# REVISITING LINEAR DECISION BOUNDARIES FOR FEW-SHOT LEARNING WITH TRANSFORMER HYPERNETWORKS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Few-shot learning (FSL) methods aim to generalize a model to new unseen classes using only a small number of support examples. In image classification settings, many FSL approaches utilize a similar architecture to standard supervised learning, learning a model composed of a feature extractor followed by a linear classifier head. A common choice for the classifier is ProtoNet-style nearest neighbor, but this may be suboptimal as it is context-independent. As an alternative, some methods train a parametric classifier (e.g. logistic regression, support vector machine) using embeddings from novel classes. However, task-specific training requires time and resources, and poses optimization challenges such as overfitting on only a few samples. Instead, we propose to generate linear classifiers for new classes using a transformer-based hypernetwork, performing context aggregation in permutation invariant manner. A transformer hypernetwork allows us to instantiate a new task-specific classifier without any additional training on novel tasks. Experiments conducted on 1-shot 5-way and 5-shot 5-way MiniImageNet, TieredImageNet, and CIFAR-FS demonstrate that transformer hypernetworks are capable of generating classifiers that achieve up to 1.4% higher accuracy than other commonly used linear classifiers. Among the group of methods that offer optimization-free meta-inference, we achieve new state-of-the-art in most cases.

## 1 INTRODUCTION

While advances in deep learning (LeCun et al., 2015) have led to significant improvements in a variety of problem domains, the typical supervised learning approaches require collecting large amounts of labeled data for training (Deng et al., 2009), often in the order of thousands of samples per class. In many applications, such large-scale labeled datasets are unavailable, and collecting them is impractical or prohibitively expensive. This is especially the case when fast adaptation to novel unseen tasks is necessary. To address this issue, few-shot learning (FSL) methods (Vinyals et al., 2016; Snell et al., 2017; Finn et al., 2017) have been proposed. FSL approaches seek to generalize to previously unseen tasks, with only a few samples available for adaptation.

For few-shot classification, many methods follow the same overall structure as conventional supervised learning. They use a convolutional feature extractor to compute embeddings, followed by a classifier to map an embedding to one of the output classes. Moreover, few-shot models are trained in a way that allows the feature extractor to adapt to new tasks using only a few examples. Several common choices for classifiers are prevalent among recent literature. Simple classifiers such as nearest neighbor, either with euclidean or cosine distance, can work surprisingly well while also having the added benefit of not requiring any additional training (Snell et al., 2017; Vinyals et al., 2016). However, nearest neighbor can often be outperformed by trained classifiers such as logistic regression (LR) or support vector machines (SVMs) (Caruana et al., 2008), and recent works have shown that this is also the case for embeddings in few-shot settings (Tian et al., 2020; Lee et al., 2019). On the other hand, LR and SVM classifiers require a separate and expensive training for each novel task, and the small number of samples available in few-shot settings may elevate the risk of overfitting. Subsequently, the learned classifier may be suboptimal.

Instead of training parametric classifiers or relying on nearest neighbors, we propose using transformer hypernetworks (Vaswani et al., 2017; Ha et al., 2017) to *generate* classifiers for novel unseen tasks. Such an approach does not require any task-specific training, while also having the flexibility of producing classifiers beyond simple nearest neighbors. We parameterize the hypernetwork using a transformer, as it has input permutation invariance and an attention mechanism capable of learning context, both desirable properties for few-shot learning. Our transformer hypernetwork takes in average support image embeddings from novel classes, also known as class prototypes, and predicts a context-dependent linear classifier. We meta-train our transformer hypernetwork in an episodic fashion with a cross-entropy loss. Additionally, we use outlier exposure (Hendrycks et al., 2018) and minimize cross entropy between the predicted class logits and soft labels with uniform probability for out-of-the-task images. Our main contributions are as follows:

- We develop a transformer hypernetwork that uses class-prototypes to predict parameters of a linear classifier for a pre-trained feature extractor. Notably, our approach infers a task-dependent model without requiring optimization on novel tasks.
- We validate the effectiveness of transformer hypernetworks for generating few-shot classifiers with experiments on MiniImageNet (Vinyals et al., 2016), TieredImageNet (Ren et al., 2018) and CIFAR-FS (Bertinetto et al., 2018) in 1-shot and 5-shot 5-ways settings.
- Remarkably, the linear classifiers generated by our transformer hypernetwork achieve up to 1.4% higher accuracy compared to the baseline classifiers operating on the same embedding space. Our complete model achieves new state-of-the-art performance in five out of six cases, outperforming other existing methods with optimization-free meta-inference.

## 2 Related works

### 2.1 Meta-learning

Many FSL methods use meta-learning, or "learning to learn" strategy, as proposed in (Vinyals et al., 2016). Specifically, meta-learning mimics few-shot inference setting during meta-training stage by sampling learning episodes from a larger training set. The meta-learning based methods are often further categorized into metric learning methods and optimization based methods.

Metric learning methods aim to use a better similarity metric to learn more transferable representations for FSL. Matching Networks (Vinyals et al., 2016) leveraged attention and a memory architecture to learn a mapping from a support set to a classification function using cosine similarity. Prototypical Networks (ProtoNets) (Snell et al., 2017) constructed a nearest neighbor classifier using euclidean distance between class-average features and test features. Relation Networks (Sung et al., 2018) proposed a transferable metric by comparing the relation between different images. Oreshkin et al. (2018) explored the effectiveness of task-dependent metric space. Li et al. (2020) and Xing et al. (2019) used semantic similarity to learn better representations that benefit FSL. While these approaches learn global representations, Li et al. (2019) proposed an image-to-class module to calculate similarities on local descriptors. A similar idea of using local descriptors, or spatially-aware features, is leveraged in Doersch et al. (2020).

Optimization-based methods aim to perform fine-tuning given a few support examples from a new task. Model-agnostic meta-learning (MAML) (Finn et al., 2017) aimed to find a set of model parameters which can be quickly adapted to novel tasks by using only a few optimization steps. Latent Embedding Optimization (LEO) method (Rusu et al., 2018) demonstrates the effectiveness of learning a low-dimensional latent generative representation space to perform gradient-based meta-learning. MetaOptNet (Lee et al., 2019) proposed to incorporate a differentiable quadratic programming solver to train the model end-to-end with a linear support vector machine as the classifier.

As we perform episodic training, our work falls into the broader category of meta-learning. However, we do not belong to either metric learning or optimization based methods. We do not rely on a similarity metric or fine-tune model parameters on a support set of novel tasks. Instead, we aim to bring the best of both worlds, offering optimization-free meta-inference similar to metric learning approaches and while also producing task-dependent models similar to optimization based methods.

### 2.2 Hypernetworks

Hypernetworks were first proposed in (Ha et al., 2017) for language modeling and image recognition. It is an approach to use one network known as a hypernetwork to generate the weights of another network. Since then, hypernetworks have been successfully applied to various tasks (Zhang et al., 2018; Nirkin et al., 2020; Spurek et al., 2020).

