FCC-GAN: A Fully Connected and Convolutional Net Architecture for GANs

Sukarna Barua
The University of Melbourne
Victoria-3010, Australia

Sarah Monazam Erfani
The University of Melbourne
Victoria-3010, Australia

James Bailey
The University of Melbourne
Victoria-3010, Australia

ABSTRACT

Generative Adversarial Networks (GANs) are a powerful class of generative models. Despite their successes, the most appropriate choice of a GAN network architecture is still not well understood. GAN models for image synthesis have adopted a deep convolutional network architecture, which eliminates or minimizes the use of fully connected and pooling layers in favor of convolution layers in the generator and discriminator of GANs. In this paper, we demonstrate that a convolution network architecture utilizing deep fully connected layers and pooling layers can be more effective than the traditional convolution-only architecture, and we propose FCC-GAN, a fully connected and convolutional GAN architecture. Models based on our FCC-GAN architecture learn both faster than the conventional architecture and also generate higher quality of samples. We demonstrate the effectiveness and stability of our approach across four popular image datasets.

KEYWORDS

Generative adversarial networks, GAN architecture, fully connected layers

1 INTRODUCTION

Generative Adversarial Networks (GANs) have attracted wide attention as powerful generative models. A GAN model consists of a generator network that generates fake samples from random noise vectors, and a discriminator network that provides probabilistic feedback about how well the generated samples resemble a real distribution. Initially formulated by [5], GANs have been successfully used for image generation [4, 27], image in-painting [25], image super-resolution [11] and text-to-image conversion [28].

Despite their attractive theoretical framework, it can be difficult to train GANs in practice, due to instability issues such as mode collapse and vanishing gradients [1, 5]. There has been considerable work investigating the stabilization of GAN training via the use of different network architectures [4, 16, 27, 35] and objective functions [1, 6, 34]. However, the interaction between the underlying network architecture and the sample generation process of GANs is still not well understood. *Indeed, the choice of what architectures are most appropriate to use for GANs remains an open problem.*

Existing research usually deploys Deep Convolutional Networks (CNNs) for image synthesis tasks in GANs. Specifically, in traditional architectures, convolution layers are mostly used, and fully connected and pooling layers are eliminated or minimized [1, 6, 18, 27]. Such practices have been largely inspired by the DCGAN model [27], which achieved significant image quality and training stability through its adoption of the all convolutional [32] architecture. A set of transposed convolution layers are used in the generator for upsampling, while strided-convolution layers are used in the discriminator to downsample images. For high resolution image synthesis, convolution networks based on ResNet [7] blocks instead of simple convolution layers are employed [6, 18, 24]. The final network architectures are usually formed with only a single fully connected layer in the generator (to receive input noise) and discriminator (to produce the output probability). Whilst the use of such convolution-only architecture has been effective for delivering high training stability and sample quality for GANs, in this paper *we demonstrate that the use of multiple fully connected layers together with convolution layers can perform even better than the conventional architecture*. Thus, we propose FCC-GAN, a fully connected and convolutional net architecture for GANs.

Conventional GAN generators approach image generation as a single process achieved by a deep convolutional network. In contrast, our philosophy is to model image generation as two separate tasks: (i) conversion of the low-dimensional input noise to a high-dimensional intermediate representation of image features, and (ii) conversion of these features to an output image. An FCC-GAN generator uses deep fully connected layers for the first task and convolution layers for the second. The fully connected network used for mapping noise vectors to image features learns a consistent relationship between different noise vectors and image features leading to more natural images. Indeed our premise is that convolution layers are not well suited for such global mapping operations, due to an over emphasis on local connectivity enforced by shared-weights.

Similarly, FCC-GAN employs deep fully connected layers after a series of convolution layers in the discriminator. In conventional discriminators, convolution layers extract high-dimensional image features which are directly passed to an output layer for classification. In contrast, an FCC-GAN discriminator uses the deep fully connected network to map the high-dimensional features to a lower-dimensional space before final classification. We demonstrate that discrimination in this lower-dimensional space can prevent the discriminator loss from becoming too low. A higher loss for the discriminator results in larger gradients and faster learning for the generator. Finally, we show that the use of pooling in the discriminator is more effective than the conventional choice of strided-convolution layers for our FCC-GAN architecture.

To summarize, we investigate and revisit two widely adopted architectural assumptions made in GANs: (i) use of convolution-only architecture minimizing the number of fully connected layers to just one in both the generator and discriminator, and (ii) use of strided-convolution replacing pooling layers. We demonstrate that these conventions are not necessarily the best choices and that more effective architectures are possible by combining *deep fully connected and pooling layers* together with convolution layers. Our proposed FCC-GAN architecture demonstrates significant improvements over conventional convolution models in terms of sample quality, learning speed, and training stability. Our contribution can be summarized as follows:

- We propose FCC-GAN, an architecture consisting of deep fully connected and convolution layers for both the generator and discriminator in GANs. Our proposed architecture generates higher quality samples than conventional architectures on a variety of benchmark image datasets.
- We demonstrate that FCC-GAN has a faster learning curve than the conventional architecture, and can produce recognizable good quality images after just a few epochs of training.
- Models based on FCC-GAN architecture perform better than those of the conventional CNN architecture on the benchmark datasets in term of Inception score and Frétchet Inception Distance.
- Our proposed architecture exhibits higher stability than the conventional architecture, and is able to avoid mode collapse for a wide variety of experimental settings.

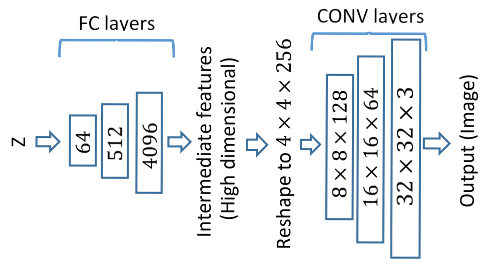
2 RELATED WORK

In this section, we briefly review the existing literature on GANs. At the end of this section, we describe how our work is different from existing research.

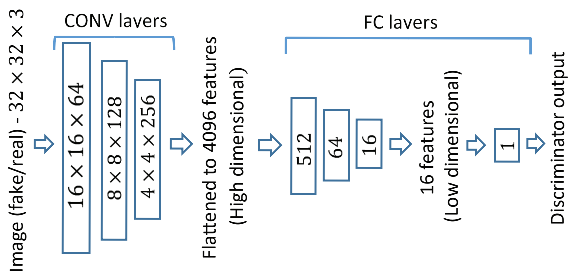
GANs were first formulated in [5], which demonstrated their potential as a generative model. GANs became popular for image synthesis based on successful use of deep convolution layers [4, 27]. The generator in the standard model maps a single noise vector to an output distribution [5, 27]. Additional information in the form of latent codes can be combined with the noise vector to control output attributes and improve the sample quality [3, 17, 24, 31]. For example, class labels combined with noise vectors as input to the generator can generate supervised data [17]. Side-information such as image captions and bounding box localization can also be combined with class information to improve the quality of images [29, 33]. Maximizing mutual information between the input latent variables and the GAN outputs can produce a disentangled and interpretable representation [3]. In semi-supervised GAN models, the discriminator is trained to predict the correct label for each real sample, in addition to discriminating real and fake data [23, 24, 30]. Such models produce better image quality than their unsupervised counterparts [24, 30].

