Generative Adversarial Networks(GANs)

Satish Kumar

November 3, 2017

LEAP Lab, IISC Bangalor

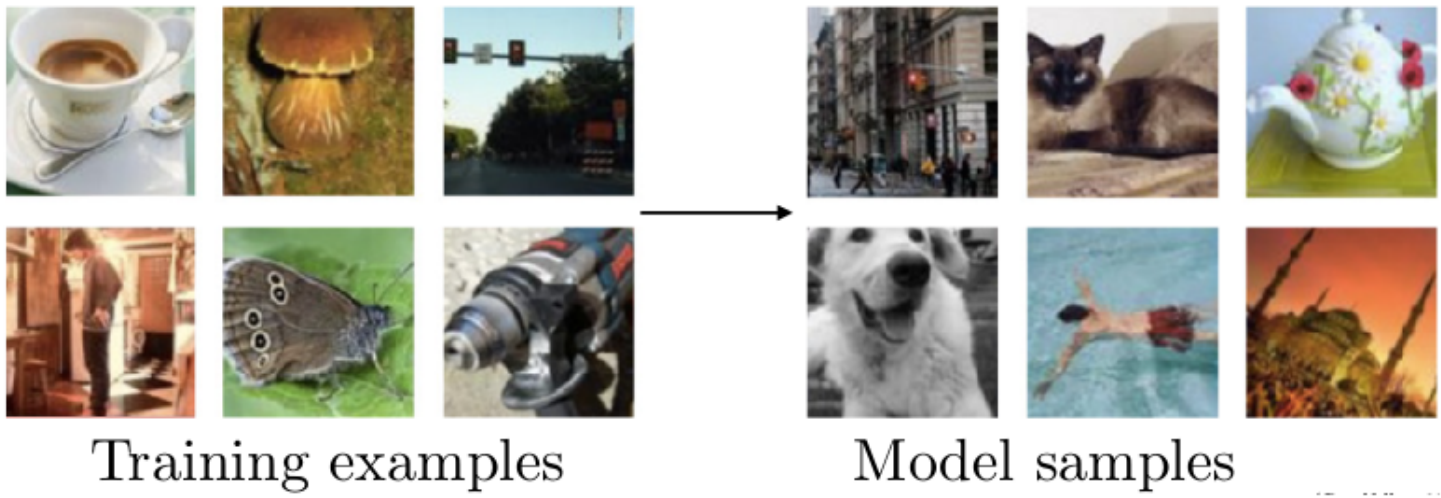
Table of contents

Generative modeling

Density Estimation

Figure 1: Some generative models perform density estimation. These models take a training set of examples drawn from an unknown data-generating distribution p_{data} and return an estimate of that distribution. The estimate p_{model} can be evaluated for a particular value of $\boldsymbol{x}$ to obtain an estimate $p_{\text{model}}(\boldsymbol{x})$ of the true density $p_{\text{model}}(\boldsymbol{x})$. This figure illustrates the process for a collection of samples of one-dimensional data and a Gaussian model.

An Ideal Generative model

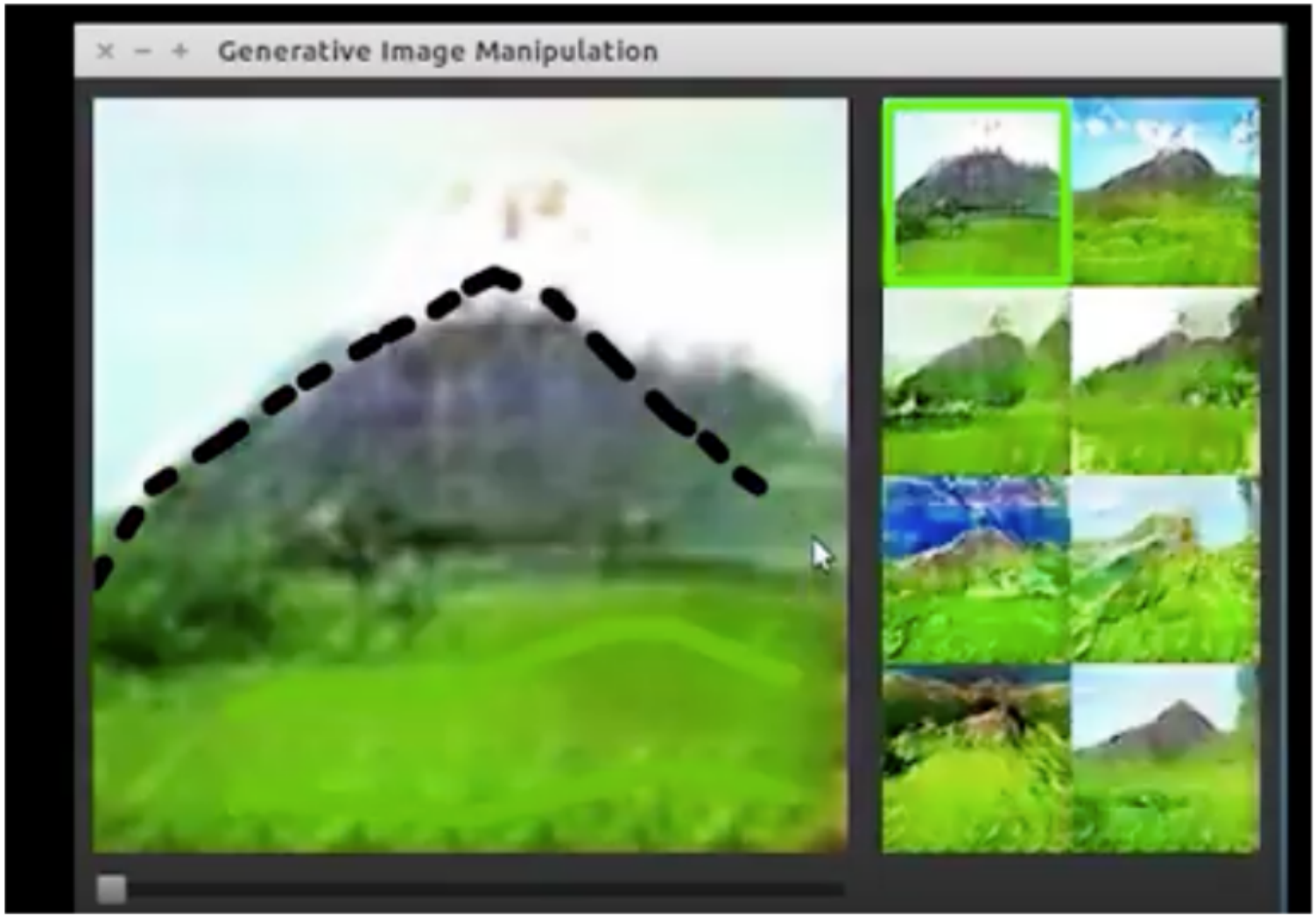


Why study Generative modeling?