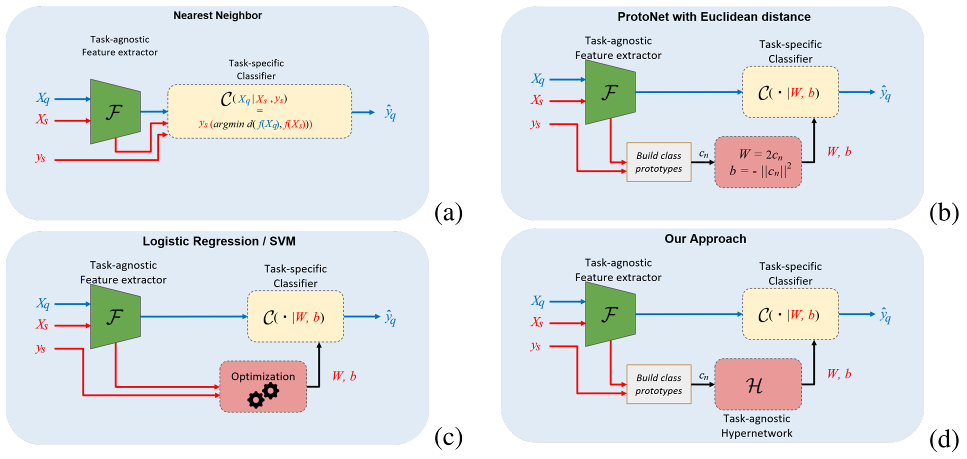
Nearest Neighbor
Task-agnostic Feature extractor
Task-specific Classifier
$X_q$
$X_s$
$y_s$
$\mathcal{F}$
$\mathcal{C}(X_q \mid X_s, y_s) = y_s(\operatorname{argmin} d(f(X_q), f(X_s)))$
$\hat{y}_q$
(a)
ProtoNet with Euclidean distance
Build class prototypes
$c_n$
$W = 2c_n$
$b = -\|c_n\|^2$
$W, b$
$\mathcal{C}(\cdot \mid W, b)$
(b)
Logistic Regression / SVM
Optimization
(c)
Our Approach
$\mathcal{H}$
Task-agnostic Hypernetwork
(d)

Figure 1: Meta-inference stage using various classifiers composed with a fixed task-agnostic feature extractor. We propose a task-agnostic hypernetwork (d) to generate a task-specific linear classifier. Note, LR and linear SVM (c) require training during meta-inference, whereas the remaining classifiers (a,b,d) are training-free during meta-inference.

Several works have leveraged hypernetworks for few-shot learning, though some of these works do not mention hypernetworks explicitly. Qiao et al. (Qiao et al., 2018) proposed using a feed-forward network to predict the classifier weights from feature activations. Gordon et al. (Gordon et al., 2018) used an amortized network to infer rows of a classifier weight matrix in a context-free manner. TADAM (Oreshkin et al., 2018) defined a dynamic feature extractor with scale and shift parameters predicted from task-dependent representations. TAFE-Net (Wang et al., 2019a) used a meta-learner, or hypernetwork, to generate the parameters of a network that compute features compatible with a fixed classifier. In contrast to TADAM and TAFE-Net, we keep a feature extractor fixed and adapt the final linear classifier to a given task. Furthermore, unlike (Qiao et al., 2018; Gordon et al., 2018), we infer context-aware decision boundaries for a multi-class classifier using transformers.

## 3 Method

The goal of a $K$-shot $N$-way classification problem is to leverage $KN$ labeled examples from $N$ novel classes to infer a classifier that can classify $QN$ unlabeled examples from the same classes. The labeled and unlabeled examples are commonly referred to as support and query examples, respectively. We denote the labeled set as $\mathcal{D}_s = \{X_s, y_s\}_{s=1}^{KN}$ and the unlabeled set as $\mathcal{D}_q = \{X_q\}_{q=1}^{QN}$, where $\{X, y\}$ is an image and its corresponding one-hot label. The goal is to infer a model $\mathcal{M}$ such that $\hat{y}_q = \mathcal{M}(X_q; \mathcal{D}_s)$.

Generally $K = 1$ or $5$ in few-shot classification tasks, which is a challengingly low number of samples to learn from. One strategy is to meta-train $\mathcal{M}$ to solve many $K$-shot $N$-way classification tasks drawn from a large labeled train set $\mathcal{D}_{\text{train}}$. The classes in $\mathcal{D}_{\text{train}}$, commonly referred to as the *base* classes, are typically different from the *novel* classes encountered during inference. Specifically, we draw labeled sets $\{\mathcal{D}_s^B, \mathcal{D}_q^B\} \sim \mathcal{D}_{\text{train}}$ and learn $\mathcal{M}^* = \operatorname{argmin}_{\mathcal{M}} \; \mathbb{E}_{\{\mathcal{D}_s^B, \mathcal{D}_q^B\}} L(\mathcal{M}(X_q^B; \mathcal{D}_s^B), y_q^B)$. At inference time, given $X_q$ and $\mathcal{D}_s$, we predict labels $\hat{y}_q = \mathcal{M}^*(X_q; \mathcal{D}_s)$.

### 3.1 Our Approach

We construct model $\mathcal{M}$ using a task-agnostic feature extractor $\mathcal{F}$ and a task-specific classifier $\mathcal{C}$. $\mathcal{F}$ computes features from support and query images, and $\mathcal{C}$ predicts class-logits. One can use either a non-parametric nearest neighbor classifier or a simple parametric linear classifier as $\mathcal{C}$ (see Figure 1(a-c)). The parameters of a linear classifier can either be optimized (e.g. with logistic regression, linear SVM), or inferred using a closed-form solution (e.g. ProtoNets (Snell et al., 2017)). As observed by Tian et al. (2020), logistic regression can learn better classifiers than nearest neighbor; however, it requires expensive task-specific optimization on novel task, often leading to overfitting due to the few training samples. On the other hand, ProtoNets offer an optimization-free solution, but one that does not account for context: the parameters specific to the decision boundary of one class are independent from the rest of the classes in the task, resulting in suboptimal performance.

We seek a context-dependent linear classifier that does not require task-specific optimization. To this end, we introduce a transformer hypernetwork $\mathcal{H}$ that uses averaged support features to produce the parameters of a classifier $\mathcal{C}$. Instead of relying on optimization, $\mathcal{H}$ predicts parameters of $\mathcal{C}$ using a single forward pass through the network, reducing overfitting and eliminating time needed for training during meta-inference of novel tasks. Similar to $\mathcal{F}$, $\mathcal{H}$ is task-agnostic and is shared by all few-shot tasks; in contrast, the generated classifier $\mathcal{C}$ is task-specific and is predicted from the support set of a given task. The complete model is shown in Fig. 1(d).

