TapNet: Neural Network Augmented with Task-Adaptive Projection for Few-Shot Learning: Supplementary Material

Sung Whan Yoon [1] **Jun Seo** [1] **Jaekyun Moon** [1]

1. Architecture Details

To obtain the results presented in the main paper, we employed ResNet-12, a residual network with 12 convolutional layers, as the feature extractor of TapNets. ResNet-12 is composed of four residual blocks and four max-pooling layers. Each residual block is constructed with three layers of 3×3 convolutions, followed by a batch normalization layer and an ReLU activation function. Each residual connection consists of a 3×3 convolutional layer and a batch normalization layer, and links the input to the last activation function. At the top of the residual block stack, global average pooling is also applied to reduce dimension. The four residual blocks have 64, 128, 256 and 512 respective channels. As mentioned in the main paper, this ResNet-12 is the same embedding network used in the task-dependent adaptive metric (TADAM) scheme.

2. Ablation Study

In this section, we show the results of ablation studies involving the following issues: composition of training episodes, learning rate optimization, regularization hyperparameters and reference vector settings. We used the Adam optimizer for all experiments. Also, $l2$ regularization and dropout are employed.

2.1. Episode Composition

Fig. 1 shows test accuracies for 5-way, 5-shot *mini*ImageNet with different numbers of training classes and query samples per class. We used the Adam optimizer with an initial learning rate of 10^{-3} and cut the learning rate by a factor of 10 every 4.0×10^4 episodes. $l2$ regularization with a weight decay rate of 5.0×10^{-4} is used and dropout with ratio 0.2 is applied to every output of the max-pooling layers. As summarized in Table 1, the best accuracy is obtained using either 20 training classes with 8 query samples per class or 25 training classes with 6 query samples per class. We simply chose the 20/8 combination for TapNet experiments.

Table 1. Best N_q for different numbers of training classes

Number of training classes	Best N_q	Accuracy
5	40	74.19%
10	14	75.73%
15	10	75.40%
20	**8**	**75.86%**
25	6	75.86%
30	4	75.09%

2.2. Hyperparameters in Experiment

Table 2 shows the hyperparameter values used for TapNet experiments. For 20-way Omniglot and 5-way *tiered*ImageNet experiments, we used step decays for the learning rate and dropout. In 5-way *mini*ImageNet experiments, $l2$ regularization is also applied in addition to dropout and learning rate decays. The dropout ratio bracket indicates the dropout ratio applied to each of the four pooling layer outputs.

2.3. Ablation Study for Reference Vectors

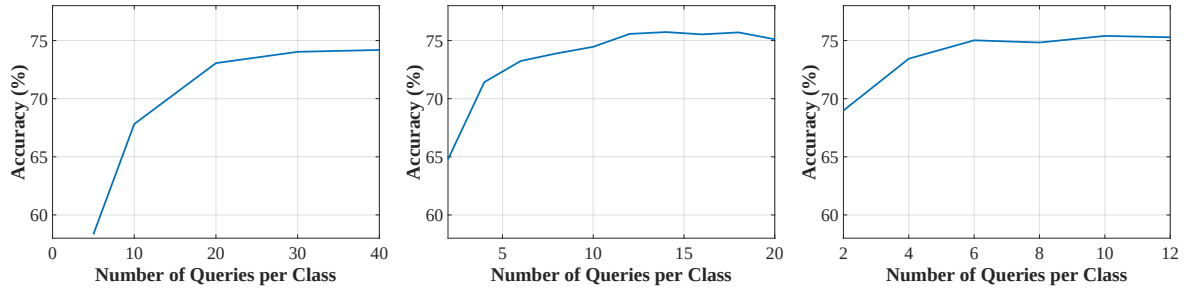
We test the 5-way, 5-shot *mini*ImageNet classification accuracy while varying the settings on the reference vectors $\mathbf{\Phi}$. In the main paper, we use the modified reference vectors $\tilde{\boldsymbol{\phi}}_k = \boldsymbol{\phi}_k - \frac{1}{N_c - 1} \sum_{l \neq k} \boldsymbol{\phi}_l$ when constructing the projection space, but use the original reference vectors $\boldsymbol{\phi}_k$ when classifying the query samples. Using the modified references in constructing projection space gives better separation among the classes in the projection space. By constructing projection space with the original reference vectors, TapNet achieves 75.53% accuracy, a degradation from 76.36%. Also, one can think of using the modified reference vectors in place of the original reference vectors during classification of the query samples. In this case, the classification accuracy also degrades to 74.61%.

