

Reinforcing Generated Images via Meta-learning for One-Shot Fine-Grained Visual Recognition

Satoshi Tsutsui, Yanwei Fu, David Crandall, *Member, IEEE*

Abstract—One-shot fine-grained visual recognition often suffers from the problem of having few training examples for new fine-grained classes. To alleviate this problem, off-the-shelf image generation techniques based on Generative Adversarial Networks (GANs) can potentially create additional training images. However, these GAN-generated images are often not helpful for actually improving the accuracy of one-shot fine-grained recognition. In this paper, we propose a meta-learning framework to combine generated images with original images, so that the resulting “hybrid” training images improve one-shot learning. Specifically, the generic image generator is updated by a few training instances of novel classes, and a Meta Image Reinforcing Network (MetaIRNet) is proposed to conduct one-shot fine-grained recognition as well as image reinforcement. Our experiments demonstrate consistent improvement over baselines on one-shot fine-grained image classification benchmarks. Furthermore, our analysis shows that the reinforced images have more diversity compared to the original and GAN-generated images.

Index Terms—Fine-grained visual recognition, One-shot learning, Meta-learning.

1 INTRODUCTION

The availability of vast labeled datasets has been crucial for the recent success of deep learning. However, there will always be learning tasks for which labeled data is sparse. Fine-grained visual recognition is one typical example: when images are to be classified into many very specific categories (such as species of birds), it may be difficult to obtain training examples for rare classes, and producing the ground truth labels may require significant expertise (e.g., from ornithologists). One-shot learning is thus very desirable for fine-grained visual recognition.

A recent approach to address data scarcity is meta-learning [1]–[4], which trains a parameterized function called a meta-learner that maps labeled training sets to classifiers. The meta-learner is trained by sampling small training and test sets from a large dataset of a base class. Such a meta-learned model can be adapted to recognize novel categories with a single training instance per class. Another way to address data scarcity is to synthesize additional training examples, for example by using Generative Adversarial Networks (GANs) [5], [6]. However, classifiers trained from GAN-generated images are typically inferior to those trained with real images, possibly because the distribution of generated images is biased towards frequent patterns (modes) of the original image distribution [7]. This is especially true in one-shot fine-grained recognition where a tiny difference (e.g., beak of a bird) can make a large difference in class.

In this paper, we develop an approach to apply off-the-shelf generative models to synthesize training data in a way

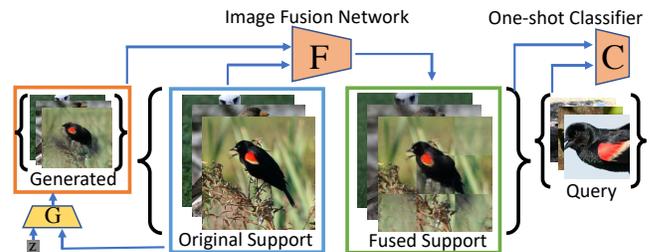


Fig. 1. Our Meta Image Reinforcing Network (MetaIRNet) consists of an image fusion network and a one-shot classifier. The image fusion network reinforces generated images to make them more beneficial for the one-shot classifier by diversifying the images (Figure 2), while the one-shot classifier learns representations that are suitable to classify unseen examples with few examples. Both networks are trained end-to-end, so the loss back-propagates from classifier to the fusion network.

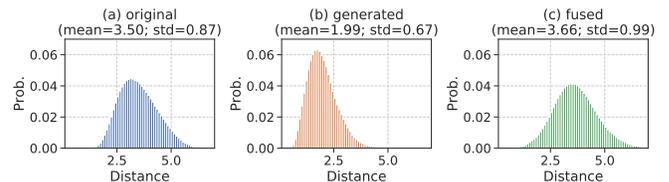


Fig. 2. Distribution of pairwise distances for (a) original set, (b) generated set, and (c) fused set. Our fused images have greater diversity, while generated images are not as diverse as the originals.

that improves one-shot fine-grained classifiers. We begin by conducting a pilot study in which we investigate using a generator pre-trained on ImageNet in a one-shot scenario. We show that the generated images can indeed improve the performance of a one-shot classifier when used with a manually designed rule to combine the generated images with the originals using the weights of a 3×3 block matrix (like Fig. 3 (g)). These preliminary results lead us to consider optimizing these block matrices in a data-driven manner. Thus, we propose a meta-learning approach to learn these

- S. Tsutsui and D. Crandall are with Indiana University, Bloomington, USA. E-mail: {stsutsui,djcran}@indiana.edu
- Y. Fu is with School of Data Science, and MOE Frontiers Center for Brain Science, Shanghai Key Lab of Intelligent Information Processing, Fudan University, China. E-mail: yanweif@fudan.edu.cn.

Manuscript received April 19, 2005; revised August 26, 2015.

block matrices to reinforce the generated images effectively for few-shot classification.

Our approach has two steps. First, an off-the-shelf generator trained from ImageNet is updated towards the domain of novel classes by using only a single image (Sec. 4.1). Second, since previous work and our pilot study (Sec. 3) suggest that simply adding synthesized images to the training data may not improve one-shot learning, the synthesized images are “mixed” with the original images in order to bridge the domain gap between the two (Sec. 4.2). The effective mixing strategy is learned by a meta-learner, which essentially boosts the performance of fine-grained categorization with a single training instance per class. We experimentally validate that our approach can achieve improved performance over baselines on fine-grained classification datasets in one-shot situations (Sec. 5). Moreover, we empirically analyze the mixed images and investigate how our learned mixing strategy reinforces the original images (Sec. 6). As highlighted in Figure 2, we show that while the GAN-generated images lack diversity compared to the original, our mixed images effectively introduce additional diversity.

Contributions of this paper are that we: 1) Introduce a method to transfer a pre-trained generator with a single image; 2) Propose a meta-learning method to learn to complement real images with synthetic images in a way that benefits one-shot classifiers; 3) Demonstrate that these methods improve one-shot classification accuracy on fine-grained visual recognition benchmarks; and 4) Analyze our resulting mixed images and empirically show that our method can help diversify the dataset. A preliminary version of this paper appeared in NeurIPS [8].

