Masked Feature Generation Network for Few-Shot Learning

Yunlong Yu[1] , **Dingyi Zhang**[1] , **Zhong Ji**[2]
[1] College of Information Science and Electronic Engineering, Zhejiang University
[2] School of Electrical and Information Engineering, Tianjin University
{yuyunlong, dyzhang}@zju.edu.cn, jizhong@tju.edu.cn

Abstract

In this paper, we present a feature-augmentation approach called **M**asked **F**eature **G**eneration **N**etwork (**MFGN**) for Few-Shot Learning (FSL), a challenging task that attempts to recognize the novel classes with a few visual instances for each class. Most of the feature-augmentation approaches tackle FSL tasks via modeling the intra-class distributions. We extend this idea further to explicitly capture the intra-class variations in a one-to-many manner. Specifically, MFGN consists of an encoder-decoder architecture, with an encoder that performs as a feature extractor and extracts the feature embeddings of the available visual instances (the unavailable instances are seen to be masked), along with a decoder that performs as a feature generator and reconstructs the feature embeddings of the unavailable visual instances from both the available feature embeddings and the masked tokens. Equipped with this generative architecture, MFGN produces nontrivial visual features for the novel classes with limited visual instances. In extensive experiments on four FSL benchmarks, MFGN performs competitively and outperforms the state-of-the-art competitors on most of the few-shot classification tasks.

1 Introduction

Few-Shot Learning (FSL) is a challenging task that aims at recognizing the novel target classes from a few instances with the assistance of a set of disjoint data-rich base classes. As a promising direction of alleviating the data-hungry issue in the deep models, FSL has attracted a lot of attention in recent years.

Typically, FSL is tackled by learning a general model from the base data and then generalizing to the target classes. Thus, most of the existing FSL approaches are fallen into the transfer learning branch. The metric-learning-based approaches [Snell *et al.*, 2017; Sung *et al.*, 2018] dominate the state-of-the-art methods and have made significant advances. Though promising results have been achieved, the metric-learning-based approaches do not explore more information for the novel target classes, which may restrict their generalization ability when predicting the target classes.

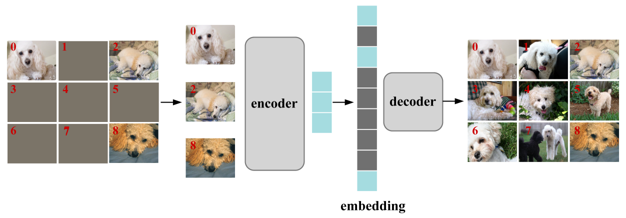


Figure 1: Visualization of basic idea of generating visual instances. A set of instances from the same class are seen as the input and the unavailable instances are viewed to be masked. The model is used to reconstruct them.

A straightforward way for FSL is to augment more data for the data-insufficient target classes. A typical strategy is to generate new samples or visual features based on the adversarial framework. For example, [Schwartz *et al.*, 2018] and [Li *et al.*, 2020] assume that the intra-class variations are shared across categories and attempt to capture the class distribution in either a pair-wise manner or specific regularizers. However, the performances of the most existing data-augmented approaches are slightly worse than the state-of-the-art methods due to the mode collapse issue, resulting in that the generated instances or visual features hardly capture the intra-class variations.

As demonstrated in [Tian *et al.*, 2020], the good feature embedding is a key to addressing FSL tasks. We argue that the properties of good feature embedding for FSL should be both discriminant and diverse. In this paper, we formulate this goal with an encoder-decoder framework, where the encoder learns a discriminative feature extractor via training a classification task and the decoder disturbs the feature embedding space to produce diverse feature embeddings.

Inspired by the success of MAE [He *et al.*, 2021] that reconstructs the masked patches with an asymmetric encoder-decoder architecture and leads to an efficient pre-trained vision learner, we follow and extend the idea to take a set of samples from the same class as the input and regard each sample as a patch; and the model is used to reconstruct the masked samples, as illustrated in Fig. 1. However, considering that the implementation of FSL is achieved in the visual embedding space, thus we relax the architecture and design the decoder to reconstruct the feature embeddings instead of visual instances. Specifically, the encoder is performed as a backbone to extract the visual features and the decoder is performed as a generator to reconstruct the feature embeddings of the masked visual instances. The designs of both the encoder and decoder are flexible. In this paper, the encoder is a CNN backbone and the decoder is a transformer. Selecting the transformer as the decoder is due to that the transformer takes different position embeddings as input that is beneficial in reconstructing diverse visual features. In a nutshell, the highlights of this work are summarized:

1. We propose a novel framework called masked feature generation network (MFGN) for FSL. MFGN addresses the FSL via reconstructing both discriminative and diverse visual features.
2. We apply the transformer to reconstruct diverse visual features and verify that the transformer is capable to transfer the high-level visual features.
3. We conduct extensive experiments on both coarse-grained and fine-grained benchmarks. The experimental results show that the proposed MFGN could augment some meaningful visual features, which facilitates the approach to beat most of the competitors with a significant margin.

