Image Generation

Semester 2, 2022

Kris Ehinger and Tom Drummond

Demos

- https://www.nvidia.com/en-us/research/ai-demos/

- https://www.midjourney.com/ (requires sign up for limited no of uses)

Outline

- Background: Generative models
- Autoencoders

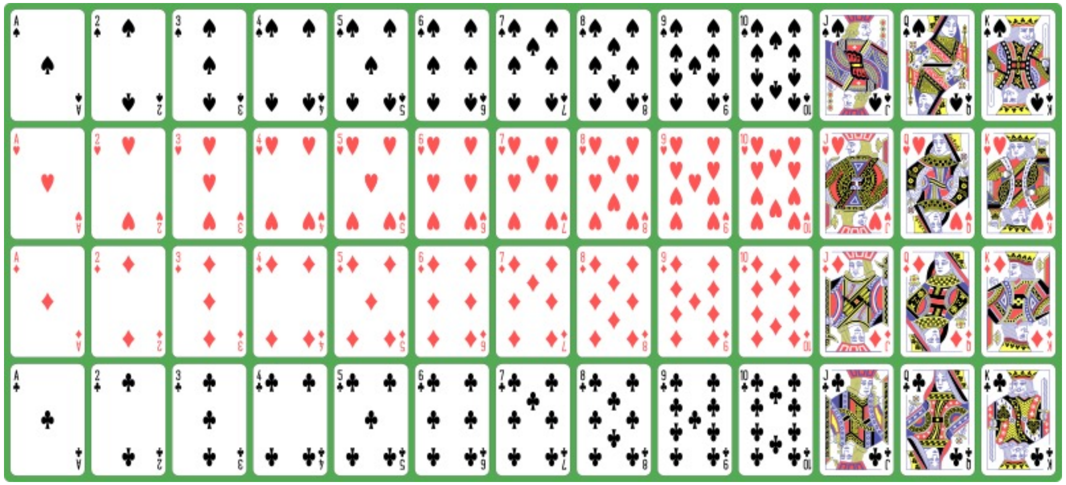
Learning outcomes

- Explain the differences between discriminative and generative models
- Explain how regular and variational autoencoders work, and how they differ from each other

Background: Generative models

Probability notation

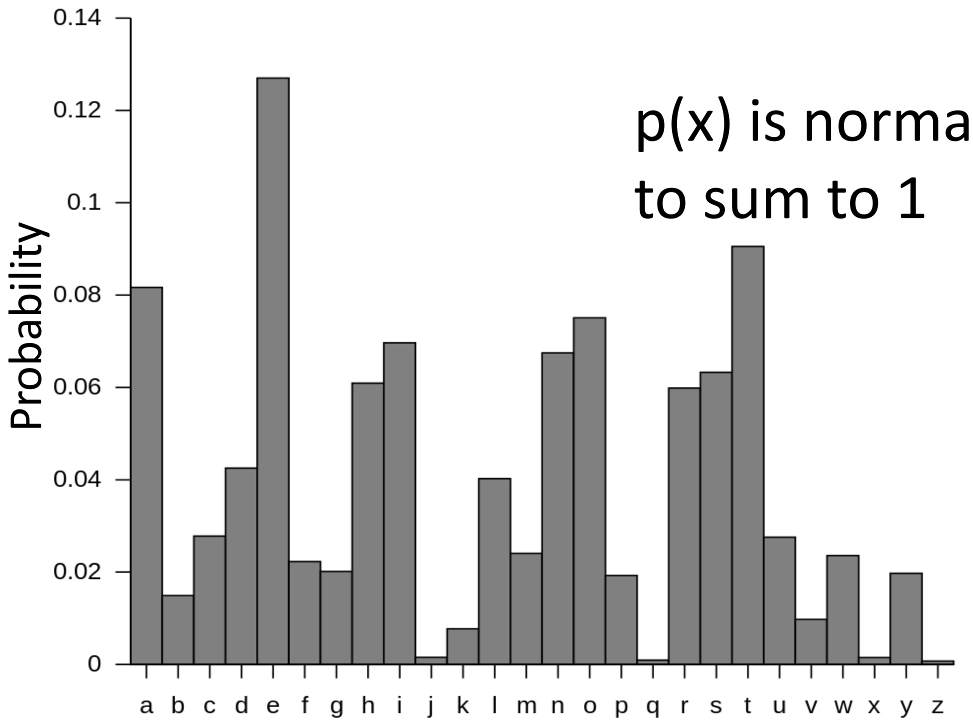
- $P(x)$ = probability of an event x
- Joint probability: $P(x,y)=P(x\cap y)$
  - Probability of both x and y occurring
- Conditional probability: $P(x|y)=\frac{P(x\cap y)}{P(y)}$
  - Probability of x occurring, given y



Probability distribution

- Probability distribution p(x)
  - A function that assigns a non-negative value to each possible x that represents the likelihood of x

Example: Probability distribution for occurrence of letters (a-z) in English text

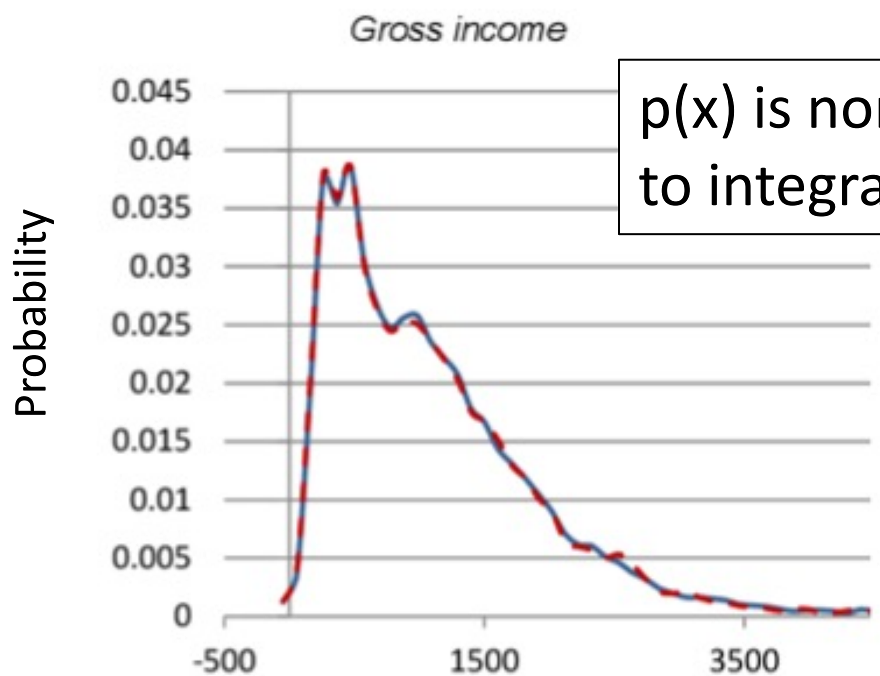


p(x) is normalised to sum to 1

Probability density function

- Probability density function p(x) for continuous x
  - A function that assigns a non-negative value to each possible x that represents the likelihood of x

