Paper Note

"Model-Agnostic Boundary-Adversarial Sampling for Test-Time Generalization in Few-Shot learning"

Lisen Dai

Oct 2020

1 General Discription of Few-shot

Paper link:ECCV 2020

1.1 Few-shot problem

Few-shot learning is a general name for tasks in which people hope to train a machine learning model with just very small amount of data.

1.2 Previous Research

1. **Data augmentation**. Generating fake labaled data.
2. Distance metric methods.
3. Meta-learning methods.

1.3 Limitations of data augmentation few-shot

1. Such methods learn to generate additional examples with the meta-training set, and thus may not be effective if meta-test domains are far from the meta-training domain.
2. Since they do not update the trained parameters of the base classifier models at test time, they have no chance to correct the errors that exist in the base classifiers (e.g. overfitting of the embedding functions to the meta-training set).
3. These methods need to be re-trained for each base classifier to generate fake labeled data optimal for the base model.

2 Innovative Designs

A novel model-agnostic sample generation approach named **MABAS** (Model-Agnostic Boundary-Adversarial Sampling).

2.1 Test-time Fine-tuning of Embedding Functions

Since each meta-test task consists of the classes that are never seen during metatraining, it is a common approach in few-shot learning to fine-tuning the learned parameters using the support samples of the novel task. This work aims at fine-tuning the learned parameters of the base few-shot learner adaptively to novel tasks but limited to update the parameters of the embedding function (or the feature extractor).

2.2 Fine-tuning by Boundary-Adversatial Samples

Once we have the adversatial samples, we can update the parameter ϕ of embedding function via gradient descent. The fine-tuning loss $\mathcal{L}^{fine-tune}$ is defined as:

$$\mathcal{L}^{fine-tune}(\mathcal{S},\phi) := \mathcal{L}^{adv}(\mathcal{S},\phi) + \eta \cdot \frac{1}{\mathcal{S}} \sum_{(\mathbf{X},y)} \in \mathcal{S} \| f_\phi(\mathbf{x}) Vert^2,$$

and,

$$\mathcal{L}^{adv}(\mathcal{S},\phi) = \frac{1}{\mathcal{S}} \sum_{(} (\mathbf{x},y) \in \mathcal{S})[\frac{1}{K-1} \sum_{k' \neq y} \{\alpha_{\mathbf{X},y} - \min m(\mathbf{z}^{adv}_{y,k'}, y, k|\psi,\mathcal{S})\}_+]$$

where, $\alpha_{\mathbf{X},y} = \frac{1}{K-1} \sum_{k \neq y} m(\mathbf{z}, y, k|\phi,\mathcal{S})$
feature: not only prevents the exces- sive expansion of the supports' embedding space but also stabilizes the updates of the embedding space.
Below is an illustration picture of the MABAS method.

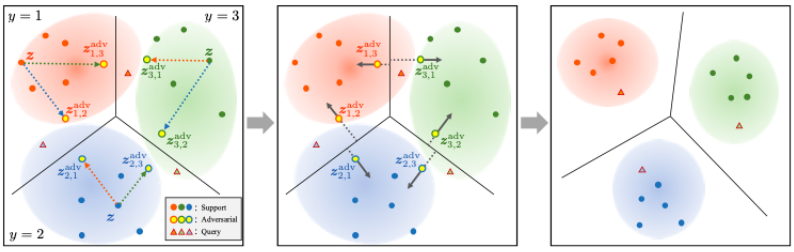


Fig. 1. Conceptual visualization of our MABAS approach in the embedding space. The solid circles, triangles, and yellow circles are the support, query, and boundary-adversarial samples, respectively. Each shaded area represents a region to which most samples from the class are mapped by the embedding function. The black lines are the classification boundaries computed from the supports. *Left*: The boundary-adversarial samples are generated by moving each support sample toward the classification boundaries (Equation (4)). *Middle*: The embedding function is updated in the direction that increases the margins of the adversarial samples (Equation (6)) while holding the support embeddings (Equation (5)). *Right*: After the fine-tuning of the embedding function, data embeddings are denser around the support embeddings, and the classification boundaries are updated to better separate queries from different classes.