2 Related Work

As a promising way to alleviate the data demands of the deep models, FSL has been attracting a lot attention in recent years. Based on the way of training the base data, the existing FSL approaches can be roughly divided into three categories, i.e., meta-learning-based approaches, metric-based approaches, and the data-hallucination-based approaches.

The meta-learning-based approaches address FSL via training a meta-learner with the meta-learning paradigm, which attempts to adjust the optimization algorithms so that the models quickly adapt to the FSL tasks, such as model-agnostic meta-learner (MAML) [Finn *et al.*, 2017], closed-form solvers [Bertinetto *et al.*, 2018], convex learners [Lee *et al.*, 2019], and contextual learner [Cai *et al.*, 2018].

The metric-learning-based approaches address the FSL tasks via learning a discriminative visual embedding space with the base data and expect that the learned embedding space also works for the novel classes. The existing metric-learning-based approaches differ in learning the visual embedding space, such as ProtoNet [Snell *et al.*, 2017], MatchingNet [Vinyals *et al.*, 2016], RelationNet [Sung *et al.*, 2018], and DN4 [Li *et al.*, 2019a]. Some nontrivial techniques are further combined with the pipeline, such as the attention [Zhu *et al.*, 2020; Dong *et al.*, 2021; Xu *et al.*, 2021; Hou *et al.*, 2019] and semantic information guidance [Xing *et al.*, 2019]. Recently, some approaches [Chen *et al.*, 2018; Tian *et al.*, 2020] reveal that the visual embedding space learned with a simple classification task could achieve superior performance and the fine-tuning techniques on the pre-trained model could further improve the performances [Sun *et al.*, 2019]. The proposed MFGN is trained end-to-end from the scratch without fine-tuning, thus has the potential to further improve if the fine-tuning techniques are applied.

The data-hallucination-based approaches address the FSL via synthesizing the visual features [Schwartz *et al.*, 2018; Li *et al.*, 2020] to capture the class-agnostic patterns in either an adversarial method or a pair-wise manner. However, most of the hallucination-based approaches are inferior to the state-of-the-art competitors. The proposed MFGN also falls into this branch. Differently, MFGN inherits the discriminative property of the metric-learning-based approaches and focuses on capturing intra-class variations in a one-to-many manner instead of revealing the real-like feature distribution.

The most related work to ours is the masked autoencoder (MAE) [He *et al.*, 2021], a scalable self-supervised learner for visual feature extraction. Our MFGN is inspired by MAE and also follows an encoder-decoder framework but differs in both purpose and implementation. First, our MFGN is designed for the data-limited tasks and attempt to capture the intra-class variations while MAE is designed for the data-hungry models to obtain a scalable visual learner. Second, MAE masks the patches of an image and enforces the model to reconstruct the masked patches; while MFGN masks the visual features of the input instances and learns to reconstruct the visual features with both the available visual features and the mask tokens. Finally, MFGN is trained in a supervised way to extract more discriminant visual features while MAE is an unsupervised approach.

3 Methodology

Our masked feature generation network (MFGN) follows the traditional autoencoder framework that consists of an encoder and a decoder. Different from the traditional autoencoder that reconstructs the original signal with the input observations, our MFGN attempts to reconstruct the variation of the input (i.e., the feature embeddings of the input visual images) given some partial observations. MFGN shares a similar asymmetric encoder and decoder framework with MAE [He *et al.*, 2021], but has a specific design. In MFGN, the encoder only extracts the feature embeddings of the available visual instances and the decoder reconstructs the feature embeddings of both available and unavailable visual instances from the available visual embeddings and the mask tokens. The basic framework is illustrated in Fig. 2.

3.1 MFGN Encoder

We take some instances from the same classes as the input that is analogous to the input image in the ViT [Dosovitskiy *et al.*, 2020], and here each instance is taken as a patch. The encoder performs as a feature extractor that maps the input instances to their corresponding feature embeddings. In this paper, we take the convolutional neural networks (e.g., ResNet12 and ConvNet4) as the encoder for fair comparison. We randomly sample a few of instances from one class and then feed them to the encoder to obtain their corresponding visual features. Then, the visual features of the several randomly sampled instances are masked and replaced with mask tokens. The remaining visual features and the mask tokens are fed into the decoder. In order to learn an efficient decoder, we follow the high masking ratio strategy applied in MAE [He *et al.*, 2021] and only feed one visible feature representation into the decoder to reconstruct the visual features of the masked instances. Such a high masking radio makes