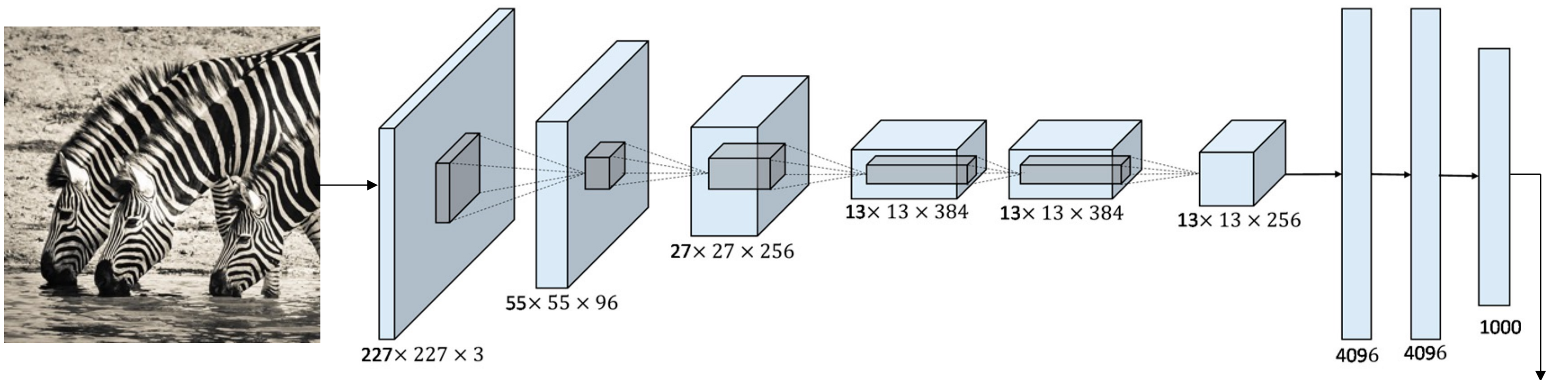
Example: Probability density function for weekly income in Australia



p(x) is normalised to integrate to 1

Discriminative vs. Generative

- **Discriminative** models
  - Learn conditional probability of class Y given attributes X: p(y|x)

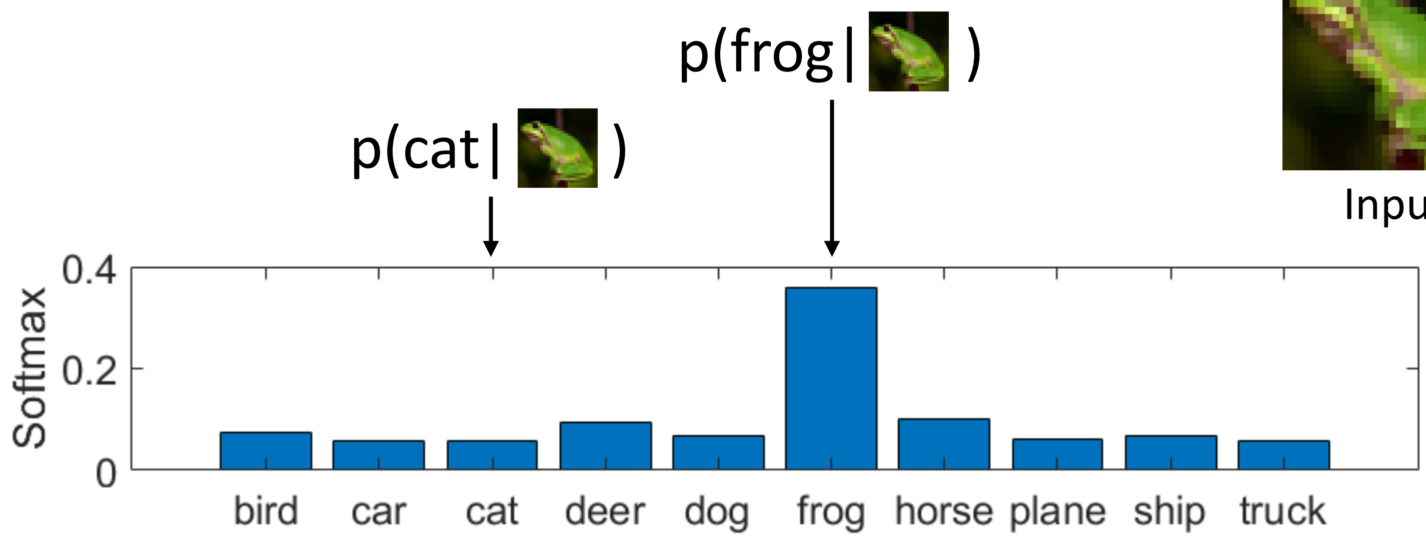


“zebra”

Image: Han, Zhong, Cao, & Zhang (2017)

Discriminative model

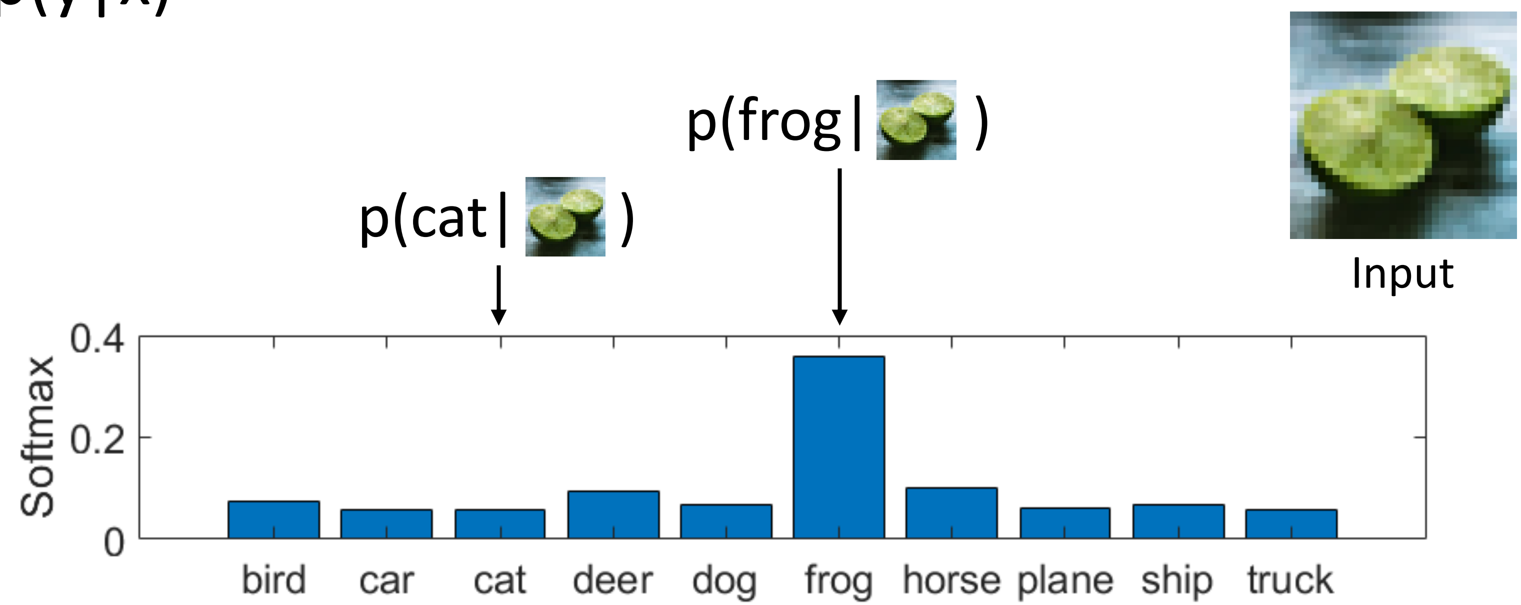
- Input is an image
- Output is a probability density function over labels p(y|x)



