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

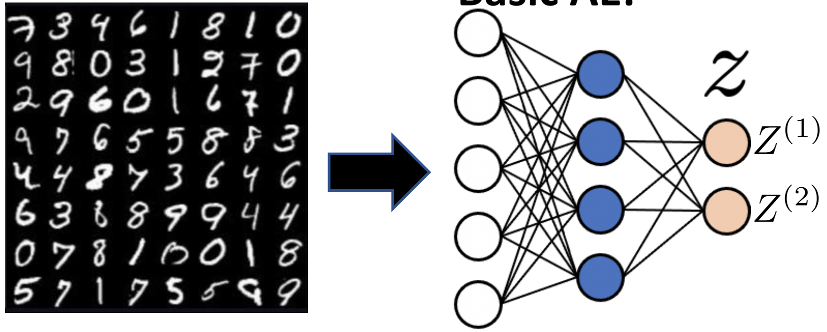
- Architecture of a Variational Auto-Encoder (VAE)
- How to train VAEs
- How do different terms of training loss influence what VAE learns
- How does a VAE relate to the basic AE

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

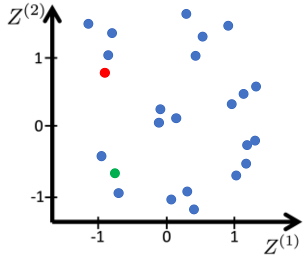
Basic AE:

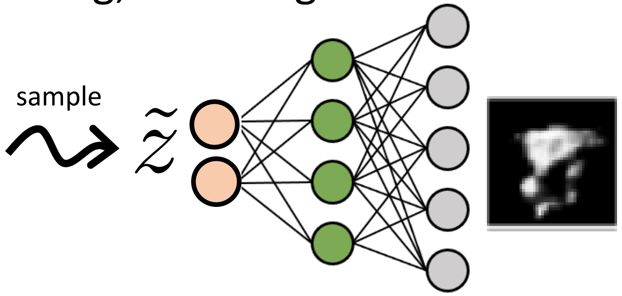


During training, learn to map each input to a point.

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

Generation: If z is sampled from space left empty during training, decoding will fail

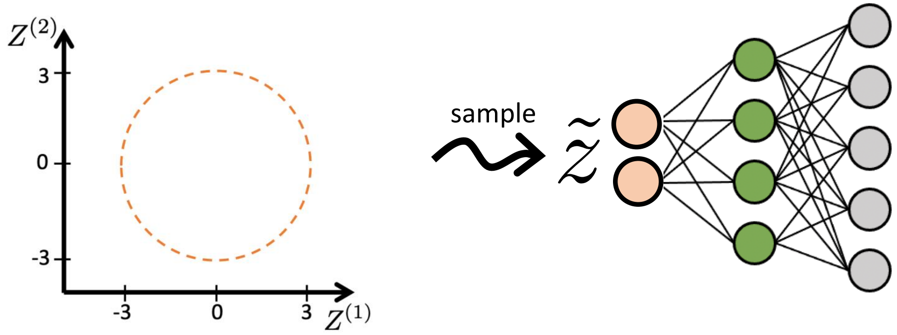




From basic deterministic AE to Variational AE

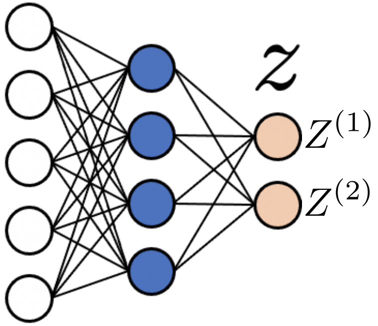
VAE (informally)**:**

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



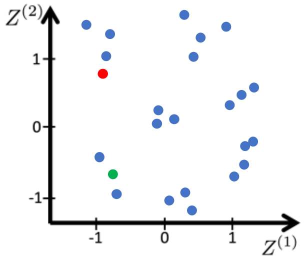
Generation: We can then sample from anywhere in the area. Decoding should then work.

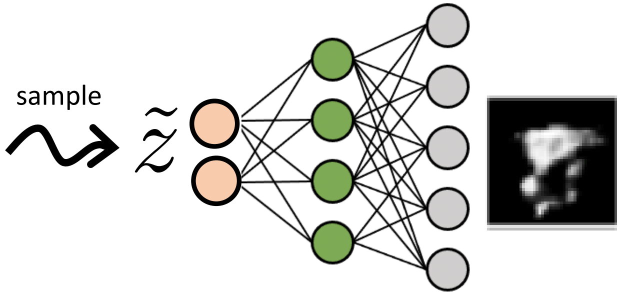
Basic AE:



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

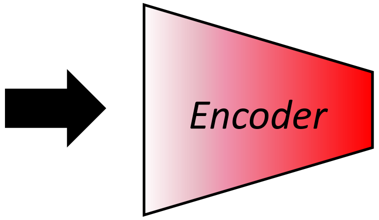
Encoder leaves space between mapped points, to enable reconstruction without confusion





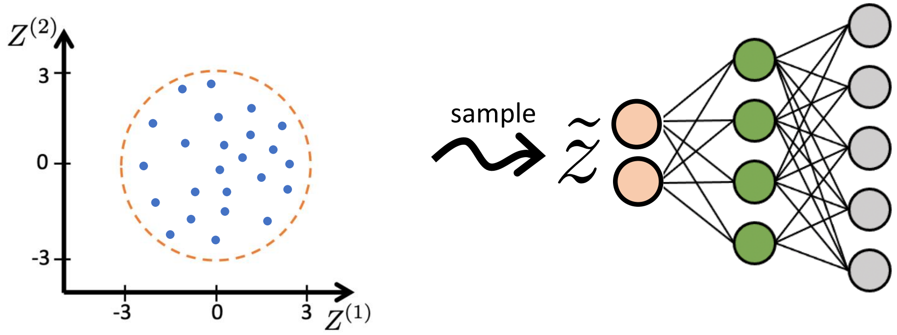
From basic deterministic AE to Variational AE

VAE (informally)**:**



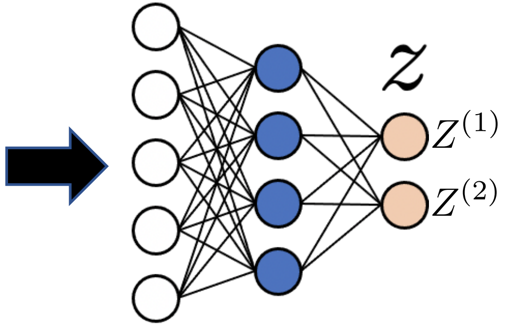
Instead of using Encoder that maps each input to *a single point* like in basic AE…

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



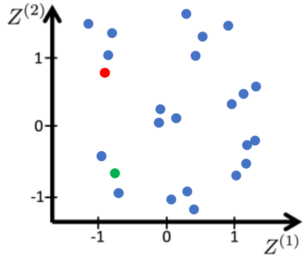
Generation: We can then sample from anywhere in the area. Decoding should then work.

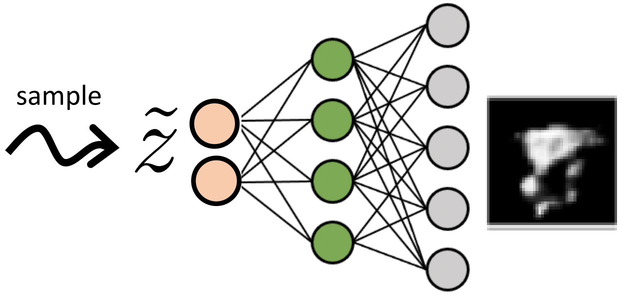
Basic AE:



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

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

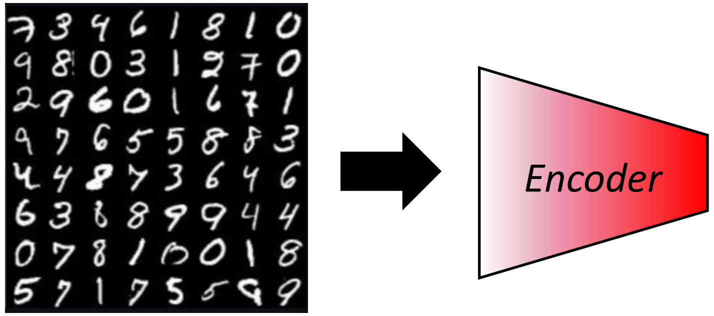




From basic deterministic AE to Variational AE

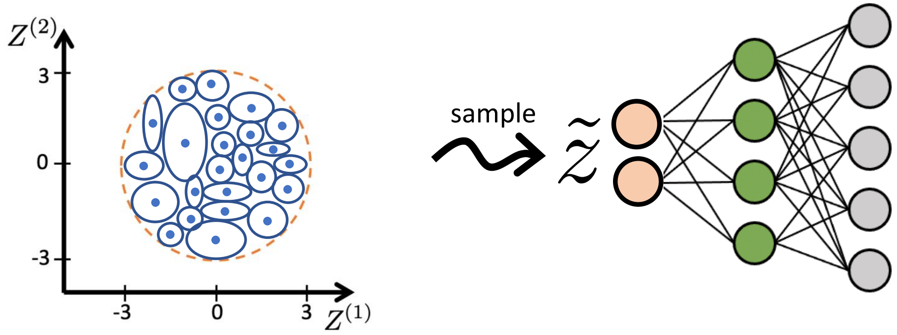
VAE (informally)**:**

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

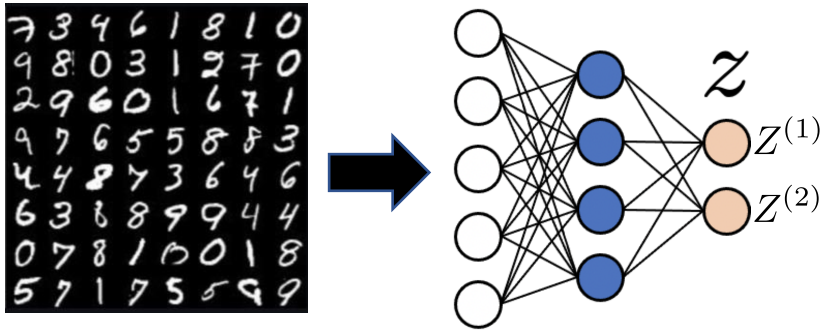


Make encoder that maps each input to an ‘area’

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

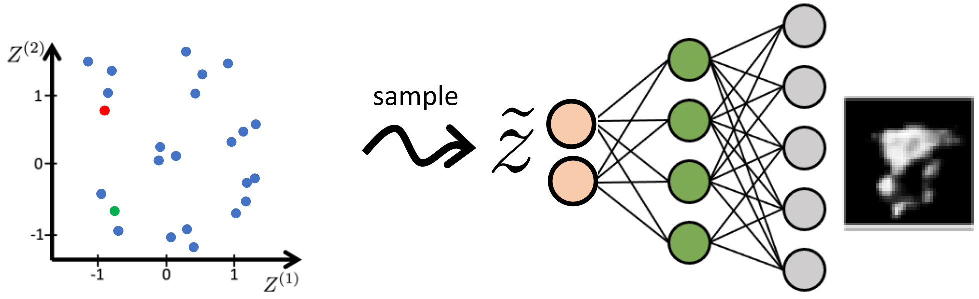


Generation: We can then sample from anywhere in the area. Decoding should then work.



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

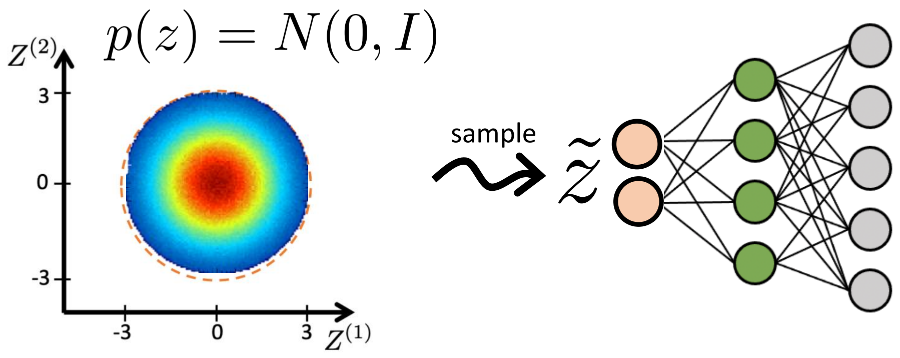
Encoder leaves space between mapped points, to enable reconstruction without confusion



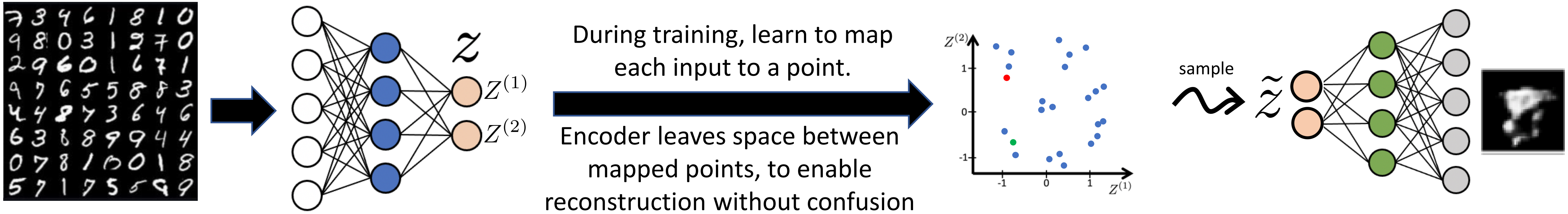
From basic deterministic AE to Variational AE

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!



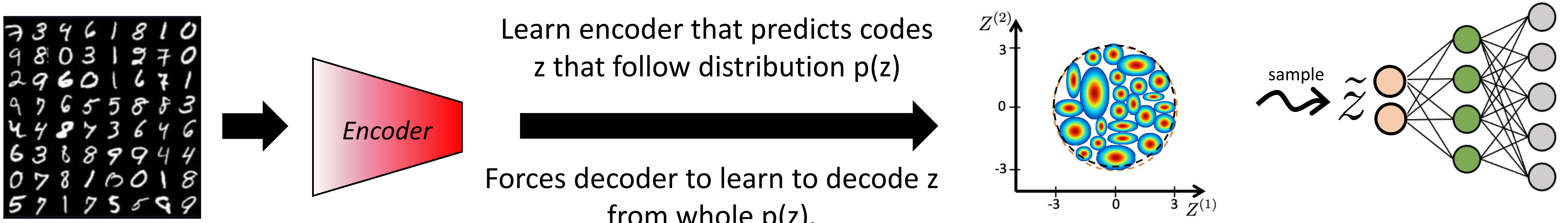
During training, learn to map each input to a point.

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

From basic deterministic AE to Variational AE

VAE (more accurately)**:**

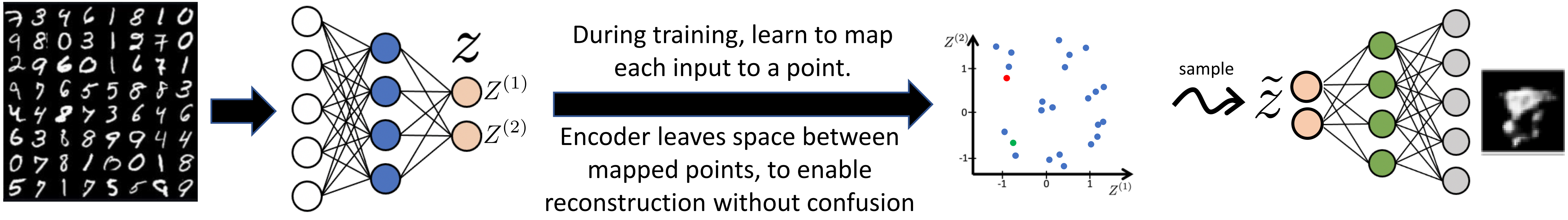
Training: Enforce codes z to follow a **“prior” distribution p(z)**



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.