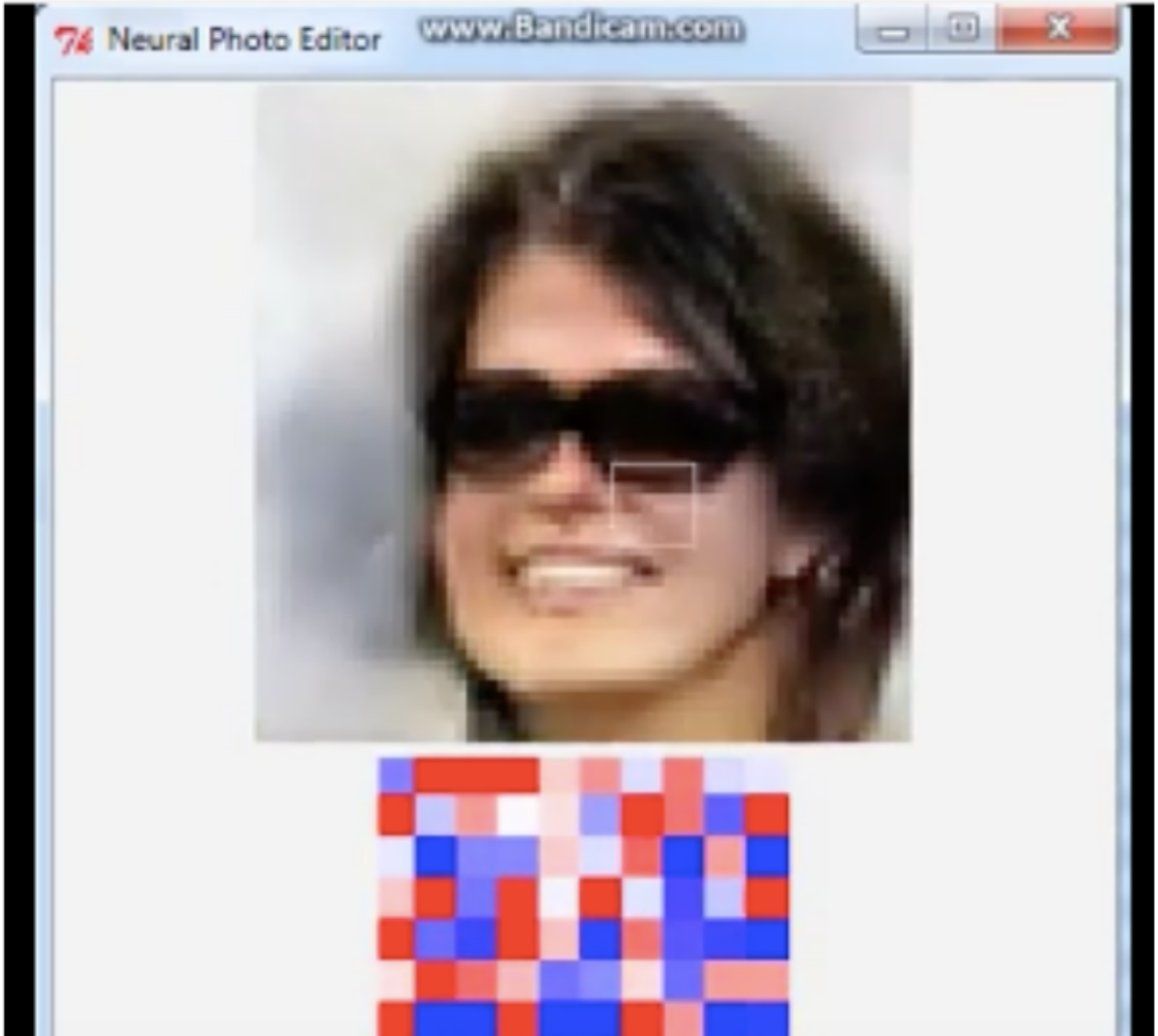
- Training and sampling from generative models is an excellent test of our ability to represent and manipulate high-dimensional probability distributions.
- Generative models can be trained with missing data and can provide predictions on inputs that are missing data.
- It enable machine learning to work with multi-modal outputs
- Realistic generation of samples from some distribution.

Examples of sample generation-1

Examples of sample generation-2



Examples of sample generation-3



Examples of sample generation-4

How generative models work

Maximum Lieklihood

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{m} p_{\text{model}}\left(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right) \tag{1}$$

$$= \arg\max_{\boldsymbol{\theta}} \log \prod_{i=1}^{m} p_{\text{model}}\left(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right) \tag{2}$$

$$= \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{m} \log p_{\text{model}}\left(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right). \tag{3}$$

