# Visual AI

CPSC 533R

**Lecture 9.** Representation learning

Helge Rhodin

# Discriminative vs. Generative

Disc.: ‘reconstruct a given image’

- Supervised learning
  - classification
  - regression
  - SVM, decision trees, …

Gen.:‘model distribution of images’

- GAN
- VAE
- PCA

‘generate images’?

‘… and their reconstruction’

- analysis by synthesis

Probabilistic definition

- conditional probability distribution p(y|x)
  - density networks, …

3D pose

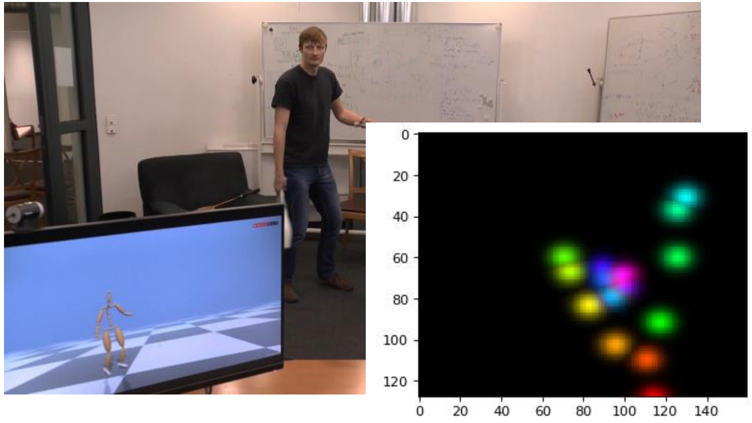
image

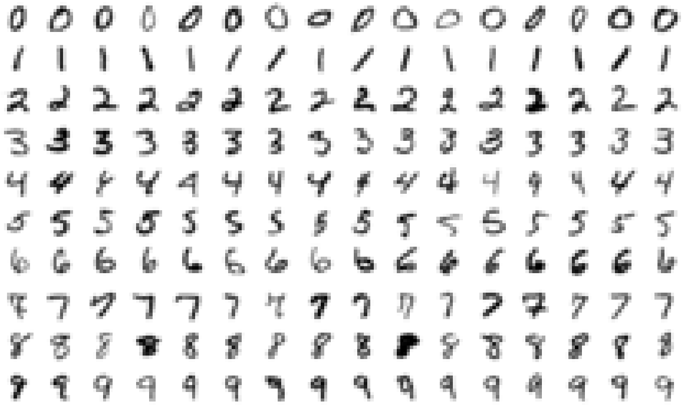
- joint probability distribution p(x,y)
  - e.g, p(x,y) = p(x|y) p(y)

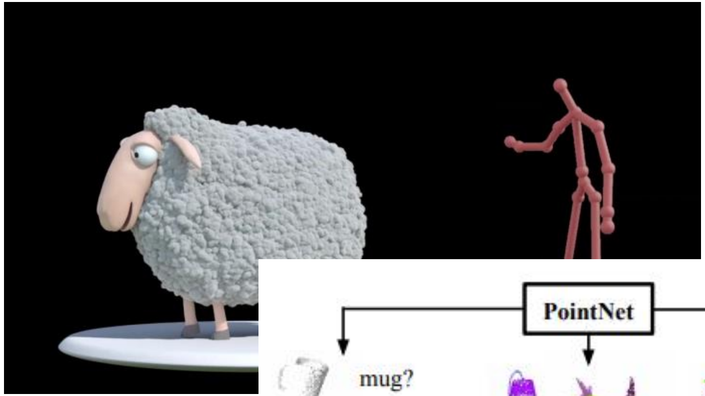
likelihood of image x given pose y

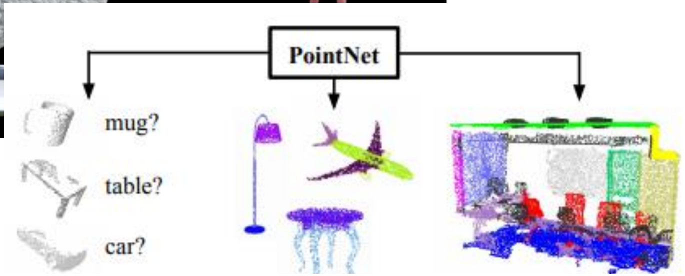
prior over all possible 3D poses

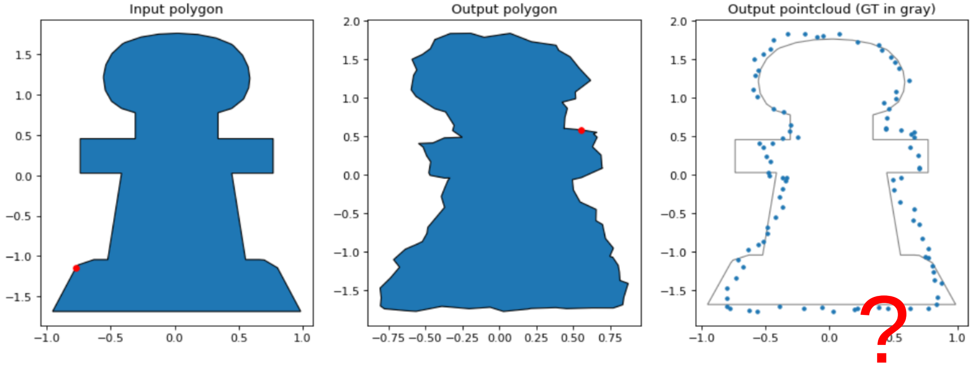
# Discriminative models covered in class



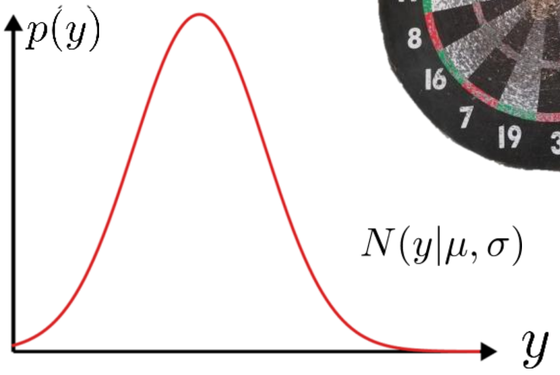




# Probabilistic interpretation of least squares regression

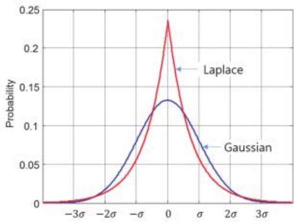
**Regression:** minimize the negative log-likelihood