Input

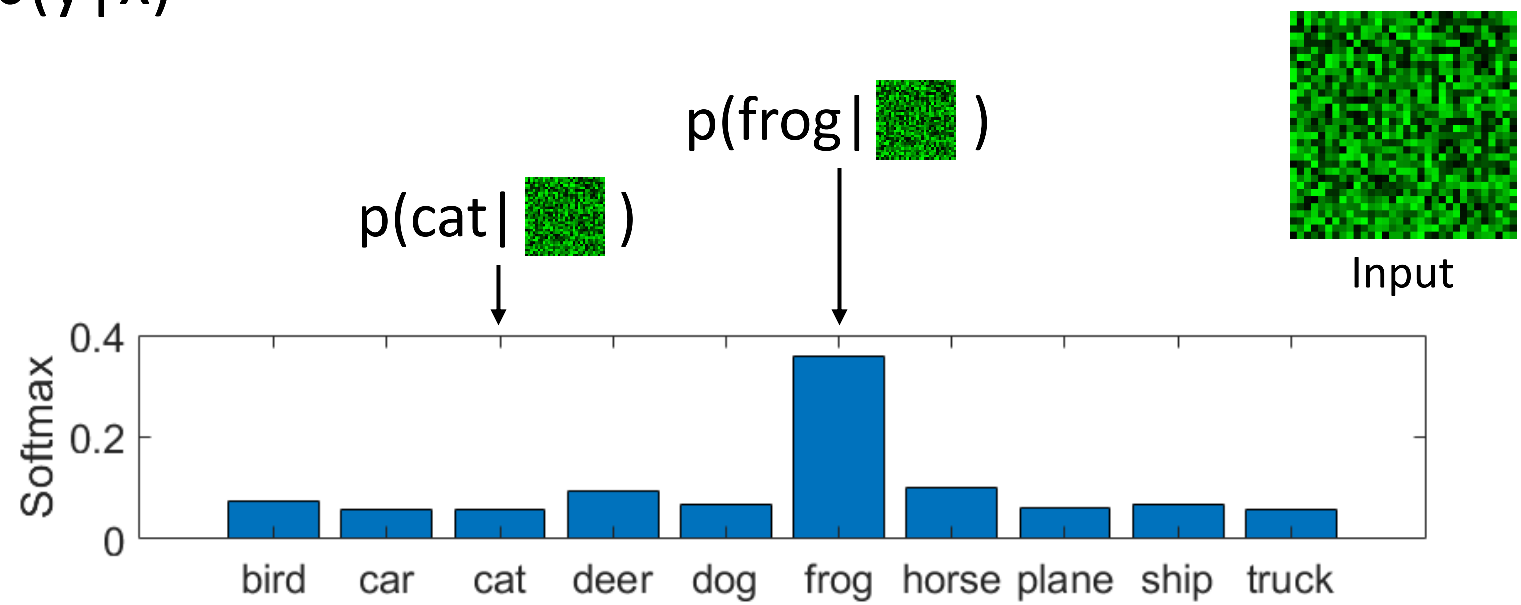
Discriminative model

- Input is an image
- Output is a probability density function over labels p(y|x)



Discriminative model

- Input is an image
- Output is a probability density function over labels p(y|x)

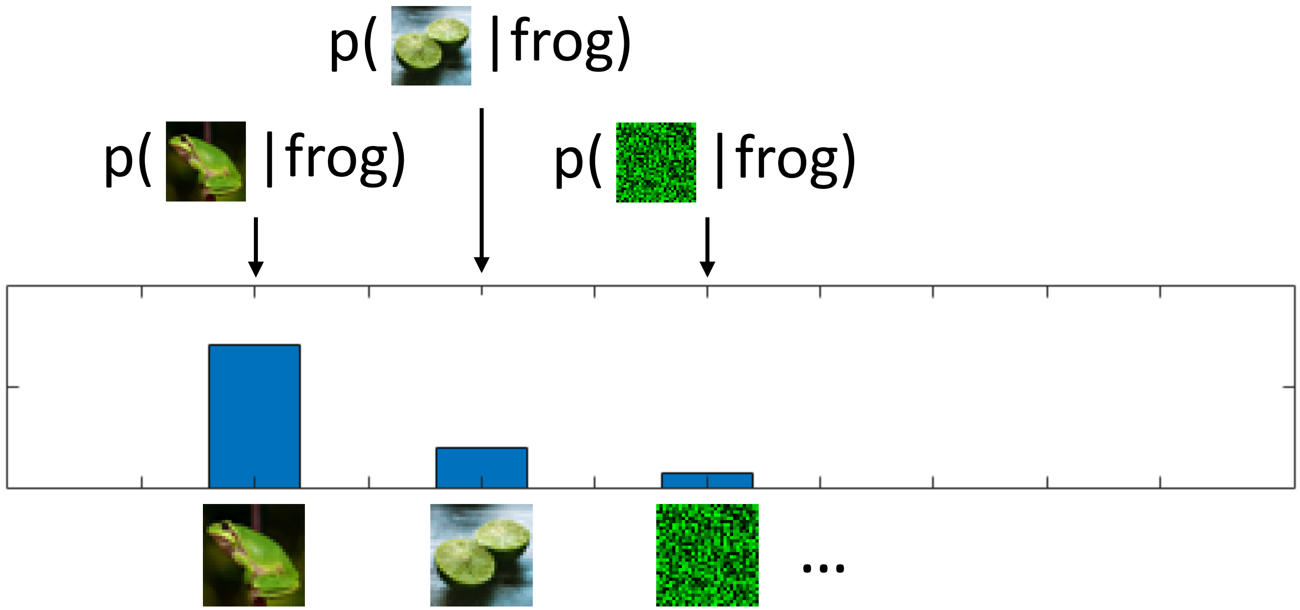


Discriminative vs. Generative

- **Discriminative** models
  - Learn conditional probability of class Y given attributes X: $p(y|x)$
- **Generative** models
  - Learn joint probability of attributes X and class Y: $p(x,y)$
- Generative model contains discriminative model: you can use the joint probability to get $p(y|x)$
- AND generative can do the reverse: $p(x|y)$

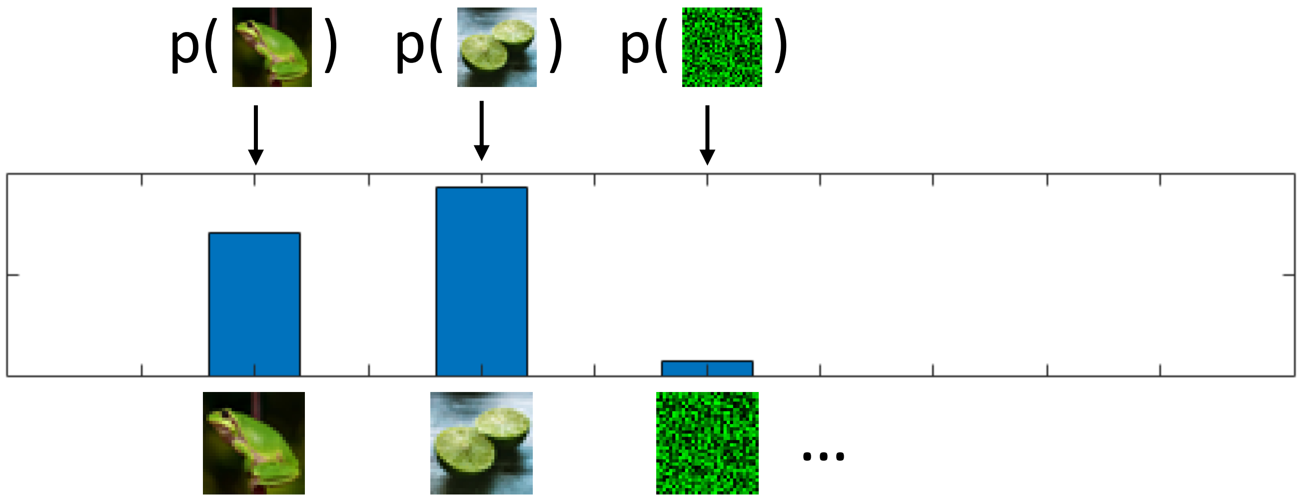
(Conditional) generative model

- Input is a label
- Output is a probability density function over images p(x|y)



(Unconditional) generative model

- Output is a probability distribution p(x)
- What is the probability that this is an image?



