An exploration of the intersection of few-shot learning and domain adaptation

Pimi Yvan, Eleni Triantafillou
ypimi@aimsammi.org, etriantafillou@google.com
African Institute for Mathematical Sciences

March 31, 2023

Abstract

Few-Shot Learning and Domain adaptation are two important fields of transfer learning, the first is used to learn with a few example while the second adapts the model to unseen distribution. Combining these fields could create a model that adapts quickly to an unseen target domain while using a few sample of data, so the accuracy of this model could be improved. it is in this perspective that we decided to combine a few-shot classification method (Prototypical network [Jak17]) and some adaptation techniques, we propose firstly a combination of Prototypical-Network and TENT a Fully test time adaptation model ([Jak17]), then we propose another combination of Prototypical Network and Visual domain bridge ([Mos20]), a source free domain adaptation and finally we fine-tune Prototypical Network. We were successfully able to improve the accuracy of prototypical network by more than 2 percent by trying these combinations.

Keywords : Prototypical Network, TENT, Visual Domain Bridge, Fine-tuning

1 Introduction

Few-shot learning and domain adaptation both fall under the common umbrella of transfer learning, where we want to transfer from a data-rich supervised source task or domain, to a different target task or domain. Specifically, in few-shot classification, we aim to transfer to new tasks that have previously-unseen classes, and offer only a few 'shots', or training examples, of each new class. On the other hand, domain adaptation assumes that the target task that we want to solve has the same set of classes as the source task, but the 'domain' is different, causing a covariate shift, e.g. between natural images and sketches of the same set of classes. Further, domain adaptation assumes access only to unlabeled data from the target domain. Source-free domain adaptation is a more realistic variant where, instead of receiving the source data, we assume a model pretrained on that data as the input, and we aim to adapt it to the target task via only unlabeled data.

Cross-domain few-shot classification is a harder variant of few-shot classification, where we aim to transfer to test tasks that, not only have just a few examples available, but also originate from different datasets/domains compared to the one(s) used for training. This challenging setting can be viewed as a hybrid between few-shot learning and domain adaptation, but to the additional domain shift between training and testing. Inspired by this observation, we set out to apply source-free domain adaptation methods for adapting the feature extractor in the context of cross-domain few-shot learning. More concretely, we investigate using a model that is pretrained on ImageNet and adapting it to various test tasks from Meta-Dataset.

Previous approaches to (cross-domain) few-shot learning can be divided into different categories, based on whether they adapt the feature extractor. Metric learning approaches like Matching Networks([Ori17]) and Prototypical Networks([Jak17]) keep the feature extractor frozen and make

[0]implemetation: https://github.com/PimiYvan/proto_adaptation

predictions within each task based on distances in the learned (and fixed) embedding space. On the other hand, fine-tuning is a common approach that adapts the feature extractor to the new task via gradient descent, using the cross-entropy loss on the small training set (the few available 'labeled 'shots'). Instead, our work aims to investigate adapting the feature extractor with unlabeled data, using a source-free domain adaptation approach such as TENT, Visual Domain Bridge. In doing so, we aim to investigate whether, in cross-domain few-shot settings, this type of unsupervised transfer learning can be beneficial over using labeled data, e.g. in preventing overfitting. in the same way, we investigate for adapting the model by using a supervised method such as fine-tuning, in order to check if a supervised method can also improved the adaptation.

1.1 Meta-Dataset

Meta-Dataset [Ele20] is a diverse collection of large datasets representing a benchmark for training and evaluating a model in a few-shot classification context. Evaluation can be done by using some few-shot learning algorithms such as : Prototypical-Networks, MAML, Proto-MAML, RelationNet, fo-Proto-MAML, fo-MAML, MatchingNet, k-NN baseline, Fine-tune baseline
Meta-Dataset contains some large datasets such as :

- Omniglot : datasets of some special characters.
- Aircraft : is an imagenet's dataset of Aircraft.
- Fungi : datasets of fungi images.
- Cu birds : imagenet's dataset of birds.
- DTD : is an evolving collection of textural images in the wild.
- VGG Flowers : is an imagenet's dataset of flower.
- Quickdraw : is a collection of drawings.
- Ilsvrc 2012 : an imagenet's dataset.
- MSCOCO, Traffic sign.

