EE390A 2019 Tutorial Paper on Generative Adversarial Networks (GANs)

Debashish Reang (170228)
Department of Electrical Engineering
Indian Institute of Technology Kanpur
Kanpur, Uttar Pradesh 208016
E-mail: reang@iitk.ac.in

***Abstract*—Since its invention by Goodfellow et al. (2014) [1], Generative Adversarial Nets more popularly known as GANs have come out to be one of the very famous methods in Deep Learning. GANs have been used in semi-supervised learning [2], fully supervised learning [3] and reinforcement learning [4] settings. Although introduced initially as a novel method for training generative models, GANs have been successfully used to super-resolve images, predict next frame in a video, image style conversion [3] etc. This article describes the math and intuition behind GANs, motivates the study of generative models and concludes with a few example use cases. Anyone with a basic background in probability theory should be able to understand the material very easily. No prior expertise in Deep Learning is assumed.**

I. Introduction

Let's say we have some high dimensional space $\mathcal{X}$ and let X be some datapoints in that space. We can define a distribution $P(X)$ that models the datapoints X. "Generative modeling" deals with such distributions $P(X)$. We will use images as our datapoint to motivate the discussion going forward. There is nothing special about images, any kind of data can be modeled using generative models. It is however, convenient to use images for illustrative purposes. A "datapoint" (an image) can be made of hundreds or thousands of pixels and we wish that our generative model be able to relate these pixels e.g., pixels that have similar density may represent a blob and hence should be grouped together. One model can be that we assign a high probability to X's that look like real images and assign a low probability to the ones that look like random noise. This seems like a reasonable thing to do. However, knowing that one image has low probability does not help us in producing a new image that has a high probability.

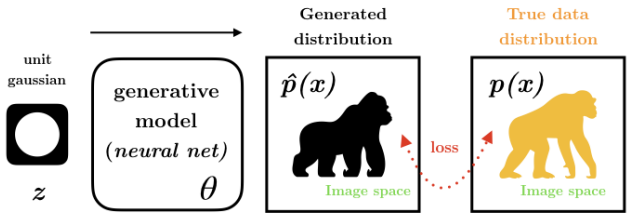

Fig. 1. Intuitive depiction of Generative Modeling

Instead of the above approach, we can train a model that generate images similar to the ones that is already there in the database. We don't care if it is not exactly the same, only that the generated image should be similar to the rest. For instance, we can try to produce next scene in video games based on what we have seen so far or produce more handwritten text after seeing many handwritten texts. Formally, say we have X from some unknown distribution $P_{un}(X)$ and we want to learn a model M such that M and $P_{un}(X)$ are as similar as possible. We give an intuitive depiction of generative modeling in Fig. 1.

The machine learning community had been interested in training the these kind of models. Many approaches had been tried out in this regard. A taxonomy of models had been shown in Fig. 2. Most of them had one of the three serious downsides. One, the model makes very strong assumptions. This is especially a problem if the model inherently assumes a particular distribution of the data. Two, the models may make approximations about values obtained during training. This can possibly end up in suboptimal models because approximations were made. Three, the models may end up using Markov Chain Monte Carlo methods which are computationally very expensive. A lot of progress has been made in training neural nets through backpropagation. The recent work by Krizhevsky et al. [5] is an example. These motivated the deep learning community in building generative models based on backpropagation methods.

One of such approaches is the Generative Adversarial Nets approach, which is the subject of this report. This approach has advantages over other approaches in that samples can be generated in parallel, it does not require Markov Chains and is not bounded by any Variational Bounds. [6]

This report is not intended to be a scientific paper but rather an informal introduction to Generative Adversarial Nets (GANs).