Optimizing the standard GAN objective function was shown to be similar to minimizing the Jensen Shannon (JS) divergence [5] between the real distribution and generator's distribution. Moreover, the traditional optimization process of GANs is a special case of more general variational divergence estimation [21]. Minimizing the JS divergence of two distributions, which have non-overlapping support can result in vanishing gradients hurting the training of GANs [1]. A loss function based on Wasserstein distance (WGAN) was introduced to improve stability [1]. WGAN training was further improved by penalizing the gradients in the loss function [6] and adding a consistency term [34]. GANs can also be trained using auto-encoder based discriminators [35], various types of f-divergences [22], and other objective functions such as Loss-Sensitive GAN [26] and Least Squares GAN [15]. Weight normalization in the discriminator [18] and large batch sizes [2] has been shown to be very useful in high resolution image synthesis.

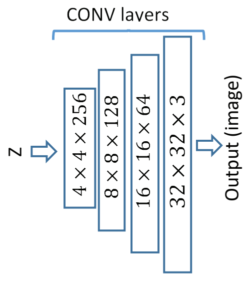
However, most existing works on image synthesis have adopted the conventional deep convolution architecture in the generator and discriminator of GANs (not using or minimizing the use of fully connection layers). In this aspect, our work is different since we investigate an architecture composed of both deep fully connected and convolution layers. *To the best of our knowledge, our work is the first to demonstrate a network combining convolution layers with three or more fully connected layers is effective for GANs.*



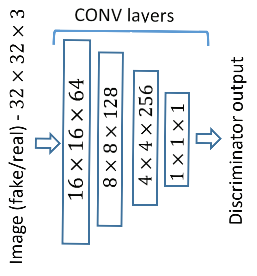
(a) FCC-GAN generator



(b) FCC-GAN discriminator



(c) Conventional generator



(d) Conventional discriminator

Figure 1: A simple example of generator and discriminator networks in the proposed FCC-GAN architecture (a-b) and conventional CNN architecture (c-d). The models generate $32 \times 32 \times 3$ RGB images from random noise vector z. Numbers inside the boxes denote the output shape of the layer (for CONV layers) or number of nodes (for FC layers). Note that the first (last) layer of the generator (discriminator) in conventional architecture can be considered a fully connected layer, which is employed at the input (output). For high resolution image synthesis, simple convolution layers are usually replaced with ResNet [7] blocks.

3 FCC-GAN: A FULLY CONNECTED AND CONVOLUTONAL GAN ARCHITECTURE

We propose two simple modifications to the conventional principles of GAN architecture by (i) incorporating multiple fully connected layers in both the GAN generator and discriminator networks and (ii) using pooling layers with unit stride convolution in place of strided convolution layers in the discriminator. Figures 1(a-b) visualizes an example model of our proposed FCC-GAN architecture for generating $32 \times 32 \times 3$ images. The generator consists of two parts: a series of fully connected (FC) layers, followed by a series of convolution (CONV) layers. The FC part receives a low-dimensional (e.g., 100) noise vector z and progressively converts it to a high-dimensional (4096 features) intermediate representation of image features. This representation is then reshaped ($4 \times 4 \times 256$) for a convolution block. A series of transposed convolution layers convert the intermediate features to an output image ($32 \times 32 \times 3$). The discriminator is constructed with a stack of convolution layers followed by a stack of FC layers. The convolution layers extract high-dimensional ($4\times4\times256$) features from input images ($32\times32\times3$). The extracted features are flattened and fed to an FC part which progressively maps them to a lower-dimensional space for classification by an output layer. Average pooling layers are used in the discriminator for downsampling operation which implies unit-stride convolution layers are used instead of strided convolutions. A comparison of FCC-GAN architecture versus the conventional CNN architecture is provided in Figures 1(c-d).

Specifically, we propose the following architectural modifications to the conventional deep convolution architecture:

- Use of multiple deep fully connected layers before convolution layers in the generator to convert the low-dimensional noise vector to a high-dimensional representation of image features.
- Use of multiple deep fully connected layers in the discriminator to map the high-dimensional features extracted by convolution layers to a lower-dimensional space before classification.
- Use of average pooling with unit-stride convolution in the discriminator, instead of the conventional choice of strided-convolution.

The architecture shown in Figures 1(a-b) can be adapted for images of different resolutions by appropriately modifying the shape and depth of the convolution stack. In this study, we demonstrate the approach on four popular image datasets of three different resolutions (28×28, 32×32, and 64×64 pixels).

3.1 Rationale and Benefits of the FCC-GAN Architecture

(i) Mapping noise to intermediate image features— The FC network in the generator of FCC-GAN serves three purposes. First, it learns the required non-spatial mapping from random noise vectors to intermediate image features. Second, it captures the inherent relationship between different noise vectors that should be mapped to similar features of the same class of images. Third, it address the problem of shared weights in the conventional architecture, which prevent the convolution layers from generating subtle variations in different spatial zones of the same convolution filter. These variations are needed to produce realistic images. Inclusion of initial fully connected layers facilitates capture of these subtle variations, and enforces them in the intermediate features. This helps the generator to produce more natural images. Fully connected layers are better than convolution layers to realize these objectives because their mapping is more non-spatial and the shared weights of convolution layers restrict their capacity to learn such general transformations.

(ii) Discrimination in a low-dimensional feature space— Conventional discriminators extract high-dimensional features from an input image using a series of strided-convolution layers. These features are then passed to an output layer, either a convolution layer or a fully connected node. Building a decision boundary in this high-dimensional feature space to separate real and fake data samples has two disadvantages: (1) Constructing the boundary is easier (too easy) in this high-dimensional space, therefore the discriminator loss decreases quickly to very small values. For GANs, a very small discriminator loss is harmful for training the generator since the gradients become very small or vanish completely. (2) In a high-dimensional space, the distance between the decision boundary and class regions is more likely to be large. Thus, the gradients may point to random directions and may not be very accurate to train the generator, deteriorating convergence rate. Such situations are more likely to occur at the beginning of training when the generator's distribution is completely different from the real one.

The fully connected layers in the FCC-GAN discriminator are used to map the high-dimensional image features to a lower dimensional space before classification. Such forcing of the final output node to discriminate in a low-dimensional space serves two purposes: (1) it brings the decision boundary closer to class regions, thus making the gradient directions more accurate, and (2) it is comparatively harder to build the decision boundary in a low-dimensional space. This prevents the discriminator loss from becoming small too quickly, especially at the initial stages of training.

Our motivations for using FC layers is partly empirical (see later experiments), and partly due to the following reasons. First, having no shared-weights, FC layers are more independent to learn the required non-spatial dimensional mapping task. Second, FC layers are typically harder to train for images than convolution layers are. Hence, they assist in preventing the low discriminator loss. Indeed a larger loss for the discriminator after the addition of FC layers is clearly evident in all of our experiments. In addition, this is complemented by a low generator loss at the beginning, suggesting that the higher loss actually helps the generator learn quickly.

(iii) Average pooling in the discriminator— Conventional GAN models have mostly favored the use of strided-convolution in the discriminator for downsampling images, e.g., [1, 27]. As we show later in experiments, use of average pooling results in a performance boost for our FCC-GAN architecture. Average pooling adds regularization in the feature extraction process by averaging features from adjacent spatial zones. We believe that such regularization becomes particularly effective when followed by deep fully connected layers.