The standard evaluation protocol is to draw a number of k-shot N-way episodes from each dataset, where k denotes the number of 'shots' (i.e. the number of labeled examples per class that are available for training) and N the 'way' (i.e. the number of classes). Each episode also has a number of query examples, which are held-out examples from each class that are used for evaluation.

1.2 Batch Normalization

Batch Normalization is a technique for standardizing the input features of a layer in a neural network model. It is composed of two steps: First, the input is normalized, and then we apply an affine transformation of the normalization result.
let $S = \{(x_1, y_1) \dots (x_N, y_N)\}, x_i \in D^s$ our batch example from a source domain D^s and Θ a deep neural network. Batch Normalization at the layer l during training is given by :

$$BN(h_l^s) = \gamma^s \times \frac{h_l^s - \mu_l^s}{\sqrt{\sigma_l^{2^s} + \epsilon}} + \beta^s \tag{1}$$

h_l being the input feature of the layer l. $\mu_l^s, \sigma_l^{2^s}$ being respectively the mean and the variance of the input feature.

$$\mu_l^s = \frac{1}{NHW} \sum_i h_{l_i}^s \tag{2}$$

$$\sigma_l^s = \sqrt{\frac{1}{NHW} \sum_i (h_{l_i}^s - \mu_l^s)^2} \tag{3}$$

where H, W are the spacial dimension of the input feature h_l^s

During the test time, the batch normalization layer use the last estimated mean and variance computed during the training time. this estimated mean and variance can be defined by :

$$\mu_l^{est} = \mu_l^{batch} * \alpha + \mu_l^{est} * (1 - \alpha) \tag{4}$$

$$\sigma_l^{est} = \sigma_l^{batch} * \alpha + \sigma_l^{est} * (1 - \alpha) \tag{5}$$

where α is the momentum of the layer. then the batch norm during test time is equal to :

$$BN(h_l^s) = \gamma^s \times \frac{h_l^s - \mu_l^{est}}{\sqrt{\sigma_l^{2^{est}} + \epsilon}} + \beta^s \tag{6}$$

While batch normalization is a useful component for neural network architectures, it can be tricky to use in the context of domain adaptation, since the statistics of the source and target datasets are different. In this section, we review the standard usage of batch normalization and we later describe a different variation of a batch normalization layer, specifically designed for domain adaptation.

1.3 Prototypical Network

Each training task in a few-shot learning context contains a support set and a query set. for a N-way-K-shot classification, each task includes N classes with K examples of each class in the support set. let $S = \{(x_1, y_1) \dots (x_n, y_n)\}$ our labelled support set, $x_i \in R^D$ is the D-dimensional feature vector of an example and $y_i \in \{1, \dots, K\}$ is the corresponding label

Prototypical networks compute an M-dimensional representation $c_k \in R^M$ or prototype, of each class through an embedding function $f_\phi : R^D \to R^M$ with learnable parameters ϕ. Each prototype is the mean vector of the embedded support points belonging to its class [Jak17]:

$$c_k = \frac{1}{|S_k|} \sum_{(x_i, y_i) \in S_k} f_\phi(x_i) \tag{7}$$

where S_k denotes the set of examples labelled with class k.

Given a distance $d : R^M \times R^M \to [0, \infty)$ and a vector x, The model's probability of the query example x belonging to class k, is given by :

$$IP(y = k|x) = \frac{\exp(-d(f_\phi(x), c_k))}{\sum_{k'} \exp(-d(f_\phi(x), c'_k))} \tag{8}$$

1.3.1 Reinterpretation as a Linear Model

By considering the Euclidean distance $d(x, x') = \|x - x'\|^2$

$$-\|f_\phi(x) - c_k\|^2 = -f_\phi(x)^T f_\phi(x) + 2c_k^T f_\phi(x) - c_k^T c_k \tag{9}$$

$$-\|f_\phi(x) - c_k\|^2 = w_k^T f_\phi(x) + b_k \tag{10}$$

with $b_k = -c_k^T c_k, w_k = 2c_k$. So Prototypical Network can be written as a Linear model, this equation will be use-full for changing the top layer of our pretrained model.

