1A3-2

# Depth and Complexity of Deep Generative Adversarial Networks

Hiroyuki V. Yamazaki, Takahira Yamaguchi

Keio University

Although generative adversarial networks (GANs) have achieved state-of-the-art results in generating realistic looking images, models often consist of neural networks with few layers compared to those for classification. We evaluate different architectures for GANs with varying depths using residual blocks with shortcut connections in order to train GANs with higher capacity. While training tend to oscillate and not benefit from additional capacity of naively stacked layers, we show that GANs are capable of generating images of higher visual fidelity with proper regularization and simple techniques such as minibatch discrimination. In particular, we show that an architecture similar to the standard GAN with residual blocks in the hidden layers consistently achieve higher inception scores than the standard model without noticeable mode collapse. The source code is made available on `https://github.com/hvy/gan-complexity`.

## 1. Introduction

Generative adversarial networks (GANs) are generative models that are trained to estimate data distributions using two functions, a data generating function and an adversarial function called the generator and the discriminator [5]. They have in particular been successful in modeling distributions of real images yielding sharper results than variational autoencoders [9].

These two functions, the generator and the discriminator are often trained as neural networks. The generator and the discriminator are jointly trained with the objective function defined by the minimax game $\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_x(x)}[log(D(x))] + \mathbb{E}_{z\sim p_z(z)}[log(1-D(G(z)))]$ where $D$ denotes the probability that the discriminator assigns to its given data, $G$ the data sampled from the generator and $z$ random input noise. The objective of the generator is to map the noise $z$ from an easily sampled distribution $p_z$ such as a Gaussian or a uniform distribution to the data generating distribution. The discriminator is an adversarial classifier, trained to discriminate between samples from the true distribution and samples from the generator. The parameters are updated using gradient descent and GANs converge when those functions reach an equilibrium in this minimax game, a saddle point in the value function.

This is known to be difficult and the training is not guaranteed to converge. It could be the case that the discriminator pushes the generator around in the data space, never allowing the generator to spread its mass to cover the real distribution. Another failure case is having the generator collapsing to a single or few modes, basically mapping all possible inputs to nearby points that the discriminator thinks is real.

Because of the training instability, various architectures and techniques have been proposed to alleviate these problems [1, 13, 18]. However, they are often small in terms of number of parameters compared to neural networks used for other tasks such as classification.

Contact: Hiroyuki V. Yamazaki, Keio University, hvy@keio.jp

We argue that this is not optimal in the setting of modeling real image distributions and that GANs are capable of generating images of higher visual fidelity given more capacity. In this work, we attempt to train larger models inspired by the shortcut connections in residual networks in order to generate images of higher visual fidelity.

The main contributions of this work are as follows.

- Evaluation in terms of inception scores for GANs with different depths.
- A simple GAN architecture with shortcut connections that consistently achieves higher inception scores compared to a standard model without noticeable mode collapse, using minibatch discrimination.

## 2. Related Work

DCGAN was proposed as an extension to GANs with an emphasis on convolutional layers to generate images. They used deconvolutions in the generator to upsample noise and successfully trained a generative model that could generate sharp realistic looking images of bedrooms and human faces [15]. In this work, we will refer the standard GAN to this architecture.

Other techniques involve training an encoder that maps data from the data generating distribution to a low dimensional latent space, which can then be used as the noise to reconstruct the original data as in [3, 4, 10] while [2, 7, 17] rely on intermediate supervision by splitting up the generator into several smaller networks, each one with a less difficult task than the standard GAN to improve the training.

### 2.1 Shortcut Connections

Shortcut connections, or skip connections are used in the residual blocks of residual networks allowing neural networks of hundreds or over a thousands layers to train and improve in terms of classification accuracy over plain feed forward networks [6]. Residual networks group few convolutional layers, often two or three, into so called residual blocks. A residual block uses a shortcut connections to directly pass the input of a block to its output, adding them

together and reparameterizing the block to learn the residual.

Later works have shown that shortcut connections break symmetry in the loss landscape, reducing the number of local minima and improving training [14].

