Structured Disentangled Representations

Jan-Willem van de Meent






Babak Esmaeli   Hao Wu   Sarthak Jain   Alican Bozkurt

Northeastern University

Siddharth N.

Oxford

Brooks Paige

Alan Turing Institute

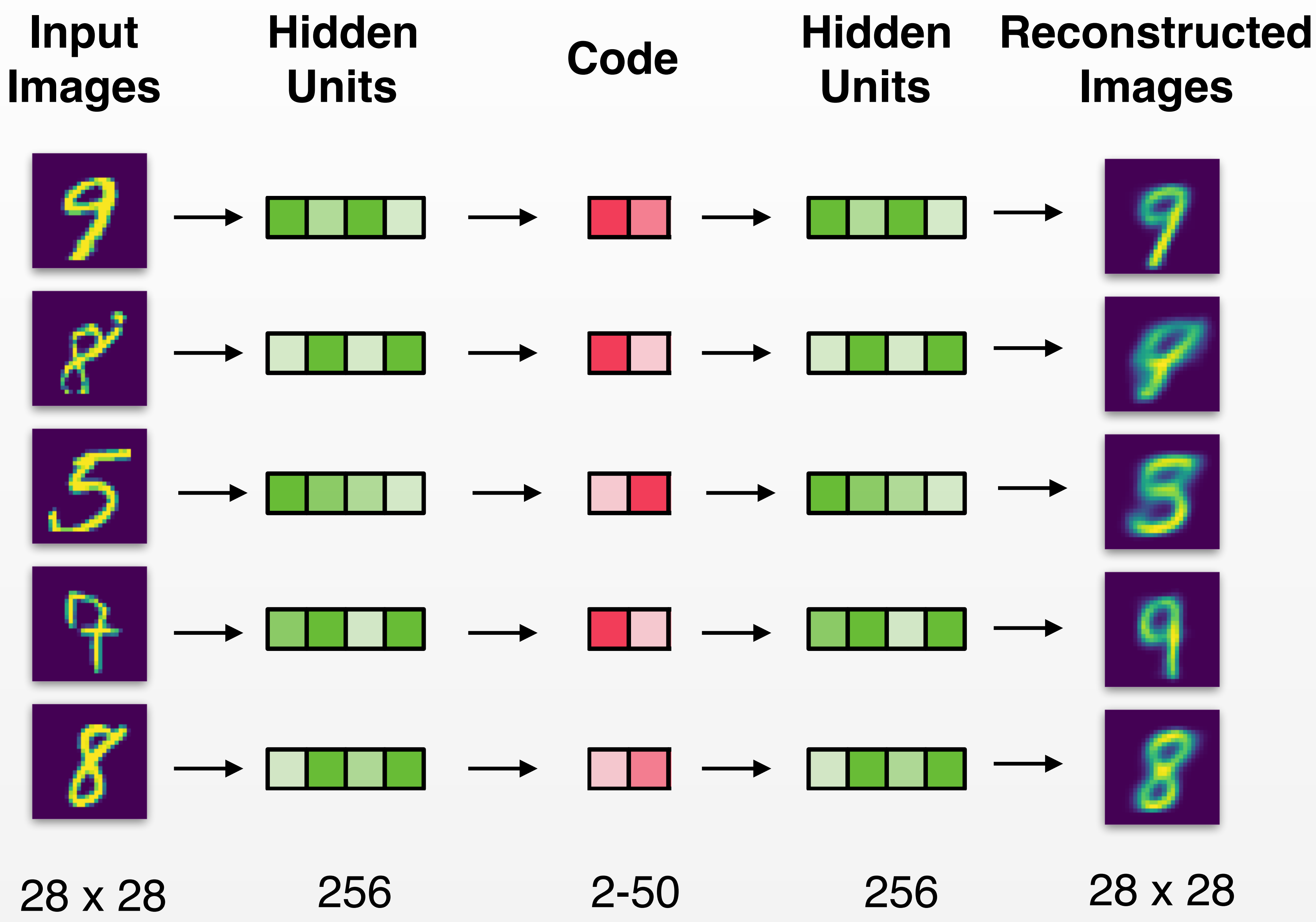
Learning Representations

Everyone's Favorite Example: MNIST

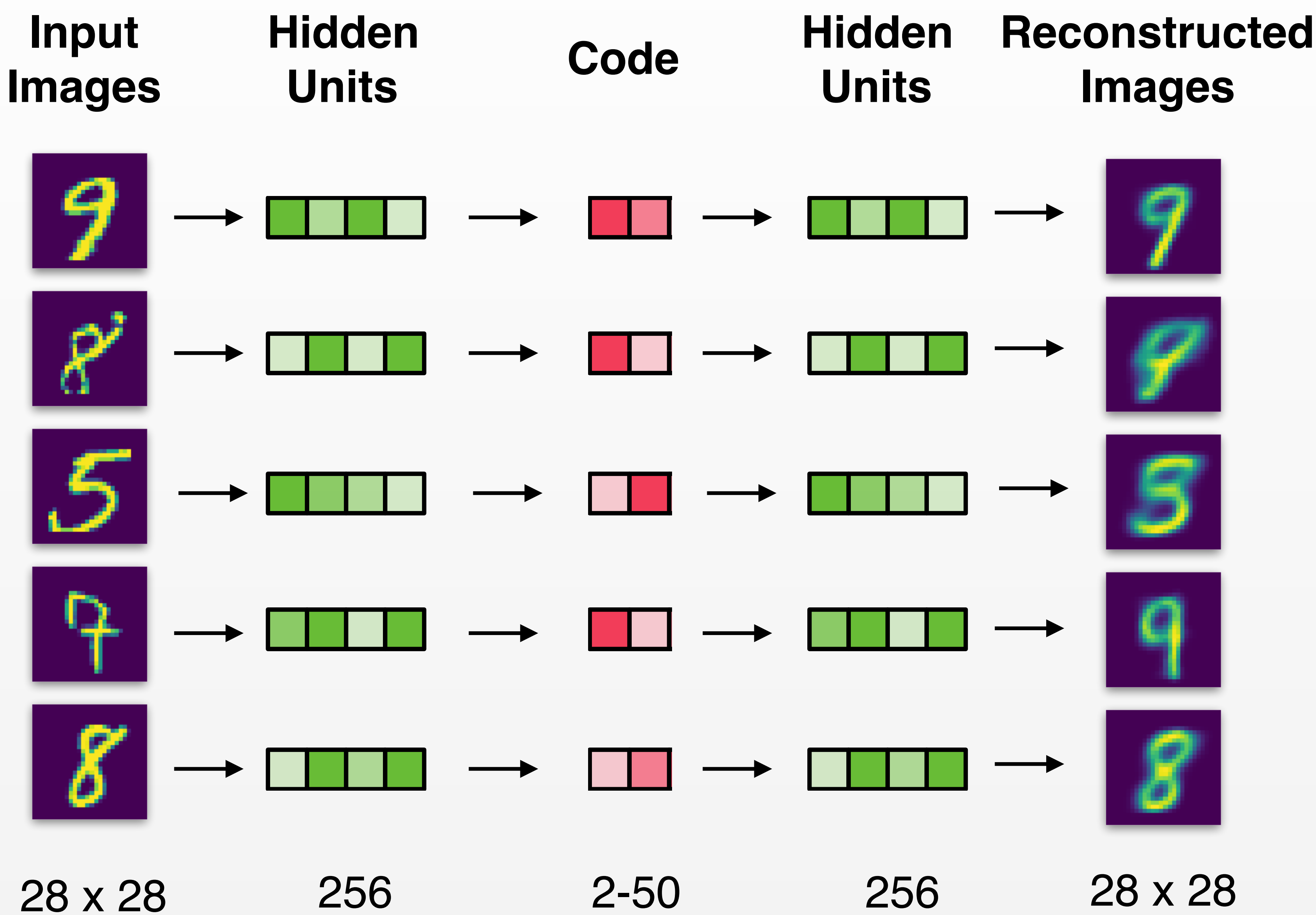
Goal: Learn features that capture ***similarities*** and ***dissimilarities***

Requirement: Objective that defines notion of *utility* (*task-dependent*)

Autoencoders

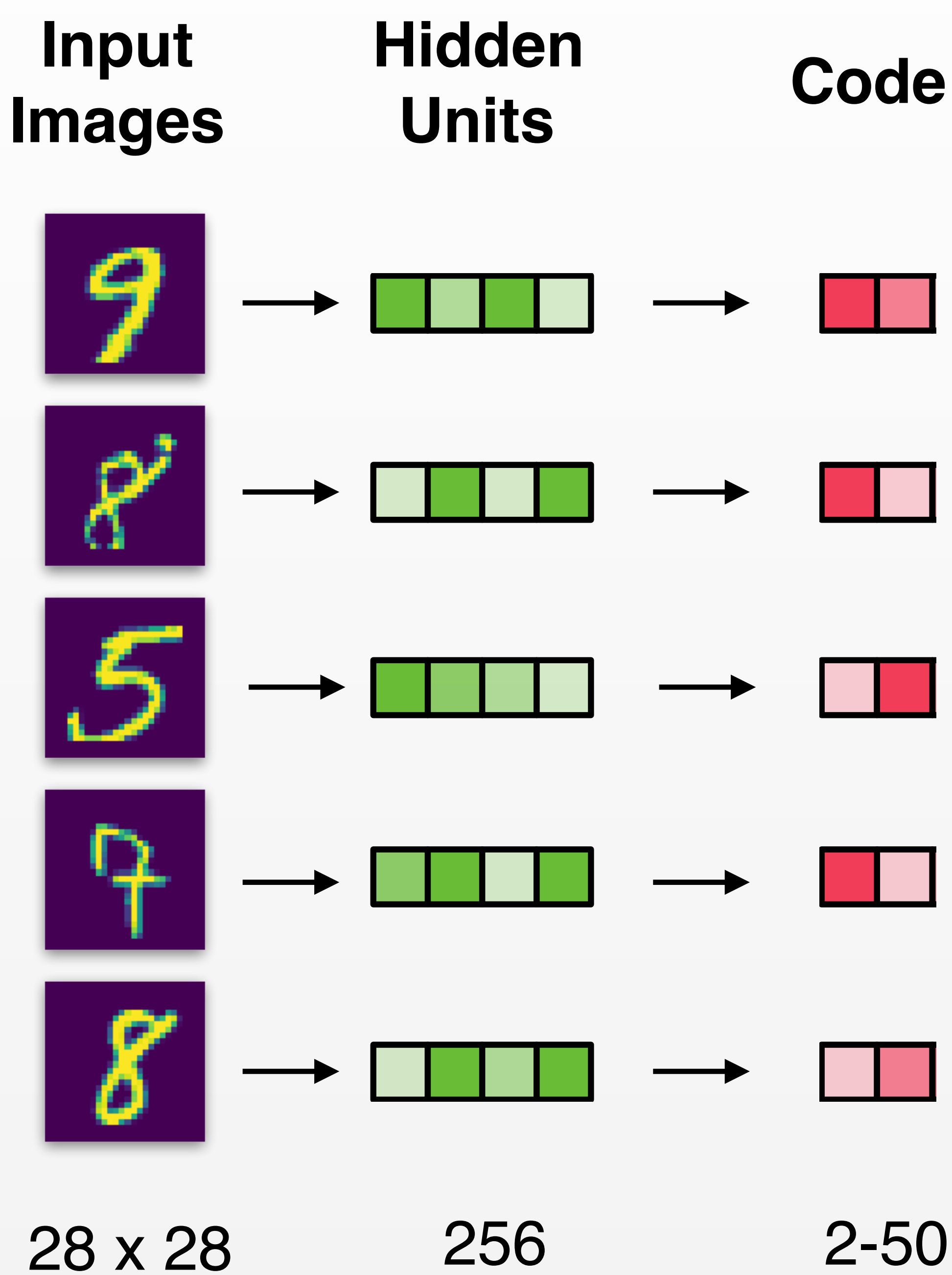


Autoencoders



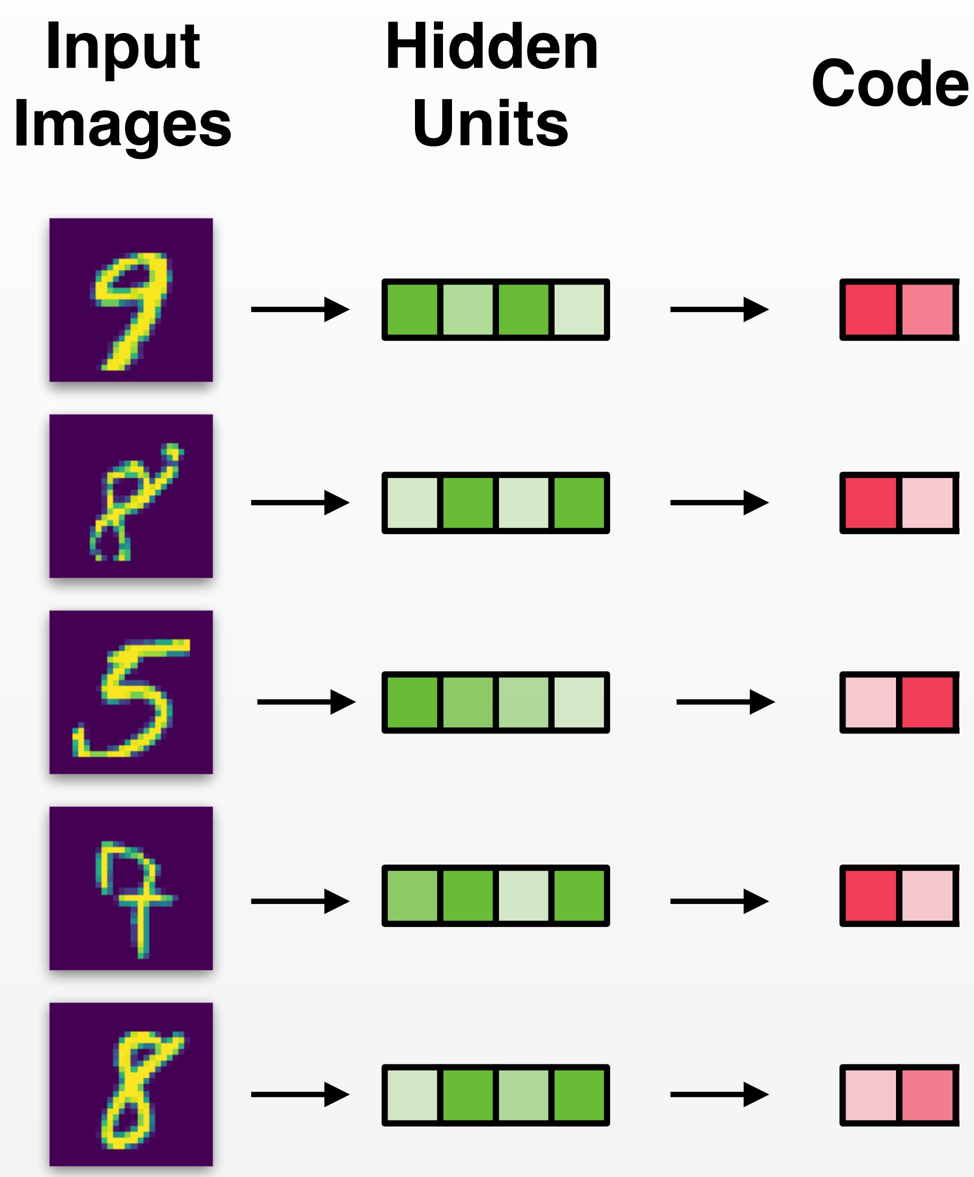
Notion of Utility: Ability to reconstruction of pixels from code (features are a *compressed* representation of original data)

Autoencoders

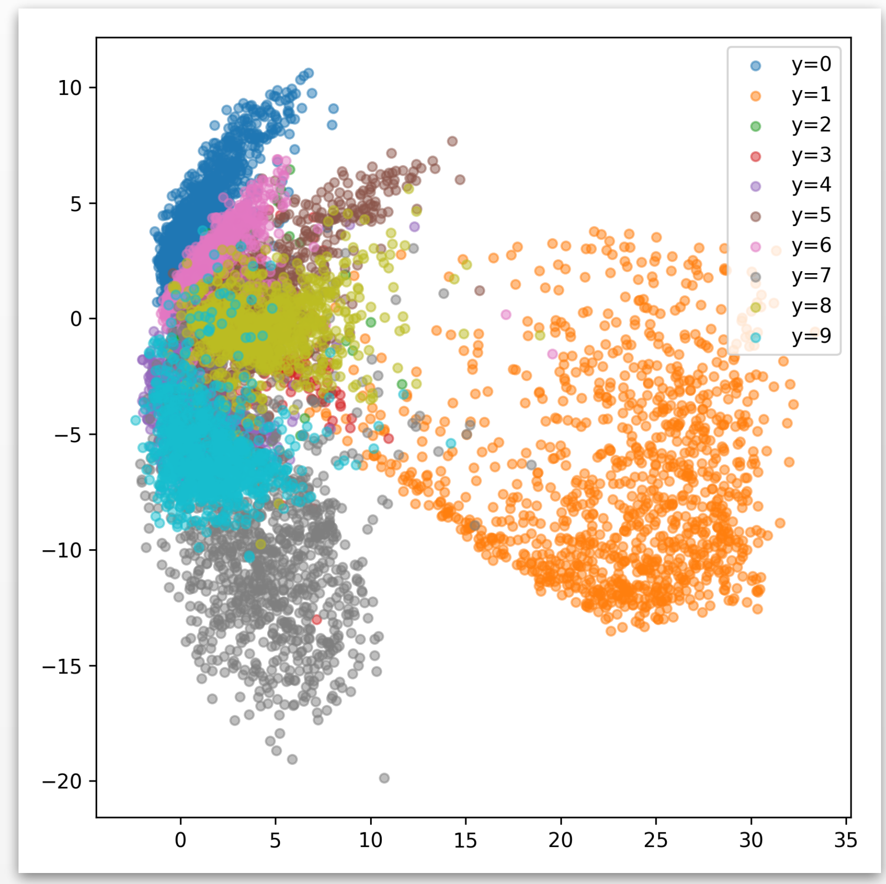


Notion of Utility: Ability to reconstruction of pixels from code
(features are a *compressed* representation of original data)

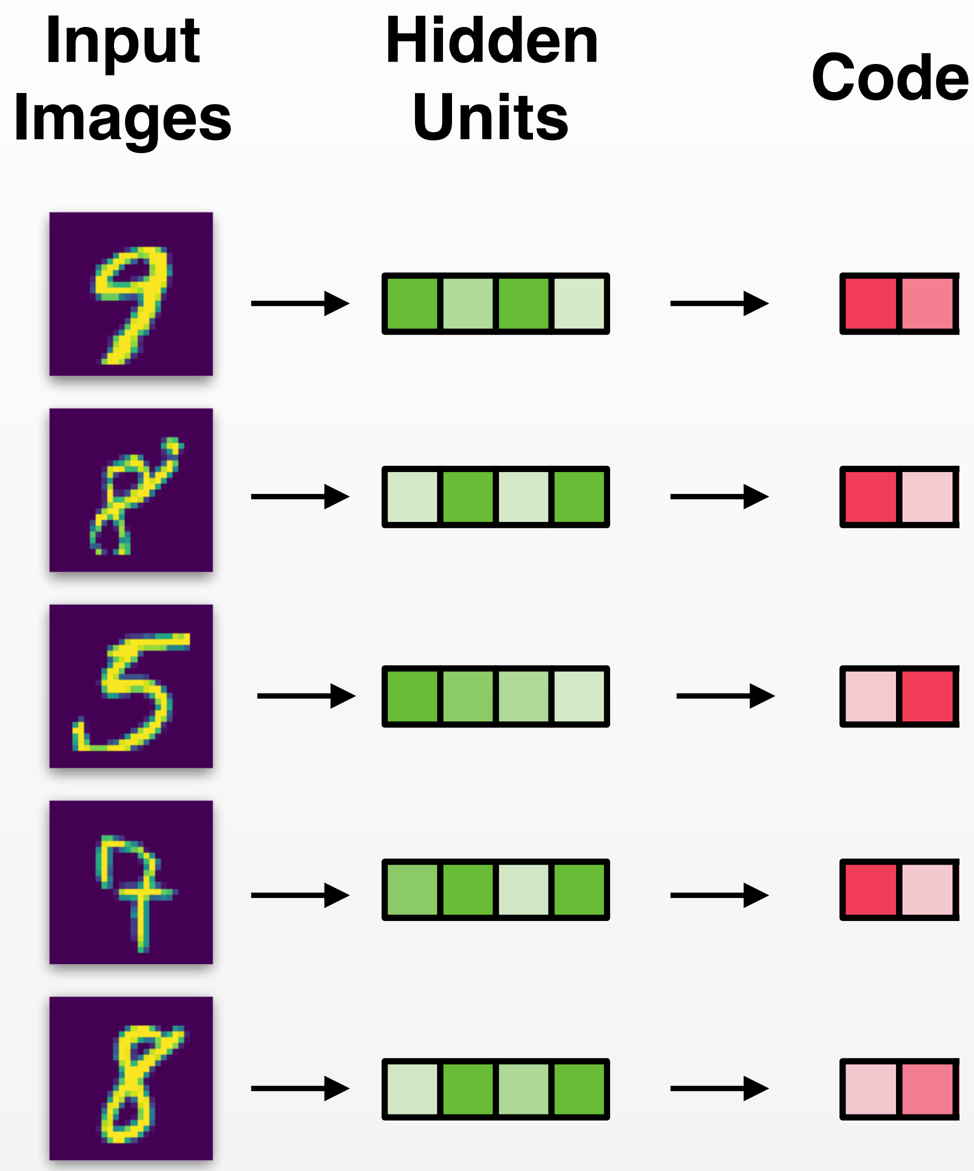
Autoencoders



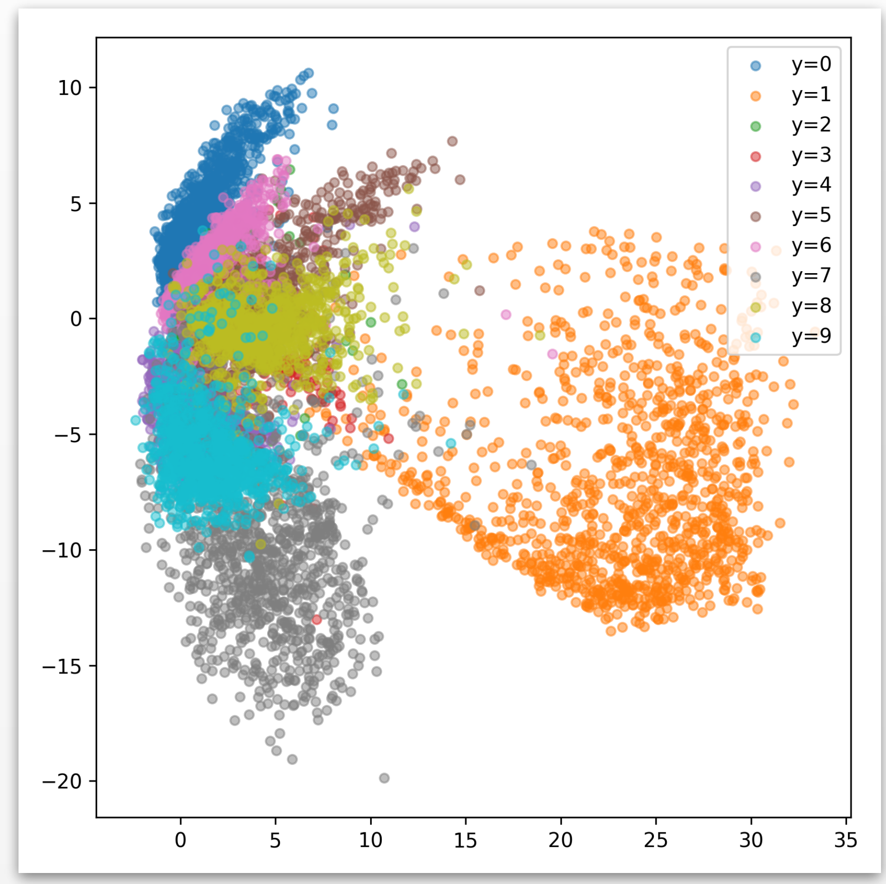
Learned Representation



Autoencoders



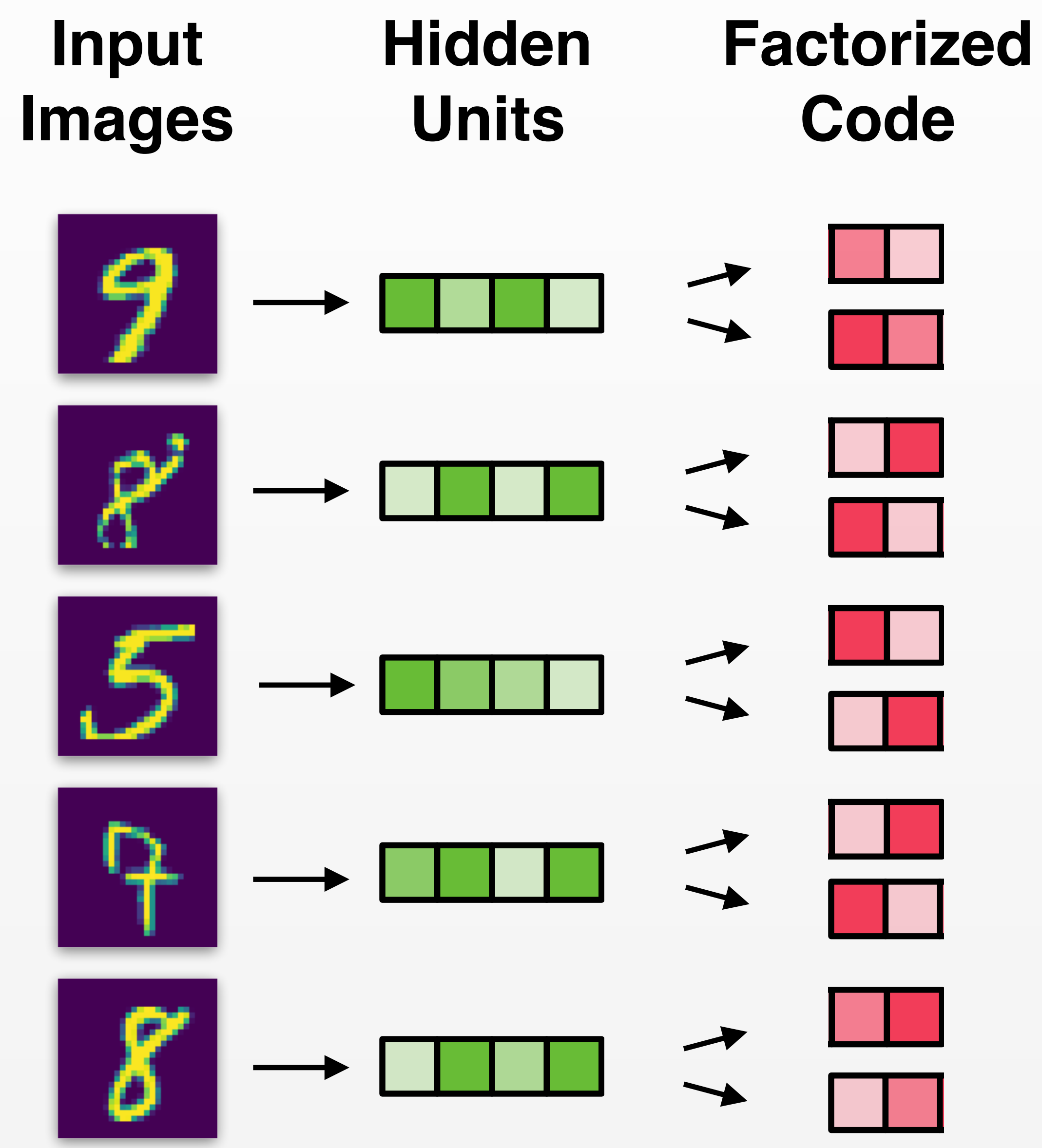
Learned Representation



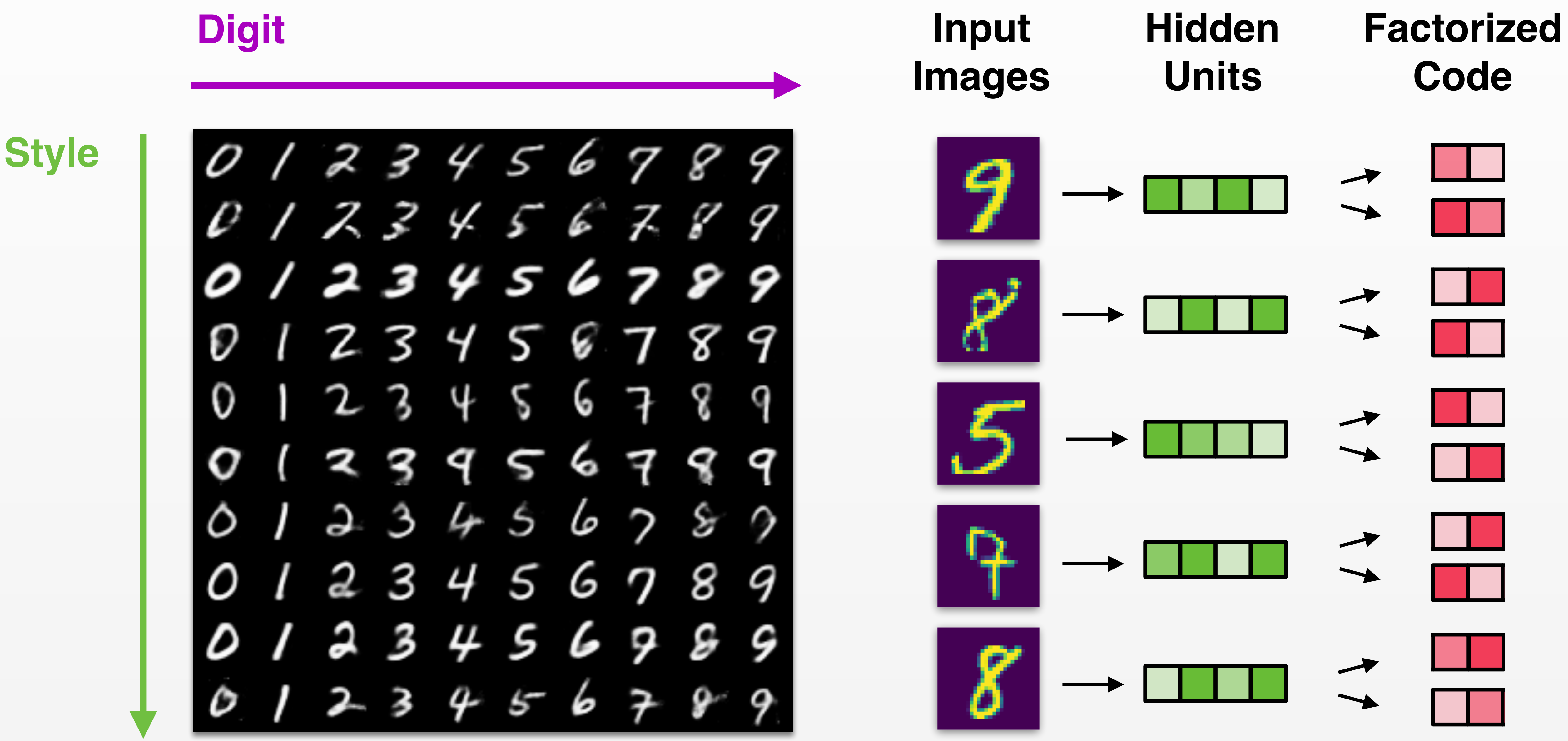
Entangled Representation: Individual dimensions in code encode some unknown combination of features in the data.

