Noise-robust few-shot classification via Variational Adversarial Data Augmentation

Renjie Xu
China University of Petroleum (East China)
Qingdao 266580, China.
xurj1998@163.com

Baodi Liu
China University of Petroleum (East China)
Qingdao 266580, China.
thu.liubaodi@gmail.com

Kai Zhang
China University of Petroleum (East China)
Qingdao 266580, China.
zhangkai@upc.edu.cn

Honglong Chen
China University of Petroleum (East China)
Qingdao 266580, China.
chenhl@upc.edu.cn

Dapeng Tao
Yunnan University
Yunnan 650504, China
dapeng.tao@gmail.com

Weifeng Liu
China University of Petroleum (East China)
Qingdao 266580, China.
liuwf@upc.edu.cn

Abstract

Few-Shot classification models trained with clear samples have a poor effect to classify the samples from the real world which carry different scales of noise. To enhance the model for recognizing noisy samples, researchers usually utilize data augmentation or use noisy samples generated by adversarial training for model training. However, the existing methods still have the following problems: 1. The effect of data augmentation on the robustness of the model is limited; 2. The noise generated by adversarial training will usually cause overfitting and reduce the generalization ability of the model, which is very significate for the Few-Shot classification; 3. Most of the existing methods cannot generate appropriate noise adaptively. Given the above three points, to increase the robustness of the model and avoid reducing the generalization ability of the model, this paper proposes a noise-robust Few-Shot classification algorithm based on Variational Adversarial Data Augmentation (VADA). Different from the existing methods, VADA utilizes a Variational Noise Generator to generate adaptive noise distribution according to different samples based on adversarial learning and optimizes the variational automatic noise generator by minimizing the expectation of the empirical risk, to improve the robustness of the model and ensuring its generalization ability. Applying VADA for training can cause the Few-Shot classification model more robust when dealing with noisy data, while not losing the generalization ability. In this paper, we utilize FEAT and ProtoNet as baseline models, and the accuracy is verified on common Few-Shot classification datasets, e.g., MiniImageNet, TieredImageNet, and CUB. After training with VADA, the accuracy of the models for classifying samples with different scales of noise increases.

***Keywords:** Few-Shot learning, Adversarial learning, Robustness, Variational method.*

1. Introduction

The Few-Shot classification algorithm [9, 30] aims to utilize a few novel class samples to train the model so that the model can classify novel class samples. Because each Few-Shot classification task contains only a small number of samples, the noisy samples in Few-Shot tasks will greatly affect the classification effect of the model. In reality, it is very difficult to ensure that the samples in the task do not contain noise. Therefore, to better classify noisy samples, researchers usually apply data augmentation [17, 22, 36] or utilize noisy samples generated by adversarial training [6, 23] to train the model in purpose to increase its robustness to noise.

Training with samples generated with data-augmented samples is the most extensively utilized method to enhance the generalization ability and robustness of the classification model. Generally, most researchers [17, 36] apply specific types of data augmentation types, e.g., flipping, rotating, and adding random noise, to generate novel samples and shift the distribution of samples. During the training

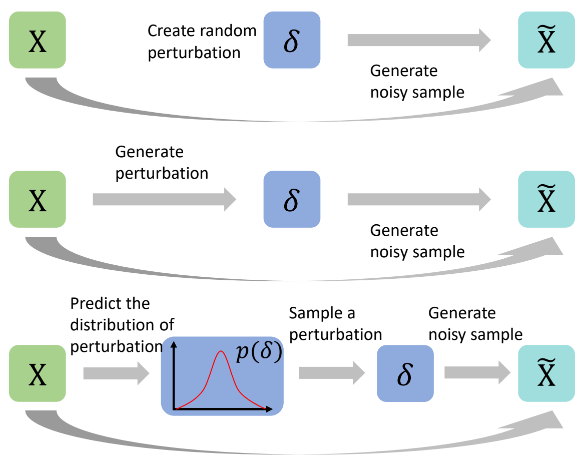

Figure 1. The first row represents the method of generating noisy samples with random perturbation. With this method, both the generalization ability and the robustness can be enhanced, while the robustness is enhanced slightly. To further increase the robustness, researchers generate controllable perturbation with the input samples as shown in the second row. Although this line of methods can increase the robustness, it would decrease the generalization ability of the model. To combine the advantages of both, we propose VADA, which is a method that generates controllable random perturbation via a variable noise generator.

progress, these models sample augmentation methods randomly, which can be invalid for model training. Furthermore, to apply data augmentation more effectively, another line of methods, e.g., efficient data augmentation network (EDANet) [4] applies automatically selected augmentation approaches to achieve optimal performance. Although the above augmentation methods are efficient in providing a model from overfitting and increasing generalization ability, their effect of enhancing robustness is insufficient.

Thus, focusing on enhancing the robustness, adversarial training [7, 35] is proposed as another approach to increase the noise-robustness of the model. In adversarial training methods, researchers aim to generate noises according to the samples to attack the classification model. After attacking, the model is trained to recognize the noised samples. With the repeating of such adversarial training, the model becomes harder and harder to be attacked, and thus the robustness of the model can be significantly enhanced. Such as the Fast Gradient Sign Method (FGSM) [12], with which the sample is perturbed by adding a small value along the gradient direction of its classification loss function, and then the perturbed samples are input into the model for training.

Although adversarial training can largely enhance the robustness of the classification model. The existing adversarial training methods still face two major problems: Firstly, some existing works [19, 21, 27] proposed that the adversarial training on small datasets may cause an overfitting phenomenon called "robust overfitting". With this phenomenon, the classification model becomes too robust and loses generalization ability, whereas the few-shot classification relies tightly upon it. Therefore, the adversarial training methods are hard to be utilized in few-shot classification directly. Secondly, most existing works are not self-adaptive, which represents that the attribute of the generated noise will not change during the training process, while we hold that the same noise will not benefit the model during training.

