Meta Convolutional Neural Networks for Single Domain Generalization

Chaoqun Wan[1], Xu Shen[1], Yonggang Zhang[2], Zhiheng Yin[3], Xinmei Tian[2]*,
Feng Gao[4], Jianqiang Huang[1], Xian-Sheng Hua[1]*
[1]Alibaba Cloud Computing Ltd,
[2]University of Science and Technology of China, [3]University of Michigan, [4]Zhejiang Lab
{qionglong.wcq,shenxu.sx,jianqiang.hjq,xiansheng.hxs}@alibaba-inc.com,
yonggang@mail.ustc.edu.cn, yzhiheng@umich.edu, xinmei@ustc.edu.cn, gaof@zhejianglab.com

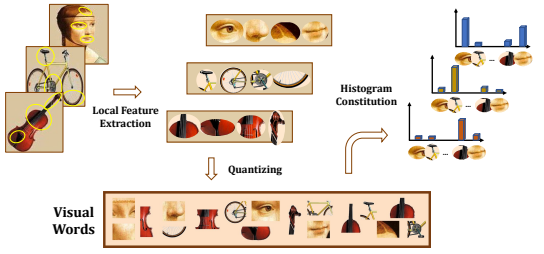
Abstract

In single domain generalization, models trained with data from only one domain are required to perform well on many unseen domains. In this paper, we propose a new model, termed meta convolutional neural network, to solve the single domain generalization problem in image recognition. The key idea is to decompose the convolutional features of images into meta features. Acting as "visual words", meta features are defined as universal and basic visual elements for image representations (like words for documents in language). Taking meta features as reference, we propose compositional operations to eliminate irrelevant features of local convolutional features by an addressing process and then to reformulate the convolutional feature maps as a composition of related meta features. In this way, images are universally coded without biased information from the unseen domain, which can be processed by following modules trained in the source domain. The compositional operations adopt a regression analysis technique to learn the meta features in an online batch learning manner. Extensive experiments on multiple benchmark datasets verify the superiority of the proposed model in improving single domain generalization ability.

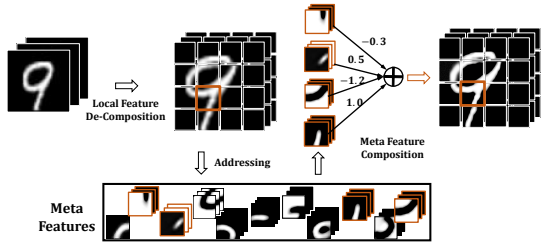
1. Introduction

Deep learning models are widely used for vision tasks in recent years [11, 13, 19, 36, 38], with the assumption that training data and testing data are from the same or similar distributions. However, when applied to unseen or out-of-distribution (OOD) test domains, the performance of the model trained on the source domain can be significantly degraded. In practice, domain shift problem is very common because of the change of illuminations, object appearance, or background [4, 30]. To solve this problem, many domain adaptation [2, 4, 23, 42] and domain generalization [3, 5, 14, 33, 41, 45, 46, 49] methods have been proposed. These methods differ in strategies of transferring knowledge from multiple source domains to the target domain. However, it is more plausible to consider a more realistic scenario where only one single domain is available for training, and the trained models are required to perform well on multiple unseen domains, *i.e.*, single domain generalization.

Single domain generalization is an important and chal-



(a) Bag of Visual Words (BoVW)



(b) Bag of Meta Features (BoMF)

Figure 1. Bag of visual words and the proposed bag of meta features. Both follow a three-step process to construct general representations for the input. (a) Local features are firstly extracted by SIFT [24], and then quantized into visual words. A frequency histogram of visual words is constituted as the final representation. (b) Input features are firstly decomposed into local features through a sliding window. Then, the addressing operation selects meta features which are related to these local features. Finally, a linear regression model is adopted to compose the output representation based on the selected meta features.

*Corresponding author.

lenging problem. For the purpose of large-scale vision applications in practical scenarios, we focus on improving the single domain generalization ability of CNNs. Recently, only a few works are proposed to solve this problem, including data augmentation [32, 43, 45] and regularization [32, 39]. Data augmentation based approaches generate more diversified data from many "fictitious" domains in the input space for training. Regularization based methods mainly focus on developing losses for the consistency among features from source data and augmented data.

In this paper, we solve the single domain generalization from a new perspective: developing a novel convolutional model, termed Meta CNN. We are motivated by stacked capsule autoencoder (SCAE) [17], in which images are segmented into constituent parts and reconstructed by a composition of general part templates. Similarly, we believe convolutional features of input images can also be decomposed into universal and elemental visual features. Then these elemental visual features are used as "templates" to compose general representations of images in the trained domain. However, due to the domain gap, features from a different domain are unable to be directly reconstructed by these trained templates. Following metadata normalization [25], the domain shift effects of such local features can be "regressed out" by a generalized linear model (GLM) [26] $\boldsymbol{f} = \beta \boldsymbol{M} + \boldsymbol{r}$, where $\boldsymbol{f}$ is the local features, β is a learnable set of linear parameters, $\boldsymbol{M}$ is the templates of trained domain (termed as "meta features"), and $\boldsymbol{r}$ is the irrelevant features effected by the domain gap. As $\beta \boldsymbol{M}$ corresponds to the component in $\boldsymbol{f}$ explained by the meta features, the features from shifted domain is reconstructed in the source domain by eliminating $\boldsymbol{r}$. Consequently, for a CNN block, the effect of domain shift in input features is eliminated by feeding $\beta \boldsymbol{M}$ instead of $\boldsymbol{f}$ into the following convolutional operations. To achieve this goal, we follow the process of bag of visual words (BoVW) [6, 24, 37], where each image is represented by the histogram of "visual words" through 3 steps (shown in Fig. 1a). We propose corresponding 3 steps of compositional operations in CNN blocks: 1) decompose the convolutional features of images into local meta features as "templates", 2) take meta features as reference, eliminate irrelevant features of local convolutional features by an addressing process, 3) reformulate the feature maps as a composition of related meta features. In this way, images are universally coded without biased information from the unseen domain, which can be processed by following modules trained in the source domain (shown in Fig. 1b).