Table 1: Generator and discriminator network used for CIFAR-10 and SVHN datasets for CNN and FCC-GAN models. CONV(x, y, z) is a convolution layer with filters=x, kernel=$y \times y$, and stride=z. CONVT(x, y, z) is a transposed convolution layer with filters=x, kernel=$y \times y$, and stride=z. FC(x) is a fully connected layer with x output nodes. BN implies a batch-normalization layer and R implies reshape. For simplicity, the activation functions are now shown. The architecture of FCC-GAN model is shown with strided convolution.

CNN Model	FCC-GAN Model
Generator network	
Input: Z(100)	Input: Z(100)
R(1,1,100)	FC(64), FC(512), FC(4096), BN
CONVT(256,4,1), BN	R(4,4,256)
CONVT(128,4,2), BN	CONVT(128,4,2), BN
CONVT(64,4,2), BN	CONVT(64,4,2), BN
CONVT(3,4,3)	CONVT(3,4,3)
Output: (32, 32, 3)	Output: (32, 32,3)
Discriminator network	
Input: (32,32,3)	Input: (32,32,3)
CONV(64,4,2), BN	CONV(64,4,2), BN
CONV(128,4,2), BN	CONV(128,4,2), BN
CONV(256,4,2), BN	CONV(256,4,2), BN
CONV(1,4,1)	FC(512), FC(64), FC(16), FC(1)
Output: 1	Output: 1

4 EXPERIMENTS

We evaluate the effectiveness of FCC-GAN architecture, comparing it with the conventional CNN architecture. We train GAN models using both architectures on four benchmark image datasets: MNIST, CIFAR-10, SVHN [20], and CelebA [13]. To demonstrate the generalizability of FCC-GAN architecture, we test models using two different objective functions used widely in the literature: the standard GAN objective function [5] and the Wasserstein distance from WGAN [1]. For this study, we only focus on unsupervised GAN training. Codes for FCC-GAN implementation can be found in *https://github.com/sukarnabarua/fccgan*.

Experimental Settings: We train the FCC-GAN models twice: once using strided convolution in the discriminator and once using average pooling with unit-stride convolution. This was done to demonstrate the impact of pooling. While reporting the results for both of these models, we differentiate them as FCC-GAN-S (with strided convolution) and FCC-GAN-P (with pooling). In other places, we simply use FCC-GAN to refer to the latter model with pooling. Table 1 shows the network architecture of the CNN and FCC-GAN-S models used for CIFAR-10 and SVHN datasets. For FCC-GAN-P model, we just replace each strided CONV(x, y, 2) layer of the discriminator with a CONV(x, y, 1) layer followed by an average pooling layer with pool size of 2×2.

For building the CNN architecture, we adopted the suggestions of the DCGAN and WGAN methods. Batch-normalization [9] was used in both the generator and discriminator (see Table 1). In all layers except the output, ReLU [19] activation was used in the generator and LeakyReLU [14] with a leaky slope of 0.2 in the discriminator. The generator used Tanh output whilst in the discriminator, Sigmoid was used for standard GAN training and no activation was used for WGAN training. For MNIST dataset, we changed the kernel size of the convolution layers to match the dimension of the MNIST dataset (28×28). Note that the first (last) convolution layer in the generator (discriminator) of CNN models can be considered equivalent to a fully connected layer. For further details on architecture, please see Appendix A.

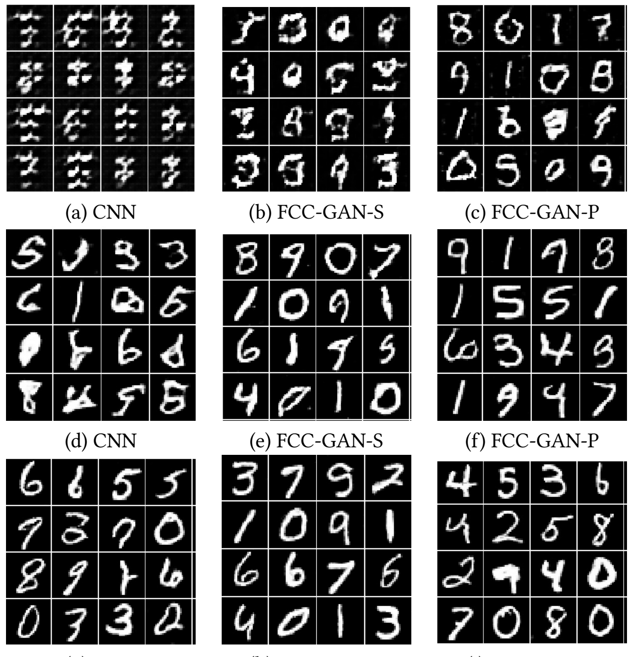


Figure 2: 28×28 pixel MNIST images for standard GAN training. Images generated by conventional CNN and FCC-GAN models after 1 epoch (a-c), 5 epochs (d-f), and 50 epochs (g-i)

For all experiments, we used the ADAM [10] optimizer for standard GAN training and RMSProp for WGAN training, following the suggestions from DCGAN and WGAN methods. For a fair comparison, we kept all training parameters such as batch size, learning rate, and number of iterations, the same for all compared models during training. For standard GAN training, we ran a different number of iterations for different datasets: 60 epochs (approx. $94K$ iterations) on MNIST, 100 epochs (approx. $228K$ iterations) on SVHN, and 150 epochs (approx. $234K$ iterations) on CIFAR-10. The number of epochs were kept the same for WGAN training.

We compare the architectures based on the visual quality of generated samples, as well as the Inception score [30] and Frétchet Inception Distance (FID) [8], two popular metrics for measuring visual quality and diversity of GAN outputs. We compute the Inception score for CIFAR-10 and SVHN datasets following the method proposed by [30]. For computing the score on the MNIST dataset, we followed the approach of [12]. For computing FID, we followed the approach proposed by the original authors [8].

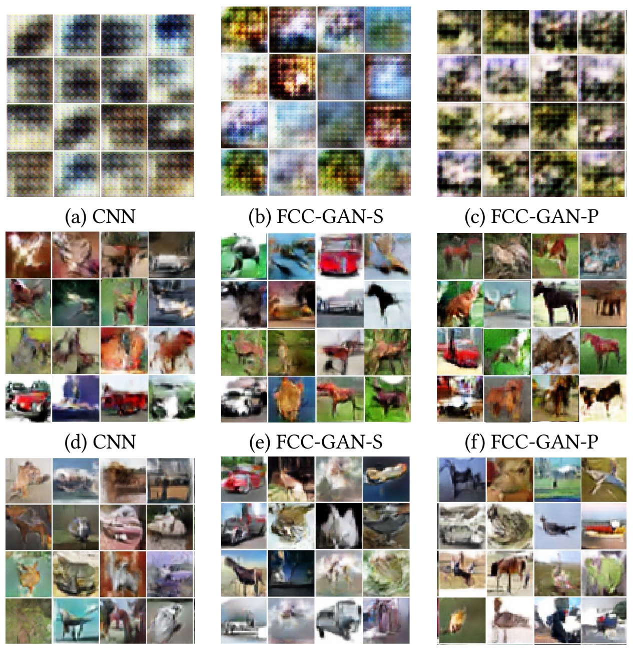


Figure 3: 32×32 pixel CIFAR-10 images for standard GAN training. Images generated by conventional CNN and proposed FCC-GAN models after 1 epoch (a-c), 35 epochs (d-f), and 150 epochs (g-i)

Results on Standard GAN Training: We summarize results of standard GAN training for the two architectures. Figure 2 shows the images generated after epoch 1, 5, and 50 for the MNIST dataset. We observe that after just one epoch, the images generated by the FCC-GAN models contain some recognizable digit structures, whereas such structures are not present for the CNN model. Both the FCC-GAN models learn the distribution much more quickly than the CNN model. After five epochs, FCC-GAN models generate clearly recognizable digits, while the CNN model does not. After epoch 50, all models generate good images, though FCC-GAN models still outperform the CNN model in terms of image quality.

