VAEs and data generation

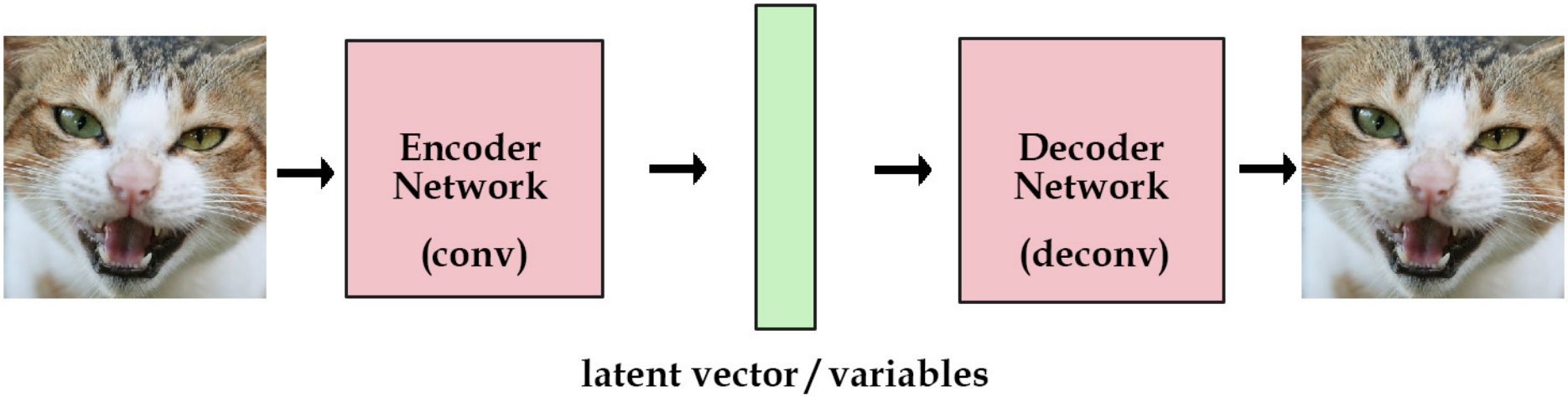
Yale

CPSC/AMTH 452/552

CBB 663

Variational Autoencoders (VAEs)

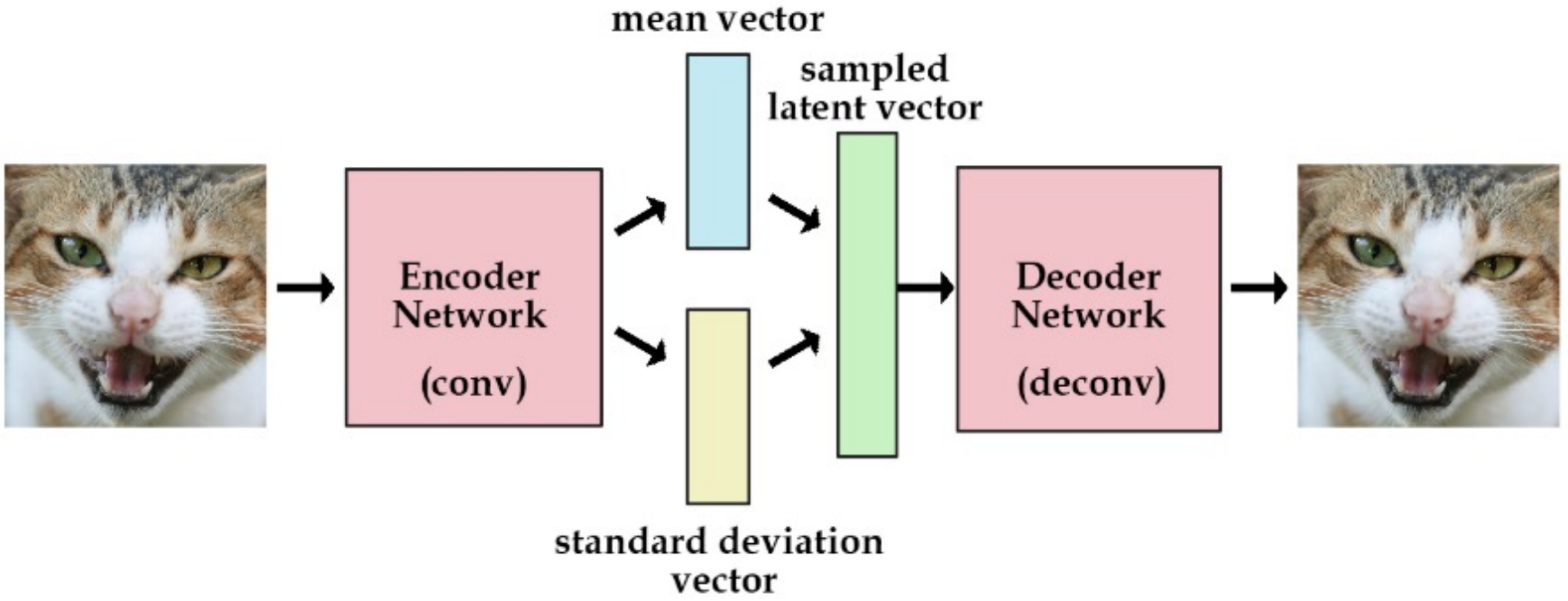
- Standard autoencoder



- We want to generate new examples
- Can we use an autoencoder?

Variational Autoencoders (VAEs)

- Force the latent vectors to have (roughly) a unit Gaussian distribution
- Generate images by sampling a unit Gaussian and passing it into the decoder



Variational Autoencoders (VAEs)

- Tradeoff between accuracy and the unit Gaussian approximation
- Build this directly into the loss
  - Kullback-Leibler (KL) divergence: a measure of difference between probability distributions P and Q

$$D_{KL}(P||Q) = -\sum_{i} P(i) \log\left(\frac{Q(i)}{P(i)}\right)$$

  - Include a regularization term that penalizes the KL divergence between a unit Gaussian and the latent vector distribution
  - How do we do this in practice?

Variational Autoencoders (VAEs)

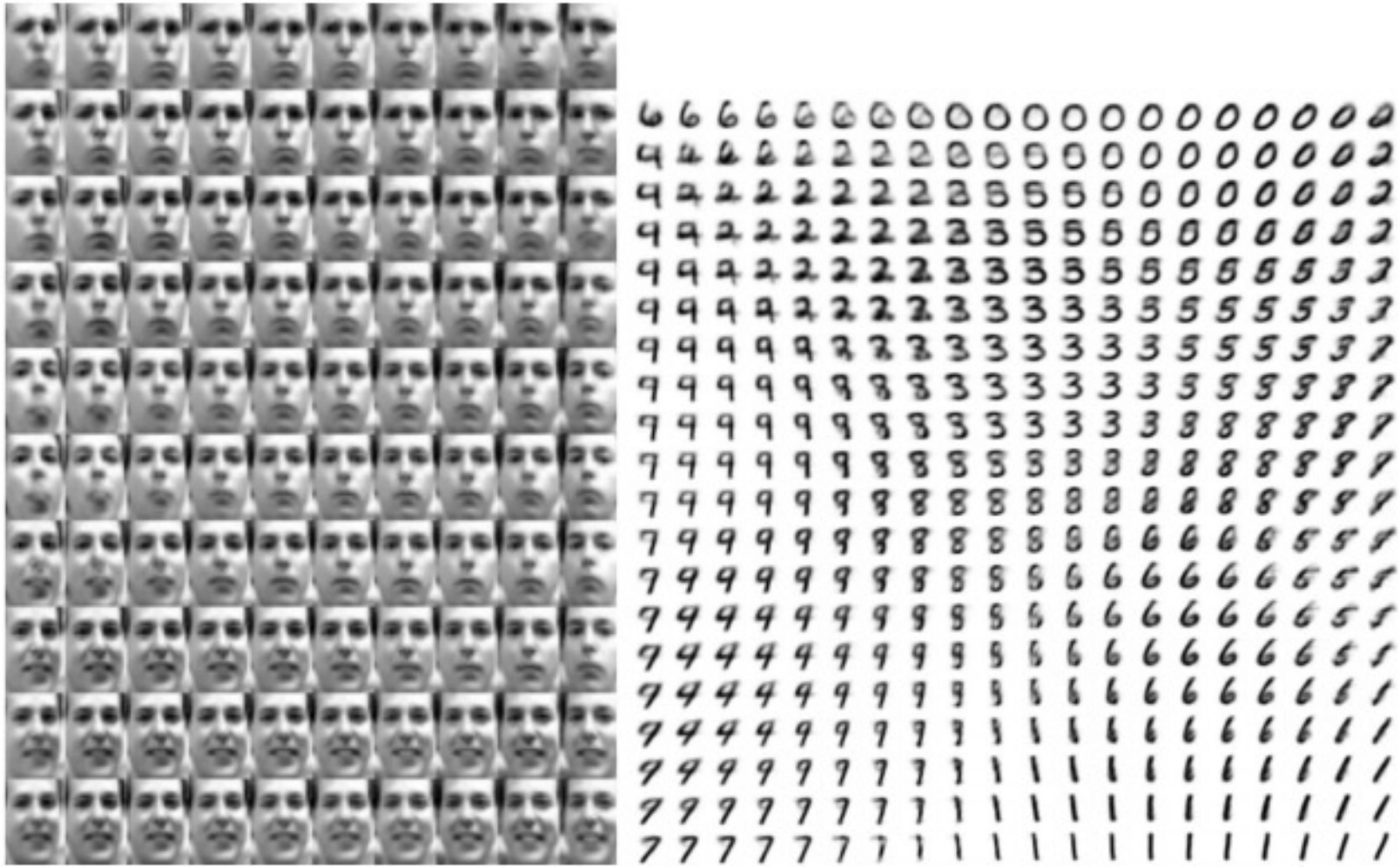
1st Epoch

9th Epoch

Original

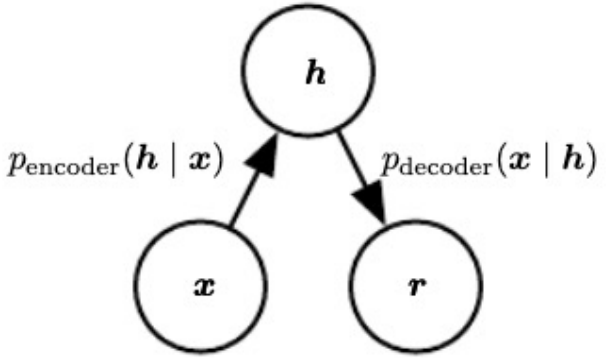
Variational Autoencoders (VAEs)