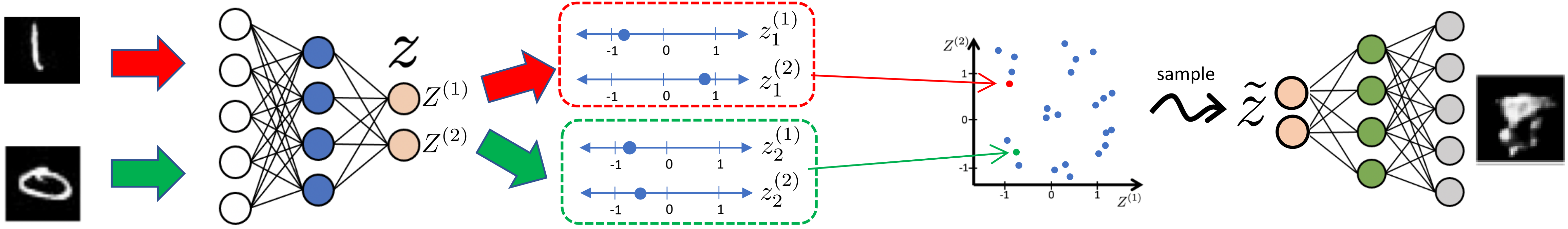
Encoder leaves space between mapped points, to enable reconstruction without confusion

1. Probabilistic Encoder

Basic AE:

Each input x is mapped to a point z

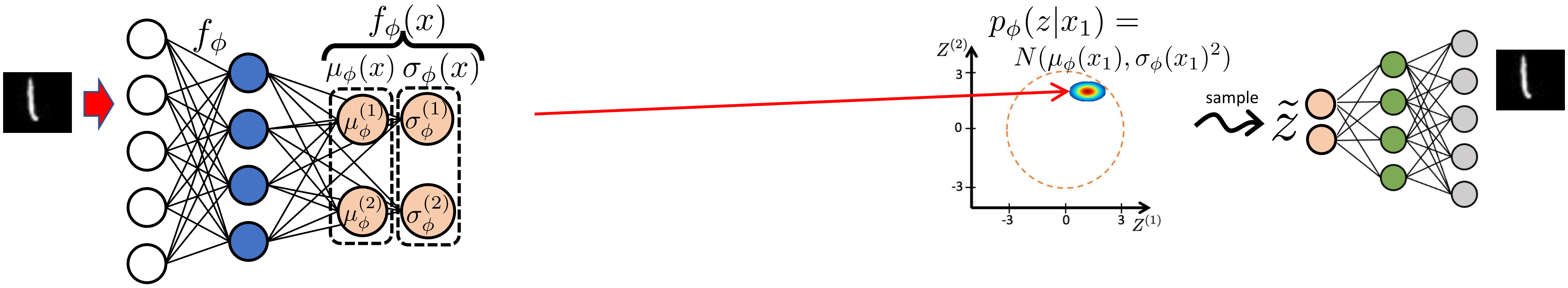
The whole model is deterministic.



1. Probabilistic Encoder

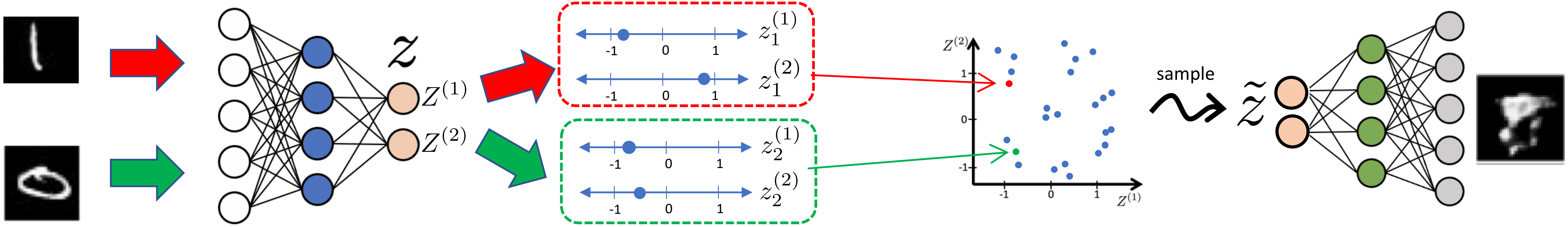
“**posterior**” distribution of z,
a.k.a. “**conditional**” distribution of z given x

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)


f_ϕ
$f_\phi(x)$
$\mu_\phi(x)$ $\sigma_\phi(x)$
$\mu_\phi^{(1)}$ $\sigma_\phi^{(1)}$
$\mu_\phi^{(2)}$ $\sigma_\phi^{(2)}$
$p_\phi(z|x_1) =$
$N(\mu_\phi(x_1), \sigma_\phi(x_1)^2)$
$Z^{(2)}$
3
0
-3
-3
0
3
$Z^{(1)}$
sample
$\tilde{z}$

Basic AE:

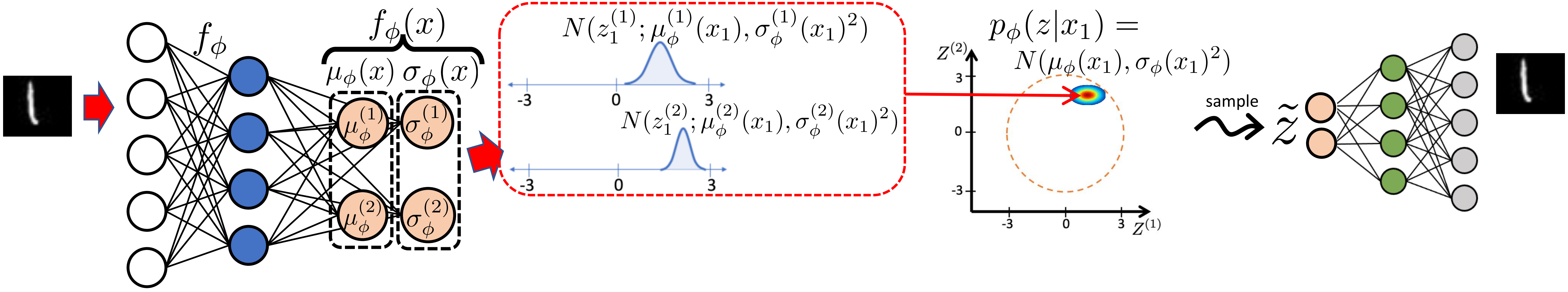
Each input x is mapped to a point z



1. Probabilistic Encoder

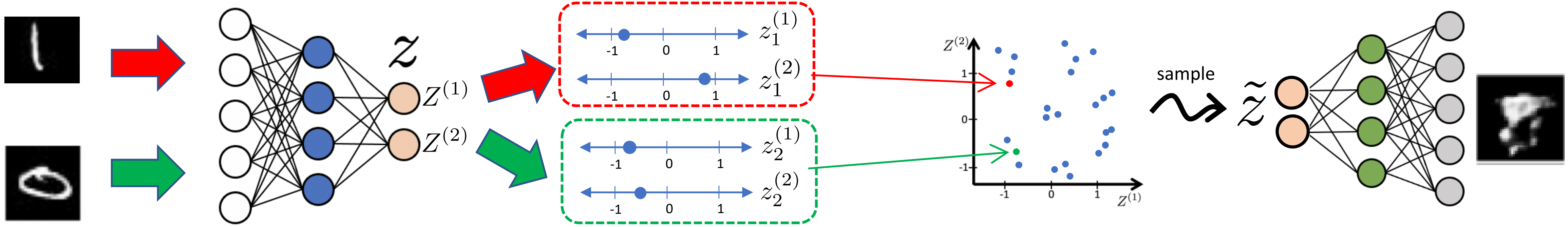
“**posterior**” distribution of z,
a.k.a. “**conditional**” distribution of z given x

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)


f_ϕ
$f_\phi(x)$
$\mu_\phi(x)$ $\sigma_\phi(x)$
$\mu_\phi^{(1)}$
$\sigma_\phi^{(1)}$
$\mu_\phi^{(2)}$
$\sigma_\phi^{(2)}$
$N(z_1^{(1)}; \mu_\phi^{(1)}(x_1), \sigma_\phi^{(1)}(x_1)^2)$
-3 0 3
$N(z_1^{(2)}; \mu_\phi^{(2)}(x_1), \sigma_\phi^{(2)}(x_1)^2)$
-3 0 3
$p_\phi(z|x_1) =$
$N(\mu_\phi(x_1), \sigma_\phi(x_1)^2)$
$Z^{(2)}$
3
0
-3
-3 0 3
$Z^{(1)}$
sample
$\tilde{z}$

Basic AE:

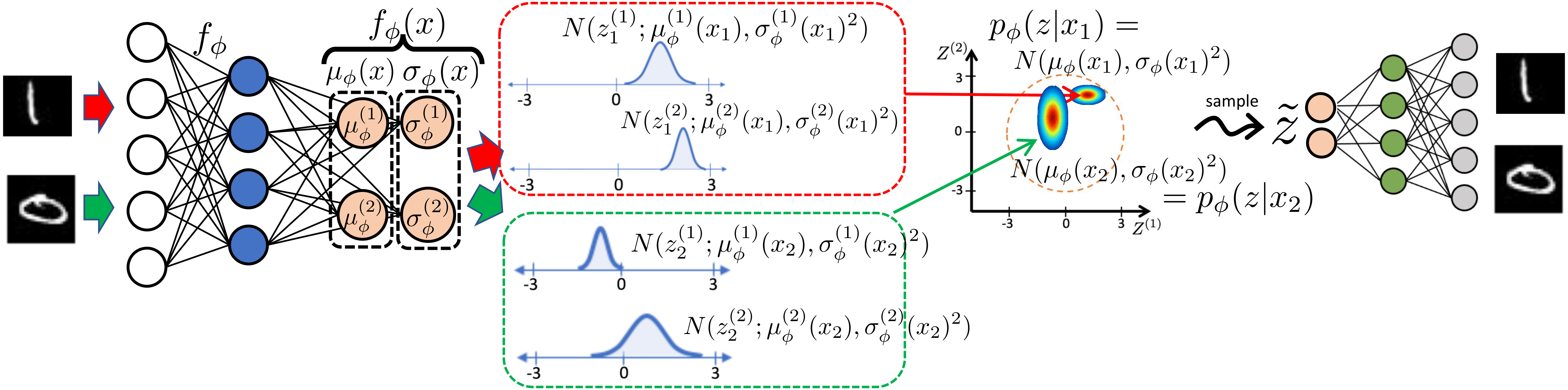
Each input x is mapped to a point z



1. Probabilistic Encoder

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

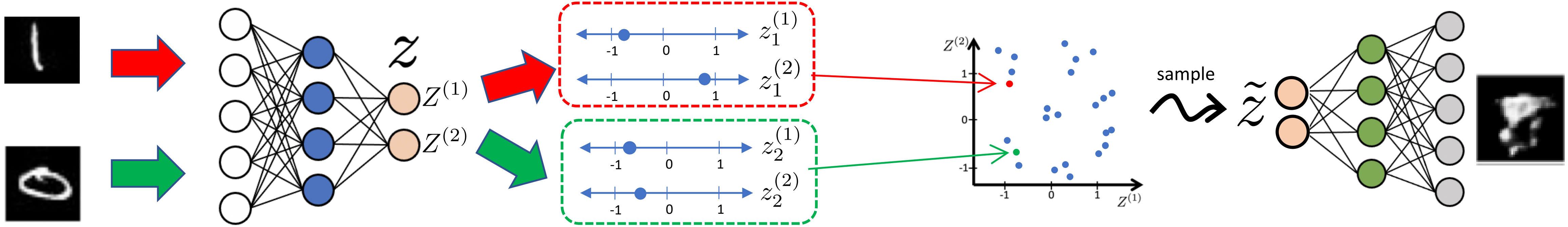
Each x is mapped to a multi-variate Normal/Gaussian distribution of z, $p_\phi(z|x) = N(\mu_\phi(x), \sigma_\phi(x)^2)$
by **predicting** a **mean** and **standard deviation** for each dimension of z. (Note: diagonal covariance matrix, i.e. each dimension is uncorrelated)



f_ϕ
$\overbrace{\mu_\phi(x)\ \sigma_\phi(x)}^{f_\phi(x)}$
$\mu_\phi^{(1)}$
$\sigma_\phi^{(1)}$
$\mu_\phi^{(2)}$
$\sigma_\phi^{(2)}$
$N(z_1^{(1)}; \mu_\phi^{(1)}(x_1), \sigma_\phi^{(1)}(x_1)^2)$
-3 0 3
$N(z_1^{(2)}; \mu_\phi^{(2)}(x_1), \sigma_\phi^{(2)}(x_1)^2)$
-3 0 3
$N(z_2^{(1)}; \mu_\phi^{(1)}(x_2), \sigma_\phi^{(1)}(x_2)^2)$
-3 0 3
$N(z_2^{(2)}; \mu_\phi^{(2)}(x_2), \sigma_\phi^{(2)}(x_2)^2)$
-3 0 3
$p_\phi(z|x_1) =$
$N(\mu_\phi(x_1), \sigma_\phi(x_1)^2)$
$Z^{(2)}$
3
0
-3
-3 0 3
$Z^{(1)}$
$N(\mu_\phi(x_2), \sigma_\phi(x_2)^2)$
$= p_\phi(z|x_2)$
sample
$\tilde{z}$

Basic AE:

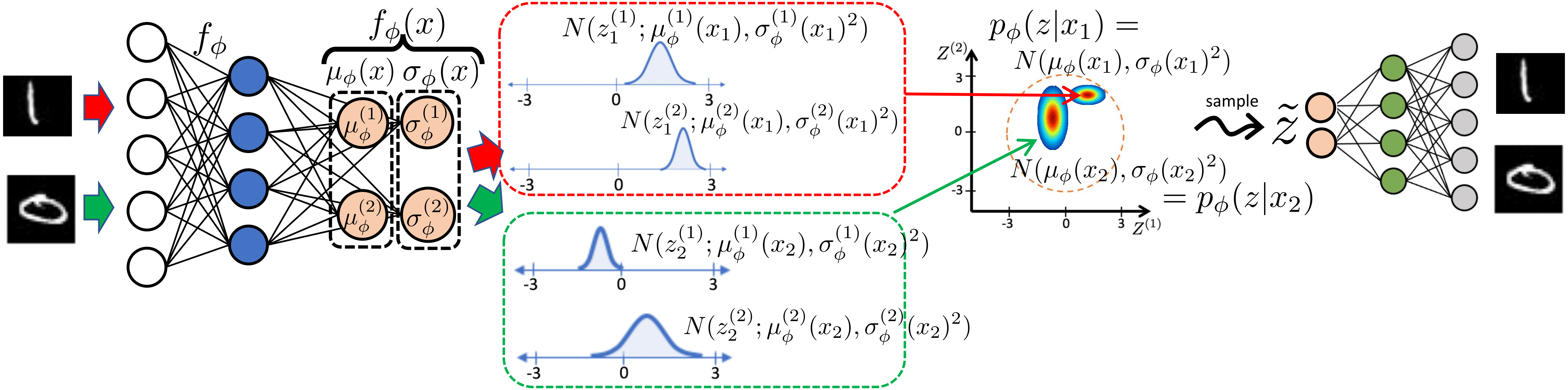
Each input x is mapped to a point z



1. Probabilistic Encoder

"**posterior**" distribution of z,
a.k.a. "**conditional**" distribution of z given x

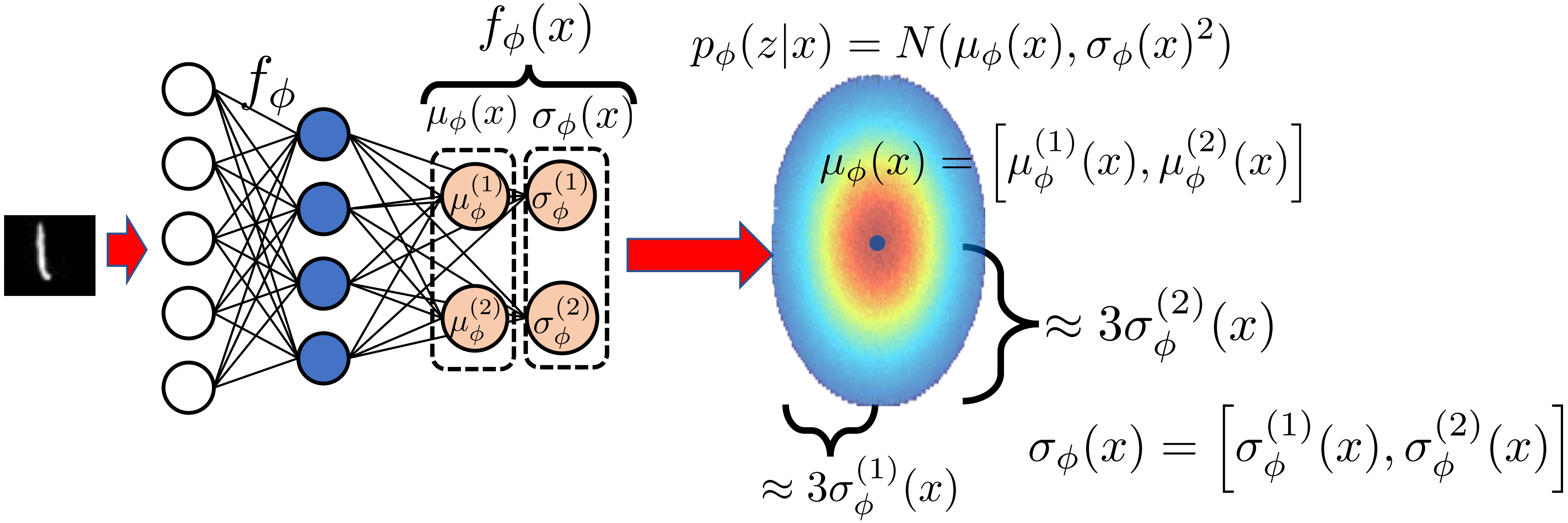
Each x is mapped to a multi-variate Normal/Gaussian distribution of z, $p_\phi(z|x) = N(\mu_\phi(x), \sigma_\phi(x)^2)$
by **predicting** a **mean** and **standard deviation** for each dimension of z. (Note: diagonal covariance matrix, i.e. each dimension is uncorrelated)



f_ϕ
$f_\phi(x)$
$\mu_\phi(x)$ $\sigma_\phi(x)$
$\mu_\phi^{(1)}$
$\sigma_\phi^{(1)}$
$\mu_\phi^{(2)}$
$\sigma_\phi^{(2)}$
$N(z_1^{(1)}; \mu_\phi^{(1)}(x_1), \sigma_\phi^{(1)}(x_1)^2)$
-3 0 3
$N(z_1^{(2)}; \mu_\phi^{(2)}(x_1), \sigma_\phi^{(2)}(x_1)^2)$
-3 0 3
$N(z_2^{(1)}; \mu_\phi^{(1)}(x_2), \sigma_\phi^{(1)}(x_2)^2)$
-3 0 3
$N(z_2^{(2)}; \mu_\phi^{(2)}(x_2), \sigma_\phi^{(2)}(x_2)^2)$
-3 0 3
$p_\phi(z|x_1) =$
$N(\mu_\phi(x_1), \sigma_\phi(x_1)^2)$
$Z^{(2)}$
3
0
-3
-3
0
3
$Z^{(1)}$
$N(\mu_\phi(x_2), \sigma_\phi(x_2)^2)$
$= p_\phi(z|x_2)$
sample
$\tilde{z}$