II. Generative Adversarial Nets

This framework has two neural nets competing against each other — the generator and the discriminator. The generator is the generative model that we previously described an the discriminator is an adversary whose purpose is is tell us whether the generated data came from the model data or the training data. To give an intuitive understanding let us consider an examination setting. The student who is cheating in the test can be thought of as the generator and the instructor as the discriminator. The job and the student is to convince the instructor that he did not

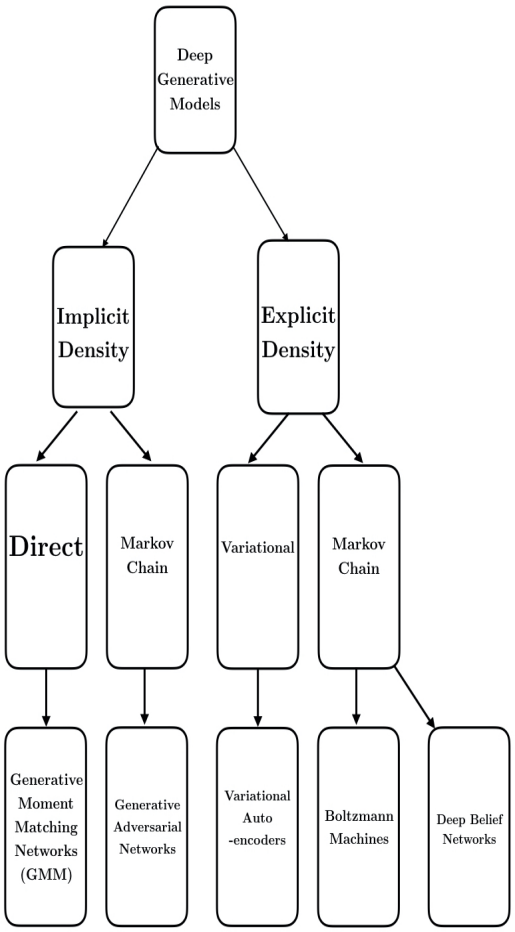

Fig. 2. Taxonomy of generative models

copy in the test and the instructor's job to catch the student if he had copied. The two compete against each other until the student convice the instructor that he had not copied while in reality he had in fact copied in the test.

Let us now formalize the above intuition. Define a multilayer perceptron $G(\boldsymbol{z};\theta_g)$, a differentiable function parameterized by θ_g. We also define a prior on the input noise variables. Let us call this $p_{\boldsymbol{z}(\boldsymbol{z})}$. We can then learn the generator's distribution p_g over the data $\boldsymbol{x}$. Now define a second multilayer perceptron $D(\boldsymbol{x};\theta_d)$. The output of D is a single scalar. The probability that $\boldsymbol{x}$ came from the data rather than p_g is represented by $D(\boldsymbol{x})$. D and G play the following game with the cost function $C(G,D)$:

$$\min_G \max_D C(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}(\boldsymbol{x})[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}(\boldsymbol{z})[\log(1-D(G(\boldsymbol{z})))]. \quad (1)$$

The Generative Adversarial Nets framework is summed up in Fig. 3.

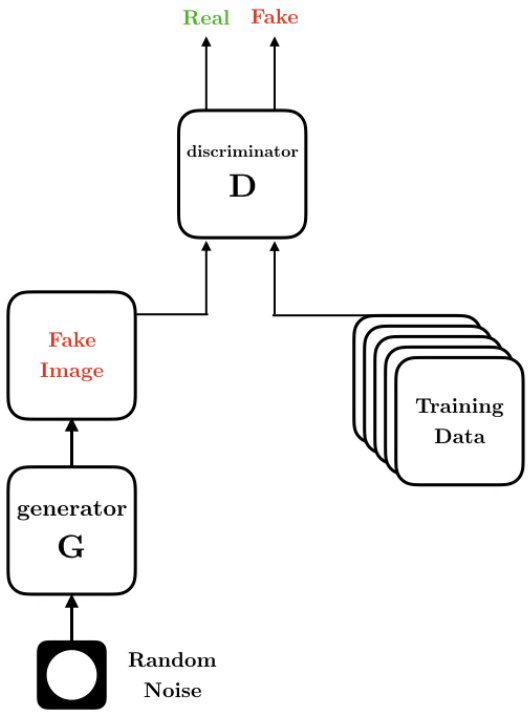

Fig. 3. Generative Adversarial Nets framework

All of the theoretical results are summed up in the next section.