Table 2: Inception scores (higher is better) for standard GAN training for conventional CNN and proposed FCC-GAN architectures.

Model	MNIST	CIFAR-10	SVHN
CNN	8.68 ± 0.03	6.37 ± 0.07	2.97 ± 0.03
FCC-GAN-S	9.34 ± 0.02	6.72 ± 0.08	3.08 ± 0.04
FCC-GAN-P	**9.48 ± 0.03**	**7.25 ± 0.05**	**3.23 ± 0.03**

Figure 3 shows the images generated by the three models for the CIFAR-10 dataset. Comparing the images generated after epoch 1 and 35, we observe that the FCC-GAN models' outputs are visually superior to those of the CNN model. The results on SVHN show similar trends to the experiments for MNIST and CIFAR-10 (see Figure 4 for the SVHN images produced by the three models for epochs 1, 10, and 100).

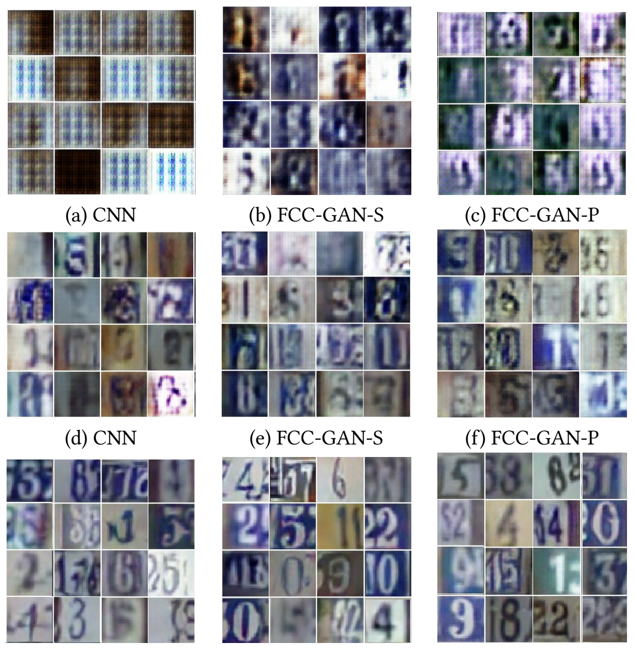


Figure 4: 32×32 pixel SVHN images for standard GAN training. Images generated by conventional CNN and proposed FCC-GAN models after 1 epoch (a-c), 10 epochs (d-f), and 100 epochs (g-i).

Table 3: FIDs (lower is better) for standard GAN training for conventional CNN and proposed FCC-GAN architectures.

Model	MNIST	CIFAR-10	SVHN
CNN	8.07 ± 0.20	40.84 ± 0.34	19.22 ± 0.16
FCC-GAN-S	4.69 ± 0.12	41.47 ± 0.17	18.46 ± 0.11
FCC-GAN-P	**4.23 ± 0.02**	**34.07 ± 0.18**	**15.56 ± 0.11**

In Table 2 and Table 3, we report the best Inception scores and FIDs obtained by the three models. FCC-GAN models obtain better scores than the CNN model. Specifically, FCC-GAN-P model obtains the best results and outperforms the other models in terms of both Inception score and FID.

Table 4: Inception scores (higher is better) for WGAN training for conventional CNN and proposed FCC-GAN models.

Model	MNIST	CIFAR-10	SVHN
CNN	7.00 ± 0.03	5.33 ± 0.06	2.83 ± 0.02
FCC-GAN-S	7.25 ± 0.04	5.42 ± 0.06	**3.03 ± 0.02**
FCC-GAN-P	**8.67 ± 0.03**	**5.60 ± 0.07**	3.00 ± 0.01

Results on WGAN Training: Results for WGAN training are similar to those for standard GAN. Tables 4 and 5 show the best Inception scores and FIDs obtained by the three models. We see

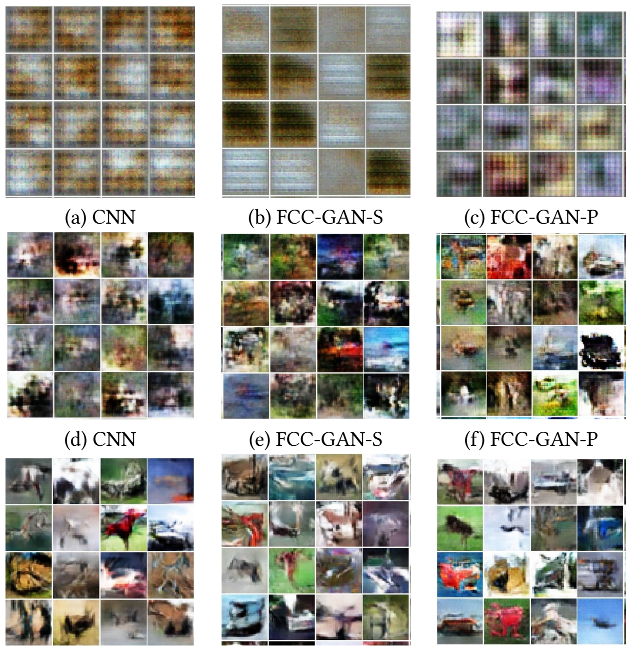

(a) CNN (b) FCC-GAN-S (c) FCC-GAN-P
(d) CNN (e) FCC-GAN-S (f) FCC-GAN-P
(g) CNN (h) FCC-GAN-S (i) FCC-GAN-P

Figure 5: 32×32 pixel CIFAR-10 images for WGAN training. Images generated by conventional CNN and proposed FCC-GAN models after 1 epoch (a-c), 10 epochs (d-f), and 150 epochs (g-i)

Table 5: FIDs (lower is better) for WGAN training for conventional CNN and proposed FCC-GAN models.

Model	**MNIST**	**CIFAR-10**	**SVHN**
CNN	30.17 ± 0.25	62.21 ± 0.20	53.60 ± 0.12
FCC-GAN-S	30.76 ± 0.01	58.33 ± 0.15	**43.82 ± 0.48**
FCC-GAN-P	**20.26 ± 0.18**	**56.89 ± 0.22**	69.60 ± 0.30

that for all experiments, FCC-GAN models achieved better scores than the CNN model in WGAN training as well. In particular, FCC-GAN-P model was superior among the three models outperforming the other two in two of the three datasets. For space constraints, we only show the generated images for CIFAR-10 in Figure 5. We observe that the images generated by the FCC-GAN models are visually superior to those of the CNN models.

Higher Discriminator Loss: In this section, we empirically demonstrate that the FCC-GAN discriminator has higher loss values than the CNN discriminator. Consider Figure 6(a) where we plot the loss of CNN and FCC-GAN (with pooling) models over different epochs of standard GAN training on MNIST. We observe that the discriminator of the FCC-GAN model has a higher loss values than that of the CNN model across all epochs. On average, FCC-GAN model has 62% higher loss than the CNN model. This higher loss has the potential to provide larger gradients to help the generator learn the data distribution faster.