With the success of deep residual architectures in training for non-adversarial settings, we study the effects of those architectures when put in the settings of GANs. More specifically, we compare different models with varying number of residual blocks by plotting the inception scores during the course of training.

### 2.2 Minibatch Discrimination

Since early results show that naively stacking residual blocks increase the amount of oscillation and mode collapse as shown in Section 4.1, we also train the same networks with a minibatch discrimination layer right before the fully connected layer in the discriminator to alleviate the mode collapse problem [16]. The minibatch discrimination layer allows the discriminator to look at whole minibatches of images before assigning probabilities, which allows the generator to mimic the statistics of each minibatch from the training dataset.

### 2.3 Inception Score

The training performance is measured using the inception score [16] which has been used in [7, 16–18] to evaluate trained models. The inception score is defined as $\exp(\mathbb{E}_x[D_{KL}(p(y|x) \;||\; p(y))])$ [16]. The exponential of the expected Kullback-Leibler divergence from $p(y)$, the marginal distribution over all predictions in a set of samples to $p(y|x)$, the conditional distribution over classes given by a pre-trained Inception network. Intuitively, this means that a classifier should make highly confident predictions, assign low entropy to $p(y|x)$, but with variations, high entropy to $p(y)$, for well optimized generative models that yield realistic images. Higher is better.

## 3. Models

All models are based on the standard GAN architecture with four deconvolution and four convolution layers with no pooling layers, but where the last convolution in the discriminator is replaced with a fully connected layer with a single output, namely the probability $D(\cdot)$. The generator uses batch normalization followed by the ReLU activation [12] in all layers except for the last where there is no batch normalization and the final output is given by a tanh activation. The batch normalization is removed from the last layer since we do not want to add noise to the data after the final trainable layer. Similarly, the discriminator uses batch normalization in all layers except for the first, with leaky ReLU [11] activations after each layer to allow gradients to propagate backwards to the generator. The final probability is given by the sigmoid function ensuring that the output is in range $[0, 1]$.

To train deeper GANs with higher capacity, shortcut connections are added to the two middle-most deconvolutional and convolutional layers with N$\in \{1, 2, 3, 5\}$ blocks as shown in Table 1. Higher N means higher capacity. Residual blocks are avoided in the first and the last layers, similar to [6]. Each block consist of two deconvolutional layers and two convolutional layers in the generator and the discriminator respectively. In order to keep the number of parameters in the same order for both of the networks, architectures are always mirrored (same N) during the experiments. The architecture of the generator with N = 1 is shown in Figure 1.

The models are updated using the binary cross entropy loss with the Adam update rule [8] with a learning rate $\alpha = 0.0002, \beta_1 = 0.5, \beta_2 = 0.999$ similar to the settings in [15]. All models are trained with CIFAR-10 with a minibatch size of 64 over 300 epochs. We use weight decay with a decay rate of 0.00001. Both the generator and the discriminator are trained using the same settings.

| Generator, Input $z \in \mathcal{R}^{100}$ | | |
|---|---|---|
| Stage | Output | Block |
| dc1 | $4 \times 4$ | $\begin{bmatrix} 4 \times 4, 256 \end{bmatrix}$ |
| dc2 | $8 \times 8$ | $\begin{bmatrix} 4 \times 4, 128 \\ 3 \times 3, 128 \end{bmatrix}, \begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times (\text{N}-1)$ |
| dc3 | $16 \times 16$ | $\begin{bmatrix} 4 \times 4, 64 \\ 3 \times 3, 64 \end{bmatrix}, \begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times (\text{N}-1)$ |
| dc4 | $32 \times 32$ | $\begin{bmatrix} 4 \times 4, 3 \end{bmatrix}$ |
| Discriminator, Input $x \in \mathcal{R}^{32 \times 32}$ | | |
| Stage | Output | Block |
| c1 | $16 \times 16$ | $\begin{bmatrix} 4 \times 4, 64 \end{bmatrix}$ |
| c2 | $8 \times 8$ | $\begin{bmatrix} 4 \times 4, 128 \\ 3 \times 3, 128 \end{bmatrix}, \begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times (\text{N}-1)$ |
| c3 | $4 \times 4$ | $\begin{bmatrix} 4 \times 4, 256 \\ 3 \times 3, 256 \end{bmatrix}, \begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times (\text{N}-1)$ |
| fc | $1 \times 1$ | - |