Bayes' rule

- Relationships between these models:

Conditional generative model

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

Prior over labels

Discriminative model

Unconditional generative model

Discriminative model

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

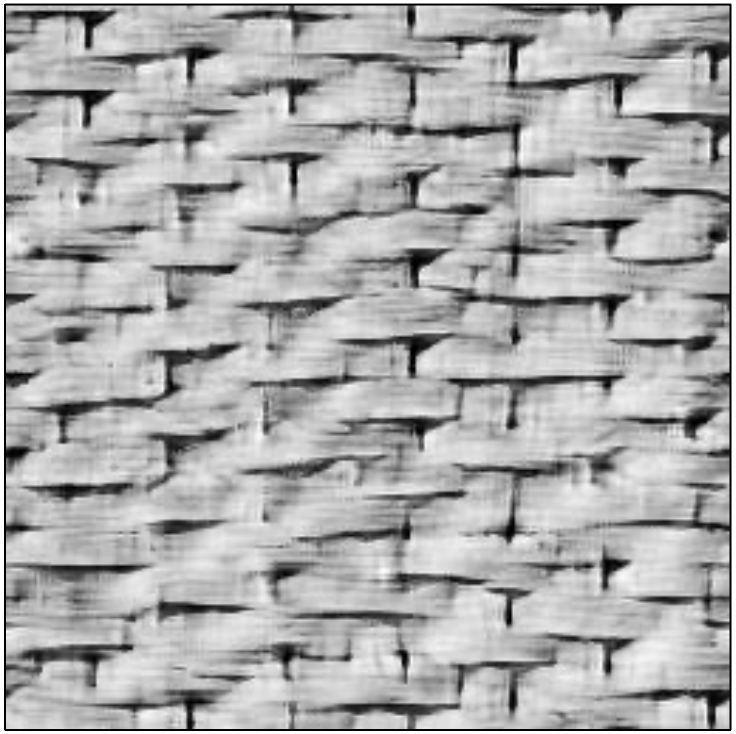
Unconditional generative model

Conditional generative model

Prior over labels

Generative models

- Generative models can generate new samples from the learned distribution



"Basket"
Parametric texture model

Portilla & Simoncelli (2000)
https://www.cns.nyu.edu/~lcv/texture/

Generative models

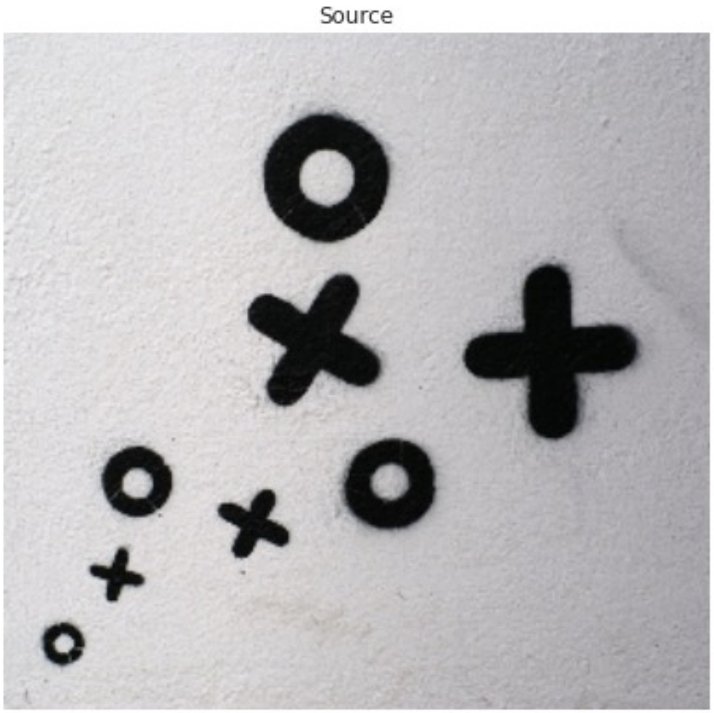
- Generative models can generate new samples from the learned distribution

"Face"
Parametric texture model

Portilla & Simoncelli (2000)
https://www.cns.nyu.edu/~lcv/texture/

Generative models

- Generative models can generate new samples from the learned distribution



Images: http://bethgelab.org/deeptextures/

Summary

- Discriminative models produce a probability distribution over labels, given an image
- Generative models produce a probability distribution over images, given a label (conditional) or in general (unconditional)
- Difficult problem – what makes one set of pixels more probable than another?

Autoencoders

Unsupervised learning

- Learn a model for unlabelled data
- Goal is to find a model that represents the data as well as possible, usually with fewer parameters
- Uses:
  - Simpler model for another ML system
  - Form of dimensionality reduction
  - Potentially better generalization

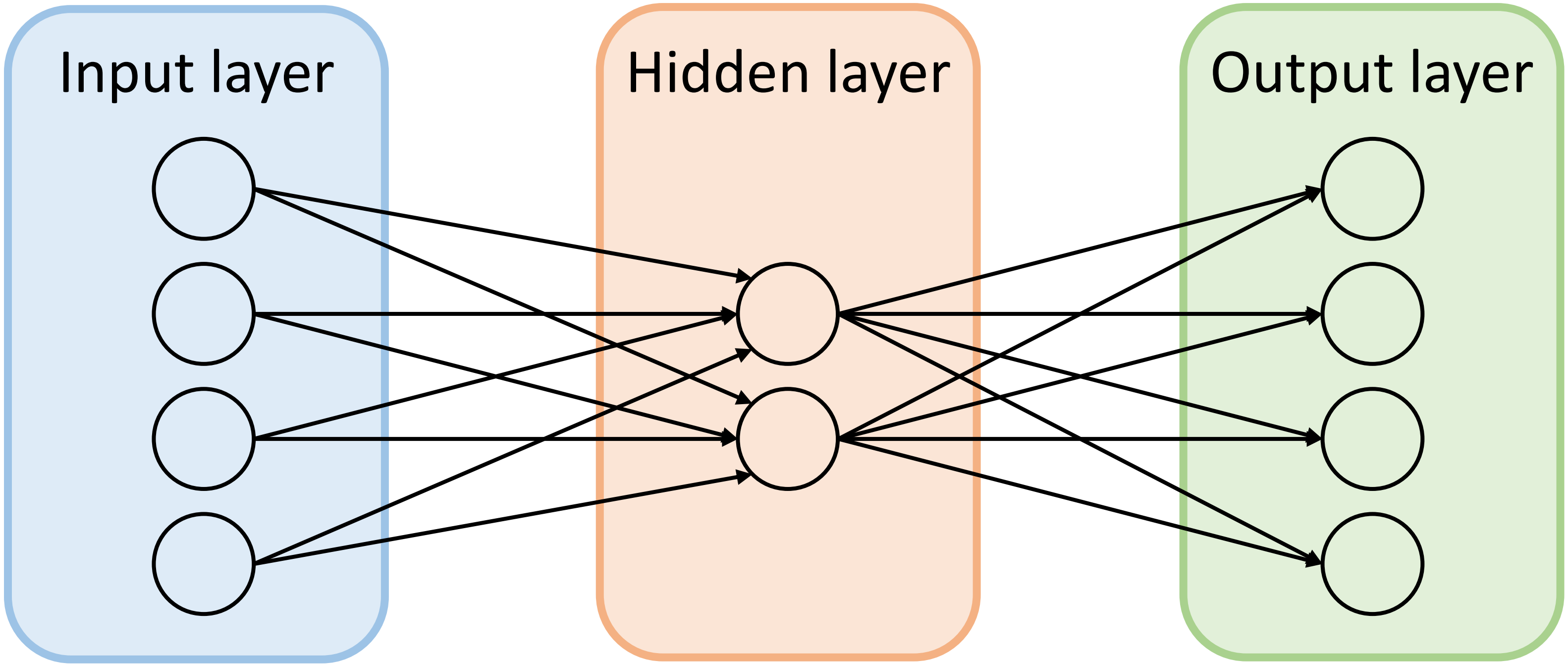
Unsupervised learning + NNs

- Like supervised machine learning algorithms, unsupervised algorithms may not work well in “raw” input spaces (text, images)
- Why?
- Solution? Embeddings (e.g., from neural networks) might work better
  - But we have no labels to learn the embeddings for our task
  - Embeddings learned for a different task may not give a complete representation of the data

