Generative Models

Sunil Srinivasa

Guess the celebrities.

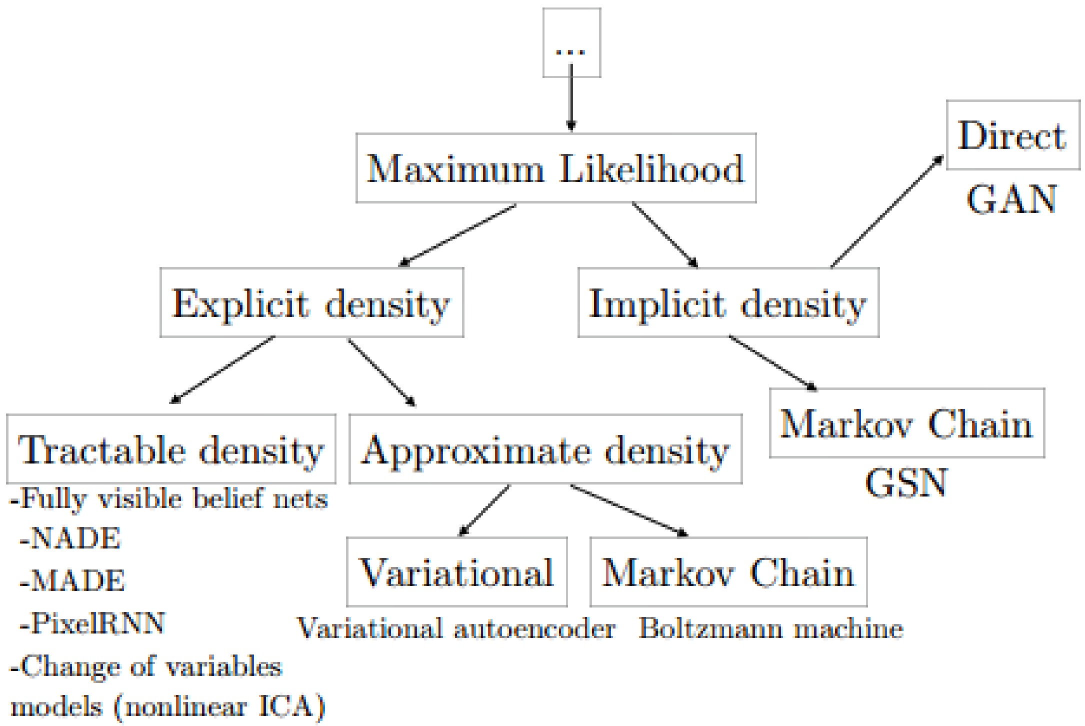
TOP ROW: GENERATED IMAGES
BOTTOM ROW: REAL CELEBRITY IMAGES

Nvidia paper: Progressive growing of GANs submitted to ICLR 2018 - https://arxiv.org/abs/1710.10196

The context of generative models

- **Supervised** Learning Models
  - Given input X and labels y, find a model that maps X to y.
  - E.g., Regression, SVM, CNN, RNN, …
- **Unsupervised** Learning Models
  - Given input X alone (no labels!), find a model that finds some underlying structure in date
  - E.g., Clustering, Principal Component Analysis (PCA), feature leaning, …
- **Semi-supervised** Learning Models
  - Given (a few) inputs X with their labels, generate (many more) samples that are similar to the inputs (similar in terms of the probability distribution).
  - E.g., RBM, P-RNN, GAN, VAE, …

Taxonomy of Deep Generative Networks



All these methods attempt to minimize the divergence between p_{data} and p_{model}.

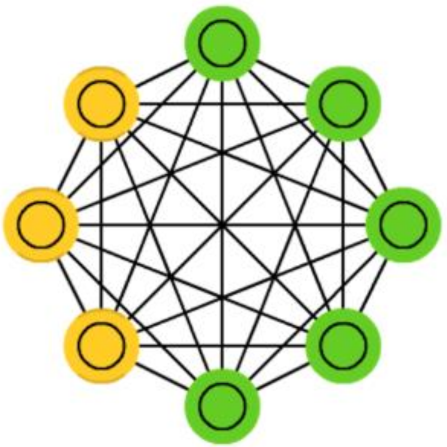
Pic courtesy: Ian Goodfellow's NIPS keynote talk - https://arxiv.org/pdf/1701.00160.pdf

Timeline

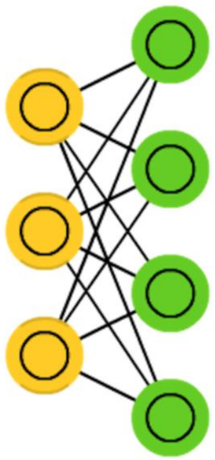
- Gaussian mixture models
- Hidden Markov models
- …
- 1986 – Boltzmann machines (Geoffrey Hinton)
- 2013 – Variational Auto Encoders (Diederik Kingma)
- 2014 – Generative Adversarial Networks (Ian Goodfellow)
- 2016 – Pixel RNNs (Aaron van den Oord)

- VAEs and GANs are the most popular generative models to-date!