Lower Generator Loss: We compare the loss of the generators of the two architectures in Figure 6(b) for the same experiment on MNIST. The FCC-GAN generator achieves lower loss values compared to the CNN model. In particular, comparing the loss pattern of the two generators at the early stages of training (epochs 1-5), we see that the loss of the FCC-GAN model sharply decreases in contrast to an increase of loss for the CNN model. As training progresses, the discriminator improves and the generator loss starts to increase (after epoch 7 in the figure).

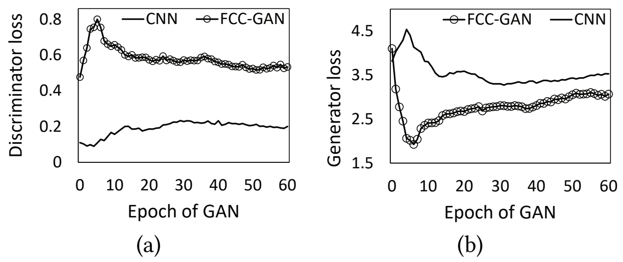


Figure 6: (a) Discriminator loss of the CNN and FCC-GAN models for MNIST dataset (b) Generator loss of the CNN and FCC-GAN models for MNIST dataset

Table 6: Number of epochs needed by the CNN and FCC-GAN models to achieve a desired Inception score for MNIST dataset. FCC-GAN model converges much faster than the CNN model.

	Inception score (higher is better)				
Method	≥ 6.00	≥ 7.00	≥ 8.0	≥ 8.5	≥ 9.0
FCC-GAN	1	2	3	4	8
CNN	3	5	12	26	-

Faster Convergence: The loss pattern of the generator and discriminator of FCC-GAN is actually associated with a faster learning of the data distribution. To illustrate this, in Table 6, we report the number of epochs needed by both models to achieve a desired Inception score for a standard GAN training on MNIST. The Inception score for the MNIST real dataset is 9.95. From Table 6, we see that, FCC-GAN obtained an Inception score above 6.00 after just 1 epoch of training, whilst CNN requires 3 epochs. After 3 epochs, FCC-GAN achieves an score above 8.0 compared to 11 epochs needed by CNN. Notably, after just 8 epochs, Inception score of the FCC-GAN reached above 9.0. The CNN model did not achieve an Inception score above 9.0 even after running the model for 100 epochs (above 187K generator iterations). Table 7 provides similar comparisons for a standard GAN training on CIFAR-10. We see that FCC-GAN achieves Inception scores above 6.5 after epoch 41 and 7.0 after epoch 112. The CNN model could not obtain an Inception score above 6.5 even after running for 150 epochs (above 234K generator iterations). These results demonstrate a huge speedup in convergence of FCC-GAN compared to the standard CNN architecture.

Experiments on 64×64 pixel CelebA dataset: To assess the performance of FCC-GAN on higher resolution datasets, we evaluated it on 64×64 resolution CelebA [13] daataset. We trained both

Table 7: Number of epochs needed by the CNN and FCC-GAN models to achieve a desired Inception score for CIFAR-10 dataset. FCC-GAN model converges much faster than the CNN model.

	Inception score (higher is better)				
Method	≥ 5.0	≥ 5.5	≥ 6.0	≥ 6.5	≥ 7.0
FCC-GAN	18	24	32	41	112
CNN	26	34	81	-	-

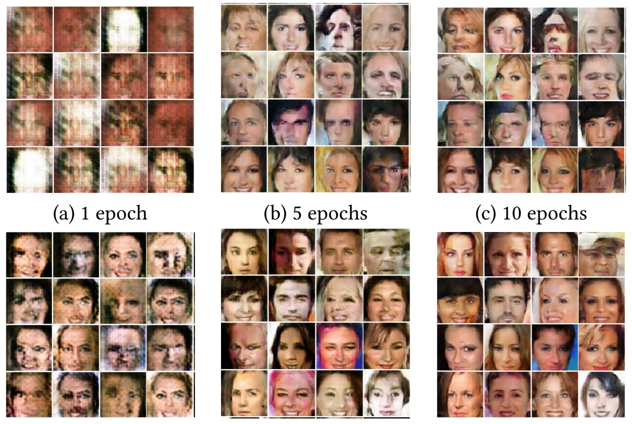


Figure 7: 64×64 **pixel CelebA images generated by conventional CNN model (a-c) and proposed FCC-GAN model (d-f) after different epochs of an standard GAN training**

CNN and FCC-GAN models for 50 epochs using standard GAN training. The models were similar to previous experiments with an additional convolution layer for upsampling (downsampling) in the generator (discriminator) to match with the higher resolution. Detailed architecture can be found in Appendix A.3. Adam optimizer was used with a learning rate of 0.0001 and decay of 0.0001. Figure 7 shows the generated images by both CNN and FCC-GAN models after different epochs of the training. The higher image quality and faster learning of the FCC-GAN model are clearly visible in the figure for all epochs.

Experiments with ResNet: In this section, we demonstrate the performance of FCC-GAN architecture with ResNet models. Previously, GAN models [6, 18] based on ResNet architecture showed significant performance improvement over models build with simple convolution layers. To verify whether FCC-GAN architecture can benefit ResNet GAN models, we modify the ResNet model used by the WGAN-GP method for CIFAR10 dataset, and add fully connected layers at the beginning (end) of the generator (discriminator) network. We added three FC layers in the generator having 64, 512, and 2048 nodes, respectively, in each layer. The FC configuration is similar to the one shown in Table 1 except that the last (third) FC layer has 2048 nodes to match with the input filter of the subsequent ResNet blocks (as used by WGAN-GP experiments). We added two FC layers having 16 and 1 node at the end of the ResNet

Table 8: Inception scores (higher is better) for CNN and FCC-GAN models on CIFAR10 dataset over different architectures of varying convolution layers. N_1 **and** N_2 **denote the number of convolution layers used in the generator and discriminator of CNN model, respectively. Note that for FCC-GAN model, the first (last) convolution layer in the generator (discriminator) is removed to accommodate three fully connected layers. The models were trained for 50 epochs using standard GAN training.**

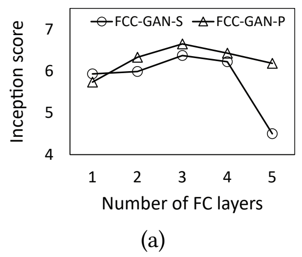
	Number of convolution layers, $N_1 + N_2$			
Method	4+4	4+7	7+4	7+7
CNN	5.98	5.63	3.69	4.24
FCC-GAN	6.69	6.91	6.84	6.84

discriminator. The ResNet blocks in the original WGAP-GP discriminator outputs a vector of 128 dimensions, thus we only needed two FC layers to reduce the dimension further to 16 and then produce the final output. The value of all hyper-parameters, e.g., batch size, optimization algorithm, learning rate, and decay, were kept similar to the ones used in the original experiments [6]. The ResNet GAN model was run for $100K$ generator iterations on the CIFAR10 dataset. We found the standard GAN loss function to perform better than the WGAN loss function for FCC-GAN ResNet models. Table 9 shows the best Inception scores obtained on the CIFAR10 dataset by the original WGAN-GP ResNet model and FCC-GAN ResNet model in our unsupervised GAN training. Our FCC-GAN architecture on ResNet achieved higher (better) Inception scores than the original ResNet model used by WGAN-GP experiments. The results demonstrate that the proposed FCC-GAN architecture can substantially improve the performance of ResNet GAN models.

