Bayesian Deep Learning

Going full Bayesian

- Bayes = Probabilities

- Bayes Theorem

Hypothesis = Model

$$p(H|E) = \frac{p(E|H)p(H)}{p(E)}$$

Evidence = data

Going full Bayesian

- Start with a prior on the model parameters $p(\theta)$

- Choose a statistical model $p(x|\theta)$

  data

- Use data to refine my prior, i.e., compute the posterior

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{\cancel{p(x)}}$$

No dependence on parameters

Going full Bayesian

- Start with a prior on the model parameters $p(\theta)$

- Choose a statistical model $p(x|\theta)$

  data

- Use data to refine my prior, i.e., compute the posterior

$$\underbrace{p(\theta|x)}_{\text{posterior}} = \underbrace{p(x|\theta)}_{\text{likelihood}}\underbrace{p(\theta)}_{\text{prior}}$$

Going full Bayesian

- 1. Learning: Computing the posterior
  - Finding a point estimate (MAP) → what we have been doing so far!

$$p(\theta|x) = p(x|\theta)p(\theta)$$

  - Finding a probability distribution of θ

This lecture

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)}$$

What have we learned so far?

- Advantages of Deep Learning models
  - Very expressive models
  - Good for tasks such as classification, regression, sequence prediction
  - Modular structure, efficient training, many tools
  - Scales well with large amounts of data
- But we have also disadvantages…
  - "Black-box" feeling
  - We cannot judge how “confident” the model is about a decision

Modeling uncertainty

- Wish list:
  - We want to know what our models know and what they do not know

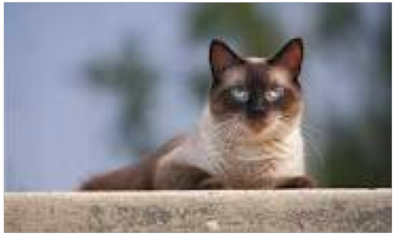
Modeling uncertainty

- Example: I have built a dog breed classifier

Bulldog

German sheperd

Chihuaha



What answer will my NN give?

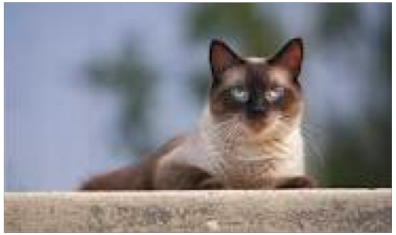
Modeling uncertainty

- Example: I have built a dog breed classifier

Bulldog

German sheperd

Chihuaha



I would rather get as an answer that my model is not certain about the type of dog breed

Modeling uncertainty

- Wish list:
  - We want to know what our models know and what they do not know

- Why do we care?
  - Decision making
  - Learning from limited, noisy, and missing data
  - Insights on why a model failed

Modeling uncertainty

- Finding the posterior
  - Finding a point estimate (MAP) → what we have been doing so far!
  - Finding a probability distribution of θ

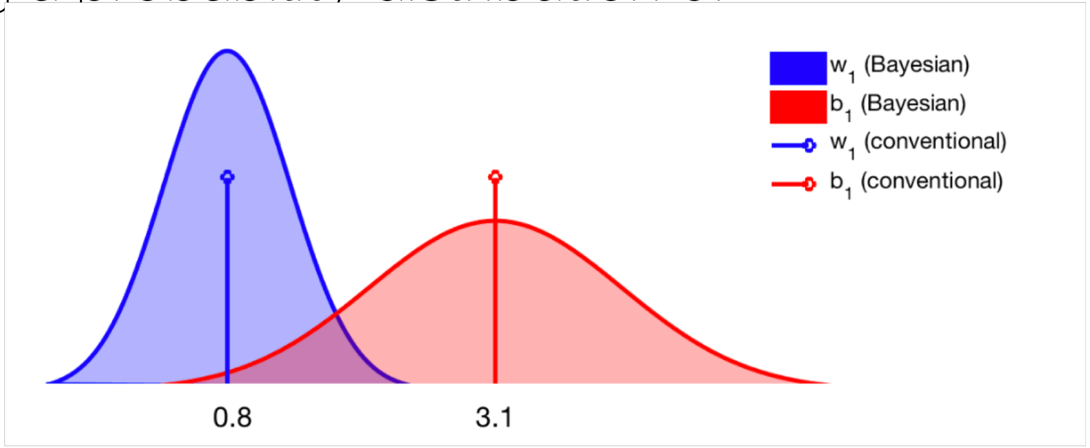


Image: https://medium.com/@joeDiHare/deep-bayesian-neural-networks-952763a9537

Modeling uncertainty

- We can sample many times from the distribution and see how this affects our model's predictions
- If predictions are consistent = model is confident

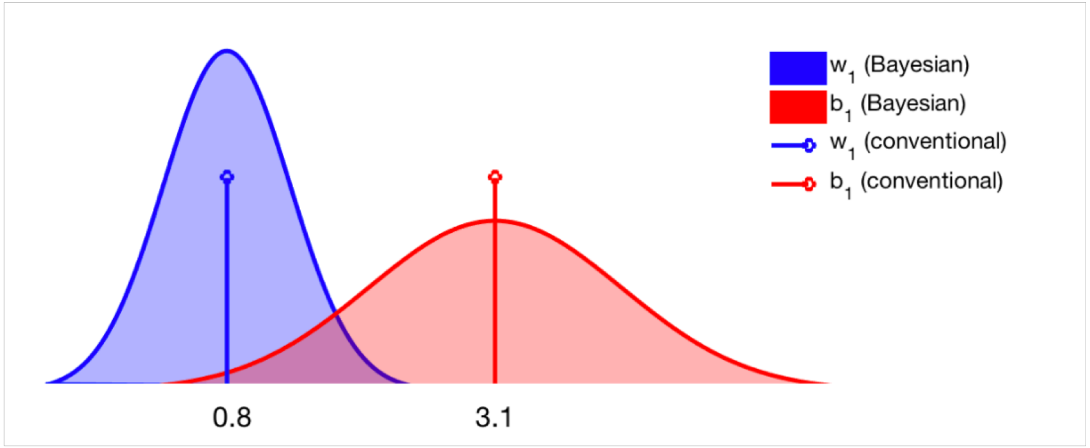
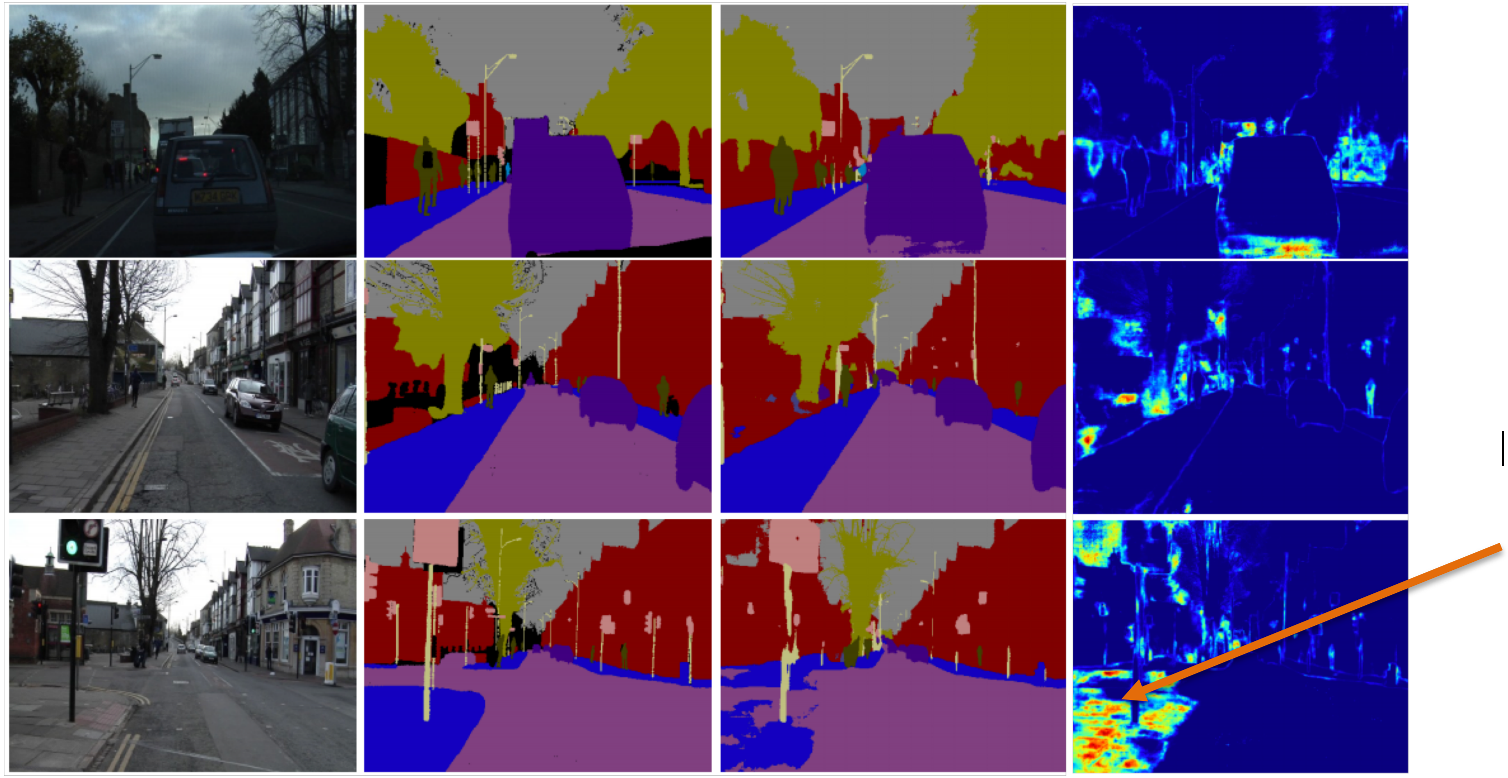


Image: https://medium.com/@joeDiHare/deep-bayesian-neural-networks-952763a9537

Modeling uncertainty



I am not really sure

Kendal & Gal. "What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?" NIPS 2016

How do we get the posterior?