2D latent mapping of results from VAEs



Recall

- Autoencoders can be thought of as latent variable models if the encoder is stochastic

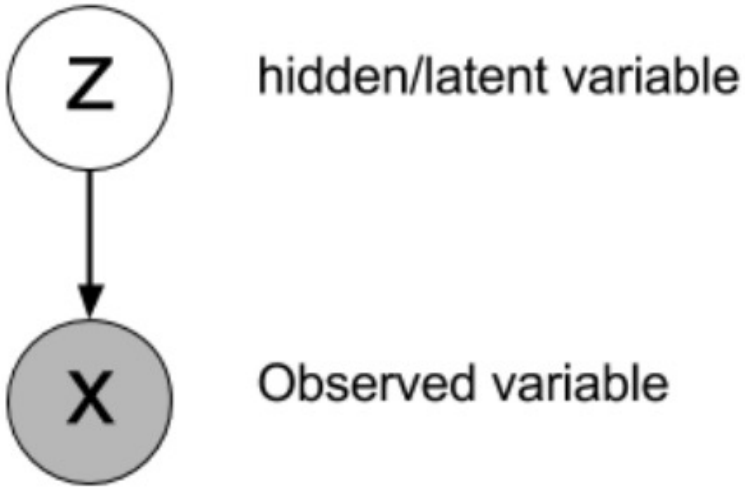


posterior dist

- Then the autoencoder training actually corresponds to minimizing negative log likelihood

$J = \min \quad -\log p_{\text{decoder}}(\boldsymbol{x} \mid \boldsymbol{h})$

VAEs and Variational Inference



$$p(Z|X) = \frac{p(X|Z)p(Z)}{p(X)} = \frac{p(X|Z)p(Z)}{\int_Z p(X,Z)}$$

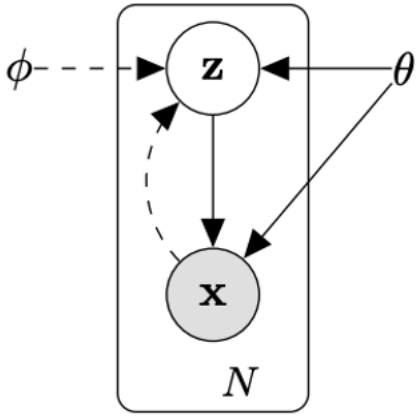
Intractible to Compute! But can use sampling!

$P(\mathbf{Z} \mid \mathbf{X}) \approx Q(\mathbf{Z})$.

parametrized dist which can be optimized by SGD

Posterior approximated with a **variational distribution** Q(Z) which has a fairly simple form, i.e. from a family of Gaussians.

VAEs and Variational Inference



Solid lines are the generative model $p_{\boldsymbol{\theta}}(\mathbf{z})p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$

Dashed lines are the variational approximation $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$

Basic Idea of VAEs

- Use a parametric form of Q(Z)
- There are two kinds of parameters
  - Variational parameters
  - Neural network parameters
- Use gradient descent to optimize it
  - But neural networks don't do well with equality constraints so use an inequality to approximate it called ELBO (evidence based lower bound)

$$P(\mathbf{Z} \mid \mathbf{X}) \approx Q(\mathbf{Z}).$$

$$p(Z|X) = \frac{p(X|Z)p(Z)}{p(X)} = \frac{p(X|Z)p(Z)}{\int_Z p(X,Z)}$$

- Use sampling of datapoints and variational parameters to come up with all other terms!

sample from variational latent space

Variational Bayes (Detour)

- Most common form minimizes KL-divergence

between dist Q and P $D_{\mathrm{KL}}(Q \parallel P) \triangleq \sum_{\mathbf{Z}} Q(\mathbf{Z}) \log \frac{Q(\mathbf{Z})}{P(\mathbf{Z} \mid \mathbf{X})}.$

- Given that $P(\mathbf{Z} \mid \mathbf{X}) = \frac{P(\mathbf{X}, \mathbf{Z})}{P(\mathbf{X})}$

- KL divergence can be written as:

$$D_{\mathrm{KL}}(Q \parallel P) = \sum_{\mathbf{Z}} Q(\mathbf{Z}) \left[\log \frac{Q(\mathbf{Z})}{P(\mathbf{Z}, \mathbf{X})} + \log P(\mathbf{X}) \right] = \sum_{\mathbf{Z}} Q(\mathbf{Z}) \left[\log Q(\mathbf{Z}) - \log P(\mathbf{Z}, \mathbf{X}) \right] + \underbrace{\sum_{\mathbf{Z}} Q(\mathbf{Z})}_{1} \left[\log P(\mathbf{X}) \right]$$

- Since P(X) is a constant and Q(z) sums to 1 over all z's

$$D_{\mathrm{KL}}(Q \parallel P) = \sum_{\mathbf{Z}} Q(\mathbf{Z}) \left[\log Q(\mathbf{Z}) - \log P(\mathbf{Z}, \mathbf{X}) \right] + \log P(\mathbf{X})$$

Variational Bayes Continued

- This can be rewritten in terms of expectation over Q(Z):

$$D_{\mathrm{KL}}(Q \parallel P) = \mathbb{E}_{\mathbf{Q}}\left[\log Q(\mathbf{Z}) - \log P(\mathbf{Z}, \mathbf{X})\right] + \log P(\mathbf{X})$$

- Rearranging terms to get log P(X) on one side gives:

$$\log P(\mathbf{X}) = D_{\mathrm{KL}}(Q \parallel P) - \mathbb{E}_{\mathbf{Q}}\left[\log Q(\mathbf{Z}) - \log P(\mathbf{Z}, \mathbf{X})\right] = D_{\mathrm{KL}}(Q \parallel P) + \mathcal{L}(Q)$$

- Here the last term is called the variational free energy
- Maximizing variational free energy minimizes divergence this term because log of evidence is fixed
- In this sense the variational free energy forms a bound log P(X)>L(Q)

log likelihood of data

ELBO

log P(X) is lower bound by L(Q)

max L(Q) ⇒ min $D_{KL}(Q||P)$ ⇒ Q close to P

VAEs

Auto-Encoding Variational Bayes

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

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

Autoencoding Variational Bayes

- Implements VB in an autoencoder
- Log likelihood computed as sum of value for individual points for a given nnet parameter setting θ, ϕ are parameters of Q

estimated posterior — true posterior

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) = D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}^{(i)})) + \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)})$$

encoder — likelihood

loss used in VAE

likelihood is lower bound, maximize it

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) \geq \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[-\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) + \log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})\right]$$

- Can also be written as divide prior over z $p_\theta(z)$

decoder

$$\mathcal{L}(\theta, \phi; x^{(i)}) = -D_{KL}(q_\phi(z|x^{(i)}) \| p_\theta(z)) + E_{q_\phi(z|x^{(i)})}[\log p_\theta(x^{(i)}|z)]$$

Autoencoding Variational Bayes

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) \geq \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[-\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) + \log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})\right]$$

- Can condition the whole analysis over the z, given a prior on z, p(z)
- In a neural network prior p(z) corresponds to the way the latent variables are desired to be shaped

$$\text{Log}\,\frac{p_\theta(x^{(i)})}{p(z)} \geq \frac{\mathcal{L}(\theta,\phi,x^{(i)})}{p(z)} = E_{q(Z|X)} - \frac{\log q(Z|X)}{p(z)} + \frac{\log p(x,z)}{p(z)}$$

- This gives:

$$\text{Log}\,\frac{p_\theta(x^{(i)})}{p(z)} \geq \quad -D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z})) + \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z})\right]$$

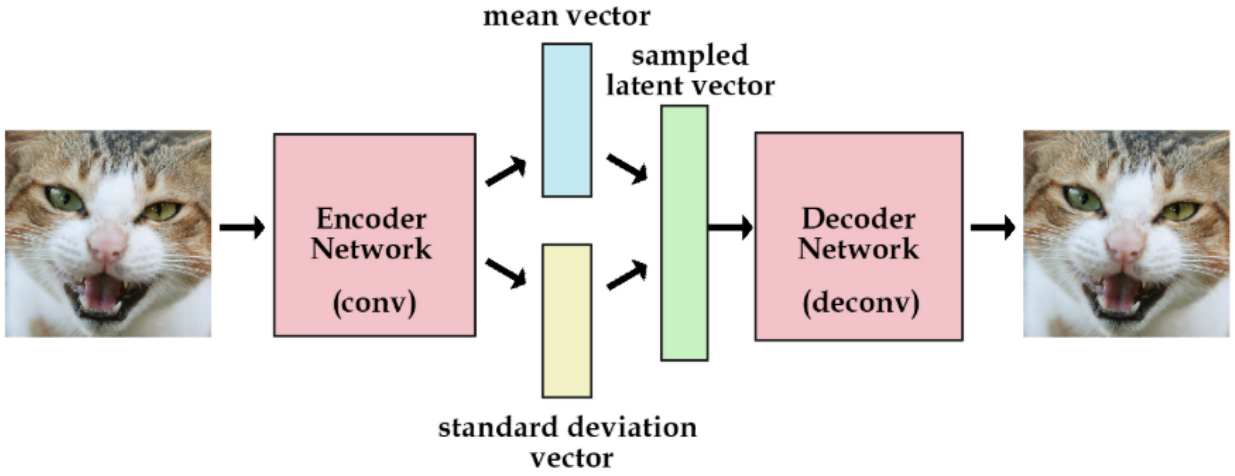
Autoencoding Variational Bayes

- VAEs use gradient descent to maximize variational free energy
- Use a parametric (neural network parameterized form) Q(Z) and sampling-based estimate of P(Z,X)

Algorithm 1 Minibatch version of the Auto-Encoding VB (AEVB) algorithm. Either of the two SGVB estimators in section 2.3 can be used. We use settings $M = 100$ and $L = 1$ in experiments.

