

# Variational Auto-Encoders (Part 1)

# What we'll learn about Variational Auto-Encoders:

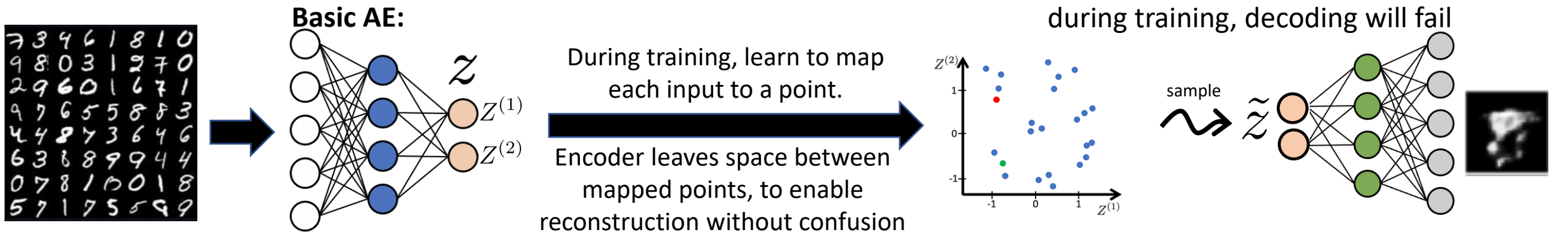
Part 1 (this video): The “simple” explanation of a VAE, as a regularized AE

- Architecture of a Variational Auto-Encoder (VAE)
- How to train VAEs
- How do different terms of training loss influence what VAE learns
- How does a VAE relate to the basic AE

Part 2: Applications of VAE for...

- Generation of new data points
- Modifying data via interpolating between 2 inputs
- Modifying specific feature of an input
- Compression and Reconstruction

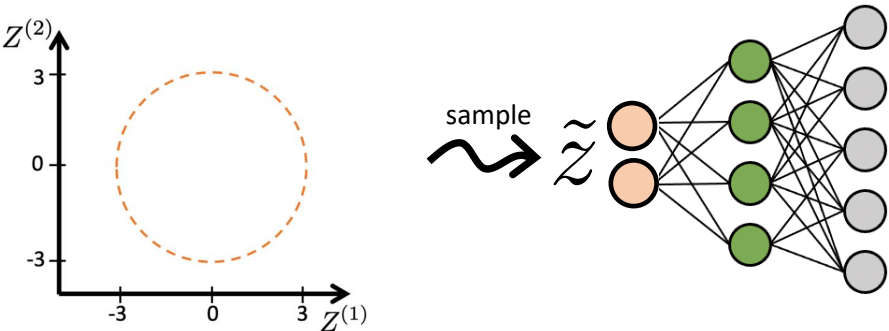
# From basic deterministic AE to Variational AE



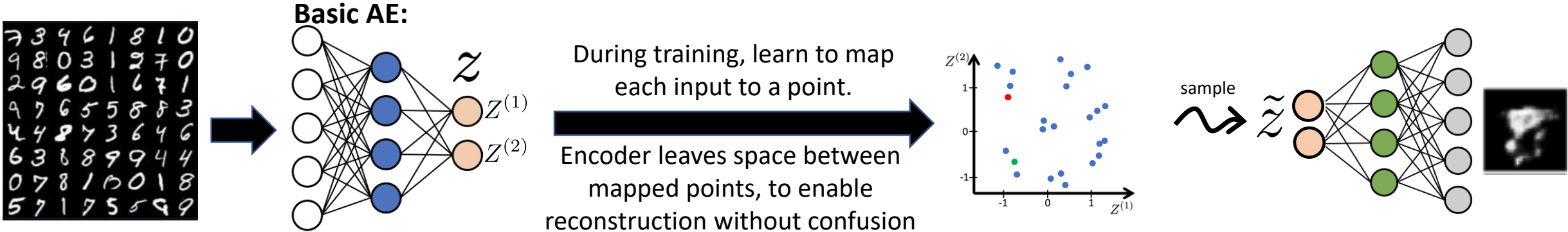
# From basic deterministic AE to Variational AE

**VAE** (informally):

**Training:** Enforce codes  $z$  to cover a specified area

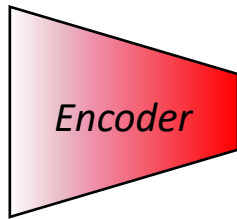
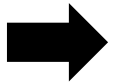


**Generation:** We can then sample from anywhere in the area. Decoding should then work.

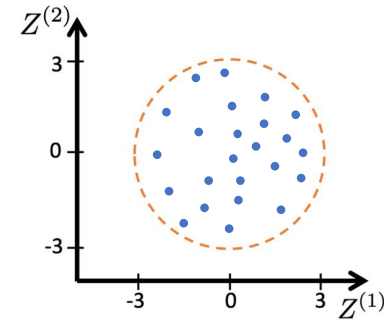


# From basic deterministic AE to Variational AE

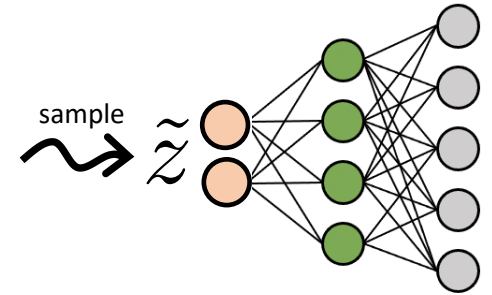
**VAE (informally):**



Instead of using Encoder that maps each input to *a single point* like in basic AE...



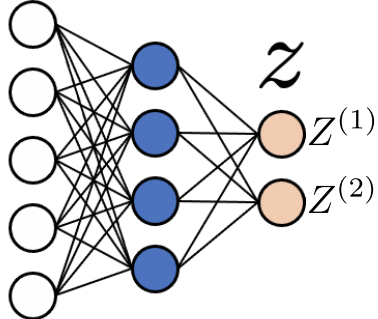
**Training:** Enforce codes  $z$  to cover a specified area



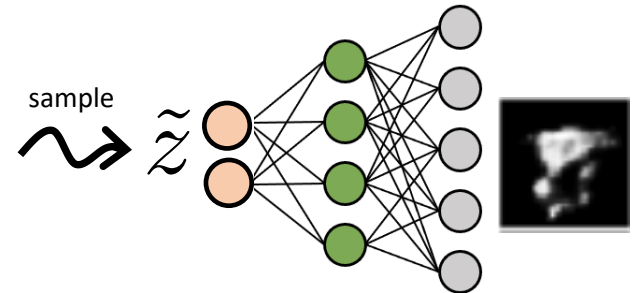
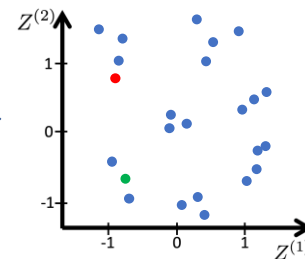
**Generation:** We can then sample from anywhere in the area. Decoding should then work.



**Basic AE:**

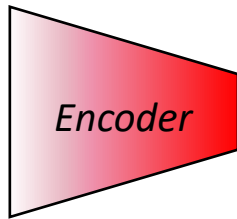
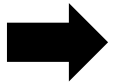


During training, learn to map each input to a point.  
Encoder leaves space between mapped points, to enable reconstruction without confusion



# From basic deterministic AE to Variational AE

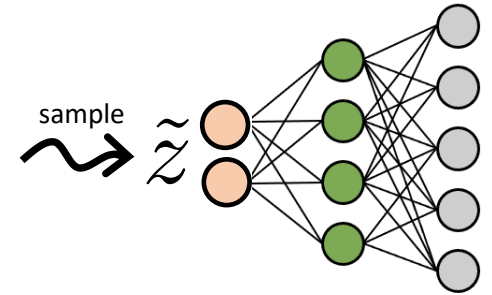
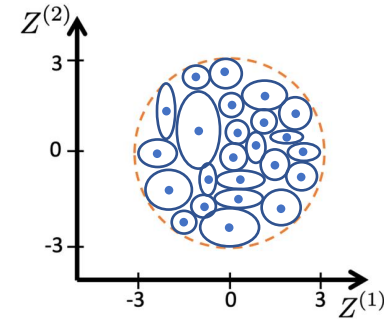
VAE (informally):



Make encoder that maps  
each input to an 'area'

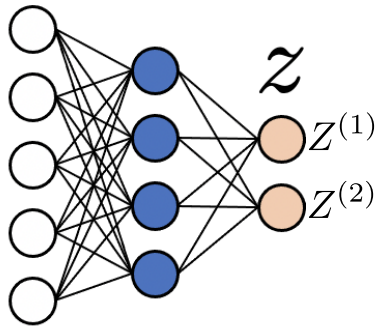


Forces decoder to learn to decode  
 $z$  from whole covered area.

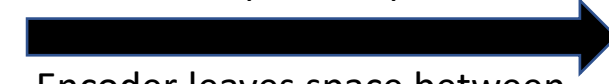


**Training:** Enforce codes  $z$  to  
cover a specified area

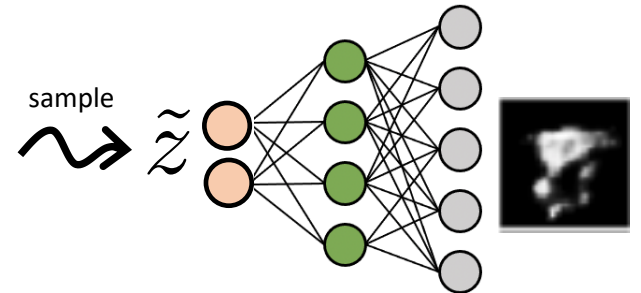
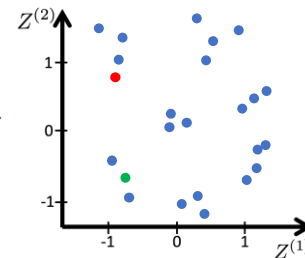
**Generation:** We can then sample from anywhere  
in the area. Decoding should then work.



During training, learn to map  
each input to a point.



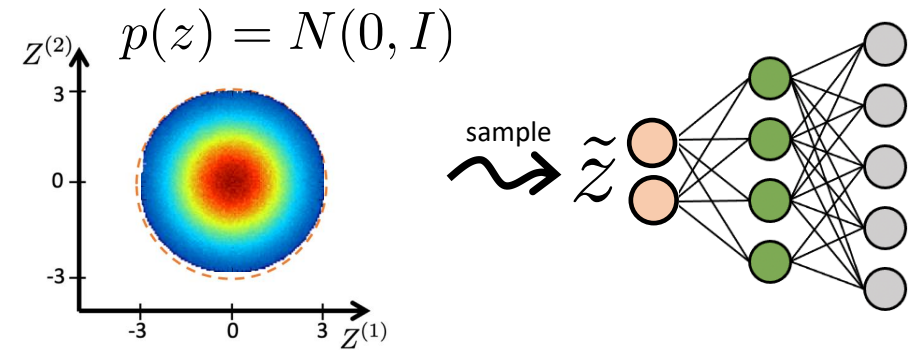
Encoder leaves space between  
mapped points, to enable  
reconstruction without confusion



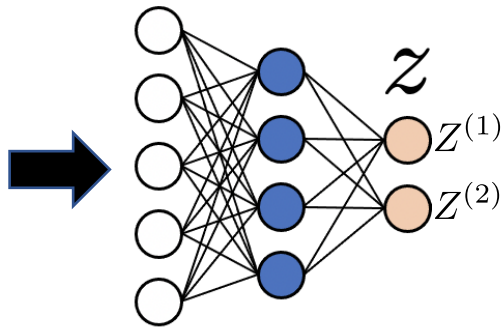
# From basic deterministic AE to Variational AE

**VAE** (more accurately):

**Training:** Enforce codes  $z$  to follow a “prior” distribution  $p(z)$

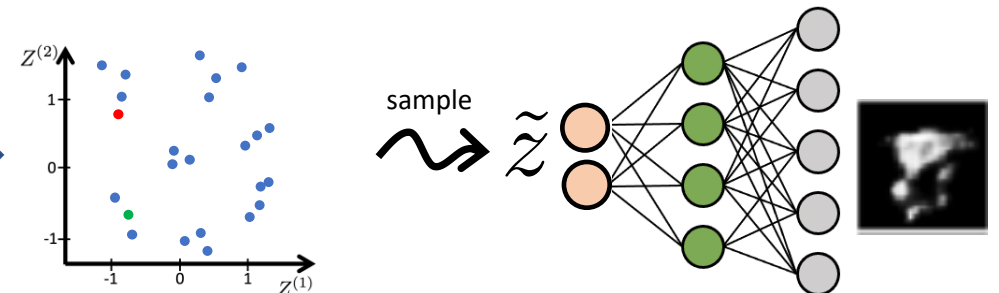


**Generation:** We can then sample from prior  $p(z)$  and decode!



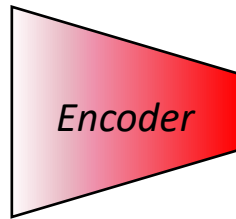
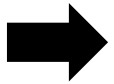
During training, learn to map each input to a point.

Encoder leaves space between mapped points, to enable reconstruction without confusion



# From basic deterministic AE to Variational AE

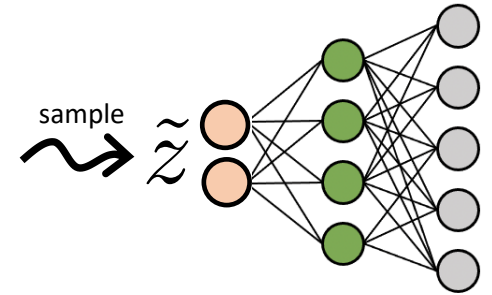
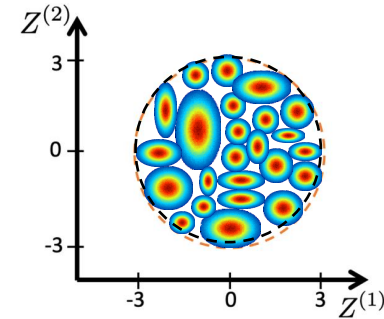
**VAE** (more accurately):



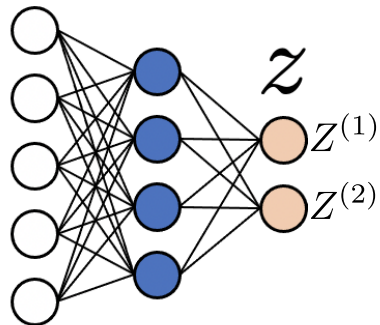
Learn encoder that predicts codes  $z$  that follow distribution  $p(z)$



Forces decoder to learn to decode  $z$  from whole  $p(z)$ .



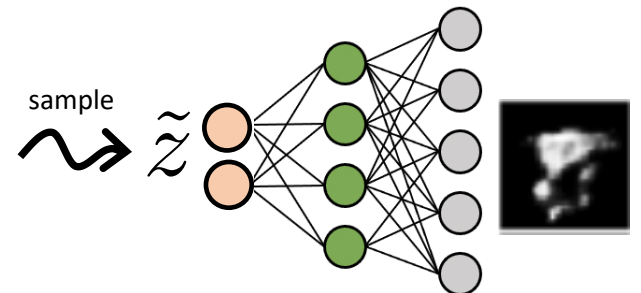
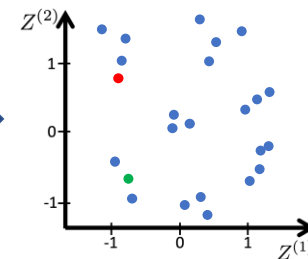
**Generation:** We can then sample from prior  $p(z)$  and decode!



During training, learn to map each input to a point.