For meta-training of TapNets, we adopted the higher way training. As a result, the number of prepared reference vectors is larger than the actual number of distinct classes

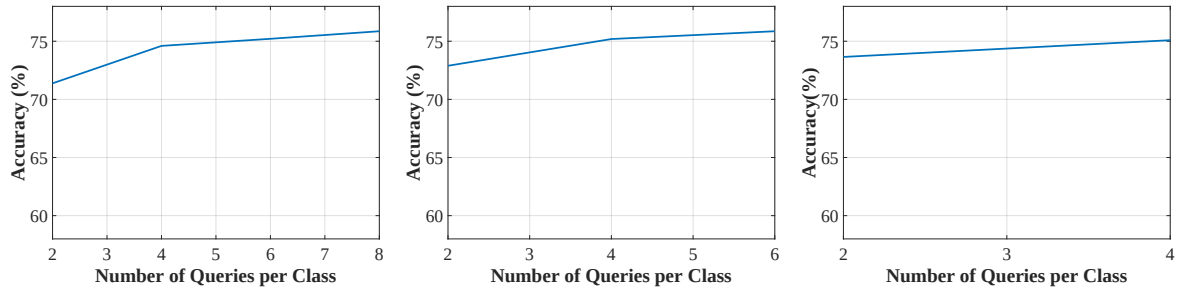
[1]School of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea. Correspondence to: Sung Whan Yoon <shyoon8@kaist.ac.kr>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

Figure 1. Accuracies with different episode compositions



(a) 5-way training (b) 10-way training (c) 15-way training



(d) 20-way training (e) 25-way training (f) 30-way training

Table 2. Hyperparameter settings for meta-training

Model	N_c	N_q	learning rate	lr decay step	lr decay ratio	$l2$ decay rate	dropout ratio
Omniglot 1-shot case	60	15	1e-3	40000	0.5	-	[0.2,0.2,0.2,0.2]
Omniglot 5-shot case	60	15	1e-3	40000	0.5	-	[0.2,0.2,0.2,0.2]
***mini*ImageNet** 1-shot case	20	12	1e-3	20000	0.1	5e-4	[0.2,0.2,0.2,0.2]
***mini*ImageNet** 5-shot case	20	8	1e-3	40000	0.1	5e-4	[0.3,0.2,0.2,0.2]
***tiered*ImageNet** 1-shot case	30	8	1e-3	40000	0.1	-	[0.2,0.2,0.2,0.2]
***tiered*ImageNet** 5-shot case	20	8	1e-3	30000	0.1	-	[0.2,0.2,0.2,0.2]

during evaluation. This leads to a study of the effect of particular reference vector selection. In the main paper, we selected those reference vectors closest to the class average vectors. We test two other possible reference selections here: one is simple random selection and the other is to select the farthest references to the class average. On the 5-way, 5-shot *mini*ImageNet classification task, random selection shows a slightly degraded accuracy of 75.78% while the farthest reference selection yields a bit more drop to 75.11%. In summary, although choosing the reference vectors closest to the class averages gives the best performance, any particular selection seems to make only a small difference.

In Section 4.3 of the main paper, we studied the learning trend of the reference $\mathbf{\Phi}$ based on minimum distance growth as well as visualization of the vector trajectories. As easily seen from the figures, the norm of each reference vector increased as training progressed. This naturally raises a question: would initializing the references with a larger norm be beneficial? The answer, however, turned out to be no; when the reference vectors were initialized to have a norm as large as the fully meta-trained reference vectors, the performance of TapNet actually dropped slightly to 75.78%.

3. Experimental Results for Varying Network Sizes

We now evaluate TapNets with varying embedding network sizes. The first option is the convolutional neural network (CNN) widely used in prior works such as Matching Networks or Prototypical Networks. It is based on four convolutional blocks, each of which consists of a 3×3 convolutional layer with 64 filters, stride 1 and padding along with a batch normalization layer, a ReLU activation and a 2×2 max-pooling. This CNN is denoted as "Conv4" in Table 3. The

Table 3. Few-shot classification accuracies for 5-way *mini*ImageNet

	5-way *mini*ImageNet	
Methods	1-shot	5-shot
Matching Networks	43.56 ± 0.84%	55.31 ± 0.73%
MAML	48.70 ± 1.84%	63.15 ± 0.91%
Prototypical Networks	49.42 ± 0.78%	68.20 ± 0.66%
TapNet (Ours, Conv4)	**50.68 ± 0.11%**	**69.00 ± 0.09%**
Relation Networks	50.44 ± 0.82 %	65.32 ± 0.70%
Transductive Propagation Nets	**55.51 ± 0.86%**	**69.86 ± 0.65%**
SNAIL	55.71 ± 0.99%	68.88 ± 0.92%
adaResNet	56.88 ± 0.62%	71.94 ± 0.57%
TapNet (Ours, ResNet-12-small)	**59.47 ± 0.12%**	**72.79 ± 0.10%**
TADAM-α	56.8 ± 0.3%	75.7 ± 0.2%
TADAM-TC	58.5 ± 0.3%	**76.7 ± 0.3%**
TapNet (Ours, ResNet-12)	**61.65 ± 0.15%**	**76.36 ± 0.10%**