Disentangled Representations

Disentangled Representations

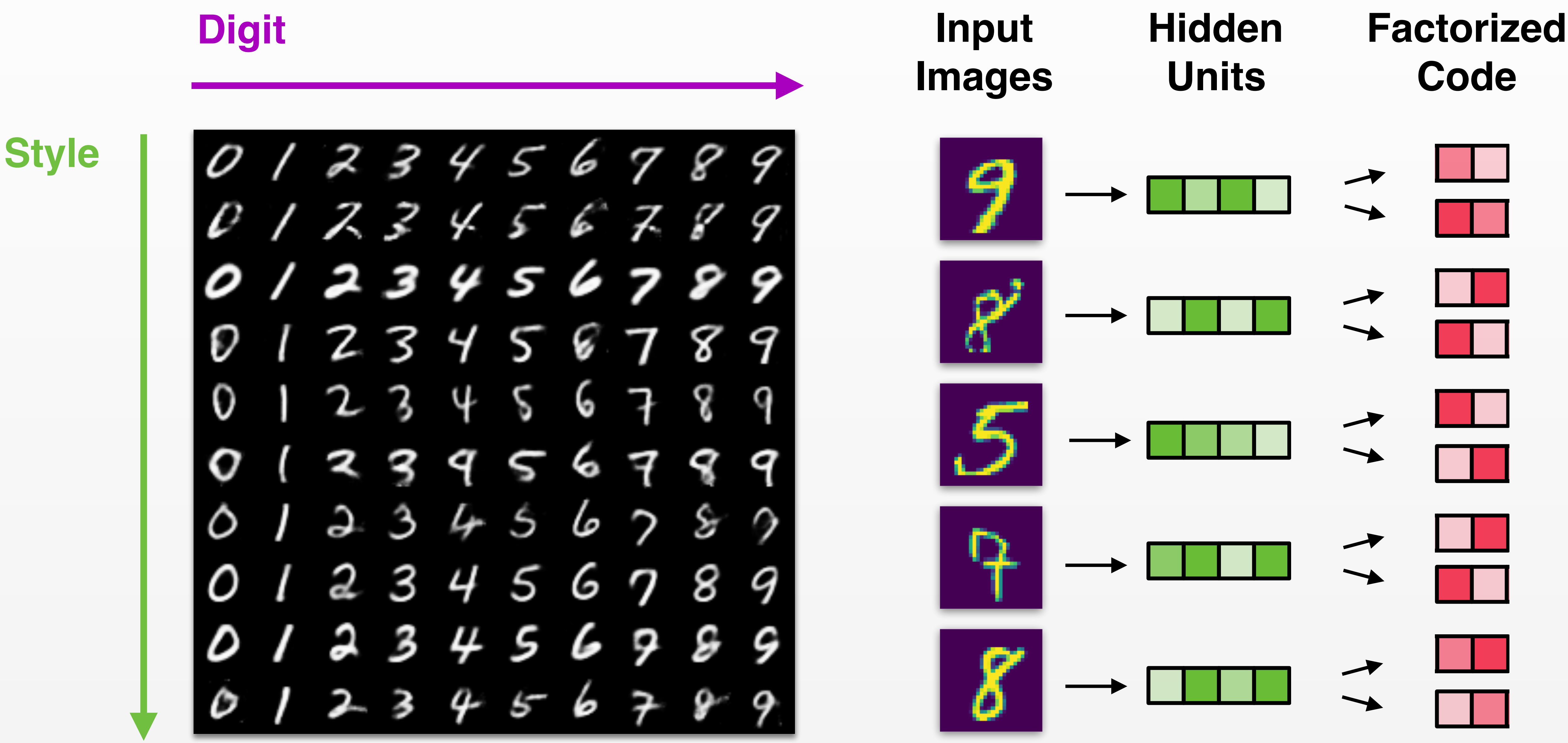


Disentangled Representations



Goal: Learn features that correspond to *distinct* factors of variation

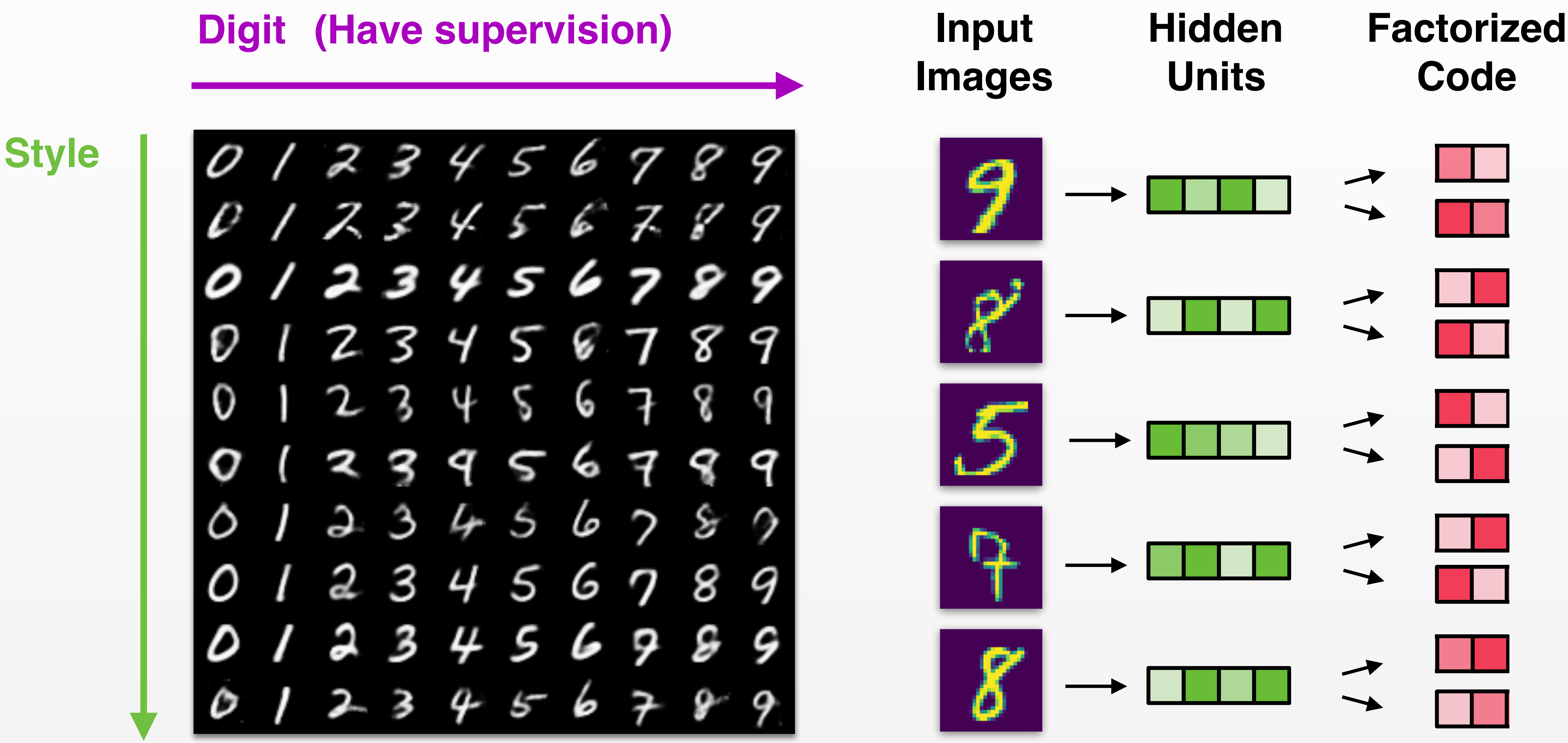
Disentangled Representations



Goal: Learn features that correspond to *distinct* factors of variation

One Notion of Utility: Statistical independence

Disentangled Representations



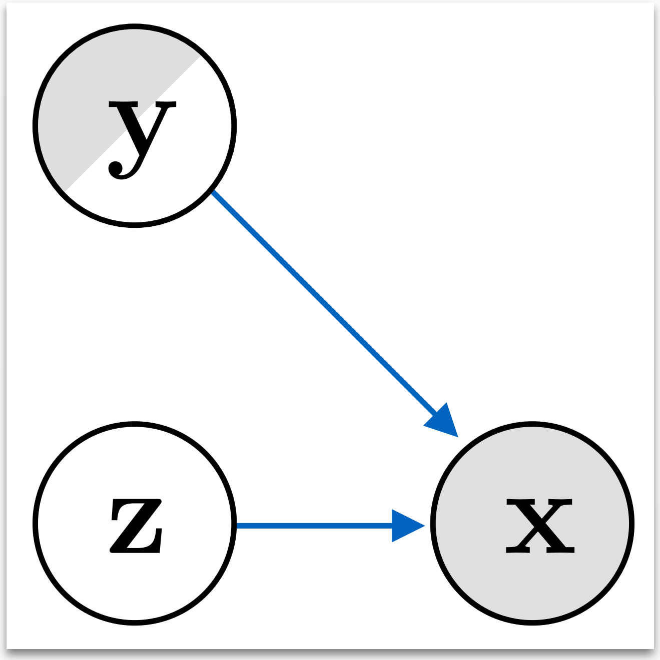
Goal: Learn features that correspond to *distinct* factors of variation

One Notion of Utility: Statistical independence

Semi-supervised Learning

Generative Model (Decoder)

$$p_\theta(\boldsymbol{x} \mid \boldsymbol{y}, \boldsymbol{z})$$



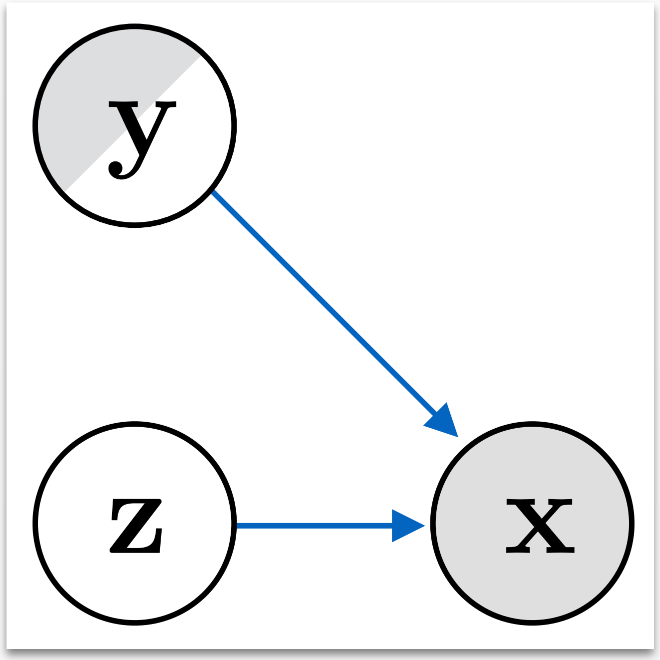
Assume independence between digit **y** and style **z**

[Kingma, Mohamed, Jimenez-Rezende, Welling, *Semi-supervised Learning with Deep Generative Models,* NIPS 2014]

Semi-supervised Learning

Generative Model (Decoder)

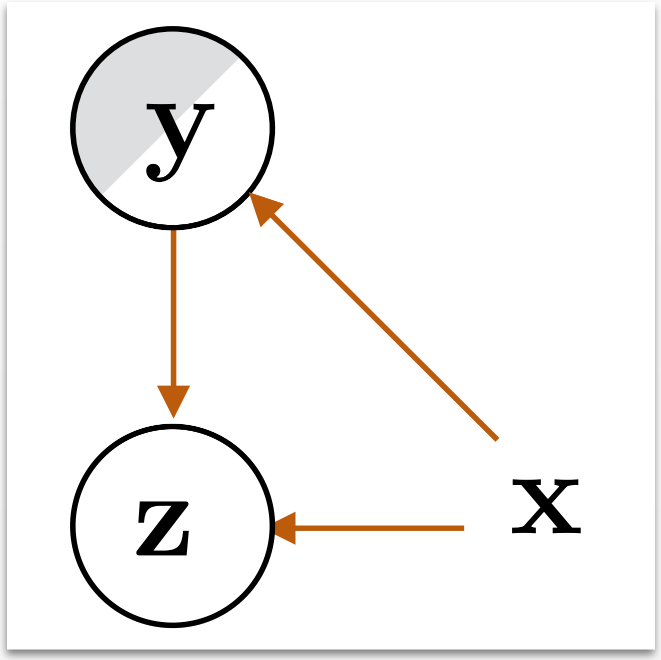
$p_\theta(\boldsymbol{x} \mid \boldsymbol{y}, \boldsymbol{z})$



Assume independence between digit **y** and style **z**

Inference Model (Encoder)

$q_\phi(\boldsymbol{y}, \boldsymbol{z} | \boldsymbol{x})$



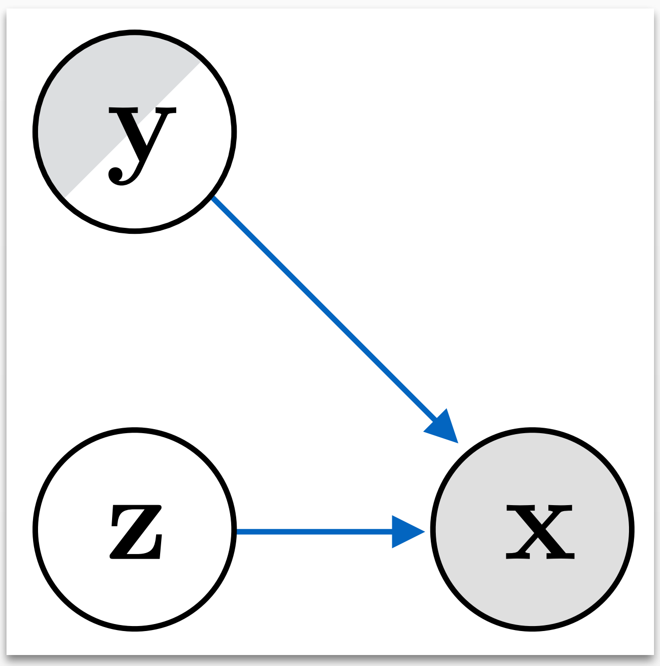
Infer **y** from pixels **x**, and **z** from **y** and **x**

[Kingma, Mohamed, Jimenez-Rezende, Welling, *Semi-supervised Learning with Deep Generative Models,* NIPS 2014]

Semi-supervised Learning

Generative Model (Decoder)

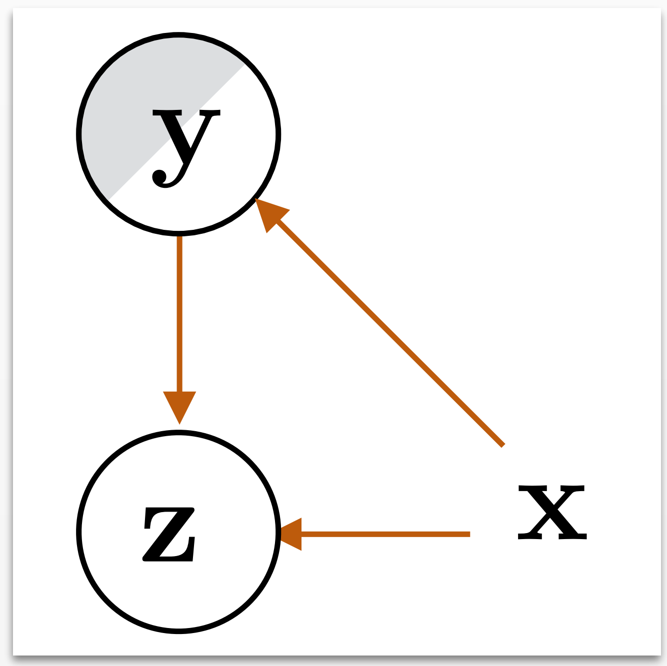
$p_\theta(\boldsymbol{x} \mid \boldsymbol{y}, \boldsymbol{z})$



Assume independence
between digit **y** and style **z**

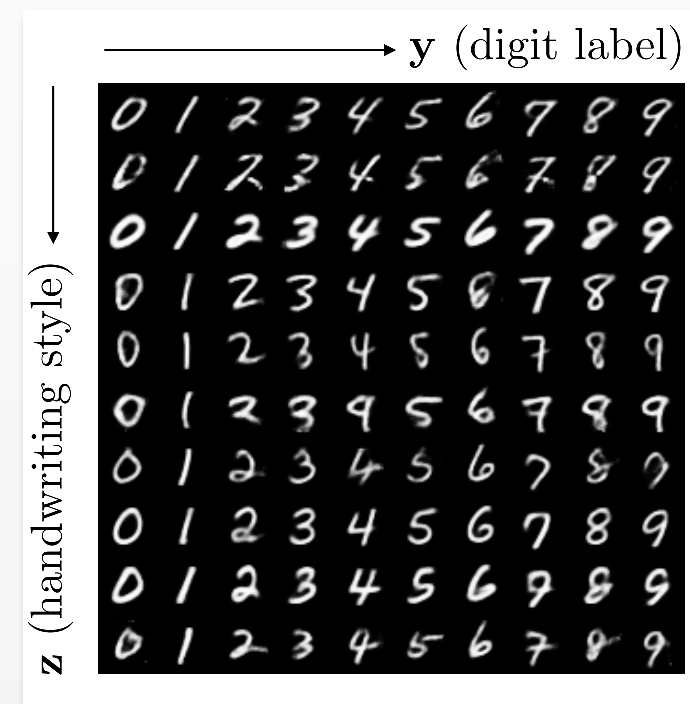
Inference Model (Encoder)

$q_\phi(\boldsymbol{y}, \boldsymbol{z} | \boldsymbol{x})$



Infer **y** from pixels **x**,
and **z** from **y** and **x**

Disentangled
Representation



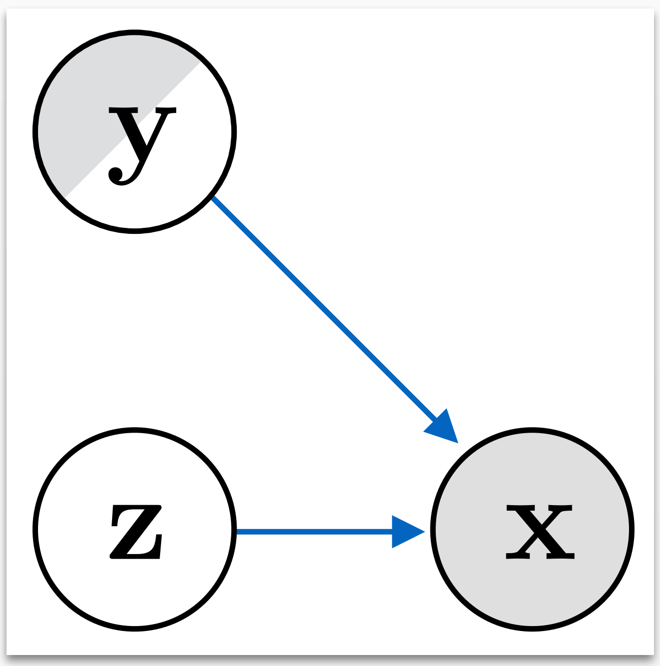
Separate interpretable **y**
from "nuisance" variables **z**

[Kingma, Mohamed, Jimenez-Rezende, Welling, *Semi-supervised Learning with Deep Generative Models,* NIPS 2014]

Semi-supervised Learning

Generative Model (Decoder)

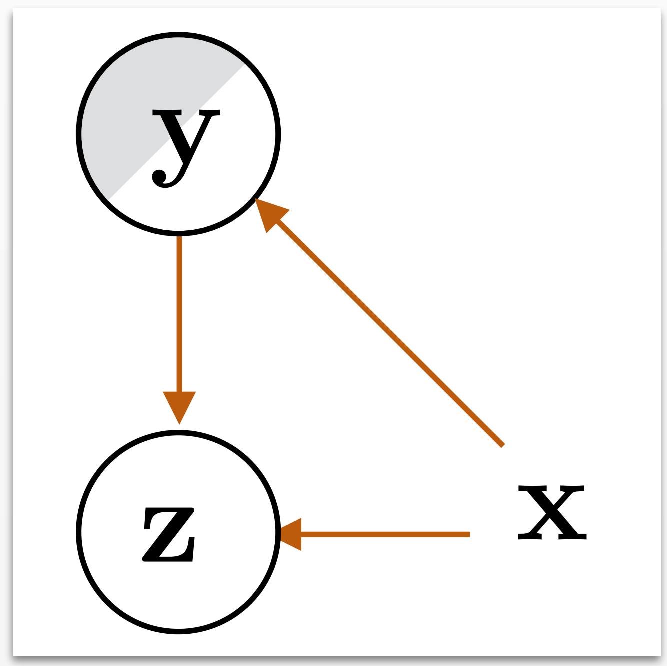
$p_\theta(\mathbf{x} \mid \mathbf{y}, \mathbf{z})$



Assume independence between digit **y** and style **z**

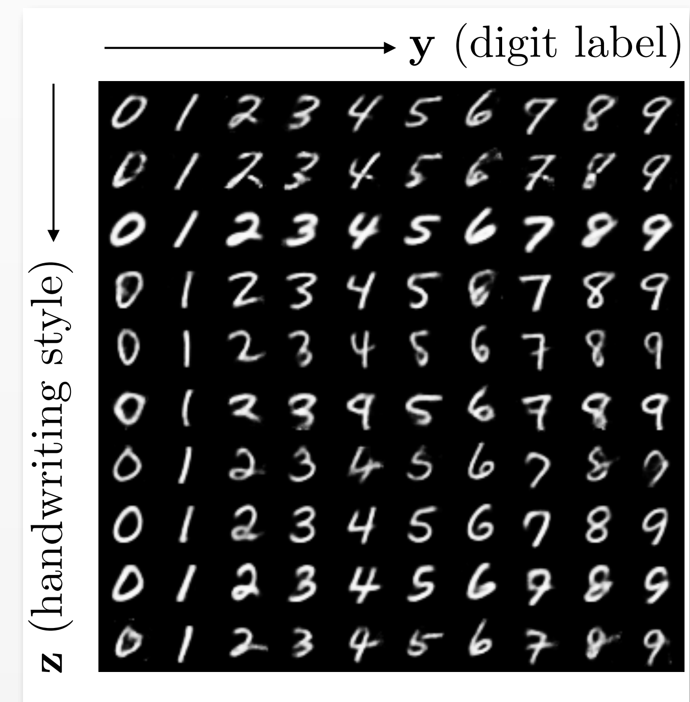
Inference Model (Encoder)

$q_\phi(\mathbf{y}, \mathbf{z} \mid \mathbf{x})$



Infer **y** from pixels **x**, and **z** from **y** and **x**

Disentangled Representation



Separate interpretable **y** from "nuisance" variables **z**

Hypothesis: Assuming a statistical independence under the prior induces disentangled representations.