2 RELATED WORK

Our paper relates to three main lines of work: GAN-synthesized images for training, few-shot meta-learning, and data augmentation to diversify training examples.

2.1 Image Generation by GANs

Learning to generate realistic images is challenging because it is difficult to define a loss function that accurately measures perceptual photo realism. Generative Adversarial Networks (GANs) [5] address this issue by learning not only a generator but also a loss function — the discriminator — that helps the generator to synthesize images indistinguishable from real ones. This adversarial learning is intuitive but is often unstable in practice [9]. Recent progress includes better CNN architectures [6], [10], training stabilization [9], [11]–[13], and exciting applications (e.g. [14], [15]). BigGAN [6] trained on ImageNet has shown visually-impressive generated images with stable performance on generic image generation tasks. Several studies [16], [17] have explored generating images from few examples, but their focus has not been on one shot classification. Several papers [16], [18], [19] also use the idea of adjusting batch normalization layers, which helped to inspire our work. Finally, work has investigated using GANs to help image classification [7], [20]–[23]; ours differs in that we apply an off-the-shelf generator pre-trained from a large and generic dataset.

2.2 Few-shot Meta-learning

Few shot classification [24] with meta-learning has received much attention after the introduction of MetaDataset [25]. The task is to train a classifier with only a few examples per class. Unlike the typical classification setup, the classes in the training and test sets have no overlap, and the model is trained and evaluated by sampling many few-shot tasks (or episodes). For example, when training a dog breed classifier, an episode might train to recognize five dog species with only a single training image per class — a 5-way-1-shot setting. A meta-learning method trains a meta-model by sampling many episodes from training classes and is evaluated by sampling many episodes from other unseen classes. With this episodic training, we can choose several possible approaches to “learn to learn.” For example, “learning to compare” methods learn a metric space (e.g., [26]–[29]), while other approaches learn to fine-tune (e.g., [3], [30]–[32]) or learn to augment data (e.g., [23], [33]–[36]). Our approach also explores data augmentation by mixing the original images with synthesized images produced by a fine-tuned generator, but we find that the naive approach of simply adding GAN-generated images to the training dataset does not improve performance. However, by carefully combining generated images with the original images, we find that we can synthesize examples that do increase the performance. Thus we employ meta-learning to learn the proper combination strategy.

2.3 Data Augmentation

Data augmentation is often an integral part of training deep CNNs; in fact, AlexNet [37] describes data augmentation as one of “the two primary ways in which we combat overfitting.” Since then, data augmentation strategies have been explored [38], but they are manually designed and thus not scalable for many domain-specific tasks. Recent work uses automated approaches to search for the optimal augmentation policy using reinforcement learning [39] or by directly optimizing the augmentation policy by making it differentiable [40]. Moreover, some researchers perform empirical analysis on data augmentation as a distributional shift [41], or develop a theoretical framework of data augmentation as a Markov process [42]. Our work is most closely related to augmentation based on mixing images. For example, Mixup [43] linearly mixes two random images with a random weight. Manifold Mixup performs a similar operation to the CNN representation of the images [44]. CutMix [45] overlays randomly-cropped images onto other images at random locations. While these methods randomly mix images while ignoring the content of them, our method learns to adjust the mixing technique for given images via meta-learning that optimizes the parameterized mixing strategy to help one-shot learning.

3 PILOT STUDY

To explain how we arrived at our approach, we describe the initial experimentation which motivated our methods.

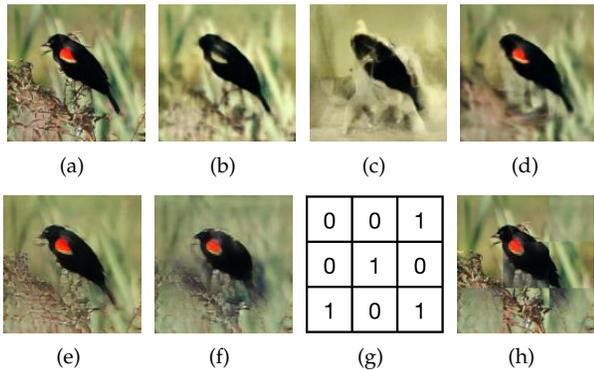


Fig. 3. Samples described in Sec. 3. (a) Original image. (b) Result of tuning noise only. (c) Result of tuning the whole network. (d) Result of tuning batch norm only. (e) Result of tuning batch norm with perceptual loss. (f) Result of slightly disturbing noise from (e). (g) a 3×3 block weight matrix w . (h) Result of mixing (a) and (f) as $w \times (f) + (1 - w) \times (a)$.

TABLE 1

CUB 5-way-1-shot classification accuracy (%) using ImageNet features. Simply adding generated images to the training set does not help, but adding hybrid images, as in Fig. 3 (h), can.

Training Data	Nearest Neighbor	Logistic Regression	Softmax Regression
Original	70.72 ± 0.51	75.39 ± 0.47	74.61 ± 0.48
Original + Generated	70.84 ± 0.51	74.08 ± 0.48	73.55 ± 0.48
Original + Mixed	71.50 ± 0.50	76.07 ± 0.47	75.40 ± 0.47

3.1 How to transfer knowledge from pre-trained GANs?

We aim to quickly generate training images for few-shot classification. Performing adversarial learning (*i.e.*, training a generator and discriminator initialized with pre-trained weights) is not practical when we only have one or two examples per class. Instead, we want to develop a method that does not depend on the number of images at all; in fact, we consider the extreme case where only a single image is available, and want to generate variants of the image using a pre-trained GAN. We tried fixing the generator weights and optimizing the noise so that it generates the target image, under the assumption that slightly modifying the optimized noise would produce a variant of the original. However, naively implementing this idea with BigGAN did not reconstruct the image well, as shown in the example in Fig. 3(b). We then tried also fine-tuning the generator weights, but this produced even worse images stuck in a local minimum, as shown in Fig 3(c).

