Few-Shot Learning via Feature Hallucination with Variational Inference

Qinxuan Luo[1, 2] luoqinxuan2018@ia.ac.cn

Lingfeng Wang[1, 3] lfwang@nlpr.ia.ac.cn

Jingguo Lv[4] lvjingguo@bucea.edu.cn

Shiming Xiang[1, 2] smxiang@nlpr.ia.ac.cn

Chunhong Pan[1] chpan@nlpr.ia.ac.cn

[1]NLPR, Institute of Automation, Chinese Academy of Sciences

[2]School of Artificial Intelligence, University of Chinese Academy of Sciences

[3]Key Laboratory of Knowledge Automation for Industrial Processes, Ministry of Education

[4]School of Geomatics and Urban Spatial Informatics, Beijing University of Civil Engineering and Architecture

Abstract

Deep learning has achieved huge success in the field of artificial intelligence, but the performance heavily depends on labeled data. Few-shot learning aims to make a model rapidly adapt to unseen classes with few labeled samples after training on a base dataset, and this is useful for tasks lacking labeled data such as medical image processing. Considering that the core problem of few-shot learning is the lack of samples, a straightforward solution to this issue is data augmentation. This paper proposes a generative model (VI-Net) based on a cosine-classifier baseline. Specifically, we construct a framework to learn to define a generating space for each category in the latent space based on few support samples. In this way, new feature vectors can be generated to help make the decision boundary of classifier sharper during the fine-tuning process. To evaluate the effectiveness of our proposed approach, we perform comparative experiments and ablation studies on mini-ImageNet and CUB. Experimental results show that VI-Net does improve performance compared with the baseline and obtains the state-of-the-art result among other augmentation-based methods.

1. Introduction

In recent years, deep learning [20] has made great progress in the computer vision community. Deep networks, especially Convolution Neural Networks (CNN), are playing important roles in various visual tasks such as recognition [19][32][37][13], detection [29][1][22] and segmentation [44][2][12]. It is undeniable that deep learning is the main trend now in the field of artificial intelligence. However, the excellent performance of deep learning are heavily dependent on the amount of training data while collecting enough labeled data is generally high-cost, and this problem severely limits its application in practice. Although models can be pre-trained with open-source datasets, the large domain gap between datasets cannot be ignored. Few-shot learning [7][40] has therefore been proposed to alleviate such problems.

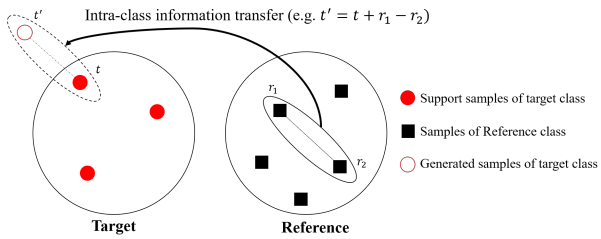

Figure 1. When the intra-class information is directly transferred from the reference category to the target one, the generated sample may be farther away from the distribution center in some cases.

Generally, few-shot learning aims to enable machines to learn the concepts from limited labeled data to reduce the dependence on training data. In the common setting of few-shot learning, a single task ought to contain a support set and a query set with shared categories, and the model expects to predict the category of query samples according to the support set. A task is usually called the N-way-K-shot task when its support set only contains N categories with K samples per class. This way to construct tasks accords with the motivation mentioned above and has been applied in most studies on few-shot learning.

In the research of few-shot learning, the main challenge is how to seek an appropriate feature space and a classifier to quickly adapt to a novel domain with a small number

of labeled samples. Some works attempt to learn a good metric [39][33][36] and some construct meta-learners to improve adaptive capacity [28][8]. Considering that the critical problem of few-shot learning is the lack of labeled data, augmentation-based approaches are naturally effective and straightforward in theory. In fact, there exists many methods based on this idea, and some of them try to transfer intra-class information from the reference categories to the target ones [31][11][3]. As shown in Figure 1, in this augmentation mode, the generated sample may be farther from the center of the distribution because the direction of transfer is hard to control.

In this paper, we propose a generative model named VI-Net ,which aims at defining a generating space for each category in the latent space that satisfies the regular distribution and then generating new feature vectors based on it. The generated features can be used to extend support samples per class and finally serve for fine-tuning classifier. It is worth mentioning that VI-Net does not need external data from other categories during the generation process.

The main contributions of our work are as follows:

1) We constrain the mapping of the samples belonging to the same category in the low-dimensional latent space within a Gaussian distribution by variational inference;

2) We propose a generation strategy based on random sampling within Gaussian distribution in latent space;

3) Our work is tested on two benchmark datasets mini-ImageNet and CUB [42] by comparison and ablation studies, and the results prove the effectiveness of our method.

2. Related Work

2.1. Metric Learning

The core idea of metric learning is to transform the process of recognition into the comparison of similarity between features. This kind of methods expects to find a feature space where samples of the same category are clustered. Typically, they utilize metrics, such as Euclidean distance, to measure similarities between samples.

