements we implement in ProtoContext, but neither contains both. The first is a *PCA baseline*, which performs PCA on the prototypes and only keeps the principal components with singular values greater than 30% of the highest singular value (threshold found through grid search). Prototypes are replaced with the sum of their projections onto these components and the head is then intialized through Equation (4). This produces prototypes more specific to the task at hand, thresholding out noise, and is used as a comparison to our model's process of producing task-specific prototypes before initialization.

We also test a *Context Only* model which uses our contextualization scheme but does *not* re-initialize the head at every task, instead updating it in the same manner as MAML. This is meant to evaluate the importance of the task-specific feature-adaptation of our contextualizer.

In all models, our feature extractor is the same VGG [51] architecture used in MAML++. In the case of ProtoMAML, ProtoContext, and PCA Baseline the head of the architecture is initialized in the manner detailed in the main paper. This means the head is not meta trained, as it is re-initialized at the beginning of each task. For our ProtoContext and Context Only models, we make use of the Transformer architecture and learning rate scheduler introduced in [65] for our attention mechanism. Although we experimented with implementation of a full multi-head attention mechanism, we found that dropping some pieces of it improved performance. Our final results are reported using dot-product self-attention, and then feeding the results of that self-attention through a layernorm layer [4] and a feedforward layer with a skip connection around our attention mechanism and around our feedforward layer. We do not make use of multiple attention heads or stack several sub-layers instead opting for a single attention head that attends to samples and contextualizers. In all experiments, the key dimension is 64. The value dimension (equal to the feature size) is 64 for Omnliglot and 1200 for the remaining datasets.

B. Varying inner loop steps

Although our primary focus is on task adaptation with a single inner loop step, we also test our models with more inner loop steps to see how they make use of the additional gradient updates. This is a useful domain to explore, as the fine-tuning time scales linearly with the number of inner loop steps. When fewer gradient steps are allowed, it is likely that the contextualizer is more involved in the classification of query samples, as the model weights are less adapted to the task. With more inner loop steps, there is risk of overfitting to $\mathbf{S}$, and we seek to understand if our ProtoContext models suffer from this issue. We experimented with as many as 10 inner loop steps on all datasets, while all of our main results are with 1 step.

C. Intra-class similarity

The success of any method that relies on class representation, whether prototypes or contextualizers, is dependent on the ability of $f_{\boldsymbol{\theta}}(\cdot)$ to produce prototypes for each class that align closely to the samples of that class in the latent space. If this is not the case, then it will be difficult to classify new samples based off of the prototype alone. To measure the quality of this clustering, we propose a new metric as follows

$$\text{intra-class}(a) := \frac{1}{k}\sum_{j=1}^{k} \cos^{-1}\left(\frac{f_{\boldsymbol{\theta}}\left(\mathbf{x}_j^{(a)}\right)}{\left\|f_{\boldsymbol{\theta}}\left(\mathbf{x}_j^{(a)}\right)\right\|_2}\cdot\frac{\mathbf{p}^{(a)}}{\left\|\mathbf{p}^{(a)}\right\|_2}\right).$$

We use the measure of angular similarity rather than Euclidean distance as in [52] because logits are calculated through dot product with a head initialized to the prototypes, not through distance in Euclidean space. We also prefer this metric to cosine similarity since it has a more interpretable result (the angle between two feature vectors), although it is clear that the two metrics are directly related. We measure the intra-class similarity for the features produced by our ProtoContext model and ProtoMAML before fine-tuning on $\mathbf{S}$. For fair comparison, we only measure the features produced by $f_{\boldsymbol{\theta}}$ and do NOT measure the features modified by the context in ProtoContext. Evaluating this measure allows us to understand if the addition of a contextualizer produces feature extractors that create stronger latent representations.

VII. Results

A. 1-shot and 5-shot classification

In Table II we present our results on 1-shot and 5-shot classifications. In 1-shot classification, we outperform the baselines across all datasets. The 1-shot setting is challenging for gradient-based meta learners as there are not many data samples to use for adaptation and overfitting is a high risk. It becomes very useful in such a setting to have a task-specifc method of augmenting prediction such as the one we introduce through contextualization. In the 5-shot setting, we outperform on Omniglot, Tiered-ImageNet, and Airplanes. On Mini-ImageNet all gradient based models are outperformed by Prototypical Networks. In order to understand the limited performance of our model on this dataset, we present figures of the additional inner loop steps in the Supplemental Materials which suggest that ProtoContext is unable to develop a sufficiently strong initialization in this setting. We should note that our models outperform ProtoMAML and the PCA baseline, which also rely on the head initialization.

The two forms of contextualization we explore perform roughly similarly on Omniglot and Mini-ImageNet. On Tiered-ImageNet, using the head as contextualization leads to considerable benefits, while on Airplanes, using contextualized prototypes produces better results. We also see that in general, the Context Only baseline is more successful when given the head as contextualization. This makes sense, since although there is no explicit initialization of the head in this model, contextualization allows us to augment features with information about the head. Thus, features of samples in a given class can be made to align strongly with the weights in the head for that class. A surprising benefit of our contextualization method is the low variance across runs. Many methods have large standard deviations in the 1 and 5 shot settings (up to 9.73% for MAML). In contrast, across all datasets our ProtoContext model has low variance in its test accuracy.