Let $f(X) \in \mathbb{R}^{d\times 1}$ be $\ell_2$-normalized features of an image $X$, $f(X) = \mathcal{F}(X)/||\mathcal{F}(X)||_2$. Let $c_n = \frac{1}{K}\sum_{s=1}^{KN} \mathbb{1}_{[y_s=n]} f(X_s)$ be the class-prototype of the $n^{th}$ class. We disentangle direction and norm of $c_n$ to build representation $p_n = \left[\frac{c_n}{||c_n||_2}; ||c_n||_2 - 1\right]$. This choice is motivated by the ProtoNet-style linear classifier, where $W$ and $b$ depend on the direction and the norm of individual class-prototypes. The hypernetwork processes representations $P = [p_1, ..., p_N]$ as follows.

$$\widehat{P} = \frac{1}{\sqrt{d}}\, W_2 \mathcal{T}(\sqrt{d} W_1 P) \tag{1}$$

Here, $W_1 \in \mathbb{R}^{t\times(d+1)}$ and $W_2 \in \mathbb{R}^{(d+1)\times t}$ are projection matrices and $\mathcal{T}$ is a transformer (Vaswani et al., 2017). We initialize $W_1$ as an orthogonal matrix and $W_2 = W_1^T$, ensuring that Eq. 1 results in pseudo-identity at initialization. This initialization helps the hypernetwork find a solution closer to the ProtoNet solution, yet more optimal. Note that $W_1$ and $W_2$ are not tied and are free to vary independently during training. Finally, we split $\widehat{P}$ to predict parameters of the linear classifier $\mathcal{C}$.

$$\widehat{W}, \widehat{b} = \widehat{P}_{(1:d,:)}, \widehat{P}_{(d+1,:)} \tag{2}$$

$$W, b = \frac{\widehat{W}}{||\widehat{W}||_2}, \text{sigmoid}(\widehat{b}) \tag{3}$$

We predict class-logits for query examples as $\hat{y}_q = \mathcal{C}(f(X_q)|W, b) = f(X_q)^T W - b$.

## 3.2 Training and Inference

We train feature extractor $\mathcal{F}$ and hypernetwork $\mathcal{H}$ sequentially. $\mathcal{F}$ is pre-trained to perform a conventional classification on the base classes, as in previous works (Ye et al., 2020; Sun et al., 2019; Oreshkin et al., 2018; Tian et al., 2020). $\mathcal{H}$ is meta-trained to predict $K$-shot $N$-way linear classifiers while keeping parameters of $\mathcal{F}$ frozen. Once trained, the hypernetwork can generate a task-specific linear classifier for any novel task without requiring additional training.

**Pre-training of $\mathcal{F}$.** Self-supervised learning (SSL) has proven effective for few-shot classification (Gidaris et al., 2019; Mangla et al., 2020; Rodríguez et al., 2020; Rajasegaran et al., 2020). Inspired by this observation, we train $\mathcal{F}$ using a combination of supervised and self-supervised criteria as proposed in (Rajasegaran et al., 2020). To this end, we augment image-label pair $\{X, y\} \in \mathcal{D}_{\text{train}}$ to produce $\{X^\theta, y^\theta, r^\theta\}$, where $X^\theta$ is an image $X$ rotated by $\theta$ degrees, $y^\theta = y$, $r^\theta$ is a one-hot rotation label and $\theta \in \{0, 90, 180, 270\}$. Then, we introduce a stack of two linear classifiers on top of $\mathcal{F}$. $\mathcal{F}$ extracts features from image $X^\theta$ and passes them to the first classifier. The first classifier predicts class logits $\hat{y}^\theta$ and passes them to the second classifier. The second classifier predicts rotation logits $\hat{r}^\theta$. We trained the entire model using $L_{CCE}(\text{softmax}(\hat{y}^\theta), y) + \alpha L_{BCE}(\text{sigmoid}(\hat{r}^\theta), r^\theta)$.

**Meta-training of $\mathcal{H}$.** We meta-train $\mathcal{H}$ in an episodic fashion while keeping the parameters of $\mathcal{F}$ frozen. We draw a batch of $M$ few-shot tasks with non-overlapping classes, $\{\mathcal{D}^B_{s(m)}, \mathcal{D}^B_{q(m)}\}_{m=1}^M \sim \mathcal{D}_{\text{train}}$. We extract embeddings of support and query examples using $\mathcal{F}$, and let $\mathcal{H}$ generate parameters of $M$ task-specific classifiers $\{\mathcal{C}_m\}_{m=1}^M$ as discussed in Section 3.1. The generated classifier $\mathcal{C}_m$ predicts class logits $\hat{y}^B_{q(m)}$ for query images $X^B_{q(m)}$. We use the cross-entropy loss as our training criterion. Additionally, we also use outlier exposure (Hendrycks et al., 2018), which yields a small but consistent improvement. To this end, we build a task-specific outlier set $\mathcal{D}^B_{o(m)} = \{X^B_{q(l\neq m)}, u | l \in M\}, \forall m \in M$ using the query images in the existing batch. Here $u$ is a soft label with uniform probability. The generated classifier $\mathcal{C}_m$ also predicts class logits $\hat{y}^B_{o(m)}$ for outlier images. Finally, we combine cross-entropy losses evaluated on query and outlier sets to form a training objective $L = \frac{1}{M}\sum_{m=1}^M L_{CCE}(\text{softmax}(\hat{y}^B_{q(m)}), y^B_{q(m)}) + \lambda L_{CCE}(\text{softmax}(\hat{y}^B_{o(m)}), u)$.

**Meta-inference.** We evaluate our model on novel few-shot tasks without further optimization. Let's consider a novel dataset $\{\{X_s, y_s\}_{s=1}^{KN}, \{X_q\}_{q=1}^{QN}\}$. The feature extractor $\mathcal{F}$ extracts features $f(X_s)$ and $f(X_q)$ from $X_s$ and $X_q$. The hypernetwork uses $f(X_s)$ to predict parameters $W$ and $b$ of a linear classifier $\mathcal{C}$. We use the generated classifier to predict class labels $\hat{y}_q = \mathrm{argmax}\ \mathcal{C}(f(X_q)|W, b)$.

## 4 EXPERIMENTS

We perform experiments on 1-shot 5-way and 5-shot 5-way classification tasks on three datasets. First, we provide details about the experiment setup. Then, we compare various linear classifiers for a fixed $\mathcal{F}$. Finally, we perform ablation on various design choices made in our hypernetwork and present a state-of-the-art comparison of a complete model.

### 4.1 EXPERIMENT SETUP

**Datasets.** We perform experiments on three popular few-shot classification datasets: MiniImageNet (Vinyals et al., 2016), TieredImageNet (Ren et al., 2018), and CIFAR-FS (Bertinetto et al., 2018). MiniImageNet and TieredImageNet contain color images of size 84×84 obtained from ImageNet (Russakovsky et al., 2015). MiniImageNet includes 60K images from 100 classes divided into train, validation, and test sets with 64, 16, and 20 classes respectively (Ravi & Larochelle, 2017). TieredImageNet contains ~0.78M images from 608 classes divided into train, validation, and test sets with 351, 97, and 160 classes respectively. CIFAR-FS is derived from the CIFAR-100 dataset (Krizhevsky et al., 2009), containing 60K 32 × 32 color images. The dataset comprises of 100 classes divided into 64, 16, and 20 classes for training, validation, and testing. We use the same number of ways during meta-training and meta-inference; experiments demonstrating interpolation and extrapolation to fewer and more classes during meta-inference are shown in Appendix A.