Siamese Network [18] used parameter-shared CNNs to extract features from a pair of images and determines if they are of the same class based on distance. Matching Network [39] adopted LSTM [14] to encode features based on the relationship between support images and query images. Prototypical Network [33], a classical metric-learning method, calculated the average of embedding features of each class as representation and measured the similarity by Euclidean distance. Besides using fixed metrics, Relation Network [36] designed a learnable metric module to calculate the relation score. The above methods focused on learning feature space or similarity metric. TADAM [26] introduced a TEN network to dynamically adjust parameters of feature extractor based on task representation. Wertheimer and Hariharan [43] used the attention module to reduce negative effects from complex background.

From another point of view, GNN [17] is a practical tool to model the relationship between samples. Garcia and Bruna [9] firstly converted discrete points to a graph, where nodes represented features with one-hot labels and edges represented hidden relationships. The predictions could be extracted from nodes after the propagation of the graph. EGNN [15] took a more direct definition of graphs, it used edges to represent the intra/inter-class relations between connected nodes. TPN [23] was a special graph method, it utilized label propagation to predict class and performed transductive learning to improve the accuracy rate.

2.2. Meta-Learning

Meta-learning [38] based approaches construct meta-learners and meta-tasks to make models achieve faster adaption to novel tasks. The significant commonality of these methods is to make hyper-parameters learnable.

Ravi and Larochelle [28] proposed an LSTM-based learnable optimizer, which used loss value and partial derivatives as input, to replace traditional parameter optimizations. MAML [8] provided a very classic meta-learning method for few-shot learning. In MAML, the initial parameter of the model was the main training object, which was updated after fine-tuning based on several meta tasks. LEO [30] and MTL [35] were all inspired by MAML. LEO constructed a low-dimensional space for parameters while MAML updated parameters directly in high-dimensional space. MTL divided weights of CNN into two parts, one of which was frozen after pre-training and the other participated in meta-learning.

2.3. Data Augmentation

The critical limitation of few-shot learning is the lack of training data, hence data augmentation is a direct and effective means. Different from image translation, such methods do not need to generate realistic images, and how to guarantee the authenticity of the generated features is the main argument instead.

Wang et al. [41] considered that generated features just need to be useful for class distinctions and designed a simple hallucinator with noise and support samples as input, which could be meta-trained jointly with the classifier. AFHN [21] added WGAN [24] on this basis and utilized multiple regularizers to constraint the generator. Another solution is to hallucinate novel images with the help of external datasets, in other words, is to transform relationships between external samples to the target ones. Delta-Encoder [31] expected to extract inter-class deformation between two classes based on conditional autoencoder, and used reconstruction loss to constraint encoder and decoder. Hariharan and Girshick [11] proposed a kind of hallucinator

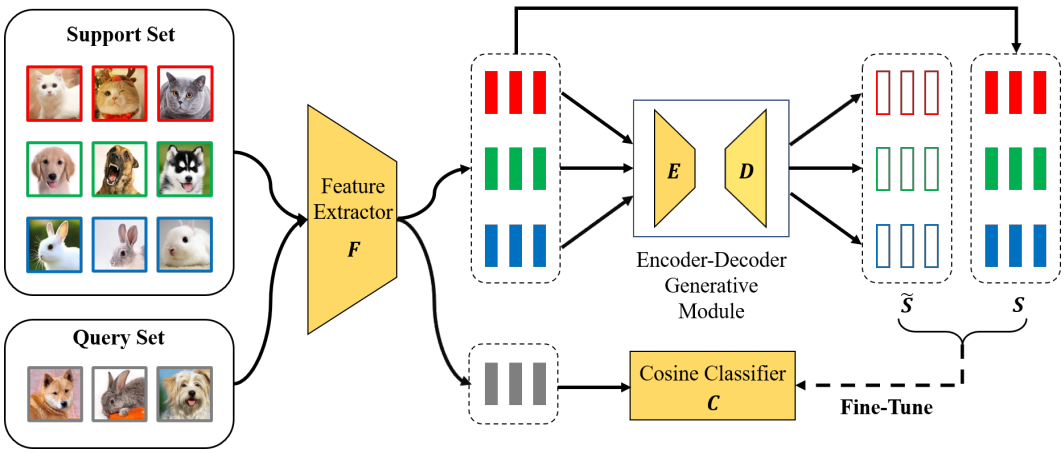

Figure 2. **Framework of VI-Net.** This illustration describes the data flow during a 3-shot-3-way task. After given a set of $\{S, Q\}$, VI-Net feeds all samples into feature extractor and obtains feature embeddings. Considering only support samples are labeled, VI-Net divides support features into several clusters according to their categories and then utilizes an encoder-decoder generative module to augment the support set. At last, the combined support set $S \cup \tilde{S}$ are utilized to fine-tune the cosine-classifier before recognizing query samples.

that is similar to Delta-Encoder. Instead of inter-class deformation, they used intra-class deformation from a pair of images of other class to cause hallucination. DTN [3] improved on this basis, it firstly mapped feature vectors into low-dimensional space and then fused features using addition and subtraction operation such as $A+(B-C)$, and the final generated features could be decoded from them. Different from the above methods based on relationships between features, IDeMe-Net [5] directly combined two images using learnable weights after dividing them into several patches. SalNet [45] employed saliency detection to extract foreground and background features from images respectively, and exchanged them with each other to compose new features.

VI-Net is mainly inspired by variational FSL [46], which used variational inference to predict the distribution of samples. Differently, it expects to construct a sampling space for each category directly based on few samples instead of extracting the relationship of external sample pairs. Gaussian distribution parameters of each category can be gained by a specific fusion strategy, and work for generation finally.