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

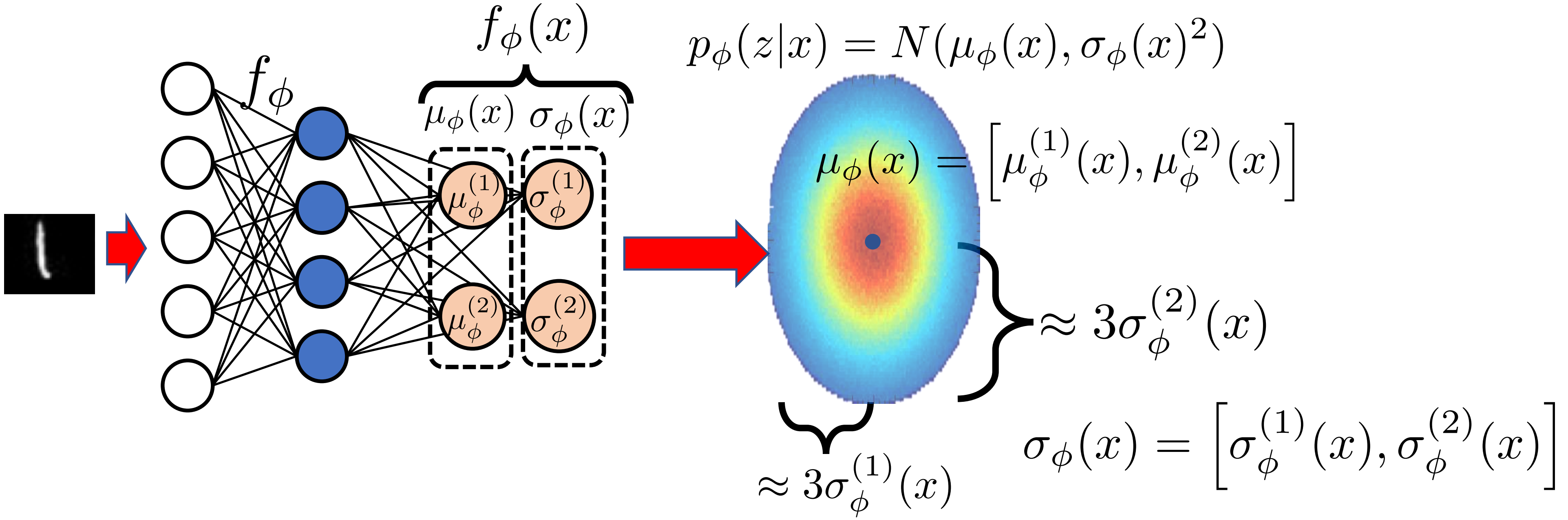


$$p_\phi(z|x) = N(\mu_\phi(x), \sigma_\phi(x)^2)$$

$$\mu_\phi(x) = \left[\mu_\phi^{(1)}(x), \mu_\phi^{(2)}(x)\right]$$

$$\sigma_\phi(x) = \left[\sigma_\phi^{(1)}(x), \sigma_\phi^{(2)}(x)\right]$$

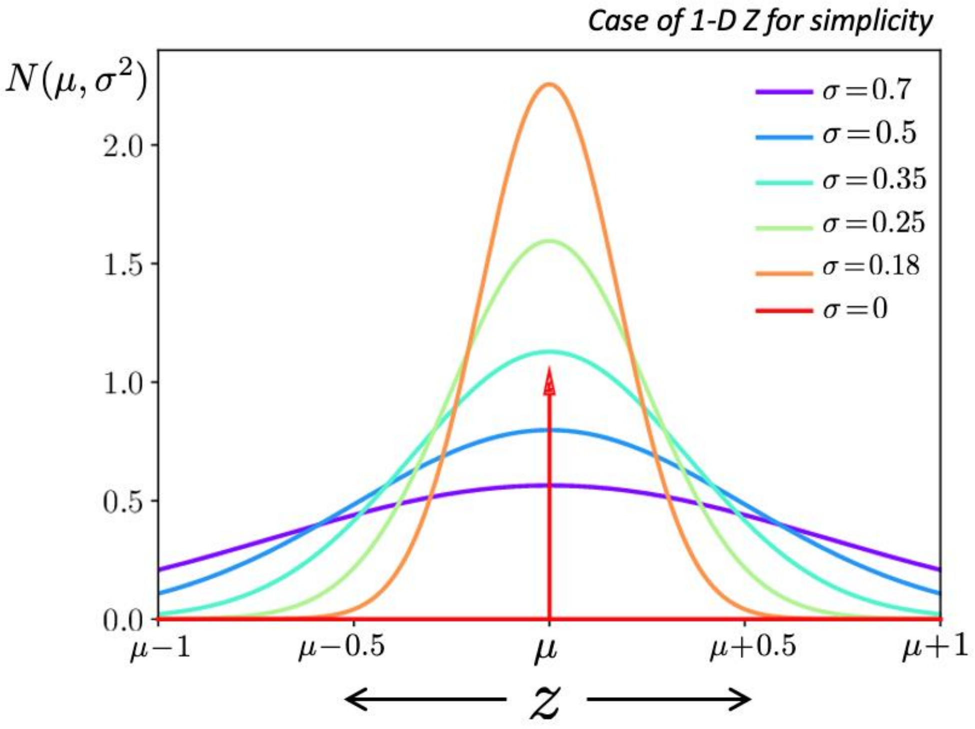
How to interpret that for each input x, the encoder predicts a probability distribution for z?



$f_\phi(x)$
f_ϕ
$\mu_\phi(x)$ $\sigma_\phi(x)$
$\mu_\phi^{(1)}$
$\sigma_\phi^{(1)}$
$\mu_\phi^{(2)}$
$\sigma_\phi^{(2)}$
$p_\phi(z|x) = N(\mu_\phi(x), \sigma_\phi(x)^2)$
$\mu_\phi(x) = \left[\mu_\phi^{(1)}(x), \mu_\phi^{(2)}(x)\right]$
$\approx 3\sigma_\phi^{(2)}(x)$
$\approx 3\sigma_\phi^{(1)}(x)$
$\sigma_\phi(x) = \left[\sigma_\phi^{(1)}(x), \sigma_\phi^{(2)}(x)\right]$

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 ϕ.

Uncertainty (low probability) in encoding

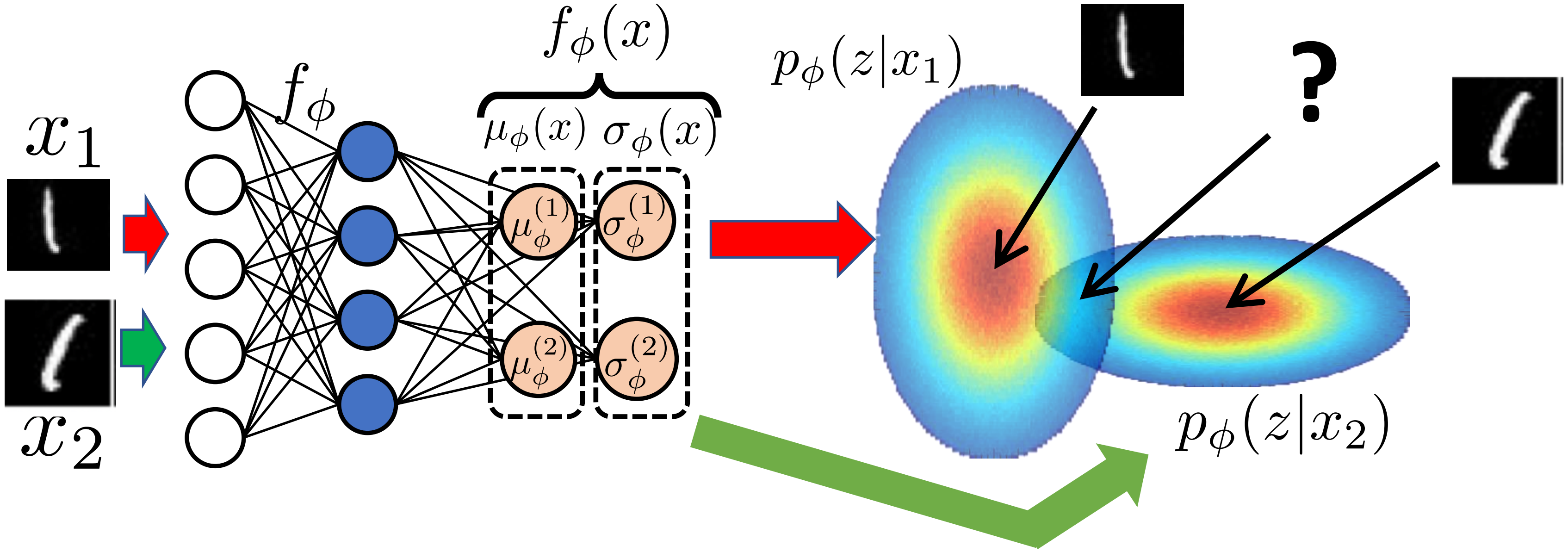


Q: If predicted std. deviation $\sigma_\phi(x) = 0$?

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.

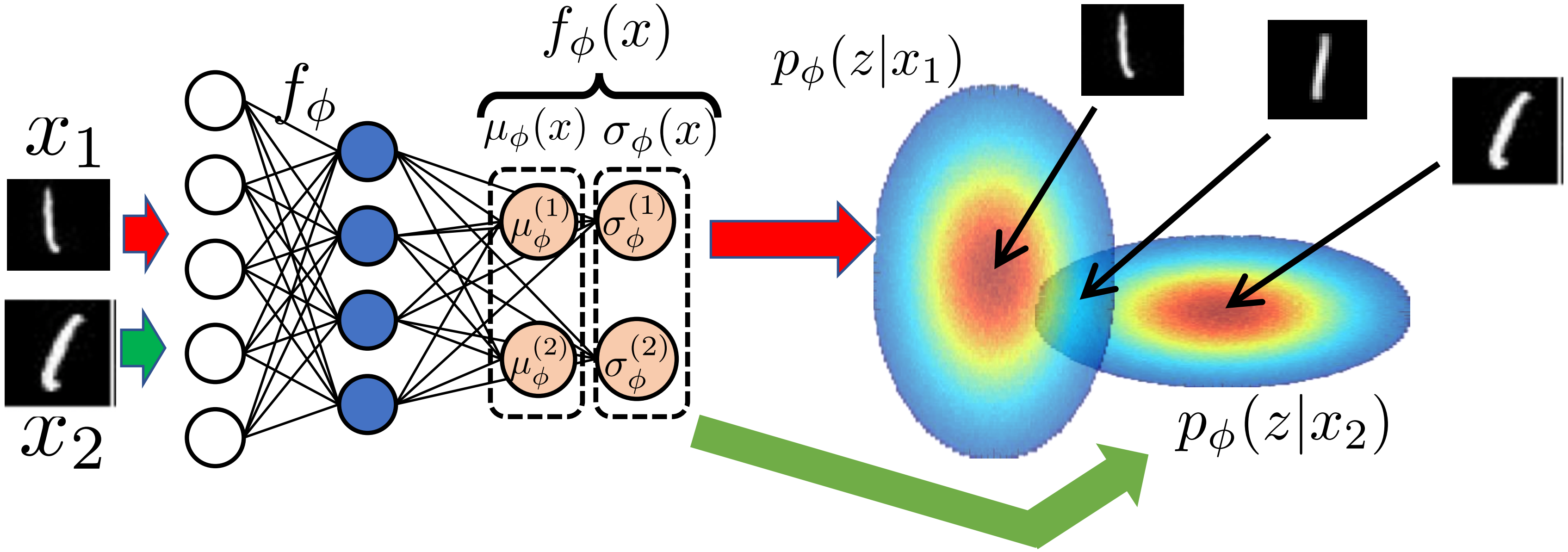
How to interpret “overlaps” of predicted distributions for z?



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?

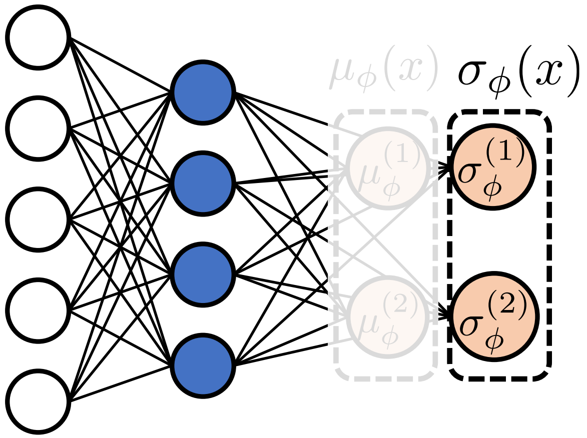


This formulation of the encoder leads to existence of “overlaps” in the encoding of different samples.
(in fact, Gaussians extend to infinity, even for very small values, so in fact, there are overlaps everywhere)

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?

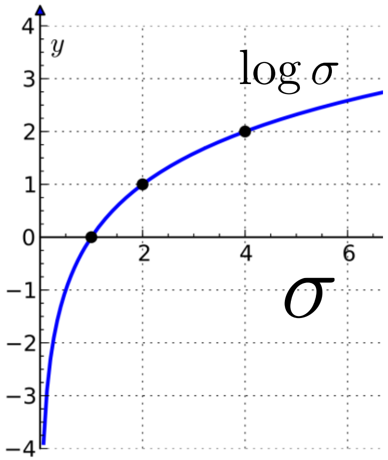
A: Appearance that would be in-between those of x_1 and x_2

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 $\log(\sigma_\phi(x))$

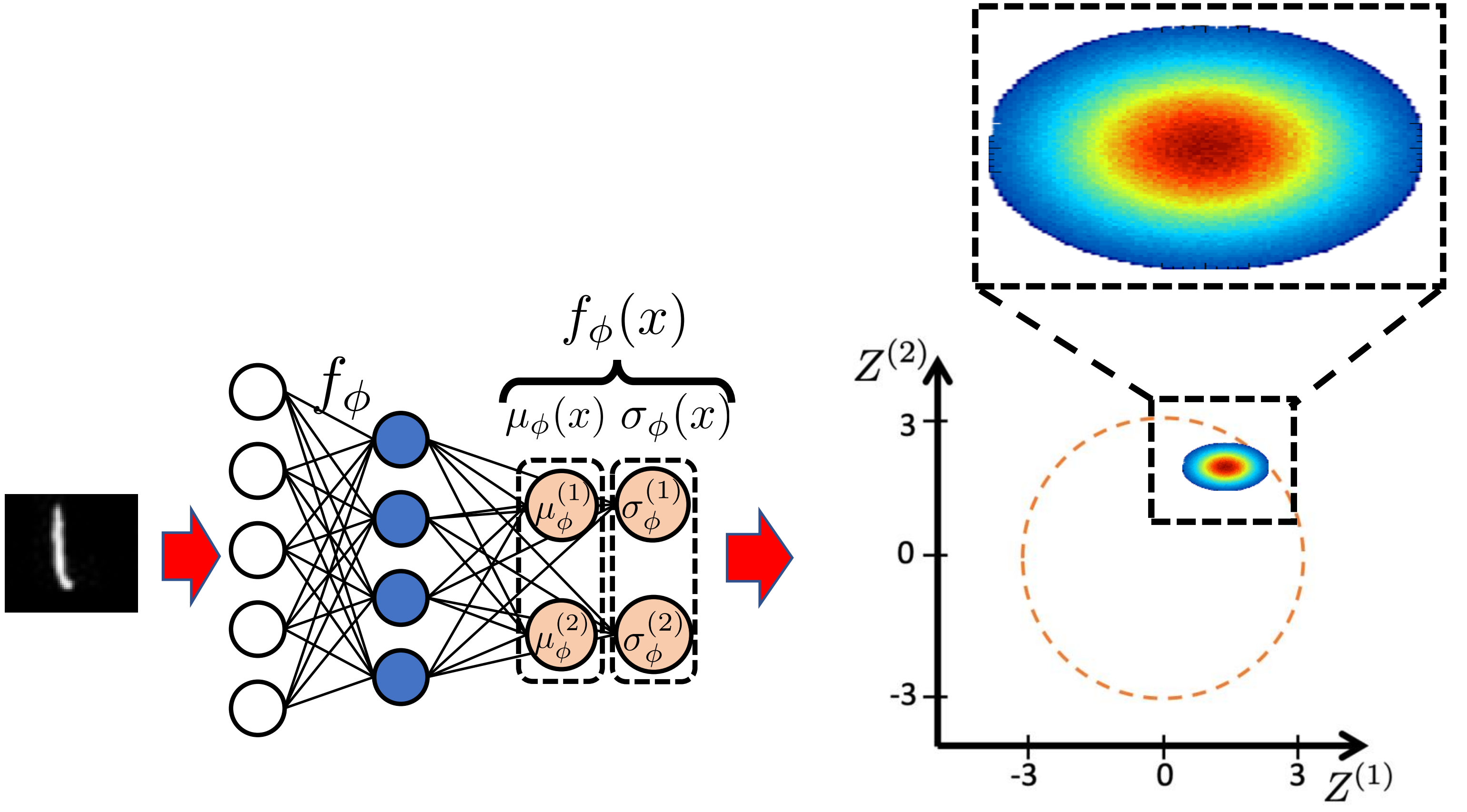
```
log_stddev = matrix_mult(act_layer1 , w_layer2)
```