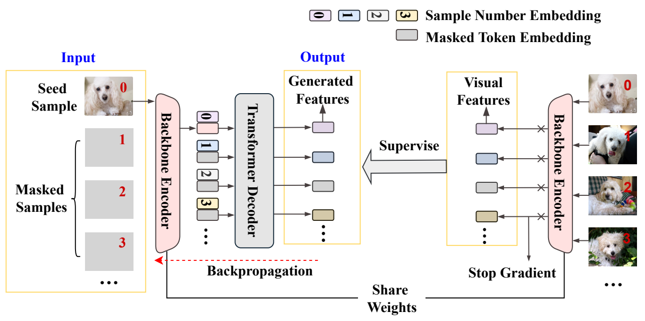


Figure 2: The basic framework of MFGN. The encoder maps an available instance to the latent feature embedding that performs as a seed. The decoder reconstructs the visual features of the masked instances with the seed, sample number embedding, and the mask token shared with all masked instances.

the reconstruction task difficult, thus enforcing the decoder to learn more meaningful patterns.

3.2 MFGN Decoder

The decoder in our framework is composed of a series of transformer blocks, which takes the feature embeddings of the unmasked visual instances, the sample number embedding, and the mask token as the input and produces the feature embeddings of both the masked and unmasked instances. The mask token is a shared, learned vector that indicates the presence of masked visual instances. We also add the sample number embeddings for all the instances. Each number embedding is a vector that indicates the item number of the instance.

In this way, the framework has endowed the capability of generating more visual features from the seed feature embedding, thus could be used for addressing FSL tasks, as introduced in 3.5.

3.3 Loss Functions

Instead of reconstructing the original pixel values of the input instances, our MFGN predicts the feature embeddings of the masked visual instances. Suppose that a seed (unmasked) instance and its corresponding masked instances from the same class are provided, we apply the popular mean squared error (MSE) as the loss function to measure the differences between the predicted feature embeddings and the real feature embeddings extracted from the encoder. The reconstruction loss function is formulated as:

$$L_{mse} = \sum_i \sum_{j=1}^{m} \|g(f(x_i), t, s_j) - f(x_i^j)\|_2^2, \quad (1)$$

where x_i denotes the seed instance; x_i^j denotes the j-th masked instance from the same class as x_i; t denotes the mask token, s_j is the sample number embedding of j; f and g denote the encoder and decoder, respectively.

In order to extract more discriminant feature representations, the whole framework is trained under the supervision of the ground-truth labels of the input instances. Thus, we also add the classification loss function to enforce the encoder to produce discriminant feature representations. Specifically, the classification loss function is formulated as:

$$L_{ce} = -\sum_i y_i \log \frac{e^{-h(f(x_i))}}{\sum_k e^{-h(f(x_k))}}, \quad (2)$$

where y_i is the class label of x_i; h denotes the classifier, which is a fully connected layer.

Furthermore, to ensure the reconstructed visual features to be discriminant, we also regularize the logits of the reconstructed visual features produced by the classifier to be close to those of the corresponding masked real feature embeddings, which is formulated as:

$$L_{reg} = \sum_i \sum_{j=1}^{m} \|h(g(f(x_i), t, s_j)) - h(f(x_i^j))\|_2^2. \quad (3)$$

Note that the logits of the masked real feature embeddings are detached, which means no gradients are back propagated. Using this regularizer improves the performances in our experiments, please see 4.4.

Overall, the objective of our approach is:

$$\arg\min L_{ce} + \lambda L_{mse} + \gamma L_{reg}, \quad (4)$$

where λ and γ are two hyperparameters. The pseudo code is provided in Alg. 1.

Algorithm 1 Pseudocode of MFGN in a PyTorch-like style.

```
# f: CNN Encoder, g: Transformer Decoder
# h: Classifier
# lamb, gamma: hyperparameter
# t: masked token, s: sample number embedding
for x_i, x_j, y in loader:
    # x_i, x_j are from class y
    # feature embedding
    f_i = f.forward(x_i)
    f_j = f.forward(x_j)
    [g_i, g_j] = g.forward([f_i+s_i],[t+s_j])
    # logit
    h_i, h_j = h.forward(f_i), h.forward(f_j)
    z_i, z_j = h.forward(g_i), h.forward(g_j)
    # loss
    l_ce = CE(h_i, y)
    l_mse = MSE(g_i-f_i)+MSE(g_j-f_j.detach())
    l_reg = MSE(z_i-h_i)+MSE(z_j-h_j.detach())
    # objective
    loss = l_ce + lamb*l_mse + gamma*l_reg
    loss.backward() # SGD update
    update f, g, h, t, s
```

3.4 Simple Implementation

Our MFGN is trained end-to-end from scratch and does not require fine-tuning during the inference, thus is efficient. During training, we randomly sample a batch of instances, each of which is seen as a seed (unmasked) instance. Meanwhile, we also sample a few instances for each seed instance from the same class as the masked visual instances. As noted, no gradients are back propagated through the masked visual instances. In other words, the masked visual instances are only used for supervision and are detached during training.