3 Other Applications of Few-shot

To show the flexibility and generality of our MABAS approach, this paper applies it to three representative few-shot learning methods:
1. MetaOptNet
2. Few-Shot without Forgetting
3. Standard Transfer Learning

In each method, the paper give out the original equation of implementation and then the three ways below about how to optimize it by MABAS:
1. Boundary-adversarial fine-tuning.
2. Variation.
3. Setting of δ

And here's an example of *MetaOptNet*:

Original methodology. The Few-Shot without Forgetting (FSwF) [16] learns not only the embedding function f_ϕ but also the attention-based classification weight generator. The role of the weight generator G with parameter θ is to compute the classification weight vector $\boldsymbol{w}_k$ for the novel class k using two types of input: (i) the classification weights for the B base classes (*i.e.* $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_B$) trained from $\mathcal{D}^{\text{train}}$ and (ii) the support of novel class k (*i.e.* $S_k = \{(\boldsymbol{x}, y) | y = k, \forall (\boldsymbol{x}, y) \in \mathcal{S}\}$) in each few-shot task:

$$G_\theta(\mathcal{S}, k, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_B) = \theta^{avg} \odot \boldsymbol{w}_k^{avg} + \theta^{att} \odot \boldsymbol{w}_k^{att}, \tag{11}$$

$$\boldsymbol{w}_k^{avg} = \frac{1}{|S_k|} \sum_{(\boldsymbol{x}, y) \in S_k} \frac{f_\phi(\boldsymbol{x})}{\|f_\phi(\boldsymbol{x})\|}, \tag{12}$$

$$\boldsymbol{w}_k^{att} = \frac{1}{|S_k|} \sum_{(\boldsymbol{x}, y) \in S_k} \sum_{b=1}^{B} \text{Att}(\theta^{att} \frac{f_\phi(\boldsymbol{x})}{\|f_\phi(\boldsymbol{x})\|}, \boldsymbol{\theta}_b^{key}) \cdot \frac{\boldsymbol{v}_b}{\|\boldsymbol{v}_b\|}, \tag{13}$$

where $\odot$ is the element-wise product and Att() is the attention kernel. The learnable parameters include ϕ and $\theta = \{\theta^{avg}, \theta^{att}, \boldsymbol{\theta}_1^{key}, \ldots, \boldsymbol{\theta}_B^{key}\}$, where θ is trained on meta-training tasks and not updated in the meta-test phase. Finally, the novel class weight vector is $\boldsymbol{w}_k = G_\theta(\mathcal{S}, k, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_B)$.

The score function of FSwF for the task becomes

$$h\left(f_\phi(\boldsymbol{x}), k | \psi, \mathcal{S}\right) \coloneqq \frac{\boldsymbol{w}_k^\top f_\phi(\boldsymbol{x})}{\|\boldsymbol{w}_k\| \|f_\phi(\boldsymbol{x})\|}. \tag{14}$$

Boundary-adversarial fine-tuning. By applying the definition of h from Equation (14) to Equations (4) and (6), $\boldsymbol{z}_{y,k'}^{\text{adv}}$ for FSwF is defined by

$$\begin{aligned} \boldsymbol{z}_{y,k'}^{\text{adv}} &= \boldsymbol{z} - \delta \cdot \nabla_{\boldsymbol{z}} \left(\left(\frac{\boldsymbol{w}_y}{\|\boldsymbol{w}_y\|} - \frac{\boldsymbol{w}_{k'}}{\|\boldsymbol{w}_{k'}\|} \right)^\top \frac{\boldsymbol{z}}{\|\boldsymbol{z}\|} \right) \\ &= \boldsymbol{z} - \delta \cdot \left(\frac{I_d}{\|\boldsymbol{z}\|} - \frac{\boldsymbol{z}\boldsymbol{z}^\top}{\|\boldsymbol{z}\|^3} \right) \left(\frac{\boldsymbol{w}_y}{\|\boldsymbol{w}_y\|} - \frac{\boldsymbol{w}_{k'}}{\|\boldsymbol{w}_{k'}\|} \right) \end{aligned} \tag{15}$$

where $\boldsymbol{z} \in \mathbb{R}^d$, and the adversarial loss $\mathcal{L}^{\text{adv}}$ becomes

$$\begin{aligned} \mathcal{L}^{\text{adv}}(\mathcal{S}, \phi) = \frac{1}{|\mathcal{S}|} \sum_{(\boldsymbol{x}, y) \in \mathcal{S}} \Bigg[\frac{1}{K-1} \sum_{k' \neq y} \\ \left\{ \alpha_{\boldsymbol{x}, y} - \min_{k \neq y} \left(\frac{\boldsymbol{w}_y}{\|\boldsymbol{w}_y\|} - \frac{\boldsymbol{w}_k}{\|\boldsymbol{w}_k\|} \right)^\top \frac{\boldsymbol{z}_{y,k'}^{\text{adv}}}{\|\boldsymbol{z}_{y,k'}^{\text{adv}}\|} \right\}_+ \Bigg]. \end{aligned} \tag{16}$$

Similarly to the derivation for MetaOptNet in Section 4.1, the adversarial gradient in Equation (15) is derived while fixing the classification weights of $\boldsymbol{w}_y$ and $\boldsymbol{w}_{k'}$ with respect to $\boldsymbol{z}$.