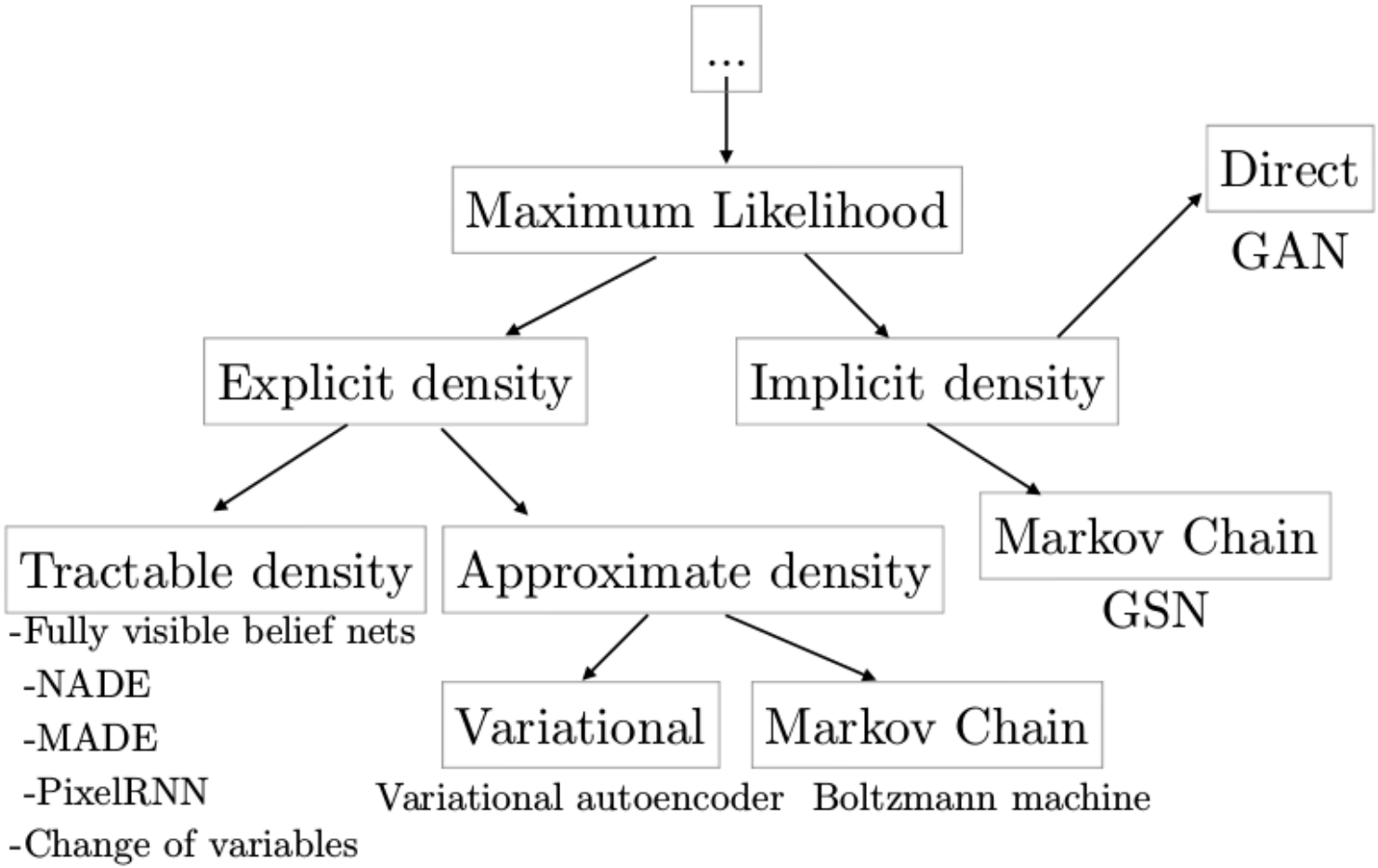
We can also think of maximum likelihood estimation as minimizing the **KL divergence** between the data generating distribution and the model:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} D_{\text{KL}}\left(p_{\text{data}}(\boldsymbol{x}) \| p_{\text{model}}(\boldsymbol{x}; \boldsymbol{\theta})\right). \tag{4}$$

Maximum Lieklihood

- We restrict our attention to deep generative models that work by maximizing the likelihood

A taxonomy of deep generative models



Explicit density models

- We simply plug the models definition of the density function into the expression for the likelihood, and follow the gradient uphill.
- Difficulty in explicit density models is designing a model that can capture all of the complexity of the data to be generated while still maintaining computational tractability
- Strategies to confront this challenge:
- 1. Careful construction of models whose structure guarantees their tractability
- 2. Models that admit tractable approximations to the likelihood and its gradients

Tractable explicit models

- Models that define an explicit density function that is computationally tractable
- Example:Fully visible belief networks

Fully visible belief networks

FVBNs are models that use the chain rule of probability to decompose a probability distribution over an n-dimensional vector x into a product of one-dimensional probability distributions:

$$p_{\text{model}}(\boldsymbol{x}) = \prod_{i=1}^{n} p_{\text{model}}\left(x_i | x_1, \ldots, x_{i-1}\right)$$

- FVBNs are one of the three most popular approaches to generative modeling, alongside GANs and variational autoencoders.
- Used in DeepMind and WaveNet
- Drawback of FVBNs is that samples must be generated one entry at a time: first x_1 , then x_2 , etc.,
- WaveNet requires two minutes of computation time to generate one second of audio, and cannot yet be used for interactive conversations.
- GANs were designed to be able to generate all of $\boldsymbol{x}$ in parallel, yielding greater generation speed

Explicit models requiring approximation

- Still uses an explicit density function that is intractable but requires the use of approximations to maximize the likelihood
- Two categories:
- 1. Using deterministic approximations,Exp:Variational approximations
- 2. Using stochastic approximations,Exp:Markov chain approximations

Variational approximations

- Variational methods define a lower bound

$$\mathcal{L}(\boldsymbol{x}; \boldsymbol{\theta}) \leq \log p_{\text{model}}(\boldsymbol{x}; \boldsymbol{\theta}).$$

- It is possible to define an L that is computationally tractable even when the log-likelihood is not
- Example: Variational autoencoder
- The gap between $\mathcal{L}$ and the true likelihood can result in p_{model} learning something other than the true p_{data}
- In practice, variational methods often obtain very good likelihood, but are regarded as producing lower quality samples
- GANs are generally regarded as producing better samples(Subjective opinion)
- Compared to FVBNs, VAEs are regarded as more difficult to optimize, but GANs are not an improvement in this respect

Markov chain approximations

- Some models require the generation of more expensive samples, using Markov chain techniques
- A Markov chain is a process for generating samples by repeatedly drawing a sample $\boldsymbol{x}' \sim q(\boldsymbol{x}'|\boldsymbol{x})$
- By repeatedly updating $\boldsymbol{x}$ according to the transition operator q, Markov chain methods can sometimes guarantee that $\boldsymbol{x}$ will eventually converge to a sample from $p_{\text{model}}(\boldsymbol{x})$
- This convergence can be very slow
- Example:Boltzmann Machines

Implicit density models

- Some models can be trained without needing to explicitly define a density functions
- These models instead offer a way to train the model while interacting only indirectly with p_{model}, usually by sampling from it
- Example:Generative Stochastic Network
- The family of implicit models that can generate a sample in a single step
- Example:GANs

GANs

GANs Framework

- The basic idea of GANs is to set up a game between two players G and D
- The generator creates samples that are intended to come from the same distribution as the training data
- The discriminator examines samples to determine whether they are real or fake
- Counterfeiter and Police analogy

GANs Framework

- Formally, GANs are a structured probabilistic model containing latent variables z and observed variables x