Table 9: Inception scores obtained by the original WGAN-GP ResNet model and FCC-GAN ResNet model in unsupervised GAN training on CIFAR10 dataset.

Model	**Inception Score**
WGAN-GP ResNet [6]	7.86 ± 0.07
FCC-GAN ResNet	**8.02 ± 0.08**

Impact of convolution network depth on CNN and FCC-GAN architecture: One may argue that the performance benefit of FCC-GAN model results from higher network depths (achieved due to the inclusion of FC layers in both generator and discriminator). To evaluate this argument, we perform GAN experiments by increasing the network depth of CNN models and compare the results with those of FCC-GAN models. We increase the number of convolution layers of CNN models from 4 (as shown in Table 1) to 7 in the generator network, discriminator network, and both networks together. We achieved this by adding an unit-stride convolution layer with 3×3 kernel after (before) each upsampling (downsampling) convolution layer in the generator (discriminator) (Please see Appendix A.4 for detailed network architecture). Then we trained the models on CIFAR10 dataset using standard GAN training for 50 epochs. The best Inception scores are shown in the third row of Table 8. We see that increasing the depth of CNN model did not result in better Inception scores. In fact, CNN models with 7 convolution layers in the generator did not converge (obtained low Inception scores of 3.69 and 4.64). However, similar increase of depth for FCC-GAN models results in a performance boost. As can be seen from the Inception scores of last row of Table 8, larger architectures (4+7, 7+4, and 7+7) get better Inception scores than the smallest (4+4) architecture for FCC-GAN models. We also observe that the smallest FCC-GAN model (4+4) obtains better Inception score than all four CNN models. These results imply that the benefits of FCC-GAN architecture results necessarily from the inclusion of FC layers, and not merely from the increase of network depth. Choosing a different set of hyper-parameters might have an impact on the performance of the higher depth CNN models; however, for fair comparison we stick to the same hyper-parameters for all experiments. Although increasing convolution network depth did not improve performance of the simple CNN models (as used in our experiments), high depth networks are usually effective for complex architectures such as ResNet [6], particularly in generating high resolution images [2, 18, 24].



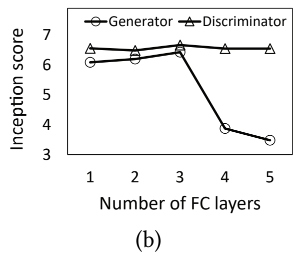


Figure 8: (a) Inception score for varying the number of FC layers in both the generator and discriminator simultaneously (b) Inception scores for varying the number of FC layers in a single network (e.g., generator) keeping the other network (e.g., discriminator) fixed to a CNN model. The results reported are for FCC-GAN model with pooling.

Impact of FC network depth on FCC-GAN architecture: It may be interesting to examine how the performance of the FCC-GAN model changes when the number of layers of the FC network is varied. We conduct two different studies to examine the impact of the number of FC layers in the FCC-GAN architecture. In one, we keep the number of FC layers in both the generator and discriminator same, and vary them from one to five layers. In the other, we vary the number of FC layers in one network, i.e., generator (discriminator), while we use a fixed CNN model for the other network, i.e., discriminator (generator). For both studies, we train the FCC-GAN models on CIFAR-10 dataset using standard GAN training for up to 50 epochs. Figure 8 shows the plot of Inception scores obtained over number of FC layers. As can be seen from Figure 8(a), the best score for CIFAR-10 is obtained when the network has three FC layers in the generator and discriminator. Increasing the number of layers beyond four decreases the scores for FCC-GAN-S drastically, though FCC-GAN-P still achieves moderately good scores. It shows FCC-GAN-P is a more robust architecture than FCC-GAN-S in terms of such variations. For the other study, we find from Figure 8(b) that, the performance of FCC-GAN degrades significantly for four or more FC layers in the generator. However, the performance remains very stable (good) for such variations in the discriminator. Interestingly, for both studies, the best Inception score is obtained at three FC layers.

4.1 Stability of FCC-GAN Architecture

We examined the stability of the proposed FCC-GAN models with respect to a variety of experimental settings. First, we examined the stability with respect to the presence and absence of batch-normalization (BN) layers in the generator and discriminator, following the similar experiments by [1]. Then, we varied the optimization algorithm. We ran experiments on CIFAR-10 dataset for up to 50 epochs to verify stability of the models. Table 10 reports the Inceptions scores and Figure 9 shows some comparisons of generated samples. The results are descried below.

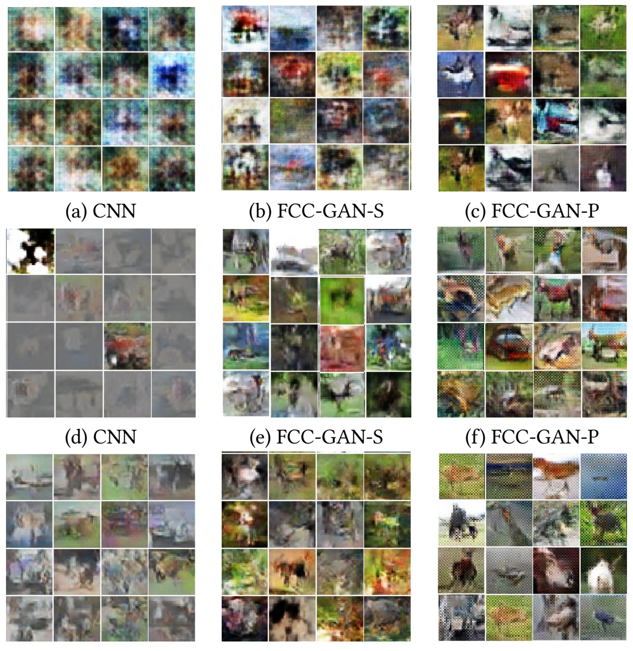


Figure 9: Images generated for (a-c) SGD optimization algorithm, (d-f) No BN in discriminator, and (g-i) No BN in both the generator and discriminator

Impact of BN layers. For this study, we trained the models after sequentially removing BN from the generator (NBN-BN), discriminator (BN-NBN), and from both the generator and discriminator (NBN-NBN).

- No BN in the generator: When we removed BN from the generator only, all models obtained good quality images and inception scores. However, the scores obtained by the FCC-GAN models were better than those of CNN model. Another point to note is that the scores found after removing BN from the generator was worse than that of the

Table 10: Inception scores on CIFAR-10 dataset for stability experiments. Standard deviations are omitted. Very low Inception scores (* marked) indicate the corresponding model completely failed to produce any recognizable images.

Method	CNN	FCC-GAN-S	FCC-GAN-P
RMSProp	5.9512224	5.863833	**6.860175**
SGD	1.6547798*	2.4576838*	**3.8250222**
NBN-BN	5.717898	6.0637116	**6.1960044**
BN-NBN	2.213803*	3.8931298	**4.8963203**
NBN-NBN	3.4395053*	3.522677	**4.0727606**

generator with BN. This implies that the use of BN in the generator has a substantial impact in improving performance of GAN models both in terms image quality and inception scores.

- No BN in the discriminator: Wen we removed BN from the discriminator, the CNN model failed to produce any recognizable outputs (see figure 9(d)). Our FCC-GAN model also failed to produce very good quality images. However, FCC-GAN-P model was still able to produce some recognizable images (see figure 9(f)). The images were worse than those of BN-inclusive discriminator.
- No BN in the generator and discriminator: The results obtained after removing BN from both the generator and discriminator was similar to the results obtained after removing BN from the discriminator only. The CNN model failed to capture the distribution while our FCC-GAN models, specifically the FCC-GAN-P was able to produce some recognizable images (see Figure 9(i)).