B. Inner loop steps

In Figure 3 we demonstrate our results from varying the number of inner loop steps the model is allowed before testing on the query set on Tiered-ImageNet. We see that in both the 1-shot and the 5-shot settings, our model both begins at and stays at a high level of accuracy. These results indicate that ProtoContext is able to effectively develop a very strong initialization for the head that requires little task-specific fine-tuning. In the 1-shot setting the accuracy of MAML and ProtoMAML decreases as we add more inner loop steps, which may be due to overfitting. While both versions of ProtoContext do suffer from this to a degree, the losses in

TABLE II

ACCURACY FOR 1- AND 5-SHOT EXPERIMENTS. OMNIGLOT IS 20-WAY AND THE REST ARE 5-WAY. MEDIAN AND STANDARD DEVIATION FROM THREE TRIALS ON THE FULL TEST SET (FIVE TRIALS FOR EXPERIMENTS WITH LARGE DEVIATION). BELOW THE PENULTIMATE LINES FOR EACH EXPERIMENT ARE OUR MODELS: PROTOCONTEXT WITH CONTEXTUALIZED PROTOTYPES AND HEAD. GREEN INDICATES IMPROVEMENTS FROM PROTOMAML++, OUR MOST DIRECT BASELINE.

Accuracy for 1-shot (%)					
Experiment type	Model	Omniglot	Mini-ImageNet	Tiered-ImageNet	Airplanes
Our implementation	Prototypical Networks	80.9 ± 4.00	49.9 ± 0.25	50.3 ± 0.38	42.8 ± 2.76
Our implementation	MAML++	75.3 ± 2.19	48.5 ± 2.95	49.2 ± 0.66	39.8 ± 8.50
Our implementation	ProtoMAML++	70.2 ± 5.08	47.8 ± 0.25	49.7 ± 0.61	53.1 ± 0.96
Our ablation	PCA baseline	69.4 ± 4.27	48.5 ± 0.30	50.0 ± 0.38	39.5 ± 1.54
Our contrib. (abl.)	Context Only (*Contex. Proto.*)	94.9 ± 0.65	45.4 ± 2.19	47.4 ± 5.05	41.3 ± 2.56
Our contrib. (abl.)	Context Only (*Head*)	94.9 ± 0.12	49.1 ± 0.77	50.3 ± 0.36	52.6 ± 0.15
Our contrib. (main)	ProtoContext (*Contex. Proto.*)	96.0 ± 0.40 (+25.8)	**50.1** ± 1.13 (+2.3)	52.1 ± 1.07 (+2.4)	**58.3** ± 0.14 (+5.2)
Our contrib. (main)	ProtoContext (*Head*)	**96.7** ± 0.22 (+26.5)	**50.1** ± 0.21 (+2.3)	**54.3** ± 0.32 (+4.6)	56.6 ± 0.19 (+3.5)
Accuracy for 5-shot (%)					
Experiment type	Model	Omniglot	Mini-ImageNet	Tiered-ImageNet	Airplanes
Our implementation	Prototypical Networks	91.5 ± 2.89	**66.4** ± 0.40	70.1 ± 0.40	63.0 ± 0.86
Our implementation	MAML++	84.6 ± 2.40	65.6 ± 0.39	57.3 ± 6.87	58.2 ± 9.73
Our implementation	ProtoMAML++	77.5 ± 4.78	60.7 ± 0.99	65.0 ± 0.31	63.3 ± 0.57
Our ablation	PCA baseline	79.1 ± 4.06	61.5 ± 0.13	65.8 ± 0.25	62.2 ± 2.06
Our contrib. (abl.)	Context Only (*Context. Proto.*)	98.1 ± 0.22	60.0 ± 0.30	59.8 ± 0.21	62.7 ± 4.39
Our contrib. (abl.)	Context Only *(Head)*	98.3 ± 0.08	63.0 ± 0.12	66.8 ± 0.28	63.0 ± 3.07
Our contrib. (main)	ProtoContext (*Contex. Proto.*)	**98.4** ± 0.15 (+20.9)	63.9 ± 0.17 (+3.2)	70.0 ± 0.19 (+5.0)	**70.1** ± 0.61 (+6.8)
Our contrib. (main)	ProtoContext (*Head*)	**98.4** ± 0.04 (+20.9)	64.5 ± 0.39 (+3.8)	**71.4** ± 0.32 (+6.4)	69.3 ± 0.32 (+6.0)

accuracy are not nearly as dramatic. This suggests that the use of contextualizers may provide a method of augmenting prediction that is not as prone to overfitting as fine-tuning parameters through gradient descent, or that the ProtoContext head initializations are more general than those produced by ProtoMAML and MAML.

VIII. EMPIRICAL ANALYSIS

In this section our goal is to understand the role of contextualizers in conjuction with (*i*) the feature extactor and (*ii*) the self-attention mechanism. The former we study through our intra-class metric, while the latter we observe via attention heatmaps. We describe both below.

A. Intra-class Metric

Table III shows the results of measuring the intra-class metric of the features extracted by our ProtoContext model as well as ProtoMAML. Across every dataset, we see that ProtoContext consistently produces features with a lower intra-class metric (better), indicating that it has learned a feature extractor that is discriminative. These numbers suggest that the use of our contextualization mechanism may have some upstream benefit to the feature extractor. We can understand this by realizing that, because our contextualizers are initialized with versions of the prototypes, properly making use of the contextualization mechanism requires prototypes that well represent the samples given.