**Feature extractor $\mathcal{F}$.** Similar to past FSL works (Mishra et al., 2018; Ye et al., 2020; Tian et al., 2020; Oreshkin et al., 2018; Lee et al., 2019; Ravichandran et al., 2019), we implement $\mathcal{F}$ as a ResNet-12 (He et al., 2016), which consists of four residual blocks of three convolutional layers with 64, 160, 320, and 640 3 × 3 kernels. The first three blocks are followed by 2 × 2 max-pooling layers, and the last block is followed by a global average pooling layer, resulting in features of size $d = 640$. We use Dropblock (Ghiasi et al., 2018) regularization while training $\mathcal{F}$ and follow the optimization procedure of (Rajasegaran et al., 2020). We set batch-size to 64 and rotate each image three times, resulting in an effective batch-size of 256. The hyperparameter $\alpha$ in the training objective is set to 2. We train $\mathcal{F}$ for 65 epochs for MiniImageNet, 60 epochs for TieredImageNet, and 65 epochs for CIFAR-FS. We start with an initial learning rate of 0.05 and decay the learning rate by 0.1 at 60 epochs for MiniImageNet and CIFAR-FS, and 0.1 each at 30, 40 and 50 epochs for TieredImageNet. Pre-training on each dataset is done on a Tesla V100 GPU with 16GB memory in less than a day. Experiments with a ResNet-18 backbone can be found in Appendix B.

**Hypernetwork $\mathcal{H}$.** For projection, we use $t = 1024$ for 1-shot and $t = 2048$ for 5-shot tasks. Transformer $\mathcal{T}$ has a single encoder layer with two residual blocks (Vaswani et al., 2017). The first residual branch includes a self-attention layer with a single attention head, and the second residual branch includes an MLP with hidden dimensions equal to $4t$. Furthermore, we place LayerNorm (Ba et al., 2016) inside the residual branch (Xiong et al., 2020). Experiments with other architectural instantiations of the hypernetwork can be found in Appendix C. We meta-train $\mathcal{H}$ for 10K iterations, halving the learning rate every 2K iterations and monitoring validation accuracy for early stopping. We use a Stochastic Gradient Descent (SGD) optimizer with Nesterov momentum (Sutskever et al., 2013) of 0.9 and weight decay of 5e-4 for both training phases. In addition to four rotations, the input images are augmented using random crop, color jitter, and horizontal flip. We perform cross-validation to choose the initial learning rate, batch-size $M$, and hyperparameter $\lambda$. Meta-training is done on a V100 GPU with 16GB memory in less than 5 hours.

**Evaluation Protocol.** Many prior approaches report accuracy on only 600 test episodes (Snell et al., 2017; Sun et al., 2019; Zhang et al., 2020; Finn et al., 2017; Triantafillou et al., 2017), but it is also common to report accuracy on 1000 (Tian et al., 2020; Simon et al., 2020), 2000 (Ravichandran et al., 2019), or 3000 (Tian et al., 2020) test episodes. Following the recommendations of recent work (Ye et al., 2020; Rusu et al., 2018), we report average accuracy with a 95% confidence interval on 10,000 test episodes for a lower variance estimate of test performance. For each episode, we draw 15 query examples, using the same protocol for all experiments.

Table 1: Comparison of various classification heads with a fixed pretrained feature extractor. We report average accuracy with 95% confidence interval on the test split of each dataset. Classifiers that require optimization on novel support set are indicted with ✓in the second column. ‡ Nearest Neighbor classifier and ProtoNet classifiers with euclidean and cosine distances are equivalent.

| Classifier | ⚙ | MiniImageNet | TieredImageNet | CIFAR-FS |
|---|---|---|---|---|
| | | 1-shot 5-way | | |
| LR | ✓ | 64.95±0.20 | 71.54±0.22 | 72.78±0.21 |
| LR-Aug | ✓ | 64.93±0.20 | 71.69±0.22 | 72.84±0.21 |
| Linear SVM | ✓ | 64.66±0.20 | 70.95±0.22 | 73.04±0.21 |
| Linear SVM-Aug | ✓ | 64.13±0.20 | 70.60±0.22 | 72.10±0.22 |
| NN / ProtoNet ‡ | | 64.66±0.20 | 70.95±0.22 | 73.23±0.21 |
| Ours | | **66.33±0.20** | **72.06±0.22** | **74.36±0.21** |
| | | 5-shot 5-way | | |
| LR | ✓ | 81.86±0.14 | 86.34±0.15 | 86.98±0.15 |
| LR-Aug | ✓ | 82.05±0.13 | **86.67±0.15** | 87.14±0.15 |
| SVM | ✓ | 81.20±0.14 | 85.46±0.15 | 86.44±0.15 |
| SVM-Aug | ✓ | 81.15±0.14 | 86.04±0.15 | 86.22±0.15 |
| NN | | 76.50±0.15 | 82.24±0.17 | 83.51±0.16 |
| ProtoNet-euclidean | | 82.06±0.14 | 86.36±0.15 | 87.01±0.15 |
| ProtoNet-cosine | | 82.02±0.14 | 86.38±0.15 | 87.16±0.15 |
| Ours | | **82.19±0.14** | 86.50±0.15 | **87.46±0.15** |

### 4.2 Classification Head Comparison with a Fixed Feature Extractor $\mathcal{F}$

**Baselines.** We compare performance of various linear classifiers that operate on the $\ell_2$-normalized feature space of the same pre-trained feature extractor $\mathcal{F}$ in Table 1, isolating it from the effect of the linear classifier choice. To this end, we draw novel few-shot tasks from the test splits of the datasets and use a pre-trained $\mathcal{F}$ to extract features from the support and query examples. Then, we infer the following linear classifiers using the support image features and labels: (i) a logistic regression classifier (LR), (ii) a linear SVM, (iii) a ProtoNet-style linear classifier with euclidean distance, (iv) a ProtoNet-style linear classifier with cosine distance, (v) a nearest neighbor classifier (NN), and (vi) a linear classifier produced by our hypernetwork $\mathcal{H}$. Note that the first two classifiers require training on the support set of novel classes while the remaining classifiers are training-free. We also train LR and SVM classifiers using a combination of five independently augmented (random crops, color jitter and horizontal flip) copies of the support set. For ProtoNet-style classifiers, we compute class prototypes and measure euclidean and cosine distances between the prototypes and the query features. We emphasize that ProtoNet-style classifiers are used only during meta-inference. We do not train feature extractor $\mathcal{F}$ using the ProtoNet objective.

**Results.** Among the baseline classifiers, logistic regression with an augmented support set performs the best. Augmentation leads to marginal but consistent improvement in LR, as similarly observed by Tian et al. (2020). We do not observe the same trend for linear SVM. As seen by Tian et al. (2020), the performance gap between NN and LR is larger for 5-shot tasks compared to 1-shot tasks. ProtoNet classifiers show competitive performance even though the feature extractor is not trained with the ProtoNet objective, importantly demonstrating the effectiveness of the ProtoNet approach in solving few-shot tasks (Snell et al., 2017; Wang et al., 2019b). In the case of 5-shot, ProtoNet classifiers perform comparably to LR-Aug on two datasets without using any data augmentation.

Finally, our hypernetwork produces a classifier that outperforms the baseline classifiers in five out of six cases, with the exception of LR-Aug for 5-shot TieredImageNet. However, LR-Aug requires feature extraction on a 5$\times$ larger augmented support set and training on the support set, whereas a hypernetwork facilitates training-free meta-inference. Inferring an optimal classifier is more challenging in 1-shot cases compared to 5-shot cases. Our hypernetwork infers a classifier that gains >1% higher accuracy on two out of three 1-shot tasks. The results suggest that hypernetworks can find a solution for the classifier parameters that is more optimal than the baseline solutions.