- Compute the posterior over the weights

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)}$$

- Probability of observing our data under all possible model parameters

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{\int_\theta p(x|\theta)p(\theta)d\theta}$$

How do we compute this?

How do we get the posterior?

- How do we compute this?

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{\int_\theta p(x|\theta)p(\theta)d\theta}$$

- Denominator = we cannot compute all possible combinations

- Two ways to compute the approximation of the posterior:

$\boxed{\text{Markov Chain Monte Carlo}}$

$\boxed{\text{Variational Inference}}$

How do we get the posterior?

- Markov Chain Monte Carlo (MCMC)
  - A chain of samples

$$\theta_t \rightarrow \theta_{t+1} \rightarrow \theta_{t+2} \quad \dots$$

SLOW

that converge to $p(\theta|x)$

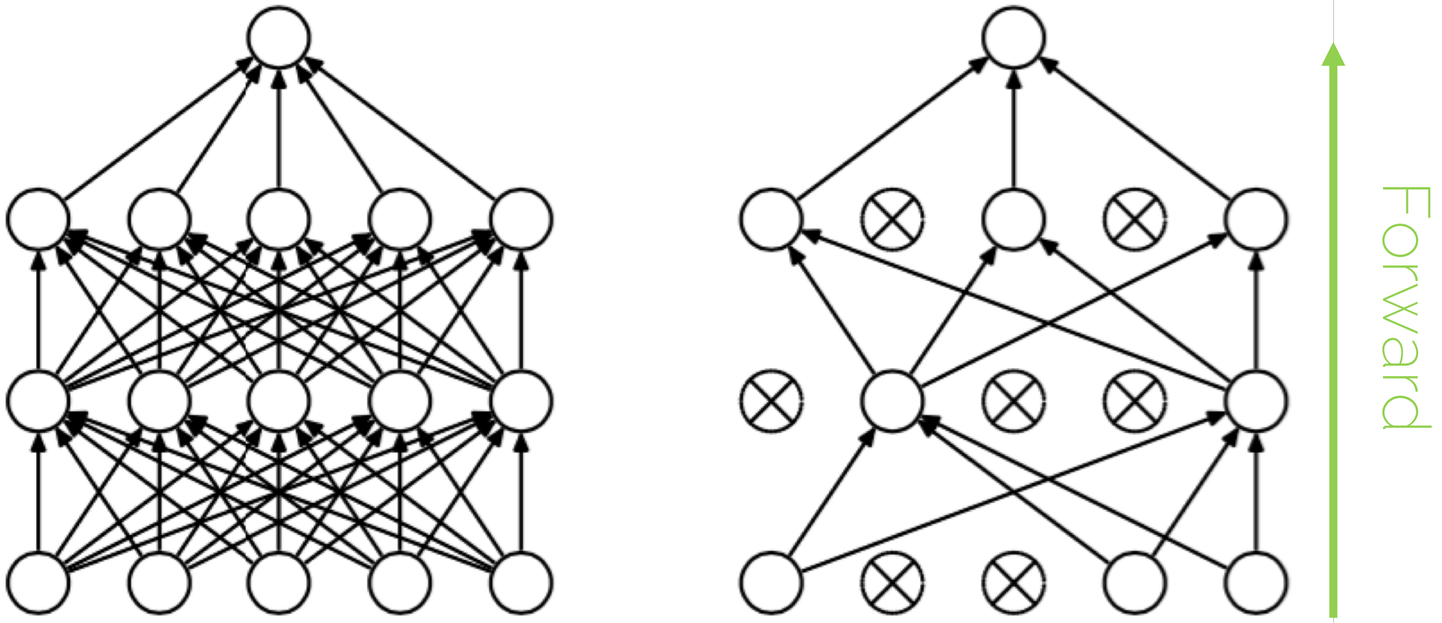
- Variational Inference
  - Find an approximation $q(\theta)$ that:

$$\arg\min KL(q(\theta)||p(\theta|x))$$

Dropout for Bayesian Inference

Recall: Dropout

- Disable a random set of neurons (typically 50%)



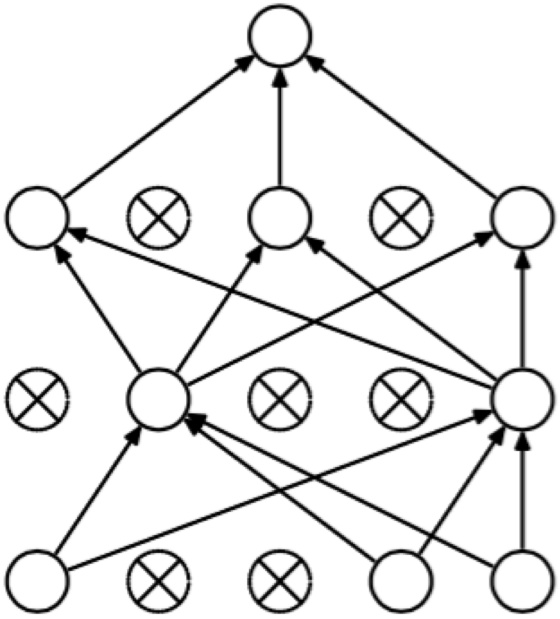
(a) Standard Neural Net (b) After applying dropout.

Srivastava 2014

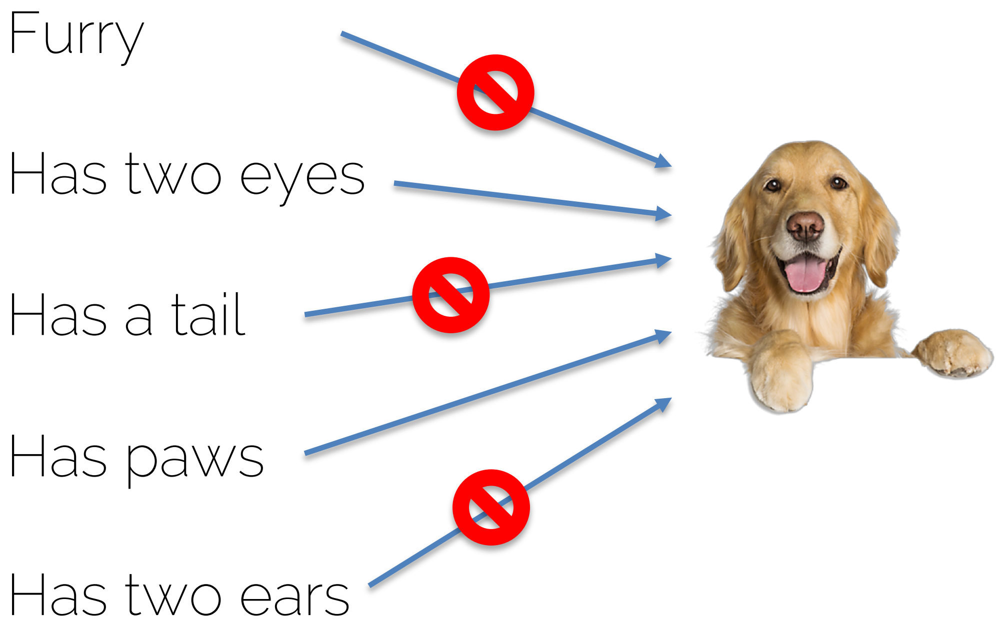
Recall: Dropout

- Using half the network = half capacity

Redundant representations



(b) After applying dropout.



