Cyclic Data Augmentation Generative Adversarial Network

Siyuan Zhou 120033910030
Yuanhang Yin 120033910057
Yunfan He 120033910183

Abstract

Few-shot generation problems, mainly based on deep generative models, have become an attractive direction of artificial intelligence in recent years, which could strongly augment training data, alleviate overfitting, and provide creative images or music. However these problems are difficult to solve due to the lack of data. Antoniou et al. proposed an effective model Data Augmentation Generative Adversarial Networks (DAGAN) using generative adversarial network, but suffer from unrealistic results. Inspired by DAGAN, we proposed Cyclic Data Augmentation Generative Adversarial Network (CDAGAN), which add a cyclic supervisor between input and output to learn the implicit invariance among images of the original class. Experimental results of our proposed method on various datasets Omniglot, EMNIST, and VGG-Faces show remarkable improvements compared to DAGAN, which demonstrate the effectiveness of our model.

1. Introduction

Deep neural networks (DNN) have achieved extraordinary results, through the training of large amounts of data. In the past decade, DNN has been widely used in image classification, speech recognition, object detection, machine translation, and other fields. However, in some cases, it is difficult to obtain a large amount of data, which creates the problem of few-shot learning. In these problems such as few-shot image classification, we need to learn a model based on a very limited dataset to achieve the goal. Due to the lack of data and the complexity of traditional deep neural network models, methods based on traditional neural networks are prone to overfit, resulting in poor results in the test set. Much research in the field of few-shot learning has obtained notable results. For example, the MAML [1] proposed by Finn et al. has achieved excellent results in the field of few-shot image classification through meta-learning. Data augmentation can also be applied to few-shot learning, based on a given small number of training samples, more diverse and realistic new samples can be generated. This requires us to use the method of generative models to obtain the implicit distribution of the training data.

Deep generative models are one of the most powerful and promising directions in the field of generative models, which have achieved unprecedented performance and have been more and more widely used in recent years. In these models by using the combination of conditional distributions defined by deep neural networks, the generation process learns the density function of the samples through deep neural networks (Variational Auto Encoder) or learns an implicit density model to directly generate new samples in the same distribution (Generate Adversarial Network). Due to the powerful fitting capabilities of deep neural networks, deep generative models can generate realistic samples. Among them, the GAN-based models have achieved promising results in various domains like image generation, attracting a lot of attention in the field of machine learning. However, a major bottleneck of these generative models is the need for a large amount of training data and training time to better fit these data and generate samples from the same distribution, which limits the use of deep generative models in the field of data augmentation. Therefore, it is necessary to learn a general generative model so that it can generate new images of the same category based on only a few images of a new category. This is the problem of few-shot image generation, which is the task we focus on.

In the few-shot image generation problem, the model is first trained in the source domain with sufficient training samples of seen categories. Then, the model will be applied to the target domain, generate more realistic images for unseen categories based on a few samples in these categories. Previous work such as DAWSON [2] used a GAN, which first trains the model on seen categories, and then transfers the trained model to unseen categories for fine-tuning, has quick adaption to the new unseen categories, and can generate similar images of these categories. DAGAN [3] uses a random Gaussian noise to be concatenated with features extracted from a conditional image to generate more distinctive images. However, the image produced by this method has small differences compared with the original image, and it is difficult to obtain satisfactory results that are both realistic and different from the original image.

In this paper, we follow the architecture of DAGAN and make further modifications by considering the underlying invariance between the input images and the output ones. To be specific, we map the generated images back to the original domain and add a reconsturction restriction between the input domain and the cyclic reconstructed domain. We change the one-directional mapping to a bi-directional one to better investigate the invariance knowledge when more diverse and realistic new samples are generated. In few-shot image generation, it is crucial to find the class-agnostic knowledge for transferring from seen categories to unseen novel categories. We find that the cyclic invariance we investigate can help with the transferring process and facilitate generation for novel categories.