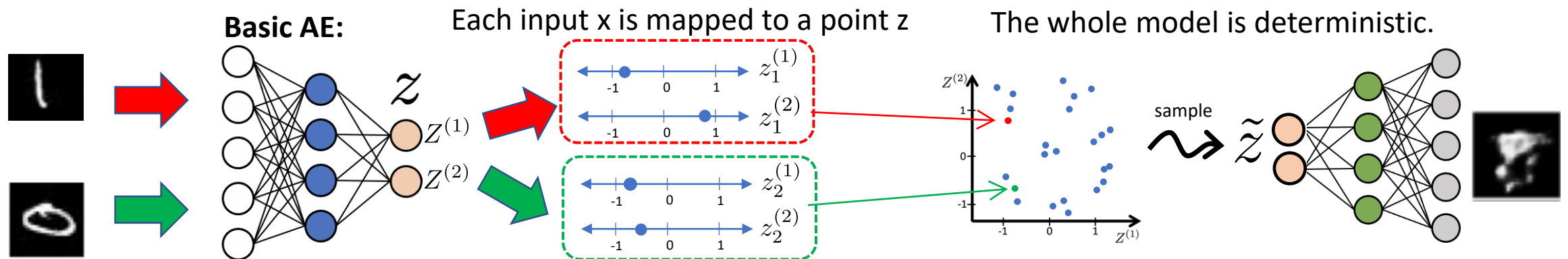


Encoder leaves space between mapped points, to enable reconstruction without confusion





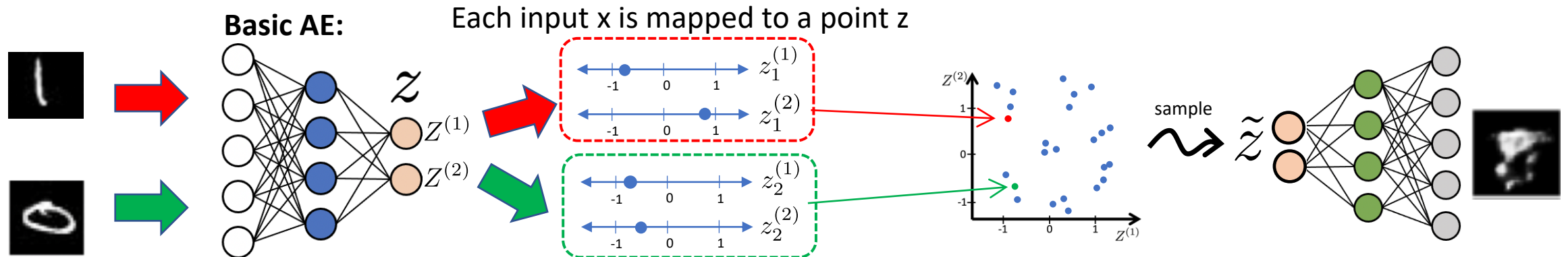
# 1. Probabilistic Encoder



# 1. Probabilistic Encoder

“posterior” distribution of  $z$ ,  
a.k.a. “conditional” distribution of  $z$  given  $x$

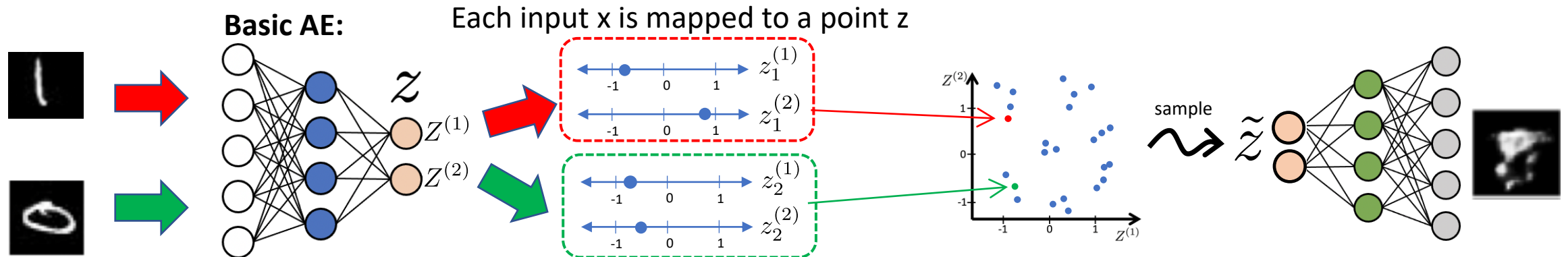
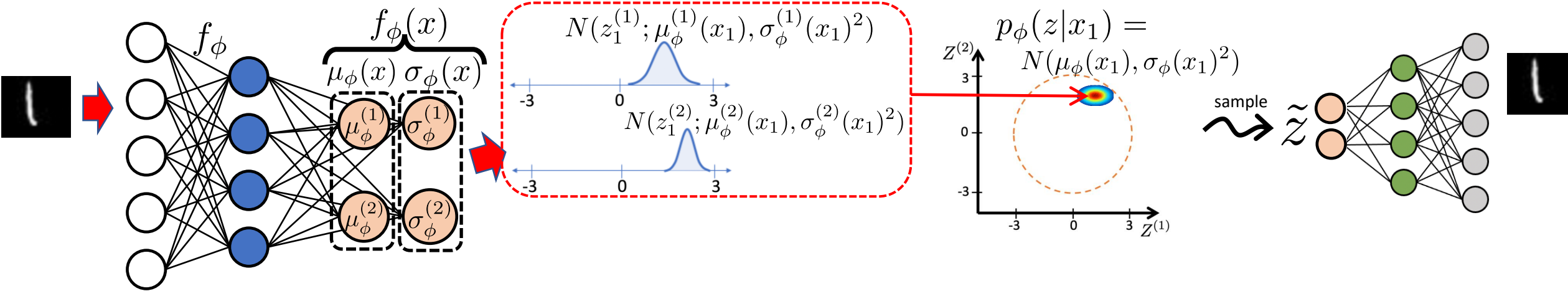
Each  $x$  is mapped to a multi-variate Normal/Gaussian distribution of  $z$ ,  $p_\phi(z|x) = N(\mu_\phi(x), \sigma_\phi(x)^2)$   
by predicting a mean and standard deviation for each dimension of  $z$ . (Note: diagonal covariance matrix, i.e. each dimension is uncorrelated)



# 1. Probabilistic Encoder

“posterior” distribution of  $z$ ,  
a.k.a. “conditional” distribution of  $z$  given  $x$

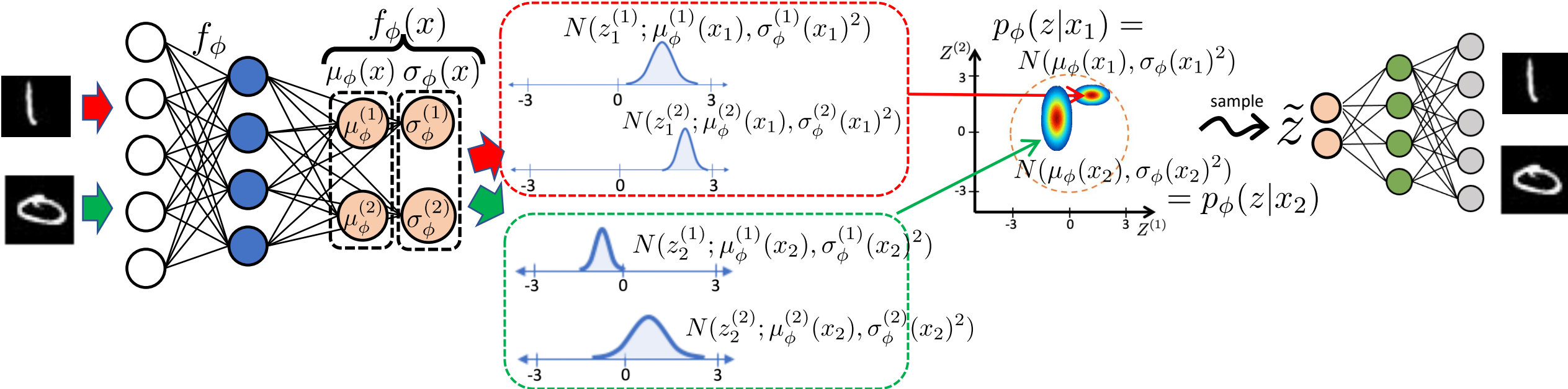
Each  $x$  is mapped to a multi-variate Normal/Gaussian distribution of  $z$ ,  $p_\phi(z|x) = N(\mu_\phi(x), \sigma_\phi(x)^2)$   
by predicting a mean and standard deviation for each dimension of  $z$ . (Note: diagonal covariance matrix, i.e. each dimension is uncorrelated)



# 1. Probabilistic Encoder

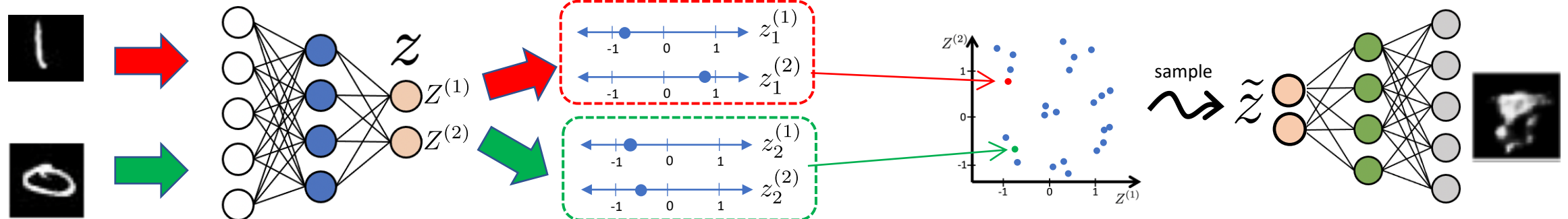
“posterior” distribution of  $z$ ,  
a.k.a. “conditional” distribution of  $z$  given  $x$

Each  $x$  is mapped to a multi-variate Normal/Gaussian distribution of  $z$ ,  $p_\phi(z|x) = N(\mu_\phi(x), \sigma_\phi(x)^2)$   
by predicting a mean and standard deviation for each dimension of  $z$ . (Note: diagonal covariance matrix, i.e. each dimension is uncorrelated)



Basic AE:

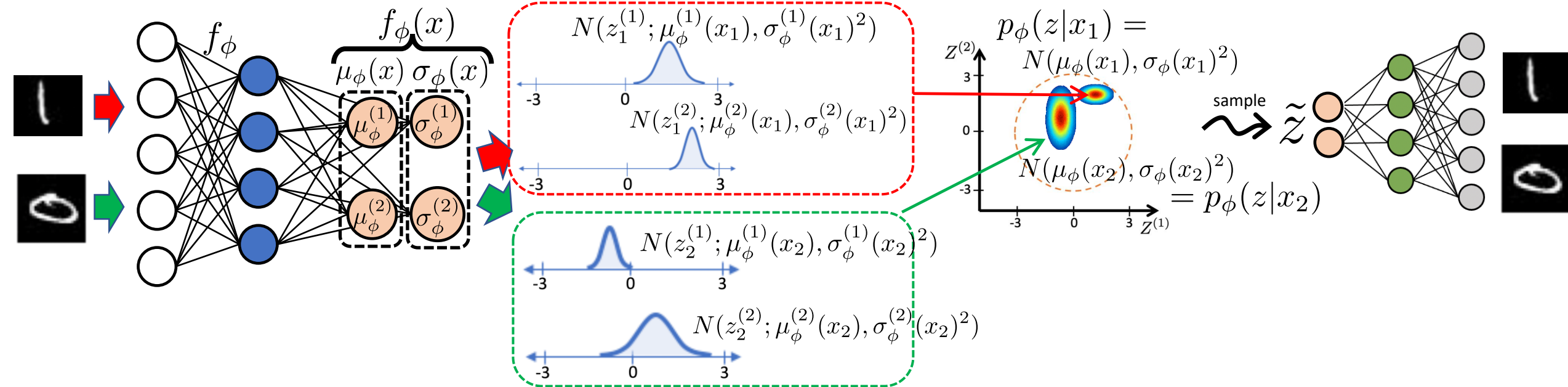
Each input  $x$  is mapped to a point  $z$



# 1. Probabilistic Encoder

“posterior” distribution of  $z$ ,  
a.k.a. “conditional” distribution of  $z$  given  $x$

Each  $x$  is mapped to a multi-variate Normal/Gaussian distribution of  $z$ ,  $p_\phi(z|x) = N(\mu_\phi(x), \sigma_\phi(x)^2)$   
by predicting a mean and standard deviation for each dimension of  $z$ . (Note: diagonal covariance matrix, i.e. each dimension is uncorrelated)



The final layer of VAE's encoder has **2 times the number of neurons** than the encoder of a standard AE.

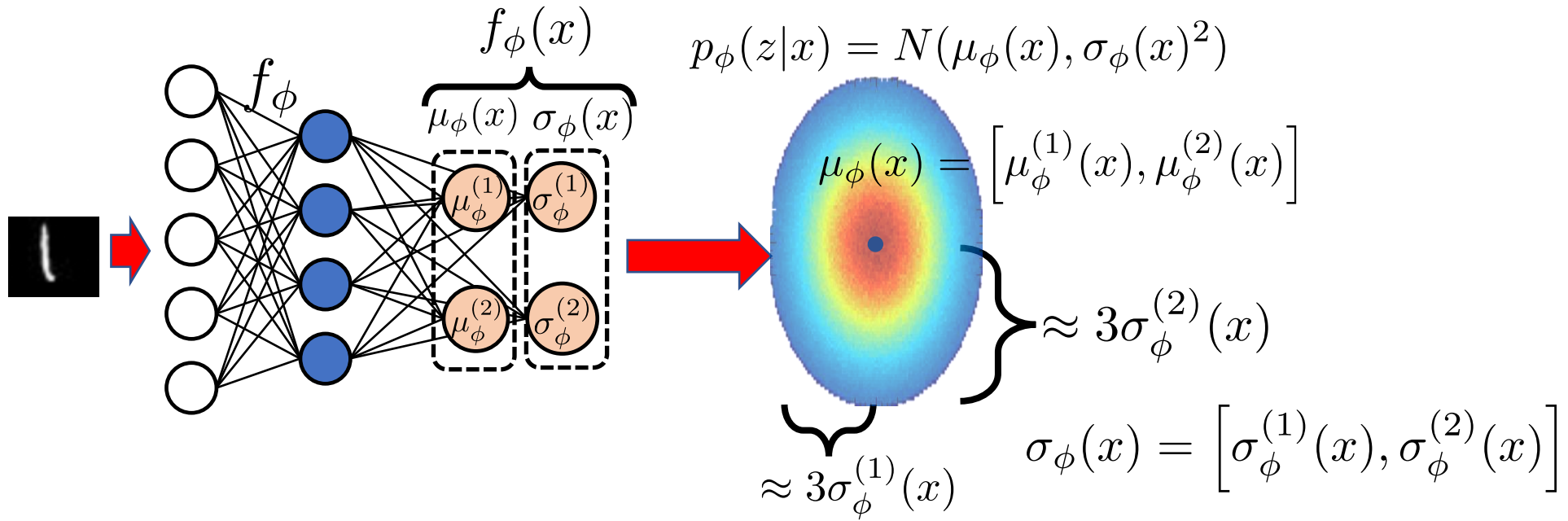
For  $z$  of dimension  $v$ ,  $z \in \mathbb{R}^v$ , encoder of VAE has  $2 \times v$  neurons.

For each dimension of  $z$ , the **standard AE's** encoder predicts **one value, the code**. The AE's encoder is **deterministic**.

For each dimension of  $z$ , the VAE's encoder predicts the **mean** and **standard deviation** of a **Gaussian distribution**.

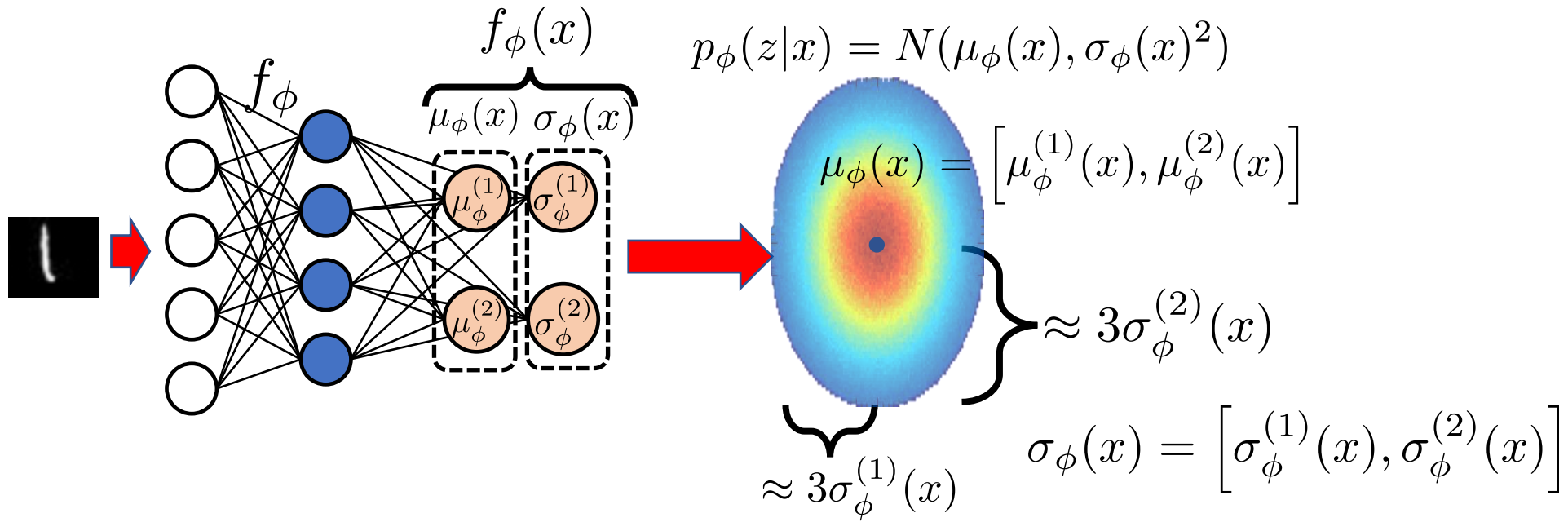
This distribution describes “**which values of  $z$  are likely the correct code**” for input  $x$ . VAE's encoder is **probabilistic/stochastic**.

# A closer look





How to interpret that for each input  $x$ , the encoder predicts a probability distribution for  $z$ ?