$\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Initialize parameters
repeat
  $\mathbf{X}^M \leftarrow$ Random minibatch of M datapoints (drawn from full dataset)
  $\boldsymbol{\epsilon} \leftarrow$ Random samples from noise distribution $p(\boldsymbol{\epsilon})$
  $\mathbf{g} \leftarrow \nabla_{\boldsymbol{\theta},\boldsymbol{\phi}} \widetilde{\mathcal{L}}^M(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}^M, \boldsymbol{\epsilon})$ (Gradients of minibatch estimator (8)) compute gradient
  $\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Update parameters using gradients $\mathbf{g}$ (e.g. SGD or Adagrad [DHS10])
until convergence of parameters $(\boldsymbol{\theta}, \boldsymbol{\phi})$
return $\boldsymbol{\theta}, \boldsymbol{\phi}$

Autoencoding Variational Bayes



- Think of the
  - encoder computing q(z|x)
  - decoder as computing p(x|z) to reobtain x
  - p(z) is a prior distribution on z
- Maximizing the ***variational lower bound L(q)*** of a probability distribution
- Two terms in the loss function

$$-D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_\theta(\mathbf{z}))$$

KL Divergence Term

penalize encoder output different from prior

$$\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log p_\theta(\mathbf{x}^{(i)}|\mathbf{z})\right]$$

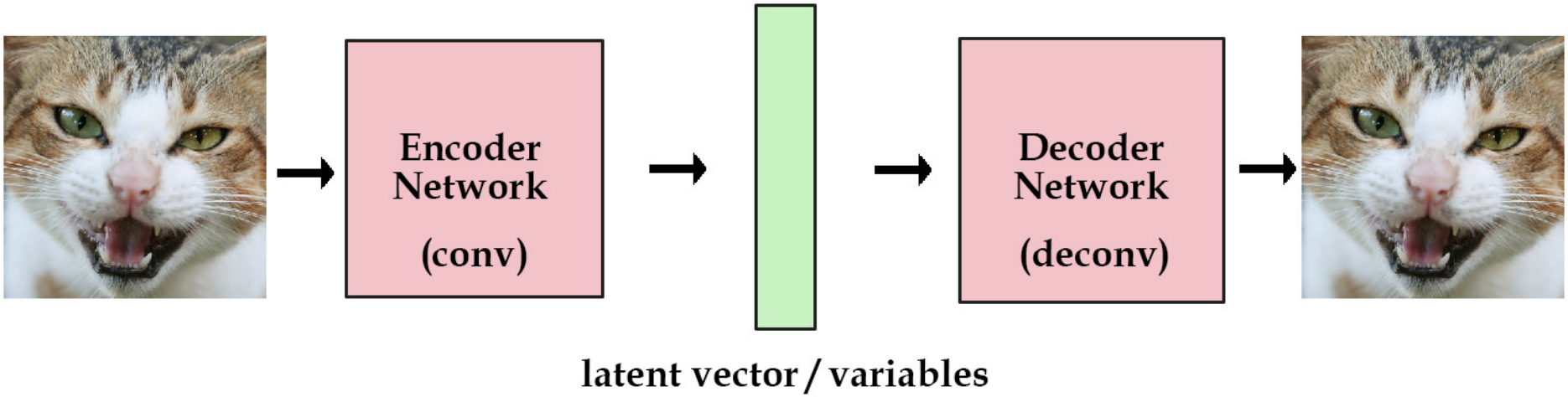
Log Likelihood Term

minimize reconstruction loss

Data Generation with Autoencoders

Variational Autoencoders (VAEs)

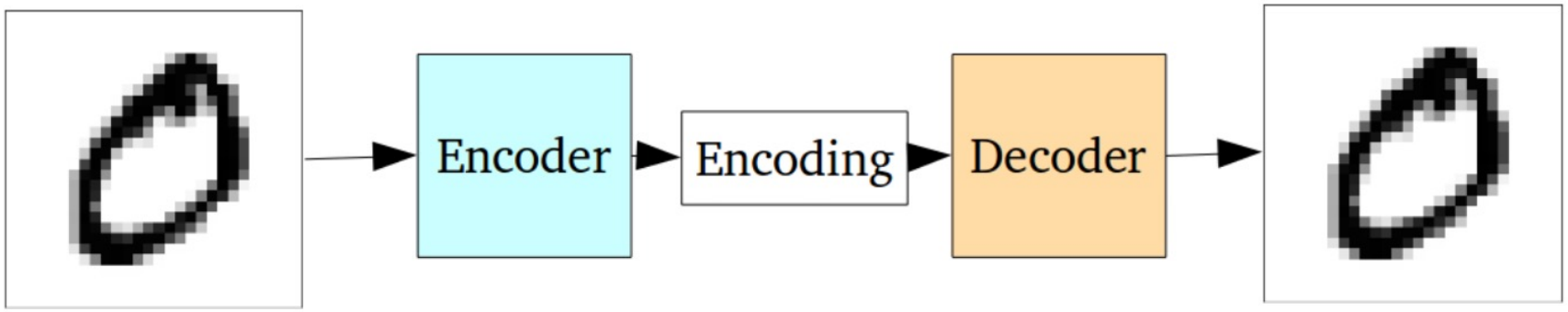
- Learning latent variables that allow for sampling



- We want to generate new examples just by sampling in the latent space

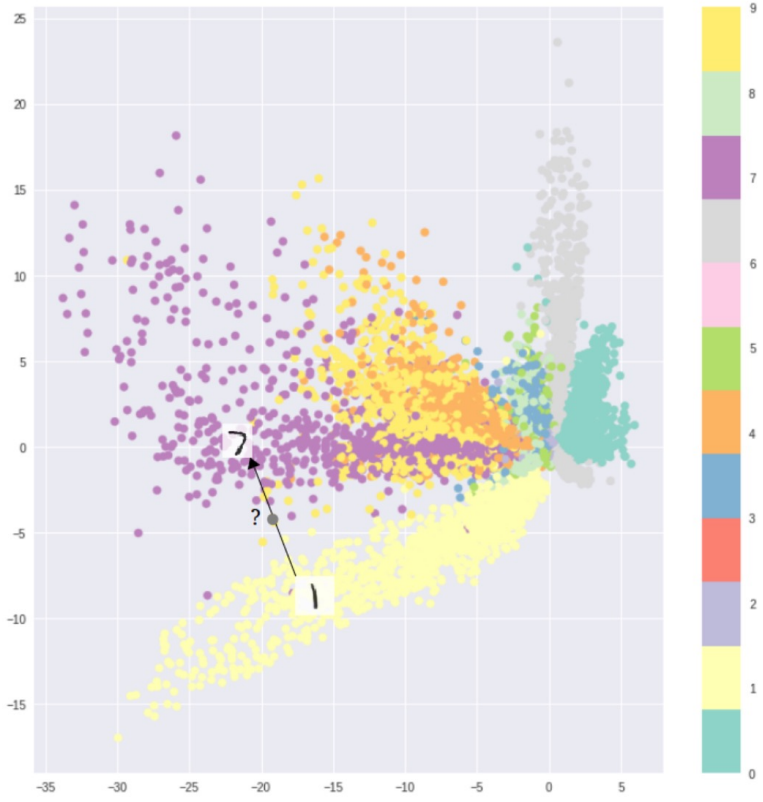
common latent dist: Gaussian and Uniform

Why not regular autoencoders?



- Optimized with Reconstruction Loss
- Not explicitly penalized for Generative Purposes

Latent Space Not Optimal for Generation



Optimizing purely for reconstruction loss

Formation of clusters helps decoding

Does not help generation

Latent space cannot be interpolated

lots of empty space in latent space

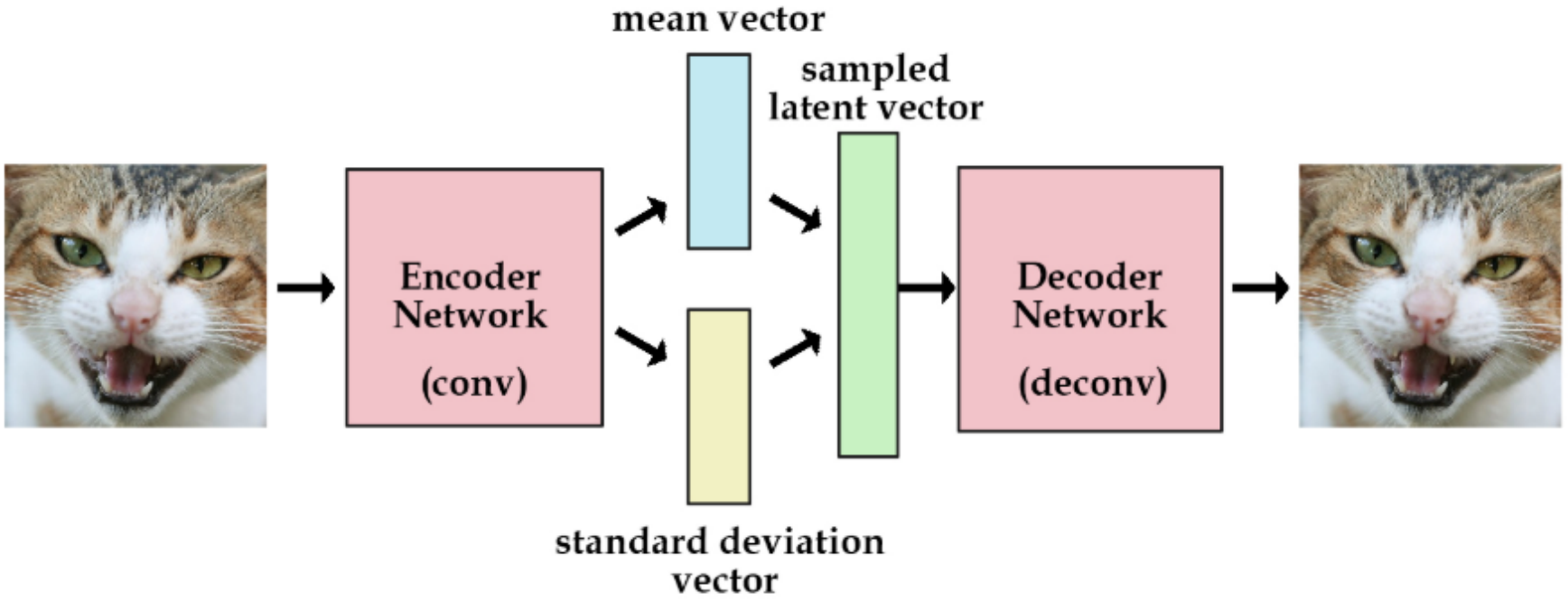
Variational Autoencoders (VAEs)

