COMP 432 Machine Learning

Variational Autoencoders

Computer Science & Software Engineering
Concordia University, Fall 2021

Variational Autoencoders (VAEs)

Auto-Encoding Variational Bayes

"The" VAE paper

Diederik P. Kingma
Machine Learning Group
Universiteit van Amsterdam
dpkingma@gmail.com

Max Welling
Machine Learning Group
Universiteit van Amsterdam
welling.max@gmail.com

ICLR 2014
>10,000 citations

Abstract

How can we perform efficient inference and learning in directed probabilistic models, in the presence of continuous latent variables with intractable posterior distributions, and large datasets? We introduce a stochastic variational inference and learning algorithm that scales to large datasets and, under some mild differentiability conditions, even works in the intractable case. Our contributions is two-fold. First, we show that a reparameterization of the variational lower bound yields a lower bound estimator that can be straightforwardly optimized using standard stochastic gradient methods. Second, we show that for i.i.d. datasets with continuous latent variables per datapoint, posterior inference can be made especially efficient by fitting an approximate inference model (also called a recognition model) to the intractable posterior using the proposed lower bound estimator. Theoretical advantages are reflected in experimental results.

Variational Autoencoders (VAEs)

Intuition: for an input example $\mathbf{x}$, we may be more certain about some "latent attributes" than others

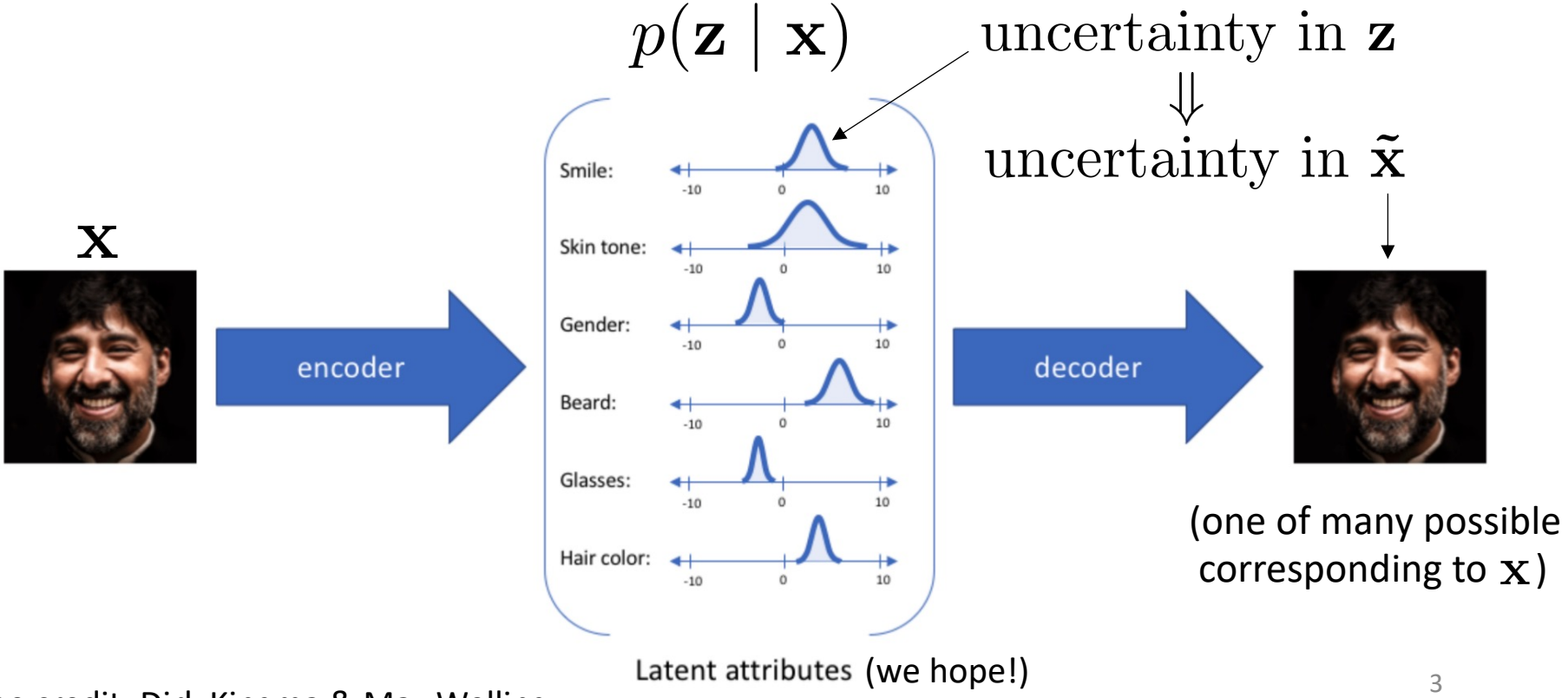


Image credit: Dirk Kingma & Max Welling

Variational Autoencoders (VAEs)

Idea 1: Encoder should output a distribution over codes

For AE code $\mathbf{z} = f(\mathbf{x}, \mathbf{w})$ is *deterministic* function.
For VAE it is a *stochastic* function, typically

$$p(\mathbf{z} \mid \mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2) \quad \text{where} \quad \underbrace{\boldsymbol{\mu}, \boldsymbol{\sigma}} = f(\mathbf{x}, \mathbf{w})$$

encoder determines parameters of **z**'s stochastic distribution, rather than determining **z** itself.

Reconstruction is now:

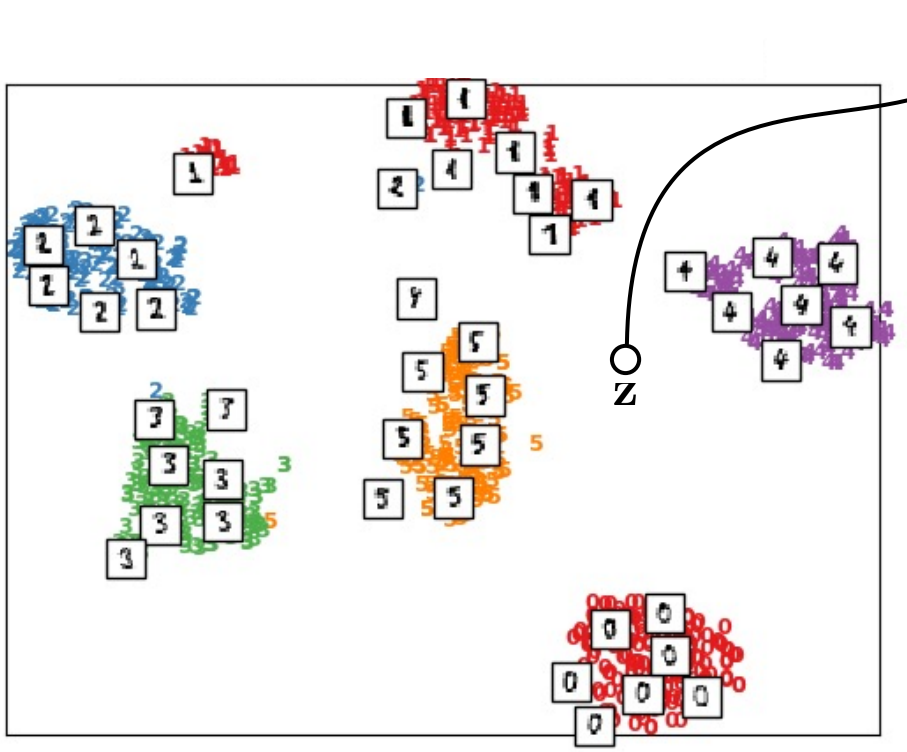
$$\boldsymbol{\mu}, \boldsymbol{\sigma} = f(\mathbf{x}, \mathbf{w}_1)$$
$$\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$$
$$\tilde{\mathbf{x}} = g(\mathbf{z}, \mathbf{w}_2)$$

```
mu, sigma = f(x)
z = sample_normal(mu, sigma)
x_recon = g(z)
```

slight abuse of notation: means $\mathcal{N}(\boldsymbol{\mu}, \mathrm{diag}(\boldsymbol{\sigma}^2))$

Variational Autoencoders (VAEs)

Idea 1 alone doesn't solve the problem that code space can be arbitrarily empty, filled with nonsense regions

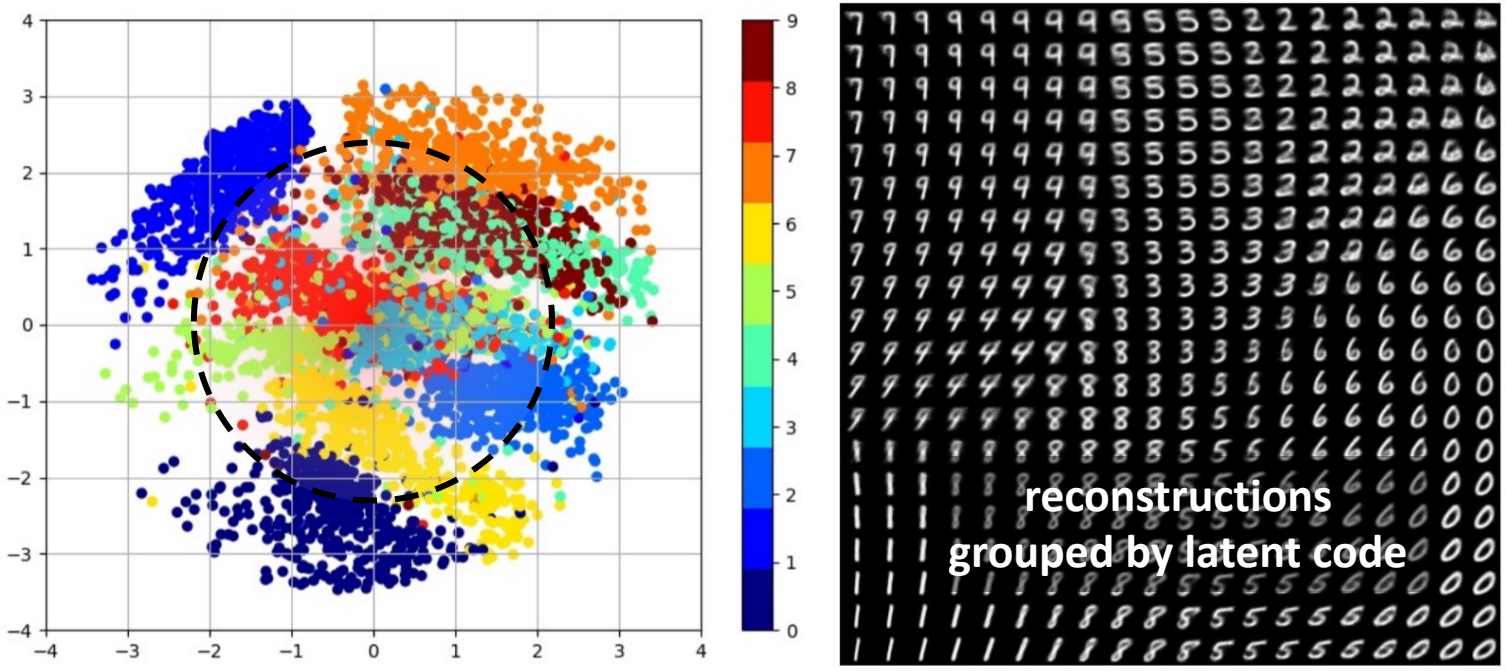


Rather than coding for nonsense, wouldn't we want this particular $\mathbf{z}$ to encode an image that looks *"halfway between a 5 and a 4"*?

What if we could make it easier to generate new $\tilde{\mathbf{x}}$ at the same time?

Variational Autoencoders (VAEs)

<u>Idea 2</u>: Pack the codes together so that they overlap the high-density regions of a standard distribution