2. Related Work

Data Augmentation. Data augmentation is a technique that increases the number of training samples by slightly modifying existing data, or generating new data directly based on existing data. In training deep learning models, data augmentation can help suppress over-fitting to obtain models with better generalization capabilities. Traditional data augmentation methods, including cropping, rotation, and adding Gaussian noise, can increase the diversity of training data. However, this diversity is very limited. In recent years, advanced data augmentation methods, such as deep generative models, can dig out the internal distribution of data, thereby generating more diverse and realistic new samples. Our work is based on this new data augmentation method, which can generate more images for training sets.

Generative Adversarial Network. Generative Adversarial Network (GAN) [4] is a deep generative model based on adversarial learning, which defines two networks generator and discriminator. GAN can learn the complex density distribution of the training samples, while the discriminative network distinguishes the candidates generated by the generative network based on the training samples. Early unconditional GAN is based on a random vector z, which can generate realistic-looking new samples. Since there is no need to know the actual data distribution, it has been widely used in image processing, sequence data, and computer vision. Furthermore, conditional GAN [] can generate target images based on one or more conditional images. Some recent conditional GAN attempts to solve few-shot image generation, i.e. learn the distribution of images with several conditional images based on a few training images to generate new images. The focus of our work is to use GAN to solve the task of few-shot image generation.

Few-Shot Generation. Few-shot generation is a new direction of few-shot learning in which feature information is extracted from a few samples to obtain the potential distribution of data, thereby generating new samples. Early work of few-shot learning, such as Bayesian Program Learning [5] by Lake et al., learned simple concepts such as pen stroke by inputting both the images and stroke data, and hierarchically combined these simple concepts to generate new images. Rezende et al. [6] used a sequence generation model and constructed a Variational Auto Encoder (VAE) with an attention module to serially infer the images to be generated. Generative Adversarial Network (GAN) based methods such as DAGAN by Antoniou et al. [3] used an image conditional generative adversarial network, can learn meta information from samples in the source domain to generate new samples in the target domain. Novelly, the matching networks of Bartunov et al. [7] used a memory-assisted network to quickly learn new concepts through an attention mechanism. Methods based on combining meta-learning and GAN, such as DAWSON [2], can quickly adapt to new domains and obtain new samples by applying the MAML algorithm [1] to GAN. More recently, Hong et al. proposed an F2GAN [8] that defines a two-step model that first fuse conditional images and then fill high-level details to obtain more realistic and diverse generated images. Our work is based on DAGAN and further adds an additional cyclic reconstruction process to learn the invariance knowledge during the image generation process.

3. Methodology

3.1. Basic Representation

Our work starts from Generative Adversarial Methods, which are one approach for learning to generate examples that can match the density of a training dataset $D = \{x_1, x_2, \cdots, x_{N_D}\}$ of size denoted by N_D. The objective is to minimize a distribution discrepancy measure between the generated data and the true data. To clearly explain our method, we first introduce a typical Generative Adversarial Network (GAN) which can be formed by

$$z = \tilde{\mathbf{N}}(0, I) \tag{1}$$

$$v = f(z) \tag{2}$$

where v are generated vectors and z are latent variables which follow Gaussian distribution. f is a mapping functions, usually implemented as a neural network. As expected, v should be able to represent the distribution of data D and z provides the variation to the generation results.

Specifically, a generative adversarial network can be used to map out a data augmentation manifold, *i.e.*, DAGAN (see Figure 1). Given an input x, we can learn a latent representation of input $r = g(x)$. By integrating the representation of input into the basic form of GAN, we can acquire the combined formulas in the form of conditional GAN:

$$r = g(x) \tag{3}$$

$$z = \tilde{\mathbf{N}}(0, I) \tag{4}$$

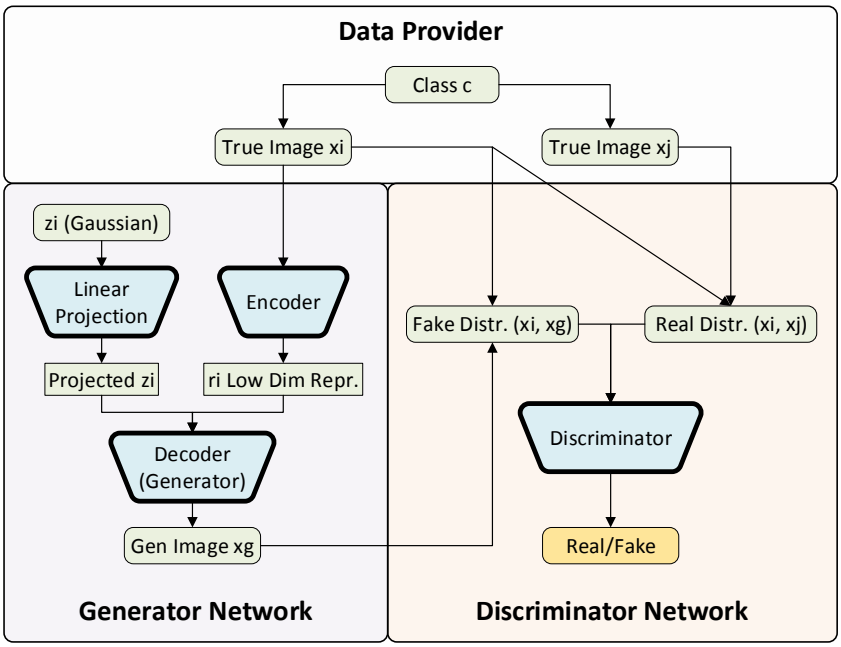

Figure 1. **Architecture of DAGAN.** Left: the generator network is composed of an encoder taking an input image (from class c), projecting it down to a lower dimensional manifold (bottleneck). A random vector (z_i) is transformed and concatenated with the bottleneck vector; these are both passed to the decoder network which generates an augmentation image. Right: the adversarial discriminator network is trained to discriminate between the samples from the *real* distribution (other real images from the same class) and the *fake* distribution (images generative from the generator network). Adversarial training leads the network to generate new images from an old one that appear to be within the same class (whatever that class is), but look different enough to be a different sample.

$$x_g = f(z, r) \tag{5}$$

Now f takes random z and representation r as input to reconstruct x. These formulas follow the basic idea of the classic conditional GAN, where r is the condition to clarify the target domain of generation. Intuitively, r can specifie the domain distribution of the generated images in the conditional GAN model of DAGAN. By virtue of this model, we can acquire the meaningful representation $r^* = g(x^*)$ of any new input x^*. In addition to that, we can generate extra augmentation data $v_1^*, v_2^*, \ldots$ that supplements the original x^* by sampling z from the standard Gaussian distribution and feed it to the generative network.

3.2. Cyclic Data Augmentation

As discussed in the last section, traditional DAGAN augment the input domain by mapping the latent representation to a new generated domain, implying a kind of unidirectivity knowledge. However, the correlation and coherence between the original domain and the new domain are not explicitly supervised. Only the discriminator can tell the implicit distinction between the two domains.

In order to further investigate the data augmentation ability of the generative model, we aim to exploit an additional explicit consistency on the top of the generator network of DAGAN. Our basic idea is to form a symmetry cycle between the input and output of the generation process. To be specific, our goal is to further map the generated value x_g back to the original domain of input x. In this way, we can form a cycle between the input and the generated output. We call this architecture cyclic-DAGAN (see Figure 2 for details). This kind of cyclic consistency can help the training process of the generated images because it forces an additional supervision on the generated domain which helps to share the discrimination difficulty of the discriminator. The cyclic-DAGAN in Figure 2 can be represented in the mathematical form:

$$r = g_1(x) \tag{6}$$

$$z = \tilde{\mathbf{N}}(0, I) \tag{7}$$

$$x_g = f_1(z, r) \tag{8}$$

$$s = g_2(x_g) \tag{9}$$

$$z' = \tilde{\mathbf{N}}(0, I) \tag{10}$$

$$x_c = f_2(z', s) \tag{11}$$

The first three formulas are in the same form as DAGAN. That is, with a representation function g_1 and a mapping

