Lecture 19: Variational Autoencoders

CS109B Data Science 2

Pavlos Protopapas and Mark Glickman

Outline

Motivation for Variational Autoencoders (VAE)

Mechanics of VAE

Seperability of VAE

The math behind everything

Generative models

Outline

Motivation for Variational Autoencoders (VAE)

Mechanics of VAE

Seperability of VAE

The math behind everything

Generative models

Outline

Motivation for Variational Autoencoders (VAE)

Mechanics of VAE

Seperability of VAE

The math behind everything

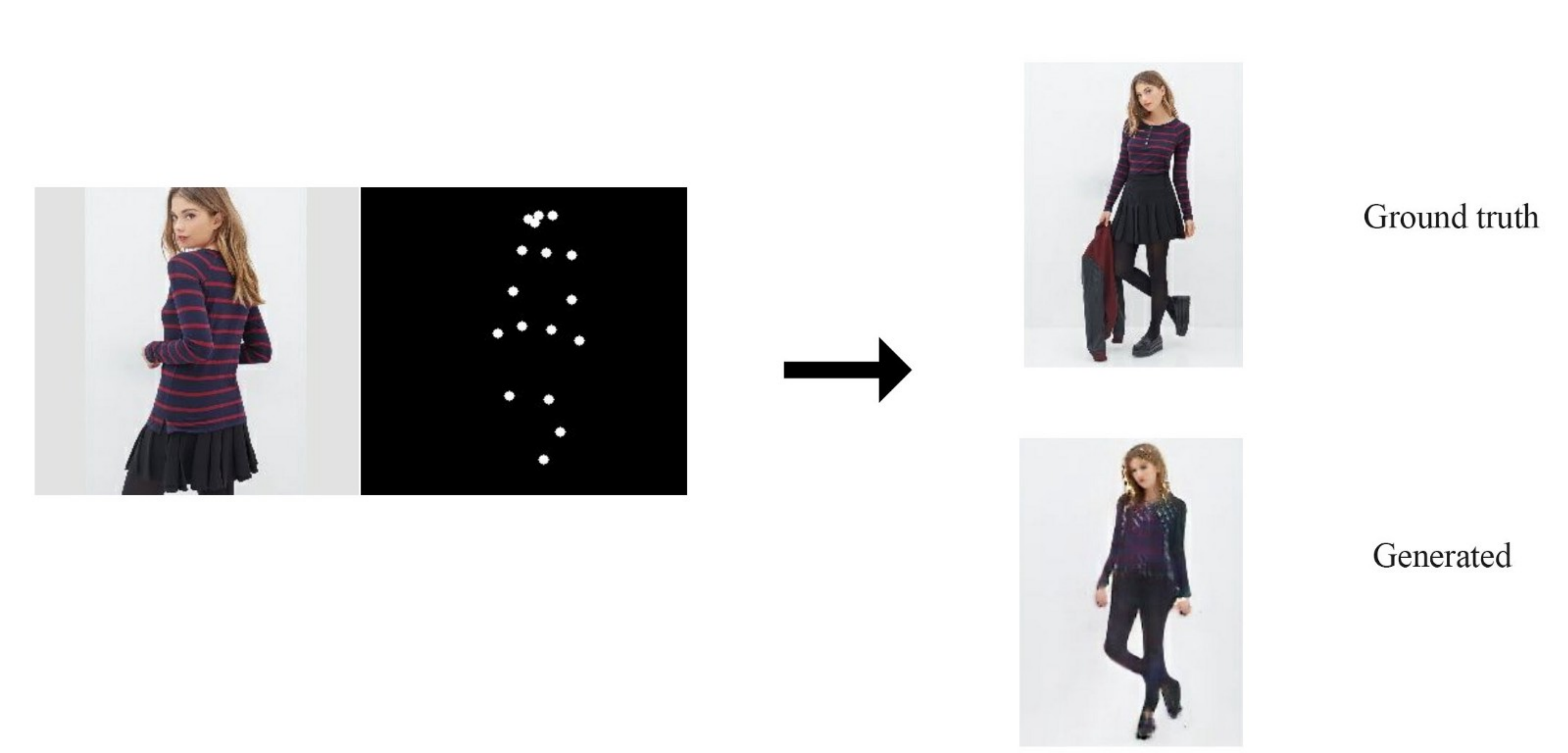
Generative models

Generating Data (is exciting)

Figure 7: Generated samples

https://arxiv.org/pdf/1708.05509.pdf

Generating Data (is exciting)



Generating Data (is exciting)

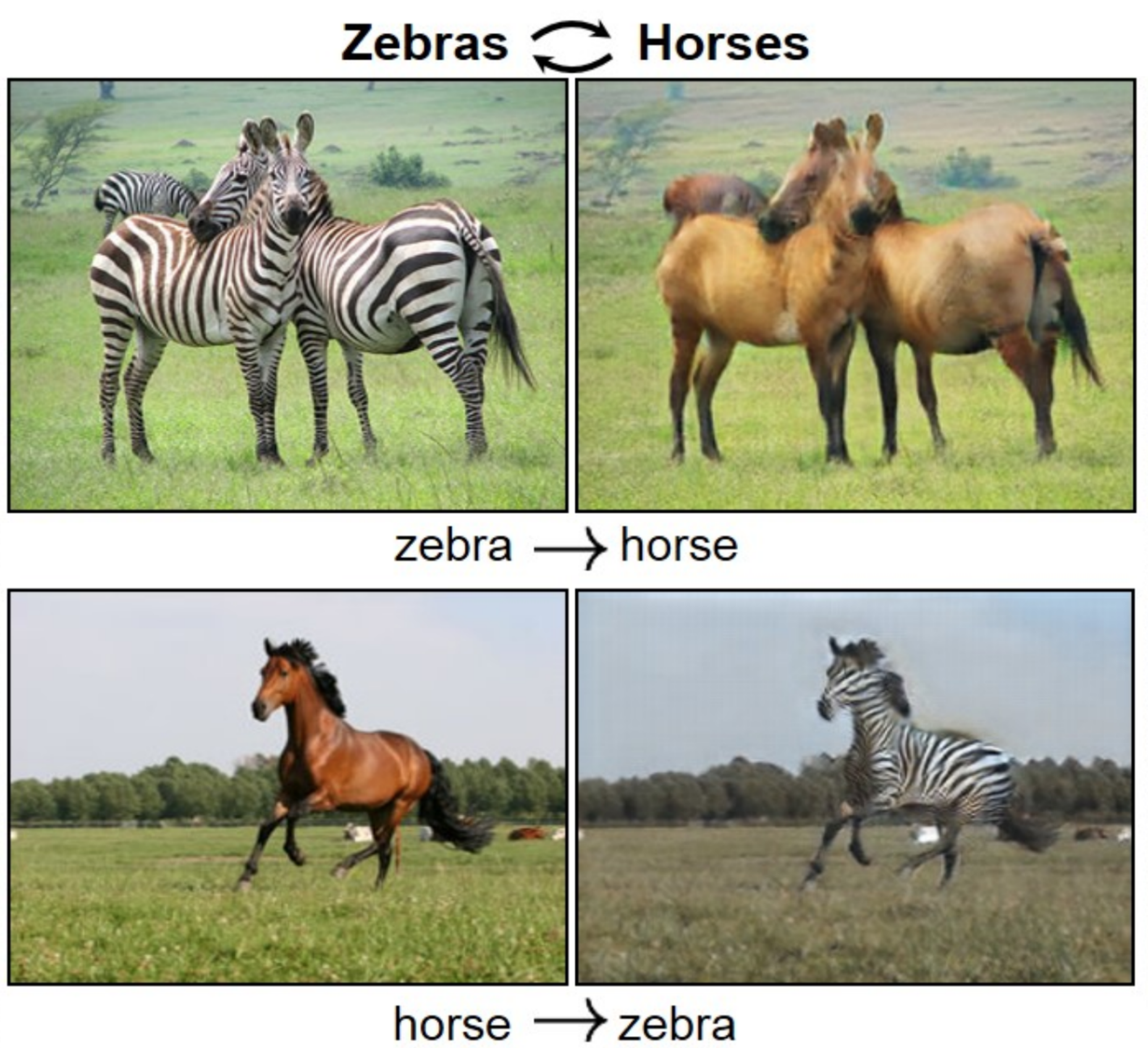


Figure 5: 1024×1024 images generated using the CELEBA-HQ dataset. See Appendix F for a larger set of results, and the accompanying video for latent space interpolations.

(a) MidiNet model 1

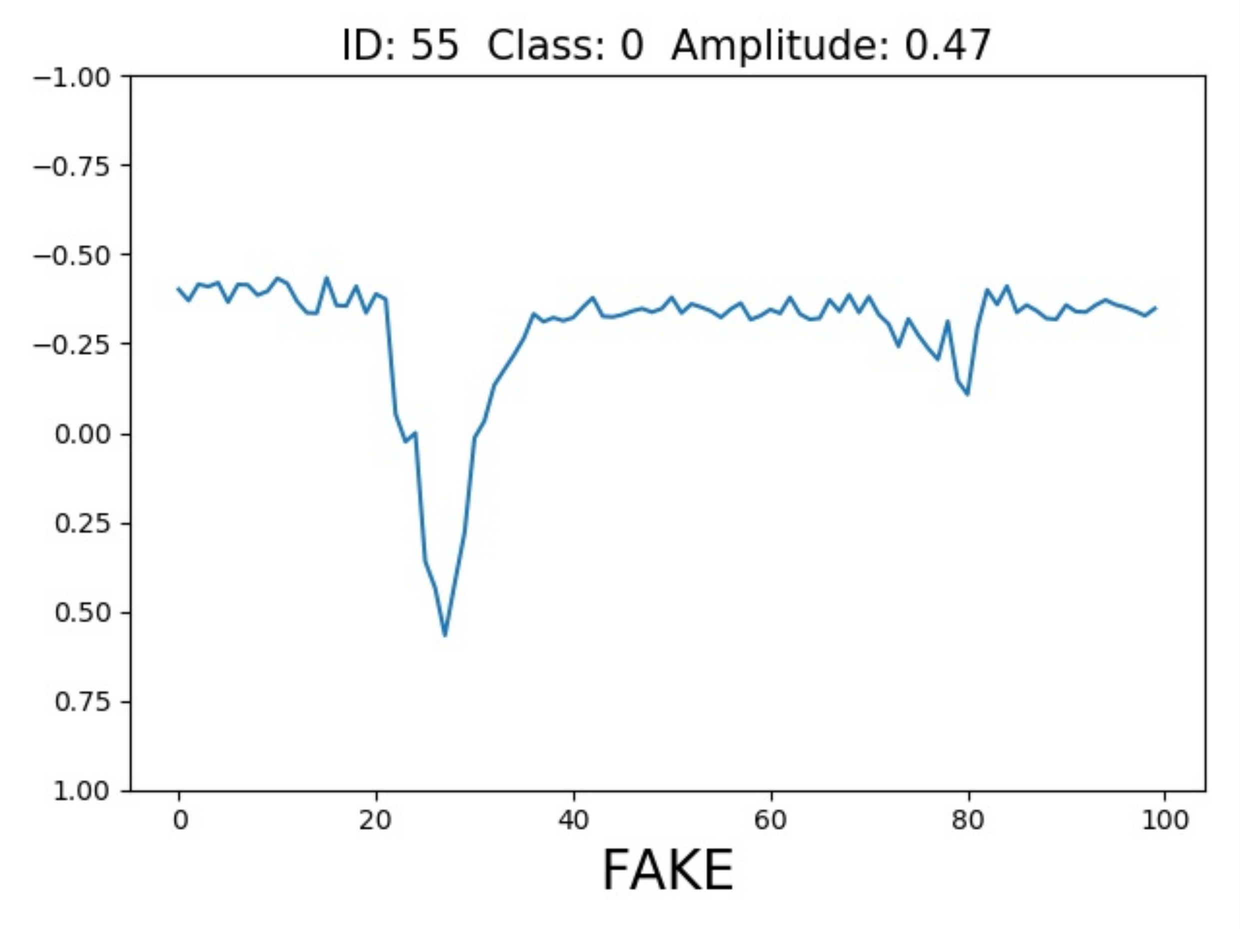
(b) MidiNet model 2

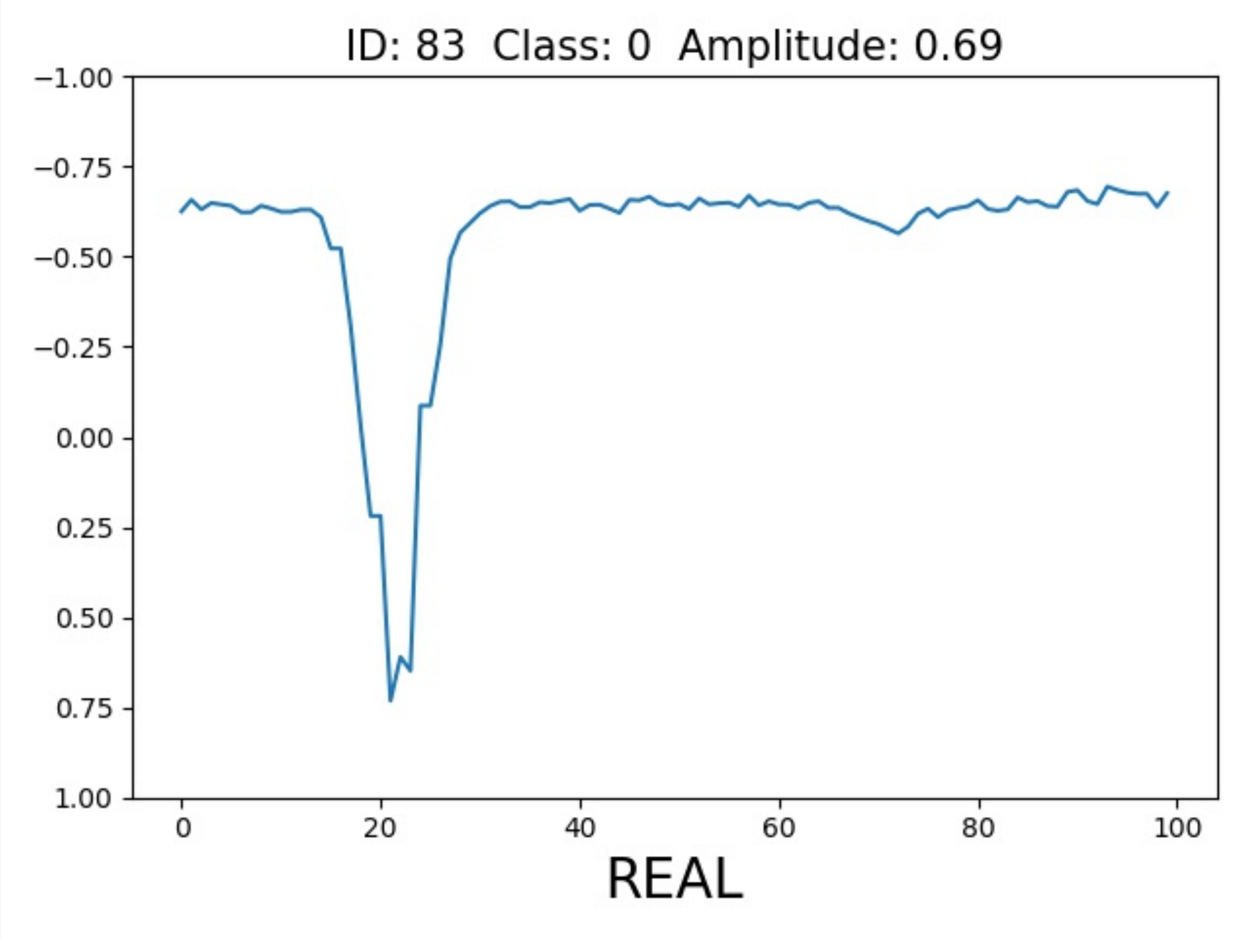
(c) MidiNet model 3

Figure 3. Example result of the melodies (of 8 bars) generated by different implementations of MidiNet.

