Few-shot Learning with Noisy Labels
— Supplemental Material —

Kevin J Liang[1] Samrudhdhi B. Rangrej[2] Vladan Petrovic[1] Tal Hassner[1]
[1]Facebook AI Research [2]McGill University
kevinjliang@fb.com

We include supplemental material for our work here. Sec. A shows the mislabeled samples in the 5-way 5-shot support set in Fig. 1 of the main paper, as well as two more example noisy support sets. We discuss computational complexity considerations for iteratively solving for the median in Sec. B. In Sec. C, we investigate noisy few-shot performance for different numbers of shots from the 5-shot setting considered in Sec. 6.2 of the main paper. Sec. D contains descriptions and additional implementation details of the baselines that we compare against. We perform further ablation studies beyond Sec. 6.4 of the main paper, investigating feature extractors, hyperparameter settings, and architectural design choices of TraNFS in Sec. E.

A. Noisy support set examples

Noisy few-shot learning is a challenging problem. Even before adding noise, there can be significant variation within a class, largely due to the manner in which the ImageNet [2] dataset (from which MiniImageNet [10] and TieredImageNet [7] are derived) was constructed. Some images in the *clean* version of ImageNet are mislabeled due to human error, but even among the correctly labeled objects, there are non-canonical views, images with multiple objects (possibly from multiple ImageNet classes), and classes that are close to synonymous. We provide several examples of noisy support sets from MiniImageNet with 40% symmetric label swap noise [9] in Fig. 1, with the clean and noisy samples framed in green and red, respectively. While humans are generally able to separate the noisy samples from the clean samples with some scrutiny, this is in large part due to prior conceptual understandings of the classes depicted. Few-shot models presented with support sets such as those in Fig. 1 are tasked with learning how to distinguish the depicted classes *without having previously seen these classes*, a much more difficult problem.

B. A Note On Median Complexity

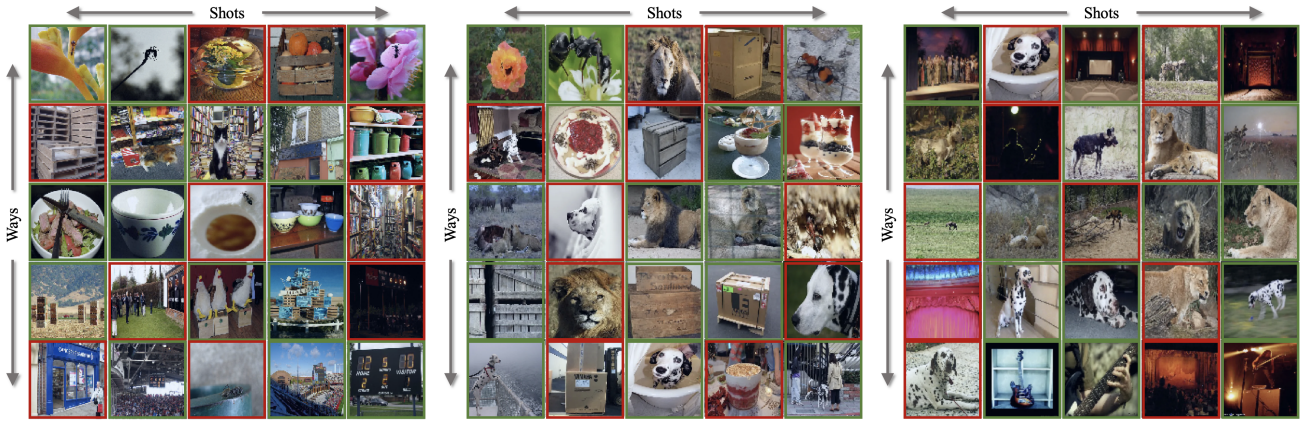
As discussed in Sec. 4.1 in the main paper, median computation has to be performed iteratively since no closed form solution exists. We have chosen the 2^{nd}- over 1^{st}-order optimization as the former provides an optimal step size at each iteration, speeding up convergence. This choice may seem costly at first glance, but computational complexity analysis of Eq. (7) shows negligible 2^{nd}-order method overhead. Each median update iteration takes $4DK+2K-D$ flops for gradient computation, K flops for optimal step calculation (2^{nd}-order method overhead), and $2D$ flops for parameter update. We emphasize that this optimization is done to calculate a median prototype (as opposed to updating the model weights); D and K are both fairly small.

Table 1. Few-shot performance with symmetric label swap noise on 5-way 3-shot MiniImageNet [10].

Model \ Noise Proportion	0%	33.3%
Oracle	62.60 ± 0.17	56.89 ± 0.18
Nearest $k = 1$	52.98 ± 0.18	39.92 ± 0.18
Nearest $k = 3$	50.59 ± 0.18	38.76 ± 0.16
Nearest $k = 5$	50.20 ± 0.17	40.05 ± 0.16
Linear Classifier	61.54 ± 0.17	46.06 ± 0.17
Matching Networks [10]	57.86 ± 0.18	44.92 ± 0.18
MAML [3]	59.79 ± 0.20	40.41 ± 0.17
Vanilla ProtoNet [8]	62.54 ± 0.18	48.78 ± 0.19
RNNP [6]	62.57 ± 0.17	48.76 ± 0.19
Median	62.60 ± 0.17	50.40 ± 0.19
Absolute $T = 10.0$	61.77 ± 0.17	50.93 ± 0.19
Absolute $T = 25.0$	62.54 ± 0.17	50.84 ± 0.19
Absolute $T = 50.0$	62.69 ± 0.17	50.06 ± 0.19
Euclidean $T = 10.0$	62.58 ± 0.17	50.83 ± 0.19
Euclidean $T = 25.0$	62.62 ± 0.18	50.06 ± 0.19
Euclidean $T = 50.0$	62.62 ± 0.17	49.51 ± 0.19
Cosine $T = 0.2$	62.75 ± 0.17	49.63 ± 0.19
Cosine $T = 0.5$	62.55 ± 0.17	49.15 ± 0.19
Cosine $T = 1.0$	62.52 ± 0.17	49.20 ± 0.19
Cosine $T = 2.0$	62.63 ± 0.17	49.05 ± 0.19
Cosine $T = 5.0$	62.54 ± 0.17	48.96 ± 0.19
TraNFS-2	64.17 ± 0.18	53.35 ± 0.21
TraNFS-3	$\mathbf{64.28 \pm 0.18}$	$\mathbf{53.84 \pm 0.21}$

C. Different number of shots

While the experiments in Sec. 6.2 are conducted with 5 shots, many of our findings on noisy FSL apply to other numbers of shots as well. We provide additional results be-



(a) (b) (c)

Figure 1. **Noisy support set examples.** Images with green boxes are clean samples from the original class, while red boxes are mislabeled samples due to symmetric label swaps. (a) is the support set shown in Fig. 1 of the main paper.