B. Use of contextualizers in self-attention

Figure 4 shows heatmaps of the attention weights in the self-attention mechanism of our ProtoContext model. The diagonal

TABLE III

INTRA-CLASS METRIC IS NOTICEABLY SMALLER (BETTER) FOR PROTOCONTEXT THAN PROTOMAML. EXPERIMENTS ARE 5-SHOT. PC IS SHORT FOR PROTOCONTEXT, *CP* IS SHORT FOR *Contextualized Prototypes* AND *H* IS SHORT FOR *Head*. ALL MEASURES IN RADIANS. MEAN AND STANDARD DEVIATION FROM THREE TRIALS ON THE FULL TEST SET.

Dataset	PC (*CP*)	PC (*H*)	ProtoMAML
Omniglot	0.37 ± 0.006	0.28 ± 0.001	0.84 ± 0.011
Mini-ImageNet	0.99 ± 0.001	0.82 ± 0.003	1.24 ± 0.006
Tiered-ImageNet	0.98 ± 0.007	0.83 ± 0.002	1.22 ± 0.003
Airplanes	0.93 ± 0.128	0.70 ± 0.005	1.15 ± 0.019

patterns of attention indicate high weight is given to the contextualizers that correspond to the class of a given sample. The context is also highly updated by the samples in its class, although this is more prominent in the 1-shot than the 5-shot setting. We present additional heatmaps in the Supplemental Materials that show a range of patterns can emerge, and show how applying regularization on the heatmaps can lead to improvements. We also find that the query almost never attends the query during self-attention. This is especially important as this is a fully learned behavior, not one we code into the model. We believe this demonstrates that the self-attention mechanism is learning what we expect: that context is a much more useful tool than other samples for making classification decisions.

C. Lessons learned from HPC experimentation

In our work we considered a large variety of datasets and baselines. Running and keeping track of a full cross product of all variants required careful SLURM scripting on

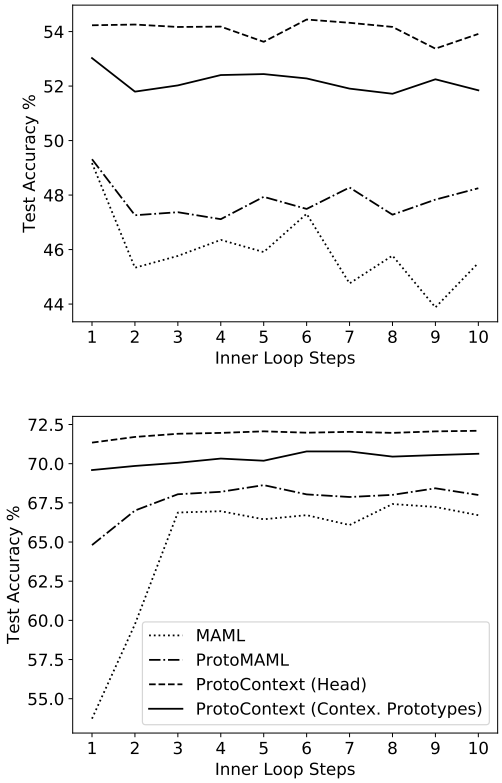

Fig. 3. Test Accuracy versus the number of inner loop steps for 1-shot (left) and 5-shot (right) on Tiered-ImageNet. ProtoContext shows strong results with just one adaptation step. In addition, ProtoContext seems to suffer from significantly less overfitting and shower lower variance across inner loop steps than ProtoMAML and MAML.

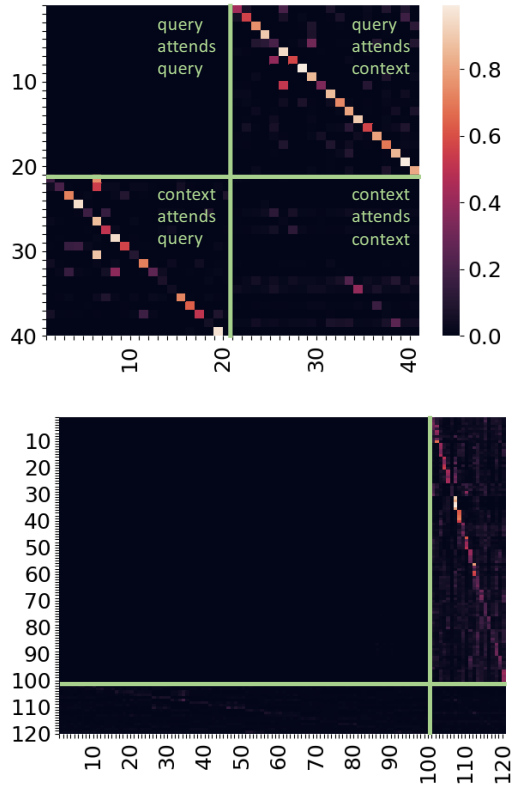

Fig. 4. Heatmaps of the attention weights for Omniglot provide evidence to support our conjecture in Equation (5): significant weight is given to the contextualizer corresponding to each example's class when contextualizing the features. This is seen in the strong diagonals in the heatmaps, which are a result of samples being presented in the order of class number.

MIT SuperCloud, which we have highighted in our publicly available code on GitHub.