Another use of generating new data is to give us ideas and options. Suppose we're planning a house. We can give the computer the space we have available, and its location. From this, the computer can can give us some ideas.

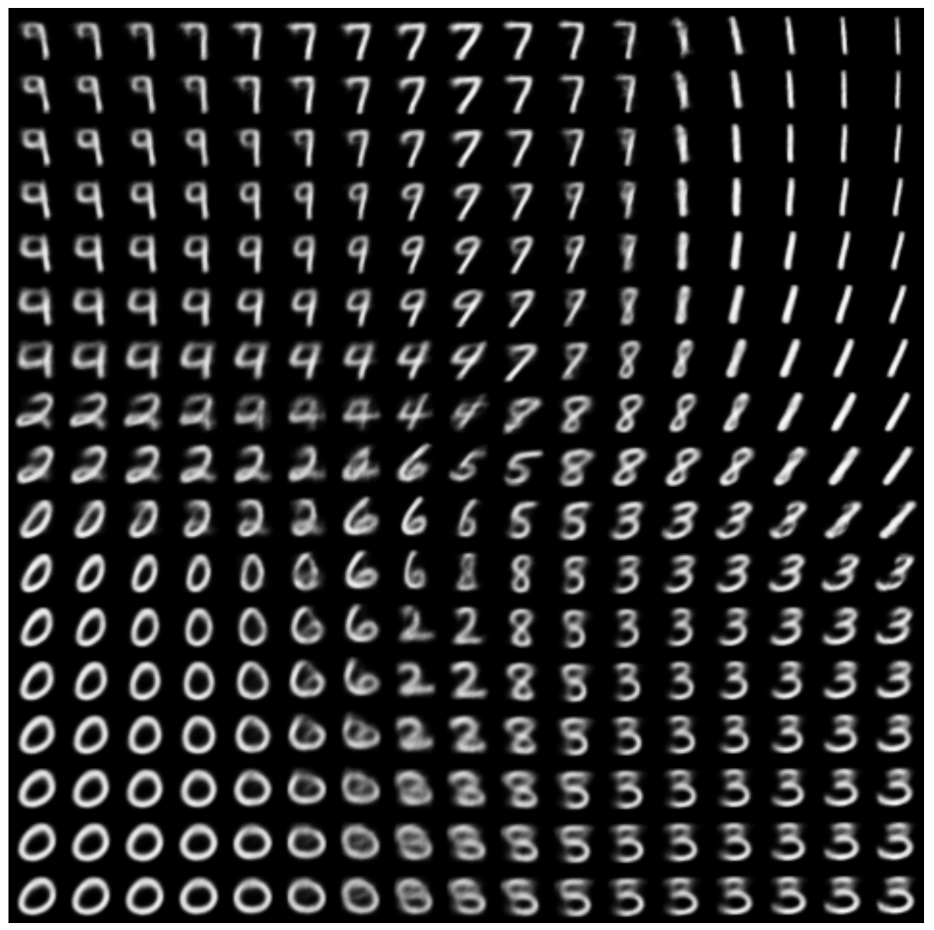
Big networks require big data, and getting high-quality, labeled data is difficult. If we're generating that data our selves, we can make as much of it as we like.





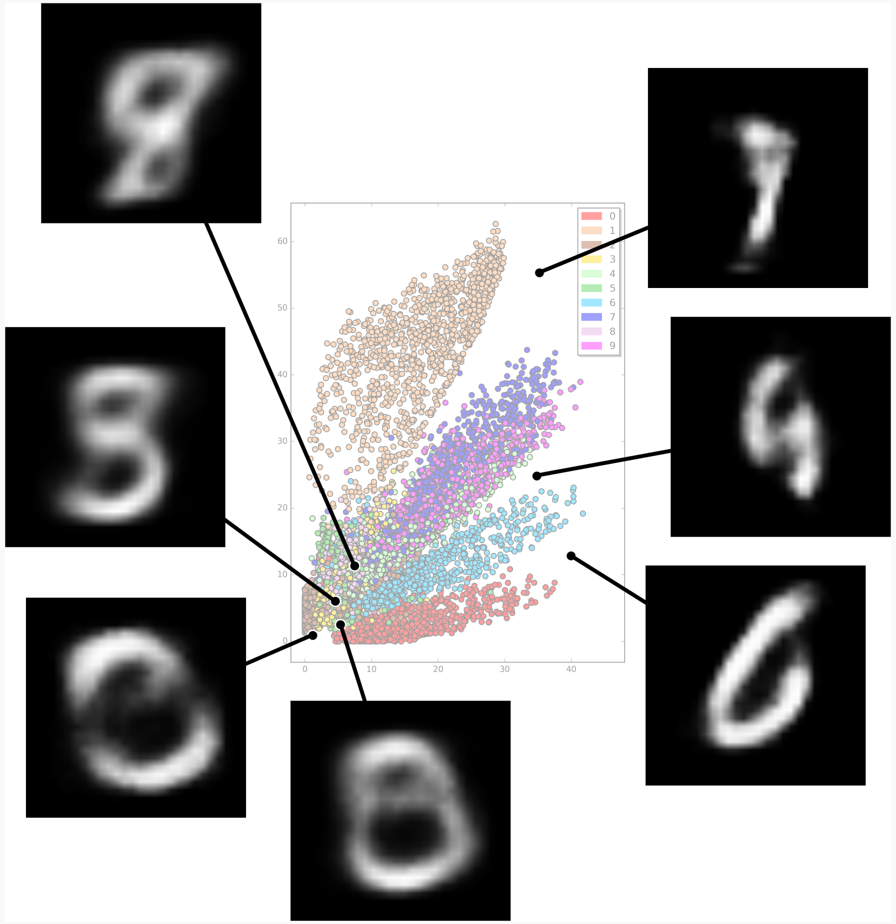
Generating Data

We saw how to generate new data with a AE in Lecture 18.



Problems with Autoencoders

- Gaps in the latent space
- Discrete latent space
- Separability in the latent space



Outline

Motivation for Variational Autoencoders (VAE)

Mechanics of VAE

Seperability of VAE

The math behind everything

Generative models

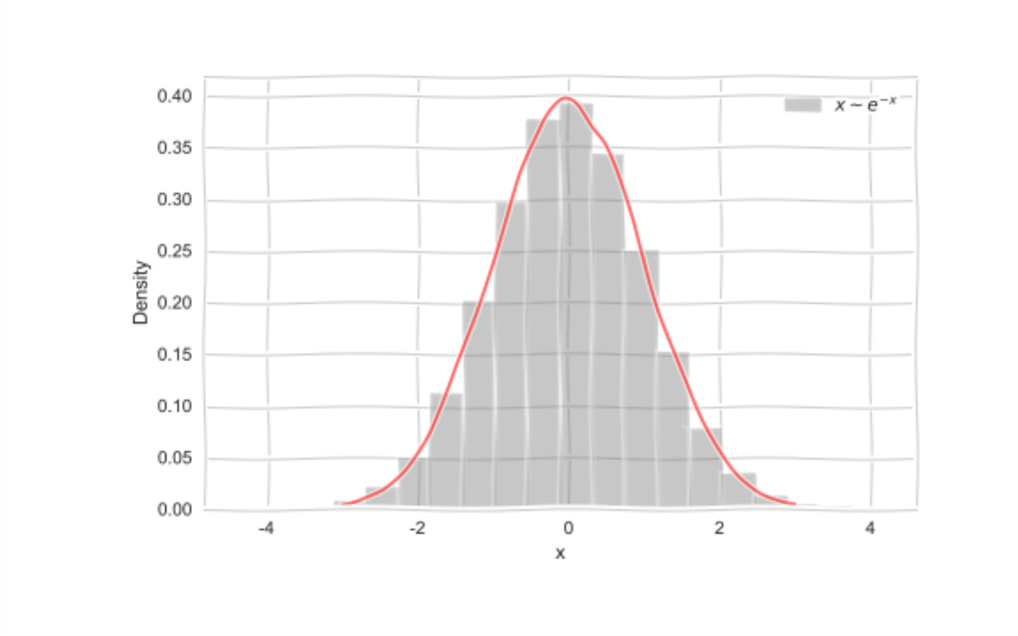
Generative models

Imagine we want to generate data from a distribution,

$$x \sim p(x)$$

e.g.

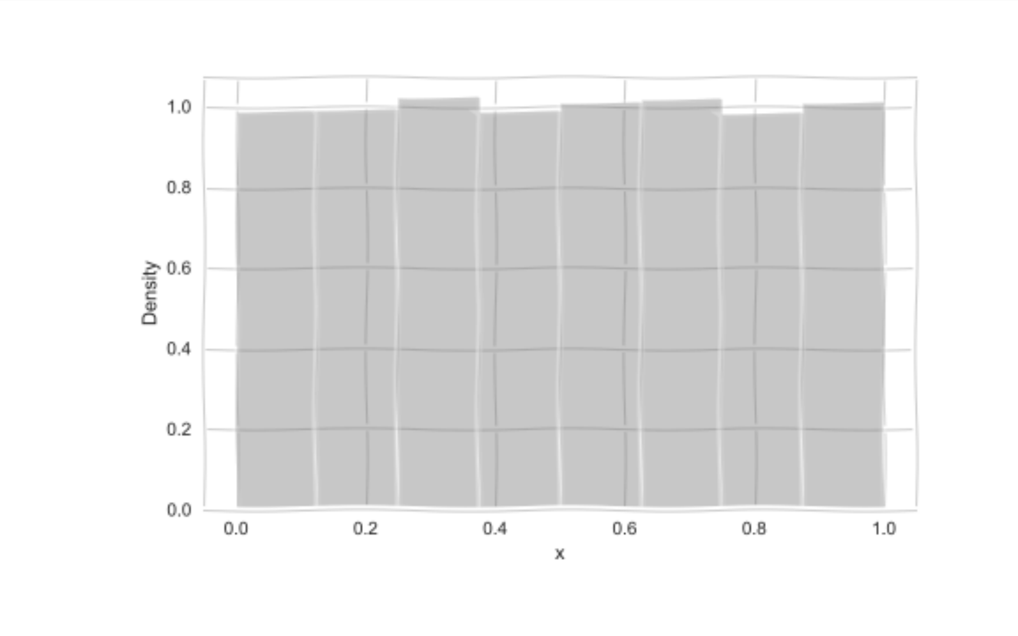
$$x \sim \mathcal{N}(\mu, \sigma)$$



Generative models

But how do we generate such samples?

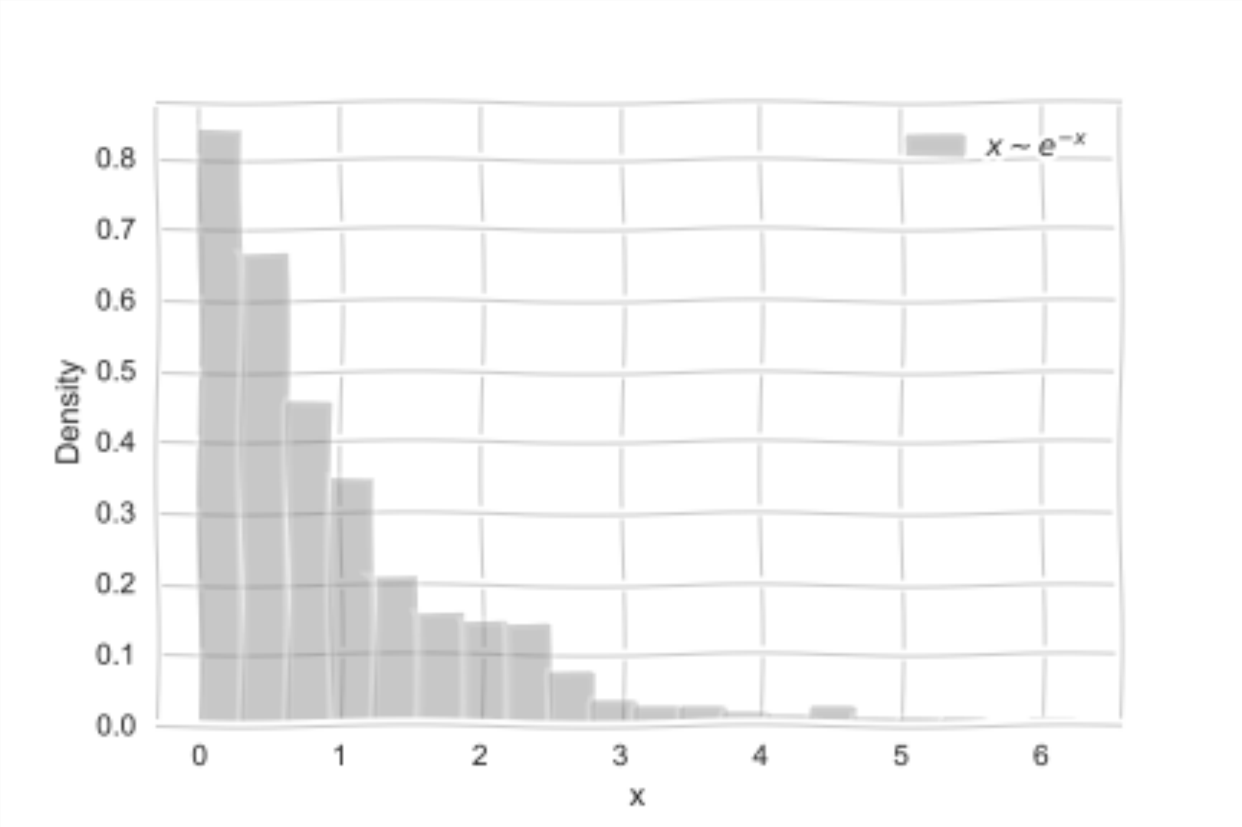
$$z \sim \mathrm{Unif}(0, 1)$$



Generative models

But how do we generate such samples?