[Kingma, Mohamed, Jimenez-Rezende, Welling, *Semi-supervised Learning with Deep Generative Models,* NIPS 2014]

Deep Probabilistic Programs

Inference Model (Encoder)

```
class Encoder(torch.nn.Module):
  def __init__(self, x_sz, h_sz, y_sz, z_sz):
    # intializes layers: h, y_log_weights, ...
    ...

  def forward(self, x, y_values=None):
    q = probtorch.Trace()
    h = self.h(x)
    y = q.concrete(
          self.y_log_weights(h), 0.66,
          value=y_values, name='y')
    hy = torch.cat([h, y], -1)
    z = q.normal(self.z_mean(hy),
                 self.z_std(hy),
                 name='z')
    return q
```

Generative Model (Decoder)

```
class Decoder(torch.nn.Module):
  def __init__(self, x_sz, h_sz, y_sz, z_sz):
    # intializes layers: h, x_mean, ...
    ...

  def forward(self, x, q):
    p = probtorch.Trace()
    y = p.concrete(self.y_log_weights, 0.66,
                   value=q['y'], name='y')
    z = p.normal(0.0, 1.0,
                 value=q['z'], name='z')
    h = self.h(torch.cat([y, z], -1))
    x = p.loss(self.bce,
               self.x_mean(h), x,
               name='x')
    return p
```

Probabilistic Torch

PROB
TORCH

https://github.com/probtorch/probtorch

Deep Probabilistic Programs

Inference Model (Encoder)

```
class Encoder(torch.nn.Module):
  def __init__(self, x_sz, h_sz, y_sz, z_sz):
    # intializes layers: h, y_log_weights, ...
    ...

  def forward(self, x, y_values=None):
    q = probtorch.Trace()
    h = self.h(x)
    y = q.concrete(
          self.y_log_weights(h), 0.66,
          value=y_values, name='y')
    hy = torch.cat([h, y], -1)
    z = q.normal(self.z_mean(hy),
                 self.z_std(hy),
                 name='z')
    return q
```

Generative Model (Decoder)

```
class Decoder(torch.nn.Module):
  def __init__(self, x_sz, h_sz, y_sz, z_sz):
    # intializes layers: h, x_mean, ...
    ...

  def forward(self, x, q):
    p = probtorch.Trace()
    y = p.concrete(self.y_log_weights, 0.66,
                   value=q['y'], name='y')
    z = p.normal(0.0, 1.0,
                 value=q['z'], name='z')
    h = self.h(torch.cat([y, z], -1))
    x = p.loss(self.bce,
               self.x_mean(h), x,
               name='x')
    return p
```

Probabilistic Torch

PROB TORCH

https://github.com/probtorch/probtorch

Deep Probabilistic Programs

Inference Model (Encoder)

```python
class Encoder(torch.nn.Module):
  def __init__(self, x_sz, h_sz, y_sz, z_sz):
    # intializes layers: h, y_log_weights, ...
    ...

  def forward(self, x, y_values=None):
    q = probtorch.Trace()
    h = self.h(x)
    y = q.concrete(
          self.y_log_weights(h), 0.66,
          value=y_values, name='y')
    hy = torch.cat([h, y], -1)
    z = q.normal(self.z_mean(hy),
                 self.z_std(hy),
                 name='z')
    return q
```

Generative Model (Decoder)

```python
class Decoder(torch.nn.Module):
  def __init__(self, x_sz, h_sz, y_sz, z_sz):
    # intializes layers: h, x_mean, ...
    ...

  def forward(self, x, q):
    p = probtorch.Trace()
    y = p.concrete(self.y_log_weights, 0.66,
                   value=q['y'], name='y')
    z = p.normal(0.0, 1.0,
                 value=q['z'], name='z')
    h = self.h(torch.cat([y, z], -1))
    x = p.loss(self.bce,
               self.x_mean(h), x,
               name='x')
    return p
```

Probabilistic Torch

PROB TORCH

https://github.com/probtorch/probtorch

Deep Probabilistic Programs

Inference Model (Encoder)

```
class Encoder(torch.nn.Module):
  def __init__(self, x_sz, h_sz, y_sz, z_sz):
    # intializes layers: h, y_log_weights, ...
    ...

  def forward(self, x, y_values=None):
    q = probtorch.Trace()
    h = self.h(x)
    y = q.concrete(
          self.y_log_weights(h), 0.66,
          value=y_values, name='y')
    hy = torch.cat([h, y], -1)
    z = q.normal(self.z_mean(hy),
                 self.z_std(hy),
                 name='z')
    return q
```

Generative Model (Decoder)

```
class Decoder(torch.nn.Module):
  def __init__(self, x_sz, h_sz, y_sz, z_sz):
    # intializes layers: h, x_mean, ...
    ...

  def forward(self, x, q):
    p = probtorch.Trace()
    y = p.concrete(self.y_log_weights, 0.66,
                   value=q['y'], name='y')
    z = p.normal(0.0, 1.0,
                 value=q['z'], name='z')
    h = self.h(torch.cat([y, z], -1))
    x = p.loss(self.bce,
               self.x_mean(h), x,
               name='x')
    return p
```

Probabilistic Torch

PROB TORCH

https://github.com/probtorch/probtorch

Deep Probabilistic Programs

Inference Model (Encoder)

```
class Encoder(torch.nn.Module):
  def __init__(self, x_sz, h_sz, y_sz, z_sz):
    # intializes layers: h, y_log_weights, ...
    ...

  def forward(self, x, y_values=None):
    q = probtorch.Trace()
    h = self.h(x)
    y = q.concrete(
          self.y_log_weights(h), 0.66,
          value=y_values, name='y')
    hy = torch.cat([h, y], -1)
    z = q.normal(self.z_mean(hy),
                 self.z_std(hy),
                 name='z')
    return q
```

Generative Model (Decoder)

```
class Decoder(torch.nn.Module):
  def __init__(self, x_sz, h_sz, y_sz, z_sz):
    # intializes layers: h, x_mean, ...
    ...

  def forward(self, x, q):
    p = probtorch.Trace()
    y = p.concrete(self.y_log_weights, 0.66,
                   value=q['y'], name='y')
    z = p.normal(0.0, 1.0,
                 value=q['z'], name='z')
    h = self.h(torch.cat([y, z], -1))
    x = p.loss(self.bce,
               self.x_mean(h), x,
               name='x')
    return p
```

Probabilistic Torch

PROB TORCH

https://github.com/probtorch/probtorch

Deep Probabilistic Programs

Inference Model (Encoder)

```
class Encoder(torch.nn.Module):
  def __init__(self, x_sz, h_sz, y_sz, z_sz):
    # intializes layers: h, y_log_weights, ...
    ...

  def forward(self, x, y_values=None):
    q = probtorch.Trace()
    h = self.h(x)
    y = q.concrete(
          self.y_log_weights(h), 0.66,
          value=y_values, name='y')
    hy = torch.cat([h, y], -1)
    z = q.normal(self.z_mean(hy),
                 self.z_std(hy),
                 name='z')
    return q
```

Generative Model (Decoder)

```
class Decoder(torch.nn.Module):
  def __init__(self, x_sz, h_sz, y_sz, z_sz):
    # intializes layers: h, x_mean, ...
    ...

  def forward(self, x, q):
    p = probtorch.Trace()
    y = p.concrete(self.y_log_weights, 0.66,
                   value=q['y'], name='y')
    z = p.normal(0.0, 1.0,
                 value=q['z'], name='z')
    h = self.h(torch.cat([y, z], -1))
    x = p.loss(self.bce,
               self.x_mean(h), x,
               name='x')
    return p
```

Probabilistic Torch

PROB TORCH

https://github.com/probtorch/probtorch

Deep Probabilistic Programs

Inference Model (Encoder)

```
class Encoder(torch.nn.Module):
  def __init__(self, x_sz, h_sz, y_sz, z_sz):
    # intializes layers: h, y_log_weights, ...
    ...

  def forward(self, x, y_values=None):
    q = probtorch.Trace()
    h = self.h(x)
    y = q.concrete(
          self.y_log_weights(h), 0.66,
          value=y_values, name='y')
    hy = torch.cat([h, y], -1)
    z = q.normal(self.z_mean(hy),
                 self.z_std(hy),
                 name='z')
    return q
```

Generative Model (Decoder)

```
class Decoder(torch.nn.Module):
  def __init__(self, x_sz, h_sz, y_sz, z_sz):
    # intializes layers: h, x_mean, ...
    ...

  def forward(self, x, q):
    p = probtorch.Trace()
    y = p.concrete(self.y_log_weights, 0.66,
                   value=q['y'], name='y')
    z = p.normal(0.0, 1.0,
                 value=q['z'], name='z')
    h = self.h(torch.cat([y, z], -1))
    x = p.loss(self.bce,
               self.x_mean(h), x,
               name='x')
    return p
```

Probabilistic Torch

PROB TORCH

https://github.com/probtorch/probtorch

Deep Probabilistic Programs

Inference Model (Encoder)

```
class Encoder(torch.nn.Module):
  def __init__(self, x_sz, h_sz, y_sz, z_sz):
    # intializes layers: h, y_log_weights, ...
    ...

  def forward(self, x, y_values=None):
    q = probtorch.Trace()
    h = self.h(x)
    y = q.concrete(
          self.y_log_weights(h), 0.66,
          value=y_values, name='y')
    hy = torch.cat([h, y], -1)
    z = q.normal(self.z_mean(hy),
                 self.z_std(hy),
                 name='z')
    return q
```

Generative Model (Decoder)

```
class Decoder(torch.nn.Module):
  def __init__(self, x_sz, h_sz, y_sz, z_sz):
    # intializes layers: h, x_mean, ...
    ...

  def forward(self, x, q):
    p = probtorch.Trace()
    y = p.concrete(self.y_log_weights, 0.66,
                   value=q['y'], name='y')
    z = p.normal(0.0, 1.0,
                 value=q['z'], name='z')
    h = self.h(torch.cat([y, z], -1))
    x = p.loss(self.bce,
               self.x_mean(h), x,
               name='x')
    return p
```

Probabilistic Torch

PROB
TORCH

https://github.com/probtorch/probtorch

Deep Probabilistic Programs

Inference Model (Encoder)

```python
class Encoder(torch.nn.Module):
  def __init__(self, x_sz, h_sz, y_sz, z_sz):
    # intializes layers: h, y_log_weights, ...
    ...

  def forward(self, x, y_values=None):
    q = probtorch.Trace()
    h = self.h(x)
    y = q.concrete(
          self.y_log_weights(h), 0.66,
          value=y_values, name='y')
    hy = torch.cat([h, y], -1)
    z = q.normal(self.z_mean(hy),
                 self.z_std(hy),
                 name='z')
    return q
```

Generative Model (Decoder)

```python
class Decoder(torch.nn.Module):
  def __init__(self, x_sz, h_sz, y_sz, z_sz):
    # intializes layers: h, x_mean, ...
    ...

  def forward(self, x, q):
    p = probtorch.Trace()
    y = p.concrete(self.y_log_weights, 0.66,
                   value=q['y'], name='y')
    z = p.normal(0.0, 1.0,
                 value=q['z'], name='z')
    h = self.h(torch.cat([y, z], -1))
    x = p.loss(self.bce,
               self.x_mean(h), x,
               name='x')
    return p
```

Probabilistic Torch

PROB TORCH

https://github.com/probtorch/probtorch

Deep Probabilistic Programs

Inference Model (Encoder)

```
class Encoder(torch.nn.Module):
  def __init__(self, x_sz, h_sz, y_sz, z_sz):
    # intializes layers: h, y_log_weights, ...
    ...

  def forward(self, x, y_values=None):
    q = probtorch.Trace()
    h = self.h(x)
    y = q.concrete(
          self.y_log_weights(h), 0.66,
          value=y_values, name='y')
    hy = torch.cat([h, y], -1)
    z = q.normal(self.z_mean(hy),
                 self.z_std(hy),
                 name='z')
    return q
```

Generative Model (Decoder)

```
class Decoder(torch.nn.Module):
  def __init__(self, x_sz, h_sz, y_sz, z_sz):
    # intializes layers: h, x_mean, ...
    ...

  def forward(self, x, q):
    p = probtorch.Trace()
    y = p.concrete(self.y_log_weights, 0.66,
                   value=q['y'], name='y')
    z = p.normal(0.0, 1.0,
                 value=q['z'], name='z')
    h = self.h(torch.cat([y, z], -1))
    x = p.loss(self.bce,
               self.x_mean(h), x,
               name='x')
    return p
```

Probabilistic Torch

PROB TORCH

https://github.com/probtorch/probtorch

Deep Probabilistic Programs

Inference Model (Encoder)

```python
class Encoder(torch.nn.Module):
  def __init__(self, x_sz, h_sz, y_sz, z_sz):
    # intializes layers: h, y_log_weights, ...
    ...

  def forward(self, x, y_values=None):
    q = probtorch.Trace()
    h = self.h(x)
    y = q.concrete(
          self.y_log_weights(h), 0.66,
          value=y_values, name='y')
    hy = torch.cat([h, y], -1)
    z = q.normal(self.z_mean(hy),
                 self.z_std(hy),
                 name='z')
    return q
```

Generative Model (Decoder)

```python
class Decoder(torch.nn.Module):
  def __init__(self, x_sz, h_sz, y_sz, z_sz):
    # intializes layers: h, x_mean, ...
    ...

  def forward(self, x, q):
    p = probtorch.Trace()
    y = p.concrete(self.y_log_weights, 0.66,
                   value=q['y'], name='y')
    z = p.normal(0.0, 1.0,
                 value=q['z'], name='z')
    h = self.h(torch.cat([y, z], -1))
    x = p.loss(self.bce,
               self.x_mean(h), x,
               name='x')
    return p
```

Probabilistic Torch

PROB TORCH

https://github.com/probtorch/probtorch

Deep Probabilistic Programs

Inference Model (Encoder)

```
class Encoder(torch.nn.Module):
  def __init__(self, x_sz, h_sz, y_sz, z_sz):
    # intializes layers: h, y_log_weights, ...
    ...

  def forward(self, x, y_values=None):
    q = probtorch.Trace()
    h = self.h(x)
    y = q.concrete(
          self.y_log_weights(h), 0.66,
          value=y_values, name='y')
    hy = torch.cat([h, y], -1)
    z = q.normal(self.z_mean(hy),
                 self.z_std(hy),
                 name='z')
    return q
```

Generative Model (Decoder)

```
class Decoder(torch.nn.Module):
  def __init__(self, x_sz, h_sz, y_sz, z_sz):
    # intializes layers: h, x_mean, ...
    ...

  def forward(self, x, q):
    p = probtorch.Trace()
    y = p.concrete(self.y_log_weights, 0.66,
                   value=q['y'], name='y')
    z = p.normal(0.0, 1.0,
                 value=q['z'], name='z')
    h = self.h(torch.cat([y, z], -1))
    x = p.loss(self.bce,
               self.x_mean(h), x,
               name='x')
    return p
```

Edward

https://github.com/bleilab/edward

Probabilistic Torch

https://github.com/probtorch/probtorch

Deep Probabilistic Programs

Inference Model (Encoder)

```
class Encoder(torch.nn.Module):
  def __init__(self, x_sz, h_sz, y_sz, z_sz):
    # intializes layers: h, y_log_weights, ...
    ...

  def forward(self, x, y_values=None):
    q = probtorch.Trace()
    h = self.h(x)
    y = q.concrete(
          self.y_log_weights(h), 0.66,
          value=y_values, name='y')
    hy = torch.cat([h, y], -1)
    z = q.normal(self.z_mean(hy),
                 self.z_std(hy),
                 name='z')
    return q
```

Generative Model (Decoder)

```
class Decoder(torch.nn.Module):
  def __init__(self, x_sz, h_sz, y_sz, z_sz):
    # intializes layers: h, x_mean, ...
    ...

  def forward(self, x, q):
    p = probtorch.Trace()
    y = p.concrete(self.y_log_weights, 0.66,
                   value=q['y'], name='y')
    z = p.normal(0.0, 1.0,
                 value=q['z'], name='z')
    h = self.h(torch.cat([y, z], -1))
    x = p.loss(self.bce,
               self.x_mean(h), x,
               name='x')
    return p
```

Edward

https://github.com/bleilab/edward

Probabilistic Torch

PROB TORCH

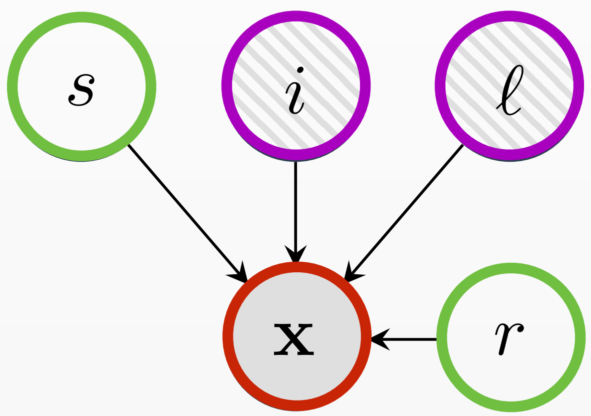
https://github.com/probtorch/probtorch

Pyro

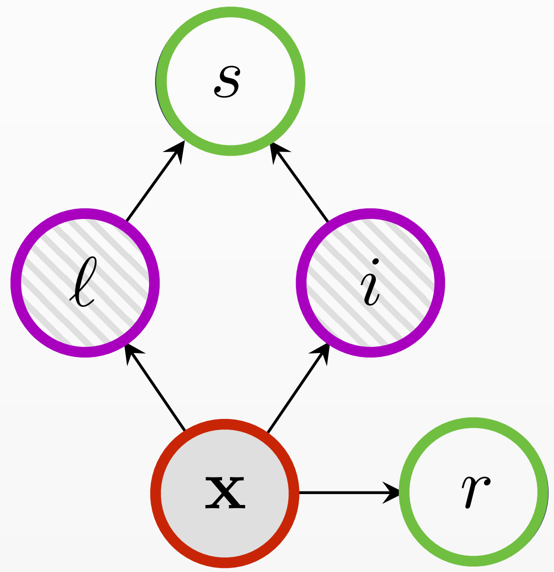
https://github.com/uber/pyro

Example: Yale B Faces

Generative Model



Inference Model



Semi-supervised:
Identity (i), Lighting (l)

Unsupervised:
Shading (s), Reflectance (r)

Data:
Pixels (x)

Original

Reconstruction

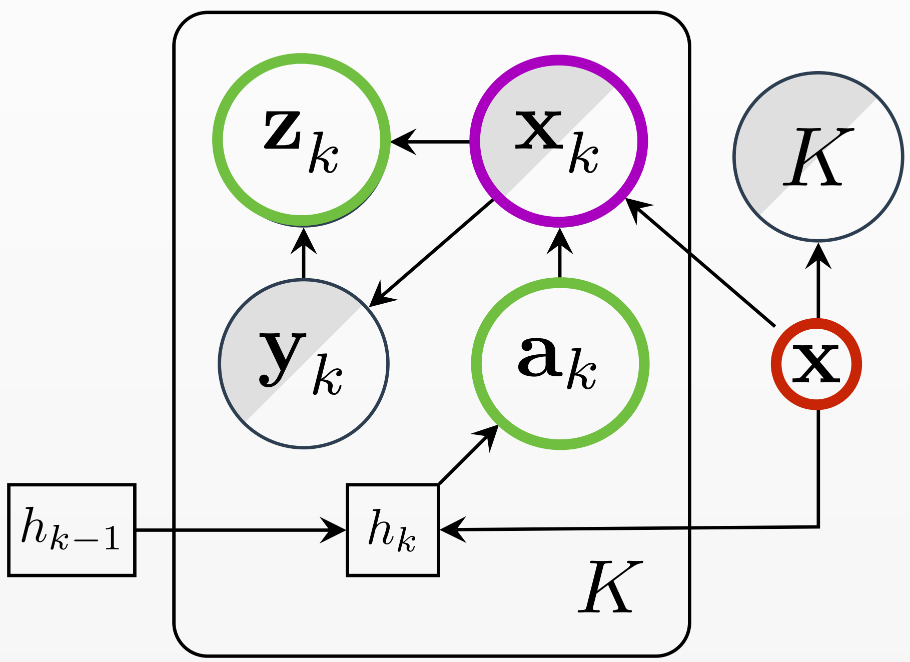
Same Lighting, Different Identity

Same Identity, Different Lighting

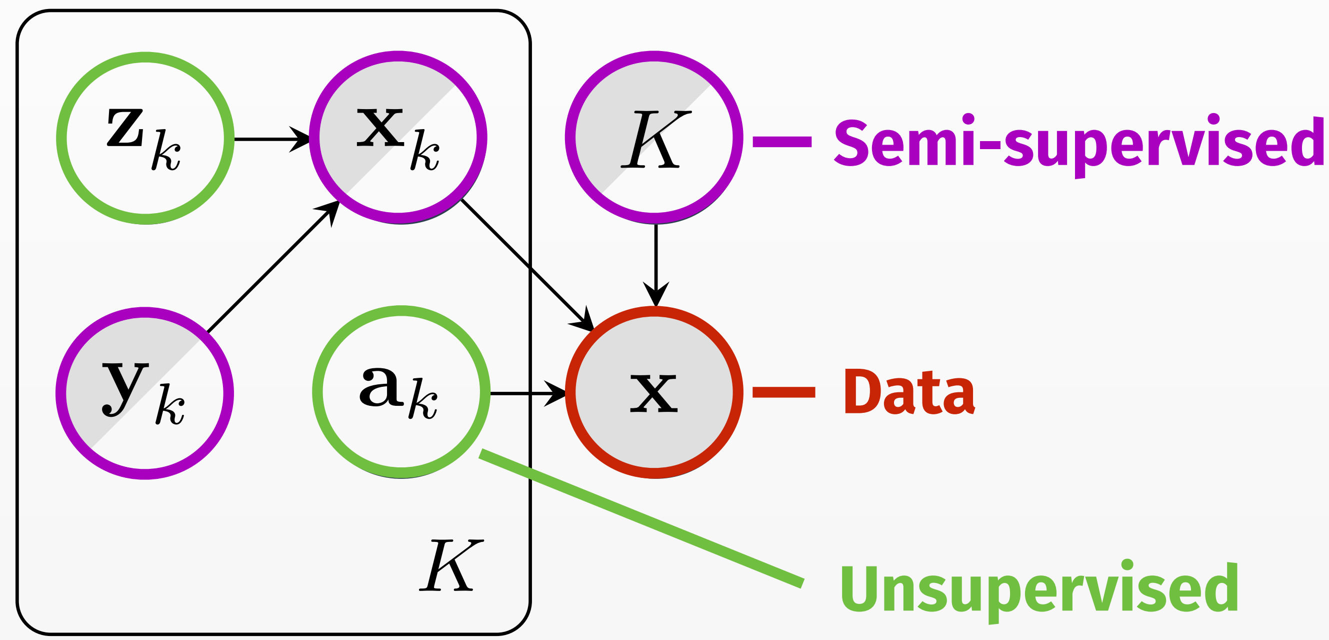
[Siddharth, Paige, van de Meent, Desmaison, Wood, Goodman, Kohli, Torr, NIPS 2017]

Example: Multiple MNIST Digits

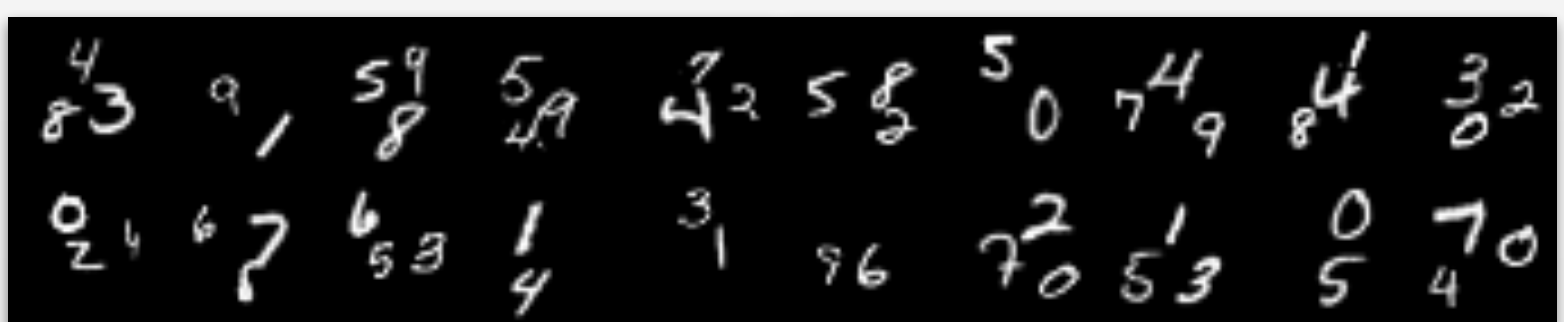
Inference Model



Generative Model



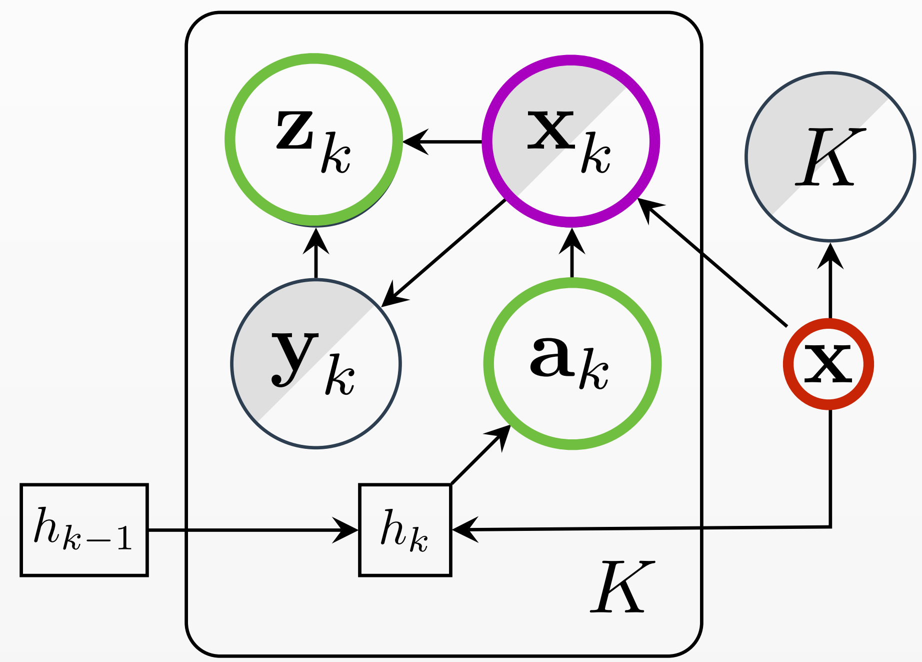
Multiple MNIST digits



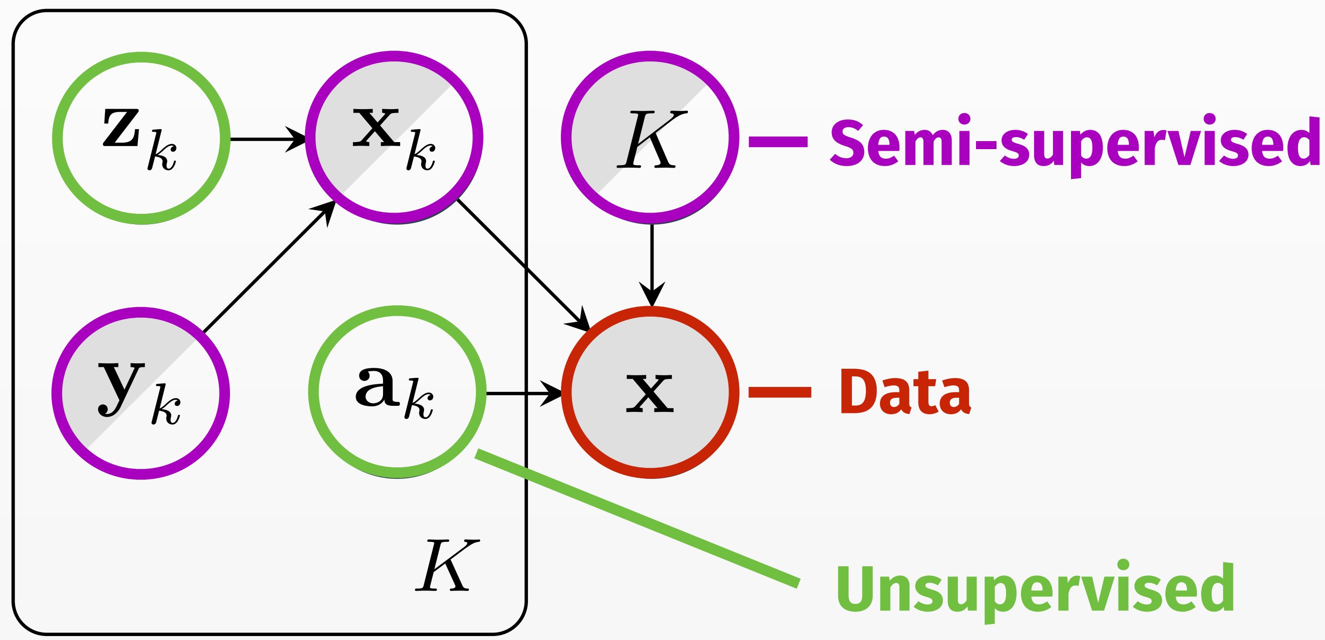
[Siddharth, Paige, van de Meent, Desmaison, Wood, Goodman, Kohli, Torr, NIPS 2017]

Example: Multiple MNIST Digits

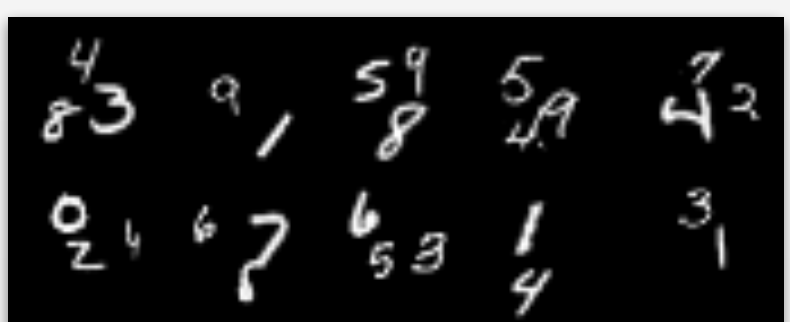
Encoder



Decoder



Data



Decomposition

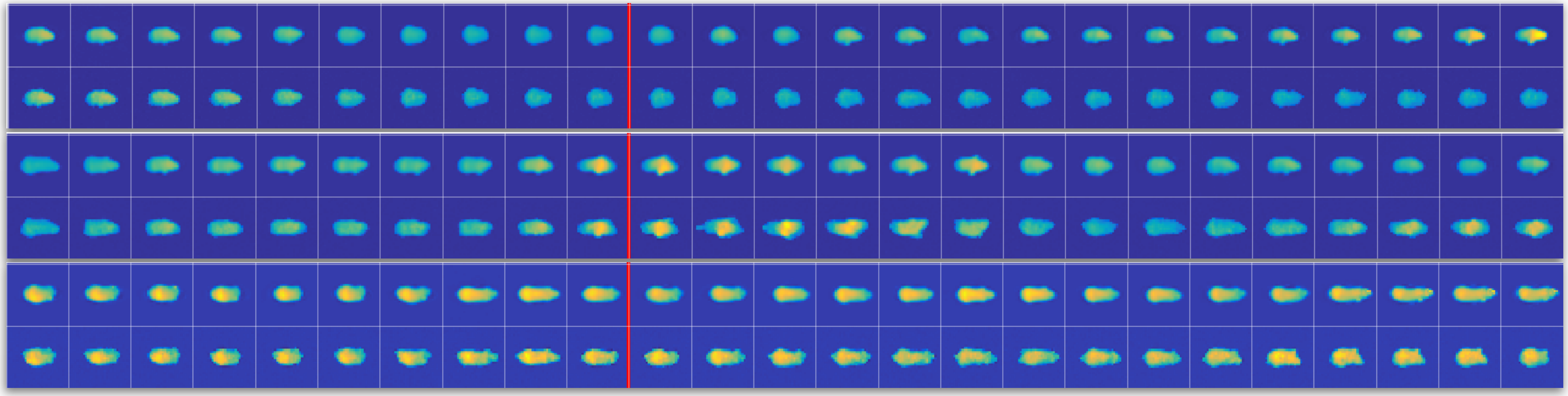
Accuracy

$\frac{M}{M+N}$	Count Error (%)	
	w/o MNIST	w/ MNIST
0.1	85.45 ($\pm$ 5.77)	76.33 ($\pm$ 8.91)
0.5	93.27 ($\pm$ 2.15)	80.27 ($\pm$ 5.45)
1.0	99.81 ($\pm$ 1.81)	84.79 ($\pm$ 5.11)

[Siddharth, Paige, van de Meent, Desmaison, Wood, Goodman, Kohli, Torr, NIPS 2017]

Deep Probabilistic Models

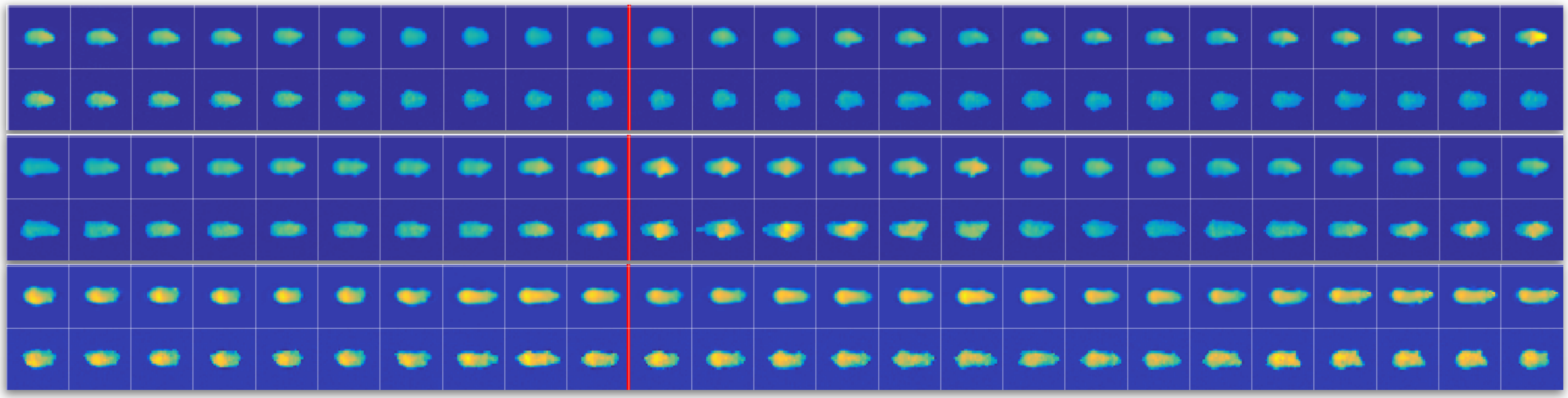
Example: Modeling Laboratory Mice with Deep State Space Models



[Johnson, Duvenaud, Wiltschko, Adams, Datta, NIPS 2016]

Deep Probabilistic Models

Example: Modeling Laboratory Mice with Deep State Space Models



[Johnson, Duvenaud, Wiltschko, Adams, Datta, NIPS 2016]

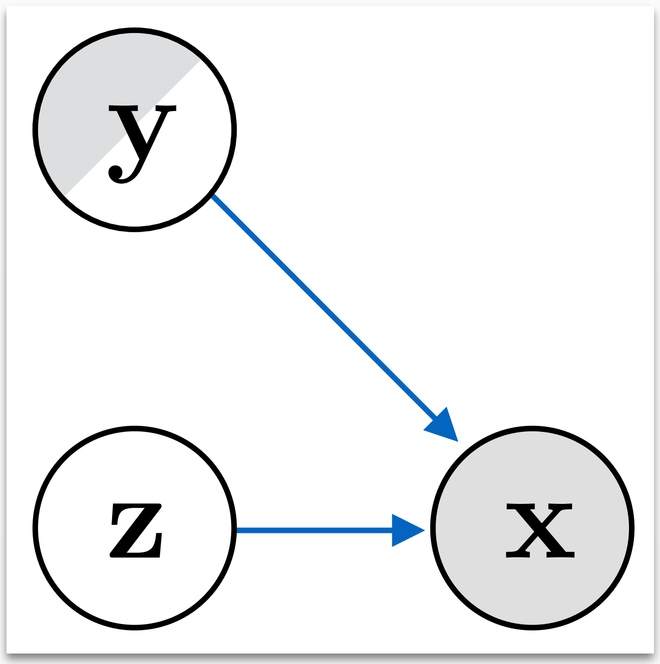
Data Science ❤️ Probabilistic Modeling + Deep Learning?