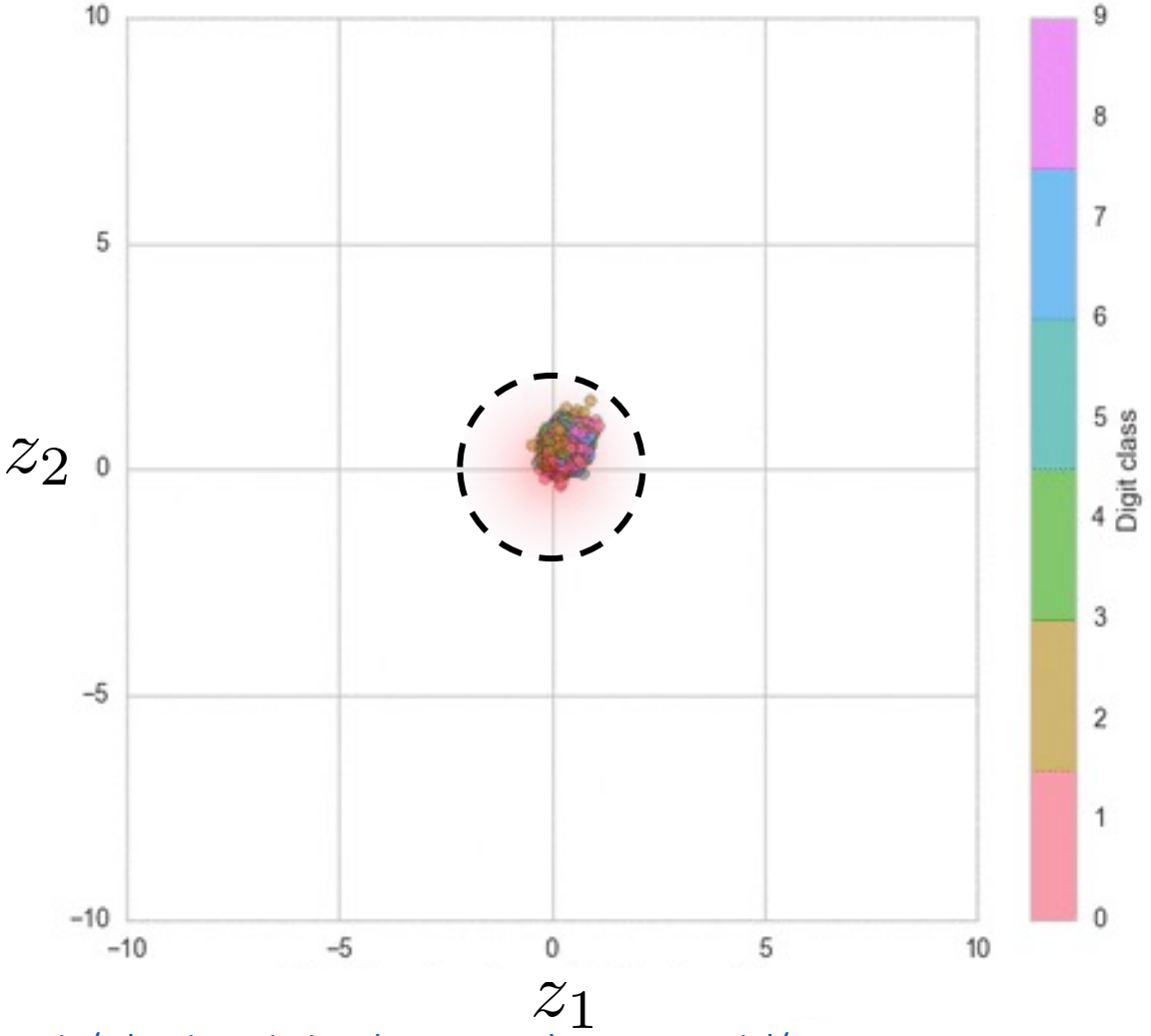


Typically choose $\mathcal{N}(\mathbf{0}, \mathbf{I})$ as standard distribution

Image source: Tschannen *et al.* 2018 https://arxiv.org/pdf/1812.05069.pdf

Variational Autoencoders (VAEs)

Animation of how the means of the latent codes shift around during learning



(latent codes of MNIST digits, where mean of code is coloured by its digit)

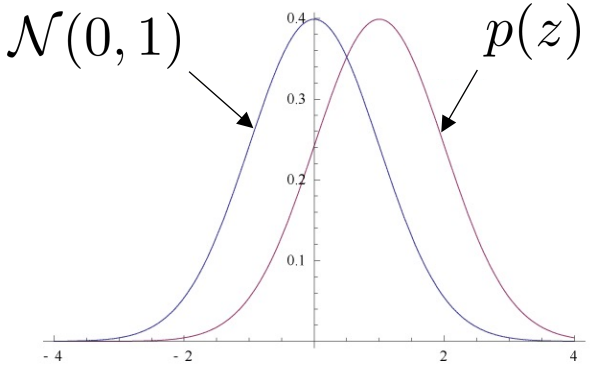
Image source: https://jaan.io/what-is-variational-autoencoder-vae-tutorial/

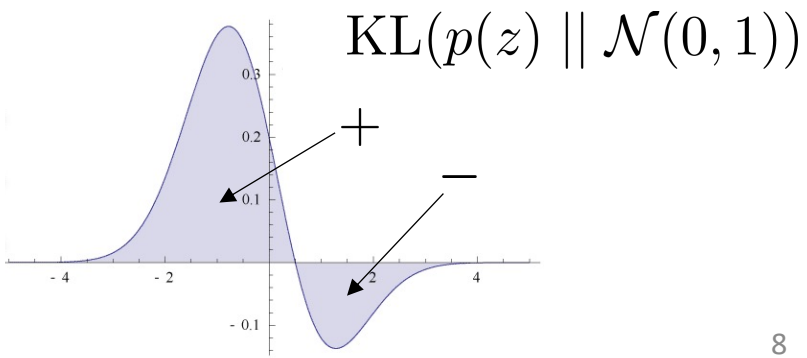
Variational Autoencoders (VAEs)

Typical VAE loss combines "reconstruction" term with a "how well the code distributions pack into $\mathcal{N}(\mathbf{0}, \mathbf{I})$" term

Overlap between each code distribution $p(\mathbf{z}_i \mid \mathbf{x}_i)$ and the "code prior" distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$ is measured by the *Kullback-Leibler divergence* between them

$$D_{\mathrm{KL}}(p(\mathbf{z}_i \mid \mathbf{x}_i) \,||\, \mathcal{N}(\mathbf{0}, \mathbf{I}))$$





- Entropy (in bits) of a discrete distribution $p(\mathbf{z})$

$$H(p) = -\sum_{\mathbf{z}} p(\mathbf{z}) \log_2 p(\mathbf{z}) = \mathbb{E}_{\mathbf{z}\sim p}\left[\log_2 \frac{1}{p(\mathbf{z})}\right]$$

sum over all possible states

higher entropy means "harder to compress data sampled from p using a dictionary-like scheme"

expected # bits to encode state $\mathbf{z}$ when sampled from p and when using a dictionary designed for p

code length of state $\mathbf{z}$ when using a dictionary designed to compress data based on frequencies under p

- Cross-entropy (in bits) of $q(\mathbf{z})$ relative to $p(\mathbf{z})$

$$H(p, q) = -\sum_{\mathbf{z}} p(\mathbf{z}) \log_2 q(\mathbf{z}) = \mathbb{E}_{\mathbf{z}\sim p}\left[\log_2 \frac{1}{q(\mathbf{z})}\right]$$

$H(p, p) = H(p)$

$H(p, q) \geq H(p)$

since dictionary for q might not be ideal for encoding data from p

expected # bits to encode state $\mathbf{z}$ when sampled from p and when using a dictionary designed for q

code length of state $\mathbf{z}$ when using a dictionary designed to compress data based on frequencies under q