We speculate that the best approach may be somewhere in between the two extremes of tuning noise only and tuning both noise and weights. Inspired by previous work [16], [18], [19], we propose to fine-tune only scale and shift parameters in the batch normalization layers. This strategy produces better images, as shown in Fig. 3(d). Finally, again inspired by previous work [16], we not only minimize the pixel-level distance but also the distance of a pre-trained CNN representation (perceptual loss [46]), yielding slightly improved results (Fig. 3(e)). We can generate slightly different versions by adding random perturbations to the tuned noise (e.g., the “fattened” version of the same bird in Fig. 3(f)). The entire training process requires fewer than 500 iterations and

takes less than 20 seconds on an NVidia Titan Xp GPU. We explain the generation strategy that we developed based on this pilot study in Sec. 4.

3.2 Do generated images help few-shot learning?

Our goal is not to generate images, but to augment the training data for few shot learning. A naive way to do this is to apply the above generation technique for each training image, in order to double the training set. We tested this idea on a validation set (split the same as [24]) from the Caltech-UCSD bird dataset [47] and computed mean accuracy and 95% confidence intervals on 2000 episodes of 5-way-1-shot classification. We used pre-trained ImageNet features from ResNet18 [48] with nearest neighbor, one-vs-all logistic regression, and softmax regression (or multi-class logistic regression). As shown in Table 1, the accuracy actually drops for two of the three classifiers when we double the size of our training set by generating synthetic training images, suggesting that the generated images are harmful for training classifiers.

3.3 How to synthesize images for few-shot learning?

Given that the synthetic images *appear* meaningful to humans, we conjecture that they can benefit few shot classification when properly mixed with originals to create hybrid images. To empirically test this hypothesis, we devised a random 3×3 grid to combine the images, which is inspired by 3×3 visual jigsaw pretraining [49]. As shown in Fig. 3(h), images (a) and (f) were combined by taking a linear combination within each cell of the grid shown in (g). Finally, we added mixed images like (h) into the training data, and discovered that this produced a modest increase in accuracy (last row of Table 1). While the increase is marginal, these mixing weights were binary and manually selected, and thus likely not optimal. In Sec. 4.2, we show how to learn this mixing strategy in an end-to-end manner using a meta-learning framework.

4 METHOD

The results of the pilot study in the last section suggested that producing synthetic images could be useful for few-shot fine-grained recognition, but only if done in a careful way. In this section, we use these findings to propose a novel technique that does this effectively. We propose a GAN fine-tuning method that works with a single image (Sec. 4.1), and a meta-learning method to not only learn to classify with few examples, but also to learn to reinforce the generated images (Sec. 4.2).

4.1 FinetuneGAN: Fine-tuning Pre-trained Generator for Target Images

GANs typically have a generator G and a discriminator D . Given an input signal $z \sim \mathcal{N}(0, 1)$, a well-trained generator synthesizes an image $G(z)$. In our tasks, we adapt an off-the-shelf GAN generator G that is pre-trained on the ImageNet-2012 dataset in order to generate more images in a target, data-scarce domain. Note that we do not use the discriminator, since adversarial training with a few images is unstable

and may lead to model collapse. Formally, we fine-tune z and the generator G such that G generates an image \mathbf{I}_z from an input vector z by minimizing the distance between $G(z)$ and \mathbf{I}_z , where the vector z is randomly initialized. Inspired by previous work [11], [16], [46], we minimize a loss function \mathcal{L}_G with \mathcal{L}_1 distance and perceptual loss \mathcal{L}_{perc} with earth mover regularization \mathcal{L}_{EM} ,

$$\mathcal{L}_G(G, \mathbf{I}_z, z) = \mathcal{L}_1(G(z), \mathbf{I}_z) + \lambda_p \mathcal{L}_{perc}(G(z), \mathbf{I}_z) + \lambda_z \mathcal{L}_{EM}(z, r), \quad (1)$$

where λ_p and λ_z are coefficients of each term. The first term is the L1 distance of $G(z)$ and \mathbf{I}_z using pixels. The second term is basically L2 distance of $G(z)$ and \mathbf{I}_z but using, instead of pixels, the intermediate feature maps from all convolution layers of ImageNet-trained VGG16 [50]. The last term is the earth mover distance [11] of z and $r \sim \mathcal{N}(0, 1)$ (random noise sampled from the normal distribution).

Since only a few training images are available in the target domain, only scale and shift parameters of the batch normalization of G are updated in practice. Specifically, only the γ and β of each batch normalization layer are updated in each layer,

$$\gamma \left(\frac{x - \mathbb{E}(x)}{\sqrt{\text{Var}(x) + \epsilon}} \right) + \beta, \quad (2)$$

where x is the input feature from the previous layer, and \mathbb{E} and Var indicate the mean and variance functions, respectively. Intuitively and in principle, updating γ and β only is equivalent to adjusting the activation of each neuron in a layer. Once updated, the $G(z)$ would be synthesized to reconstruct the image \mathbf{I}_z . Empirically, a small random perturbation ϵ is added to z as $G(z + \epsilon)$. Examples of \mathbf{I}_z , $G(z)$ and $G(z + \epsilon)$ are illustrated in Fig. 3 (a), (e), and (f), respectively.

4.2 Meta-Reinforced Synthetic Data

4.2.1 One-shot Learning Defined

One-shot classification is a meta-learning problem that divides a dataset into two sets: meta-training (or base) set and meta-testing (or novel) set. The classes in the base and novel sets are disjoint. In other words,

$$\mathcal{D}_{base} = \{(\mathbf{I}_i, y_i), y_i \in \mathcal{C}_{base}\}, \quad (3)$$

$$\mathcal{D}_{novel} = \{(\mathbf{I}_i, y_i), y_i \in \mathcal{C}_{novel}\}, \quad (4)$$

where $\mathcal{C}_{base} \cup \mathcal{C}_{novel} = \emptyset$.