**Analysis.** Predicting a context-aware linear classifier in a training-free manner sets our approach apart from ProtoNet and LR. In Figure 2, we visualize a 1-shot 5-way classification task drawn from

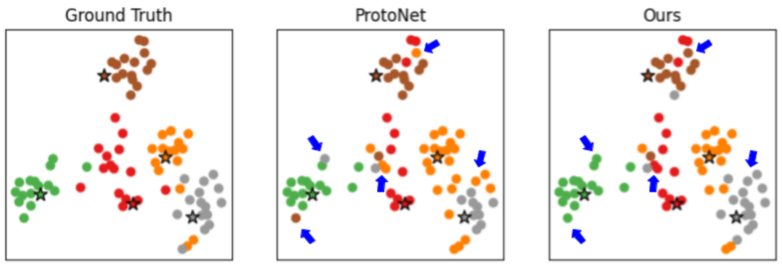

Figure 2: T-SNE projection (Van der Maaten & Hinton, 2008) of support (star) and query (circles) features, color-coded by classes in a novel 1-shot 5-way task from MiniImageNet. Our context-aware approach achieves higher accuracy than context-unaware ProtoNet.

Table 2: Accuracy with 95% confidence interval on support (S) and query (Q) sets of novel 5-shot tasks from MiniImageNet.

| | S | Q |
|---|---|---|
| LR | 99.34±0.03 | 81.76±0.14 |
| Ours | **98.92±0.04** | **82.16±0.14** |
| Ours+LR | 99.37±0.03 | 81.99±0.14 |

the MiniImageNet test set. The context-aware linear classifier produced by our model accurately classifies many query images where context-unaware ProtoNet struggles. While LR also produces context-aware classifier, it requires task-specific optimization which is expensive and prone to overfitting. On Intel(R) Xeon(R) CPU, a single forward pass through a PyTorch implementation of our hypernetwork is $2\times$ faster in predicting a linear classifier for 1-shot tasks than optimization using scikit-learn implementation of LR. In Table 2, we show accuracy of our model and LR on support and query sets for novel 5-shot tasks. It is evident that LR overfits on the support set, leading to reduced generalizability and lower performance on the query set. In contrast, the linear classifier produced by our hypernetwork is more generalizable. Additionally, we demonstrate that further task-specific optimization on the linear classifier predicted by our model using LR leads to overfitting, indicating that classifier produced by our model in a training-free manner is already excellent and further task-specific optimization only leads to sub-optimal solutions.

## 4.3 Ablation Study

We perform ablation on certain elements of our hypernetwork on the TieredImageNet dataset. First, we verify benefits of orthogonal initialization of $W_1$ and $W_2$ and constraining $W$ and $b$ as in Eq 3. Next, we analyze the effect of outlier exposure and batch-size $M$ on the performance of our model. Finally, we present an ablation study of various architecture design choices of the transformer $\mathcal{T}$.

**Initialization of $W_1$ and $W_2$.** We initialize $W_1$ with an orthogonal matrix and $W_2 = W_1^T$, ensuring that the hypernetwork is a pseudo-identity map at initialization. Table 3 (left) presents comparison of orthogonal initialization with Xavier initialization (Glorot & Bengio, 2010). The orthogonal initialization achieves significantly higher accuracy compared to standard initialization.

**Constraints on $W$ and $b$.** In Eq 3, we restrict $||W||_2 = 1$ and $b \in (0, 1)$. Here, we build a classifier using unconstrained $\widehat{W}$ and $\widehat{b}$ from Eq 2 and compare it against a classifier built using $W$ and $b$ from Eq 3. The results are presented in Table 3 (middle). We observe consistent improvement by constraining parameters of the generated classifier.

**Outlier exposure (OE).** We include an outlier exposure training criterion while meta-training $\mathcal{H}$. Table 3 (right) compares the performance of our model with and without outlier exposure. We observe a small but consistent gain by using outlier exposure, justifying its inclusion.

**Batch size $M$.** We notice that the batch size $M$ in the meta-training phase influences the performance of our model. In Figure 3(a), we compare performance with various batch sizes. Observe that the performance improves as one increases batch-size up to a certain value, after which the performance starts to drop. We perform validation to select the value of $M$.

**Transformer Architecture.** In Figure 3(b), we compare performance of our model when the transformer $\mathcal{T}$ has a hidden dimensionality $t$ set to 512, 1024 and 2048. We observe that performance improves with increasing $t$. To maintain a trade-off between accuracy and efficiency, we use $t = 1024$ for 1-shot tasks and $t = 2048$ for 5-shot tasks. In Figure 3(c), we experiment with 1 and 8 heads

Table 3: (Left) Comparison of initialization methods for $W_1$ and $W_2$. (Middle) Comparison of $\{\widehat{W}, \widehat{b}\}$ and $\{W, b\}$ from Eq. 2 and 3. (Right) Use of outlier exposure (OE) in meta-training of $\mathcal{H}$. We report accuracy with 95% confidence interval for 5-way tasks on TieredImageNet test split.

| | Initialization | | Classifier | | Outlier Exposure | |
|---|---|---|---|---|---|---|
| | (Glorot & Bengio, 2010) | Orthogonal | $C(\cdot\|\widehat{W}, \widehat{b})$ | $C(\cdot\|W, b)$ | Without OE | With OE |
| 1-shot | 65.48±0.24 | **72.06±0.22** | 71.31±0.22 | **72.06±0.22** | 71.96±0.22 | **72.06±0.22** |
| 5-shot | 82.04±0.17 | **86.50±0.15** | 85.10±0.16 | **86.50±0.15** | 86.38±0.15 | **86.50±0.15** |

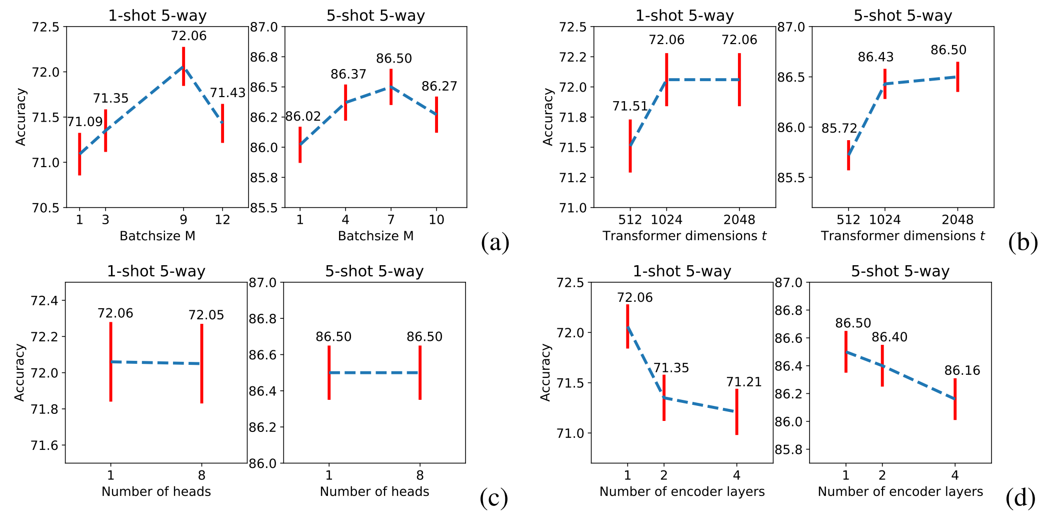

Figure 3: (a) Effect of batch size $M$ on model performance. Ablation study on architecture of the transformer $\mathcal{T}$: (b) hidden dimensions $t$ (c) number of heads (d) number of encoder layers. We report accuracy with 95% confidence interval on TieredImageNet test split.

in the multi-head attention layer of the transformer $\mathcal{T}$. We do not observe any gain in the accuracy by using more than one head. In Figure 3(d), we experiment with 1, 2, and 4 encoding layers in the transformer $\mathcal{T}$. We observe a drop in the performance with more than one layer. Hence, we use a single layer architecture with a single attention head in our experiments.