III. Theoretical Results

Algorithm 1 Training of generative adversarial nets. Here, mini-batch stochastic gradient descent is used. n is the number of steps to apply to the discriminator net. n is a hyperparameter.

i = number of iterations
for i **do**
  for n steps **do**
    •Get k noise samples $\{\boldsymbol{z}^{(1)},\ldots,\boldsymbol{z}^{(k)}\}$ from $p_g(\boldsymbol{z})$.
    •Get k examples $\{\boldsymbol{x}^{(1)},\ldots,\boldsymbol{x}^{(k)}\}$ from $p_{\text{data}}(\boldsymbol{x})$.
    •The discriminator is updated using the gradient:

$$\nabla_{\theta_d}\frac{1}{k}\sum_{i=1}^{k}\left[\log D\left(\boldsymbol{x}^{(i)}\right)+\log\left(1-D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right].$$

  end for
  •Get k noise samples $\{\boldsymbol{z}^{(1)},\ldots,\boldsymbol{z}^{(k)}\}$ from $p_g(\boldsymbol{z})$.
  •The generator is updated using:

$$\nabla_{\theta_g}\frac{1}{k}\sum_{i=1}^{k}\log\left(1-D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

A. $p_g = p_{\text{data}}$ *is globally optimal*

Proposition 1. *Call the optimal discriminator as D_G^* and keep the generator G fixed. Then you get the discriminator as given by this:*

$$D_G{}^*(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})} \tag{2}$$

Proof. For whatever generator G is given, the discriminator D should maximize the variable $C(G, D)$:

$$\begin{aligned} C(G,D) &= \int_{\boldsymbol{x}} p_{\text{data}} \log(D(\boldsymbol{x}))dx + \int_{\boldsymbol{z}} p_{\boldsymbol{z}}(\boldsymbol{z}) \log(1 - D(g(\boldsymbol{z})))dz \\ &= \int_{\boldsymbol{x}} \left[p_{\text{data}} \log(D(\boldsymbol{x})) + \log(1 - D(\boldsymbol{x}))\right] dx \end{aligned} \tag{3}$$

Outside $Supp(p_{\text{data}}) \cup Supp(p_g)$ we do not require the discriminator to be defined. Also, the function $w \to p\log(w) + q\log(1-w)$ becomes highest in $[0, 1]$ at $\frac{p}{p+q}$ for any $(p, q) \in \mathbb{R}^2 \setminus \{0, 0\}$. **Hence, proved.** [1]

Observe that

$$\begin{aligned} C(G) &= \max_D C(G, D) \\ &= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}}[\log(1 - D_G^*(G(\boldsymbol{z})))] \\ &= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x} \sim p_g}[\log(1 - D_G^*(\boldsymbol{x}))] \\ &= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}[\log \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})}] \\ &+ \mathbb{E}_{\boldsymbol{x} \sim p_g}[\log \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})}] \end{aligned} \tag{4}$$

Theorem 1. *When $p_g = p_{\text{data}}$ the global minimum criterion is achieved and $C(G)$ get an optimal value equal to $-\log 4$.*

Proof. Look at eq. 2 When we put $p_g = p_{\text{data}}$ in that equation, we get the value of $D_G^*(\boldsymbol{x})$ to be half, in other words $D_G^*(\boldsymbol{x}) = \frac{1}{2}$.