- Compute sigma as: $\sigma_\phi(x) = \exp(\log(\sigma_\phi(x)))$

```
stddev = exp(log_stddev)
```

2. Stochastic Decoder (during training)

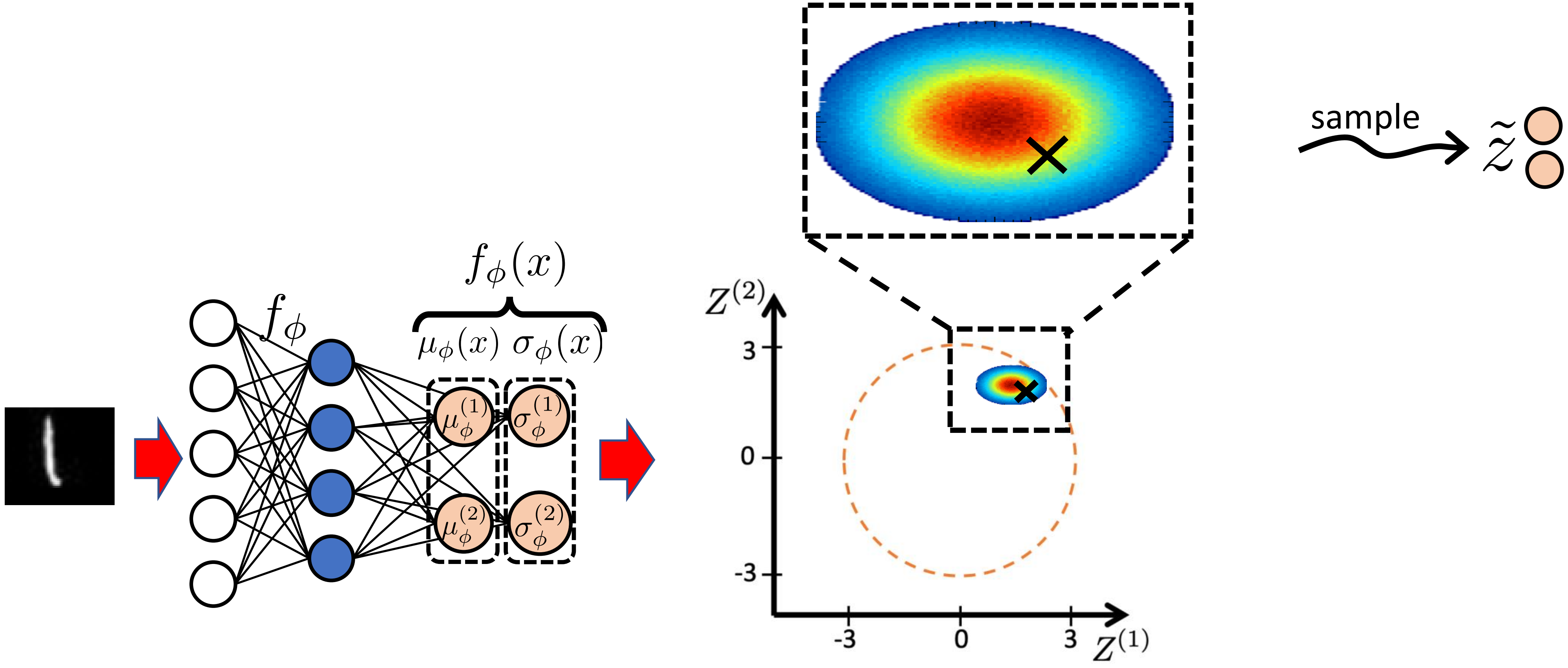
$$p_\phi(z|x) = N(\mu_\phi(x), \sigma_\phi(x)^2)$$



2. Stochastic Decoder (during training)

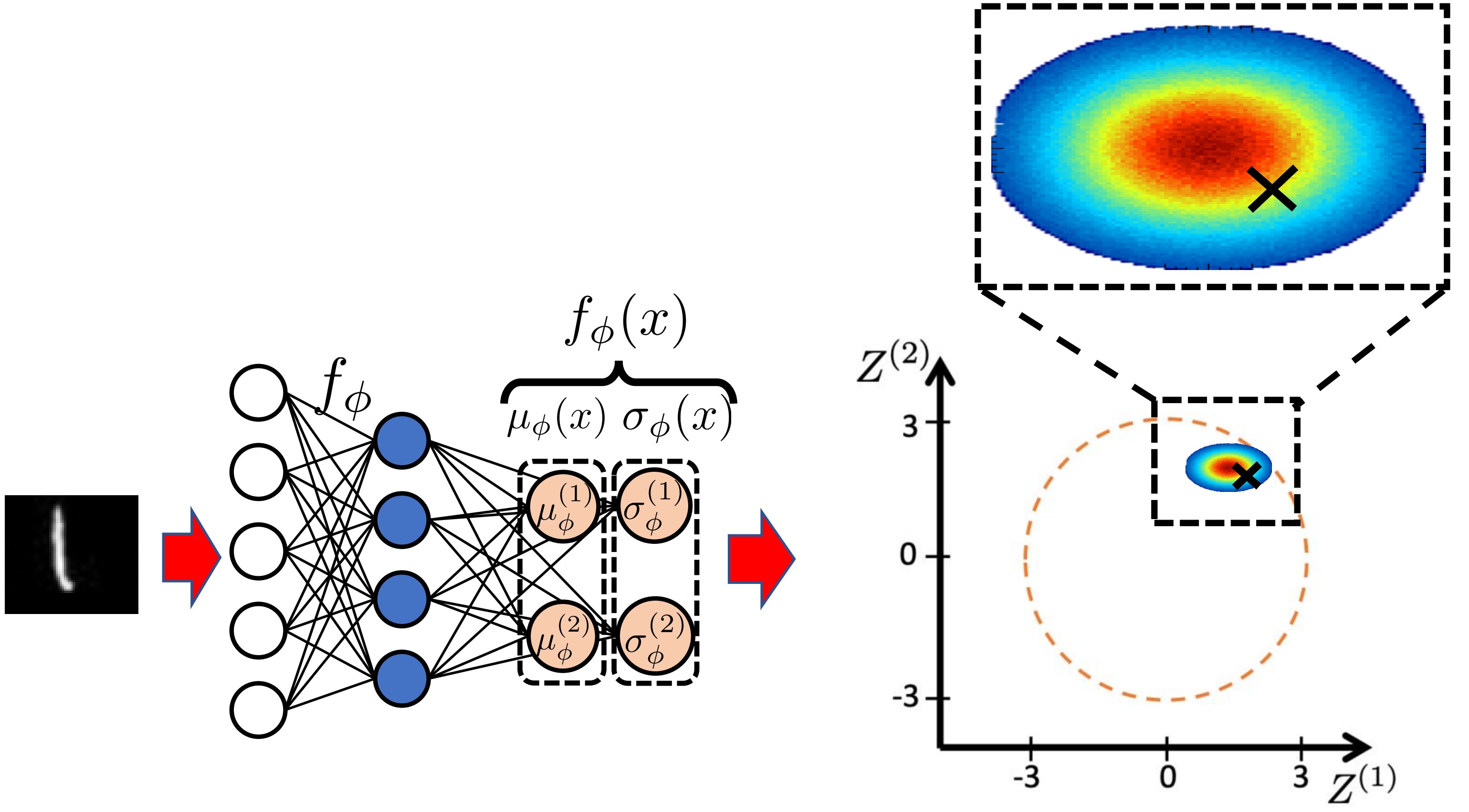
Step 1. Sample a code $\tilde{z}$ from the predicted Gaussian $p_\phi(z|x)$

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

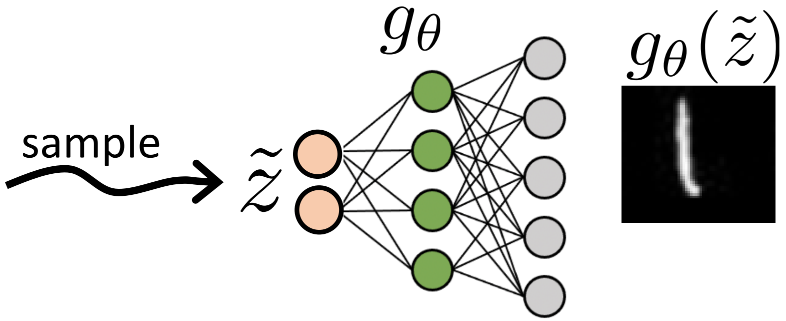


2. Stochastic Decoder (during training)

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



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

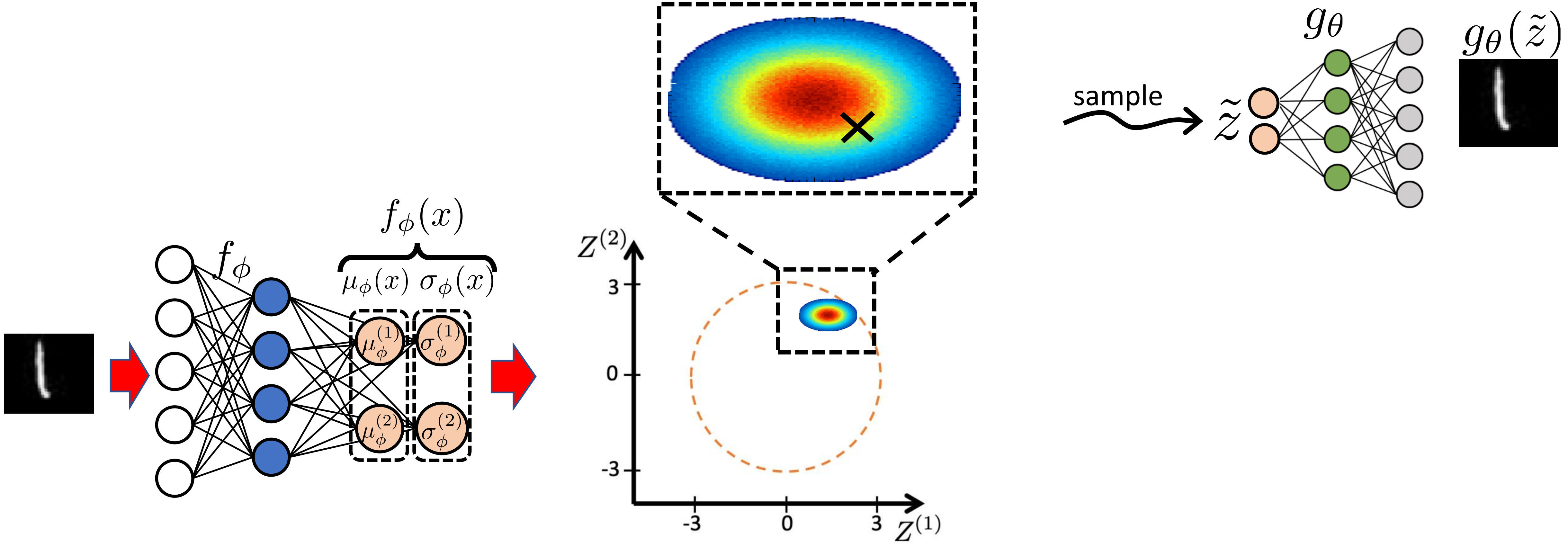


Step 2. $\tilde{z}$
Decode $\tilde{z}$ with a standard decoder, similar to a basic AE.

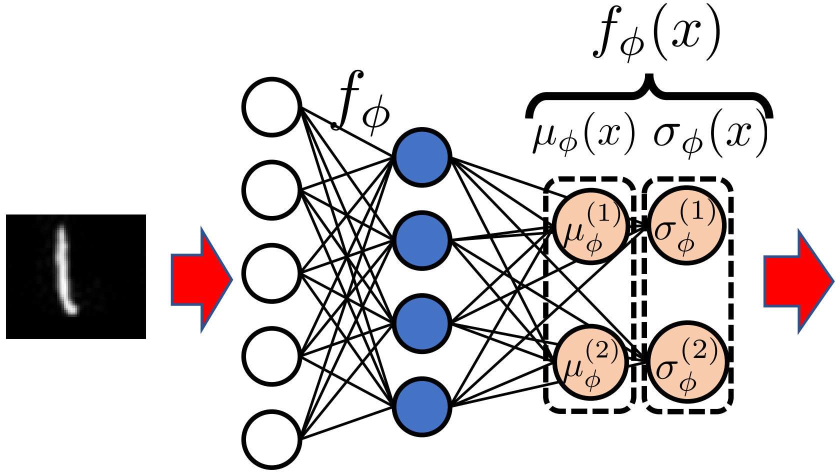
2. Stochastic Decoder (during training)

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

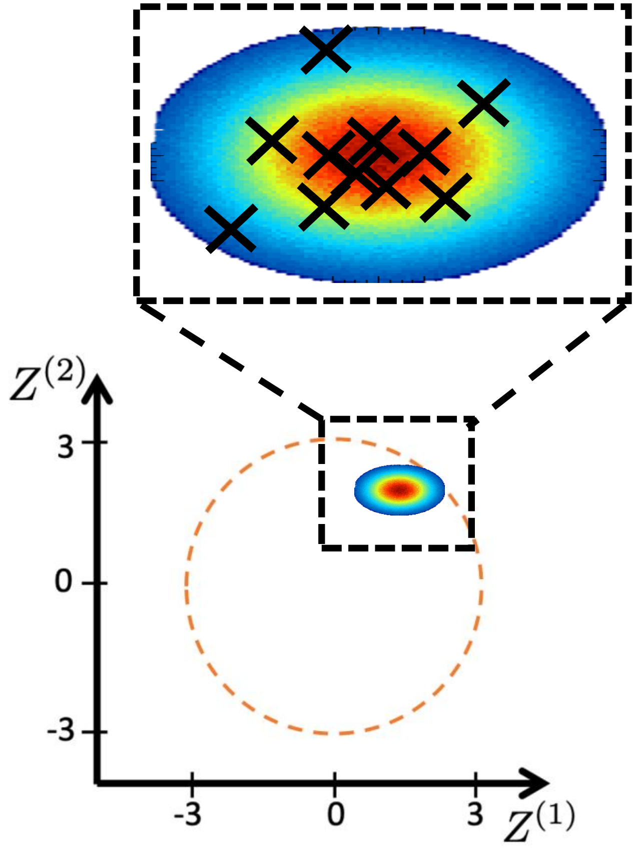
$$p_\phi(z|x) = N(\mu_\phi(x), \sigma_\phi(x)^2)$$



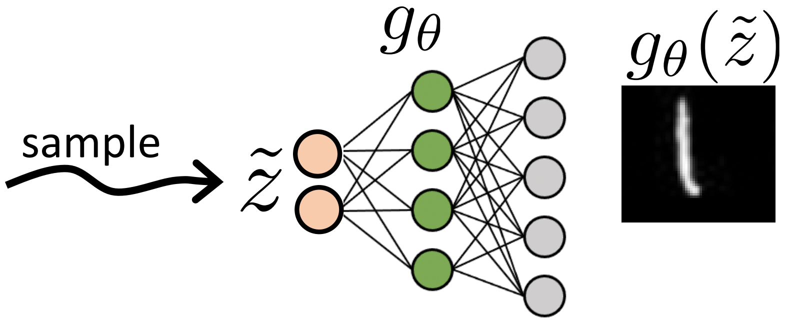
2. Stochastic Decoder (during training)



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



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

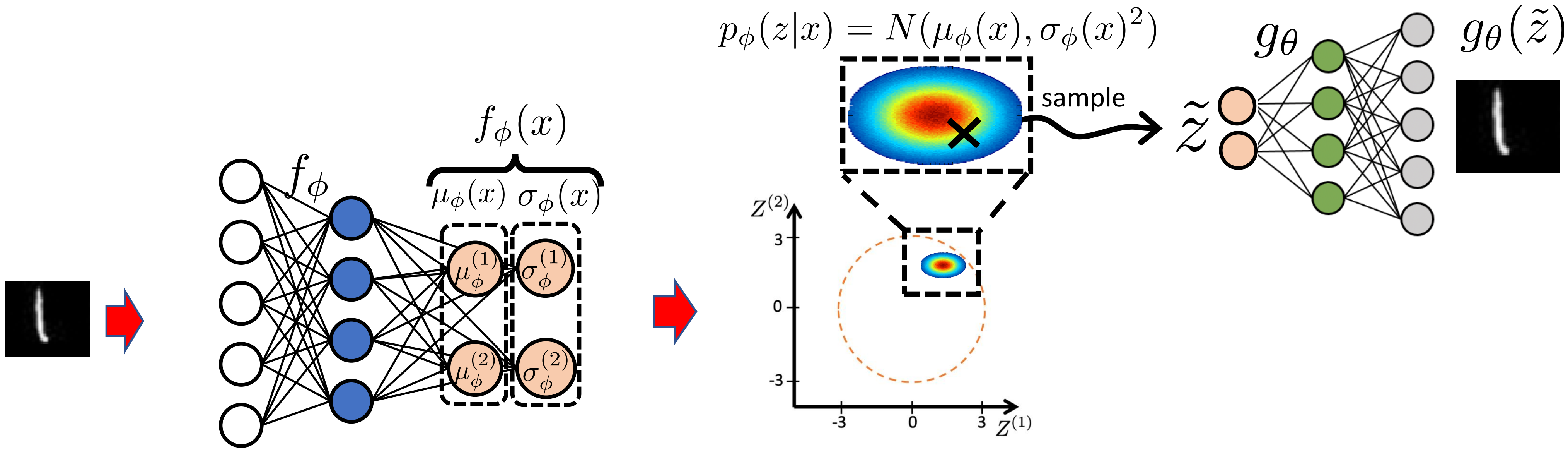


However, at different SGD iterations, different samples may be drawn for the same x.

