Brief Review of Machine Learning

Yong Zhuang

Machine Learning

In this age of modern technology, there is one resource that we have in abundance: a large amount of structured and unstructured data. In the second half of the twentieth century, machine learning evolved as a subfield of artificial intelligence that involved the development of self-learning algorithms to gain knowledge from that data in order to make predictions. Instead of requiring humans to manually derive rules and build models from analyzing large amounts of data, machine learning offers a more efficient alternative for capturing the knowledge in data to gradually improve the performance of predictive models, and make data-driven decisions. Not only is machine learning becoming increasingly important in computer science research but it also plays an ever greater role in our everyday life. Thanks to machine learning, we enjoy robust e-mail spam filters, convenient text and voice recognition software, reliable Web search engines, challenging chess players, and, hopefully soon, safe and efficient self-driving cars.

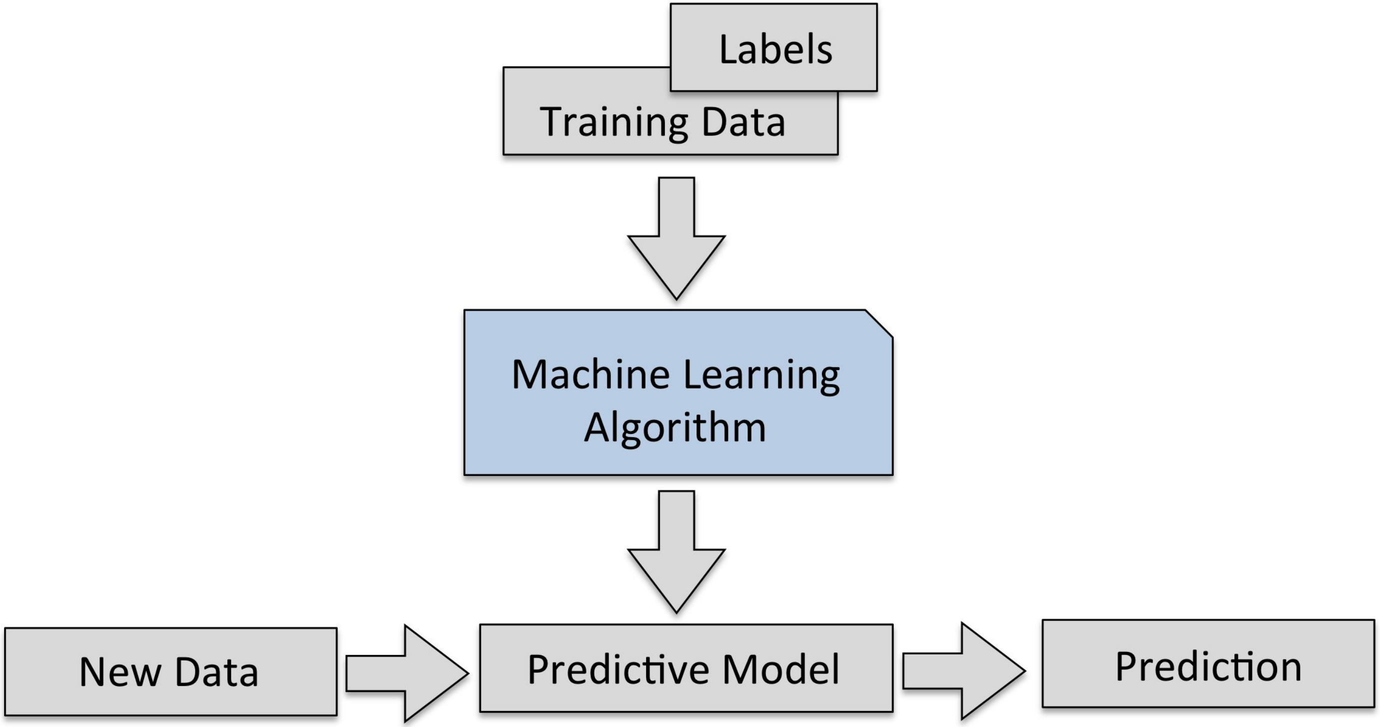
The three different types of machine learning