- KL-divergence (in bits) of $q(\mathbf{z})$ relative to $p(\mathbf{z})$

$$D_{\mathrm{KL}}(p \,||\, q) = \sum_{\mathbf{z}} p(\mathbf{z}) \log_2 \frac{p(\mathbf{z})}{q(\mathbf{z})} = H(p, q) - H(p)$$

expected # *extra* bits to encode state $\mathbf{z}$ when sampled from p and when using a dictionary designed for q

Kullbach-Leibler (KL) divergence

- Measures dissimilarity of two probability distributions:

$$D_{\mathrm{KL}}(p \,||\, q) = \int p(\mathbf{z}) \ln \frac{p(\mathbf{z})}{q(\mathbf{z})} \mathrm{d}\mathbf{z}$$

zero when $p(\mathbf{z}) = q(\mathbf{z})$

$D_{\mathrm{KL}}(\cdot \,||\, \cdot) \geq 0$

or: $\int p(\mathbf{z}) \left(\ln p(\mathbf{z}) - \ln q(\mathbf{z})\right) \mathrm{d}\mathbf{z}$

$D_{\mathrm{KL}}(p \,||\, q) = 0$ iff $p = q$ a.e. ("almost everwhere")

$D_{\mathrm{KL}}(p \,||\, q) \neq D_{\mathrm{KL}}(q \,||\, p)$ so not symmetric, not a 'distance'

- Additive for independent distributions, i.e., when

$$p(\mathbf{z}_1, \ldots, \mathbf{z}_N) = p_1(\mathbf{z}_1) \cdots p_N(\mathbf{z}_N)$$
$$q(\mathbf{z}_1, \ldots, \mathbf{z}_N) = q_1(\mathbf{z}_1) \cdots q_N(\mathbf{z}_N)$$

then $D_{\mathrm{KL}}(p \,||\, q) = \sum_{i=1}^{N} D_{\mathrm{KL}}(p_i \,||\, q_i)$

Variational Autoencoder loss

- Typical VAE loss is therefore

but what's the actual formula for *this* term?

$$\ell(\mathbf{w}, \boldsymbol{\omega}) = \sum_{i=1}^{N} \frac{1}{2} \left\| \mathbf{x}_i - \tilde{\mathbf{x}}_i \right\|^2 + \beta D_{\mathrm{KL}}\left(\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i^2) \,||\, \mathcal{N}(\mathbf{0}, \mathbf{I})\right)$$

where $\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i = f(\mathbf{x}_i, \mathbf{w})$ are the encoded params,

$\mathbf{z}_i \sim \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i^2)$ are the sampled latent codes

$\tilde{\mathbf{x}}_i = g(\mathbf{z}_i, \boldsymbol{\omega})$ are the reconstructed data

$\beta \geq 0$ is the strength of KL "packing force"

important VAE hyperparameter introduced by "β-VAE" paper

(original VAE just assumed $\beta = 1$)

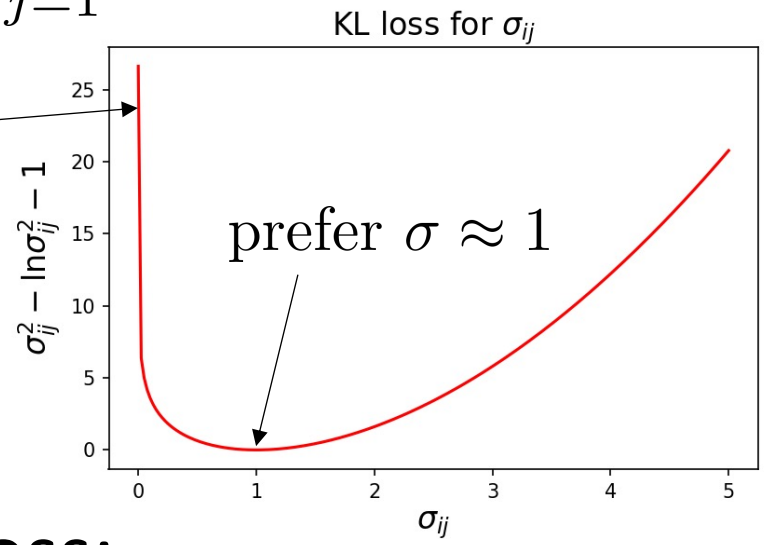
Higgins et al ICLR 2017

$(\text{assume } \mathbf{z} \in \mathbb{R}^M)$

Variational Autoencoder loss

$$D_{\mathrm{KL}}(\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2) \,||\, \mathcal{N}(\mathbf{0}, \mathbf{I})) = \frac{1}{2} \sum_{j=1}^{M} \left(\mu_j^2 + \sigma_j^2 - \ln \sigma_j^2 - 1\right)$$

KL automatically prevents $\sigma \to 0$



Choosing $\mathcal{N}$ gives specific loss:

$$\ell(\mathbf{w}, \boldsymbol{\omega}) = \sum_{i=1}^{N} \frac{1}{2} \left\|\mathbf{x}_i - \tilde{\mathbf{x}}_i\right\|^2 + \frac{\beta}{2} \left(\left\|\boldsymbol{\mu}_i\right\|^2 + \left\|\boldsymbol{\sigma}_i\right\|^2 - \sum_{j=1}^{M} (\ln \sigma_{ij}^2 + 1) \right)$$

Backpropagate through "$\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$"

- VAE performs a sampling operation between encoding and reconstruction.
- How do we "backpropagate" through this step?
  - How would we get $\frac{\partial \ell}{\partial \mu_j}$ and $\frac{\partial \ell}{\partial \sigma_j}$ and thereby $\nabla_{\mathbf{w}} \ell$?
- **<u>Reparameterization trick</u>:** rewrite $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ as

```
mu, sigma = f(x)
eps = sample_normal(0, 1)
z = mu + sigma*eps
x_recon = g(z)
```