3. Method

3.1. Few-Shot Learning Setup

Considering the particularity of few-shot tasks, the conventional definition mode of tasks no longer applies. In few-shot learning, a sufficient labeled dataset D_{base} is supplied when training the model. However, during the process of inference, such a model has to work on a dataset D_{novel} which is disjoint from D_{base} completely. Different from transfer learning, a few-shot inference task samples a limited subset from D_{novel} every time and divides it into a support-set S and a query-set Q which shares categories, and aims to correctly recognize query samples from Q just based on S directly. It means that the model has to adapt useful knowledge for each subtask using very few samples. Vinyals et al. [39] proposed a standard N-way-K-shot classication scenario to define the format of S as $\{x_1^c, \cdots, x_K^c; y_c\}_{c=1}^N$. This definition is widely used in the study of few-shot learning, which properly shows the characteristics of tasks.

3.2. Cosine-Similarity based Baseline

Chen et al. [4] have shown that distance-based classifiers can also perform well just with the convolution networks. Here we choose a cosine classifier C_ω as our predictor and extract features using convolution network f_θ, where ω and θ mean parameters of networks. Given an image x, the similarity score of the i-th class could be computed as:

$$s_i = \cos\left(f_\theta(x), \omega_i\right) = \frac{f_\theta(x) \cdot \omega_i}{\|f_\theta(x)\| \cdot \|\omega_i\|}, \tag{1}$$

where ω_i means the weight vector of the i-th class. In the process of prediction, the calculating formula of possibility can be expressed as:

$$p(y=i|x) = \frac{e^{\gamma s_i}}{\sum_{j=1}^N e^{\gamma s_j}}, \tag{2}$$

where N is the number of categories and γ represents scale factor. Following this method, a basic classification model can be easily constructed, and the target of pre-training is just to minimize the cross-entropy loss L_{cla} over images from D_{base}:

$$L_{cla} = \mathbb{E}_{(x,y)\sim D_{base}} \frac{1}{N} \sum_{j=1}^{N} -y \log p(y=j|x). \quad (3)$$

Considering the purpose of few-shot learning, the pre-trained extractor needs to perform on D_{novel}. The most common approach is to fine-tune a new classifier based on support-set S during the testing process, and the prediction process is similar to the commoin classification process.

As shown in section 4.2, this baseline can obtain a similar result with many classical models without external modules. By observing the fine-tuning operation during the testing stage, it is found that the limitation of data is easy to cause overfitting. To this end we propose a hallucination method to generate diverse and effective samples that can make the decision boundary of the classifier sharper.

3.3. Feature Hallucination based Method

3.3.1 Overview of Network

The overall structure of VI-Net is shown in Figure 2. It mainly contains four modules: feature extractor (F), distribution encoder (E), feature decoder (D) and classifier (C). Given a set of samples $X = \{S, Q\}$, where $S = \{(x_i, y_i)|i = 1, 2, \cdots, N \times K\}$ means support set and $Q = \{x_i|i = 1, 2, \cdots, N_q\}$ means query set, feature vectors can be extracted by F directly ($f = F(x) \in \mathbb{R}^H$). As mentioned above, there are only K samples of each category can be used to adjust the parameters of the N-class classifier while all categories are novel for the model. Thus we feed support features into E to establish distribution in L-dimensional latent space, where $L \ll H$, and utilize D to generate abundant new features $\tilde{f}$ based on distribution information for each category. At last, C could be fine-tuned by all combined features set $f_s \cup \tilde{f}_s$ and used to complete the recognition task on Q. Obviously, if new samples can express category characteristics and possess both diversity and discriminability, they may effectively alleviate the problem of underfitting due to the lack of samples. For convenience, we directly use x_i to represent features in the following.

3.3.2 Feature Hallucination Module

Zhang et al. [46] proposed that the distributions of categories can be predicted in embedding space by variational inference. Instead of directly predicting categories in latent space by probability density, we take the distribution in latent space as the basis to generate a certain class of features. To achieve this goal, the mapping of samples in latent space should be guided to meet the following conditions: 1) Latent vectors belonging to the same category have clustering characteristics and can be described using regular distribution; 2) Latent vectors sampled from distribution can be recognized by the classifier after being mapped back to the feature space.

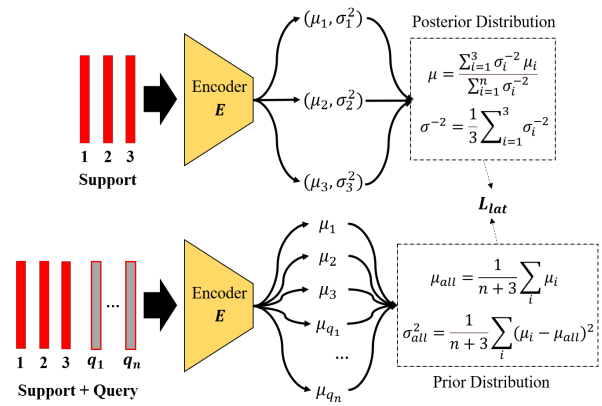

Figure 3. **Encoder to extract distribution.** During the base-training stage, we suppose that the labels of the query samples are known and get a larger set of each class containing the support set and query set at the same time. For a 3-shot task, there are 3 support samples and n query samples for red class, and posterior distribution as well as prior distribution can be obtained by an encoder based on the support set and the larger set.