Boltzmann Machine

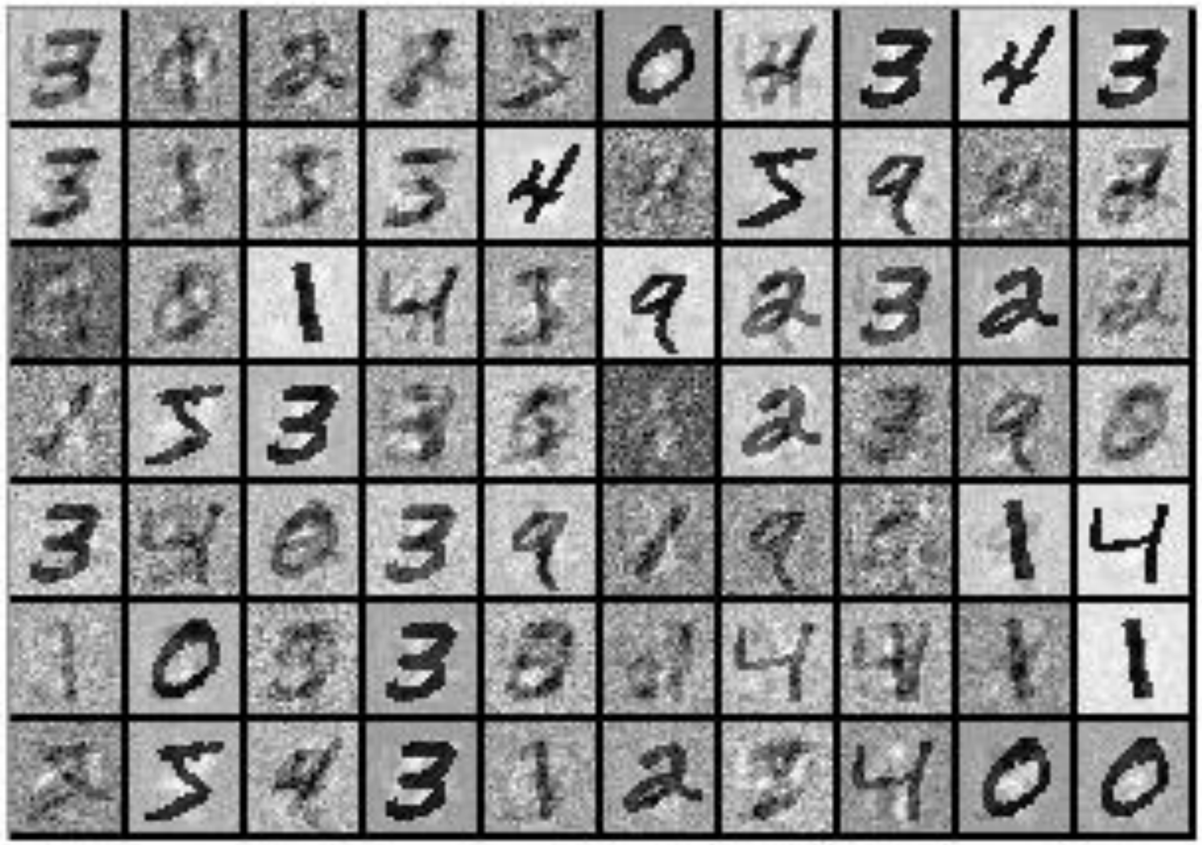


- Back-fed input units (Yellow)
  - Also becomes the output unit after training!
- Probabilistic hidden units (Green)
- Each edge is governed by a trainable weight.
- Training ("Wake-Sleep" algorithm):
  - Given visible states (from inputs), compute the hidden states.
  - Given the computed hidden states, compute the visible states.
  - Repeat until equilibrium is attained.



