Generative Adversarial Networks

Stephan Halbritter
stephan.halbritter@haw-hamburg.de
Hamburg University of Applied Sciences, Dept. of Computer Science
Berliner Tor 7, 20099 Hamburg, Germany

I. Introduction

Artificial Neural Networks (ANN) and *Deep Learning* have attracted a great deal of attention in recent years, at the latest since 2012 AlexNet (Krizhevsky, Sutskever, and Hinton 2012) was the first ANN to win the *ImageNet Large Scale Visual Recognition Challenge (ILSVRC)*. It is a very fast moving field with an interesting mix of academic and industry sponsored research. Many of the research results are quickly finding their way into everyday applications.

A lot of the ideas and theories behind ANNs and Deep Learning are not new and go back to the 1960s. It's the major improvements in computational power like in GPUs as well as the availability of large amounts of data that made the recent breakthrough of Deep Learning possible.

As a result of these first breakthroughs, research focused for some time on problems with respect to these large amounts of data. *Conditional* or *Discriminative Models* can perform very well e.g. in computer vision, text and speech recognition, for pattern detection and data clustering.

In contrast to this, a *Generative Model* tries to create new samples. Most use the idea of maximum likelihood to estimate the probability distribution of the input data of the training set. Explicit density models can explicitly represent that probability distribution. Examples are *Boltzmann Machines* in different variants, *Deep Belief Networks* and *Variational Autoencoders* (Goodfellow, Bengio, and Courville 2016, 654ff).

Implicit density models have to find indirect ways to represent the probability distribution. In general this means they can produce samples from the distribution, but not the distribution itself (Goodfellow 2017). A very promising representative of implicit density models are *Generative Adversarial Networks (GAN)*.

II. Generative Adversarial Networks

The GAN framework was proposed by Ian Goodfellow and colleagues in May 2014 in a paper of the same name, »Generative Adversarial Networks« (Goodfellow et al. 2014). It was presented later that year at NIPS conference.

Since the publication of »Generative Adversarial Networks« until August 2017 more than 160 papers were published, which not only dealt with GANs but also gave their own name to their approach. The visualization in fig. 1 might give an idea of how popular the topic is right now. This is underlined by statements such as those by Geoffrey E. Hinton, who calls GANs »one of sort of biggest ideas in deep learning« and that they »have been a big breakthrough« (G. Hinton and Ng 2017).

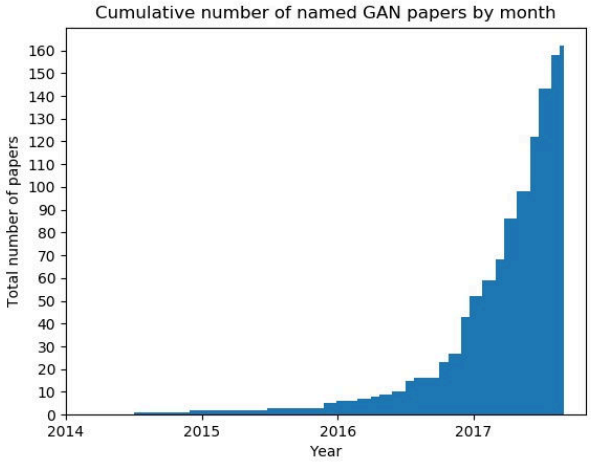

Figure 1. Over 160 papers that named GANs were published until Aug. 28, 2017 (Hindupur 2017)

A. Basic idea

The basic idea is to use not just a single artificial neural net, but two nets that train each other. One net, called a generator, takes noise as input and generates samples from it. The other one, called a discriminator, takes input

from a training set as well as from the generator. It then tries to decide if the input is from the training set and real or from the generator and fake. This process is like a competitive game in which on the one hand the generator learns how fool the discriminator with more and more realistic synthetic samples and on the other hand the discriminator gets better and better in distinguishing fake and real samples. In the long run and if everything goes well, this leads to a situation where the generator produces output which resembles the training data, leaving the discriminator with no other option but to guess blindly.

One often heard metaphor is that of comparing the generator to a cash counterfeiter, and comparing the discriminator to the police. As the counterfeiter improves his technique to produce counterfeit, so does the police in detecting it. This rat race forces the forger to develop better and better methods to produce ever more realistic-looking counterfeits that are increasingly difficult to distinguish from real money.