We also benchmark the latency of our experiments on MIT SuperCloud. We use a single Nvidia V100 32G GPU and 20 workers for our experiments. We work with the harderOmniglot dataset with 5 inner loop steps and 20-way 5-shot classification setting. Every other hyperparameter follows the training setting from Figure 3. We run 100 steps of minibatch training and report mean and standard deviations of *iterations per second*.

In Figure 5 we present the results of our experiment and observe that our methods are a factor of 0.7 slower than the baseline. We believe that this slowdown is justified given the significant improvements we obtain over MAML and ProtoMAML across our experiments in this paper.

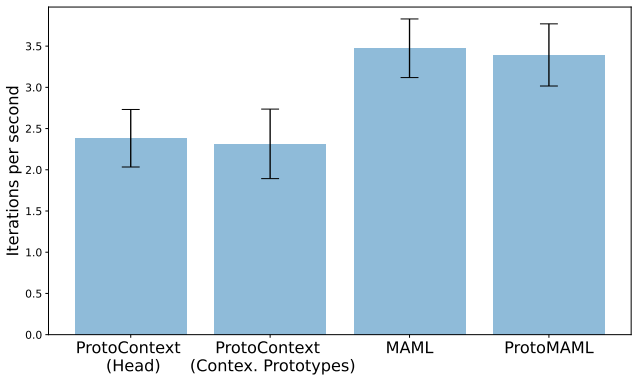

Fig. 5. Benchmarking the number of iterations per second for our ProtoContext-based and the MAML-based methods. Error bars denote the standard deviation of iterations per second.

IX. Conclusion and Future Work

We have presented a novel framework, Contextualization, for approaching few shot classification which allows for task-specific initialization and feature modification. We tested two forms of contextualization and showed that both outperform strong baselines. In addition, we presented other benefits of contextualization including resilience to overfitting, potential upstream benefits for feature extractors, and use of context to boost classification decisions. We believe that with proper contextualizers, contextualization can be extended to other problems such as regression, reinforcement learning and self-supervised learning, which we hope to explore in future work.

ACKNOWLEDGEMENTS

The authors acknowledge the MIT SuperCloud and Lincoln Laboratory Supercomputing Center [49] for providing HPC and consultation resources that have contributed to the research results reported within this paper/report.

We would like to thank Peter Lu, Charlotte Loh, Ileana Rugina, Kristian Georgiev, Brian Cheung, Alireza Fallah, Séb Arnold, and Chelsea Finn for fruitful conversations.

Research was sponsored in part by the United States Air Force Research Laboratory and was accomplished under Cooperative Agreement Number FA8750-19-2-1000. This material is also based upon work supported in part by the U. S. Army Research Office through the Institute for Soldier Nanotechnologies at MIT, under Collaborative Agreement Number W911NF-18-2-0048. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the United States Air Force or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

REFERENCES

[1] Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012.

[2] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning, 2022. *URL https://arxiv. org/abs/2204.14198*.

[3] Antreas Antoniou, Harrison Edwards, and Amos Storkey. How to train your MAML. In *ICLR*, New Orleans, LA, USA, 2019.

[4] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. In *NeurIPS 2016 Deep Learning Symposium*, 2016.

[5] Harkirat Singh Behl, Atilim Günes Baydin, and Philip H. S. Torr. Alpha MAML: adaptive model-agnostic meta-learning. In *ICML workshop*, 2019.

[6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[7] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

[8] Soravit Changpinyo, Wei-Lun Chao, and Fei Sha. Predicting visual exemplars of unseen classes for zero-shot learning. In *ICCV*, 2017.

[9] Yu Cheng, Mo Yu, Xiaoxiao Guo, and Bowen Zhou. Few-shot learning with meta metric learners. In *NeurIPS workshop*, 2017.

[10] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

[11] Yann N Dauphin and Samuel Schoenholz. MetaInit: Initializing learning by learning to initialize. In *NeurIPS*, 2019.

[12] Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Peter Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, et al. Scaling vision transformers to 22 billion parameters. In *International Conference on Machine Learning*, pages 7480–7512. PMLR, 2023.

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *arXiv:2010.11929*, 2021.

[14] Nikita Dvornik, Cordelia Schmid, and Julien Mairal. Selecting relevant features from a multi-domain representation for few-shot classification. In *arXiv:2003.09338*, 2020.

[15] Chelsea Finn and Sergey Levine. Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm. In *ICLR*, 2018.

[16] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

[17] Sebastian Flennerhag, Andrei A. Rusu, Razvan Pascanu, Francesco Visin, Hujun Yin, and Raia Hadsell. Meta-learning with warped gradient descent. In *International Conference on Learning Representations*, 2020.

[18] Kevin Frans, Jonathan Ho, Xi Chen, Pieter Abbeel, and John Schulman. Meta learning shared hierarchies. In *ICLR*, 2018.

[19] Weifeng Ge, Weilin Huang, Dengke Dong, and Matthew R. Scott. Deep metric learning with hierarchical triplet loss. *Lecture Notes in Computer Science*, page 272–288, 2018.

[20] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *CVPR*, 2018.

[21] Spyros Gidaris, Andrei Bursuc, Gilles Puy, Nikos Komodakis, Matthieu Cord, and Patrick Pérez. Online bag-of-visual-words generation for unsupervised representation learning, 2020.

[22] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010.

