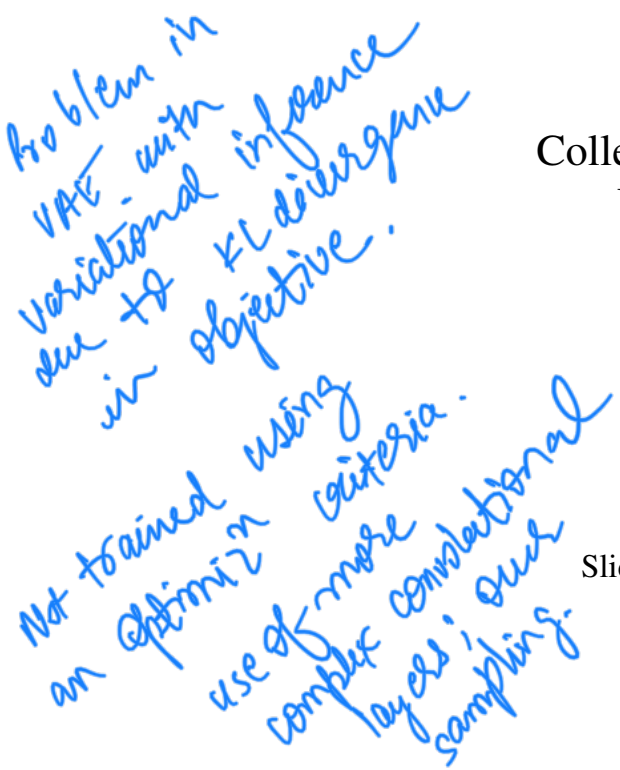
COMPSCI 689
Lecture 22: Generative Adversarial Networks

Benjamin M. Marlin

College of Information and Computer Sciences
University of Massachusetts Amherst



Slides by Benjamin M. Marlin (marlin@cs.umass.edu).

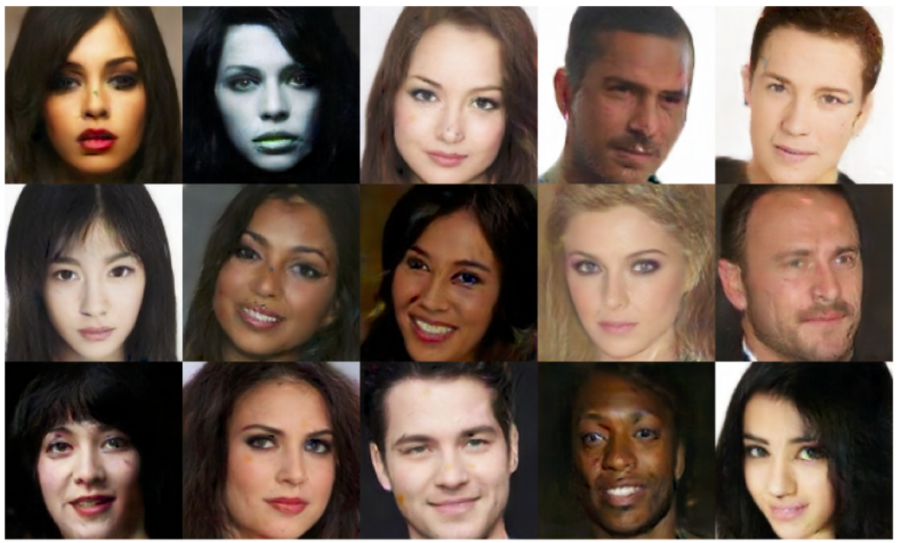
Outline

1 Introduction

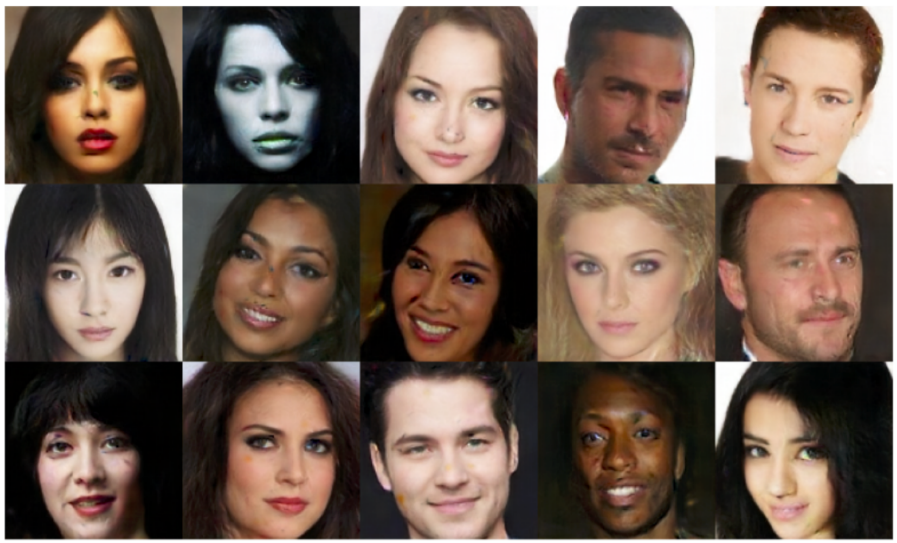
2 GANs

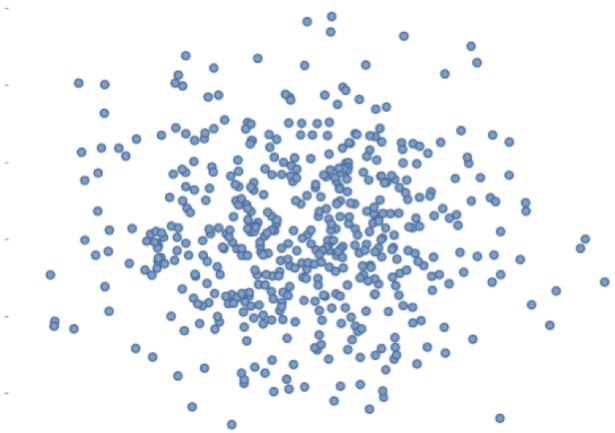
3 Convolutional GANs

Which Images are Real?