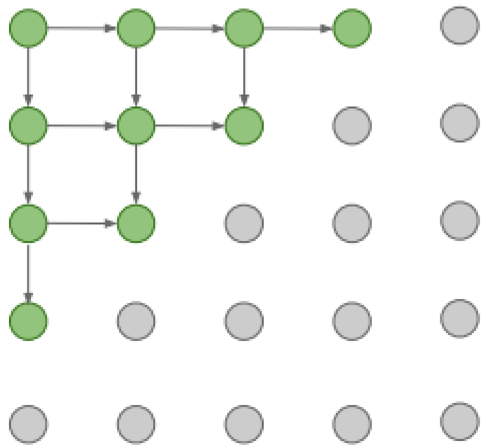
- Restricted Boltzmann machine
  - No intra-layer connection between hidden/visible units
  - Prevents over fitting, faster training
- Tricky to train!

MNIST samples generated by RBMs



Pixel RNN/CNN

- Generate value of a pixel based on neighboring pixels

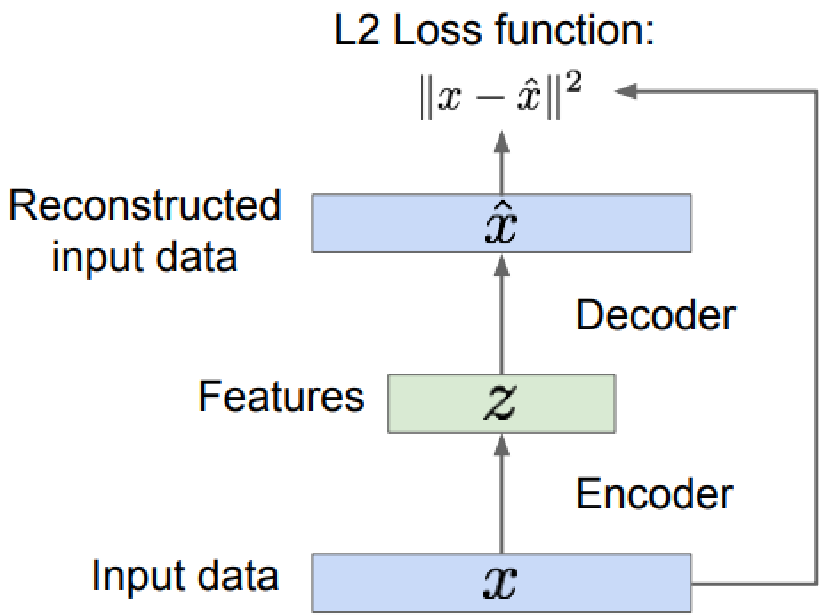


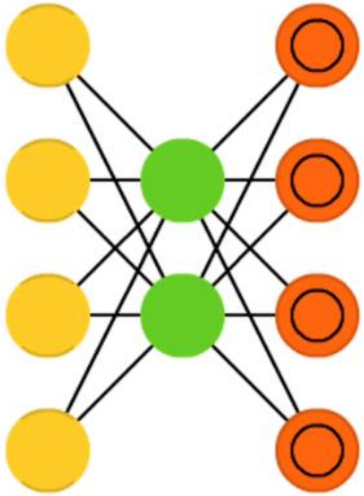
- RNN architecture used to model the joint probability distribution.
- Explicit probability density and good-looking samples.
- Sequential generation is slow!
  - Pixel CNN improves training time (only by a little bit).

Sample images generated by a model trained on ImageNet 32x32 images

Autoencoders

- Attempts to reconstruct input data ($x \rightarrow \hat{x}$).
- Finds a meaningful (and low dimensional) hidden representation z.
- Encoder and decoder are often modeled via deep networks.
- Useful for classification/transformation, but not generation.





Variational autoencoders

- Change the loss function to one that captures KL divergence between p_{data} and p_{model} and minimize it.
  - Probabilistic hidden layer
  - Results in a evidence lower bound (ELBO) that needs to be maximized!

$$\begin{aligned}
\log p_\theta(x^{(i)}) &= \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z) \\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad (\text{Bayes' Rule}) \\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad (\text{Multiply by constant}) \\
&= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad (\text{Logarithms}) \\
&= \underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)},\theta,\phi)} + \underbrace{D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))}_{\geq 0}
\end{aligned}$$

Reconstruct the input data

Make approximate posterior distribution close to prior

$$\log p_\theta(x^{(i)}) \geq \mathcal{L}(x^{(i)},\theta,\phi)$$

Variational lower bound ("ELBO")

$$\theta^*, \phi^* = \arg\max_{\theta,\phi} \sum_{i=1}^{N} \mathcal{L}(x^{(i)},\theta,\phi)$$