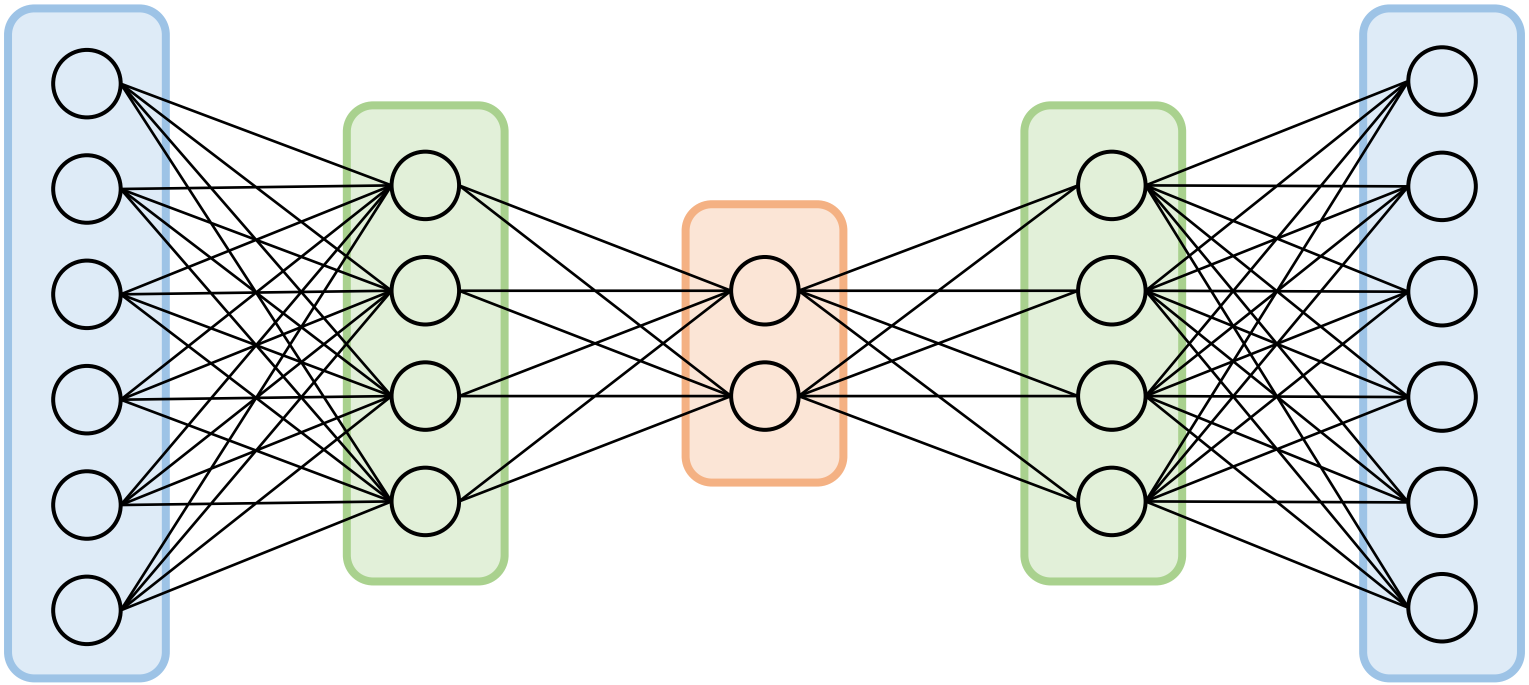
Autoencoders

- Essentially, neural networks for unsupervised learning
- Sometimes called “self-supervised” learning
- Output of the network is whatever was passed to the network (e.g., an image)
- Hidden layer learns a lower-dimensional representation of the input

Basic autoencoder architecture



Deeper autoencoder architecture



---------- Encoder ----------   ---------- Decoder ----------

Autoencoders

- Encoder/decoder architecture
  - **Encode** in a hidden layer
  - Hidden layer is smaller than the input (fewer neurons)
  - **Decode** to an output layer
  - Often the encoding and decoding weights are forced to be the same
- Goal: output the input

Hidden layer

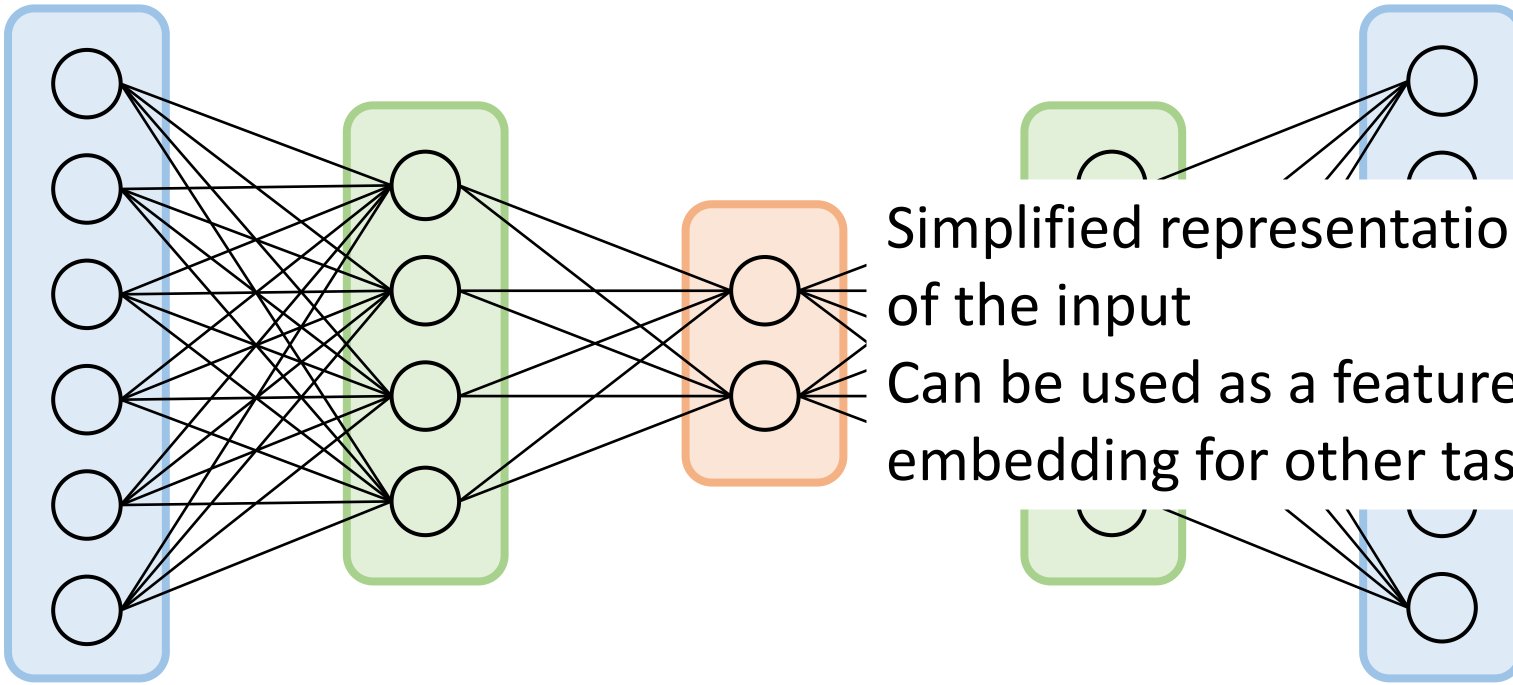
- “Bottleneck” layer – smaller than the input
- Represents the input in terms of **latent variables**
- In the simplest case (one hidden layer with linear activation functions), this layer learns PCA

- Why does this layer need to be smaller than the input?