- Force the latent vectors to have (roughly) a unit Gaussian distribution

completely defined by

- Generate images by sampling a unit Gaussian and passing it into the decoder

mean and variance



Variational Autoencoders (VAEs)

reconstruction error

- Tradeoff between accuracy and the unit Gaussian approximation

constrain latent space to be Gaussian

- Build this directly into the loss
  - Kullback-Leibler (KL) divergence: a measure of difference between probability distributions P and Q

$$D_{KL}(P||Q) = -\sum_{i} P(i) \log\left(\frac{Q(i)}{P(i)}\right)$$

  - Include a regularization term that penalizes the KL divergence between a unit Gaussian and the latent vector distribution
  - How do we do this in practice?

Variational Autoencoders (VAEs)

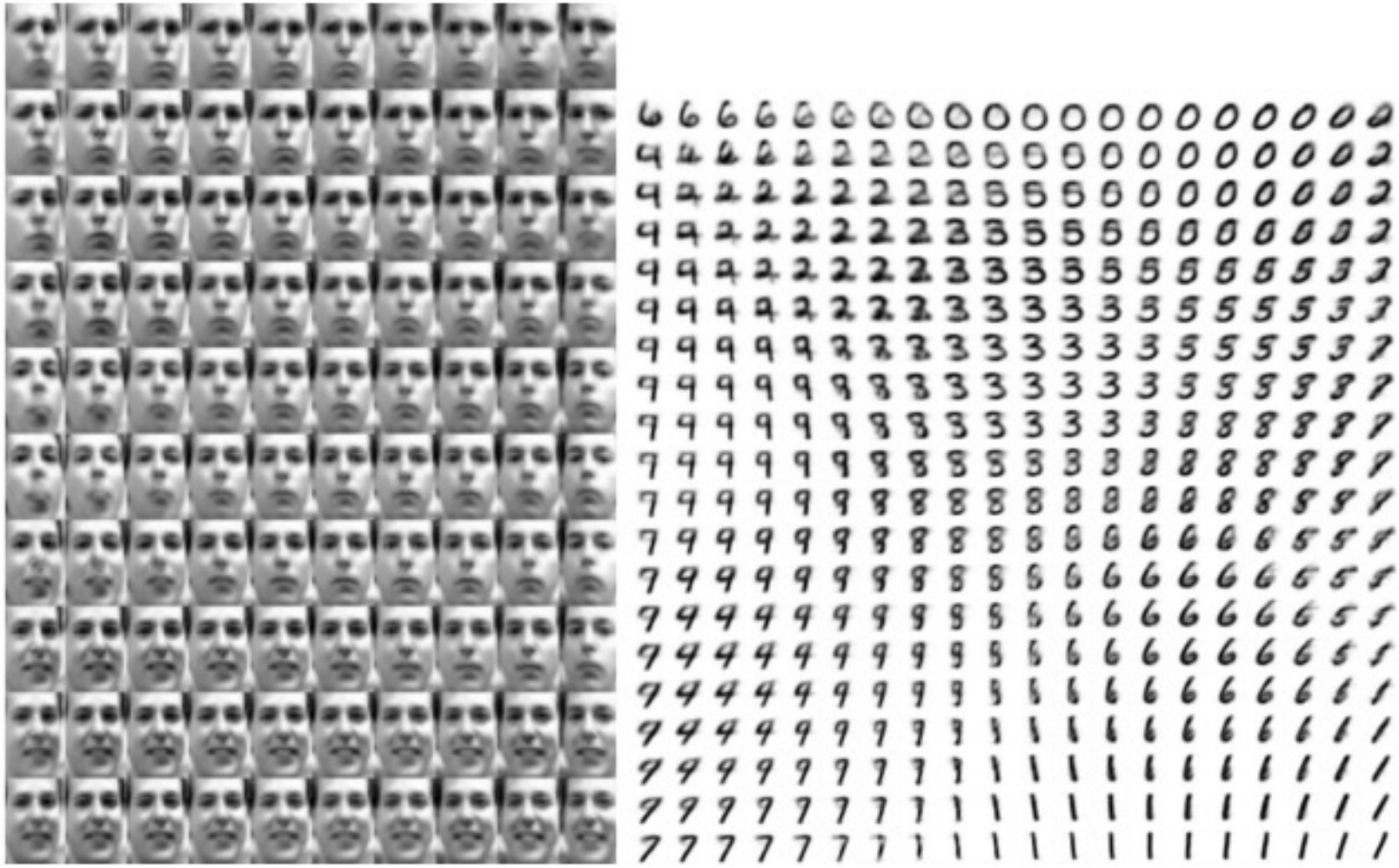
1st Epoch

9th Epoch

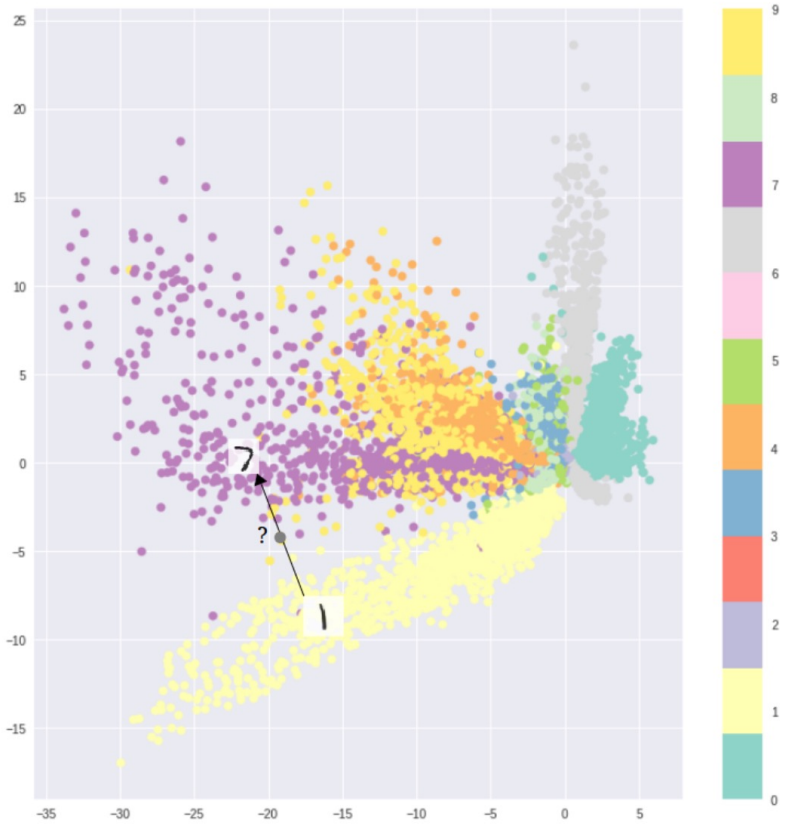
Original

Variational Autoencoders (VAEs)