The task is to train a classifier on \mathcal{D}_{base} that can quickly generalize to unseen classes in \mathcal{C}_{novel} with one or few examples. To do this, a meta-learning algorithm performs meta-training by sampling many one-shot tasks from \mathcal{D}_{base} , and is evaluated by sampling many similar tasks from \mathcal{D}_{novel} . Each sampled task (called an episode) is an n -way- m -shot classification problem with q queries, meaning that we sample n classes with m training and q test examples for each class. In other words, an episode has a support (or training) set S and a query (or test) set Q , where $|S| = n \times m$ and $|Q| = n \times q$. One-shot learning means $m = 1$. The notation S_c means the support examples only belong to the class c , so $|S_c| = m$.

4.2.2 Meta Image Reinforcing Network (MetaIRNet)

We propose a Meta Image Reinforcing Network (MetaIRNet), which not only learns a few-shot classifier, but also learns to reinforce generated images by combining real and generated images. MetaIRNet is composed of two modules: an image fusion network F , and a one-shot classifier C .

Image Fusion Network F combines a real image \mathbf{I} and a corresponding generated image \mathbf{I}_g into a new image $\mathbf{I}_{syn} = F(\mathbf{I}, \mathbf{I}_g)$, which will be added into the support set, which means the size of the support set will be doubled. Note that for each real image (regardless of whether it is a positive or negative example) in the support set, we use a single image generated by FinetuneGAN for mixing. While there could be many possible ways to mix the two images (i.e., the design decision of F), we were inspired by 3×3 visual jigsaw pretraining [49] and its data augmentation applications [4]. Thus, as shown in Figure 3(g), we divide the images into a 3×3 grid and linearly combine the cells with the weights \mathbf{w} produced by a CNN conditioned on the two images,

$$\mathbf{I}_{syn} = \mathbf{w} \odot \mathbf{I} + (1 - \mathbf{w}) \odot \mathbf{I}_g, \quad (5)$$

where \odot is element-wise multiplication, and \mathbf{w} is resized to the image size keeping the block structure. The CNN that produces \mathbf{w} extracts the feature vectors of \mathbf{I} and \mathbf{I}_g , concatenates them, and uses a fully-connected layer to produce a weight corresponding to each of the nine cells in the 3×3 grid. Finally, for each real image \mathbf{I}^i , we generate n_{aug} synthetic images, and assign the same class label y_i to each synthesized image $\mathbf{I}_{syn}^{i,j}$ to obtain an augmented support set,

$$\tilde{S} = \left\{ (\mathbf{I}^i, y^i), \left\{ (\mathbf{I}_{syn}^{i,j}, y^i) \right\}_{j=1}^{n_{aug}} \right\}_{i=1}^{n \times m}. \quad (6)$$

One-Shot Classifier C maps an input image \mathbf{I} into feature maps $C(\mathbf{I})$, and performs one-shot classification. Although any one-shot classifier can be used, we choose the non-parametric prototype classifier of Snell *et al.* [27] due to its superior performance and simplicity. During each episode, given the sampled S and Q , the image fusion network produces an augmented support set \tilde{S} . This classifier computes the prototype vector \mathbf{p}_c for each class c in \tilde{S} as an average feature vector,

$$\mathbf{p}_c = \frac{1}{|\tilde{S}_c|} \sum_{(\mathbf{I}_i, y_i) \in \tilde{S}_c} C(\mathbf{I}_i). \quad (7)$$

For a query image $\mathbf{I}_i \in Q$, the probability of belonging to a class c is estimated as,

$$P(y_i = c | \mathbf{I}_i) = \frac{\exp(-\|C(\mathbf{I}_i) - \mathbf{p}_c\|)}{\sum_{k=1}^n \exp(-\|C(\mathbf{I}_i) - \mathbf{p}_k\|)}, \quad (8)$$

where $\|\cdot\|$ is the Euclidean distance. Then, for a query image, the class with the highest probability becomes the final prediction of the one-shot classifier.

Training. In the meta-training phase, we jointly train F and C end-to-end, minimizing a cross-entropy loss,

$$\min_{\theta_F, \theta_C} \frac{1}{|Q|} \sum_{(\mathbf{I}_i, y_i) \in Q} -\log P(y_i | \mathbf{I}_i), \quad (9)$$

where θ_F and θ_C are the learnable parameters of F and C .

TABLE 2
5-way-1-shot accuracy (%) on CUB and NAB datasets with ImageNet pre-trained ResNet18.

Method (+Data Augmentation)	CUB Acc.	NAB Acc.
Nearest Neighbor	79.00 \pm 0.62	80.58 \pm 0.59
Logistic Regression	81.17 \pm 0.60	82.70 \pm 0.57
Softmax Regression	80.77 \pm 0.60	82.38 \pm 0.57
ProtoNet	81.73 \pm 0.63	87.91 \pm 0.52
ProtoNet (+Flip)	82.66 \pm 0.61	88.55 \pm 0.50
ProtoNet (+FinetuneGAN)	79.40 \pm 0.69	85.40 \pm 0.59
ProtoNet (+Gaussian)	81.75 \pm 0.63	87.90 \pm 0.52
ProtoNet (+Mixup)	82.65 \pm 0.59	88.12 \pm 0.52
ProtoNet (+Manifold Mixup)	81.78 \pm 0.58	85.31 \pm 0.51
ProtoNet (+CutMix)	80.81 \pm 0.71	86.12 \pm 0.53
MetaIRNet (+FreezeDGAN)	81.93 \pm 0.62	88.26 \pm 0.57
MetaIRNet (+Jitter)	82.69 \pm 0.63	88.31 \pm 0.55
Ours: MetaIRNet (+FinetuneGAN)	84.13 \pm 0.58	89.19 \pm 0.51
Ours: MetaIRNet (+FinetuneGAN, Flip)	84.80 \pm 0.56	89.57 \pm 0.49

TABLE 3
5-way-1-shot accuracy (%) on CUB dataset with Conv4 and without ImageNet pre-training.