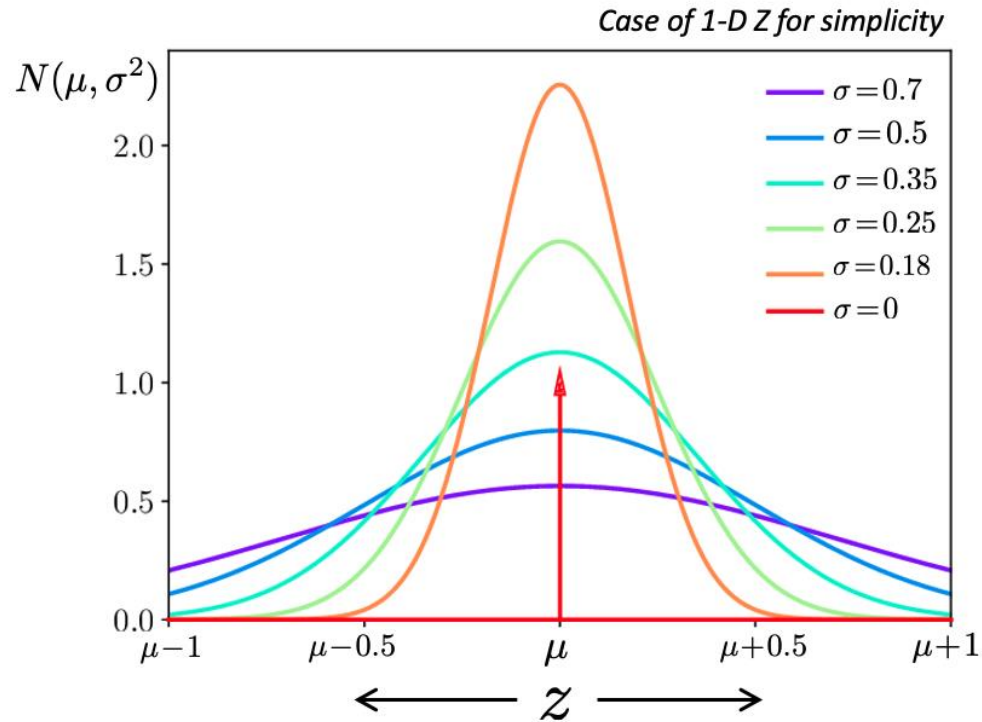


VAE's encoder expresses “**uncertainty**”:

For each sample  $x$ , the Gaussian distribution of  $z$  predicted by the encoder,  $p_\phi(z|x) = N(\mu_\phi(x), \sigma_\phi(x)^2)$  expresses **how “likely” is that a specific value of  $z$  is the correct code for  $x$ .**

In probabilistic terms, it expresses the **conditional probability of  $z$  given  $x$** ,  $p_\phi(z|x)$ , according to the model (encoder) with parameters  $\phi$ .

# Uncertainty (low probability) in encoding



Q: If predicted std. deviation  $\sigma_\phi(x) = 0$  ?

A: Totally **certain (prob=1)** that **z is the predicted mean!**  
This is the **basic Auto-Encoder's behavior!**

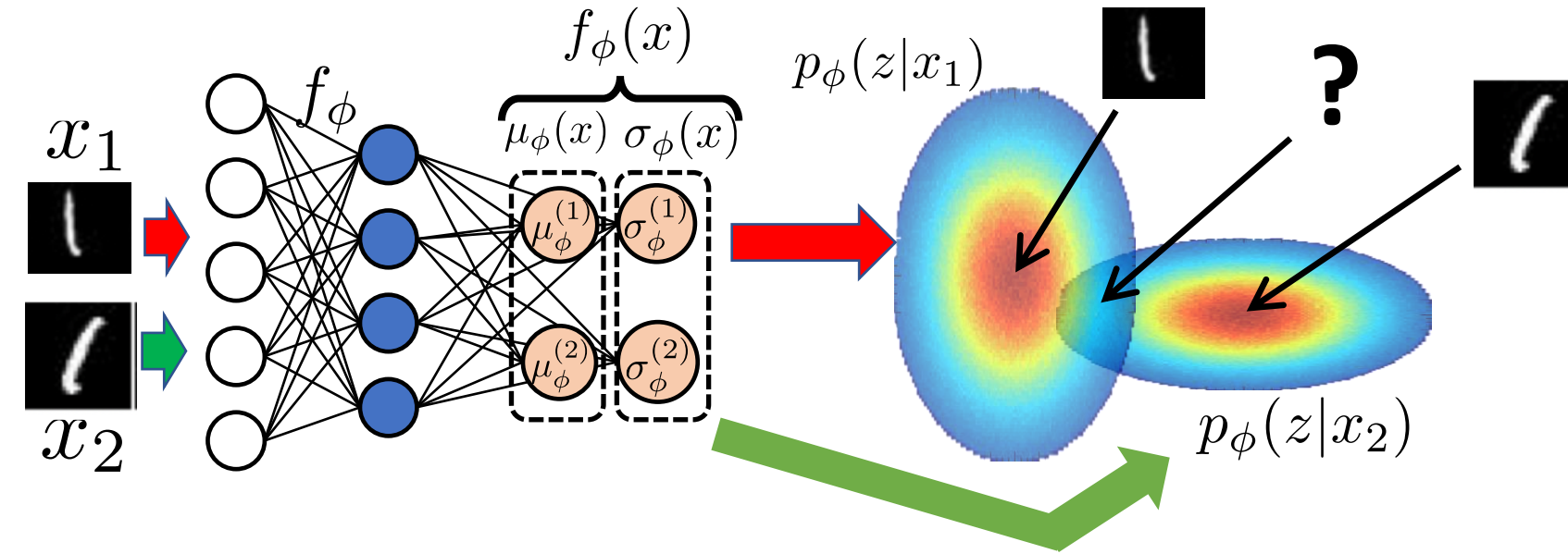
**Larger** standard deviation =>

larger range of values  $z$  may be the correct code of  $x$  =>

less certainty (probability) each of these  $z$  values is the “ideal” code for  $x$ .



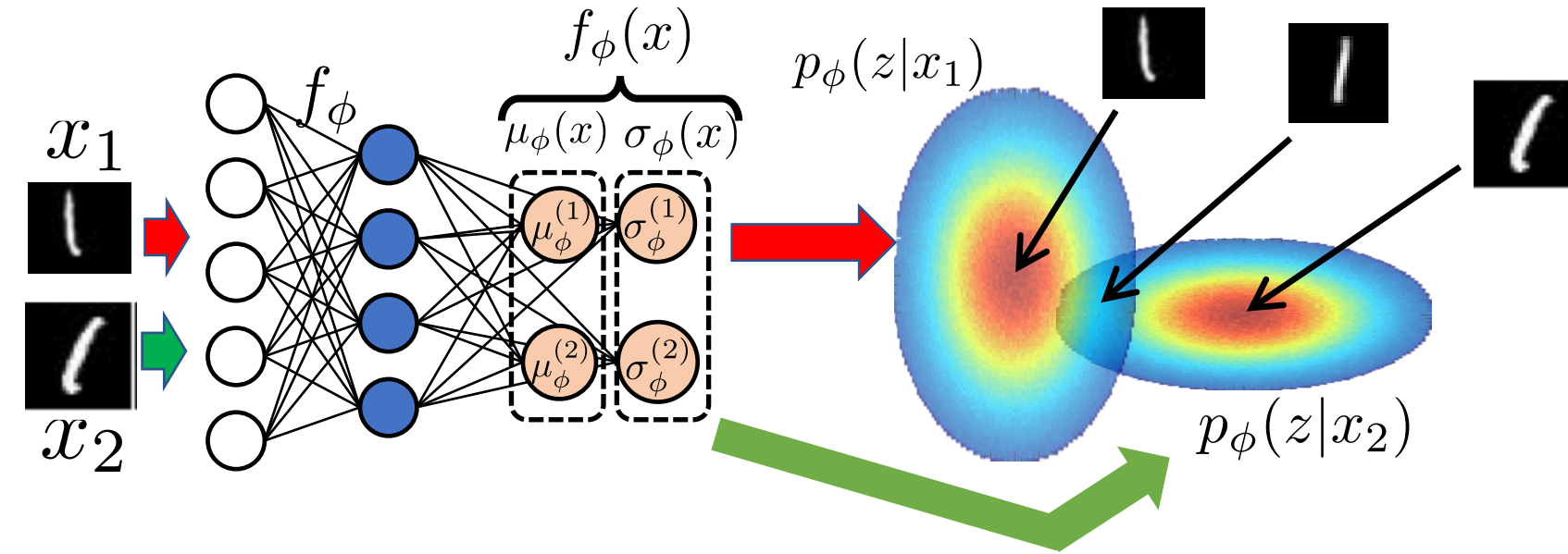
# How to interpret “overlaps” of predicted distributions for $z$ ?



This formulation of the encoder leads to existence of “overlaps” in the encoding of different samples.  
(in fact, Gaussians extend to infinity, even for very small values, so in fact, there are overlaps everywhere)

**Q: What would be the appearance of an image that would be mapped with a mean in the overlapping area of  $x_1$  and  $x_2$ ?**

# How to interpret “overlaps” of predicted distributions for $z$ ?

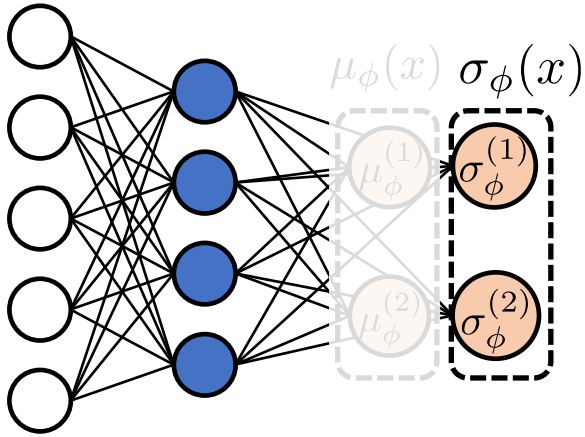


This formulation of the encoder leads to existence of “overlaps” in the encoding of different samples.  
(in fact, Gaussians extend to infinity, even for very small values, so in fact, there are overlaps everywhere)

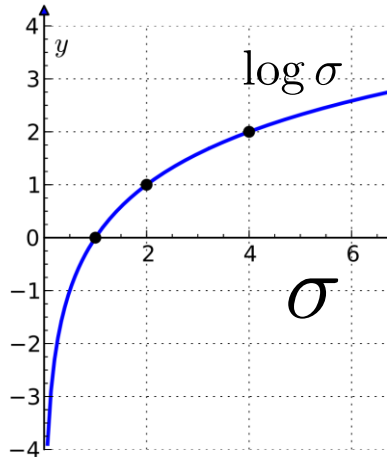
**Q:** What would be the appearance of an image that would be mapped with a mean in the overlapping area of  $x_1$  and  $x_2$ ?

**A:** Appearance that would be in-between those of  $x_1$  and  $x_2$

# Implementation trick: predict $\log(std.dev.)$



1. Standard deviation *\*must\** be positive.  
=> Cant use activation functions that allow negatives.
2. Standard deviations for each  $x$  tend to be small, close to zero  
=> Cant use ReLU. Not good gradients around 0, but std.devs can and will take such low values.

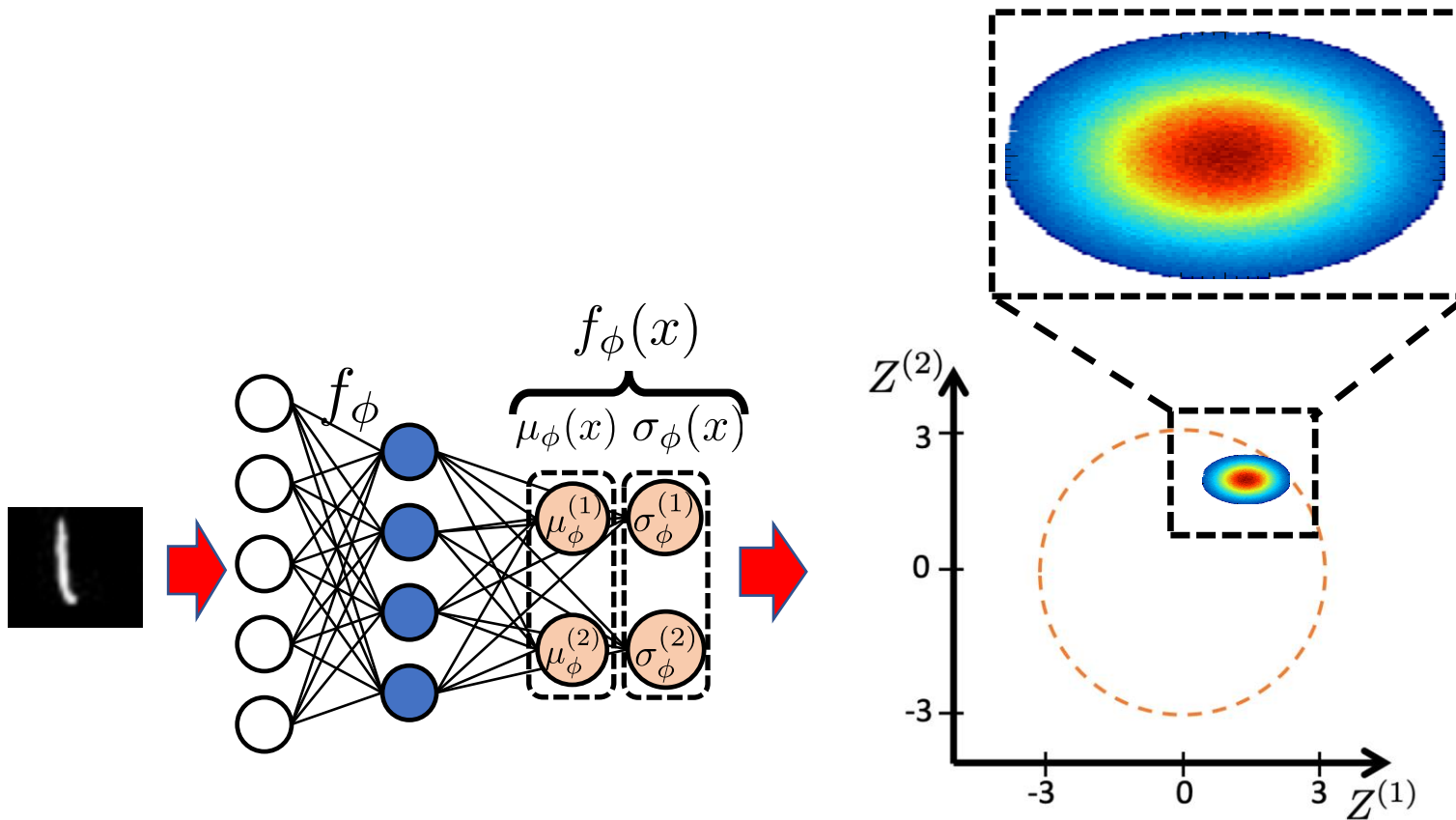


Common solution in implementations:

- Treat Encoder as if it predicts  $\log(\sigma_\phi(x))$   
`log_stddev = matrix_mult(act_layer1 , w_layer2)`
- Compute sigma as:  $\sigma_\phi(x) = \exp(\log(\sigma_\phi(x)))$   
`stddev = exp(log_stddev)`

## 2. Stochastic Decoder (during training)

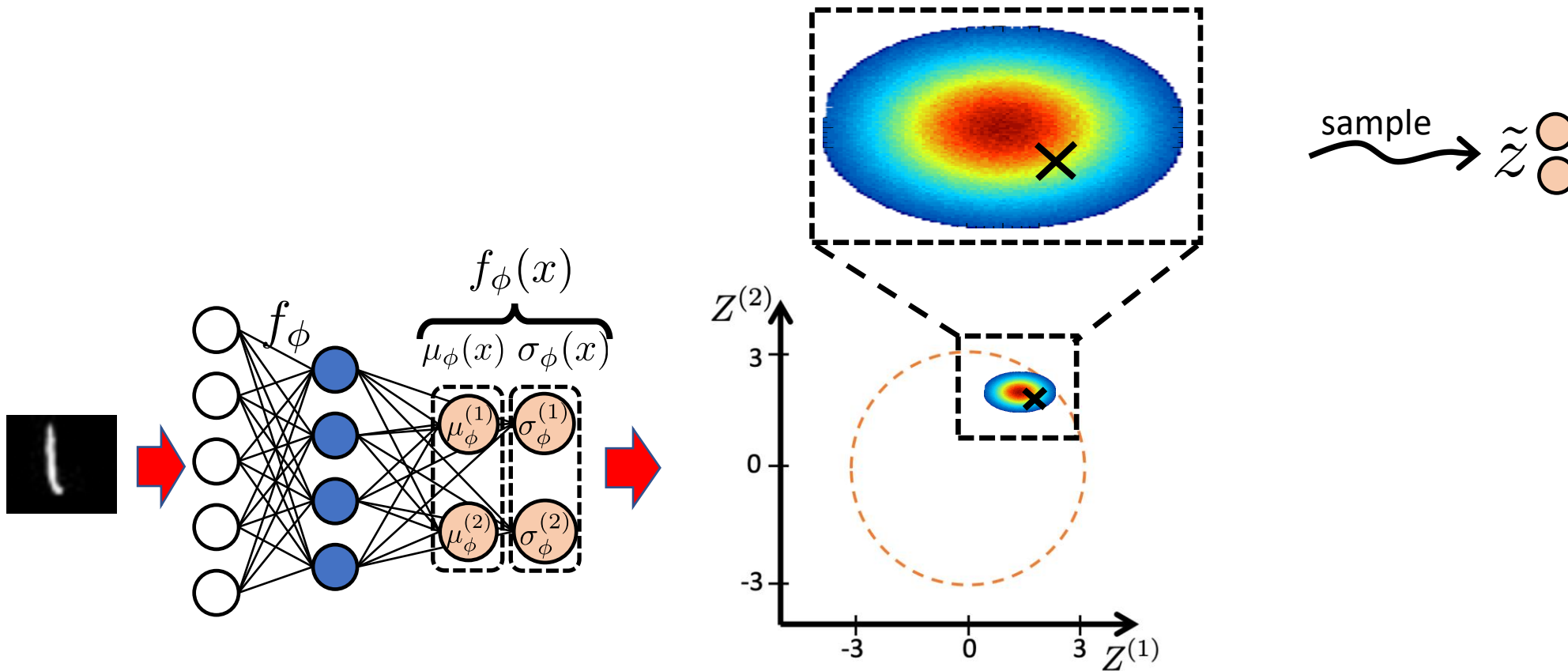
$$p_{\phi}(z|x) = N(\mu_{\phi}(x), \sigma_{\phi}(x)^2)$$