Table 2. Few-shot performance with outlier noise on 5-way 3-shot MiniImageNet [10].

Model \ Noise Proportion	0%	33.3%
Oracle	62.60 ± 0.17	56.89 ± 0.18
Nearest $k = 1$	53.07 ± 0.18	44.66 ± 0.18
Nearest $k = 3$	50.40 ± 0.18	41.59 ± 0.17
Nearest $k = 5$	50.24 ± 0.17	42.26 ± 0.17
Linear Classifier	61.58 ± 0.17	51.21 ± 0.18
Matching Networks [10]	57.82 ± 0.18	48.56 ± 0.19
MAML [3]	59.76 ± 0.19	47.08 ± 0.19
Vanilla ProtoNet [8]	62.43 ± 0.17	52.78 ± 0.19
RNNP [6]	62.55 ± 0.17	52.88 ± 0.19
Median	62.53 ± 0.17	53.82 ± 0.19
Absolute $T = 10.0$	61.54 ± 0.17	53.76 ± 0.19
Absolute $T = 25.0$	62.47 ± 0.17	54.07 ± 0.19
Absolute $T = 50.0$	62.69 ± 0.17	53.73 ± 0.19
Euclidean $T = 10.0$	62.56 ± 0.18	54.10 ± 0.19
Euclidean $T = 25.0$	62.57 ± 0.17	53.72 ± 0.18
Euclidean $T = 50.0$	62.76 ± 0.17	53.55 ± 0.19
Cosine $T = 0.2$	62.58 ± 0.17	53.46 ± 0.19
Cosine $T = 0.5$	62.50 ± 0.17	53.03 ± 0.19
Cosine $T = 1.0$	62.50 ± 0.17	52.84 ± 0.19
Cosine $T = 2.0$	62.72 ± 0.17	53.16 ± 0.19
Cosine $T = 5.0$	62.63 ± 0.18	53.19 ± 0.19
TraNFS-2	$\mathbf{63.63 \pm 0.18}$	$\mathbf{54.75 \pm 0.20}$
TraNFS-3	63.61 ± 0.18	54.72 ± 0.20

low for MiniImageNet [10] with $K = \{3, 10\}$ shots.

C.1. 3-shot MiniImageNet

We show 5-way 3-shot performance on MiniImageNet with symmetric label swap (Table 1) and outlier (Table 2) noise. Note that we do not show results for paired label swap noise, as at 33.3% noise, paired label noise is identical to symmetric, and at 66.7%, the clean class is dominated by the noisy class.

We observe similar trends in the 3-shot setting as in the 5-way 5-shot experiments reported in Tables 1, 2, and 3 of the main paper. The baseline methods suffer dramatically from replacing a clean sample in the support set with a single noisy sample, with ProtoNet [8] suffering almost a 14% drop in accuracy in the 33.3% symmetric label swap noise setting, as compared to the 5.71% drop in accuracy from removing a shot. Our proposed ProtoNet variants at various temperatures T all outperform vanilla ProtoNet. On the other hand, our TraNFS surpasses vanilla ProtoNet by 5.06% and impressively is only 3.05% short of the Oracle, despite not having knowledge of the noisy samples within the support set.

C.2. 10-shot MiniImageNet

We show 5-way 10-shot performance on MiniImageNet with symmetric label swap (Table 3), paired label swap (Table 4), and outlier (Table 5) noise. Note that we only show 20%, 30%, and 40% noise proportion for paired label swap noise, as at 0% and 10%, paired label swapping is no different from symmetric swapping (Table 3) for 10 shots, and at 50% and above the noisy class would have either a share of or the outright majority.

Our proposed TraNFS shines with 10-shot tasks as well. As in the 5-shot case, our method does especially well in moderate to high noise levels. In particular, we observe over 5% absolute improvement from TraNFS over vanilla ProtoNet at 40% and 50% symmetric label swap noise and an impressive 6.21% improvement for 40% paired label swap noise. TraNFS is also the best method for rejecting outlier noise as well.

D. Method descriptions

Fig. 2 shows a visual comparison of some of the baselines we compare against. We discuss implementation details below.

Oracle. When noise appears in a support set, accuracy of the few-shot model is reduced for two reasons: (1) misla-

Table 3. Few-shot performance with symmetric label swap noise on 5-way 10-shot MiniImageNet [10].