- Structured priors model problem domain
- Bayesian inference for uncertainty estimates
- Neural likelihood models for data such as text and images
- Neural inference models predict values for latent variables

Today: Unsupervised Learning

Generative Model (Decoder)

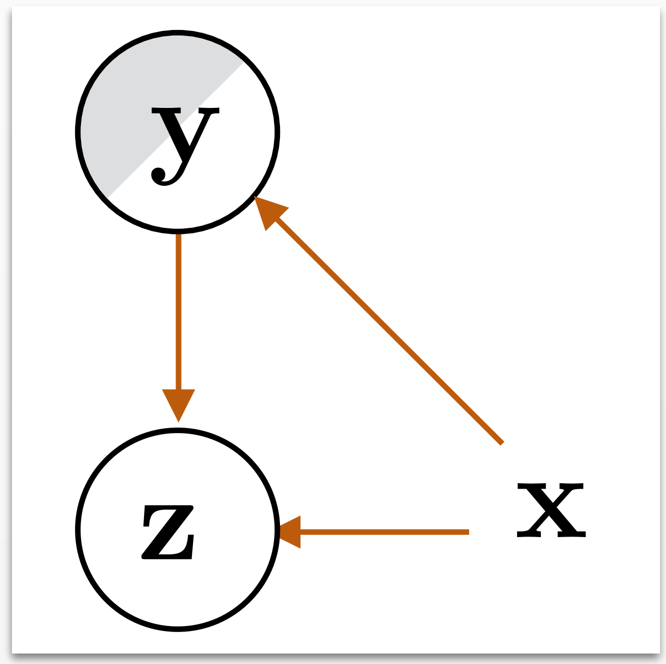
$p_\theta(\boldsymbol{x} \mid \boldsymbol{y}, \boldsymbol{z})$



Assume independence between digit **y** and style **z**

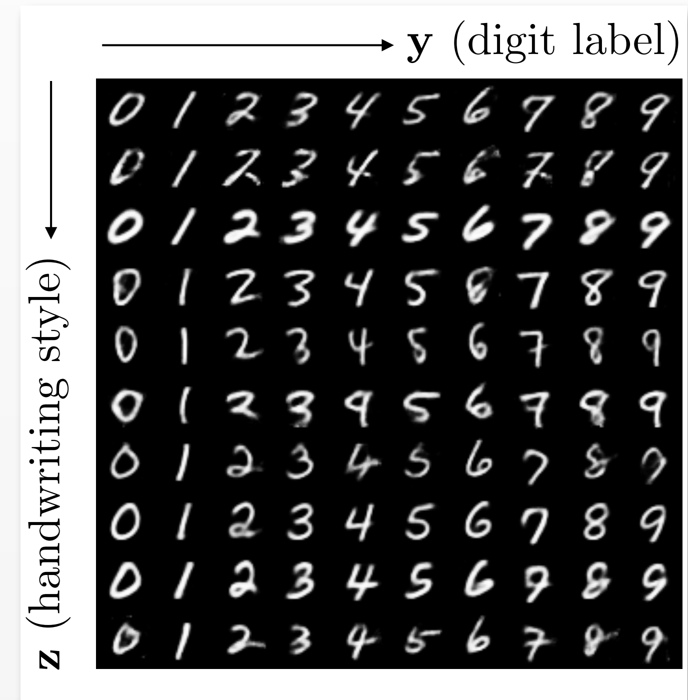
Inference Model (Encoder)

$q_\phi(\boldsymbol{y}, \boldsymbol{z} \mid \boldsymbol{x})$



Infer **y** from pixels **x**, and **z** from **y** and **x**

Disentangled Representation



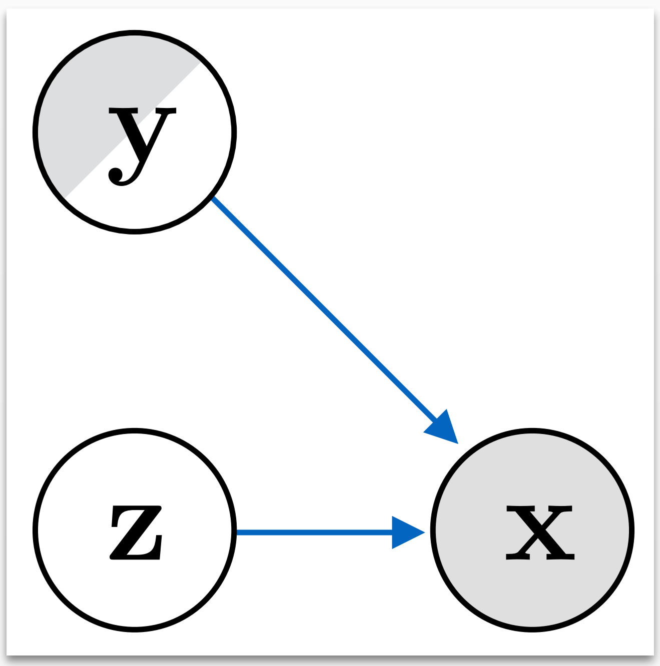
Separate interpretable **y** from "nuisance" variables **z**

Hypothesis: Assuming a statistical independence under the prior induces disentangled representations.

Today: Unsupervised Learning

Generative Model (Decoder)

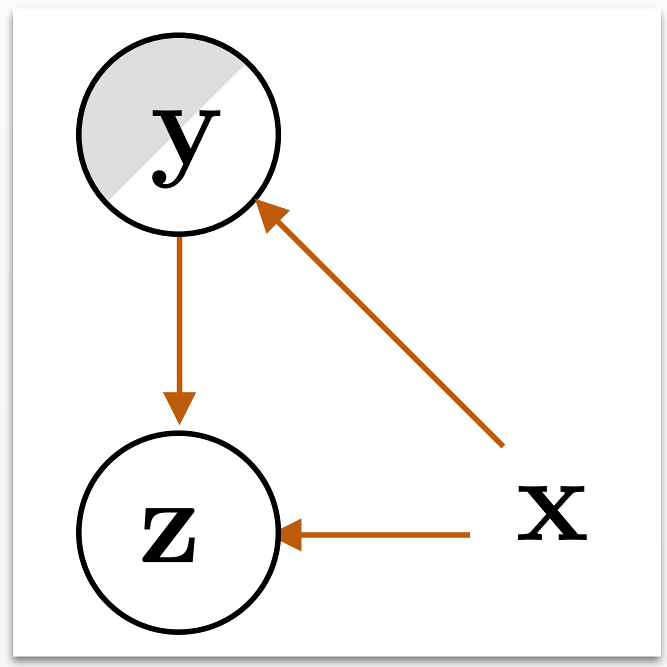
$p_\theta(\boldsymbol{x} \,|\, \boldsymbol{y}, \boldsymbol{z})$



Assume independence between digit **y** and style **z**

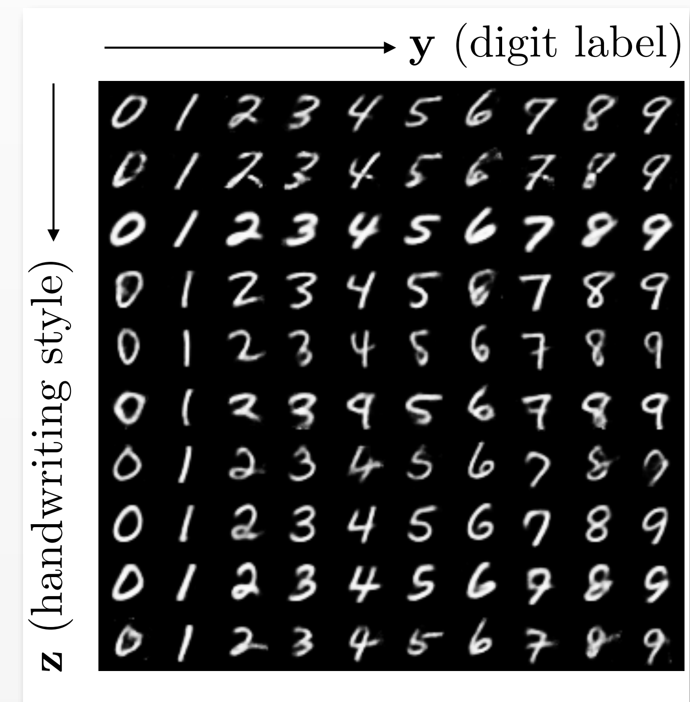
Inference Model (Encoder)

$q_\phi(\boldsymbol{y}, \boldsymbol{z} | \boldsymbol{x})$



Infer **y** from pixels **x**, and **z** from **y** and **x**

Disentangled Representation



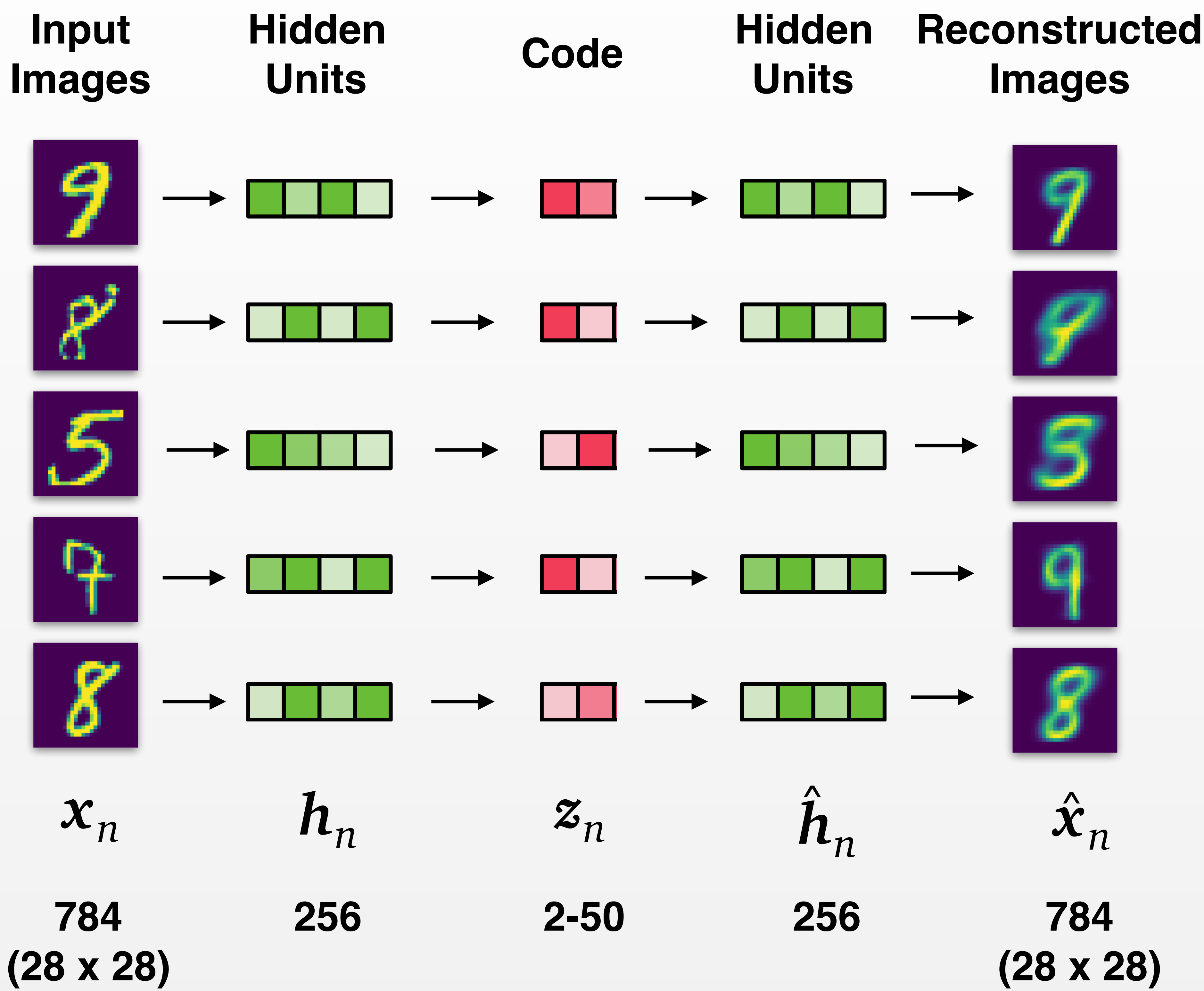
Separate interpretable **y** from “nuisance” variables **z**

Problem: Unsupervised learning (with same model) does *not* disentangle digit from style

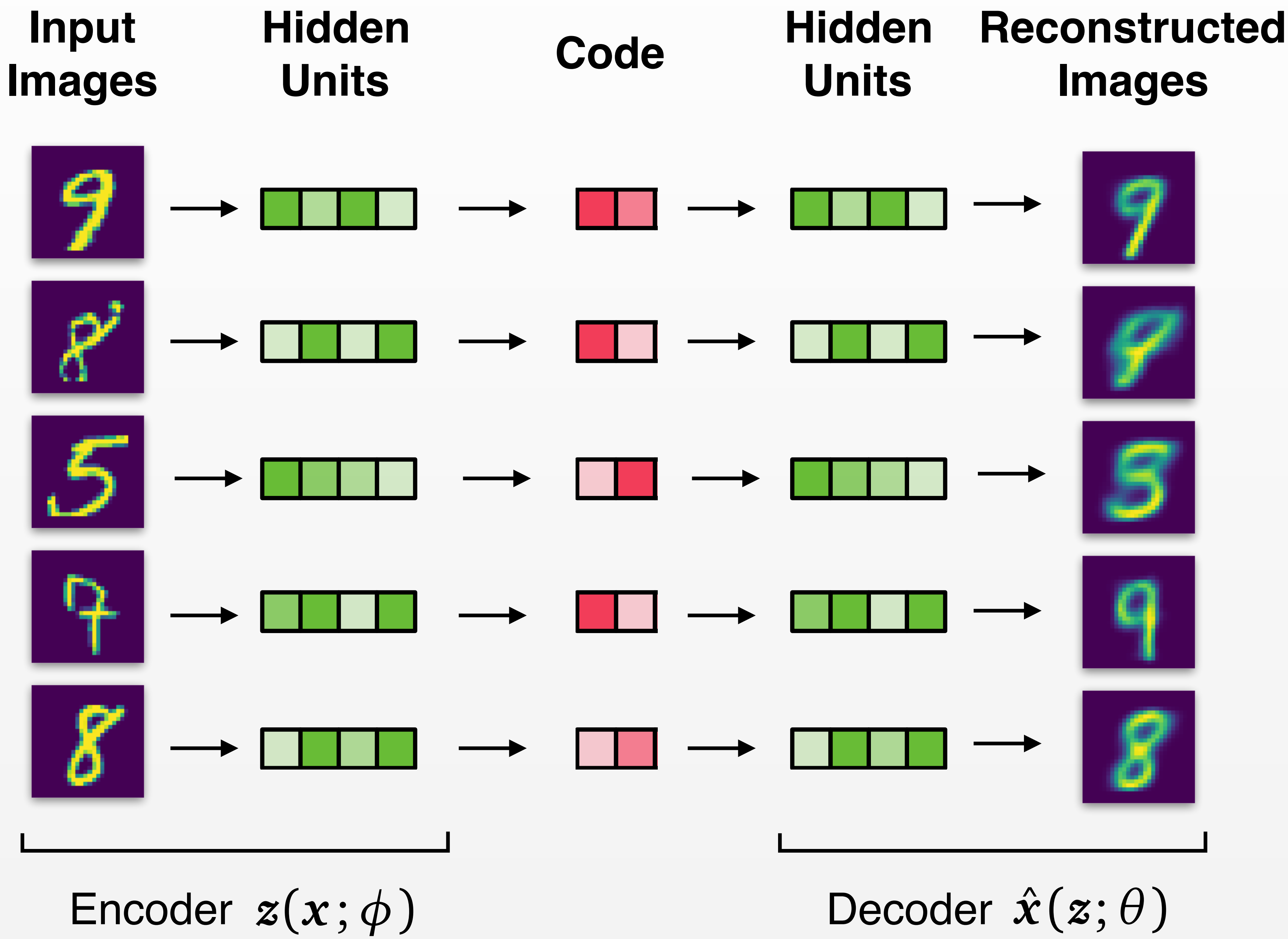
Variational Autoencoders

(a.k.a. Deep Latent-Variable Models)

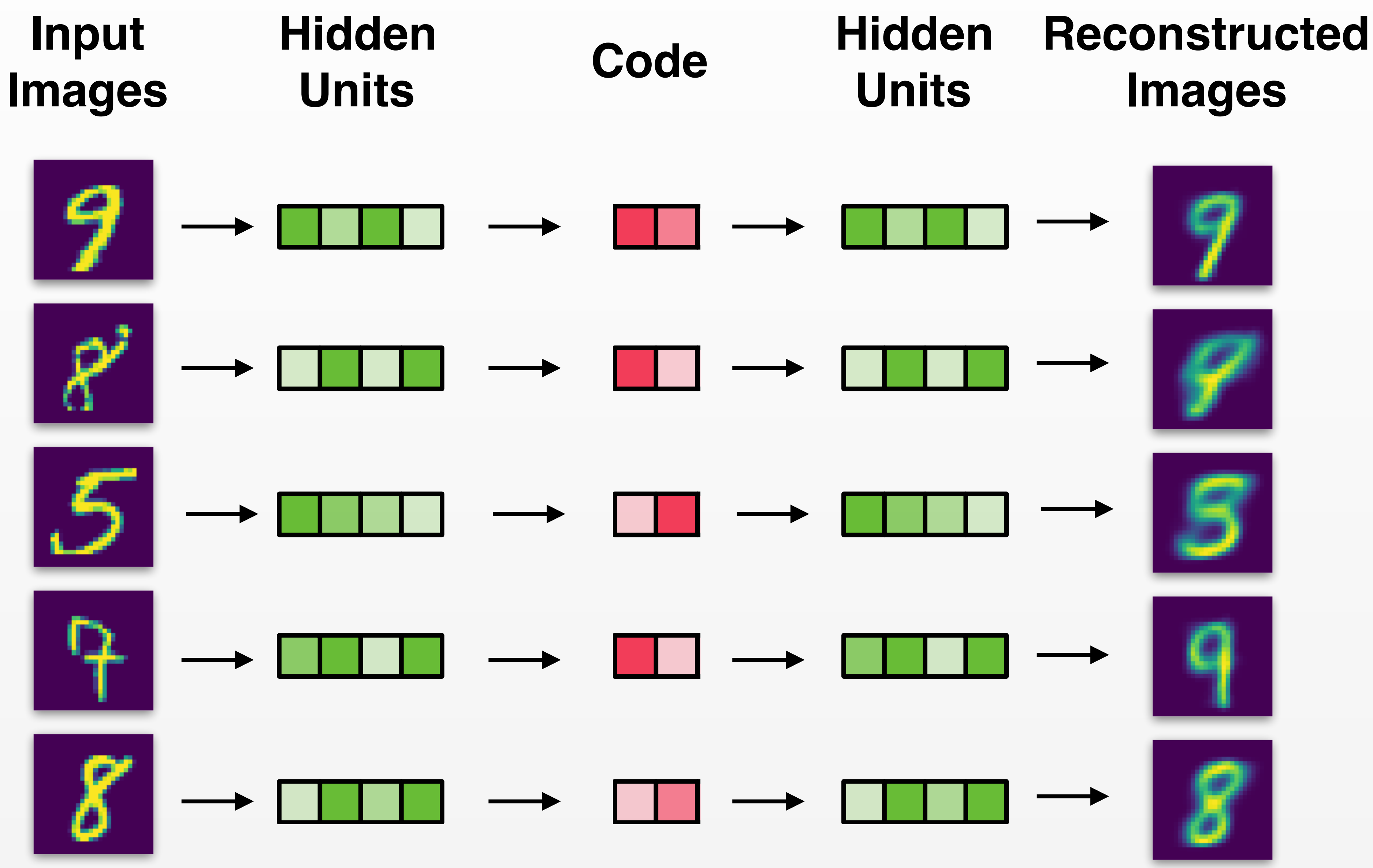
Autoencoders



Autoencoders



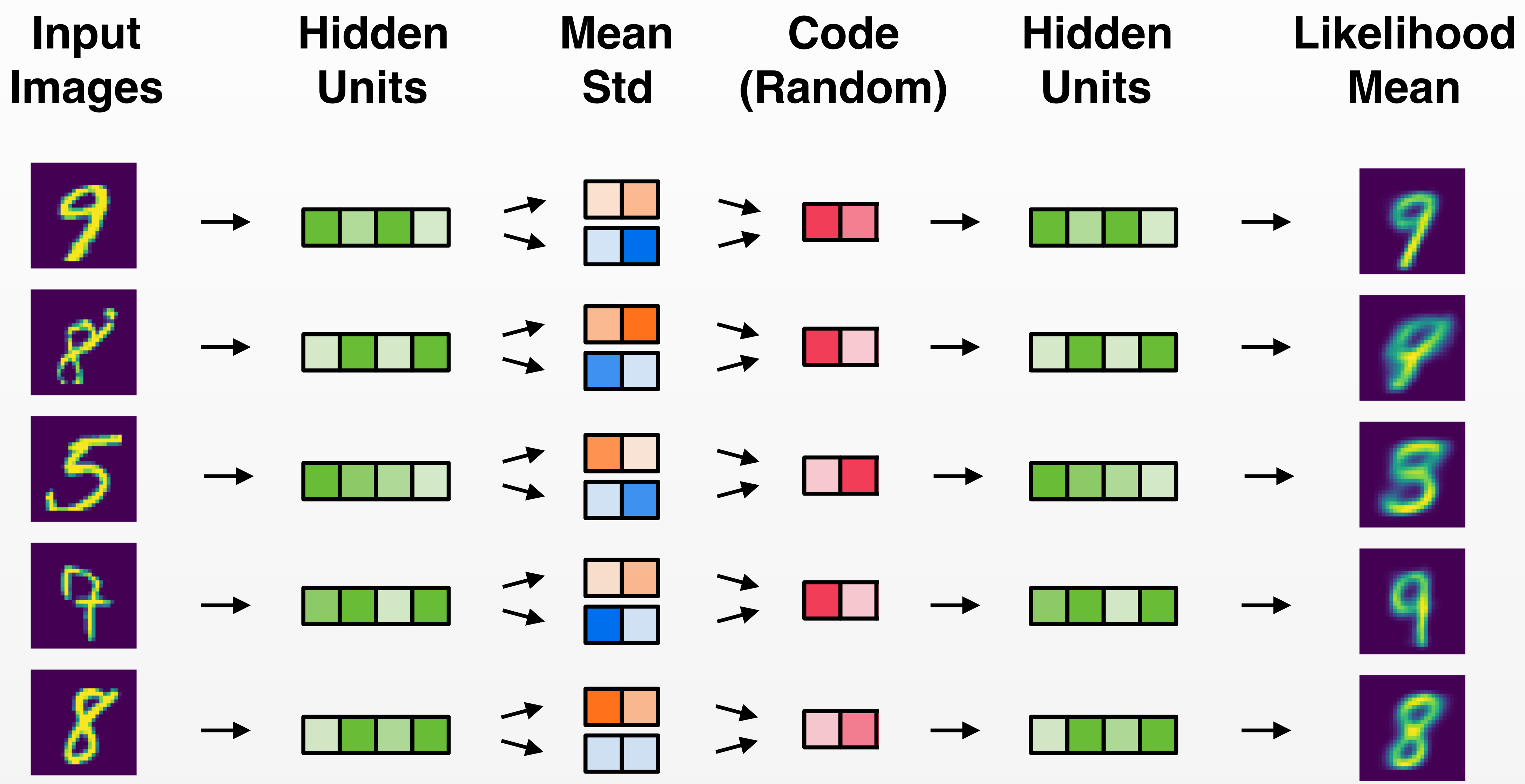
Autoencoders



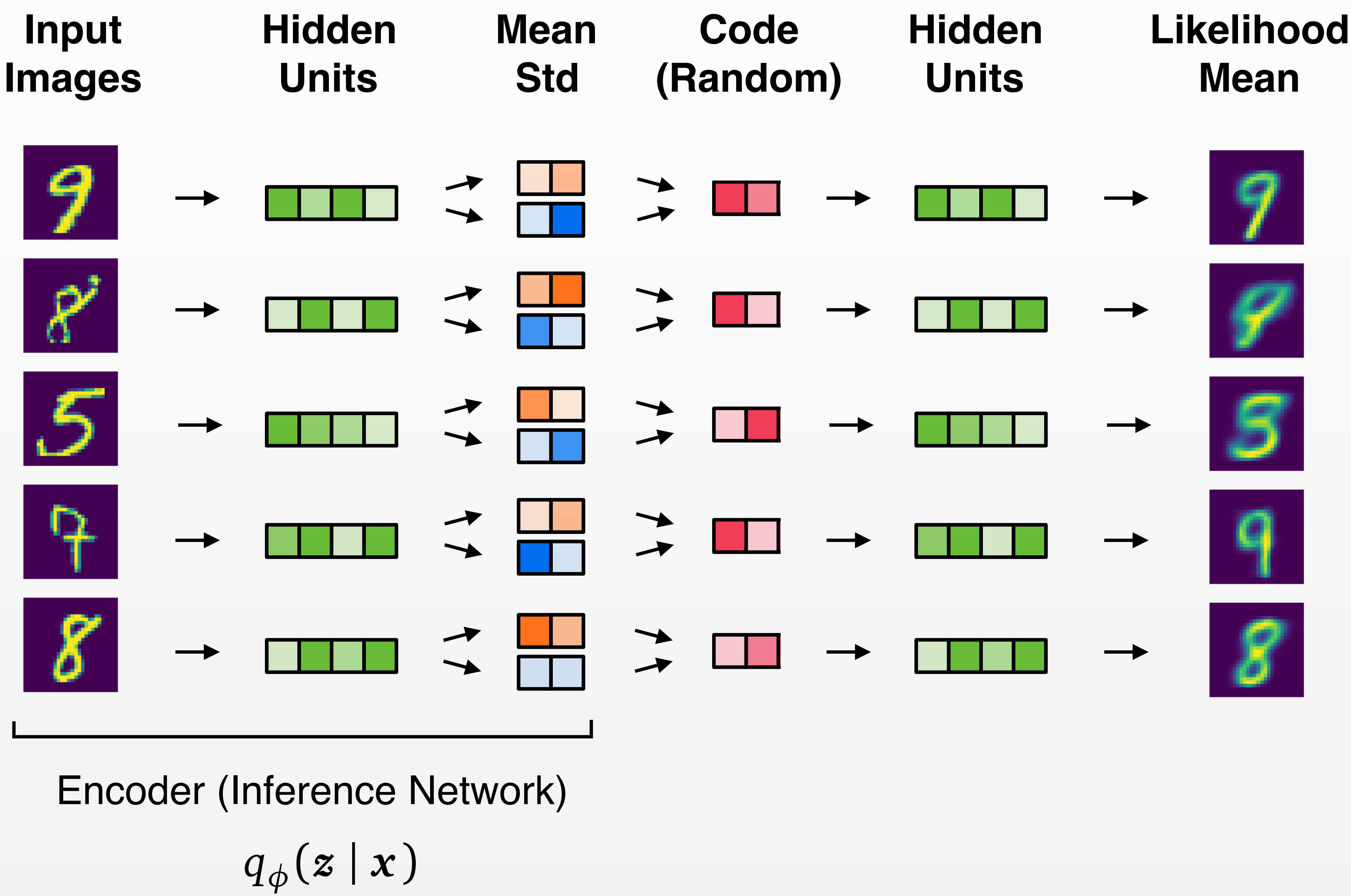
Objective: Maximize Reconstruction Quality

$$\max_{\phi,\theta} \frac{1}{N} \sum_{n=1}^{N} \sum_{p=1}^{P} \boldsymbol{x}_{n,p} \log \hat{\boldsymbol{x}}_{n,p}(\boldsymbol{z}(\boldsymbol{x}_n; \phi); \theta)$$