1.4 TENT

Deep networks can achieve high accuracy on training and testing data from the same distribution, otherwise this accuracy can be low due to the distribution shift since we are testing on another distribution. To fix this issue we need to adapt this model with the new target distribution.

TENT or fully Test-time adaptation by entropy minimization minimizes the entropy of the model prediction $y = f(x)$ by updating statistics(μ, σ) and affine parameters(γ, β) of every batch norm layer of the model[Deq21].

TENT uses Shannon entropy $H(y) = p(y)log(p(y))$ for the entropy minimization, and has two main steps, the first is the normalization and the second is the affine transformation. normalization consists of changing the statistics of the batch norm layer l by using the input feature h_l^t from the target domain such as :

$$\mu_l^t = \frac{1}{NHW} \sum_i h_{l_i}^t \tag{11}$$

$$\sigma_l^t = \sqrt{\frac{1}{NHW} \sum_i (h_{l_i}^t - \mu_l^t)^2} \tag{12}$$

The affine transformation is the same as a batch normalization layer 1, but here the affine parameters(γ, β) are updating during back propagation by using the gradient of the entropy H such as:

$$\gamma = \gamma + \frac{\partial H}{\partial \gamma} \tag{13}$$

$$\beta = \beta + \frac{\partial H}{\partial \beta} \tag{14}$$

This minimization of the entropy of the model enable TENT to adapt the model to a new domain.

1.5 Visual Domain Bridge

Assume there are D^s and D^t respectively the source and the target domain, in a domain adaptation context during the training of the target domain, the Batch normalization is supposed to be :

$$BN(h_l^s) = \gamma^s \times \frac{h_l^t - \mu_l^s}{\sqrt{\sigma_l^{2^s} + \epsilon}} + \beta^s \tag{15}$$

h_l^t being the input feature of the target domain.
This results in an inconsistency between the input feature h_l^t and the statistical elements of the source domain captured inside the BN layers, this is seen as the internal state Mismatch. Visual Domain Bridge proposed a solution to fix this inconsistency by transferring the statistics of the target feature h_l^t (the mean μ_l^t and the variance σ_l^t) into the statistic of the source domain. Visual Domain Bridge is defined by the formula [Mos20]:

$$BN(h_l^{trans}) = \gamma^s \times \frac{h_l^{trans} - \hat{\mu}_l^{trans}}{\sqrt{\hat{\sigma}_l^{2^{trans}} + \epsilon}} + \beta^s \tag{16}$$

Where

$$h_l^{trans} = \sigma_l^{2^s} \times \frac{h_l^t - \hat{\mu}_l^t}{\sqrt{\hat{\sigma}_l^{2^t} + \epsilon}} + \mu_l^s \tag{17}$$

$$\hat{\mu}_l^t = \alpha \mu_l^t + (1 - \alpha) \mu_l^s \tag{18}$$

$$\hat{\sigma}_l^t = \alpha \sigma_l^t + (1 - \alpha) \sigma_l^s \tag{19}$$

α being the momentum factor of the batch norm layer l.

2 Our work

2.1 Combination of Prototypical Network and TENT

TENT minimize the entropy of a model by updating the statistics and the affine parameters of every batch normalization layer. it is useful for a test time adaptation of a model to a target domain.

In order to combine prototypical network and TENT, we decided to consider a model pretrained on ImageNet, since this dataset is known to transfer well to several other tasks, then we have to change the top layer(classifier) of this model to a linear model representation of a prototypical network by using the support sample and their labels 10.

After obtained prototypical network, we minimise the entropy of this model by using TENT, TENT will adapt the affine transformation parameters of each batch normalization layer. In addition, by setting batch normalization to 'training mode' during this phase, it uses the statistics from the (support set of the) target domain.

Knowing that we are adapting with a very small dataset (the support set is not large), we decided to repeat the adaptation process many times with the same support set, in order to minimize quickly the entropy of the model.

2.2 Combination of Prototypical Network and Visual Domain Bridge

Visual Domain Bridge is a technique for transferring the statistics of the target domain to the source domain. This is done by updating the batch normalization layer in order to take account of the statistics of the target feature (see Visual Domain Bridge).

In order to combine it with prototypical network, We first need to define a model where all the batch norm layer is done with Visual Domain Bridge formula(see Visual Domain Bridge), then we load a pretrained weight of a similar model into our new model, Finally we update the top layer of this last model by using the linear network formula of prototypical network 10.