Model \ Noise Proportion	0%	10%	20%	30%	40%	50%	60%	70%
Oracle	73.62 ± 0.14	72.78 ± 0.15	71.78 ± 0.15	70.82 ± 0.15	69.27 ± 0.16	64.70 ± 0.17	60.59 ± 0.17	53.88 ± 0.18
Nearest $k = 1$	53.02 ± 0.19	49.04 ± 0.18	45.02 ± 0.18	40.87 ± 0.18	37.28 ± 0.17	33.13 ± 0.17	29.07 ± 0.16	24.64 ± 0.15
Nearest $k = 3$	53.79 ± 0.19	50.84 ± 0.18	47.24 ± 0.18	43.21 ± 0.17	38.58 ± 0.17	33.72 ± 0.16	29.00 ± 0.15	24.24 ± 0.13
Nearest $k = 5$	55.03 ± 0.20	53.08 ± 0.19	50.29 ± 0.19	46.50 ± 0.18	41.97 ± 0.17	36.51 ± 0.16	30.75 ± 0.15	25.26 ± 0.14
Linear Classifier	72.08 ± 0.14	68.56 ± 0.15	64.17 ± 0.16	58.90 ± 0.16	52.68 ± 0.16	45.43 ± 0.16	37.20 ± 0.15	29.18 ± 0.14
Matching Networks [10]	62.63 ± 0.19	60.81 ± 0.19	58.21 ± 0.19	54.79 ± 0.19	50.05 ± 0.19	43.47 ± 0.18	35.90 ± 0.17	28.70 ± 0.15
MAML [3]	64.37 ± 0.18	64.42 ± 0.18	55.27 ± 0.18	44.17 ± 0.18	44.10 ± 0.18	44.01 ± 0.18	32.03 ± 0.16	20.04 ± 0.13
Vanilla ProtoNet [8]	$\mathbf{73.65 \pm 0.14}$	71.80 ± 0.15	69.19 ± 0.15	65.28 ± 0.16	59.52 ± 0.17	51.42 ± 0.18	41.43 ± 0.18	32.29 ± 0.18
RNNP [6]	73.47 ± 0.14	71.80 ± 0.15	69.37 ± 0.16	65.88 ± 0.17	60.51 ± 0.18	52.25 ± 0.19	41.74 ± 0.19	32.47 ± 0.19
Median	73.54 ± 0.14	71.90 ± 0.15	69.30 ± 0.15	65.59 ± 0.16	59.88 ± 0.17	51.42 ± 0.18	41.13 ± 0.19	31.99 ± 0.18
Absolute $T = 10.0$	71.12 ± 0.15	69.58 ± 0.16	66.77 ± 0.17	62.27 ± 0.18	54.91 ± 0.20	45.13 ± 0.21	35.05 ± 0.20	28.20 ± 0.18
Absolute $T = 25.0$	73.10 ± 0.14	71.66 ± 0.15	69.13 ± 0.16	65.15 ± 0.17	58.63 ± 0.18	49.02 ± 0.19	38.40 ± 0.20	30.05 ± 0.18
Absolute $T = 50.0$	73.49 ± 0.14	71.88 ± 0.15	69.42 ± 0.16	65.52 ± 0.16	59.54 ± 0.18	50.65 ± 0.19	40.04 ± 0.19	31.34 ± 0.18
Euclidean $T = 10.0$	73.11 ± 0.15	71.60 ± 0.15	69.28 ± 0.16	65.57 ± 0.17	59.59 ± 0.18	50.45 ± 0.19	39.73 ± 0.19	30.88 ± 0.18
Euclidean $T = 25.0$	73.57 ± 0.14	71.98 ± 0.15	69.50 ± 0.16	65.78 ± 0.16	60.02 ± 0.18	51.59 ± 0.18	40.96 ± 0.19	31.98 ± 0.18
Euclidean $T = 50.0$	73.64 ± 0.14	71.96 ± 0.15	69.36 ± 0.16	65.68 ± 0.16	59.95 ± 0.17	51.58 ± 0.18	41.30 ± 0.19	32.18 ± 0.18
Cosine $T = 0.2$	73.62 ± 0.14	71.94 ± 0.15	69.44 ± 0.15	65.65 ± 0.16	59.91 ± 0.17	51.49 ± 0.18	41.14 ± 0.19	32.13 ± 0.18
Cosine $T = 0.5$	73.60 ± 0.14	71.85 ± 0.15	69.26 ± 0.15	65.46 ± 0.16	59.64 ± 0.17	51.50 ± 0.18	41.23 ± 0.19	32.18 ± 0.18
Cosine $T = 1.0$	73.57 ± 0.14	71.78 ± 0.15	69.13 ± 0.15	65.36 ± 0.16	59.62 ± 0.17	51.56 ± 0.18	41.44 ± 0.18	32.24 ± 0.18
Cosine $T = 2.0$	$\mathbf{73.65 \pm 0.14}$	71.83 ± 0.15	69.08 ± 0.16	65.25 ± 0.16	59.58 ± 0.17	51.26 ± 0.18	41.36 ± 0.18	32.10 ± 0.18
Cosine $T = 5.0$	73.55 ± 0.14	71.73 ± 0.15	69.06 ± 0.15	65.19 ± 0.16	59.42 ± 0.17	51.38 ± 0.18	41.31 ± 0.19	32.19 ± 0.18
TraNFS-2	72.80 ± 0.15	71.86 ± 0.15	70.54 ± 0.16	68.25 ± 0.17	64.29 ± 0.19	57.04 ± 0.21	$\mathbf{45.84 \pm 0.24}$	35.09 ± 0.23
TraNFS-3	73.17 ± 0.15	$\mathbf{72.14 \pm 0.15}$	$\mathbf{70.71 \pm 0.16}$	$\mathbf{68.48 \pm 0.17}$	$\mathbf{64.59 \pm 0.18}$	$\mathbf{57.45 \pm 0.21}$	45.80 ± 0.24	$\mathbf{35.12 \pm 0.23}$

Table 4. Few-shot performance with paired label swap noise on 5-way 10-shot MiniImageNet [10].

Model \ Noise Proportion	20%	30%	40%
Oracle	71.78 ± 0.15	70.82 ± 0.15	69.27 ± 0.16
Nearest $k = 1$	44.85 ± 0.18	40.80 ± 0.18	36.58 ± 0.17
Nearest $k = 3$	46.96 ± 0.18	42.31 ± 0.17	37.32 ± 0.16
Nearest $k = 5$	49.88 ± 0.18	45.21 ± 0.17	39.47 ± 0.17
Linear Classifier	63.54 ± 0.16	56.70 ± 0.16	47.85 ± 0.16
Matching Networks [10]	57.74 ± 0.19	52.80 ± 0.18	45.37 ± 0.17
MAML [3]	55.05 ± 0.18	41.95 ± 0.18	41.83 ± 0.18
Vanilla ProtoNet [8]	68.34 ± 0.16	62.59 ± 0.16	52.73 ± 0.17
RNNP [6]	68.89 ± 0.16	63.86 ± 0.17	54.06 ± 0.18
Median	69.04 ± 0.15	63.50 ± 0.16	53.61 ± 0.17
Absolute $T = 50.0$	69.07 ± 0.16	63.62 ± 0.17	53.78 ± 0.18
Absolute $T = 25.0$	69.00 ± 0.16	63.75 ± 0.17	53.88 ± 0.18
Absolute $T = 10.0$	66.94 ± 0.17	61.82 ± 0.18	52.28 ± 0.20
Euclidean $T = 50.0$	68.86 ± 0.16	63.20 ± 0.17	53.24 ± 0.17
Euclidean $T = 25.0$	69.12 ± 0.16	63.48 ± 0.17	53.58 ± 0.17
Euclidean $T = 10.0$	68.91 ± 0.16	63.73 ± 0.17	53.81 ± 0.18
Cosine $T = 5.0$	68.50 ± 0.15	62.72 ± 0.16	52.76 ± 0.17
Cosine $T = 2.0$	68.42 ± 0.15	62.63 ± 0.16	52.87 ± 0.17
Cosine $T = 1.0$	68.48 ± 0.16	62.59 ± 0.17	52.86 ± 0.17
Cosine $T = 0.5$	68.58 ± 0.15	62.76 ± 0.16	52.90 ± 0.17
Cosine $T = 0.2$	68.82 ± 0.16	63.14 ± 0.17	53.27 ± 0.17
TraNFS-2	70.13 ± 0.16	66.20 ± 0.17	56.97 ± 0.20
TraNFS-3	$\mathbf{70.38 \pm 0.16}$	$\mathbf{67.03 \pm 0.18}$	$\mathbf{58.94 \pm 0.21}$

beled samples provide the model with misleading information about the class, and (2) clean samples that would have otherwise been informative were removed from the support set. FSL performance can be heavily influenced by the number of shots, especially in the low-data regime, so we find it important to separate out the aforementioned two sources of performance degradation. For this purpose, we include in our results tables an *Oracle* model consisting of a vanilla ProtoNet [8] with prototypes produced from only the correctly labeled samples in the support set. Note that the Oracle requires knowing the identities of the noisy samples, which cannot be reasonably expected in many real-world settings and is thus not a fair comparison with the other methods, but we include it to give a sense of constructive information content still available in the support set after noise corruption.