MetaIRNet	ProtoNet [27]	MatchingNet [26]	MAML [3]	RelationNet [28]
65.86 \pm 0.72	63.50 \pm 0.70	61.16 \pm 0.89	55.92 \pm 0.95	62.45 \pm 0.98

5 EXPERIMENTS

To investigate the effectiveness of our approach, we perform 1-shot-5-way classification following the meta-learning experimental setup described in Sec. 4.2. We perform 1000 episodes in meta-testing, with 16 query images per class per episode, and report average classification accuracy and 95% confidence intervals. We use the fine-grained classification dataset of Caltech UCSD Birds (CUB) [47] for our main experiments, and another fine-grained dataset, North American Birds (NAB) [51], for secondary experiments. CUB has 11,788 images with 200 classes, and NAB has 48,527 images with 555 classes.

5.1 Implementation Details

While our fine-tuning method introduced in Sec. 4.1 can generate images for each step in meta-training and meta-testing, it takes around 20 seconds per image, so we apply the generation method ahead of time to make our experiments more efficient. This means that the generator is trained independently. We use a BigGAN pre-trained on ImageNet, using the publicly-available weights. We set $\lambda_p = 0.1$ and $\lambda_z = 0.1$, and perform 500 gradient descent updates with the Adam [52] optimizer with learning rate 0.01 for z and 0.0005 for the fully connected layers, to produce scale and shift parameters of the batch normalization layers. We manually chose these hyper-parameters by trying random values from 0.1 to 0.0001 and visually checking the quality of a few generated images. We only train once for each image, generate 10 random images by perturbing z , and randomly use one of them for each episode ($n_{aug} = 1$). For image classification, we use ResNet18 [48] pre-trained on ImageNet for the two CNNs in F and one in C . Note that we do not share weights among the three CNNs, which means that our model has three ResNets inside. We train F and C with Adam with a default learning rate of 0.001.

We select the best model based on the validation accuracy, and then compute the final accuracy on the test set. We use the same train/val/test split used in previous studies [8], [24] for CUB and NAB, respectively. Further implementation details are available as supplemental source code.¹

5.2 Baselines

Non-meta learning classifiers. We directly train the same ImageNet pre-trained CNN used in F to classify images in \mathcal{D}_{base} , and use it as a feature extractor for \mathcal{D}_{novel} . We then use the following off-the-shelf classifiers: (1) **Nearest Neighbor**; (2) **Logistic Regression** (one-vs-all classifier); (3) **Softmax Regression** (also called multi-class logistic regression).

Meta-learning classifiers. We try the meta-learning method of prototypical network (**ProtoNet** [27]). ProtoNet computes an average prototype vector for each class and performs nearest neighbor with the prototypes. We note that our MetaIRNet adapts ProtoNet as a choice of F so this is an ablative version of our model (MetaIRNet without the image fusion module).

Data augmentation. We compare against simply using the generated images as data augmentation, as well as applying typical data augmentations. Moreover, because our MetaIRNet uses meta-learning to find the best way to mix the original and GAN-generated images, we compare against several alternative ways of mixing them: (1) **Flip** horizontally flips the images; (2) **Gaussian** adds Gaussian noise with standard deviation 0.01 into the CNN features; (3) **FinetuneGAN** (introduced in Sec. 4.1) generates augmented images by fine-tuning the ImageNet-pretrained BigGAN with each support set; (4) **Mixup** [43] mixes two images with a randomly sampled weight; (5) **Manifold Mixup** [44] does mixup in the CNN representation of images; and (6) **CutMix** [45] mixes the two images with randomly sampled locations. We do these augmentations in the meta-testing stage to increase the support set. For fair comparison, we use ProtoNet as the base classifier of all these baselines.

Mix with other images. To evaluate the utility of our generated images, we use our meta-learning technique (MetaIRNet) to mix with images that are *not* from our FinetuneGAN: (1) **FreezeDGAN** [53] fine-tunes GANs by performing adversarial training using a stabilization technique of freezing the discriminator; and (2) **Jitter** produces data-augmented images by randomly jittering the original images.

5.3 Results

As shown in Table 2, our MetaIRNet is superior to all baselines including the meta-learning classifier of ProtoNet (84.13% vs. 81.73%) on the CUB dataset. It is notable that while ProtoNet has worse accuracy when simply using the generated images as data augmentation, our method shows an accuracy increase from ProtoNet, which is equivalent to MetaIRNet without the image fusion module. This indicates that our image fusion module can effectively complement the original images while removing harmful elements from

1. <http://vision.soic.indiana.edu/metairnet/>

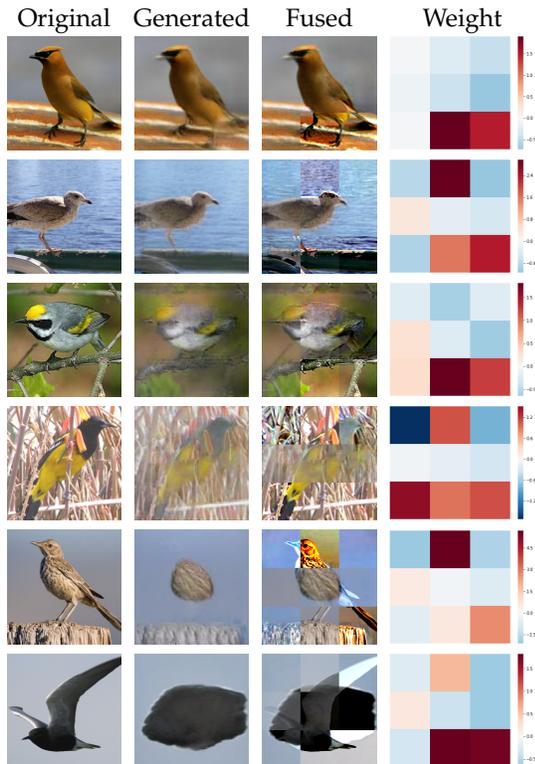


Fig. 4. Samples of original image, generated image, fused image, and mixing weight w . Higher weight (red) means more of the original image was used, and lower weight (blue) means more generated image.

generated ones. Interestingly, horizontal flip augmentation yields nearly a 1% accuracy increase for ProtoNet. Because flipping cannot be learned directly by our method, we conjectured that our method could also benefit from it. The final row of the table shows an additional experiment with our MetaIRNet combined with random flip augmentation, showing an additional accuracy increase from 84.13% to 84.80%. This suggests that our method provides an improvement that is orthogonal to flip augmentation.