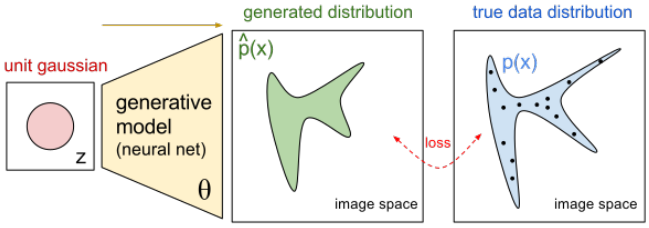

Figure 2. The basic idea of the GAN framework (Karpathy et al. 2016)

In principle, GANs are not restricted in the type of data they generate. However, most examples and research paper apply them to the creation of images. A main reason for this is because images are easy to validate by just looking at them and therefore give good examples to visualize results in a printed publication.

B. Training process

Some details of the training process as proposed in the original paper have been rectified several times. This overview refers to the improvements presented by Radford, Metz, and Chintala (2015). According to Goodfellow (2017), most GANs today are based on this DCGAN architecture (more about DCGAN in sec. III-C and sec. V-A).

The generator network G takes an n-dimensional vector z as input. z comes from a known probability distribution p_z, usually this is Gaussian noise. The idea of Radford, Metz, and Chintala (2015) is to use *fractionally-strided convolutions* to upsample z to the output dimensions by adding zeros between the pixels before convoluting over them, see fig. 4. G is modelled by putting multiple of these convolutional layers behind each other until we reach the desired output size. The example shown in fig. 3 upsamples to a final size of of 64×64 pixels and a depth of 3 for the color channels red, green and blue.

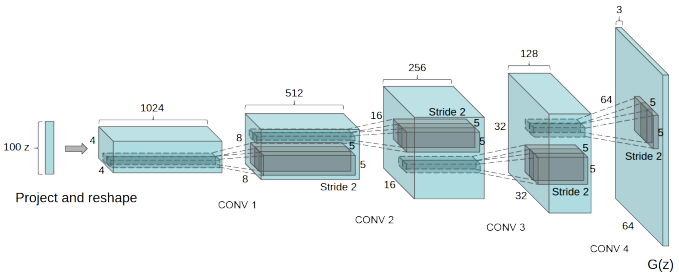

Figure 3. DCGAN generator using four fractional-strided convolutions (Radford, Metz, and Chintala 2015)

The discriminator network D takes a sample x and returns the probability that it is a real image from the training set with the unknown probability distribution p_{data}. For image generation in research, image databases like *ImageNet* or one of its subsets, e.g. *Caltech-UCSD Birds 200* or *Standford Dogs Dataset* are used most of the time. As they are well known, results can be easily compared with other methods.

We want to train the discriminator so that it maximizes $D(x)$ for every $x \sim p_{data}$ and minimizes for every $x \sim p_g$. For this task it uses the normal cross entropy cost function.

Using game theory, the whole training process can be interpreted as a zero-sum game as G's ability to fool D is balanced by D's ability to categorize its input into fake or true. A zero-game can be solved with the *Minimax Theorem* by finding the *Nash Equilibrium*:

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

The two neural networks are trained using *Stochastic Gradient Descent (SGD)*. By taking the respective gradients from eq. 1 they accordingly adjust their parameters θ_d and θ_g. It is also possible to use another variation of SGD, for example *Momentum* or *ADAM*. Every training iteration has two phases. In phase 1, D is updated for k steps. In practice most of the time k equals 1. In phase 2 G is updated. Algorithm 1 outlines this process.

If both networks D and G have unlimited resources, they might reach a Nash equilibrium. That means that D

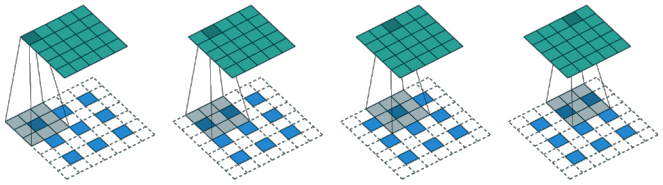

Figure 4. Example of fractionally-strided convolution from 3×3 input in blue to 5×5 output in green (Dumoulin and Visin 2016)

always produces an almost identical output for real as well as for fake data, i.e. being unable to distinguish between them. Or from the generator's perspective, G would generate samples with the same probability distribution as the training data, meaning that $p_g = p_{data}$ (Goodfellow et al. 2014).