To solve the problems mentioned above, we propose a Variational Adversarial Data Augmentation (VADA) Algorithm which considers both the robustness and generalization ability of the classification model. With our VADA, we apply a Variational Noise Generator (VNG) to generate self-adaptive noise distribution for each sample (shown in Fig. 1). Then the noise is sampled from the generated distribution and added to the original sample to form the perturbed sample. Because the noise is sampled from a distribution according to the input sample, the controllable randomness can enhance both the generalization ability and the robustness. To obtain noise that can influence the accuracy of the classification model, we utilize adversarial training as the training strategy: The VNG is trained to fool the classification model while the classification model is trained against the fooling. During utilizing VADA, we suppose that appropriate noised samples are generated as the input of the classification model in each epoch of the training process as data augmentation. After training with our VADA, the adversarial robustness of the classification model will be enhanced without sacrificing the generalization ability. To validate the effect of our VADA, we conduct extensive experiments on popular Few-Shot classification datasets, e.g., MiniImageNet, TieredImageNet, and CUB. Utilizing our VADA, the testing accuracy against the different scales of noise raises on all the datasets.

The major contributions of this paper are summarized below.

- We propose a Variational Noise Generator (VNG) to generate noises that can perturb the classification performance of the Few-Shot model. VNG can generate proposal noises according to the input samples adaptively.
- We propose a Variational adversarial data augmentation (VADA) algorithm with our proposed VNG to enhance the robustness of Few-Shot models. Our VADA can enhance the adversarial robustness while not sacrificing the generalization ability.
- We conduct extensive experiments to verify the effect of VADA on popular Few-Shot datasets. After training with our VADA, the Few-Shot model becomes more robust against different scales of noise.

In this paper, we organize the subsequent of this paper as follows: We introduce the related works in Section 2. In Section 3, we introduce the details of VADA, which is mainly about the Variational Noise Generator. After introducing our VADA, the experimental results with our VADA are shown in Section 4. Finally, we construct the conclusion and prospect in Section 6.

2. Related works

In this section, we introduce the existing works which are related to our VADA, e.g., the adversarial robustness and the Variational Auto-Encoder.

2.1. Adversarial Robustness

Adversarial Robustness represents the ability of models against adversarial attacks. In former studies, researchers solve the robustness against adversarial attack problems with dual optimization [23]. During the optimization process of the adversarial training, a perturbation δ is searched to change the decision of the classification model:

$$\max_{||\delta|| \le \epsilon} \mathcal{L}(\mathrm{F}_\phi(x+\delta), y) \tag{1}$$

where F_ϕ is the classification model with parameter ϕ, $\mathcal{L}$ is the objective loss function of the model, and ϵ is a const which limits the size of the perturbation δ.

After the perturbation, the empirical adversarial risk, which is the objective function of the classification model, is represented in the following min-max setting:

$$\min_{\phi} \max_{||\delta|| \le \epsilon} \mathbb{E}_{x \sim p(x)}[\mathcal{L}(\mathrm{F}_\phi(x+\delta), y)] \tag{2}$$

The Projected Gradient Descend (PGD) is a typical method with this min-max setting. The PGD utilizes a multi-step FGSM to upgrade the noise, which is computed with the following iterative method:

$$\delta := P(\delta + \alpha \nabla_\delta \mathcal{L}(\mathrm{F}_\phi(x+\delta), y)) \tag{3}$$

where P is a projection over the ball of interest.

So far, many works [5, 11, 33, 34] aim at improving robustness by utilizing PGD to generate adversarial perturbation. In the field of Few-shot classification, Goldblum et al. [11] apply PGD in the inner loop of meta-learning, while computing gradient and updating the model in the outer loop with the adversarial query data. Zhang et al. [34] utilized PGD to generate samples to minimize the distance between classes as the perturbation.

Although adversarial training like PGD can generate effective noise easily with backpropagation, the noise generated this way has a shortcoming: Once the parameter of the classification model is determined, the noise is determined. Thus, this noise can let to robustness overfitting because of its certainty.

2.2. Variational Auto-Encoder

Variational Auto-Encoder (VAE) [1, 14] is a generative model which consists of an encoder that projects samples into a latent vector and a decoder that generates a sample according to the latent vector. The main idea of Variational Auto-Encoder is that the latent vector z is sampled from distribution and the generated sample should be similar to the original sample:

$$-\mathbb{E}_{z \sim q(z|x)}[\log p(x|z)] + KL(q(z|x)||N(0,1)) \tag{4}$$

In VAE, researchers assume that the probability of a latent vector can be represented as $p(z|x)$, which can be predicted with a Bayesian network $q(z|x)$. Then the decoder aims to keep $p(x|z)$ have the maximum certainty. It can be noted that the latent vector z has the following performance: 1. It is sampled from a distribution that depends on the input sample. 2. The distribution can be fitted with the Gaussian distribution. In another word, VAE grants us the ability to generate controllable randomness.

Although most of the works [10] apply VAE as an auto-encoder, some researchers [2] also discover its potential for noise. In this work, different from other existing works, the decoder of the VAE is not utilized. Thus, instead of generating noisy samples with the decoder, we apply the output of the encoder as the noise, while the output is a distribution matrix with the same size as the samples and not vectors in other works.

2.3. The Reparameterization Trick

Although researchers utilize VAE to generate distribution, sampling from the distribution will cut off the back-propagation of the gradient, which prevent the encoder from upgrading. To optimize the encoder, VAE utilizes the reparameterization trick to avoid this phenomenon.

In the reparameterization trick, researchers decouple the sampling into two steps: 1. sampling ξ from a distribution $p(\xi)$. 2. Generate z with $z = g_\theta(\xi)$, where g_θ is a function of ξ. Thus, the learnable distribution is divided into distributions with no learnable parameter and a learnable function. Only the function is trained during the optimization process.

3. Variational Adversarial Data Augmentation

We briefly introduce our Variational Adversarial Data Augmentation in this section, which includes problem definition, objective function, the Variational Noise Generator, and the training process of VADA.