4 Experiment

4.1 Setup

4.1.1 Dataset

1. miniImageNet: consists of 100 classes each of which has 600 images with a size of $84 \times 84 \times 3$. It adopts the same class split used by [36, 22]: 64, 16 and

20 classes for training, validation and the test, respectively.
2. CIFAR-FS: splits all of the classes in CIFAR100 [20] into 64 training, 16 validation and 20 test sets, respectively.
3. FC100: is another CIFAR100-based dataset. Classes are split into 60, 20 and 20 for training, validation and test, respectively. This class split is designed to minimize the overlap of information between all three subsets, to be more challenging than CIFAR-FS for few-shot learning.

4.1.2 Embedding functions

ResNet-12, but for STL and MetaOptNet, different average pooling methods are applied in the last residual blocks.

4.1.3 Meta-training and meta-validation phase

For MetaOptNet- Proto, it uses a learning decay rate of 0.1 with a decay period of 15 epochs for simplicity. For the STL models, it trains for 100 epochs with a batch size of 256 and a learning rate of 0.001 using the cosine annealing decay.

4.1.4 Meta-test phase

5-way 5-shot and 5-way 1-shot classification tasks. Each meta-test run consists of 600 tasks sampled from D^{test}, and a single task contains 15 query samples per each of the 5 classes.

4.1.5 Boundary-adversarial fine-tuning

In all experiments, they update the embedding function for 150 steps and use the step-based learning rate decay with a decay rate of 0.8 and a period of 5.

4.2 Result

Table 1. Meta-test accuracies (%) of the four base methods before and after applying our MABAS method with the 95% confidence interval. We also report the accuracy gains (%p) by MABAS. We obtain all results using the source codes provided by the original authors to precisely measure the accuracy improvements by our method.

Method	miniImageNet, 5-way		CIFAR-FS, 5-way		FC100, 5-way	
	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
STL [6, 42]	55.98 ± 0.78	76.19 ± 0.60	64.32 ± 0.92	83.62 ± 0.61	38.83 ± 0.70	54.54 ± 0.72
+ **Ours**	60.19 ± 0.79	79.34 ± 0.57	67.41 ± 0.91	84.29 ± 0.65	40.76 ± 0.68	58.16 ± 0.78
Gain (%p)	4.21	3.15	3.09	0.67	1.93	3.62
FSwF [16]	55.64 ± 0.82	69.94 ± 0.68	69.23 ± 0.90	82.52 ± 0.68	37.91 ± 0.77	49.75 ± 0.72
+ **Ours**	60.45 ± 0.82	78.28 ± 0.61	70.71 ± 0.89	85.25 ± 0.65	40.63 ± 0.75	54.95 ± 0.75
Gain (%p)	4.81	8.34	1.48	2.73	2.72	5.20
MetaOptNet [22]	62.25 ± 0.82	78.55 ± 0.58	72.11 ± 0.96	84.32 ± 0.65	40.15 ± 0.71	54.92 ± 0.75
+ **Ours**	64.21 ± 0.82	81.01 ± 0.57	73.24 ± 0.95	85.65 ± 0.65	41.74 ± 0.73	57.11 ± 0.75
Gain (%p)	1.96	2.46	1.13	1.33	1.59	2.19
MetaOptNet-Proto	61.68 ± 0.85	78.36 ± 0.59	72.46 ± 0.91	84.02 ± 0.67	40.60 ± 0.75	55.04 ± 0.75
+ **Ours**	65.08 ± 0.86	82.70 ± 0.54	73.51 ± 0.92	85.49 ± 0.68	42.31 ± 0.75	57.56 ± 0.78
Gain (%p)	3.40	4.34	1.05	1.47	1.71	2.52

invariant to scaling of inputs, we set $\eta = 0$. Its initial learning rate is 0.0003 and the adversarial step size is $\delta = 10$. In the STL experiments, we perform the fine-tuning with an initial learning rate of 0.0001, $\eta = 0.01$ and $\delta = 10$.

Table 2. Comparison with the state-of-the-art few-shot learning methods. We present the average meta-test accuracy with its 95% confidence interval. ‡ denotes the results from [36, 22, 42], while all the other baseline scores are referred to their original papers. The numbered superscripts denote the architecture of f_ϕ: [1]Conv-4, [2]Conv-4+MetaNet, [3]ResNet-12, [4]WRN-28-10, [5]Conv4+MetaGAN. Note that WRN-28-10 involves three times more parameters than ResNet-12.