The challenge of applying compositional operations of meta features within a CNN building block lies in four folds: 1) *local feature extraction*. For BoVW, local features are firstly located by key point detection and then extracted by handcrafted operations of local image patches. Both operations are non-differentiable, making it infeasible for end-to-end batch learning in CNN blocks. As representation and location are encoded simultaneously in a grid manner for convolutional feature maps, we propose to decompose feature maps into local patches with a sliding window, which is compatible with following convolution/pooling operations in CNN blocks. 2) *local feature addressing*. In BoVW, each local feature is mapped to a certain visual word, where a dense and large enough visual word set is required for small mapping error. However, in the deep learning scenario, the storage of batch training and inference are limited by GPUs. Therefore, we propose to map local features to a combination of meta features, where the expressive power of the meta features set is enlarged. In this way, a moderate meta features set is allowed with feasible storage occupancy. 3) *meta feature composition*. In BoVW, image is represented by a frequency histogram of visual words, which is non-differentiable and lack of the spatial and content information of image patches. To keep the content and spatial information of convolutional features, we propose to represent local patch features by a linear interpolation of meta features. 4) *meta feature learning*. In BoVW, clustering is performed over all the local features in the dataset, the center of each cluster is used as visual words. However, deep learning models are trained in an online batch learning manner, where only a very small portion of the dataset are feasible. Consequently, we adopt a regression analysis with maximum likelihood estimation to update the meta features during training.

Extensive experiments on multiple benchmark datasets indicate the superiority of the proposed model in tackling single domain generalization problems. More importantly, these results reveals the potential of convolutional meta features for general image representations.

2. Related Work

Bag of Visual Words (BoVW) is one of the most widely used model for image recognition in computer vision. To represent an image using the BoVW model, an image can be treated as a document, containing a series of "visual words". The whole procedure of BoVW can be summarized into three steps: local feature extraction, feature quantization, and histogram representation, as shown in Fig. 1a. For local feature extraction, Scale-Invariant Feature Transform (SIFT) [24], invented by David Lowe, is the most impressive method to transform an image into a large collection of disordered feature vectors. Because the features are invariant to image scale and rotation [24]. These robust features are then quantized based on visual words, *i.e.*, finding the nearest neighbor for each extracted feature. Finally, the image can be represented by a frequency histogram of the visual words. To generate reliable visual vocabulary, k-means clustering can be performed on all extracted features. Visual vocabulary are then defined as the centers of the learned

clusters. One of the notorious disadvantages of BoVW is that it ignores the spatial relationships among the patches. This issue was carefully considered when we implemented meta CNN.

Domain Generalization is one of the most general problem of discussing the stability and robustness of the convolutional neural networks. It considers the generalization capacities to unseen domains of deep models. Existing domain generalization methods can be rough classified into two categories: learning the domain invariant representation and data augmentation. Learning the domain invariant representation is to reduce the discrepancy between representations of different source domains. Classical methods includes kernel-based method [27], domain reconstruction [10], contrastive semantic alignment loss [33], domain agnostic representation learning [41], and Maximum Mean Discrepancy (MMD). Recently, meta-learning procedure has been studied to solve domain generalization problem. Li *et al.* propose and develop a gradient-based model agnostic meta-learning algorithm for domain generalization [22]. Dou *et al.* exploit the episodic training scheme, which enforces features keep alignment from the view of the local and global [8]. Du *et al.* incorporate variational information bottleneck with meta-learning to narrow the domain gap between the source domains [46].

The other category is data augmentation. Methods in this category generally aim to generate various styles of samples to enlarge the training distribution. These samples are used to train the network along with the source samples to improve the generalization ability. For instance, Riccardo *et al.* propose to generate "hard" samples for the classifier based on the adversarial training scheme [40]. Shankar *et al.* focus on the direction of the domain change, and proposed to augment the source samples along this direction [34]. Zhou *et al.* exploit a conditional generative adversarial network (GAN) to synthesize data from pseudo-novel domain [50]. Fabio *et al.* exploit an auxiliary self-supervision training signal from solving a jigsaw puzzle [5].

Single Domain Generalization is a more challenging yet realistic domain generalization task [32,48]. In this task, the network is trained on only a single source domain, and evaluated on multiple unseen domains. To deal with this challenging problem, gradient-based image augmentation is an effective strategy to improve the model generalization. Qiao *et al.* propose to encourage semantic consistency between the augmented and source images in the latent space through Wasserstein autoencoder [32]. Zhao *et al.* introduce entropy maximization in adversarial training framework to generate challenging perturbations of the source samples [48]. [32, 43] propose to learn the various styles for the generation of more diverse images. Different from existing single domain generalization methods, our method aims to discover the stability of deep models from the perspective of model architecture.

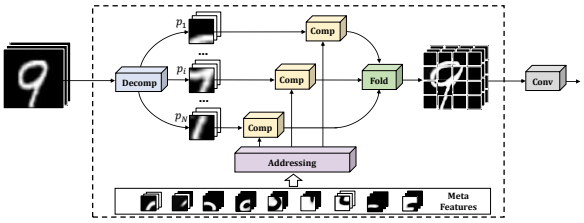


Figure 2. Overview of the proposed BoMF operations. The input convolutional feature maps are firstly decomposed into local features. Then local features are refined by a composition of related meta features. Finally, refined local features are folded to generate the final output convolutional feature maps.