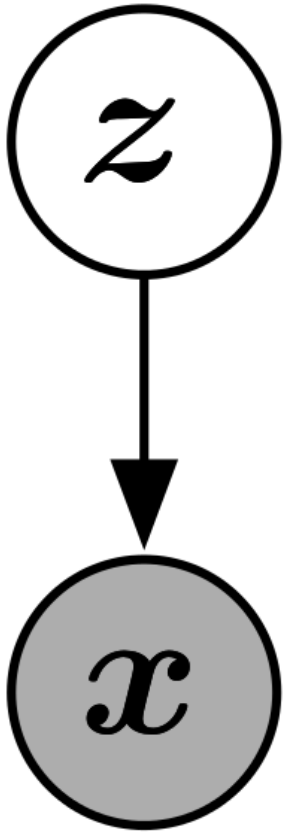
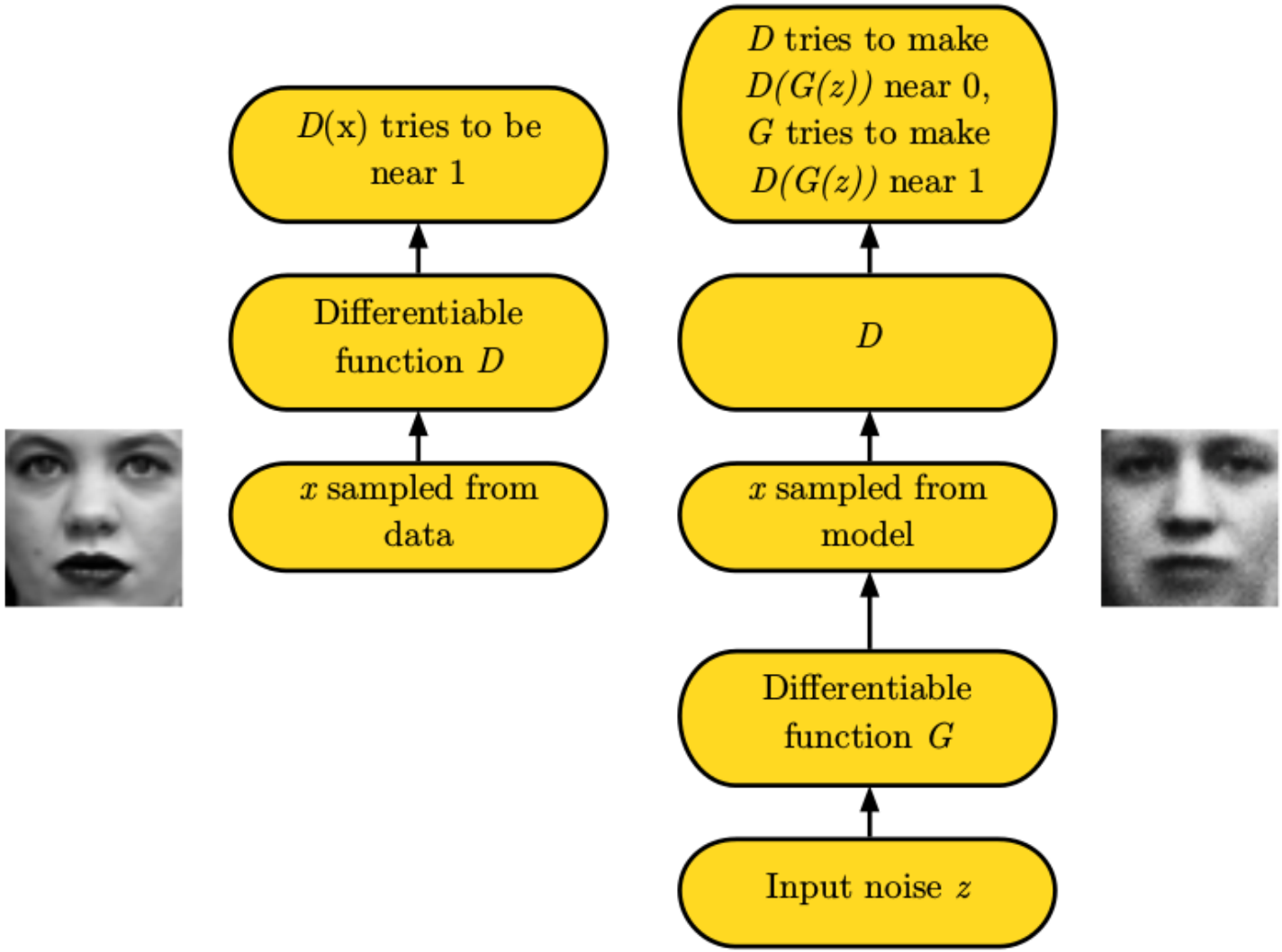


Figure 13: The graphical model structure of GANs, which is also shared with VAEs, sparse coding, etc. It is directed graphical model where every latent variable influences every observed variable. Some GAN variants remove some of these connections.

-

GANs Framework Continued.....



The Generator

- The generator is simply a differentiable function G
- Typically, a deep neural network is used to represent G
- Inputs may be provided at any point throughout the network

The Cost Function

- The cost used for the discriminator is

$$J^{(D)}(\boldsymbol{\theta}^{(D)}, \boldsymbol{\theta}^{(G)}) = -\frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{data}} \log D(\boldsymbol{x}) - \frac{1}{2}\mathbb{E}_{z} \log\left(1 - D(G(z))\right)$$

- And our goal is to minimize this
- A standard cross-entropy cost that is minimized when training a standard binary classifier with a sigmoid output
- The only difference is that the classifier is trained on two minibatches of data:
- – one coming from the dataset,where the label is 1 for all examples
- – and one coming from the generator, where the label is 0 for all examples
- All versions of the GAN game encourage the discriminator to minimize equation

The Nash equilibrium

- The Nash equilibrium is one of the foundational concepts in game theory
- Informally, a strategy profile is a Nash equilibrium if no player can do better by unilaterally changing his or her strategy.