Unsupervised Learning

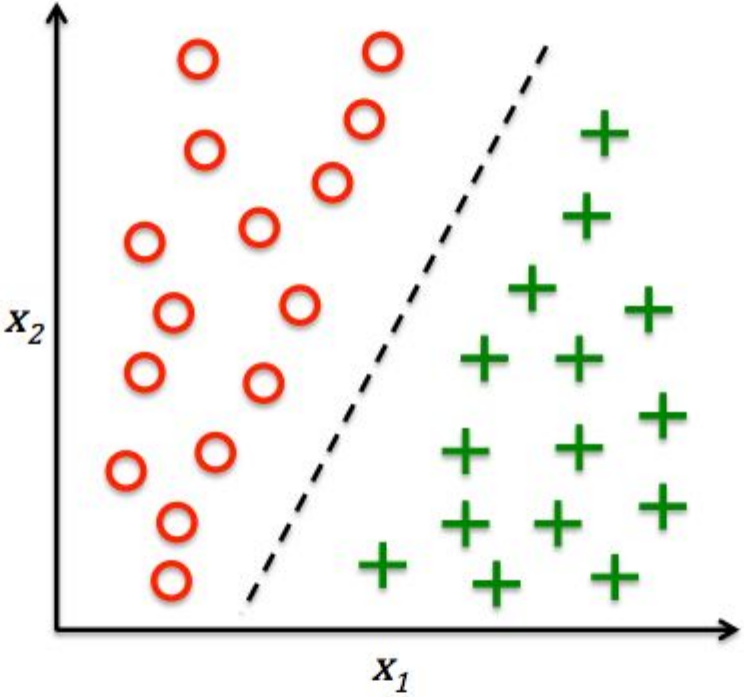
Supervised Learning

Reinforcement Learning

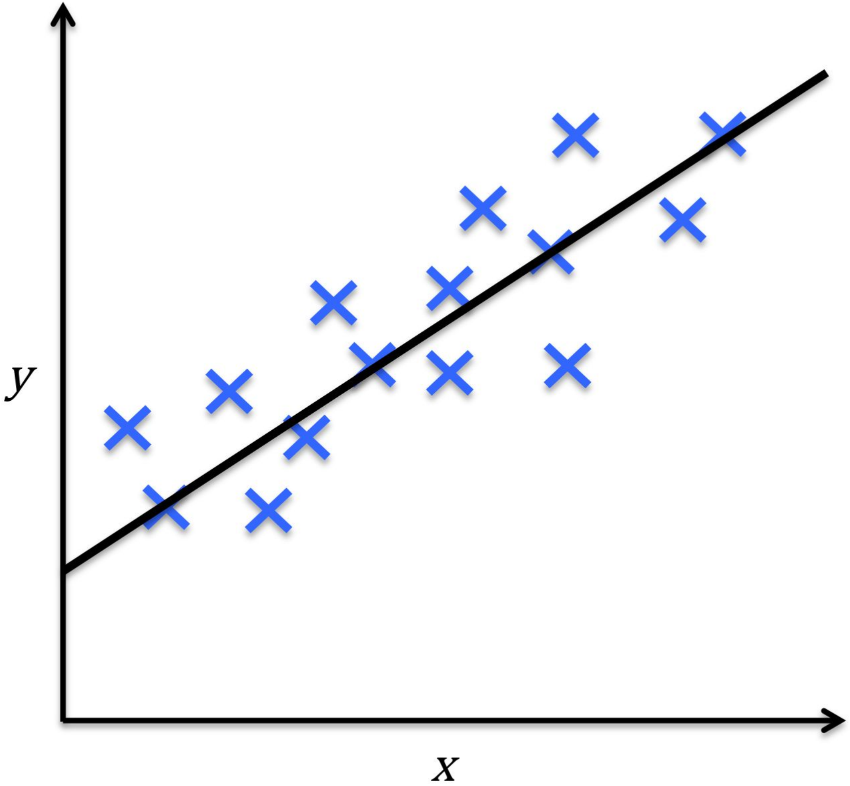
Supervised Learning



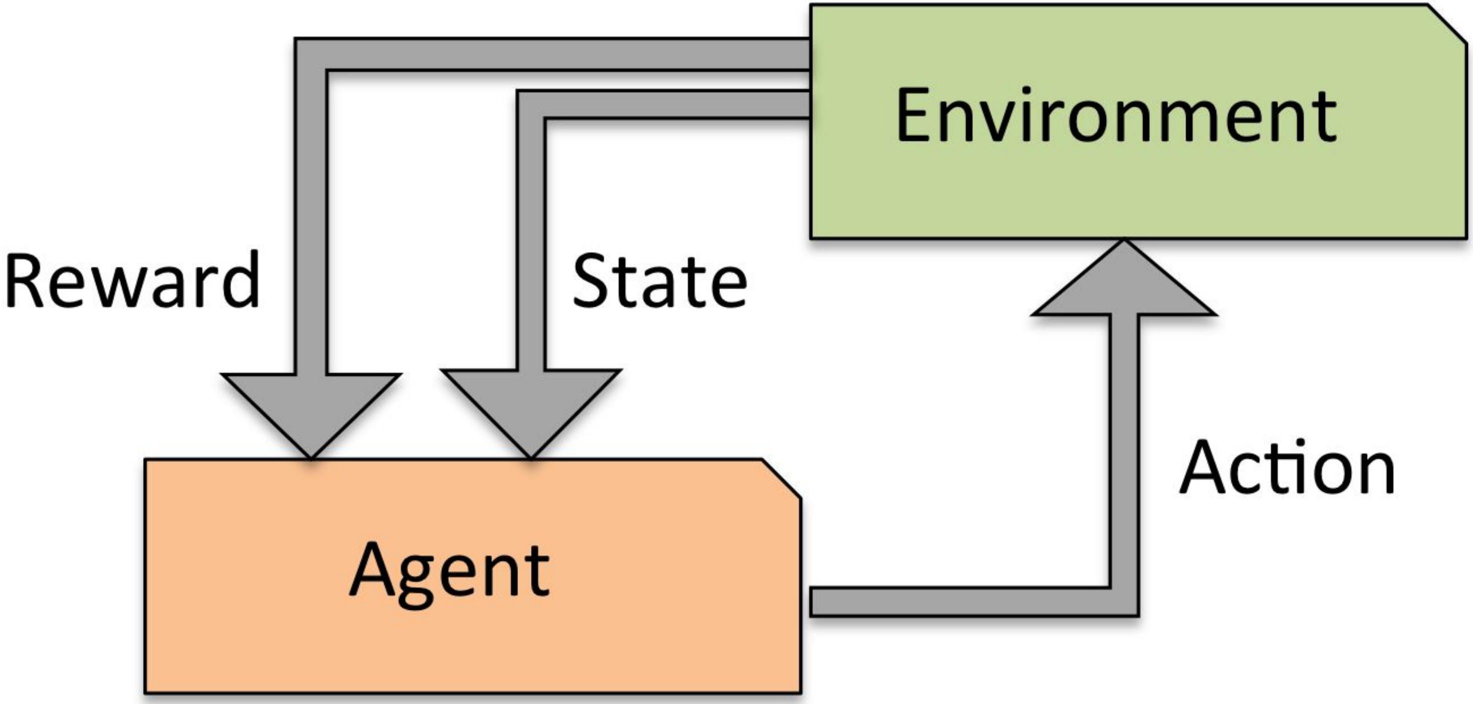
Classification for predicting class labels



Regression for predicting continuous outcomes



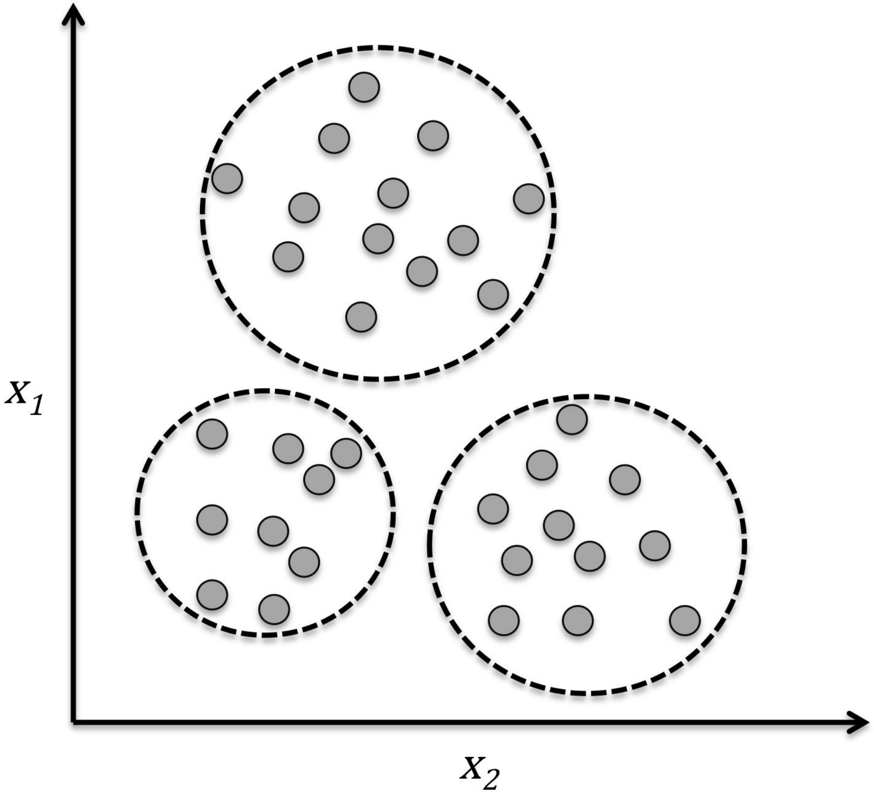
Solving interactive problems with reinforcement learning



Unsupervised learning

In supervised learning, we know the right answer beforehand when we train our model, and in reinforcement learning, we define a measure of reward for particular actions by the agent. In unsupervised learning, however, we are dealing with unlabeled data or data of unknown structure. Using unsupervised learning techniques, we are able to explore the structure of our data to extract meaningful information without the guidance of a known outcome variable or reward function.

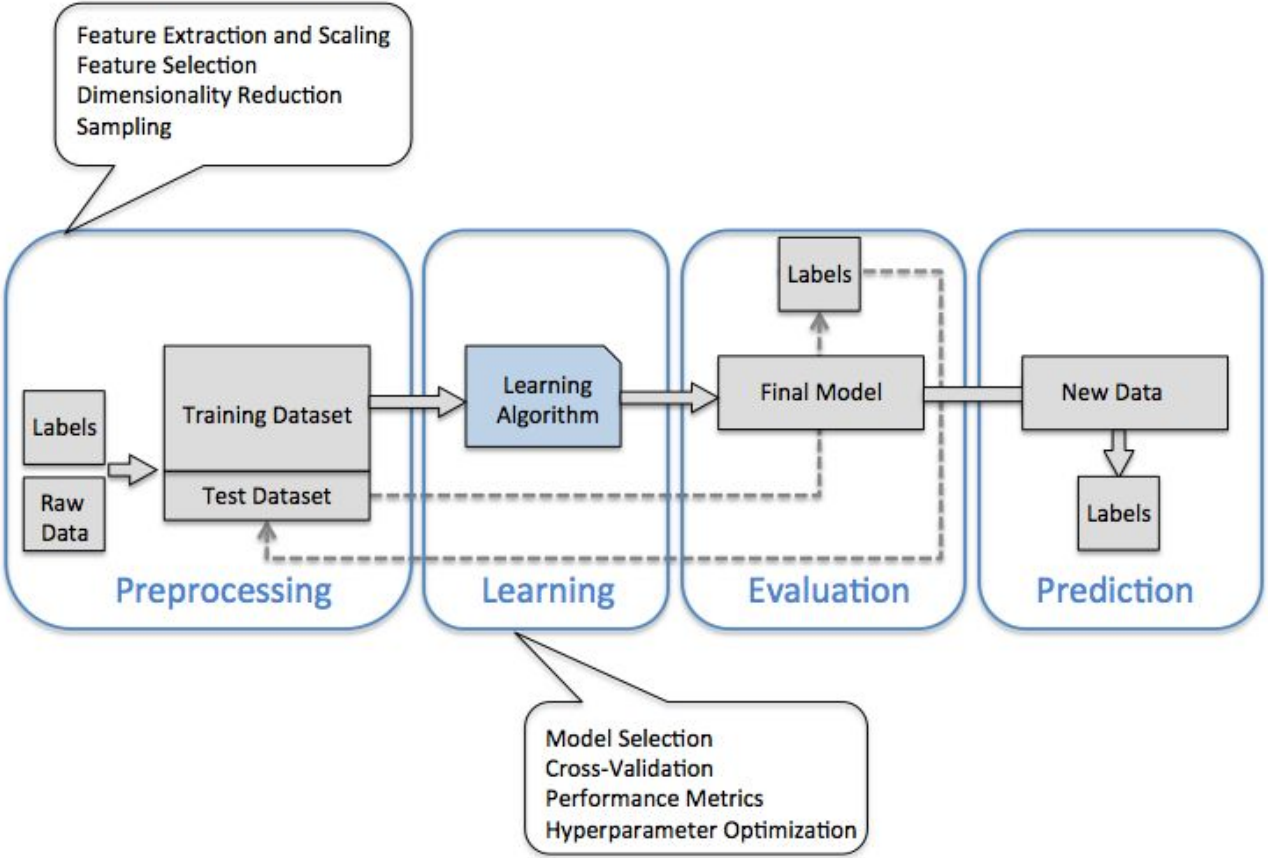
Finding subgroups with clustering



Dimensionality reduction

Another subfield of unsupervised learning is dimensionality reduction. Often we are working with data of high dimensionality—each observation comes with a high number of measurements—that can present a challenge for limited storage space and the computational performance of machine learning algorithms. Unsupervised dimensionality reduction is a commonly used approach in feature preprocessing to remove noise from data, which can also degrade the predictive performance of certain algorithms, and compress the data onto a smaller dimensional subspace while retaining most of the relevant information.