3. Method

Consider input convolutional feature maps $\boldsymbol{f} \in R^{C\times H\times W}$, and meta feature set $\boldsymbol{M} \in R^{M\times D}$, where M is the number of meta features and D is the dimension of each meta feature. The goal of BoMF is to reformulate the convolutional representations as a composition of meta features, *i.e.*, $\hat{\boldsymbol{f}} = \mathbf{BoMF}(\boldsymbol{f}; \boldsymbol{M})$. The proposed BoMF operations consist of four steps, as shown in Fig. 2: 1) local feature decomposition aims to split feature maps into local patches. 2) local feature addressing selects most related meta features from the meta feature set. 3) meta feature composition generates refined local patches by integrating the selected meta features. In addition, the entire output feature maps are generated by folding all the refined local features. 4) meta feature learning is designed to update meta features from random initialization based on batch stochastic gradient descent. Details are described in the following.

3.1. Local Feature Decomposition

Local feature decomposition aims to extract local features $\{\boldsymbol{p}_i\}_{i=1}^N$ from the input convolutional feature maps $\boldsymbol{f}$ through a sliding window (shown in Fig. 1b), N is the number of local patches. Similar to convolution/pooling, the sliding window in local feature decomposition is defined by a window size $k_c \times k_h \times k_w$ and a step size $s_c \times s_h \times s_w$. This process corresponds to the unfolding[1] operation in PyTorch [31], $\{\boldsymbol{p_i}\}_{i=1}^N = unfold(\boldsymbol{f})$. Fig. 1b shows an example of extracting 16 patches from input feature maps through a sliding window with $k_c \times k_h \times k_w = C \times \frac{1}{4}H \times \frac{1}{4}W$ and $s_c \times s_h \times s_w = 1 \times \frac{1}{4}H \times \frac{1}{4}W$.

3.2. Local Feature Addressing

Given a specific local feature $\boldsymbol{p} \in R^{c\times h\times w}, D = c \times h \times w$ and the entire meta feature set $\boldsymbol{M}$, local feature addressing aims to selects most related meta features $\boldsymbol{M}_p$ from $\boldsymbol{M}$. To improve the efficiency of composition and the

[1] https://pytorch.org/docs/stable/generated/torch.nn.Unfold.html

express power of meta features, it's natural to select fewest meta features (*i.e.*, $\min |\boldsymbol{M}_p|$) with minimized fitting error of $\boldsymbol{p}$. Thus, this selection process can be formulated as a sparse coding problem [1, 15], that is, estimating a coefficients vector $\boldsymbol{\Gamma}$ based on $\boldsymbol{p} = \boldsymbol{\Gamma}\boldsymbol{M} + \boldsymbol{r}$ and

$$\min_{\boldsymbol{\Gamma}} ||\boldsymbol{p} - \boldsymbol{\Gamma}\boldsymbol{M}||_2 \; s.t. \; ||\boldsymbol{\Gamma}||_1 < \epsilon. \tag{1}$$

Here, $\boldsymbol{r}$ is the irrelevant features, and the $\mathcal{L}_1$ constraint of $\boldsymbol{\Gamma}$ controls the number of selected meta features. To solve this problem, an ordinary solution is Iterative Shrinkage Thresholding Algorithm (ISTA) [15, 35]. Given an initial $\boldsymbol{\Gamma}_0 = \boldsymbol{0}$, ISTA iterates the recursive equation $\boldsymbol{\Gamma}_{k+1} = \mathcal{S}_{\frac{\lambda}{c}}(\boldsymbol{\Gamma}_k + \frac{1}{c}(\boldsymbol{p} - \boldsymbol{\Gamma}_k\boldsymbol{M})\boldsymbol{M}^T)$ several times. After this iteration, meta features with corresponding non-zero values in $\boldsymbol{\Gamma}$ are incorporated to construct $\boldsymbol{M}_p$.

3.3. Meta Feature Composition

Meta feature composition firstly integrates related meta features $\boldsymbol{M}_p$ to refine $\boldsymbol{p}$ and then compose the refined local features to feature maps based on their spatial relationship.

To remove the irrelevant features of $\boldsymbol{p}$ without the loss of content, a general linear model (GLM) [26, 28] is adopted to associate local feature $\boldsymbol{p}$ and meta features $\boldsymbol{M}_p$ by $\boldsymbol{p} = \beta\boldsymbol{M}_p + \boldsymbol{r}$. The optimal linear coefficients β is given by the closed-form solution $\beta = (\boldsymbol{M}_p^T\boldsymbol{M}_p)^{-1}\boldsymbol{M}_p^T\boldsymbol{p}$. Therefore, the refined local feature $\hat{\boldsymbol{p}}$ without irrelevant features $\boldsymbol{r}$ based on selected meta features $\boldsymbol{M}_p$ is

$$\hat{\boldsymbol{p}} = \beta\boldsymbol{M}_p = p\boldsymbol{M}_p^T(\boldsymbol{M}_p\boldsymbol{M}_p^T)^{-1}\boldsymbol{p}\boldsymbol{M}_p. \tag{2}$$

To maintain the spatial relationship of refined local features, the final output feature maps $\hat{\boldsymbol{f}}$ is constructed by locating $\{\hat{\boldsymbol{p}}\}_{i=1}^N$ to the same position as $\{\boldsymbol{p}\}_{i=1}^N$ in $\boldsymbol{f}$ (shown in the right side of Fig. 1b). This is an inverse process of local feature decomposition and corresponds to the folding[2] operation in PyTorch [31], *i.e.*, $\hat{\boldsymbol{f}} = fold(\{\hat{\boldsymbol{p}}_i\}_{i=1}^N)$.