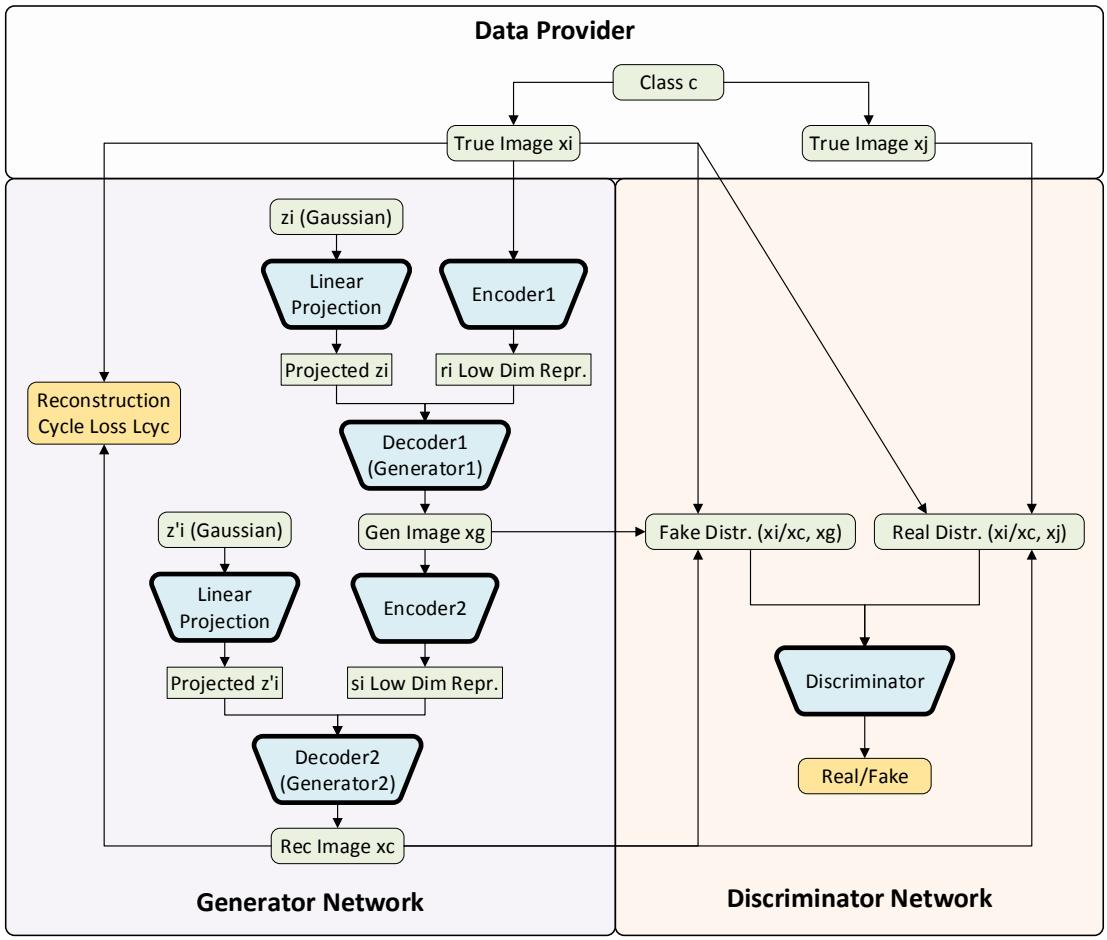

Figure 2. **Architecture of our cyclic-DAGAN.** Cyclic-DAGAN is built based on the architecture of DAGAN (see Figure 1). Left: The augmentation image x_g goes through a secondary encoder to obtain anther lower dimensional manifold (bottleneck) s_i. A random vector (z_i') is transformed and concatenated with s_i; these are both passed to a secondary decoder network which outputs the reconstruction image x_c. A reconstruction cycle loss is employed between x_i and x_c. Right: The input of the discriminator contains both the original image x_i and the reconstruction image x_c. We hope that the discriminator can not distinguish between x_i and x_c after enough steps of training.