[23] Jonathan Gordon, John Bronskill, Matthias Bauer, Sebastian Nowozin, and Richard Turner. Meta-learning probabilistic inference for prediction. In *ICLR*, 2019.

[24] Yanming Guo, Yu Liu, Ard Oerlemans, Songyang Lao, Song Wu, and Michael S Lew. Deep learning for visual understanding: A review. *Neurocomputing*, 187:27–48,

2016.

[25] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):5149–5169, 2021.

[26] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, pages 448–456, 2015.

[27] Muhammad Abdullah Jamal and Guo-Jun Qi. Task agnostic meta-learning for few-shot learning. In *CVPR*, 2019.

[28] Bingyi Kang and Jiashi Feng. Transferable meta learning across domains. In *UAI*, 2018.

[29] Leonid Karlinsky, Joseph Shtok, Sivan Harary, Eli Schwartz, Amit Aides, Rogerio Feris, Raja Giryes, and Alex M. Bronstein. RepMet: Representative-based metric learning for classification and few-shot object detection. In *CVPR*, 2019.

[30] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[31] Gregory Koch. Siamese neural networks for one-shot image recognition. In *ICML workshop*, 2015.

[32] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[33] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *CVPR*, 2019.

[34] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *arXiv:1904.03758*, 2019.

[35] Wenbin Li, Lei Wang, Jinglin Xu, Jing Huo, Yang Gao, and Jiebo Luo. Revisiting local descriptor based image-to-class measure for few-shot learning. In *CVPR*, 2019.

[36] Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-sgd: Learning to learn quickly for few shot learning. *arXiv preprint arXiv:1707.09835*, 2017.

[37] Yann Lifchitz, Yannis Avrithis, Sylvaine Picard, and Andrei Bursuc. Dense classification and implanting for few-shot learning. In *CVPR*, 2019.

[38] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

[39] Yair Movshovitz-Attias, Alexander Toshev, Thomas K Leung, Sergey Ioffe, and Saurabh Singh. No fuss distance metric learning using proxies. In *ICCV*, 2017.

[40] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

[41] Boris Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. TADAM: task dependent adaptive metric for improved few-shot learning. In *NeurIPS*, 2018.

[42] Min Peng, Zhan Wu, Zhihao Zhang, and Tong Chen. From macro to micro expression recognition: Deep learning on small datasets using transfer learning. In *SCCG*, 2018.

[43] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. Film: Visual reasoning with a general conditioning layer. In *AAAI*, 2018.

[44] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8): 9, 2019.

[45] Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. Rapid learning or feature reuse? towards understanding the effectiveness of maml. In *ICLR*, 2020.

[46] S. Rahman, S. Khan, and F. Porikli. A unified approach for conventional zero-shot, generalized zero-shot, and few-shot learning. *IEEE Transactions on Image Processing*, 27(11):5652–5667, 2018.

[47] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2017.

[48] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. Meta-learning for semi-supervised few-shot classification. In *ICLR*, 2018.

[49] Albert Reuther, Jeremy Kepner, Chansup Byun, Siddharth Samsi, William Arcand, David Bestor, Bill Bergeron, Vijay Gadepally, Michael Houle, Matthew Hubbell, et al. Interactive supercomputing on 40,000 cores for machine learning and data analysis. In *HPEC*, 2018.

[50] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, , and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *ICLR*, 2019.

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

[52] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *NeurIPS*, 2017.

[53] Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning. In *NIPS*, 2017.

[54] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *NeurIPS*, 2016.

[55] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. 2016.

[56] Xingyou Song, Wenbo Gao, Yuxiang Yang, Krzysztof Choromanski, Aldo Pacchiano, and Yunhao Tang. ES-MAML: Simple hessian-free meta learning. In *ICLR*, 2019.

[57] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *arXiv:1707.02968*, 2017.

[58] Qianru Sun, Yaoyao Liu, Tat-Seng Chua, and Bernt Schiele. Meta-transfer learning for few-shot learning. In *CVPR*, 2019.

[59] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip H.S. Torr, and Timothy M. Hospedales. Learning

to compare: Relation network for few-shot learning. In *CVPR*, 2018.

[60] Yonglong Tian, Yue Wang, Dilip Krishnan, Joshua B. Tenenbaum, and Phillip Isola. Rethinking few-shot image classification: A good embedding is all you need? In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 266–282, Cham, 2020. Springer International Publishing. ISBN 978-3-030-58568-6.

[61] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *arXiv:2012.12877*, 2021.

[62] Eleni Triantafillou, Richard Zemel, and Raquel Urtasun. Few-shot learning through an information retrieval lens. In *NeurIPS*, 2017.

[63] Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, and Hugo Larochelle. Meta-dataset: A dataset of datasets for learning to learn from few examples. In *ICLR*, 2020.

[64] Joaquin Vanschoren. Meta-learning: A survey. *arXiv preprint arXiv:1810.03548*, 2018.

[65] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[66] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *NeurIPS*, 2016.

[67] Evan Vogelbaum, Rumen Dangovski, Li Jing, and Marin Soljačić. Contextualizing enhances gradient based meta learning. *arXiv preprint arXiv:2007.10143*, 2020.

[68] Jian Wang, Feng Zhou, Shilei Wen, Xiao Liu, and Yuanqing Lin. Deep metric learning with angular loss. *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. doi: 10.1109/iccv.2017.283.

