Supplementary Materials for Meta Convolutional Neural Networks for Single Domain Generalization

Algorithm 1 Algorithm of BoMF

```
Input: input features $\boldsymbol{F} = \{\boldsymbol{f}_i\}_{i=0}^{B}$, meta features $\boldsymbol{M}$ initiated by gaussian noise
Output: learned meta features $\boldsymbol{M}$, and compositional outputs $\hat{\boldsymbol{F}} = \{\hat{\boldsymbol{f}}_i\}_{i=0}^{B}$
 1: repeat
 2:     Fetch a new batch of data $\boldsymbol{F}$
        Meta Feature Composition:
 3:     for i = 1,...,B do
 4:         Obtain local features $\{\boldsymbol{p}\}$ from $\boldsymbol{f}$ through local feature decomposition (Sec. 3.1).
 5:         Select a group of meta features $\boldsymbol{M}_p$ for each $\boldsymbol{p}$ based on local feature addressing (Sec. 3.2).
 6:         Compose $\boldsymbol{M}_p$ through GLM to obtain $\hat{\boldsymbol{p}}$ and fold $\{\hat{\boldsymbol{p}}\}$ into $\hat{\boldsymbol{f}}_{\boldsymbol{i}}$ (Sec. 3.3).
 7:     end for
        Meta Feature Learning:
 8:     Obtain local features $\{\boldsymbol{p}\}$ from the whole batch.
 9:     Estimate the coefficients based on Sec. 3.1, 3.2, 3.3.
10:     Update the meta features $\boldsymbol{M}$ through the back-propagation of the gradient based on Eq. 3
11: until The end of batch
```

1. Algorithm of BoMF

The algorithm of the proposed BoMF, including local feature decomposition, local feature addressing, meta feature composition and meta feature learning, is summarized in Alg. 1.

During the training process, the whole network is trained in an end-to-end manner. For the objective function, the parameters in BoMF are updated by both classification and reconstruction loss, other parameters are updated by classification loss only. All operations in BoMF are differentiable. Local Feature Decomposition is the img2col operation in convolution, implemented by *unfold* in Pytorch. Local Feature Addressing is a result of selection, gradients can be propagated back only to $\mathbf{M}_p$. Meta Feature Composition involves the col2img in convolution (*fold*), matrix multiplication (*mul*) and inversion (*inverse*). All are differentiable operations in Pytorch.

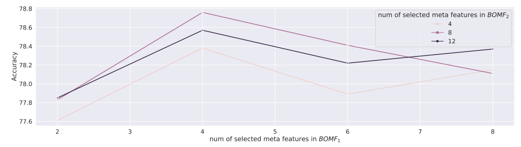
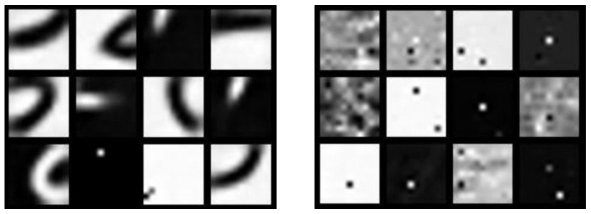


Figure 1. The effect of the number of selected meta features.



(a) w/ LRUA (b) w/o LRUA

Figure 2. Comparison of the learned meta features w & w/o LRUA. 16 meta features are randomly selected from the whole meta feature set. In (b), many meta features will receive zero gradients and be optimized to one points. These meta features are hardly selected during the composition process.

2. Implementation Details

Average number of selected meta features $|\bar{\boldsymbol{M}}_p|$. In the local feature addressing step, λ and k control the average number of selected meta features $|\bar{\boldsymbol{M}}_p|$. Considering both efficiency and effectiveness, λ is set relatively large, while k is small (generally less than 10). As a result, $|\bar{\boldsymbol{M}}_p|$ of two BoMF operations are controlled around 4 and 8, respectively. The effect of $|\bar{\boldsymbol{M}}_p|$ is shown in Fig 1, and $4/8$ are the best for $BoMF_1/BoMF_2$ within the experiments on Digits. Since patterns in digital images are simple, small $|\bar{\boldsymbol{M}}_p|$ is enough to handle. When facing more complicated tasks, it is suggested to set relative large $|\bar{\boldsymbol{M}}_p|$.