2D latent mapping of results from VAEs

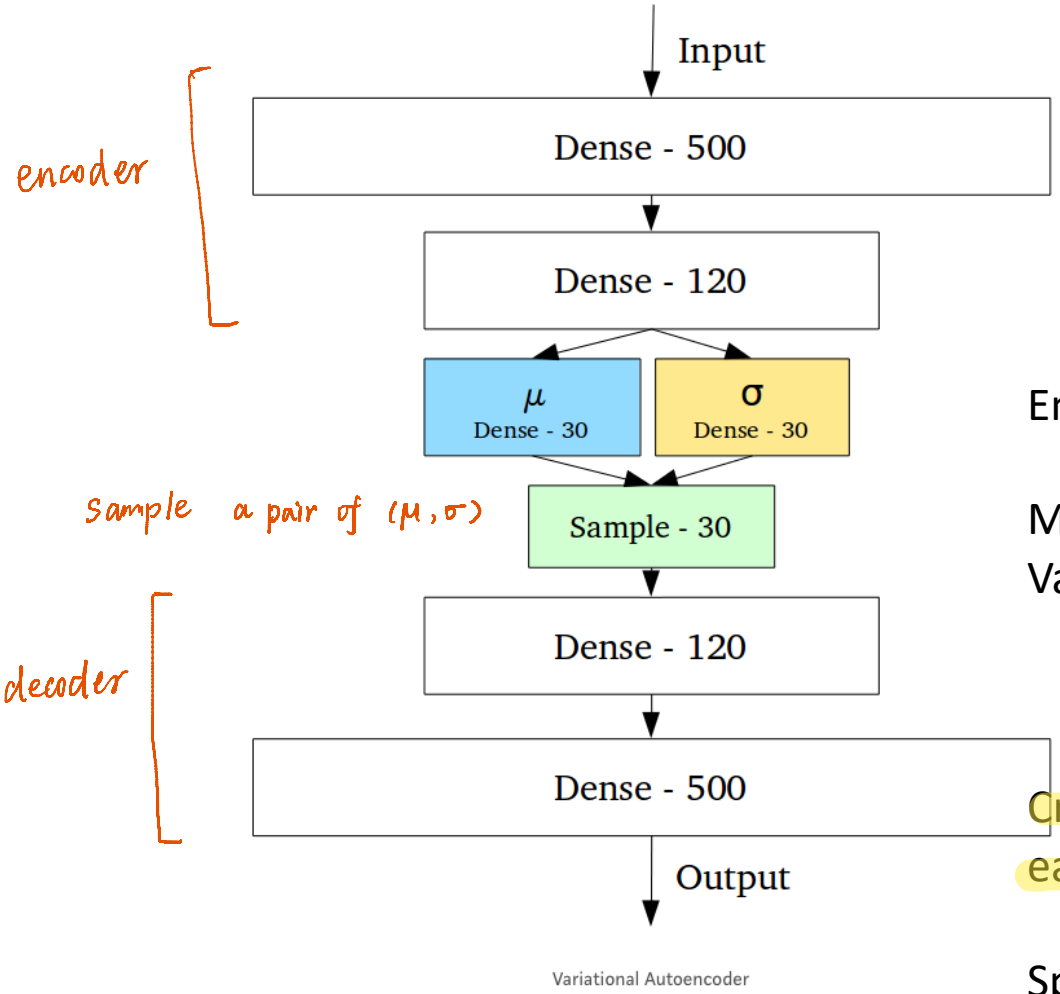


VAEs Revisited- Want Dense Middle Layer



Optimizing purely for reconstruction loss

Designed for Sampling



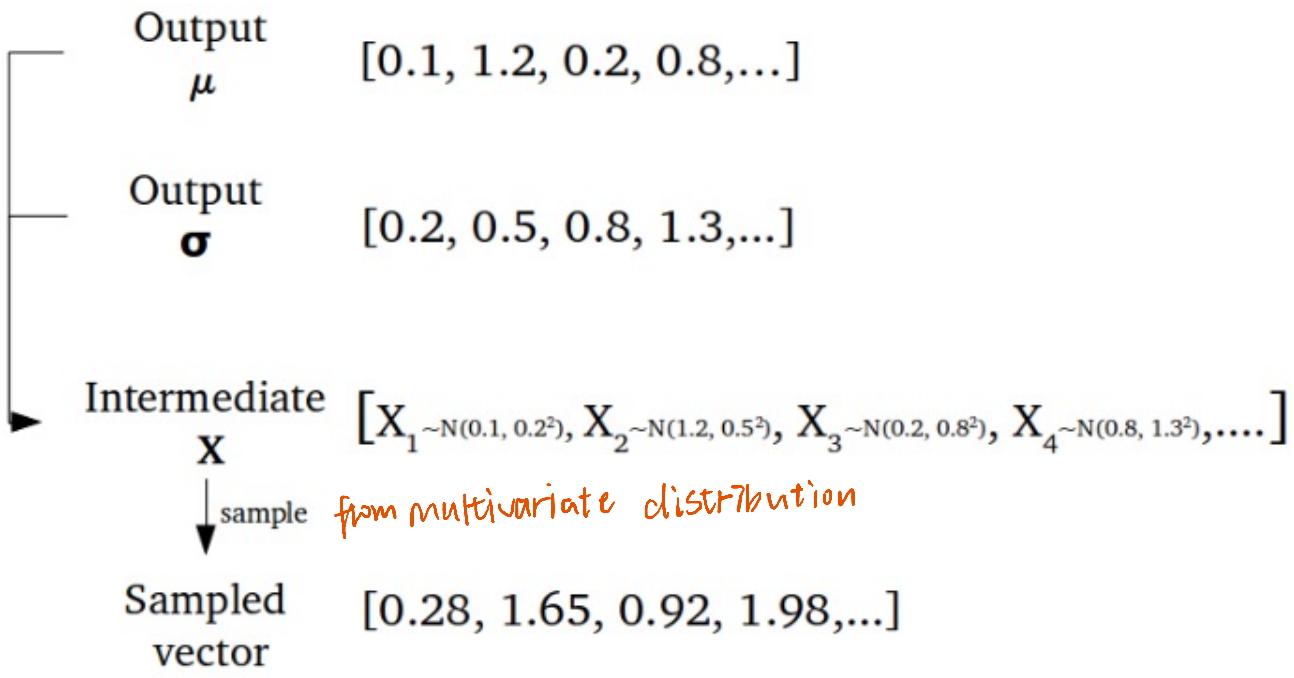
Encoding layer outputs two vectors

Means: $\{\mu_1, \mu_2, \mu_3 \ldots \mu_k\}$
Variances: $\{\sigma_1, \sigma_2, \sigma_3 \ldots \sigma_k\}$