3.1. Problem definition

In Few-Shot learning, researchers utilize a set of N-way K-shot Few-Shot tasks T_b which consisted of the base classes as the training tasks. After training the classification model with the training tasks, the model

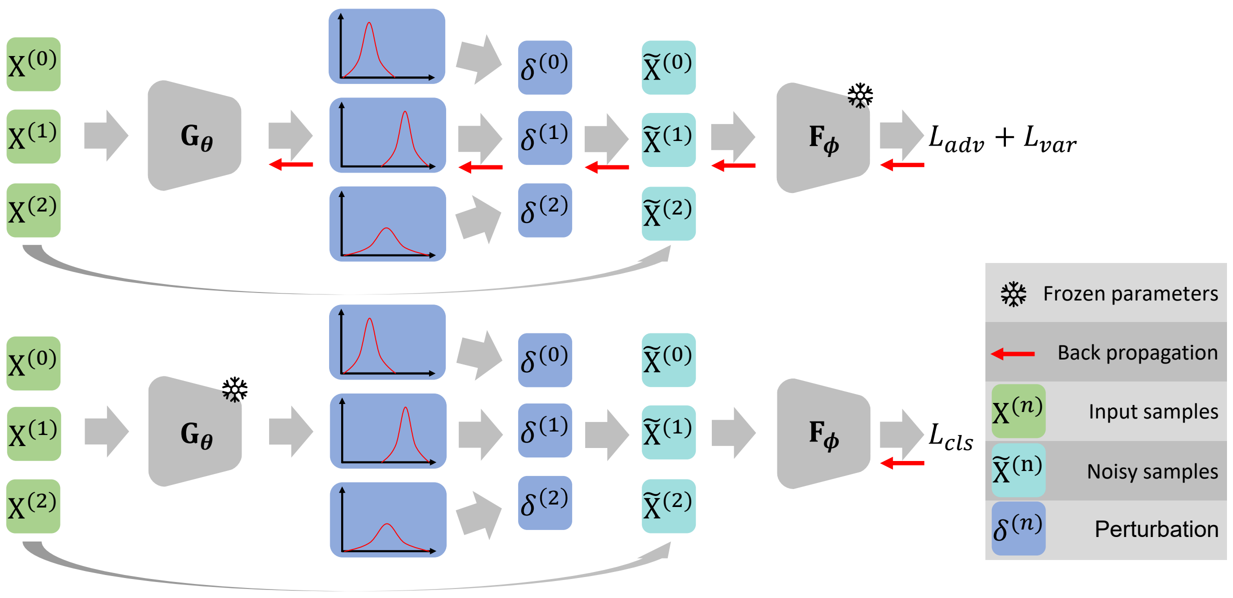

Figure 2. **The total training process of VADA:** Our VADA consists of a Variational Noise Generator G_θ and a classification model F_ϕ. The first row represents the training process of the VNG. While training the VNG, we compute the loss function according to the function (10). Then we utilize backpropagation to upgrade the parameter of VNG while freezing the parameter of the classification model. We utilize a classical training process to optimize the classification model in the second row. In this process, the VNG is frozen and generates noisy samples as data augmentation.

is tested with the Few-Shot tasks T_n with novel classes. And each of the N-way K-shot tasks contains K labeled samples in each of the N different base classes. In this paper, we represent each training task as $T = \{(x^{(1)}, y^{(1)}), \dots, (x^{(N*(K+M))}, y^{(N*(K+M))})\}$, in which the first $N * K$ samples are called support set and the $N * K + 1$ to $N * (K + M)$ are called query set. With the noise-robust setting, the samples from the base classes are clear while the samples from the novel classes carry noises with different scales. Because the novel classes have no intersection with the base classes, The effect of the model on the novel classes is tightly related to the generalization ability. Thus, it is important to enhance the robustness while keeping the generalization ability.

3.2. Objective function of VADA

Following the function (2), we write our objective function in a min-max formula:

$$\begin{aligned}&\min_{\phi}\max_{\theta}\mathbb{E}_{x\sim p(x)}[\mathcal{L}(F_\phi(G_\theta(x)), y)]\\ =&\min_{\phi}\mathbb{E}_{x\sim p(x)}\max_{p(\hat{x}|x)}\mathbb{E}_{\hat{x}\sim p(\hat{x}|x)}[\mathcal{L}(F_\phi(\hat{x}), y)]\end{aligned} \tag{5}$$

where the $p(x)$ is the prior of x, the $\hat{x} = x + \delta = G_\theta(x)$ is the noisy sample which carries noise. G_θ is the attacker model, which is our Variational Noise Generator. In this paper, the attacker model targets fooling the classification model via generating noise which follows the probability $\hat{x} \sim p(\hat{x}|x)$ to maximize the classification loss. And the classification model is betaken against attacking by minimizing the perturbed loss $\mathcal{L}(F_\phi(\hat{x}), y)$.

3.3. The Variational Noise Generator

To generate the distribution of the noise, we propose a Variational Noise Generator (VNG) to solve the maximizing problem, which utilizes a Gaussian distribution to fit the original noise distribution. Noticing that the maximizing problem can be rewritten as a minimizing problem:

$$\begin{aligned}&\max_{p(\hat{x}|x)}\mathbb{E}_{\hat{x}\sim p(\hat{x}|x)}[\mathcal{L}(F_\phi(\hat{x}), y)]\\ =&\min_{p(\hat{x}|x)}\mathbb{E}_{\hat{x}\sim p(\hat{x}|x)}[\mathcal{L}(F_\phi(\hat{x}), \bar{y})]\end{aligned} \tag{6}$$

where the $\bar{y}$ represents the labels except the correct label y. After rewriting, we can represent the rewritten function with a negative log-likelihood formula:

$$\begin{aligned}&\min_{p(\hat{x}|x)}\mathbb{E}_{\hat{x}\sim p(\hat{x}|x)}[\mathcal{L}(F_\phi(\hat{x}), \bar{y})]\\ =&\min_{p(\hat{x}|x)} - \int p(\hat{x}|x)\log p(\bar{y}|\hat{x})d\hat{x}\\ =&\min_{p(\hat{x}|x)} - \int p(\hat{x}|x)\log\frac{p(\bar{y},\hat{x})}{p(\hat{x}|x)}d\hat{x}\end{aligned} \tag{7}$$