$$\text{step 1: } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

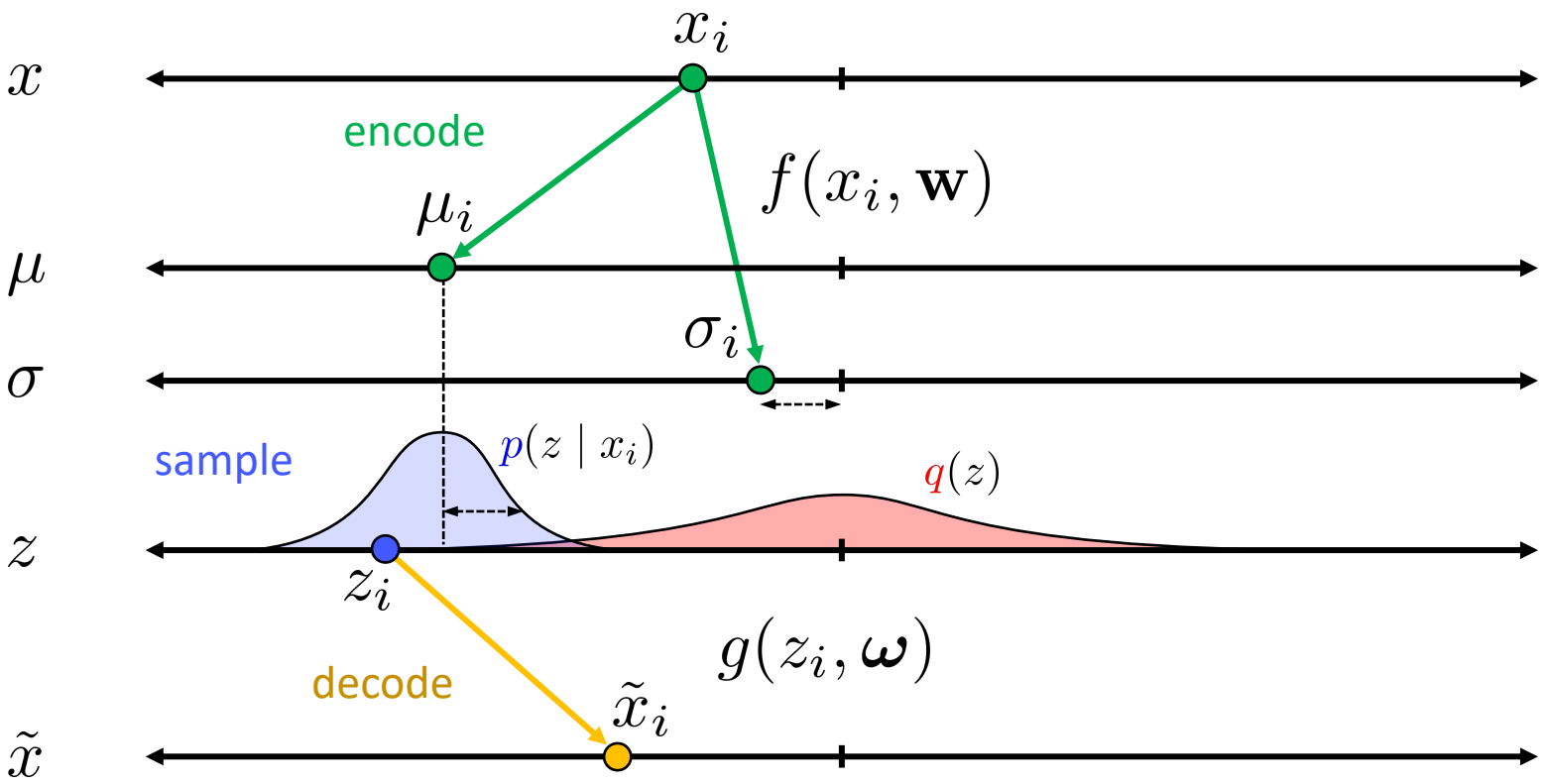
$$\text{step 2: } \mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$$

(symbol for elementwise multiplication)

gradients can "flow" to the encoder, through standard multiple-add ops

treated as constant

Example training a 1-1-1 linear VAE



$$\ell(\mathbf{w}, \boldsymbol{\omega}) = \sum_{i=1}^{N} \frac{1}{2}(x_i - \tilde{x}_i)^2 + D_{\mathrm{KL}}\big(\mathcal{N}(\mu_i, \sigma_i^2) \,||\, \mathcal{N}(0,1)\big)$$

VAEs are generative models

before training

after 10 epochs

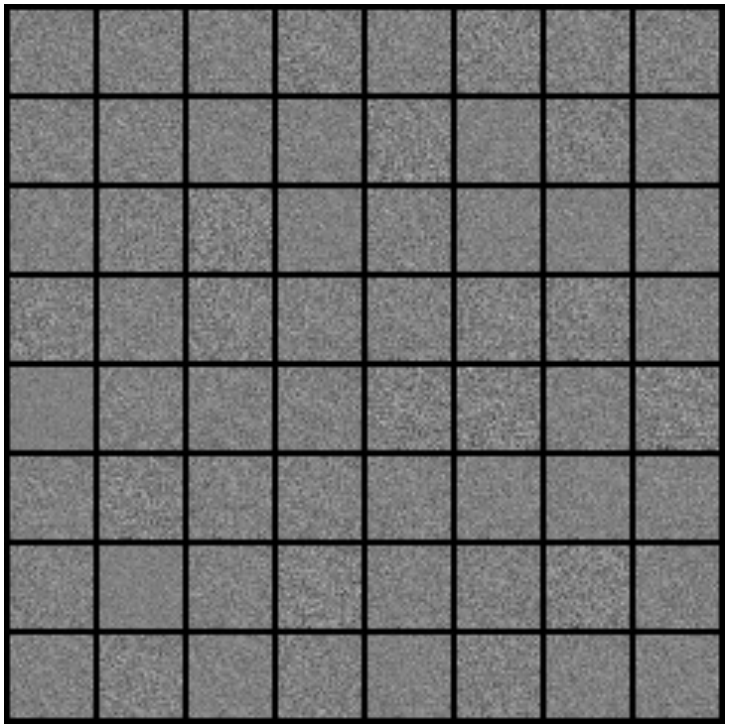
$\mathbf{x}_i$

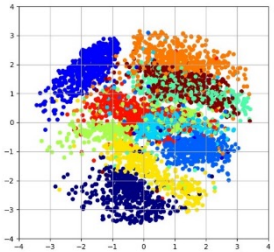
$\tilde{\mathbf{x}}_i$

samples

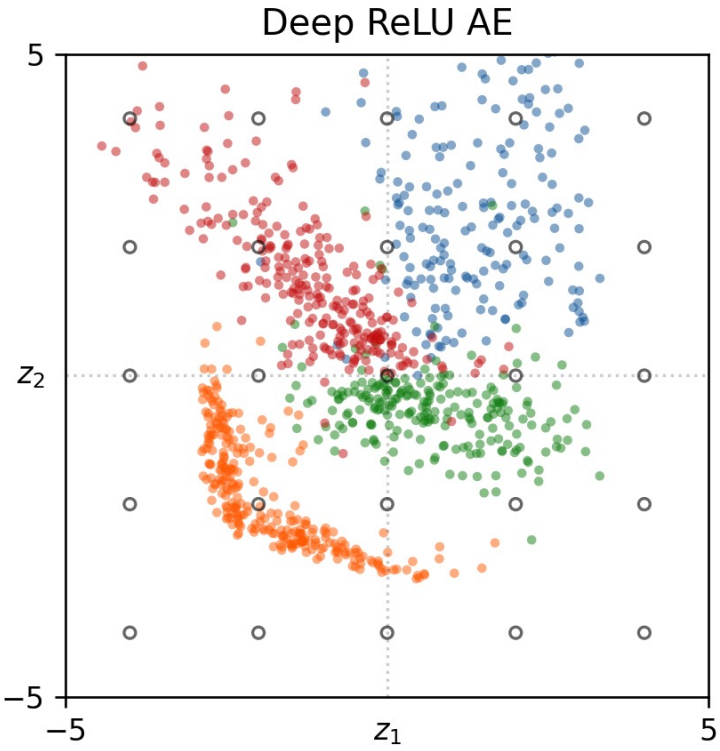
$\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