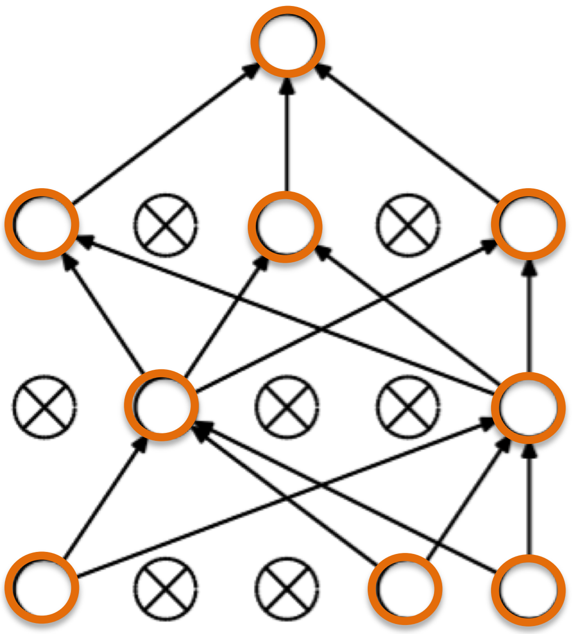
Recall: Dropout

- Using half the network = half capacity
  - Redundant representations
  - Base your scores on more features

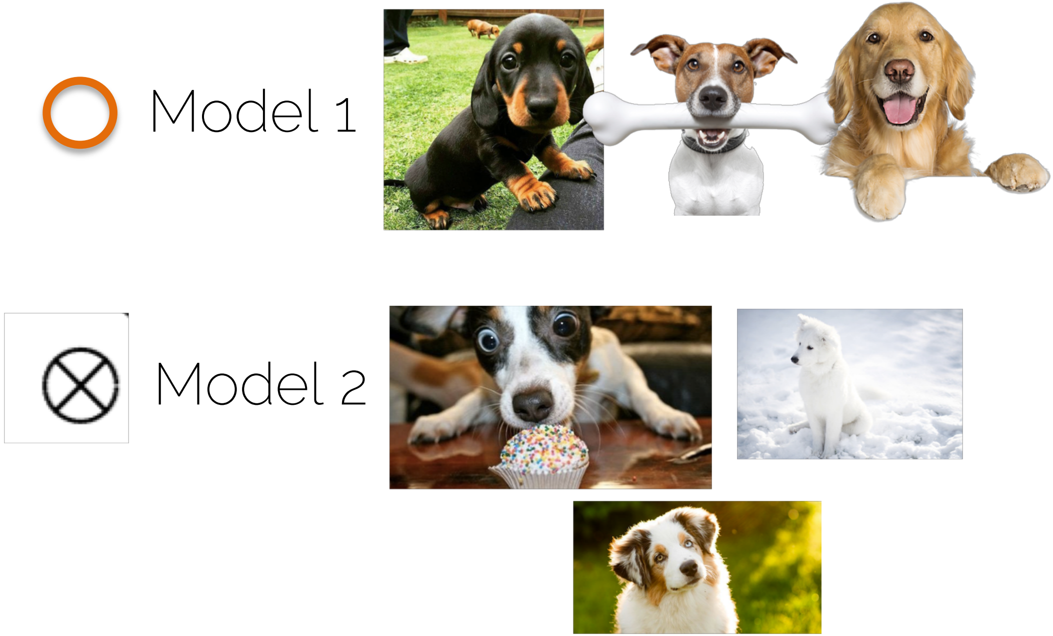
- Consider it as model ensemble

Recall: Dropout

- Two models in one



(b) After applying dropout.



MC dropout

- Variational Inference
  - Find an approximation $q(\theta)$ that

$$\arg\min KL(q(\theta)||p(\theta|x))$$

$$=$$

- Dropout training
  - The variational distribution is from a Bernoulli distribution (where the states are "on" and "off")

Y Gal, Z Ghahramani, "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning", ICML 2016

MC dropout

- 1. Train a model with dropout before every weight layer
- 2. Apply dropout at **test** time
  - Sampling is done in a Monte Carlo fashion, hence the name Monte Carlo dropout

Y Gal, Z Ghahramani, "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning", ICML 2016

MC dropout

– Sampling is done in a Monte Carlo fashion, e.g.,

$$p(y = c|x) \approx \frac{1}{T} \sum_{t=1}^{T} Softmax(f_{\hat{\theta}_t}(x))$$

classification

NN

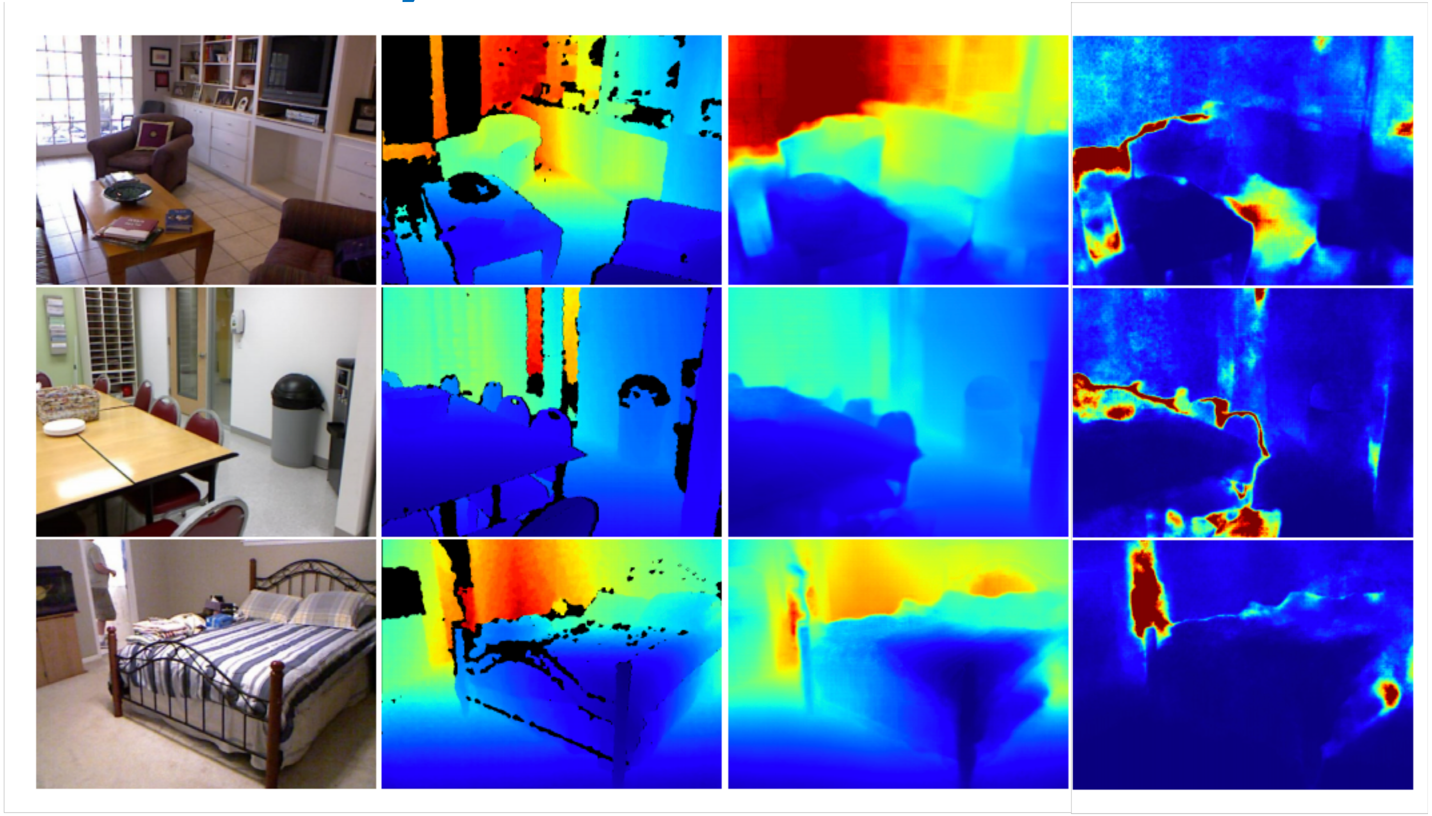
Parameter sampling

where $\hat{\theta}_t \sim q(\theta)$

and $q(\theta)$ is the dropout distribution

Y Gal, Z Ghahramani, "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning", ICML 2016

Measure your model's uncertainty



Kendal & Gal. "What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?" NIPS 2016

Another look

Let us take another look

- We know it is intractable, we approximate it

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{\int_\theta p(x|\theta)p(\theta)d\theta}$$

- The denominator expresses how my data is generated

$$p(x) = \int_\theta p(x|\theta)p(\theta)d\theta$$

Let us take another look

- We assume that the data is generated by some random process, involving an unobserved continuous random (latent) variable z

- Generation process: $$p_\theta(x) = \int_z p_\theta(x|z)p_\theta(z)dz$$

- Posterior: $$\boxed{p_\theta(z|x)} = \frac{p_\theta(x|z)p_\theta(z)}{\int_z p_\theta(x|z)p_\theta(z)dz}$$

Let us take another look

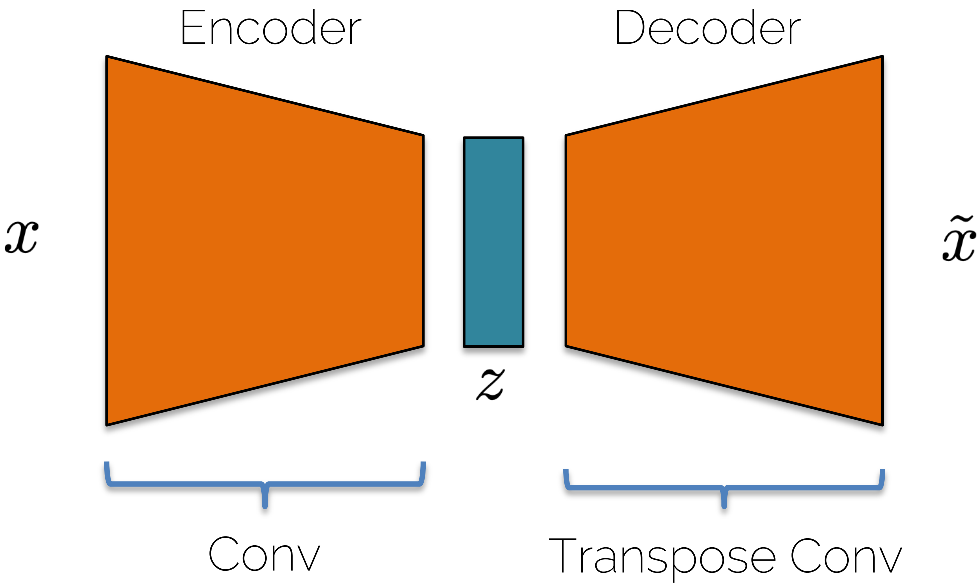
- Variational Inference
  - Find an approximation. $q_\phi(z|x)$