Let us take an example with Resnet 18, we should create first a new model(Resnet vdb 18) similar to Resnet 18 but every batch norm layer should be replace by our VDB layer, then we should load the pretrained weight of Resnet 18 into our new model (Resnet vdb 18), finally we should update the top layer of this new model 10.

2.3 Combination of Prototypical Network and Fine-tuning

Fine-tuning refer to taking the weights of a trained neural network and use it as the starting values for training a new network. This experiment consists of fine-tuning prototypical network by using the support sample from an episode for training the new model.

3 Results

3.1 Combination of Prototypical Network and TENT

3.1.1 Complete experiment with a target domain

After initializing our combination of Prototypical network and TENT, we ran it on 200 episodes, and we got the results below.

- Dataset : Cu Birds
- Number of shot : 5
- Number of way : 8
- Number of query : 15

Tent Iteration / LR	0	10	25	50	100	200
10^{-1}	**68.908 ± 0.989**	**68.571 ± 1.0174**	**68.279 ± 1.164**	**68.154 ± 1.266**	**68.1166 ± 1.294**	**68.125 ± 1.290**
10^{-2}	**69.917 ± 1.008**	**69.888 ± 1.006**	**69.883 ± 1.007**	**69.875 ± 1.007**	**69.875 ± 1.007**	**69.875 ± 1.007**
10^{-3}	**68.683 ± 1.0241**	**68.6875 ± 1.0222**	**68.6875 ± 1.0222**	**68.6875 ± 1.0222**	**68.6875 ± 1.0222**	**68.6875 ± 1.0222**
10^{-4}	**69.025 ± 0.8751**	**69.025 ± 0.8751**	**69.0208 ± 0.8752**	**69.0208 ± 0.8752**	**69.0208 ± 0.8752**	**69.0208 ± 0.8752**
10^{-5}	**69.5375 ± 1.054**	**69.5375 ± 1.054**	**69.5375 ± 1.054**	**69.5375 ± 1.054**	**69.5375 ± 1.054**	**69.5416 ± 1.0543**

Table 1: Result on Cu Birds of the combination of Prototypical Network and TENT

As we can see, this combination is not improving prototypical network, the accuracy of this combination is similar to the accuracy of prototypical network.

3.1.2 Our Best result for the other datasets

Here we recap the result obtained with other target domain(fungi, dtd, aircraft, omniglot, ...)

Dataset	Parameter	Best LR	Prototypical Value	**Proto-Tent Best Value**
Flower	**shot : 5 Number of Way: 8 number of query : 15**	**LR :** 10^{-3} **TENT iteration: 25**	**85.0583 ± 0.6311**	**85.0666 ± 0.6309**
Flower	**shot : 10 Number of Way: 8 number of query : 15**	**LR :** 10^{-2} **TENT iteration: 10**	**90.879 ± 0.497**	**90.883 ± 0.499**
Omniglot	**shot : 5 Number of Way: 8 number of query : 15**	**LR :** 10^{-2} **TENT iteration: 10**	**86.975 ± 0.6947**	**86.983 ± 0.694**
Omniglot	**shot : 10 Number of Way: 8 number of query : 15**	**LR :** 10^{-4} **TENT iteration: 50**	**86.975 ± 0.6947**	**86.983 ± 0.694**
DTD	**shot : 10 Number of Way: 5 number of query : 10**	**LR :** 10^{-4} **TENT iteration: 100**	**66.91 ± 1.411**	**66.94 ± 1.410**
DTD	**shot : 15 Number of Way: 5 number of query : 10**	**LR :** 10^{-3} **TENT iteration: 100**	**74.35 ± 1.124**	**74.42 ± 1.118**
Cu Birds	**shot : 5 Number of Way: 8 number of query : 15**	**LR :** 10^{-5} **TENT iteration: 50**	**69.5375 ± 1.054**	**69.5416 ± 1.0543**
Cu Birds	**shot : 10 Number of Way: 8 number of query : 15**	**LR :** 10^{-4} **TENT iteration: 200**	**77.516 ± 0.869**	**77.5208 ± 0.866**

Table 2: Summary of experience for other datasets

This result show us that combine Prototypical network and TENT is not really improving prototypical network, because the result are similar to what we got for prototypical network.