Finding the equilibrium corresponds to minimizing the distance of the probability distribution of p_g and p_{data}. In the original paper, *Jensen-Shannon Divergence* was used for this task. Other GAN variations use different distance functions (for example WGAN, see sec. III-D).

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets (Adapted from Goodfellow et al. 2014)

for number of training iterations **do**
  for k steps **do**
    - Sample minibatch $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from $p_g(\boldsymbol{z})$.
    - Sample minibatch $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from $p_{\text{data}}(\boldsymbol{x})$.
    - Update D by ascending its stochastic gradient:

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

  end for
  - Sample minibatch $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from $p_g(\boldsymbol{z})$.
  - Update G by descending its stochastic gradient:

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

end for

III. Problems and Improvements

Many deep learning training processes consist of a single ANN and are trying to find a single global minimum. In general, this does work well and even when the minimum is just local, the result are often sufficient.

In GANs, there are two nets trying to minimize their cost. The goal is not to just to find the respective minima of these two nets but to achieve an Nash equilibrium between them both. In practice this setup leads to problems we do not have with single net architectures.

A. Vanishing gradient

In the very beginning of the training process, G produces very low quality samples and it is very easy for D to detect them with a high confidence. $D(G(z))$ will then output a value very close to 0 almost always, even if $G(z)$ makes larger changes. This results in what is called the *Vanishing Gradient Problem*: due to the very small gradients, G cannot improve anymore. A practical solution to this problem is to change the objective function of G. Instead of minimizing $\log(1-D(G(z)))$, G's new objective is to maximize $\log D(G(z))$. Another approach proposed by Chintala (2016) is to flip the labels from real to fake and vice versa when training the generator. As it is often the case, this trick comes from experience and not from some theoretical insight.

B. Mode collapse problem

Most probability distributions are *multi-modal*, i.e. they have multiple probability density peaks. Now it can happen that G learns to fool D by imitating just a small subset of those peaks. As D gets better in recognizing a fake input, G jumps to another peak to fool D again, neglecting what it learned before about the probability distribution. And as D gets better this happens again. As a result, G never learns more than a small part of the distribution and the GAN never converges. Shown in fig. 5 is an example of the mode collapse problem, where it takes about 5000 training iterations of G focusing on one peak, D adapting to this deception attempt and G switching to the next peak.

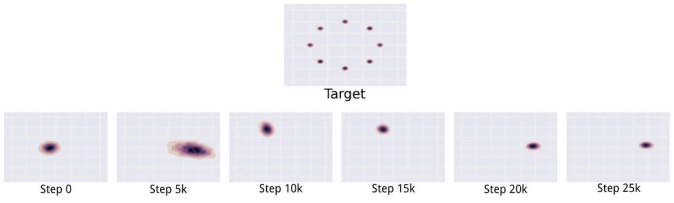

Figure 5. Mode collapse example (Goodfellow 2017)

»Improved Techniques for Training GANs« (Salimans et al. 2016) tries to overcome this problem with two methods. *Minibatch discrimination* gives the discriminator the possibility to compare the samples in a minibatch. If their entropy is too low and/or they are too similar, that might indicate that they are false. *Feature matching*

changes the objective function of G, so it tries to recreate values of an intermediate layer of D, not the end result of D. It is not clear why this works but they »hope to develop a more rigorous theoretical understanding in future work«.

C. Stabilizing training with architectural constrains

In »Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks« Radford, Metz and Chintala propose a few improvements to stabilize GANs by applying architectural constrains (Radford, Metz, and Chintala 2015). This architecture is called DCGAN. The changes include replacing all pooling layers with strided convolutions in D and fractional-strided convolutions in G as well as using ReLU activation, removing fully connected hidden layers for deeper architectures and other changes. Especially in regard of image creation, this architecture became the new standard.