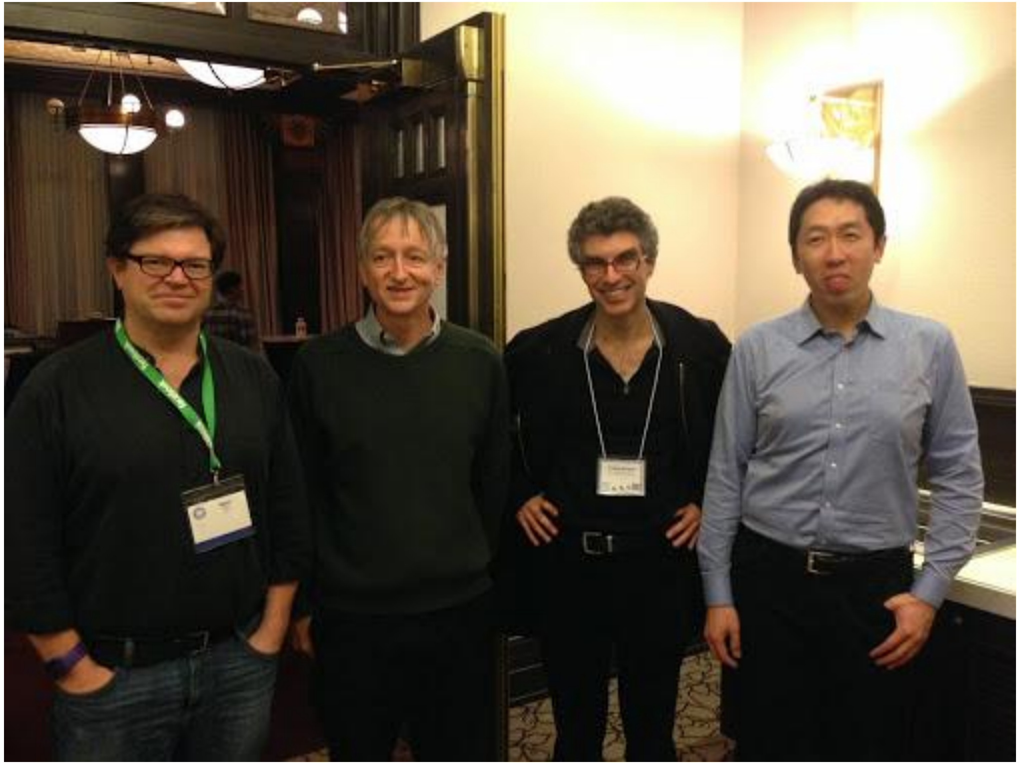
A roadmap for building machine learning systems



Generative Adversarial Networks (GAN)

Yong Zhuang

Big 5



Yann LeCun, Geoff Hinton, Yoshua Bengio, Andrew Ng

Michael I. Jordan

Generative Modeling

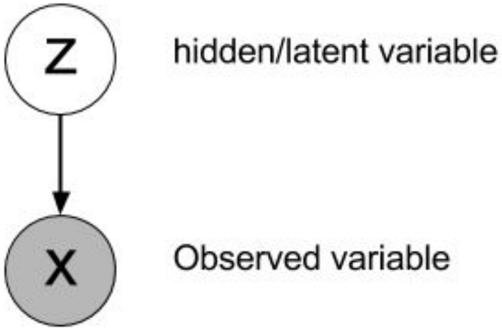
Can I use some noises to generate a picture?

Variational Inference

In short, variational inference is akin to what happens at every presentation you've attended. Someone in the audience asks the presenter a very difficult answer which he/she can't answer. The presenter conveniently reframes the question in an easier manner and gives an exact answer to that reformulated question rather than answering the original difficult question.

Variational Inference

In many interesting statistical problems, we can't directly calculate the posterior because the normalization constant is intractable. This happens often in latent variable models. For example assume that X represents a set of observations and Z represents a set of latent variables. If we are interested in the posterior P(Z|X), we know that :



$$P(Z|X) = \frac{P(Z,X)}{\int_z P(Z,X)}$$

but often times we can't calculate the denominator.

Variational Inference

Variational inference seeks to approximate the true posterior, P(Z|X), with an approximate variational distribution, which we can calculate more easily. For notation, let V be the parameters of the variational distribution.

$$P(Z|X) \approx Q(Z|V) = \prod_i Q(Z_i|V_i)$$

Variational Inference

Typically, in the true posterior distribution, the latent variables are not independent given the data, but if we restrict our family of variational distributions to a distribution that factorizes over each variable in Z (this is called a mean field approximation), our problem becomes a lot easier. We can easily pick each V_i so that Q(Z|V) is as close to P(Z|X) as possible when measured by Kullback Leibler (KL) divergence. Thus, our problem of interest is now selecting a V* such that

$$V^\star = \arg\min_V KL(Q(Z|V)||P(Z|X))$$

Once we arrive at a V*, we can use Q(Z|V*) as our best guess at the posterior when performing estimation or inference.

Connections to deep learning : Variational Autoencoders (VAE) Generative Adversarial Networks (GAN)

Generative Adversarial Networks (GAN)

GANs: Don’t work with any explicit density function!
Instead, take game-theoretic approach: learn to generate from training distribution through 2-player game

Problem: Want to sample from complex, high-dimensional training distribution. No direct way to do this!

Solution: Sample from a simple distribution, e.g. random noise. Learn transformation to training distribution.

Generative Adversarial Networks (GAN)

Problem: Want to sample from complex, high-dimensional training distribution. No direct way to do this!

Solution: Sample from a simple distribution, e.g. random noise. Learn transformation to training distribution.

What can we use to represent this complex transformation?

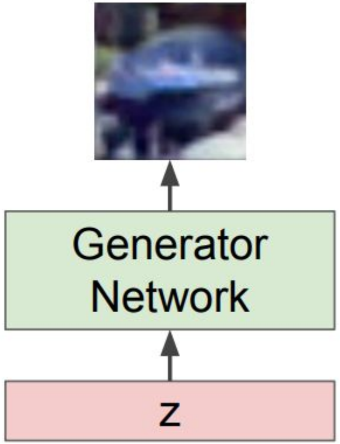
Generative Adversarial Networks (GAN)

Problem: Want to sample from complex, high-dimensional training distribution. No direct way to do this!

Solution: Sample from a simple distribution, e.g. random noise. Learn transformation to training distribution.

What can we use to represent this complex transformation?