Now look at eq. 4. When, the value of $D_G^*(\boldsymbol{x}) = \frac{1}{2}$, we have putting the value of $D_G^*(\boldsymbol{x}) = \frac{1}{2}$ in eq. 4, $C(G) = -\log 2 + -\log 2 = -\log 4$. This is the best possible value of $C(G)$. And this happens iff $p_g = p_{\text{data}}$ because

$$\mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} \left[\log \frac{1}{2}\right] + \mathbb{E}_{\boldsymbol{x} \sim p_g} \left[\log \frac{1}{2}\right] = -\log 4$$

Now subtract this equation from $C(G) = C(D_G^*)$. We get the value of $C(G)$ equal to and that by subtracting this expression from $C(G) = V(D_G^*, G)$, we obtain the value of $C(G)$ to be:

$$\log \frac{1}{4} + KL\left(p_{\text{data}} \parallel \frac{p_{\text{data}} + p_g}{2}\right) + KL\left(p_{\text{data}} \parallel \frac{p_g + p_g}{2}\right) \tag{5}$$

This shows that whenever $p_g = p_{\text{data}}$ happens $C_G^* = \log \frac{1}{4}$ and the data distribution is absolutely identical to the one that is generated by the generative model. Of course, in practice this shall never happen.

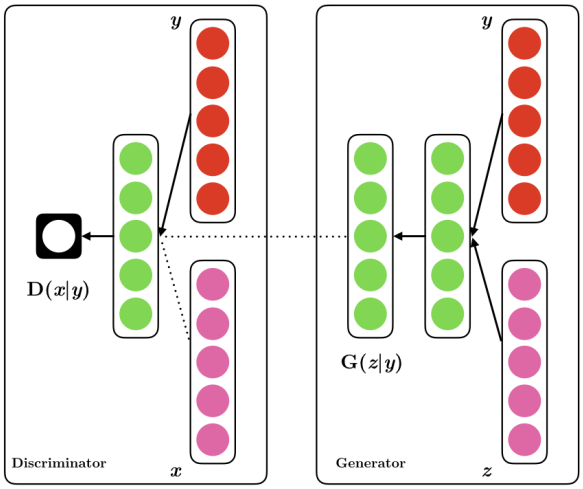

Fig. 4. Summary of the conditional generative adversarial nets framework

IV. Conditional GANs

At this point we point out that the generative adversarial nets that we described so far does not take into account any extra information. They just take in some random noise and raw input data for generating new samples. It would be good if the model also took as input extra input like the name of the class names or the number of pixels that an image is composed of. That is exactly what Mirza at al. [7] did. They devised a conditional model on top of the existing GANs framework. Let us call the extra information as $\boldsymbol{y}$. $\boldsymbol{y}$ is given as input to both D and G as a condition. This approach is popularly known as conditioning in the Deep Learning community.

$\boldsymbol{y}$ and $p_{\boldsymbol{z}(\boldsymbol{z})}$ get combined and are fed into the generator as input. The input is a joint hidden representation. The framework allows one to manipulate how the input is given to the networks. Experimental results shows that including extra condition while training helped improve the performance of the framework as compared to the bare minimum framework without any extra constraints.

The modified objective function is:

$$\begin{aligned} \min_G \max_D C(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}(\boldsymbol{x})[\log D(\boldsymbol{x}|\boldsymbol{y})] + \\ \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}}(\boldsymbol{z})[\log(1 - D(G(\boldsymbol{z}|\boldsymbol{y})))]. \end{aligned} \tag{6}$$

Figure 4 sums up our arguments.

V. Examples

A. MNIST Deep Convolutional Generative Adversarial Network (DCGAN)

We use MNIST to show the capabilities of the generative adversarial nets framework. Specifically, we use a particular flavor of GANs known as the DCGAN in short for Deep Convolutional Generative Adversarial Network. It was first proposed by Radford et al. [8] The network architecture is shown in Fig. 5. The network