Likelihood: $$N(y|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(y-\mu)^2}{2\sigma^2}}$$

Negative log likelihood:
(simplified) $$E = \frac{1}{2\sigma^2}(y - f_\theta(x))^2$$

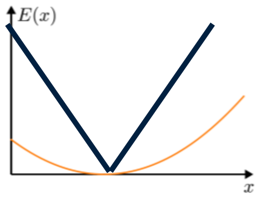
Likelihood




Gaussian distribution $\exp(-(y-\mu)^2)$

Laplace distribution $\exp(-|y-\mu|)$

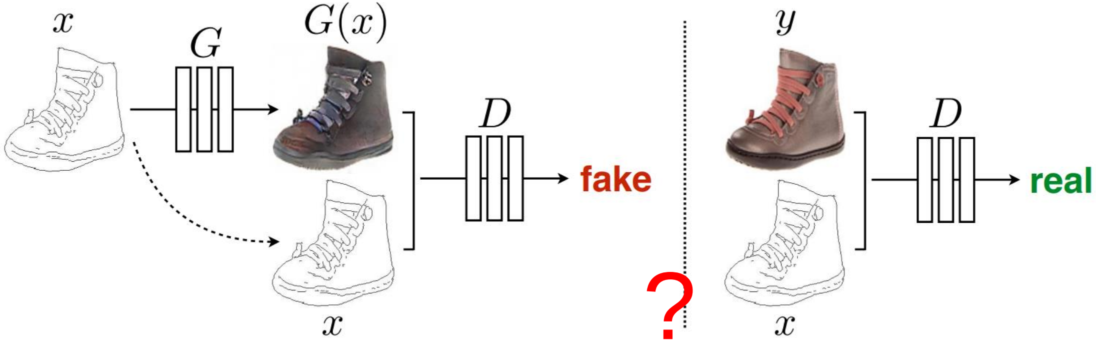
Error functions
(negative log likelihood)




Mean squared error (MSE) $(y-\mu)^2$

Mean absolute error (MAE) $|y-\mu|$

# Generative models covered

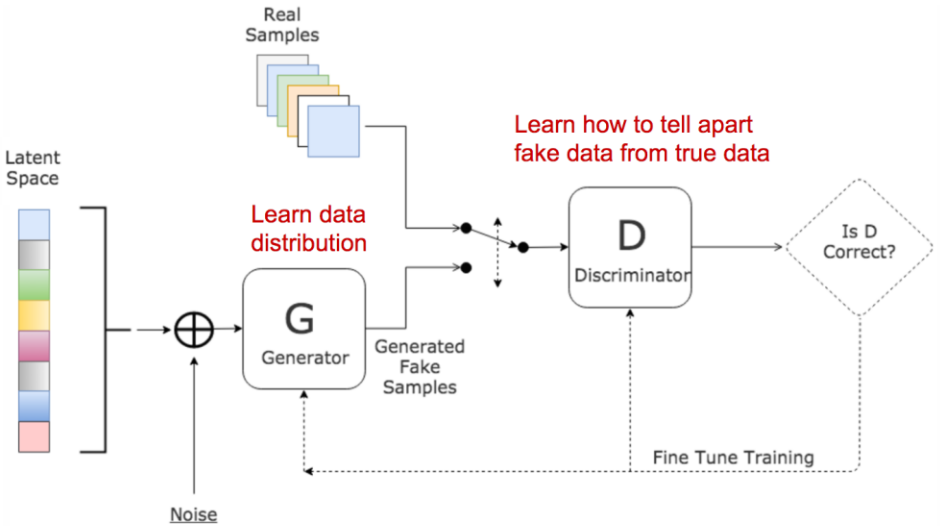



?

# Recap: GAN concept

**Goal:** Train a generator, G, that produces naturally looking images

**Idea:** Train a discriminator, D, that distinguishes between real and fake images. Use this generator to train G




https://www.kdnuggets.com/2017/01/generative-adversarial-networks-hot-topic-machine-learning.html

# From classical (JS) to Wasserstein GAN

Diverse measures exist to compare probability distributions (here generated and real image distribution)

- The *Total Variation* (TV) distance

$$\delta(\mathbb{P}_r, \mathbb{P}_g) = \sup_{A \in \Sigma} |\mathbb{P}_r(A) - \mathbb{P}_g(A)| \ .$$

- The *Kullback-Leibler* (KL) divergence

$$KL(\mathbb{P}_r \| \mathbb{P}_g) = \int \log\left(\frac{P_r(x)}{P_g(x)}\right) P_r(x) d\mu(x) \ ,$$

- The *Jensen-Shannon* (JS) divergence

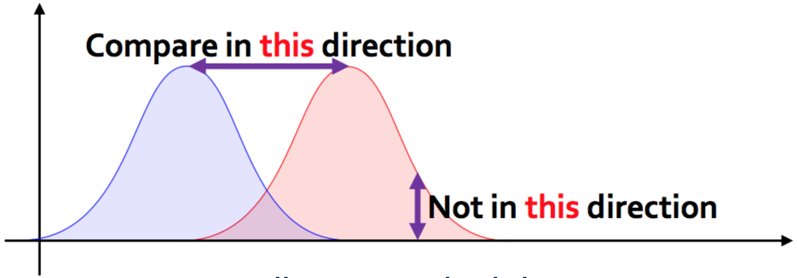
$$JS(\mathbb{P}_r, \mathbb{P}_g) = KL(\mathbb{P}_r \| \mathbb{P}_m) + KL(\mathbb{P}_g \| \mathbb{P}_m) \ ,$$

where $\mathbb{P}_m = (\mathbb{P}_r + \mathbb{P}_g)/2$

*JS is what the classical GAN optimizes*

- The *Earth-Mover* (EM) distance or Wasserstein-1

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y) \sim \gamma} \big[ \|x - y\| \big] \ ,$$




EM distance principle

[Arjovsky et al., Wasserstein GAN. 2017]

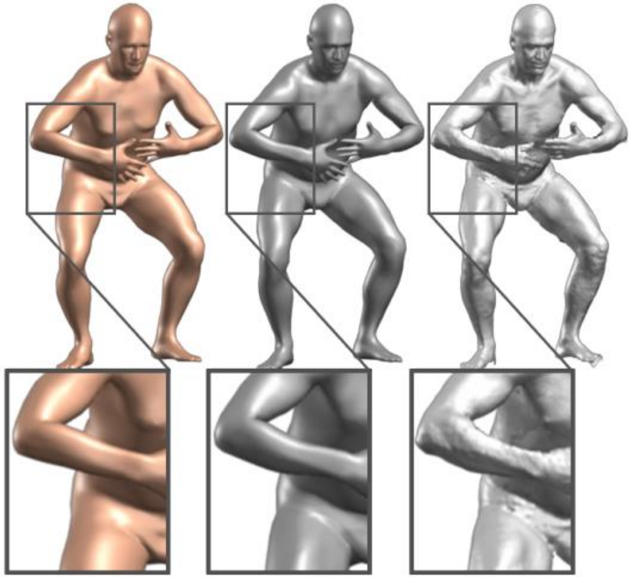
# **SMPL:** A Skinned Multi-Person Linear Model