Nearest Neighbors. In the context of FSL, nearest neighbors (Fig. 2a) is a simple, non-parametric classification technique which classifies query samples based on the labels of the k closest support samples in embedding space. Whichever class has the plurality among the k nearest neighbor support samples is the prediction, with ties broken uniformly at random among the tied classes. We report results for $k \in \{1, 3, 5\}$.

Linear Classifier. We train a single fully connected layer $\mathbb{R}^D \rightarrow \mathbb{R}^N$ on top of frozen convolutional features (Fig. 2c). For each episode, the parameters of the fully connected layer are learned with the AdamW [5] optimizer with weight decay 0.01, trained for 100 steps. Note that this approach resembles the Baseline method [1], with the primary difference being that we use the ProtoNet objective and episodic meta-training to learn the feature extractor $\mathcal{F}$, as opposed to the softmax cross entropy loss with batch learning on the base classes.

Matching Networks [10]. Matching networks (Fig. 2b) use an attention mechanism to compare the embedded query

Table 5. Few-shot performance with outlier noise on 5-way 10-shot MiniImageNet [10].

Model \ Noise Proportion	0%	10%	20%	30%	40%	50%	60%	70%
Oracle	73.62 ± 0.14	72.78 ± 0.15	71.78 ± 0.15	70.82 ± 0.15	69.27 ± 0.16	64.70 ± 0.17	60.59 ± 0.17	53.88 ± 0.18
Nearest $k = 1$	53.14 ± 0.19	50.61 ± 0.19	48.25 ± 0.18	45.62 ± 0.18	42.91 ± 0.18	39.99 ± 0.17	37.06 ± 0.17	33.66 ± 0.17
Nearest $k = 3$	53.55 ± 0.19	51.49 ± 0.18	49.16 ± 0.18	46.50 ± 0.18	43.69 ± 0.17	40.63 ± 0.17	37.07 ± 0.16	33.15 ± 0.15
Nearest $k = 5$	54.81 ± 0.20	53.31 ± 0.19	51.46 ± 0.19	49.18 ± 0.18	46.35 ± 0.18	43.13 ± 0.17	39.32 ± 0.16	35.05 ± 0.16
Linear Classifier	71.90 ± 0.15	69.62 ± 0.15	66.94 ± 0.16	63.70 ± 0.16	59.86 ± 0.16	55.31 ± 0.17	49.84 ± 0.17	43.42 ± 0.17
Matching Networks [10]	62.68 ± 0.19	61.37 ± 0.19	59.58 ± 0.19	57.52 ± 0.19	54.58 ± 0.19	51.12 ± 0.19	46.48 ± 0.19	40.68 ± 0.18
MAML [3]	64.30 ± 0.18	64.43 ± 0.18	58.82 ± 0.18	51.30 ± 0.19	51.37 ± 0.19	51.36 ± 0.19	42.05 ± 0.19	30.89 ± 0.18
Vanilla ProtoNet [8]	73.67 ± 0.14	72.27 ± 0.15	70.55 ± 0.15	68.08 ± 0.16	64.93 ± 0.16	60.66 ± 0.17	55.28 ± 0.18	47.94 ± 0.19
RNNP [6]	73.35 ± 0.14	71.92 ± 0.15	70.16 ± 0.15	67.97 ± 0.16	64.90 ± 0.17	60.81 ± 0.17	55.34 ± 0.18	48.07 ± 0.19
Median	$\mathbf{73.69 \pm 0.14}$	$\mathbf{72.50 \pm 0.15}$	70.78 ± 0.15	68.47 ± 0.15	65.26 ± 0.16	61.07 ± 0.17	55.46 ± 0.18	47.92 ± 0.19
Absolute $T = 50.0$	73.56 ± 0.14	72.44 ± 0.15	70.82 ± 0.15	68.60 ± 0.16	65.48 ± 0.16	61.33 ± 0.17	55.62 ± 0.18	48.19 ± 0.19
Absolute $T = 25.0$	73.26 ± 0.14	72.14 ± 0.15	70.65 ± 0.15	68.57 ± 0.16	65.53 ± 0.17	61.29 ± 0.17	55.45 ± 0.18	47.89 ± 0.19
Absolute $T = 10.0$	71.10 ± 0.15	69.96 ± 0.15	68.48 ± 0.16	66.29 ± 0.17	63.36 ± 0.17	58.83 ± 0.18	52.58 ± 0.19	44.80 ± 0.20
Euclidean $T = 50.0$	73.62 ± 0.14	72.39 ± 0.15	70.59 ± 0.15	68.28 ± 0.16	65.21 ± 0.16	60.94 ± 0.17	55.37 ± 0.18	48.04 ± 0.19
Euclidean $T = 25.0$	73.58 ± 0.14	72.36 ± 0.15	70.70 ± 0.15	68.40 ± 0.16	65.33 ± 0.16	61.08 ± 0.17	55.15 ± 0.18	47.79 ± 0.19
Euclidean $T = 10.0$	73.19 ± 0.15	72.03 ± 0.15	70.48 ± 0.16	68.21 ± 0.16	65.00 ± 0.17	60.50 ± 0.18	54.51 ± 0.19	46.58 ± 0.20
Cosine $T = 5.0$	73.57 ± 0.14	72.25 ± 0.15	70.44 ± 0.15	67.97 ± 0.16	64.77 ± 0.16	60.61 ± 0.17	55.14 ± 0.18	47.94 ± 0.19
Cosine $T = 2.0$	73.63 ± 0.14	72.28 ± 0.14	70.47 ± 0.15	68.10 ± 0.16	64.79 ± 0.16	60.60 ± 0.17	55.02 ± 0.18	48.03 ± 0.19
Cosine $T = 1.0$	73.46 ± 0.14	72.19 ± 0.15	70.33 ± 0.15	67.97 ± 0.16	64.80 ± 0.16	60.59 ± 0.17	55.09 ± 0.18	47.88 ± 0.19
Cosine $T = 0.5$	73.64 ± 0.14	72.30 ± 0.15	70.53 ± 0.15	68.13 ± 0.16	65.07 ± 0.16	60.73 ± 0.17	55.16 ± 0.18	48.13 ± 0.19
Cosine $T = 0.2$	73.55 ± 0.14	72.40 ± 0.15	70.61 ± 0.15	68.37 ± 0.16	65.26 ± 0.16	61.00 ± 0.17	55.34 ± 0.18	47.97 ± 0.19
TraNFS-2	72.43 ± 0.15	71.54 ± 0.16	70.24 ± 0.16	68.56 ± 0.17	65.93 ± 0.18	62.21 ± 0.20	56.98 ± 0.21	49.41 ± 0.22
TraNFS-3	72.91 ± 0.15	72.12 ± 0.15	$\mathbf{70.92 \pm 0.16}$	$\mathbf{69.47 \pm 0.16}$	$\mathbf{67.14 \pm 0.17}$	$\mathbf{63.60 \pm 0.19}$	$\mathbf{58.68 \pm 0.20}$	$\mathbf{50.66 \pm 0.22}$

sample with embeddings of each of the support set samples, with the prediction being a linear combination of the support set labels based on the result of this attention. While this mechanism is trainable in a meta-learning setup, we found that we achieved better results than those reported in the literature by using a frozen convolutional feature extractor trained with the ProtoNet loss.