3.2 Combination of Prototypical Network and Visual Domain Bridge

The results obtained after combining Prototypical Network and Visual Domain Bridge is presented in the table below:

Dataset	Parameter	Prototypical Value	Prototypical + VDB
Flower	**shot : 10 Number of Way: 5 number of query : 10**	**94.84 ± 0.54413**	**94.99 ± 0.50505**
Omniglot	**shot : 10 Number of Way: 5 number of query : 10**	**93.95 ± 0.5622**	**94.23 ± 0.625**
Aircraft	**shot : 10 Number of Way: 5 number of query : 10**	**61.3 ± 1.4201**	**63.39 ± 1.4248**
Cu Birds	**shot : 10 Number of Way: 5 number of query : 10**	**82.37 ± 1.086**	**84.96 ± 1.0537**
DTD	**shot : 10 Number of Way: 5 number of query : 10**	**67.37 ± 1.256**	**73.39 ± 1.236**
Fungi	**shot : 5 Number of Way: 5 number of query : None**	**64.822 ± 1.872**	**70.1046 ± 1.770**

Table 3: Summary of the experiment for all datasets

We can see that the result of the combination of Prototypical Network and Visual Domain Bridge are good, the difference between the accuracy of these 2 models can be greater than 2% for some target domain(DTD, Fungi, Aircraft). This could be explained by the fact that Visual Domain Bridge is combining source statistics and target statistics.

3.3 Fine-tuning of prototypical network

we summarise the result of this combination in the table below

Dataset	Parameter	Best LR	Prototypical Value	Our Best Value
DTD	**shot : 10 Number of Way: 5 base model : resnet 18**	**LR :** 10^{-4} **# epoch : 50**	**67.88 ± 1.3710**	**69.36 ± 1.3632**
Omniglot	**shot : 10 Number of Way: 5 base model : resnet 18**	**LR :** 10^{-3} **# epoch: 10**	**94.05 ± 0.6299**	**94.13 ± 0.6079**
Fungi	**shot : 10 Number of Way: 5 base model : resnet 18**	**LR :** 10^{-4} **# epoch: 25**	**63.2003 ± 1.7742**	**63.4291 ± 1.7582**
Aircraft	**shot : 10 Number of Way: 5 base model : resnet 18**	**LR :** 10^{-3} **# epoch: 10**	**60.53 ± 1.429**	**61.0 ± 1.4005**

Table 4: Result after combining Prototypical Network and Visual Domain Bridge

The result obtained after changing the base model from resnet-18 to resnet-18 with VDB layer instead of a simple batch norm is:

Dataset	Parameter	Best LR	**Prototypical Value**	Proto-VDB	**Finetuning Proto Value**
DTD	**shot : 15 Number of Way: 5 number of query : 10**	**LR :** 10^{-3} **num epoch: 10**	**75.14 ± 1.1033**	**77.92 ± 1.2142**	**78.65 ± 1.18671**
DTD	**shot : 10 Number of Way: 5 number of query : 10**	**LR :** 10^{-3} **num epoch: 10**	**66.3 ± 1.42305**	**72.18 ± 1.3477**	**72.24 ± 1.3475**
Omniglot	**shot : 15 Number of Way: 5 number of query : 5**	**LR :** 10^{-3} **num epoch: 50**	**94.76 ± 0.6684**	**93.28 ± 0.82554**	**93.36 ± 0.7945**
Omniglot	**shot : 10 Number of Way: 5 number of query : 10**	**LR :** 10^{-3} **num epoch: 25**	**94.0 ± 0.564670**	**94.34 ± 0.55375**	**94.41 ± 0.5412**
Omniglot	**shot : 5 Number of Way: 5 number of query : 10**	**LR :** 10^{-3} **num epoch: 25**	**91.65 ± 0.8504**	**93.19 ± 0.7297**	**93.24 ± 0.7336**
Omniglot	**shot : 2 Number of Way: 5 number of query : 10**	**LR :** 10^{-3} **num epoch: 5**	**83.63 ± 1.1435**	**89.06 ± 0.8964**	**89.12 ± 0.8924**
Omniglot	**shot : 1 Number of Way: 5 number of query : 10**	**LR :** 10^{-4} **num epoch: 5**	**72.06 ± 1.419**	**79.3 ± 1.4939**	**79.3 ± 1.4939**