The Discriminator

- Optimal strategy for any $p_{\text{model}}(x)$ is always

$$D(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_{\text{model}}(\boldsymbol{x})}$$

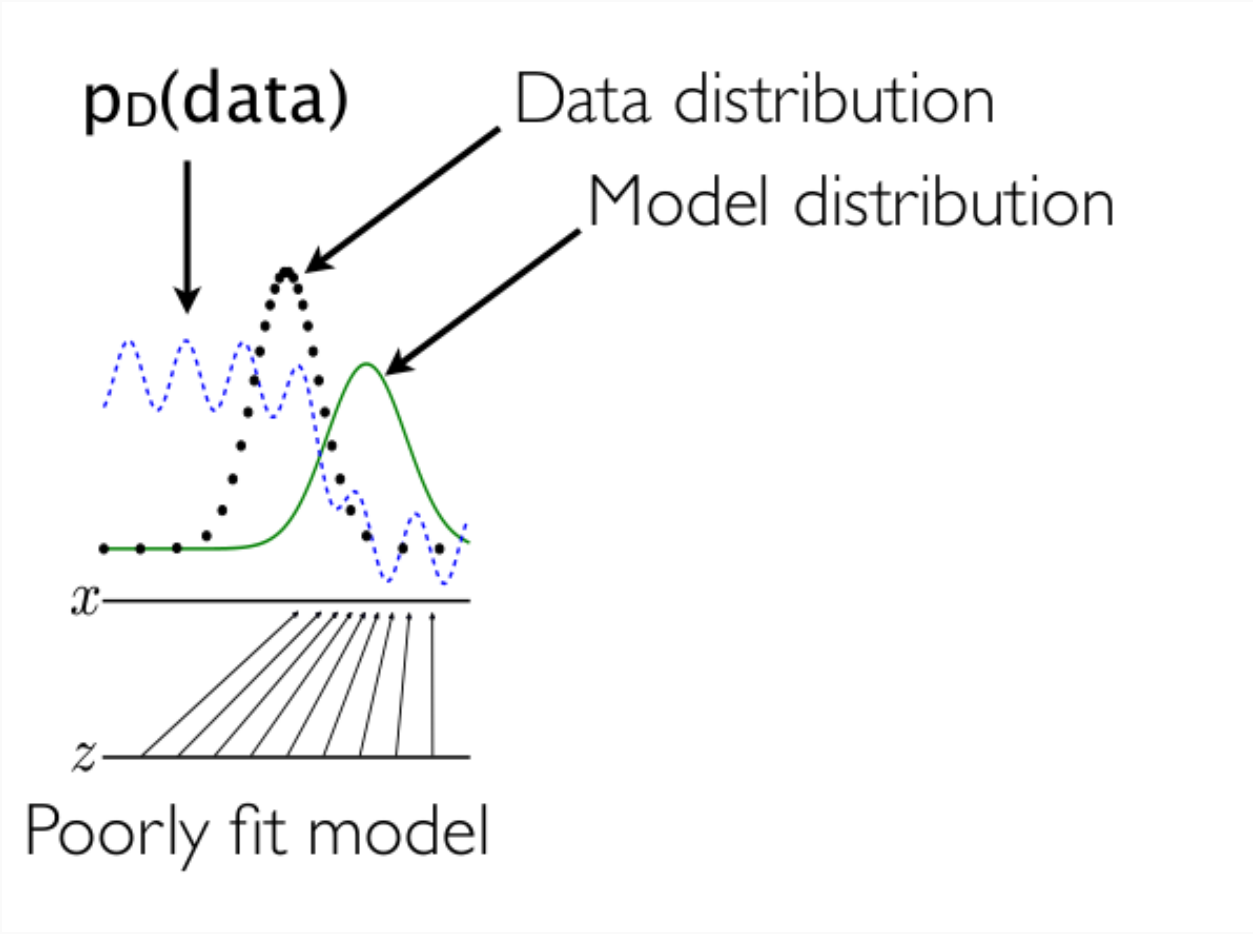
- In all cases, the discriminator has the same optimal strategy
- Other deep generative models make approximations based on lower bounds or Markov chains;
- GANs make approximations based on using supervised learning to estimate a ratio of two densities
- The GAN approximation is subject to the failures of supervised learning: overfitting and underfitting
- With perfect optimization and enough training data, these failures can be overcome

Training Process

- The training process consists of simultaneous SGD
- On each step, two minibatches are sampled:
  1. A minibatch of x values from the dataset and
  2. A minibatch of z values drawn from the model s prior over latent variables
- Then two gradient steps are made simultaneously
- In both cases, it is possible to use the gradient-based optimization algorithm of your choice(Adam)

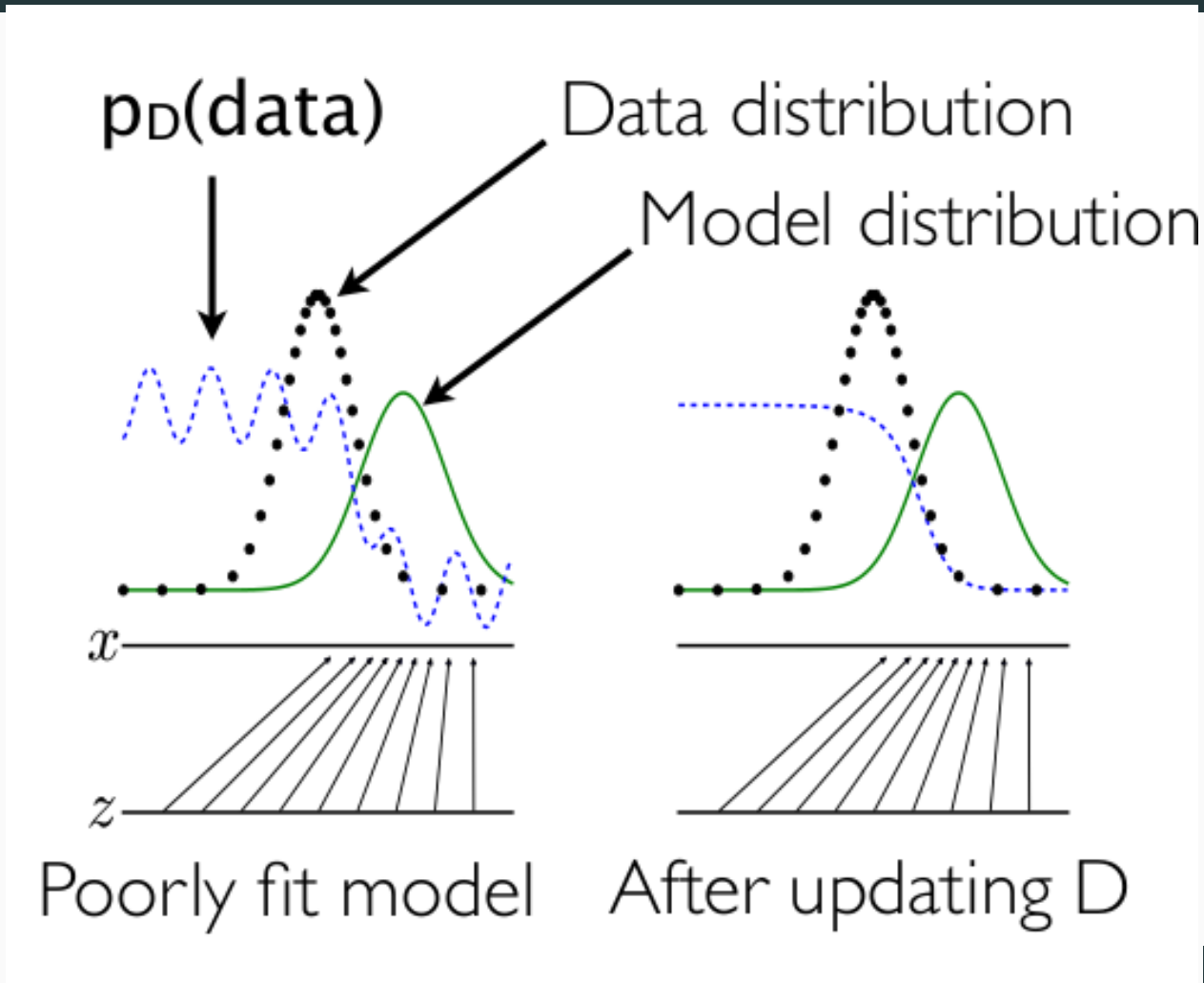
Training Process Continued...