## 2. Stochastic Decoder (during training)

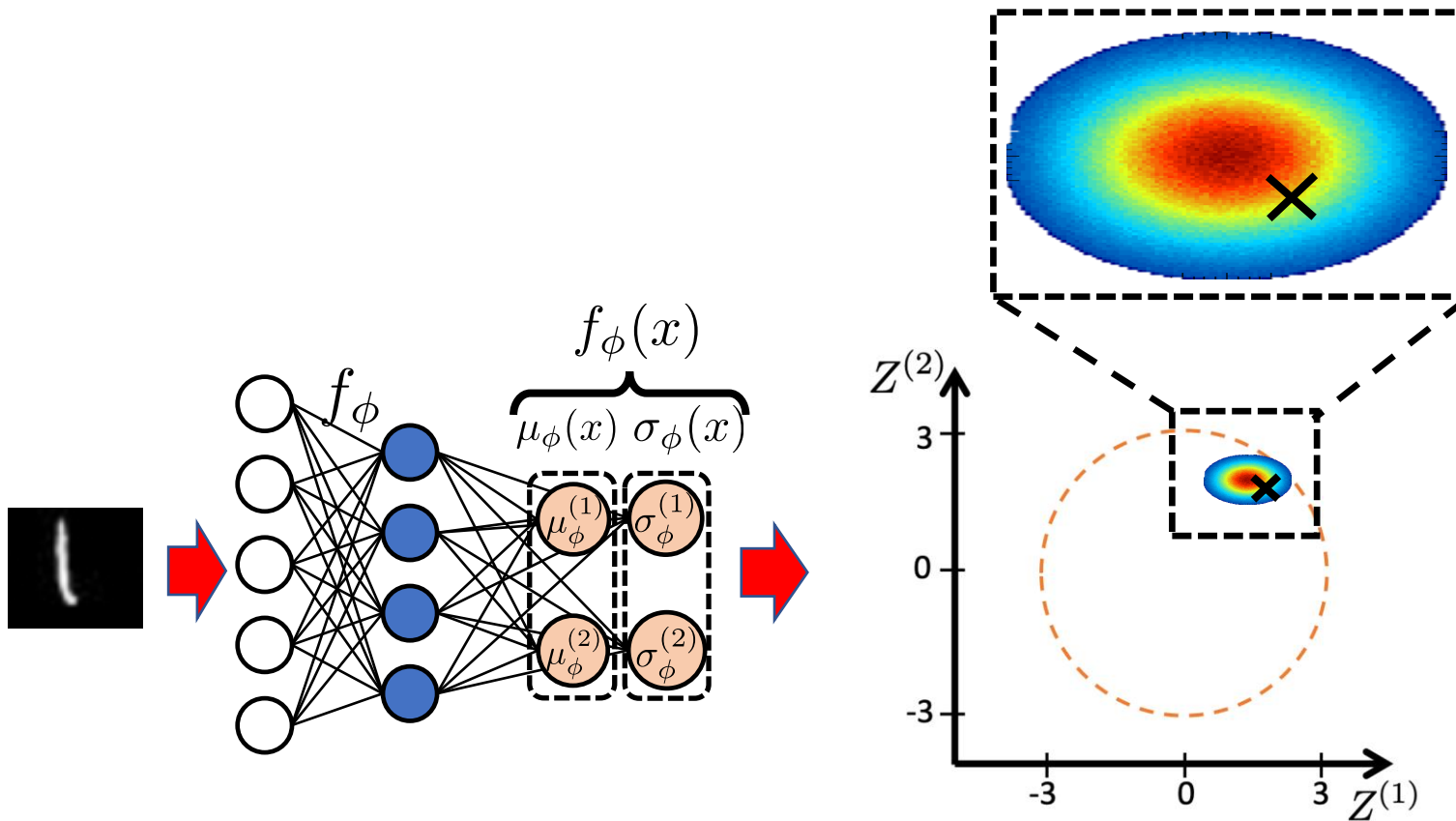
$$p_{\phi}(z|x) = N(\mu_{\phi}(x), \sigma_{\phi}(x)^2)$$

**Step 1.** Sample a code  $\tilde{z}$  from the predicted Gaussian  $p_{\phi}(z|x)$

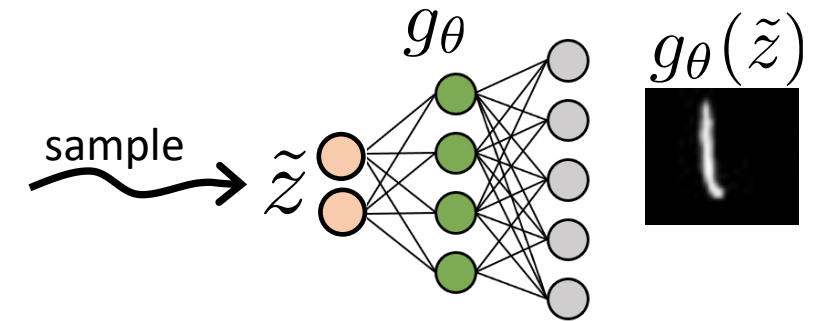


## 2. Stochastic Decoder (during training)

$$p_{\phi}(z|x) = N(\mu_{\phi}(x), \sigma_{\phi}(x)^2)$$



**Step 1.** Sample a code  $\tilde{z}$  from the predicted Gaussian  $p_{\phi}(z|x)$

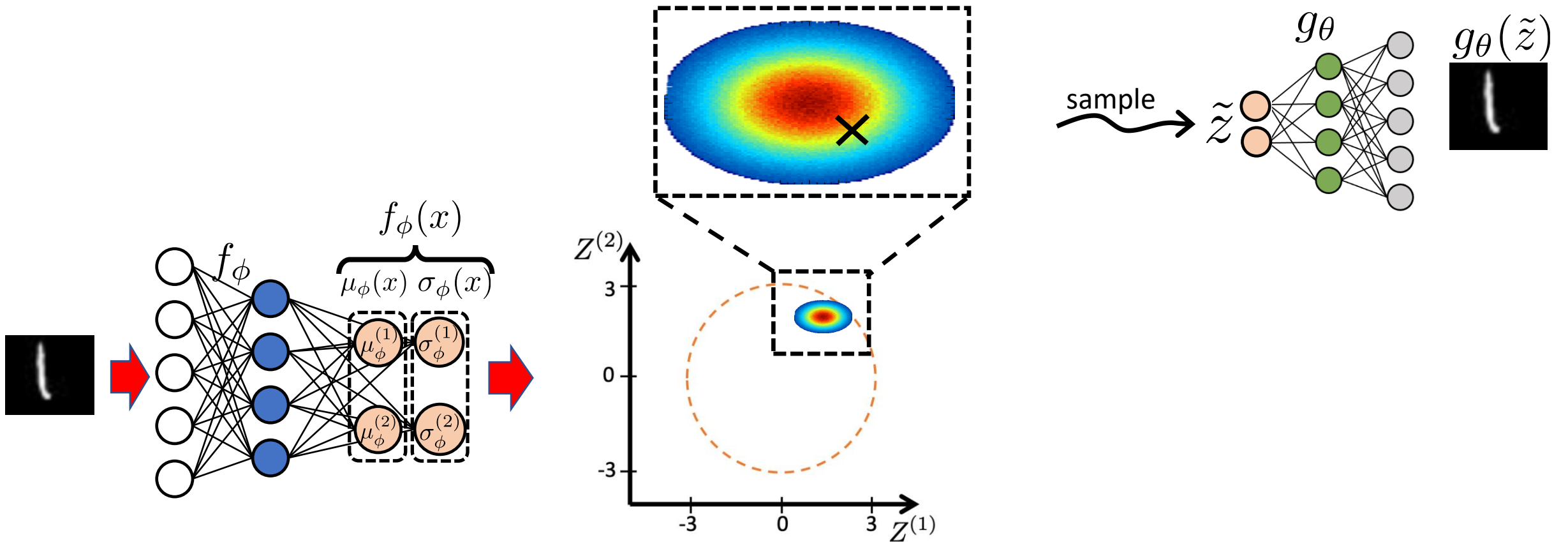


**Step 2.** Decode  $\tilde{z}$  with a standard decoder, similar to a basic AE.

## 2. Stochastic Decoder (during training)

At each training iteration of SGD, for each input  $x$  in the batch, we draw 1 sample.

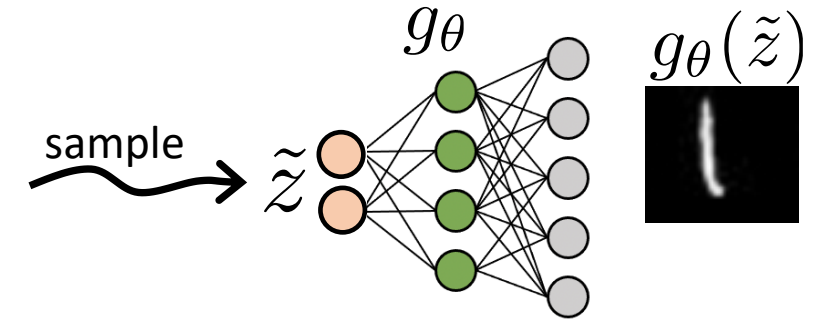
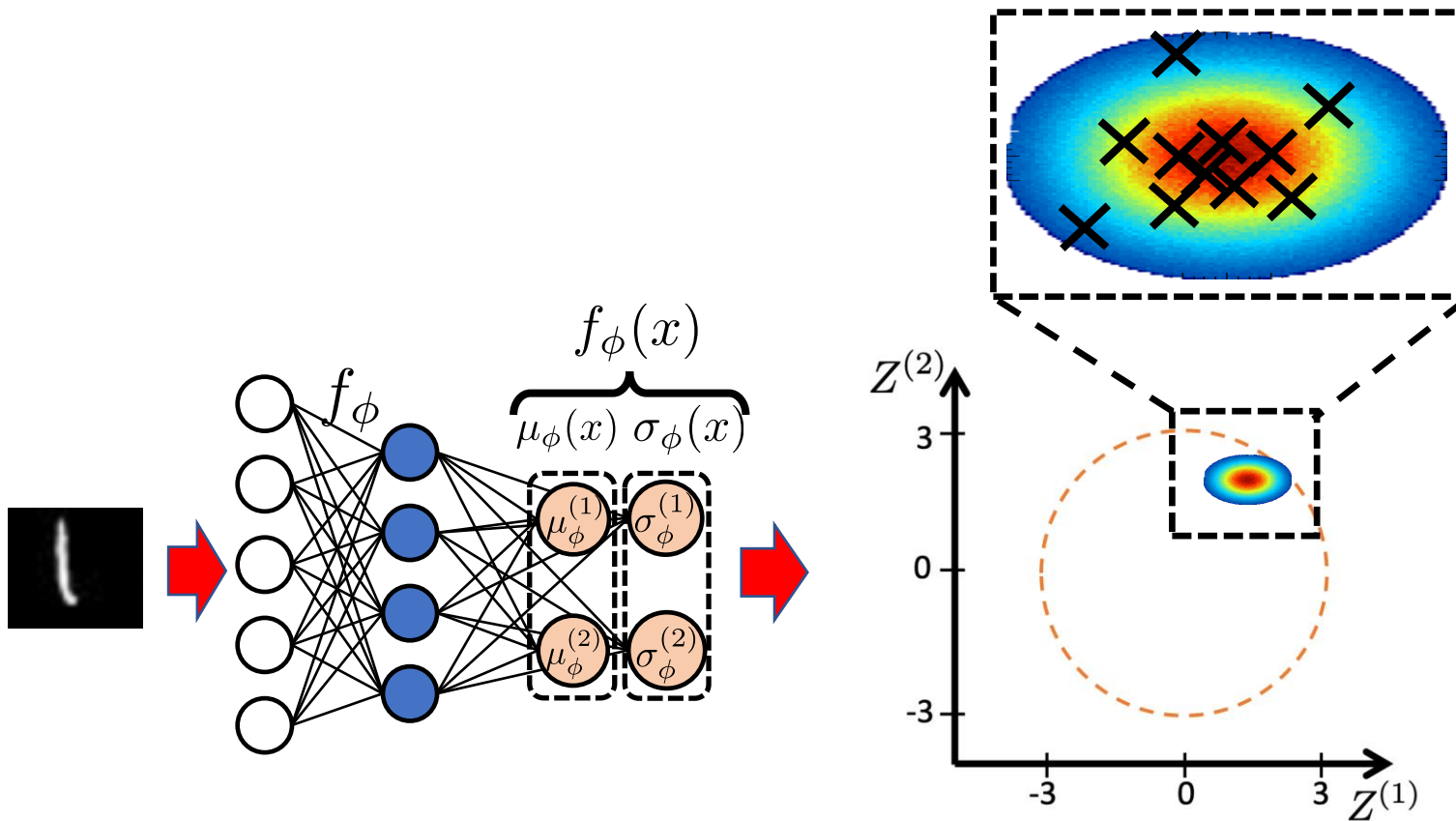
$$p_{\phi}(z|x) = N(\mu_{\phi}(x), \sigma_{\phi}(x)^2)$$



## 2. Stochastic Decoder (during training)

At each training iteration of SGD, for each input  $x$  in the batch, we draw 1 sample.

$$p_{\phi}(z|x) = N(\mu_{\phi}(x), \sigma_{\phi}(x)^2)$$



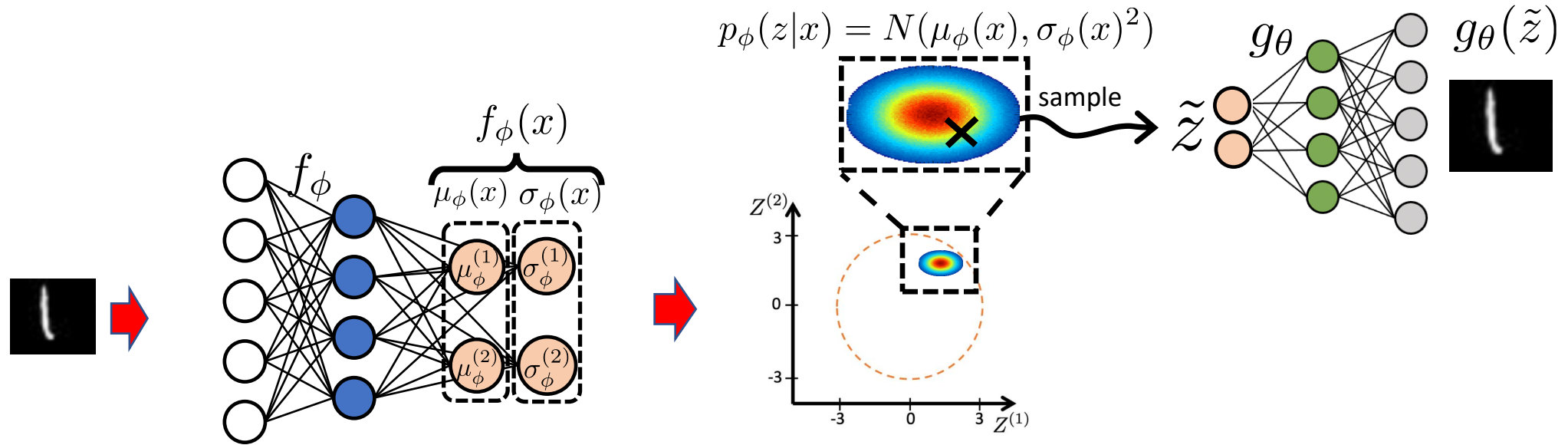
However, at different SGD iterations, different samples may be drawn for the same  $x$ .

For any input  $x$  and predicted  $p_{\phi}(z|x)$  it is **more likely** that we will sample value  $\tilde{z}$  that is close to the predicted mean  $\mu_{\phi}(x)$ , rather than further.

At different SGD iterations, the decoder should try to reconstruct the input for **any drawn sample**, even the unlikely ones.