- My approximation is parameterized by a model ϕ

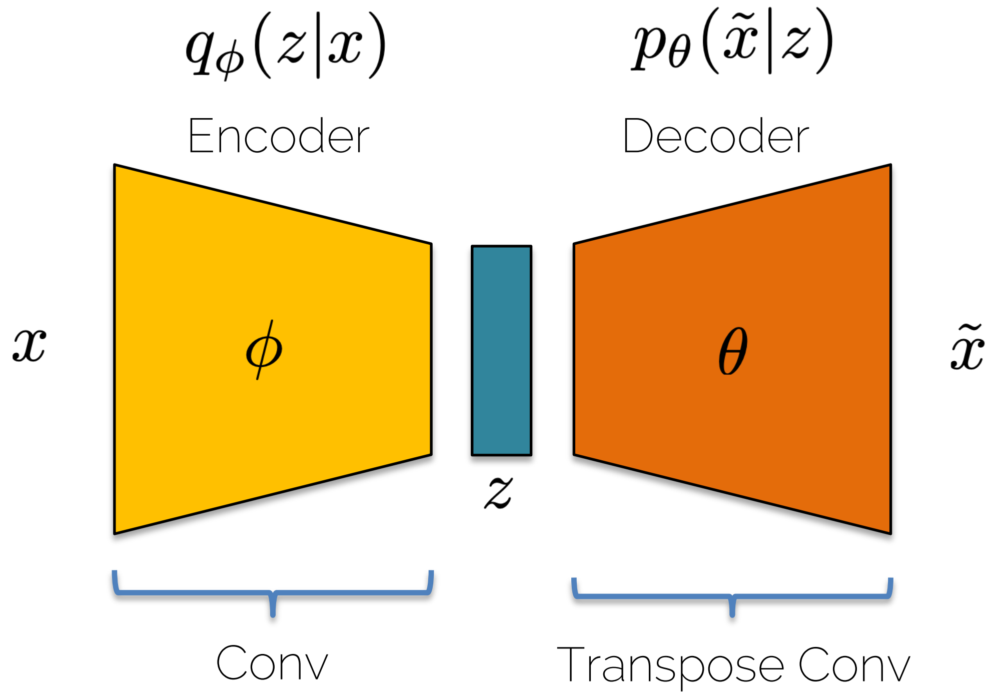
Variational Autoencoders

Recall: Autoencoders

- Encode the input into a representation (bottleneck) and reconstruct it with the decoder

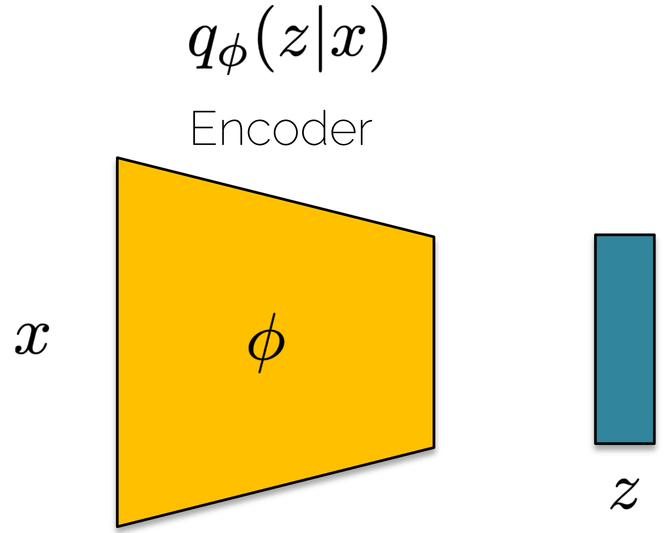


Variational Autoencoder



Variational Autoencoder

- Latent space is now a distribution
- Specifically it is a Gaussian

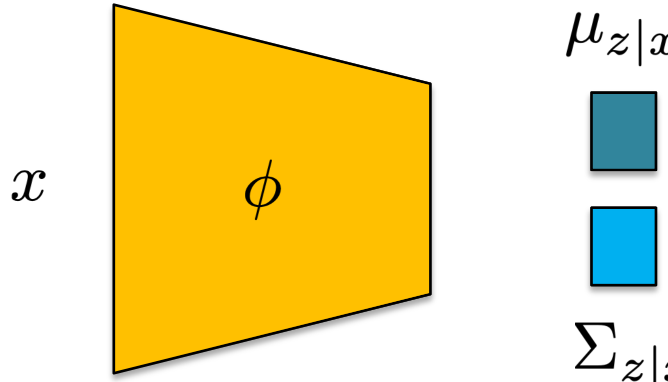


Variational Autoencoder

- Latent space is now a distribution
- Specifically it is a Gaussian

$q_\phi(z|x)$

Encoder



$\mu_{z|x}$ Mean

$$z|x \sim \mathcal{N}(\mu_{z|x}, \Sigma_{z|x})$$

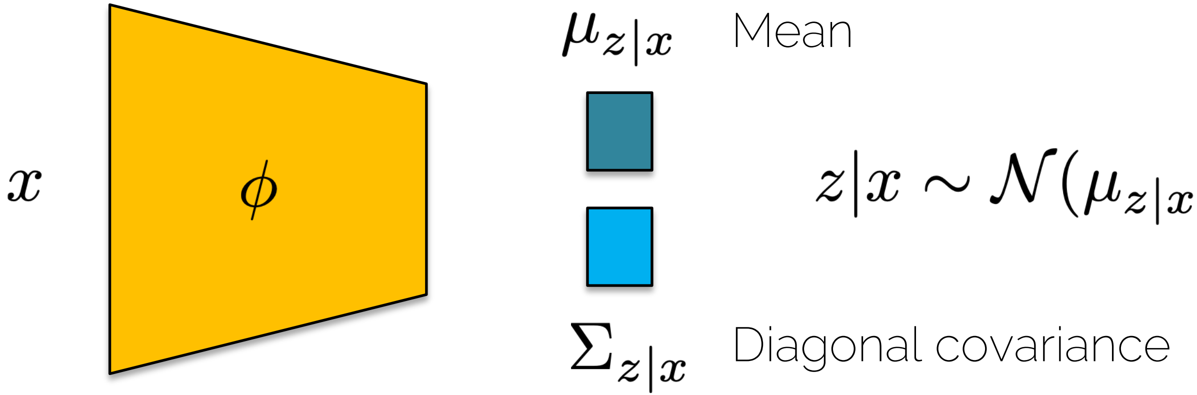
$\Sigma_{z|x}$ Diagonal covariance

Variational Autoencoder

- Latent space is now a distribution
- Specifically it is a Gaussian

$q_\phi(z|x)$

Encoder



$$z|x \sim \mathcal{N}(\mu_{z|x}, \Sigma_{z|x})$$

Variational Autoencoder

- Back to our Bayesian view, our generation process was:

$$p_\theta(x) = \int_z p_\theta(x|z)p_\theta(z)dz$$

- Which is the denominator of the posterior:

$$p_\theta(z|x) = \frac{p_\theta(x|z)p_\theta(z)}{p_\theta(x)}$$

I want to optimize

Variational Autoencoder

- Loss function for a data point $\boldsymbol{x_i}$

$$\log(p_\theta(x_i)) = \boldsymbol{E}_{z\sim q_\phi(z|x_i)}[\log(p_\theta(x_i))]$$

I draw samples of the latent variable z from my encoder

Variational Autoencoder

- Loss function for a data point x_i

$$\log(p_\theta(x_i)) = \boldsymbol{E}_{z\sim q_\phi(z|x_i)}[\log(p_\theta(x_i))]$$

$$= \boldsymbol{E}_{z\sim q_\phi(z|x_i)}\left[\log \frac{p_\theta(x_i|z)p_\theta(z)}{p_\theta(z|x_i)}\right]$$

Bayes Rule

Posterior

Variational Autoencoder

- Loss function for a data point x_i

$$\log(p_\theta(x_i)) = \boldsymbol{E}_{z\sim q_\phi(z|x_i)}[\log(p_\theta(x_i))]$$

$$= \boldsymbol{E}_{z\sim q_\phi(z|x_i)}\left[\log \frac{p_\theta(x_i|z)p_\theta(z)}{p_\theta(z|x_i)}\right]$$

$$= \boldsymbol{E}_z\left[\log \frac{p_\theta(x_i|z)p_\theta(z)}{p_\theta(z|x_i)} \frac{q_\phi(z|x_i)}{q_\phi(z|x_i)}\right]$$