function f_1, we obtain the generated image x_g from the input image x as the conventional methods do. Symmetrically, we apply another representation function g_2 and another mapping function f_2 in order to map the generated x_g to a new image domain x_c, as seen in the last three formulas. z' and s are the random vector the intermediate representation of input for the secondary generation process, respectively. For ease of representation, we can use $x_g = f_1 \circ g_1(x, z)$ and $x_c = f_2 \circ g_2(x_g, z')$ to succinctly express the two generation process. Since we hope that x_c can well represent the characteristic of x from the source domain, we further employ a reconstruction cycle loss between these two domains in the $L1$ form:

$$\mathbf{L}_{cyc} = \|x_c - x\|_1 \tag{12}$$

3.3. Optimization

The optimization of the overall cyclic-DAGAN network consists of two main parts: 1) adversarial training and 2) reconstruction training. We will discuss about these two parts as follows.

Consider a source domain consisting of data $D = \{x_1, x_2, \cdots, x_{N_D}\}$ and corresponding target values $\{t_1, t_2, \cdots, t_{N_D}\}$. Like the implementation in DAGAN, we use an improved WGAN critic to learn our cyclic data augmentation model in an adversarial approach. The only difference is that our cyclic-DAGAN not only considers the images from the source domain, but only those from the reconstructed domain. Therefore, the WGAN critic takes:

(a) some input data point x_i and a second data point x_j from the same class such that $t_i = t_j$.

(b) some input data point x_i and the output x_g of the current generator which takes x_i as an input.

(c) the reconstructed output $x_{i,c}$ and a second data point x_j from the same class such that $t_i = t_j$.

(d) the reconstructed output $x_{i,c}$ and the generated x_g of the current generator which takes x_i as an input.

The above critic not only tries to discriminate the generated points (b) from the real points (a), but also tries to discriminate the generated points (d) from the reconstructed points (c). The generator is trained to minimize this discriminative capability as measured by the Wasserstein distance $\mathbf{L}_{WGAN}$. To be specific, we use $\mathbf{L}_{WGAN,src}$ to represent the Wasserstein distance between the source domain and the generated domain. We use $\mathbf{L}_{WGAN,rec}$ to represent the Wasserstein distance between the reconstructed domain and the generated domain. These two Wasserstein distances make up the final advasarial loss. Therefore, the final WGAN loss can be formed by

$$\mathbf{L}_{WGAN} = \mathbf{L}_{WGAN,src} + \lambda \mathbf{L}_{WGAN,rec},$$

where λ is a coefficient that control the weight of the second distance. At the beginning of training, λ should be small because the reconstruction quality is low and thus does harm to the adversarial training quality. When the reconstruction loss $\mathbf{L}_{cyc}$ gets smaller as the training step goes on, the reconstruction quality becomes better, in which case λ should be larger. In our design, λ should a function inversely proportional to $\mathbf{L}_{cyc}$. For simplicity, we use a variation of the exponential function as the form of function ϕ:

$$\lambda = \phi(\mathbf{L}_{cyc}) = e^{-\mathbf{L}_{cyc}}$$

We can see that $\lambda \to 1$ when $\mathbf{L}_{cyc} \to 0$ and $\lambda \to 0$ when $\mathbf{L}_{cyc} \to \infty$.