$$z \sim \mathrm{Unif}(0,1) \quad \mathrm{x} = \ln \mathrm{z}$$



Generative models

In other words we can think that if we choose $\mathbf{z \sim \boldsymbol{Uniform}}$ then there is a mapping:

$$x = f(z)$$

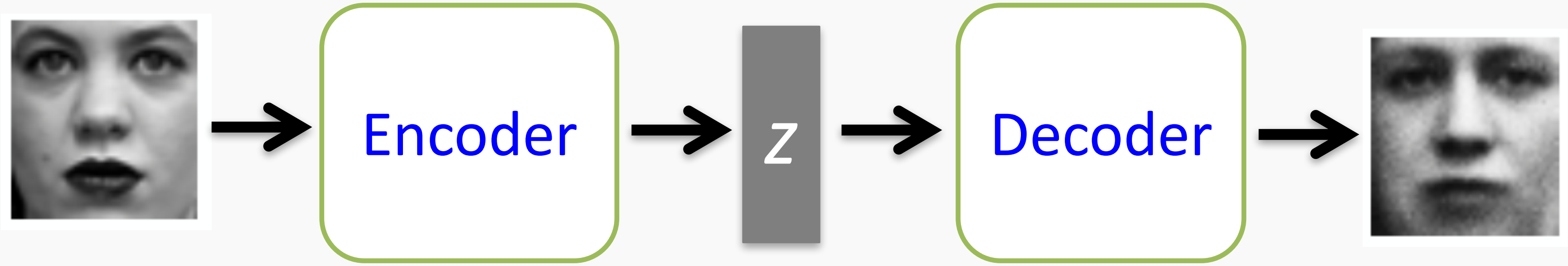
such as:

$$x \sim p(x)$$

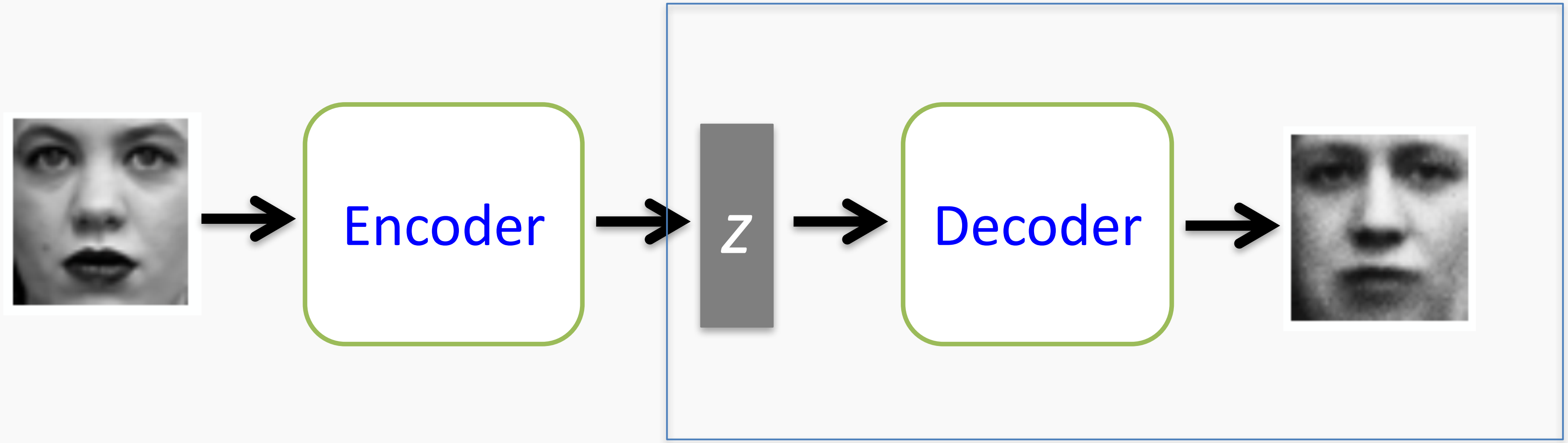
where in general f is some complicated function.

We already know that Neural Networks are great in learning complex functions.

Variational Autoencoders



Variational Autoencoders



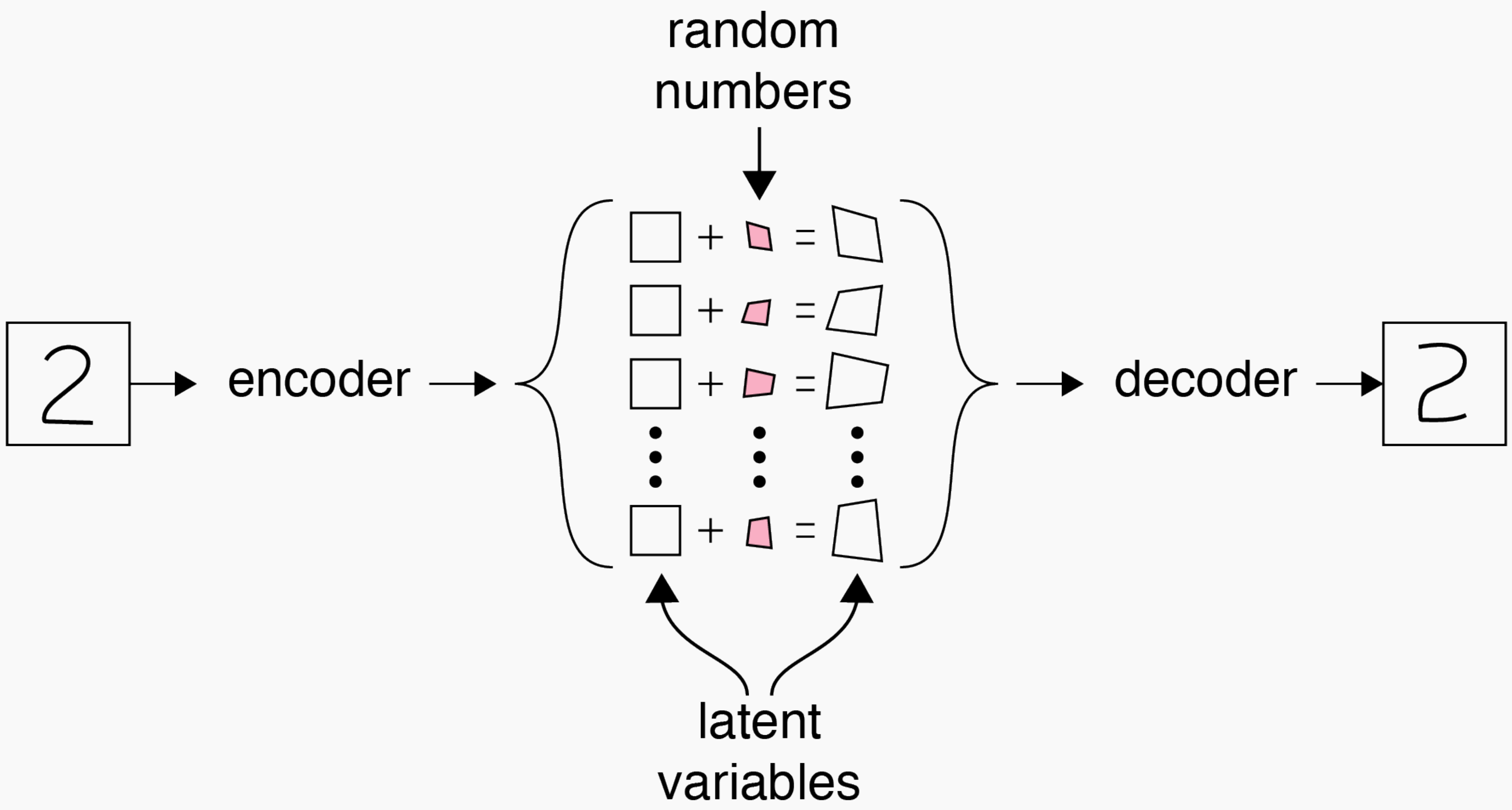
Variational Autoencoders

Sample from $P(z)$

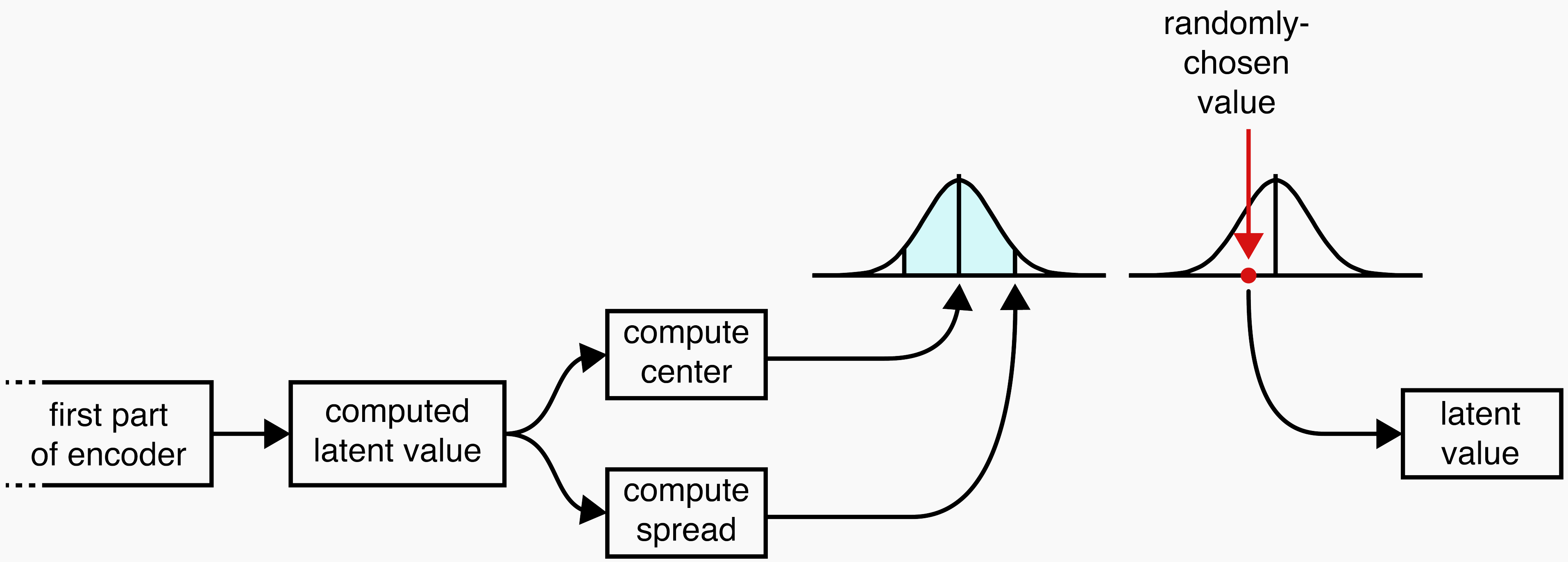
Standard Gaussian

z → Decoder $P(x|z)$ →

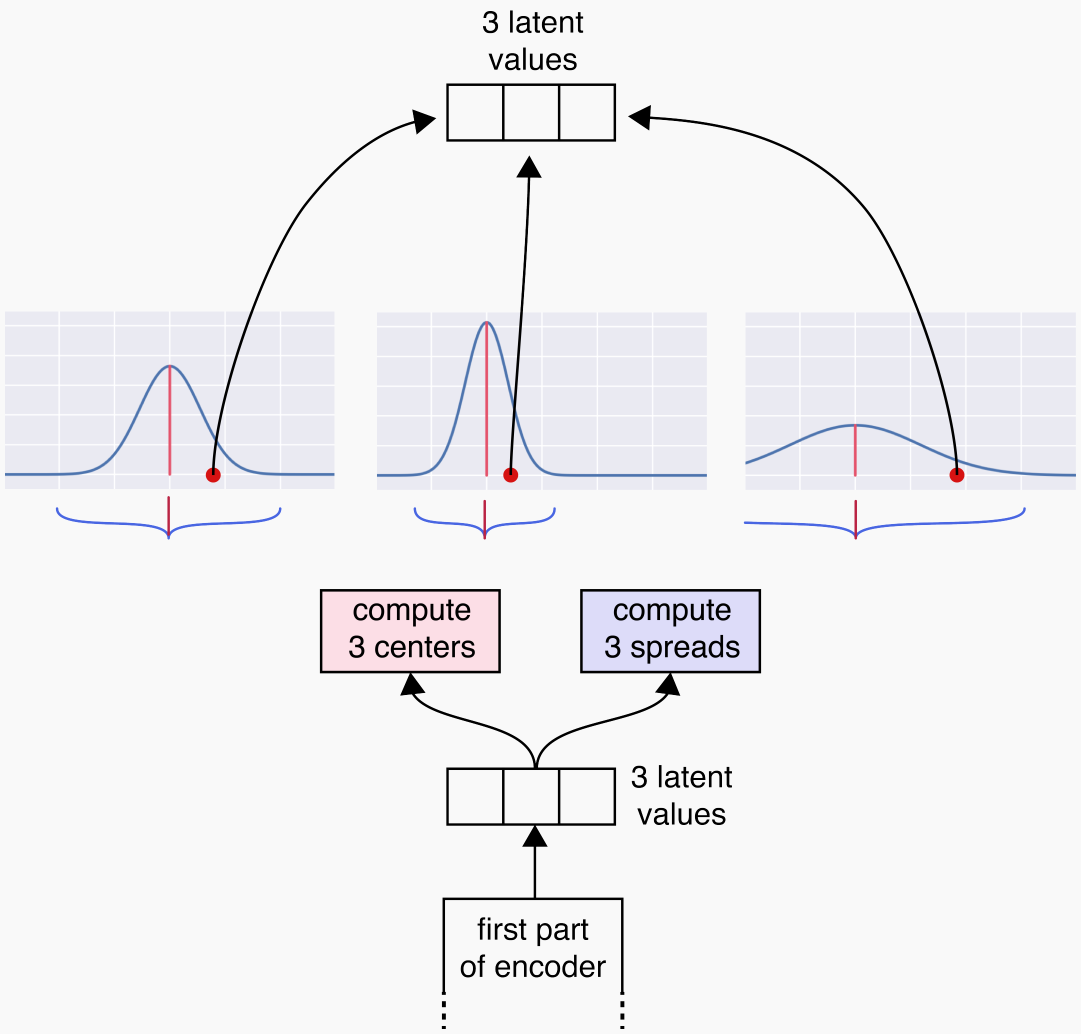
Variational Autoencoders



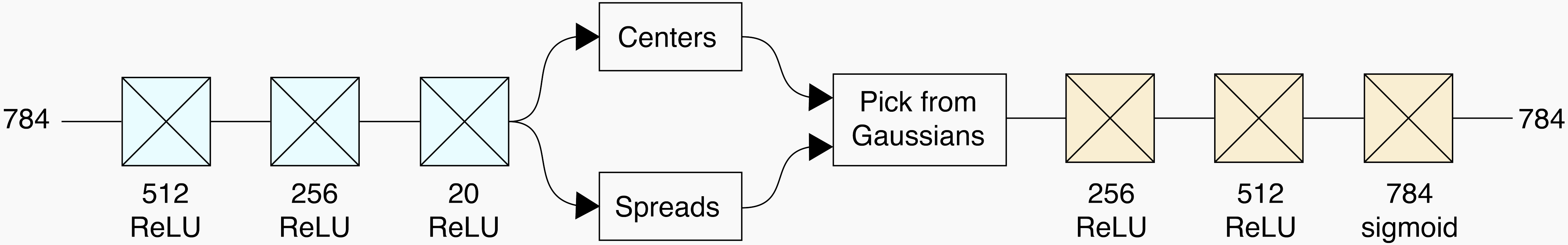
Variational Autoencoders



Variational Autoencoders



Variational Autoencoders



Outline

Motivation for Variational Autoencoders (VAE)