Output: Sample from training distribution



Input: Random noise

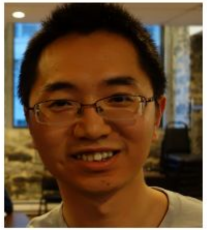
Generative Adversarial Networks (GAN)

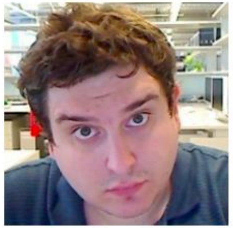
Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014


Ian
Goodfellow


Jean
Pouget-Abadie


Mehdi
Mirza


Bing
Xu


David
Warde-Farley


Sherjil
Ozair


Aaron
Courville


Yoshua
Bengio

Generative Adversarial Networks (GAN)

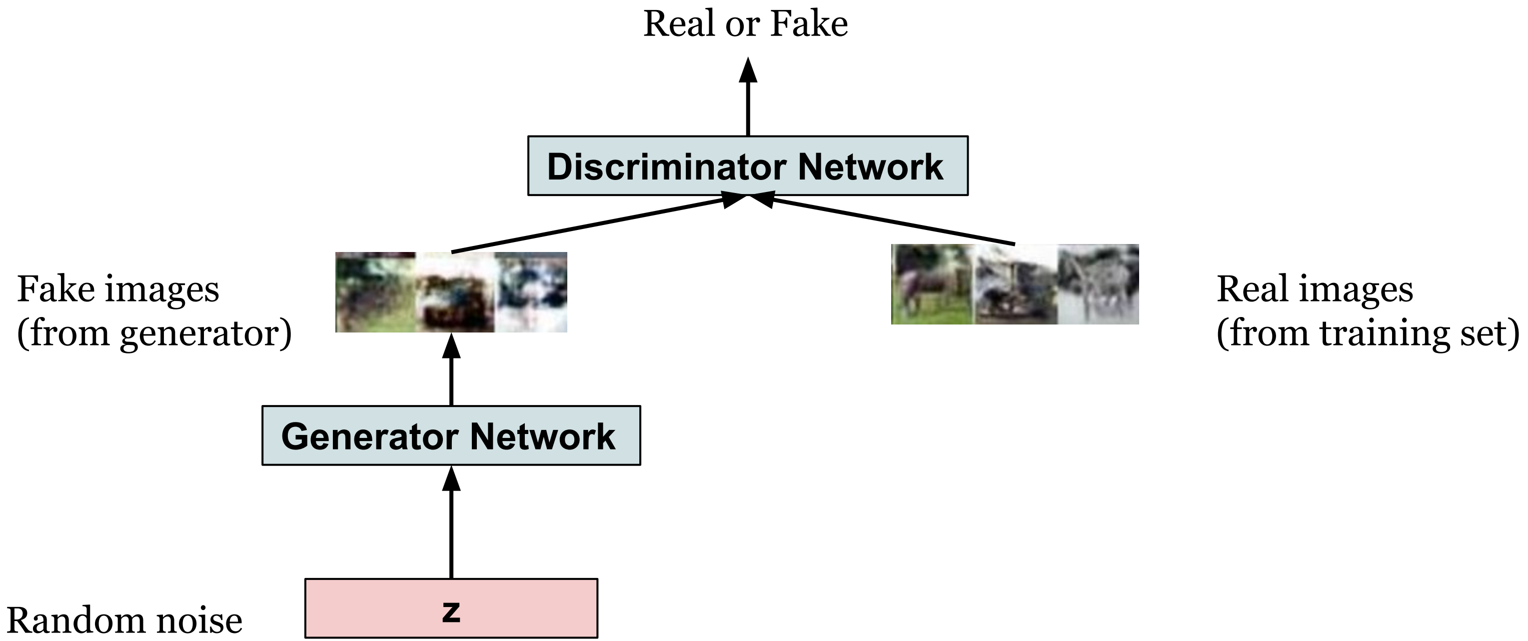
- Two networks:
  1. Discriminator D
  2. Generator G

- D tries to discriminate between:
  1. A sample from the data distribution.
  2. And a sample from the generator G.

- G try to fool the discriminator by generating real-looking images.
- D try to distinguish between real and fake images.

Generative Adversarial Networks (GAN)

- G try to fool the discriminator by generating real-looking images.
- D try to distinguish between real and fake images.



Joint Training in Minimax Game

Minimax objective function:

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

Joint Training in Minimax Game

Minimax objective function :

Discriminatory outputs likelihood in (0,1) of real image

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

Discriminatory output for real data x

Discriminatory output for generated fake data G(z)

Joint Training in Minimax Game

Minimax objective function :

Discriminatory outputs likelihood in (0,1) of real image

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

Discriminatory output for real data x

Discriminatory output for generated fake data G(z)

- Discriminator try to **maximize objective** such that D(x) is close to 1 for real data and D(G(z)) is close to 0 for fake data.
- Generator try to **minimize objective** such that D(G(z)) is close to 1

(discriminator is fooled into thinking generated G(z) is real)

Algorithm

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

    • Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.

    • Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.

    • Update the discriminator by ascending its stochastic gradient:

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

  end for

  • Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.

  • Update the generator 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

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Training

- **Gradient ascent** on discriminator

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

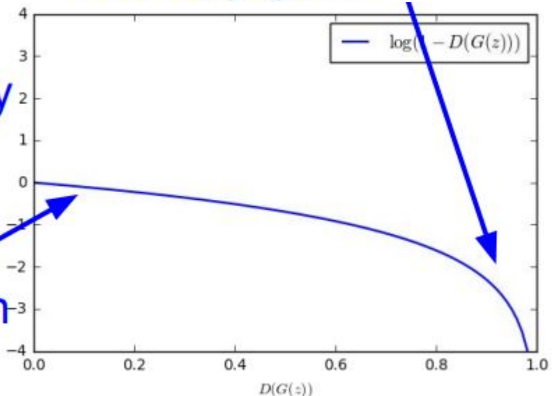
- **Gradient descent** on generator

$$\min_G \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

In practice, optimizing this generator objective does not work well!

Gradient signal dominated by region where sample is already good

When sample is likely fake, want to learn from it to improve generator. But gradient in this region is relatively flat!



UMASS BOSTON

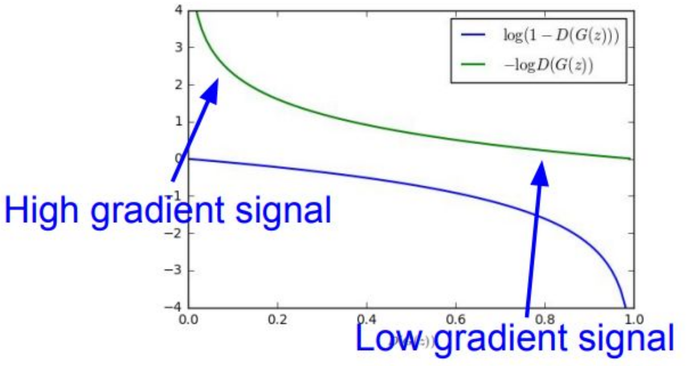
Training

- **Gradient ascent** on discriminator

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

- **Instead: Gradient ascent** on generator

$$\max_G \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(D(G(\boldsymbol{z})))]$$



Instead of minimizing likelihood of discriminator being correct, now maximize likelihood of discriminator being wrong. Same objective of fooling discriminator, but now higher gradient signal for bad samples => works much better! Standard in practice.

Algorithm

Some find k=1 more stable, others use k>1, no best rule

Recent work (e.g. Wasserstein GAN) alleviates this problem, better stability!

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

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:

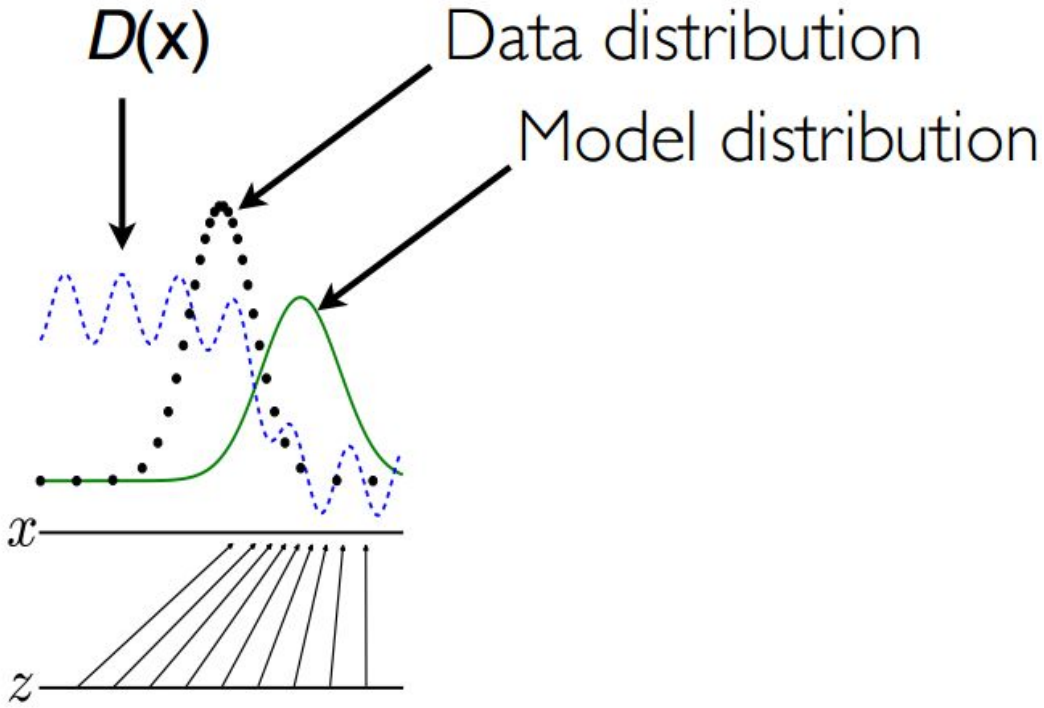
$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right].$$