Encoder to extract distribution. As mentioned in variational FSL [46], Gaussian distribution is very suitable for describing the distribution of samples. However, in high dimensional feature space, the distribution of feature vectors is complex and hard to accurately predict by few samples. Hence we construct a low-dimensional latent space and suppose that the mapped latent vectors of the same class satisfy multivariate Gaussian distribution $N(\mu, \Sigma)$. To simplified the form of the parameter, Σ is represented by L-dimensional vector σ^2 that means components in latent vectors are independent.

According to the parameter estimation method based on the maximum likelihood, the parameters of the specified distribution can be estimated based on samples, and the larger the sample set, the more accurate the result. It is known that in the training stage the number of query samples is unlimited and all labels are available, so the mapping of a large number of samples including both support and query samples in the latent space can be used to estimate a set of parameters as the prior distribution for every category. For multivariate Gaussian distribution with independent components, the calculation formulas of two key parameters are as follows:

$$\mu_c^{all} = \frac{1}{K + N_q} \sum_{i=1}^{K+N_q} x_i, \quad (4)$$

$$\sigma_c^{2\ all} = \frac{1}{K + N_q} \sum_{i=1}^{K+N_q} (x_i - \mu_c^{all})^2, \quad (5)$$

where μ_c^{all} and $\sigma_c^{2\ all}$ respectively represent the mean and variance in the prior distribution of class c. But in Few-shot Learning tasks, the available samples of the same category are very few. Thus we directly use an encoder to predict distribution by a single sample in turn and then fuse the distribution information according to the category. The encoder is a simple fully-connected layer that uses support samples as input and directly outputs the distribution parameters $(\mu_c, \sigma_c^2) \in \mathbb{R}^L$. There is no doubt that the predicted result of a single sample is unreliable. Therefore, after given $\{(\mu_i, \sigma_i^2)\}_{i=1}^K$ predicted by K samples of the same class, the posterior distribution parameters of this category can be computed as:

$$\mu_c = \frac{\sum_{i=1}^K \sigma_i^{-2} \mu_i}{\sum_{i=1}^K \sigma_i^{-2}}, \tag{6}$$

$$\sigma_c^{-2} = \frac{1}{K} \sum_{i=1}^K \sigma_i^{-2}. \tag{7}$$

Figure 3 shows the total process of parameter processing in latent space when training. To ensure that vectors in latent space can effectively reflect the distribution of each category and help the decoder exactly generate useful feature vectors, the following loss function is designed to constraint latent space:

$$L_{lat} = \frac{1}{N} \sum_{c=1}^N KL(q_\varphi(z_c|S_c) \| p_\theta(z_c|S_c, Q_c)), \tag{8}$$

where $q_\varphi(z_c|S_c)$ represents the posterior distribution of class c in latent space based on support samples, and $p_\theta(z_c|S_c, Q_c)$ stands for the assumed prior distribution of class c based on both support and query samples. Naturally, the posterior distribution is supposed to approximate the prior by Kullback-Leibler divergence (KL).

Decoder to generate new features. Follow the above encoder, features of the same class $\{(x_i, y_i)|y_i = c\}$ can be mapped to a pair of distribution parameters $(\mu_c, \sigma_c^2) \in \mathbb{R}^L$. Based on the training target described in section 3.3.2, latent vectors $z_i^c \sim N(\mu_c, \sigma_c^2)$ sampled from the latent space have to make sure that generated samples are meaningful. As shown in Figure 4, feature vectors can be directly generated by decoder D with latent vectors z_i^c as input. To ensure that the generated samples have discriminability, the classifier trained during the pre-training stage is set to compute the loss L_{cla} on new features.

It must be noted that there is no constraint to ensure that new features are close to real features in feature space and generated features would stay away from real features although can be correctly recognized. To avoid this situation, the reconstruction loss between features decoded directly

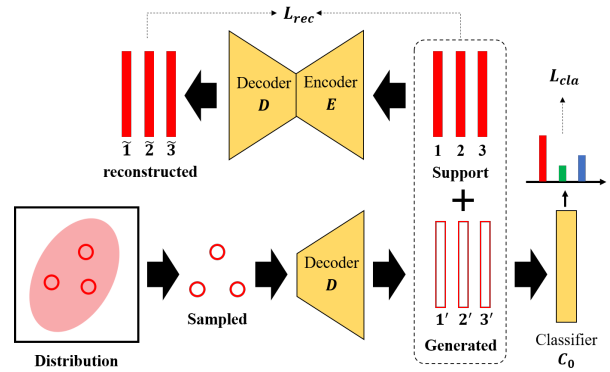

Figure 4. **Decoder to generate new features.**. Given distribution information encoded from the encoder, latent vectors can be randomly sampled from distribution and then mapped back to feature space through the decoder. The augmented set consisting of both support set and generated set can be used to fine-tune the classifier. During the base-training stage, support samples need to be fed into the generative module to be reconstructed to compute L_{rec} and generated samples have to be fed into the pre-trained classifier C_0 to compute L_{cla}.

from the means vectors μ_i and the corresponding support samples is necessary:

$$L_{rec} = \frac{1}{N \times K} \sum_{c=1}^N \sum_{i=1}^K \|D(\mu_i^c) - x_i^c\|, \tag{9}$$

where x_i^c means the i-th samples of class c and $\mu_i^c = E^\mu(x_i^c)$. With this loss, features from the decoder have to reflect the link between latent space and feature space.