Just a constant

Variational Autoencoder

- Loss function for a data point x_i

$$\log(p_\theta(x_i)) = \boldsymbol{E}_z \left[\log \frac{\boxed{p_\theta(x_i|z)}\boxed{p_\theta(z)}}{\boxed{p_\theta(z|x_i)}} \frac{\boxed{q_\phi(z|x_i)}}{\boxed{q_\phi(z|x_i)}} \right]$$

$$= \boxed{\boldsymbol{E}_z \left[\log p_\theta(x_i|z) \right]} - \boxed{\boldsymbol{E}_z \left[\log \frac{q_\phi(z|x_i)}{p_\theta(z)} \right]} + \boxed{\boldsymbol{E}_z \left[\log \frac{q_\phi(z|x_i)}{p_\theta(z|x_i)} \right]}$$

Variational Autoencoder

- Loss function for a data point x_i

$$= \boxed{\boldsymbol{E}_z\left[\log p_\theta(x_i|z)\right]} - \boxed{\boldsymbol{E}_z\left[\log \frac{q_\phi(z|x_i)}{p_\theta(z)}\right]} + \boxed{\boldsymbol{E}_z\left[\log \frac{q_\phi(z|x_i)}{p_\theta(z|x_i)}\right]}$$

$$= \boxed{\boldsymbol{E}_z\left[\log p_\theta(x_i|z)\right]} - \boxed{KL(q_\phi(z|x_i)||p_\theta(z))} + \boxed{KL(q_\phi(z|x_i)||p_\theta(z|x_i))}$$

Kullback-Leibler Divergences

Variational Autoencoder

- Loss function for a data point $\boldsymbol{x_i}$

$$= \boxed{\boldsymbol{E}_z \left[\log p_\theta(x_i|z)\right]} - \boxed{KL(q_\phi(z|x_i)||p_\theta(z))} + \boxed{KL(q_\phi(z|x_i)||p_\theta(z|x_i))}$$

Reconstruction loss

Measures how good my latent distribution is with respect to my prior

I still cannot express the shape of the distribution. But I know

≥ 0

Variational Autoencoder

- Loss function for a data point x_i

$$\underbrace{\boxed{\boldsymbol{E}_z\left[\log p_\theta(x_i|z)\right]} - \boxed{KL(q_\phi(z|x_i)||p_\theta(z))}}_{\substack{\text{Loss function (lower bound)} \\ \mathcal{L}(x_i,\phi,\theta)}} + \underbrace{\boxed{KL(q_\phi(z|x_i)||p_\theta(z|x_i))}}_{\geq 0}$$

$$\log(p(x_i)) \geq \mathcal{L}(x_i,\phi,\theta)$$

Variational Autoencoder

- Loss function for a data point x_i

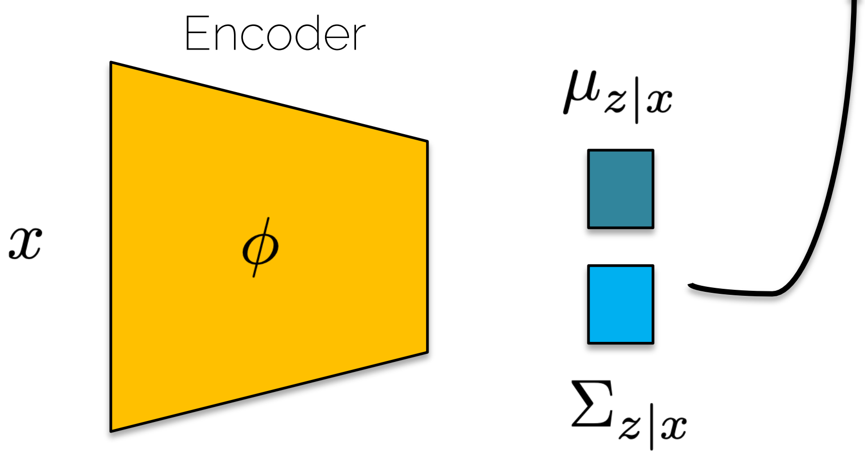
$$\underbrace{\boxed{\boldsymbol{E}_z\left[\log p_\theta(x_i|z)\right]} - \boxed{KL(q_\phi(z|x_i)||p_\theta(z))}}_{\substack{\text{Loss function (lower bound)} \\ \mathcal{L}(x_i,\phi,\theta)}} + \underbrace{\boxed{KL(q_\phi(z|x_i)||p_\theta(z|x_i))}}_{\geq 0}$$

- Optimize $\phi^*, \theta^* = \arg\max \sum_{i=1}^{N} \mathcal{L}(x_i, \phi, \theta)$

Variational Autoencoder

- Training

$$\boxed{\boldsymbol{E}_z\left[\log p_\theta(x_i|z)\right]} - \boxed{KL(q_\phi(z|x_i)||p_\theta(z))} + \boxed{KL(q_\phi(z|x_i)||p_\theta(z|x_i))}$$

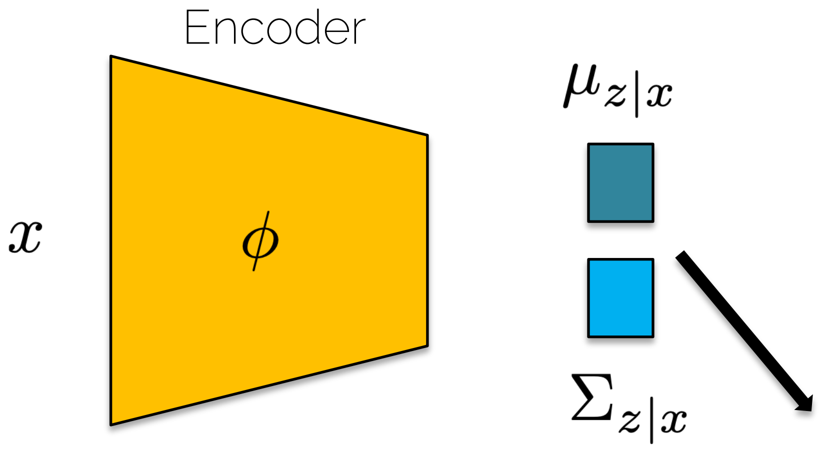


Make posterior distribution close to prior (close to unit Gaussian distribution)

Variational Autoencoder

- Training

$$\boxed{\boldsymbol{E}_z\left[\log p_\theta(x_i|z)\right]} - \boxed{KL(q_\phi(z|x_i)||p_\theta(z))} + \boxed{KL(q_\phi(z|x_i)||p_\theta(z|x_i))}$$

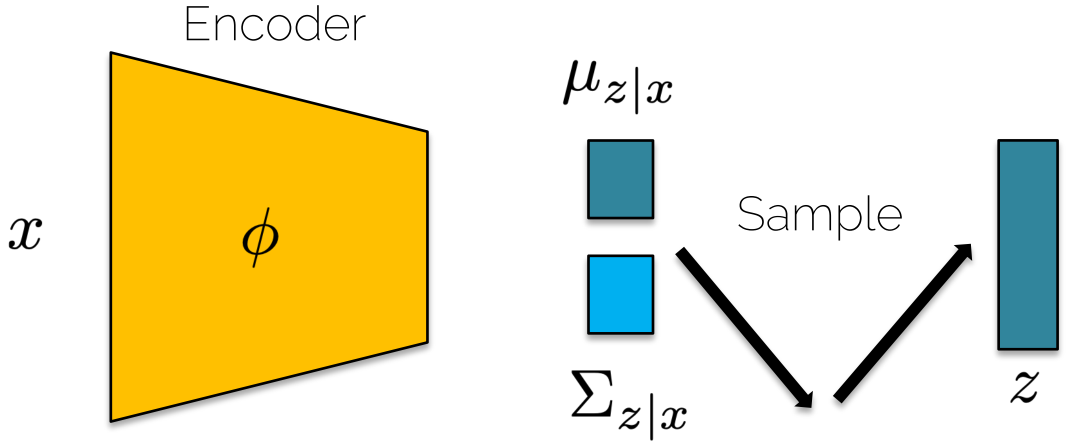


$$z|x \sim \mathcal{N}(\mu_{z|x}, \Sigma_{z|x})$$

Variational Autoencoder

- Training

$$\boxed{\boldsymbol{E}_z \left[\log p_\theta(x_i|z)\right]} - \boxed{KL(q_\phi(z|x_i)||p_\theta(z))} + \boxed{KL(q_\phi(z|x_i)||p_\theta(z|x_i))}$$