Output and loss

- Unlike a standard NN, the output is not a class or regression value – it's the same type as the input (e.g., an image)
- Activation function is chosen appropriately:
  - For a binary image, tanh or sigmoid
  - For a regular image, linear activation
- Loss function = difference between input and output (e.g., MSE)

Latent representation



Simplified representation of the input

Can be used as a feature embedding for other tasks

---------- Encoder ----------   ---------- Decoder ----------

Example: Denoising autoencoder

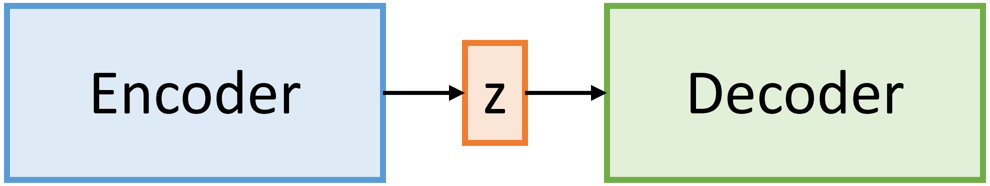
- https://cs.stanford.edu/people/karpathy/convnetjs/demo/autoencoder.html

Autoencoder

- Unsupervised learning (no labels)
- Learns a latent representation from data: lower-dimensional set of features that explains the data
- Not a true generative model – no way to sample new data
  - You could “sample” by giving random latent variable values to the decoder, but no guarantee that these will produce real images

Variational autoencoder (VAE)

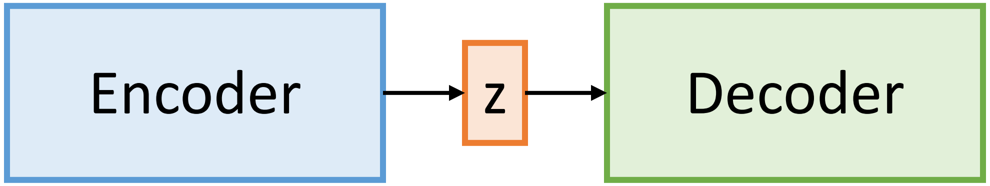
- Probabilistic version of an autoencoder: learn latent representation and sample from the model to generate new images



- Assume images are generated from some distribution over latent variables z
- Assume a simple prior p(z), e.g., uniform or Gaussian distribution

Variational autoencoder (VAE)

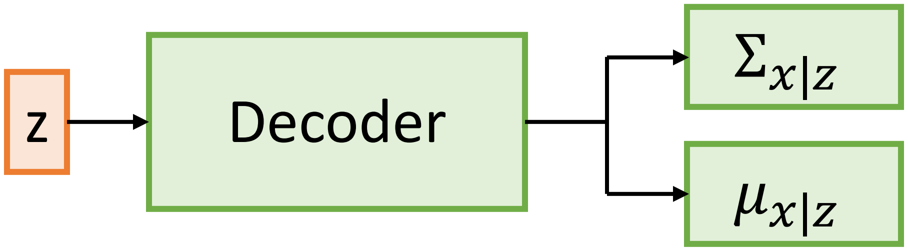
- Probabilistic version of an autoencoder: learn latent representation and sample from the model to generate new images



- Probabilistic decoder learns p(x|z)
- Probabilistic encoder learns p(z|x)
- Goal: maximize the likelihood p(x)

Probabilistic decoder

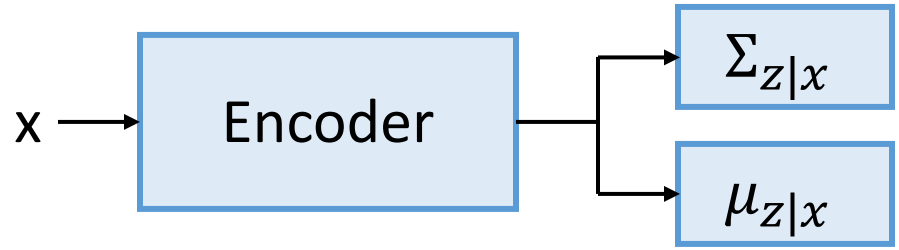
- Input: latent variables z
- Output: mean $\mu_{x|z}$ and diagonal covariance $\Sigma_{x|z}$, parameters of a Gaussian distribution that generates x conditional on z



- $p(x|z) = \mathrm{N}\left(\mu_{x|z}, \Sigma_{x|z}\right)$
- Goal: maximize $p(x) = \dfrac{p(x|z)p(z)}{p(z|x)}$

Probabilistic encoder

- Input: image x
- Output: mean $\mu_{z|x}$ and diagonal covariance $\Sigma_{z|x}$, a Gaussian distribution over latent variables conditional on x

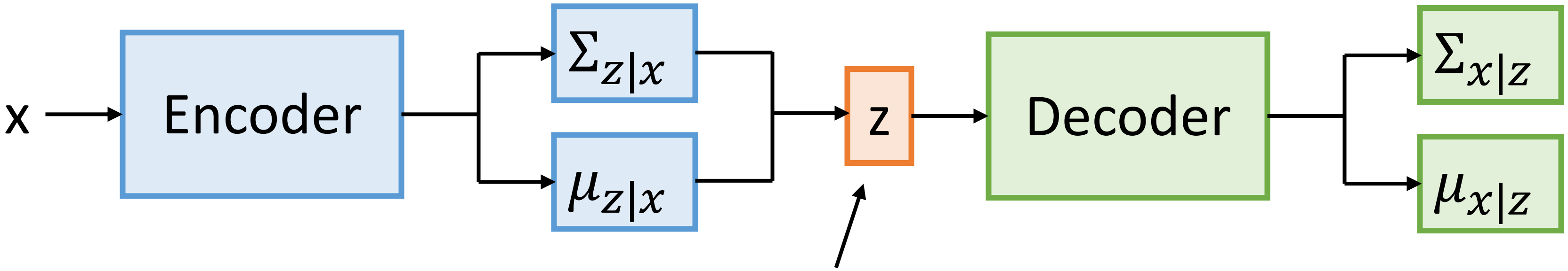


- Learns approximation of $p(z|x)$:

$$q(z|x) = \mathrm{N}\left(\mu_{z|x}, \Sigma_{z|x}\right)$$

Variational autoencoder (VAE)

- Encoder and decoder are concatenated and trained jointly



$z = \Sigma_{z|x}\alpha + \mu_{z|x}$

α is a random variable from the standard normal distribution $\mathrm{N}(0,1)$

Loss function

- Goal: maximize likelihood p(x)
- Loss is based on variational lower bound on p(x):

$$\log(p(x)) \geq E_{z\sim q(z|x)}[\log(p(x|z))] - D_{KL}\big(q(z|x), p(z)\big)$$

Log likelihood of image x	Reconstruction quality: expected log likelihood of Decoder's reconstruction, with respect to Encoder's distribution over inputs	Kullback-Leibler divergence between the Encoder's estimate of p(z|x) and our prior p(z)=standard normal distribution