MAML [3]. Model-Agnostic Meta-Learning (MAML) seeks to learn a good initialization so that the model can be quickly adapted to new tasks, with this initialization learned through second-order gradients. As such, unlike the other methods we compare against, we do not use the weights of the same frozen 4-layer convolutional feature extractor for MAML. Instead, we use the Adam optimizer to train MAML with a meta-learning rate of 3×10^{-3} and inner loop learning rate of 1×10^{-2}, using 5 adaptation steps during meta-training and 10 steps during meta-test. We use the same random horizontal flips, resized crops, and color jitters for data augmentations as the rest of our experiments.

ProtoNet [8]. ProtoNet (Fig. 2d) was introduced in Sec. 3 of the main paper. We refer to the version of ProtoNet proposed by Snell *et al*. in [8] (using the mean of the support embeddings) as *Vanilla ProtoNet* to distinguish it from the median and similarity weighted variants of ProtoNet that we propose in Sec. 4 of the main paper.

Baseline++ [1]. Baseline++ was proposed as a simple alternative to recent few-shot methods. Rather than requiring relatively complex bi-level meta-training, [1] proposed simply pre-training a feature extractor with a standard supervised cross-entropy loss, freezing the feature extractor's weights, and then fine-tuning a one-layer classifier just on top of the few examples in the novel class's support set features. In particular, the Baseline++ method uses cosine similarity and a softmax for the classifier. Such an approach has been shown to be surprisingly competitive with popular few-shot approaches. We implement this cosine similarity classifier in our framework, with the primary difference being that we use a feature extractor trained with the ProtoNet loss instead of a cross-entropy loss, in order to compare the classifier design on even terms. Note that [1] also proposed a simpler approach using a standard linear layer instead of cosine distance, which they referred to as Baseline; other than the training objective of the fixed feature extractor, the Baseline method is equivalent to our Linear Classifier baseline.

NegMargin [4]. Taking insights from the metric learning literature, [4] suggests that discriminability shortcomings of the softmax loss can be mitigated by learning with a margin. Surprisingly, NegMargin found that positive margins underperform in open-set few-shot classification scenarios, while negative margins can lead to significant improvements in performance due to improved transferability. To perform few-shot classification, NegMargin takes a similar approach to [1]–first pre-training and then freezing the feature extractor, followed by fine-tuning of a classifier for the novel support set–with the primary difference being the substitution of the standard softmax with the negative margin softmax loss during pre-training. As such, unlike the other methods

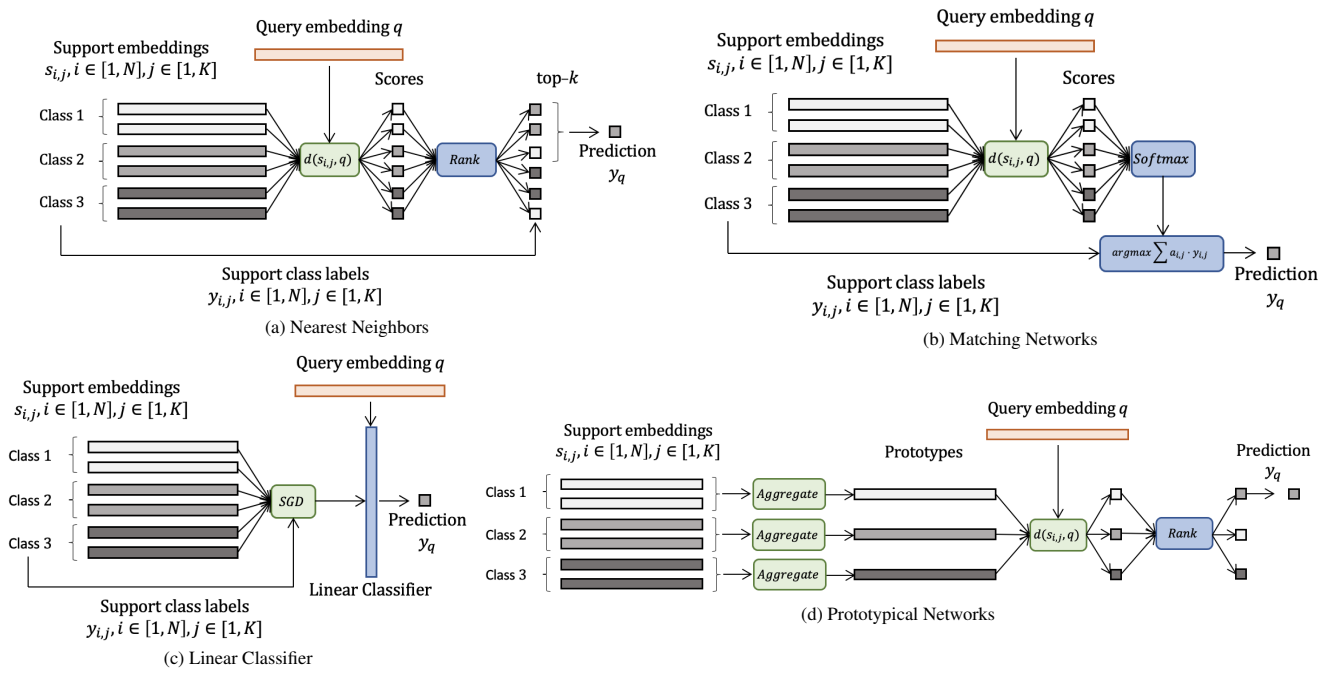


Figure 2. Visual overview of several of the few-shot method archetypes considered.

Table 6. **Temperature sweep for our ProtoNet variants: symmetric label swap noise**. 5-way 5-shot Acc. ± 95% CI on MiniImageNet [10], TieredImageNet [7]. Best viewed in color.