### 3. Training Objective

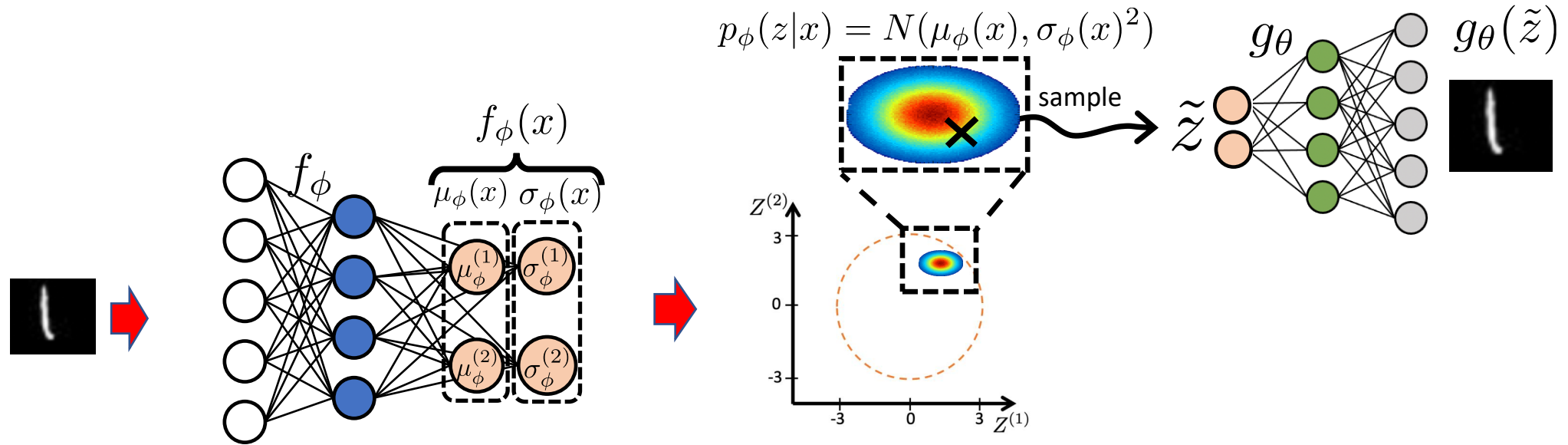


What do we want to train this model to do?

Create realistic images with the decoder

Predicted (distributions of) codes to cover the prior  $p(z)$

### 3. Training Objective



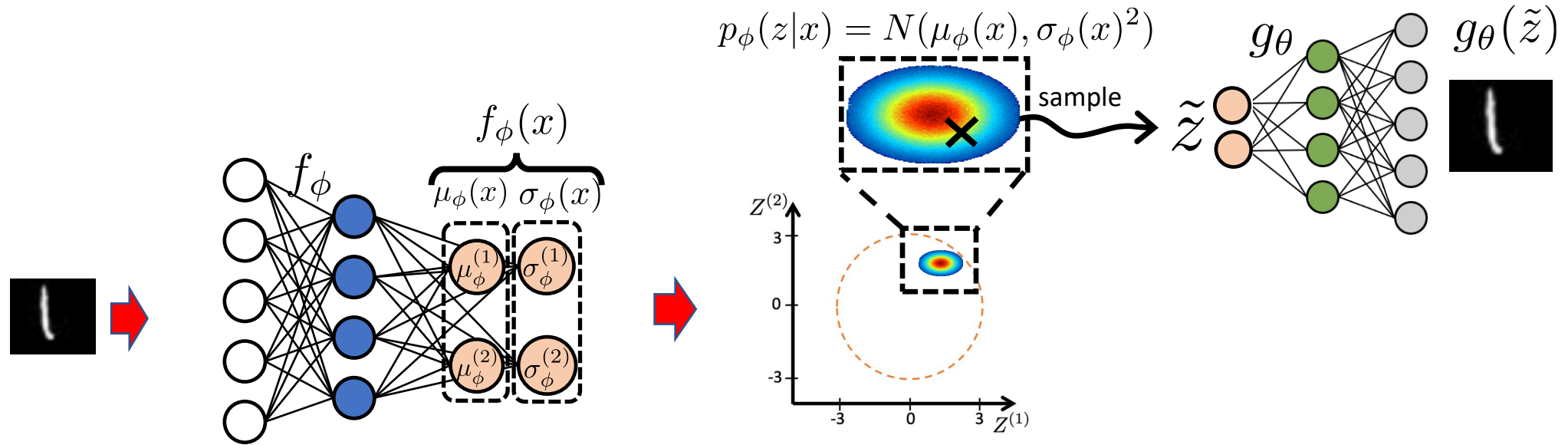
**Reconstruction loss**      **Regularizer**

$$\mathcal{L}_{VAE} = \underbrace{\mathcal{L}_{rec}}_{\text{Reconstruction loss}} + \underbrace{\lambda \mathcal{L}_{reg}}_{\text{Regularizer}}$$

↓

**Strength of Regularizer**  
(chosen by us through experimentation)

### 3. Training Objective



Reconstruction loss      Regularizer

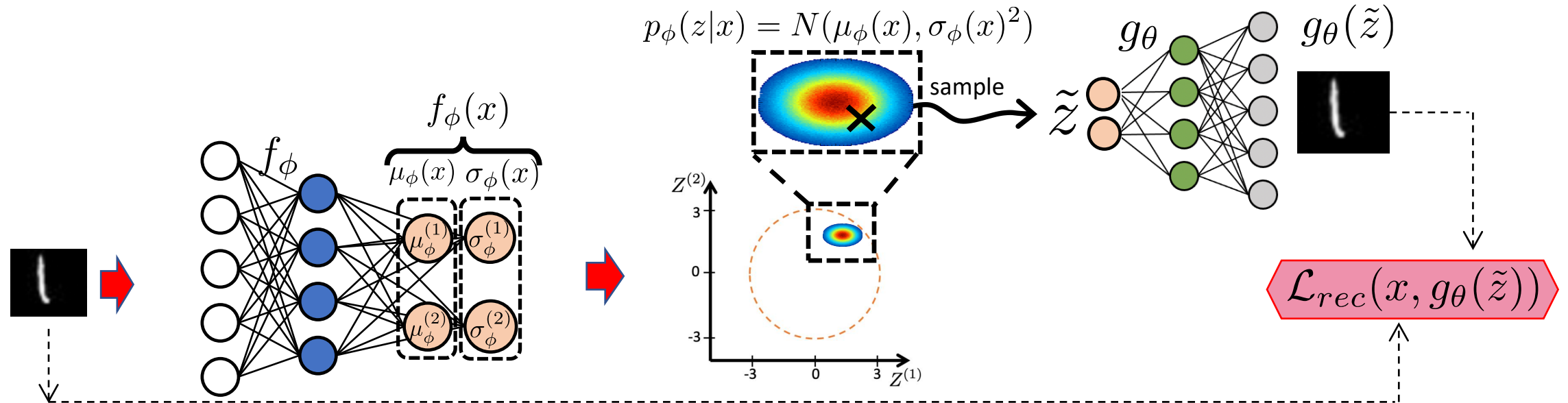
$$\mathcal{L}_{VAE} = \underbrace{\mathcal{L}_{rec}}_{\text{Reconstruction loss}} + \underbrace{\lambda \mathcal{L}_{reg}}_{\text{Regularizer}}$$

$$\{\theta', \phi'\} = \arg \min_{\theta, \phi} \underbrace{\mathbb{E}_{x \sim p_{data}} [\mathcal{L}_{VAE}]}_{\text{Expected Value}} = \arg \min_{\theta, \phi} \mathbb{E}_{x \sim p_{data}} [\mathcal{L}_{rec}] + \lambda \mathbb{E}_{x \sim p_{data}} [\mathcal{L}_{reg}]$$

“Minimize the **Expected Value** of the VAE loss, when  $x$  is sampled from the data distribution”

In other words: “Minimize the VAE loss **on average** over the training data  $x$ ”

### 3. Training Objective: Reconstruction loss



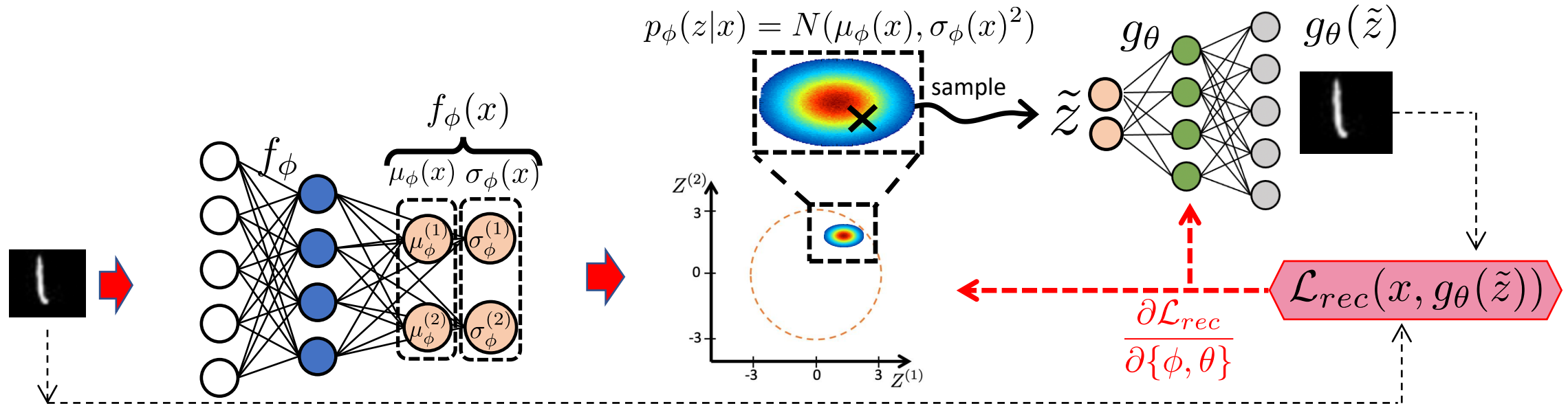
$$\{\theta', \phi'\} = \arg \min_{\theta, \phi} \mathbb{E}_{x \sim p_{data}} [\mathcal{L}_{VAE}] = \arg \min_{\theta, \phi} \mathbb{E}_{x \sim p_{data}} [\mathcal{L}_{rec}] + \lambda \mathbb{E}_{x \sim p_{data}} [\mathcal{L}_{reg}]$$

**Reconstruction loss:**

$$\mathcal{L}_{rec} = \frac{1}{d} \sum_{j=1}^d (x^{(j)} - g_\theta(\tilde{z})^{(j)})^2 \quad \text{for } \tilde{z} \sim p_\phi(z|x) \quad \text{Reconstruct input } x, \text{ for any } \tilde{z} \text{ sampled from } p_\phi(z|x).$$

Dimensionality of  $x$

### 3. Training Objective: Reconstruction loss



$$\{\theta', \phi'\} = \arg \min_{\theta, \phi} \mathbb{E}_{x \sim p_{data}} [\mathcal{L}_{VAE}] = \arg \min_{\theta, \phi} \mathbb{E}_{x \sim p_{data}} [\mathcal{L}_{rec}] + \lambda \mathbb{E}_{x \sim p_{data}} [\mathcal{L}_{reg}]$$

**Reconstruction loss:**

$$\mathcal{L}_{rec} = \frac{1}{d} \sum_{j=1}^d (x^{(j)} - g_\theta(\tilde{z})^{(j)})^2$$

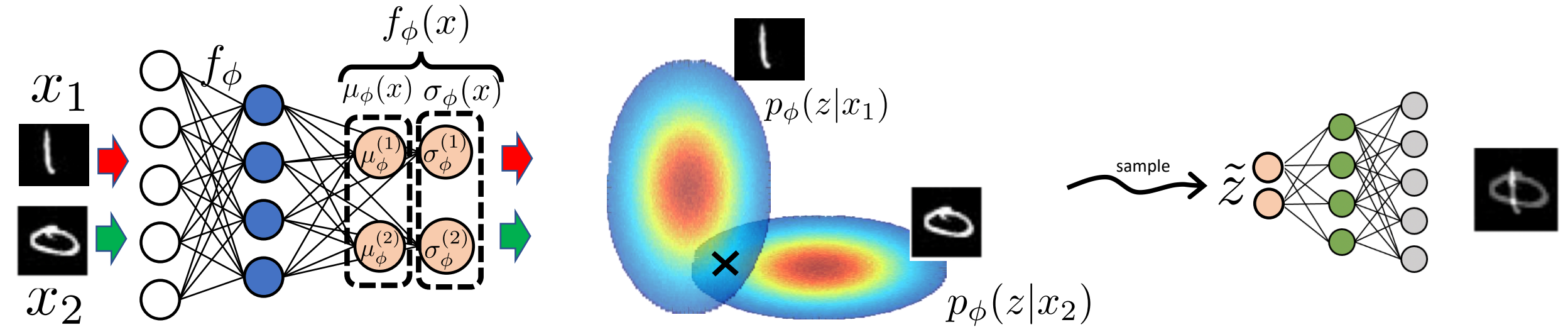
Function of  $\theta$   
(decoder params)

Function of  $\phi$   
(encoder params)

The Reconstruction loss is a **function of all parameters**.

Therefore it **trains BOTH** the encoder **AND** the decoder.

# Effect of Reconstruction loss:

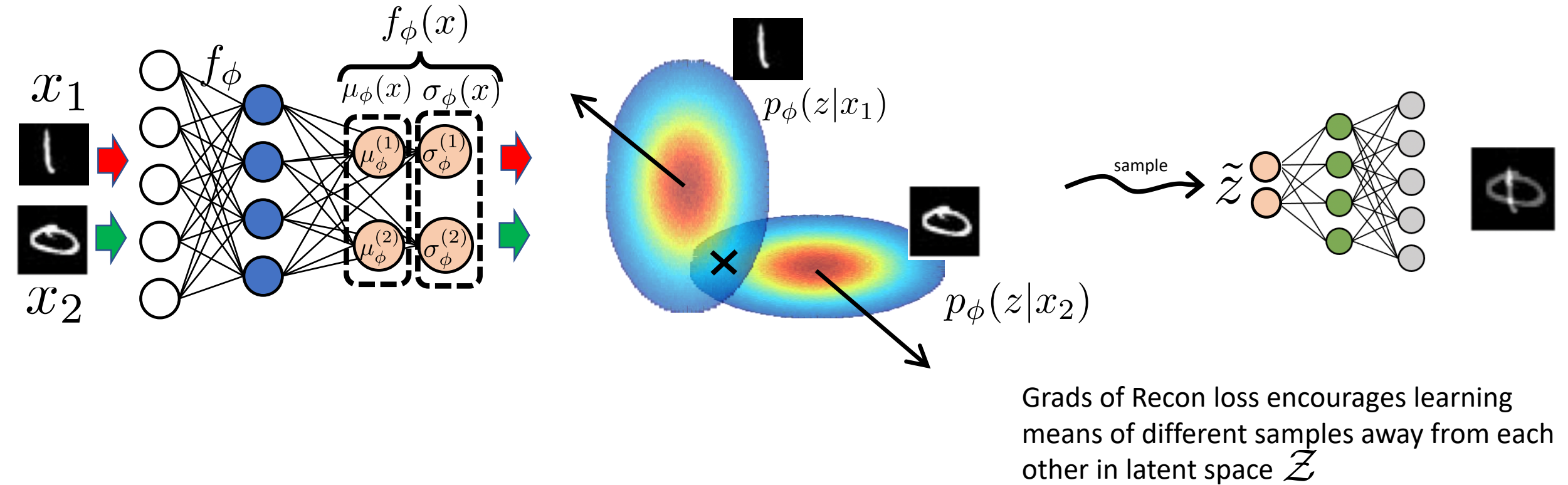


Assume that an encoder learns to encode these two digits as shown.

**Q: For what values of  $z$  do you expect *high reconstruction error*?**

**A: Overlapping area**

# Effect of Reconstruction loss: on the predicted means

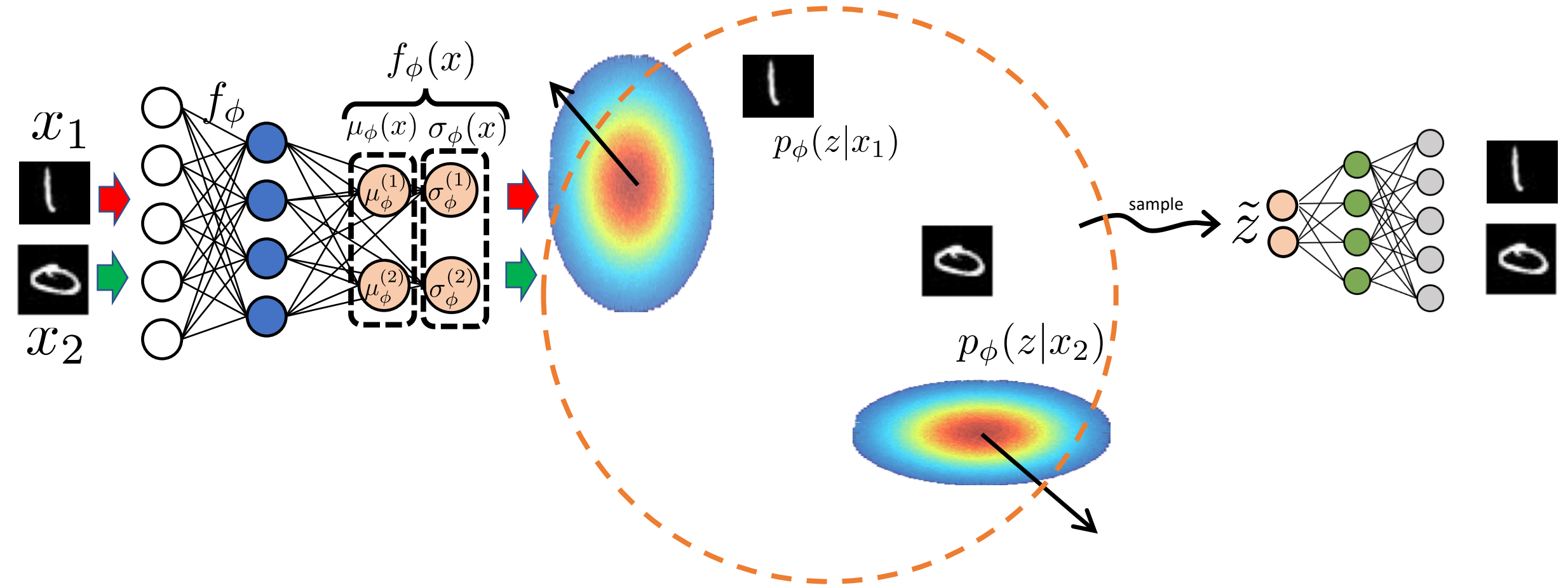


Q: How can the encoder reduce overlapping areas **by changing the predicted means**?