Loss function

- Loss consists of two terms: reconstruction loss and regularisation loss
- Reconstruction loss: encourages network to create output images as similar as possible to input images
- Regularisation loss: encourages network to learn a latent representation z that is similar to the prior (standard normal distribution)

Properties of the latent space

- To be useful for generation, the latent space should be:
  - Continuous: nearby points in the latent space correspond to similar images
  - Complete: every point in the latent space corresponds to a valid image
- Standard normal distribution satisfies both of these requirements
- Use of diagonal covariance matrices ensures latent variables are independent

Application: Sampling new images

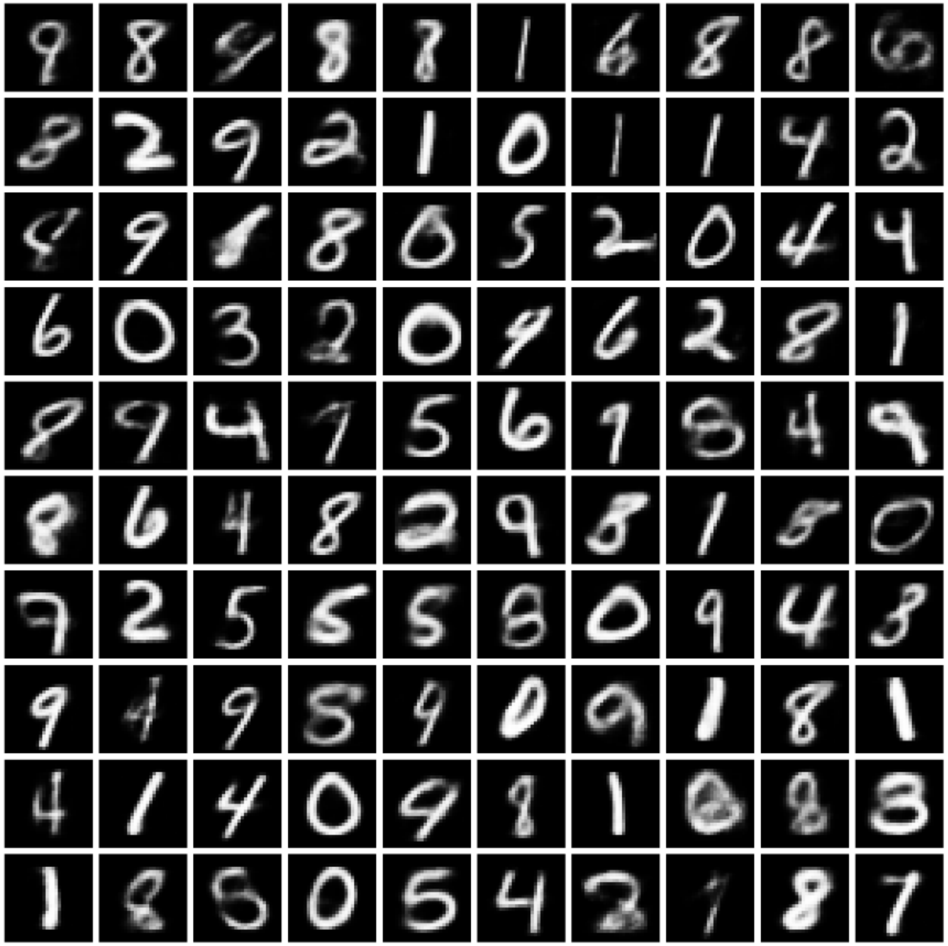


Image: Doersch (2016)

Application: Sampling new images

Image: Hou, Shen, Sun, & Qiu (2017)

Latent space visualisation

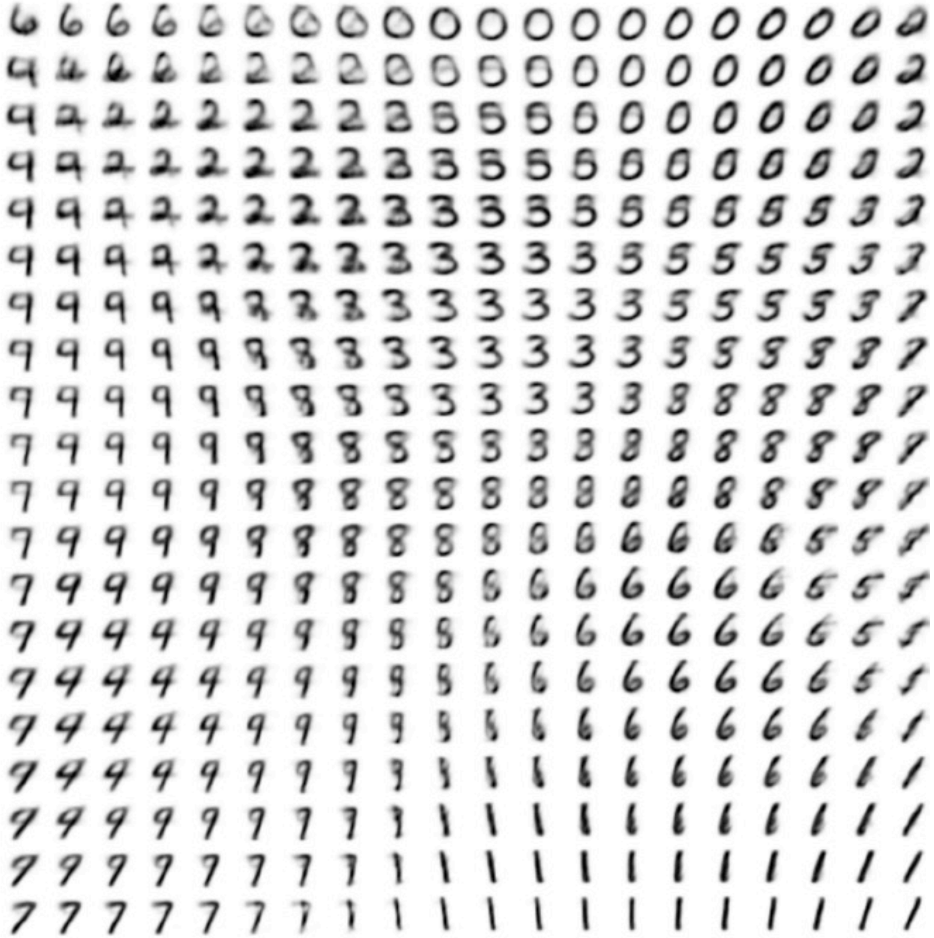


Image: Kingma & Welling (2014)

Application: Image manipulation

- Given the latent-variable representation z of image x, can change values of z to create variations on x
- Nearby points in latent space correspond to similar images (continuity requirement) and axes are independent
- But directions in latent space may not correspond to recognizable image properties (without additional constraints)

Image: Kulkarni, Whitney, Kohli, & Tenenbaum (2015)