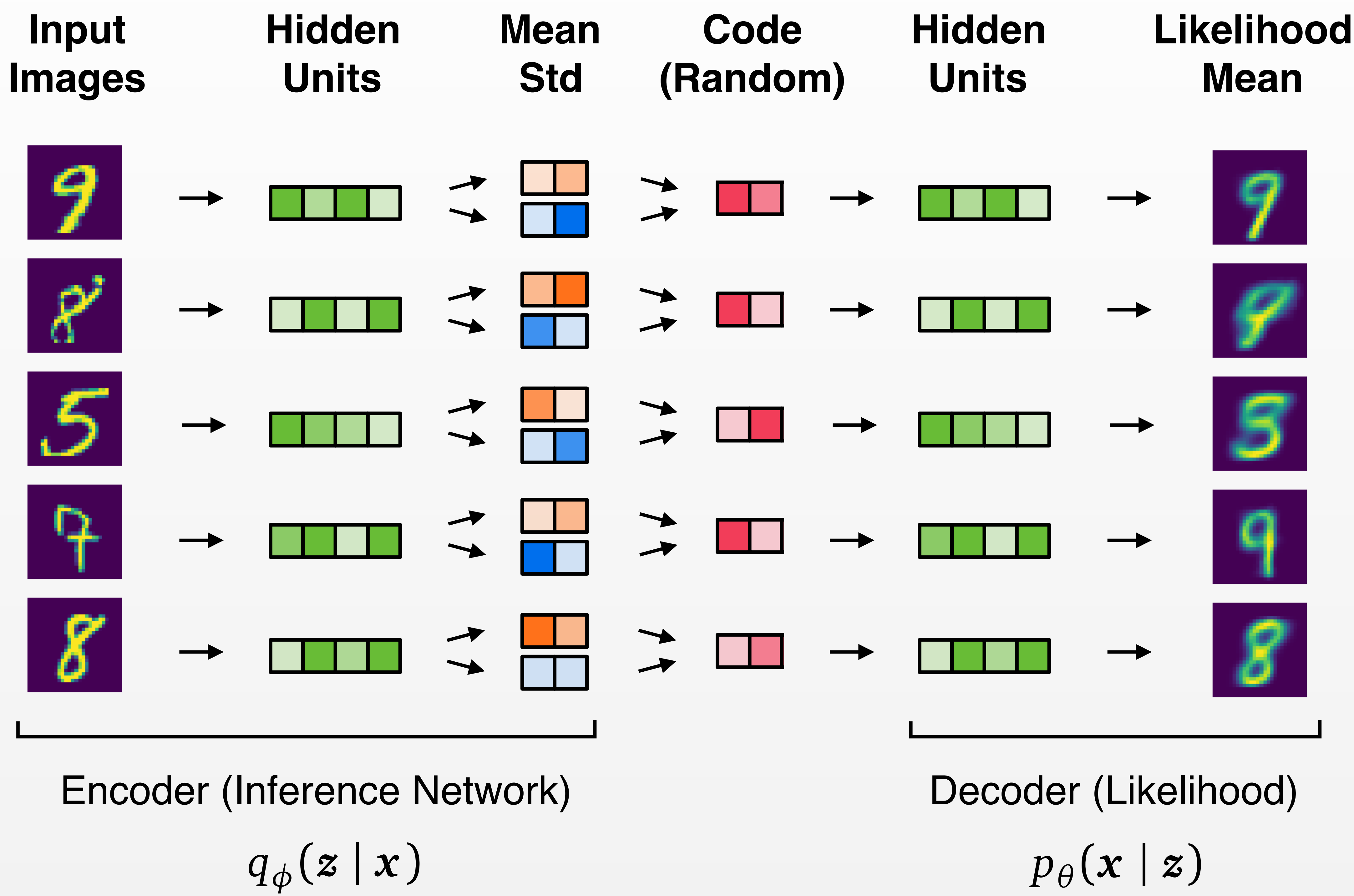
Variational Autoencoders



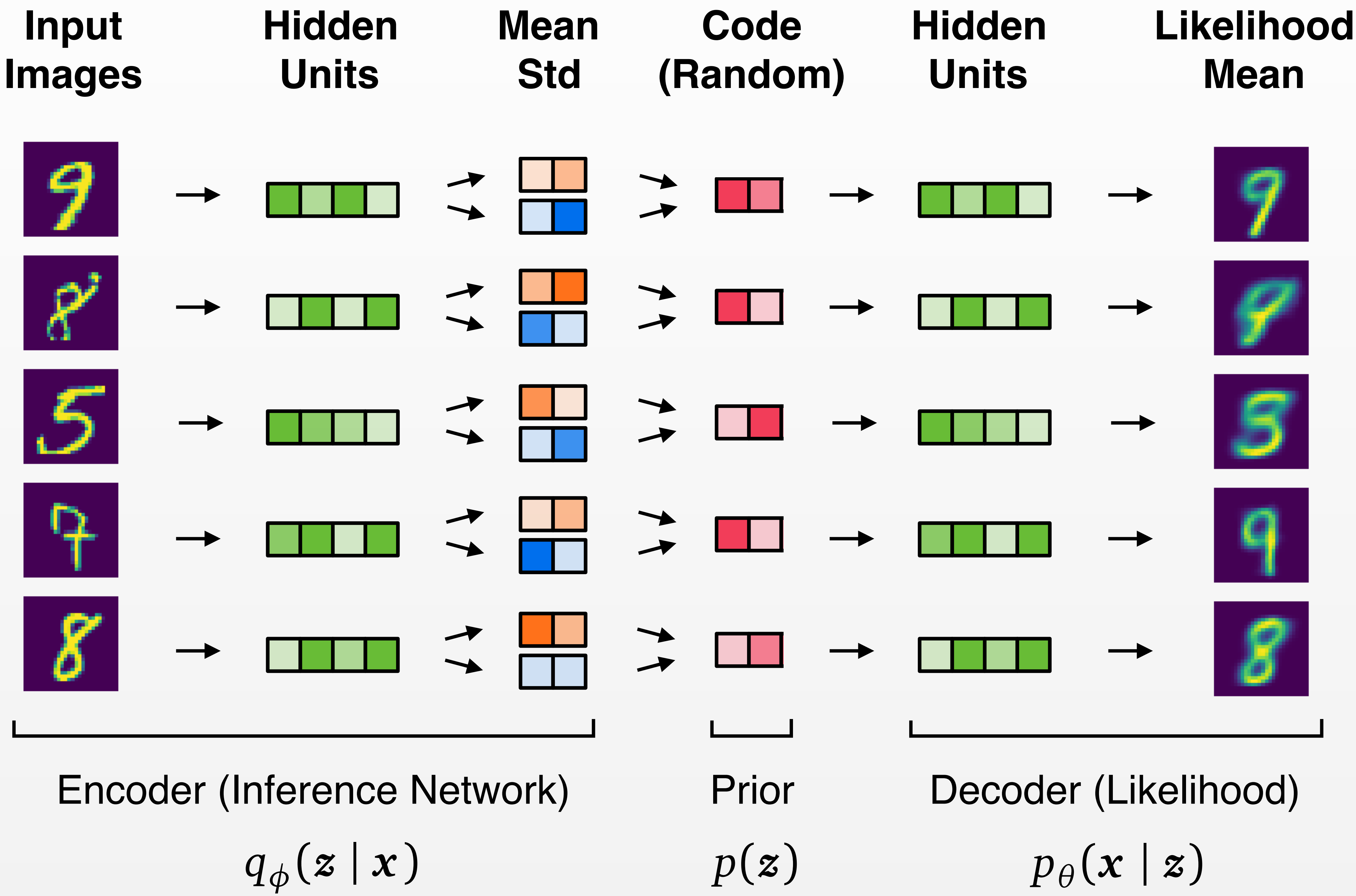
Variational Autoencoders



Variational Autoencoders



Variational Autoencoders



Variational Inference

Objective: Maximize Evidence Lower Bound

$$\mathscr{L}(\theta, \phi) := \mathbb{E}_{p^{\text{data}}(\boldsymbol{x})}\Big[\log p_\theta(\boldsymbol{x}) - \mathrm{KL}(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \,||\, p_\theta(\boldsymbol{z} \mid \boldsymbol{x}))\Big]$$

Variational Inference

Objective: Maximize Evidence Lower Bound

$$\mathcal{L}(\theta, \phi) := \mathbb{E}_{p^{\text{data}}(\boldsymbol{x})}\Big[\log p_\theta(\boldsymbol{x}) - \text{KL}(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \mid\mid p_\theta(\boldsymbol{z} \mid \boldsymbol{x}))\Big]$$

$$= \mathbb{E}_{p^{\text{data}}(\boldsymbol{x})}\left[\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p_\theta(\boldsymbol{x})p_\theta(\boldsymbol{z} \mid \boldsymbol{x})}{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})}\right]\right]$$

Variational Inference

Objective: Maximize Evidence Lower Bound

$$\mathcal{L}(\theta, \phi) := \mathbb{E}_{p^{\text{data}}(\boldsymbol{x})}\Big[\log p_\theta(\boldsymbol{x}) - \text{KL}(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \,||\, p_\theta(\boldsymbol{z} \mid \boldsymbol{x}))\Big]$$

$$= \mathbb{E}_{p^{\text{data}}(\boldsymbol{x})}\left[\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p_\theta(\boldsymbol{x})p_\theta(\boldsymbol{z} \mid \boldsymbol{x})}{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})}\right]\right]$$

$$= \mathbb{E}_{p^{\text{data}}(\boldsymbol{x})}\left[\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p_\theta(\boldsymbol{x}, \boldsymbol{z})}{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})}\right]\right]$$ (computable)

Variational Inference

Objective: Maximize Evidence Lower Bound

$$\mathcal{L}(\theta,\phi) := \mathbb{E}_{p^{\text{data}}(\boldsymbol{x})}\Big[\log p_\theta(\boldsymbol{x}) - \mathrm{KL}(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \,||\, p_\theta(\boldsymbol{z} \mid \boldsymbol{x}))\Big]$$

$$= \mathbb{E}_{p^{\text{data}}(\boldsymbol{x})}\left[\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p_\theta(\boldsymbol{x})p_\theta(\boldsymbol{z} \mid \boldsymbol{x})}{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})}\right]\right]$$

$$= \mathbb{E}_{p^{\text{data}}(\boldsymbol{x})}\left[\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})}\right]\right] \quad \text{(computable)}$$

$$= \mathbb{E}_{p^{\text{data}}(\boldsymbol{x})}\left[\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_\theta(\boldsymbol{x} \mid \boldsymbol{z}) + \log \frac{p(\boldsymbol{z})}{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})}\right]\right]$$

Variational Inference

Objective: Maximize Evidence Lower Bound

$$\mathcal{L}(\theta,\phi) := \mathbb{E}_{p^{\text{data}}(\boldsymbol{x})}\Big[\log p_\theta(\boldsymbol{x}) - \mathrm{KL}(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \,||\, p_\theta(\boldsymbol{z} \mid \boldsymbol{x}))\Big]$$

$$= \mathbb{E}_{p^{\text{data}}(\boldsymbol{x})}\left[\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p_\theta(\boldsymbol{x})p_\theta(\boldsymbol{z} \mid \boldsymbol{x})}{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})}\right]\right]$$

$$= \mathbb{E}_{p^{\text{data}}(\boldsymbol{x})}\left[\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})}\right]\right] \quad \text{(computable)}$$

$$= \mathbb{E}_{p^{\text{data}}(\boldsymbol{x})}\left[\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_\theta(\boldsymbol{x} \mid \boldsymbol{z}) + \log \frac{p(\boldsymbol{z})}{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})}\right]\right]$$

$$= \mathbb{E}_{p^{\text{data}}(\boldsymbol{x})}\Big[\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x} \mid \boldsymbol{z})] - \mathrm{KL}(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \,||\, p(\boldsymbol{z}))\Big]$$

Variational Inference

Objective: Maximize Evidence Lower Bound

$$\mathcal{L}(\theta,\phi) := \mathbb{E}_{p^{\text{data}}(\boldsymbol{x})}\Big[\log p_\theta(\boldsymbol{x}) - \mathrm{KL}(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \,||\, p_\theta(\boldsymbol{z} \mid \boldsymbol{x}))\Big]$$

$$= \mathbb{E}_{p^{\text{data}}(\boldsymbol{x})}\left[\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p_\theta(\boldsymbol{x})p_\theta(\boldsymbol{z} \mid \boldsymbol{x})}{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})}\right]\right]$$

$$= \mathbb{E}_{p^{\text{data}}(\boldsymbol{x})}\left[\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})}\right]\right] \quad \text{(computable)}$$

$$= \mathbb{E}_{p^{\text{data}}(\boldsymbol{x})}\left[\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_\theta(\boldsymbol{x} \mid \boldsymbol{z}) + \log \frac{p(\boldsymbol{z})}{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})}\right]\right]$$

$$= \mathbb{E}_{p^{\text{data}}(\boldsymbol{x})}\Big[\underbrace{\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x} \mid \boldsymbol{z})]}_{\text{Reconstruction Error}} - \mathrm{KL}(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \,||\, p(\boldsymbol{z}))\Big]$$

Variational Inference

Objective: Maximize Evidence Lower Bound

$$\mathcal{L}(\theta,\phi) := \mathbb{E}_{p^{\text{data}}(\boldsymbol{x})}\Big[\log p_\theta(\boldsymbol{x}) - \mathrm{KL}(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \,||\, p_\theta(\boldsymbol{z} \mid \boldsymbol{x}))\Big]$$

$$= \mathbb{E}_{p^{\text{data}}(\boldsymbol{x})}\left[\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p_\theta(\boldsymbol{x})p_\theta(\boldsymbol{z} \mid \boldsymbol{x})}{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})}\right]\right]$$

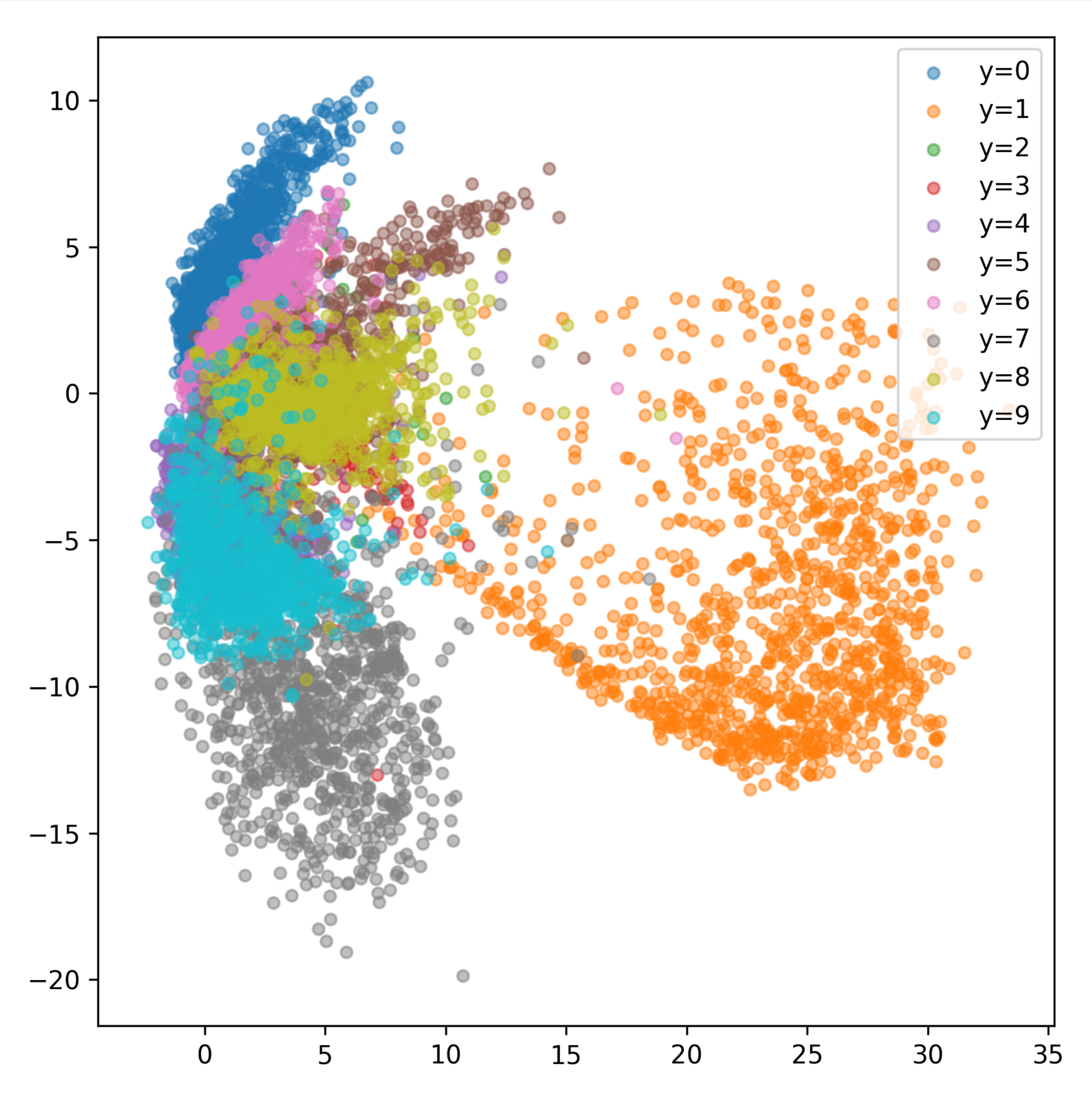
$$= \mathbb{E}_{p^{\text{data}}(\boldsymbol{x})}\left[\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})}\right]\right] \quad \text{(computable)}$$

$$= \mathbb{E}_{p^{\text{data}}(\boldsymbol{x})}\left[\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_\theta(\boldsymbol{x} \mid \boldsymbol{z}) + \log \frac{p(\boldsymbol{z})}{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})}\right]\right]$$

$$= \mathbb{E}_{p^{\text{data}}(\boldsymbol{x})}\Bigg[\underbrace{\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x} \mid \boldsymbol{z})]}_{\text{Reconstruction Error}} - \underbrace{\mathrm{KL}(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \,||\, p(\boldsymbol{z}))}_{\text{KL Regularization}}\Bigg]$$

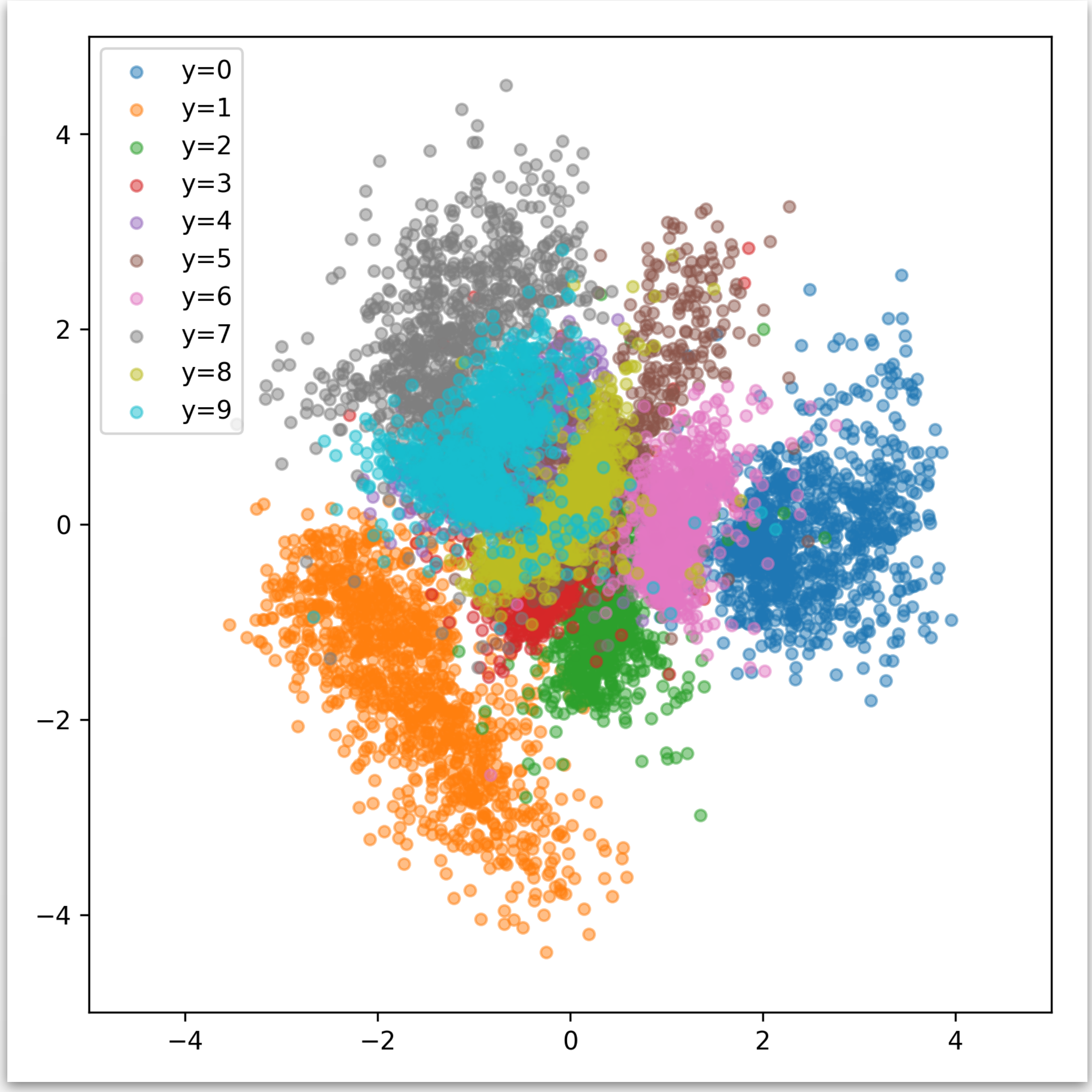
Regular vs Variational Autoencoders

Autoencoder (2-dim)



Arbitrary scale (no "typical" values)

VAE (2-dim)

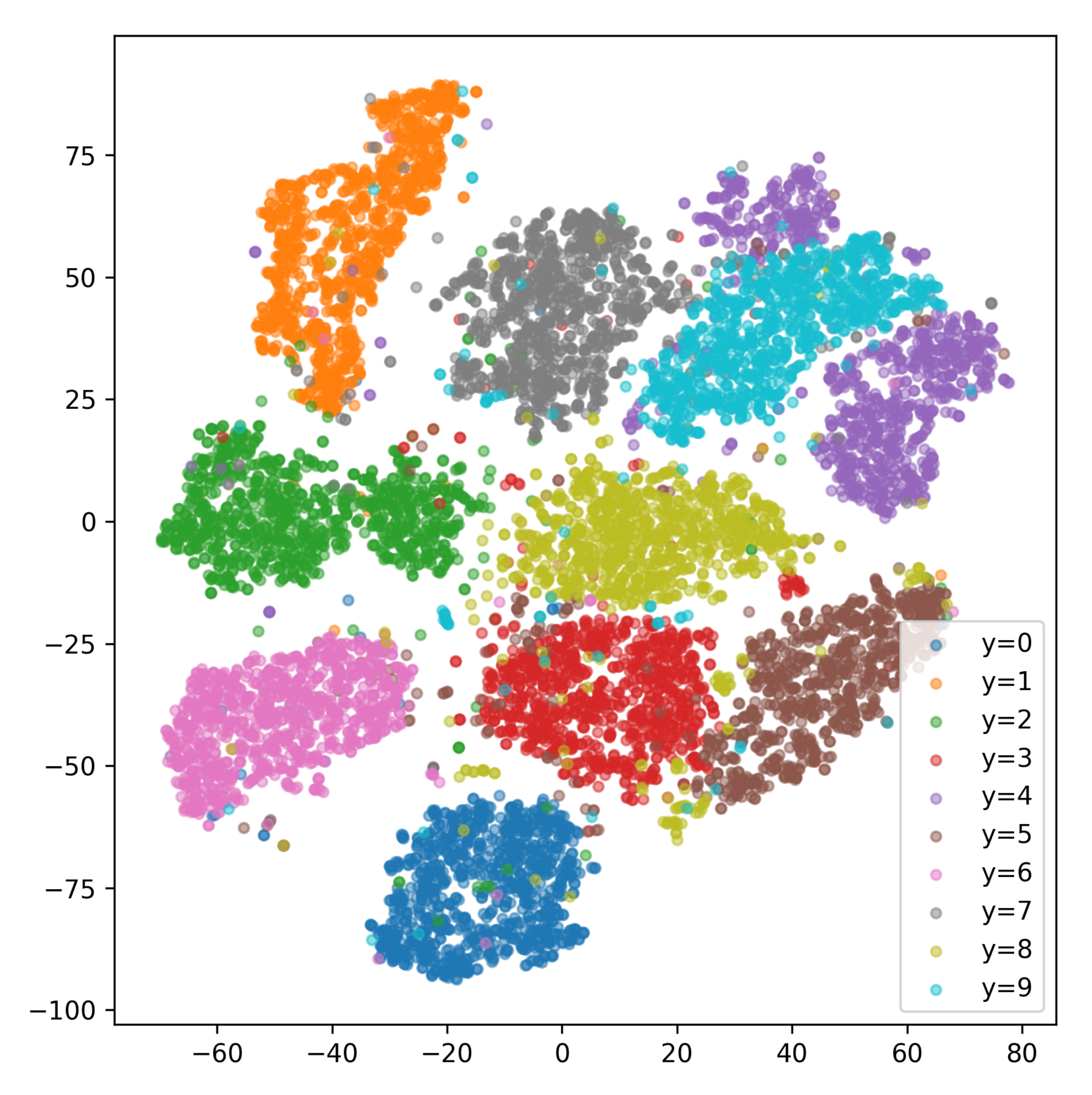


Well-defined scale (-4 < z < 4)