The final loss of our cyclic-DAGAN model is the summation of the WGAN loss and the cyclic reconstruction loss:

$$\begin{aligned}\mathbf{L}_{overall} &= \mathbf{L}_{WGAN} + \mathbf{L}_{cyc} \\ &= \mathbf{L}_{WGAN,src} + \lambda \mathbf{L}_{WGAN,rec} + \mathbf{L}_{cyc} \\ &= \mathbf{L}_{WGAN,src} + e^{-\mathbf{L}_{cyc}} \mathbf{L}_{WGAN,rec} + \mathbf{L}_{cyc}\end{aligned}$$

Our optimization objective is to minimize $\mathbf{L}_{overall}$:

$$\min \quad \mathbf{L}_{WGAN,src} + e^{-\mathbf{L}_{cyc}} \mathbf{L}_{WGAN,rec} + \mathbf{L}_{cyc}$$

4. Experiment Setups

4.1. Implementation Details

Baseline. We follow DAGAN for most of the settings and thus directly compare the experimental results of our Cyclic-DAGAN with them.

Architecture. The two generators we use share the same architecture but have separate parameters to update. Each generator consists of a UNet and ResNet, which we henceforth call a UResNet. The UResNet generator has a total of 8 blocks, each block having 4 convolutional layers (with leaky rectified linear (relu) activations and batch renormalisation (batchrenorm) followed by one downscaling or upscaling layer. Downscaling layers (in blocks 1-4) were convolutions with stride 2 followed by leaky relu, batch normalisation and dropout. Upscaling layers were stride 1/2 replicators, followed by a convolution, leaky relu, batch renormalisation and dropout. For Omniglot and EMNIST experiments, all layers had 64 filters. For the VGG-Faces the first 2 blocks of the encoder and the last 2 blocks of the decoder had 64 filters and the last 2 blocks of the encoder and the first 2 blocks of the decoder 128 filters. In addition, each block of the UResNet generator had skip connections. As with a standard ResNet, a strided 1x1 convolution also passes information between blocks, bypassing the between block non-linearity to help gradient flow. Finally skip connections were introduced between equivalent sized filters at each end of the network.

We used a DenseNet discriminator, using layer normalization instead of batch normalization; the latter would break the assumptions of the WGAN objective function. The DenseNet was composed of 4 Dense Blocks and 4 Transition Layers. We used a growth rate of k = 64 and each Dense Block had 4 convolutional layers within it. For the discriminator we also used dropout at the last convolutional layer of each Dense Block as we found that this improves sample quality.

We trained each Cyclic-DAGAN for 500 epochs, using a learning rate of 0.0001, and an Adam optimizer with Adam parameters of $\beta_1 = 0$ and $\beta_2 = 0.9$. For each classification experiment we used a DenseNet classifier composed of 4 Dense Blocks and 4 Transition Layers with a k = 64, each Dense Block had 3 convolutional layers within it. The classifiers were a total of 17 layers (i.e. 16 layers and 1 softmax layer). Furthermore, we applied a dropout of 0.5 on the last convolutional layer in each Dense Block. The classifier was trained with standard augmentation: random Gaussian noise was added to images (with 50% probability), random shifts along x and y axis (with 50% probability), and random 90 degree rotations (all with equal probability of being chosen). Classifiers were trained for 200 epochs, a learning rate of 0.001, and an Adam optmizer with $\beta_1 = 0.9$ and $\beta_2 = 0.99$.

Datasets. We use the same datasets as in DAGAN, *i.e.*, Omniglot, EMNIST, and VGG-Faces. All datasets were split randomly into source domain sets, validation domain

Experiment ID	Samples Per Class	Test Accuracy
Omni_5_Standard	5	0.689904
Omni_5_DAGAN_Augmented	5	0.821314
Omni_5_CDAGAN_Augmented	5	0.842060
Omni_10_Standard	10	0.794071
Omni_10_DAGAN_Augmented	10	0.862179
Omni_10_CDAGAN_Augmented	10	0.869816
Omni_15_Standard	15	0.819712
Omni_15_DAGAN_Augmented	15	0.874199
Omni_15_CDAGAN_Augmented	15	0.883094