Creates a K-dimensional Gaussian ball for each datapoint

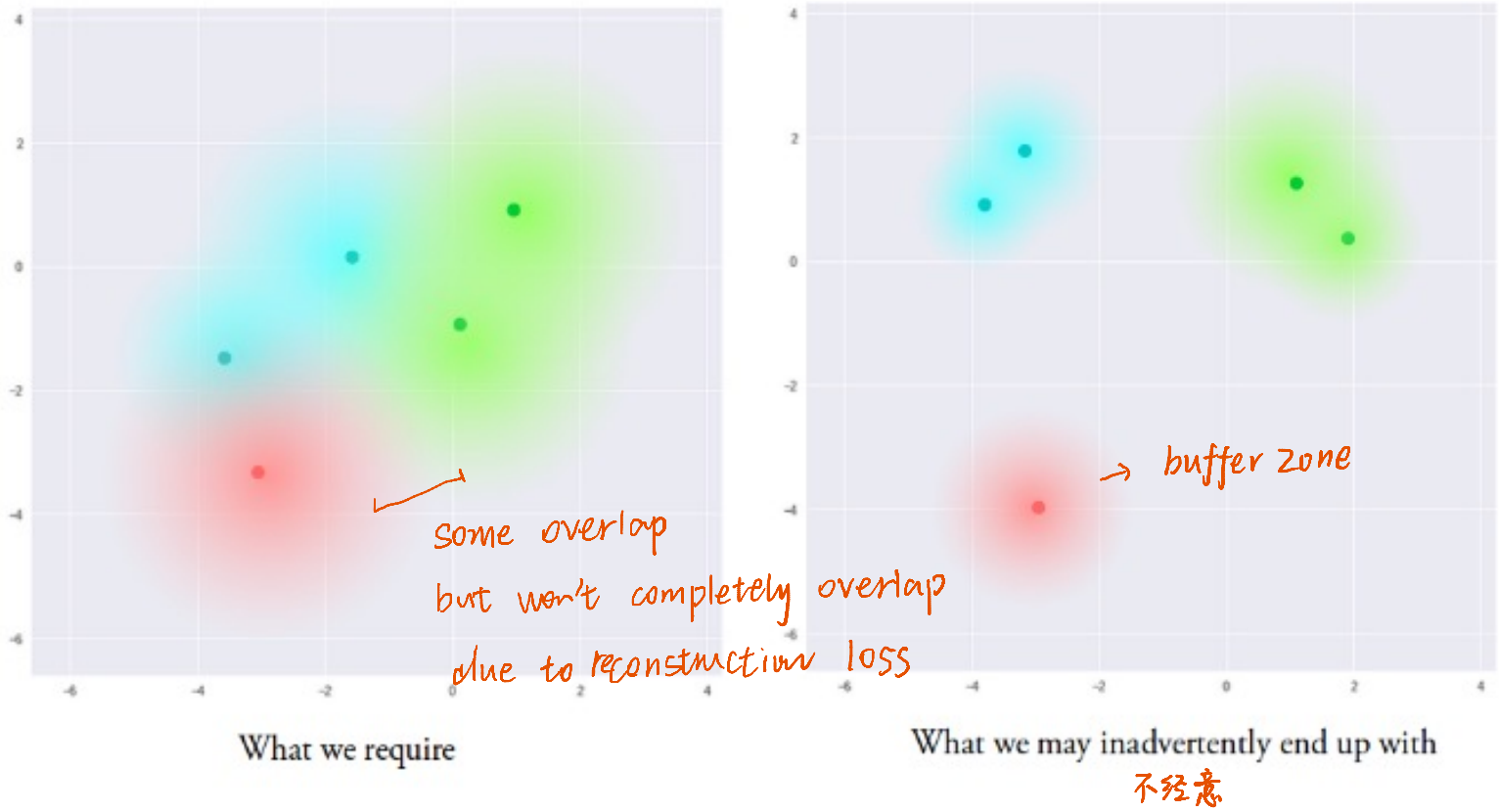
Splits latent space into Gaussian balls

Sampling a vector in latent space



Stochastically generating encoding vectors

Discouraging Clusters



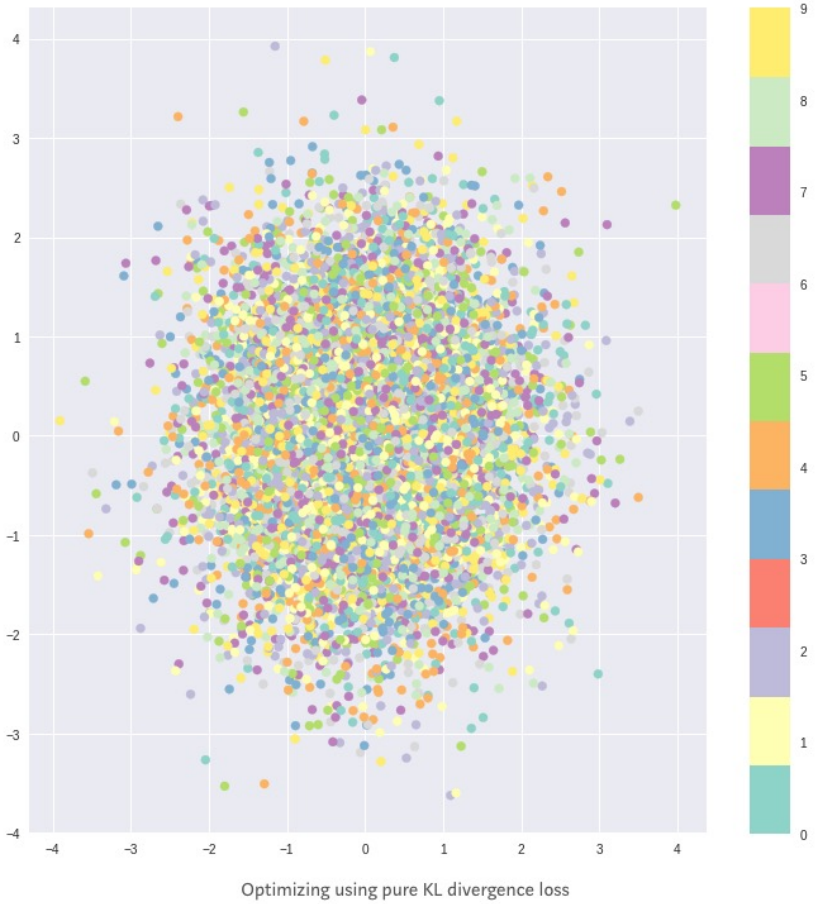
Solution: Penalize the Gaussian balls if they don't overlap!

KL Divergence Penalty

$$\mathbb{KL}(q_\theta(z \mid x_i) \mid\mid p(z))$$

Penalizing distributions
In the latent space from being
too far from a standard normal

$$p(z) = Normal(0, 1).$$

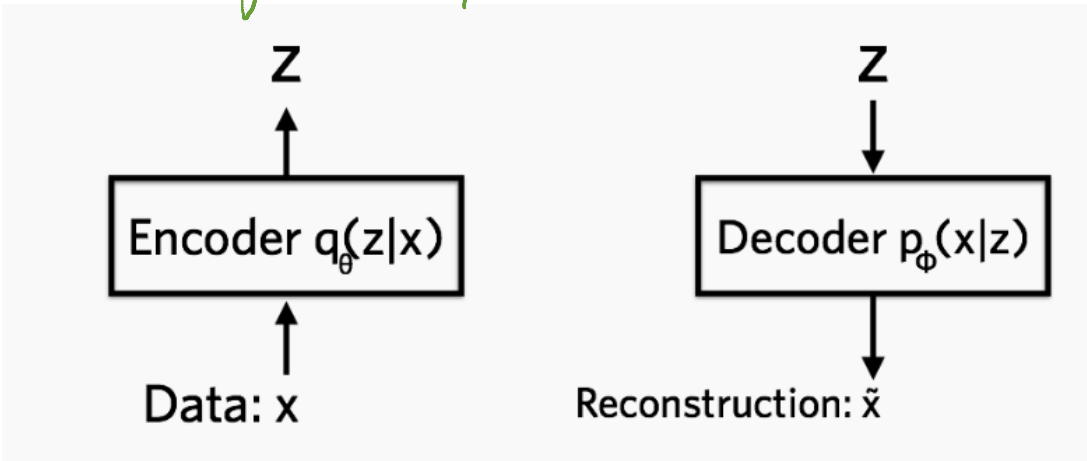


Encourages every ball to be the same!

Loss Function of VAE

$$l_i(\theta, \phi) = -\mathbb{E}_{z \sim q_\theta(z|x_i)}[\log p_\phi(x_i \mid z)] + \mathbb{KL}(q_\theta(z \mid x_i) \,||\, p(z))$$

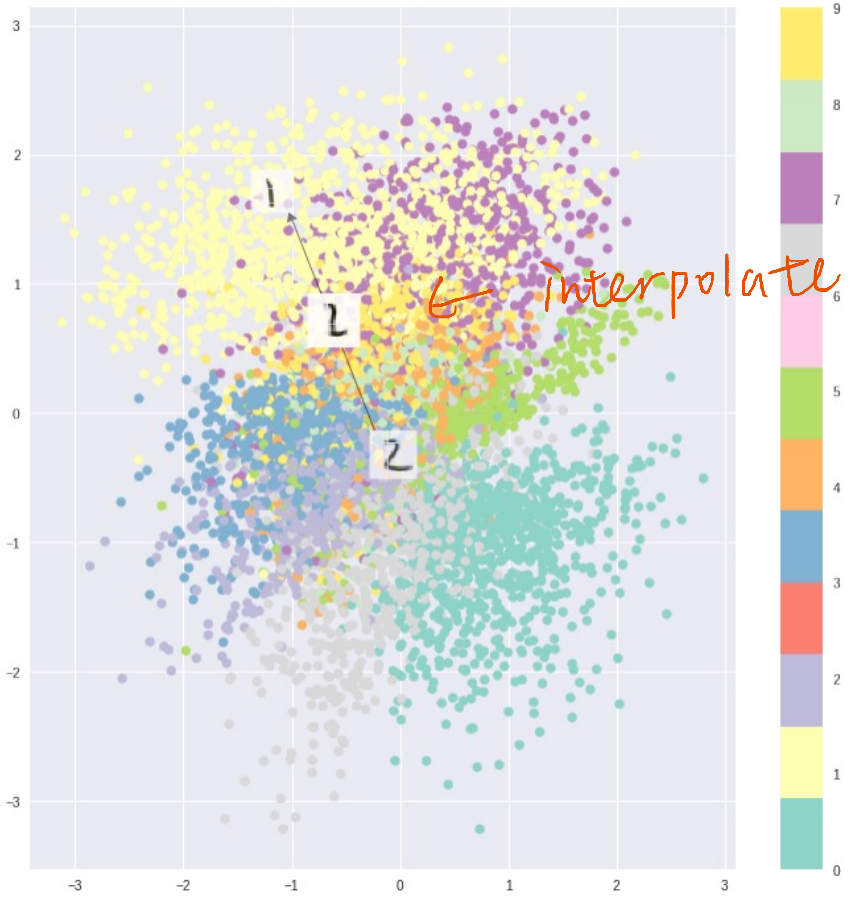
encourage separate clusters close



Reconstruction penalty encourages separation

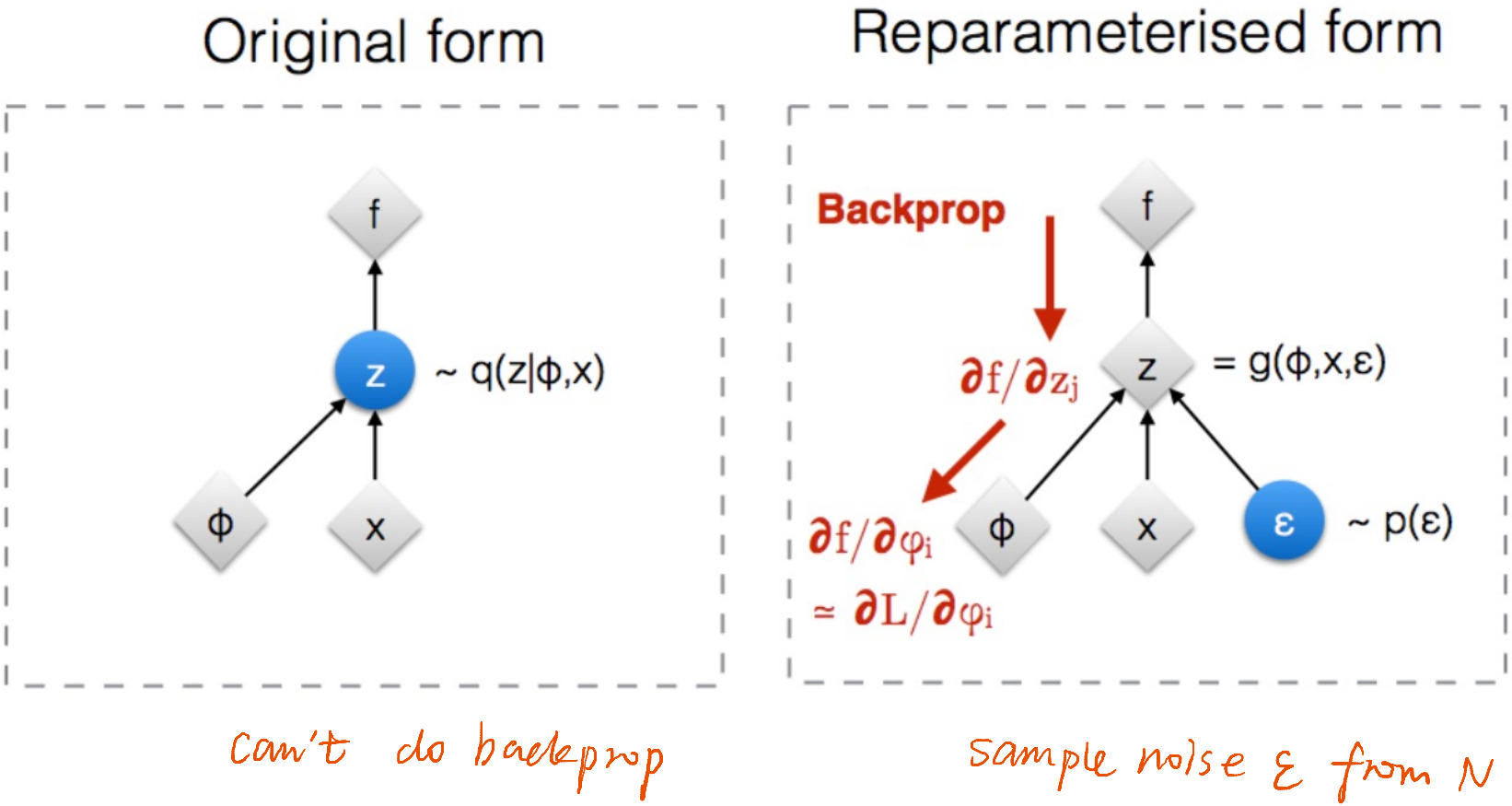
$$-\mathbb{E}_{z \sim q_\theta(z|x_i)}[\log p_\phi(x_i \mid z)]$$