Table 1: Architectures for the generator and the discriminator. N corresponds to the number of residual blocks in a stage. In our experiments, we use N$\in \{1, 2, 3, 5\}$. dc stands for deconvolutional blocks, c for convolutional blocks and fc for the fully connected layer with a single output.

## 4. Results

The inception scores are plotted over the course of the training for all architectures in order to compare different models. While the inception scores usually require a large number of samples, we reduce that number down to 5000 from 50000 [16] in favor of computation. We can still clearly see the trend in how the training progresses and are able to compare the different architectures.

### 4.1 Stacking Residual Blocks

As shown in Figure 2, naively increasing the depth using residual blocks tend to degrade the results causing oscillation and mode collapse. In other words, a batch of images

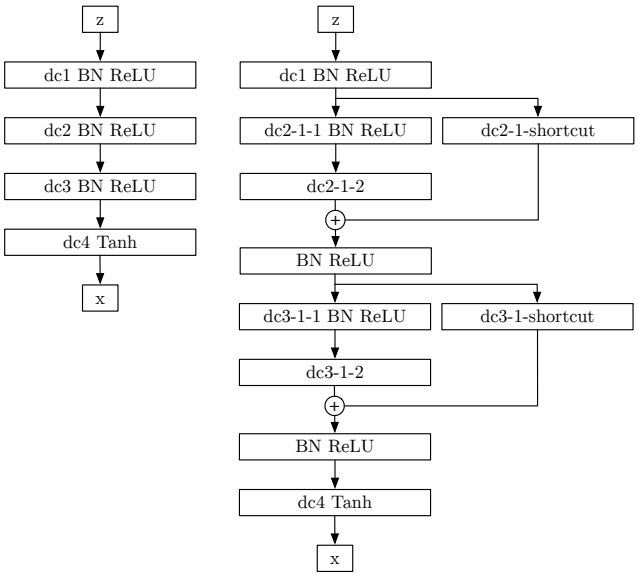



Figure 1: Left: Generator of the standard GAN. Right: Generator of the proposed model with residual blocks, N = 1. dc stands for deconvolutional blocks, and BN for batch normalization. The + signs represent element-wise additions and the xs are the generated images.

being sampled from the generator at any given time has low diversity while at the same time looking completely different when sampled at a different time. This is most likely caused by the generator being able to move its mass towards a tiny region of the manifold where the discriminator assign high probability for almost any given input noise z as the capacity of the network increases. By visually inspecting the images, we observe that mode collapse often is correlated with oscillation, meaning that if either of these occur, that the other is likely to occur as well.

The key observation is that the model with N = 1 as shown in Figure 1 (Right), achieves roughly the same inception score as the standard GAN, Figure 1 (Left), even though its depth is increased by two layers in each of the two networks although it starts degrading towards the end. This is especially surprising given that N = 2 performs considerably worse.

With minibatch discrimination, this model even surpasses the standard GAN consistently where it successfully benefit from the additional capacity. Images sampled from this model are shown in Figure 3.

An unexpected behavior is the standard GAN suffering from minibatch discrimination. If the minibatch discrimination acts as a regularizer, this could mean that the regularization was too strong, allowing the generator to spread its mass but never converge to generate images of high visual fidelity.

We also observed that images that look like noise with low fidelity early in the training have a hard time moving its mass to regions of the true data distribution. Conversely, images that look somewhat realistic usually converge to realistic looking images. This is caused by the discriminator rejecting bad images with high confidence leaving no gradients for the generator to improve.