Lastly, while most of our experiments focus on the 1-shot cases, we also tested 5-shot and obtained an accuracy of $93.09 \pm 0.30\%$, which is higher than baselines. More details are in Supplementary Material.

Case Studies. We show some sample visualizations in Fig. 4. We observe that image generation often works well but sometimes completely fails. An advantage of our technique is that even in these failure cases, our fused images often maintain some of the object’s shape, even if the images themselves do not look realistic. In order to investigate the quality of generated images in more detail, we randomly pick two classes, sample 100 images for each class, and show a t-SNE visualization of real images (\bullet), generated images (\blacktriangle), and augmented fused images ($+$) in Fig. 7, with classes shown in red and blue. It is reasonable that the generated images are closer to the real ones, because our loss function in Equation (1) encourages this to be so. Interestingly, perhaps due to artifacts of 3×3 patches, the fused images are distinctive from the real and generated images, which extends the decision boundary.

Increasing the number of generated examples. Does our

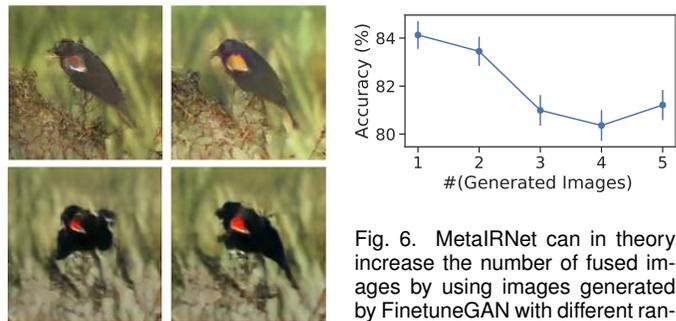


Fig. 5. By fluctuating the input noise, FinetuneGAN can obtain slightly different variants of the original image, which we show in Figure 3(a).

Fig. 6. MetaIRNet can in theory increase the number of fused images by using images generated by FinetuneGAN with different random noise values. However, this does not increase accuracy, presumably because they are conditioned on the same image so adding many of them does not increase diversity.

method benefit by increasing the number of examples generated by FinetuneGAN with different random noise values (Figure 5)? We tried $n_{aug} = 1, 2, 3, 4, 5$ on CUB and the accuracies are shown in Figure 6. Having too many augmented images seems to bias the classifier, and we conclude that the performance gain is marginal or even harmful when increasing n_{aug} . This effect could be because all generated images are conditioned on the same original image, so adding many of them does not significantly increase diversity.

Comparing with other meta-learning classifiers. It is a convention in our community to compare any new technique with previous methods using the accuracies reported in the corresponding literature. The accuracies in Table 2, however, cannot be directly compared with other papers’ reported accuracies as we use ImageNet-pre-trained CNNs. While it is a natural design decision for us to use the pretrained model because our focus is how to use ImageNet pre-trained generators for improving fine-grained one-shot classification, which assumes ImageNet as an available resource off-the-shelf, much of the one-shot learning literature focuses on improving the one-shot algorithms themselves and thus trains from scratch. To provide a comparison, we cite a benchmark study [24] reporting accuracy of other well-known meta-learners [3], [26], [28] on the CUB dataset. To compare with these scores, we experimented with our MetaIRNet and the ProtoNet baseline using the same four-layered CNN. As shown in Table 3, our MetaIRNet performs better than the other methods with more than 2% absolute improvement. We note that this comparison is not totally fair because we use images generated from a generator pre-trained from ImageNet, so one can argue that we use more data than others. However, our contribution is not to establish a new state-of-the-art score but to present the idea of transferring an ImageNet pre-trained GAN for improving one shot classifiers, so we believe this is still informative as it provides a reference score for future work to compare to.

Results on NAB. We also performed similar experiments on the NAB dataset, which is more than four times larger than CUB, and the results are shown in the last column of Table 2. We observe similar results as on CUB, and that our method improves classification accuracy from a ProtoNet baseline (89.19% vs. 87.91%).

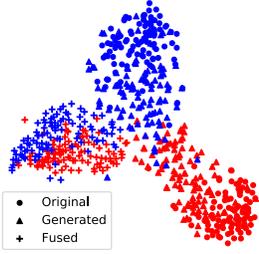


Fig. 7. t-SNE plot of two random classes. We use colors (red and blue) to represent the classes, and use different markers for the three types of images – real images (\bullet), generated images (\blacktriangle), and augmented fused images ($+$)

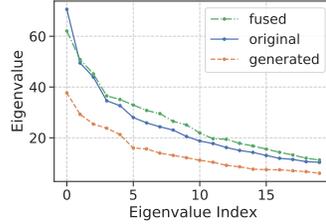


Fig. 8. The sorted eigenvalues of principal component analysis (PCA) for each set. The higher eigenvalues mean that the manifold in the feature space is wider, which suggests greater diversity. Our fused images are at least as diverse as original.