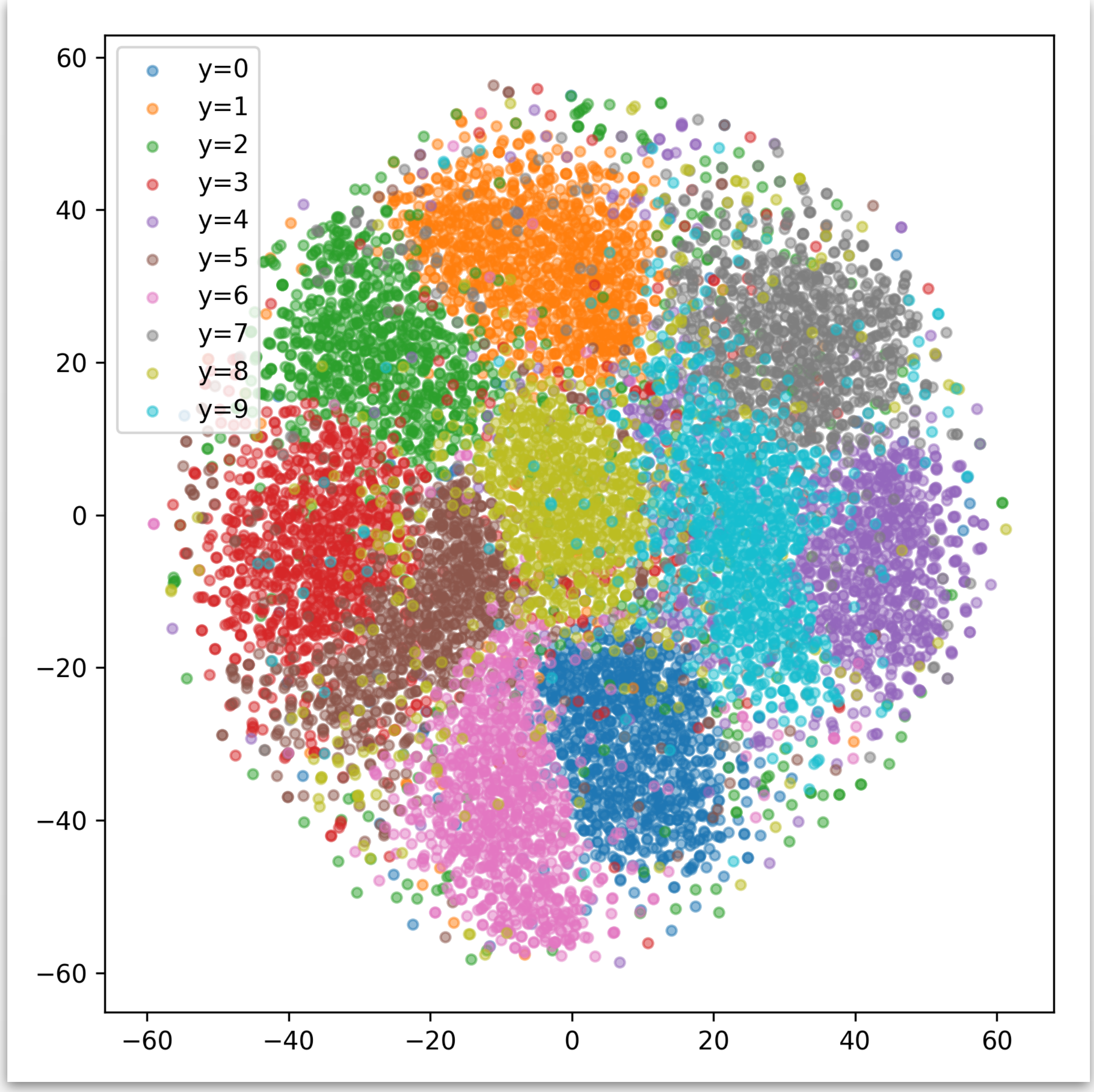
KL regularization constrains values of latent codes

Regular vs Variational Autoencoders

Autoencoder (50-dim TSNE)

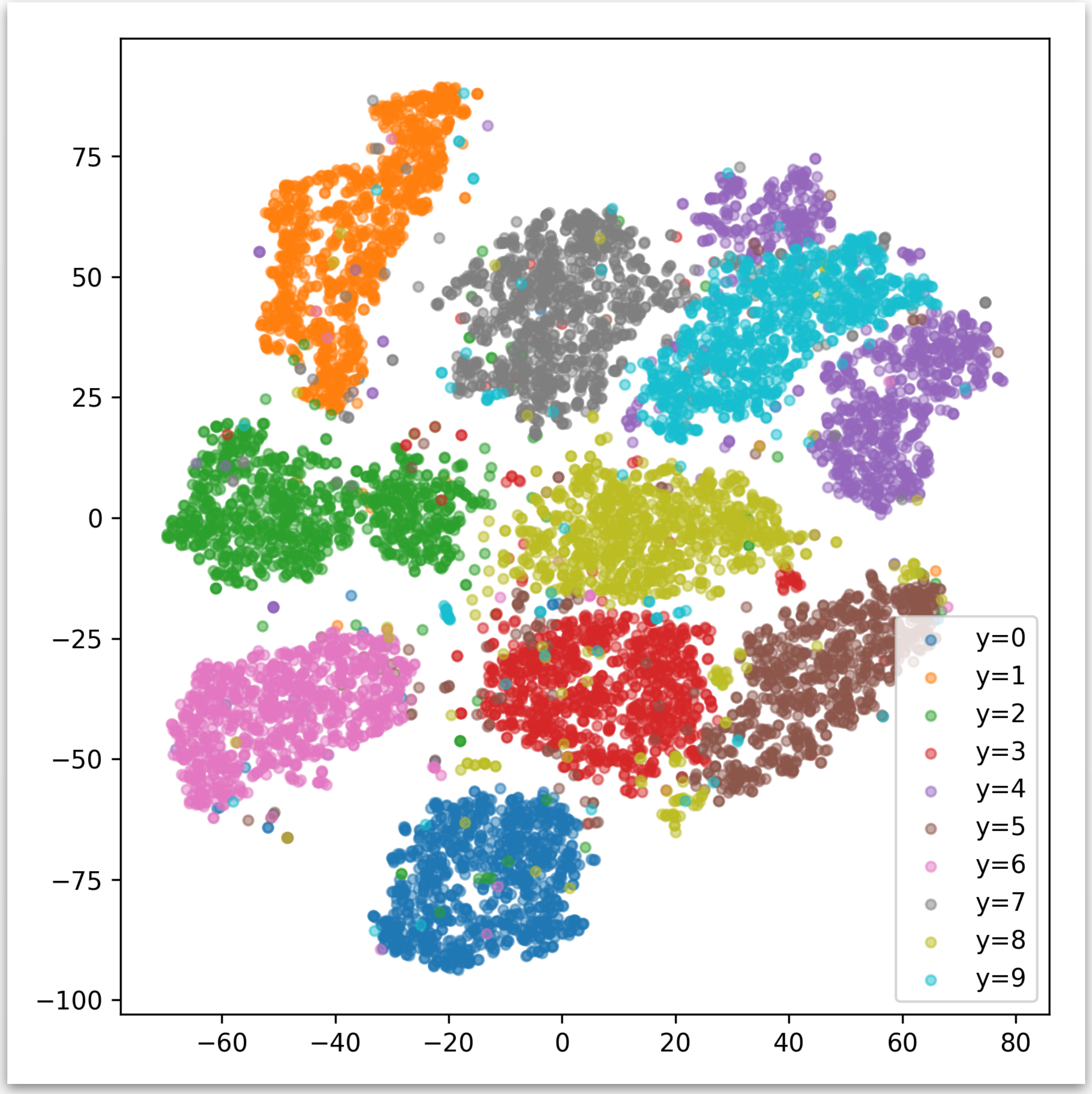


VAE (50-dim TSNE)

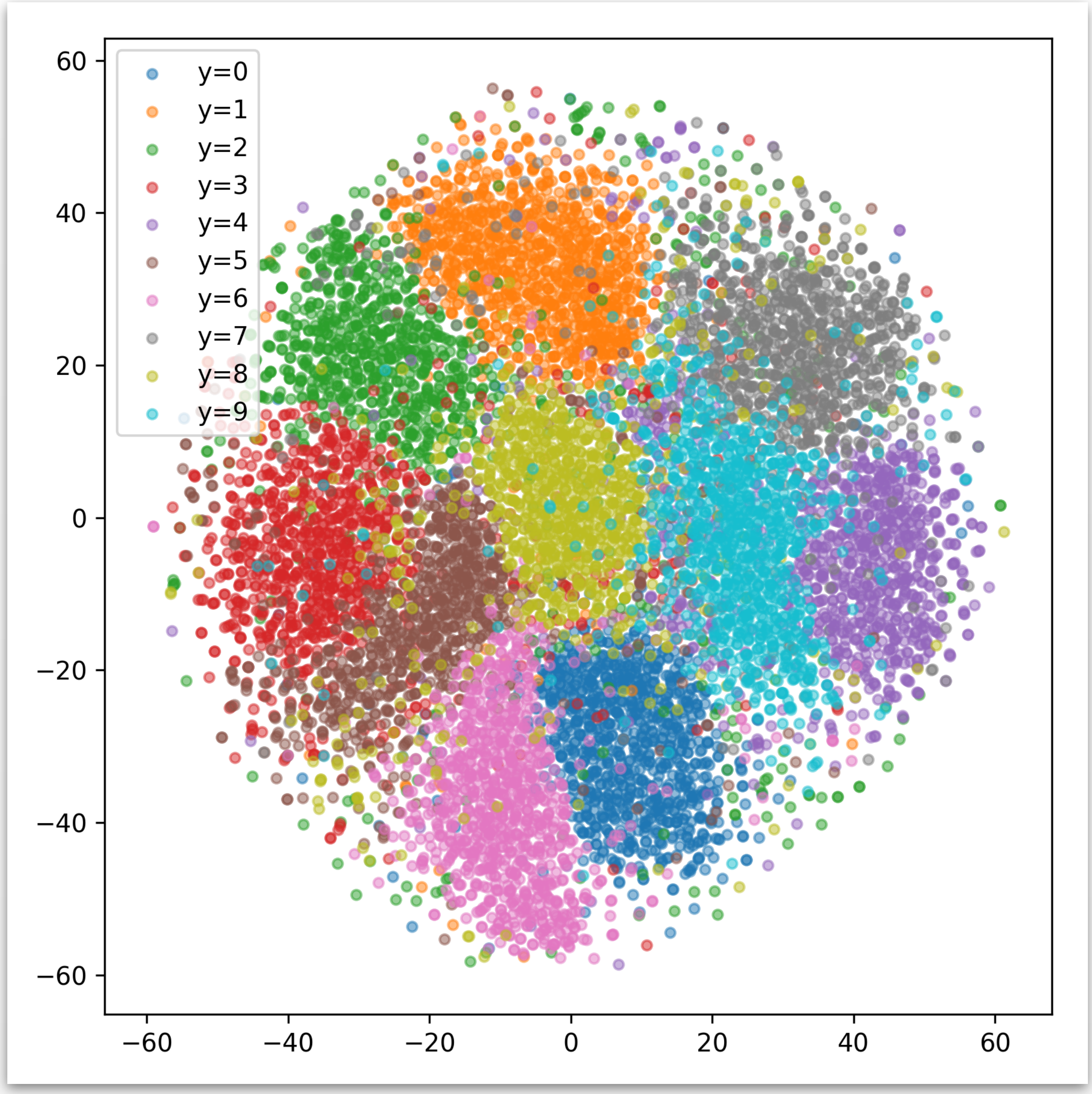


Regular vs Variational Autoencoders

Autoencoder (50-dim TSNE)



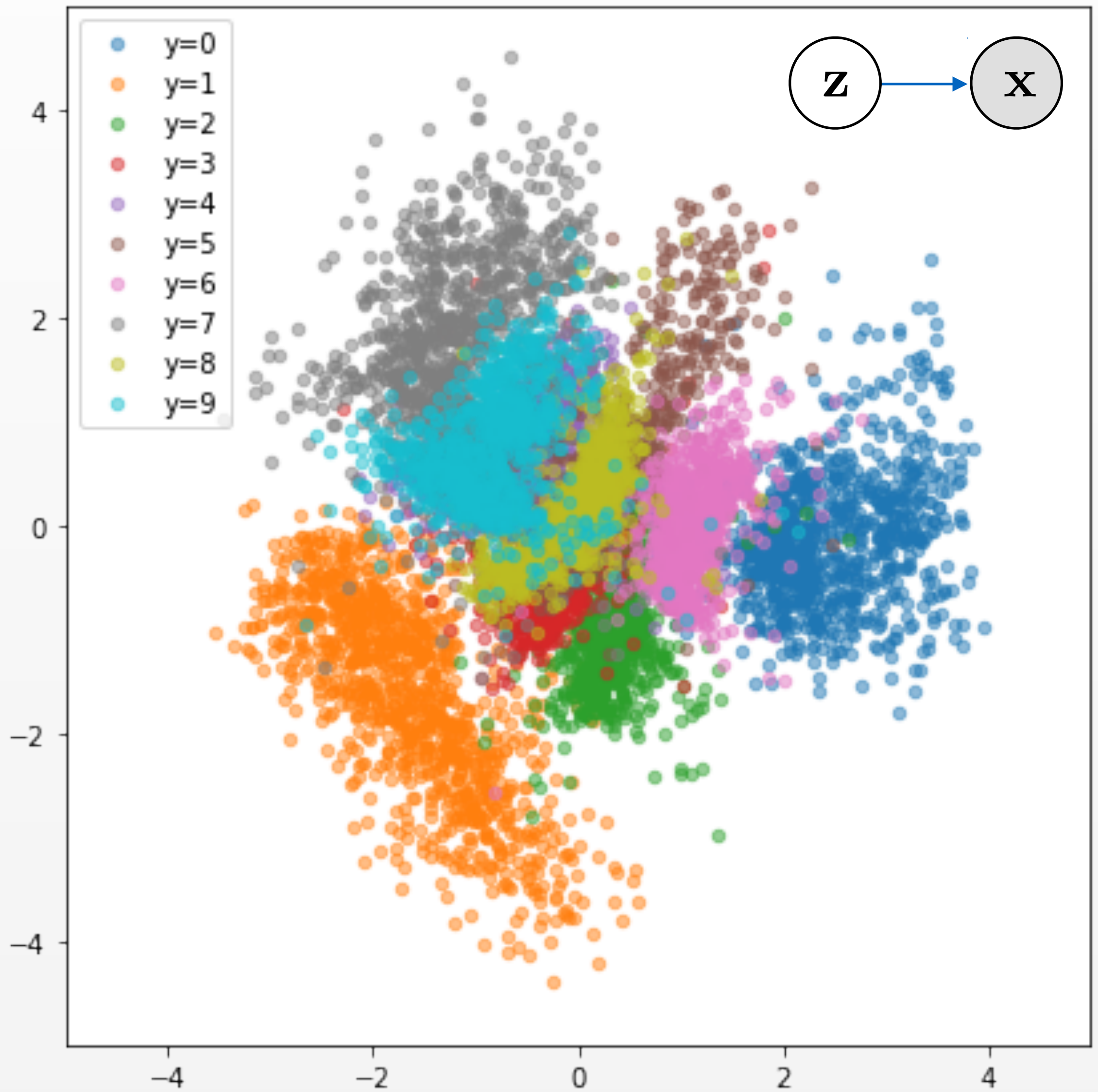
VAE (50-dim TSNE)