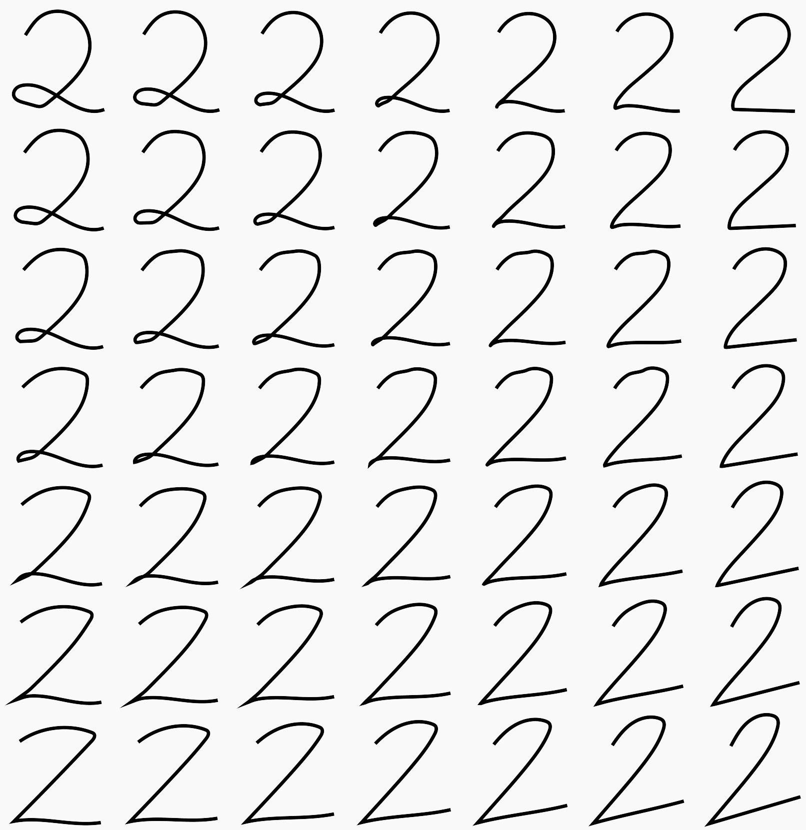
Mechanics of VAE

Seperability of VAE

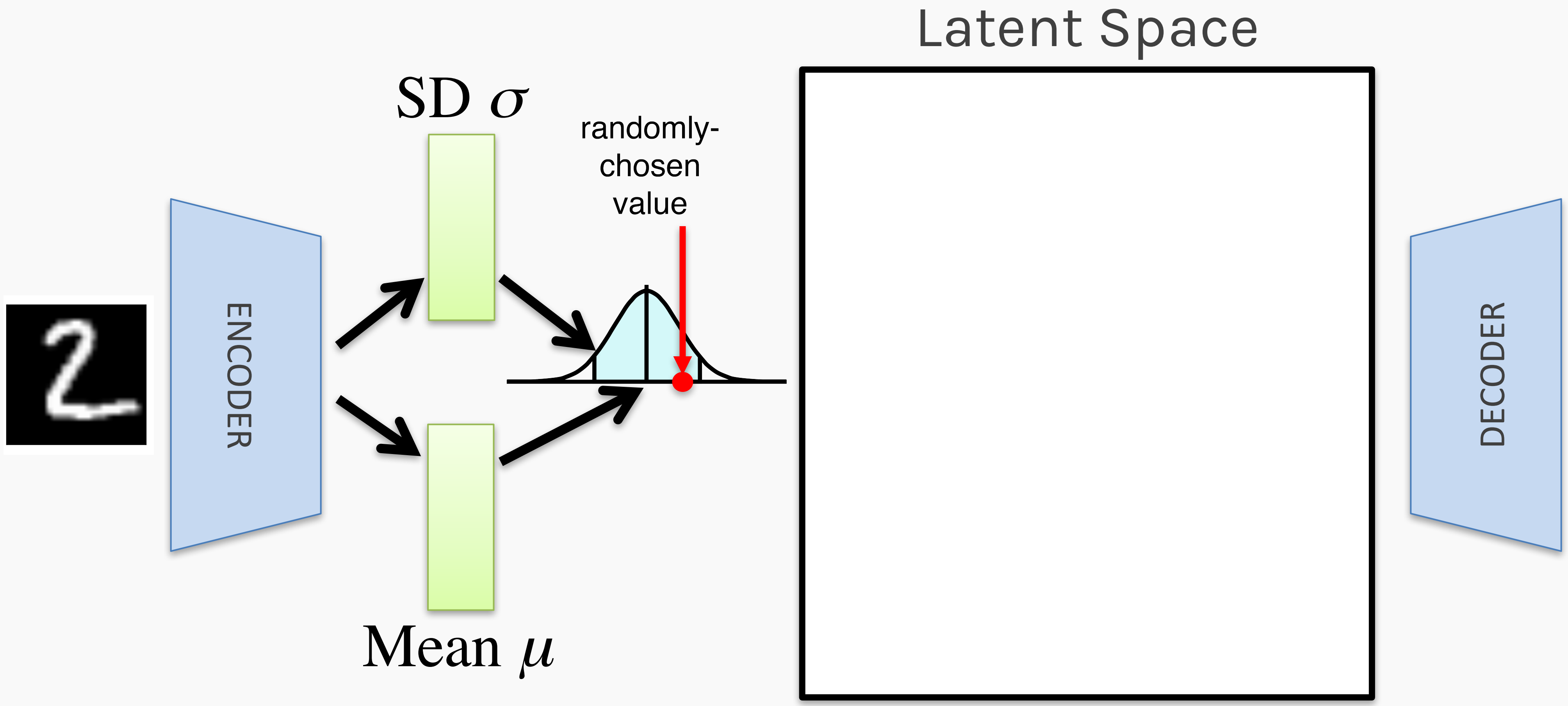
The math behind everything

Generative models

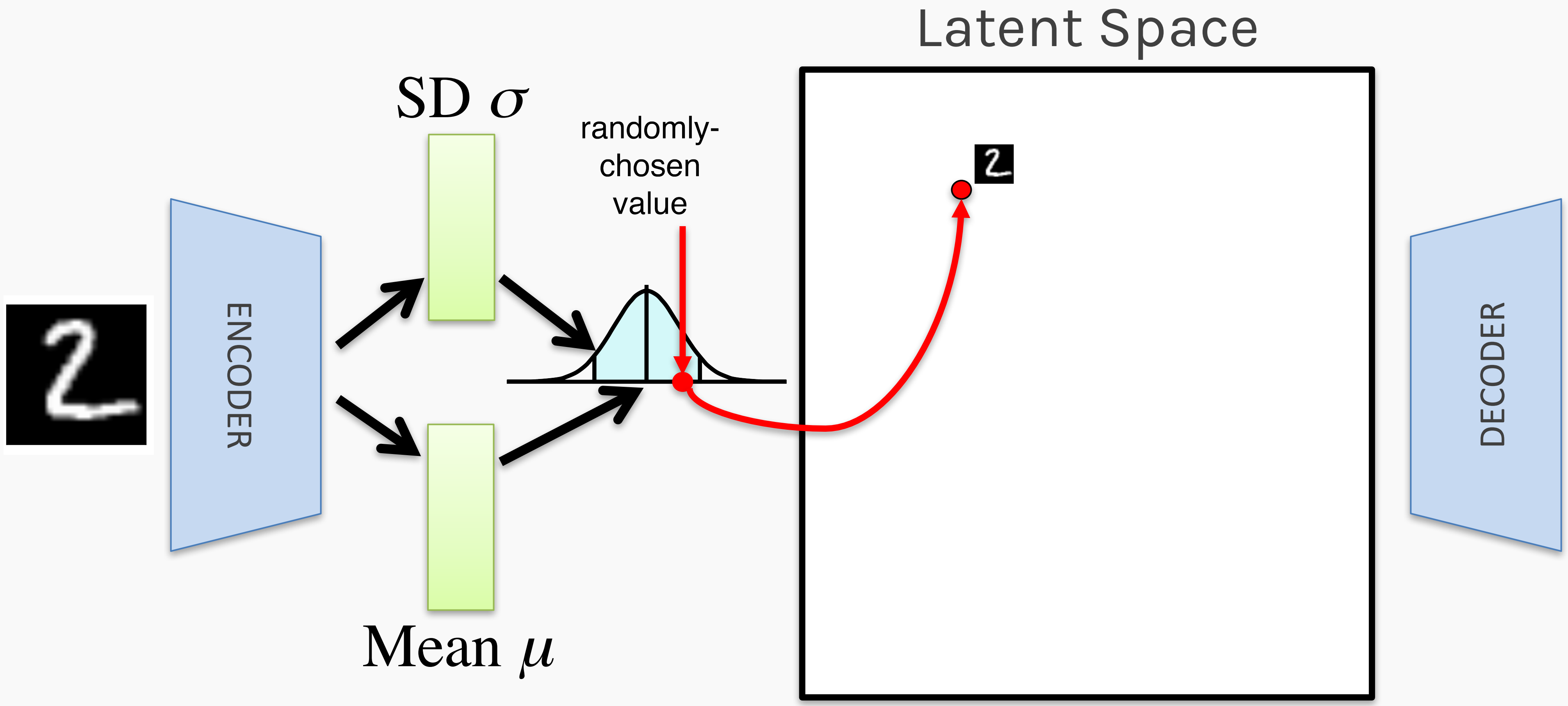
Variational Autoencoders



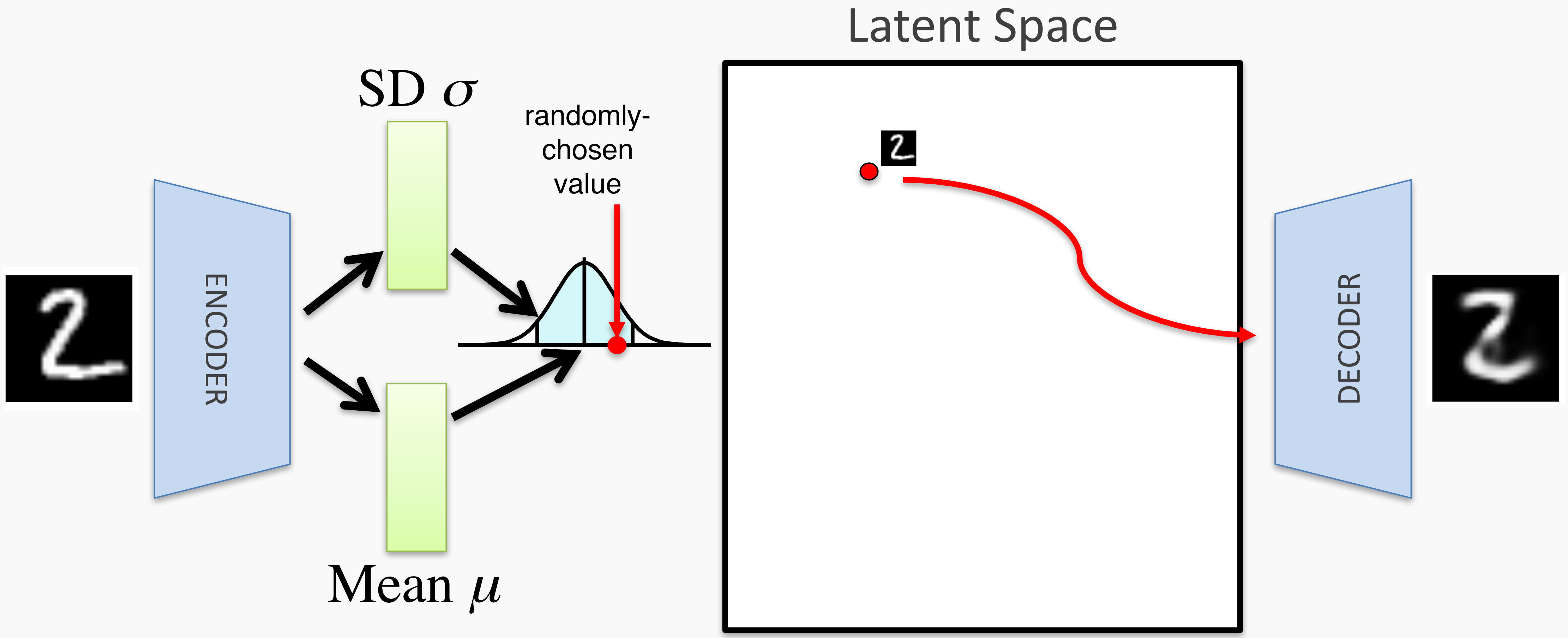
Blending Latent Variables



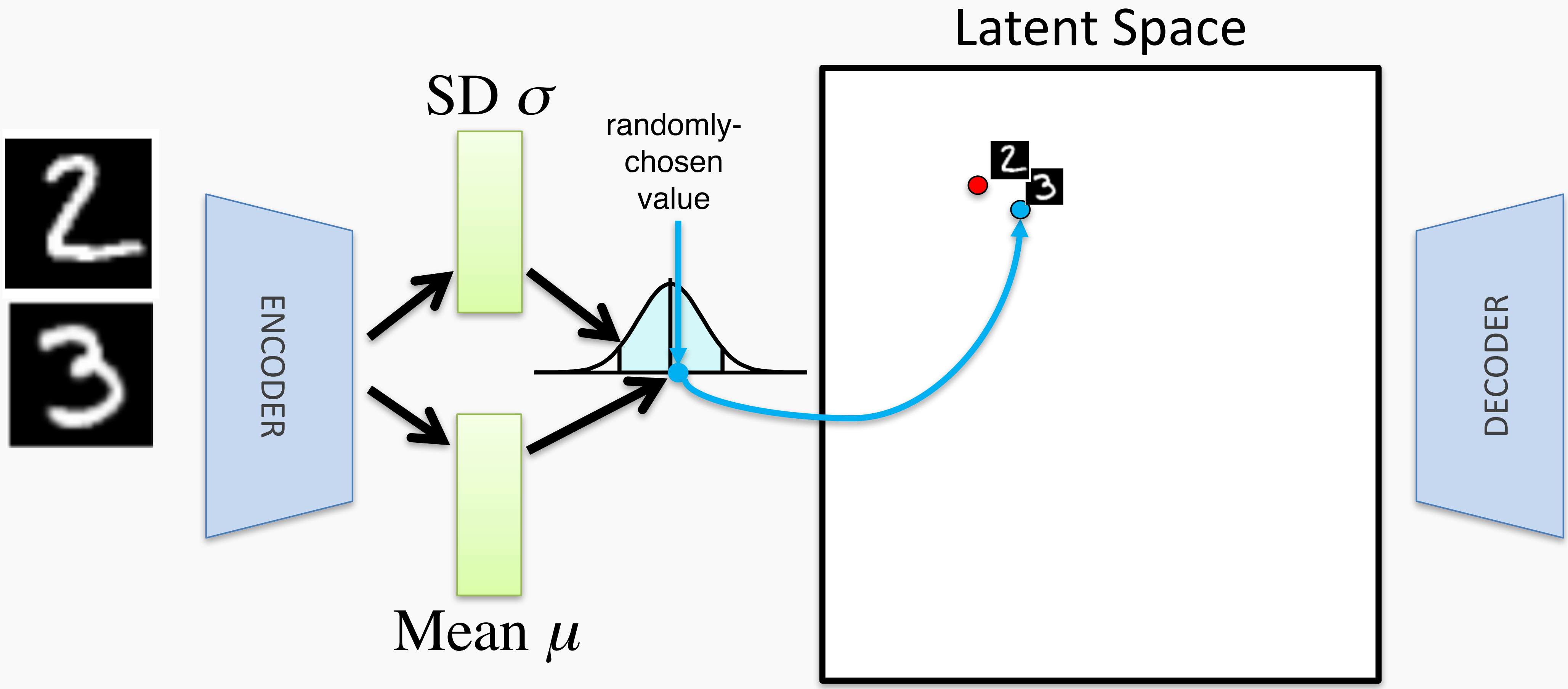
Blending Latent Variables



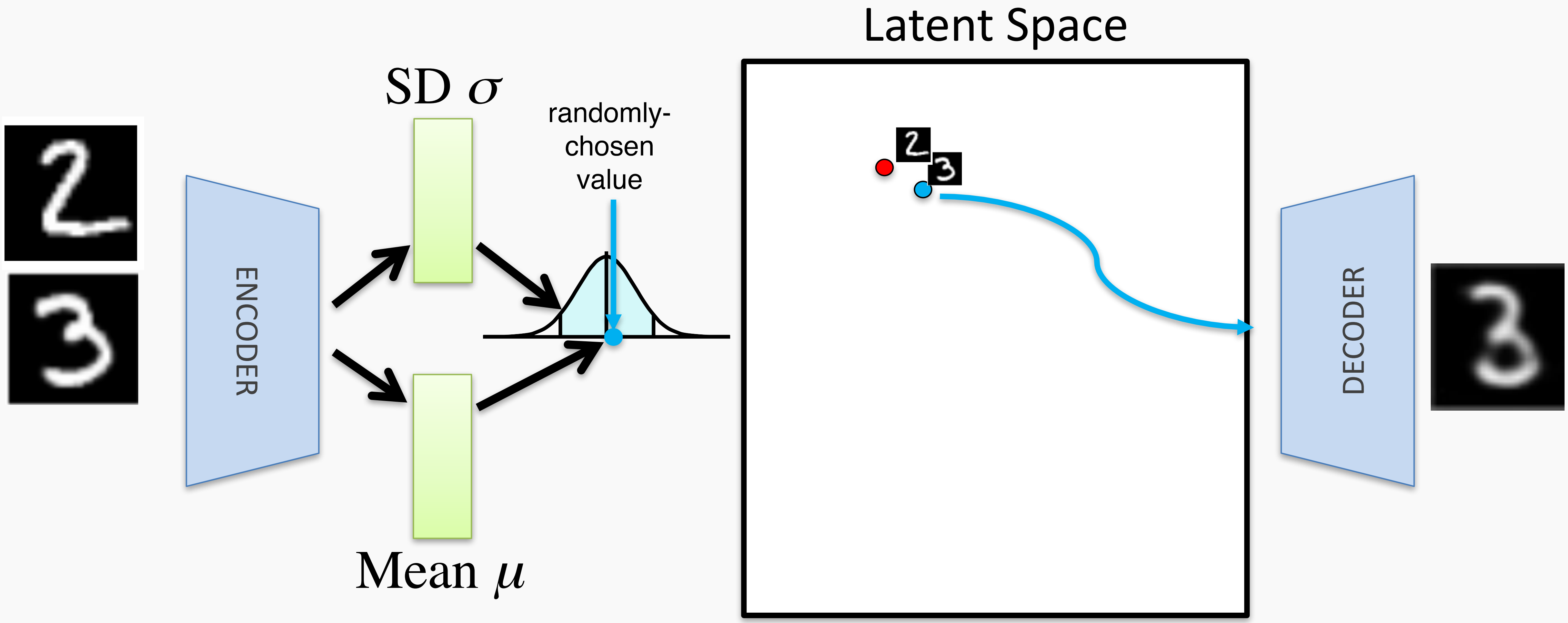
Blending Latent Variables



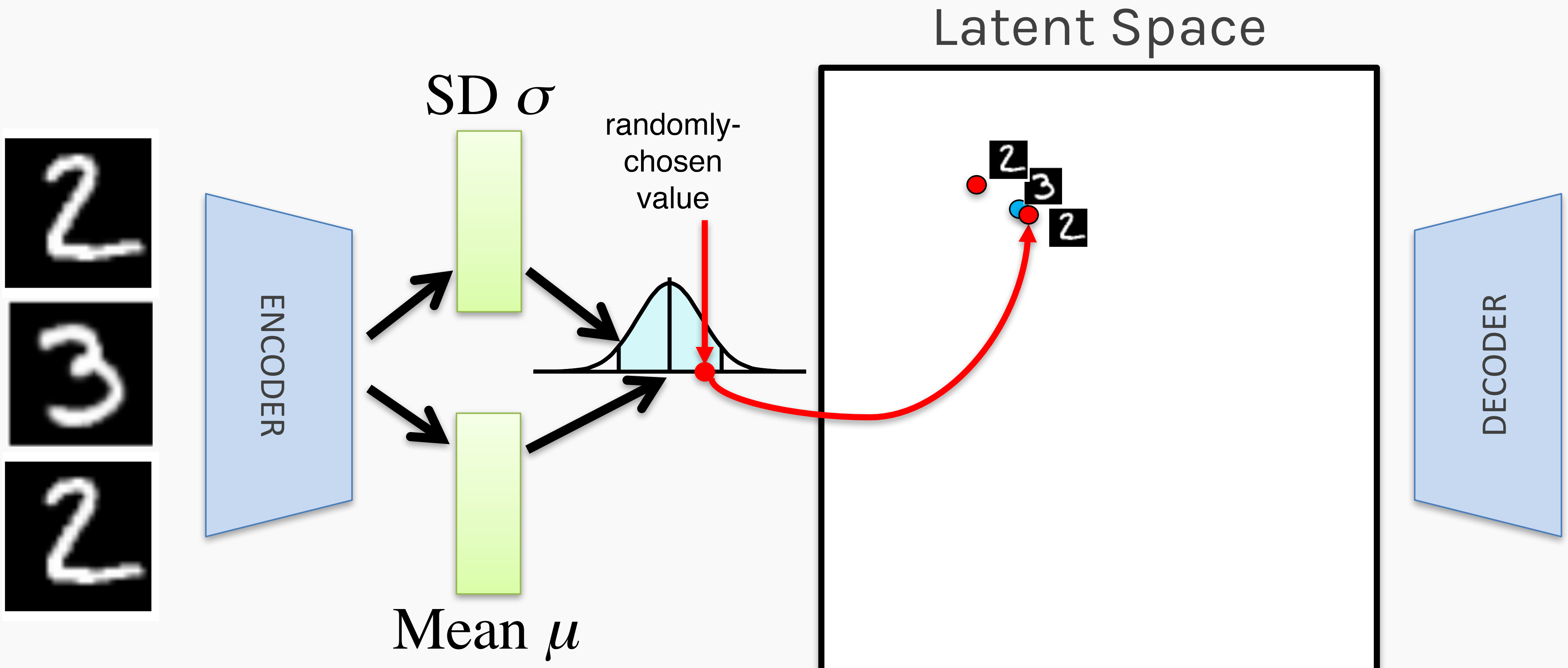
Blending Latent Variables



Blending Latent Variables

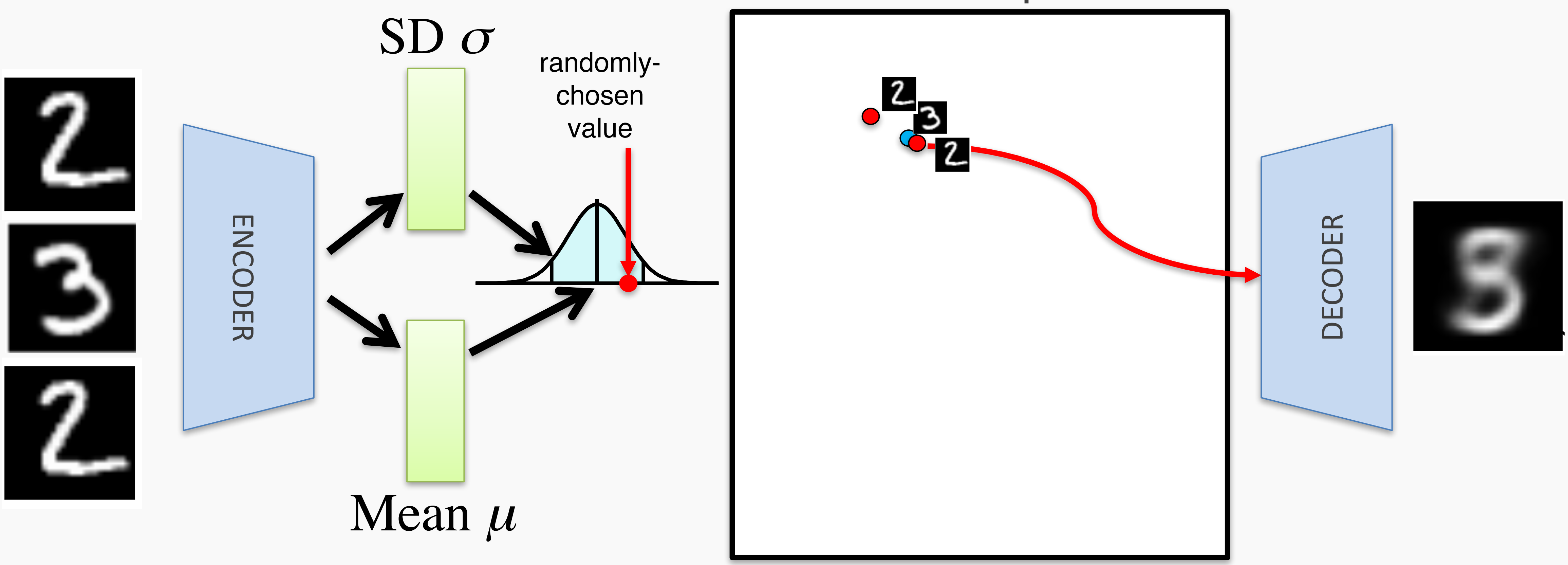


Blending Latent Variables

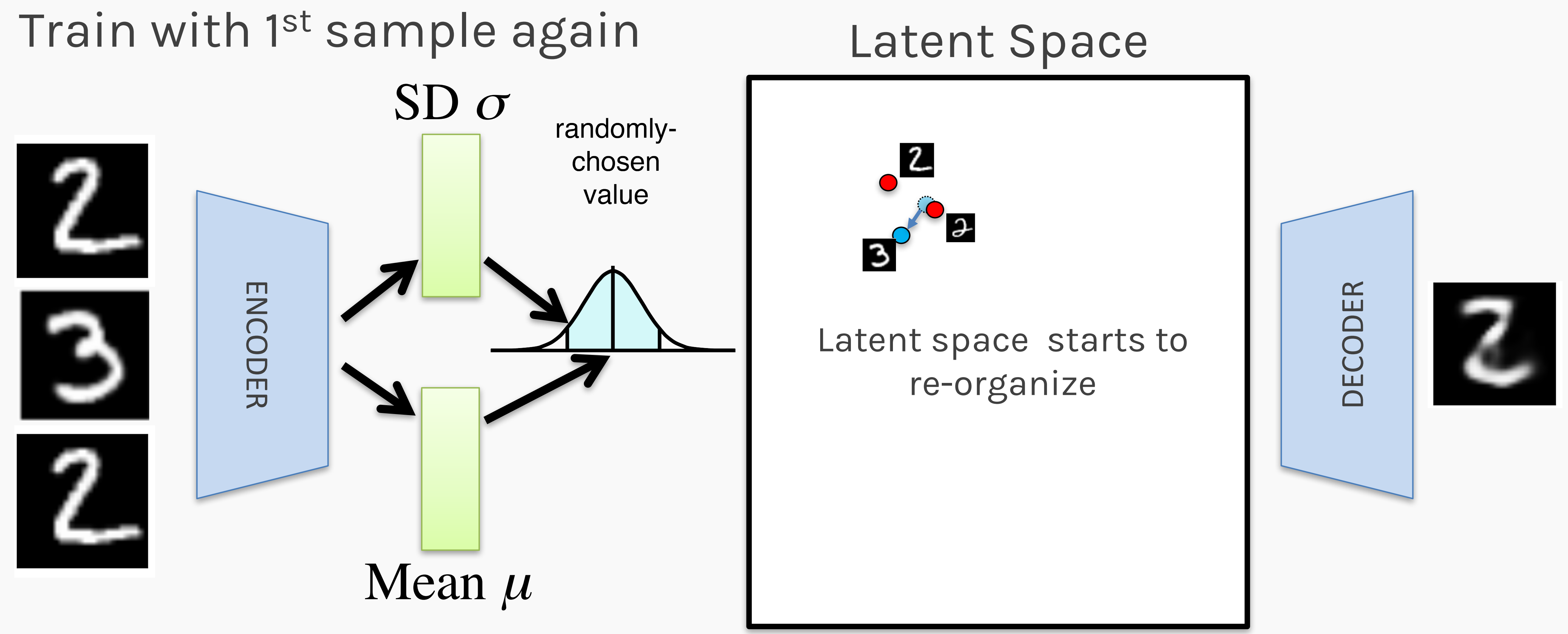


Blending Latent Variables

Train with 1st sample again

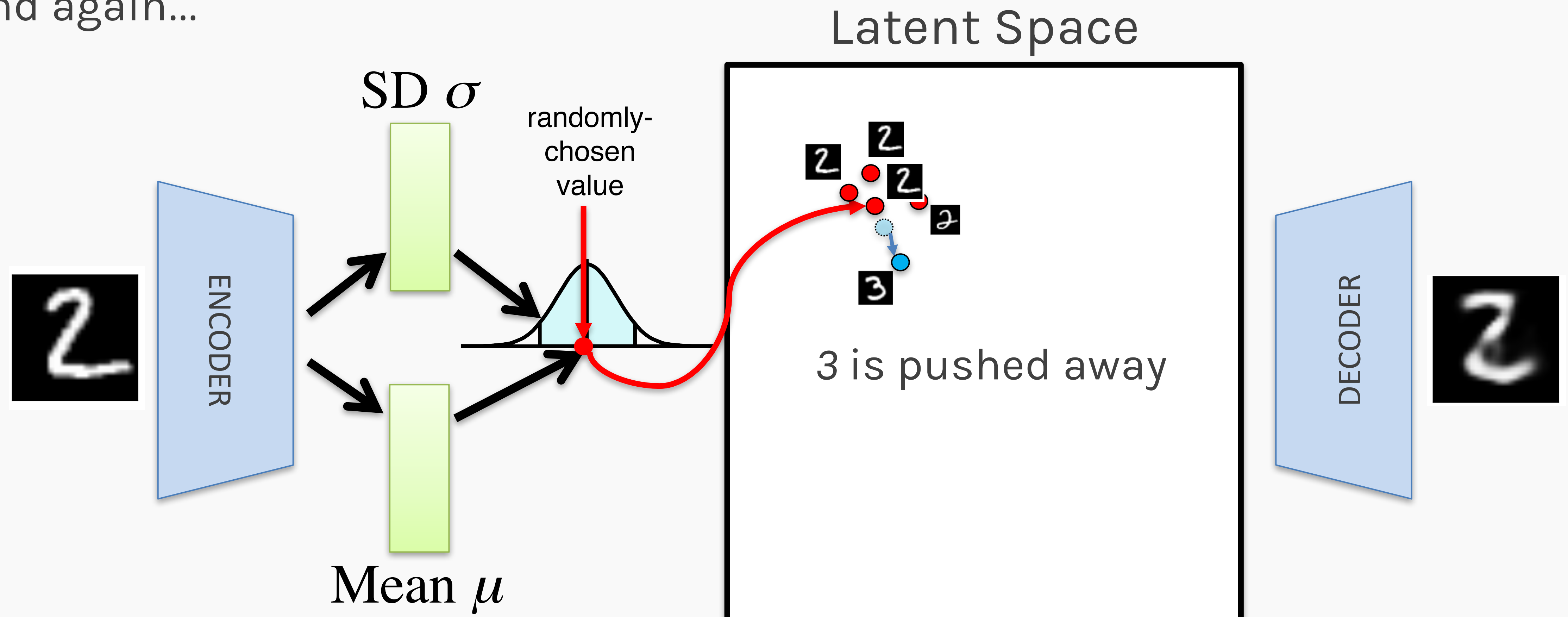