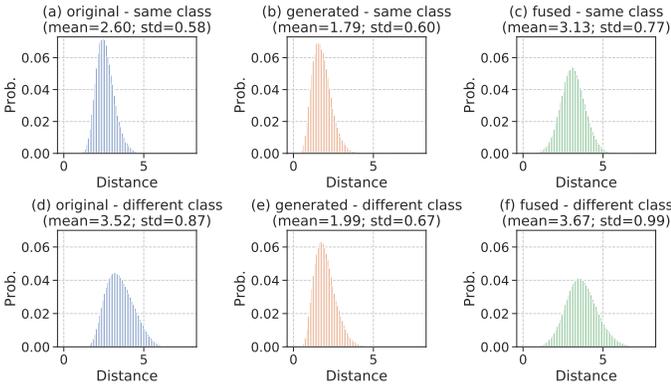


Fig. 9. Distribution of pairwise distances divided by intra-class (same-class) pairwise comparisons and inter-class (different-class) pairwise comparisons. Similar to Figure 2, our fused images have a wider pairwise distribution while generated images are not as diverse as originals, but the amount of increase from the original is more on the same-class comparisons.

6 ANALYSIS

Our proposed technique for reinforcing images generated from fine-tuned GANs improved the few-shot recognition accuracy, but what causes this performance improvement? We hypothesized that the images generated by GANs are not diverse enough on their own, and our learned technique that mixes them with original images helps to diversify the dataset. To validate this hypothesis, we perform several studies investigating the diversity of three image sets: original images, images generated by GANs, and images fused by our method. To measure diversity, we use pairwise distance distributions and principal component analysis (PCA).

6.1 Pairwise distance distribution

One way of quantifying the diversity of an image set is to compute the distance between all possible pairs of images, and then examine the resulting distribution. We compute the Euclidean distances of all possible pairs of images in a set using pretrained CNN representations. If the distribution of the pairwise distances of a set is longer-tailed than others, then we regard the set as more diverse. Figures 2(a), (b), and (c) plot the distributions of original, generated, and fused images, respectively, using the CUB dataset. We observe

that generated images (Figure 2(b)) do not increase pairwise distances from the original set (Figure 2(a)), and actually lower the mean distance from 3.50 to 1.99 and standard deviation from 0.87 to 0.67. In contrast, our fused images (Figure 2(c)) slightly diversify the original set (Figure 2(a)), and increase the mean distance from 3.50 to 3.66 and the standard deviation from 0.87 to 0.99.

6.2 Eigenvalues of PCA

We also employ a quantitative measure of the diversity of the data. For a given image set, we compute its covariance matrix and apply principal component analysis (PCA). PCA can help interpret a high dimensional space by decomposing it into orthogonal subspaces based on the variance of the data. The largest eigenvalues generated by PCA are a measure of the variance in the original dataset. We show a plot of the largest eigenvalues, sorted in decreasing order, in Figure 8. The figure confirms that the generated images have significantly lower eigenvalues than the original and fused images, and the fused images have slightly higher eigenvalues than the original (except for the highest eigenvalue). These observations indicate that the generated images are not diverse on their own, but our technique of fusing makes them at least as diverse as the originals.

6.3 Inter- and Intra-Class Diversity

Figure 9 shows the same histograms of pairwise distances, but split into image pairs that are within the same class and pairs that are in different classes. For the same type of images, it is intuitive that same-class distances are lower than different-class distances. Moreover, for both same-class and different-class, the overall trend is the same as Sec. 6.1: generated images are not as diverse as the originals, but our fused images are more diverse. However, when we compare the distribution of the original and fused images, the same-class comparison sees a greater increase in distance than the different-class comparison: the mean of different-class distances increases from 3.52 (Figure 9(d)) to 3.68 (Figure 9(f)), for a change of +0.16, while the mean of same-class distances increases from 2.60 (Figure 9(a)) to 3.12 (Figure 9(c)), for a change of +0.53. This suggests that our fused images significantly widen the manifolds per class by efficiently mixing the original and generated images.

7 CONCLUSION

We introduce an effective way to employ an ImageNet-pretrained image generator for the purpose of improving fine-grained one-shot classification when data is scarce. Our pilot study found that adjusting only scale and shift parameters in batch normalization can produce visually realistic images. This technique works with a single image, making the method less dependent on the number of available images. Furthermore, although naively adding the generated images into the training set does not improve performance, we show that it can improve performance if we mix generated with original images to create hybrid training exemplars. In order to learn the parameters of this mixing, we adapt a meta-learning framework. We implement this idea and demonstrate a consistent and significant improvement over

several classifiers on two fine-grained benchmark datasets. Furthermore, our analysis suggests that the increase in performance may be because the mixed images are more diverse than the original and the generated images.