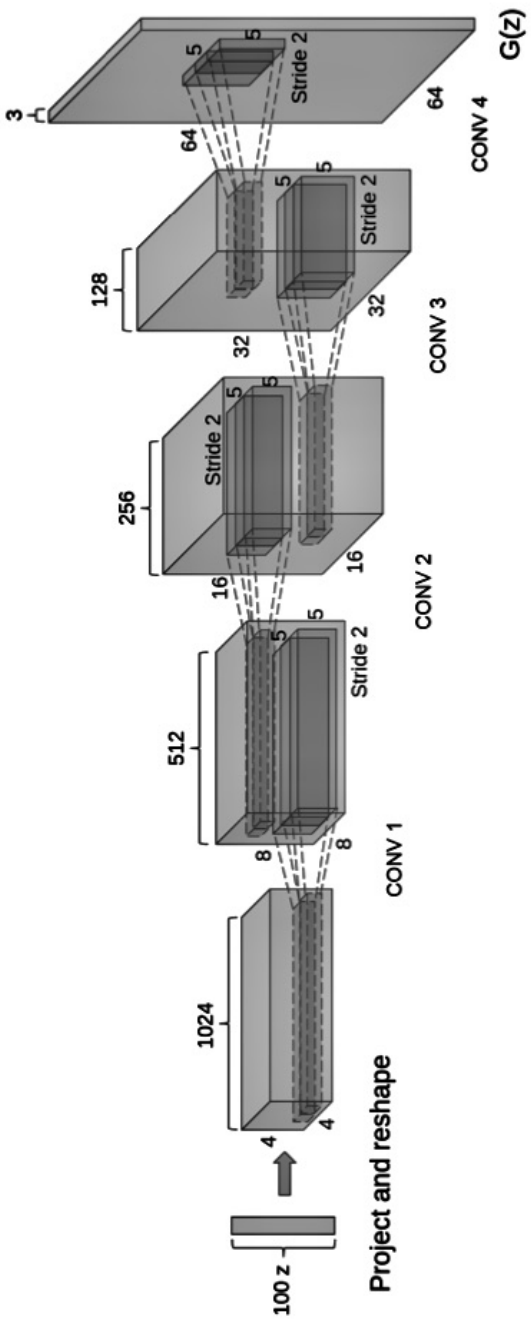

Fig. 5. Deep Convolutional Generative Adversarial Nets (DCGANs)

first takes in random noise as input and generate some random noisy image which the discriminator is able to differentiate initially. Over several training steps the network generates better and better images. after about 20 training steps the MNIST data generated looks almost as real as the training data.

DCGAN takes as input random noise sampled from a distribution (usually uniform distribution) and then projects and reshapes the samples into pixels. At the beginning of the training process, these are just random values that have no relationship with each other and do not make meaningful representations. However, as the training progresses, these random numbers undergo convolutions and starts to make sense. In case of MNIST we get 32 by 32 pixels at the output layer each of these pixels are related to each other and over several epochs of training they start resembling the data that is there in the MNIST dataset. The upsampling and convolution is however not done in one step. They are spread over several layers. This allows for the flexibility to learn one feature at a time. DCGAN makes use of both the effectiveness of CNNs as well as GANs. The results generated by the network is shown in Fig. 6.