3.5 MFGN for FSL

In an FSL task, a support set consisting of N classes is provided, each class has K instances, the task is to predict the query instances from the same N classes with the classifiers derived from the support set. Once the proposed MFGN is trained, we obtain both a feature extractor (encoder) and a feature generator (decoder) and the combination of them can be used for augmenting more feature embeddins of the support classes. Specifically, the implementation has the following steps. First, each support sample is taken as the seed instance to feed the encoder to obtain the feature embedding (which can be directly used for classification, see 4.4). Second, both the seed feature embedding and the learned mask token, along with some random number embeddings are fed into the decoder to obtain a series of augmented feature embeddins. Finally, both the feature embedding of the seed instance and the augmented feature embeddins are used for predicting the query instances. In the experiments, each support class is represented as a prototype that is obtained by averaging the seed and the augmented feature embeddins from the same class. The nearest neighbor classifier is applied and the cosine distance is used for measuring the similarity between the query instances and the prototypes.

4 Experiments

In this section, we conduct a set of experiments to evaluate the proposed approach. We first document the benchmarks and the implementation details and then compare the proposed approach with the existing competitors. Finally, further analysis are provided.

4.1 Datasets

We have conducted experiments on four datasets that are Caltech UCSD Birds (CUB) [Wah *et al.*, 2011], StanfordDogs [Khosla *et al.*, 2011], miniImageNet [Vinyals *et al.*, 2016], and CIFAR-FS [Bertinetto *et al.*, 2018]. These datasets consist of collections of images comprising a varied set of categories in different scopes: birds, dogs, and objects respectively. Both CUB and StanfordDogs are fine-grained datasets, and both miniImageNet and CIFAR-FS are two coarse-grained datasets. The CUB dataset contains 11,788 images from 200 bird species. Following the split proposed in [Chen *et al.*, 2018], we divide the data into three disjoint sets, 100, 50, and 50 categories, respectively for training, validation, and test. The StanfordDogs consists of 120 dog breeds with a total number of 20,580 images. We follow [Li *et al.*, 2019a] and split the dataset into 70, 20, and 30 classes for training, validation, and test, respectively. The miniImageNet dataset is a subset of ImageNet, which consists of 100 classes with 600 images per class. Following the split proposed in [Ravi and Larochelle, 2016], we divide the dataset with 64, 16, and 20 classes for training, validation, and test, respectively. The CIFAR-FS is a few-shot classification dataset built on the CIFAR100. Following [Bertinetto *et al.*, 2018], we use 64/16/20 classes for training/validation/test, respectively.

4.2 Implementation Details

For fair comparisons with the other competitors, we conduct experiments with three popular architectures as the feature extractor, i.e., ConvNet4, ResNet12, and ResNet18. Specifically, ConvNet4 consists of four convolutional blocks, each of which consists of a convolutional layer, a batch normalization layer and a ReLU activation function, and a 2×2 max-pooling operation. The dimensionality of the visual features extracted with ConvNet4 is 64. ResNet12 consists of four residual blocks, each of which is composed of three convolutional layers with 3×3 kernels; a 2×2 max-pooling layer is applied after each of the first three blocks; a global average-pooling layer is on the top of the final block. The architecture of the ResNet18 is similar with that of the ResNet12 and consists of four residual blocks. Differently, the last two blocks are composed of four convolutional layers. The dimensionality of the feature embeddings extracted with both ResNet12 and ResNet18 is 640.

To train the model, we apply the SGD optimizer with a momentum of 0.9 and a weight decay of $5e^{-4}$. The batch size is set to 32. For the fine-grained datasets, we train the models with 80 epochs. The learning rate is initialized as 0.05 and decayed with a factor of 0.1 at 60 and 70 epochs, respectively. For the coarse-grained datasets, we train the models with 60 epochs. The learning rate is initialized as 0.05 and decayed with a factor of 0.1 at 40 and 50 epochs, respectively. In the training stage, we adopt the standard data augmentation strategy as [Lee *et al.*, 2019], including random crop, color jittering, and random horizontal flip. The images from CUB-200-2011, StandfordDogs, and miniImageNet are resized as 84×84 and the size of the images from CIFAR-FS is 32×32.

To evaluate the model, we conduct 600 random few-shot classification tasks and report the average classification accuracy and 95% confidence interval. If not specific, both λ and γ are set to 1; the number of synthesized visual features is set to 5 for all datasets.