Table 1. Omniglot CDAGAN Augmented Classification

Experiment ID	Samples Per Class	Test Accuracy
EMNIST_Standard	15	0.739353
EMNIST_DAGAN_Augmented	15	0.760701
EMNIST_CDAGAN_Augmented	15	0.772471
EMNIST_Standard	25	0.783539
EMNIST_DAGAN_Augmented	25	0.802598
EMNIST_CDAGAN_Augmented	25	0.805145
EMNIST_Standard	50	0.815055
EMNIST_DAGAN_Augmented	50	0.827832
EMNIST_CDAGAN_Augmented	50	0.832134
EMNIST_Standard	100	0.837787
EMNIST_DAGAN_Augmented	100	0.848009
EMNIST_CDAGAN_Augmented	100	0.857231

Table 2. EMNIST CDAGAN Augmented Classification

sets and test domain sets.

For classifier networks, all data for each character (handwritten or person) was further split into 2 test cases (for all datasets), 3 validation cases and a varying number of training cases depending on the experiment. Classifier training was done on the training cases for all examples in all domains, with hyperparameter choice made on the validation cases. Finally test performance was reported only on the test cases for the target domain set. Case splits were randomized across each test run.

For one-shot networks, Cyclic-DAGAN training was done on the source domain, and the meta learning done on the source domain, and validated on the validation domain. Results were presented on the target domain data. Again in the target domain a varying number of training cases were provided and results were presented on the test cases.

Omniglot dataset contains 1623 differents handwritten characters from 50 different alphabets. Each of the 1623 characters was drawn online via Amazon's Mechanical Turk by 20 different people. Each image is paired with stroke data, a sequences of $[x, y, t]$ coordinates with time (t) in milliseconds. The Omniglot data was split into source domain and target domain. The order of the classes was shuffled such that the source and target domains contain diverse samples (i.e. from various alphabets). The first 1200 were used as a source domain set, 1201-1412 as a validation domain set and 1412-1623 as a target domain test set.

The EMNIST dataset is a set of handwritten character digits derived from the NIST Special Database 19 and converted to a 8x28 pixel image format and dataset structure that directly matches the MNIST dataset. The EMNIST data was split into a source domain that included classes 0-34 (after random shuffling of the classes), the validation domain set included classes 35-42 and the test domain set included classes 42-47. Since the EMNIST dataset has thousands of samples per class we chose only a subset of 100 for each class, so that we could make our task a low-data one.

In the VGG-Face dataset case, we randomly chose 100 samples from each class that had 100 uncorrupted images, resulting in 2396 of the full 2622 classes available in the dataset. After shuffling, we split the resulting dataset into a source domain that included the first 1802 classes. The test domain set included classes 1803-2300 and the validation domain set included classes 2300-2396.

Experiment ID	Samples Per Class	Test Accuracy
VGG-Face_Standard	5	0.0446948
VGG-Face_DAGAN_Augmented	5	0.125969
VGG-Face_CDAGAN_Augmented	5	0.123470
VGG-Face_Standard	15	0.39329
VGG-Face_DAGAN_Augmented	15	0.429385
VGG-Face_CDAGAN_Augmented	15	0.438982
VGG-Face_Standard	25	0.579942
VGG-Face_DAGAN_Augmented	25	0.584666
VGG-Face_CDAGAN_Augmented	25	0.601597

Table 3. Face CDAGAN Augmented Classification

Technique Name	Test Accuracy
Pixel Distance	0.267
Pixel Distance + DAGAN Augmentations	0.605
Pixel Distance + CDAGAN Augmentations	0.624
Matching Nets	0.938
Neural Statistician	0.931
Prototypical Networks	0.96
Siam-I	0.884
Siam-II	0.92
GR + Siam-I	0.936
GR + Siam-II	0.912
SRPN	0.948
Matching Nets (Local Reproduction)	0.969
Matching Nets + DAGAN Augmentations	0.974
Matching Nets + CDAGAN Augmentations	0.976

Table 4. Omniglot One-shot Result. Note that DAGAN's local implementation of matching networks substantially outperforms the matching network results presented in the original paper. Our CDAGAN experiment is based on this local implementation.