[69] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

[70] Olga Wichrowska, Niru Maheswaranathan, Matthew W. Hoffman, Sergio Gómez Colmenarejo, Misha Denil, Nando de Freitas, and Jascha Sohl-Dickstein. Learned optimizers that scale and generalize. In *ICLR*, 2017.

[71] Zhongwen Xu, Hado P van Hasselt, and David Silver. Meta-gradient reinforcement learning. In *NeurIPS*, 2018.

[72] Han-Jia Ye, Hexiang Hu, De-Chuan Zhan, and Fei Sha. Few-shot learning via embedding adaptation with set-to-set functions. In *CVPR*, 2020.

[73] Mingzhang Yin, George Tucker, Mingyuan Zhou, Sergey Levine, and Chelsea Finn. Meta-learning without memorization. In *ICLR*, 2020.

[74] Jaesik Yoon, Taesup Kim, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian model-agnostic meta-learning. In *NeurIPS*. 2018.

[75] Luisa Zintgraf, Kyriacos Shiarli, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. Fast context adaptation via meta-learning. In *ICML*, 2019.

Algorithm 1 ProtoContext: Head Contextualization's Training Step for one Inner Loop Step. PyTorch-like Pseudocode.

```
# theta: feature extractor f's parameters
# phi: self-attention s's parameters
# w, b: head h's weights & biases parameters
# tasks: few shot tasks, N: ways
# D: features dimension
# K: support shots, Kq: query shots
# alpha: inner lr, beta: outer lr
# loss: cross-entropy

mgrads = [] # collect meta gradients
for t in tasks: # loops over tasks
    # extract support & query
    s, q = t.support, t.query

    # inner loop
    fs = f(theta)(s.inputs) # features: NxKxD
    p = mean(fs, dim=1) # prototypes: NxD
    fs = fs.view(N * K, D) # NKxD
    # context: NxD
    c = s(phi)(cat([p, fs], dim=0))[:N,:]
    # weights: NxD, biases: N
    w, b = proto_maml_init(c)
    # contextualized support: NKxD
    cs = s(phi)(cat([fs, w], dim=0))[:NK,:]
    ls = loss(h(w,b)(cs), s.labels)
    params_q = [theta, phi, w, b]
    params_q -= alpha * ls.grad(params)
    theta_q, phi_q, w_q, b_q = params_q

    # outer loop
    Nq = N * Kq
    fq = f(theta_q)(q.inputs).view(Nq,D)
    s_input = cat([fq, w_q], dim=0)
    cq = s(phi_q)(s_input)[:Nq,:]
    lq = loss(h(w_q,b_q)(cq), q.labels)
    mgrads.append(lq.grad(params_q))

[theta, phi, w, b] -= beta * mean(mgrads)
```

`cat`: concatenation, `proto_maml_init`: Algorithm 3, `mean`: mean of tensor (and list), `append`: append to list

APPENDIX

A. Algorithms

In this section, we show through analysis of our gradient the impact the addition of a self attention mechanism can have on our parameter updates. Our analysis is focused on the 1-shot setting for the sake of simplicity of notation. All of the calculations extend to the 5-shot case naturally.

Algorithm 2 ProtoContext: Prototype Contextualization's Training Step for an Inner Loop Step. Py-Torch-like.

```
# replace red code in Algorithm 1
# follow the rest of Algorithm 1 exactly

# first red line in Algorithm 1 becomes
cs = s(phi)(cat([fs, c], dim=0))[:NK,:]
c_new = s(phi)(cat([fs, c], dim=0))[NK:,:]

# second red line in Algorithm 1 becomes
s_input = cat([fq, c_new], dim=0)
```

Algorithm 3 ProtoMAML Initialization. PyTorch-like.

```
# c: contextualized weights (NxD)

w = 2 * c # weights: NxD
b = -norm(c, dim=1, p=2) ** 2 # biases: N
return w, b
```

`norm`: L_p norm, p is given by `p`

B. Notation for the 1-shot setting

Note that the inner loop updates depend only on the support set $\mathbf{S}$, and since in the 1-shot setting we have a single example from each class a, we can write the support set as follows $\mathbf{S} = (\mathbf{X}, \mathbf{y}) \equiv \left\{\left(\mathbf{x}^{(a)}, a\right)\right\}_{a=1}^{n}$, where $a = y^{(a)}$ without loss of generality, i.e. the target for the input $\mathbf{x}^{(a)}$ is the index of its class, which is a. Likewise, we denote the context as $\mathbf{C} \equiv \left\{\mathbf{c}^{(a)}\right\}_{a=1}^{n}$. Finally, let $\mathcal{C}^{(a)}$ be the contextualization of input example $\mathbf{x}^{(a)}$, i.e. $\mathcal{C}^{(a)} = s_{\boldsymbol{\phi}}\left(f_{\boldsymbol{\theta}}\left(\mathbf{x}^{(a)}\right), \mathbf{C}\right)$ for the contextualization algorithm and $\mathcal{C}^{(a)} = f_{\boldsymbol{\theta}}\left(\mathbf{x}^{(a)}\right)$ for any other gradient based algorithm (MAML, ProtoMAML).

C. General form of the loss and the gradient updates

Our approach is to impose the structure of the classification head in order to obtain an explicit form of the loss function, which will consequentially yield the gradient updates for the parameters of our model. We proceed with our analysis below.