3.3.3 Algorithm Analysis: Variational Inference

In our generation process, the most important question is how to infer the distribution in the latent space. Obviously, variational inference is a suitable approximating approach, which has been fully discussed in VAE [16][34]. Given a set of $\{S, Q\}$, the target is to generate new features $\tilde{S}$ from S useful for training a new classifier for Q. Based on optimization target $\log p(\tilde{S})$, the variational lower bound can be written as:

$$\log p(\tilde{S}) \geq \mathbb{E}_{q_\varphi(z|S)} \log p(\tilde{S}|z) - KL(q_\varphi(z|S) \| p_\theta(z)), \tag{10}$$

where z denotes the distribution. In this formula, the former item can be approximated by L_{cla} and L_{rec}, and the latter item just means Kullback-Leibler divergence between the posterior distribution based on support samples $q_\varphi(z|S)$ and the prior distribution $p_\theta(z)$. But in the actual implementation, the prior distribution $p_\theta(z)$ is hard to achieve. In VAE such prior is manually fixed as $N(0, 1)$, which is

Algorithm 1 Training Stage

Require: D_{base}

//Pre-training
while not convergent **do**
  for $I_i \in D_{base}$ **do**
    Extract feature $x_i = F(I_i)$
    Compute scores s_i^c for all classes using Eq. 1
  end for
  Update $F\&C$ by minimizing Eq. 3
end while

//Base-training
Frozen $F\&C$
while done **do**
  Sample a K-shot-N-way task $\{S, Q\}$ from D_{base} randomly
  Extract posterior distribution $\{(\mu_c, \sigma_c^2)\}_{c=1}^N$ and prior distribution $\{(\mu_c^{all}, \sigma_c^{2\ all})\}_{c=1}^N$ using Eq. 4 - Eq. 7
  Sample n latent vectors $\{z_j\}_{j=1}^n \sim N(\mu_c, \sigma_c^2)$ for each class
  Generate $n \times N$ new features and obtain augmented set $S \cup \tilde{S}$
  Compute L_{cla} based on $\tilde{S}$ using Eq. 3
  Compute L_{lat} using Eq. 8
  Compute L_{rec} using Eq. 9
  Update $E\&D$ by minimizing Eq. 11
end while

not applicable for VI-Net because the discrepancy between D_{base} and D_{novel} is unpredictable and huge. Therefore, the prior is set as $N(\mu_c^{all}, \sigma_c^{2\ all})$ to enhance generalization of the model, as L_{lat} shows.

3.4. Train and Test Strategy

Training stage: The training process based on the base dataset D_{base} can be divided into two stages: pre-training and base-training. The goal of pre-training is to obtain a feature extractor F and a base classifier C serving for base-training using the same train mode as baseline.

During the base-training stage, the task as described in part 3.1 are constructed to train both encoder E and decoder D. Specifically, for each episode in training, a set of support-set $S = \{(x_i, y_i)\}_{i=1}^{N \times K}$ and a query-set $Q = \{(x_i, y_i)\}_{i=1}^{N_q}$ are sampled from D_{base}. It is worth noting that labels of query samples are available during the training stage. Then we fix the parameters of F and C and feed S into the encoder-decoder module to obtain new features, and minimize total loss to train E and D:

$$L_{total} = L_{cla} + \alpha L_{rec} + \beta L_{lat}, \tag{11}$$

Algorithm 2 Testing Stage

Require: D_{novel}, F, E, D

while done **do**
  Sample a task $\{S, Q\}$ from D_{novel}
  Initialize a new classifier C
  Extract class distribution $\{(\mu_c, \sigma_c^2)\}_{c=1}^N$ using Eq. 6 - Eq. 7
  Sample n latent vectors $\{z_j\}_{j=1}^n \sim N(\mu_c, \sigma_c^2)$ for each class
  Generate $n \times N$ new features and obtain augmented $\tilde{S}$
  while $epoch \leq 100$ **do**
    for $x_i \in S \cup \tilde{S}$ **do**
      Compute scores s_i^c for all classes using Eq. 1
    end for
    Update C by minimizing Eq. 3
  end while
  Output C and recognize samples in Q
end while

where α is set to 0.75, and β is set to 0.25 in our experiments. Algorithm 1 describes the training process in detail.

Testing stage: In the testing stage, when facing a few-shot learning task $\{S, Q\}$, we first use the encoder-decoder module to augment the support set and then achieve a new task $\{S \cup \tilde{S}, Q\}$ with more samples per class. Like other fine-tuning based method, a new cosine classifier is built and trained with L_{cla} based on the augmented set, and finally, can recognize samples in Q. Algorithm 2 provides the testing process.

4. Experiments

4.1. Implementation Details

Just like other few-shot methods, the experiments are only performed on two kinds of tasks: 1-shot-5-way and 5-shot-5-way. We take average accuracy over the above tasks as the performance of VI-Net and prove that it has obtained the state-of-the-art result among augmentation-based algorithms by fair comparison.

Datasets: VI-Net is tested on two common datasets: mini-ImageNet and CUB [42], which are usually used to evaluate models about few-shot learning.