Application: Image manipulation

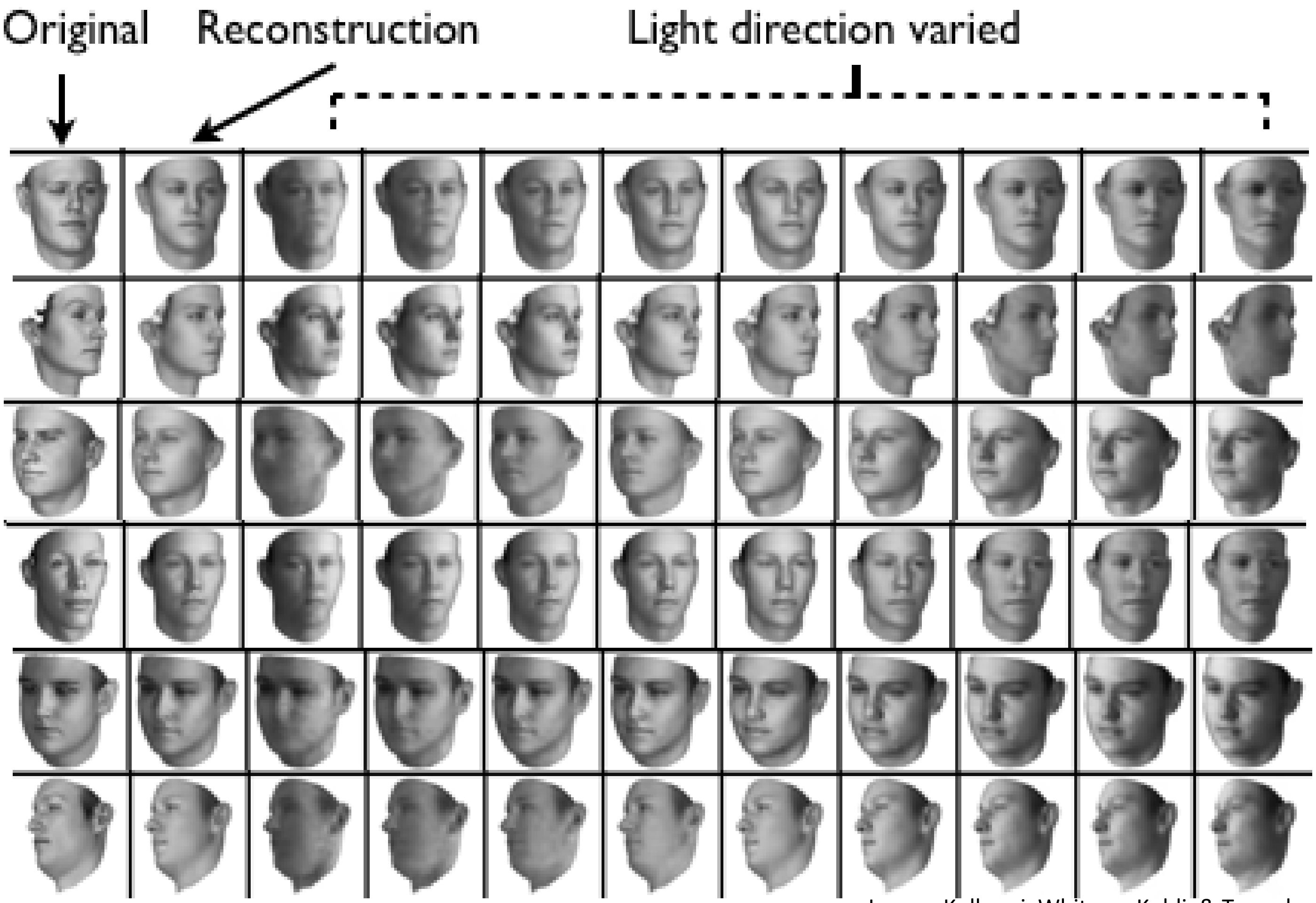


Image: Kulkarni, Whitney, Kohli, & Tenenbaum (2015)

Application: Image manipulation

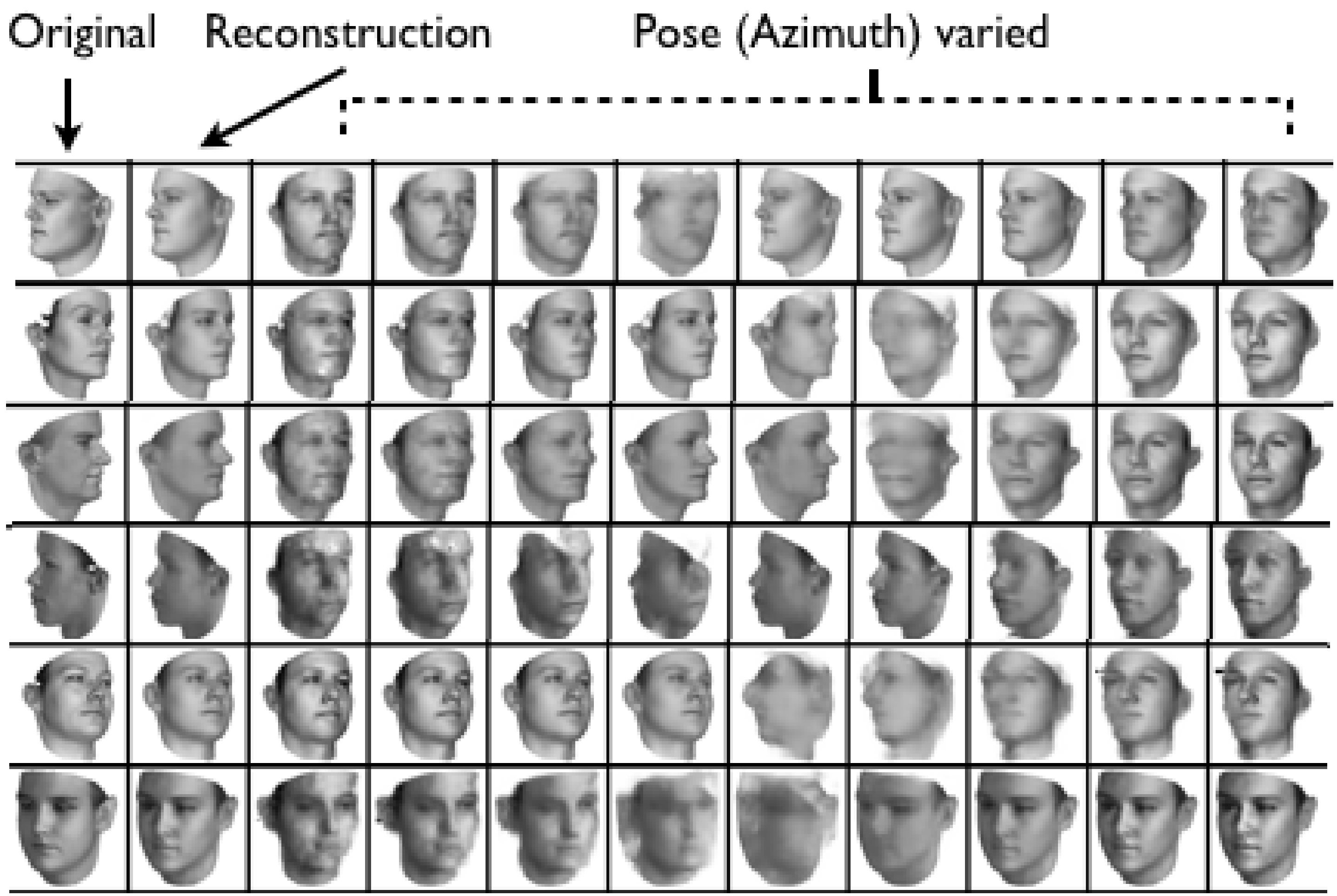


Image: Kulkarni, Whitney, Kohli, & Tenenbaum (2015)

Variational autoencoder (VAE)

- Advantages
  - Learns approximations of $p(z|x)$ and $p(x|z)$, where z is a latent variable representation of the input x
  - Can be used to generate new instances of x
- Disadvantages
  - Outputs often blurry (why?)

Final note

- More recent VAEs use better image representations to reduce blur

Image: Razavi, van den Oord, & Vinyals (2019)

Summary

- Discriminative models predict labels from images, generative models predict the probability distribution of images
- Autoencoders assume images can be generated from a low-dimensional space of latent variables
- Regular autoencoder – learns latent representation to reconstruct images
- Variational autoencoder – probabilistic version of autoencoder, can sample from the latent space