3.4. Meta Feature Learning

The operations described in Sec. 3.2 and Sec. 3.3 are built based on a known meta feature set $\boldsymbol{M}$. An unsolved problem is to learn these meta features from random initialization. Meta feature learning aims to learn the set of meta features from random initialization based on batch stochastic gradient descent. Given the latest meta features $\boldsymbol{M}$ and a batch of input features $\{\boldsymbol{f}_i\}_{i=1}^B$, β is firstly estimated in the forward pass based on operations in Sec. 3.1, 3.2, and 3.3. Then the estimated β is fixed, meta features can be updated by aggregating the gradients from the reconstruction error in Eq. 2 and the back-propagated gradients from the supervised classification loss $\mathcal{L}_{cls}$:

$$\mathcal{L} = \alpha_1||\beta\boldsymbol{M}_p - \boldsymbol{p}||^2 + \alpha_2\mathcal{L}_{cls}, \tag{3}$$

[2] `https://pytorch.org/docs/stable/generated/torch.nn.Fold.html`

Table 1. Our meta CNN model for Digits categorization. The baseline model shares the same parameter settings for the convolutional and fully connected modules. The k, s and $\#k$ represent the kernel size, stride and the output channels. The n denotes the number of meta features.

Layer		Parameters
$\mathbf{BoMF}_1$	decomp	$k = 9 \times 9 \times 3$, $s = 2 \times 2 \times 1$
	meta feat	$n = 100$
	conv	$k = 5 \times 5$, $s = 1$; $\#k = 64$
$\mathbf{maxpool}_1$		$k = 2 \times 2$
$\mathbf{BoMF}_2$	decomp	$k = 5 \times 5 \times 4$, $s = 1 \times 1 \times 1$
	meta feat	$n = 200$
	conv	$k = 5 \times 5$, $s = 1$; $\#k = 128$
$\mathbf{maxpool}_2$		$k = 2 \times 2$
$\mathbf{fc}_1$		3200×1024
$\mathbf{fc}_2$		1024×1024
$\mathbf{fc}_3$		1024×10

where α_1, α_2 is the weights for reconstruction loss and classification loss, respectively. The balance between these two objectives pushes the meta features to learn both general and discriminative local patterns.

3.5. Instantiation

To understand the behavior of Meta CNN, we conduct comprehensive ablation experiments on the generalization of Digits classification [7, 9, 20, 29]. First we describe our baseline network architectures for this task, and then extend them to meta convolutional neural networks.

Plain Networks. As in [34, 50], the baseline convolutional neural networks consist of two convolutional blocks. Each convolutional layer is followed by ReLU activation and 2×2 max-pooling. The classifier is a cascade of three fully connected layers.

The plain networks share the same settings ("conv", "maxpool" and "fc" in Table 1) with Meta CNNs except those of BoMF operations.

Meta CNN. In meta CNN, two extra BoMF modules are inserted to the convolutional blocks. The setting of the local feature decomposition ("decomp") and the number of meta features ("meta feat") are presented in Table 1.

4. Experiments

In this section, experimental setups and implementation details are introduced in Sec. 4.1 and 4.2. In Sec. 4.3, experiments are conducted on three widely used benchmarks of domain generalization. Further analysis and visualizations are provided in Sec. 4.4.

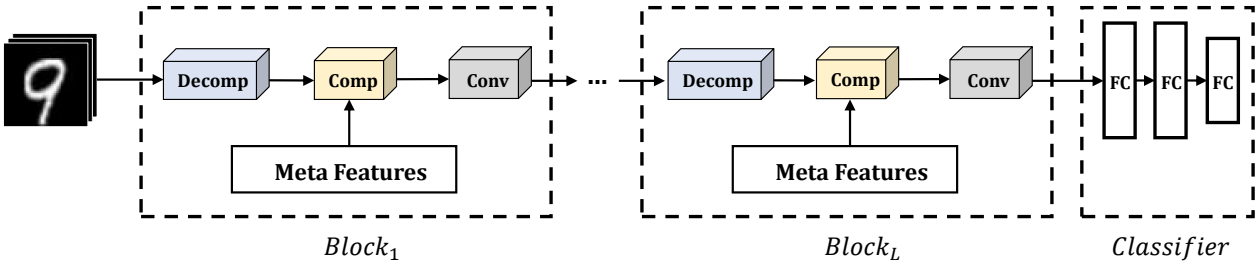


Figure 3. The framework of Meta Convolutional Neural Networks. Each block consists of three steps: decomposition of input, composition of local features based on meta features and convolutional operations. This network is trained in an end-to-end manner with a general classification loss.

4.1. Datasets and Settings

Digits consists of 5 different datasets, including MNIST [20], SVHN [29], MNIST-M [9], SYN [9] and USPS [7]. Images in different datasets have different font styles, scales, backgrounds, stroke colors, *etc*. Following [32,43,45], the first $10,000$ images of the training set of MNIST are selected to train the models, and other datasets are used to evaluate the generalization of performance. All the images are converted into RGB, and resized to 32×32 during data preprocessing.

CIFAR-10-C, also termed as corrupted CIFAR-10 [12] is a robustness benchmark consisting of 19 types of corruptions on the test set of CIFAR-10 [18]. These corruptions are from 4 main categories, *weather, blur, noise*, and *digital*. Each corruption has five-level severities and "5" indicates the most corrupted level. Following [43], the training set of CIFAR-10 is selected as the source dataset for training, while images in CIFAR-10-C with the corruption of level "5" are used for evaluation. For simplicity, only the accuracy of each category and the average accuracy of all categories are reported.

PACS [21] is a recent proposed domain generalization benchmark dataset that has four domains, *photo, art painting, cartoon*, and *sketch*. Each domain contains 224×224 images belonging to seven categories, and there are $9,991$ images in total. Compared with the digits dataset, PACS is a more challenging dataset due to the large style shift between domains. Following [43], images in photo are selected as the source domain for training, while other images are used for evaluation.