Table 5: Result after fine-tuning the combination of Prototypical Network and VDB

The first table(4) show the result obtained when we fine-tune the prototypical network model, we see that the accuracy can be improved by more than 1 % depending of the target domain. On the second table(5) where VDB is used as the base model before fine-tuning, we can see that fine-tuning only work when the number of shot is high and can increase by up to 0.5% the accuracy of Proto-VDB (depending of the target domain), if the number of shot is low, it will have the same accuracy than the combination of Prototypical network and VDB.

4 Conclusion

Domain Adaptation and Few-shot learning are used respectively for adapting a model to a target domain and for training a model by using a few example of data. this works aims to investigate whether we can improve the accuracy of a few-shot model by combining it with some domain adaptation techniques. We try first to combine Prototypical network with TENT a fully test time adaptation technique which minimize the entropy of the model, the result obtained by this combination was closed to the accuracy of prototypical network, then we combine prototypical network and Visual Domain Bridge a source free domain adaptation technique which transfer the statistic of the target domain to the statistic of the source domain, the result obtained from this combination was very good, for some target domain(Fungi, DTD, Aircraft) it improved the accuracy of prototypical network model by up to 2%. after we fine-tune the prototypical network model with the support sample data, we succeed to improve the accuracy of prototypical model by up to 1% for some target domain. and finally we fine-tune the combination of Visual Domain Bridge and prototypical network and we have succeeded in improving the accuracy of the combination of prototypical network and Visual Domain Bridge depending of the number of shot 4.

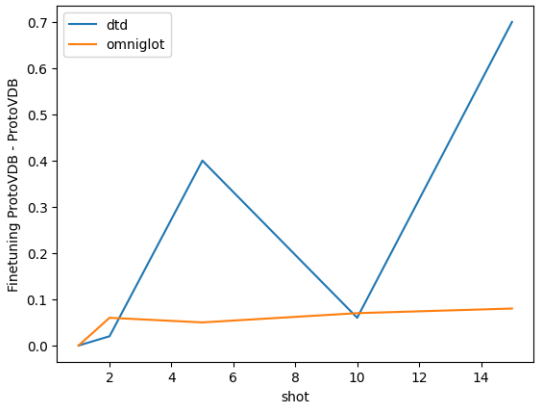

Figure 1: Difference between the accuracy of Finetuning proto-VDB and Proto-VDB in function of the number of shot

References

[Deq21] Shaoteng Liu] Dequan Wang Evan Shelhamer. "TENT, FULLY TEST-TIME ADAPTATION BY ENTROPY MINIMIZATION". In: *arXiv* 15.3 (2021), p. 15. DOI: https://arxiv.org/pdf/2006.10726.pdf.

[Ele20] Vincent Dumoulin Eleni Triantafillou Tyler Zhu. "META-DATASET, A DATASET OF DATASETS FOR LEARNING TO LEARN FROM FEW EXAMPLES". In: *arXiv* 24.3 (2020), p. 24. DOI: https://arxiv.org/pdf/2006.10726.pdf.

[Jak17] Richard S. Zemel Jake Snell Kevin Swersky. "Prototypical Networks for Few-shot Learning". In: *arXiv* 13.2 (2017), p. 13. DOI: https://arxiv.org/pdf/1703.05175.pdf.

[Mos20] Parham Moradi Moslem Yazdanpanah. "Visual Domain Bridge: A source-free domain adaptation for cross-domain few-shot learning". In: *Open Access* 13 (2020), p. 4. DOI: https://openaccess.thecvf.com/content/CVPR2022W/FaDE-TCV/papers/Yazdanpanah_Visual_Domain_Bridge_A_Source-Free_Domain_Adaptation_for_Cross-Domain_Few-Shot_CVPRW_2022_paper.pdf.

[Ori17] Timothy Lillicrap Oriol Vinyals Charles Blundell. "Matching Networks for One Shot Learning". In: *arXiv* 12.2 (2017), p. 12. DOI: https://arxiv.org/pdf/1606.04080.pdf.