A: Learn parameters that predict **means that are away from each other**.



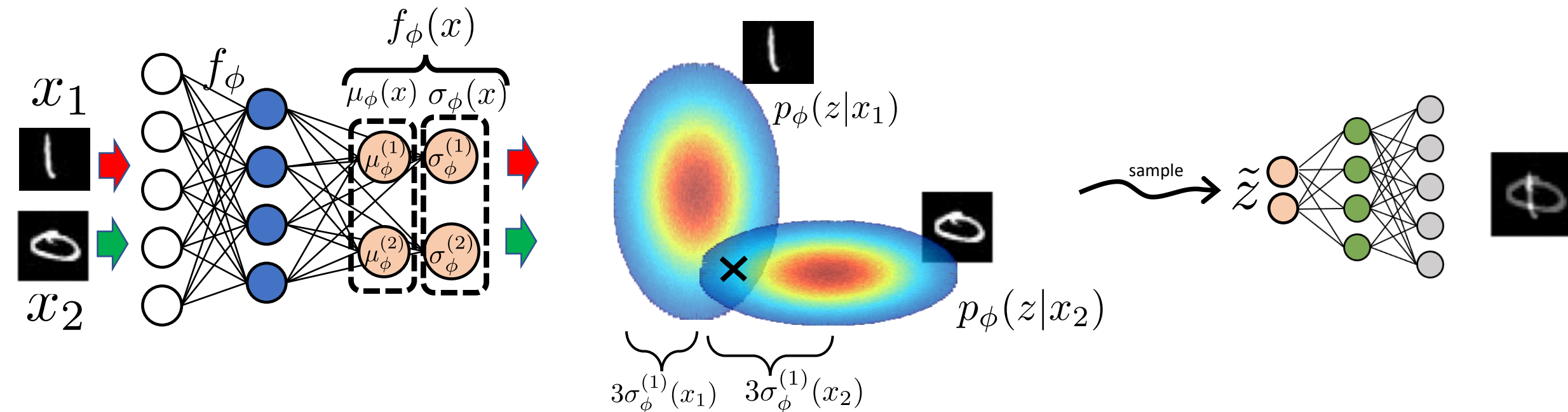
# Effect of Reconstruction loss: on the predicted means



If we would minimize ONLY the Reconstruction loss, encoder can learn to predict means of  $p(z|x)$  that are **arbitrarily far**. They can take values outside the area covered by the “prior”  $p(z)$ .



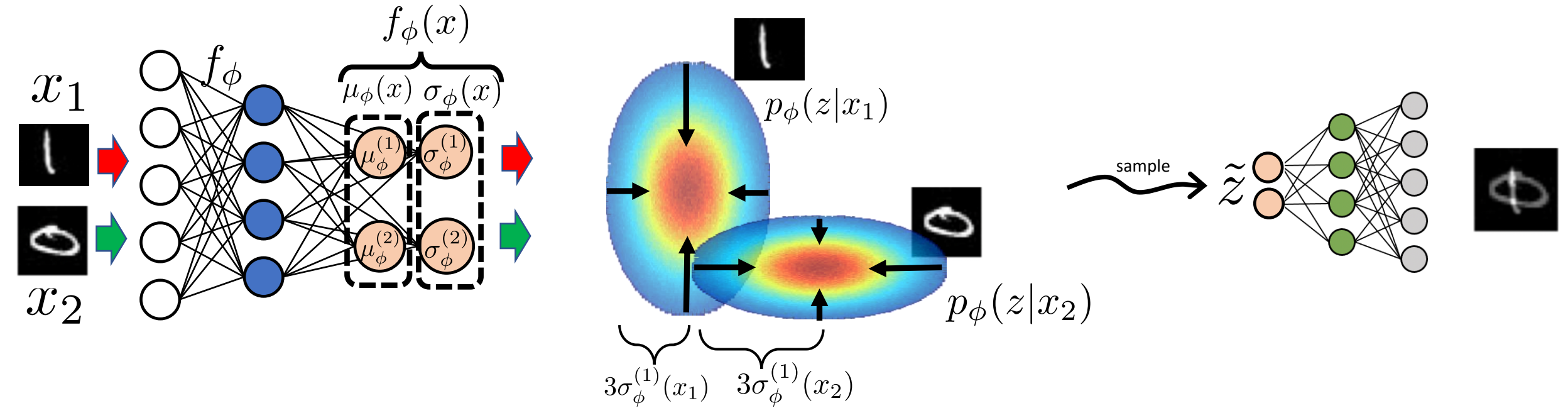
## Effect of Reconstruction loss: on the predicted std. deviations



Q: How can the encoder reduce overlapping areas *by changing the predicted standard deviations*?

A: Learn parameters that predict *small standard deviations*.

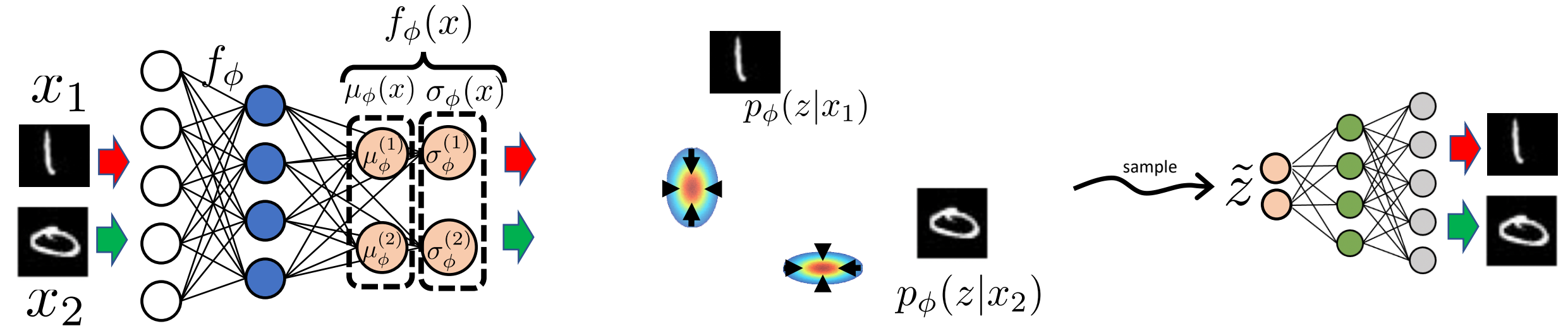
# Effect of Reconstruction loss: on the predicted std. deviations



Q: How can the encoder reduce overlapping areas **by changing the predicted standard deviations**?

A: Learn parameters that predict **small standard deviations**.

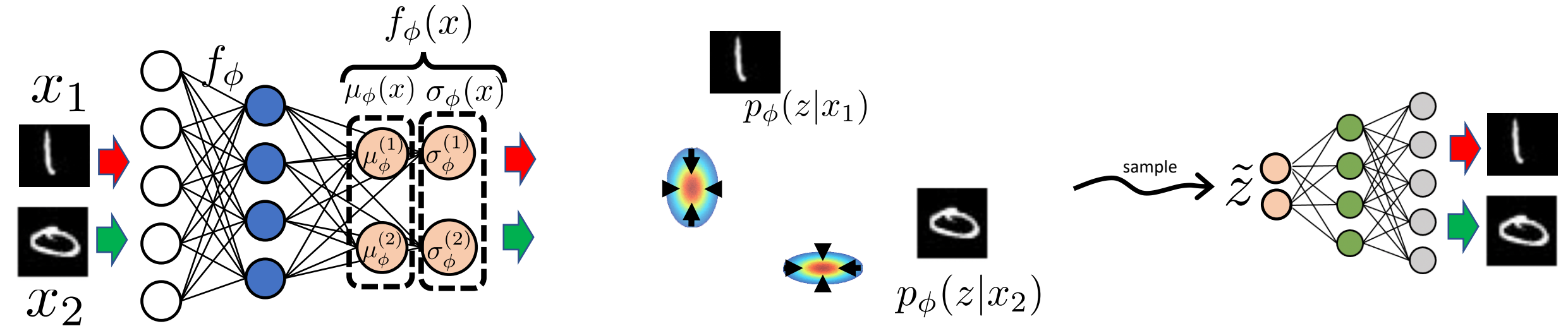
# Effect of Reconstruction loss: on the predicted std. deviations



Q: How can the encoder reduce overlapping areas **by changing the predicted standard deviations**?

A: Learn parameters that predict **small standard deviations**.

# Effect of Reconstruction loss: on the predicted std. deviations

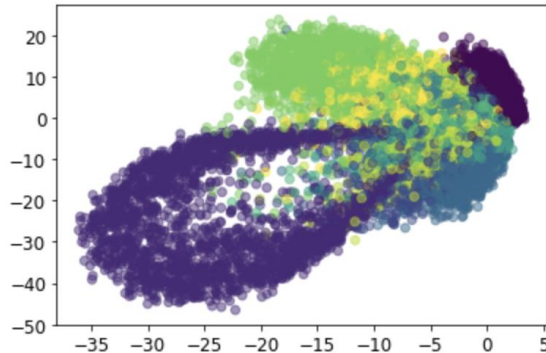


If we would minimize **ONLY** the Reconstruction loss,  
“ideal” standard deviations for reconstruction loss are zero for every  $x$ !

# Influence of Reconstruction loss:

If we would train the VAE to only minimize the Reconstruction loss:

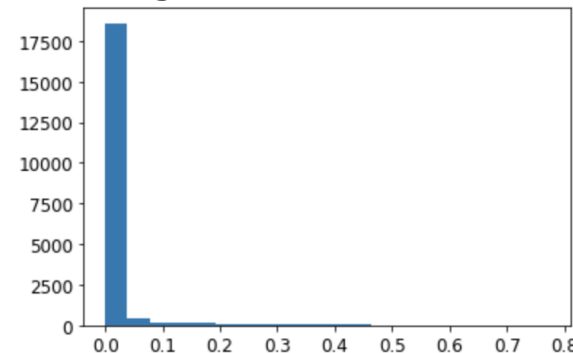
**Means in 2D space of Z:**



For each  $x$ , predicted **means**:

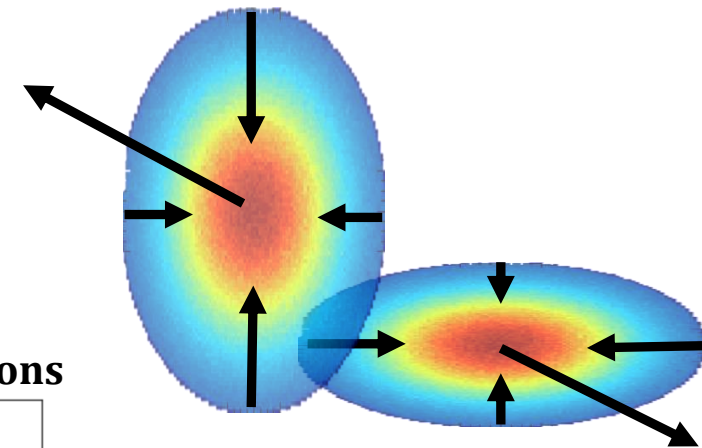
- Dissimilar  $x$  to different means of  $z$ .  
More similar samples can be closer.
- Means are pushed to **arbitrarily high values**

**Histogram of Std. Deviations**



For each  $x$ , predicted **standard deviations**:

- Tend to be small, ideally 0.
- Distributions  $p(z|x)$  tend to collapse to their mean.  
Therefore **cant cover prior  $p(z)$** .
- The encoder tends to become **deterministic**.



**If we would train a VAE only with the Reconstruction loss, its learned parameters would make it behave exactly like the basic deterministic AE!**

**So, the “probabilistic encoder” architecture is not sufficient! We need the Regularizer!**

### 3. Training Objective: Regularizer

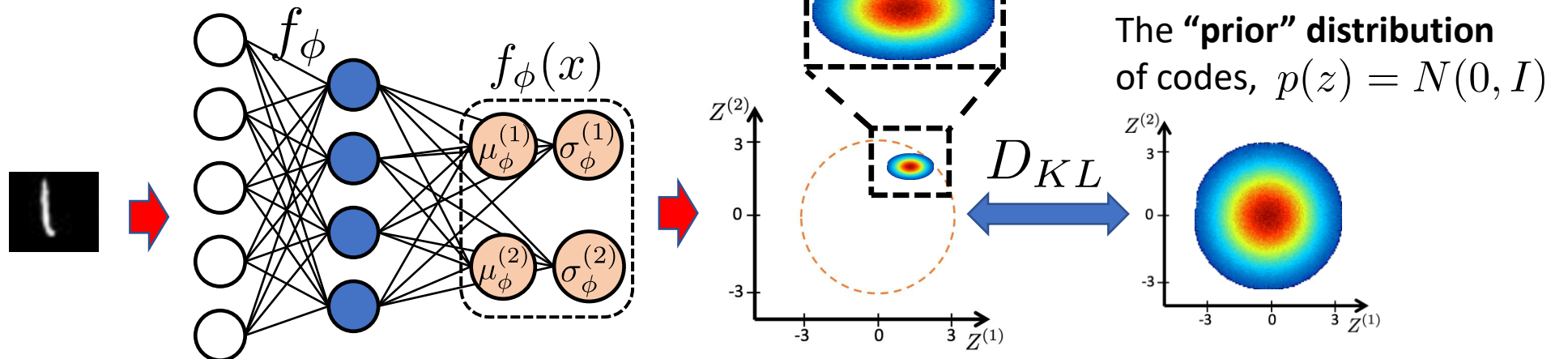
$$\{\theta', \phi'\} = \arg \min_{\theta, \phi} \mathbb{E}_{x \sim p_{data}} [\mathcal{L}_{VAE}] = \arg \min_{\theta, \phi} \mathbb{E}_{x \sim p_{data}} [\mathcal{L}_{rec}] + \lambda \mathbb{E}_{x \sim p_{data}} [\mathcal{L}_{reg}]$$

“Expected value (average over training data) of the **Kullback-Leibler (KL) Divergence**” between two distributions

$$\mathbb{E}_{x \sim p_{data}} D_{KL} [p_{\phi}(z|x) || N(0, I)]$$

Measure of difference  
between **2 distributions**

$$p_{\phi}(z|x) = N(\mu_{\phi}(x), \sigma_{\phi}(x)^2)$$

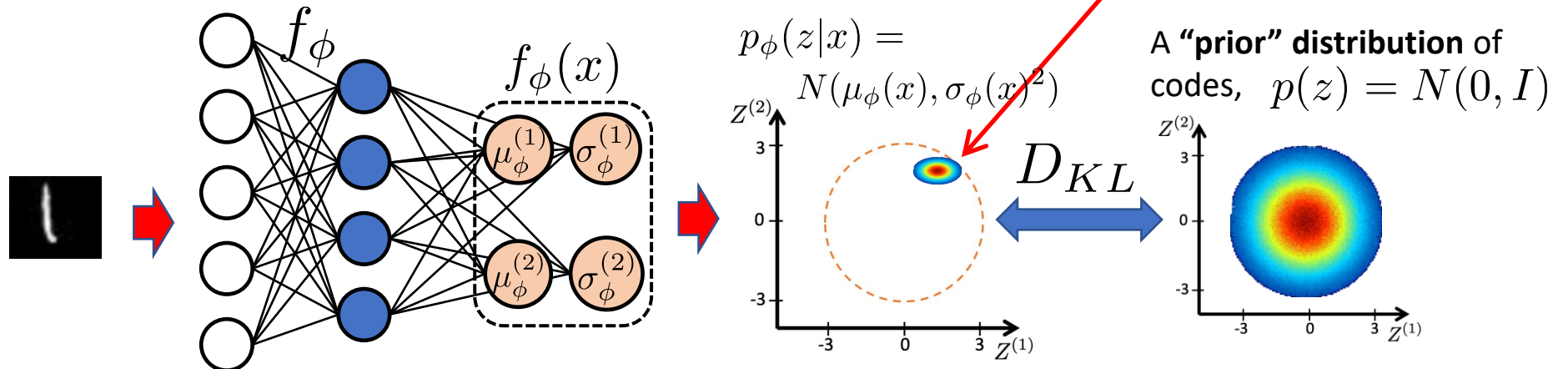


### 3. Training Objective: Regularizer

$$\{\theta', \phi'\} = \arg \min_{\theta, \phi} \mathbb{E}_{x \sim p_{data}} [\mathcal{L}_{VAE}] = \arg \min_{\theta, \phi} \mathbb{E}_{x \sim p_{data}} [\mathcal{L}_{rec}] + \lambda \mathbb{E}_{x \sim p_{data}} [\mathcal{L}_{reg}]$$

#### What this regularizer does:

“Learn encoder parameters  $\phi$  such that on average over the training data, we minimize the “Kullback Leibler (KL) Divergence” between the predicted posterior distribution of z codes,  $p_\phi(z|x)$ , and the Gaussian distribution  $N(0, I)$ ”