Which Images are Real?





Which Images are Real?

Image Credit: `https://arxiv.org/pdf/1912.00953.pdf`

Which Images are Real?

Image Credit: `https://arxiv.org/pdf/1912.00953.pdf`

Outline

1 Introduction

2 GANs

3 Convolutional GANs

Generative Adversarial Networks

- Generative adversarial networks are an approach to learning generative models for complex data like images.

Generative Adversarial Networks

- Generative adversarial networks are an approach to learning generative models for complex data like images.
- GANs use samples from a true distribution $p_*(\mathbf{X})$ to learn a generator that can transform random noise into new samples from the same distribution $p_*(\mathbf{X})$.

Generative Adversarial Networks

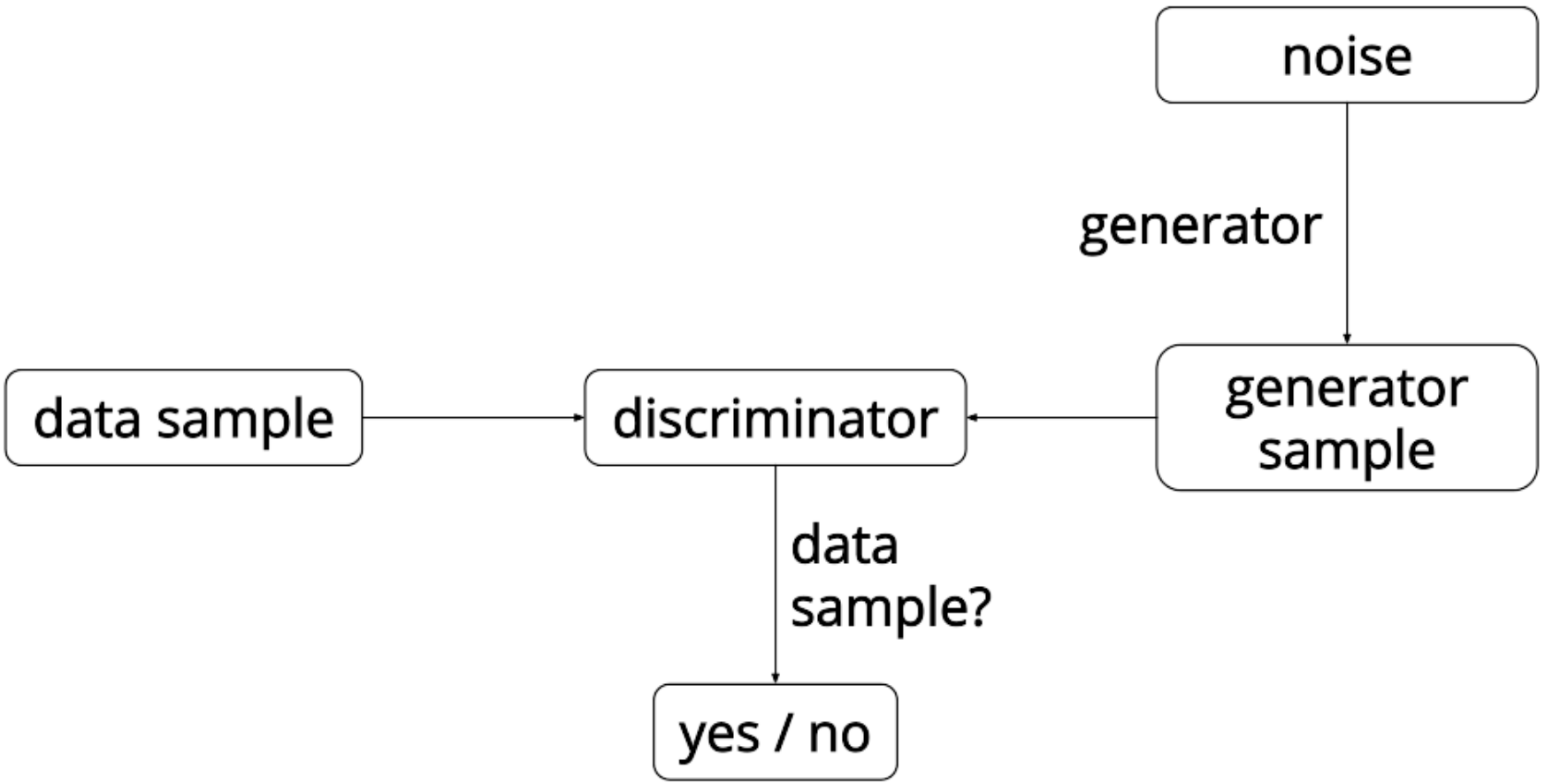
- Generative adversarial networks are an approach to learning generative models for complex data like images.
- GANs use samples from a true distribution $p_*(\mathbf{X})$ to learn a generator that can transform random noise into new samples from the same distribution $p_*(\mathbf{X})$.
- However, GANs do not provide an explicit probabilistic model $p(\mathbf{X}|\theta)$, and they require optimization to infer latent variables/codes.