- An illustration of how the discriminator estimates a ratio of densities



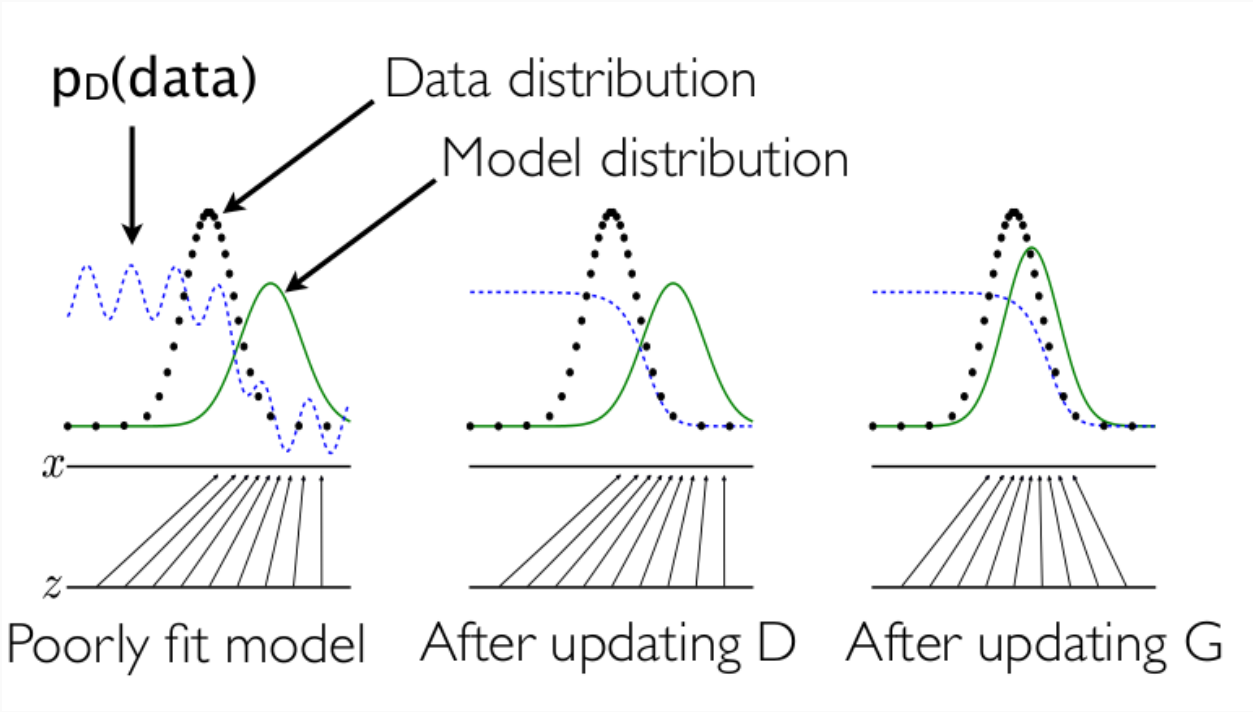
Consider an adversarial pair near convergence: Model Distribution is similar to Data Distribution and D is a partially accurate classifier

Training Process Continued....



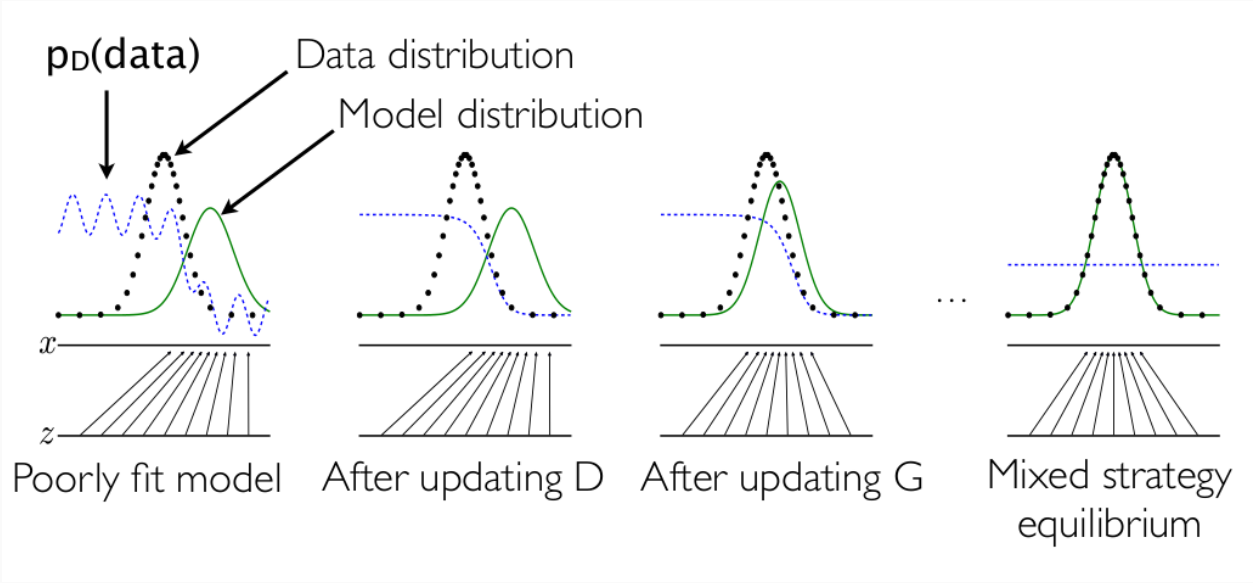
In the inner loop of the algorithm D is trained to discriminate samples from data, converging to D(x)

Training Process Continued.....