  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)**:

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

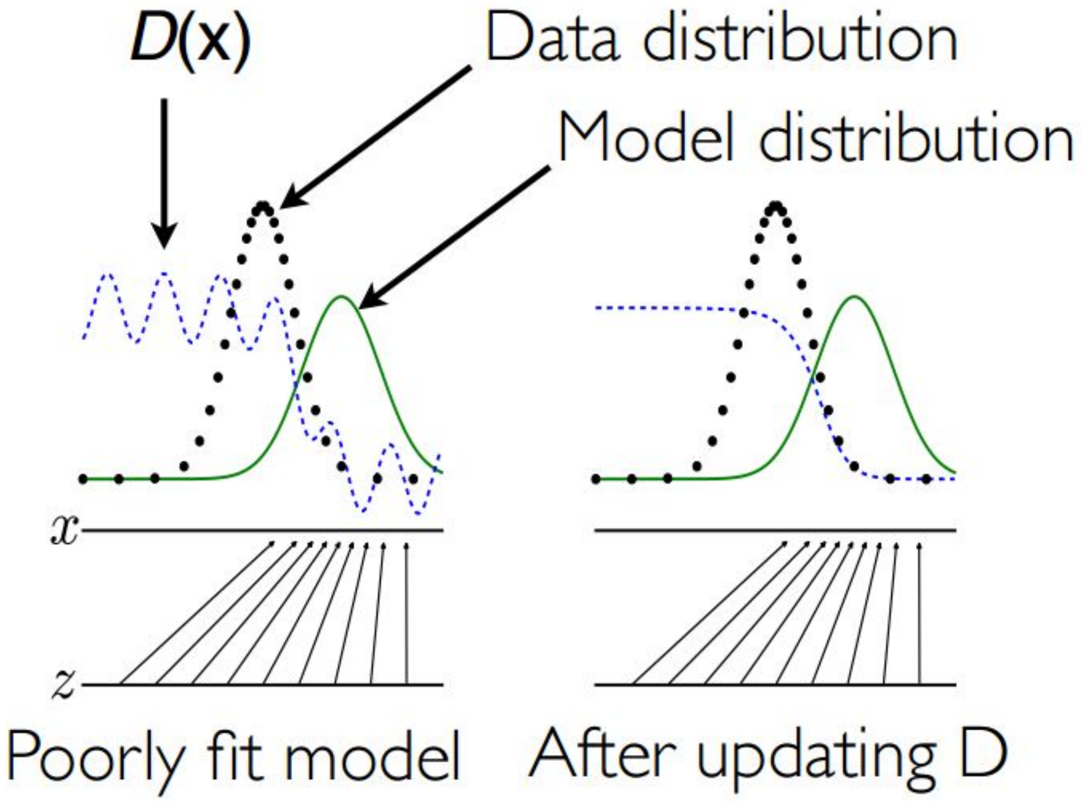
end for
The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Learning Process