KL+Reconstruction Loss



← interpolate between digit 1 and 2

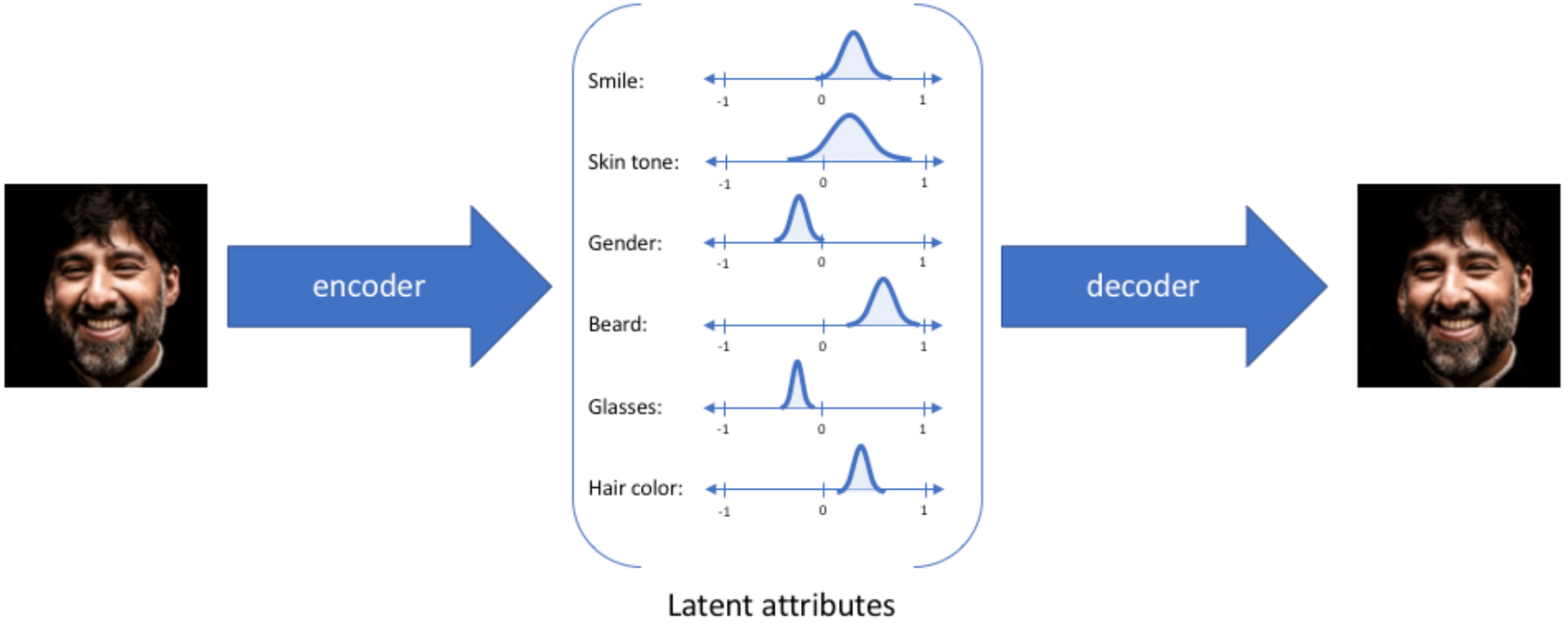
Training a VAE with sampling



can't do backprop

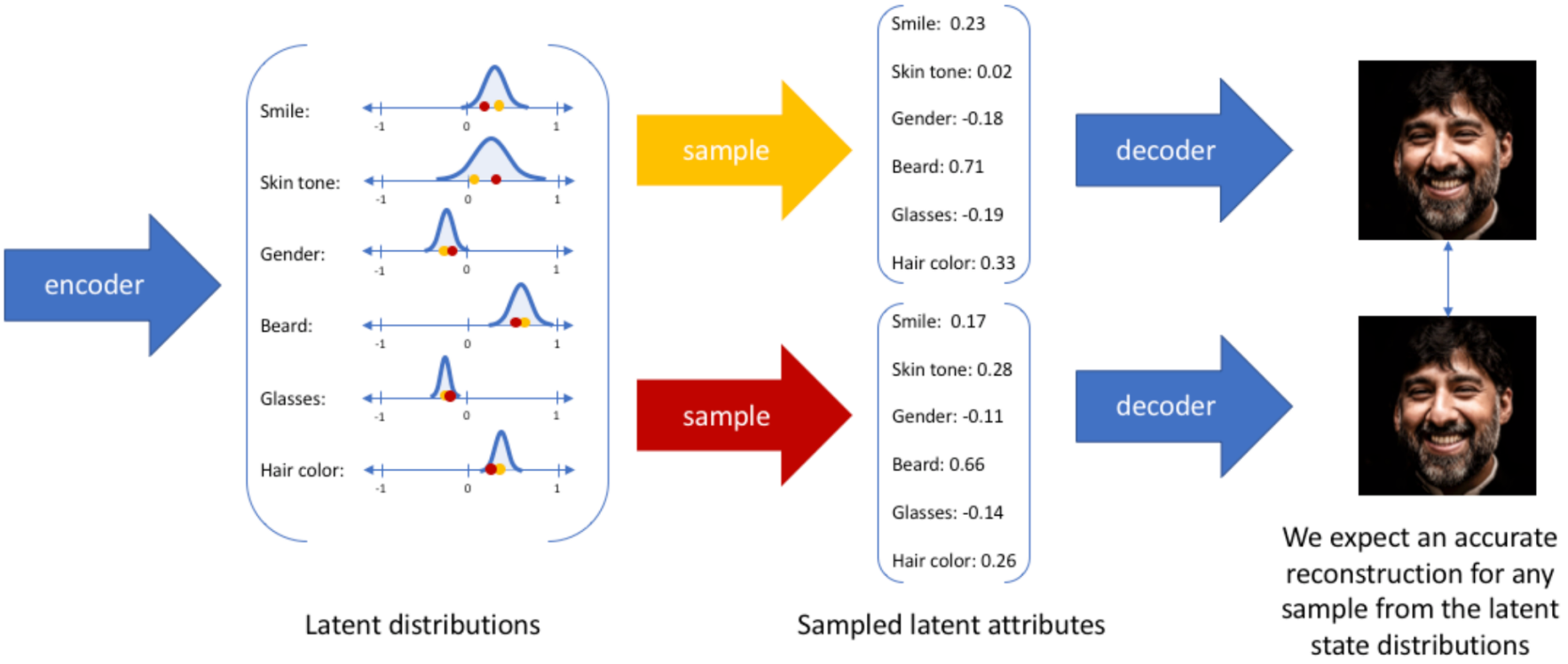
sample noise ε from $N(0, I)$

Example



From: https://www.jeremyjordan.me/variational-autoencoders/

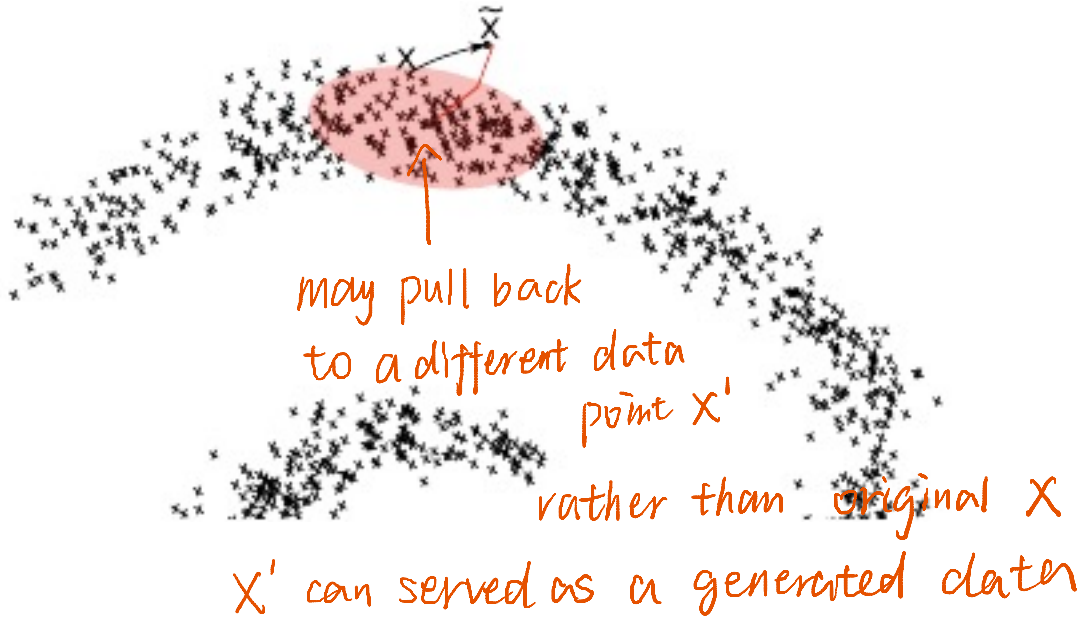
Sampling from latent space



Can we modify a regular autoencoder for generation?