Utilizing the distribution generated by VNG $p(\hat{x}|x, \theta)$ to fit the noise distribution $p(\hat{x}|x)$, we obtain:

$$\begin{aligned}
&\min_{p(\hat{x}|x)} - \int p(\hat{x}|x) \log \frac{p(\bar{y}, \hat{x})}{p(\hat{x}|x)} d\hat{x} \\
=& \min_{\theta} - \int p(\hat{x}|x,\theta) \log \frac{p(\bar{y}, \hat{x})}{p(\hat{x}|x,\theta)} d\hat{x} \\
&- \int p(\hat{x}|x,\theta) \log \frac{p(\hat{x}|x,\theta)}{p(\hat{x}|x)} d\hat{x}
\end{aligned} \tag{8}$$

where the first term of equation (8) can be represented as a Cross-Entropy (CE) loss $\mathcal{L}_{adv}$, and the second term can be written as a Kullback-Leibler divergence $\mathcal{L}_{var}$

$$\begin{aligned}
&\mathcal{L}_{VNG} = \mathcal{L}_{adv} + \mathcal{L}_{var} \\
=& - \int p(\hat{x}|x,\theta) \log \frac{p(\bar{y}, \hat{x})}{p(\hat{x}|x,\theta)} d\hat{x} \\
&+ \int p(\hat{x}|x,\theta) \log \frac{p(\hat{x}|x)}{p(\hat{x}|x,\theta)} d\hat{x}
\end{aligned} \tag{9}$$

The $\mathcal{L}_{VNG}$ in equation (9) is the objective function of VNG, which is consisted of two parts. The $\mathcal{L}_{adv}$ hold that the generated noise can fool the classifier and the $\mathcal{L}_{var}$ aims at limiting the distribution. To optimize the parameter of VADA with gradient descent, we make an unbiased estimation:

$$\begin{aligned}
\mathcal{L}_{VNG} =& \frac{1}{N} \sum_{x} -p(\hat{x}_0|x,\theta) \log p(\bar{y}|\hat{x}_0) \\
&+ \mathrm{KL}(\mathcal{N}(\hat{x}; x+\mu, \Sigma) || \mathcal{N}(x, \Gamma)) \\
=& \frac{1}{N} \sum_{x} -p(\hat{x}_0|x,\theta) \log p(\bar{y}|\hat{x}_0) \\
&+ \mathrm{KL}(\mathcal{N}(\delta; \mu, \Sigma) || \mathcal{N}(0, \Gamma))
\end{aligned} \tag{10}$$

where the $\mathcal{N}(\hat{x}; x+\mu, \Sigma) = p(\hat{x}|x,\theta)$ is the distribution generated by VNG, and $\hat{x}_0$ is a random value selected from the distribution $p(\hat{x}|x,\theta)$, $p(\hat{x}_0|x,\theta)$ shows the probability of selecting $\hat{x}_0$ from $p(\hat{x}|x,\theta)$. We assume that the noise distribution to be fitted can be represented as a Gaussian distribution $\mathcal{N}(0,\Gamma)$, where the variance is expressed as Γ.

With our VNG, we can generate controllable randomness according to the input sample x. After training, the VNG obtains the ability to generate noise distribution which can most perturb the samples. During VADA, we train VNG and the classification model alternately.

3.4. Training with VADA

After introducing VNG, we would introduce the training process of the VADA, which is shown in Algorithm 1.

In this paper, we regard the optimization of the objective function as a biconvex problem. As shown in Fig. 2, We train the classification model or VNG while freezing the parameter of the other. During training, We apply the loss equation (10) to update VNG. For easier adjusting, we fix the Γ with a small value and change the weight γ to construct a soft formula of equation (10):

Algorithm 1: The proposed VADA algorithm

```
Input: A set of Training tasks $\{T_b\}$ which consisted
       with base classes, and Validation tasks $\{T_v\}$
       which consisted with Validation classes. A
       Classification model $F_\phi$, and a VNG $G_\theta$.
Output: The parameter $\phi$ of classification model
        $F_\phi$.
1  while Epoch $\leq$ Total Epoch do
2      for Each $T_b \in \{T_b\}$ do
3          /*Optimizing the VNG                              */
4          if Epoch % Update Frequency == 0 then
5              Generate the noisy samples $\{\hat{x}^{(i)}\}$ with
                equation (12) for each $x^{(i)} \in T_b$;
6              Compute the probability
                $\{p(\hat{x}^{(i)}|x^{(i)},\theta)\}$ for pair of $\hat{x}^{(i)}$ and
                $x^{(i)}$;
7              Predict the likelihood $\{p(\bar{y}|\hat{x})\}$ for each
                $\hat{x}^{(i)}$ via the Classification model $F_\phi$;
8              Compute the KL divergence between
                $\{p(\hat{x}^{(i)}|x^{(i)},\theta)\}$ and $\mathcal{N}(x^{(i)},1)$;
9              Compute the $\mathcal{L}_{VNG}$ with equation (10);
10             Update the parameter of VNG $\theta$ with
                $\mathcal{L}_{VNG}$;
11         end
12         /*Optimizing the Classification model             */
13         Generate the noisy samples $\{\hat{x}^{(i)}\}$ with
            equation (12) for each $x^{(i)} \in T_b$;
14         Predict the likelihood $\{p(y|\hat{x})\}$ for each $\hat{x}^{(i)}$
            via the Classification model $F_\phi$;
15         Compute the $\mathcal{L}_{cls}$ via equation (13);
16         Update the parameter of classification model
            $\phi$ with $\mathcal{L}_{cls}$;
17     end
18     Evaluate the accuracy with Validation tasks
        $\{T_v\}$;
19     Epoch += 1;
20 end
21 return The parameter $\phi$ with best validation
    accuracy.
```

$$\begin{aligned}
&\mathcal{L}_{VNG} \\
=& \frac{1}{N} \sum_{x} -p(\hat{x}_0|x,\theta) \log p(\bar{y}|\hat{x}_0) \\
&+ \gamma * \mathrm{KL}(\mathcal{N}(\delta; \mu, \Sigma) || \mathcal{N}(0, \Gamma))
\end{aligned} \tag{11}$$