Blending Latent Variables



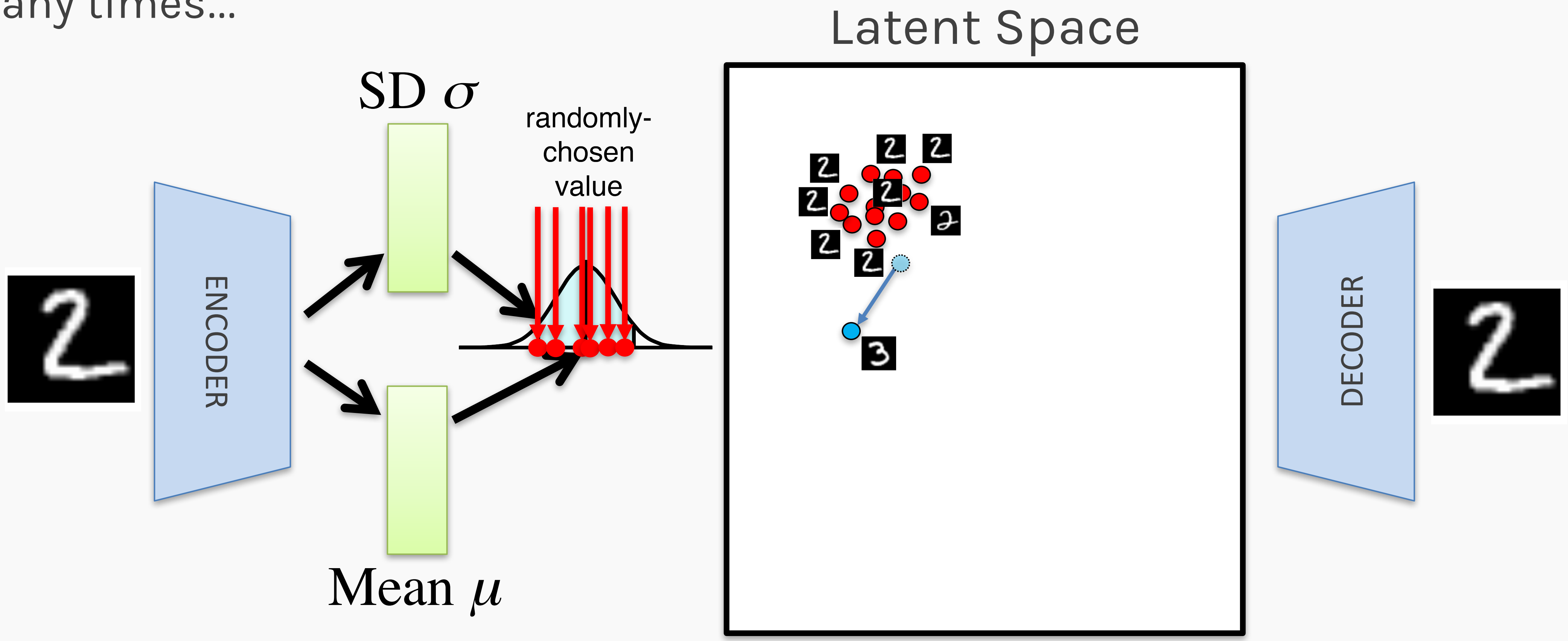
Blending Latent Variables

And again…



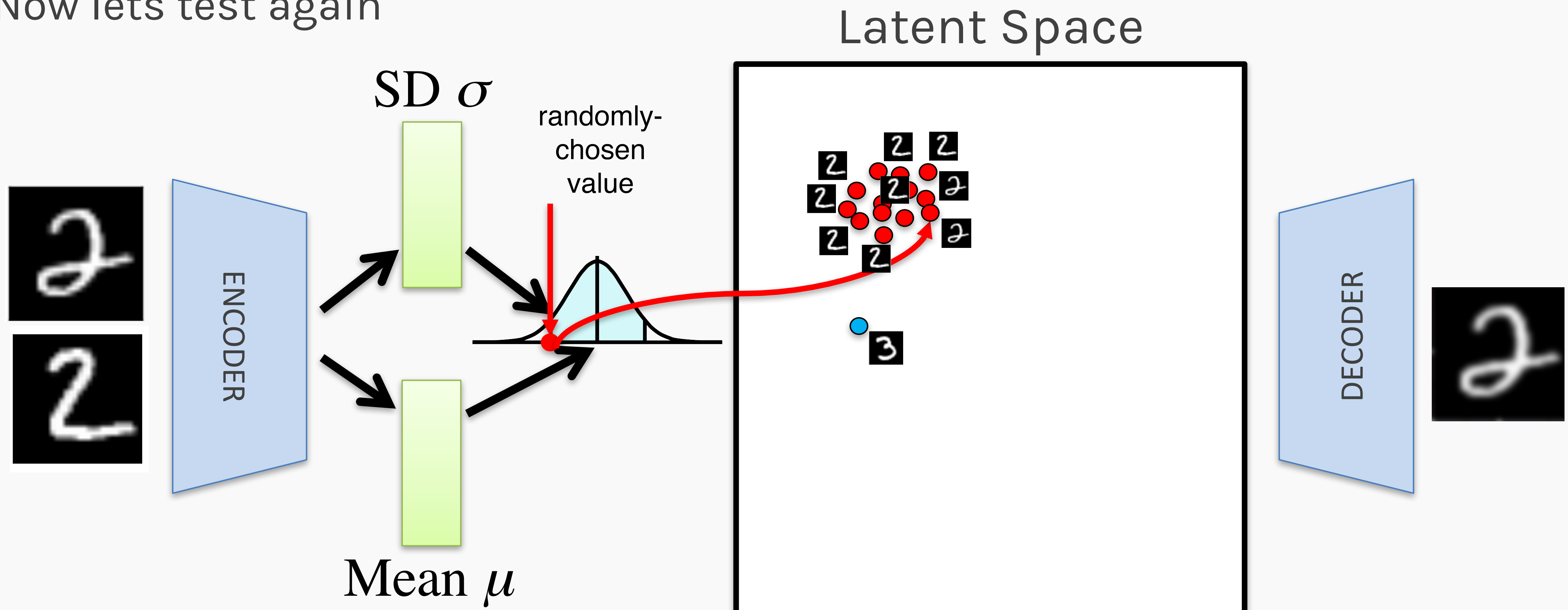
Blending Latent Variables

Many times…



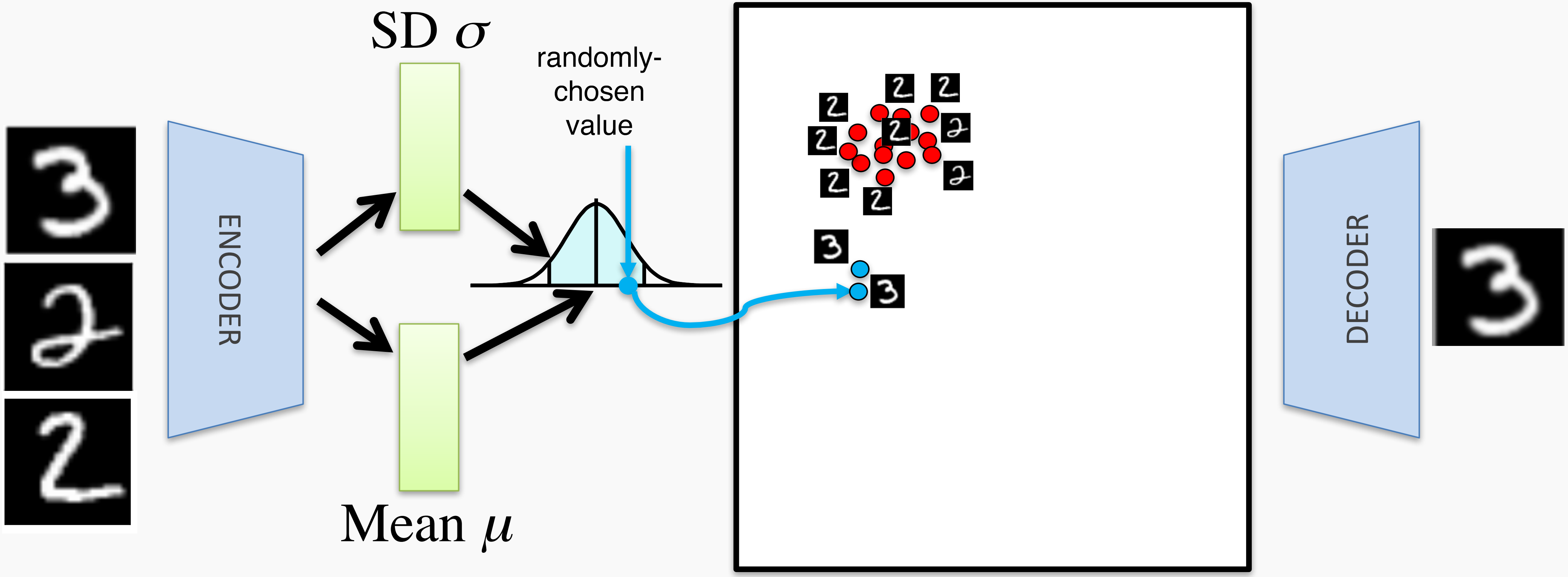
Blending Latent Variables

Now lets test again



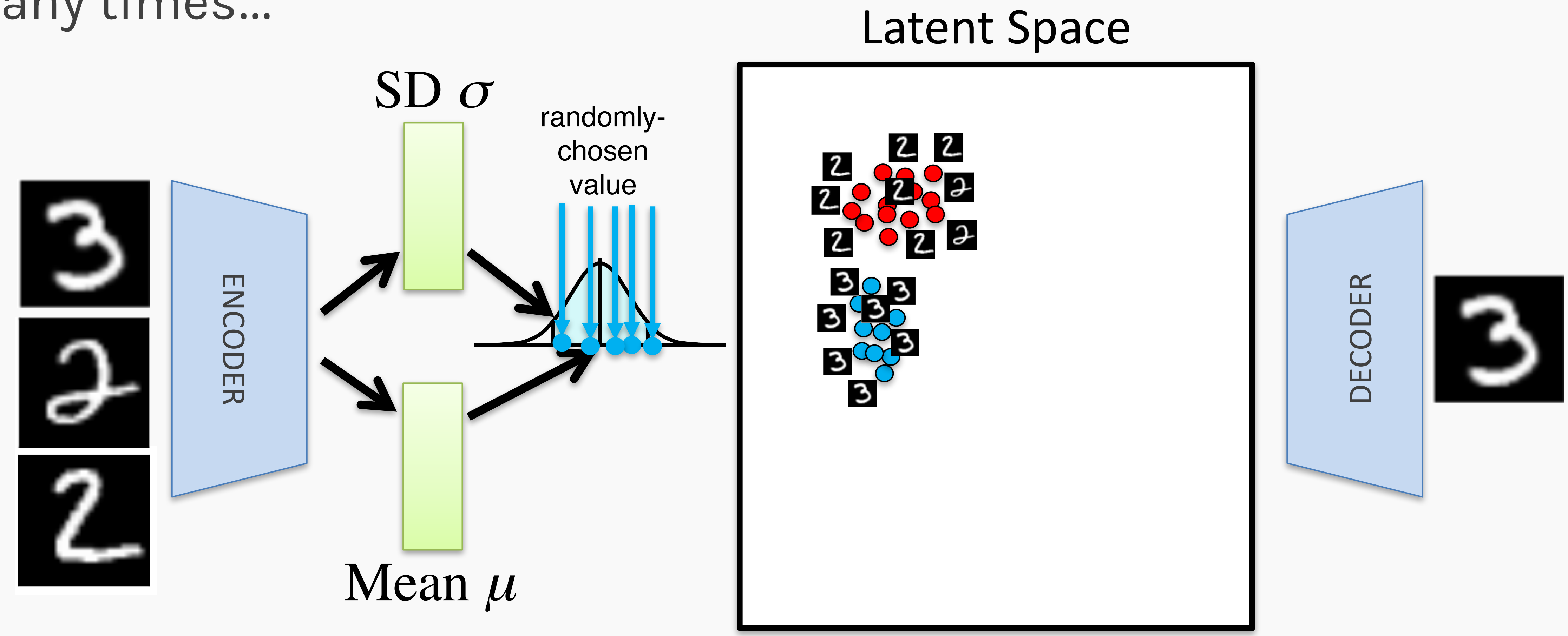
Blending Latent Variables

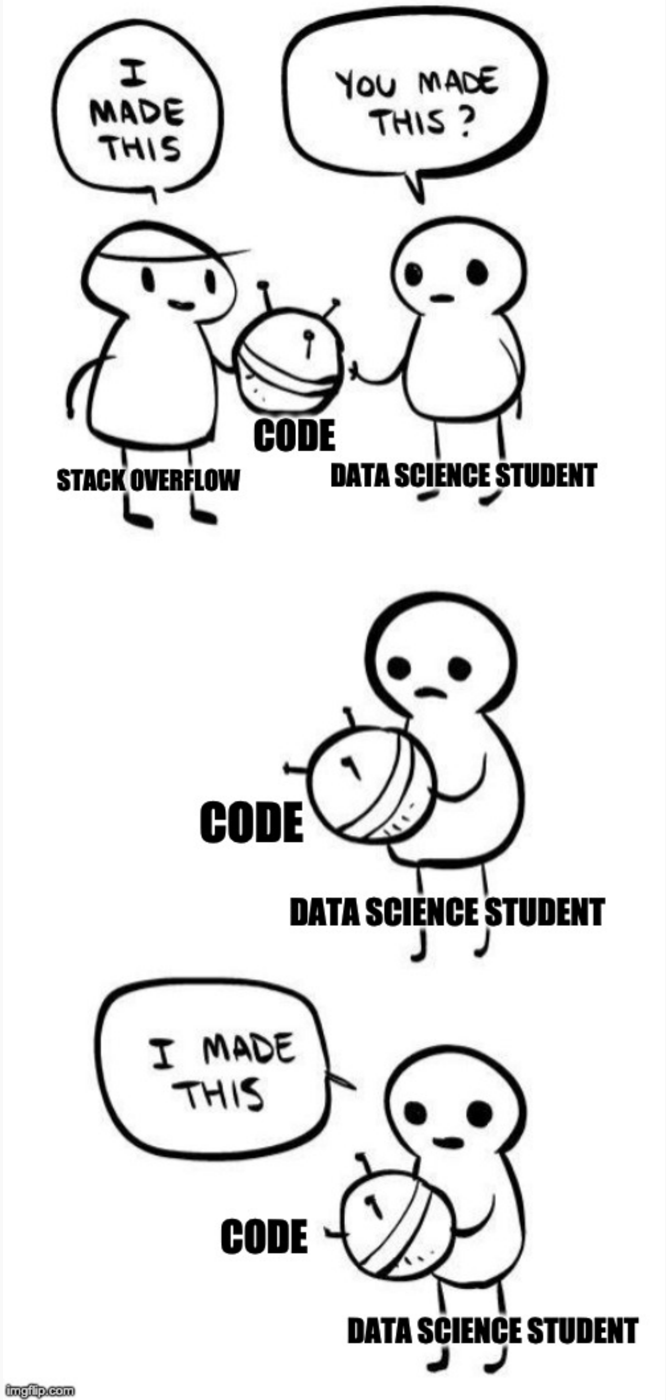
Training on 3's again



Blending Latent Variables

Many times…




I MADE THIS
YOU MADE THIS ?
CODE
STACK OVERFLOW
DATA SCIENCE STUDENT
CODE
DATA SCIENCE STUDENT
I MADE THIS
CODE
DATA SCIENCE STUDENT
imgflip.com

Outline

Motivation for Variational Autoencoders (VAE)

Mechanics of VAE

Seperability of VAE

The math behind everything