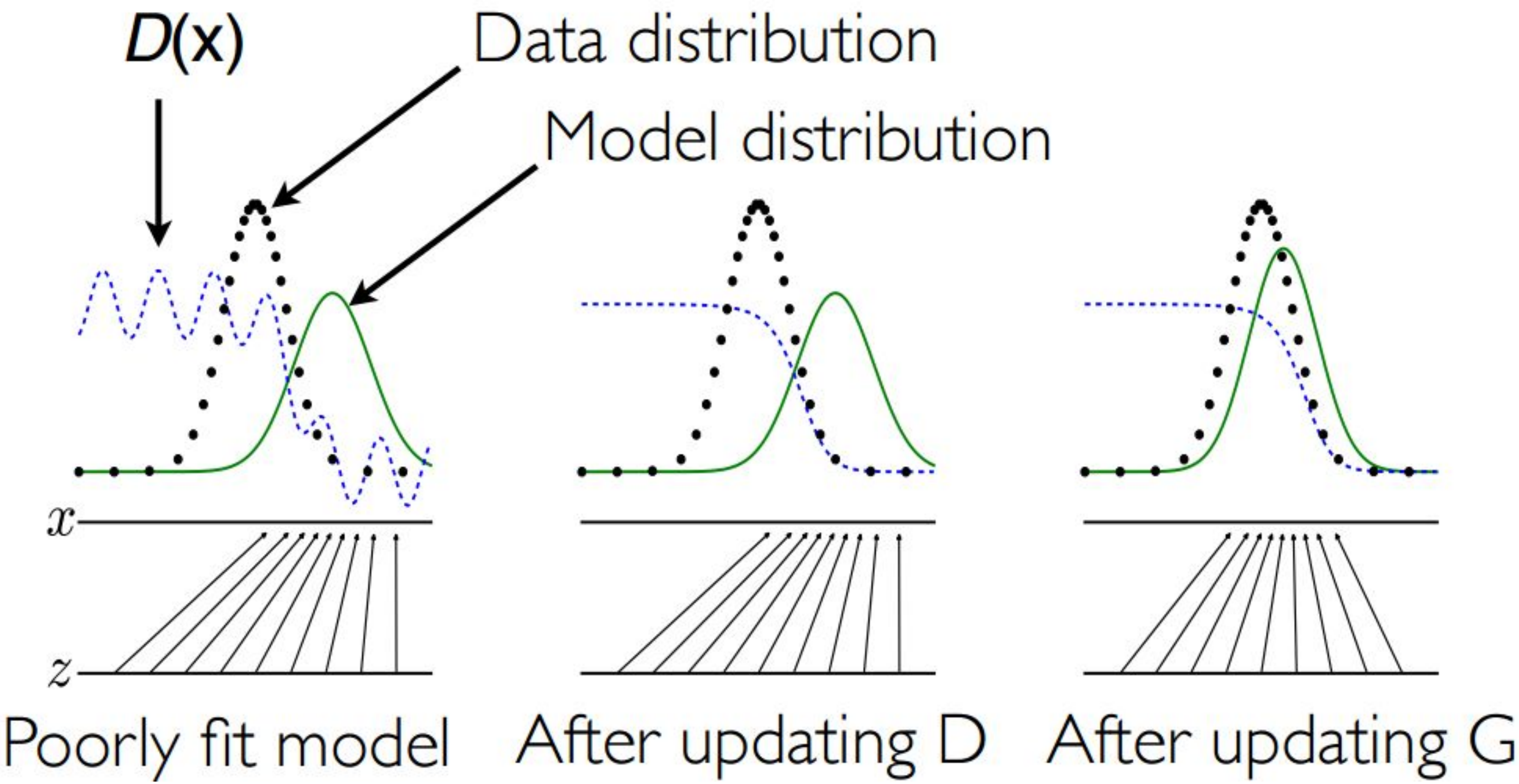


Poorly fit model

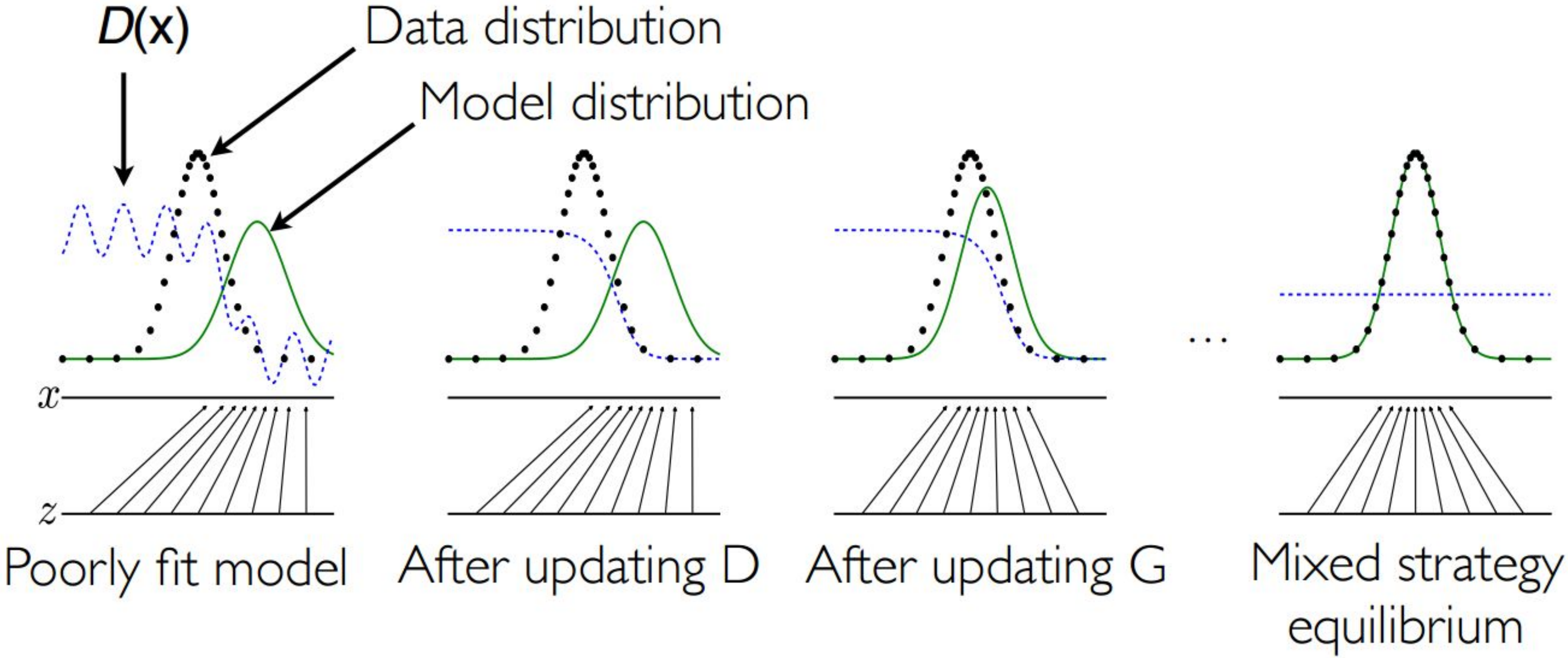
Learning Process



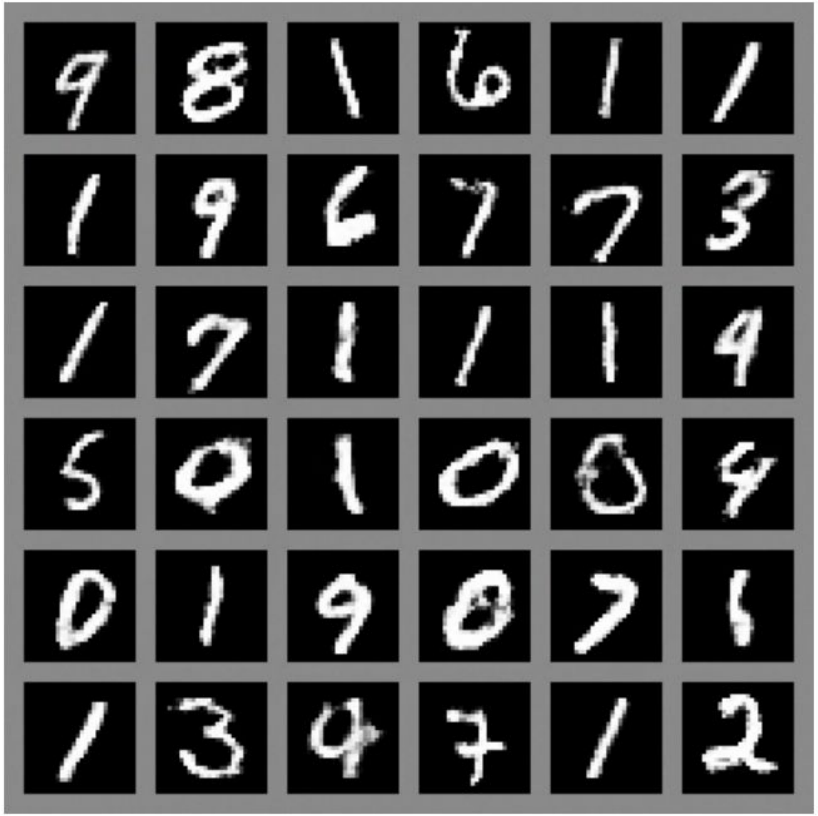
Learning Process



Learning Process



Generated Samples



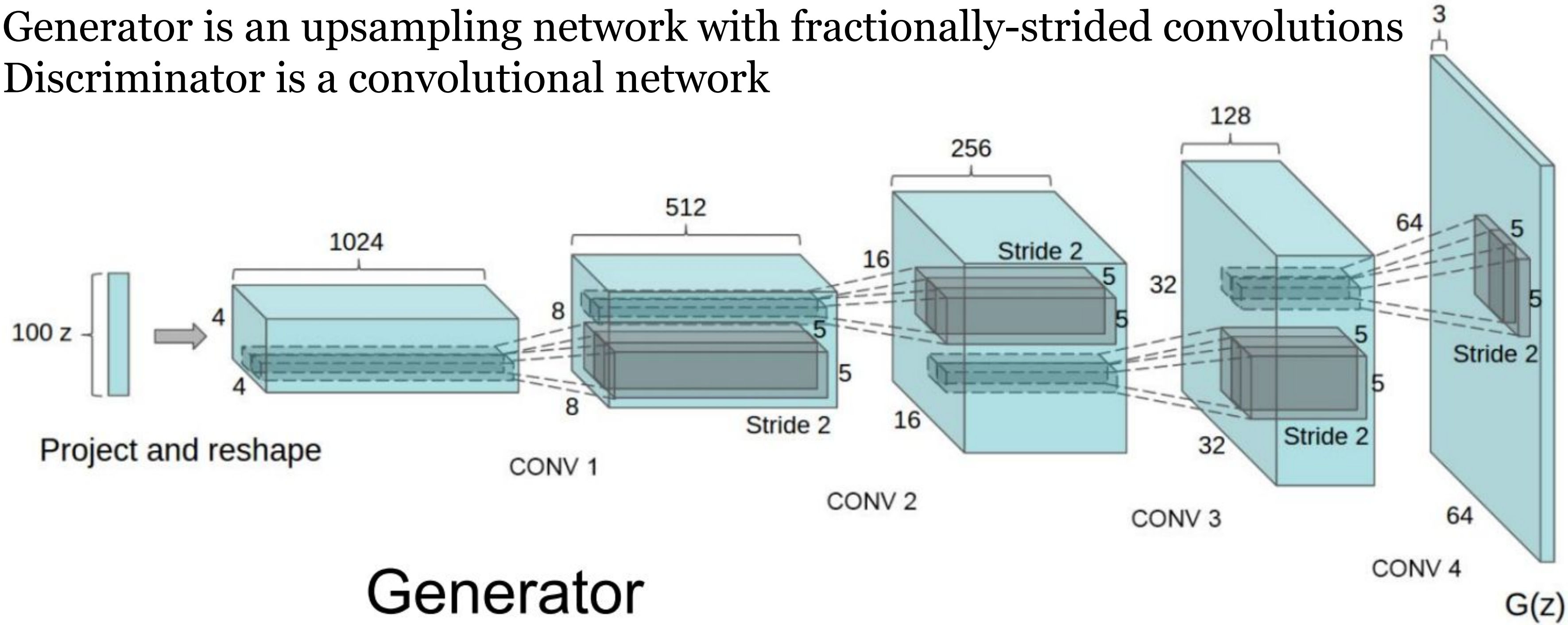
MNIST digit dataset

Toronto Face Dataset (TFD)

Generative Adversarial Nets: Convolutional Architectures

Radford et al., "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks", ICLR 2016

Generator is an upsampling network with fractionally-strided convolutions
Discriminator is a convolutional network