We utilize the reparameterization trick to keep the VNG

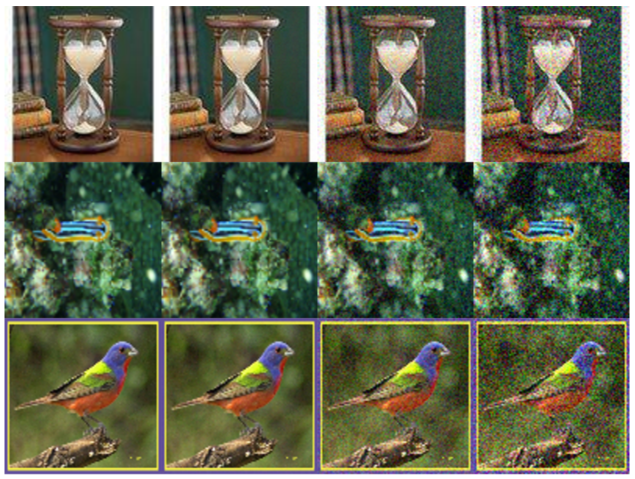
Figure 3. In this figure, we show the origin and noisy samples from datasets utilized in this paper. The columns represent the original sample and the noisy samples with noise scale $\epsilon = 2/255, 16/255, 32/255$ from left to the right. And the rows represent different datasets, e.g., MiniImageNet, TieredImageNet, and CUB from up to down.

gradable. We split the VNG G_θ into two parts: Firstly, we sample ξ from a distribution $\mathcal{N}(\xi; 0, 1)$, Then, Our VNG generates the mean μ and variance Σ:

$$\begin{aligned} &\hat{x} = x + \mu + \xi * \Sigma \\ s.t. \quad &\mu, \Sigma = \mathrm{G}_\theta(x). \end{aligned} \tag{12}$$

But, for short, we mark the whole process of generating $\hat{x}$ as $\hat{x} = \mathrm{G}_\theta(x)$. Therefore, the loss function we utilized for training the classification model is a Cross-entropy loss:

$$\begin{aligned} \mathcal{L}_{cls} =&\mathbb{E}_{x\sim p(x)}[\mathcal{L}(\mathrm{F}_\phi(\mathrm{G}_\theta(x), y)] \\ =&\frac{1}{N}\sum_x -\log p(y|\mathrm{G}_\theta(x)) \end{aligned} \tag{13}$$

4. Experiments

In this section, we exhibit the experimental results to evaluate the training effect of our VADA. The subsections include Dataset Descriptions, Baseline Information, Experimental Details, and Experimental Results.

4.1. Dataset Descriptions

Three popular Few-Shot Datasets are utilized in this paper, e.g., MiniImageNet, TieredImageNet, and CUB.

MiniImageNet [15] is a subset of the ImageNet, which is one of the most commonly utilized datasets in image recognition. Its subset, MiniImageNet, is mainly utilized in the Few-Shot classification field. In MiniImageNet, 60000 images of different objects are included in 100 classes, which represents 600 images per class. In this paper, we divide the classes into three subsets which contain 60, 16, and 24 classes respectively for training, validation, and testing.

TieredImageNet [18] is a larger dataset than MiniImageNet, which is also a subset of ImageNet. The TieredImageNet dataset consists of 779,165 images. The images are divided into 608 classes and 34 higher-level nodes. Following the proposed works [32], we divided the TieredImageNet into training, validation, and testing sets according to different nodes. Specifically, 20 nodes are included in the training set while 6 and 8 nodes are in the validation and testing sets.

CUB [31] is consisted of 200 subcategories and a total of 11,788 images of birds. Therefore, researchers apply the CUB dataset for fine-grained visual categorization. In this paper, we split CUB into training, Validation, and testing sets according to different subcategories. Following [3, 26], We utilize the same split method, in which 100 classes are used as the training set, 50 classes are used as the validation set, and 50 classes are applied as the testing set.

With the above datasets, we make the following settings: To simulate the training process on the ideal dataset and the testing process on samples from the real world, the training and validation sets are selected from the dataset directly and without adding any noise, and the testing samples are added different scales of Gaussian noise.

4.2. Baseline Information

In this paper, we utilize widely recognized Few-Shot classification models, e.g., FEAT and ProtoNet as baselines to observe how our VADA influence their robustness.

ProtoNet [24] is a classical Few-Shot classification model. With ProtoNet, the model obtains the prototype of each class via the support set samples. Then, the distance between the prototype and samples from the query set is computed to predict the labels.

FEAT (Few-Shot Embedding Adaptation with Transformer) [32] is a recently popular metric-based model. Similar to ProtoNet, FEAT utilizes the feature extracted from the support set to obtain prototypes for each class. Then a transformer [28] is utilized to adjust the prototypes to generate a better embedding.

In this paper, we set the hyperparameters of ProtoNet and FEAT the same as in the original paper of FEAT.

4.3. Experimental Details

In the experiments, we utilize the FEAT and ProtoNet with Conv4 and Resnet12 [13] as the backbone and test the accuracy on MiniImageNet, TieredImageNet, and CUB for both 1-shot and 5-shot settings. We apply different hyperparameters in different experiments. In every experiment in this section, we update the VNG every 4 training steps and set the updating rate as 0.0001.

Table 1. Percentage Accuracy of noisy data with Standard Deviation on MiniImageNet.