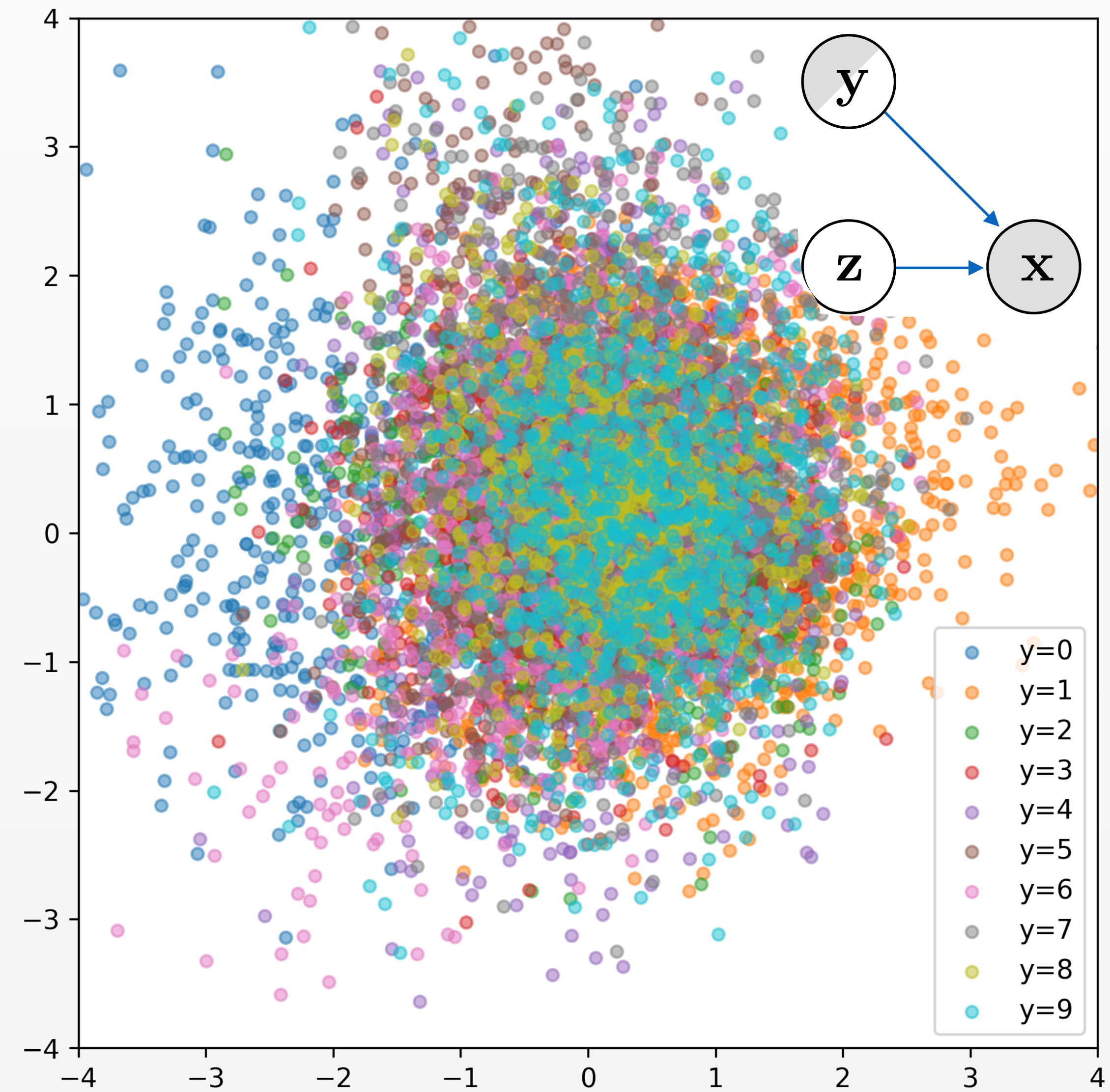
Representations are still entangled

Unsupervised vs Semi-Supervised

Unsupervised, Entangled

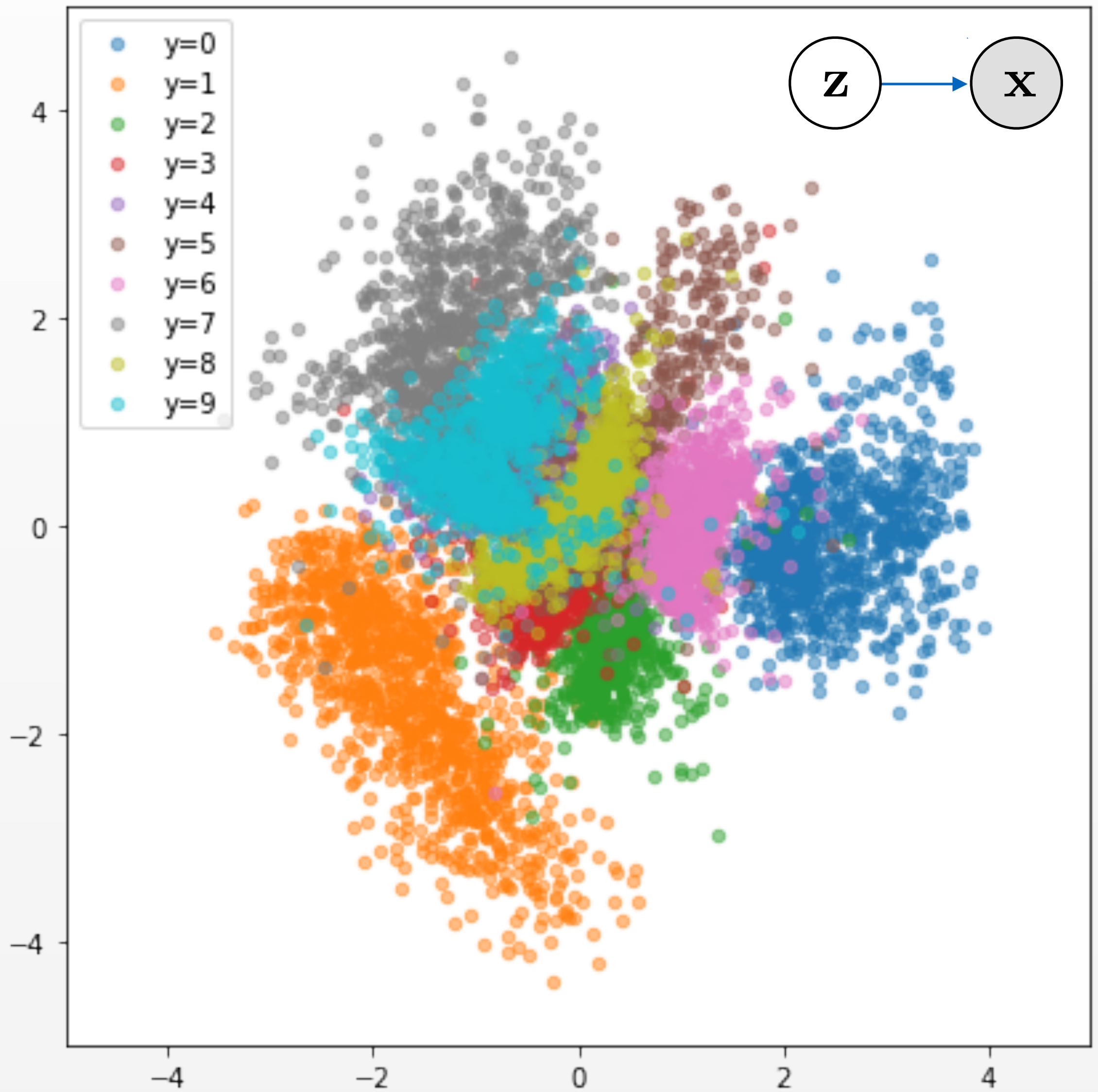


Semi-supervised, Disentangled

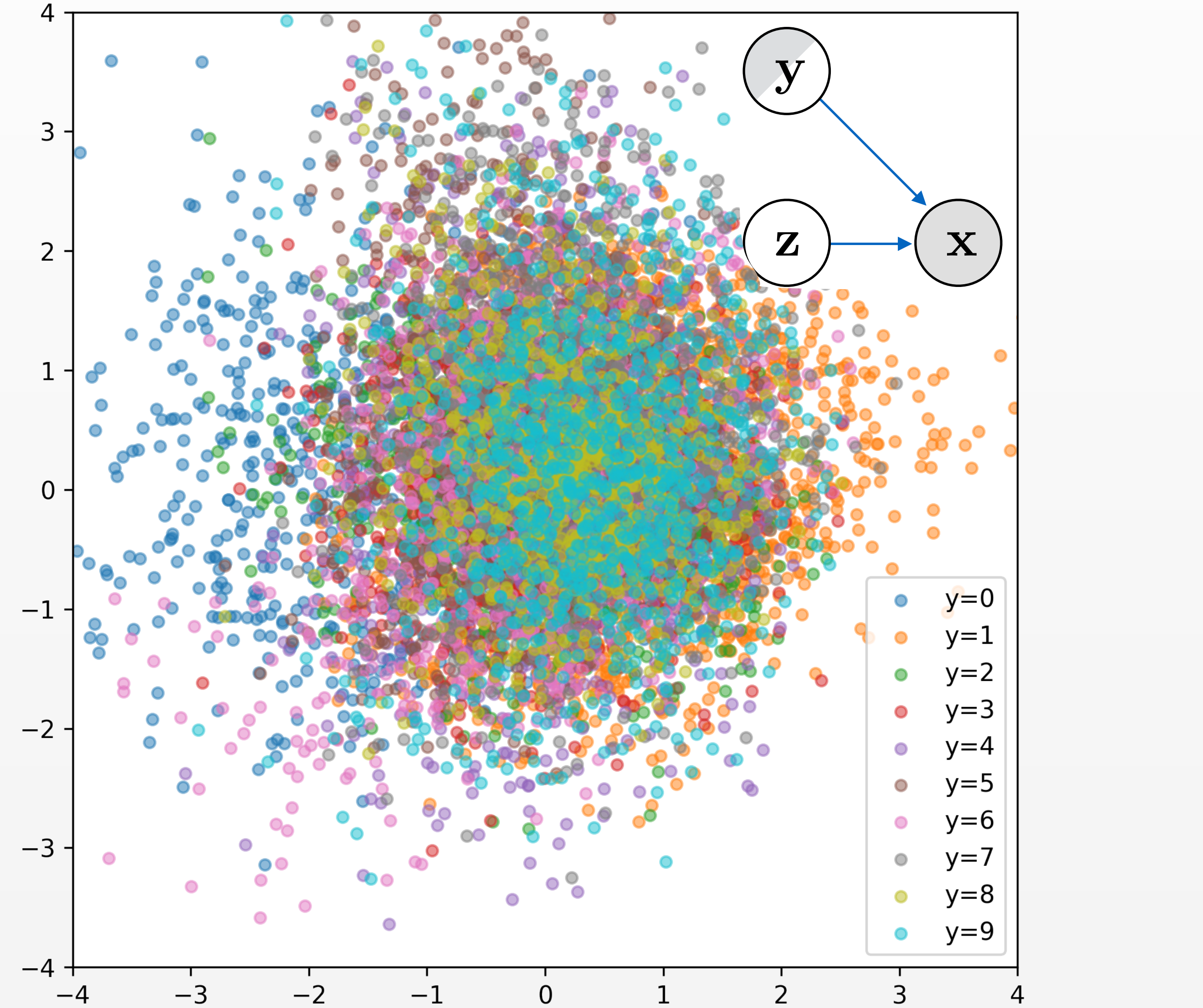


Unsupervised vs Semi-Supervised

Unsupervised, Entangled



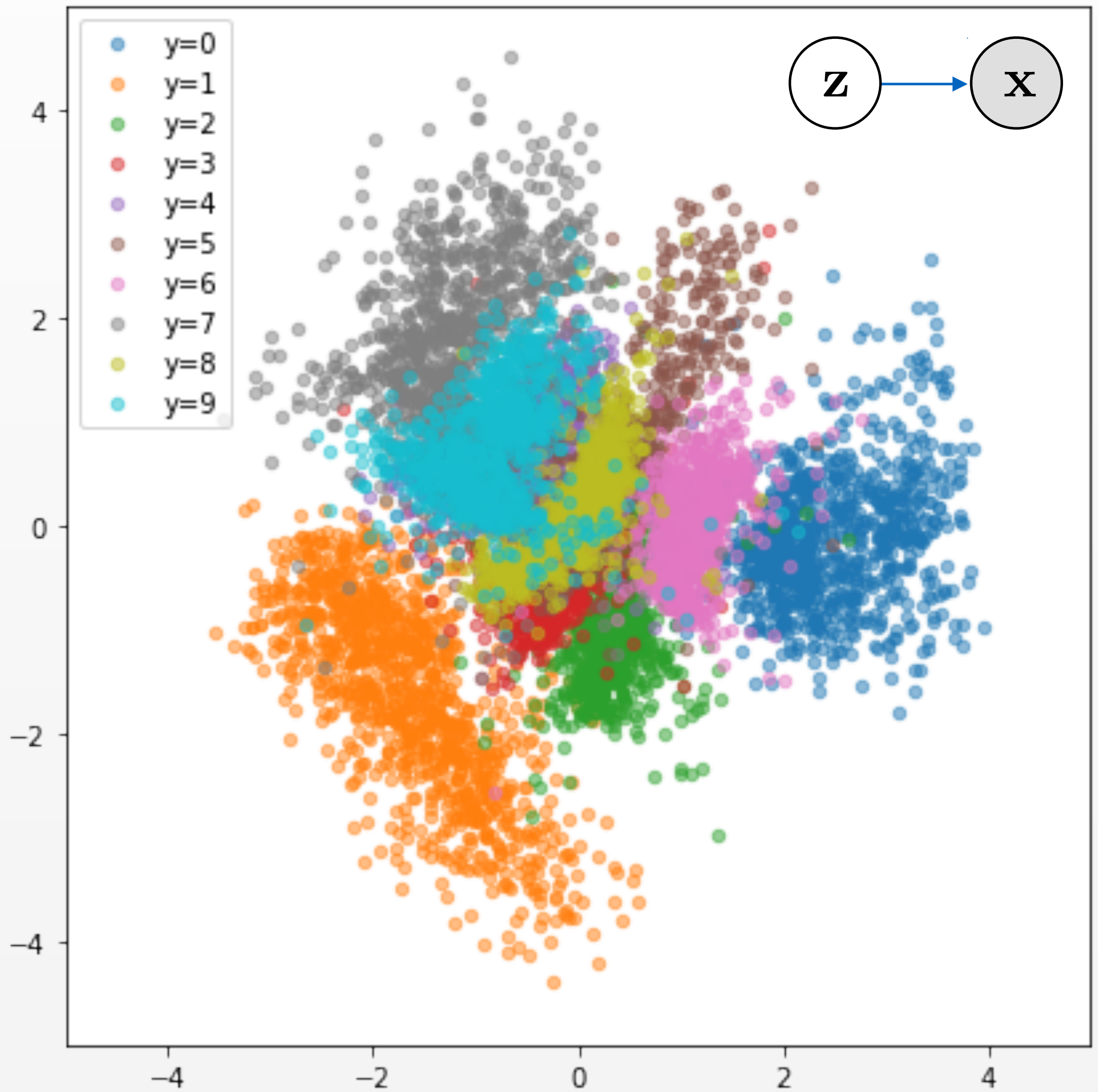
Semi-supervised, Disentangled



Latent code ***z*** represents both style and digit

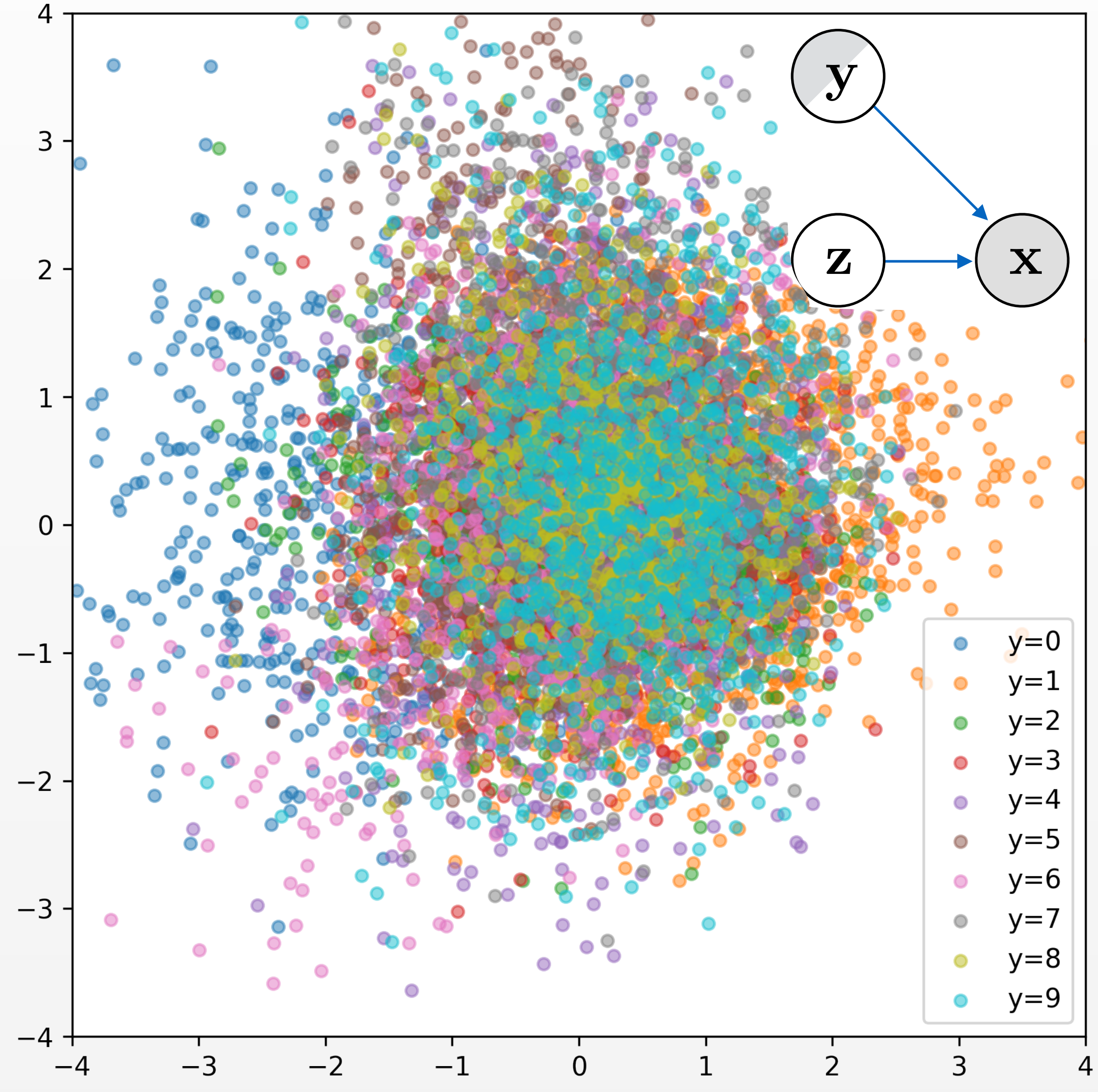
Unsupervised vs Semi-Supervised

Unsupervised, Entangled



Latent code ***z*** represents both style and digit

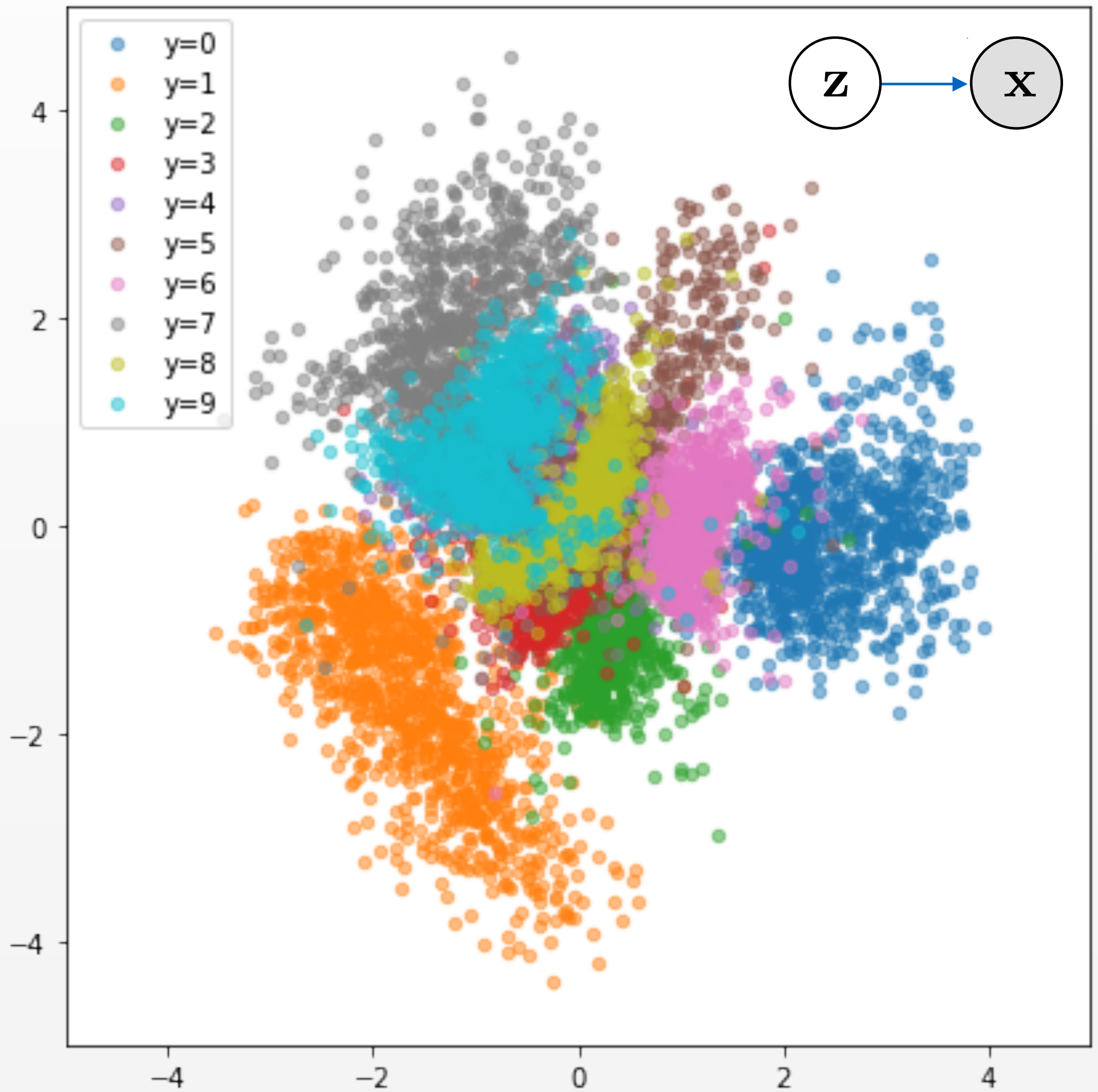
Semi-supervised, Disentangled



Style variable ***z*** is conditionally independent from digit ***y*** (*)

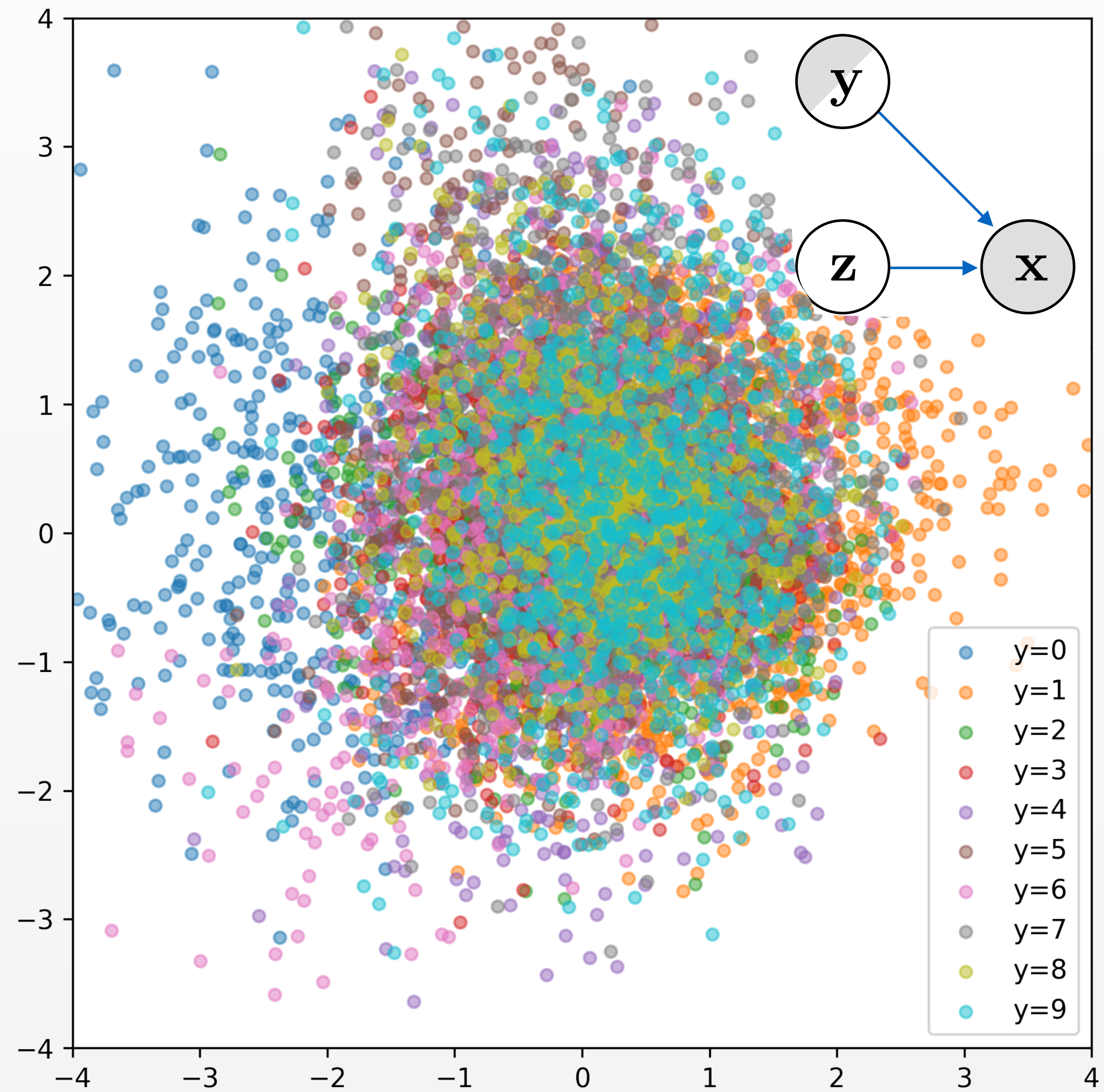
Unsupervised vs Semi-Supervised

Unsupervised, Entangled



Latent code ***z*** represents both style and digit

Semi-supervised, Disentangled

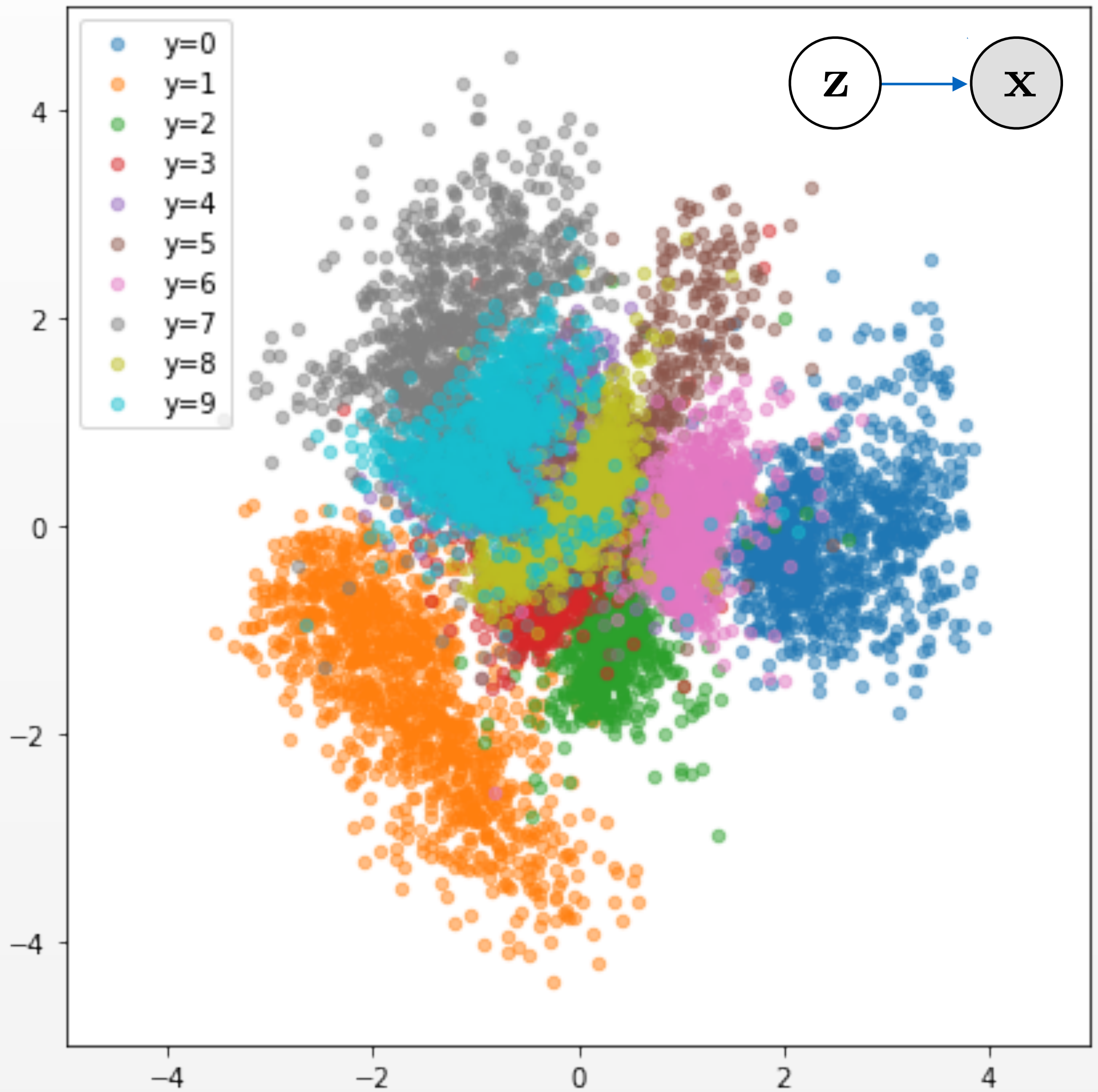


Style variable ***z*** is conditionally independent from digit ***y*** (*)

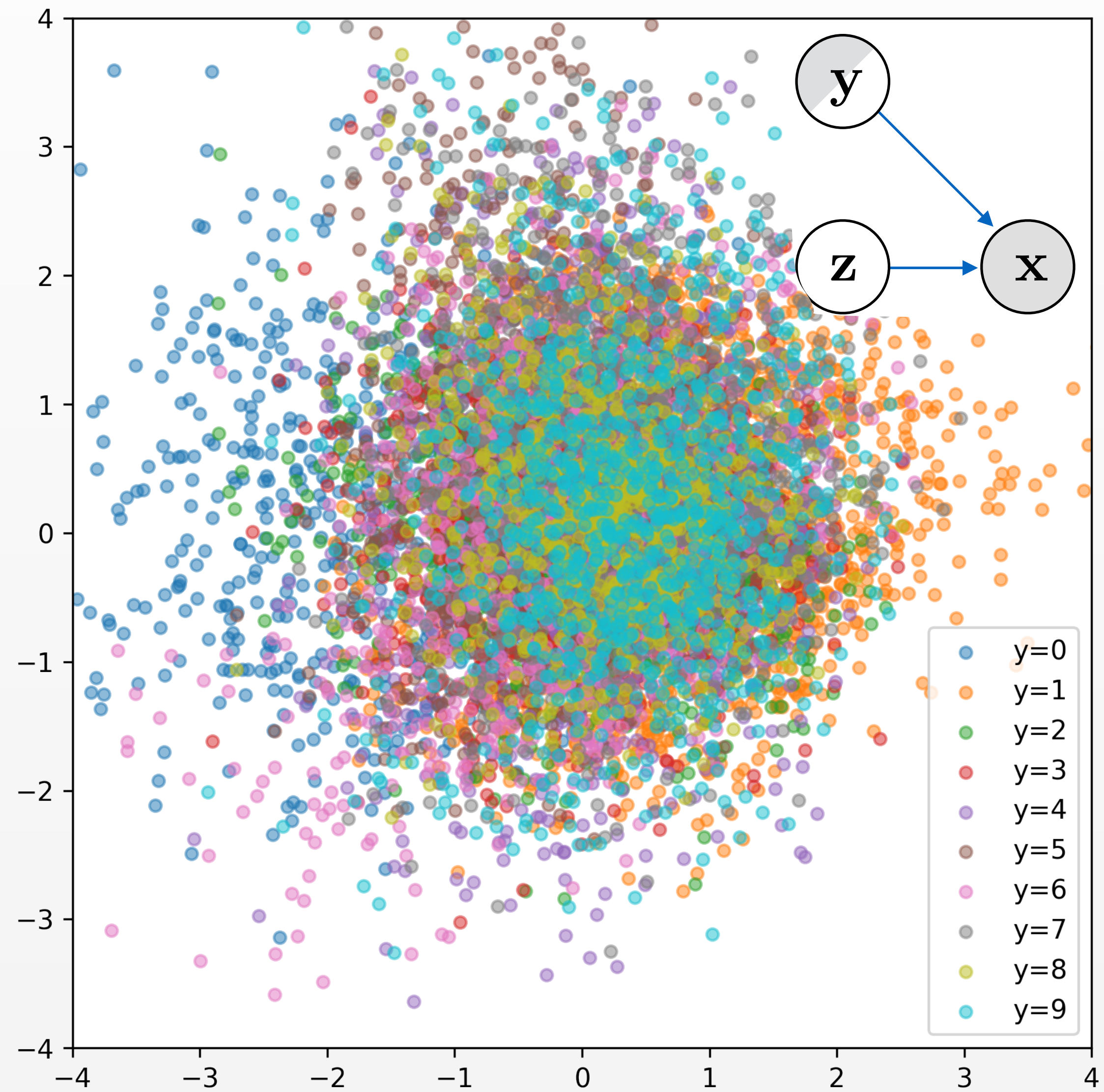
(* *but not without supervision*)

Unsupervised vs Semi-Supervised

Unsupervised, Entangled



Semi-supervised, Disentangled



In both models, prior on ***z*** assumes uncorrelated dimensions

$$p(\boldsymbol{z}) = \prod_d p(\boldsymbol{z}_d)$$

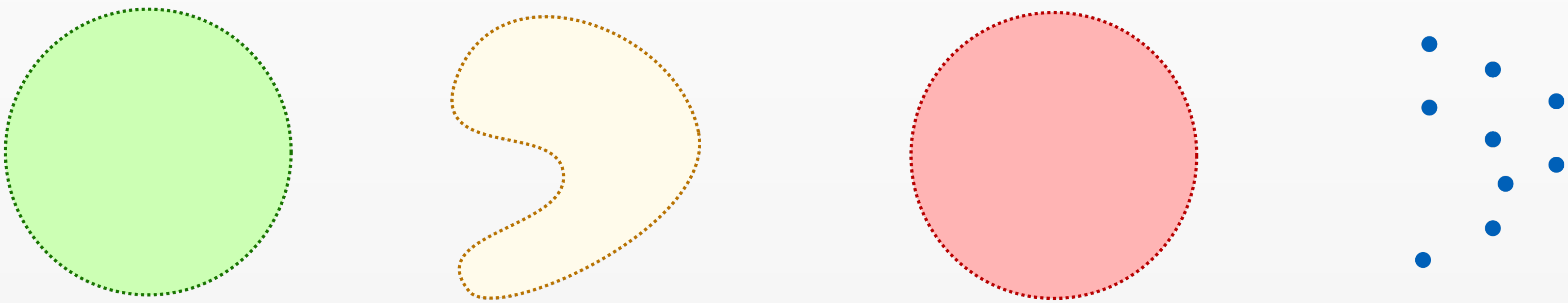
Learning Statistically Independent Factors

Deep Latent-Variable Models

Generative Model

Inference Model

$p(\boldsymbol{z})$ $p_\theta(\boldsymbol{x})$ $q_\phi(\boldsymbol{z})$ $q(\boldsymbol{x})$



Prior (*known*)

Data Distribution (*learned*)

Inference Marginal (*learned*)

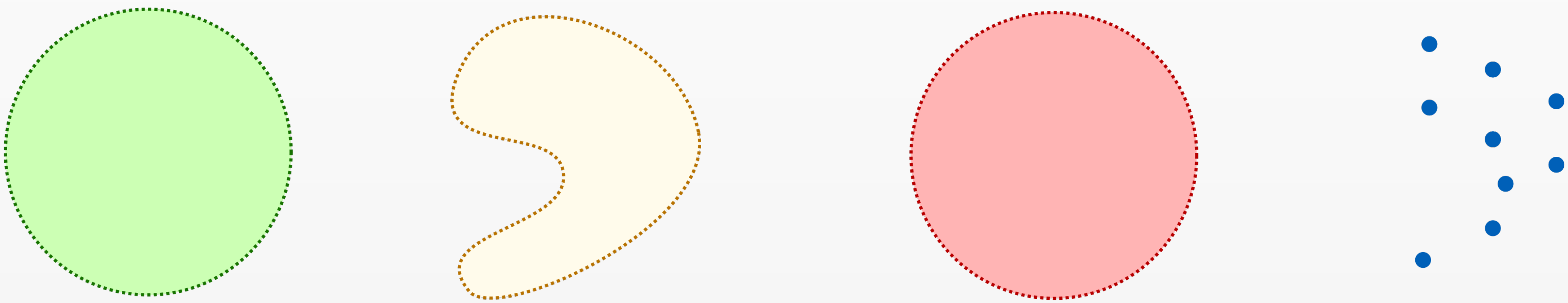
Empirical Distribution (*known*)

Deep Latent-Variable Models

Generative Model

Inference Model

$p(\boldsymbol{z})$ $p_\theta(\boldsymbol{x})$ $q_\phi(\boldsymbol{z})$ $q(\boldsymbol{x})$



Prior
(*known*)

Data Distribution
(*learned*)

Inference Marginal
(*learned*)

Empirical Distribution
(*known*)

$$\boldsymbol{x}_n \sim p^{\text{data}}(\boldsymbol{x})$$

Data Sampled from
Unknown Distribution

$$q(\boldsymbol{x}) := \frac{1}{N}\sum_{n=1}^{N} \delta_{\boldsymbol{x}_n}(\boldsymbol{x})$$

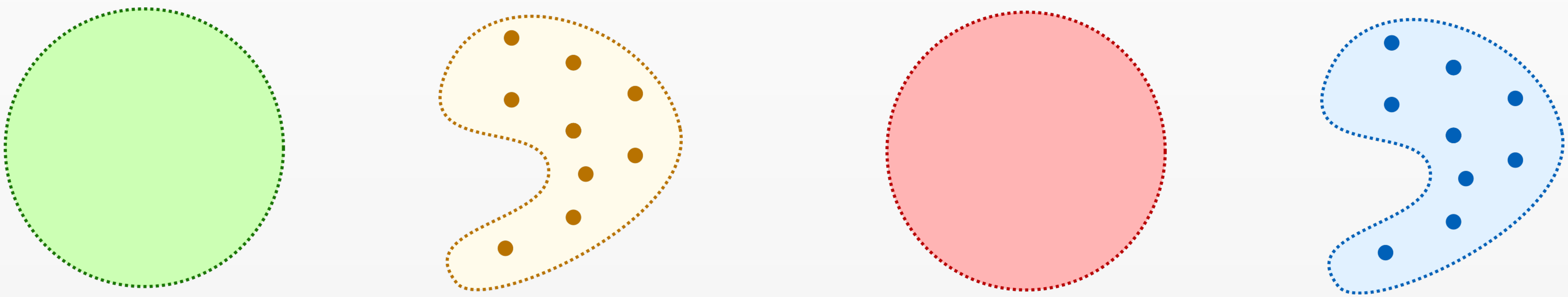
Approximation of
Data Distribution

Deep Latent-Variable Models

Generative Model

Inference Model

$p(\boldsymbol{z})$ $p_\theta(\boldsymbol{x})$ $q_\phi(\boldsymbol{z})$ $q(\boldsymbol{x})$



Prior (*known*)

Data Distribution (*learned*)

Inference Marginal (*learned*)

Empirical Distribution (*known*)

$$\boldsymbol{x}_n \sim p^{\text{data}}(\boldsymbol{x})$$

Data Sampled from Unknown Distribution

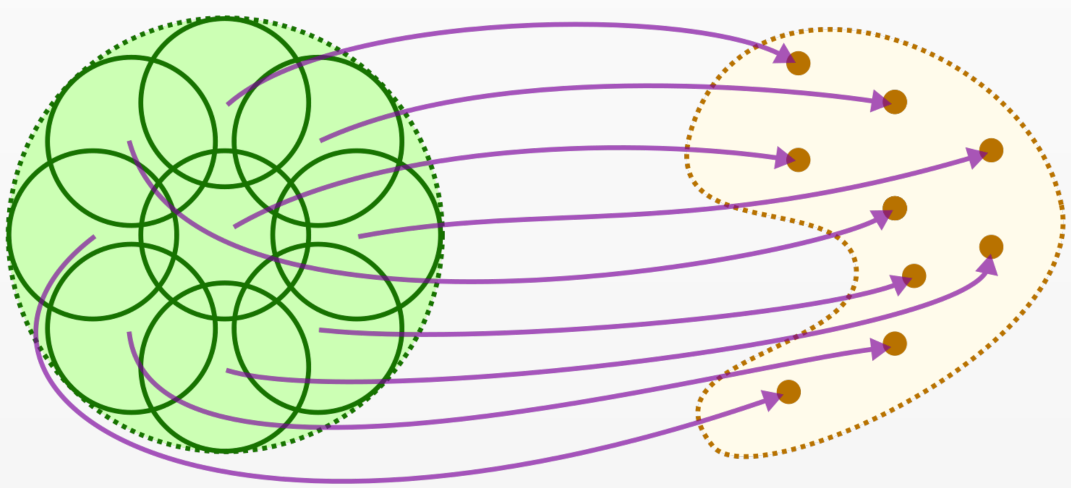
$$q(\boldsymbol{x}) := \frac{1}{N} \sum_{n=1}^{N} \delta_{\boldsymbol{x}_n}(\boldsymbol{x})$$

Approximation of Data Distribution

Deep Latent-Variable Models

Generative Model

$p(\boldsymbol{z})$ $\quad p_\theta(\boldsymbol{x} \mid \boldsymbol{z})$ $\quad p_\theta(\boldsymbol{x})$