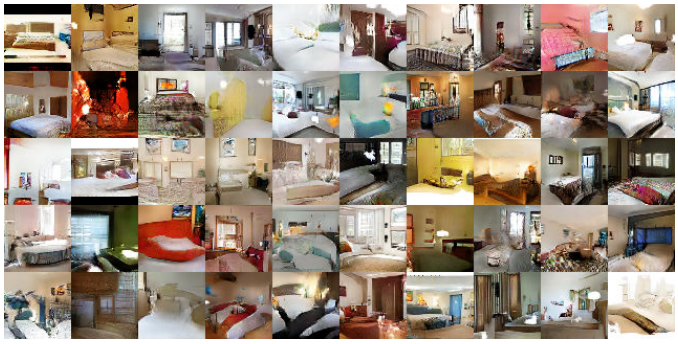

Figure 6. Generated images of bedrooms using a DCGAN (Radford, Metz, and Chintala 2015)

D. Better training by using Wasserstein distance

A recent approach to improve the training process is presented by Arjovsky, Chintala, and Bottou (2017) and is called *Wasserstein GAN (WGAN)*. They replace the Jensen-Shannon divergence with the Wasserstein or *Earth Mover Distance (EM)*. As a result, mode collapse is »drastically reduced«. One »of the most compelling practical benefits of WGANs is the ability to continuously estimate the EM distance by training the discriminator to optimality«. In practice, this means there is no need to carefully keep a balance between the discriminator and the generator while training. This makes designing GANs a lot easier.

This idea was once again improved by Gulrajani et al. (2017).

IV. Disadvantages and advantages

- As there is no explicit representation of a probability distribution p_{data} like in it is not clear how how to compare and validate results except by inspecting visual samples.
- It is hard to train a GAN as in practice the objective of $D(G(z))$ does not steadily improve but oscillates around the Nash equilibrium. Although this problem is addressed e.g. with the WGAN, there is also the question if a pure equilibrium does exist at all in general (Arora 2017).
- Sample generation is quite fast, as it only requires one pass through the model while Boltzmann machines need to repeatedly apply a Markov chain operator and PixelRNNs can only generate one pixel at a time!
- The general impression is that GANs generate images of higher quality. They are sharper and contain more details. This is true especially in contrast to generative models that use *mean squared error*, as this results in blurred images.

V. Interesting papers

As already mentioned in sec. II, there is no shortage of publications on the subject of GANs. In the following is a small and subjective selections of papers I found interesting and astounding.

A. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks (Radford, Metz, and Chintala 2015)

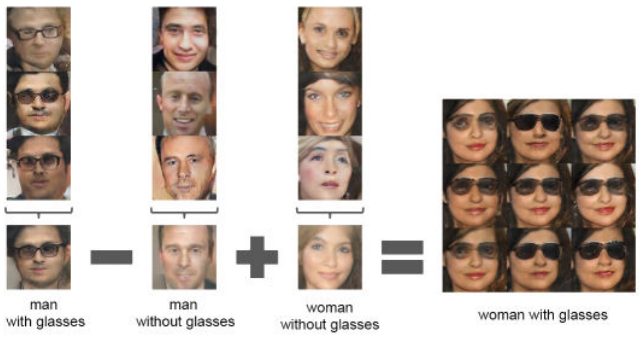

Figure 7. Doing vector math on input vectors representing a specific visual concept like »man with glasses« (Radford, Metz, and Chintala 2015)

A result of the DCGAN paper that is not mentioned so far is their evaluation of what the network learnt. On a trained model, each n-dimensional input vector z maps

to a specific output, e.g. a face. Averaging three different input vectors that represented similar visual concept like »man with glasses«, they could to simple math with those vectors as demonstrated in fig. 7. Similar to this they were able to generated smooth transitions between images by interpolation between a series of points in z.

B. Semantic Image Inpainting with Deep Generative Models (Yeh et al. 2016)

This paper proposes a solution for filling missing parts in images using a DCGAN. A very good introduction into this paper (and GANs in general) is provided by Amos (2016). Basic idea is the intuition that photo-realistic images are just samples of a specific probability distribution. As a GAN can learn to generate samples given a specific training set e.g. with faces, it can also learn to fill in missing parts of unknown face images.