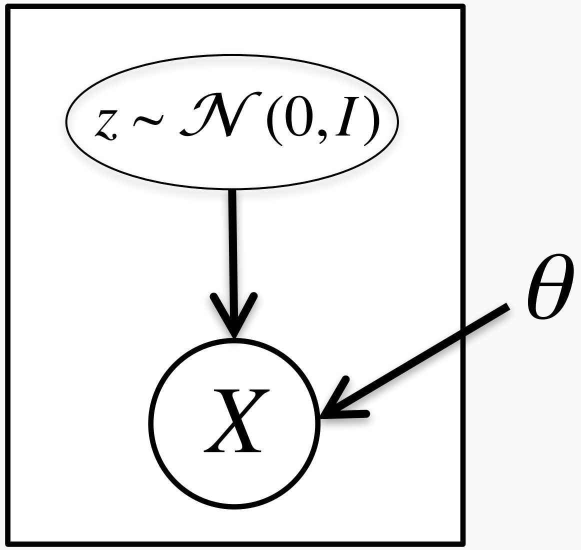
Generative models

VAE Likelihood

Neural network

$$p_\theta(x) = \int_z p_\theta\left(x|z\right)\boxed{p_\theta(z)}\,dz$$

Difficult to approximate in high dim through sampling



For most *z* values *p*(*x*|*z*) close to 0

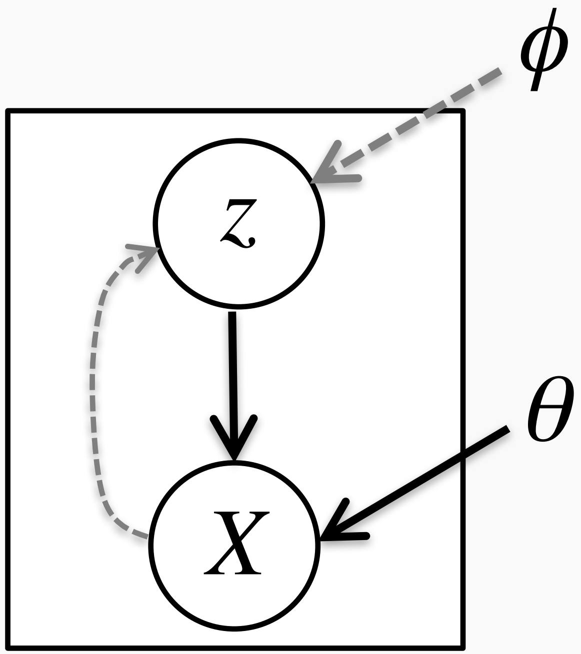
VAE Likelihood

Another neural net

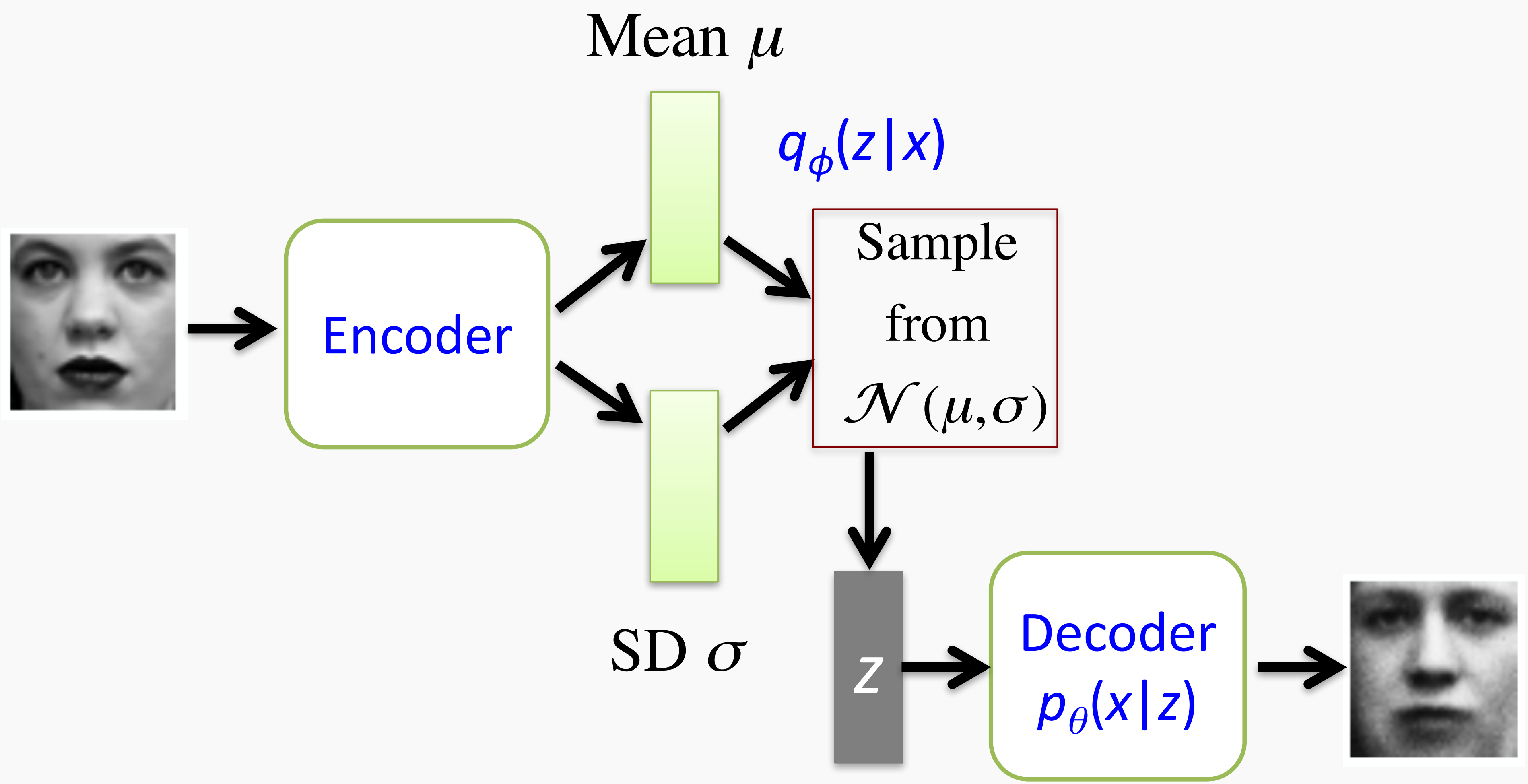
$$p_\theta(x) = \int_z p_\theta\left(x|z\right) \boxed{q_\phi\left(z|x\right)} dz$$

Proposal distribution:

Likely to produce values of *x* for which *p*(*x*|*z*) is non-zero



VAE Architecture



VAE Loss

Reconstruction Loss

$$-\mathbf{E}_{z\sim q_{\phi}(z|x)}\log\left(p_{\theta}\left(x|z\right)\right)$$

VAE Loss

Reconstruction Loss