Training: Maximize lower bound

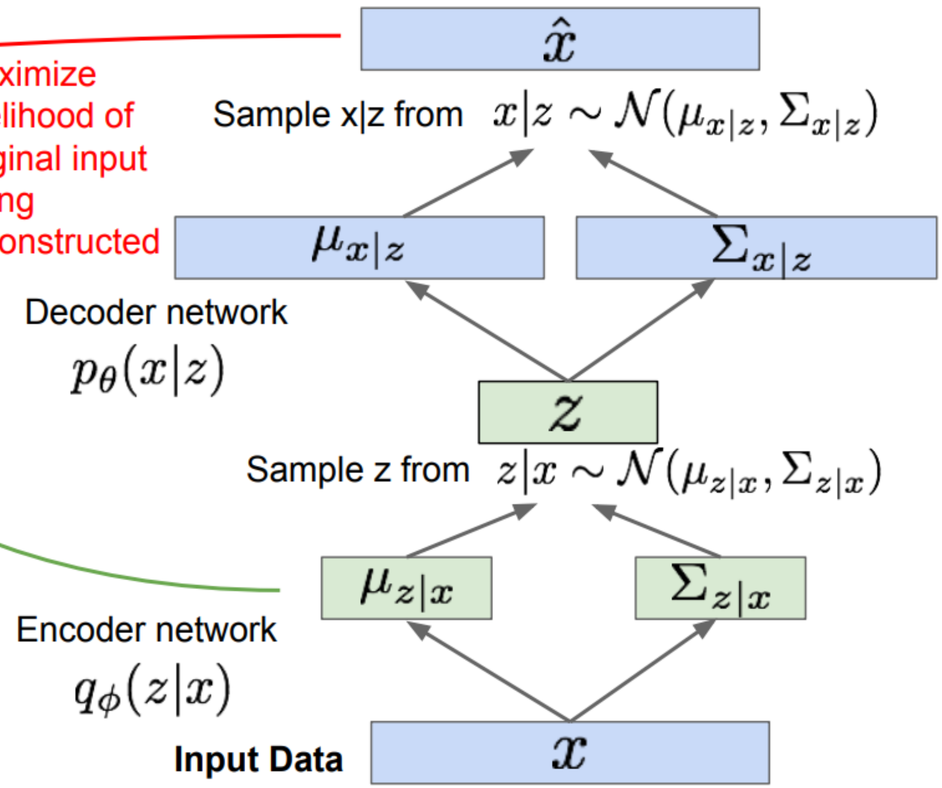
VAE Training

Putting it all together: maximizing the likelihood lower bound