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

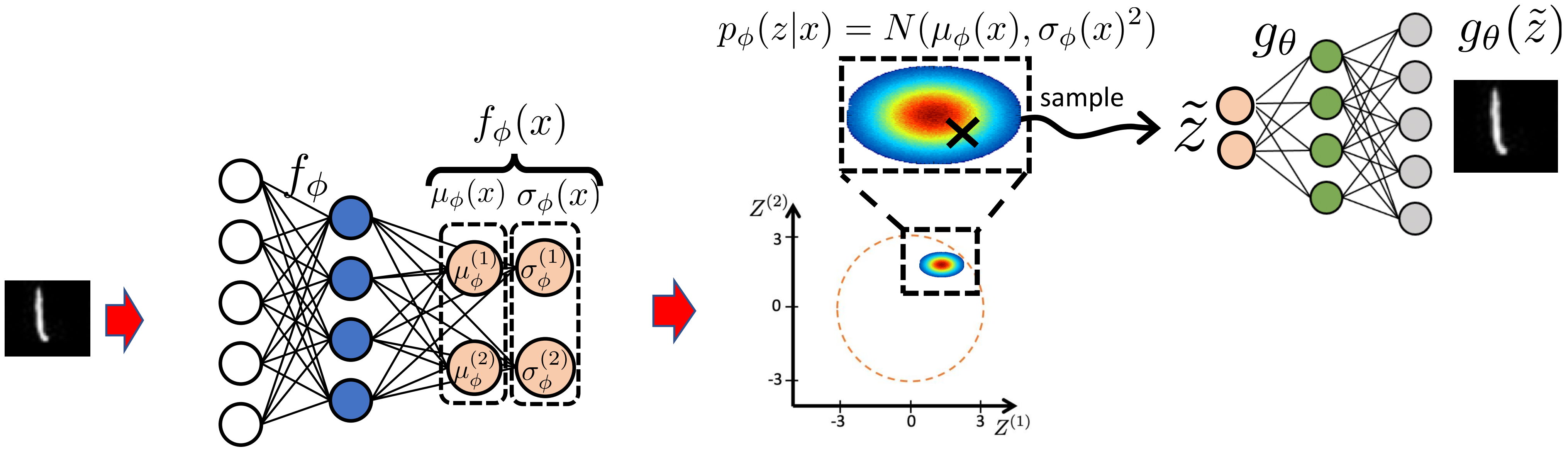


What do we want to train this model to do?

Create **realistic images** with the decoder

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

3. Training Objective

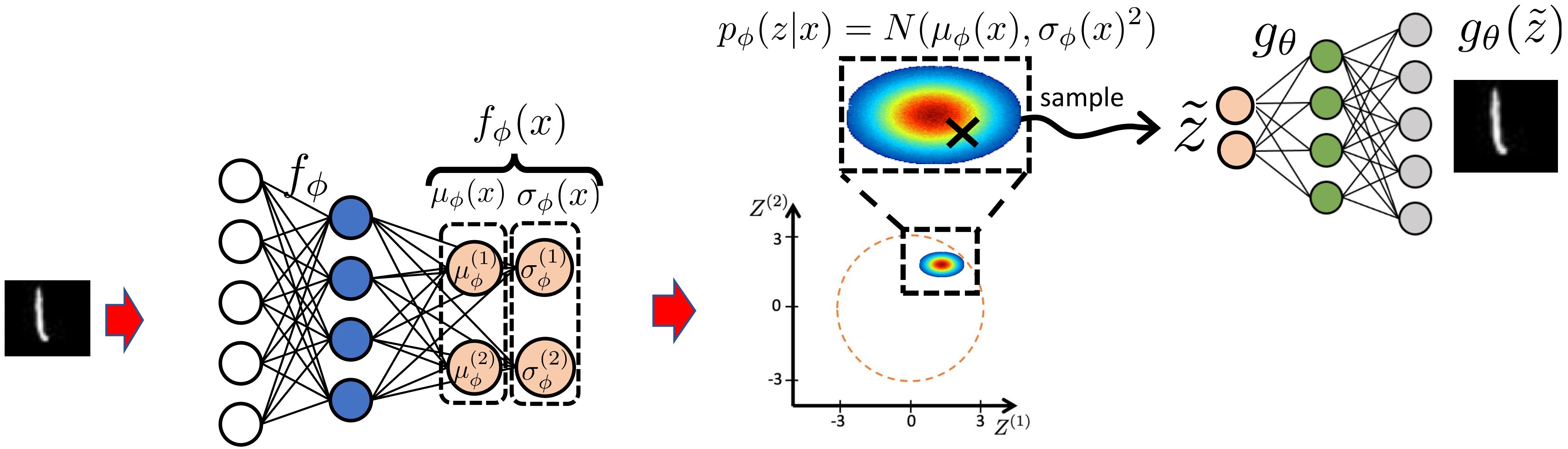


$$\mathcal{L}_{VAE} = \overbrace{\mathcal{L}_{rec}}^{\textbf{Reconstruction loss}} + \underset{\downarrow}{\lambda}\overbrace{\mathcal{L}_{reg}}^{\textbf{Regularizer}}$$

Strength of Regularizer

(chosen by us through experimentation)

3. Training Objective



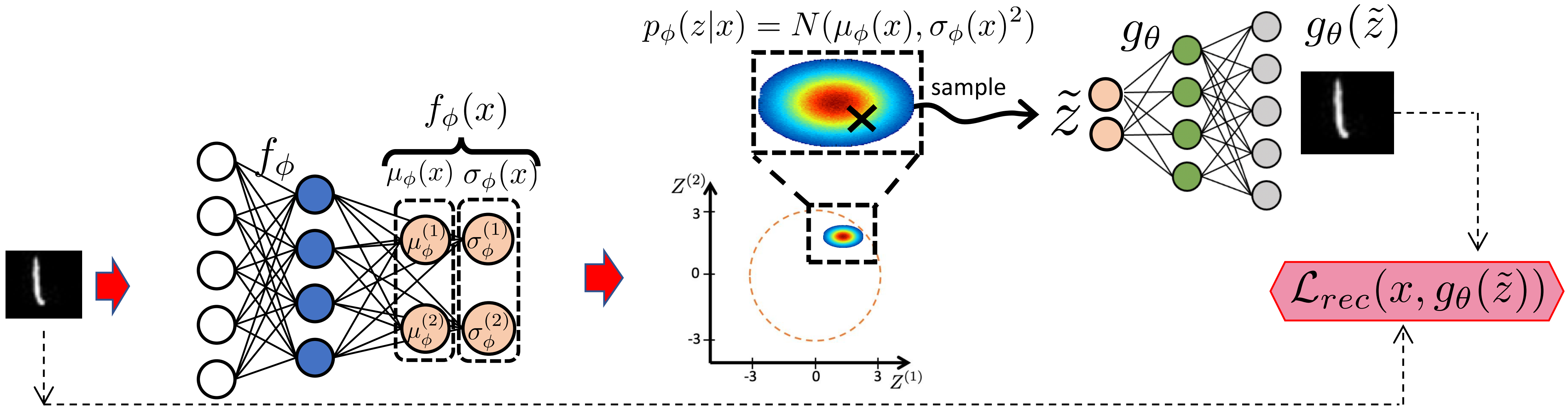
$$\mathcal{L}_{VAE} = \overbrace{\mathcal{L}_{rec}}^{\textbf{Reconstruction loss}} + \lambda \overbrace{\mathcal{L}_{reg}}^{\textbf{Regularizer}}$$

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

*“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



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

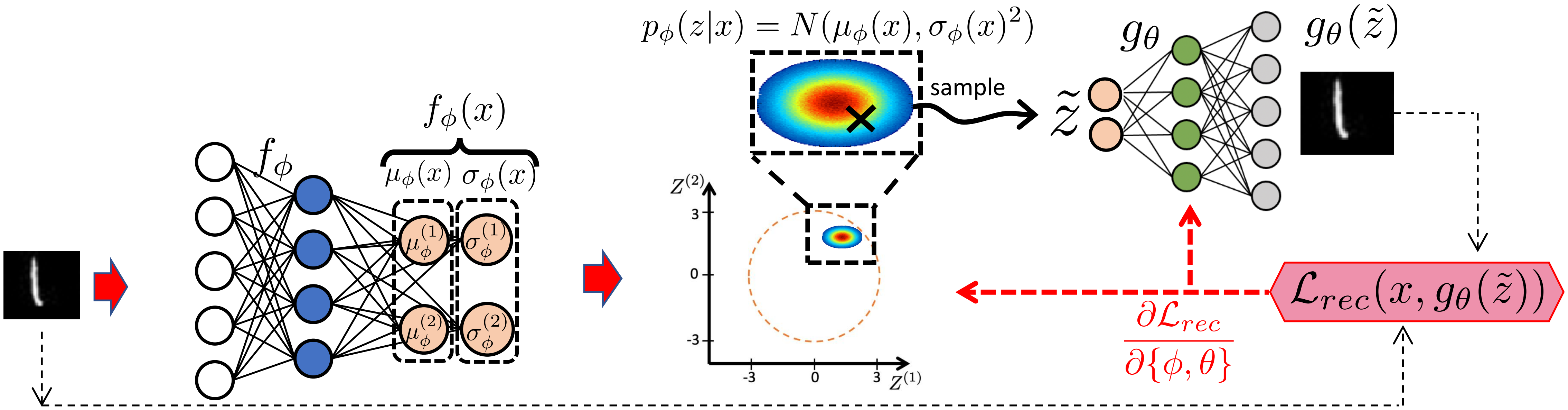
Reconstruction loss:

$$\mathcal{L}_{rec} = \frac{1}{d}\sum_{j=1}^{d}\left(x^{(j)} - g_\theta(\tilde{z})^{(j)}\right)^2 \quad \text{for } \tilde{z} \sim p_\phi(z|x)$$

Reconstruct input x, for any $\tilde{z}$ sampled from $p_\phi(z|x)$.

Dimensionality of x

3. Training Objective: Reconstruction loss



$p_\phi(z|x) = N(\mu_\phi(x), \sigma_\phi(x)^2)$
sample
$\tilde{z}$
g_θ
$g_\theta(\tilde{z})$
$f_\phi(x)$
f_ϕ
$\mu_\phi(x)$ $\sigma_\phi(x)$
$\mu_\phi^{(1)}$
$\sigma_\phi^{(1)}$
$\mu_\phi^{(2)}$
$\sigma_\phi^{(2)}$
$Z^{(2)}$
3
0
-3
-3
0
3
$Z^{(1)}$
$\mathcal{L}_{rec}(x, g_\theta(\tilde{z}))$
$\frac{\partial \mathcal{L}_{rec}}{\partial \{\phi, \theta\}}$

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

Reconstruction loss:

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

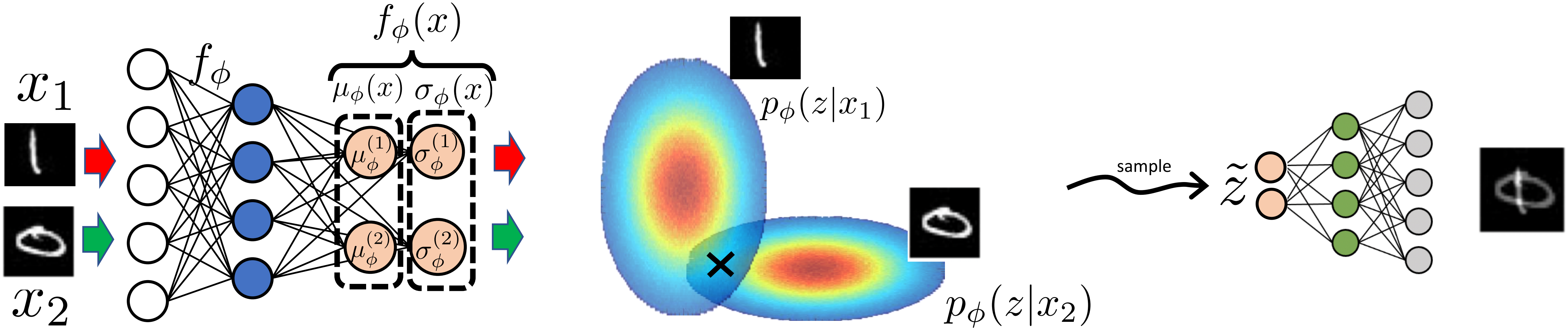
Function of θ
(decoder params)

Function of ϕ
(encoder params)

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

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

Effect of Reconstruction loss:

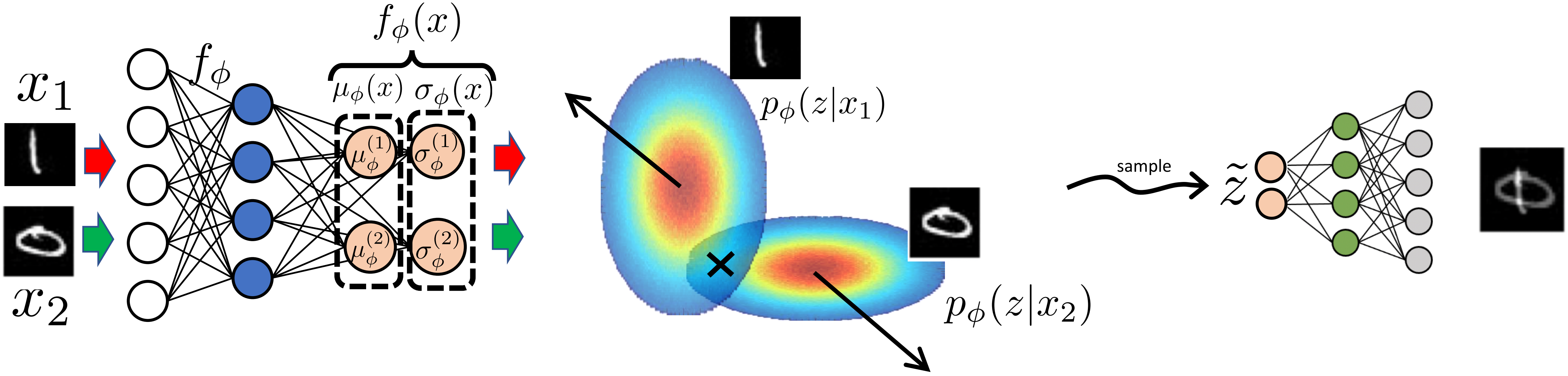


Assume that an encoder learns to encode these two digits as shown.