Generative Adversarial Networks

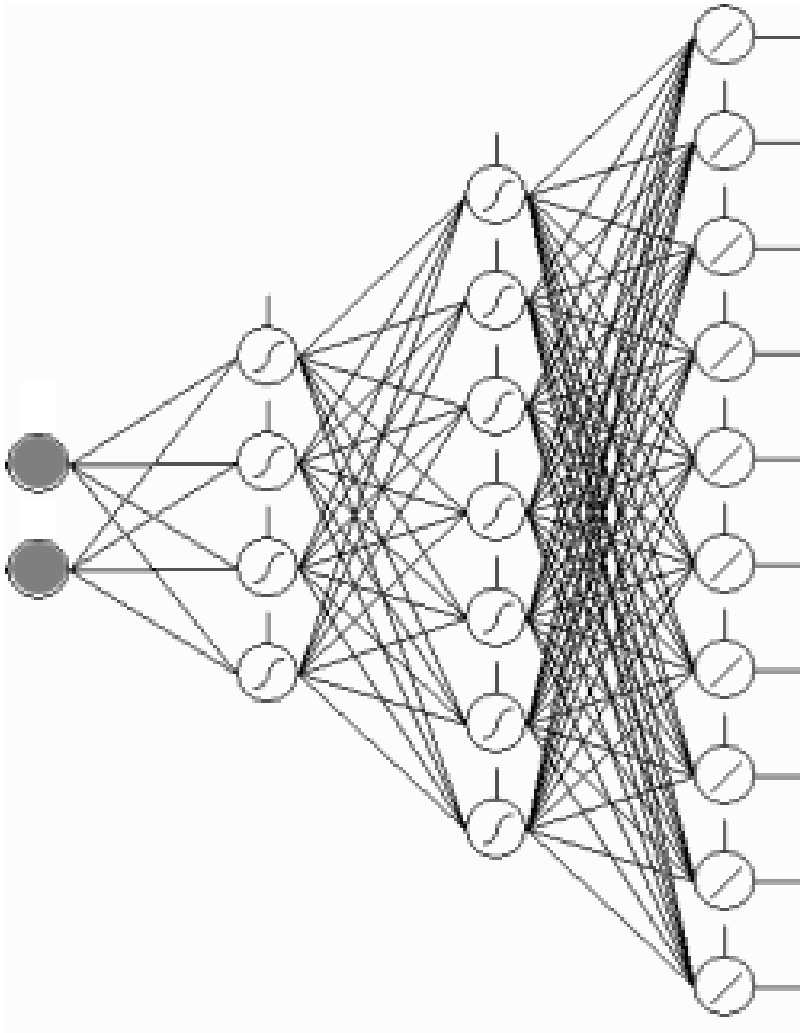
- Generative adversarial networks are an approach to learning generative models for complex data like images.
- GANs use samples from a true distribution $p_*(\mathbf{X})$ to learn a generator that can transform random noise into new samples from the same distribution $p_*(\mathbf{X})$.
- However, GANs do not provide an explicit probabilistic model $p(\mathbf{X}|\theta)$, and they require optimization to infer latent variables/codes.
- They are sometimes referred to as *implicit* probabilistic models.

don't give out a PDF.

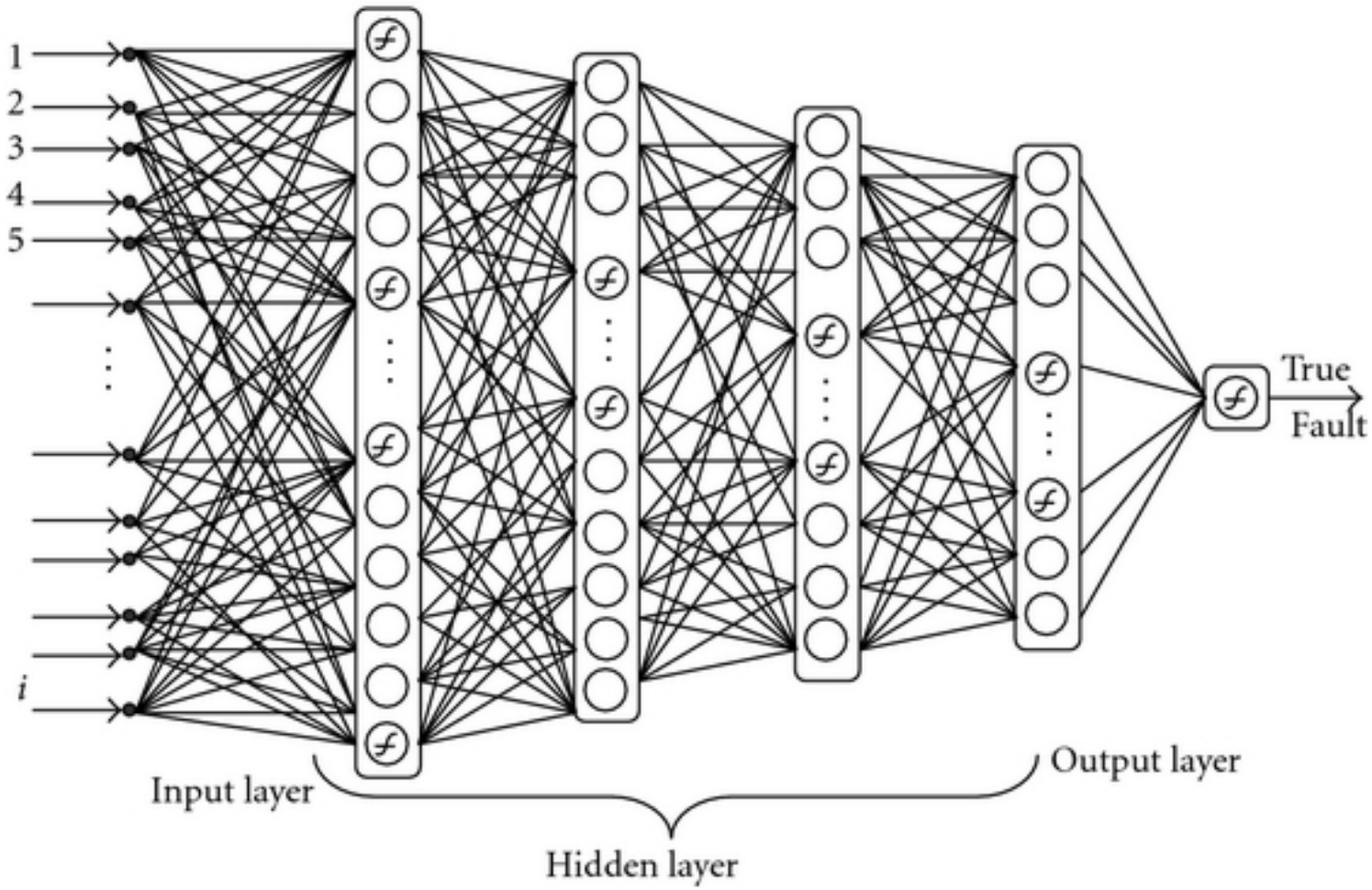
GAN Overview



GAN Generator



GAN Discriminator



GAN Learning

- The job of the the generator is to generate data samples from the model.