Let the weights of the head be $\left\{\boldsymbol{w}^{(a'')}\right\}_{a''=1}^{n}$ and their corresponding biases be $\left\{b^{(a'')}\right\}_{a''=1}^{n}$. Then, we have that for a single sample

$$p_{\boldsymbol{w},b}\left(s_{\boldsymbol{\phi}}\left(f_{\boldsymbol{\theta}}\left(\mathbf{x}^{(a)}\right)\right)\right) = \operatorname{softmax}\left(\begin{bmatrix}\boldsymbol{w}^{(1)} \cdot \mathcal{C}^{(a)} + b^{(1)} \\ \vdots \\ \boldsymbol{w}^{(n)} \cdot \mathcal{C}^{(a)} + b^{(n)}\end{bmatrix}\right).$$

Note that in our setting the cross entropy loss takes the form

$$-\sum_{a''=1}^{n} \log\left[p_{\boldsymbol{w},b}\left(s_{\boldsymbol{\phi}}\left(f_{\boldsymbol{\theta}}\left(\mathbf{x}^{(a'')}\right)\right)\right)\right]_{a''}, \tag{6}$$

where $[\mathbf{z}]_{a''}$ means that we take the a''-th component of the vector $\mathbf{z}$, and $p_{\boldsymbol{w},b}, s_{\boldsymbol{\phi}}$, and $f_{\boldsymbol{\theta}}$ are the predictor (head), contextualization mechanism, and feature extractor respectively as defined above. Now, using Equation (C) in Equation (6), then using the form of the softmax and simplifying the expression we obtain the following loss function L viewed as a function of the support set $\mathbf{S}$ as follows

$$\begin{aligned} L(\mathbf{S}) = &-\sum_{a''=1}^{n} \boldsymbol{w}^{(a'')} \cdot \mathcal{C}^{(a'')} + b^{(a'')} \\ &+ \sum_{a''=1}^{n} \log\left(\sum_{\widetilde{a}=1}^{n} \exp\left(\boldsymbol{w}^{(\widetilde{a})} \cdot \mathcal{C}^{(a'')} + b^{(\widetilde{a})}\right)\right). \end{aligned} \tag{7}$$

Hence, the form of this loss in Equation (7) is amenable to analysis for the predictor and feature extractor of our model.

D. Gradient Updates for the Predictor

Differentiating Equation (7) with respect to the head weights $\boldsymbol{w}^{(a)}$, corresponding to class a, we obtain the following

$$\nabla_{\boldsymbol{w}^{(a)}} L(\mathbf{S}) = -\mathcal{C}^{(a)} + \mathcal{D}^{(a)}, \tag{8}$$

where we introduced the following notation

$$\begin{aligned} \mathcal{D}^{(a)} &= \sum_{a''=1}^{n} \frac{\exp\left(\boldsymbol{w}^{(a)} \cdot \mathcal{C}^{(a'')} + b^{(a)}\right)}{\sum_{\widetilde{a}=1}^{n} \exp\left(\boldsymbol{w}^{(\widetilde{a})} \cdot \mathcal{C}^{(a'')} + b^{(\widetilde{a})}\right)} \mathcal{C}^{(a'')} \\ &\equiv \sum_{a''=1}^{n} \frac{\exp\left(\boldsymbol{w}^{(a)} \cdot \mathcal{C}^{(a'')} + b^{(a)}\right)}{\mathcal{Z}} \mathcal{C}^{(a'')}, \end{aligned}$$

where $\mathcal{Z}$ is the partition function. Equation (8) is significant since it tells us that the contextualization algorithm can control the gradient updates through the self-attention mechanism, because the contextualization $\mathcal{C}^{(a)}$ depends on the parameters $\boldsymbol{\phi}$ of the self-attention mechanism. Under some assumptions, this behavior might yield simplifications, amenable to analysis. In that spirit, we proceed with the following

Proposition 1. *Assume our contextualizations are orthogonal. In the 1-shot setting, the inner loop updates for the weights in the head for class a move in a direction of **positive** correlation with the contextualization of its support example $\mathbf{x}^{(a)}$.*

Proof. Note that $0 < \exp\left(\boldsymbol{w}^{(a)} \cdot \mathcal{C}^{(a)} + b^{(a)}\right) / \mathcal{Z} < 1$. Combined with our assumption of orthogonality of contextualizations, we then get that the correlation

$$C := \mathcal{C}^{(a)} \cdot (-\nabla_{\boldsymbol{w}^{(a)}} L(\mathbf{S}))$$

with the gradient update (ignoring the learning rate) is

$$\begin{aligned} C &= \mathcal{C}^{(a)} \cdot \left(\mathcal{C}^{(a)} - \mathcal{D}^{(a)}\right) \\ &= \mathcal{C}^{(a)} \cdot \mathcal{C}^{(a)} - \mathcal{C}^{(a)} \cdot \mathcal{D}^{(a)} \\ &= \left\|\mathcal{C}^{(a)}\right\|_2^2 - \mathcal{C}^{(a)} \cdot \mathcal{D}^{(a)} \\ &= \left\|\mathcal{C}^{(a)}\right\|_2^2 - \frac{\exp\left(\boldsymbol{w}^{(a)} \cdot \mathcal{C}^{(a)} + b^{(a)}\right)}{\sum_{\widetilde{a}=1}^{n} \exp\left(\boldsymbol{w}^{(\widetilde{a})} \cdot \mathcal{C}^{(a)} + b^{(\widetilde{a})}\right)} \left\|\mathcal{C}^{(a)}\right\|_2^2 \\ &= \left(1 - \frac{\exp\left(\boldsymbol{w}^{(a)} \cdot \mathcal{C}^{(a)} + b^{(a)}\right)}{\sum_{\widetilde{a}=1}^{n} \exp\left(\boldsymbol{w}^{(\widetilde{a})} \cdot \mathcal{C}^{(a)} + b^{(\widetilde{a})}\right)}\right) \left\|\mathcal{C}^{(a)}\right\|_2^2 \\ &> 0, \end{aligned}$$