Proposal distribution should resemble a Gaussian

$$-\mathbf{E}_{z \sim \boxed{q_\phi(z|x)}} \log\left(p_\theta(x|z)\right) + KL\left(\boxed{q_\phi(z|x)} \| p_\theta(z)\right)$$

VAE Loss

Reconstruction Loss

Proposal distribution should resemble a Gaussian

$$-\mathbf{E}_{z\sim \boxed{q_\phi(z|x)}} \log\left(p_\theta(x|z)\right) + KL\left(\boxed{q_\phi(z|x)} \| p_\theta(z)\right)$$

$$\geq -\log p_\theta(x)$$

Variational upper bound
on loss we care about!

Training VAE

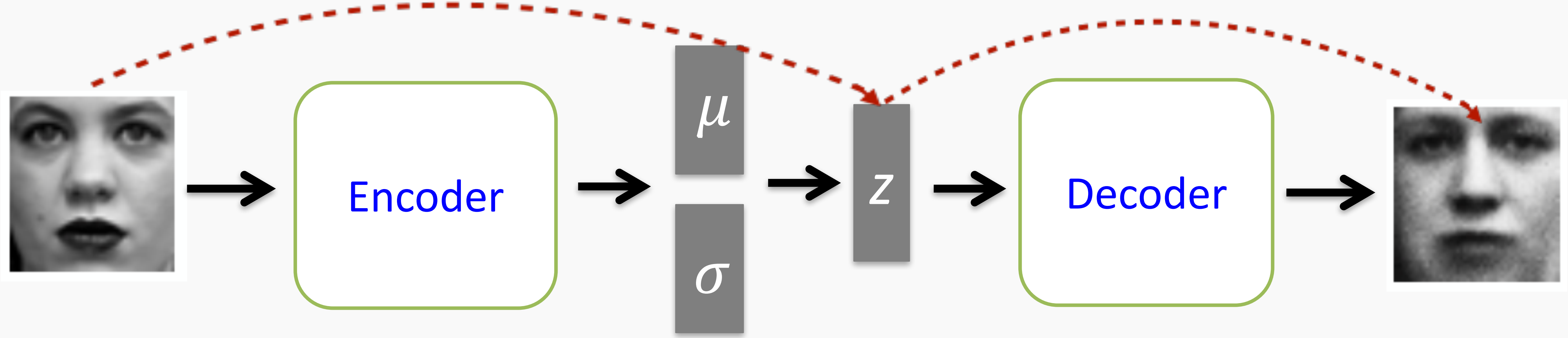
Apply stochastic gradient descent

Sampling step not differentiable

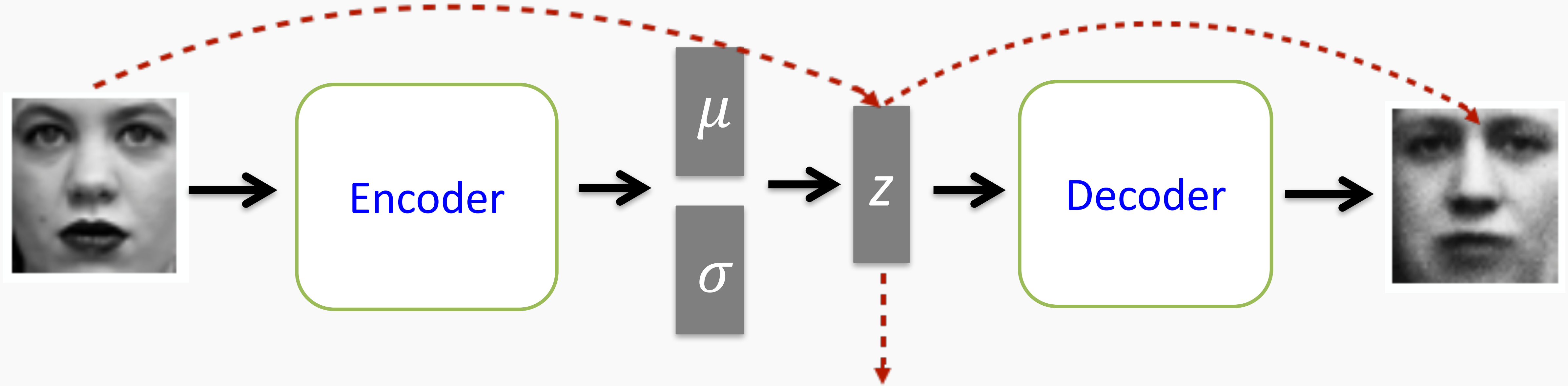
Use a re-parameterization trick

- Move sampling to input layer, so that the sampling step is independent of the model