GAN Learning

- The job of the the generator is to generate data samples from the model.
- The job of the discriminator is to tell real data samples from generated samples.

GAN Learning

- The job of the the generator is to generate data samples from the model.
- The job of the discriminator is to tell real data samples from generated samples.
- The learning procedure for GANs does not follow any of the principles we've seen to date.

GAN Learning

- The job of the the generator is to generate data samples from the model.
- The job of the discriminator is to tell real data samples from generated samples.
- The learning procedure for GANs does not follow any of the principles we've seen to date.
- The learning problem is framed as a minimax game between the generator and discriminator.

GAN Learning

- The value function for the discriminator in the classical GAN formulation is given below. The generator receives the negative of this function as its reward.

$$\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

GAN Learning

- The value function for the discriminator in the classical GAN formulation is given below. The generator receives the negative of this function as its reward.

$$\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

- Here $D_{\theta_d}(x)$ is the probability that the discriminator thinks that x is real data.

GAN Learning

- The value function for the discriminator in the classical GAN formulation is given below. The generator receives the negative of this function as its reward.

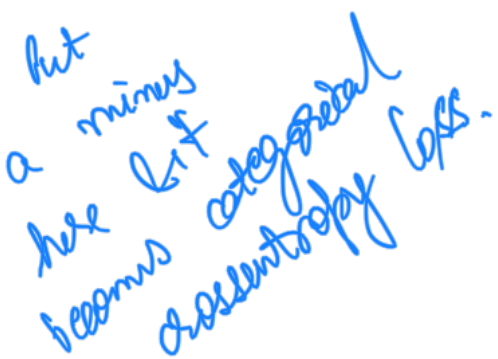
$$\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

- Here $D_{\theta_d}(x)$ is the probability that the discriminator thinks that x is real data.
- $G_{\theta_g}(z)$ is a sample generated by the GAN given noise sample z as input.

GAN Learning

- The reward functions are basically the (negative) cross entropy between the true labels indicating whether data examples are real or generated and the corresponding probabilities that data examples are real or generated according to the discriminator.

$$\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$



GAN Learning

- The reward functions are basically the (negative) cross entropy between the true labels indicating whether data examples are real or generated and the corresponding probabilities that data examples are real or generated according to the discriminator.

$$\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

- The larger this value is, the better the discriminator is at telling generated data from real data. The smaller, the better the generator is at fooling the discriminator.

GAN Learning

- Learning then takes the form of solving a minimax game as shown below.

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

GAN Learning

- Learning then takes the form of solving a minimax game as shown below.

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

- Solutions to the game take the form of parameters for the discriminator and the generator that are in *equilibrium* (e.g., a saddle point).

GAN Learning

- Most current learning algorithm for GANs try to reach such an equilibrium by alternately maximizing over the discriminator parameters and minimizing over the generator parameters.

GAN Learning

- Most current learning algorithm for GANs try to reach such an equilibrium by alternately maximizing over the discriminator parameters and minimizing over the generator parameters.
- The optimization problem for the discriminator is a standard binary classification problem:

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

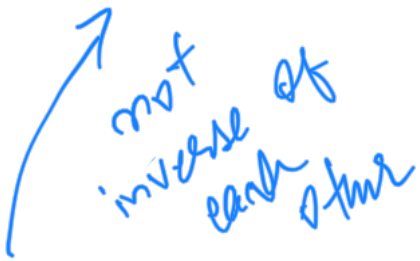
GAN Learning

- Most current learning algorithm for GANs try to reach such an equilibrium by alternately maximizing over the discriminator parameters and minimizing over the generator parameters.
- The optimization problem for the discriminator is a standard binary classification problem:

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

- The optimization problem for the generator is to minimize the "number of times" the discriminator is correct:

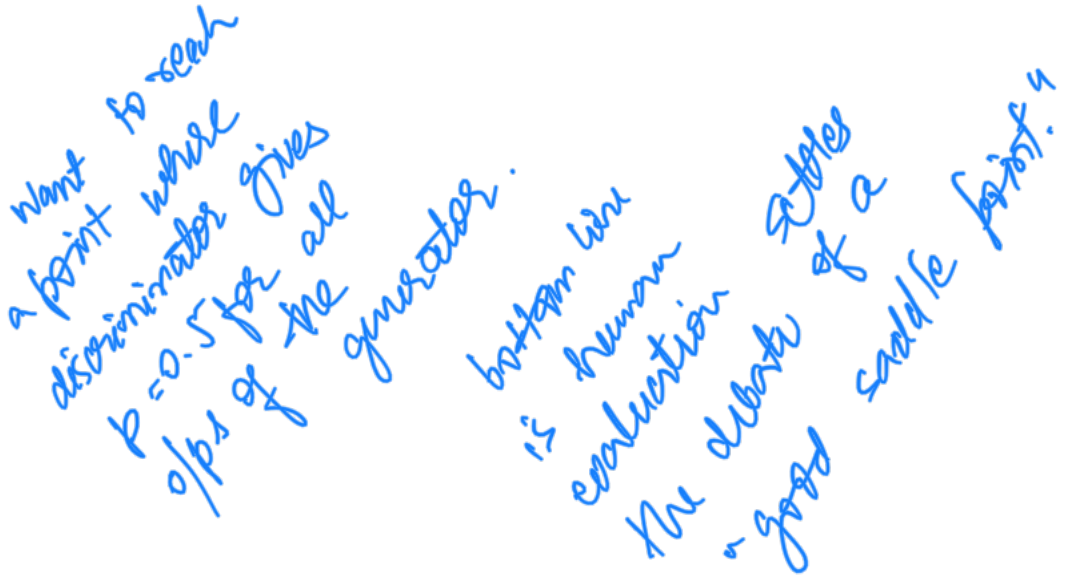
$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$



GAN Learning

- Due to issues with the scaling of the gradients during learning, having the generator maximize the number of times the discriminator is incorrect works better in practice:

$$\max_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(D_{\theta_d}(G_{\theta_g}(z)))$$