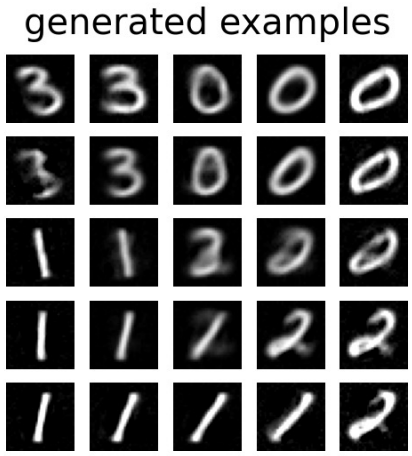
$\tilde{\mathbf{x}} = g(\mathbf{z}, \boldsymbol{\omega})$



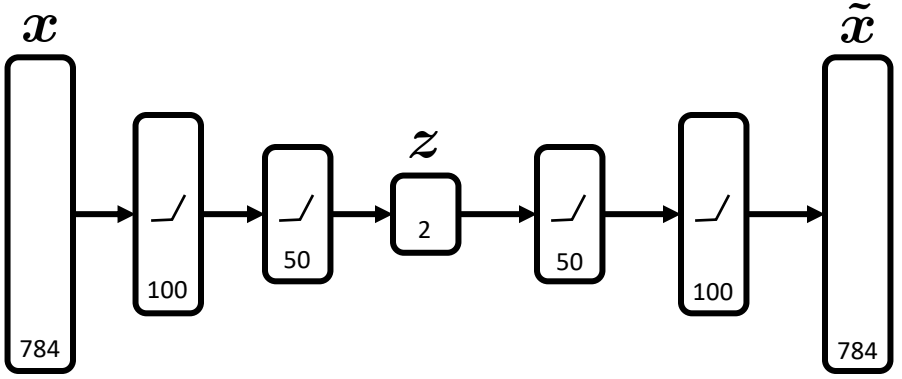


Example: Deep ReLU Autoencoder



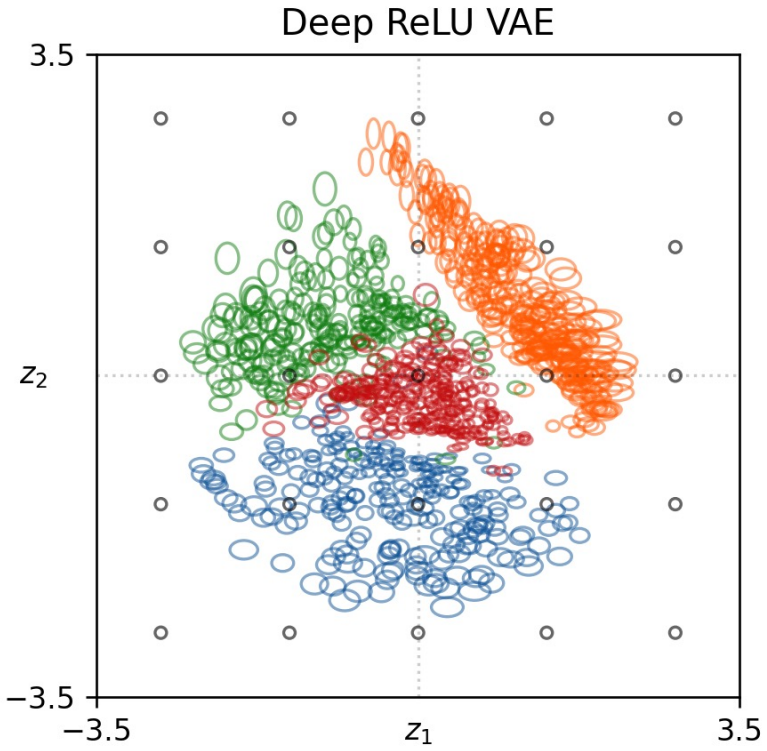


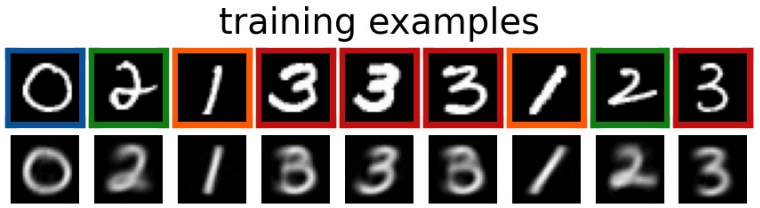
'0' digit
'1' digit
'2' digit
'3' digit
generated

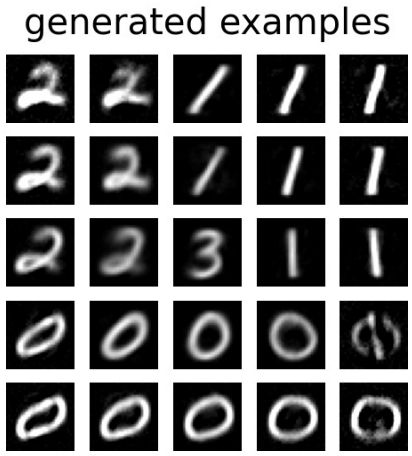


(this slide is a video)