$$\underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)}$$

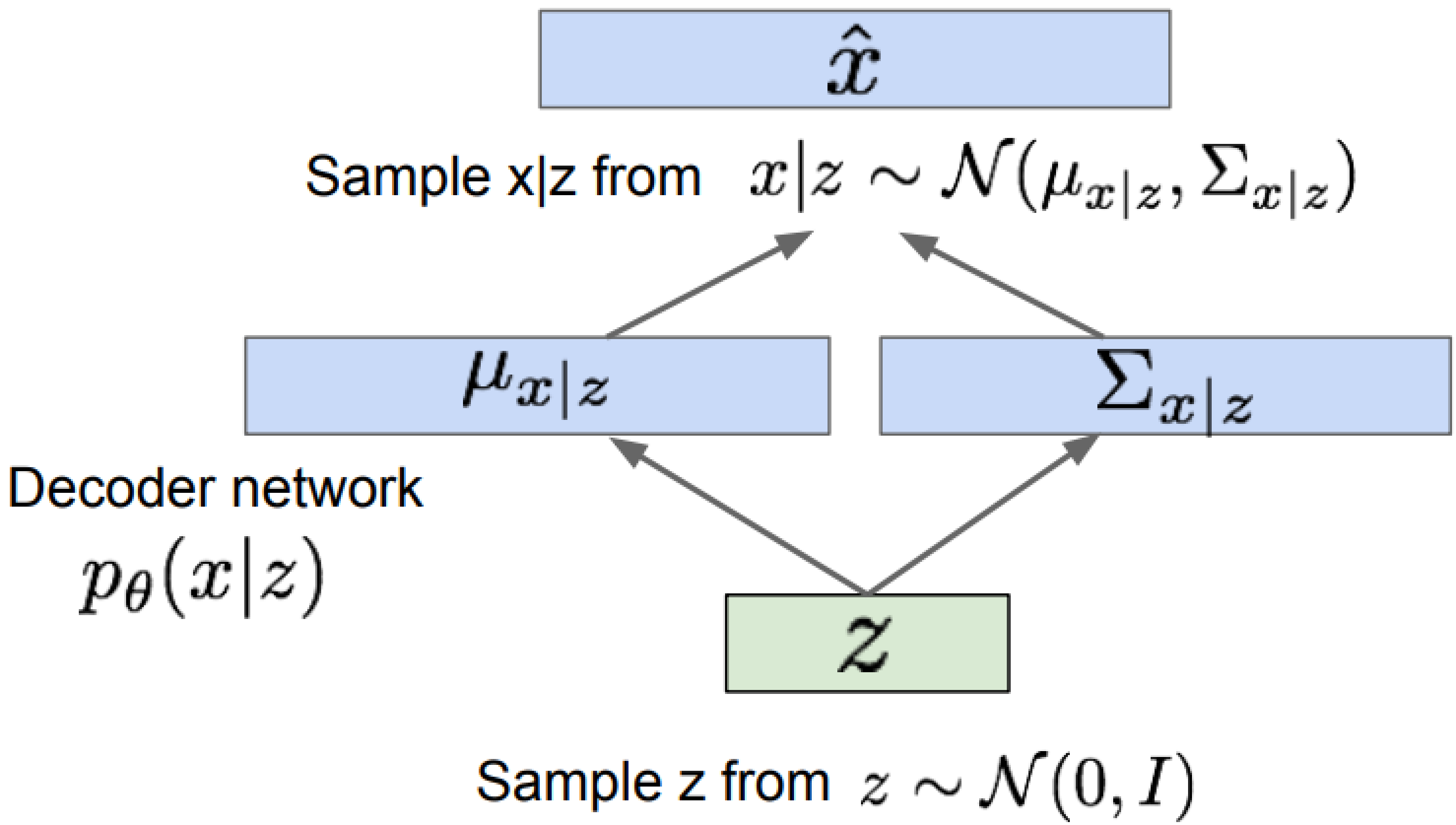
Maximize likelihood of original input being reconstructed