GAN Learning Algorithm

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D_{\theta_d}(x^{(i)}) + \log(1 - D_{\theta_d}(G_{\theta_g}(z^{(i)}))) \right]$$

end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by ascending its stochastic gradient (improved objective):

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(D_{\theta_d}(G_{\theta_g}(z^{(i)})))$$

end for

Basic GAN Results: Digits

Source: https://arxiv.org/abs/1406.2661 (2014)

Basic GAN Results: Faces

Source: https://arxiv.org/abs/1406.2661 (2014)

Outline

1 Introduction

2 GANs

3 Convolutional GANs

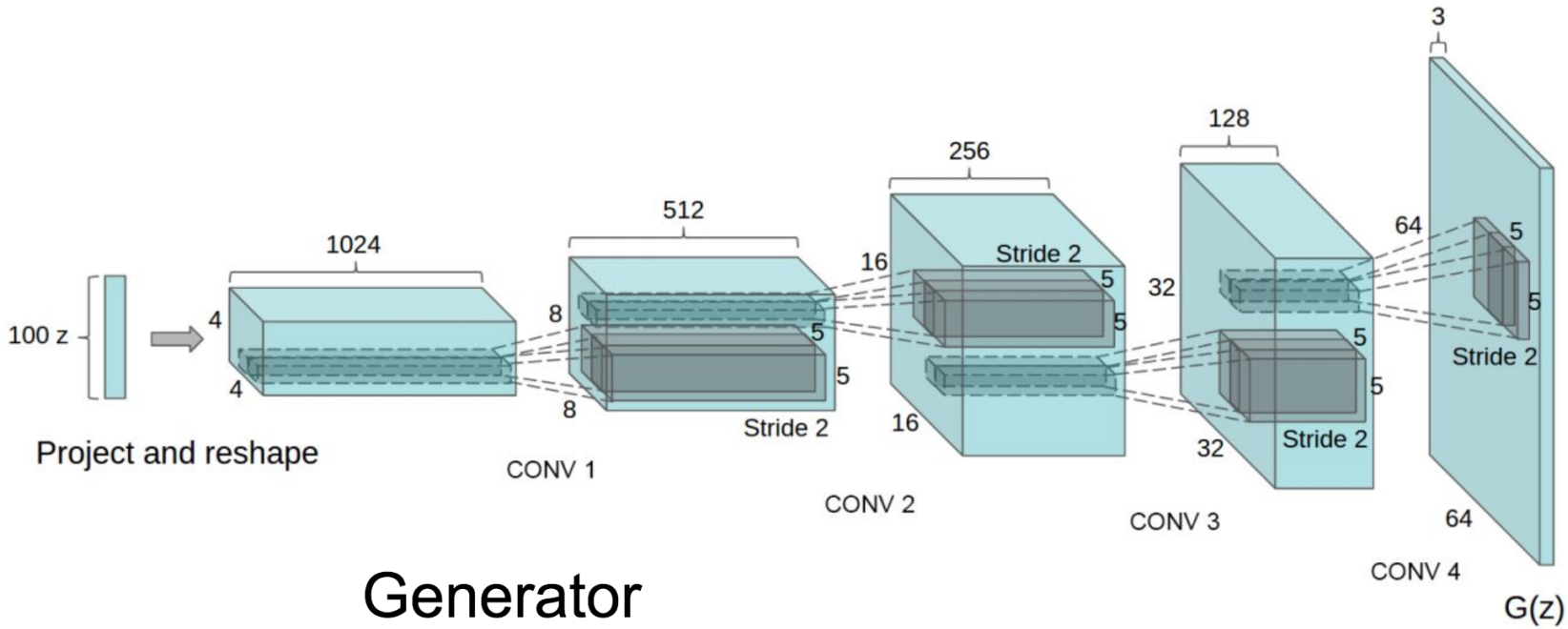
Improving GANs for Images

- To build a better GAN for images, we can leverage deep (de-)convolutional architectures.

Improving GANs for Images

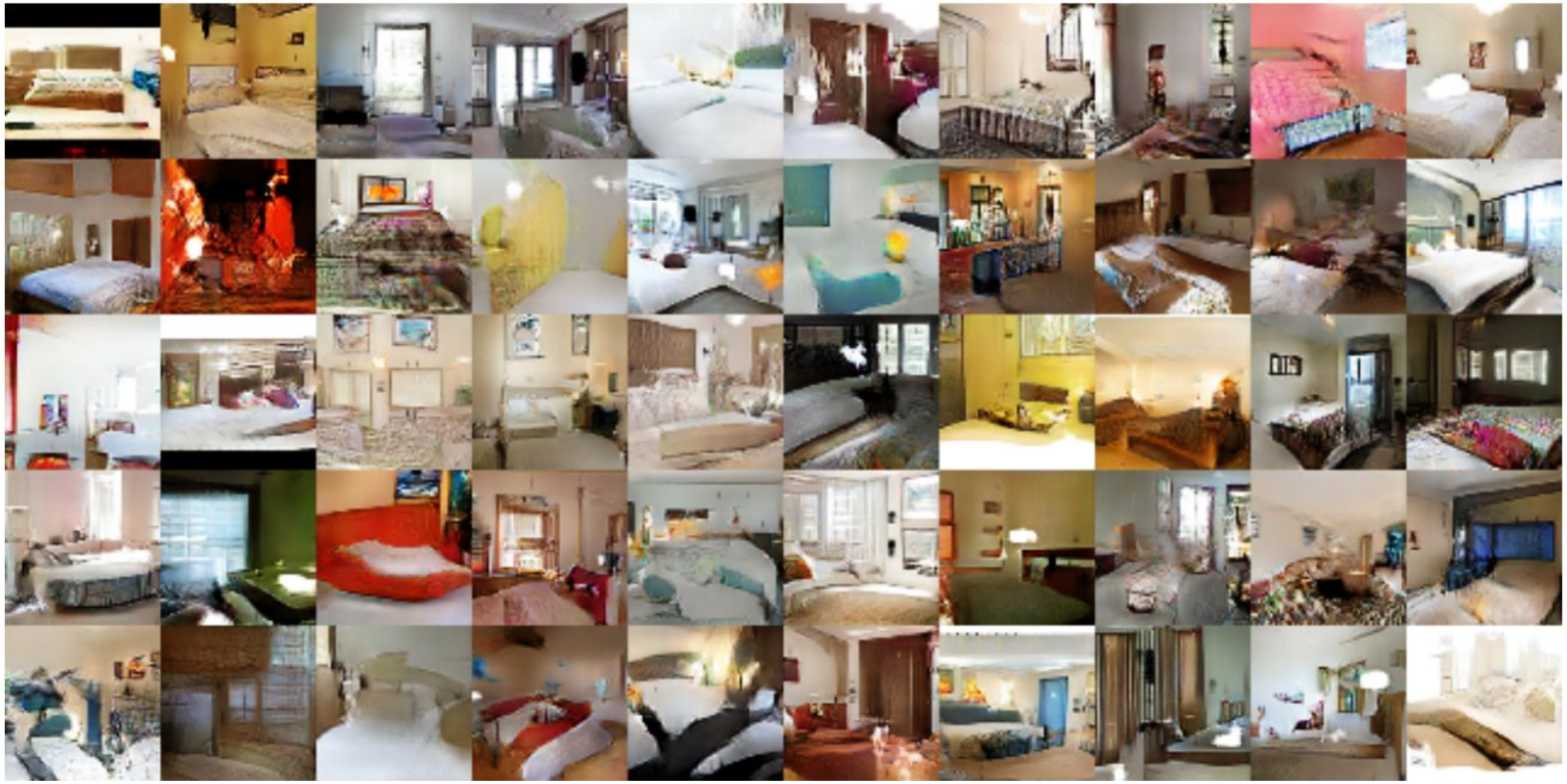
- To build a better GAN for images, we can leverage deep (de-)convolutional architectures.
- The DC-GAN and BE-GAN are two different convolutional GAN architectures that have been fine-tuned to generate (small) photo-realistic outputs.