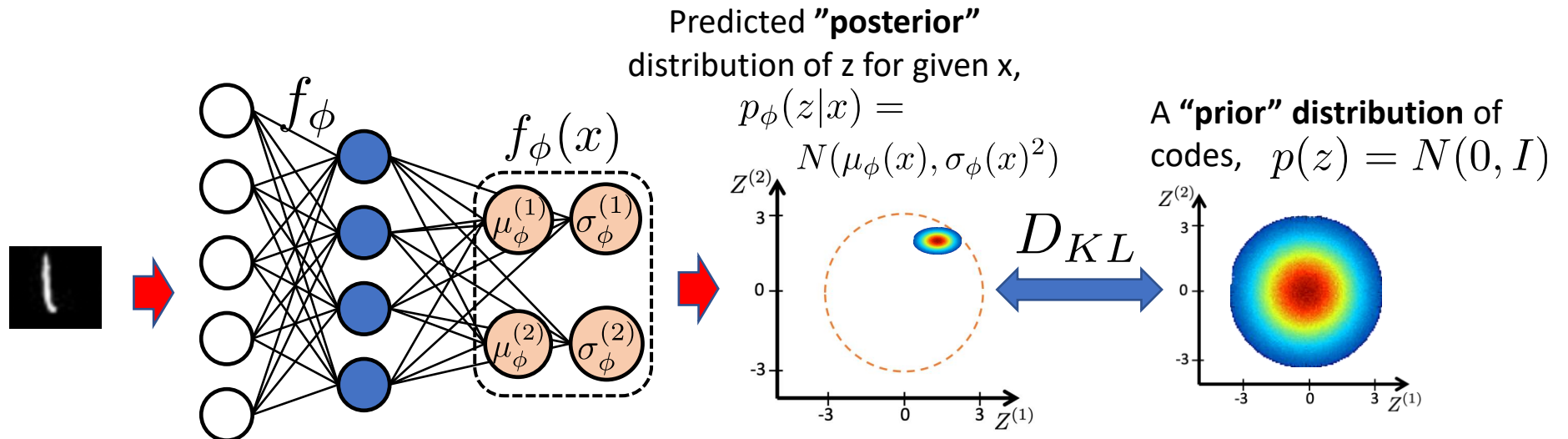
### 3. Training Objective: Regularizer

$$\{\theta', \phi'\} = \arg \min_{\theta, \phi} \mathbb{E}_{x \sim p_{data}} [\mathcal{L}_{VAE}] = \arg \min_{\theta, \phi} \mathbb{E}_{x \sim p_{data}} [\mathcal{L}_{rec}] + \lambda \mathbb{E}_{x \sim p_{data}} [\mathcal{L}_{reg}]$$

In the specific case of KL divergence between two Gaussian distributions, it is equal to:

$$D_{KL} [p_{\phi}(z|x) || N(0, I)]$$

$$= \frac{1}{2} \sum_{j=1}^v \left[ (\mu_{\phi}^{(j)}(x))^2 + (\sigma_{\phi}^{(j)}(x))^2 - 2 \log \sigma_{\phi}^{(j)}(x) - 1 \right]$$





### 3. Training Objective: Regularizer

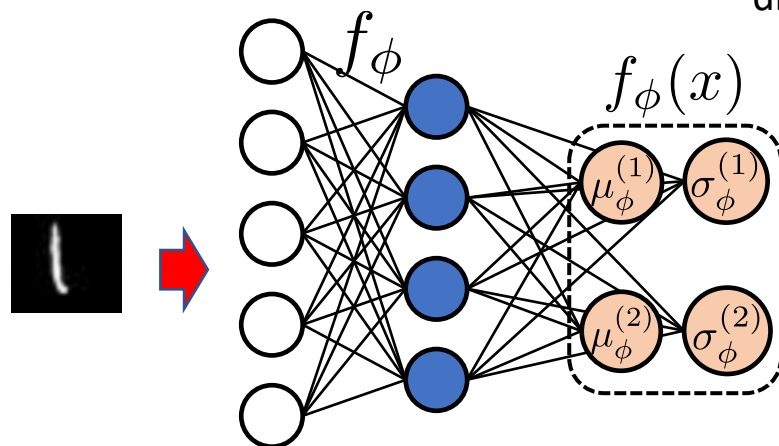
$$\{\theta', \phi'\} = \arg \min_{\theta, \phi} \mathbb{E}_{x \sim p_{data}} [\mathcal{L}_{VAE}] = \arg \min_{\theta, \phi} \mathbb{E}_{x \sim p_{data}} [\mathcal{L}_{rec}] + \lambda \mathbb{E}_{x \sim p_{data}} [\mathcal{L}_{reg}]$$

The regularizer is **ONLY** a function of  $\phi$ ,  
the parameters of the encoder!

It does **NOT** train the decoder!

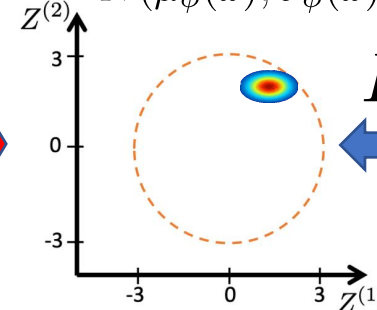
$$D_{KL} [p_{\phi}(z|x) || N(0, I)]$$

$$= \frac{1}{2} \sum_{j=1}^v \left[ (\mu_{\phi}^{(j)}(x))^2 + (\sigma_{\phi}^{(j)}(x))^2 - 2 \log \sigma_{\phi}^{(j)}(x) - 1 \right]$$

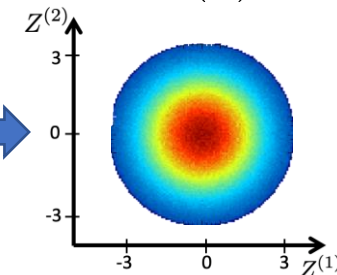


Predicted "posterior"  
distribution of  $z$  for given  $x$ ,

$$p_{\phi}(z|x) = N(\mu_{\phi}(x), \sigma_{\phi}(x)^2)$$



A "prior" distribution of  
codes,  $p(z) = N(0, I)$



$D_{KL}$

# Effect of Regularizer on means and standard deviations:

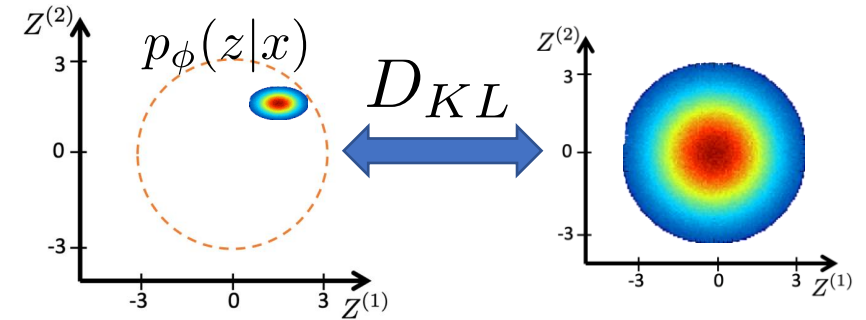
$$\begin{aligned}\phi' &= \arg \min_{\phi} \mathcal{L}_{reg}(x) \Rightarrow \\ &= \arg \min_{\phi} D_{KL} [p_{\phi}(z|x) || N(0, I)] \Rightarrow \\ &= \arg \min_{\phi} \frac{1}{2} \sum_{j=1}^v \left[ \underbrace{(\mu_{\phi}^{(j)}(x))^2}_{\text{mean}^2} + (\sigma_{\phi}^{(j)}(x))^2 - 2 \log \sigma_{\phi}^{(j)}(x) - 1 \right]\end{aligned}$$

To minimize the regularizer....

...for each of  $v$  dimensions of  $z$ ...

...mean<sup>2</sup> must be minimized, which **means ideally** the predicted means must be....????

**...must be 0!**



# Effect of Regularizer on means and standard deviations:

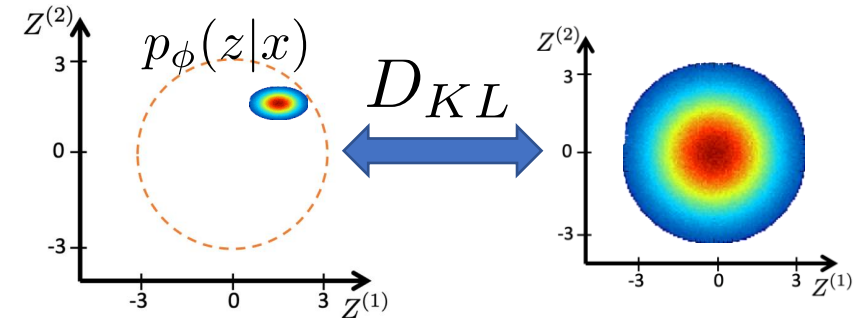
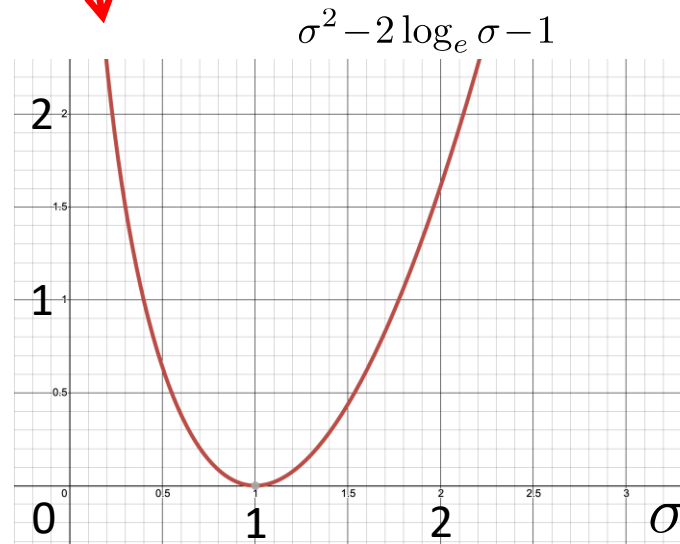
$$\begin{aligned}\phi' &= \arg \min_{\phi} \mathcal{L}_{reg}(x) \Rightarrow \\ &= \arg \min_{\phi} D_{KL} [p_{\phi}(z|x) || N(0, I)] \Rightarrow \\ &= \arg \min_{\phi} \frac{1}{2} \sum_{j=1}^v \left[ \underbrace{(\mu_{\phi}^{(j)}(x))^2}_{\text{mean}^2} + \underbrace{(\sigma_{\phi}^{(j)}(x))^2 - 2 \log \sigma_{\phi}^{(j)}(x) - 1}_{\sigma^2 - 2 \log_e \sigma - 1} \right]\end{aligned}$$

To **minimize** the regularizer....

...for each of  $v$  dimensions of  $z$ ...

...mean<sup>2</sup> must be minimized, which **means ideally** the predicted means must be....????

...must be 0!

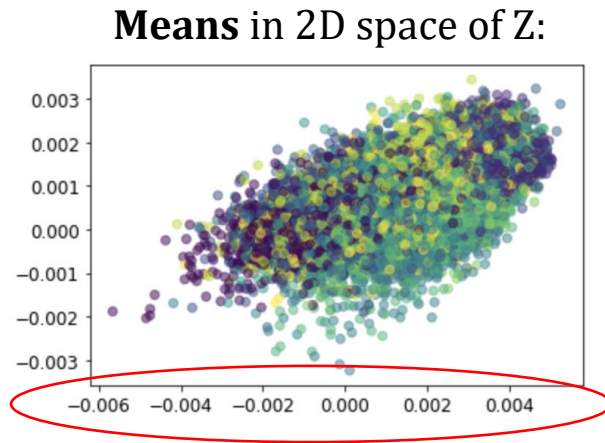


...the predicted standard deviation must **ideally** be ????.

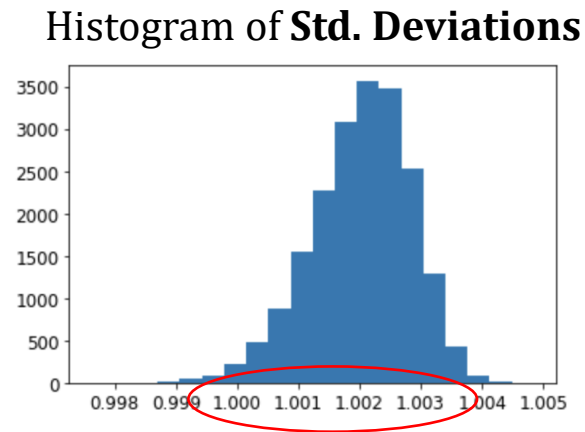
... must be =1 !

# Effect of Regularizer:

If we would train the VAE to only minimize the Regularizer:

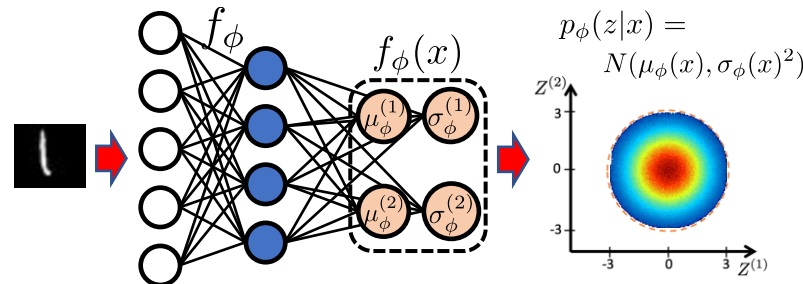


For each  $x$ , predicted **mean tends to 0**.



For each  $x$ , predicted **std. deviation tends to 1**.

For every sample, the predicted posterior  $p(z|x)$  takes the shape of the prior  $p(z)$



Useless by itself!  
But useful if combined with Reconstruction..!

# Combination of the loss terms:

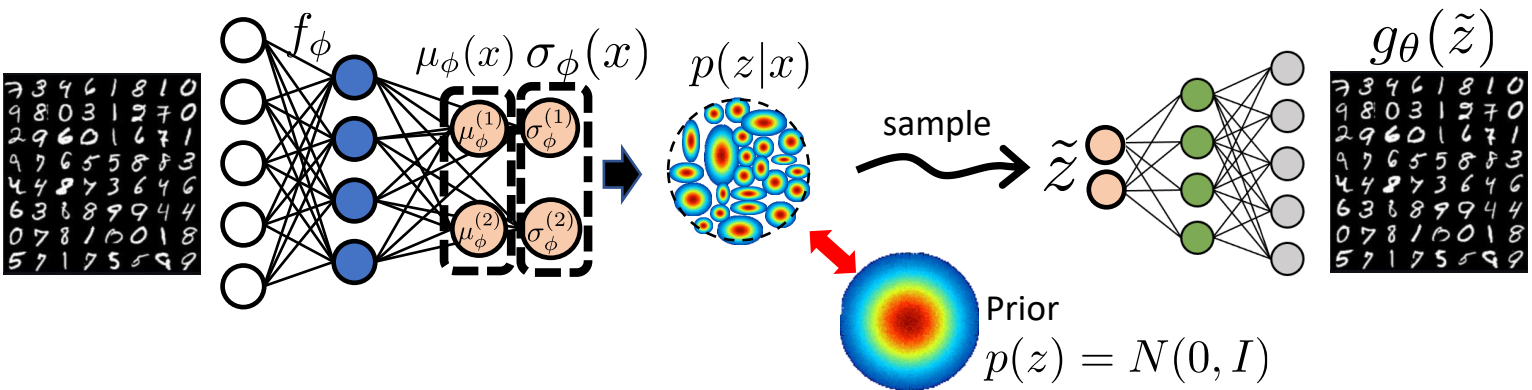
$$\{\theta', \phi'\} = \underset{\theta, \phi}{arg\min} \mathbb{E}_{x \sim p_{data}} [\mathcal{L}_{rec}] + \lambda \mathbb{E}_{x \sim p_{data}} [\mathcal{L}_{reg}]$$

Means that “cluster”  
the samples

Small Std.Devs. that  
reduce overlap

Means around 0

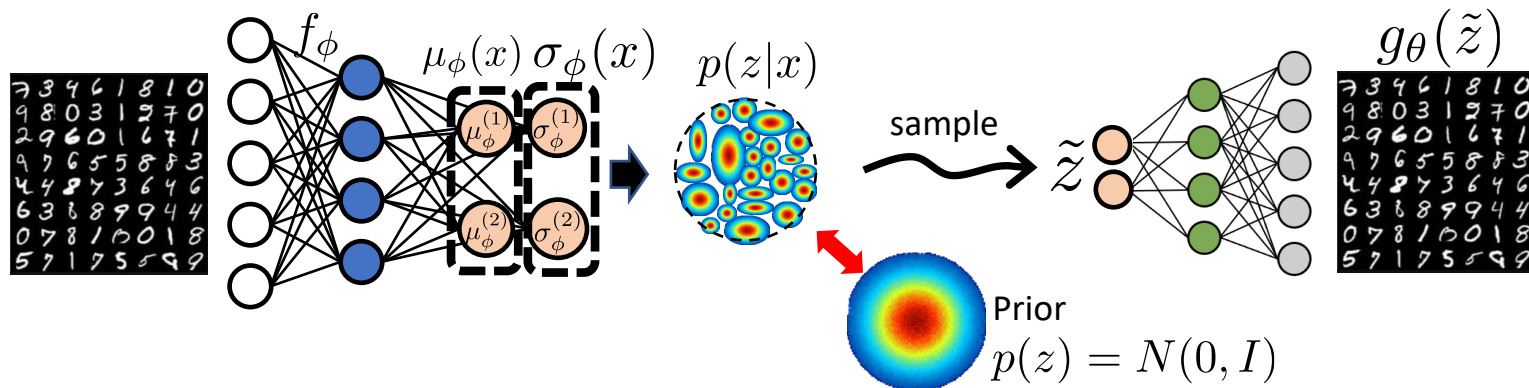
Std.Devs that cover  
the prior



# Combination of the loss terms:

$$\{\theta', \phi'\} = \arg \min_{\theta, \phi} \mathbb{E}_{x \sim p_{data}} [\mathcal{L}_{rec}] + \lambda \mathbb{E}_{x \sim p_{data}} [\mathcal{L}_{reg}]$$

Means that “cluster” the samples      Small Std.Devs. that reduce overlap      Means around 0      Std.Devs that cover the prior



Under **\*ideal\*** optimization: **and appropriate choice of  $\lambda$  (via cross-val observing rec loss, means and stds of  $z$ )**

This **combination** leads that **all together** (not each individually) the **posteriors  $p(z|x)$**  of the whole training database **cover the “prior”  $N(0, I)$** .