Consists of

- a mesh template
- a low-dimensional set of deformation parameters
- a skeleton rig

It can be used as a generative model

- prior over shape deformations
- prior over joint angles
- likelihood of point cloud compared to model parameterized by angles and shape weights

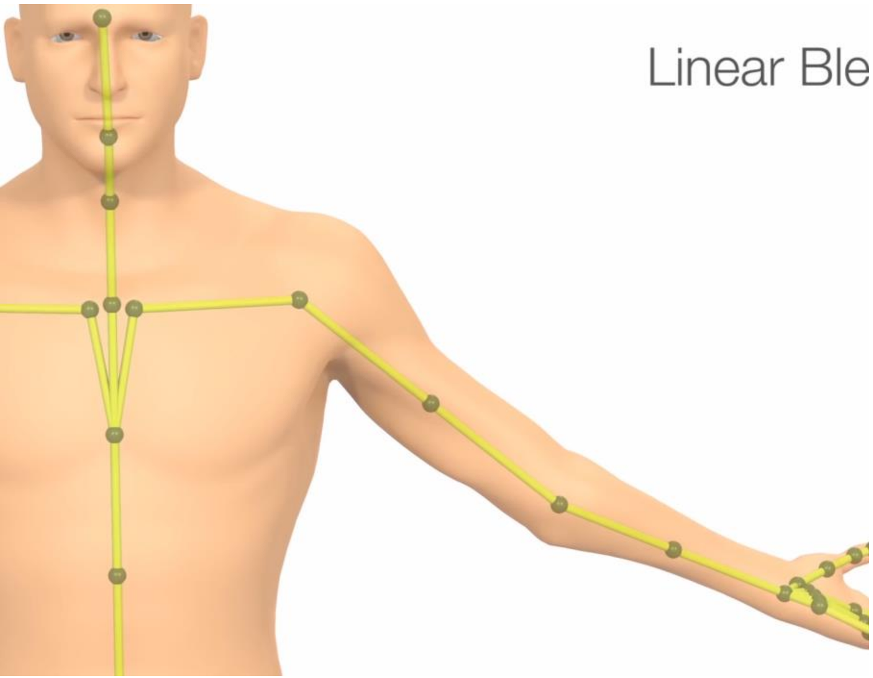


$$p(x,y) = p(x|y)\, p(y)$$

point cloud/mesh

joint & shape parameters

# Skinning

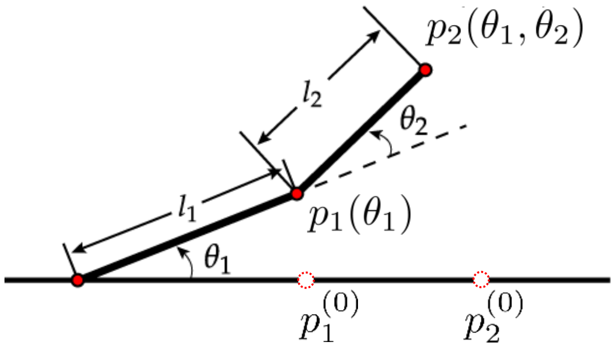



# Forward kinematics, linear or not?

Forward kinematics

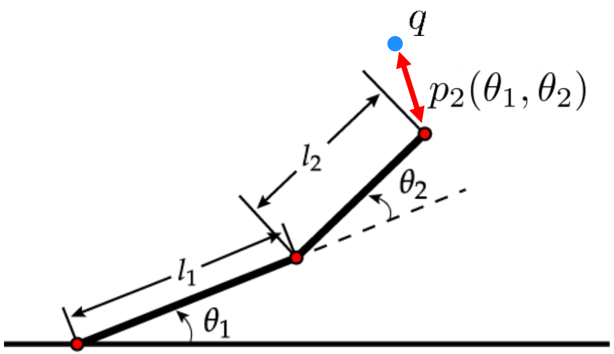
- non-linear in the angle (due to cos and sin)

$$R_1 = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 \\ \sin\theta_1 & \cos\theta_1 \end{bmatrix} \qquad R_2 = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 \\ \sin\theta_2 & \cos\theta_2 \end{bmatrix}$$




- linear/affine given a set of rotation matrices

$$p_2(\theta_1, \theta_2) = R_1 p_1^{(0)} + R_2 R_1 \left(p_2^{(0)} - p_1^{(0)}\right)$$




Inverse kinematics

- minimize objective to reach goal location $q$

$$O(\theta_1, \theta_2) = \|q - p_2(\theta_1, \theta_2)\|$$

- difficult, due to nonlinear dependency on theta

Bonus: implement IK yourself (perhaps use PyTorch?)
https://rgl.s3.eu-central-1.amazonaws.com/media/pages/hw4/CS328_-_Homework_4_3.ipynb

# Recap: Surface mesh

Representation: Vertices connected by edges forming faces (usually triangles)

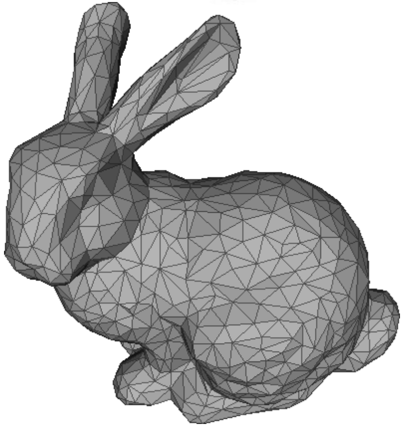
- Size: N x D + F x 3 (N: # points, D: space dimension, F: #triangles)
- A 3D surface parametrization (can be higher-dimensional)
    - Piece-wise linear with adaptive detail; triangle faces are usual

Benefits

- Good for single and multi-view reconstruction
- Provides orientation information (surface normal)
- Graph convolutions possible