In summary, for all cases, FCC-GAN models obtained better image quality and Inception scores (see Table 10). In particular, FCC-GAN-P outperformed others.

Impact of optimization algorithm. We changed the default optimization algorithm from ADAM to SGD and RMSProp with a learning rate of 0.0001 and a decay of 0.00001. Table 10 shows the Inception scores obtained after 50 epochs of the training for the three optimization algorithms. For RMSProp, all three models are stable and obtain good Inception scores. However, our FCC-GAN models achieve better scores than the CNN model, and FCC-GAN-P outperforms the others. For SGD, the CNN model fails to produce any recognizable images after 50 epochs (Figure 9(a)). FCC-GAN-S seems to have captured some properties of the data, and produces comparably better images than CNN (Figure 9(b)). In contrast, FCC-GAN-P generates very good quality images (Figure 9(c)).

Summary of results: We verified the effectiveness of FCC-GAN architecture on four benchmark image datasets. We also verified the applicability of the architecture with respect to two different objective functions commonly used in GAN training algorithms. In all experiments, FCC-GAN architecture generated higher quality images and obtained better Inception scores and FIDs than the conventional CNN architecture used be existing GAN models. In particular, FCC-GAN converged much faster than the CNN architecture. The training stability of the proposed architecture was also verified across a variety of experiment settings, which shows a higher stability of FCC-GAN compared to CNN.

5 CONCLUSION

In this paper, we proposed FCC-GAN architecture for GANs. In our architecture, the generator and discriminator networks consist of deep fully connected and convolution layers, in contrast to conventional deep convolution networks. We performed experiments on four benchmark image datasets and showed that our proposed architecture generates higher quality samples, obtains better Inception scores and Frétchet Inception Distances, and converges faster than the conventional architecture. We also empirically validated its stability in a wide variety of experimental settings.

One key advantage is that the proposed architecture can be applied in combination with any other GAN settings. For example, in this study, all of our experiments were limited to unsupervised training (without using labels). Thus, a straightforward extension of the work is to study the effectiveness of the FCC-GAN architecture in semi-supervised and conditional GAN settings. Another possible extension is to adapt FCC-GAN architecture for more complex networks such as ResNet.

REFERENCES

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein gan. *arXiv preprint arXiv:1701.07875* (2017).

[2] Andrew Brock, Jeff Donahue, and Karen Simonyan. 2018. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096* (2018).

[3] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *NIPS*. 2172–2180.

[4] Emily L Denton, Soumith Chintala, Arthur Szlam, and Rob Fergus. 2015. Deep generative image models using a laplacian pyramid of adversarial networks. In *NIPS*. 1486–1494.

[5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *NIPS*. 2672–2680.

[6] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. 2017. Improved training of wasserstein gans. In *NIPS*. 5767–5777.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*. 770–778.

[8] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 2017. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *NIPS*. 6626–6637.

[9] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*.

[10] Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

[11] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew P Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. 2017. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network.. In *CVPR*, Vol. 2. 4.

[12] Chunyuan Li, Hao Liu, Changyou Chen, Yunchen Pu, Liqun Chen, Ricardo Henao, and Lawrence Carin. 2017. ALICE: Towards Understanding Adversarial Learning for Joint Distribution Matching. In *NIPS*.

[13] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. 2015. Deep Learning Face Attributes in the Wild. In *Proceedings of International Conference on Computer Vision (ICCV)*.

[14] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *ICML*, Vol. 30. 3.

[15] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. 2017. Least squares generative adversarial networks. In *ICCV*. 2813–2821.

[16] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. 2017. Unrolled generative adversarial networks. In *ICLR*.

[17] Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784* (2014).

[18] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. 2018. Spectral normalization for generative adversarial networks. In *ICLR*.

[19] Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *ICML*. 807–814.

[20] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. 2011. Reading digits in natural images with unsupervised feature

learning. In *NIPS (workshop track)*, Vol. 2011. 5.
[21] XuanLong Nguyen, Martin J Wainwright, and Michael I Jordan. 2010. Estimating divergence functionals and the likelihood ratio by convex risk minimization. *IEEE Trans. on Information Theory* 56, 11 (2010), 5847–5861.
[22] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. 2016. f-gan: Training generative neural samplers using variational divergence minimization. In *NIPS*. 271–279.
[23] Augustus Odena. 2016. Semi-supervised learning with generative adversarial networks. *arXiv preprint arXiv:1606.01583* (2016).
[24] Augustus Odena, Christopher Olah, and Jonathon Shlens. 2017. Conditional image synthesis with auxiliary classifier gans. In *ICML*.
[25] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. 2016. Context encoders: Feature learning by inpainting. In *CVPR*. 2536–2544.
[26] Guo-Jun Qi. 2017. Loss-sensitive generative adversarial networks on lipschitz densities. *arXiv preprint arXiv:1701.06264* (2017).
[27] Alec Radford, Luke Metz, and Soumith Chintala. 2016. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*.
[28] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. 2016. Generative adversarial text to image synthesis. In *ICML*.
[29] Scott E Reed, Zeynep Akata, Santosh Mohan, Samuel Tenka, Bernt Schiele, and Honglak Lee. 2016. Learning what and where to draw. In *NIPS*.
[30] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved techniques for training gans. In *NIPS*. 2234–2242.
[31] Jost Tobias Springenberg. 2016. Unsupervised and semi-supervised learning with categorical generative adversarial networks. In *ICLR*.
[32] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. 2015. Striving for simplicity: The all convolutional net. In *ICLR (Workshop track)*.
[33] Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. 2016. Conditional image generation with pixelcnn decoders. In *NIPS*. 4790–4798.
[34] Xiang Wei, Boqing Gong, Zixia Liu, Wei Lu, and Liqiang Wang. 2018. Improving the Improved Training of Wasserstein GANs: A Consistency Term and Its Dual Effect. In *ICLR*.
[35] Junbo Zhao, Michael Mathieu, and Yann LeCun. 2017. Energy-based generative adversarial network. In *ICLR*.

A DETAILS OF NETWORK ARCHITECTURE AND EXPERIMENTAL SETTINGS

A.1 Model architecture used for CIFAR-10 and SVHN

Table 11 shows the FCC-GAN model architecture for CIFAR-10 and SVHN dataset. The images have the dimension 32×32×3. In the generator, we used three FC layers that map the 100-dimensional noise to 4096-dimensional intermediate features. The linear features are then reshaped to a spatial extent of dimension 4×4×256. Then four transposed convolution layers map these features to output images of dimension 32×32×3. ReLU activation is used for all layers except the output which uses Tanh activation. Batch-normalization (BN) is used after the last FC layer and all the convolution layers except the final output. In the discriminator, three convolution layers extract image features of dimension $4 \times 4 \times 256$. The features are then flattened to a 4096-dimensional feature vector. These features are then mapped to a low-dimensional space by four FC layers before the final output. LeakyReLU (LReLU) with a slope of 0.2 is used as the activation function for all layers except the final output. Sigmoid is used as the output activation for standard GAN [5] training while no activation is used for Wasserstein GAN [1]. For FCC-GAN-P model, we use average pooling instead of strided-convolution for downsampling. This implies we replace each CONV(x,y,2) layer in the discriminator with an equivalent CONV(x,y,1) layer followed by an average pooling with a pool size of 2×2 for downsampling. Table 12 shows the conventional CNN model used for CIFAR-10 and SVHN. Note that the first (last) convolution layer in the generator (discriminator) of CNN models can be considered a fully connected layer.