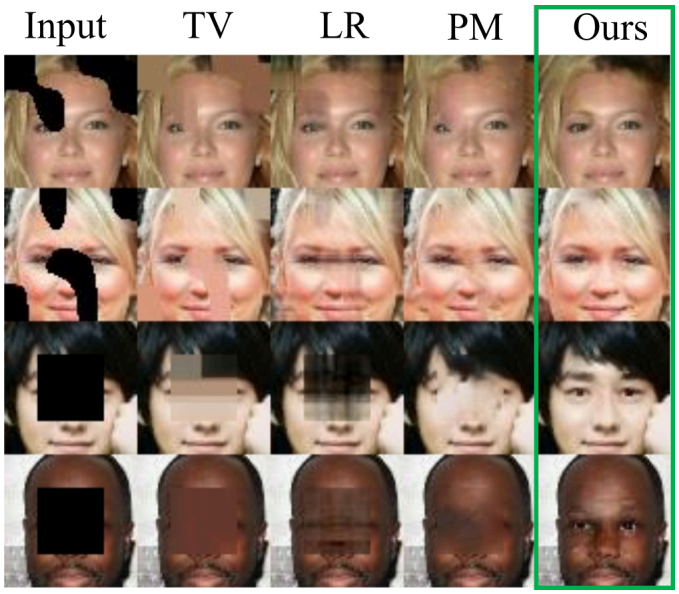

Figure 8. Image inpainting results compared with other methods (Yeh et al. 2016)

C. Beyond Face Rotation: Global and Local Perception GAN for Photorealistic and Identity Preserving Frontal View Synthesis (R. Huang et al. 2017)

This paper proposes a GAN architecture called *Two-Pathway Generative Adversarial Network (TPGAN)* for photo-realistic generation of frontal face views from a face image under a different pose. The architecture is an example for the increasing complexity of recent GANs. The generator consists of two pathways. One pathway processes the global structure, the other uses four networks to process local facial landmarks: left and right eyes, nose and mouth. The two pathways run simultaneously, their results get fused. The discriminator tries to distinguish between generated and real frontal views. See fig. 9 for a rather impressive example.

Figure 9. Synthesized frontal face view (left) created from a profile image (middle). On the right the ground truth frontal face (R. Huang et al. 2017)

D. StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks (H. Zhang et al. 2016)

StackGAN is a text-to-image GAN which can generate photo-realistic images of a size of up to 256x256px by interpreting text input. As its title implies, it stacks two GANs. The Stage-I GAN interprets the text description to generate a low-resolution image outlining the basic shapes and colors. The generator in Stage-II samples the results of the first one and learns to add additional details. In fig. 11 you see images generated for the input text »A small yellow bird with a black crown and a short pointed beak«. In the first row are results of Stage-I, below of Stage-II.

The results are very good and with great and correct detail, even when we impute they are curated. The basic idea to divide a larger goal into smaller subtask - generate a rough outline first, fill in details later - seems to be a promising approach to tackle larger problems. There are clear parallels between the problems solved with StackGAN and TPGAN. It would be interesting to see how they perform on solving the other task.

VI. Current research topics

By and large, research is divided in theoretical and practical in research.

On the practical side, people try to improve on existing results, e.g. generate high quality images larger than

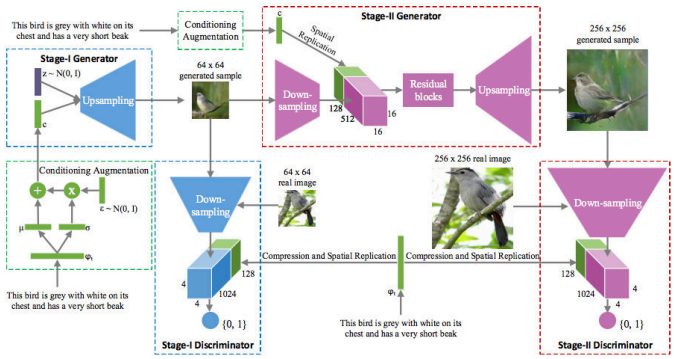

Figure 10. The architecture of StackGAN gets quite complicated (H. Zhang et al. 2016)

256×256px. There are also approaches to apply GANs on sequencing media like sound and videos. Some interesting research takes place in the field of medicine, e.g. for anomaly detection (Schlegl et al. 2017) or image synthesis to minimize the use of (radiation exposing) CT scans (Nie et al. 2016).

Theoretical research is focused on questions how things work. Large parts of Deep Learning in general are still not well understood. A lot of quite functioning improvements and best practices regarding hyperparameters and architecture design come from experience and lack a solid theoretical justification. They bear the danger of becoming blackboxes and being applied »just because«.