Drawbacks

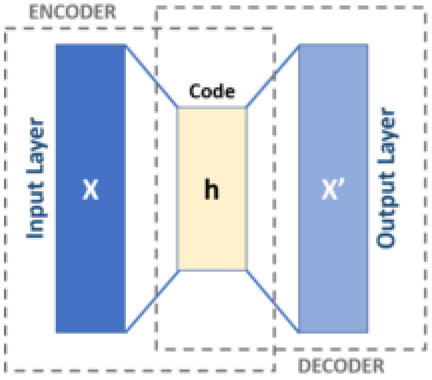
- Irregular structure (number of neighbors, edge length, face area)
- Difficult to change topology
  (shape changes require to create new vertices and edges)



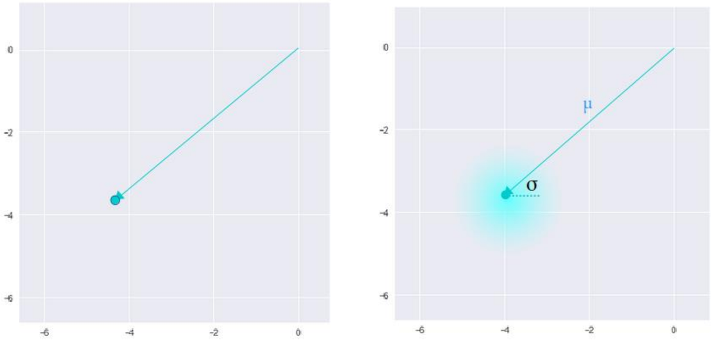
# Short break

Go through the list of presentations and classify them into generative or discriminative approaches!

# Variational Autoencoder (VAE) concept




- mapping to a latent variable distribution
  - a parametric distribution
    - usually a Gaussian
      - with variable mean and std parameters
  - impose a prior distribution on the latent variables
    - usually a Gaussian
      - with fixed mean=0 and std=1
- Enables the generation of new samples
  - draw a random sample from the prior
    - pass it through the decoder
  - or draw a sample from the posterior
    - pass it through the decoder




Standard Autoencoder
(direct encoding coordinates)

Variational Autoencoder
(μ and σ initialize a probability distribution)

https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf

# VAE examples

Generating unseen faces

https://github.com/yzwxx/vae-celebA

Generating music

[Roberts et al., Hierarchical Variational Autoencoders for Music]

# The Variational Autoencoder (VAE)

VAE Objective (general)

Regularizer / prior term

$$\mathcal{L}(\phi, \theta, \mathbf{x}) = -\mathbf{E}_{\mathbf{h} \sim q_\phi(\mathbf{h}|\mathbf{x})} \big( \log p_\theta(\mathbf{x}|\mathbf{h}) \big) + D_{\mathrm{KL}}(q_\phi(\mathbf{h}|\mathbf{x}) \| p(\mathbf{h}))$$

Expectation over q

Data term / log likelihood

Kullback–Leibler divergence (relative entropy)

- a dissimilarity measure between distributions
  - not symmetric, KL(p,q) != KL(q,p)
- Definition for continuous distributions

$$D_{\mathrm{KL}}(P \parallel Q) = \int_{-\infty}^{\infty} p(x) \log \left( \frac{p(x)}{q(x)} \right) dx$$

probability density of Q

Common parametrization

- Normal distributions

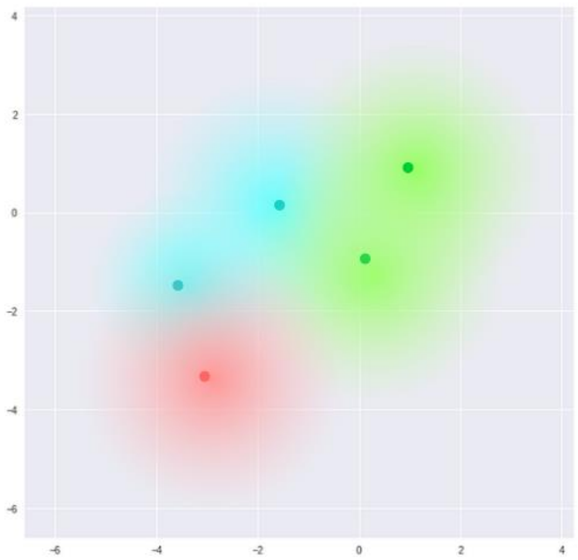
$$p(\mathbf{h}) = \mathcal{N}(0, 1)$$
$$q_\phi(\mathbf{h}|\mathbf{x}) = \mathcal{N}(\boldsymbol{e}(\mathbf{x}), \boldsymbol{\omega}(\mathbf{x})\mathbf{I})$$
$$p_\theta(\mathbf{x}|\mathbf{h}) = \mathcal{N}(\boldsymbol{d}(\mathbf{h}), \boldsymbol{\sigma}\mathbf{I})$$

- parametrized by neural networks
  - encoder e
  - decoder d

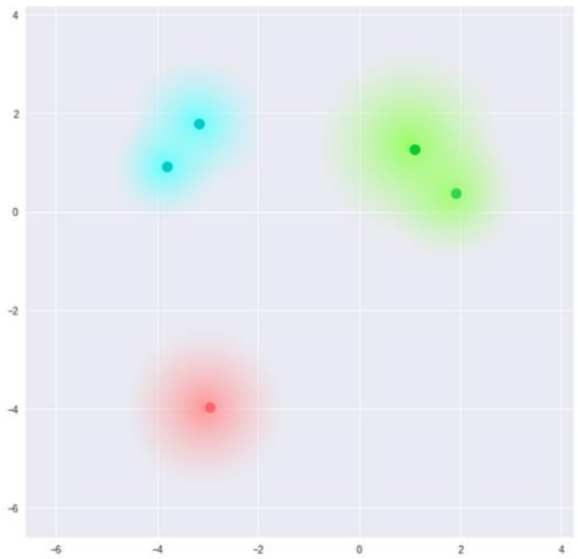
# The effect of the prior

Create a dense and smooth latent space

- without holes
    - all samples will make sense
        - e.g., will reconstruct to plausible images