Reparametrization Trick

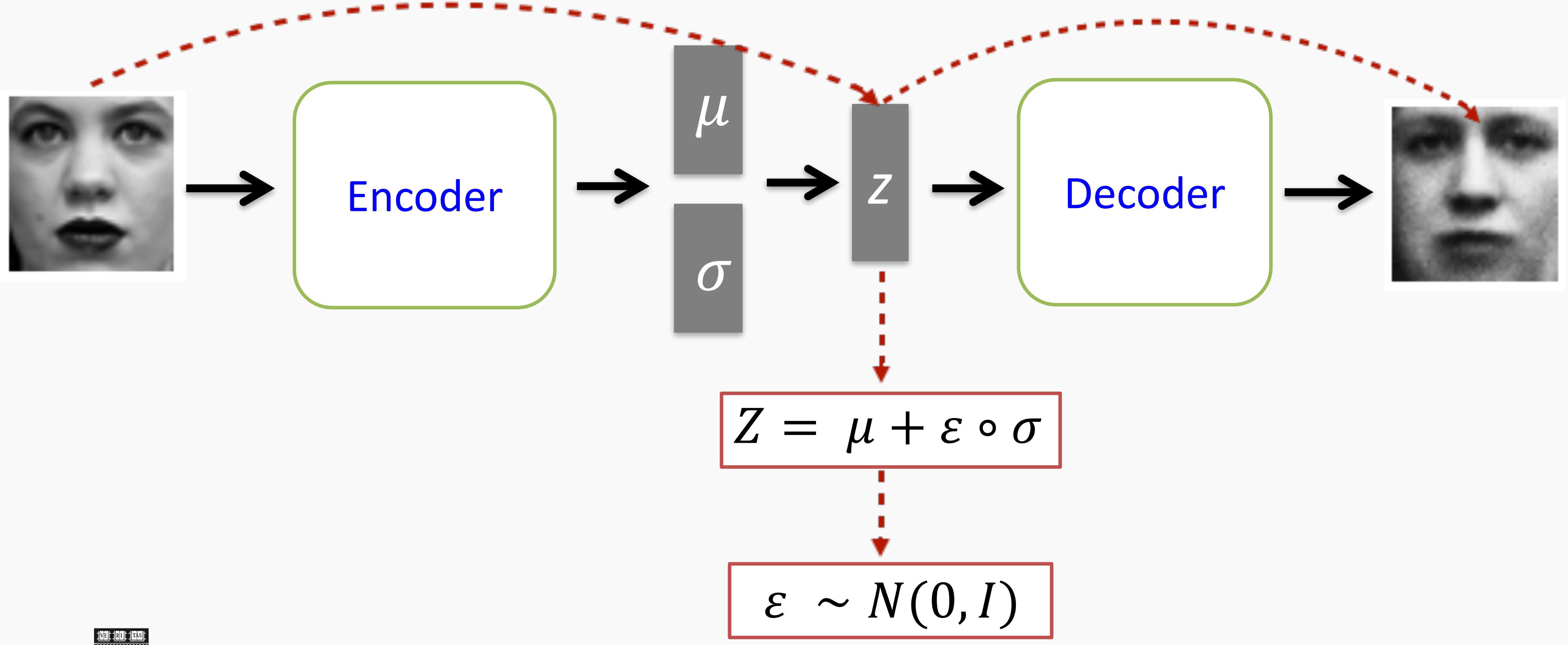


Reparametrization Trick

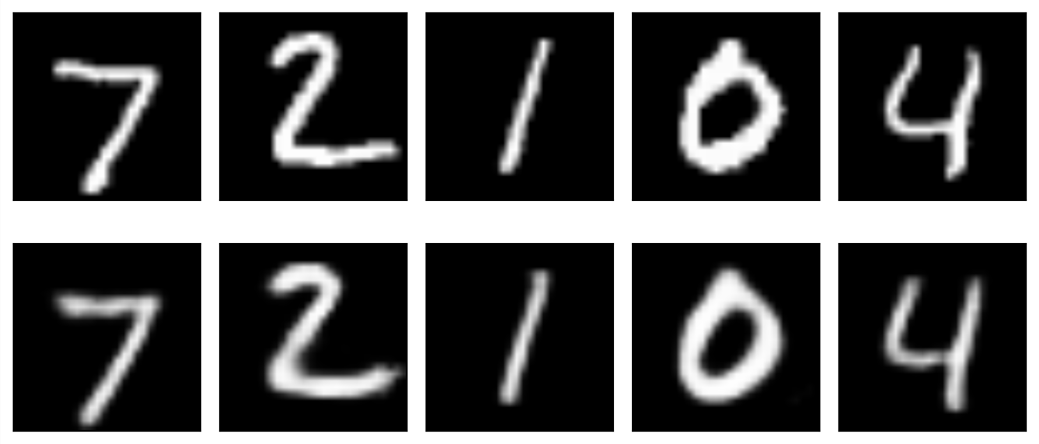


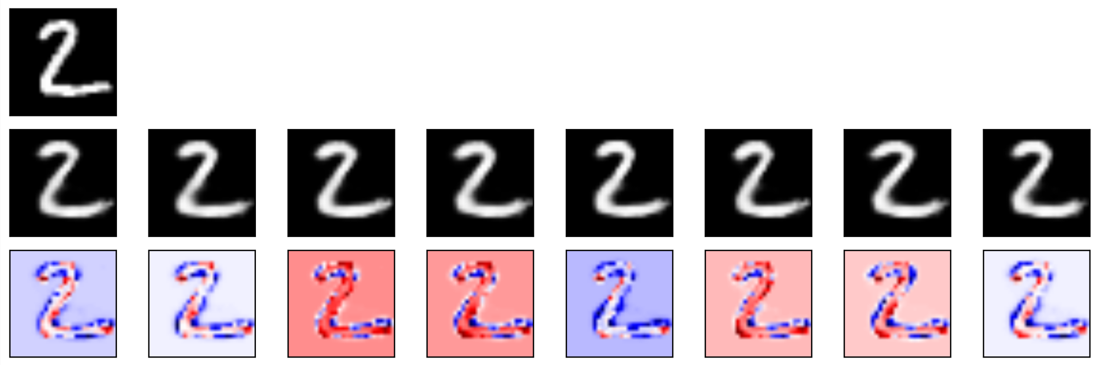
$$Z = \mu + \varepsilon \circ \sigma$$

Reparametrization Trick

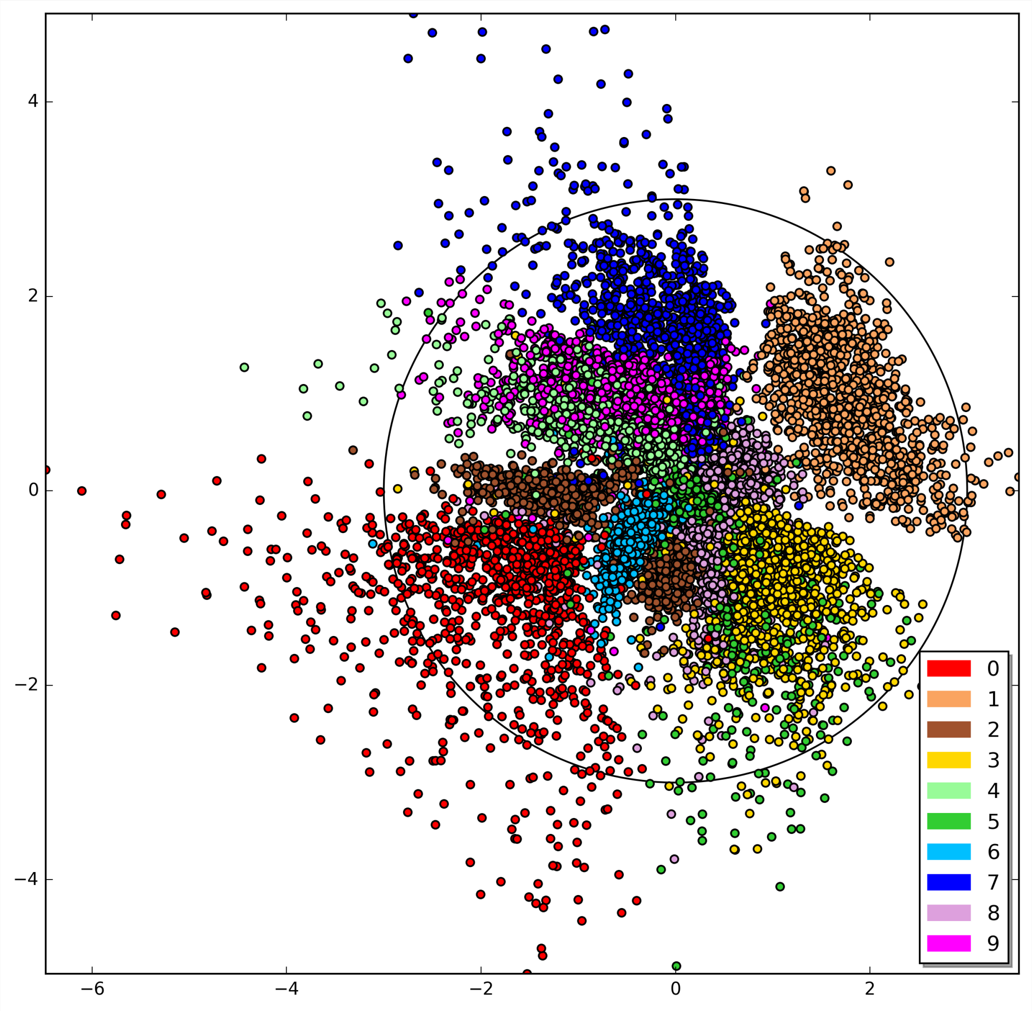


Training VAE

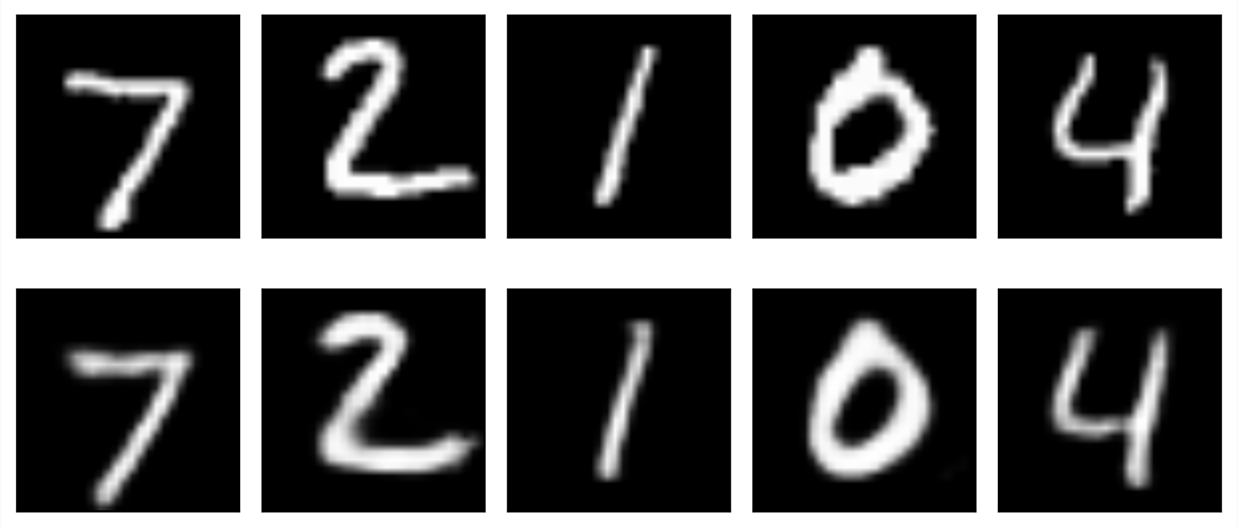




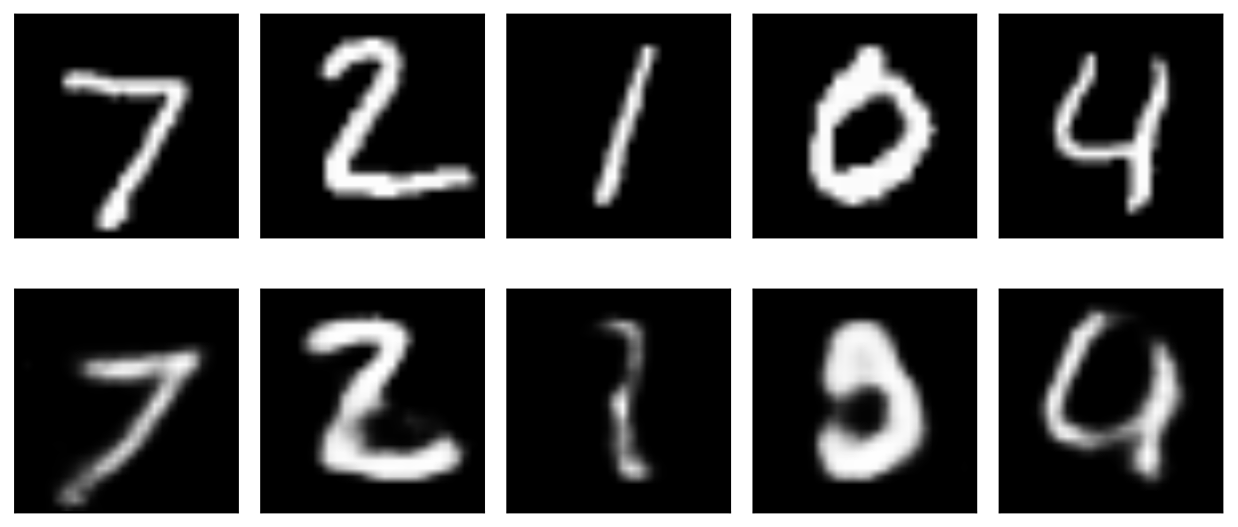
Parameter space VAE



Training VAE



10% error



30% error

Parameter space VAE