Model \ Noise Proportion	0%		20%		40%		60%	
Absolute $T = 50.0$	68.18 ± 0.16	71.24 ± 0.18	62.98 ± 0.17	66.56 ± 0.20	51.68 ± 0.19	54.97 ± 0.21	39.24 ± 0.20	41.59 ± 0.21
Absolute $T = 25.0$	68.24 ± 0.16	71.27 ± 0.18	**63.46 ± 0.17**	66.87 ± 0.20	52.06 ± 0.20	55.26 ± 0.22	39.78 ± 0.20	42.54 ± 0.22
Absolute $T = 10.0$	67.15 ± 0.17	70.15 ± 0.19	62.96 ± 0.18	66.10 ± 0.20	52.08 ± 0.20	55.08 ± 0.23	**39.92 ± 0.21**	42.49 ± 0.23
Absolute $T = 5.0$	63.89 ± 0.17	66.56 ± 0.19	59.63 ± 0.18	62.67 ± 0.21	51.30 ± 0.20	53.83 ± 0.22	37.99 ± 0.21	39.91 ± 0.23
Absolute $T = 1.0$	50.26 ± 0.20	51.39 ± 0.22	47.04 ± 0.20	48.40 ± 0.23	40.40 ± 0.21	41.45 ± 0.23	31.03 ± 0.20	31.75 ± 0.21
Euclidean $T = 50.0$	68.31 ± 0.16	71.31 ± 0.18	62.78 ± 0.17	66.36 ± 0.19	51.86 ± 0.19	55.19 ± 0.21	38.90 ± 0.20	41.19 ± 0.21
Euclidean $T = 25.0$	68.32 ± 0.16	**71.48 ± 0.18**	63.02 ± 0.17	66.69 ± 0.19	52.09 ± 0.19	55.62 ± 0.21	39.33 ± 0.20	41.75 ± 0.21
Euclidean $T = 10.0$	68.23 ± 0.16	71.18 ± 0.19	**63.46 ± 0.17**	**67.04 ± 0.20**	52.24 ± 0.20	55.78 ± 0.22	39.87 ± 0.20	42.53 ± 0.22
Euclidean $T = 5.0$	67.53 ± 0.16	70.54 ± 0.18	63.00 ± 0.18	66.56 ± 0.20	**53.79 ± 0.20**	**57.37 ± 0.22**	39.63 ± 0.21	42.31 ± 0.22
Euclidean $T = 1.0$	56.75 ± 0.19	59.17 ± 0.21	52.31 ± 0.19	54.82 ± 0.22	44.06 ± 0.20	46.09 ± 0.23	32.88 ± 0.20	33.99 ± 0.21
Cosine $T = 10.0$	68.24 ± 0.16	71.27 ± 0.18	62.47 ± 0.17	66.16 ± 0.19	51.41 ± 0.19	54.96 ± 0.21	38.38 ± 0.19	40.74 ± 0.21
Cosine $T = 5.0$	68.31 ± 0.16	71.16 ± 0.18	62.51 ± 0.17	65.99 ± 0.20	51.51 ± 0.19	54.78 ± 0.21	38.55 ± 0.19	40.81 ± 0.21
Cosine $T = 2.0$	68.28 ± 0.16	71.22 ± 0.18	62.57 ± 0.17	66.24 ± 0.19	51.59 ± 0.19	55.06 ± 0.21	38.71 ± 0.19	40.99 ± 0.21
Cosine $T = 1.0$	68.21 ± 0.16	71.21 ± 0.18	62.70 ± 0.17	66.47 ± 0.19	51.72 ± 0.19	55.27 ± 0.21	38.92 ± 0.19	41.32 ± 0.21
Cosine $T = 0.5$	**68.42 ± 0.16**	71.31 ± 0.18	63.13 ± 0.18	66.81 ± 0.20	52.08 ± 0.19	55.60 ± 0.22	39.36 ± 0.20	42.14 ± 0.22
Cosine $T = 0.2$	68.20 ± 0.16	70.59 ± 0.18	**63.46 ± 0.17**	66.62 ± 0.20	52.42 ± 0.20	55.78 ± 0.22	39.90 ± 0.20	**42.56 ± 0.22**
Cosine $T = 0.1$	67.52 ± 0.16	69.30 ± 0.19	63.07 ± 0.18	65.25 ± 0.20	52.22 ± 0.20	54.24 ± 0.23	39.85 ± 0.21	41.79 ± 0.23

we compare against, we do not use the weights of the same frozen 4-layer convolutional feature extractor for NegMargin. We use the official NegMargin codebase,[1] modifying their code to inject artificial noisy labels into support sets during meta-test evaluation.

RNNP [6]. Robust Nearest Neighbor Prototype (RNNP) creates hybrid examples by interpolating between samples within each support set, somewhat similarly to mixup. Using ProtoNet prototypes of the original support embeddings as initialization for the class centers, k-means is then used to refine the prototypes in an unsupervised manner. We reproduce RNNP, using the suggested $K - 1$ hybrids per support sample and mixing ratio of 0.8 when producing hybrids.

[1] https://github.com/bl0/negative-margin.few-shot

E. Additional ablation studies

Feature extractor training objective. We consider the performance of few-shot learning methods within the context of support set noise primarily with a frozen feature

Table 7. **Temperature sweep for our ProtoNet variants: outlier noise**. 5-way 5-shot Acc. ± 95% CI on MiniImageNet [10], TieredImageNet [7]. Best viewed in color.