After an update to G, gradient of D has guided G(z) to flow to regions that are more likely to be classified as data

Training Process Continued.....



After several steps of training, if G and D have enough capacity, they will reach a point at which both cannot improve

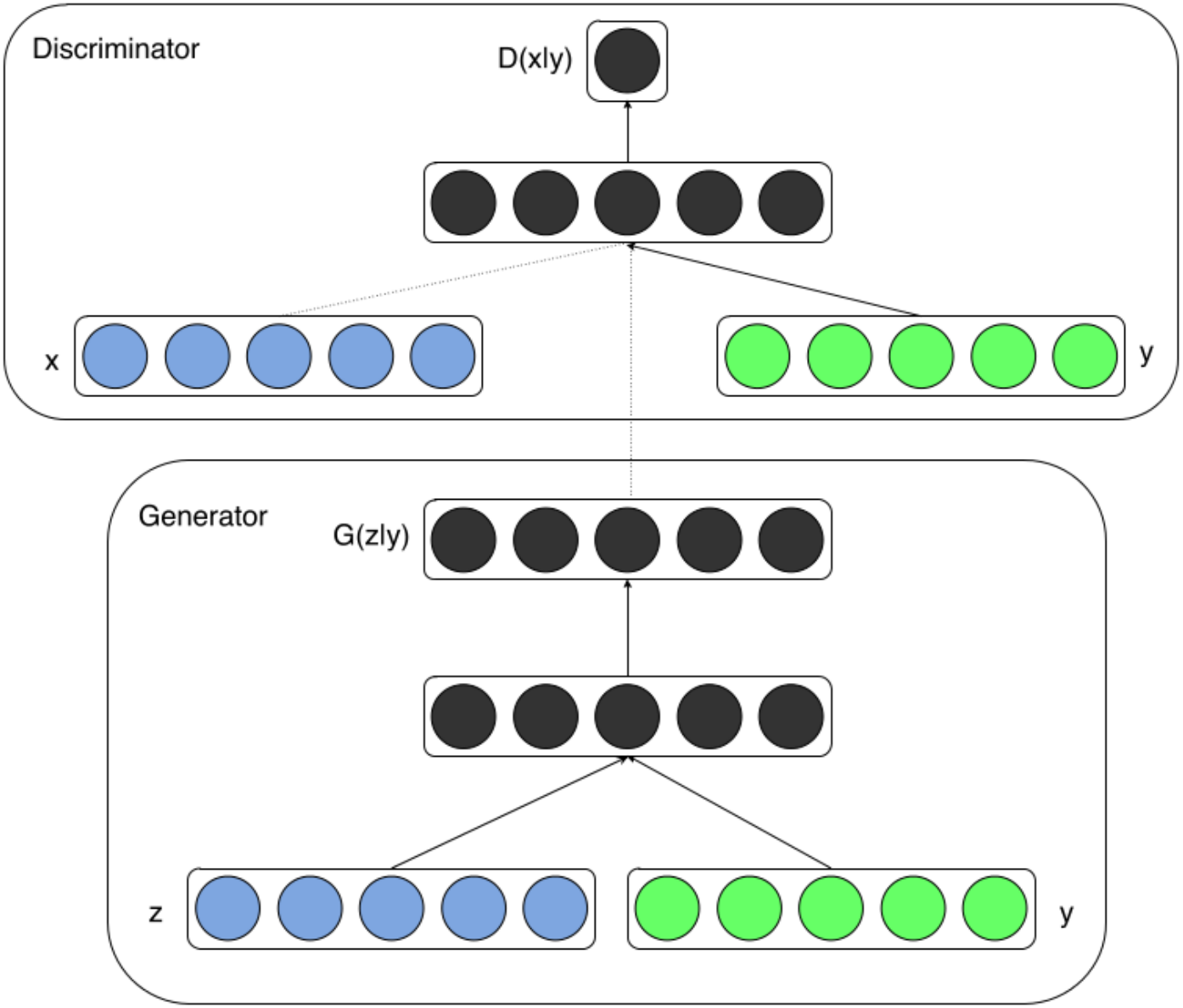
Conditional GANs

cGANs

- Generative adversarial nets can be extended to a conditional model if both the generator and discriminator are conditioned on some extra information y.

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

cGANs



cGANs Continued....

- Learns $p(x|y)$
- Discriminator is trained on (x,y) pairs
- Generator net gets y and z as input
- Useful for: Translation, speech synth, image segmentation

How GANs compare to other generative models

Comparing GANs to other models

- GANs can generate samples in parallel, instead of using runtime proportional to the dimensionality of x. This is an advantage relative to FVBNs.
- No Markov chains are needed. This is an advantage relative to Boltzmann machines and GSNs.
- No variational bound is needed, and specific model families usable within the GAN framework are already known to be universal approximators, so GANs are already known to be asymptotically consistent.
- Some VAEs are conjectured to be asymptotically consistent, but this is not yet proven.

Comparing GANs to other models

- The design of the generator function has very few restrictions. This is an advantage relative to Boltzmann machines, for which few probability distributions admit tractable Markov chain sampling
- GANs are subjectively regarded as producing better samples than other methods.

Disadvantages of GANs

- Training GANs requires finding the Nash equilibrium of a game, which is a more difficult problem than optimizing an objective function.

Tips and tricks to make GANs work

Tips continued.....

- Normalize between -1 and 1
- Use Tanh as the last layer of the generator output
- In GAN papers, the loss function to optimize G is $min(log(1 - D))$, but in practice folks practically use $max(log(D))$ because the first formulation has vanishing gradients early on
- Flip labels when training generator: real = fake, fake = real

Tips continued.....

- The stability of the GAN game suffers if you have sparse gradients
- So use LeakyReLU = good (in both G and D)
- For Downsampling, use: Average Pooling, Conv2d + stride
- For Upsampling, use: PixelShuffle, ConvTranspose2d + stride

Tips continued.....

- Label Smoothing, i.e. if you have two target labels: Real=1 and Fake=0, then for each incoming sample, if it is real, then replace the label with a random number between 0.7 and 1.2, and if it is a fake sample, replace it with 0.0 and 0.3 (for example).
- Make the labels the noisy for the discriminator: occasionally flip the labels when training the discriminator
- Use SGD for discriminator and ADAM for generator

Tips continued.....