## 4.4 State of the Art Comparison of a Complete Model

In this section, we compare our complete model ($\mathcal{F} + \mathcal{H}$) with existing state-of-the-art methods. To make the comparison fair, we include only inductive approaches that use ResNet-12 as a backbone and do not use additional unlabeled data for training. Table 4 presents a comparison for MiniImageNet and TieredImageNet. Table 5 presents a comparison for CIFAR-FS. We outperform existing methods that offer optimization-free meta-inference in five out of six cases. Remarkably, we also outperform methods that use task-dependent optimization in four out of six cases.

**MiniImageNet.** Our model achieves competitive performance with state-of-the-art methods. Among the methods that offer optimization-free meta-inference, FEAT (Ye et al., 2020) achieves the best performance. While FEAT outperforms our transformer hypernetwork on 1-shot tasks, we perform comparably to FEAT on 5-shot tasks[1]. Notably, we outperform MABAS (Kim et al., 2020), a state-of-the-art method that requires task-specific optimization, on 1-shot tasks with margin >1%.

**TieredImageNet.** We achieve state-of-the-art performance on this dataset. While FEAT performs significantly better than many other approaches, we outperform FEAT with margin >1% on 1-shot tasks and >1.5% on 5-shot tasks[1]. Among the optimization-based methods, RFS (Tian et al., 2020) achieves the best performance on this dataset. We outperform RFS on both tasks by margin of $\sim$0.5%, while not requiring task-specific optimization.

[1]For thorough comparison, we apply SSL objective from Section 3.2 to FEAT and find it to be ineffective (see Appendix D), perhaps because FEAT already uses contrastive objective during fine-tuning.

Table 4: State-of-the-Art comparison on test split of MiniImageNet and TieredImageNet. We divide previous approaches into those that require task-dependent optimization (indicated with a ✓in the second column) and those that do not. Our method belongs to the latter. We highlight state-of-the-art methods in each group separately. We report our accuracy with a 95% confidence interval on 10K tasks, setting a new state-of-the-art in three out of four cases when compared to each of the two groups. $^\diamond$ Results on MiniImageNet and TieredImageNet datasets are reported from original paper and Xing et al. (2019) respectively. $^\dagger$Results are reported from Ye et al. (2020).

| | | MiniImageNet | | TieredImageNet | |
|---|---|---|---|---|---|
| Method | ⚙ | 1-shot 5-way | 5-shot 5-way | 1-shot 5-way | 5-shot 5-way |
| MTL (Sun et al., 2019) | ✓ | 61.20±1.80 | 75.50±0.80 | - | - |
| MetaOptNet-SVM (Lee et al., 2019) | ✓ | 62.64±0.61 | 78.63±0.46 | 65.99±0.72 | 81.56±0.53 |
| MCRNet-SVM (Zhong et al., 2021) | ✓ | 62.53±0.64 | 80.34±0.47 | - | - |
| Neg-Cosine (Liu et al., 2020) | ✓ | 63.85±0.81 | 81.57±0.56 | - | - |
| RFS (Tian et al., 2020) | ✓ | 64.82±0.60 | 82.14±0.43 | **71.52±0.69** | **86.03±0.49** |
| MABAS (Kim et al., 2020) | ✓ | **65.08±0.86** | **82.70±0.54** | - | - |
| SNAIL (Mishra et al., 2018) | | 55.71±0.99 | 68.88±0.92 | - | - |
| TADAM (Oreshkin et al., 2018) $^\diamond$ | | 58.50±0.30 | 76.70±0.30 | 62.13±0.31 | 81.92±0.30 |
| Shot-Free (Ravichandran et al., 2019) | | 59.04±0.43 | 77.64±0.39 | 66.87±0.43 | 82.64±0.43 |
| TapNet (Yoon et al., 2019) | | 61.65±0.15 | 76.36±0.10 | 63.08±0.15 | 80.26±0.12 |
| DSN (Simon et al., 2020) | | 62.64±0.66 | 78.83±0.45 | 66.22±0.75 | 82.79±0.48 |
| ProtoNet (Snell et al., 2017) $^\dagger$ | | 62.39±0.21 | 80.53±0.14 | 68.23±0.23 | 84.03±0.16 |
| RelationNet2 (DCN) (Zhang et al., 2020) | | 63.92±0.98 | 77.15±0.59 | 68.58±0.63 | 80.65±0.91 |
| FEAT (Ye et al., 2020) | | **66.78±0.20** | 82.05±0.14 | 70.80±0.23 | 84.79±0.16 |
| Ours | | 66.33±0.20 | **82.19±0.14** | **72.06±0.22** | **86.50±0.15** |

Table 5: State-of-the-art comparison on the test split of CIFAR-FS. Previous approaches are divided into two groups: (i) methods that require task-dependent optimization (indicated with ✓in the second column), and (ii) methods that do not. We highlight state-of-the-art methods in each group separately. Our method falls in the second group, where we set a new state-of-the-art. We also outperform the state-of-the-art method in the first group in the 5-shot case. Accuracy reported with 95% confidence interval on 10K tasks. $^\dagger$Results are reported from (Tian et al., 2020).

| Method | ⚙ | 1-shot 5-way | 5-shot 5-way |
|---|---|---|---|
| MetaOptNet-SVM (Lee et al., 2019) | ✓ | 72.0±0.7 | 84.2±0.5 |
| MABAS (Kim et al., 2020) | ✓ | 73.5±0.9 | 85.5±0.7 |
| RFS (Tian et al., 2020) | ✓ | 73.9±0.8 | **86.9±0.5** |
| MCRNet-SVM (Zhong et al., 2021) | ✓ | **74.7±0.7** | 86.8±0.5 |
| Shot-Free (Ravichandran et al., 2019) | | 69.2±NA | 84.7±NA |
| ProtoNet (Snell et al., 2017) $^\dagger$ | | 72.2±0.7 | 83.5±0.5 |
| DSN (Simon et al., 2020) | | 72.3±0.8 | 85.1±0.6 |
| Ours | | **74.4±0.2** | **87.5±0.2** |

**CIFAR-FS.** Among the group of methods that does not require task-specific training, we achieve state-of-the-art performance. We observe $\sim$2% better accuracy than DSN (Simon et al., 2020) on both tasks. Note that we also outperform RFS (Tian et al., 2020) by a margin of $\sim$0.6% on 5-shot tasks. We emphasize that, unlike our approach, RFS requires optimization on novel tasks.