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

A: Overlapping area

Effect of Reconstruction loss: on the predicted means

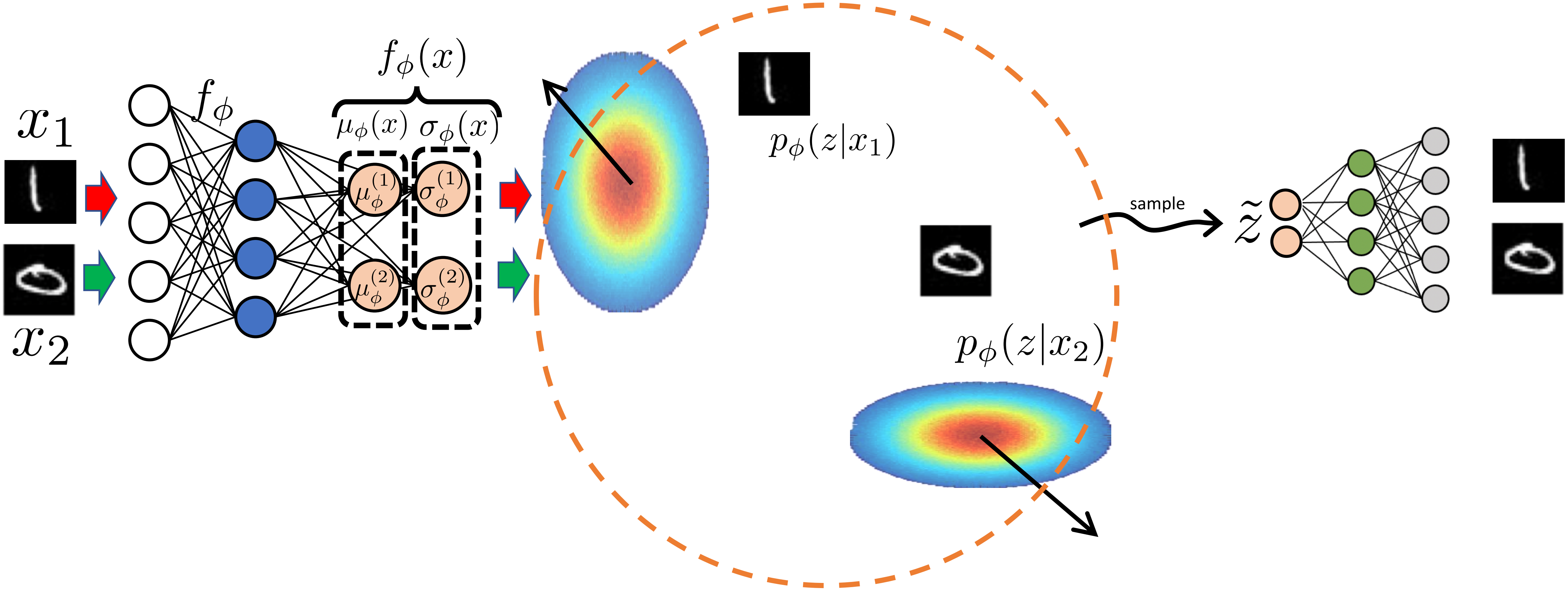


Grads of Recon loss encourages learning means of different samples away from each other in latent space $\mathcal{Z}$

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

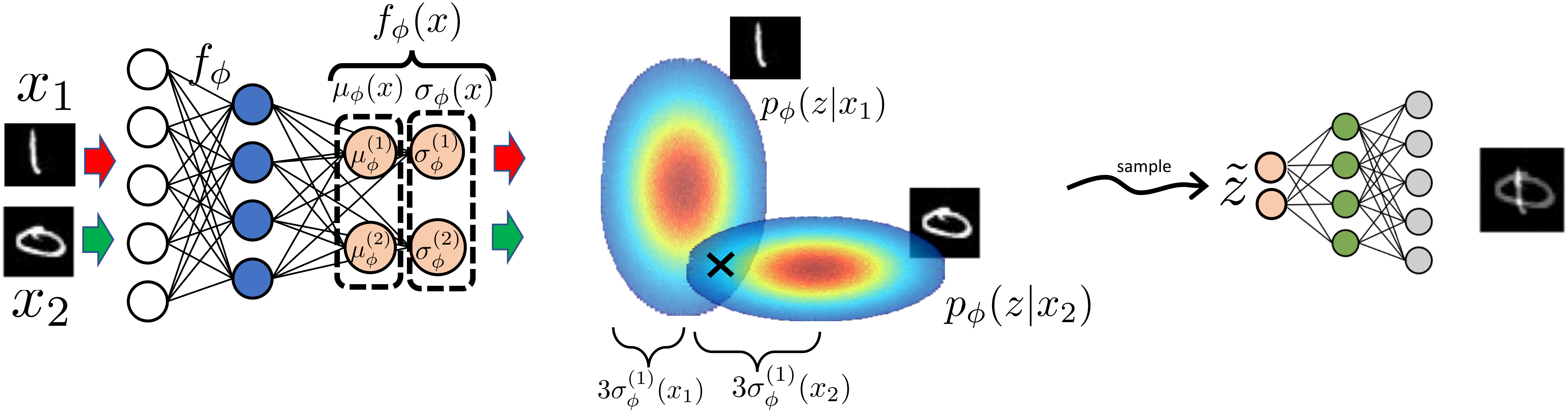
A: Learn parameters that predict ***means that are away from each other***.

Effect of Reconstruction loss: on the predicted <u>means</u>



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).

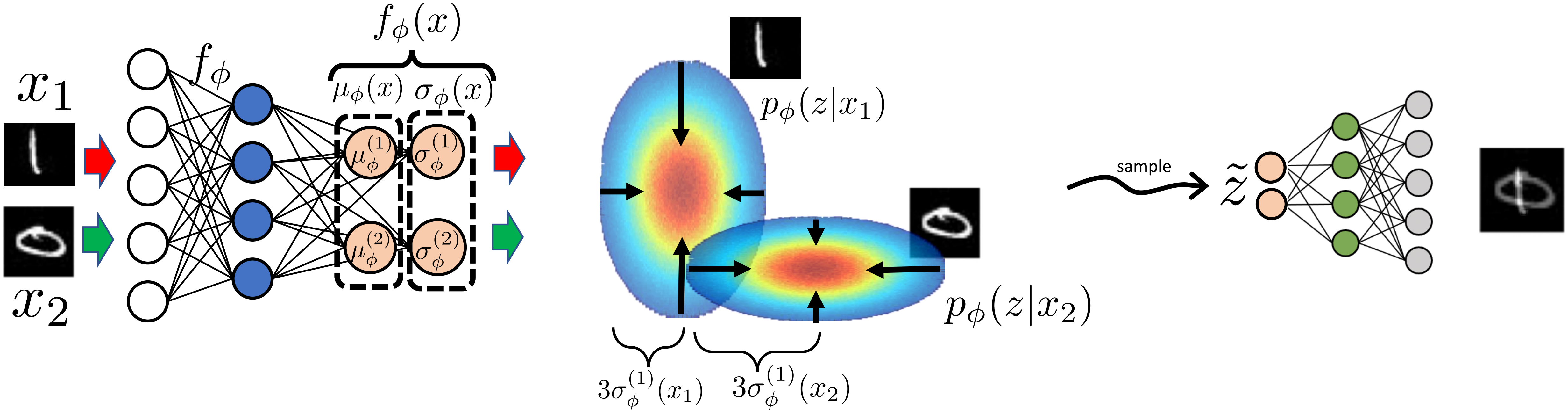
Effect of Reconstruction loss: on the predicted std. deviations



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

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

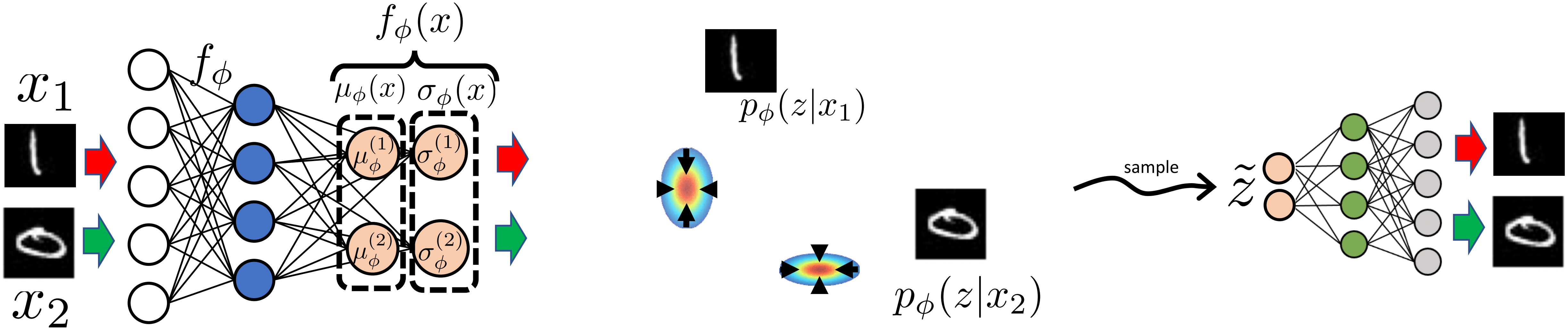
Effect of Reconstruction loss: on the predicted std. deviations



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

A: Learn parameters that predict ***<u>small</u> standard deviations***.

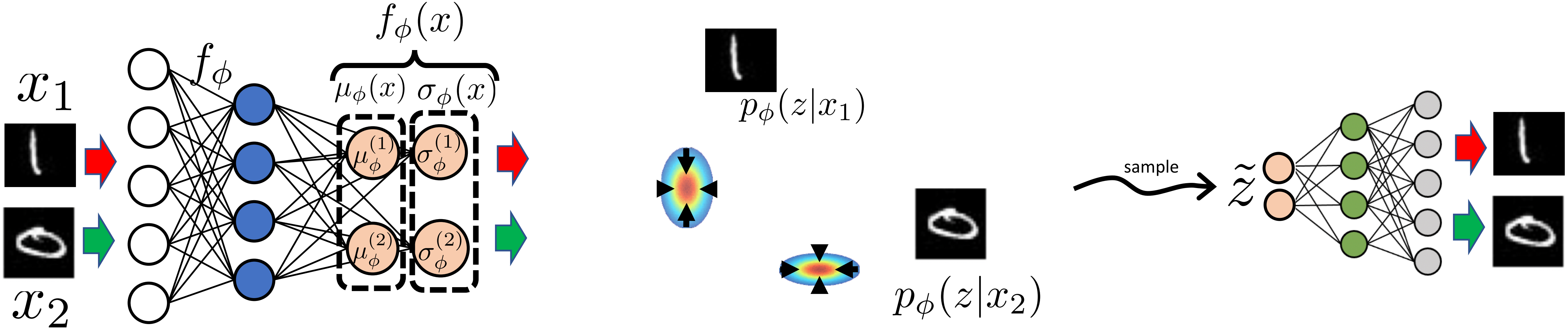
Effect of Reconstruction loss: on the predicted std. deviations



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

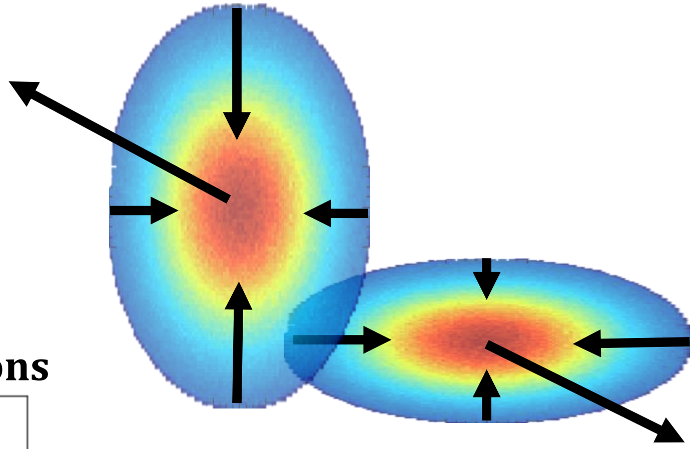
A: Learn parameters that predict ***<u>small</u> standard deviations***.

Effect of Reconstruction loss: on the predicted std. deviations



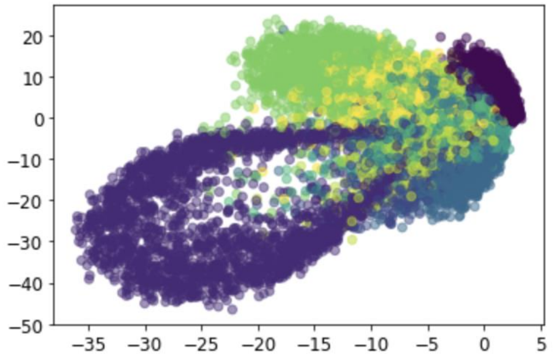
If we would minimize ONLY the Reconstruction loss,
“ideal” standard deviations for reconstruction loss are <u>zero</u> for every x!

Influence of Reconstruction loss:



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

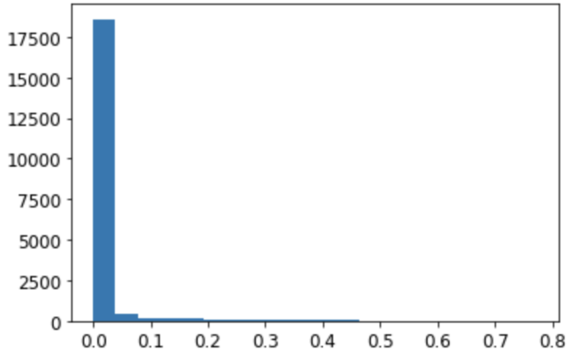
Means in 2D space of Z:



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

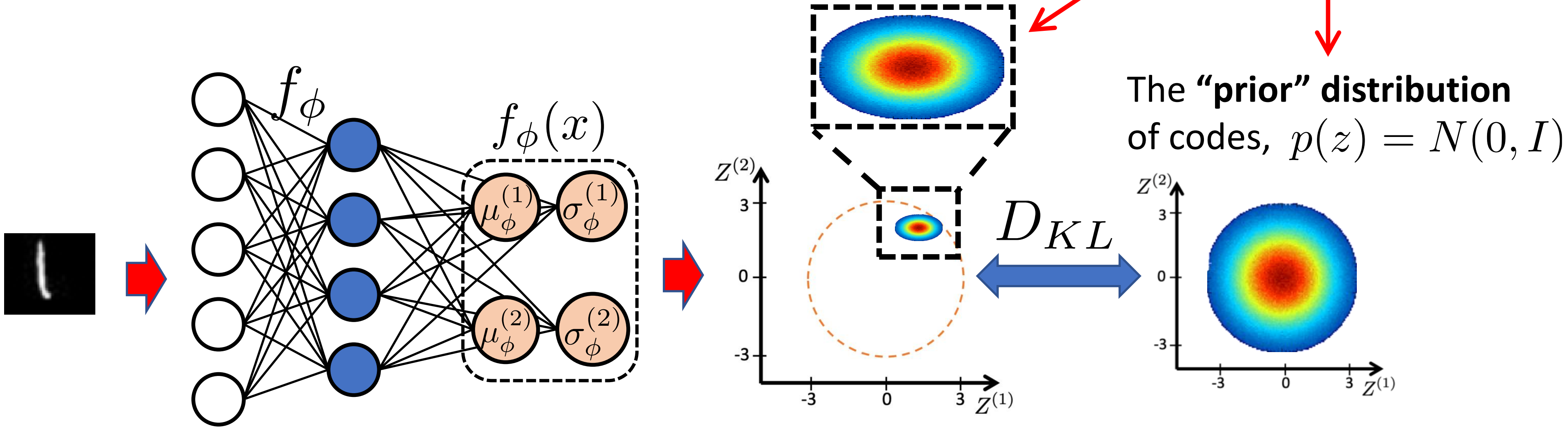
$$\{\theta', \phi'\} = arg \min_{\theta,\phi} \mathbb{E}_{x \sim p_{data}} \left[\mathcal{L}_{VAE}\right] = arg \min_{\theta,\phi} \mathbb{E}_{x \sim p_{data}} \left[\mathcal{L}_{rec}\right] + \lambda \underline{\mathbb{E}_{x \sim p_{data}} \left[\mathcal{L}_{reg}\right]}$$

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

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

Measure of difference between **2 distributions**

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



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

3. Training Objective: Regularizer

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

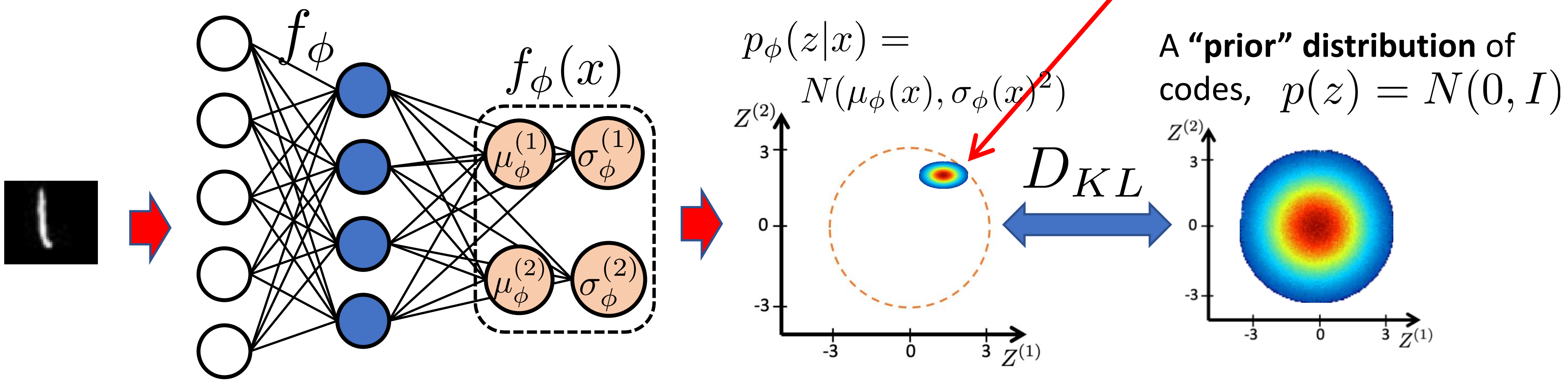
$$\downarrow$$

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

Measure of difference between **2 distributions**

What this regularizer does:

"Learn encoder parameters ϕ 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 $N(0, I)$ "



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

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

3. Training Objective: Regularizer

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

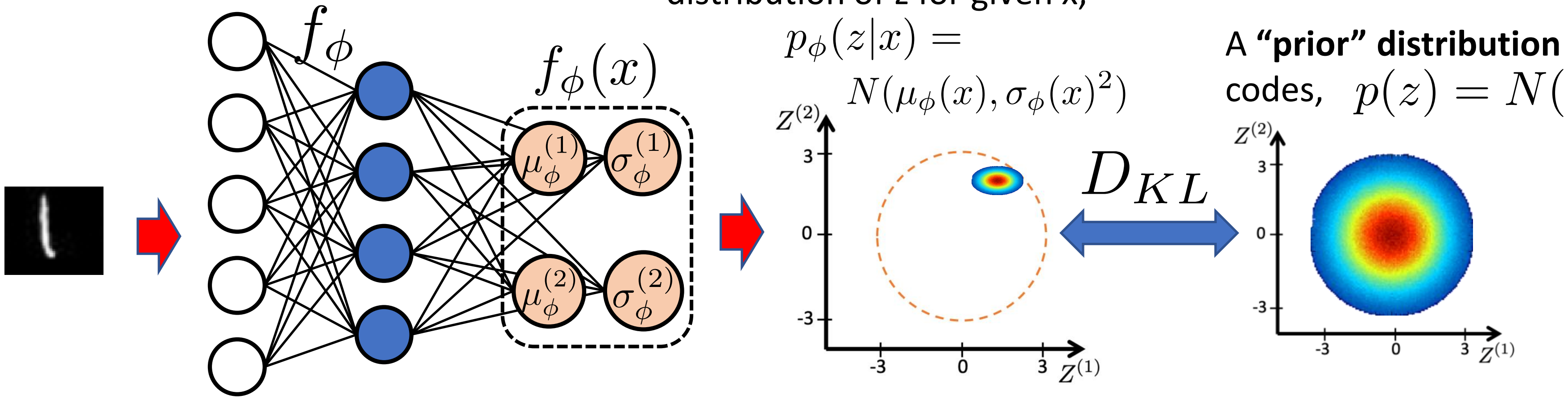
$$\Downarrow$$

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

$$\Downarrow$$

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

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



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, $p(z) = N(0, I)$

3. Training Objective: Regularizer

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

$$\Downarrow$$

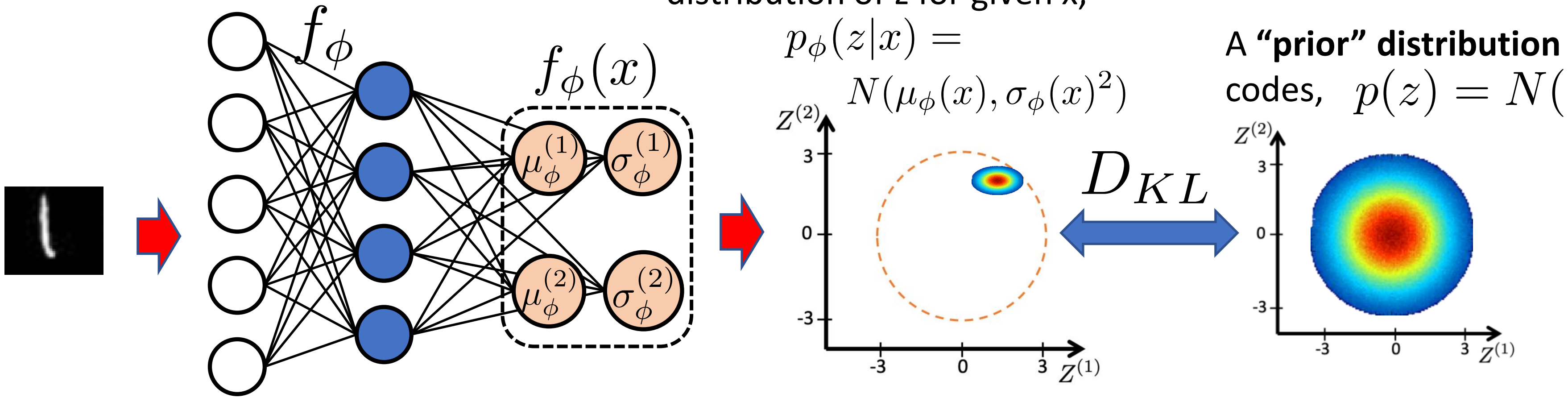
$$D_{KL}\left[p_\phi(z|x)||N(0,I)\right]$$

$$\Downarrow$$

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

The regularizer is **ONLY** a function of ϕ, **the parameters of the <u>encoder</u>!**

It does NOT train the decoder!

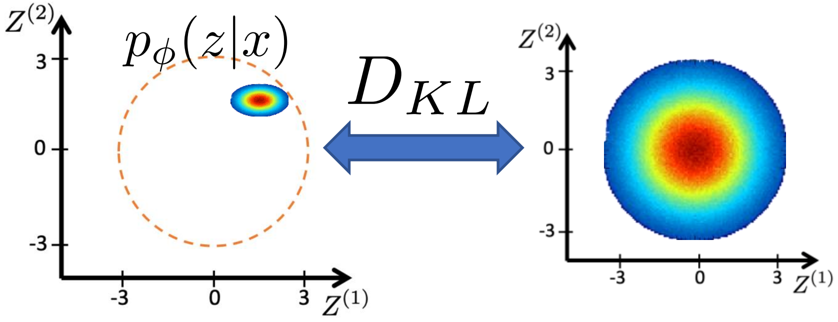


Predicted **"posterior"** distribution of z for given x, $p_\phi(z|x) = N(\mu_\phi(x), \sigma_\phi(x)^2)$

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

Effect of Regularizer on means and standard deviations:

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



To **minimize** the regularizer….

…for each of v **dimensions** of z…

…mean^2 must be minimized, which **means ideally** the predicted means must be….????

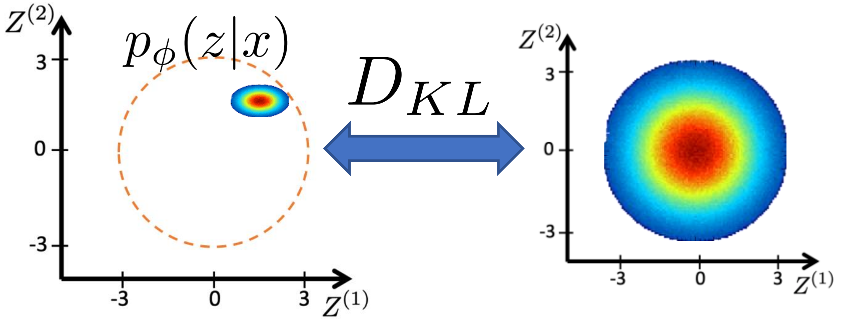
…**must be 0!**

Effect of Regularizer on means and standard deviations:

$$\phi' = arg\min_{\phi} \mathcal{L}_{reg}(x) \Rightarrow$$

$$= arg\min_{\phi} D_{KL}\left[p_{\phi}(z|x)||N(0,I)\right] \Rightarrow$$

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

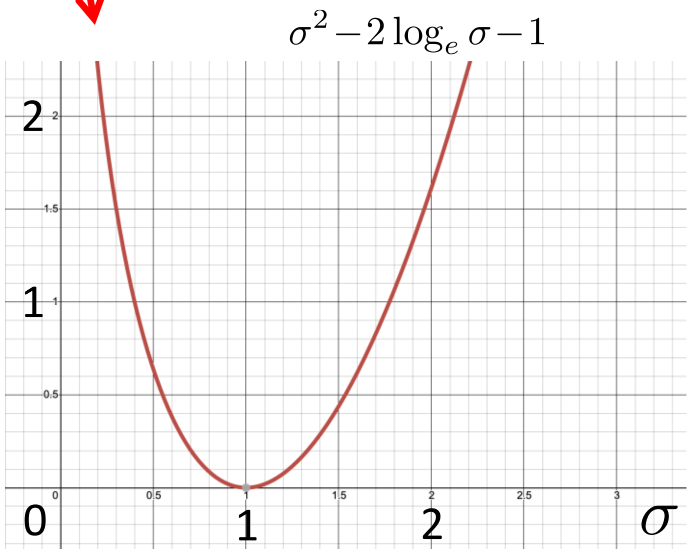


To **minimize** the regularizer….

…for each of v **dimensions** of z…

…mean^2 must be minimized, which **means ideally** the predicted means must be….????

…**must be 0!**



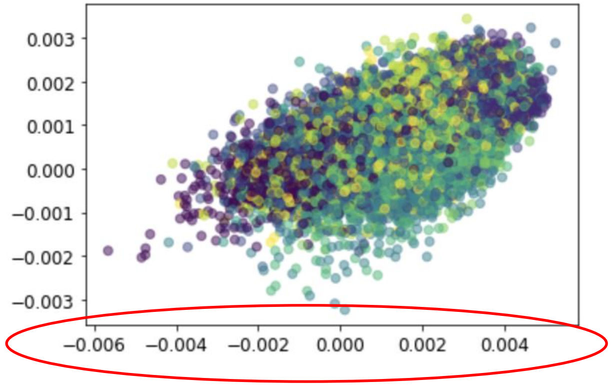
…the predicted standard deviation must **ideally be** ????….

… must be =1 !

Effect of Regularizer:

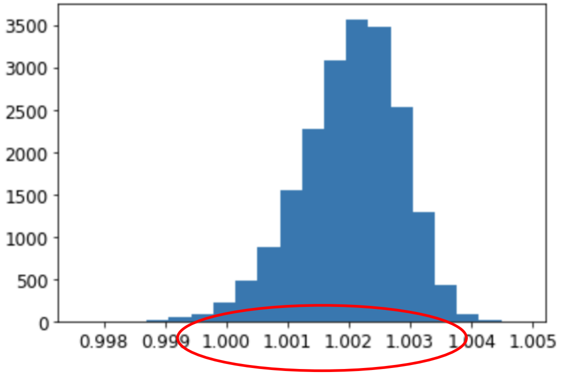
If we would train the VAE to **only** minimize the Regularizer:

Means in 2D space of Z:



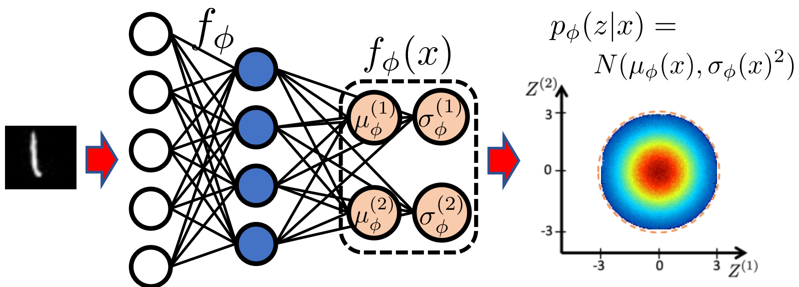
For each x, predicted **mean tends to 0.**

Histogram of **Std. Deviations**



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:

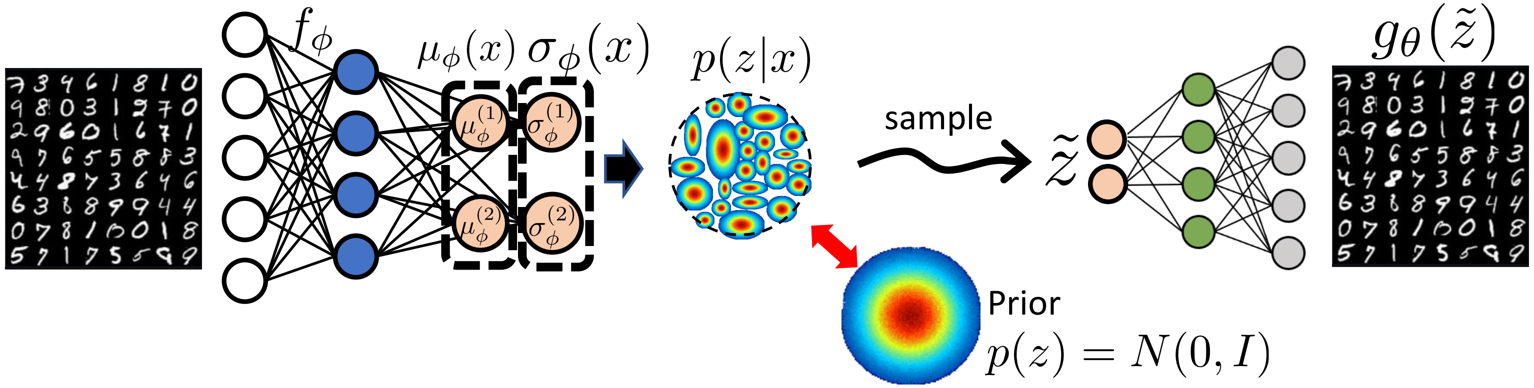
$$\{\theta', \phi'\} = arg \min_{\theta,\phi} \mathbb{E}_{x \sim p_{data}} \left[\mathcal{L}_{rec}\right] + \lambda \mathbb{E}_{x \sim p_{data}} \left[\mathcal{L}_{reg}\right]$$

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:

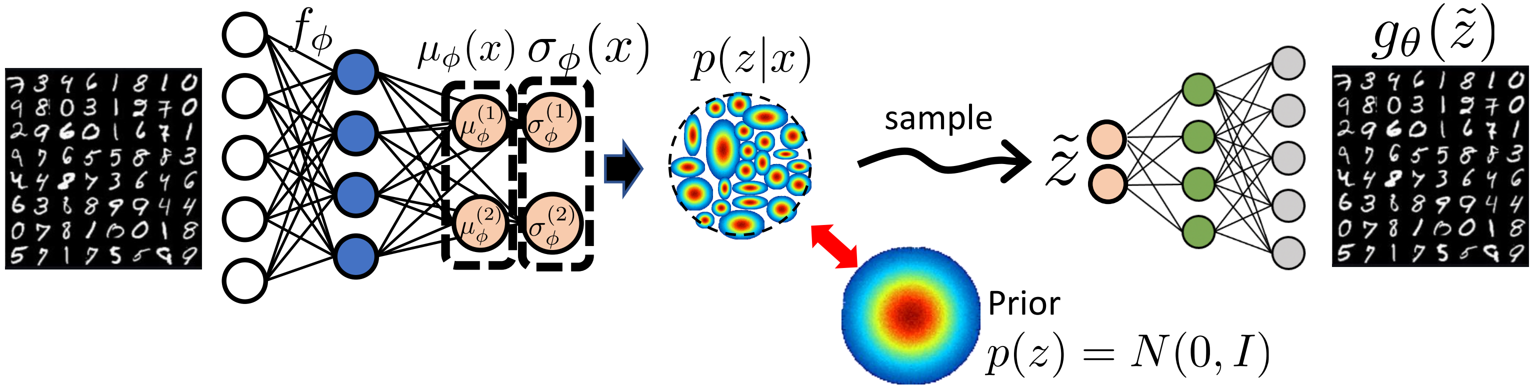
$$\{\theta', \phi'\} = arg\min_{\theta,\phi} \mathbb{E}_{x\sim p_{data}}\left[\mathcal{L}_{rec}\right] + \lambda \mathbb{E}_{x\sim p_{data}}\left[\mathcal{L}_{reg}\right]$$

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 λ (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.**

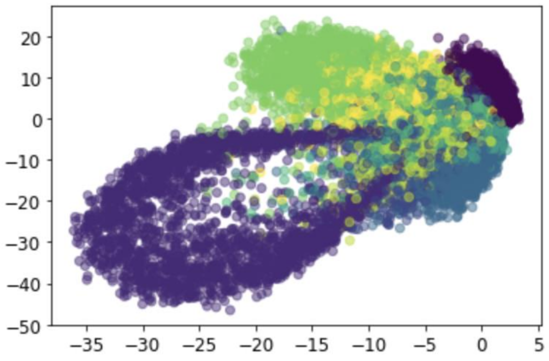
* The explanation why this happens is in the probabilistic derivation of the VAE (further reading/optional).