Make approximate posterior distribution close to prior

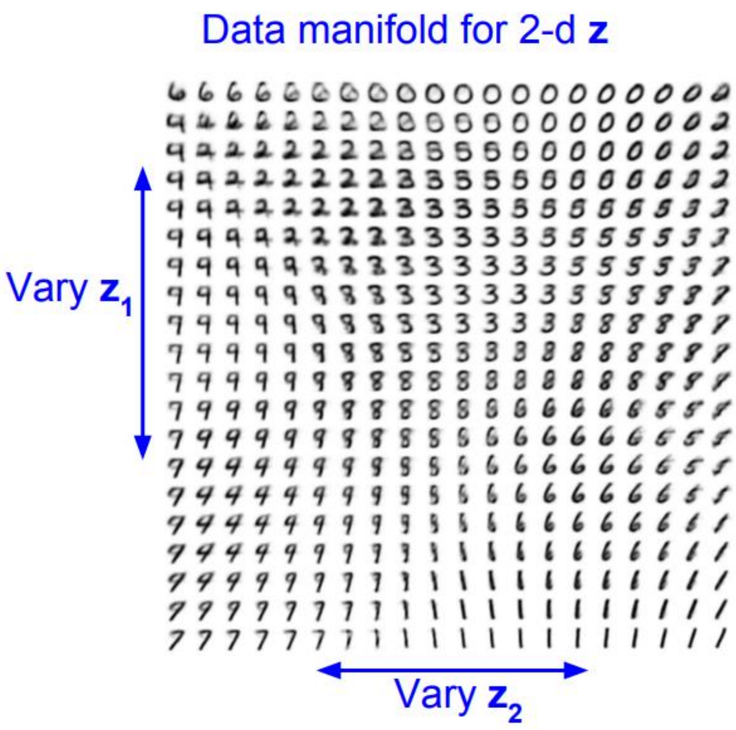


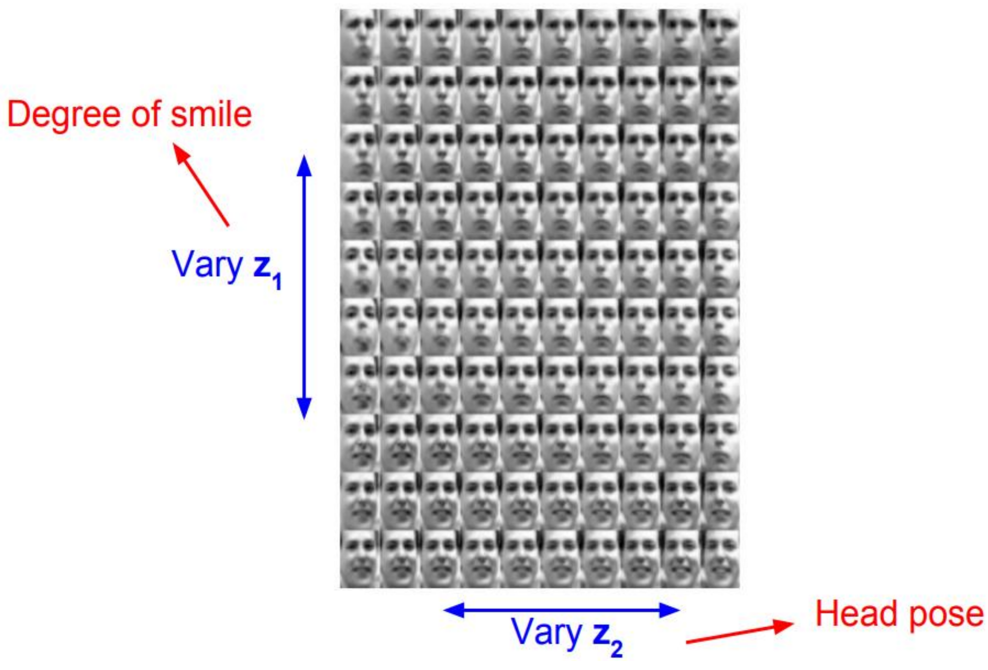
VAE generation

Use decoder network. Now sample z from prior!



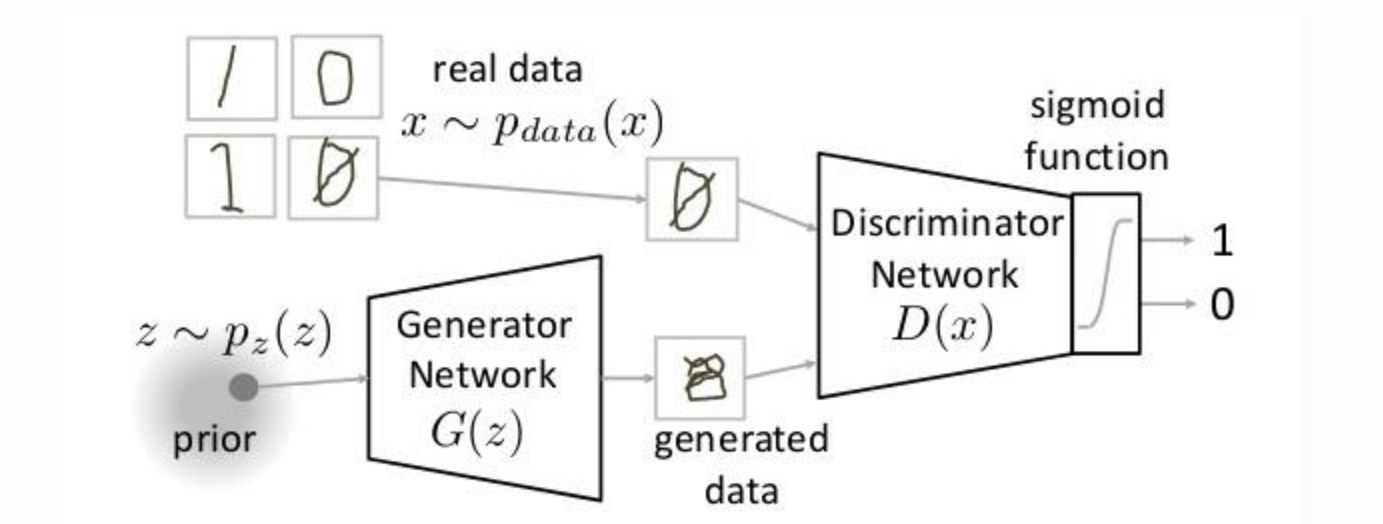
Structure incorporated in latent variables!