## 5 Conclusions

Many recent FSL methods use linear classifiers in conjunction with a fixed, task-agnostic feature extractor to classify query images but make various trade-offs. Previous training-free approaches such as ProtoNet produce this classifier without considering the context. Context-aware approaches such as Logistic Regression (LR) outperform context-unaware approaches but require expensive task-specific training that is prone to overfitting to a small support set. We present a novel transformer hypernetwork approach that combines the best of both worlds. Through experiments, we show our approach is able to achieve contextualization with fast, training-free meta-inference while offering better generalization than LR. Additionally, comparisons of our complete model with previous training-free approaches demonstrate state-of-the-art performance in five out of six cases. These strong results suggest the potential of further extensions; given the impact of the representation power of the feature extractor on our hypernetwork's performance, we believe using a hypernetwork to generate parameters for the feature extractor as well to be a promising future direction.

## References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Luca Bertinetto, Joao F Henriques, Philip Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. In *International Conference on Learning Representations*, 2018.

Rich Caruana, Nikos Karampatziakis, and Ainur Yessenalina. An empirical evaluation of supervised learning in high dimensions. In *Proceedings of the 25th international conference on Machine learning*, pp. 96–103, 2008.

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

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

Carl Doersch, Ankush Gupta, and Andrew Zisserman. Crosstransformers: spatially-aware few-shot transfer. *arXiv preprint arXiv:2007.11498*, 2020.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pp. 1126–1135. PMLR, 2017.

Victor Garcia and Joan Bruna. Few-shot learning with graph neural networks. *arXiv preprint arXiv:1711.04043*, 2017.

Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Dropblock: a regularization method for convolutional networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 10750–10760, 2018.

Spyros Gidaris, Andrei Bursuc, Nikos Komodakis, Patrick Pérez, and Matthieu Cord. Boosting few-shot visual learning with self-supervision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8059–8068, 2019.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings, 2010.

Jonathan Gordon, John Bronskill, Matthias Bauer, Sebastian Nowozin, and Richard Turner. Meta-learning probabilistic inference for prediction. In *International Conference on Learning Representations*, 2018.

David Ha, Andrew Dai, and Quoc V Le. HyperNetworks. *International Conference on Learning Representations*, 2017.

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

Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Deep anomaly detection with outlier exposure. In *International Conference on Learning Representations*, 2018.

Jaekyeom Kim, Hyoungseok Kim, and Gunhee Kim. Model-agnostic boundary-adversarial sampling for test-time generalization in few-shot learning. *Computer Vision–ECCV*, pp. 599–617, 2020.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10657–10665, 2019.

Aoxue Li, Weiran Huang, Xu Lan, Jiashi Feng, Zhenguo Li, and Liwei Wang. Boosting few-shot learning with adaptive margin loss. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12576–12584, 2020.

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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7260–7268, 2019.

Bin Liu, Yue Cao, Yutong Lin, Qi Li, Zheng Zhang, Mingsheng Long, and Han Hu. Negative margin matters: Understanding margin in few-shot classification. In *European Conference on Computer Vision*, pp. 438–455. Springer, 2020.

Puneet Mangla, Nupur Kumari, Abhishek Sinha, Mayank Singh, Balaji Krishnamurthy, and Vineeth N Balasubramanian. Charting the right manifold: Manifold mixup for few-shot learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 2218–2227, 2020.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. In *International Conference on Learning Representations*, 2018.

Yuval Nirkin, Lior Wolf, and Tal Hassner. Hyperseg: Patch-wise hypernetwork for real-time semantic segmentation. *arXiv preprint arXiv:2012.11582*, 2020.

Boris N Oreshkin, Pau Rodriguez, and Alexandre Lacoste. Tadam: task dependent adaptive metric for improved few-shot learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 719–729, 2018.

Siyuan Qiao, Chenxi Liu, Wei Shen, and Alan L Yuille. Few-shot image recognition by predicting parameters from activations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7229–7238, 2018.

Jathushan Rajasegaran, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Mubarak Shah. Self-supervised knowledge distillation for few-shot learning. *arXiv preprint arXiv:2006.09785*, 2020.

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. *International Conference on Learning Representations*, 2017.

Avinash Ravichandran, Rahul Bhotika, and Stefano Soatto. Few-shot learning with embedded class models and shot-free meta training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 331–339, 2019.

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 *International Conference on Learning Representations*, 2018.

Pau Rodríguez, Issam Laradji, Alexandre Drouin, and Alexandre Lacoste. Embedding propagation: Smoother manifold for few-shot classification. In *European Conference on Computer Vision*, pp. 121–138. Springer, 2020.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Andrei A Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. In *International Conference on Learning Representations*, 2018.

Christian Simon, Piotr Koniusz, Richard Nock, and Mehrtash Harandi. Adaptive subspaces for few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4136–4145, 2020.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 4080–4090, 2017.

Przemysław Spurek, Sebastian Winczowski, Jacek Tabor, Maciej Zamorski, Maciej Zięba, and Tomasz Trzciński. Hypernetwork approach to generating point clouds. *arXiv preprint arXiv:2003.00802*, 2020.

Qianru Sun, Yaoyao Liu, Tat-Seng Chua, and Bernt Schiele. Meta-transfer learning for few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 403–412, 2019.

Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1199–1208, 2018.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pp. 1139–1147. PMLR, 2013.

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 *European Conference on Computer Vision*, 2020.

Eleni Triantafillou, Richard Zemel, and Raquel Urtasun. Few-shot learning through an information retrieval lens. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 2252–2262, 2017.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pp. 3637–3645, 2016.

Xin Wang, Fisher Yu, Ruth Wang, Trevor Darrell, and Joseph E Gonzalez. Tafe-net: Task-aware feature embeddings for low shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1831–1840, 2019a.

Yan Wang, Wei-Lun Chao, Kilian Q Weinberger, and Laurens van der Maaten. Simpleshot: Revisiting nearest-neighbor classification for few-shot learning. *arXiv preprint arXiv:1911.04623*, 2019b.

Chen Xing, Negar Rostamzadeh, Boris Oreshkin, and Pedro O O. Pinheiro. Adaptive cross-modal few-shot learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper/2019/file/d790c9e6c0b5e02c87b375e782ac01bc-Paper.pdf`.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, pp. 10524–10533. PMLR, 2020.

Han-Jia Ye, Hexiang Hu, De-Chuan Zhan, and Fei Sha. Few-shot learning via embedding adaptation with set-to-set functions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8808–8817, 2020.

Sung Whan Yoon, Jun Seo, and Jaekyun Moon. Tapnet: Neural network augmented with task-adaptive projection for few-shot learning. In *International Conference on Machine Learning*, pp. 7115–7123. PMLR, 2019.

Chris Zhang, Mengye Ren, and Raquel Urtasun. Graph hypernetworks for neural architecture search. In *International Conference on Learning Representations*, 2018.

Xueting Zhang, Yuting Qiang, Flood Sung, Yongxin Yang, and Timothy Hospedales. Relation-net2: Deep comparison network for few-shot learning. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2020.