Method	Backbone	$\epsilon = 2/255$		$\epsilon = 16/255$		$\epsilon = 32/255$	
		1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
FEAT	Resnet12	**66.63 ± 0.20**	**82.21 ± 0.14**	61.31 ± 0.21	77.42 ± 0.15	48.83 ± 0.20	66.38 ± 0.17
FEAT+Augment		66.29 ± 0.21	81.29 ± 0.14	61.73 ± 0.20	**78.71 ± 0.15**	50.10 ± 0.20	**70.35 ± 0.16**
FEAT+PGD		63.36 ± 0.65	78.36 ± 0.48	58.92 ± 0.66	73.94 ± 0.50	50.25 ± 0.61	66.34 ± 0.53
FEAT+VADA (ours)		66.62 ± 0.20	81.97 ± 0.14	**62.15 ± 0.20**	78.01 ± 0.15	**50.67 ± 0.20**	67.58 ± 0.16
FEAT	Conv4	54.72 ± 0.20	71.63 ± 0.16	53.19 ± 0.20	70.22 ± 0.16	48.47 ± 0.20	65.85 ± 0.17
FEAT+Augment		53.86 ± 0.20	70.77 ± 0.16	52.69 ± 0.19	69.40 ± 0.16	48.58 ± 0.20	65.20 ± 0.17
FEAT+PGD		48.64 ± 0.61	63.30 ± 0.54	46.91 ± 0.61	61.86 ± 0.52	45.04 ± 0.60	59.62 ± 0.52
FEAT+VADA (ours)		**55.17 ± 0.20**	**71.88 ± 0.16**	**53.64 ± 0.19**	**70.43 ± 0.16**	**48.96 ± 0.20**	**66.14 ± 0.17**
ProtoNet	Resnet12	63.91 ± 0.21	80.27 ± 0.14	59.88 ± 0.21	**77.25 ± 0.15**	51.54 ± 0.20	69.57 ± 0.16
ProtoNet+Augment		63.45 ± 0.21	80.05 ± 0.14	**60.16 ± 0.21**	76.72 ± 0.15	52.41 ± 0.20	**69.64 ± 0.16**
ProtoNet+PGD		60.17 ± 0.68	78.74 ± 0.47	56.35 ± 0.66	75.18 ± 0.48	49.57 ± 0.63	66.34 ± 0.53
ProtoNet+VADA (ours)		**64.05 ± 0.21**	**80.33 ± 0.14**	59.96 ± 0.21	76.81 ± 0.15	**52.43 ± 0.20**	68.73 ± 0.16
ProtoNet	Conv4	54.59 ± 0.20	**71.90 ± 0.16**	**53.45 ± 0.20**	70.06 ± 0.16	48.39 ± 0.19	65.96 ± 0.16
ProtoNet+Augment		51.67 ± 0.20	69.85 ± 0.16	50.90 ± 0.20	68.67 ± 0.16	47.44 ± 0.20	64.67 ± 0.17
ProtoNet+PGD		48.60 ± 0.65	62.92 ± 0.52	47.66 ± 0.61	61.74 ± 0.53	44.05 ± 0.62	58.97 ± 0.54
ProtoNet+VADA (ours)		**54.84 ± 0.19**	71.59 ± 0.16	53.23 ± 0.19	**70.35 ± 0.16**	**49.35 ± 0.20**	**66.43 ± 0.17**

Table 2. Percentage Accuracy of noisy data with Standard Deviation on TieredImageNet and CUB.

Datasets	Methods	$\epsilon = 2/255$		$\epsilon = 16/255$		$\epsilon = 32/255$	
		1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
TieredImageNet	FEAT	**70.55 ± 0.23**	84.44 ± 0.16	69.77 ± 0.23	83.86 ± 0.16	67.14 ± 0.23	82.16 ± 0.16
	FEAT+Augment	70.17 ± 0.23	**84.90 ± 0.16**	68.91 ± 0.23	84.25 ± 0.16	66.08 ± 0.23	82.16 ± 0.16
	FEAT+PGD	60.25 ± 0.69	75.83 ± 0.55	58.24 ± 0.69	74.78 ± 0.55	55.77 ± 0.71	71.92 ± 0.56
	FEAT+VADA (ours)	70.51 ± 0.23	84.74 ± 0.16	**69.84 ± 0.23**	**84.31 ± 0.16**	**67.35 ± 0.22**	**82.47 ± 0.16**
CUB	FEAT	65.47 ± 0.23	**79.70 ± 0.17**	63.67 ± 0.24	78.11 ± 0.18	57.76 ± 0.24	72.58 ± 0.19
	FEAT+Augment	62.58 ± 0.23	78.84 ± 0.17	60.89 ± 0.23	77.13 ± 0.18	55.24 ± 0.23	71.88 ± 0.19
	FEAT+PGD	55.63 ± 0.71	71.58 ± 0.58	55.18 ± 0.71	70.20 ± 0.60	50.94 ± 0.70	66.52 ± 0.62
	FEAT+VADA (ours)	**65.66 ± 0.23**	79.46 ± 0.17	**64.13 ± 0.23**	**78.18 ± 0.18**	**58.01 ± 0.24**	**72.73 ± 0.19**

To train with MiniImageNet, we utilize different γ to limit the distribution of the generated noise. With the 1-shot setting of the ProtoNet, we apply $\gamma = 0.1$ for Conv4 and $\gamma = 0.01$ for Resnet12. For the 5-shot setting of ProtoNet, the γ is set to 0.1. With the Feat, we utilize $\gamma = 0.1$ to deal with the training of both 1-shot Conv4 and Resnet12. The 5-shot setting of γ for Conv4 and Resnet12 are appointed as 0.01 and 0.001, separately.

We only utilize Resnet12 for TieredImageNet and Conv4 for CUB. Thus, for 1-shot and 5-shot classification on TieredImageNet, we set the $\gamma = 0.005$ for both, while 0.001 and 0.01 for the 1-shot and 5-shot classification on CUB.

Table 3. Percentage Accuracy of clear data with Standard Deviation on MiniImageNet.