DC-GAN Architecture



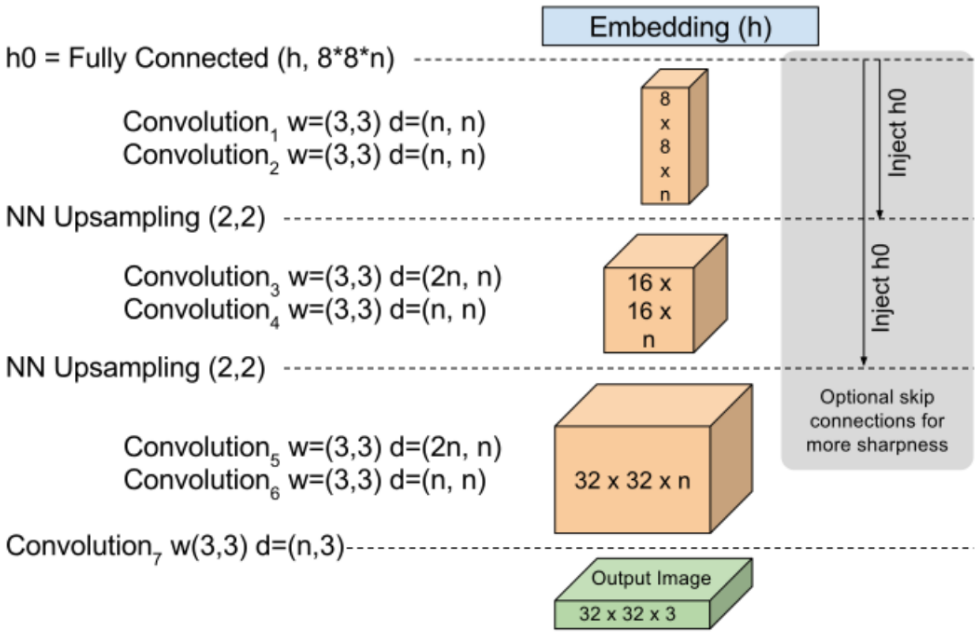
Generator

DC-GAN Examples

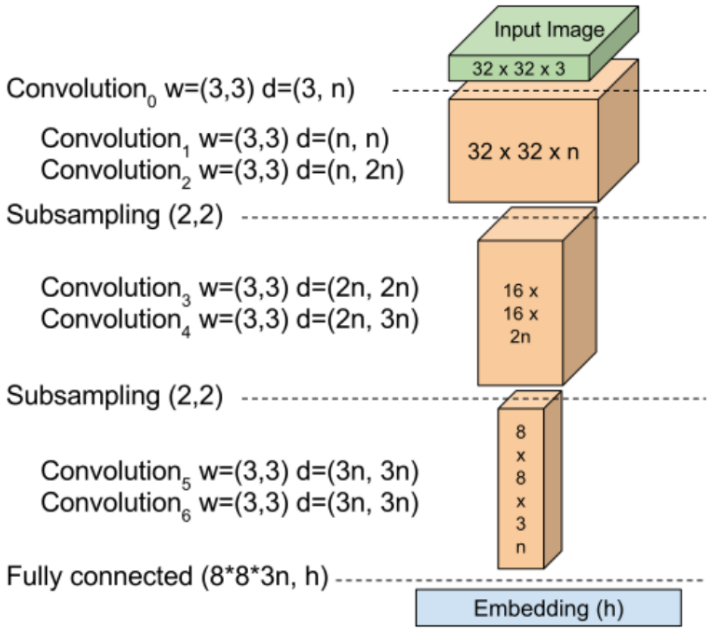


Source: `https://arxiv.org/abs/1511.06434` (2016)

BE-GAN Architecture



(a) Generator/Decoder



(b) Encoder

BE-GAN Examples

Source: `https://arxiv.org/abs/1703.10717` (2017)

BE-GAN Interpolations

Source: `https://arxiv.org/abs/1703.10717` (2017)

BigGAN and LOGAN

Image Credit: `https://arxiv.org/pdf/1912.00953.pdf`
(BigGAN 2018, LOGAN 2020)