with prior

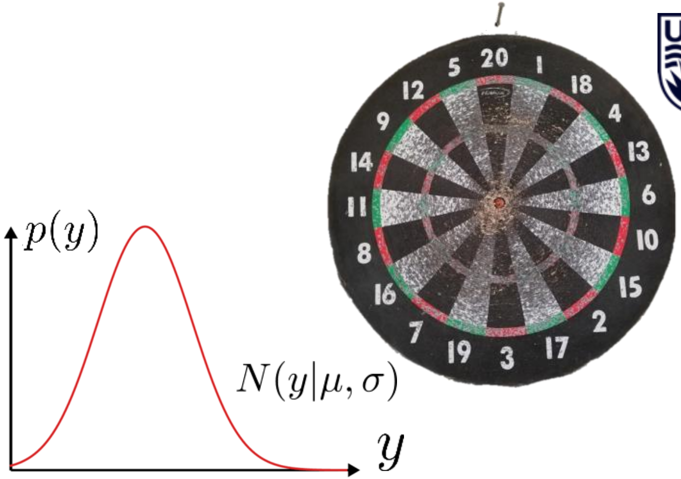


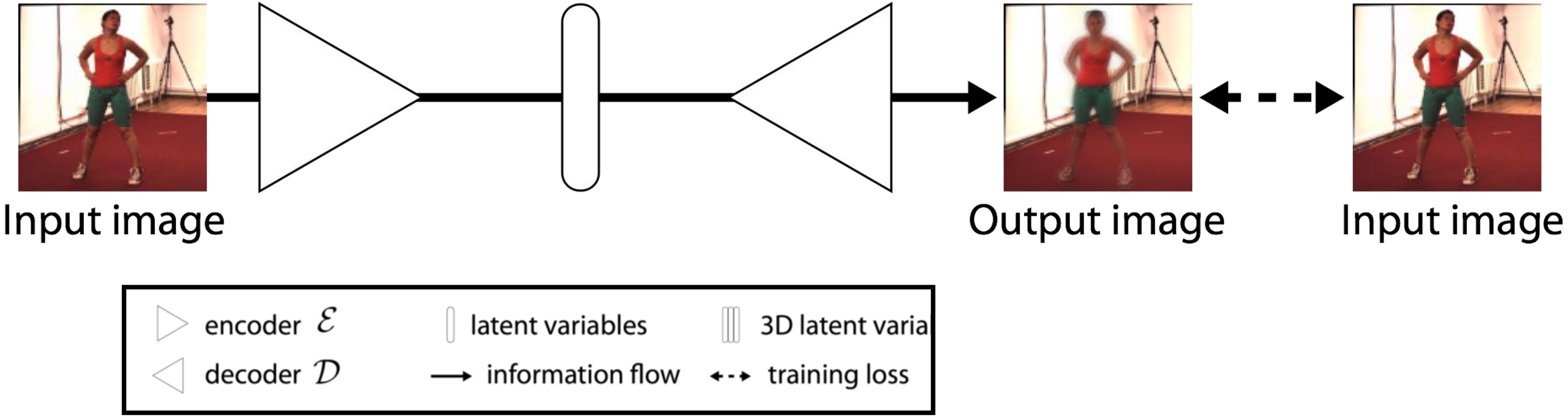
without prior

# The effect of the data term

A reconstruction loss

- similarity of prediction to the target
  - input to output similarity for an auto encoder



# The Variational Autoencoder (VAE), simplified I

VAE Objective (general)

$$\mathcal{L}(\phi, \theta, \mathbf{x}) = -\mathbf{E}_{\mathbf{h}\sim q_\phi(\mathbf{h}|\mathbf{x})}\big(\log p_\theta(\mathbf{x}|\mathbf{h})\big) + D_{\mathrm{KL}}(q_\phi(\mathbf{h}|\mathbf{x})\|p(\mathbf{h}))$$

$$\mathcal{N}(\mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

$$\log\left(\mathcal{N}(\mu, \sigma)\right) = \log\left(\frac{1}{\sigma\sqrt{2\pi}}\right) - \frac{1}{2\sigma^2}(x-\mu)^2$$

$$p_\theta(\mathbf{x}|\mathbf{h}) = \mathcal{N}(\boldsymbol{d}(\mathbf{h}), \boldsymbol{\sigma}\mathbf{I})$$

$$\Leftrightarrow \mathcal{L}(\phi, \theta, \mathbf{x}) = \mathbf{E}_{\mathbf{h}\sim q_\phi(\mathbf{h}|\mathbf{x})}\left(\frac{1}{2\sigma^2}(\mathbf{x} - \mathbf{d}(h))^2\right) + D_{\mathrm{KL}}(q_\phi(\mathbf{h}|\mathbf{x})\|p(\mathbf{h})) + C$$

$$\Leftrightarrow \mathcal{L}(\phi, \theta, \mathbf{x}) = \lambda\mathbf{E}_{\mathbf{h}\sim q_\phi(\mathbf{h}|\mathbf{x})}(\mathbf{x} - \mathbf{d}(h))^2 + D_{\mathrm{KL}}(q_\phi(\mathbf{h}|\mathbf{x})\|p(\mathbf{h})) + C$$

A simple autoencoder reconstruction loss,
the squared difference between input and output

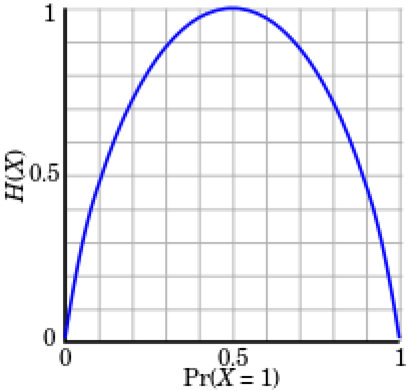
# Self-study: Kullback–Leibler divergence and entropy

Definition

$$D_{\mathrm{KL}}(P \parallel Q) = \int_{-\infty}^{\infty} p(x) \log\left(\frac{p(x)}{q(x)}\right) dx$$

Interpretation

- information gain achieved if Q is used instead of P
- relative entropy
  - Entropy: $H(p) = -\sum_{i=1}^{n} p(x_i) \log p(x_i).$
- the expected number of extra bits required to code samples from P using a code optimized for Q rather than the code optimized for P



# KL divergence between Normal distributions (univariate case)

The KL divergence can be split in two parts

$$D_{\mathrm{KL}}(P \parallel Q) = \boxed{\int_{-\infty}^{\infty} p(x) \log\left(\frac{p(x)}{q(x)}\right)} dx = \int_{-\infty}^{\infty} p(x) \log\left(p(x)\right)\, dx - \int_{-\infty}^{\infty} p(x) \log\left(q(x)\right)\, dx$$

For Gaussians $p(x) = N(\mu_1, \sigma_1)$ and $q(x) = N(\mu_2, \sigma_2)$ it holds

$$\int p(x) \log q(x) dx = -\frac{1}{2}\log(2\pi\sigma_2^2) - \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2}$$