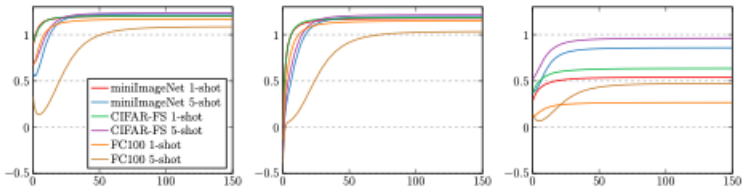
Method	miniImageNet, 5-way		CIFAR-FS, 5-way		FC100, 5-way	
	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
Meta-LSTM[1] [36]	43.44 ± 0.77	60.60 ± 0.71	-	-	-	-
MatchingNets[1]‡ [45]	43.56 ± 0.84	55.31 ± 0.73	-	-	-	-
MAML[1‡] [12]	48.70 ± 1.84	63.11 ± 0.92	58.9 ± 1.9	71.5 ± 1.0	38.1 ± 1.7	50.4 ± 1.0
ProtoNets[1‡] [40]	49.42 ± 0.78	68.20 ± 0.66	55.5 ± 0.7	72.0 ± 0.6	35.3 ± 0.6	48.6 ± 0.6
RelationNets[1‡] [43]	50.44 ± 0.82	65.32 ± 0.70	55.0 ± 1.0	69.3 ± 0.8	-	-
R2D2[1] [3]	51.20 ± 0.60	68.80 ± 0.10	65.30 ± 0.20	79.40 ± 0.10	-	-
FSwF[4] [16]	56.20 ± 0.86	73.00 ± 0.64	-	-	-	-
FSwF[3] [16]	55.45 ± 0.89	70.13 ± 0.68	-	-	-	-
Bilevel Program[3] [13]	50.54 ± 0.85	64.53 ± 0.68	-	-	-	-
MetaGAN[2] [52]	52.71 ± 0.64	68.63 ± 0.67	-	-	-	-
SNAIL[3] [29]	55.71 ± 0.99	68.88 ± 0.92	-	-	-	-
AdaResNet[3] [33]	56.88 ± 0.62	71.94 ± 0.57	-	-	-	-
TADAM[3] [34]	58.5 ± 0.3	76.7 ± 0.3	-	-	40.1 ± 0.4	56.1 ± 0.4
ProtoNets[3‡] [40]	59.25 ± 0.64	75.60 ± 0.48	72.2 ± 0.7	83.5 ± 0.5	37.5 ± 0.6	52.5 ± 0.6
MTL[3] [42]	61.2 ± 1.8	75.5 ± 0.8	-	-	**45.1 ± 1.8**	57.6 ± 0.9
TapNet[3] [49]	61.65 ± 0.15	76.36 ± 0.10	-	-	-	-
MetaOptNet[3] [22]	62.64 ± 0.61	78.63 ± 0.46	72.0 ± 0.7	84.2 ± 0.5	41.1 ± 0.6	55.5 ± 0.6
LEO[4] [38]	61.76 ± 0.08	77.59 ± 0.12	-	-	-	-
CC+rot[3] [15]	62.93 ± 0.45	79.87 ± 0.33	**76.09 ± 0.30**	**87.83 ± 0.21**	-	-
$S2M2_R$[3] [26]	64.93 ± 0.18	**83.18 ± 0.11**	74.81 ± 0.19	87.47 ± 0.13	-	-
LGM-Net[2] [23]	**69.13 ± 0.35**	71.18 ± 0.68	-	-	-	-
STL + **Ours**[3]	60.19 ± 0.79	79.34 ± 0.57	67.41 ± 0.91	84.29 ± 0.65	40.76 ± 0.68	**58.16 ± 0.78**
MetaOptNet +**Ours**[3]	64.21 ± 0.82	81.01 ± 0.57	73.24 ± 0.95	**85.65 ± 0.65**	41.74 ± 0.73	57.11 ± 0.75
−Proto + **Ours**[3]	65.08 ± 0.86	**82.70 ± 0.54**	**73.51 ± 0.92**	85.49 ± 0.68	42.31 ± 0.75	57.56 ± 0.78

Table 3. Meta-test accuracy of FSwF with different embedding functions with or without MABAS. All ConvNets share the same hyperparameters.

f_ϕ	FSwF	+ MABAS
Conv32	70.01 ± 0.66	70.05 ± 0.65
Conv64	71.89 ± 0.67	72.15 ± 0.66
Conv128	72.59 ± 0.65	72.79 ± 0.63
ResNet-12	69.94 ± 0.68	78.28 ± 0.61

Table 4. Comparison of meta-test accuracy between naive fine-tuning and MABAS using FSwF on the miniImageNet dataset.

Method	miniImageNet, 5-way	
	1-shot	5-shot
FSwF	55.64 ± 0.82	69.94 ± 0.68
+ Naive FT	55.73 ± 0.82	70.14 ± 0.67
+ Ours	60.45 ± 0.82	78.28 ± 0.61



(a) Support (b) Adversarial samples (c) Query

Fig. 2. Average classification margin with FSwF + Ours for support (*left*), adversarial (*middle*) and query (*right*) samples on 5-way tasks. The *x-axis* and *y-axis* denote the number of fine-tuning updates and the average classification margin, respectively. The margin values are averaged over all the meta-test tasks.