Generative Adversarial Networks

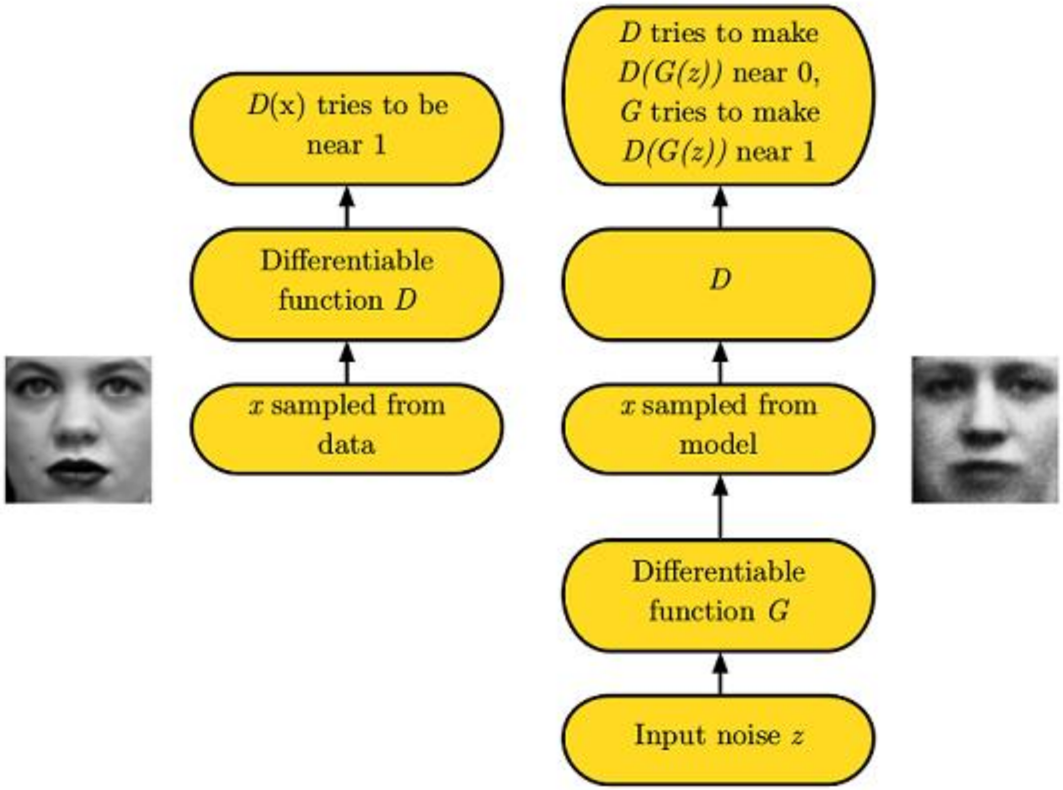
- No explicit probability density function required!
- Game theoretic approach
  - Two player game
    - Discriminator network: tries to discriminate between real and generated images.
    - Generator network: tries to fool discriminator by generating real-looking images from noise.



GAN Math

Minimax objective function:

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



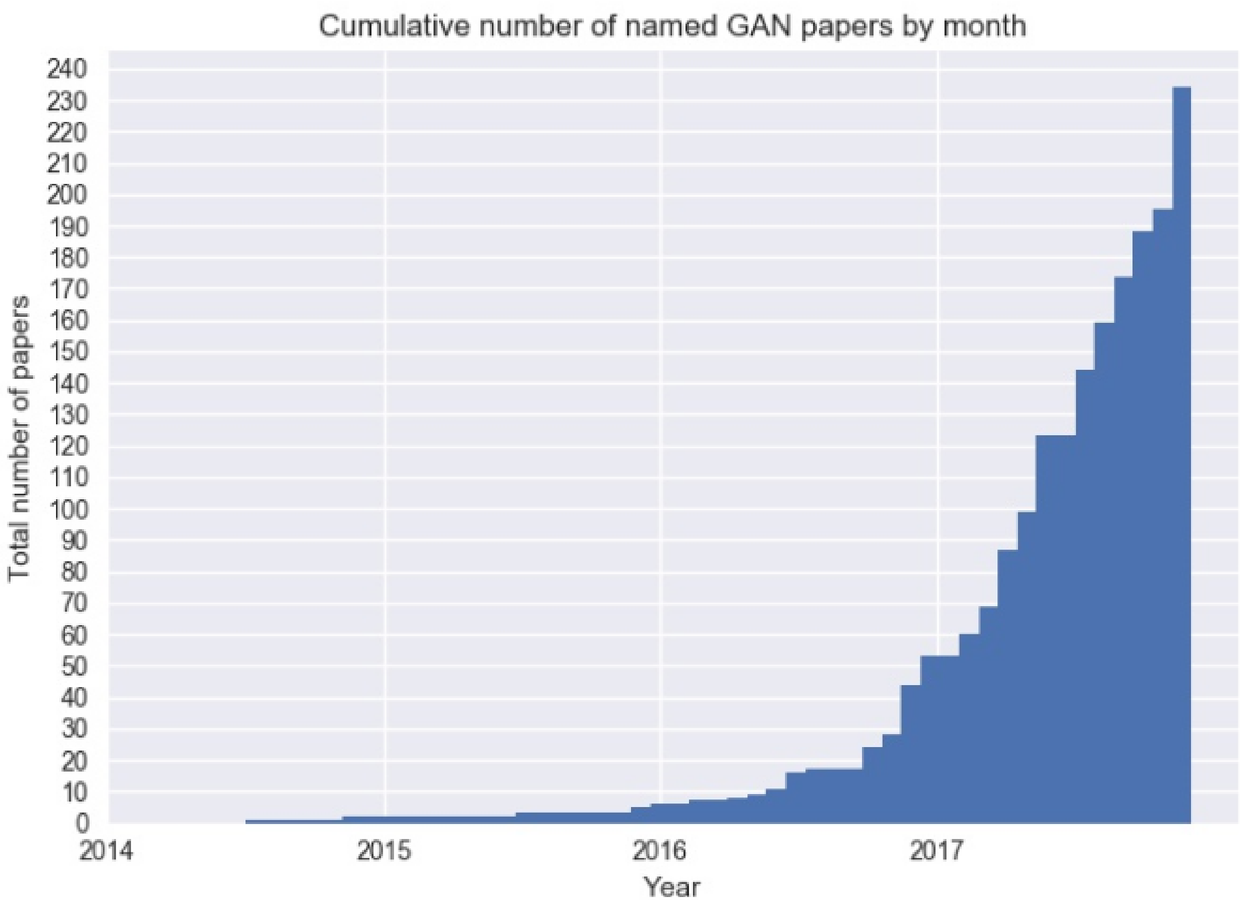
Advantages of GANs over other models

- Parallel sample generation compared to Pixel RNN
- No variational or approximate bounds needed
- No explicit density function required
- Generator design has very few restrictions
  - RBMs: Distributions that admit Markov chain sampling
  - Linear ICA: Invertible distributions
- Subjectively regarded as producing crispiest samples!

GAN versus VAE

Name	***Epoch 1***	***Epoch 10***	***Epoch 25***
VAE			
GAN			

GanZoo – 300+ types of GANs



2015

2017

https://github.com/hindupuravinash/the-gan-zoo

SOTA Generated Images

References

- CS231 course notes: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf
- Ian Goodfellow Nips 2016 Keynote: https://arxiv.org/pdf/1701.00160.pdf
- My blog article on SDSRA-AI: https://sdsra-ai.github.io/blog/2017/04/07/Inverse-Transform-Sampling-Via-Generative-Adversarial-Networks.html

Discriminative and Generative Models

- DISCRIMINATIVE - Learns a function that maps input x to output y. In probabilistic terms, it learns the conditional $p(y|x)$.
- GENERATIVE - Learns the joint distribution of x and y simultaneously. In probabilistic terms, it leans $p(x,y)$.
- Note: $p(x,y)$ can always be converted to $p(y|x)$ via Bayes rule, but more importantly, it can help create new (x,y) samples.
- Generative models are particularly useful for generating a lot of fresh data from little unlabeled data (semi-supervised learning).

Restricted Boltzmann Machine (RBM)

- Symmetric connections; no intra-layer connection between hidden units or visible units.
- **Constrastive divergence** algorithm is used for training.
  - Given v, compute the probabilities of h and sample a hidden activation vector h; compute the outer product of v and h and call this the positive gradient.
  - From h, sample a reconstruction v', then h' from this. (Gibbs sampling step)
  - Compute the outer product of v' and h' and call this the negative gradient.

Flavors of generative models

- An auto-encoder attempts to minimize the squared distance between $x_{\text{data and xmodel.}}$
- A GAN attempts to minimize the KL divergence between $p(x_{data})$ and $p(x_{model})$.
- A variational auto-encoder attempts to minimize the KL divergence between $p(z_{data} | x_{data})$ and $p(z_{model} | x_{model})$

Deep Belief Nets

- Obtained by stacking RBMs