4.2. Implementation Details

For BoFM operations, the meta features are initialized as random Gaussian noise with standard derivation of 0.01. In the local feature addressing step, the average number of selected meta features $|\bar{\boldsymbol{M}}_p|$ of two BoMF operations are controlled around 4 and 8, respectively. The normalized coefficient c is set as 1, and each meta feature is normalized through $\frac{\boldsymbol{M}_i}{||\boldsymbol{M}_i||^2}$ after every iteration [1]. In the local feature learning step, to improve the number of activated meta features and enlarge the diversity of learned semantic information, a bias term is introduced for $\boldsymbol{\Gamma}$. This bias has the same dimension as $\boldsymbol{\Gamma}$, and reflects the usage (number of update) of meta features. It assigns big values to those rarely updated meta features. Therefore, a weighted summation of $\boldsymbol{\Gamma}$ and bias can eliminate the unbalance of the update of $\mathbf{M}$. More details are presented in supplementary materials.

Table 2. Experiments of single domain generalization on digits classification. Models are trained on the first $10,000$ training images in MNIST and evaluated on the rest of the digits datasets. MetaCNN achieves the best performance, especially on SYN and USPS. The image styles and backgrounds in SYN and USPS are simple, images from these two datasets are closer to images in MNIST. It reflects that MetaCNN is more generalized to small distribution divergence.

Method	SVHN	MNIST-M	SYN	USPS	Avg
ERM [40]	27.83	52.72	39.65	76.94	49.29
CCSA [33]	25.89	49.29	37.31	83.72	49.05
d-SNE [44]	26.22	50.98	37.83	93.16	52.05
JiGen [5]	33.80	57.80	43.79	77.15	53.14
ADA [40]	35.51	60.41	45.32	77.26	54.62
M-ADA [32]	42.55	67.94	48.95	78.53	59.49
ME-ADA [48]	42.56	63.27	50.39	81.04	59.32
RandConv [45]	57.52	87.76	62.88	83.36	72.88
L2D [43]	62.86	87.30	63.72	83.97	74.46
MetaCNN (ours)	**66.50**	**88.27**	**70.66**	**89.64**	**78.76**

For model architecture, we design specific task models and different training strategies for three datasets. Specifically, models for Digits are described in Sec. 3.5. As for CIFAR-10-C and PACS, the plain networks are WideResNet (16-4) [47] and AlexNet [19] for fair comparison [43]. Similar to Sec. 3.5, the corresponding MetaCNN is constructed by inserting two extra BoMF modules to the first two convolutional blocks in WideResNet and AlexNet. Different from Table 1, the kernel size and step size of the first BoMF for PACS is set to 17 and 4 in spatial dimension.

For model training, batch size is set to 32, and all the networks are optimized through Adam [16]. The loss weight α_1 and α_2 are set to 2.0 and 0.1 initially. α_1 gradually decreases to 0 based on cosine function $cos(\frac{epoch}{|epochs|}\pi) + 1$,

Table 3. Experiments of single domain generalization on digits classification. Models are trained on CIFAR-10 and evaluated on the CIFAR-10-C. "*" means our implementation. MetaCNN achieves the best performance, especially on *blur, noise* and *digits*. Images in these three categories are closer to images in CIFAR-10. It reflects that MetaCNN is more generalized to small distribution divergence.

Method	Weather	Blur	Noise	Digits	Avg
ERM [40]	67.28	56.73	30.02	62.30	54.08
CCSA [33]	67.66	57.81	28.73	61.96	54.04
d-SNE [44]	67.90	56.59	33.97	61.83	55.07
M-ADA [32]	75.54	63.76	54.21	65.10	64.65
L2D [43]	75.98	69.16	73.29	72.02	72.61
RandConv* [45]	76.87	55.36	75.19	77.51	71.23
MetaCNN (ours)	**77.44**	**76.80**	**78.23**	**81.26**	**78.45**

while α_2 linearly increases to 1 during the first 10 epochs. In Digits, the initial learning rate is 0.0035 for conv/fc, and 0.001 for BoMF. It decays by a factor of 0.1 at 50 and 100 epochs. Training lasts 120 epochs in total. In CIFAR-10-C, the initial learning rate is 0.0001 for conv/fc, and 0.001 for BoMF. The strategy to decay the learning rate is the same as Digits. In PACS, the initial learning rate is 0.00035 for conv/fc, and 0.01 for BoMF since the scale of the training data is inadequate. The learning rate decays at 24 epochs with 30 epochs in total. All models and optimizations are implemented in PyTorch [31].

4.3. Evaluation of Single Domain Generalization

Table 2, 3 and 4 exhibit the evaluations of single domain generalization on Digits, CIFAR-10-C, and PACS, respectively. Results show that MetaCNN achieves the highest average accuracy compared to other baselines on three benchmarks. Specifically, in Table 2, improvements are 3.6%, 0.9%, 6.9%, 5.7% on SVHN, MNIST-M, SYN and USPS, compared to the previous best method L2D. In Table 3, there are 0.6%, 10.7%, 4.9%, 3.7% improvements on *weather, blur, noise and digital*, respectively. In Table 4, MetaCNN outperforms the previous methods in *cartoon, sketch*, and the average performance. The above results indicate the proposed BoMF operations improve the model generalization, reflecting the effectiveness of composing generalized output features through meta features.

In addition, there is an obvious discrepancy among different datasets rather than different benchmarks. Performance gains are appreciable on USPS/SYN [Digits], *blur/noise/digital* [CIFAR-10-C], and *cartoon/sketch* [PACS], yet insufficient on MNIST-M [Digits], *weather* [CIFAR-10-C], and *art painting* [PACS]. It is interesting that the division of these two groups of datasets reveals a certain consistent rule. In the first group, the image styles and image backgrounds are simple and close to the training images from MNIST. However, in the second group, images have complex styles and background variations which are very different from the training images. Therefore, large distribution divergence (domain gaps) increases the difficulty to recognize the convolutional features given the meta features, hindering the performance of the final classification. In conclusion, the proposed BoMF operations and its constructed MetaCNN are effective on single domain generalization, especially for data that have small distribution divergence.