Among them, the mini-ImageNet dataset is the most frequently used because of its universality. As a subset of the famous dataset ImageNet [6], mini-ImageNet contains 100 categories with 600 images per class and covers various kinds of objects. It was first proposed by Vinyals et al. [39] and has been gradually standardized in the process of the development of few-shot learning. In most works today, mini-ImageNet is randomly split into 64, 16, 20 classes respectively for training, validation and testing. To be fair, we

choose the same split file as Vinyals et al. [39] which is most wildly used.

The CUB dataset only consists of bird images and mainly serves for fine-grained recognizing. It has also been used to evaluate few-shot algorithms in recent years. Because of its small inter-class and intra-class differences, it can help us evaluate algorithms from new perspectives. CUB contains 200 categories and 11788 images in total (about 60 per class) and is usually randomly split into 100, 50, 50 classes for training, validation and testing.

Architectures: As introduced in section 3.3.1, VI-Net is composed of feature extractor (F), distribution encoder (E), feature decoder (D) and classifier (C). Because VI-Net adopts the two-stage training mode, ResNet18 [13] is more suitable as our feature extractor for more precise distribution. When images are resized to 224×224, the dimension of the output features is 512. The encoder and decoder are both fully-connected layer, with ReLu function as the activation layer, and the dimension of latent space is set to 64. The classifier is also a fully-connected layer, whose output dimension is equal to the number of classes to recognize. It should be noted that both input features and weights should be normalized before dot-multiplication to ensure that the output vector represents cosine distance.

Hyper-Parameters: We pre-train F and C with 300 epochs for mini-ImageNet and 200 epochs for CUB, using SGD optimizer with the learning rate 0.1. When training E and D, we random construction 40000 5-shot-5-way tasks with 15 generated features per class and change the optimizer to Adam with learning rate 5e-4. During fine-tuning in the testing stage, we train the new classifier by support set for 100 epochs and use SGD optimizer with learning rate 1e-2.

4.2. Comparative Results

Table 1 reports the results on mini-ImageNet compared with other existing methods. We collect lots of results of existing methods and divide them into three classes: baseline, representative methods and augmentation-based methods. The baseline is described in section 3.2, which has been studied in many works [4][10][27][25]. Results show that VI-Net performs better than baseline by about 4%, it means our proposed generative module is effective for fine-tuning classifiers. The further analysis about the functions of modules is shown in section 4.3. Representative methods are all chosen from famous methods and can reflect classic ideas in few-shot learning. It should be pointed out that some methods only provide results with ConvNet-4 backbone. ConvNet-4 is a frequently-used simple convolution structure in few-shot learning methods, which only contains 4 blocks, and each block is composed of a 3×3 convolutional layer, a batch normalization layer, a ReLu activation layer and a 2×2 max-pooling layer. However, such a simple structure is not suitable for VI-Net and we choose a deeper backbone ResNet18 [13]. It means that more attention should be paid to results with deep backbones. When compared with other augmentation-based methods, VI-Net performs better than all methods with the 5-shot-5-way setting and achieve the state-of-the-art result. However, for 1-shot-5-way tasks, VI-Net is unsatisfactory and cannot defeat DTN based on external data. This is mainly because VI-Net needs multiple samples to modify the predicted distribution, but in 1-shot tasks, only one sample is available.

	Methods	Backbone	1-shot	5-shot
Baseline	Cosine-classifer[4]	ResNet-18	51.87	75.68
Representative	Matching Network[39]	ConvNet-4	43.56	55.31
	Meta-LSTM[28]	ConvNet-4	43.44	60.60
	MAML[8]	ConvNet-4	48.70	63.11
	Prototypical Network[33]	ConvNet-4	49.42	68.20
	Relation Network[36]	ConvNet-4	50.44	65.32
	FSL-GNN[9]	ConvNet-4	50.33	66.41
	TADAM[26]	ResNet-12	58.50	76.70
	TPN[23]	ResNet-12	59.46	75.65
	Variational FSL[46]	ResNet-12	61.23	77.69
	LEO[30]	WRN-28-10	61.76	77.59
Augmentation	Meta-GAN[47]	ConvNet-4	52.71	68.63
	Delta-Encoder[31]	VGG-16	59.90	69.70
	IDeMe-Net[5]	ResNet-18	59.14	74.63
	SalNet[45]	ResNet-101	62.22	77.95
	DTN[3]	ResNet-12	63.45	77.91
	AFHN[21]	ResNet-18	62.38	78.16
	VI-Net	ResNet-18	**61.05**	**78.60**

Table 1. Classification accuracies (%) on mini-ImageNet. When a method both provides results with ConvNet-4 and a deeper backbone, we only show the deeper one.

	Methods	Backbone	1-shot	5-shot
Baseline	Cosine-classifer[4]	ResNet-18	67.02	83.58
Representative	Matching Network[39]	ConvNet-4	49.34	59.31
	Meta-LSTM[28]	ConvNet-4	40.43	49.65
	MAML[8]	ConvNet-4	38.43	59.15
	Prototypical Network[33]	ConvNet-4	45.27	56.35
Augmentation	Delta-Encoder[31]	ResNet-18	69.80	82.60
	DTN[3]	ResNet-12	72.00	85.10
	AFHN[21]	ResNet-18	70.53	83.95
	VI-Net	ResNet-18	**74.76**	**86.84**

Table 2. Classification accuracies (%) on CUB.