Augmented GANS

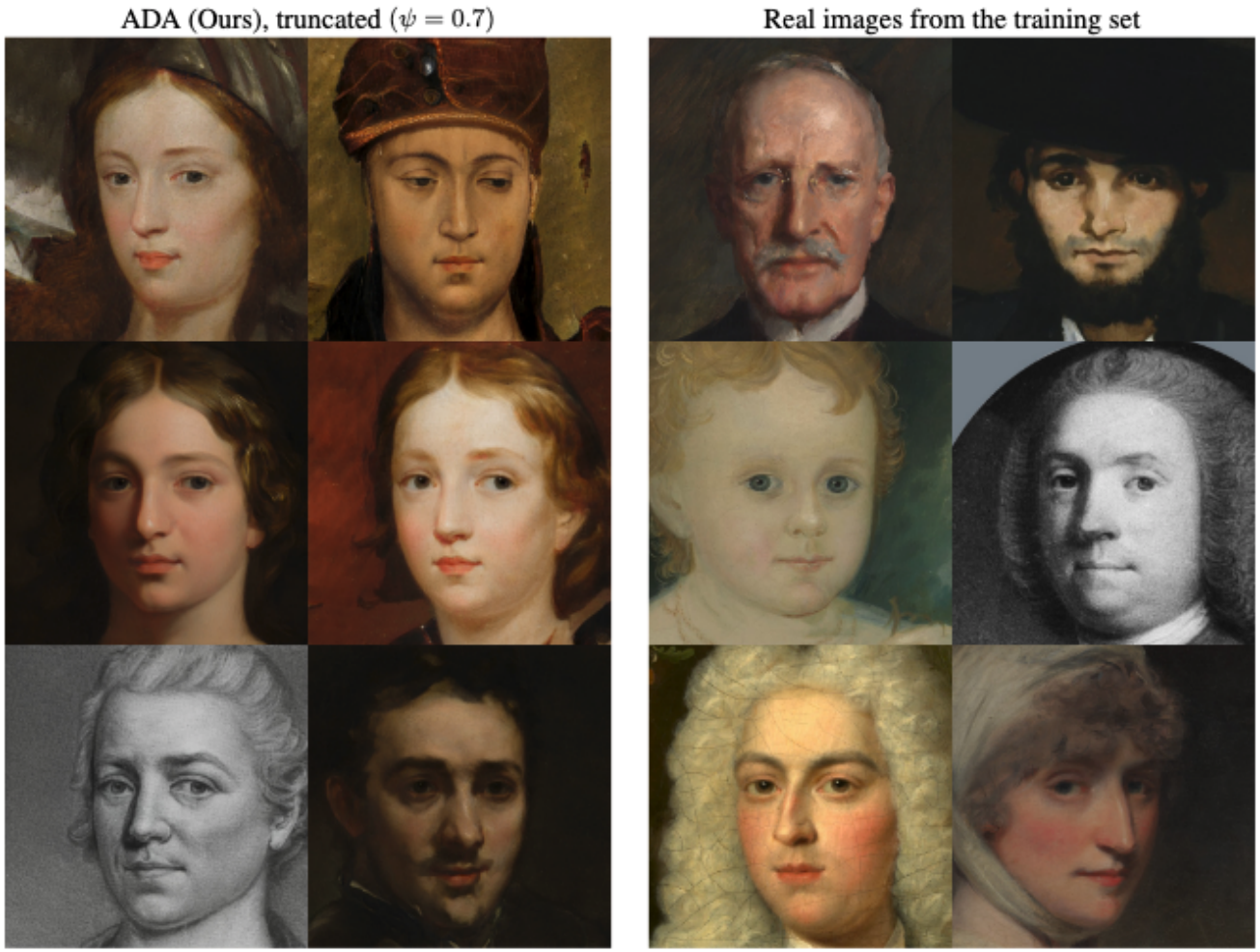


Image Credit: `https://arxiv.org/pdf/2006.06676.pdf`
(Augmented GANS 2020)