Table 4. Experiments of single domain generalization on digits classification. Models are trained on photo and evaluated on the rest of the target domains (*i.e.*, art painting, cartoon, and sketch). Best performances are highlighted in bold.

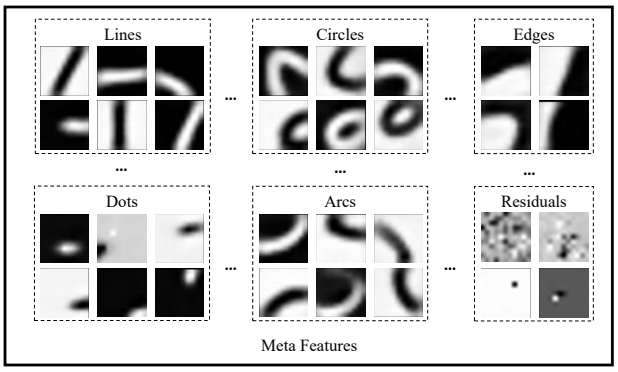
Method	A	C	S	Avg
ERM [40]	54.43	42.74	42.02	46.39
JiGen [5]	54.98	42.62	40.62	46.07
RSC []	56.26	39.59	47.13	47.66
ADA [40]	58.72	45.58	48.26	50.85
M-ADA [32]	**58.96**	44.09	49.96	51.00
L2D [43]	56.26	51.04	58.42	55.24
MetaCNN (Ours)	54.05	**53.58**	**63.88**	**57.17**

4.4. Ablation Study

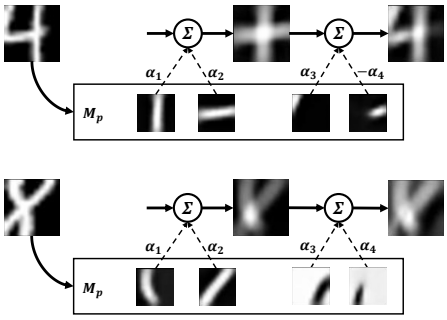
This subsection provides ablation studies and analysis of each component in BoMF.

Meta Feature Learning and **Meta Feature Composition** directly affect the outputs of BoMF. Fig. 4a exhibits the learned meta features, *e.g.*, points, straight lines, arcs, semicircles, indicating the meta feature learning is able to decouple the local features into basic elements. These basic elements are used to compose the outputs, as shown in Fig. 4b. Given a specific local feature, the most related meta features are selected for composition. For example, in the first row of Fig. 4b, the local patch of digit "4" first uses the vertical and horizontal lines to compose the major structure. Then, top left corner and middle right points are selected to complete the remaining structure. In addition, complicated local patches (patches at the middle of "8") can also be constructed given learned basic meta features. Taken together, linear regression analysis shows its efficiency in learning meta features and composing general image features. Comparing the input and output of meta feature composition, these two operations also show their superiority in reserving the domain-agnostic information (structure *etc*.) of the input and removing the domain-specific information (style, appearance *etc*.).

Local feature addressing. Table 5 compares the performances of two addressing techniques. "MetaCNN-$\boldsymbol{M}_p$"

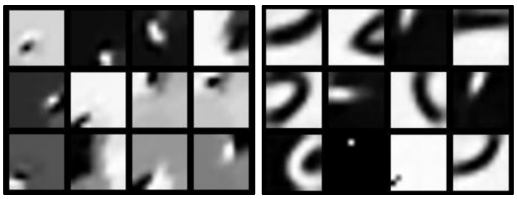


(a) Learned Meta Features



(b) Meta Feature Composition

Figure 4. (a) The learned results of meta features. Distinctive local patterns, *e.g.*, lines, circles, edges, dots, arcs, and even some residuals, can be extracted from local features. (b) The procedure of meta feature composition. Given a specific local feature, the most related meta features are selected and linearly combined to compose the output.



(a) w/o addressing (b) w/ addressing

Figure 5. Comparison of the learned meta features w & w/o local feature addressing. More meaningful and diversified local patterns are learned based on addressing.

Table 5. Experiment of evaluating the effectiveness of local feature addressing, showing that local feature addressing is essential for BoMF. "$\boldsymbol{M}$" means skipping the addressing and using all meta features. "$\boldsymbol{M}_p$" represents using the local feature addressing.

Method	SVHN	MNIST-M	SYN	USPS	Avg
MetaCNN-$\boldsymbol{M}$	62.84	85.62	65.45	83.21	74.28
MetaCNN-$\boldsymbol{M}_p$	**66.50**	**88.27**	**70.66**	**89.64**	**78.76**

represents the local feature addressing, while "MetaCNN-$\boldsymbol{M}$" represents directly composing all meta features. Results show that local feature addressing leads to better generalization ability. On the one hand, local feature addressing is implemented via sparse coding, which has already been proved beneficial for model generalization. On the other hand, the sparse constraint pushes the meta features to select more related local features and learn more discriminative patterns. Fig. 5 visualizes the learned meta features with and without local feature addressing. Fig. 5b contains more diversified local patterns, *i.e.*, points, straight lines, arcs, showing the significance of local feature addressing.