Similarly, Table 2 shows the results on CUB. Because many methods did not report results on CUB, the number of methods is much less than Table 1. From the results, we can observe that the baseline method already has good performance for the 5-shot-5-way setting, but VI-Net can still beat it by 3.26%. For the 1-shot-5-way setting, VI-Net also obtains the state-of-the-art result and beats DTN by 2.76%. The reason why VI-Net can get the state-of-the-art results both on 1-shot tasks and 5-shot tasks is that images in CUB have high intra-class similarity, and just one sample can predict the distribution well after adequate training.

From the above results, it can be observed that VI-Net is especially effective on datasets such as CUB, whose samples have smaller intra-class differences. This can be explained by the fact that the distribution of data with small intra-class differences is easier to predict.

Methods	Loss Function			Results	
	CLA	REC	LAT	1-shot	5-shot
Baseline	√			51.87	75.68
Ablation	√			58.8	76.72
	√	√		59.59	76.74
	√		√	55.43	75.42
VI-Net	√	√	√	**61.05**	**78.6**

Table 3. Classification accuracies (%) on the baseline and VI-Net with different loss-functions on mini-ImageNet. **CLA**: L_{cla}, **REC**: L_{rec}, **LAT**: L_{lat}.

	1-shot	5-shot
2	59.21	77.28
5	**61.05**	**78.6**
10	60.21	77.76
20	57.02	75.34
50	56.79	75.64

Table 4. Classification accuracies (%) with different numbers of generated samples per class.

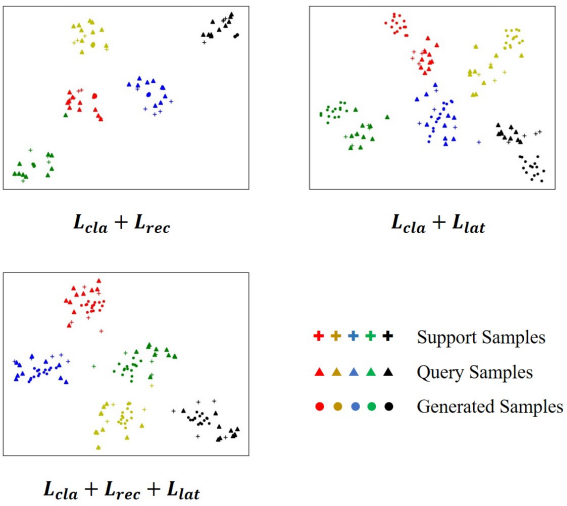

Figure 5. **t-SNE visualization of features.** Different categories are indicated by colors. It should be noted that the query samples are only used to help express the real distribution.

4.3. Ablation Study

In this section, we mainly focus on two issues: the effectiveness of the generative module and the impact of the number of generated samples per class. We design some experiments to compare results using ablation methods on mini-ImageNet, and summary results in tables and figures.

As described in section 3.3.2, there are three components in loss function to constraint encoder and decoder. Through experiments, if the loss function is separated into 3 components and only 1-2 components are used to train the generative module, the accuracy evidently decreased when compared with the complete method. In table 3, ablation methods mean different loss functions during training. Obviously, L_{cla} plays the most import role in training the generative module by raising the result to 76.72% while $L_{cla} + L_{rec}$ only raises the result to 76.74%. It is worth mentioning that L_{lat} cannot constraint decoder without L_{rec}, so only using $L_{rec} + L_{lat}$ may even reduce model accuracy. In summary, the generative module in VI-Net indeed assists the fine-tuning process and every component of the loss function is indispensable.

In order to more intuitively understand the impact of each component on the generation ability of VI-Net, we utilize t-SNE visualization tool to show the distribution status of samples in the feature space. It is easy to observe from Figure 5 that the composition of the loss function directly affects the distribution of the generated features. With $L_{cla}+L_{rec}$, all generated samples are concentrated near one point because the variance of classes will gradually disappear without L_{lat}. With $L_{cla} + L_{lat}$, as mentioned in 3.3.2, there is a clear gap between the generated samples and the real samples. Only with $L_{cla} + L_{rec} + L_{lat}$ can module generate suitable features, which have similar distribution characteristics as support features.

Although the generated features have been proven useful for the fine-tuning stage, they are not real essentially. When the number of new features is too large, they may become noise and seriously affect the performance. In Table 4, we do not change the number of generated samples during base-training, and just make the module generates different numbers of samples during testing on mini-ImageNet. From the results, VI-Net performs best when the hyperparameter is set to 5, and when the hyperparameter is over 20, there is a significant decline in performance.

5. Conclusion

In this paper, we propose a new augmentation-based model named VI-Net for few-shot learning. It utilizes a generative module based on distribution information to expand the support set. Specifically, VI-Net constructs a low-dimensional latent space to constrain samples of the same category within a multivariate Gaussian distribution by variational inference, and then generates new feature vectors by random sampling to make the decision boundary of the cosine-classifier sharper during the fine-tuning process. Through experiments, it is proved that VI-Net does achieve the state-of-the-art result when compared with other augmentation-based methods and every module in our model is meaningful.

Acknowledgement This work was supported by the National Natural Science Foundation of China under Grant 61773377 and the Fundamental Research Funds for the Central Universities under Grant FRF-BD-19-002A.