4.3 Results

Results on fine-grained datasets. Table 1 reports the 5-Way 1-Shot and 5-Way 5-Shot classification results on both CUB and StanfordDogs datasets with the three different architectures. From the results, we reach the following conclusions. First, the proposed MFGN performs competitively on the fine-grained datasets under different few-shot classification tasks. When comparing with the models trained with the ConvNet4 architecture, our MFGN performs the best on both datasets under different few-shot tasks. Specifically, MFGN achieves 69.36% and 60.22% 5-Way 1-Shot classification accuracy on CUB and StanfordDogs, respectively beats the second-best results with 3.07% and 5.38%. MFGN also obtains 0.41% and 2.83% improvements on the CUB and StanfordDogs under the 5-Way 5-Shot task, respectively. When comparing the models trained with the ResNet18 architecture, our approach improves significant margins than the competitors that are both feature augmented approaches. When comparing the models trained with the ResNet12 architecture, our MFGN performs much better than the competitors on both datasets under different tasks. Specifically, MFGN obtains respectively 5.43% and 6.22% improvements to the second-best result under the 5-Way 1-Shot classification task on CUB and StanfordDogs, which is significant. Second, the results of both ResNet12 and ResNet18 are much

Architecture	Method	CUB		StanfordDogs	
		1-Shot	5-Shot	1-Shot	5-Shot
ConvNet4	MatchingNet [Vinyals *et al.*, 2016]	45.30 ± 1.03	59.50 ± 1.01	35.80 ± 0.99	47.50 ± 1.03
	ProtoNet [Snell *et al.*, 2017]	37.36 ± 1.00	45.28 ± 1.03	37.59 ±1.00	48.19 ± 1.03
	RelationNet [Sung *et al.*, 2018]	58.99 ± 0.52	71.20 ± 0.40	43.29 ± 0.46	55.15 ± 0.39
	MAML [Finn *et al.*, 2017]	58.13 ± 0.36	71.51 ± 0.30	44.84 ± 0.31	58.61 ± 0.30
	adaCNN [Munkhdalai *et al.*, 2018]	56.76 ± 0.50	61.05 ± 0.44	42.16 ± 0.43	54.12 ± 0.39
	CovaMNet [Li *et al.*, 2019b]	52.42 ± 0.76	63.76 ± 0.64	49.10 ± 0.76	63.04 ± 0.65
	DN4 [Li *et al.*, 2019a]	46.84 ± 0.81	74.92 ± 0.64	45.41 ± 0.76	63.51 ± 0.62
	LRPABN [Huang *et al.*, 2020]	63.63 ± 0.77	76.06 ± 0.58	45.72 ± 0.75	60.94 ± 0.66
	MattML [Zhu *et al.*, 2020]	66.29 ± 0.56	80.34 ± 0.30	54.84 ± 0.53	71.34 ± 0.38
	MFGN (Ours)	**69.36 ± 0.43**	**80.75 ± 0.34**	**60.22 ± 0.45**	**73.17 ± 0.35**
ResNet18	Δ-encoder [Schwartz *et al.*, 2018]	69.80 ± 0.46	82.60 ± 0.35	68.59 ± 0.53	78.60 ± 0.78
	AFHN [Li *et al.*, 2020]	70.53 ± 1.01	83.95 ± 0.63	-	-
	MFGN (Ours)	**83.67 ± 0.37**	**91.29 ± 0.20**	**76.41 ± 0.44**	**86.14 ± 0.26**
ResNet12	Baseline++ [Chen *et al.*, 2018]	68.46 ± 0.85	81.02 ± 0.46	58.30 ± 0.35	73.77 ± 0.68
	MAML [Finn *et al.*, 2017]	71.11 ± 1.00	82.08 ± 0.72	66.56 ± 0.66	79.32 ± 0.35
	MatchingNet [Vinyals *et al.*, 2016]	72.62 ± 0.90	84.14 ± 0.50	65.87 ± 0.81	80.70 ± 0.42
	ProtoNet [Snell *et al.*, 2017]	71.57 ± 0.89	86.37 ± 0.49	65.02 ± 0.92	83.69 ± 0.48
	Δ-encoder [Schwartz *et al.*, 2018]	73.91 ± 0.87	85.60 ± 0.62	68.59 ± 0.53	78.60 ± 0.78
	MTL [Sun *et al.*, 2019]	73.31 ± 0.92	82.29 ± 0.51	54.96 ± 1.03	68.76 ± 0.65
	MetaOptNet [Lee *et al.*, 2019]	75.15 ± 0.46	87.09 ± 0.30	65.48 ± 0.49	79.39 ± 0.25
	HGNN [Yu *et al.*, 2022]	78.58 ± 0.20	90.02 ± 0.12	-	-
	MFGN (Ours)	**84.01 ± 0.39**	**91.85 ± 0.21**	**74.81 ± 0.44**	**86.52 ± 0.26**

Table 1: 5-Way 1-Shot and 5-Way 5-Shot classification accuracy (%) and 95% confidence interval on the CUB and Stanford Dogs datasets. All the results are from the exiting literature. The best performance is marked in bold.