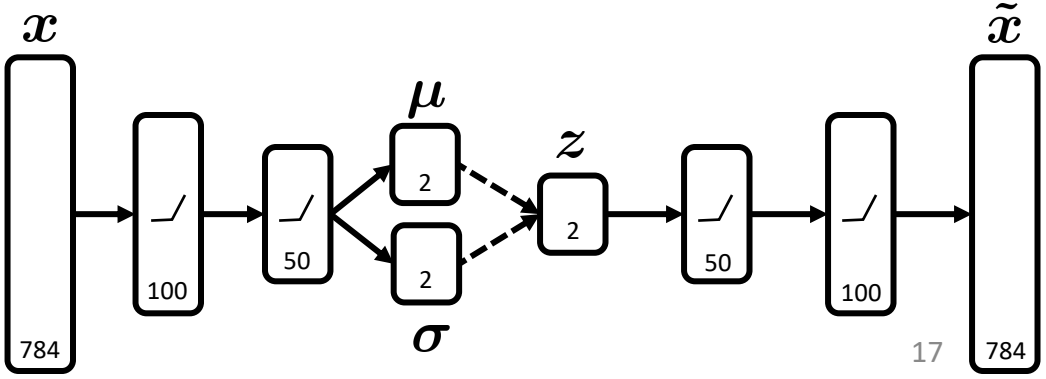
Example: Deep ReLU Variational AE





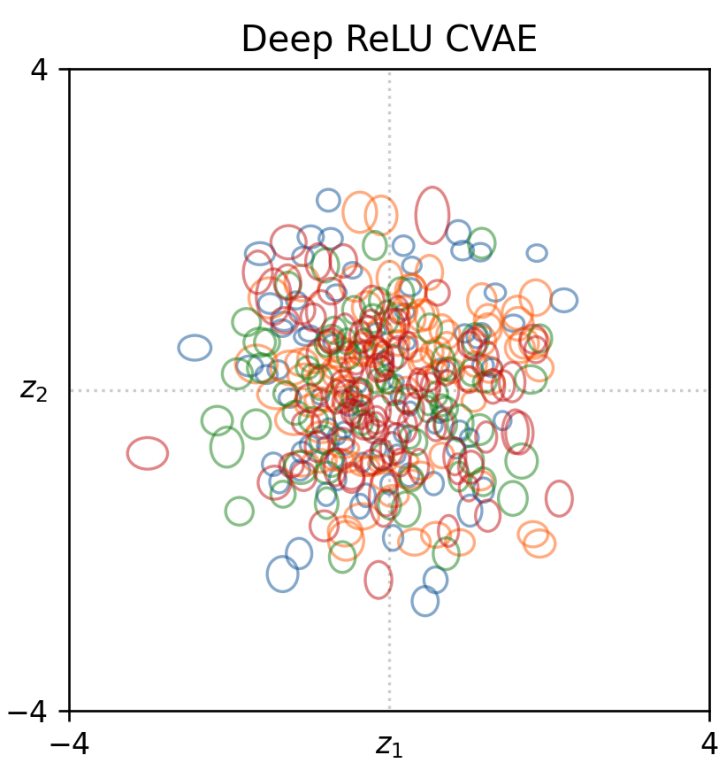


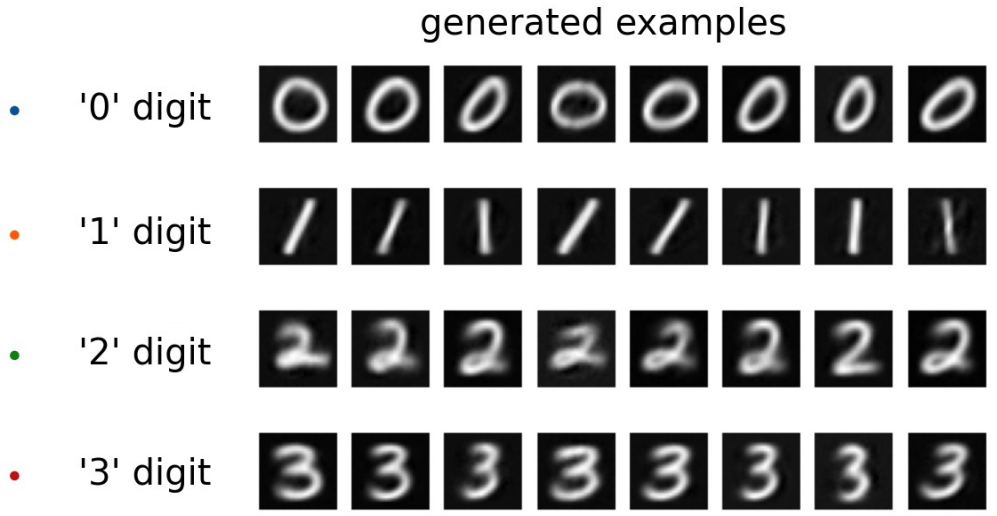
'0' digit
'1' digit
'2' digit
'3' digit
generated



(this slide is a video)

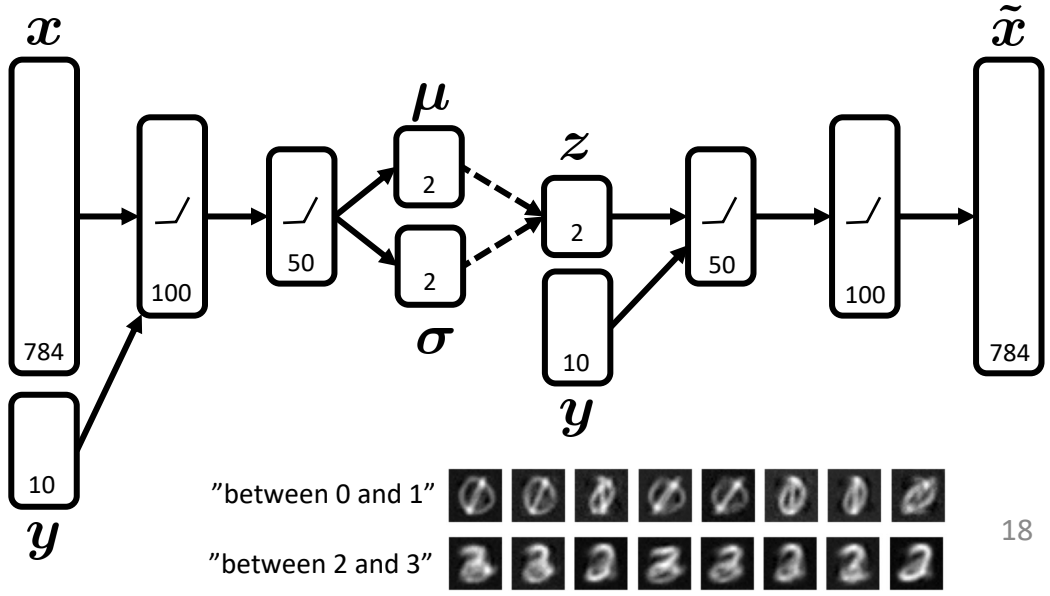
Deep ReLU *Conditional* VAE



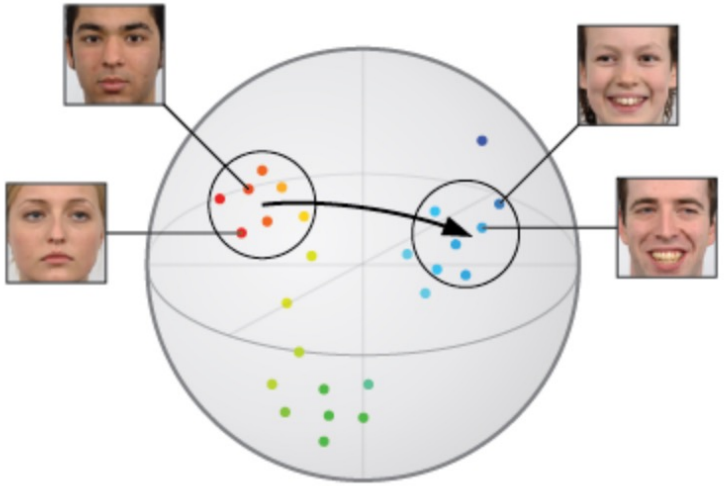


The encoder and decoder both take the class label as input, *i.e.*, they are *conditioned* on class label.