Augmented GANS

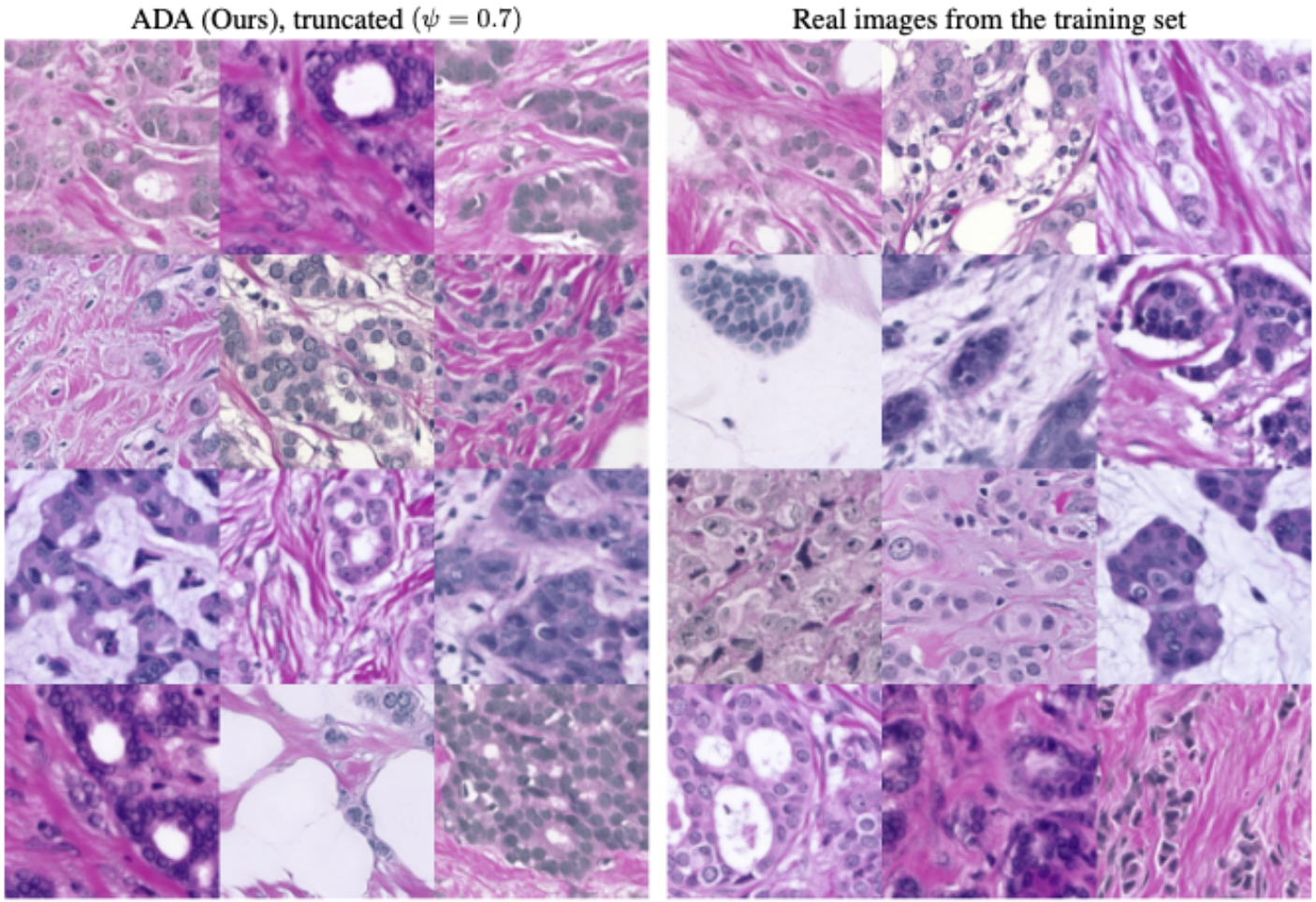


Image Credit: `https://arxiv.org/pdf/2006.06676.pdf`
(Augmented GANS 2020)

Augmented GANS

Image Credit: `https://arxiv.org/pdf/2006.06676.pdf`
(Augmented GANS 2020)

Augmented GANS

Image Credit: `https://arxiv.org/pdf/2006.06676.pdf`
(Augmented GANS 2020)

Even More GANs

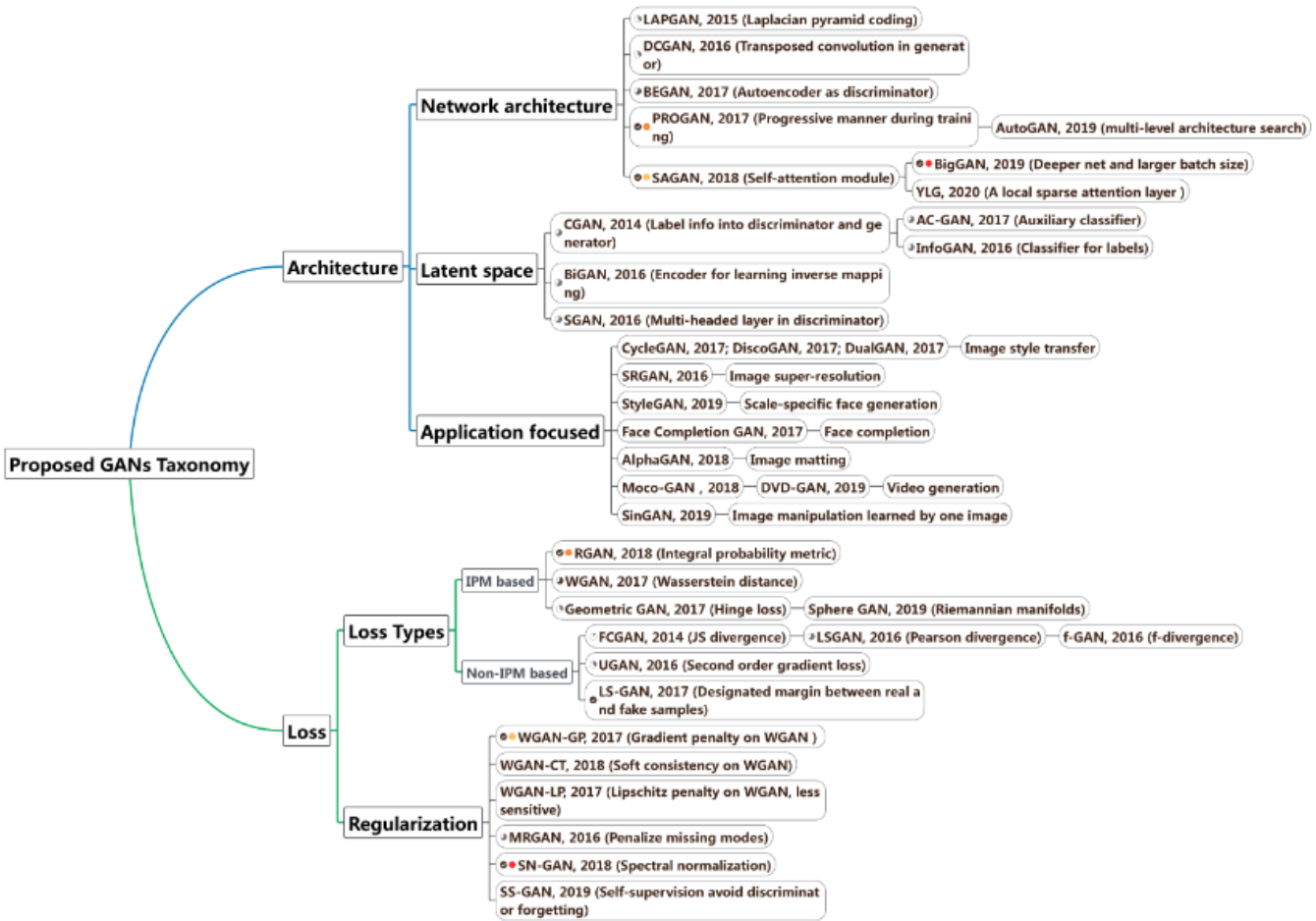


Image Credit: `https://arxiv.org/pdf/1906.01529.pdf`