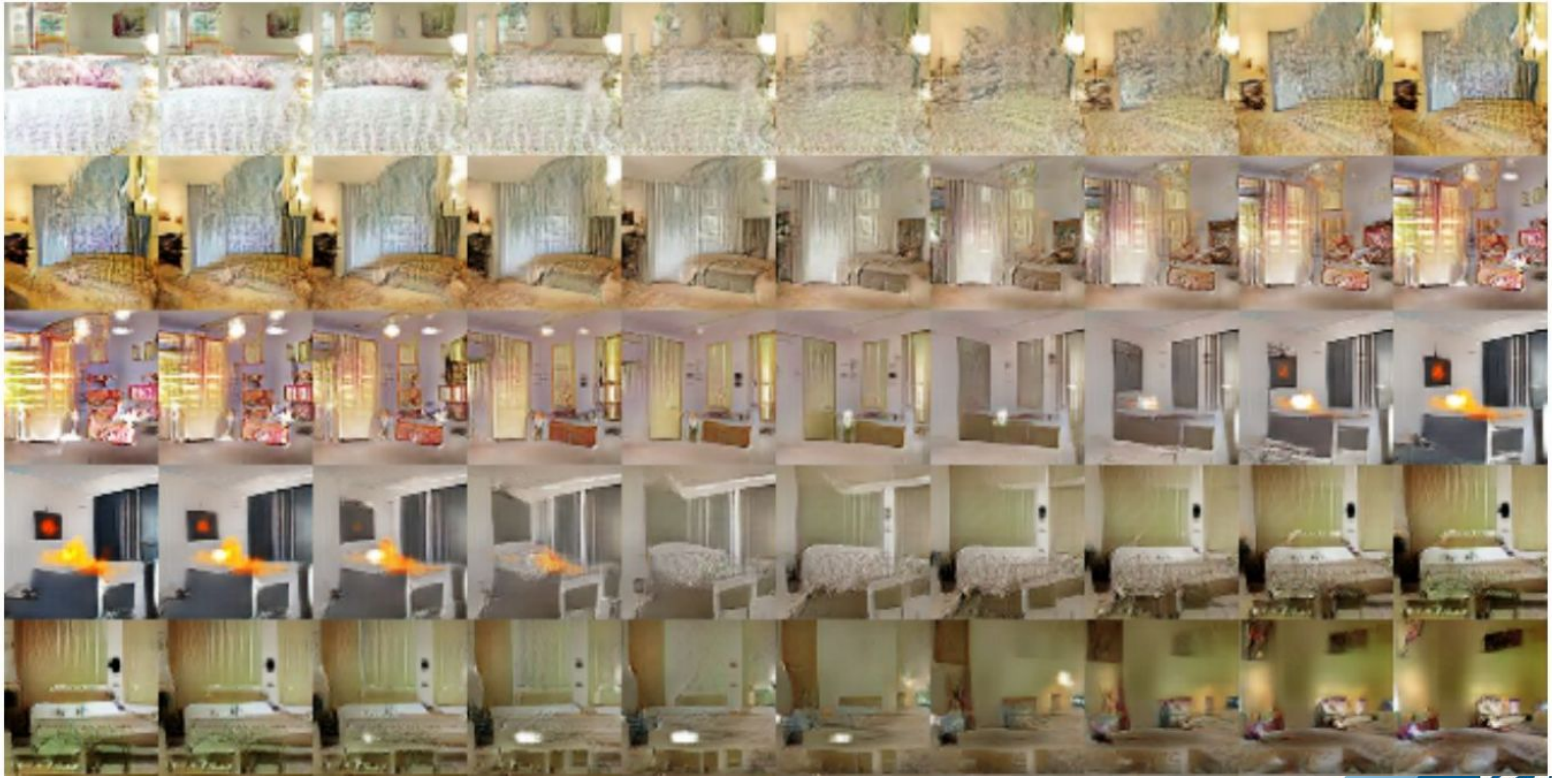


Generative Adversarial Nets: Convolutional Architectures

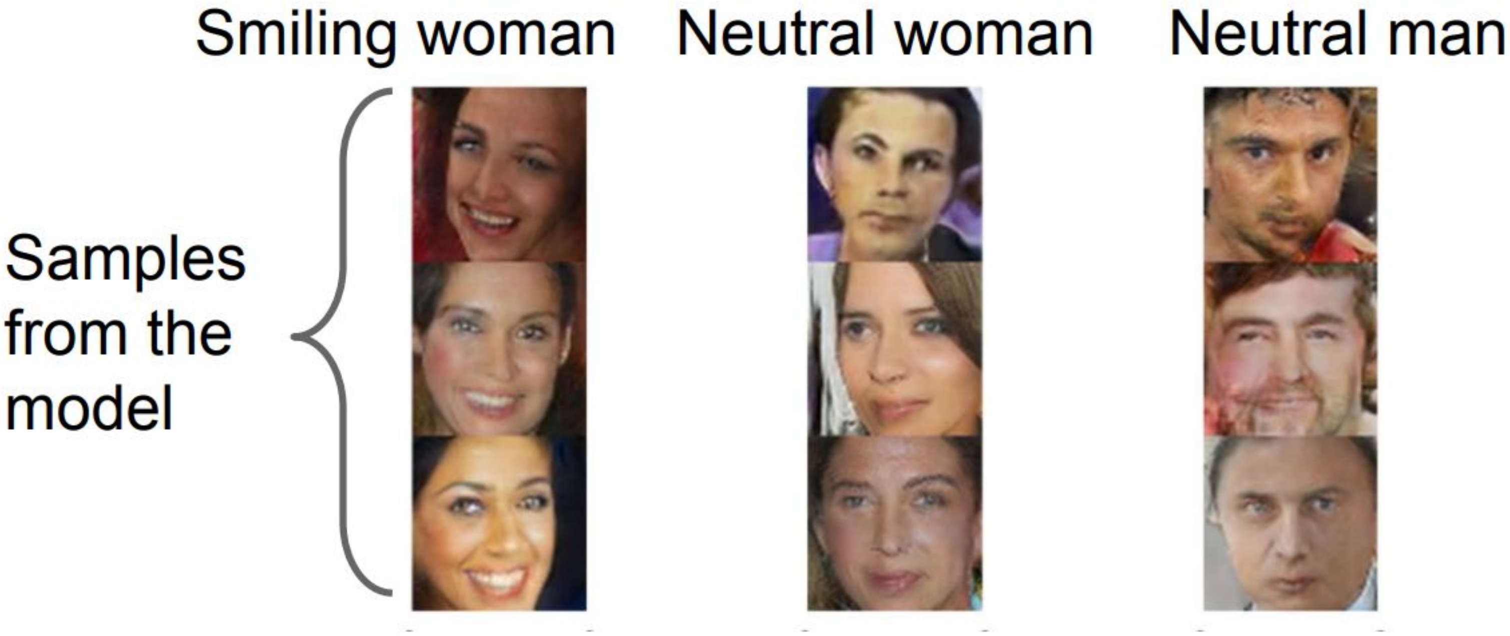
The generated samples looks amazing!