Xian Zhong, Cheng Gu, Wenxin Huang, Lin Li, Shuqin Chen, and Chia-Wen Lin. Complementing representation deficiency in few-shot image classification: A meta-learning approach. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 2677–2684. IEEE, 2021.

## A Interpolation and Extrapolation of classification ways

We analyze the capability of our transformer hypernetwork to interpolate and extrapolate the number of classification ways during meta-inference. Specifically, we meta-train our model for 20-way (5-way) few-shot tasks and meta-test for 5-way (20-way) tasks to study the interpolation (extrapolation) capability. We follow the same training procedure as discussed in Section 3.2, except for the outlier exposure. Results on TieredImageNet dataset are presented in Table 6. We observe that our model achieves greater performance during meta-inference if meta-trained with higher number of ways. Snell et al. (2017) made a similar observation for ProtoNet. In other words, our model has good interpolation capabilities. On the contrary and unsurprisingly, we see a small performance drop when we meta-train our model with lower number of ways than meta-inference.

Table 6: Accuracy of our model with 95% confidence interval on test split of TieredImageNet dataset when meta-trained and meta-tested with different number of ways.

| | 1-shot 5-way | | 5-shot 5-way | |
|---|---|---|---|---|
| Meta-train / Meta-test | 5-way | 20-way | 5-way | 20-way |
| 5-way | 71.09±0.22 | 71.49±0.23 | 86.02±0.15 | 86.15±0.15 |
| 20-way | 43.75±0.11 | 44.30±0.11 | 65.44±0.10 | 65.66±0.10 |

## B ResNet-18 Backbone

We analyze performance of our model with ResNet-18, a deeper backbone than ResNet-12. We pretrain ResNet-18 and train the hypernetwork following the same procedure and hyperparameters as stated in Section 3.2 and 4.1 of the main paper. Table 7 shows results for MiniImageNet dataset. We observe reduced performance with ResNet-18 with the current set of hyperparameters. While a dedicated hyperparameter search may further improve the performance of the model with ResNet-18 backbone, these results corroborate previous findings that shallower ResNets tend to outperform deeper ones in FSL settings for MiniImageNet (Chen et al., 2019). We hypothesize that the size and complexity of MiniImageNet may be too small to fully utilize the expanded capacity of ResNet-18 in the few-shot setting.

Table 7: Accuracy of our model with 95% confidence interval using ResNet-12 and ResNet-18 backbones on test split of MiniImageNet dataset.

| | 1-shot 5-way | 5-shot 5-way |
|---|---|---|
| ResNet-12 | **66.33±0.20** | **82.19±0.14** |
| ResNet-18 | 64.53±0.20 | 79.99±0.15 |

## C Context-aggregation and permutation invariance

Predicting a multi-class linear classifier requires that the hypernetwork is permutation invariant and aggregates context to produce task-dependent classifier. Here we analyze importance of these two properties. For clarity in the following discussion, we re-express equation 1 as follows:

$$\widehat{P} = \frac{1}{\sqrt{d}}\, W_2 \mathcal{T}(\sqrt{d} W_1 P) \tag{4}$$

$$\widehat{P} = \frac{1}{\sqrt{d}}\, W_2 \mathcal{T}(Q) \tag{5}$$

Following Section 3.1, note that $\widehat{P} \in \mathbb{R}^{(d+1)\times N}$, $P \in \mathbb{R}^{(d+1)\times N}$ and $Q \in \mathbb{R}^{t\times N}$. A hypernetwork is permutation invariant if it is insensitive to the permutations in the columns of $P$ and aggregates context if each column of $\widehat{P}$ depends on all columns $P$. To analyze these properties, we experiment with the following architectures in place of a transformer $\mathcal{T}$ in the above formulation while keeping the other components of the hypernetwork unaltered.

- **Linear** layer with $t \times t$ weight matrix processes $N$ columns of $Q$ identically and independently, formulating a permutation invariant hypernetwork that does not aggregate context.
- **Conv1D** layer with $N \times (N \times 1)$ filters processes $t$ rows of $Q$ identically and independently, formulating a hypernetwork that aggregates context but is not permutation invariant.
- **BiLSTM** considers $Q$ as a sequence of $N$ vectors (each of length $t$), formulating a hypernetwork that aggregates context but is not permutation invariant.
- **Graph Neural Network (GNN)** considers $Q$ as fully-connected graph with $N$ nodes (each corresponding to one column of $Q$), formulating a hypernetwork that aggregates context and is also permutation invariant. We follow the GNN architecture presented by Garcia & Bruna (2017). However, note that we use GNN in the hypernetwork and Garcia & Bruna (2017) used it in the main network.
- **Transformer** (our proposed method) considers $Q$ as a sequence of $N$ vectors, formulating a hypernetwork that aggregates context and is also permutation invariant.

We compare the above variants in Table 8. We observe that variants of the hypernetwork that aggregate context frequently outperform their counterpart. Transformer hypernetwork outperforms variants that are either permutation invariant or aggregate context but not both. Surprisingly, the GNN underperforms the transformer despite having both desirable properties. We attribute the difference in the performance between transformer and GNN to their architectures. Optimizing a hypernetwork is more challenging than the main network. We hypothesize the architecture of the transformer is easier to optimize and trains well when incorporated in the hypernetwork, achieving higher performance than GNN.

Table 8: Analyzing properties of context-aggregation and permutation invariance on 5-shot 5-way MiniImageNet. We report accuracy with 95% confidence interval. Transformer aggregates context in permutation invariant manner and achieves the highest accuracy.

| | Permutation invariance | Context aggregation | 1-shot 5-ways | 5-shot 5-ways |
|---|---|---|---|---|
| Linear | ✓ | | 64.93±0.20 | 80.61±0.14 |
| Conv1D | | ✓ | 66.07±0.20 | 81.42±0.14 |
| BiLSTM | | ✓ | 64.40±0.10 | 80.12±0.15 |
| GNN | ✓ | ✓ | 64.90±0.20 | 79.94±0.15 |
| Transformer | ✓ | ✓ | **66.33±0.20** | **82.19±0.14** |

## D Self-supervised pretraining

The pretraining objective of the feature extractor $\mathcal{F}$ plays a significant role in the final performance of the model (Gidaris et al., 2019). Our method adopts self-supervised pretraining of the backbone network. For thoroughness, we include self-supervised pretraining discussed in Section 3.1 in the most competitive SOTA method, i.e. FEAT(Ye et al., 2020), and observe the performance. The result presented in Table 9 shows that self-supervised pretraining does not improve FEAT. We speculate that the fine-tuning of the feature-extractor with contrastive objective as done in FEAT renders the self-supervised pretraining redundant.

Table 9: Accuracy of FEAT(Ye et al., 2020) with 95% confidence interval when the backbone is pretrained using supervised (Sup.) vs supervised and self-supervised (Sup.+Self-Sup.) objectives. Accuracy for Supp. objective is reported from (Ye et al., 2020).

| | 1-shot 5-way | | 5-shot 5-way | |
|---|---|---|---|---|
| | Sup. | Sup.+Self-Sup. | Sup. | Sup.+Self-Sup. |
| MiniImageNet | 66.78±N/A | 66.83±0.21 | 82.05±N/A | 80.14±0.16 |
| TieredImageNet | 70.80±0.23 | 70.23±0.23 | 84.79±0.16 | 84.49±0.16 |