Hence

$$KL(p,q) = -\frac{1}{2}\log(2\pi\sigma_1^2) - \frac{1}{2} + \frac{1}{2}\log(2\pi\sigma_2^2) + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2}$$

$$= \log\frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} - \frac{1}{2}$$

$$\boxed{= -\log\sigma_1 + \frac{\sigma_1^2 + \mu_1^2}{2} - \frac{1}{2}}$$

Using that $m_2$=0 and $s_2$=1

https://stats.stackexchange.com/questions/7440/kl-divergence-between-two-univariate-gaussians

# The Variational Autoencoder (VAE), simplified II

Starting point

$$\Leftrightarrow \mathcal{L}(\phi, \theta, \mathbf{x}) = \lambda \mathbf{E}_{\mathbf{h} \sim q_\phi(\mathbf{h}|\mathbf{x})} \left(\mathbf{x} - \mathbf{d}(\mathbf{h})\right)^2 + D_{\mathrm{KL}}(q_\phi(\mathbf{h}|\mathbf{x}) \| p(\mathbf{h})) + C$$

Simplification (for Gaussian prior p and Gaussian q with $\mu_1 = \boldsymbol{e}(\mathbf{x})$ and $\sigma_1 = \boldsymbol{\omega}(\mathbf{x})$)

Data term / log likelihood

$$\Leftrightarrow \mathcal{L}(\phi, \theta, \mathbf{x}) = \lambda \mathbf{E}_{\mathbf{h} \sim q_\phi(\mathbf{h}|\mathbf{x})} \left(\mathbf{x} - \mathbf{d}(\mathbf{h})\right)^2 + -\log \sigma_1 + \frac{\sigma_1^2 + \mu_1^2}{2} + C'$$

Estimating the expectation by sampling (here a single sample)

$$\approx \lambda \left(\mathbf{x} - \mathbf{d}(h)\right)^2 + -\log \sigma_1 + \frac{\sigma_1^2 + \mu_1^2}{2} + C' \text{ with } h \sim q_\phi$$

*reconstruct the image*

*'keep sigma > 0'*

*'keep sigma and mu small'*

**Expected value**

$$\mathrm{E}_{x \sim q} f(x) = \int q(x) f(x)\, dx$$
$$= \frac{1}{K} \sum_{i=1}^{K} f(x_i) \text{ with } x_i \sim q$$

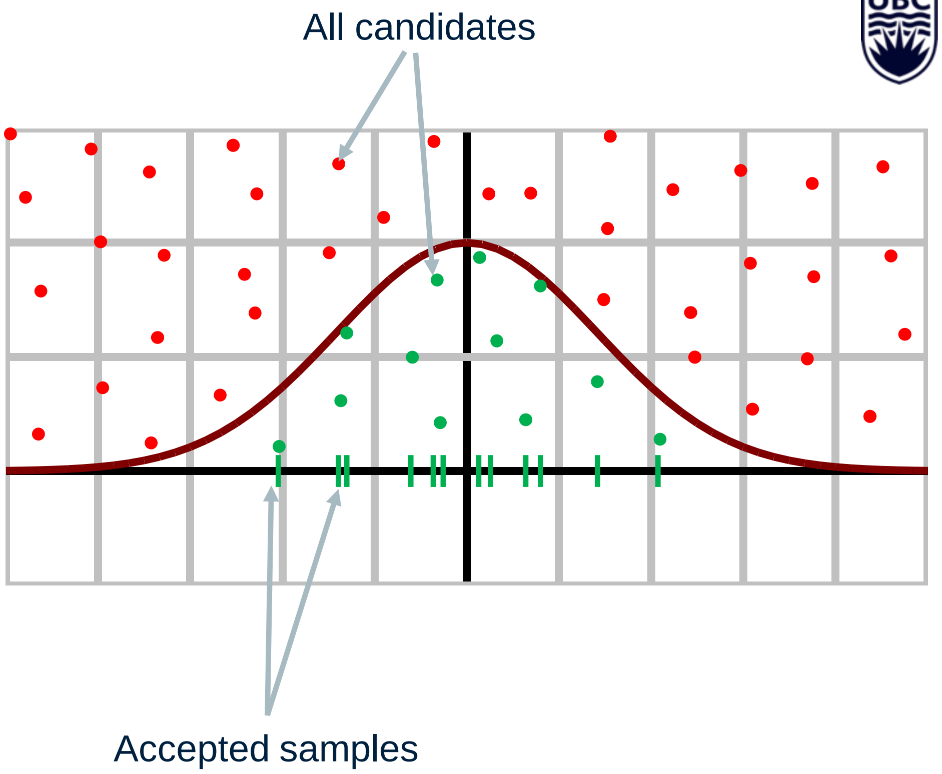
# Sampling from a Gaussian

Rejection sampling from a uniform distribution

- intuitive approach
- ignores the tails of the distribution

Better alternative:

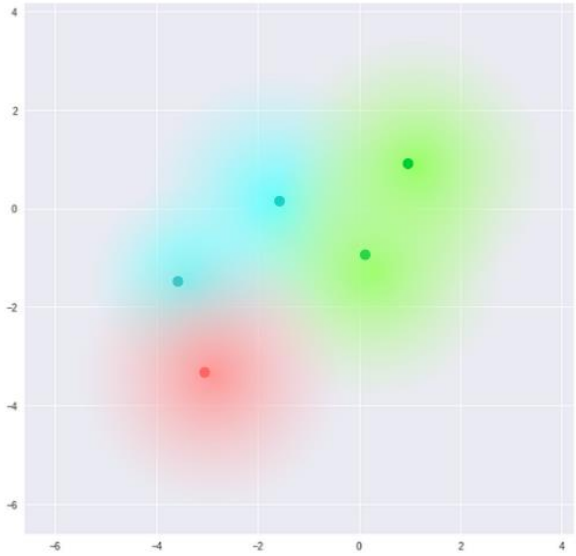
- Box-Muller Transform
  - requires only two uniform samples
  - mathematically correct
    (not an approximation)
  - efficient to compute