Combination of the loss terms:

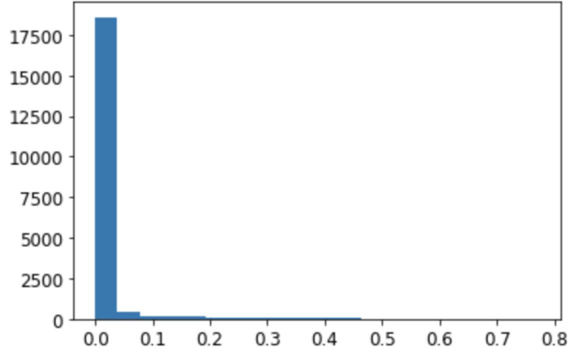
Example **in practice**, with stochastic optimization and non-optimal training parameters:
(from our Tutorial - VAEs on MNIST)

Reconstruction only

Means in 2D space of Z:

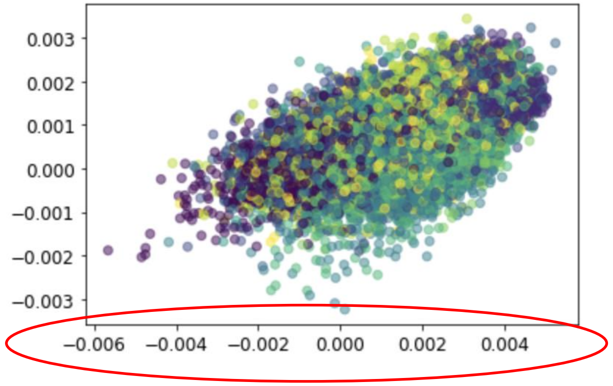


Histogram of **Std. Deviations**

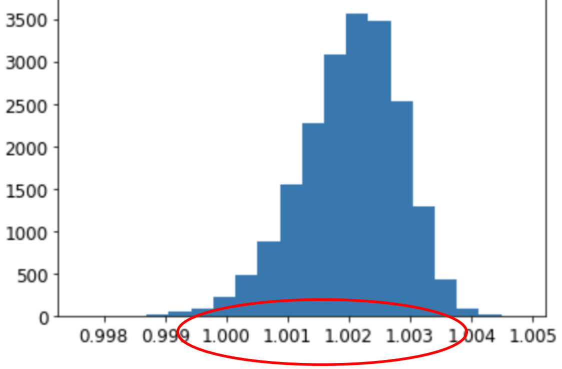


Regularizer only

Means in 2D space of Z:

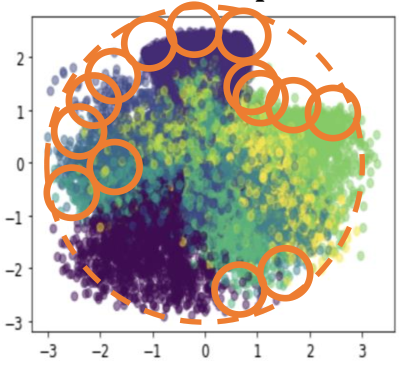


Histogram of **Std. Deviations**

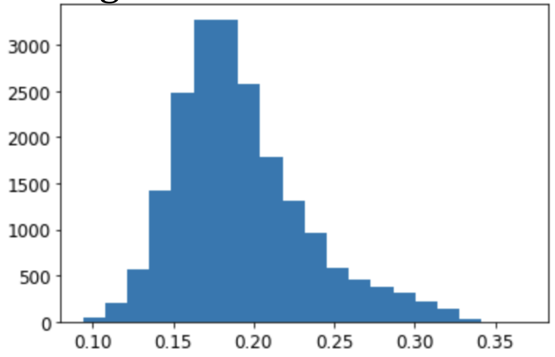


Combination

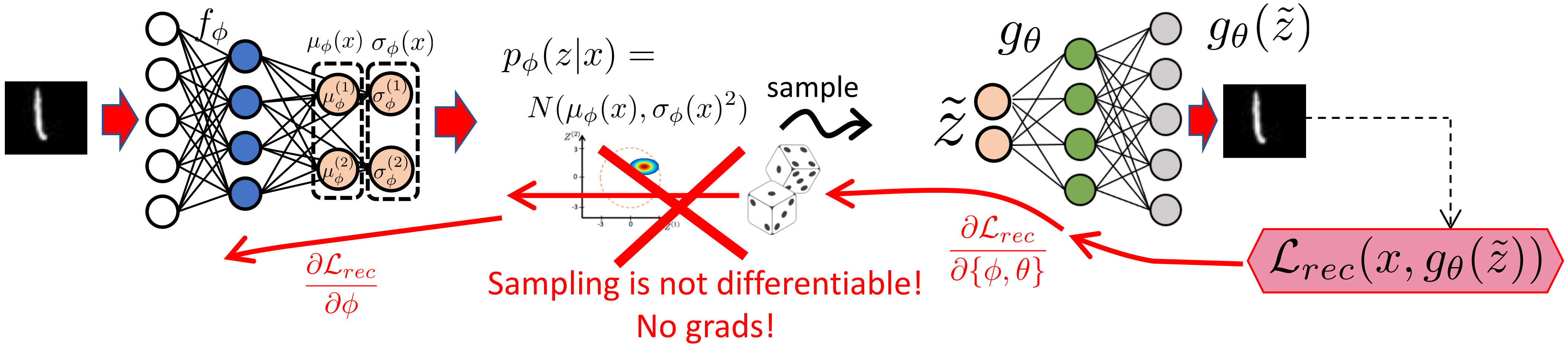
Means in 2D space of Z:



Histogram of **Std. Deviations**



4. Re-parameterization trick



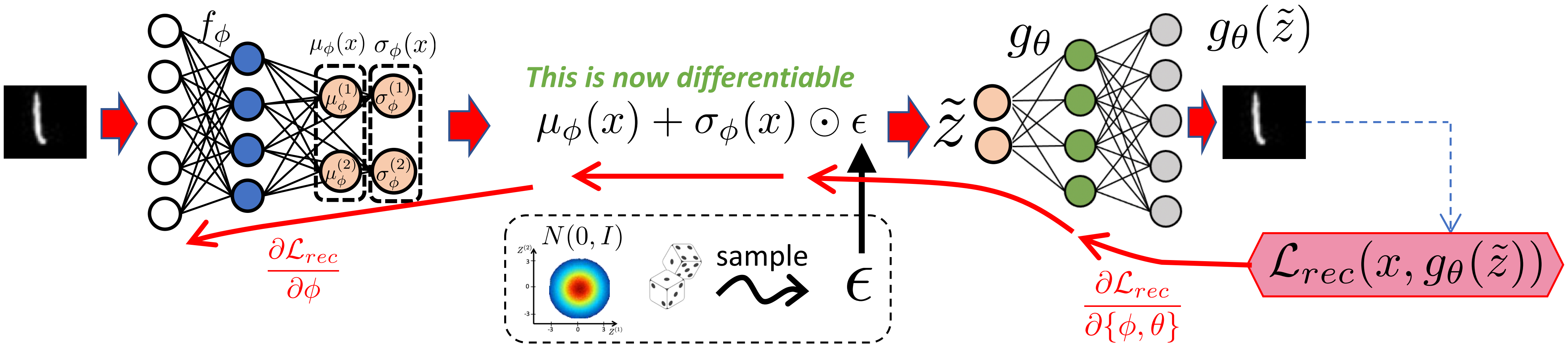
f_ϕ
$\mu_\phi(x)$ $\sigma_\phi(x)$
$\mu_\phi^{(1)}$ $\sigma_\phi^{(1)}$
$\mu_\phi^{(2)}$ $\sigma_\phi^{(2)}$
$p_\phi(z|x) =$
$N(\mu_\phi(x), \sigma_\phi(x)^2)$
sample
$\tilde{z}$
g_θ
$g_\theta(\tilde{z})$
$\mathcal{L}_{rec}(x, g_\theta(\tilde{z}))$
$\frac{\partial \mathcal{L}_{rec}}{\partial\{\phi,\theta\}}$
$\frac{\partial \mathcal{L}_{rec}}{\partial \phi}$

Sampling is not differentiable!
No grads!

Solution: The Re-parameterization trick

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

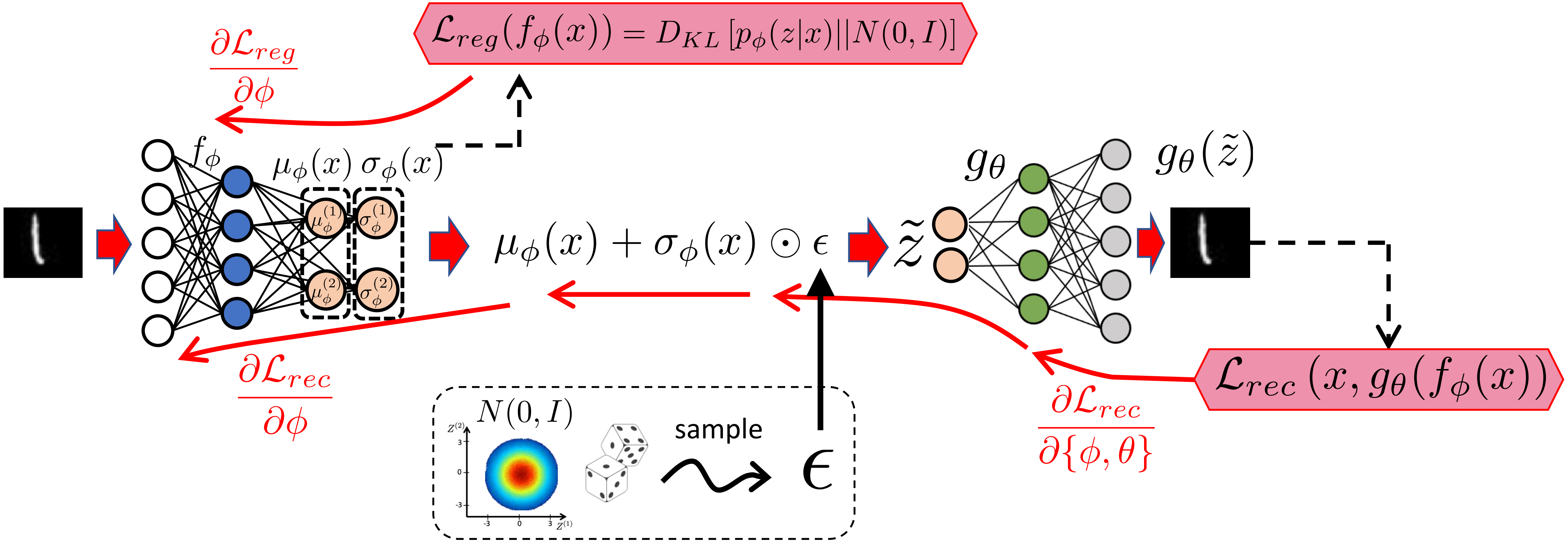
$$\tilde{z} \sim N\left(\mu_\phi(x), \sigma_\phi(x)^2\right) \Leftrightarrow \tilde{z} = \mu_\phi(x) + \sigma_\phi(x) \odot \epsilon, \text{ with } \epsilon \sim N(0, I)$$



f_ϕ
$\mu_\phi(x)$ $\sigma_\phi(x)$
$\mu_\phi^{(1)}$ $\sigma_\phi^{(1)}$
$\mu_\phi^{(2)}$ $\sigma_\phi^{(2)}$
$\mu_\phi(x) + \sigma_\phi(x) \odot \epsilon$
$N(0, I)$
sample
ϵ
$\tilde{z}$
g_θ
$g_\theta(\tilde{z})$
$\mathcal{L}_{rec}(x, g_\theta(\tilde{z}))$
$\frac{\partial \mathcal{L}_{rec}}{\partial\{\phi,\theta\}}$
$\frac{\partial \mathcal{L}_{rec}}{\partial \phi}$

This is now differentiable

Bringing everything together: Training

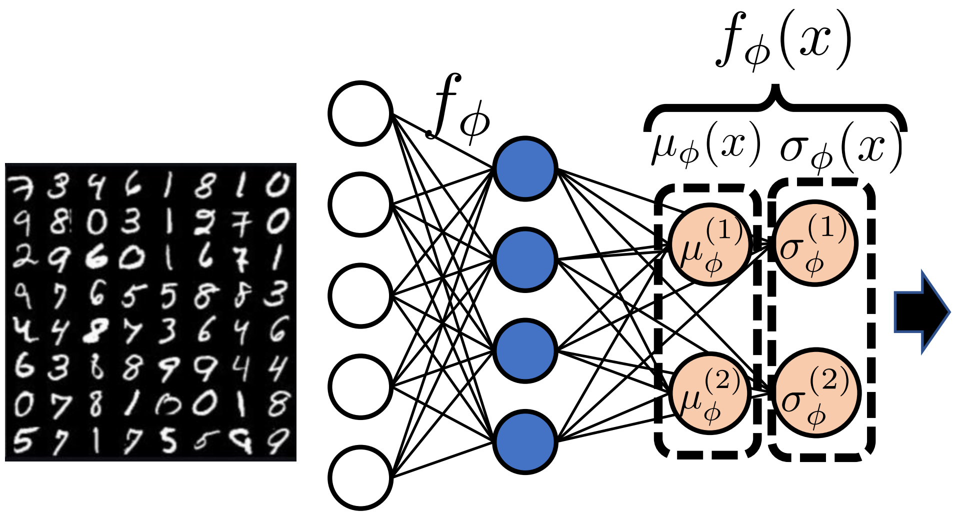


$\mathcal{L}_{reg}(f_\phi(x)) = D_{KL}\left[p_\phi(z|x)||N(0,I)\right]$
$\frac{\partial \mathcal{L}_{reg}}{\partial \phi}$
f_ϕ
$\mu_\phi(x)$ $\sigma_\phi(x)$
$\mu_\phi^{(1)}$ $\sigma_\phi^{(1)}$
$\mu_\phi^{(2)}$ $\sigma_\phi^{(2)}$
$\mu_\phi(x) + \sigma_\phi(x) \odot \epsilon$
$\tilde{z}$
g_θ
$g_\theta(\tilde{z})$
$\frac{\partial \mathcal{L}_{rec}}{\partial \phi}$
$\mathcal{L}_{rec}(x, g_\theta(f_\phi(x)))$
$\frac{\partial \mathcal{L}_{rec}}{\partial \{\phi,\theta\}}$
$N(0,I)$
$z^{(2)}$
3
0
-3
-3
0
3
$z^{(1)}$
sample
ϵ

Reparameterization trick

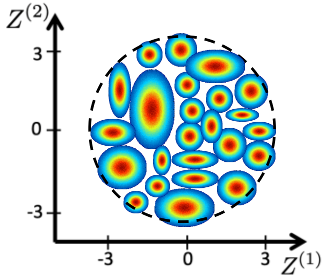
Kingma & Welling, Auto-encoding variational bayes, ICLR 2014
Rezende et al, Stochastic Backpropagation and Approximate Inference in Deep Generative Models, ICML 2014

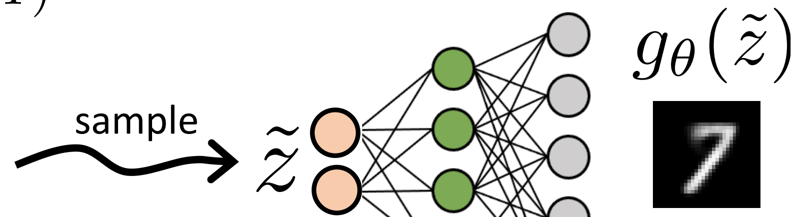
Bringing everything together: Training



Predicted distributions p(z|x) of all training data together cover the **"prior" distribution** of z,

$$p(z) = N(0, I)$$

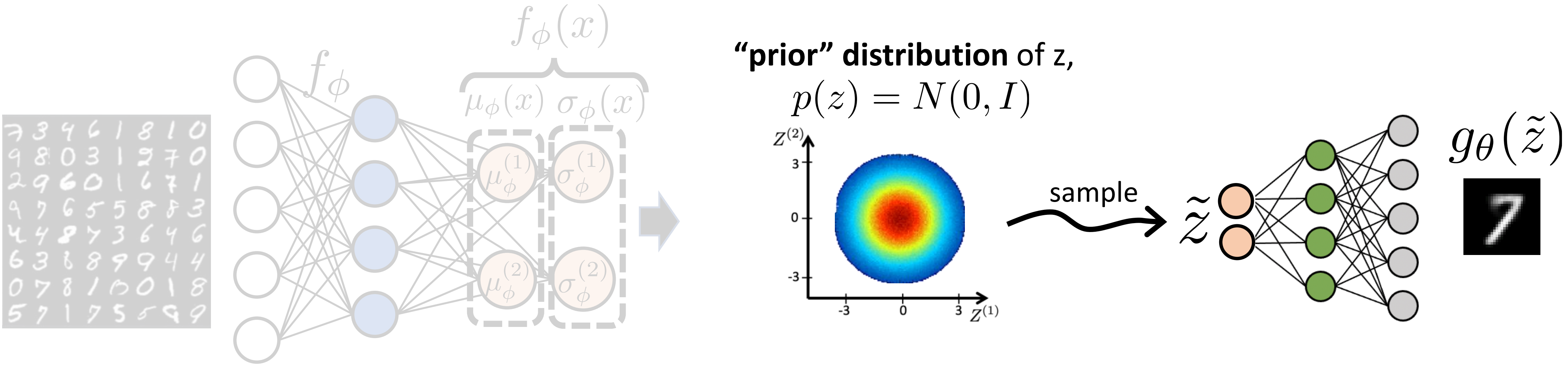




Kingma & Welling, Auto-encoding variational bayes, ICLR 2014

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

Bringing everything together: Generating new data



“prior” distribution of z,

$p(z) = N(0, I)$

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