Model \ Noise Proportion	0%		20%		40%		60%	
Absolute $T = 50.0$	68.41 ± 0.16	71.42 ± 0.19	64.62 ± 0.17	67.96 ± 0.19	58.08 ± 0.19	61.68 ± 0.21	47.33 ± 0.20	50.71 ± 0.22
Absolute $T = 25.0$	68.13 ± 0.16	71.17 ± 0.18	64.69 ± 0.17	68.00 ± 0.19	58.30 ± 0.18	61.98 ± 0.21	**47.39 ± 0.20**	50.59 ± 0.22
Absolute $T = 10.0$	67.18 ± 0.16	70.10 ± 0.19	64.14 ± 0.17	67.29 ± 0.20	58.12 ± 0.19	61.65 ± 0.21	47.02 ± 0.21	49.68 ± 0.22
Absolute $T = 5.0$	63.97 ± 0.17	66.78 ± 0.19	60.96 ± 0.18	63.88 ± 0.20	55.24 ± 0.19	58.17 ± 0.21	44.28 ± 0.21	46.50 ± 0.22
Absolute $T = 1.0$	50.02 ± 0.19	51.77 ± 0.22	47.71 ± 0.20	49.01 ± 0.22	42.90 ± 0.20	44.45 ± 0.23	34.52 ± 0.20	35.08 ± 0.21
Euclidean $T = 50.0$	68.31 ± 0.16	71.14 ± 0.18	64.25 ± 0.17	67.53 ± 0.19	57.43 ± 0.18	60.95 ± 0.21	47.06 ± 0.20	50.34 ± 0.21
Euclidean $T = 25.0$	**68.51 ± 0.16**	71.28 ± 0.18	64.57 ± 0.17	67.89 ± 0.19	58.01 ± 0.18	61.61 ± 0.20	47.25 ± 0.20	50.49 ± 0.21
Euclidean $T = 10.0$	68.19 ± 0.16	71.20 ± 0.18	64.55 ± 0.17	68.02 ± 0.19	58.17 ± 0.19	62.00 ± 0.21	47.24 ± 0.20	50.86 ± 0.22
Euclidean $T = 5.0$	67.58 ± 0.16	70.45 ± 0.18	64.25 ± 0.17	67.55 ± 0.19	57.82 ± 0.19	61.69 ± 0.21	46.34 ± 0.20	50.21 ± 0.22
Euclidean $T = 1.0$	56.94 ± 0.18	59.04 ± 0.21	53.59 ± 0.19	55.57 ± 0.22	47.23 ± 0.20	49.63 ± 0.22	37.32 ± 0.20	39.37 ± 0.22
Cosine $T = 10.0$	68.41 ± 0.16	71.20 ± 0.18	64.19 ± 0.17	67.48 ± 0.19	57.33 ± 0.18	60.87 ± 0.21	47.02 ± 0.20	50.12 ± 0.21
Cosine $T = 5.0$	68.29 ± 0.16	71.28 ± 0.19	64.04 ± 0.17	67.46 ± 0.20	57.30 ± 0.18	61.10 ± 0.21	47.08 ± 0.20	50.27 ± 0.21
Cosine $T = 2.0$	68.29 ± 0.16	71.20 ± 0.18	64.13 ± 0.17	67.53 ± 0.19	57.39 ± 0.18	61.07 ± 0.20	46.97 ± 0.20	50.23 ± 0.21
Cosine $T = 1.0$	68.30 ± 0.16	**71.54 ± 0.18**	64.23 ± 0.17	68.07 ± 0.19	57.51 ± 0.18	61.82 ± 0.20	46.89 ± 0.20	50.82 ± 0.21
Cosine $T = 0.5$	68.35 ± 0.16	71.38 ± 0.18	64.51 ± 0.17	**68.16 ± 0.19**	57.97 ± 0.18	62.13 ± 0.20	47.28 ± 0.20	**51.14 ± 0.22**
Cosine $T = 0.2$	68.20 ± 0.16	70.79 ± 0.18	**64.78 ± 0.17**	67.94 ± 0.19	58.36 ± 0.18	**62.37 ± 0.21**	47.34 ± 0.20	51.12 ± 0.22
Cosine $T = 0.1$	67.82 ± 0.16	69.33 ± 0.19	64.49 ± 0.17	66.55 ± 0.20	**58.42 ± 0.19**	61.21 ± 0.21	46.90 ± 0.21	50.00 ± 0.22

Table 8. **Temperature sweep for our ProtoNet variants: paired label swap noise**. 5-way 5-shot Acc. ± 95% CI on MiniImageNet [10], TieredImageNet [7].

Model \ Noise Proportion	40%	
Absolute $T = 50.0$	48.64 ± 0.19	51.83 ± 0.21
Absolute $T = 25.0$	49.38 ± 0.20	52.40 ± 0.22
Absolute $T = 10.0$	49.56 ± 0.20	52.54 ± 0.23
Absolute $T = 5.0$	47.18 ± 0.21	49.42 ± 0.23
Absolute $T = 1.0$	37.85 ± 0.21	38.47 ± 0.23
Euclidean $T = 50.0$	48.43 ± 0.19	51.39 ± 0.21
Euclidean $T = 25.0$	48.67 ± 0.19	51.90 ± 0.21
Euclidean $T = 10.0$	49.37 ± 0.19	52.55 ± 0.22
Euclidean $T = 5.0$	**49.75 ± 0.20**	52.57 ± 0.22
Euclidean $T = 1.0$	41.30 ± 0.21	42.92 ± 0.22
Cosine $T = 10.0$	47.75 ± 0.19	50.95 ± 0.21
Cosine $T = 5.0$	48.03 ± 0.19	51.17 ± 0.21
Cosine $T = 2.0$	48.03 ± 0.19	51.19 ± 0.21
Cosine $T = 1.0$	48.53 ± 0.19	51.71 ± 0.21
Cosine $T = 0.5$	48.90 ± 0.19	52.14 ± 0.21
Cosine $T = 0.2$	49.40 ± 0.19	**52.72 ± 0.22**
Cosine $T = 0.1$	49.71 ± 0.20	51.96 ± 0.23

Table 9. **Ablation study: Clean Prototype Loss** for a 3-layer TraNFS trained on 5-way 5-shot MiniImageNet [10].

λ_c	0%	20%	40%	60%
0.0	63.77 ± 0.18	60.67 ± 0.19	53.14 ± 0.22	39.75 ± 0.23
0.1	65.68 ± 0.18	61.94 ± 0.19	53.45 ± 0.22	39.20 ± 0.24
0.5	68.11 ± 0.17	64.56 ± 0.18	56.47 ± 0.21	41.94 ± 0.24
1.0	**68.80 ± 0.16**	**65.10 ± 0.18**	**57.26 ± 0.21**	**42.82 ± 0.24**
5.0	68.53 ± 0.17	65.08 ± 0.18	56.65 ± 0.21	42.60 ± 0.24
10.0	68.76 ± 0.17	64.87 ± 0.18	56.76 ± 0.21	42.17 ± 0.24

Table 10. **Ablation study: Binary Classification Loss** for a 3-layer TraNFS trained on 5-way 5-shot MiniImageNet [10].

λ_b	0%	20%	40%	60%
0.0	68.74 ± 0.17	64.97 ± 0.18	56.29 ± 0.21	41.88 ± 0.23
0.1	68.73 ± 0.17	65.04 ± 0.18	56.57 ± 0.21	42.23 ± 0.24
0.5	68.53 ± 0.17	**65.08 ± 0.18**	56.65 ± 0.21	**42.60 ± 0.24**
1.0	68.74 ± 0.17	64.81 ± 0.18	56.44 ± 0.21	42.26 ± 0.24
5.0	**68.75 ± 0.17**	65.06 ± 0.18	**56.71 ± 0.21**	42.42 ± 0.24

extractor, as is common practice in many previous few-shot works [1, 4, 8]. This allows us to isolate our comparison to the method, as opposed to the learned features. Nonetheless, the learned features have an impact on model performance. We compare the performance of 4-layer convolutional neural networks feature extractors [10] pre-trained with the ProtoNet [8] and NegMargin [4] objectives, observing $\{69.66 \pm 0.16, 59.88 \pm 0.18, 47.53 \pm 0.18, 35.67 \pm 0.17\}$ on 5-way 5-shot MiniImageNet [10] with $\{0\%, 20\%, 40\%, 60\%\}$ symmetric label swap noise. As reported in the literature, NegMargin outperforms the ProtoNet pre-trained feature extractor when there is no support set noise during meta-test. On the other hand, NegMargin sees a steeper decline in performance with increasing noise levels. We thus focus on the ProtoNet pre-trained feature extractor for our primary experiments. We leave further investigation into this phenomenon and the performance of other feature extractor pre-training objectives on noisy few-shot learning to future work.