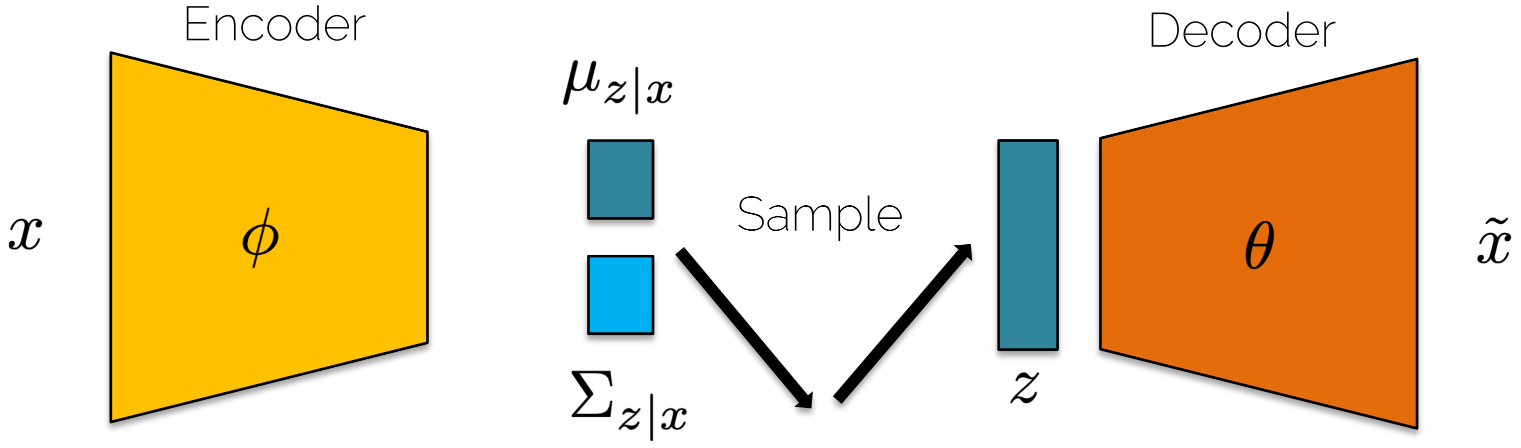


$$z|x \sim \mathcal{N}(\mu_{z|x}, \Sigma_{z|x})$$

Variational Autoencoder

- Training

$$\boxed{\boldsymbol{E}_z\left[\log p_\theta(x_i|z)\right]} - \boxed{KL(q_\phi(z|x_i)||p_\theta(z))} + \boxed{KL(q_\phi(z|x_i)||p_\theta(z|x_i))}$$



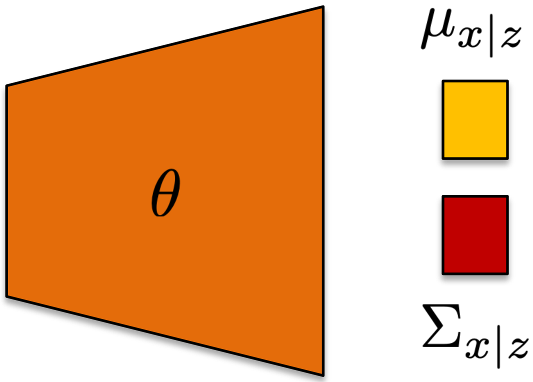
$$z|x \sim \mathcal{N}(\mu_{z|x}, \Sigma_{z|x})$$

Variational Autoencoder

- Training

$$\boxed{\boldsymbol{E}_z\left[\log p_\theta(x_i|z)\right]} - \boxed{KL(q_\phi(z|x_i)||p_\theta(z))} + \boxed{KL(q_\phi(z|x_i)||p_\theta(z|x_i))}$$

Decoder



$$x|z \sim \mathcal{N}(\mu_{x|z}, \Sigma_{x|z}) \xrightarrow{\text{Sample}} \tilde{x}$$

Output is also parameterized

Variational Autoencoder

- Training

$$\boxed{\boldsymbol{E}_z\left[\log p_\theta(x_i|z)\right]} - \boxed{KL(q_\phi(z|x_i)||p_\theta(z))} + \boxed{KL(q_\phi(z|x_i)||p_\theta(z|x_i))}$$

$\tilde{x}$

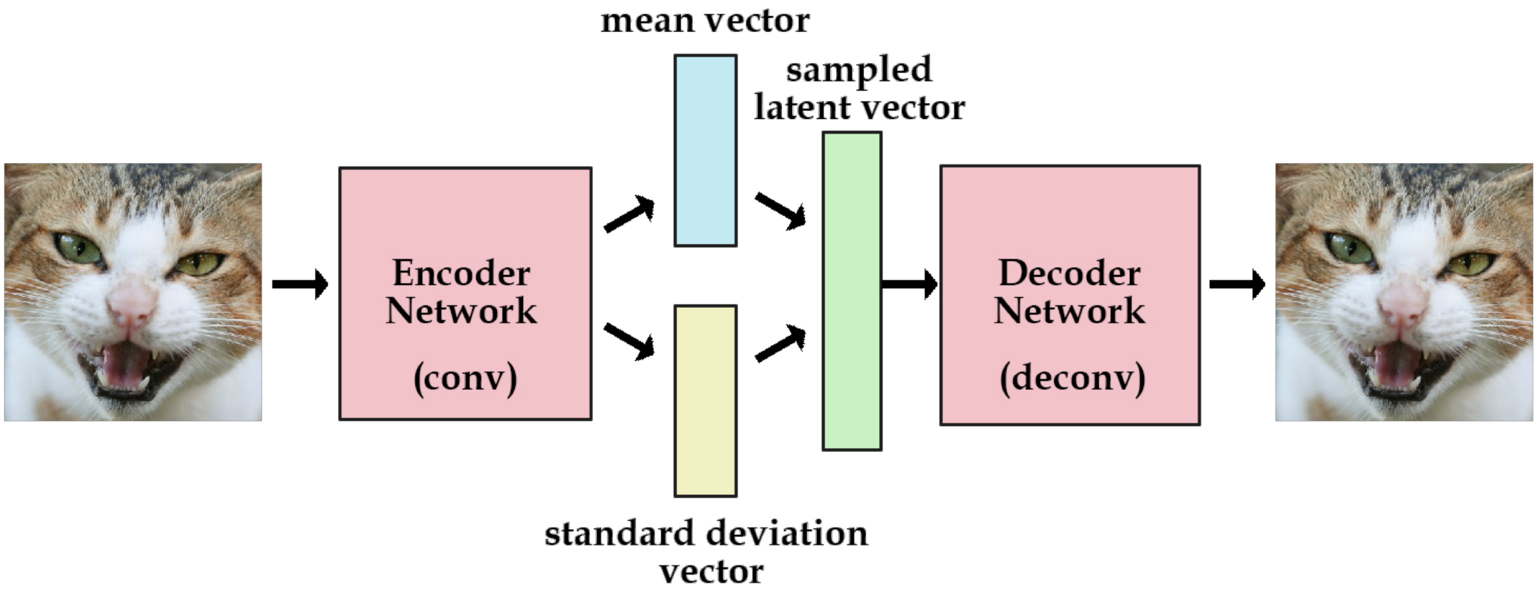
Maximize the likelihood of reconstructing the input

Variational Autoencoder

- For more details and mathematical derivation
- Reparameterization trick that allows us to backprop
- Kingman and Welling. “Auto-Encoding Variational Bayes“. ICLR 2014

How about generating data?

- Training as seen before

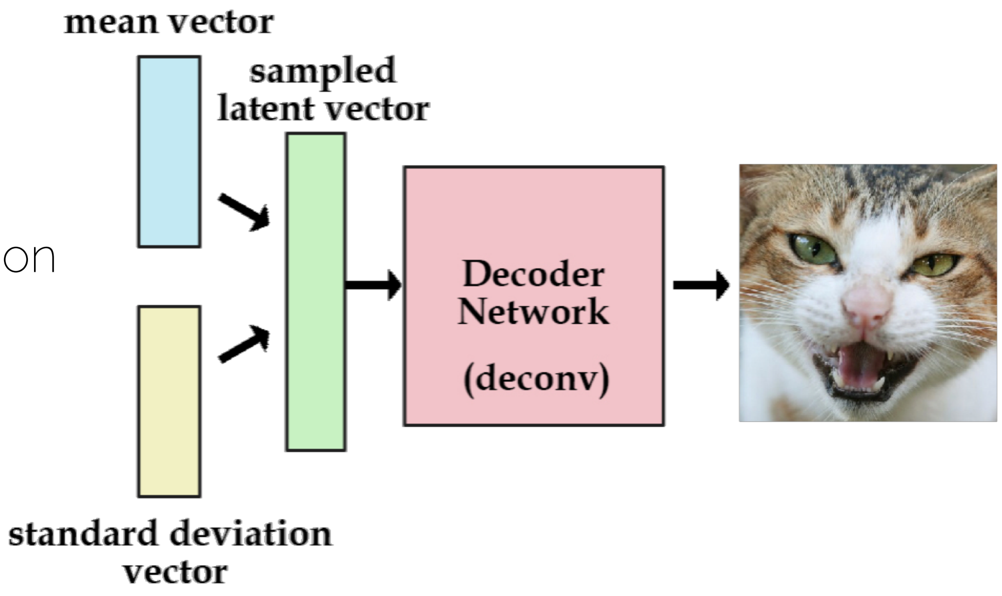


http://kvfrans.com/variational-autoencoders-explained/

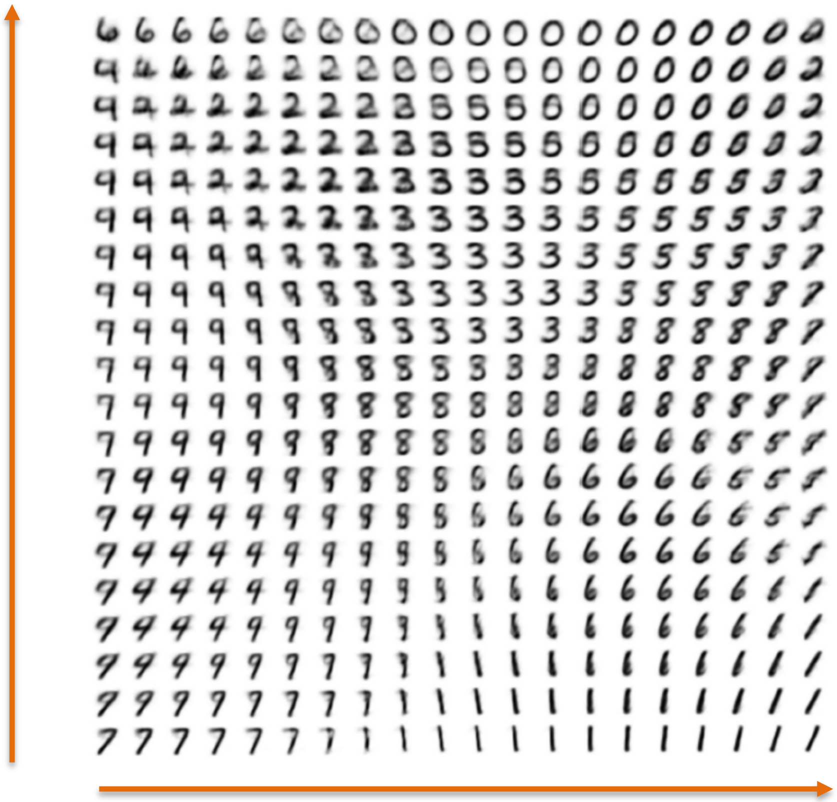
How about generating data?