- Add some artificial noise to inputs to D (Arjovsky et. al., Huszar, 2016)
- Add gaussian noise to every layer of generator (Zhao et. al. EBGAN)
- Use Dropouts in G in both train and test phase

Tips continued.....(Track failures early)

- D loss goes to 0: failure mode
- If loss of generator steadily decreases, then it's fooling D with garbage
- When things are working, D loss has low variance and goes down over time vs having huge variance and spiking
- Check norms of gradients: if they are over 100 things are screwing up

Research frontiers

Non-Convergence

- GANs require finding the equilibrium to a game with two players
- Even if each player successfully moves downhill on that players update, the same update might move the other player uphill.
- Sometimes the two players eventually reach an equilibrium, but in other scenarios they repeatedly undo each others progress without arriving anywhere useful.

Non-Convergence:Mode Collapse

- Most common form of harmful non-convergence encountered in the GAN game is mode collapse
- Mode Collapse is a problem that occurs when the generator learns to map several different input z values to the same output point.

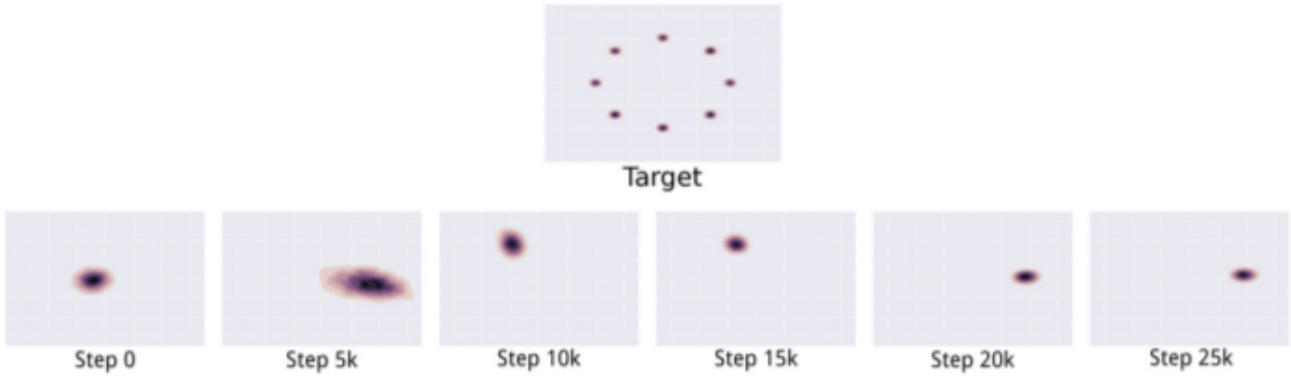


Figure 22: An illustration of the mode collapse problem on a two-dimensional toy dataset. In the top row, we see the target distribution p_{data} that the model should learn. It is a mixture of Gaussians in a two-dimensional space. In the lower row, we see a series of different distributions learned over time as the GAN is trained. Rather than converging to a distribution containing all of the modes in the training set, the generator only ever produces a single mode at a time, cycling between different modes as the discriminator learns to reject each one. Images from Metz *et al.* (2016).

Non-Convergence:Mode Collapse.....

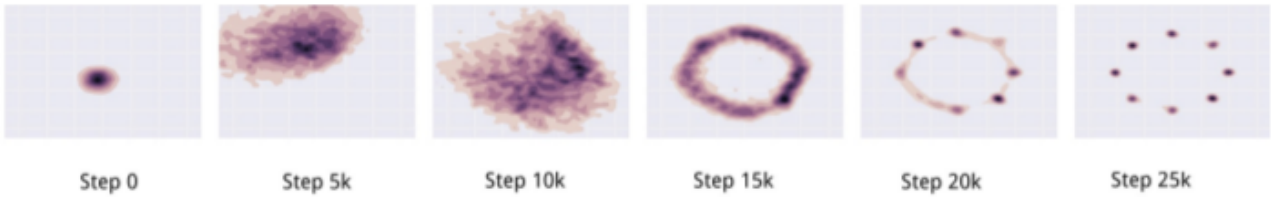


Figure 32: Unrolled GANs are able to fit all of the modes of a mixture of Gaussians in a two-dimensional space. Image reproduced from Metz *et al.* (2016).

Non-Convergence:Mode Collapse.....

Figure 24: GANs have low output diversity for text-to-image tasks because of the mode collapse problem. Image reproduced from Reed *et al.* (2016a).

Non-Convergence:Mode Collapse.....

Figure 25: StackGANs are able to achieve higher output diversity than other GAN-based text-to-image models. Image reproduced from Zhang *et al.* (2016).

Evaluation of generative models

- It is not clear how to quantitatively evaluate generative models.
- Models that obtain good likelihood can generate bad samples, and models that generate good samples can have poor likelihood
- It can be difficult to estimate the likelihood for GANs(So Harder to train than other Generative models)

Discrete outputs

- G must be differentiable
- Unfortunately, this means that the generator cannot produce discrete data, such as one-hot word or character representations
- Removing this limitation is an important research direction that could unlock the potential of GANs for NLP

Semi-supervised learning

- Feature matching GANs (Salimans et al., 2016)

Using the code

- GANs learn a representation z of the image x
- This representation can capture useful high-level abstract semantic properties of x
- One obstacle to using z is that it can be difficult to obtain z given an input x

Conclusion

- GANs are generative models that use supervised learning to approximate an intractable cost function
- Same as Boltzmann machines use Markov chains to approximate their cost and VAEs use the variational lower bound to approximate their cost
- GANs can use this supervised ratio estimation technique to approximate many cost functions,including the KL divergence used for maximum likelihood estimation
- GANs are relatively new and still require some research to reach their new potential

Thanks you.

Questions?

References

- Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. Deep learning. MIT press, 2016.
- Goodfellow, Ian. "NIPS 2016 tutorial: Generative adversarial networks." arXiv preprint arXiv:1701.00160 (2016).
- Mirza, Mehdi, and Simon Osindero. "Conditional generative adversarial nets." arXiv preprint arXiv:1411.1784 (2014).
- Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." arXiv preprint arXiv:1511.06434 (2015).
- Gauthier, Jon. "Conditional generative adversarial nets for convolutional face generation." Class Project for Stanford CS231N: Convolutional Neural Networks for Visual Recognition, Winter semester 2014.5 (2014): 2.