Simultaneously, encoder arranges posteriors of codes such that it **facilitates reconstruction**.

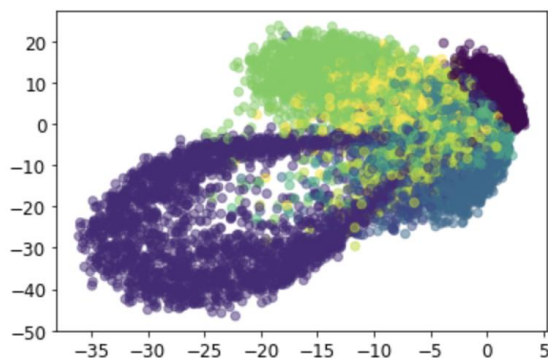
\* The explanation why this happens is in the probabilistic derivation of the VAE (further reading/optional).

# Combination of the loss terms:

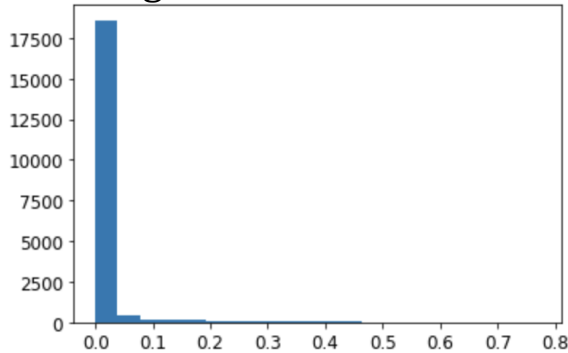
Example **in practice**, with stochastic optimization and non-optimal training parameters:  
(from our Tutorial - VAEs on MNIST)

**Reconstruction only**

**Means** in 2D space of Z:

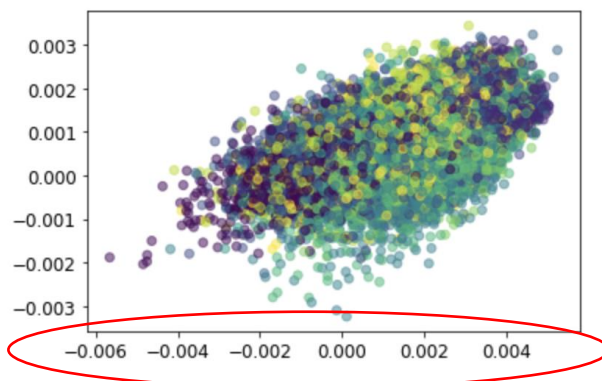


**Histogram of Std. Deviations**

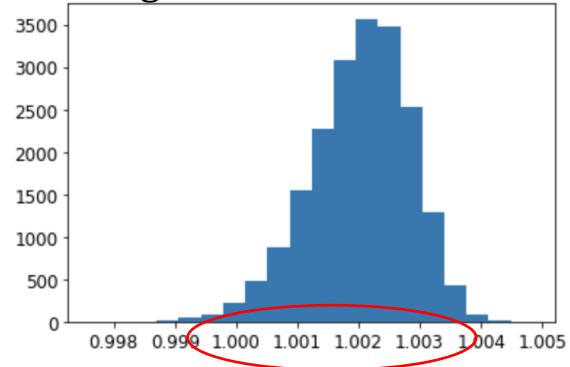


**Regularizer only**

**Means** in 2D space of Z:

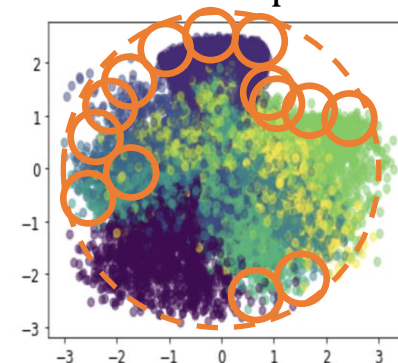


**Histogram of Std. Deviations**

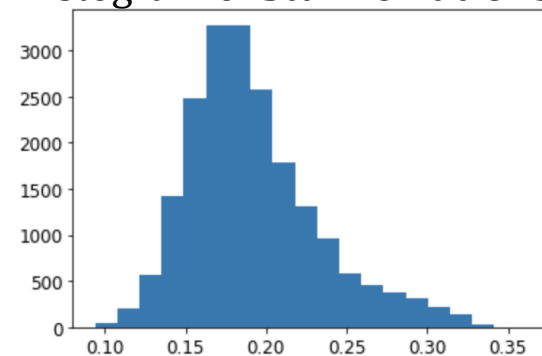


**Combination**

**Means** in 2D space of Z:

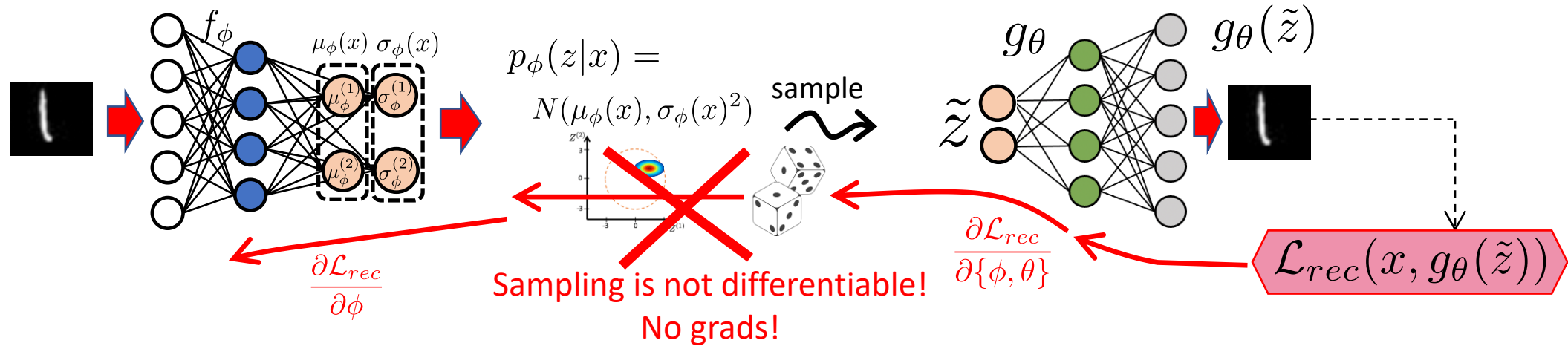


**Histogram of Std. Deviations**





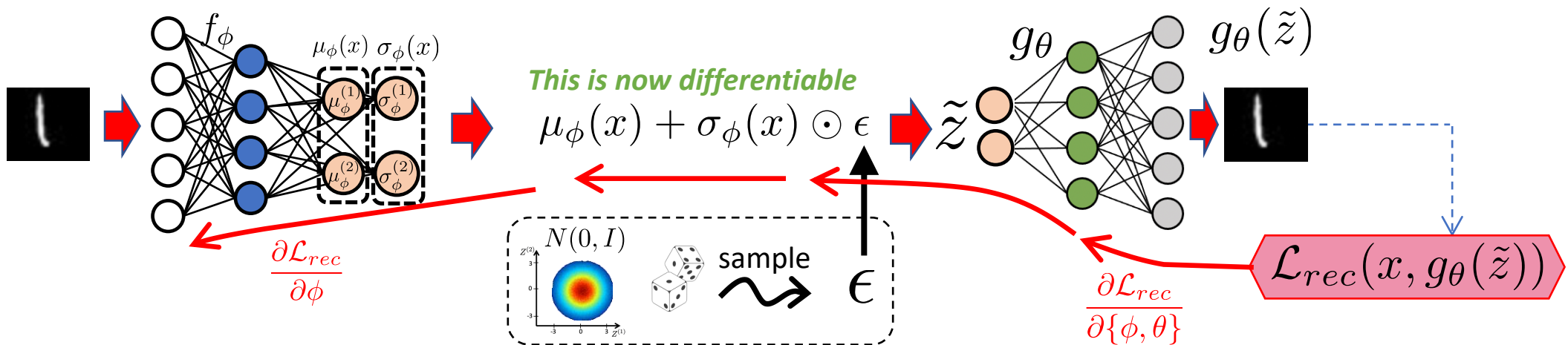
# 4. Re-parameterization trick



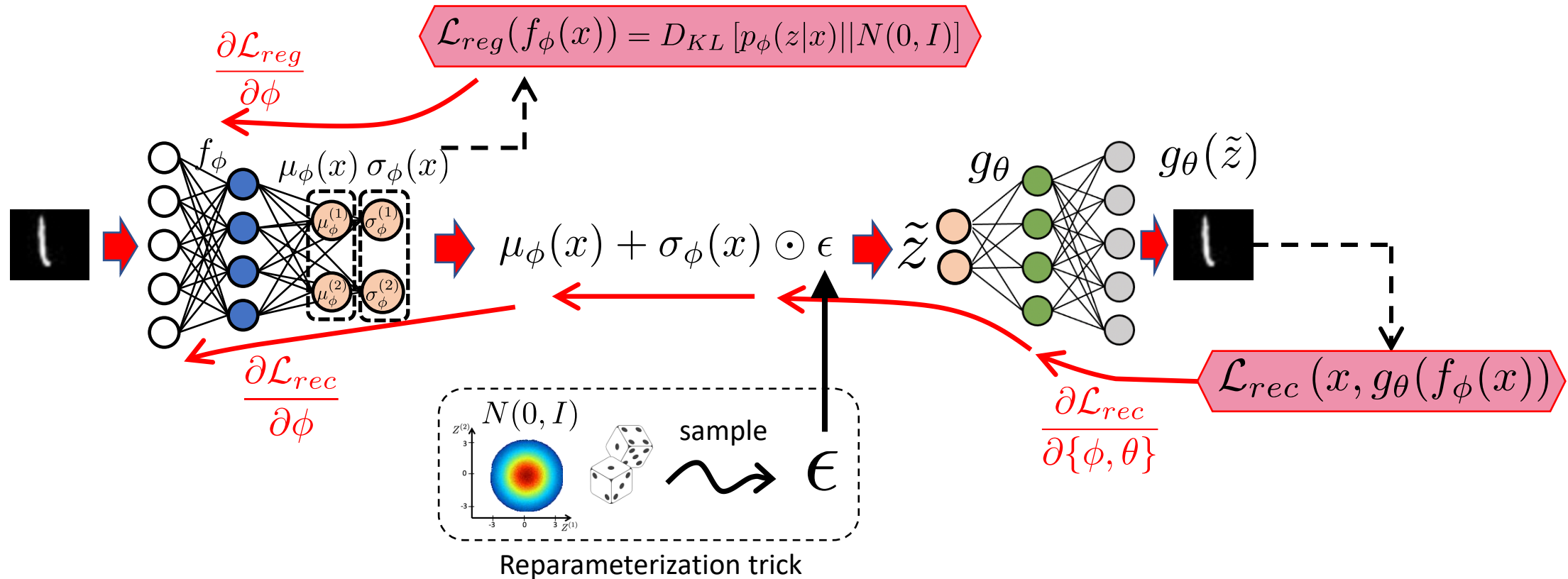
## Solution: The Re-parameterization trick

For  $p_\phi(z|x)$  a Gaussian (as in VAEs):

$$\tilde{z} \sim N(\mu_\phi(x), \sigma_\phi(x)^2) \Leftrightarrow \tilde{z} = \mu_\phi(x) + \sigma_\phi(x) \odot \epsilon, \text{ with } \epsilon \sim N(0, I)$$



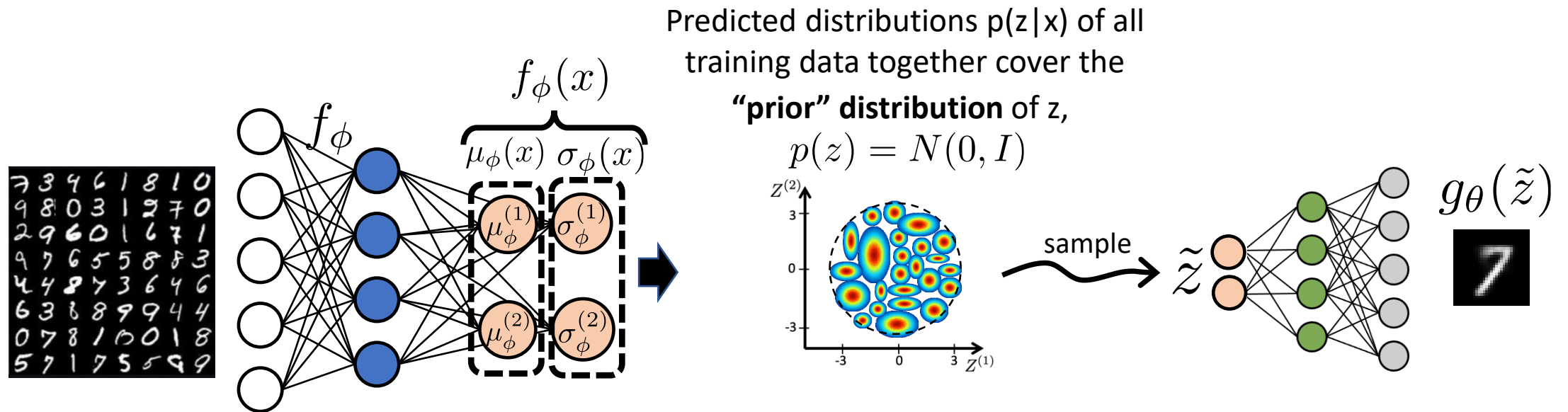
# Bringing everything together: Training



Kingma & Welling, Auto-encoding variational bayes, ICLR 2014

Rezende et al, Stochastic Backpropagation and Approximate Inference in Deep Generative Models, ICML 2014

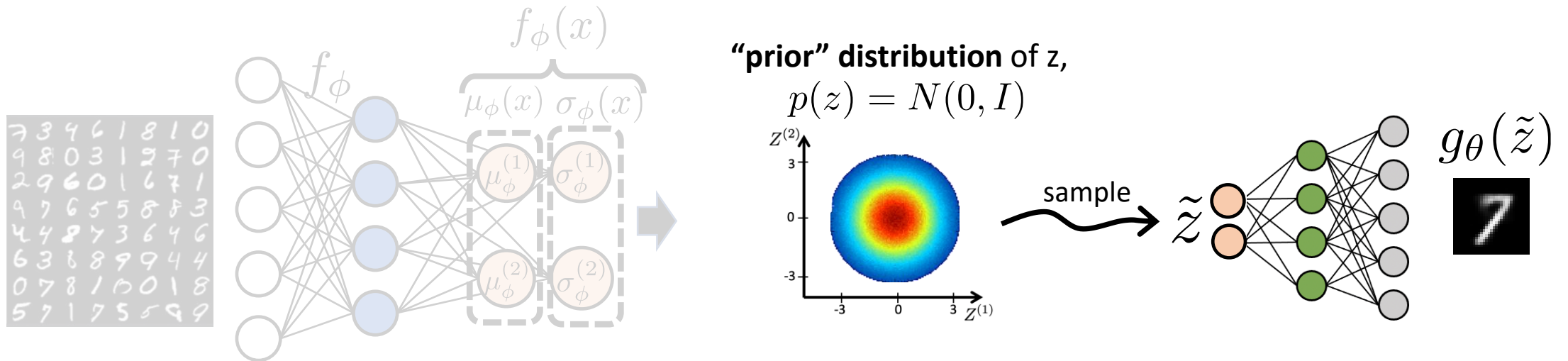
# Bringing everything together: Training



Kingma & Welling, Auto-encoding variational bayes, ICLR 2014

Rezende et al, Stochastic Backpropagation and Approximate Inference in Deep Generative Models, ICML 2014

# Bringing everything together: Generating new data



Kingma & Welling, Auto-encoding variational bayes, ICLR 2014

Rezende et al, Stochastic Backpropagation and Approximate Inference in Deep Generative Models, ICML 2014

### **In this video:**

- What is the architecture of the VAE
- What is the training loss: Reconstruction + Regularizer
- What is the effect of minimizing each loss term
- The Re-Parameterization trick for the sampling

### **Next video:**

Application of VAE for...

- Generation of new data points
- Modifying data via interpolating between 2 inputs
- Modifying specific feature of an input
- Compression and Reconstruction

Thank you very much