- After training, generate random samples

Sample from the distribution
(e.g., unit Gaussian



Generating data



Each element of z encodes a different feature

Generating data

Autoencoder vs VAE

Autoencoder

Variational Autoencoder

Ground Truth

Autoencoder Overview

- Autoencoders (AE)
  - Reconstruct input
  - Unsupervised learning
  - Latent space features are useful

- Variational Autoencoders (VAE)
  - Probability distribution in latent space (e.g., Gaussian)
  - Sample from model to generate output

Autoencoder Overview

- Autoencoders (AE)
  - Reconstruct input
  - Unsupervised learning
  - Latent space features are useful

- Variational Autoencoders (VAE)
  - Probability distribution in latent space (e.g., Gaussian)
  - Interpretable latent space (head pose, smile)
  - Sample from model to generate output

Generative models

Taxonomy of generative models

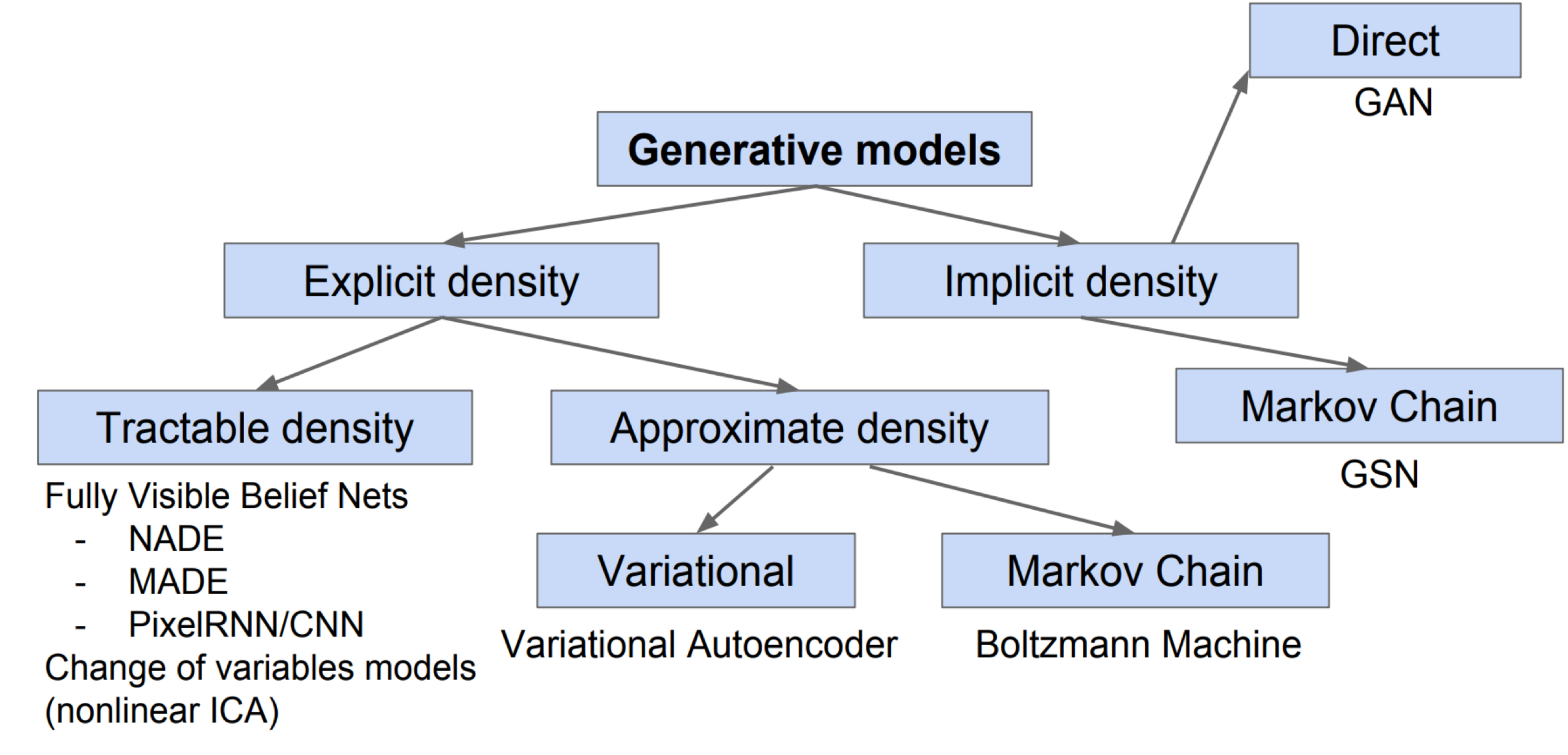


Figure from Ian Goodfellow, Tutorial on Generative Adversarial /networks, 2017

Taxonomy of generative models

$$p_\theta(x) = \int_z p_\theta(x|z)p_\theta(z)dz$$

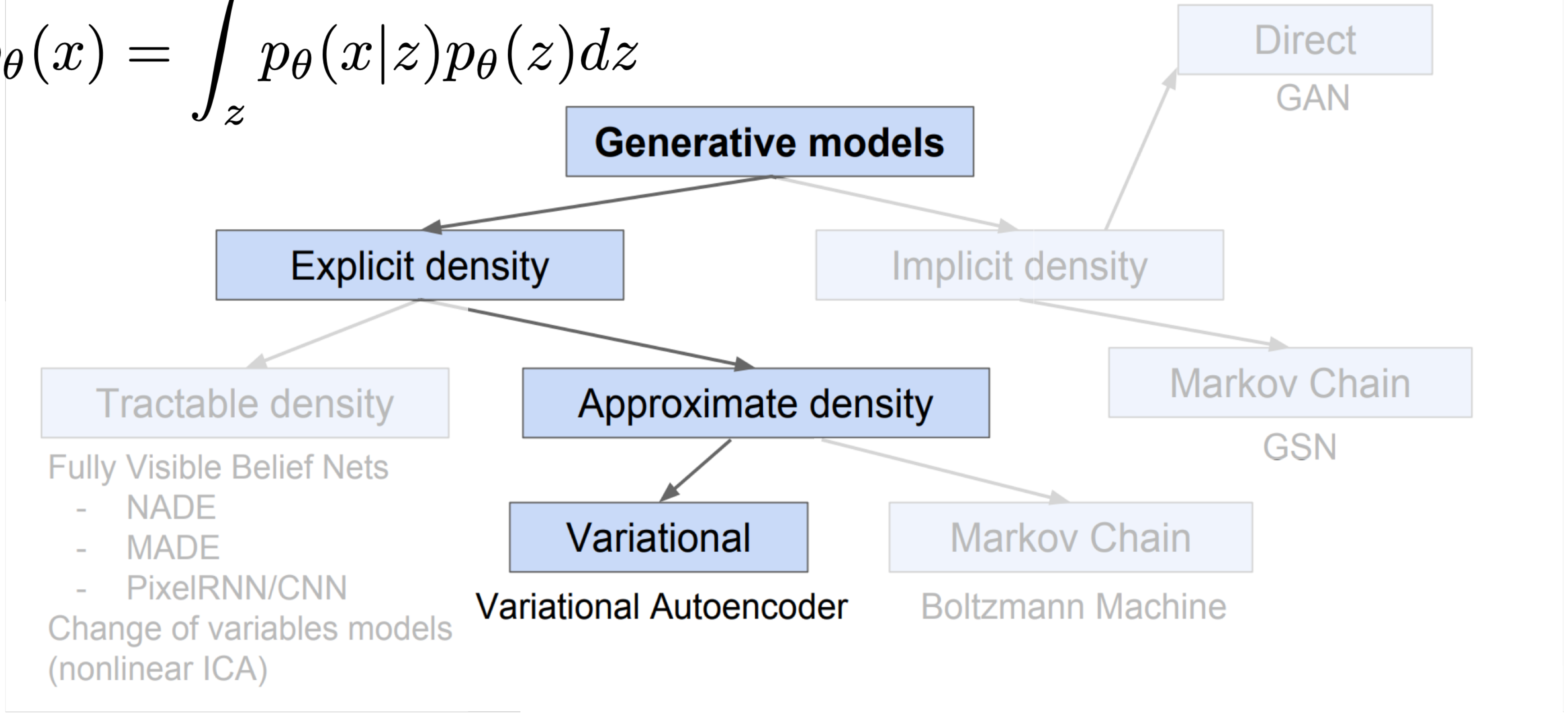


Figure from Ian Goodfellow, Tutorial on Generative Adversarial /networks, 2017

Taxonomy of generative models

Define a more tractable density function

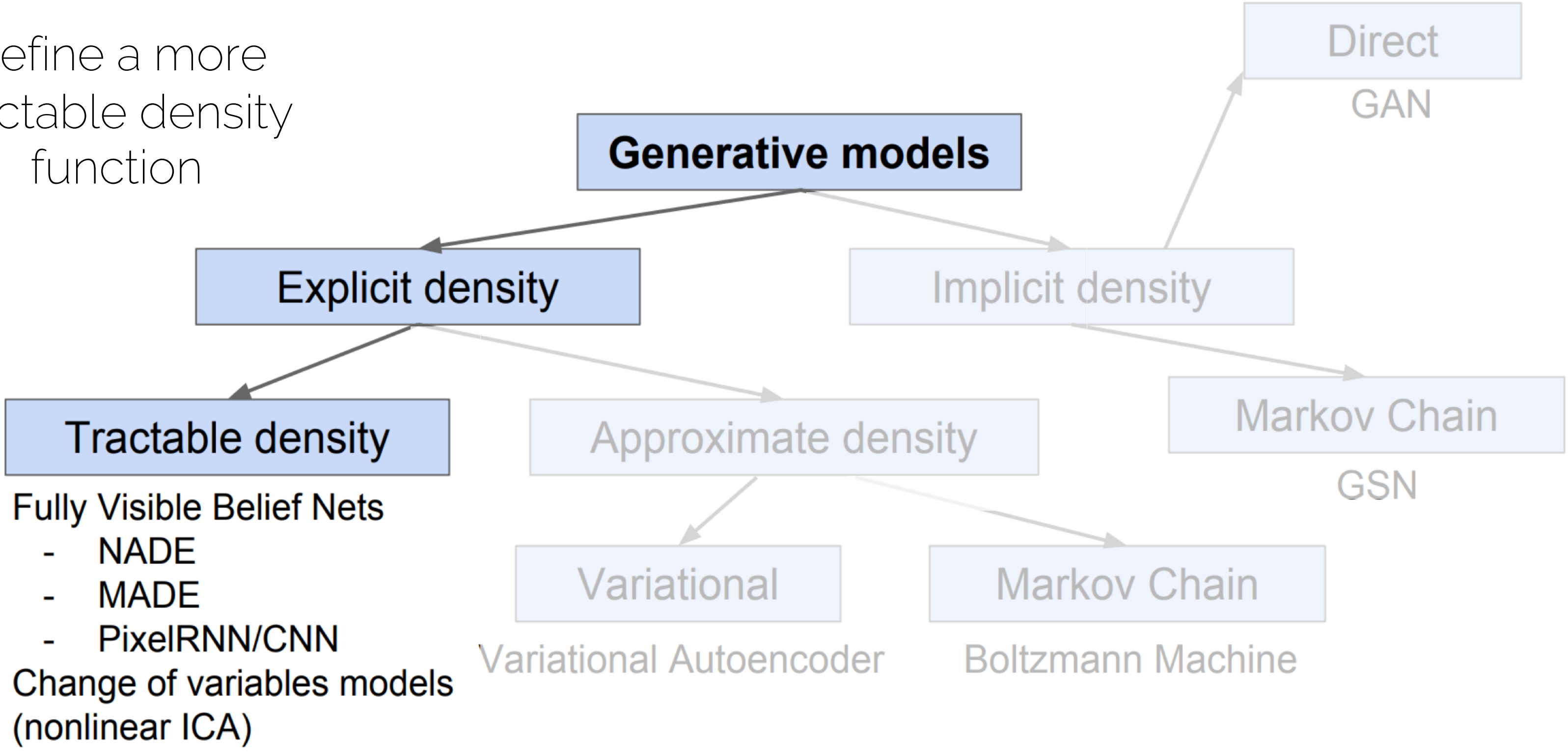


Figure from Ian Goodfellow, Tutorial on Generative Adversarial /networks, 2017

Taxonomy of generative models