4.2. Image Classification

Our first experiment is to test how well the CDAGAN can augment vanilla classifiers trained on each target domain. In the fist case, a DenseNet classifier was trained on just real data (with standard data augmentation) with 5, 10 or 15 examples per class. In the second case, the classifier was also passed generated augmented data. The real or fake label was also passed to the network, to enable the network to learn how best to emphasise true over generated data. This last step proved crucial to maximizing the potential of the augmentations. In each training cycle, varying numbers of augmented samples were provided for each real example (ranging from 1-10); the best annotation rate was selected via performance on the validation domain. The results on the held out test cases from the target domain are respectively given in Table 1 for Omniglot, Table 2 for EMNIST and Table 3 for VGG-Face. In every case the augmentation improves the classification. It can be seen that our CDAGAN based augmentation can consistently performs better than DAGAN based augmentation, verifying the effectiveness of our generation method.

4.3. One-shot learning

We follow the standard one-shot learning approach in DAGAN to execute our one-shot classification experiment. Similarly, we learn an appropriate distance between representations that can be used with a nearest neighbour classifier. We focus on the use of Matching Networks to learn a representation space, where distances in that representation space produce good classifiers. The same networks can then be used in a target domain for nearest neighbour classification. A matching network creates a predictor from a support set in the target domain by using an attention memory network to generate an appropriate comparison space for comparing a test example with each of the training examples.

By augmenting the support sets and then learning to learn from that augmented data, we can enhance the classification power of matching networks, and apply that to the augmented domain. Precisely, we train the CDAGAN on

the source domain, then train the matching networks on the source domain, along with a sample-selector neural network that selects the best representative z input used to create an additional datum that augments each case. Augmentation was used during every matching network training episode to simulate the data augmentation process. We used matching networks without full-context embedding version and stacked K GAN generated (augmented) images along with the original image. We ran experiments for 20 classes and one sample per class per episode, *i.e.*, the one shot learning setting. Table 4 shows the Omniglot one-shot results. We can see that the CDAGAN was enhancing even simple pixel distance with large improvement. Furthermore, in our matching network experiments, we saw an improvement of 0.2% over the original DAGAN.

5. Conclusion

In this paper, we presented Cyclic Data Augmentation Generative Adversarial Network, a more effective conditional generative model which can generate realistic and diverse images given input conditional images. One of the principal innovations is that we add a novel cyclic supervisor, which can exploit correlation and coherence between the original domain and the new domain. We conducted various experiments on several datasets, which demonstrate the effectiveness of our method.

References

[1] Finn C, Abbeel P, Levine S. Model-agnostic meta-learning for fast adaptation of deep networks[J]. arXiv preprint arXiv:1703.03400, 2017.

[2] Liang W, Liu Z, Liu C. DAWSON: A Domain Adaptive Few Shot Generation Framework[J]. arXiv preprint arXiv:2001.00576, 2020.

[3] Antoniou A, Storkey A, Edwards H. Data augmentation generative adversarial networks[J]. arXiv preprint arXiv:1711.04340, 2017.

[4] Goodfellow I, Pouget-Abadie J, Mirza M, et al. Generative adversarial nets[C]//Advances in neural information processing systems. 2014: 2672-2680.

[5] Lake B, Salakhutdinov R, Gross J, et al. One shot learning of simple visual concepts[C]//Proceedings of the annual meeting of the cognitive science society. 2011, 33(33).

[6] Rezende D J, Mohamed S, Danihelka I, et al. One-shot generalization in deep generative models[J]. arXiv preprint arXiv:1603.05106, 2016.

[7] Bartunov S, Vetrov D. Few-shot generative modelling with generative matching networks[C]//International Conference on Artificial Intelligence and Statistics. 2018: 670-678.

[8] Hong Y, Niu L, Zhang J, et al. F2GAN: Fusing-and-Filling GAN for Few-shot Image Generation[C]//Proceedings of the 28th ACM International Conference on Multimedia. 2020: 2535-2543.