Hyper-parameters are kernel size k and step size s in local feature decomposition, and the size of meta features M in meta feature learning. Fig. 7a and 7b evaluates

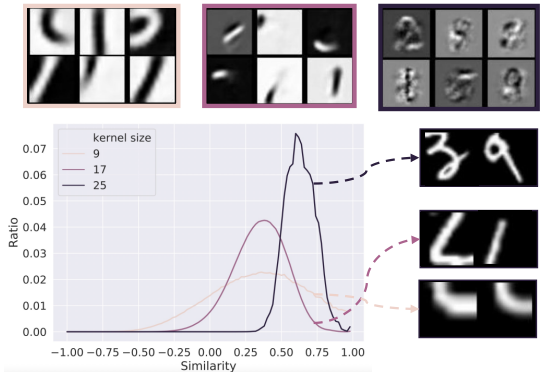
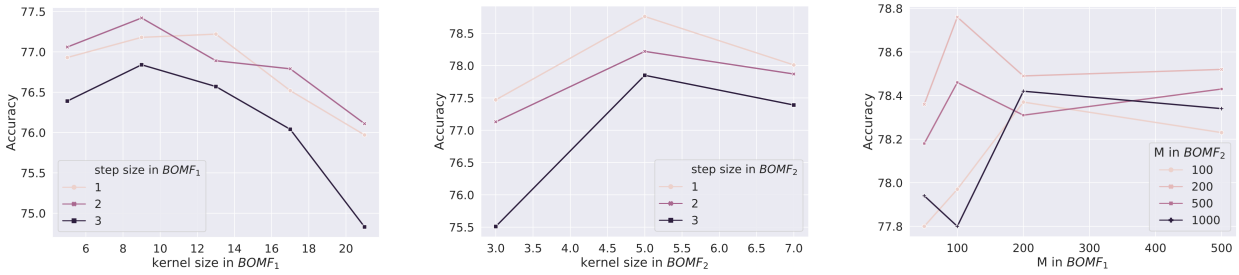


Figure 6. Similarities (Euclidean distance) among local features w.r.t. different sliding window sizes. Larger patch size presents relatively higher similarities. At right, one case of the local features with high similarities are presented for each patch size. The contrasts between dis-similar appearance and small distance reveals the necessity of decomposing input feature maps into local features with small patch size.

the generalization performances on various kernel sizes and step sizes for both BoMF operations. The results show that a medium kernel size is the best. Fig. 7c demonstrates that the accuracy becomes relatively stable after the number of meta features reaches 200. Fig. 6 explores the similarity distributions of local patches under different kernel sizes. The contrasts between dis-similar appearance and small distance reveals the necessity of decomposing input feature maps into local features with small patch size.

4.5. Deeper Meta CNNs

Previous experiments are conducted on Meta CNNs with two blocks (BoMF1 $\sim$ 2 in Table 6). To investigate the effectiveness of deeper Meta CNNs, two more compositional blocks are added to the aforementioned backbone. The kernel size and step size are $3\times3\times4$ and $1\times1\times1$ for the com-



(a) kernel size & step size in $BoMF_1$ (b) kernel size & step size in $BoMF_2$ (c) number of meta features

Figure 7. Hyper-parameter analysis for BoMF. (a) & (b): Performance on different kernel sizes and step sizes indicates medium kernel size achieves the best performance. (c) As the number of meta features increase, the performances are relatively stable. It shows that the number of meta features for $BoMF_1$ and $BoMF_2$ are sufficient at 100 and 200.

Table 6. Performance of Meta CNNs with various depths. The superior performance of deeper backbones indicates that BoMF operations are generic components for convolutional blocks.

Method	SVHN	MNIST-M	SYN	USPS	Avg
BoMF1	64.86	86.30	68.72	88.97	77.21
BoMF1$\sim$ 2	66.50	88.27	70.66	89.64	78.76
BoMF1$\sim$ 3	**67.23**	88.63	71.44	**90.07**	**79.34**
BoMF1$\sim$ 4	66.73	**88.71**	**71.76**	89.79	79.24

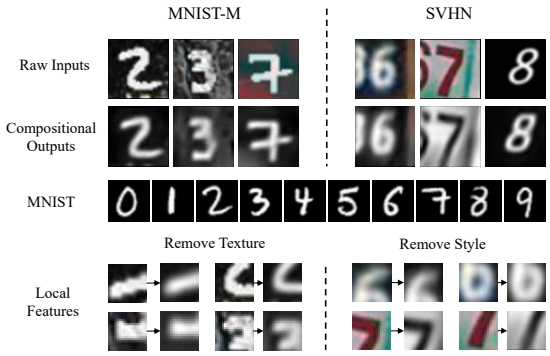


Figure 8. Visualization of meta feature composition results for unknown data (better viewed in color). Irrelevant textures and styles in MNIST-M/SVHN are removed, and the compositional outputs are close to the training images in MNIST.

positiona operations. The kernel size/stride is $3/1$ for convolutions. Table 6 shows the performance of Meta CNNs with various depth. The superior performance of deeper backbones indicates that the pro-posed BoMF operations are generic components for convolutional blocks.

4.6. Composition of unknown data

Fig. 8 shows the output of meta feature composition for unknown data. In MNIST-M, background textures lead to blurred edges of the digits. In SVHN, the styles of digits are different from which in MNIST. With the help of BoMF, most textures are removed in the compositional outputs of MNIST-M, and special styles in SVHN are removed after meta feature composition. These composed outputs are close to images in MNIST, indicating the effectiveness of the meta feature composition in eliminating domain gap.

5. Discussion

Limitation and future work. This paper focuses on generic models with high performance rather than efficiency. The limitations of this work lie in the extra computational cost of the proposed local feature decomposition and meta feature composition. This limitation can be easily alleviated by using key points to extract local features. We will discuss more options in the future.

Potential societal impact. BoMF is proved to be an effective and general operation to improve the model generalization ability. Therefore, this work possesses the potential to improve the model generalization ability on unknown data in real applications.

6. Conclusion

In this work, we propose a new perspective for single domain generalization. Compositional operation BoMF is proposed and built by four steps: local feature decomposition, local feature addressing, meta feature composition, and meta feature learning. Experiments on three single domain generalization benchmark datasets illustrate the effectiveness of the proposed model. Extensive visualizations reveal the potential of BoMF to learn expressive meta features for a general representation of images.

Acknowledgements

This work was supported by National Key R&D Program of China (No. 2020AAA0103903), partially by Alibaba Innovative Research (AIR) program and NSFC No. 61872329.

References

[1] M. Aharon, M. Elad, and A. M. Bruckstein. K-SVD: an algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, 2006.