Bias of $\boldsymbol{\Gamma}$. In the local feature learning step, it is observed that only a small set of meta features are activated and updated, while others receive zero gradients. To alleviate this problem, we are inspired by the Least Recently Used Access [2] to introduce a bias for $\boldsymbol{\Gamma}$. The bias is related to the usage/activation (the number of selection) of each meta features. Hence, less used meta features are more likely to be selected due to this bias. Specifically, we define a global statistic $\boldsymbol{u}$ to denote the number of the update of all meta features. $\boldsymbol{u}$ reflects the usage of meta features. Then the bias is calculated by the normalization

$$\boldsymbol{\Gamma}_{bias} = \frac{\sum \boldsymbol{u} - \boldsymbol{u}}{\sum \boldsymbol{u}}. \tag{1}$$

Therefore, the final estimated coefficients β is

$$\boldsymbol{\Gamma} = \gamma \boldsymbol{\Gamma}_{bias} + (1-\gamma)\boldsymbol{\Gamma}, \tag{2}$$

where γ is a balancing coefficient, and set to 0.3 during the training process. Meta features learned with and without LRUA are shown in Fig. 2. Through LRUA, many meta features can be selected and updated to learn new patterns from local features. It avoids meta features to be optimized to only one point, and almost never selected during the composition process.

Table 1. Efficiency statistics evaluated on Tesla V100 (16/batch).

Architecture	Plain CNN	+BoMF1	+BoMF2
#Params	1.81M	1.82M	1.84M
FLOPs	26M	35M	49M

3. More Ablations

Memory & Computation Cost Tab. 1 shows the efficiency statistics comparison based on Digits. The BoMF are evaluated on Tesla V100 with the batch size of 16. The extra parameters are lightweight (0.03M), and only depends on the volumes of meta features. The increase in FLOPs/Time is due to ISTA, which is used to select related meta features $\mathbf{M}_p$.

Table 2. Experiments of single domain generalization on DomainNet. Models are trained on one domain and evaluated on the others. MetaCNN achieves the best performance, especially on Clipart, Infographic, Painting, Photo. It reflects that MetaCNN is more generalized to small distribution divergence.

Model	Clip	Info	Paint	Quick	Photo	Sketch
ERM	24.7%	20.6%	27.6%	7.1%	24.8%	25.4%
MetaCNN	29.3%	27.8%	32.9%	7.7%	29.2%	26.5%

Single Domain Generalization on DomainBed. We follow the experimental settings in [1], except for leave-one-out. To conduct single domain generalization, only one domain is selected for training and the others are used for testing. Table 2 shows the comparison between traditional EMR and proposed MetaCNN. Similar experimental results are presented that MetaCNN performs much better on Clipart, Infographic, Painting, Photo than QuickDraw, Sketch.

Table 3. Ablation results of α_1 and α_2 on Digits.

α_1,α_2	2, 0	2, 1	2, 0$\sim$1	cos, 0$\sim$1
Avg Acc	74.19%	76.53%	77.91%	78.76%

Loss Weights The effect of the loss weights α_1/α_2 are shown in Tab. 3, and the strategies of cos for α_1 and warmup for α_2 increase the final accuracy about 2%. Since the whole network is trained in an end-to-end manner, features at beginning should not be used for the reconstruction loss. The warmup strategy provides 1.5% increase for training. When the model tends to converge, the BoMF module is required to learn better meta features, the decrease of classification loss can do a favor for the variety of the learned meta features.

References

[1] G. Ishaan and L. David. In search of lost domain generalization. In *ICLR*, 2021. 2

[2] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. P. Lillicrap. Meta-learning with memory-augmented neural networks. In *ICML*, 2016. 1