where going from the third to the fourth line we use the fact that $\mathcal{C}^{(a)} \cdot \mathcal{C}^{(a')} = 0$ if and only if a is different from a'. From here, since the learning rate scales each line above by $\alpha > 0$, the proof follows, as desired. □

E. Gradient Updates for the Feature Extractor

For this analysis we would like to underline the dependence of the contextualizations $\mathcal{C}^{(a'')}$ on the parameters $\boldsymbol{\theta}$ of our feature extractor, by explicitly writing the dependence as follows $\mathcal{C}_{\boldsymbol{\theta}}^{(a'')}$. Hence, differentiating Equation (7) with respect to $\boldsymbol{\theta}$ and using notation from the previous section we obtain the following expression for the gradient update

$$\nabla_{\boldsymbol{\theta}} L(\mathbf{S}) = -\sum_{a''=1}^{n} \nabla_{\boldsymbol{\theta}} \mathcal{C}_{\boldsymbol{\theta}}^{(a'')} \boldsymbol{w}^{(a'')} + \sum_{a''=1}^{n}\sum_{a'=1}^{n} \frac{\exp\left(\boldsymbol{w}^{(a')} \cdot \mathcal{C}_{\boldsymbol{\theta}}^{(a'')} + b^{(a)}\right)}{\mathcal{Z}} \nabla_{\boldsymbol{\theta}} \mathcal{C}_{\boldsymbol{\theta}}^{(a'')} \boldsymbol{w}^{(a')}.$$

We proceed with the following

Proposition 2. *With the contextualization algorithm, the gradient updates for the feature extractor share gradient information from each example in the support set.*

Proof. It suffices to analyse $\nabla_{\boldsymbol{\theta}} \mathcal{C}_{\boldsymbol{\theta}}^{(a'')}$ in the above equation. Our self-attention mechanism consists of a scaled dot product attention which yields a linear combination across the transformed (by value weights $\mathbf{W}_{\text{value}}$) inputs to the mechanism, followed by a layer normalization with mean μ and standard deivation σ, and then a linear layer with weights $\mathbf{W}$ and bias $\mathbf{b}$. Therefore, since the initalization of our context consists of self-attention over the extracted features of the support set, without loss of generality we have that

$$\mathcal{C}_{\boldsymbol{\theta}}^{(a'')} = \mathbf{W}\left(\sum_{\gamma=1}^{n} v_{\gamma}^{(a'')} \frac{\mathbf{W}_{\text{value}} f_{\boldsymbol{\theta}}\left(\mathbf{x}^{(\gamma)}\right) - \mu \mathbf{1}}{\sigma}\right) + \mathbf{b},$$

where the coefficients $v_{\gamma}^{(a'')} \equiv v_{\gamma}^{(a'')}(\boldsymbol{\theta}, \boldsymbol{\phi})$ depend both on the feature extractor and the key and query matrices. Now, after taking the gradient with respect to $\boldsymbol{\theta}$ we obtain the following expression

$$\nabla_{\boldsymbol{\theta}} \mathcal{C}_{\boldsymbol{\theta}}^{(a'')} = \mathbf{W}\left(\sum_{\gamma=1}^{n} \nabla_{\boldsymbol{\theta}} v_{\gamma}^{(a'')} \frac{\mathbf{W}_{\text{value}} f_{\boldsymbol{\theta}}\left(\mathbf{x}^{(\gamma)}\right) - \mu \mathbf{1}}{\sigma} + \boxed{\sum_{\gamma=1}^{n} \frac{v_{\gamma}^{(a'')}}{\sigma} \mathbf{W}_{\text{value}} \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}\left(\mathbf{x}^{(\gamma)}\right)}\right),$$

where we have boxed the contribution from the gradient information coming from *all* support inputs. Thus, the statement follows. □

This proposition is significant since it can explain why the feature extractor yields better intra-class similarity, as we presented in the main text. We conjecture that this is true since gradient information flows from all support exmples in a controlled manner, manifested by the coefficients $v_{\gamma}^{(a'')}/\sigma$, which are learned by the self-attention mechanism.

Contrast this with $\mathcal{C}^{(a'')} = f_{\boldsymbol{\theta}}\left(\mathbf{x}^{(a'')}\right)$ for MAML and ProtoMAML in the absence of batchnorm [26], which yields gradient information only for the support example a''. In practice, ProtoMAML and MAML use batchnorm, which may be seen as an alternative way to contextualize features, but also as an inferior way as it is demonstrated in our experiments.

F. Results in the Broader Context of Meta Learning

We should note that in this section we have described how the self-attention controls the gradient updates during fine-tuning. This emphasizes the role of the self-attention as a meta learner in a similar fashion to how LSTMs can be used as meta learners [47].