Proposed ProtoNet variants: Temperature settings. As explained in Sec. 4.2 of the main paper, the temperature T controls the diffuseness of the softmax for similarity weighted prototypes. The setting of T results in a trade-off between emphasizing more shots versus noise rejection capability and thus can have an impact on performance. We show performance of similarity weighted prototypes with absolute distance, squared euclidean distance, and cosine similarity measure on MiniImageNet and TieredImageNet at varying noise levels with symmetric label swap noise, paired label swap noise, and outlier noise in Tables 6, 8, and 7, respectively. Note that differences in scale of T for Absolute and Squared Euclidean distances versus cosine similarity is due to their scale: cosine similarity is within $[-1, 1]$, while the two distances depend on the feature dimensional-

Table 11. **Ablation study: choice of embedding for `CLS` tokens** for a 3-layer TraNFS trained on 5-way 5-shot MiniImageNet [10] with symmetric label swap noise.

`CLS` Token + `POS` Token	0%	20%	40%	60%
Prototype + Learnable	68.15 ± 0.16	64.68 ± 0.18	55.04 ± 0.21	41.12 ± 0.22
Learnable + Learnable	67.74 ± 0.17	64.28 ± 0.18	55.46 ± 0.22	41.42 ± 0.24
Random Constant + Random Constant	66.95 ± 0.17	63.34 ± 0.19	54.55 ± 0.22	40.87 ± 0.24
Random Constant + Learnable	$\mathbf{68.53 \pm 0.17}$	$\mathbf{65.08 \pm 0.18}$	$\mathbf{56.65 \pm 0.21}$	$\mathbf{42.60 \pm 0.24}$

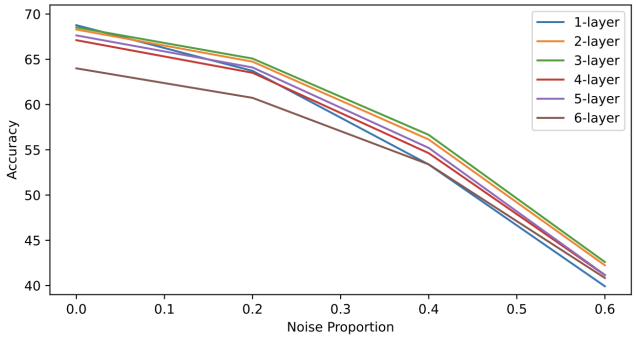


Figure 3. Sweep of number of transformer layers for 5-way 5-shot MiniImageNet [10] with symmetric label swap noise.

ity and scale.

TraNFS: Clean prototype loss. We run a hyperparameter sweep for the loss weight term λ_c, which controls the weight of the clean prototype loss (Eq. (15)). Results are reported in Table 9. We observe that the clean prototype loss is indeed helpful for encouraging the transformer to learn how to reject noisy samples, with a range of values of λ_c that work well.

TraNFS: Binary outlier detection. To test the effectiveness of the binary outlier classifier loss (Eq. (16)), we run a hyperparameter sweep for the loss weight term λ_b, reporting results in Table 10. We find that binary outlier classifier is indeed effective, with relatively low sensitivity to the setting of λ_b. Thus, we set λ_b to be 0.5 throughout our other experiments.

TraNFS: `CLS` and `POS` token embeddings. There are several options for the embeddings, corresponding to the `CLS` and `POS` tokens. In Table 11, we meta-train a 3-layer TraNFS model on 5-shot 5-way MiniImageNet with symmetric label swap noise. Each class's `CLS` token is set using one of three options: class prototypes averaged from the convolutional embeddings, a learnable parameter, and a random constant.

While we expected the ProtoNet-style prototypes to help kick-start the transformer's comparison mechanism, we were surprised to instead observe that they underperform other choices for the `CLS` embeddings. After visualizing the learning curves, we observe that using prototypes as the `CLS` embeddings results in a difficult-to-escape local minimum; we hypothesize this may be the model having minimal incentive to learn anything beyond the provided prototype. We also find that learnable `CLS` embeddings are not particularly effective: due to the random identity and shuffling of class orders between tasks, each `CLS` embedding lacks any semantic meaning beyond corresponding to a particular `POS` token's support samples; thus trying to learn some discriminitive value does not transfer between tasks and is ultimately unhelpful. As a result, it appears that a random constant value for each `CLS` token is sufficient for the transformer. For the `POS` positional encodings, however, learnable embeddings seem to work best.

TraNFS: Number of layers. Fig. 3 reports results for a sweep over the number of transformer layers in TraNFS. Matching intuition, we find that one layer is insufficient for surpassing the mean (ProtoNet) baseline. Different classes for each N-way episode mean the `CLS` embedding do not generalize across tasks. Without prior information of what each class c is in an episode, the transformer needs at least one layer to form such a concept for each position before comparisons can be made to identify samples that do not belong. Training with too many layers, however, seems to occasionally be unstable and tends to produce slightly inferior results, perhaps due to too much overparameterization and overfitting. We find two or three layers tend to perform best and thus report most of our results as such.

References

[1] 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.

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. *Conf. Comput. Vis. Pattern Recog.*, 2009.

[3] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *Int. Conf. Mach. Learning.*, 2017.

[4] Bin Liu, Yue Cao, Yutong Lin, Qi Li, Zheng Zhang, Mingsheng Long, and Han Hu. Negative margin matters: Understanding margin in few-shot classification. *Eur. Conf. Comput. Vis.*, 2020.

[5] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *Int. Conf. Learn. Represent.*, 2018.

[6] Pratik Mazumder, Pravendra Singh, and Vinay P Namboodiri. Rnnp: A robust few-shot learning approach. *Proc. Winter Conf. on Applications of Comput. Vis.*, 2021.

[7] 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. *Int. Conf. Learn. Represent.*, 2018.

[8] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Adv. Neural Inform. Process. Syst.*, 2017.

[9] Brendan Van Rooyen, Aditya Krishna Menon, and Robert C Williamson. Learning with symmetric label noise: The importance of being unhinged. *arXiv preprint arXiv:1505.07634*, 2015.

[10] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Adv. Neural Inform. Process. Syst.*, 2016.