Generative Adversarial Nets: Convolutional Architectures

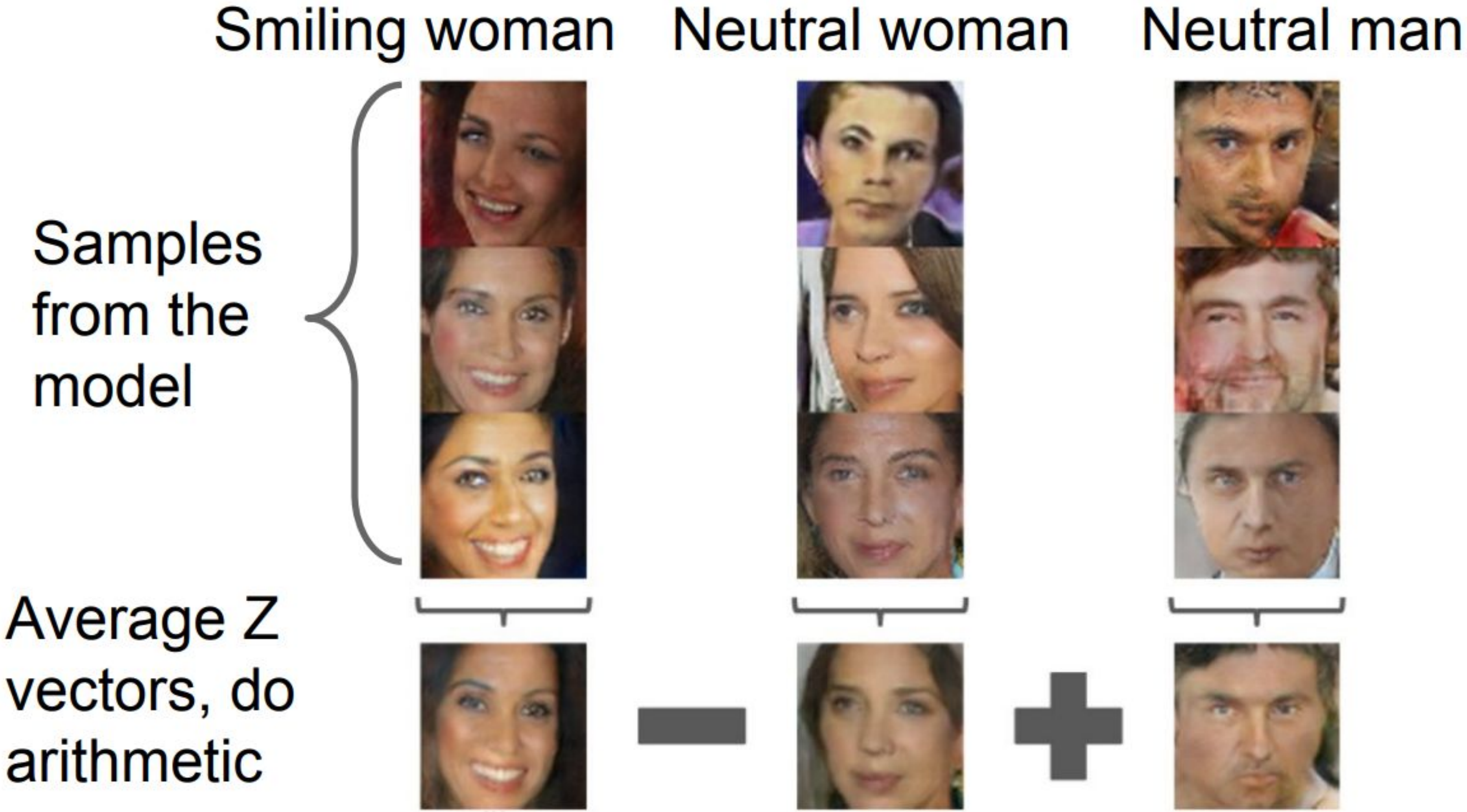
Latent space interpolation



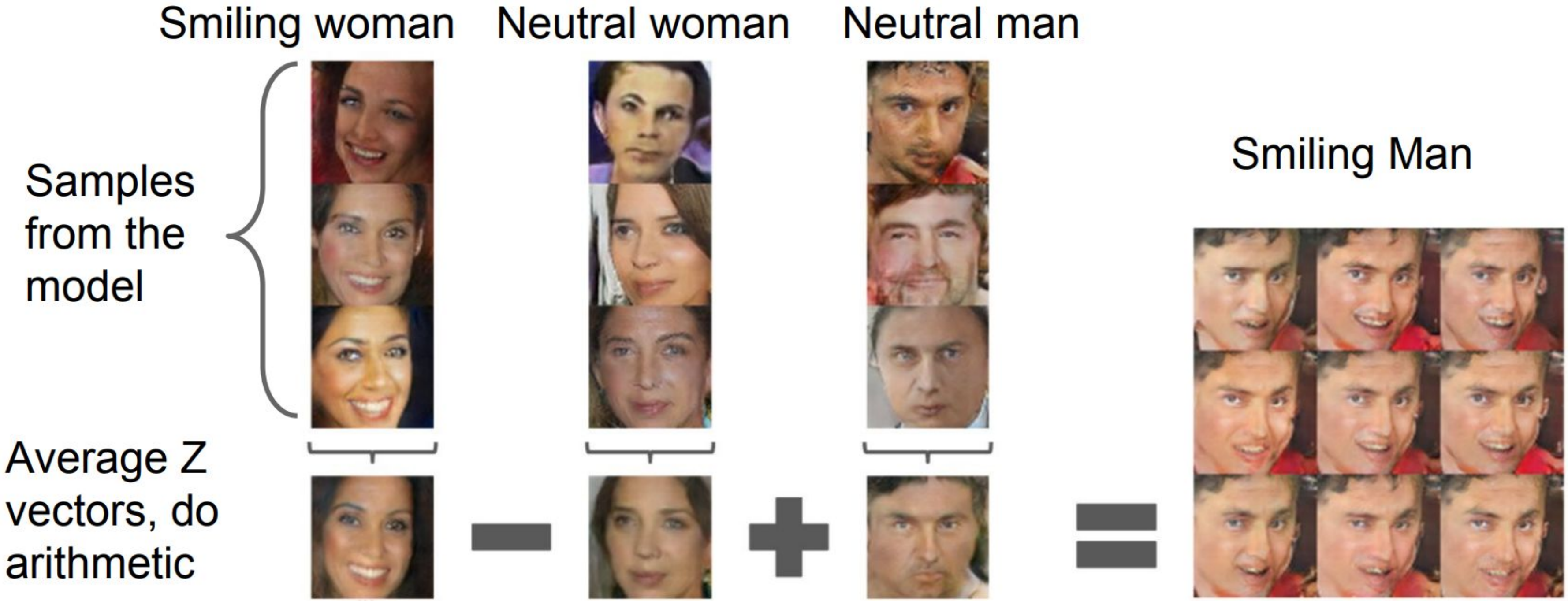
Generative Adversarial Nets: Interpretable Vector Math



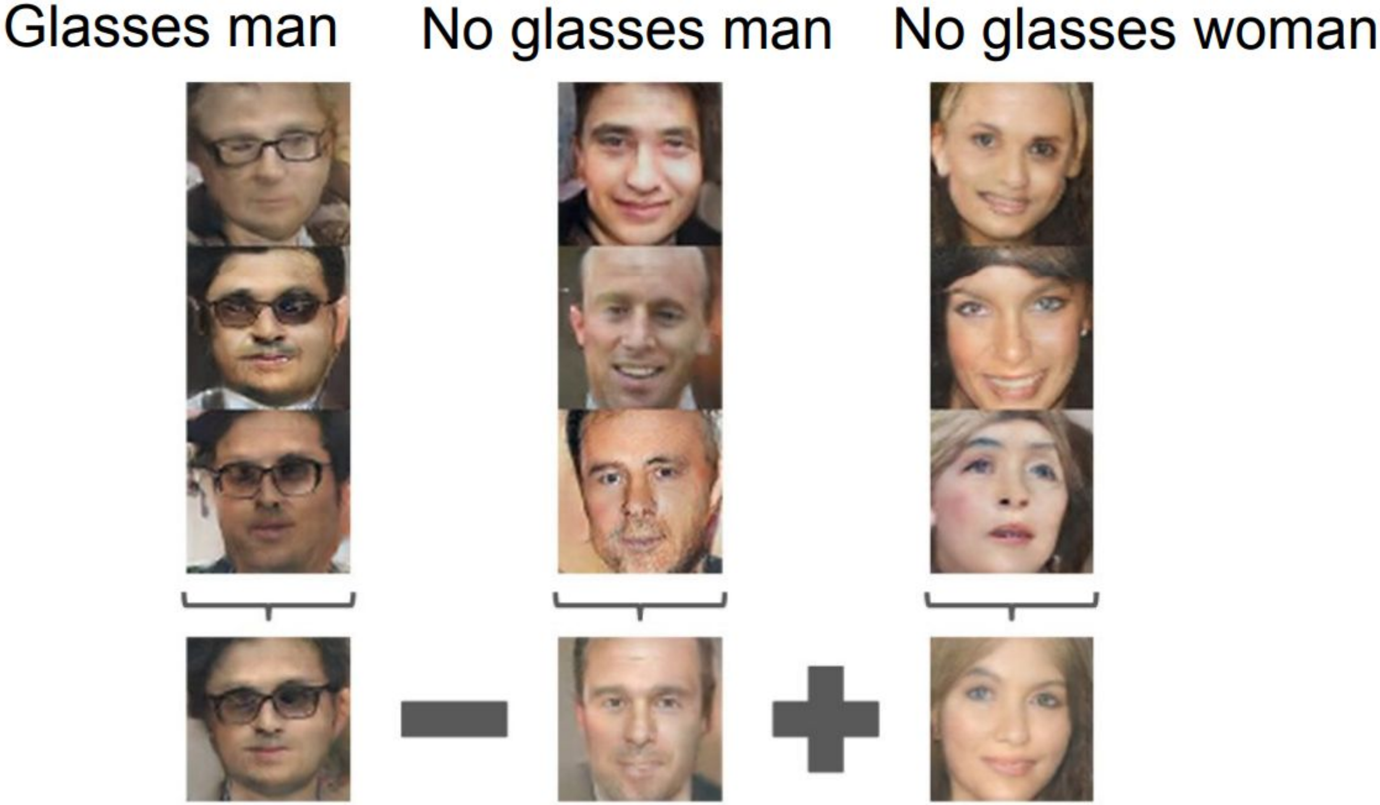
Generative Adversarial Nets: Interpretable Vector Math



Generative Adversarial Nets: Interpretable Vector Math



Generative Adversarial Nets: Interpretable Vector Math



Generative Adversarial Nets: Interpretable Vector Math

Generative Adversarial Nets: Convolutional Architectures

Tero Karras et al., Progressive Growing of GANs for Improved Quality, Stability, and Variation

UMASS BOSTON

References

Generative Adversarial Networks: https://arxiv.org/abs/1406.2661

Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks: https://arxiv.org/pdf/1511.06434.pdf

Wasserstein GAN: https://arxiv.org/pdf/1701.07875.pdf

Progressive Growing of GANs for Improved Quality, Stability, and Variation: https://arxiv.org/pdf/1710.10196.pdf

Thank You!