Latent codes *appear* to overlap, but in (2+10)-dimensional space they do not!



Example: VAE to encode faces



After training, can label some data as "not smiling" and other data as "smiling" and compute a "smile vector" (a direction in latent space) by subtracting the average latent codes…

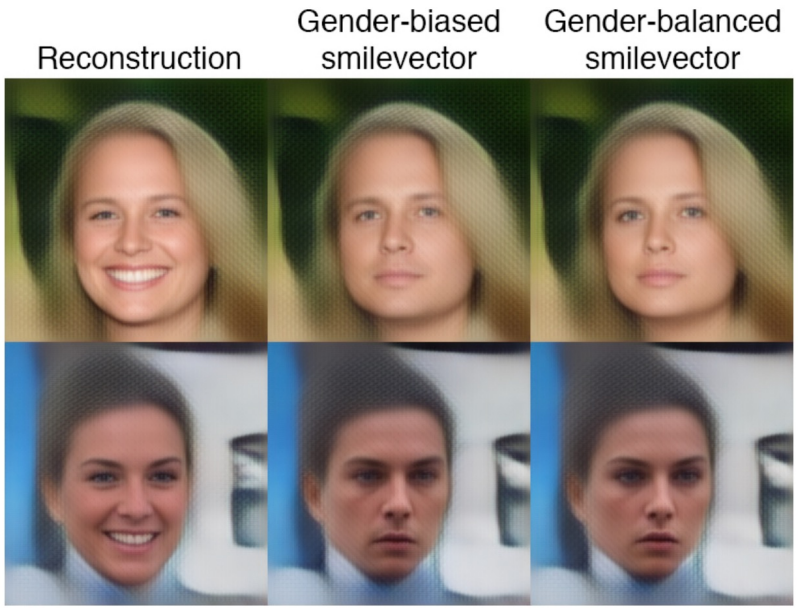


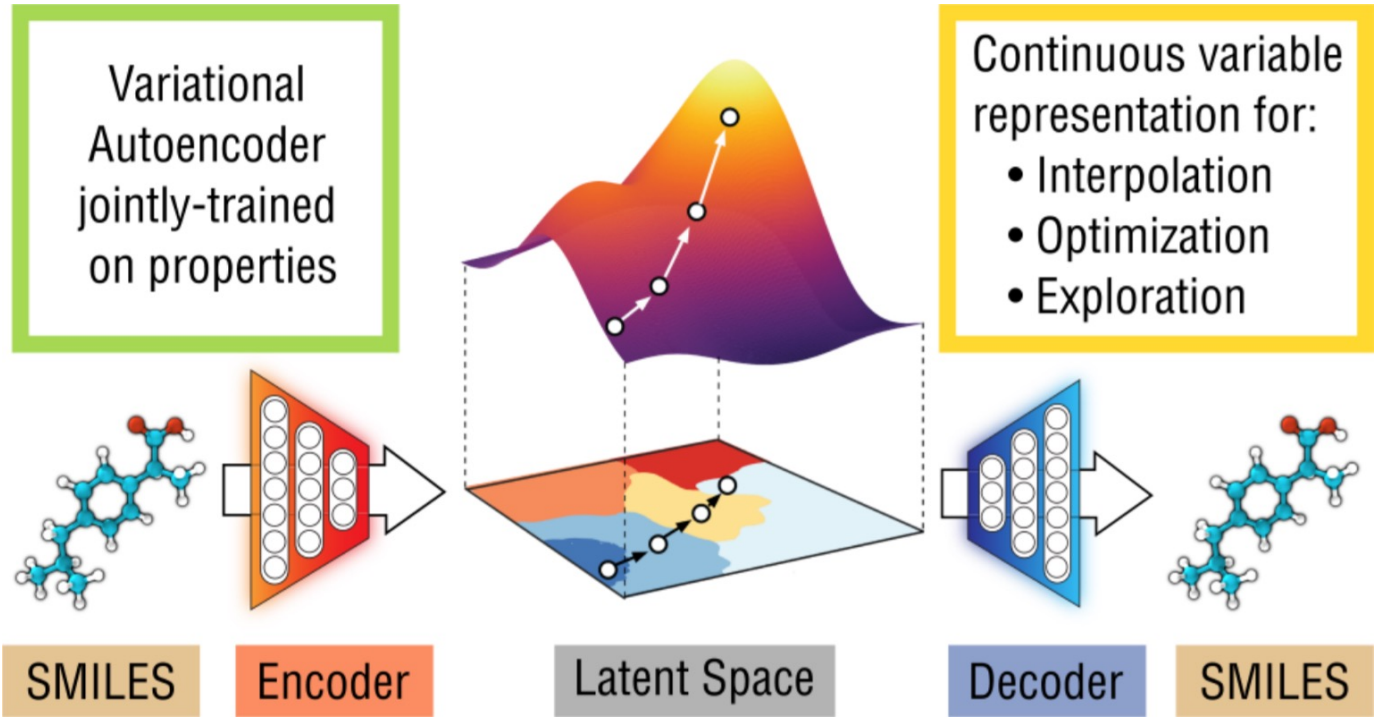
Figure 7: Initial attempts to build a smile vector suffered from sampling bias. The effect was that removing smiles from reconstructions (left) also added male attributes (center). By using replication to balance the data across both attributes before computing the attribute vectors, the gender bias was removed (right). (model: VAE from Lamb 16 on CelebA)

Image credit: Tom White https://arxiv.org/pdf/1609.04468.pdf

Example: VAE to design molecules

If we have training data with scores (e.g. molecular activity scores), map them to latent space, then search for unseen codes that are predicted to have higher score.

If we "decoded" that imaginary molecule, would it turn out to have higher activity than any of the training cases?



https://pubs.acs.org/doi/abs/10.1021/acscentsci.7b00572