Another approach

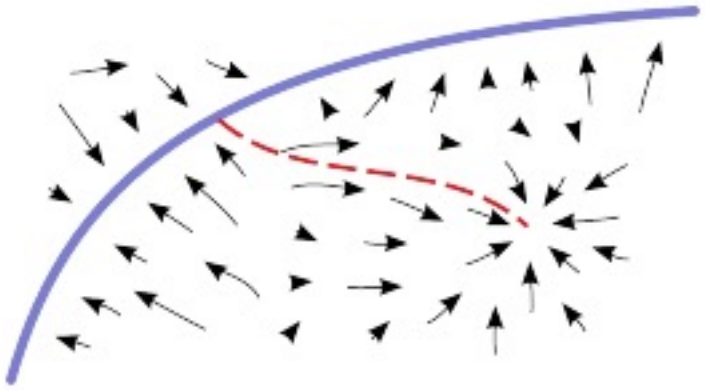
Generalized. Denoising Autoencoder
[Bengio et al. 2013]



denoising AE learn data manifold

- Basic idea: add noise to samples to push them away from the data manifold and then have them pull it back.
- For each training sample X define a corruption process C($\tilde{X}$,X) that creates a corrupted sample $\tilde{X}$.
- Train a denoising autoencoder to reverse this by using (X, $\tilde{X}$) as a training example

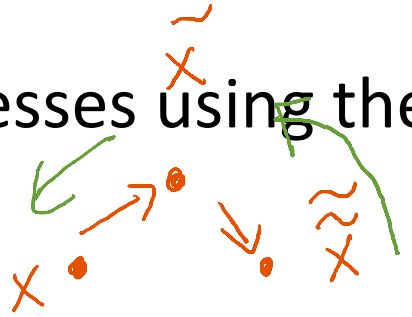
Lost away from the manifold?



- Problem: spurious modes 假模式
- The DAE may not be able to walk back well enough if you get too far away from the sample space

Walkback training

- Train the neural network to walk back from several steps away
- Create longer range corruption processes using the original corruption process
- Sample a second step $C(\tilde{X}', \tilde{X})$
- Add the training sample $(\tilde{X}', \tilde{X})$ to the training



Without walkback

With walkback

Trains the DAE to estimate conditional

- This way of training actually trains the DAE to estimate a conditional probability distribution $\mathrm{P}(\mathrm{X}|\tilde{X})$
- [Bengio et al. 2013] show that a consistent estimator of P(X), i.e. distribution of the training example can be recovered by alternating sampling from the corruption process. $\mathrm{C}(\tilde{X},\mathrm{X})$ and the denoising process $\mathrm{P}(\mathrm{X}|\tilde{X})$
- Turns out that learning a conditional distribution is a lot simpler than learning the joint distribution!
  - This idea is used in style transfer and other places
- A conditional distribution can just be a Gaussian or something simple, but this alternation allows convergence towards the joint distribution

Issue with this AE

- Data generation process is very slow
- Its like taking a slow walk through the data
- May not span the space
- Not penalized to generate the distribution

SCVI

Article | Published: 30 November 2018

Deep generative modeling for single-cell transcriptomics

Romain Lopez, Jeffrey Regier, Michael B. Cole, Michael I. Jordan & Nir Yosef

Nature Methods **15**, 1053–1058(2018) | Cite this article

Generative Process Model

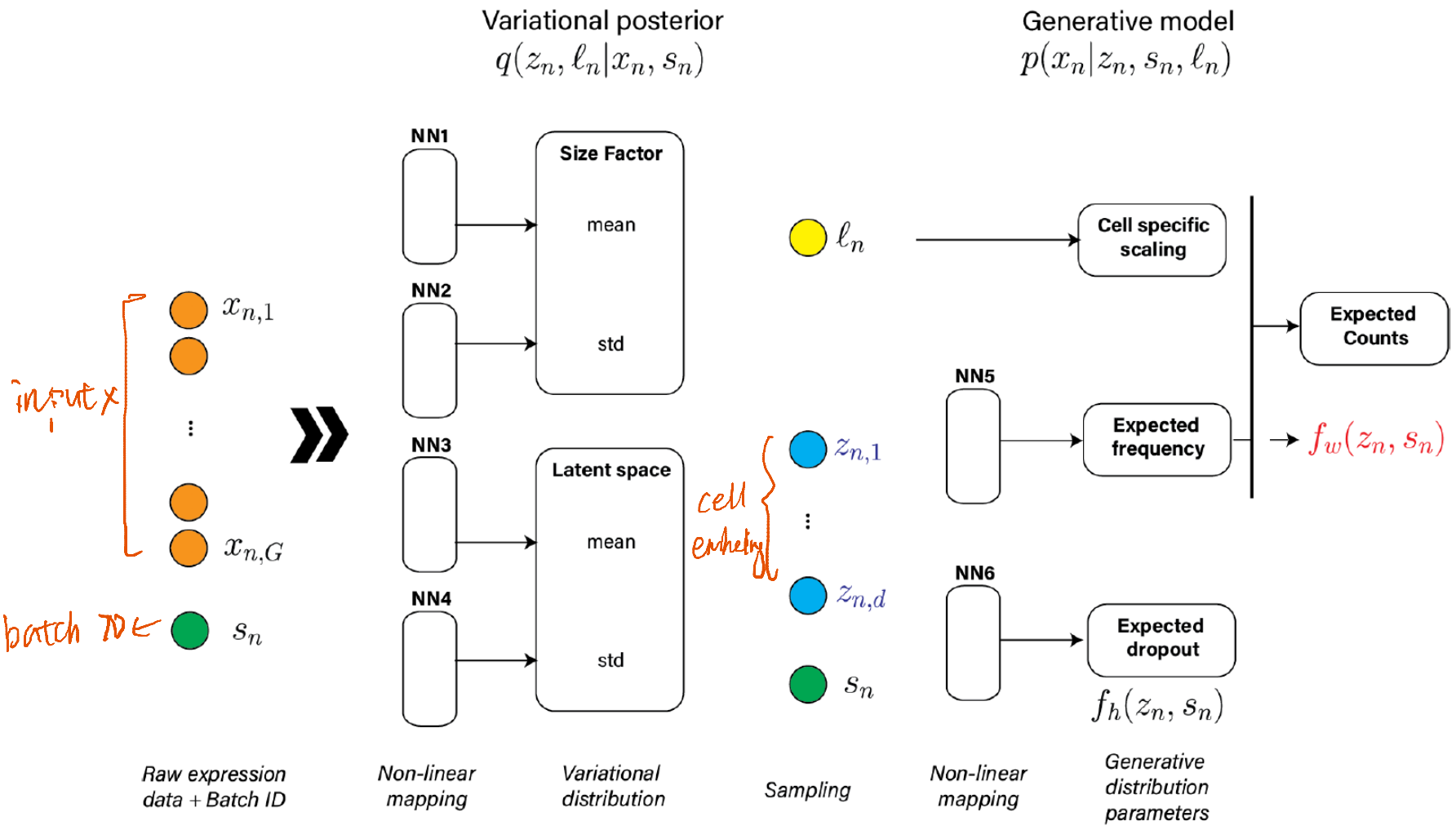
$$z_n \sim \text{Normal}(0, I)$$ Cell embedding

$$\ell_n \sim \text{LogNormal}(\ell_\mu, \ell_\sigma^2)$$ Library size

$$\rho_n = f_w(z_n, s_n)$$ Normalized expression

$$\pi_n = f_h(z_n, s_n)$$ Dropout rate

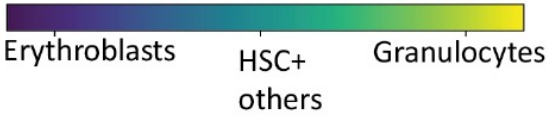
$$x_{ng} \sim \text{ZINB}(\ell_n \rho_{ng}, \theta_g, \pi_{ng})$$ Raw data



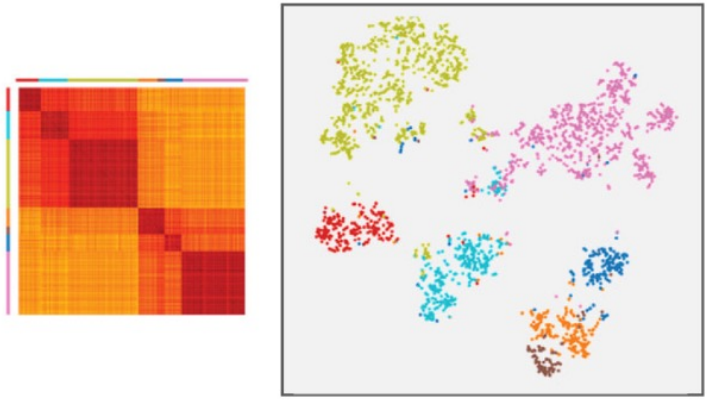
s_n can be used to correct batch effect
l_n can be used to normalize
z_n can be used to visualize

Hierarchical clusters:
~3k cortex cells
(Zeisel et al 2015)

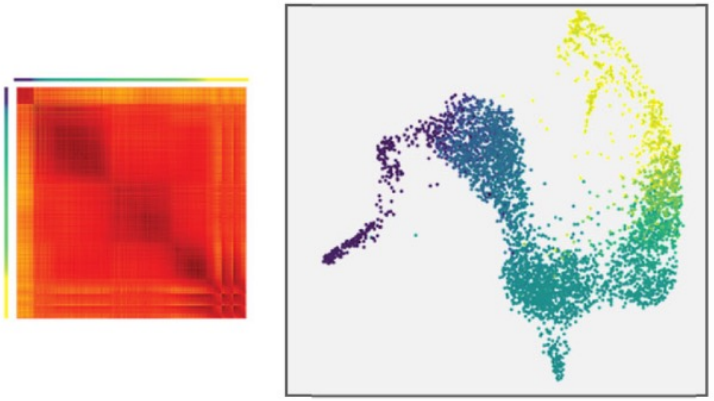
Developmental gradient:
hematopoiesis ~4k cells
(Tusi et al 2018)



Cell- cell similarity (Matrix/ tSNE)



Cell- cell similarity (Matrix/ tSNE)



Batch removal

Suggested Reading Blogs

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

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

Goodfellow et al. Chapter 20

shape latent variable using geometric or probabilistic way