Arch	Method	miniImageNet	CIFAR-FS
ConvNet4	ProtoNet	49.42 ± 0.78	55.5 ± 0.7
	RelationNet	50.44 ± 0.82	55.0 ± 1.0
	R2D2	51.20 ± 0.60	65.3 ± 0.2
	DC	54.60 ± 0.60	-
	ATLNet	54.30 ± 0.76	-
	RFS	56.32 ± 0.58	63.5 ± 0.8
	HGNN	55.63 ± 0.20	-
	MFGN (Ours)	**57.48 ± 0.37**	**66.1 ± 0.4**
ResNet18	Δ-encoder	59.90 ± n/a	66.7 ± n/a
	AFHN	62.38 ± 0.72	68.3 ± 0.9
	MFGN (Ours)	**65.97 ± 0.40**	**75.0 ± 0.5**
ResNet12	ProtoNet	60.37 ± 0.83	72.2 ± 0.7
	MetaOptNet	62.64 ± 0.61	72.6 ± 0.7
	Shot-Free	59.04 ± 0.43	69.2 ± 0.4
	MTL	61.20 ± 1.80	-
	RFS	63.59 ± 0.61	73.1 ± 0.8
	HGNN	67.02 ± 0.20	-
	MFGN (Ours)	**67.25 ± 0.36**	**76.3 ± 0.5**

Table 2: 5-Way 1-Shot classification results (%) on both miniImageNet and CIFAR-FS datasets. All the results are from the exiting literature. The best results are marked in bold.

better than the low capacity ConvNet4 architecture, which indicates the important role of the feature embeddings in the FSL tasks.

Results on coarse-grained datasets. Table 2 demonstrates the results of both miniImageNet and CIFAR-FS datasets with three different architectures under the 5-Way 1-Shot classification task. We select twelve competitors for comparison. They are ProtoNet [Snell *et al.*, 2017], RelationNet [Sung *et al.*, 2018], R2D2 [Bertinetto *et al.*, 2018], DC [Yang *et al.*, 2020], ATLNet [Dong *et al.*, 2021], RFS [Tian *et al.*, 2020], MetaOptNet [Lee *et al.*, 2019], Shot-Free [Ravichandran *et al.*, 2019], MTL [Sun *et al.*, 2019], MetaOptNet [Lee *et al.*, 2019], AFHN [Li *et al.*, 2020], and HGNN [Yu *et al.*, 2022].

From the results, we observe that the proposed MFGN also performs very competitively on the coarse-grained datasets. When comparing with the models trained with the ConvNet4, MFGN achieves 57.48% and 66.1% on miniImageNet and CIFAR-FS datasets, respectively improves the second best competitors with 1.16% and 0.8%. When comparing with the models trained with the ResNet18, MFGN beats the second best competitors with a large margin on both miniImageNet and CIFAR-FS datasets. When comparing with the models trained with the ResNet12, MFGN also achieves the best on both datasets, which has 0.23% and 3.2% accuracy boost over the second best competitors on miniImageNet and CIFAR-FS, respectively. The experimental results on both fine-grained and coarse-grained datasets reflect that the generalization ability of MFGN is better than most of the competitors in dealing with the classification tasks with extremely limited data.

4.4 Further Analysis

Impacts of training class number. To evaluate the impacts of training class number to the few-shot classification performances, we train the model with the train classes and the combination of both train and validation classes. We report the comparison results in Table 3. From the results, we conclude that training with more classes is benefit in improving

Arch	Model	CUB	Dogs	mini	CIFAR-FS
ConvNet4	Train	67.65	58.68	56.67	65.78
	Train+val	69.36	60.22	57.48	66.13
ResNet12	Train	79.68	73.61	66.56	75.82
	Train+val	84.01	74.81	67.25	76.34

Table 3: Results (%) of 5-Way 1-Shot classification on four datasets with different number of training classes. Dogs and mini are short for StanfordDogs and miniImageNet, respectively.

Data	Task	Number of synthesized visual features					
		0	1	2	3	4	5
CUB	1-Shot	66.07	69.71	69.25	69.63	69.48	69.36
	5-Shot	78.81	80.65	80.32	80.43	80.62	80.75
Dogs	1-Shot	58.59	60.47	60.94	60.66	60.53	60.22
	5-Shot	72.55	73.43	73.57.	73.26	73.47	73.17

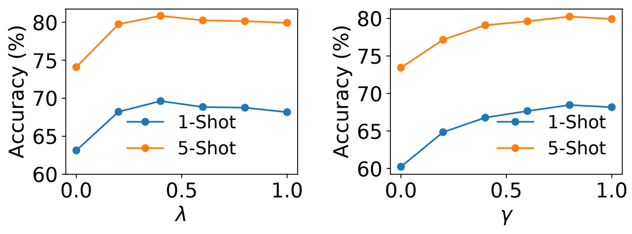
Table 4: Evaluation (%) of the number of synthesized visual features on both CUB and Dogs datasets under 5-Way classification tasks.

the FSL performances. This is reasonable due to that more training classes would provide more information and transferable patterns for the novel classes.