## 5. Conclusions

Due to the convergence difficulty of training GANs, most architectures today consist of few parameters compared to neural networks for tasks such as classification. We observe that the oscillation becomes more prominent as the number of layers increase, meaning that simply adding complexity to the networks in terms of depth makes the training worse. However, we also empirically show that GANs are capable of generating images with higher visual fidelity given certain amounts of additional complexity using proper regularization techniques and minibatch discrimination. This shows that depth plays a crucial role in generative settings.

## References

[1] Tong Che, Yanran Li, Athul Paul Jacob, Yoshua Bengio, and Wenjie Li. Mode regularized generative adversarial networks. *CoRR*, abs/1612.02136, 2016.

[2] Emily L. Denton, Soumith Chintala, Arthur Szlam, and Robert Fergus. Deep generative image models using a laplacian pyramid of adversarial networks. *CoRR*, abs/1506.05751, 2015.

[3] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *CoRR*, abs/1605.09782, 2016.

[4] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Alex Lamb, Martín Arjovsky, Olivier Mastropietro, and Aaron C. Courville. Adversarially learned inference. *CoRR*, abs/1606.00704, 2016.

[5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

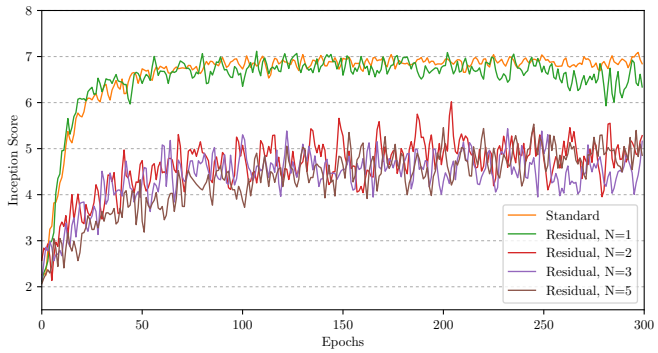
[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[7] Xun Huang, Yixuan Li, Omid Poursaeed, John E. Hopcroft, and Serge J. Belongie. Stacked generative adversarial networks. *CoRR*, abs/1612.04357, 2016.

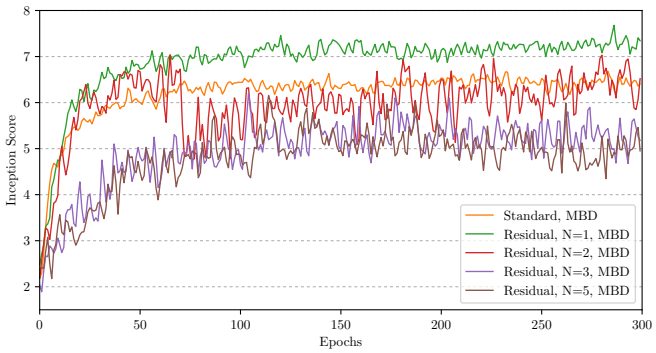
[8] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[9] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013.

[10] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, and Ole Winther. Autoencoding beyond pixels using




(a) Without minibatch discrimination.




(b) With minibatch discrimination.

Figure 2: Inception scores for models with different depths; the standard GAN and the residual GANs with N$\in \{1, 2, 3, 4, 5\}$. Inception scores are computed after the last iteration in each epoch.

Figure 3: Randomly sampled images generated from the best performing model with N = 1 with minibatch discrimination.

a learned similarity metric. *CoRR*, abs/1512.09300, 2015.

[11] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of the 30 th International Conference on Machine Learning*, 2013.

[12] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In Johannes Frnkranz and Thorsten Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814. Omnipress, 2010.

[13] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. *CoRR*, abs/1610.09585, 2016.

[14] A. Emin Orhan. Skip connections as effective symmetry-breaking. *CoRR*, abs/1701.09175, 2017.

[15] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2015.

[16] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *CoRR*, abs/1606.03498, 2016.

[17] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaolei Huang, Xiaogang Wang, and Dimitris N. Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. *CoRR*, abs/1612.03242, 2016.

[18] Junbo Jake Zhao, Michaël Mathieu, and Yann LeCun. Energy-based generative adversarial network. *CoRR*, abs/1609.03126, 2016.