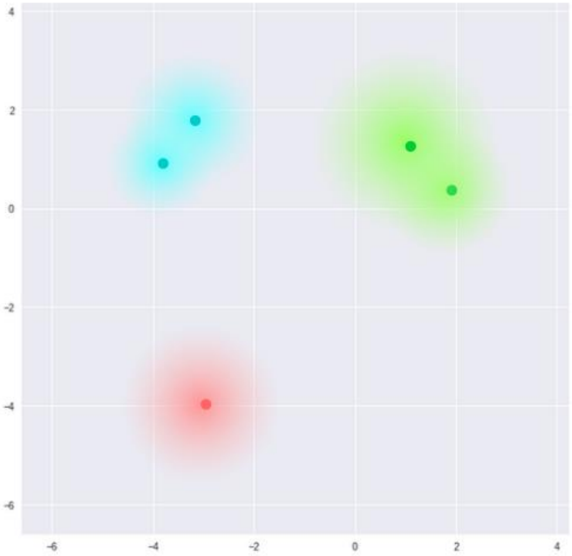
# The effect of the prior

Create a dense and smooth latent space

- without holes
  - all samples will make sense
    - e.g., will reconstruct to plausible images




with prior




without prior

# **Self study:** Deriving the VAE objective via Bayes

Goal: Compute the posterior

$$p(h \mid x) = \frac{p(x \mid h)p(h)}{p(x)}$$

*Good hidden code h, given x*

$$p(x) = \int p(x|h)p(h)dh$$

*The evidence, intractable to compute (marginalization)*

*It requires integration over all possible latent values h*

Attempt: Approximate $p(h \mid x)$ with a NN (encoder) and optimize

$$\mathbf{KL}(q_\phi(h|x)||p(h|x)) = \mathbf{E_q}[\log_{q_\phi}(h|x)] - \mathbf{E}_q[\log p(x,h)] + \log p(x)$$

- still intractable due to p(x) in the divergence between predicted and true distribution

More at https://jaan.io/what-is-variational-autoencoder-vae-tutorial/

# **Self study:** Evidence Lower BOund

Consider the term

$$ELBO(\phi) = E_q[\log p(x,z)] - E_q[\log q_\phi(z|x)]$$

(the VAE objective)

$$= -\mathbf{E}_{\mathbf{h}\sim q_\phi(\mathbf{h}|\mathbf{x})}\big(\log p_\theta(\mathbf{x}|\mathbf{h})\big)$$
$$+D_{\mathrm{KL}}(q_\phi(\mathbf{h}|\mathbf{x})\|p(\mathbf{h}))$$

This KL measures the dissimilarity to the prior!

Together with the KL divergence from before, we get log p(x) as

$$\log p(x) = ELBO(\phi) + \mathbf{KL}(q_\phi(h|x)||p(h|x))$$

This one is the KL from the previous slide (encoder vs. true posterior)!

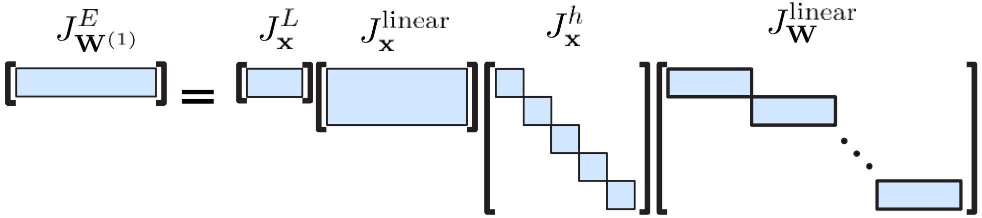
- the Kullback-Leibler divergence is always greater than or equal to zero
  - minimizing the Kullback-Leibler divergence is equivalent to maximizing the ELBO (making one term bigger must reduce the other one)

# Differentiation and sampling

Problem: How to differentiate through the sampling step?

- it's a random process, only statistically dependent on the mean and standard deviation of the sampling distribution


Normal NN differentiation (needed for backpropagation)




What is the Jacobian matrix of the sampling layer?

- How would the sample location change if the parameters of the distribution change?
  - intuition: If the distribution gets wider, the sample should move away from the mean.
    If the mean changes, the sample should move in the same way.

# Differentiation and sampling

Problem: How to differentiate through the sampling step?

- it's a random process, only statistically dependent on the mean and standard deviation of the sampling distribution

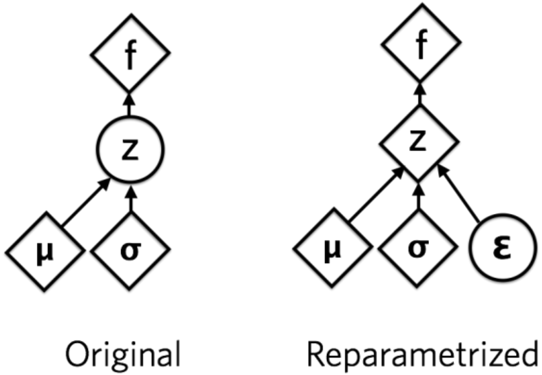
Solutions:

1. The reparameterization trick: Use

   $$h = \mu + \sigma \odot \epsilon, \text{ with } \epsilon \sim \mathcal{N}(0,1)$$

   instead of

   $$h \sim \mathcal{N}(\mu, \sigma)$$



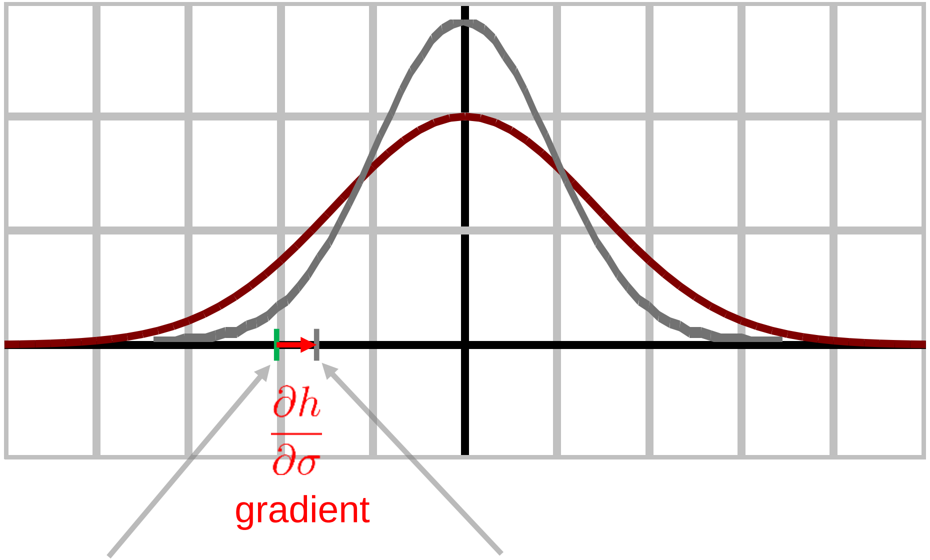

2. Monte-Carlo solution
   - related to reinforcement learning and importance sampling
   - works for discrete and continuous variables
   - we will cover it next week

# Reparameterization trick, visually and mathematically

Equation: $h = \mu + \sigma \odot \epsilon, \text{ with } \epsilon \sim \mathcal{N}(0,1)$