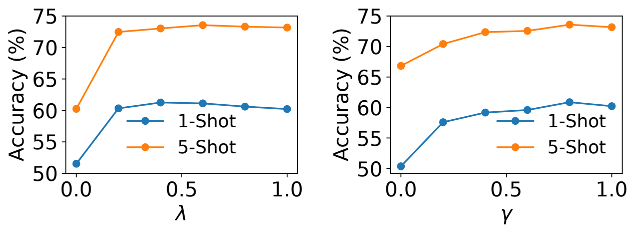
Impacts of loss functions. In this subsection, we evaluate the impacts of different loss functions. Specifically, we fix γ as 1 and range λ from 0 to 1 with an interval of 0.2 to evaluate the impacts of L_{mse}. Similarly, we fix λ as 1 and range γ from 0 to 1 with an interval of 0.2 to evaluate the impacts of L_{reg}. Fig. 3 shows the FSL results of both CUB and StanforDogs datasets with different λ and γ with ConvNet4. From the results, we observe that the performances are the best when λ is 0.4 and γ is 0.8 for both datasets and conclude that both L_{mse} and L_{ref} bring significant improvements to the classification performances for both datasets.

Impacts of augmented visual features. In this subsection, we evaluate the impacts of the augmented visual features to the few-shot classification during test. Table 4 reports the 5-Way classification results on CUB and Dogs datasets varying the number of augmented visual features with ConvNet4. From the results, we observe that the results obtained with the augmented visual features are better than those without the augmented visual features, which indicates the augmented visual features indeed provide nontrivial information for the few-shot classification. Besides, we observe that more synthesized visual features would bring little benefits in the performance improvement. In the experiments, we report the results with 5 augmented visual features.

Fig. 4 shows the visualization of the feature embeddings of a random test task from the CUB dataset obtained with the models trained with and without the decoder. From the results, we observe that the decision boundaries of the model with the decoder are sharper than that without the decoder, which indicates the decoder also boosts the discriminative of the model.



(a) Results on the CUB with different λ and γ



(b) Results on the StanfordDogs with different λ and γ

Figure 3: Impacts of hyperparameters on both CUB and StanfordDogs.

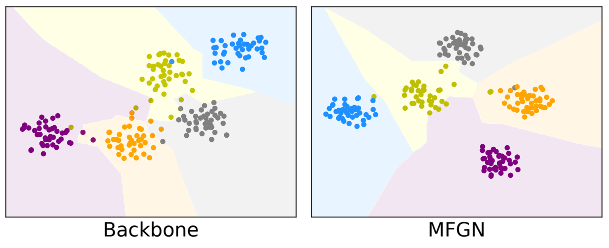


Figure 4: tSNE visualization of a FSL test task from CUB dataset. For Baseline (without decoder), the decision boundaries are produced with the feature embeddings of original 5 samples from each class. For MFGN, the decision boundaries are produced with both original and generated features.

5 Conclusion

In this paper, we have proposed a feature-augmentation approach called MFGN for FSL. MFGN addresses FSL via learning both discriminant and diverse feature embeddings with an encoder-decoder framework. Though straightforward it is, MFGN achieves quite promising FSL results on both fine-grained and coarse-grained datasets. Besides, based on the observations from the experiments, we conclude that intra-class feature variation is important for FSL and our method implemented with a transformer could effectively capture the intra-class variations.

Acknowledgments

This work is supported in part by the Key R&D Program of Zhejiang Province, China (2021C01119), the National Natural Science Foundation of China under Grant (62002320, U19B2043, 61672456), the Fundamental Research Funds for the Central Universities (2020QNA5010).

References

[Bertinetto *et al.*, 2018] Luca Bertinetto, Joao F Henriques, Philip Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. In *ICLR*, 2018.

[Cai *et al.*, 2018] Qi Cai, Yingwei Pan, Ting Yao, Chenggang Yan, and Tao Mei. Memory matching networks for one-shot image recognition. In *CVPR*, pages 4080–4088, 2018.

[Chen *et al.*, 2018] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. In *ICLR*, 2018.

[Dong *et al.*, 2021] Chuanqi Dong, Wenbin Li, Jing Huo, Zheng Gu, and Yang Gao. Learning task-aware local representations for few-shot learning. In *IJCAL*, pages 716–722, 2021.