Figure 11. StackGAN results trained on CUB dataset, synthesised from the input text »A small yellow bird with a black crown and a short pointed beak«. (H. Zhang et al. 2016)

VII. Conclusion and Outlook

In this text I outlined a quick and short overview on the topic of »Generative Adversarial Networks«, explaining the original implementation by Goodfellow et al. (2014) as well as some improvements and further developments on the topic since its publication in 2014. Also provided is some insight into a few of the current problems and solution approaches as well as interesting research results and examples.

One of the reasons I've been working on this topic was to give me a first introduction into the wide field of Artificial Neural Networks and Deep Learning, which it did provide. Although Generative Adversarial Networks are very interesting, I'm not sure at this point in time whether I will continue to work on exactly this area. But I will continue to work on the topic of Deep Learning and gain some practical experience in the upcoming semester.

VIII. Acknowledgment

This text would be much more difficult to read without the help provided by *DeepL*, a neural machine translator.

References

Amos, Brandon. 2016. “Image Completion with Deep Learning in TensorFlow.” http://bamos.github.io/2016/08/09/deep-completion.

Arjovsky, Martin, Soumith Chintala, and Léon Bottou. 2017. “Wasserstein GAN.” *arXiv:1701.07875*.

Arora, Sanjeev. 2017. “Generative Adversarial Networks (GANs), Some Open Questions.” *off the convex path*. http://www.offconvex.org/2017/03/15/GANs/.

Chintala, Soumith. 2016. “How to Train a Gan.” NIPS 2016 Workshop on Adversarial Training. Barcelona, Spain. https://www.youtube.com/watch?v=X1mUN6dD8uE&t=190.

Dumoulin, Vincent, and Francesco Visin. 2016. “A Guide to Convolution Arithmetic for Deep Learning.” *arXiv:1603.07285*.

Goodfellow, Ian. 2017. “NIPS 2016 Tutorial: Generative Adversarial Networks.” *CoRR* abs/1701.00160. http://arxiv.org/abs/1701.00160.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.

Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. “Generative Adversarial Networks.” *arXiv:1406.2661*.

Gulrajani, Ishaan, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C. Courville. 2017. “Improved Training of Wasserstein Gans.” *CoRR* abs/1704.00028. http://arxiv.org/abs/1704.00028.

Hindupur, Avinash. 2017. “The GAN Zoo.” https://github.com/hindupuravinash/the-gan-zoo.

Hinton, Geoffrey, and Andrew Ng. 2017. “Heroes of Deep Learning: Andrew Ng interviews Geoffrey Hinton.” https://youtu.be/-eyhCTvrEtE?t=1647.

Huang, Rui, Shu Zhang, Tianyu Li, and Ran He. 2017. “Beyond Face Rotation: Global and Local Perception GAN for Photorealistic and Identity Preserving Frontal View Synthesis.” *arXiv:1406.2661*.

Karpathy, Andrej, Pieter Abbeel, Greg Brockman, Peter Chen, Vicki Cheung, Rocky Duan, Ian Goodfellow, et al. 2016. “Generative Models.” *OpenAI Blog*. https://blog.openai.com/generative-models/.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton. 2012. “ImageNet Classification with Deep Convolutional Neural Networks.” In *Advances in Neural Information Processing Systems 25*, 1097–1105. Curran Associates, Inc.

Nie, Dong, Roger Trullo, Caroline Petitjean, Su Ruan, and Dinggang Shen. 2016. “Medical Image Synthesis with Context-Aware Generative Adversarial Networks.” *CoRR* abs/1612.05362. http://arxiv.org/abs/1612.05362.

Radford, Alec, Luke Metz, and Soumith Chintala. 2015. “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks.” *CoRR* abs/1511.06434.

Salimans, Tim, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. “Improved Techniques for Training Gans.” *arXiv:1606.03498*.

Schlegl, Thomas, Philipp Seeböck, Sebastian M. Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. 2017. “Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery.” *CoRR* abs/1703.05921. http://arxiv.org/abs/1703.05921.

Yeh, Raymond, Chen Chen, Teck-Yian Lim, Mark Hasegawa-Johnson, and Minh N. Do. 2016. “Semantic Image Inpainting with Perceptual and Contextual Losses.” *CoRR* abs/1607.07539. http://arxiv.org/abs/1607.07539.

Zhang, Han, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaolei Huang, Xiaogang Wang, and Dimitris Metaxas. 2016. “StackGAN: Text to Photo-Realistic Image Synthesis with Stacked Generative Adversarial Networks.” *arXiv:1612.03242 [cs.CV]*.