Methods	Backbone	1-shot	5shot
Matching Nets [29]	Conv4	43.44 ± 0.77	55.31 ± 0.73
Relation Network [25]	Conv4	50.44 ± 0.82	65.32 ± 0.70
MAML [8]	Conv4	48.70 ± 1.75	63.11 ± 0.92
Meta-LSTM [16]	Conv4	43.56 ± 0.84	60.60 ± 0.71
ProtoNet	Conv4	54.70 ± 0.20	71.81 ± 0.16
ProtoNet+Augment	Conv4	51.87 ± 0.20	70.05 ± 0.16
FEAT	Conv4	54.69 ± 0.19	71.94 ± 0.16
FEAT+Augment	Conv4	53.86 ± 0.20	70.87 ± 0.16
ProtoNet+VADA	Conv4	54.66 ± 0.20	71.86 ± 0.16
FEAT+VADA	Conv4	54.73 ± 0.20	71.91 ± 0.16
LEO [20]	Resnet12	61.76 ± 0.08	77.59 ± 0.12
Protonet	Resnet12	64.12 ± 0.21	80.56 ± 0.14
ProtoNet+Augment	Resnet12	63.64 ± 0.21	79.99 ± 0.14
FEAT	Resnet12	66.45 ± 0.20	81.67 ± 0.13
FEAT+Augment	Resnet12	66.30 ± 0.21	81.40 ± 0.14
ProtoNet+VADA	Resnet12	64.04 ± 0.21	80.50 ± 0.14
FEAT+VADA	Resnet12	66.37 ± 0.20	81.84 ± 0.14

Table 4. Percentage Accuracy of ProtoNet with Standard Deviation for different regular weight on noisy data in MiniImageNet.

Regular weight	1-shot	5shot
0.1	49.35 ± 0.20	65.62 ± 0.17
0.01	48.89 ± 0.19	66.43 ± 0.17
0.005	47.02 ± 0.20	65.90 ± 0.17
0.001	48.06 ± 0.19	65.99 ± 0.17

After training the models, 10000 tasks from the testing set are utilized to evaluate the accuracy. The testing samples are added noise with scale $\epsilon = 2/255, 16/255, 32/255$, as shown in Fig. 3.

4.4. Results on noisy data

We show the accuracy of our VADA while facing noisy samples from different datasets, e.g., MiniImageNet, TieredImageNet, and CUB, to show how data augment or our VADA enhances the robustness of the models. In both Table 1 and Table 2, blue represents that the accuracy is lower than baseline and red color means that the accuracy is higher than baseline while the bold style shows that the results are the highest.

As shown in Table 1, we evaluate our VADA with baseline models on noisy samples from the MiniImageNet. For samples that carry noise with scale $\epsilon = 2/255$, the noise can only slightly reduce the accuracy. Thus, in most cases, VADA can only lead to a similar accuracy compared with the baselines, which is 0.07% higher than the baseline average. But with the increase of the noise scale, this enhancement becomes more obvious. With $\epsilon = 16/255$, our VADA can increase the accuracy by 0.23% average. The models obtained the largest improvement when $\epsilon = 32/255$, which is a serious perturbation. In this condition, our VADA can increase the accuracy by 0.66% average. The mark "+Augment" represents the model trained with samples with data augmentation, which is the representation of (a) in Fig. 1. As shown in the table, data augmentation can also improve the accuracy in some situations, but compared with our VADA, data augmentation is not a stable enough method that can improve the robustness and will sometimes hurt the accuracy. With "+PGD", we expose the testing accuracy of model which is trained with noisy sample generated by PGD, which represents the (b) in Fig. 1. In this section, we set the PGD as the $\epsilon = 0.1$, the updating step size as 0.01, and update for 2 steps for each sample. Although training with noisy samples generated with PGD can improve the robustness of the model, the model becomes too robust to generalize to novel classes which carry uncertain noises. In this case, as shown in the table, training with PGD-perturbed samples can not improve the accuracy of a few-shot model.

Table 5. Percentage Accuracy of ProtoNet with Standard Deviation for different Update Frequencies on noisy data in MiniImageNet.

Update Frequency	1-shot	5shot
1	48.25 ± 0.20	66.17 ± 0.17
2	48.18 ± 0.20	65.82 ± 0.17
4	49.35 ± 0.20	66.43 ± 0.17
8	48.41 ± 0.20	65.71 ± 0.17
16	48.58 + 0.20	66.14 ± 0.17

The accuracy of our VADA on TieredImageNet and CUB is contained in Table 2. With the results, we can recognize that VADA can improve the 5-shot accuracy better, which increases by 0.32% on TieredImageNet and improves the 1-shot accuracy by 0.30% on the CUB dataset. Also compared with the data augmentation, our VADA can improve the accuracy and robustness more stably and does not sacrifice the generalization ability.

4.5. Results on clear data

In this subsection, we exhibit the classification accuracy of baseline models and models trained with VADA. With the experimental results, we are verifying how our VADA algorithm influences the generalization ability.

The accuracy of ProtoNet and Feat which were trained with our VADA are exhibited in Table 3. With the shown accuracy, it can be recognized that the accuracy of models trained with our VADA is close to the baseline models. which represents that our VADA has no obvious negative influence on the generalization ability. Specifically, the accuracy of the models trained with our VADA changes averagely by 0.00125%.

4.6. Results of different hyperparameters

In this section, we discuss the selection of the hyperparameters we utilize in the VADA, e.g., the update frequency and the regular weight γ. In this section, we set the scale of noise $\epsilon = 32/255$.

In Table 4, we discuss how the regular weight γ influences the classification accuracy. In this case, we can recognize that our parameter choice of the regular weight can lead to the best accuracy. The key point of the adjustment of regular weight is keeping the balance between the $\mathcal{L}_{adv}$ and $\mathcal{L}_{var}$. Thus, the VNG can generate appropriate noise which won't cause robust overfitting.

The results of updating the frequency of VNG are shown in Table 5. The accuracy fluctuates following the change in the update frequency. Generally, the update frequency is not as important as a hyperparameter as the regular weight. Therefore, we didn't change the update frequency during the former experiments.

5. conclusion

The Few-Shot classification models which were trained with clear samples had trouble recognizing noisy samples. Thus, to enhance the noise-robustness of the model, we proposed a novel data augmentation method with the variational method call Variational adversarial data Augmentation (VADA) algorithm, in which we utilized an adversarially-trained Variational Noise Generator (VNG) to generate noise which to followed adaptive appropriate noise. with our VADA, the robustness of the model could be enhanced and milder robustness overfitting would be caused. In this paper, we conducted experiments to verify the effect of our VADA, which became more obvious with the raising of the noise scale. However, our VADA is sensitive to regular weight, which is a defect that needs to be solved. In a further study, we would be working on solving the existing problems of VADA and exploring other algorithms to improve the robustness of classification models.