[Dosovitskiy *et al.*, 2020] Alexey Dosovitskiy, Lucas Beyer, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2020.

[Finn *et al.*, 2017] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, pages 1126–1135, 2017.

[He *et al.*, 2021] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. *arXiv:2111.06377*, 2021.

[Hou *et al.*, 2019] Ruibing Hou, Hong Chang, Bingpeng Ma, Shiguang Shan, and Xilin Chen. Cross attention network for few-shot classification. In *NeurIPS*, pages 4003–4014, 2019.

[Huang *et al.*, 2020] Huaxi Huang, Junjie Zhang, Jian Zhang, Jingsong Xu, and Qiang Wu. Low-rank pairwise alignment bilinear network for few-shot fine-grained image classification. *IEEE TMM*, 2020.

[Khosla *et al.*, 2011] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Fei-Fei Li. Novel dataset for fine-grained image categorization: Stanford dogs. In *CVPR Workshop*, 2011.

[Lee *et al.*, 2019] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *CVPR*, pages 10657–10665, 2019.

[Li *et al.*, 2019a] Wenbin Li, Lei Wang, Jinglin Xu, Jing Huo, Yang Gao, and Jiebo Luo. Revisiting local descriptor based image-to-class measure for few-shot learning. In *CVPR*, pages 7260–7268, 2019.

[Li *et al.*, 2019b] Wenbin Li, Jinglin Xu, Jing Huo, Lei Wang, Yang Gao, and Jiebo Luo. Distribution consistency based covariance metric networks for few-shot learning. In *AAAI*, pages 8642–8649, 2019.

[Li *et al.*, 2020] Kai Li, Yulun Zhang, Kunpeng Li, and Yun Fu. Adversarial feature hallucination networks for few-shot learning. In *CVPR*, pages 13470–13479, 2020.

[Munkhdalai *et al.*, 2018] Tsendsuren Munkhdalai, Xingdi Yuan, Soroush Mehri, and Adam Trischler. Rapid adaptation with conditionally shifted neurons. In *ICML*, pages 3664–3673, 2018.

[Ravi and Larochelle, 2016] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2016.

[Ravichandran *et al.*, 2019] Avinash Ravichandran, Rahul Bhotika, and Stefano Soatto. Few-shot learning with embedded class models and shot-free meta training. In *ICCV*, pages 331–339, 2019.

[Schwartz *et al.*, 2018] Eli Schwartz, Leonid Karlinsky, Joseph Shtok, Sivan Harary, Mattias Marder, Abhishek Kumar, Rogerio Feris, Raja Giryes, and Alex M Bronstein. δ-encoder: an effective sample synthesis method for few-shot object recognition. In *NeurIPS*, pages 2850–2860, 2018.

[Snell *et al.*, 2017] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *NeurIPS*, pages 4080–4090, 2017.

[Sun *et al.*, 2019] Qianru Sun, Yaoyao Liu, Tat-Seng Chua, and Bernt Schiele. Meta-transfer learning for few-shot learning. In *CVPR*, pages 403–412, 2019.

[Sung *et al.*, 2018] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *CVPR*, pages 1199–1208, 2018.

[Tian *et al.*, 2020] Yonglong Tian, Yue Wang, Dilip Krishnan, Joshua B Tenenbaum, and Phillip Isola. Rethinking few-shot image classification: a good embedding is all you need? In *ECCV*, pages 266–282, 2020.

[Vinyals *et al.*, 2016] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *NeurIPS*, pages 3630–3638, 2016.

[Wah *et al.*, 2011] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. http://www.vision.caltech.edu/datasets/cub_200_2011, 2011.

[Xing *et al.*, 2019] Chen Xing, Negar Rostamzadeh, Boris Oreshkin, and Pedro O Pinheiro. Adaptive cross-modal few-shot learning. *NeurIPS*, 32:4847–4857, 2019.

[Xu *et al.*, 2021] Chengming Xu, Yanwei Fu, Chen Liu, Chengjie Wang, Jilin Li, Feiyue Huang, Li Zhang, and Xiangyang Xue. Learning dynamic alignment via meta-filter for few-shot learning. In *CVPR*, pages 5182–5191, 2021.

[Yang *et al.*, 2020] Shuo Yang, Lu Liu, and Min Xu. Free lunch for few-shot learning: Distribution calibration. In *ICLR*, 2020.

[Yu *et al.*, 2022] Tianyuan Yu, Sen He, Yi-Zhe Song, and Tao Xiang. Hybrid graph neural networks for few-shot learning. In *AAAI*, 2022.

[Zhu *et al.*, 2020] Yaohui Zhu, Chenlong Liu, and Shuqiang Jiang. Multi-attention meta learning for few-shot fine-grained image recognition. In *IJCAI*, pages 1090–1096, 2020.