Table 11: FCC-GAN architecture used for CIFAR-10 and SVHN

Generator	Discriminator
Input: Z(100)	Input: Image (32,32,3)
FC(64), ReLU	CONV(64,4,2), BN, LReLU
FC(512), ReLU	CONV(128,4,2), BN, LReLU
FC(4096), BN	CONV(256,4,2), BN, LReLU
Reshape (4,4,256)	Flatten (4096)
CONVT(128,4,2), BN, ReLU	FC(512), LReLU
CONVT(64,4,2), BN, ReLU	FC(64), LReLU
CONVT(3,4,3), Tanh	FC(16), LReLU
Output: (32, 32,3)	FC(1), Sigmoid
	Output: 1

A.2 Model architecture used for MNIST

MNIST image size is $28 \times 28 \times 1$. Thus, the convolution layers used for MNIST have different kernel and filter sizes than those used for CIFAR-10 and SVHN. Table 13 shows the FCC-GAN architecture while Table 14 shows the CNN architecture.

A.3 Model architecture used for CelebA

Table 15 and 16 show the network architectures used for 64×64 pixel CelebA dataset. Note that the network configuration is similar

Table 12: CNN architecture used for CIFAR-10 and SVHN

Generator	Discriminator
Input: Z(100)	Input: (32,32,3)
R(1,1,100)	CONV(64,4,2), BN, LReLU
CONVT(256,4,1), BN, ReLU	CONV(128,4,2), BN, LReLU
CONVT(128,4,2), BN, ReLU	CONV(256,4,2), BN, LReLU
CONVT(64,4,2), BN, ReLU	CONV(1,4,1), Sigmoid
CONVT(3,4,3), Tanh	Output: 1
Output: (32, 32, 3)	

Table 13: FCC-GAN architecture used for MNIST

Generator	Discriminator
Input: Z(100)	Input: (28,28,1)
FC(64), ReLU	CONV(32,3,2), BN, LReLU
FC(512), ReLU	CONV(64,3,2), BN, LReLU
FC(1152), BN	CONV(128,3,2), BN, LReLU
Reshape (3,3,128)	Flatten (1152)
CONVT(64,3,2), BN, ReLU	FC(512), LReLU
CONVT(32,3,2), BN, ReLU	FC(64), LReLU
CONVT(1,3,2), Tanh	FC(16), LReLU
Output: (28, 28, 1)	FC(1), Sigmoid
	Output: 1

Table 14: CNN architecture used for MNIST

Generator	Discriminator
Input: Z(100)	Input: (28,28,1)
R(1,1,100)	CONV(32,3,2), BN, LReLU
CONVT(128,3,1), BN, ReLU	CONV(64,3,2), BN, LReLU
CONVT(64,3,2), BN, ReLU	CONV(128,3,2), BN, LReLU
CONVT(32,3,2), BN, ReLU	CONV(1,3,1), Sigmoid
CONVT(1,3,2), Tanh	Output: 1

to those of CIFAR10 dataset except an additional convolution layer in both the generator and discriminator.

Table 15: FCC-GAN architecture used for CelebA

Generator	Discriminator
Input: Z(100)	Input: Image (64,64,3)
FC(64), ReLU	CONV(64,4,2), BN, LReLU
FC(512), ReLU	CONV(128,4,2), BN, LReLU
FC(8192), BN	CONV(256,4,2), BN, LReLU
Reshape (4,4,512), BN	CONV(512,4,2), BN, LReLU
CONVT(256,4,2)	Flatten (8192)
CONVT(128,4,2), BN, ReLU	FC(512), LReLU
CONVT(64,4,2), BN, ReLU	FC(64), LReLU
CONVT(3,4,3), Tanh	FC(16), LReLU
Output: (32, 32,3)	FC(1), Sigmoid
	Output: 1

Table 16: CNN architecture used for CelebA

Generator	Discriminator
Input: Z(100)	Input: (64,64,3)
R(1,1,100)	CONV(64,4,2), BN, LReLU
CONVT(512,4,1), BN, ReLU	CONV(128,4,2), BN, LReLU
CONVT(256,4,1), BN, ReLU	CONV(256,4,2), BN, LReLU
CONVT(128,4,2), BN, ReLU	CONV(512,4,2), BN, LReLU
CONVT(64,4,2), BN, ReLU	CONV(1,4,1), Sigmoid
CONVT(3,4,3), Tanh	Output: 1
Output: (32, 32, 3)	

A.4 Model architecture for experiments on convolution network depth

For evaluating the impact of convolution network depth on CNN and FCC-GAN architecture, we increased the number of convolution layers in the generator and discriminator of CNN models from 4 to 7. The CNN model architecture is shown in Table 17. The FCC-GAN models for this experiment were obtained by replacing the first (last) convolution layer in the generator (discriminator) with three (four) FC layers. Table 18 shows the FCC-GAN architecture with 7 convolution layers.

Table 17: Generator and discriminator network for CNN models with 7 convolution layers

Generator	Discriminator
Input: Z(100)	Input: (32,32,3)
R(1,1,100)	CONV(64,3,1), BN, LReLU
CONVT(256,4,1), BN, ReLU	CONV(64,4,2), BN, LReLU
CONV(256,3,1), BN, ReLU	CONV(128,3,1), BN, LReLU
CONVT(128,4,2), BN, ReLU	CONV(256,4,2), BN, LReLU
CONV(128,3,1), BN, ReLU	CONV(256,3,1), BN, LReLU
CONVT(64,4,2), BN, ReLU	CONV(256,4,2), BN, LReLU
CONV(64,3,1), BN, ReLU	CONV(1,4,1), Sigmoid
CONVT(3,4,3), Tanh	Output: 1
Output: (32, 32, 3)	

A.5 Experimental settings

In standard GAN training, we used the classic GAN objective function [5] for optimization. All models were trained with a batch size of 32. Adam [10] optimizer was used with a learning rate of 0.0001 and a decay of 0.00001. For WGAN training, the Wasserstein distance was used as the objective function. All hyper-parameters were kept similar to those described in [1]: RMSProp optimizer with a learning rate of 0.00005 and a batch size of 64.

Table 18: Generator and discriminator network for FCC-GAN models with 7 convolution layers

Generator	Discriminator
Input: Z(100)	Input: (32,32,3)
FC(64), ReLU	CONV(64,3,1), BN, LReLU
FC(512), ReLU	CONV(64,4,2), BN, LReLU
FC(8192), BN	CONV(128,3,1), BN, LReLU
R(4,4,256)	CONV(128,4,2), BN, LReLU
CONV(256,3,1), BN, ReLU	CONV(256,3,1), BN, LReLU
CONVT(128,4,2), BN, ReLU	CONV(256,4,2), BN, LReLU
CONV(128,3,1), BN, ReLU	Flatten(8192)
CONVT(64,4,2), BN, ReLU	FC(512), LReLU
CONV(64,3,1), BN, ReLU	FC(64), LReLU
CONVT(3,4,3), Tanh	FC(16), LReLU
Output: (32, 32, 3)	FC(1), Sigmoid
	Output: 1