I do not care about the shape, I just want to sample!

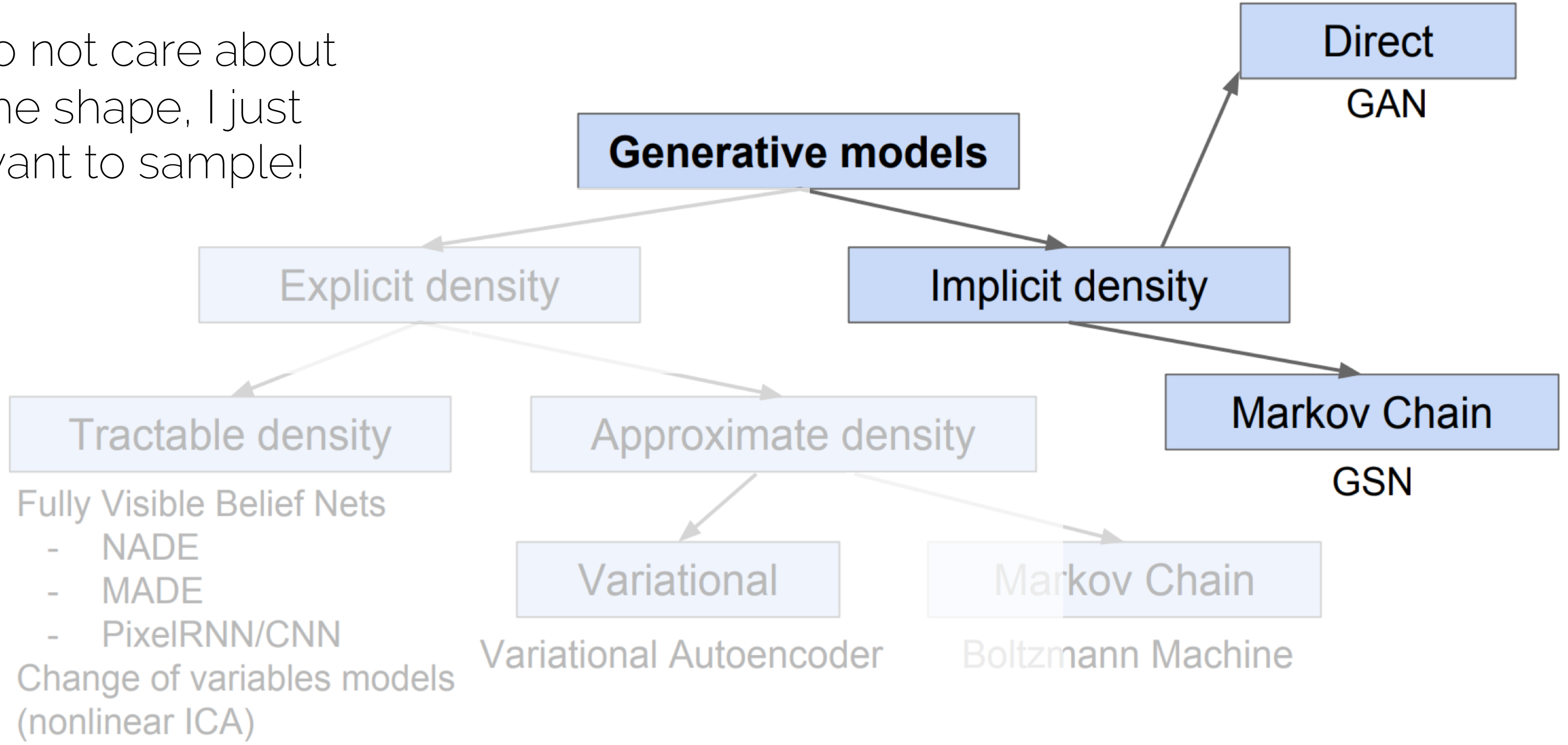


Figure from Ian Goodfellow, Tutorial on Generative Adversarial /networks, 2017

Next lectures

- Next Monday 10th, more on Generative models

- 3rd round of presentations this Friday → you will receive feedback about the presentations

- Keep working on the projects!

Other references

- Conditional Variational Autoencoders:
  - Sohn, Kihyuk, Honglak Lee, and Xinchen Yan. "Learning Structured Output Representation using Deep Conditional Generative Models." Advances in Neural Information Processing Systems. 2015.
  - Xinchen Yan, Jimei Yang, Kihyuk Sohn, Honglak Lee, Attribute2Image: Conditional Image Generation from Visual Attributes, ECCV, 2016 –

Other references

- Interesting read:
  - Jacob Walker, Carl Doersch, Abhinav Gupta, Martial Hebert, An Uncertain Future: Forecasting from Static Images using Variational Autoencoders, ECCV, 2016
  - Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, Ole Winther, Autoencoding beyond pixels using a learned similarity metric, ICML, 2016
  - Aditya Deshpande, Jiajun Lu, Mao-Chuang Yeh, David Forsyth, Learning Diverse Image Colorization, arXiv, 2016
  - Raymond Yeh, Ziwei Liu, Dan B Goldman, Aseem Agarwala, Semantic Facial Expression Editing using Autoencoded Flow, arXiv, 2016
  - Diederik P. Kingma, Danilo J. Rezende, Shakir Mohamed, Max Welling, Semi-Supervised Learning with Deep Generative Models, NIPS, 2014