6. Acknowledgements

This work was supported by the Yunnan provincial major science and technology special plan projects under Grant 202202AD080003 and the Natural Science Foundation of Shandong Province (No.ZR2022MD090).

In this paper, Renjie Xu and Weifeng Liu cooperated to design the Methodology, Baodi Liu did the conceptualization, Kai Zhang validate the experimental results, Renjie Xu and Honglong Chen writed the original draft and did the review & editing, and Dapeng Tao gives the funding acquisition of this paper.

7. Declaration of competing interest

The authors have no competing interests to declare that are relevant to the content of this article.

References

[1] H. Akrami, A. A. Joshi, J. Li, S. Aydöre, and R. M. Leahy. A robust variational autoencoder using beta divergence. *Knowledge-Based Systems*, 238:107886, 2022. 3

[2] Y. Bando, K. Sekiguchi, and K. Yoshii. Adaptive neural speech enhancement with a denoising variational autoencoder. In *INTERSPEECH*, pages 2437–2441, 2020. 3

[3] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C. F. Wang, and J.-B. Huang. A closer look at few-shot classification. In *International Conference on Learning Representations*, 2018. 6

[4] W. Cho and E. Kim. Improving augmentation efficiency for few-shot learning. *IEEE Access*, 10:17697–17706, 2022. 2

[5] J. Dong, Y. Wang, J.-H. Lai, and X. Xie. Improving adversarially robust few-shot image classification with generalizable representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9025–9034, 2022. 3

[6] Y. Dong, Q.-A. Fu, X. Yang, T. Pang, H. Su, Z. Xiao, and J. Zhu. Benchmarking adversarial robustness on image classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 321–331, 2020. 1

[7] A. Fawzi, S.-M. Moosavi-Dezfooli, and P. Frossard. Robustness of classifiers: from adversarial to random noise. *Advances in neural information processing systems*, 29, 2016. 2

[8] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017. 8

[9] M. Gaikwad and A. Doke. Survey on meta learning algorithms for few shot learning. In *2022 6th International Conference on Intelligent Computing and Control Systems (ICICCS)*, pages 1876–1879. IEEE, 2022. 1

[10] B. Ghojogh, A. Ghodsi, F. Karray, and M. Crowley. Factor analysis, probabilistic principal component analysis, variational inference, and variational autoencoder: Tutorial and survey. *arXiv preprint arXiv:2101.00734*, 2021. 3

[11] M. Goldblum, L. Fowl, and T. Goldstein. Adversarially robust few-shot learning: A meta-learning approach. *Advances in Neural Information Processing Systems*, 33:17886–17895, 2020. 3

[12] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *stat*, 1050:20, 2015. 2

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 6

[14] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *stat*, 1050:1, 2014. 3

[15] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. *Proceedings ofthe 5th International Conference on Learning Representations (ICLR)*, 2016. 6

[16] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. *International Conference on Machine Learning*, 2017. 8

[17] S.-A. Rebuffi, S. Gowal, D. A. Calian, F. Stimberg, O. Wiles, and T. A. Mann. Data augmentation can improve robustness. *Advances in Neural Information Processing Systems*, 34:29935–29948, 2021. 1

[18] M. Ren, E. Triantafillou, S. Ravi, J. Snell, K. Swersky, J. B. Tenenbaum, H. Larochelle, and R. S. Zemel. Meta-learning for semi-supervised few-shot classification. *arXiv preprint arXiv:1803.00676*, 2018. 6

[19] L. Rice, E. Wong, and Z. Kolter. Overfitting in adversarially robust deep learning. In *International Conference on Machine Learning*, pages 8093–8104. PMLR, 2020. 2

[20] A. A. Rusu, D. Rao, J. Sygnowski, O. Vinyals, R. Pascanu, S. Osindero, and R. Hadsell. Meta-learning with latent embedding optimization. In *International Conference on Learning Representations*, 2018. 8

[21] L. Schmidt, S. Santurkar, D. Tsipras, K. Talwar, and A. Madry. Adversarially robust generalization requires more data. *Advances in neural information processing systems*, 31, 2018. 2

[22] C. Shorten and T. M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019. 1

[23] S. H. Silva and P. Najafirad. Opportunities and challenges in deep learning adversarial robustness: A survey. *arXiv preprint arXiv:2007.00753*, 2020. 1, 3

[24] J. Snell, K. Swersky, and R. Zemel. Prototypical networks for few-shot learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 4080–4090, 2017. 6

[25] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1199–1208, 2018. 8

[26] E. Triantafillou, R. Zemel, and R. Urtasun. Few-shot learning through an information retrieval lens. *Advances in neural information processing systems*, 30, 2017. 6

[27] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry. Robustness may be at odds with accuracy. In *International Conference on Learning Representations*, number 2019, 2019. 2

[28] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 6

[29] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29:3630–3638, 2016. 8

[30] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)*, 53(3):1–34, 2020. 1

[31] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-ucsd birds 200. 2010. 6

[32] H.-J. Ye, H. Hu, D.-C. Zhan, and F. Sha. Few-shot learning via embedding adaptation with set-to-set functions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8808–8817, 2020. 6

[33] Z. You, J. Ye, K. Li, Z. Xu, and P. Wang. Adversarial noise layer: Regularize neural network by adding noise. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 909–913, 2019. 3

[34] H. Zhang and J. Wang. Defense against adversarial attacks using feature scattering-based adversarial training. *Advances in Neural Information Processing Systems*, 32, 2019. 3

[35] X. Zhang, Q. Wang, J. Zhang, and Z. Zhong. Adversarial autoaugment. In *International Conference on Learning Representations*, 2019. 2

[36] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang. Random erasing data augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13001–13008, 2020. 1