Table 4. Few-shot classification accuracies for 5-way *tiered*ImageNet

	5-way *tiered*ImageNet	
Methods	1-shot	5-shot
MAML	51.67 ± 1.81%	70.30 ± 1.75%
Prototypical Nets	53.31 ± 0.89%	72.69 ± 0.74%
Relation Nets	54.48 ± 0.93%	71.31 ± 0.78%
Transductive Propagation Nets	**59.91 ± 0.94%**	73.30 ± 0.75%
TapNet (Ours, Conv4)	57.11 ± 0.12%	**73.66 ± 0.09%**

second option is a smaller version of ResNet-12, like the one used in SNAIL. With this version of ResNet-12, each residual block is constructed with three 3×3 convolutional layers. Also, a 1×1 convolutional layer is used for residual connection for each block. This network consists of four residual blocks with 64, 96, 128 and 256 respective channels. We denote this network "ResNet-12-small" in comparison to the larger version of ResNet-12 in the main paper. The Adam optimizer is also used for optimizing for both Conv4 and ResNet-12-small.

We focus on 5-way miniImageNet classification and the measured accuracy results are presented in Table 3. The few-shot learners are shown in four groups, depending on the required base network size. We notice significant performance differences in general as the model size/complexity changes. The first group uses the Conv4 network. Among the methods here, our TapNet achieves best 1-shot and 5-shot accuracies. The methods in the second group, Transductive Propagation Networks (TPN) and Relation Networks, are also based on the Conv4 embedder, but require additional networks for certain purposes. TPN utilizes an additional convolutional block in the graph construction network, and Relation Networks use additional convolutional blocks in its relation module. Both these methods require significant extra learning efforts, compared to the learners relying mostly on the Conv4 base embedder. TPN achieves higher 1-shot and 5 shot accuracies than the methods in the first group. The next group of methods uses ResNet-12-small. SNAIL and adaResNet are compared with TapNet. TapNet again achieves the best 1-shot and 5-shot performance in this group. The methods in the last group utilize ResNet-12, which is the largest network among the feature extractors considered in this work. For 1-shot results, our TapNet once again provides the best accuracy. For 5-shot, TapNet's result is comparable to that of the best method, TADAM-TC, in the sense that the confidence intervals overlap. In summary, given the same base network, TapNet consistently gives either the best accuracy or one comparable to the best in 5-way *mini*ImageNet classification among the well-known methods.

In Table 4, we display additional results on 5-way *tiered*ImageNet classification with Conv4 embedding. In the *tiered*ImageNet experiment, we had a small modification to the embedding network in TapNet. We added a 2×2 average pooling layer on top of the Conv4 network. Also, for 1-shot *tiered*ImageNet classification we found that it

Table 5. Number of parameters required for the learners

Method	Feature extractor	Additional Conv layer	Additional learnable parts
Matching Networks	112k	-	-
Prototypical Networks	112k	-	-
MAML	112k	-	12k (FC layer)
TapNet (Conv4)	112k	-	46k (Parameters of $\mathbf{\Phi}$)
Relation Nets	112k	111k	4k (FC layer)
Transductive Prop. Nets	112k	37k (Graph Construction)	-
SNAIL (ResNet-12-small)	2.2M	1.3M + α (Attention + TC)	-
adaResNet	2.2M	0.3M	-
TapNet (ResNet-12-small)	2.2M	-	5k (Parameters of $\mathbf{\Phi}$)
TADAM-α	9.4M	-	-
TADAM-TC	9.4M	-	1.2M (FC layer)
TapNet (ResNet-12)	9.4M	-	10k (Parameters of $\mathbf{\Phi}$)

was beneficial to use the higher-shot training strategy; we adopted 4-shot meta-training for 1-shot classification. As a result, TapNet achieves the best accuracy for 5-shot classification, and the second best accuracy for 1-shot among the methods using the same Conv4 embedding network.

4. Number of Network Parameters

It would be useful to understand the required complexity levels of the learning methods compared in this work. We in particular look at the number of network parameters used in each method. We focus on the number of parameters for the convolutional layer and other important learnable parts which are directly related to the learning efforts of the network. In particular, the other learnable parts include the fully connected (FC) layers in some cases and the stand-alone linear weights for the class reference vectors in TapNets. The Conv4 network requires 112,320 parameters in total. The residual networks rely on considerably larger numbers of learnable parameters. ResNet-12-small runs on 2.2 million parameters, while ResNet-12 requires 9.4 million parameters approximately. Table 5 includes the number of parameters necessary for implementing any additional convolutional layers or learnable parts for each method. In the case of SNAIL, we marked the number of additional parameters simply as α, since the numbers of parameters used for attention blocks and temporal convolution blocks there are hard to estimate due to the lack of available detail descriptions. We note that for the same number of parameters, the convolutional layer, which utilizes a sliding window to repeat many multiply/add operations, requires substantially higher computational complexity than the other types of learnable parts considered in Table 5.