[2] M. Baktashmotlagh, M. T. Harandi, B. C. Lovell, and M. Salzmann. Unsupervised domain adaptation by domain invariant projection. In *ICCV*, 2013.

[3] Y. Balaji, S. Sankaranarayanan, and R. Chellappa. Metareg: Towards domain generalization using meta-regularization. In *NeurIPS*, 2018.

[4] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan. A theory of learning from different domains. *Machine learning*, 79(1):151–175, 2010.

[5] F. Maria Carlucci, A. D'Innocente, S. Bucci, B. Caputo, and T. Tommasi. Domain generalization by solving jigsaw puzzles. In *CVPR*, 2019.

[6] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *ECCV Workshop*, 2004.

[7] J. S. Denker, W. R. Gardner, H. P. Graf, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, H. S. Baird, and I. Guyon. Neural network recognizer for hand-written zip code digits. In *NeurIPS*, 1989.

[8] Q. Dou, D. Coelho de Castro, K. Kamnitsas, and B. Glocker. Domain generalization via model-agnostic learning of semantic features. In *NeurIPS*, 2019.

[9] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. In *ICML*, 2015.

[10] M. Ghifary, W. B. Kleijn, M. Zhang, and D. Balduzzi. Domain generalization for object recognition with multi-task autoencoders. In *ICCV*, 2015.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[12] D. Hendrycks and T. G. Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *ICLR*, 2019.

[13] G. Huang, Z. Liu, L. v. d. Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

[14] Z. Huang, H. Wang, E. P. Xing, and D. Huang. Self-challenging improves cross-domain generalization. In *ECCV*, 2020.

[15] D. Ingrid, D. Michel, and D.M. Christine. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 57(11):1413–1457, 2004.

[16] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[17] A. Kosiorek, S. Sabour, Y. W. Teh, and G. E. Hinton. Stacked capsule autoencoders. In *NeurIPS*, 2019.

[18] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012.

[20] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[21] D. Li, Y. Yang, Y. Song, and T. M. Hospedales. Deeper, broader and artier domain generalization. In *ICCV*, 2017.

[22] D. Li, Y. Yang, Y. Song, and T. M. Hospedales. Learning to generalize: Meta-learning for domain generalization. In *AAAI*, 2018.

[23] M. Long, Y. Cao, J. Wang, and M. Jordan. Learning transferable features with deep adaptation networks. In *ICML*, 2015.

[24] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[25] M. Lu, Q.u Zhao, J. Zhang, K. M. Pohl, Li Fei-Fei, J. C. Niebles, and E. Adeli. Metadata normalization. In *CVPR*, 2021.

[26] R. McNamee. Regression modelling and other methods to control confounding. *Occupational and Environmental Medicine*, 62(7):500–506, 2005.

[27] K. Muandet, D. Balduzzi, and B. Schölkopf. Domain generalization via invariant feature representation. In *ICML*, 2013.

[28] J. Neter, M. H. Kutner, C.J. Nachtsheim, W. Wasserman, and et al. Applied linear statistical models. 1996.

[29] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Ng. Reading digits in natural images with unsupervised feature learning. In *NeurIPS Workshop*, 2011.

[30] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[31] A. Paszke, S. Gross, T Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

[32] F. Qiao, L. Zhao, and X. Peng. Learning to learn single domain generalization. In *CVPR*, 2020.

[33] M. Saeid, P. Marco, A. Adjeroh Donald, and D. Gianfranco. Unified deep supervised domain adaptation and generalization. In *ICCV*, 2017.

[34] S. Shankar, V. Piratla, S. Chakrabarti, S. Chaudhuri, P. Jyothi, and S. Sarawagi. Generalizing across domains via cross-gradient training. In *ICLR*, 2018.

[35] D. Simon and M. Elad. Rethinking the CSC model for natural images. In *NeurIPS*, 2019.

[36] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[37] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *ICCV*, 2003.

[38] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[39] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell. Adversarial discriminative domain adaptation. In *CVPR*, 2017.

[40] R. Volpi, H. Namkoong, O. Sener, J. C. Duchi, V. Murino, and S. Savarese. Generalizing to unseen domains via adversarial data augmentation. In *NeurIPS*, 2018.

[41] H. Wang, Z. He, Z. C. Lipton, and E. P. Xing. Learning robust representations by projecting superficial statistics out. In *ICLR*, 2019.

[42] M. Wang and W. Deng. Deep visual domain adaptation: A survey. *Neurocomputing*, 312:135–153, 2018.

[43] Z. Wang, Y. Luo, R. Qiu, Z. Huang, and M. Baktashmotlagh. Learning to diversify for single domain generalization. In *ICCV*, 2021.

[44] X. Xu, X. Zhou, R. Venkatesan, G. Swaminathan, and O. Majumder. d-sne: Domain adaptation using stochastic neighborhood embedding. In *CVPR*, 2019.

[45] Z. Xu, D. Liu, J. Yang, C. Raffel, and M. Niethammer. Robust and generalizable visual representation learning via random convolutions. In *ICLR*, 2021.

[46] D. Yu, J. Xu, H. Xiong, Q. Qiu, X. Zhen, C. Snoek, and L. Shao. Learning to learn with variational information bottleneck for domain generalization. In *ECCV*, 2020.

[47] S. Zagoruyko and N. Komodakis. Wide residual networks. In *BMVC*, 2016.

[48] L. Zhao, T. Liu, X. Peng, and D. N. Metaxas. Maximum-entropy adversarial data augmentation for improved generalization and robustness. In *NeurIPS*, 2020.

[49] K. Zhou, Y. Yang, T. Hospedales, and T. Xiang. Learning to generate novel domains for domain generalization. In *ECCV*, 2020.

[50] K. Zhou, Y. Yang, T. M. Hospedales, and T. Xiang. Learning to generate novel domains for domain generalization. In *ECCV*, 2020.