Influence

- changing mu
  - increase -> moves sample right
  - decrease -> moves sample left
- changing sigma
  - increase -> moves away from center
  - decrease -> moves to the center




current sample

sample after downscaling sigma

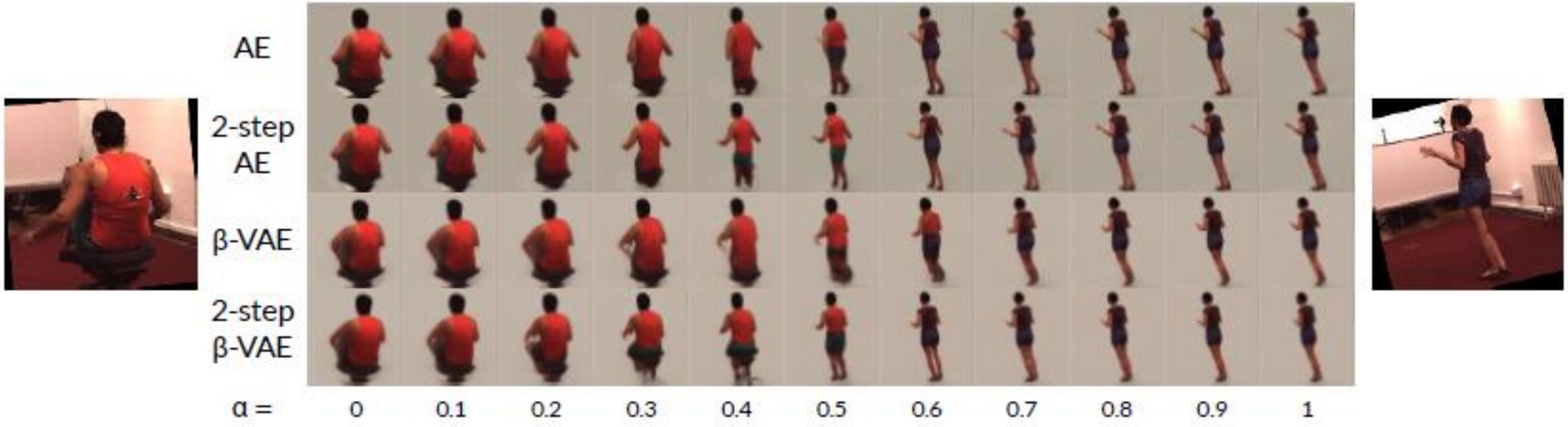
Gradient

$$\frac{\partial h}{\partial \sigma} = \epsilon, \text{ with } \epsilon \sim \mathcal{N}(0,1)$$

$$\frac{\partial h}{\partial \mu} = 1$$

# VAE results

Mixed appearance generation




Interpolation

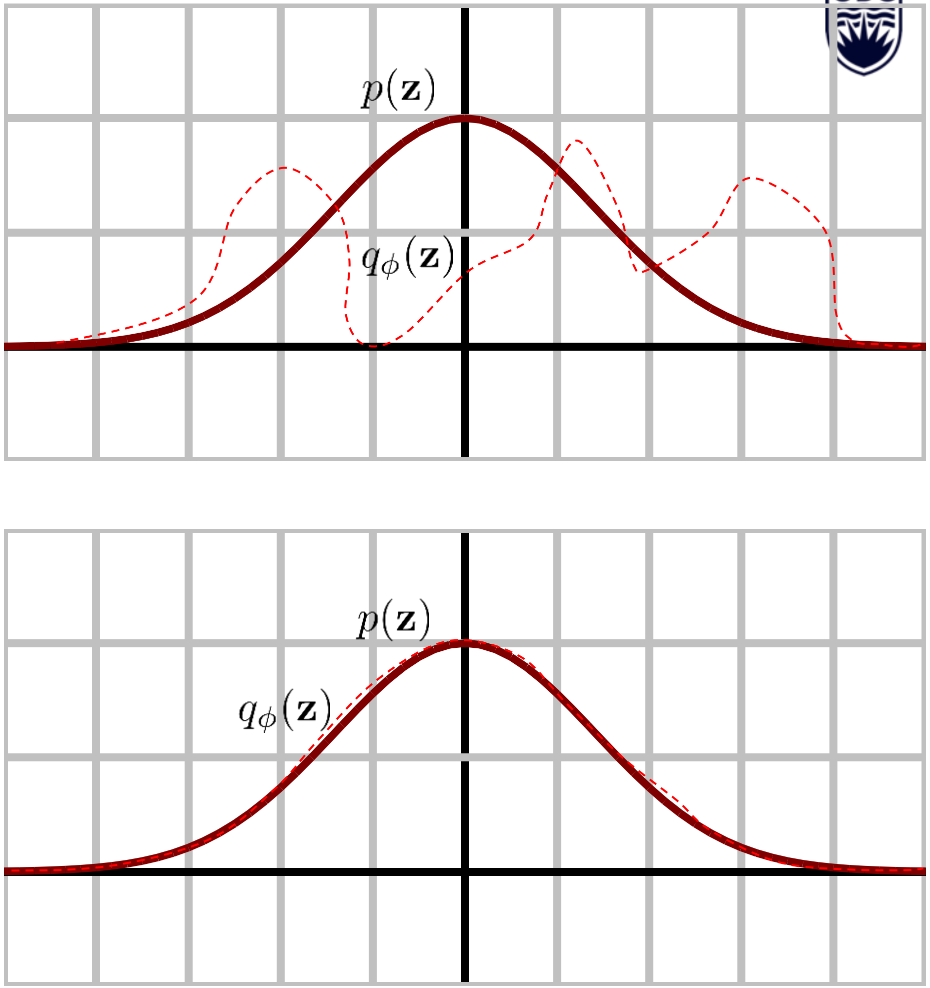
# VAE limitations

Generating human pose and appearance

# VAE Limitations II

Tradeoff between data and prior term

- high weight on data term (big lambda):
  - crisp reconstruction of training data
  - but latent code is not Gaussian
    - the reconstruction of latent code samples from a Gaussian will be incorrect
- high weight on prior term (small lambda):
  - blurry reconstruction
  - but latent code  follows a Gaussian distribution
    - sampling leads to expected outcomes (as good as training samples)



# Summary - the big picture

- Discriminative and generative models
- Supervised, self-supervised, weakly-supervised, and unsupervised approaches
- Representing objects sparsely and densely
- Representing 2D and 3D objects
- Fully-connected, deep convolutional nets, and everything in-between
- Probability theory, ML fundamentals
- How to become a good researcher