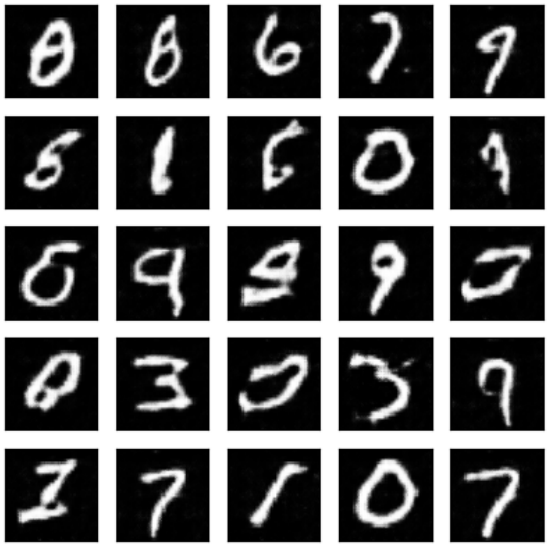

Fig. 6. MNIST data generated by DCGAN after 20 epochs

VI. MNIST Conditional DCGAN

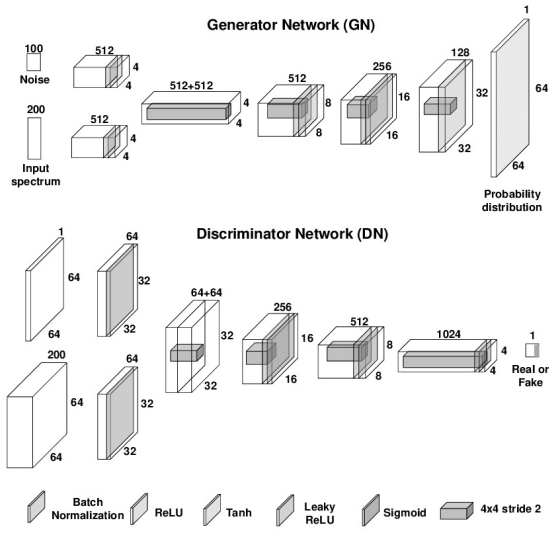

Fig. 7. Conditional DCGAN architecture (CDCGAN)

The Conditional DCGAN or CDCGAN as it is popularly know in the literature is an extension of the Deep Convolutional

Generative Adversarial Nets framework.The generator now takes as input the input spectrum in addition to the random noise as earlier. The discriminator network also takes as input the input spectrum. Different non linear activations like ReLU, $\tanh$, leaky-ReLU, Sigmoid are used in the discriminator. The output of the CDCGAN is subjectively better than that of DCGAN.

ReLU vs. Leaky ReLU

- ReLU takes the maximum of the input value and zero. The network may get in one state where all the outputs are zeros. This is called the dying state in the literature.
- This situation can be prevented by using what is called a leaky ReLU function, it allows some values to pass through. The generator will start learning only if it receives gradients from the discriminator. However, if the gradients are all zeros the network will not learn anything but zeros!
- How much negative value to allow is controlled by α in leaky ReLU. It will allow all positive values, but when it comes to negative values only values above a certain α will be allowed. In this way we can tune the performance of the network.

Which activation and architecture to choose depends largely on the application and the performance that we desire.

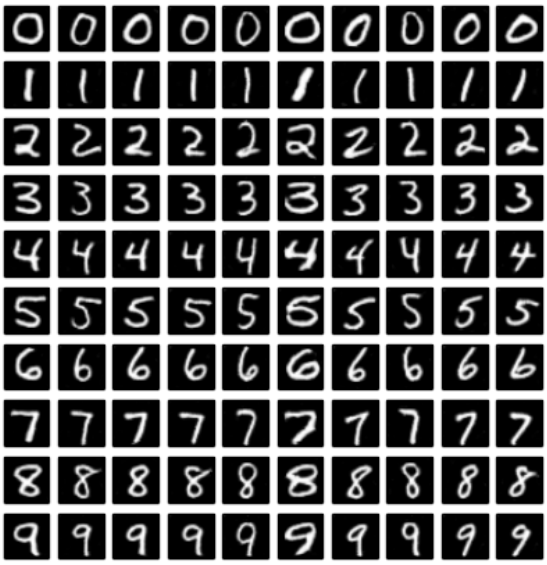

Fig. 8. MNIST data generated by CDCGAN after 20 epochs

Acknowledgements: I would like to thank Prof. Adrish Banerjee for allowing us to write a tutorial paper on the topic of our choice. Had it not been for his efforts, we would not have known how hard paper writing process actually is. We would also like to thank the student of EE390A for their feedback and helpful discussion during the writing of this article.

REFERENCES

[1] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.

[2] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," *CoRR*, vol. abs/1606.03498, 2016.

[3] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *arxiv*, 2016.

[4] J. Ho and S. Ermon, "Generative adversarial imitation learning," *CoRR*, vol. abs/1606.03476, 2016.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[6] I. J. Goodfellow, "NIPS 2016 tutorial: Generative adversarial networks," *CoRR*, vol. abs/1701.00160, 2017.

[7] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *CoRR*, vol. abs/1411.1784, 2014.

[8] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2015.