REFERENCES

- [1] Y. Wang and M. Hebert, "Learning from small sample sets by combining unsupervised meta-training with cnns," in *NeurIPS*, 2016.
- [2] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "Meta-learning with memory-augmented neural networks," in *ICML*, 2016.
- [3] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *ICML*, 2017.
- [4] Z. Chen, Y. Fu, Y.-X. Wang, L. Ma, W. Liu, and M. Hebert, "Image deformation meta-networks for one-shot learning," in *CVPR*, 2019.
- [5] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *NeurIPS*, 2014.
- [6] A. Brock, J. Donahue, and K. Simonyan, "Large scale GAN training for high fidelity natural image synthesis," in *ICLR*, 2019.
- [7] K. Shmelkov, C. Schmid, and K. Alahari, "How good is my GAN?" in *ECCV*, 2018.
- [8] D. C. Satoshi Tsutsui, Yanwei Fu, "Meta-Reinforced Synthetic Data for One-Shot Fine-Grained Visual Recognition," in *NeurIPS*, 2019.
- [9] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of Wasserstein GANs," in *NeurIPS*, 2017.
- [10] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *ICLR*, 2016.
- [11] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," *arXiv preprint arXiv:1701.07875*, 2017.
- [12] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," in *ICLR*, 2018.
- [13] D. Wang, X. Qin, F. Song, and L. Cheng, "Stabilizing training of generative adversarial nets via langevin stein variational gradient descent," *IEEE Trans Neural Netw Learn Syst*, 2020.
- [14] U. Ojha, Y. Li, C. Lu, A. A. Efros, Y. J. Lee, E. Shechtman, and R. Zhang, "Few-shot image generation via cross-domain correspondence," in *CVPR*, 2021.
- [15] T. Park, J.-Y. Zhu, O. Wang, J. Lu, E. Shechtman, A. A. Efros, and R. Zhang, "Swapping autoencoder for deep image manipulation," in *NeurIPS*, 2020.
- [16] A. Noguchi and T. Harada, "Image generation from small datasets via batch statistics adaptation," in *ICCV*, 2019.
- [17] Y. Wang, C. Wu, L. Herranz, J. van de Weijer, A. Gonzalez-Garcia, and B. Raducanu, "Transferring GANs: generating images from limited data," in *ECCV*, 2018.
- [18] H. De Vries, F. Strub, J. Mary, H. Larochelle, O. Pietquin, and A. C. Courville, "Modulating early visual processing by language," in *NeurIPS*, 2017.
- [19] V. Dumoulin, J. Shlens, and M. Kudlur, "A learned representation for artistic style," in *ICLR*, 2017.
- [20] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training," in *CVPR*, 2017.
- [21] A. Antoniou, A. Storkey, and H. Edwards, "Augmenting image classifiers using data augmentation generative adversarial networks," in *ICANN*, 2018.
- [22] R. Zhang, T. Che, Z. Ghahramani, Y. Bengio, and Y. Song, "MetaGAN: An Adversarial Approach to Few-Shot Learning," in *NeurIPS*, 2018.
- [23] H. Gao, Z. Shou, A. Zareian, H. Zhang, and S.-F. Chang, "Low-shot learning via covariance-preserving adversarial augmentation networks," in *NeurIPS*, 2018.
- [24] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C. F. Wang, and J.-B. Huang, "A closer look at few-shot classification," in *ICLR*, 2019.
- [25] E. Triantafillou, T. Zhu, V. Dumoulin, P. Lamblin, K. Xu, R. Goroshin, C. Gelada, K. Swersky, P.-A. Manzagol, and H. Larochelle, "Meta-dataset: A dataset of datasets for learning to learn from few examples," in *ICLR*, 2020.
- [26] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra *et al.*, "Matching networks for one shot learning," in *NeurIPS*, 2016.
- [27] J. Snell, K. Swersky, and R. S. Zemel, "Prototypical networks for few-shot learning," in *NeurIPS*, 2017.
- [28] 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 *CVPR*, 2018.
- [29] P. Bateni, R. Goyal, V. Masrani, F. Wood, and L. Sigal, "Improved few-shot visual classification," in *CVPR*, 2020.
- [30] A. A. Rusu, D. Rao, J. Sygnowski, O. Vinyals, R. Pascanu, S. Osindero, and R. Hadsell, "Meta-learning with latent embedding optimization," in *ICLR*, 2018.
- [31] C. Finn, K. Xu, and S. Levine, "Probabilistic model-agnostic meta-learning," in *NeurIPS*, 2018.
- [32] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," in *ICLR*, 2017.
- [33] Y.-X. Wang, R. Girshick, M. Hebert, and B. Hariharan, "Low-shot learning from imaginary data," in *CVPR*, 2018.
- [34] B. Hariharan and R. Girshick, "Low-shot visual recognition by shrinking and hallucinating features," in *ICCV*, 2017.
- [35] Z. Chen, Y. Fu, K. Chen, and Y.-G. Jiang, "Image block augmentation for one-shot learning," in *AAAI*, 2019.
- [36] E. Schwartz, L. Karlinsky, J. Shtok, S. Harary, M. Marder, A. Kumar, R. Feris, R. Giryes, and A. Bronstein, "Delta-encoder: an effective sample synthesis method for few-shot object recognition," in *NeurIPS*, 2018.
- [37] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in *NeurIPS*, 2012.
- [38] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," in *AAAI*, 2020.
- [39] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation strategies from data," in *CVPR*, 2019.
- [40] Y. Li, G. Hu, Y. Wang, T. Hospedales, N. M. Robertson, and Y. Yang, "Differentiable automatic data augmentation," in *ECCV*, 2020.
- [41] R. Gontijo-Lopes, S. Smullin, E. D. Cubuk, and E. Dyer, "Tradeoffs in data augmentation: An empirical study," in *ICLR*, 2021.
- [42] T. Dao, A. Gu, A. Ratner, V. Smith, C. De Sa, and C. Ré, "A kernel theory of modern data augmentation," in *ICML*, 2019.
- [43] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "Mixup: Beyond empirical risk minimization," in *ICLR*, 2018.
- [44] V. Verma, A. Lamb, C. Beckham, A. Najafi, I. Mitliagkas, D. Lopez-Paz, and Y. Bengio, "Manifold mixup: Better representations by interpolating hidden states," in *ICML*, 2019.
- [45] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "Cutmix: Regularization strategy to train strong classifiers with localizable features," in *ICCV*, 2019.
- [46] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *ECCV*, 2016.
- [47] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The Caltech-UCSD Birds-200-2011 Dataset," California Institute of Technology, Tech. Rep. CNS-TR-2011-001, 2011.
- [48] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.
- [49] M. Noroozi and P. Favaro, "Unsupervised learning of visual representations by solving jigsaw puzzles," in *ECCV*, 2016.
- [50] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2014.
- [51] G. Van Horn, S. Branson, R. Farrell, S. Haber, J. Barry, P. Ipeiritis, P. Perona, and S. Belongie, "Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection," in *CVPR*, 2015.
- [52] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.
- [53] S. Mo, M. Cho, and J. Shin, "Freeze the discriminator: a simple baseline for fine-tuning GANs," in *CVPR Workshop AICC*, 2020.