References

[1] Alexey Bochkovskiy, Chien Yao Wang, and Hong Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. 2020.

[2] L. C. Chen, G Papandreou, I Kokkinos, K Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848, 2018.

[3] Mengting Chen, Yuxin Fang, Xinggang Wang, Heng Luo, Yifeng Geng, Xinyu Zhang, Chang Huang, Wenyu Liu, and Bo Wang. Diversity transfer network for few-shot learning. In *Association for the Advance of Artificial Intelligence*, pages 10559–10566, 2020.

[4] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. *arXiv preprint arXiv:1904.04232*, 2019.

[5] Zitian Chen, Yanwei Fu, Yu-Xiong Wang, Lin Ma, Wei Liu, and Martial Hebert. Image deformation meta-networks for one-shot learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 8680–8689, 2019.

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. Ieee, 2009.

[7] Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611, 2006.

[8] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.

[9] Victor Garcia and Joan Bruna. Few-shot learning with graph neural networks. *arXiv preprint arXiv:1711.04043*, 2017.

[10] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4367–4375, 2018.

[11] Bharath Hariharan and Ross Girshick. Low-shot visual recognition by shrinking and hallucinating features. In *IEEE International Conference on Computer Vision*, pages 3018–3027, 2017.

[12] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *IEEE International Conference on Computer Vision*, 2017.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[14] S Hochreiter and J Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[15] Jongmin Kim, Taesup Kim, Sungwoong Kim, and Chang D Yoo. Edge-labeling graph neural network for few-shot learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 11–20, 2019.

[16] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[17] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[18] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *International Conference on Machine Learning, deep learning workshop*, volume 2. Lille, 2015.

[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[20] Y Lecun, Y Bengio, and G Hinton. Deep learning. *Nature*, 521(7553):436, 2015.

[21] Kai Li, Yulun Zhang, Kunpeng Li, and Yun Fu. Adversarial feature hallucination networks for few-shot learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 13470–13479, 2020.

[22] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, and Alexander C. Berg. Ssd: Single shot multibox detector. In *European Conference on Computer Vision*, 2016.

[23] Yanbin Liu, Juho Lee, Minseop Park, Saehoon Kim, Eunho Yang, Sung Ju Hwang, and Yi Yang. Learning to propagate labels: Transductive propagation network for few-shot learning. *arXiv preprint arXiv:1805.10002*, 2018.

[24] SC Martin Arjovsky and Leon Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, 2017.

[25] Thomas Mensink, Jakob Verbeek, Florent Perronnin, and Gabriela Csurka. Metric learning for large scale image classification: Generalizing to new classes at near-zero cost. In *European Conference on Computer Vision*, pages 488–501. Springer, 2012.

[26] Boris Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. In *Advances in Neural Information Processing Systems*, pages 721–731, 2018.

[27] Hang Qi, Matthew Brown, and David G Lowe. Low-shot learning with imprinted weights. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[28] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2016.

[29] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6), 2015.

[30] Andrei A Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. *arXiv preprint arXiv:1807.05960*, 2018.

[31] Eli Schwartz, Leonid Karlinsky, Joseph Shtok, Sivan Harary, Mattias Marder, Abhishek Kumar, Rogerio Feris, Raja Giryes, and Alex Bronstein. Delta-encoder: an effective sample synthesis method for few-shot object recognition. In *Advances in Neural Information Processing Systems*, pages 2845–2855, 2018.

[32] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[33] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077–4087, 2017.

[34] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems*, pages 3483–3491, 2015.

[35] Qianru Sun, Yaoyao Liu, Tat-Seng Chua, and Bernt Schiele. Meta-transfer learning for few-shot learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 403–412, 2019.

[36] 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 *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1199–1208, 2018.

[37] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[38] Joaquin Vanschoren. Meta-learning: A survey. *arXiv preprint arXiv:1810.03548*, 2018.

[39] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638, 2016.

[40] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys (CSUR)*, 53(3):1–34, 2020.

[41] Yu-Xiong Wang, Ross Girshick, Martial Hebert, and Bharath Hariharan. Low-shot learning from imaginary data. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 7278–7286, 2018.

[42] Peter Welinder, Steve Branson, Takeshi Mita, Catherine Wah, Florian Schroff, Serge Belongie, and Pietro Perona. Caltech-ucsd birds 200. 2010.

[43] Davis Wertheimer and Bharath Hariharan. Few-shot learning with localization in realistic settings. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 6558–6567, 2019.

[44] Huikai Wu, Junge Zhang, Kaiqi Huang, Kongming Liang, and Yizhou Yu. Fastfcn: Rethinking dilated convolution in the backbone for semantic segmentation. 2019.

[45] Hongguang Zhang, Jing Zhang, and Piotr Koniusz. Few-shot learning via saliency-guided hallucination of samples. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2770–2779, 2019.

[46] Jian Zhang, Chenglong Zhao, Bingbing Ni, Minghao Xu, and Xiaokang Yang. Variational few-shot learning. In *IEEE International Conference on Computer Vision*, pages 1685–1694, 2019.

[47] Ruixiang Zhang, Tong Che, Zoubin Ghahramani, Yoshua Bengio, and Yangqiu Song. Metagan: An adversarial approach to few-shot learning. In *Advances in Neural Information Processing Systems*, pages 2365–2374. 2018.