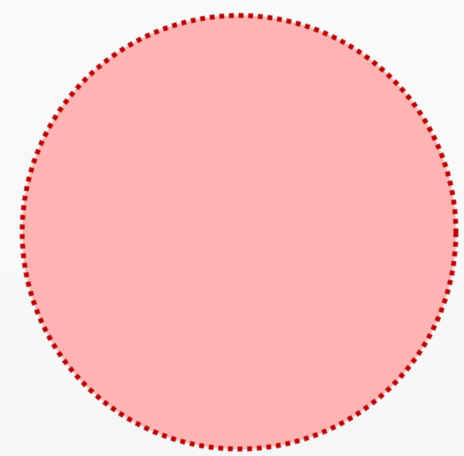


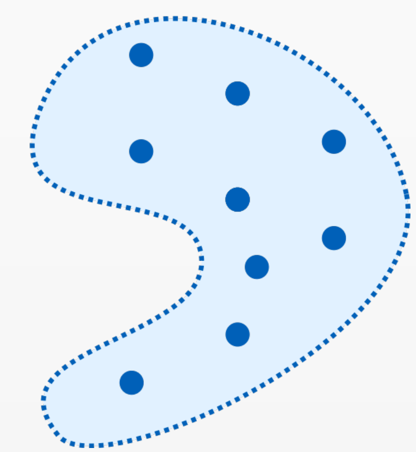
Prior
(*known*)

Data Distribution
(*learned*)

Inference Model

$q_\phi(\boldsymbol{z})$ $\quad q(\boldsymbol{x})$





Inference Marginal
(*learned*)

Empirical Distribution
(*known*)

$$\boldsymbol{x}_n \sim p^{\text{data}}(\boldsymbol{x})$$

Data Sampled from
Unknown Distribution

$$q(\boldsymbol{x}) := \frac{1}{N}\sum_{n=1}^{N} \delta_{\boldsymbol{x}_n}(\boldsymbol{x})$$

Approximation of
Data Distribution

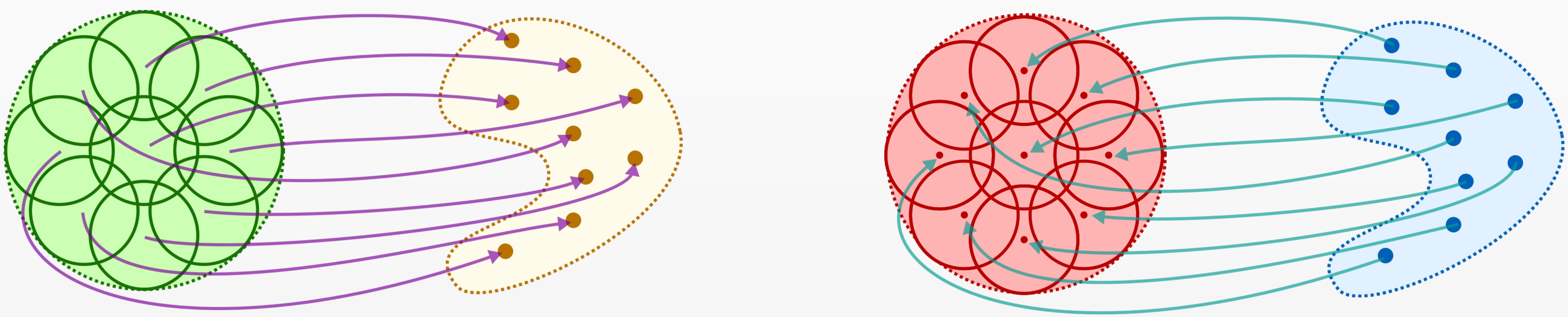
Deep Latent-Variable Models

Generative Model

$p(\boldsymbol{z})$ $\quad p_\theta(\boldsymbol{x} \mid \boldsymbol{z})$ $\quad p_\theta(\boldsymbol{x})$

Inference Model

$q_\phi(\boldsymbol{z})$ $\quad q_\phi(\boldsymbol{z} \mid \boldsymbol{x})$ $\quad q(\boldsymbol{x})$



Prior
(*known*)

Data Distribution
(*learned*)

Inference Marginal
(*learned*)

Empirical Distribution
(*known*)

$$\boldsymbol{x}_n \sim p^{\text{data}}(\boldsymbol{x})$$

Data Sampled from
Unknown Distribution

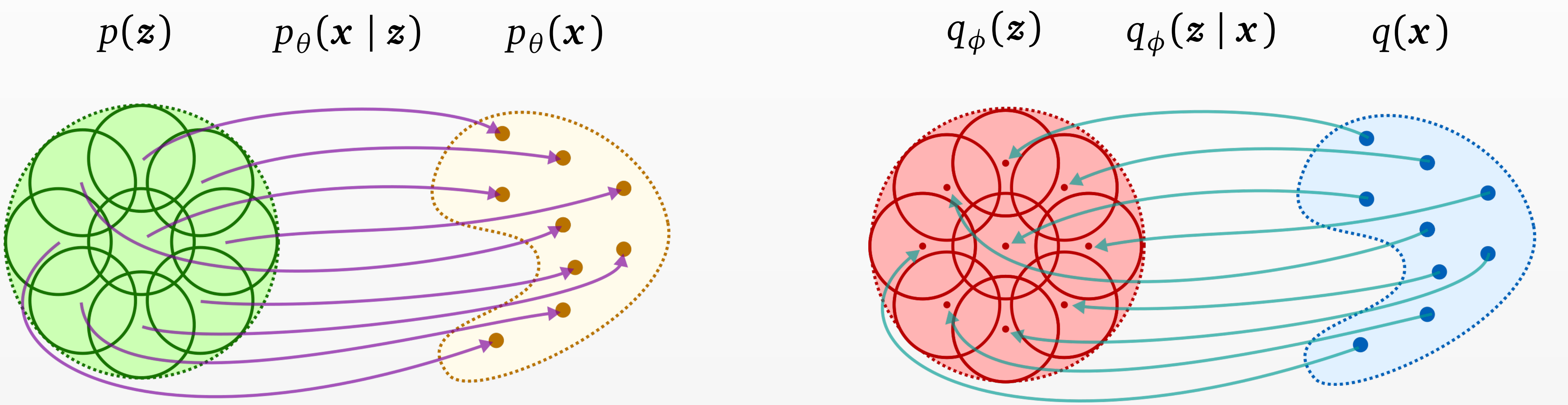
$$q(\boldsymbol{x}) \;:=\; \frac{1}{N}\sum_{n=1}^{N} \delta_{\boldsymbol{x}_n}(\boldsymbol{x})$$

Approximation of
Data Distribution

Deep Latent-Variable Models

Generative Model

Inference Model



Prior (*known*)

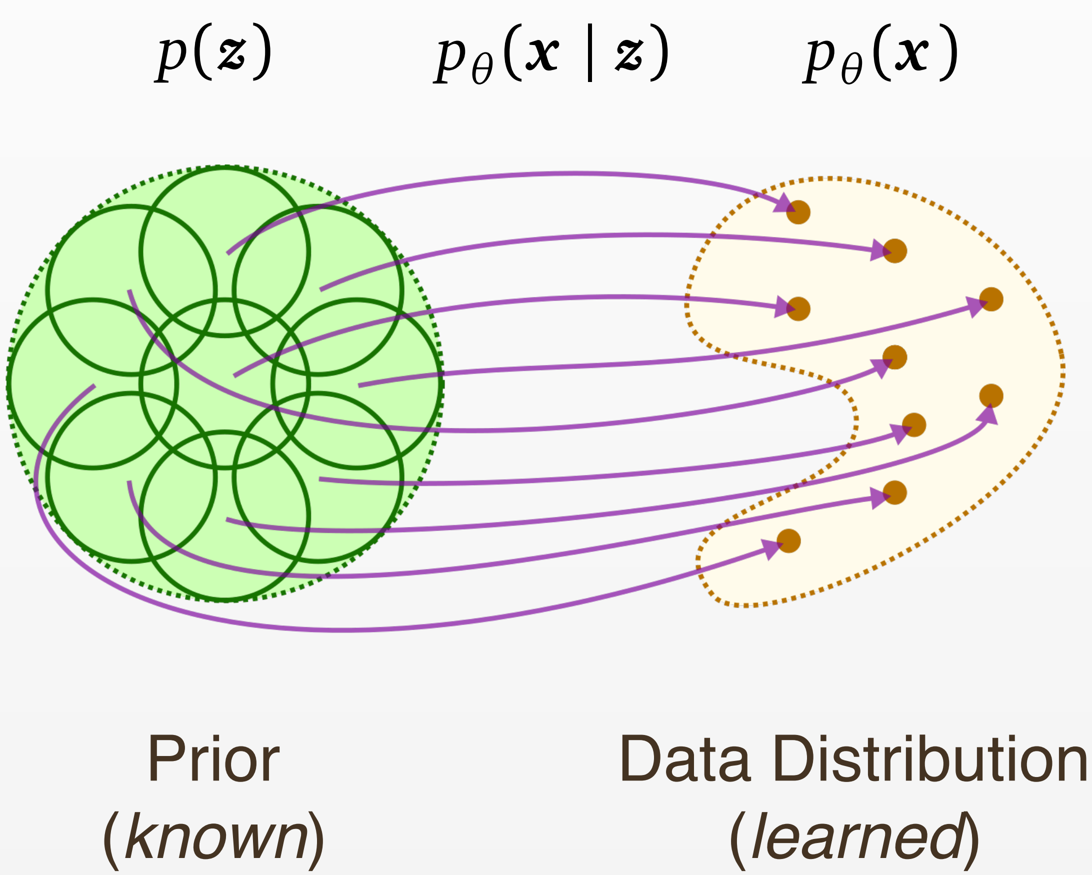
Data Distribution (*learned*)

Inference Marginal (*learned*)

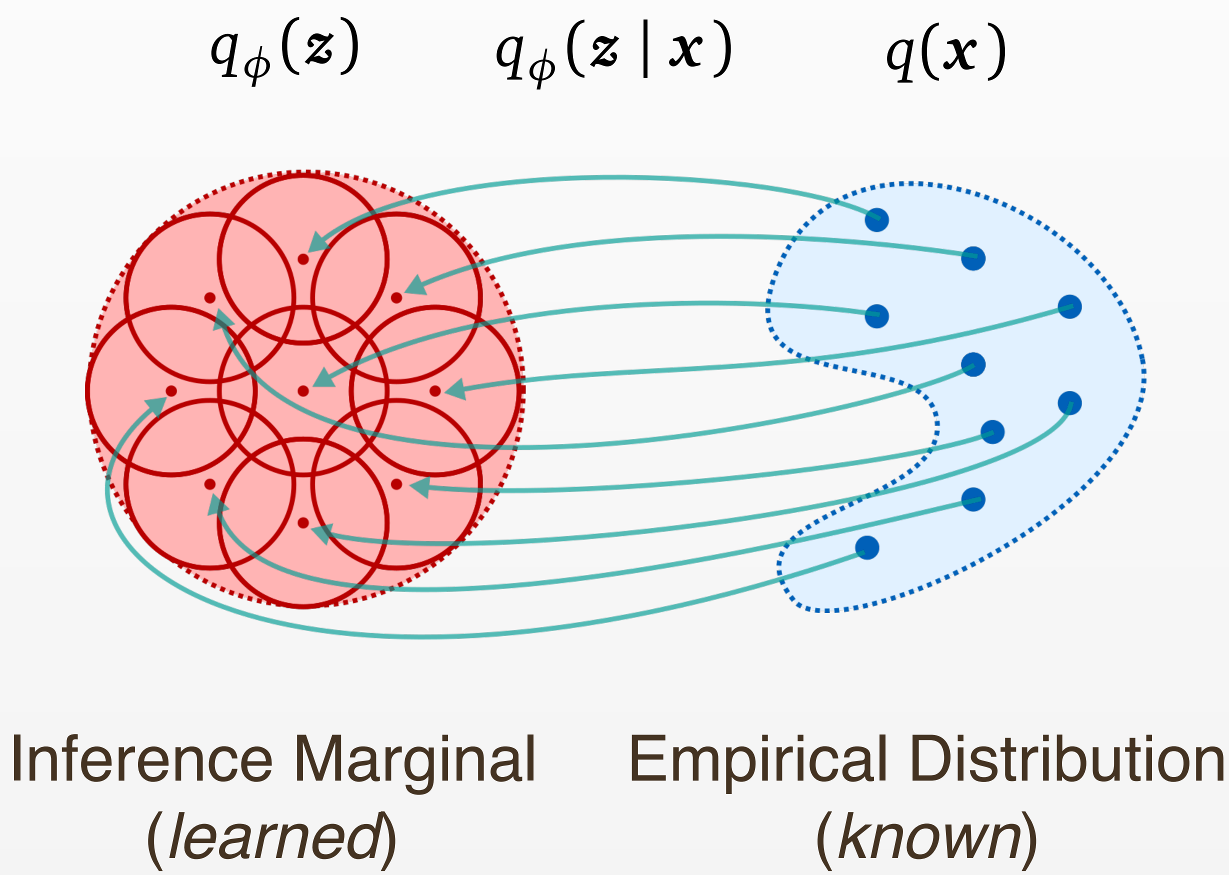
Empirical Distribution (*known*)

Deep Latent-Variable Models

Generative Model



Inference Model



Constraints: Models are Equivalent when

$p_\theta(\boldsymbol{x}) = q(\boldsymbol{x}) \qquad q_\phi(\boldsymbol{z}) = p(\boldsymbol{z}) \qquad p_\theta(\boldsymbol{z} \mid \boldsymbol{x}) = p_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \ \forall \ \boldsymbol{x}$

Implicit Tradeoffs in the Variational Objective

Classical View: Maximize Evidence Lower Bound

$$\mathcal{L}(\theta,\phi) = \mathbb{E}_{q(\boldsymbol{x})}\Big[\underbrace{\mathbb{E}_{q_\phi(\boldsymbol{z}\mid\boldsymbol{x})}[\log p_\theta(\boldsymbol{x} \mid \boldsymbol{z})]}_{\text{Reconstruction Error}} - \underbrace{\mathrm{KL}(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \,||\, p(\boldsymbol{z}))}_{\text{KL Regularization}} \Big]$$

Implicit Tradeoffs in the Variational Objective

Classical View: Maximize Evidence Lower Bound

$$\mathscr{L}(\theta, \phi) = \mathbb{E}_{q(\boldsymbol{x})}\Big[\underbrace{\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x} \mid \boldsymbol{z})]}_{\text{Reconstruction Error}} - \underbrace{\mathrm{KL}(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \,||\, p(\boldsymbol{z}))}_{\text{KL Regularization}} \Big]$$

Alternate View: KL Divergence Between Two Models

Implicit Tradeoffs in the Variational Objective

Classical View: Maximize Evidence Lower Bound

$$\mathcal{L}(\theta,\phi) = \mathbb{E}_{q(\boldsymbol{x})}\Big[\underbrace{\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x} \mid \boldsymbol{z})]}_{\text{Reconstruction Error}} - \underbrace{\mathrm{KL}(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \,||\, p(\boldsymbol{z}))}_{\text{KL Regularization}} \Big]$$

Alternate View: KL Divergence Between Two Models

$$\mathcal{L}(\theta,\phi) = \mathbb{E}_{q_\phi(\boldsymbol{x},\boldsymbol{z})}\left[\log \frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})}\right]$$

Implicit Tradeoffs in the Variational Objective

Classical View: Maximize Evidence Lower Bound

$$\mathcal{L}(\theta,\phi) = \mathbb{E}_{q(\boldsymbol{x})}\Big[\underbrace{\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x} \mid \boldsymbol{z})]}_{\text{Reconstruction Error}} - \underbrace{\mathrm{KL}(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \,||\, p(\boldsymbol{z}))}_{\text{KL Regularization}}\Big]$$

Alternate View: KL Divergence Between Two Models

$$\mathcal{L}(\theta,\phi) = \mathbb{E}_{q_\phi(\boldsymbol{x},\boldsymbol{z})}\left[\log \frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})}\right] = \mathbb{E}_{q_\phi(\boldsymbol{x},\boldsymbol{z})}\left[\log \frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})} + \log \frac{q(\boldsymbol{x})}{q(\boldsymbol{x})}\right]$$

Implicit Tradeoffs in the Variational Objective

Classical View: Maximize Evidence Lower Bound

$$\mathcal{L}(\theta,\phi) = \mathbb{E}_{q(\boldsymbol{x})}\Big[\underbrace{\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x} \mid \boldsymbol{z})]}_{\text{Reconstruction Error}} - \underbrace{\mathrm{KL}(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \,||\, p(\boldsymbol{z}))}_{\text{KL Regularization}} \Big]$$

Alternate View: KL Divergence Between Two Models

$$\mathcal{L}(\theta,\phi) = \mathbb{E}_{q_\phi(\boldsymbol{x},\boldsymbol{z})}\left[\log \frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})}\right] = \mathbb{E}_{q_\phi(\boldsymbol{x},\boldsymbol{z})}\left[\log \frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})} + \log \frac{q(\boldsymbol{x})}{q(\boldsymbol{x})}\right]$$

$$= \mathbb{E}_{q_\phi(\boldsymbol{x},\boldsymbol{z})}\left[\log \frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{q_\phi(\boldsymbol{x},\boldsymbol{z})}\right] + \mathbb{E}_{q(\boldsymbol{x})}[\log q(\boldsymbol{x})]$$

Implicit Tradeoffs in the Variational Objective

Classical View: Maximize Evidence Lower Bound

$$\mathscr{L}(\theta,\phi) = \mathbb{E}_{q(\boldsymbol{x})}\Big[\underbrace{\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x} \mid \boldsymbol{z})]}_{\text{Reconstruction Error}} - \underbrace{\mathrm{KL}(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \,||\, p(\boldsymbol{z}))}_{\text{KL Regularization}} \Big]$$

Alternate View: KL Divergence Between Two Models

$$\mathscr{L}(\theta,\phi) = \mathbb{E}_{q_\phi(\boldsymbol{x},\boldsymbol{z})}\left[\log \frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})}\right] = \mathbb{E}_{q_\phi(\boldsymbol{x},\boldsymbol{z})}\left[\log \frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})} + \log \frac{q(\boldsymbol{x})}{q(\boldsymbol{x})}\right]$$

$$= \underbrace{\mathbb{E}_{q_\phi(\boldsymbol{x},\boldsymbol{z})}\left[\log \frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{q_\phi(\boldsymbol{x},\boldsymbol{z})}\right]}_{-\,\mathrm{KL}(q_\phi(\boldsymbol{x},\boldsymbol{z})\,||\,p_\theta(\boldsymbol{x},\boldsymbol{z}))} + \mathbb{E}_{q(\boldsymbol{x})}[\log q(\boldsymbol{x})]$$

Implicit Tradeoffs in the Variational Objective

Classical View: Maximize Evidence Lower Bound

$$\mathcal{L}(\theta,\phi) = \mathbb{E}_{q(\boldsymbol{x})}\Big[\underbrace{\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x} \mid \boldsymbol{z})]}_{\text{Reconstruction Error}} - \underbrace{\mathrm{KL}(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \,||\, p(\boldsymbol{z}))}_{\text{KL Regularization}} \Big]$$

Alternate View: KL Divergence Between Two Models

$$\mathcal{L}(\theta,\phi) = \mathbb{E}_{q_\phi(\boldsymbol{x},\boldsymbol{z})}\left[\log \frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})}\right] = \mathbb{E}_{q_\phi(\boldsymbol{x},\boldsymbol{z})}\left[\log \frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})} + \log \frac{q(\boldsymbol{x})}{q(\boldsymbol{x})}\right]$$

$$= \underbrace{\mathbb{E}_{q_\phi(\boldsymbol{x},\boldsymbol{z})}\left[\log \frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{q_\phi(\boldsymbol{x},\boldsymbol{z})}\right]}_{-\,\mathrm{KL}\left(q_\phi(\boldsymbol{x},\boldsymbol{z})\,||\,p_\theta(\boldsymbol{x},\boldsymbol{z})\right)} + \underbrace{\mathbb{E}_{q(\boldsymbol{x})}[\log q(\boldsymbol{x})]}_{H_q(\boldsymbol{x}) \,=\, \log N}$$

Implicit Tradeoffs in the Variational Objective

$$\mathcal{L}(\theta, \phi) := -\mathrm{KL}\big(q_\phi(\boldsymbol{z}, \boldsymbol{x}) \,||\, p_\theta(\boldsymbol{x}, \boldsymbol{z})\big) = \mathbb{E}_{q_\phi(\boldsymbol{z}, \boldsymbol{x})}\left[\log \frac{p_\theta(\boldsymbol{x}, \boldsymbol{z})}{q_\phi(\boldsymbol{z}, \boldsymbol{x})}\right]$$

[Hoffman, Johnson, NIPS AABI Workshop 2016], [Chen, Li, Grosse, Duvenaud, Arxiv 2018]
[Esmaeli, Wu, Jain, Bozkurt, Siddharth, Paige, Brooks, Dy, van de Meent, Arxiv 2018]

Implicit Tradeoffs in the Variational Objective

$$\mathcal{L}(\theta,\phi) := -\mathrm{KL}\Big(q_\phi(\boldsymbol{z},\boldsymbol{x}) \,\Big|\Big|\, p_\theta(\boldsymbol{x},\boldsymbol{z})\Big) = \mathbb{E}_{q_\phi(\boldsymbol{z},\boldsymbol{x})}\left[\log \frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{q_\phi(\boldsymbol{z},\boldsymbol{x})}\right]$$

$$= \mathbb{E}_{q_\phi(\boldsymbol{z},\boldsymbol{x})}\left[\log \frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{p_\theta(\boldsymbol{x})p(\boldsymbol{z})} + \log \frac{q_\phi(\boldsymbol{z})q(\boldsymbol{x})}{q_\phi(\boldsymbol{z},\boldsymbol{x})} + \log \frac{p_\theta(\boldsymbol{x})}{q(\boldsymbol{x})} + \log \frac{p(\boldsymbol{z})}{q_\phi(\boldsymbol{z})}\right]$$

[Hoffman, Johnson, NIPS AABI Workshop 2016], [Chen, Li, Grosse, Duvenaud, Arxiv 2018]
[Esmaeli, Wu, Jain, Bozkurt, Siddharth, Paige, Brooks, Dy, van de Meent, Arxiv 2018]

Implicit Tradeoffs in the Variational Objective

$$\mathcal{L}(\theta,\phi) := -\mathrm{KL}\big(q_\phi(\boldsymbol{z},\boldsymbol{x}) \,||\, p_\theta(\boldsymbol{x},\boldsymbol{z})\big) = \mathbb{E}_{q_\phi(\boldsymbol{z},\boldsymbol{x})}\left[\log \frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{q_\phi(\boldsymbol{z},\boldsymbol{x})}\right]$$

$$= \mathbb{E}_{q_\phi(\boldsymbol{z},\boldsymbol{x})}\left[\log \frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{p_\theta(\boldsymbol{x})p(\boldsymbol{z})} + \log \frac{q_\phi(\boldsymbol{z})q(\boldsymbol{x})}{q_\phi(\boldsymbol{z},\boldsymbol{x})} + \log \frac{p_\theta(\boldsymbol{x})}{q(\boldsymbol{x})} + \log \frac{p(\boldsymbol{z})}{q_\phi(\boldsymbol{z})}\right]$$

$$= \mathbb{E}_{q_\phi(\boldsymbol{z},\boldsymbol{x})}\Bigg[\underbrace{\log \frac{p_\theta(\boldsymbol{z}\mid\boldsymbol{x})}{p(\boldsymbol{z})}}_{①} - \underbrace{\log \frac{q_\phi(\boldsymbol{z}\mid\boldsymbol{x})}{q_\phi(\boldsymbol{z})}}_{②}\Bigg] \underbrace{- \mathrm{KL}\big(q(\boldsymbol{x}) \,||\, p_\theta(\boldsymbol{x})\big)}_{③} \underbrace{- \mathrm{KL}\big(q_\phi(\boldsymbol{z}) \,||\, p(\boldsymbol{z})\big)}_{④}$$

[Hoffman, Johnson, NIPS AABI Workshop 2016], [Chen, Li, Grosse, Duvenaud, Arxiv 2018]
[Esmaeli, Wu, Jain, Bozkurt, Siddharth, Paige, Brooks, Dy, van de Meent, Arxiv 2018]

Implicit Tradeoffs in the Variational Objective

$$\mathcal{L}(\theta,\phi) := -\mathrm{KL}\big(q_\phi(\boldsymbol{z},\boldsymbol{x}) \,\big|\big|\, p_\theta(\boldsymbol{x},\boldsymbol{z})\big) = \mathbb{E}_{q_\phi(\boldsymbol{z},\boldsymbol{x})}\left[\log \frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{q_\phi(\boldsymbol{z},\boldsymbol{x})}\right]$$

$$= \mathbb{E}_{q_\phi(\boldsymbol{z},\boldsymbol{x})}\left[\log \frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{p_\theta(\boldsymbol{x})p(\boldsymbol{z})} + \log \frac{q_\phi(\boldsymbol{z})q(\boldsymbol{x})}{q_\phi(\boldsymbol{z},\boldsymbol{x})} + \log \frac{p_\theta(\boldsymbol{x})}{q(\boldsymbol{x})} + \log \frac{p(\boldsymbol{z})}{q_\phi(\boldsymbol{z})}\right]$$

$$= \mathbb{E}_{q_\phi(\boldsymbol{z},\boldsymbol{x})}\Bigg[\underbrace{\log \frac{p_\theta(\boldsymbol{z}\mid\boldsymbol{x})}{p(\boldsymbol{z})}}_{①} - \underbrace{\log \frac{q_\phi(\boldsymbol{z}\mid\boldsymbol{x})}{q_\phi(\boldsymbol{z})}}_{②}\Bigg] \underbrace{- \mathrm{KL}\big(q(\boldsymbol{x}) \,||\, p_\theta(\boldsymbol{x})\big)}_{③} \underbrace{- \mathrm{KL}\big(q_\phi(\boldsymbol{z}) \,\big|\big|\, p(\boldsymbol{z})\big)}_{④}$$

$$p_\theta(\boldsymbol{x}) = q(\boldsymbol{x})$$

[Hoffman, Johnson, NIPS AABI Workshop 2016], [Chen, Li, Grosse, Duvenaud, Arxiv 2018]
[Esmaeli, Wu, Jain, Bozkurt, Siddharth, Paige, Brooks, Dy, van de Meent, Arxiv 2018]

Implicit Tradeoffs in the Variational Objective

$$\mathcal{L}(\theta,\phi) := -\mathrm{KL}\big(q_\phi(\boldsymbol{z},\boldsymbol{x}) \,\big|\big|\, p_\theta(\boldsymbol{x},\boldsymbol{z})\big) = \mathbb{E}_{q_\phi(\boldsymbol{z},\boldsymbol{x})}\left[\log \frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{q_\phi(\boldsymbol{z},\boldsymbol{x})}\right]$$

$$= \mathbb{E}_{q_\phi(\boldsymbol{z},\boldsymbol{x})}\left[\log \frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{p_\theta(\boldsymbol{x})p(\boldsymbol{z})} + \log \frac{q_\phi(\boldsymbol{z})q(\boldsymbol{x})}{q_\phi(\boldsymbol{z},\boldsymbol{x})} + \log \frac{p_\theta(\boldsymbol{x})}{q(\boldsymbol{x})} + \log \frac{p(\boldsymbol{z})}{q_\phi(\boldsymbol{z})}\right]$$

$$= \mathbb{E}_{q_\phi(\boldsymbol{z},\boldsymbol{x})}\Big[\underbrace{\log \frac{p_\theta(\boldsymbol{z} \mid \boldsymbol{x})}{p(\boldsymbol{z})}}_{①} \underbrace{- \log \frac{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})}{q_\phi(\boldsymbol{z})}}_{②}\Big] \underbrace{- \mathrm{KL}\big(q(\boldsymbol{x}) \,||\, p_\theta(\boldsymbol{x})\big)}_{③} \underbrace{- \mathrm{KL}\big(q_\phi(\boldsymbol{z}) \,\big|\big|\, p(\boldsymbol{z})\big)}_{④}$$

$$p_\theta(\boldsymbol{x}) = q(\boldsymbol{x}) \qquad q_\phi(\boldsymbol{z}) = p(\boldsymbol{z})$$

[Hoffman, Johnson, NIPS AABI Workshop 2016], [Chen, Li, Grosse, Duvenaud, Arxiv 2018]
[Esmaeli, Wu, Jain, Bozkurt, Siddharth, Paige, Brooks, Dy, van de Meent, Arxiv 2018]

Implicit Tradeoffs in the Variational Objective

$$\mathcal{L}(\theta, \phi) := -\mathrm{KL}\big(q_\phi(\boldsymbol{z}, \boldsymbol{x}) \,||\, p_\theta(\boldsymbol{x}, \boldsymbol{z})\big) = \mathbb{E}_{q_\phi(\boldsymbol{z},\boldsymbol{x})}\left[\log \frac{p_\theta(\boldsymbol{x}, \boldsymbol{z})}{q_\phi(\boldsymbol{z}, \boldsymbol{x})}\right]$$

$$= \mathbb{E}_{q_\phi(\boldsymbol{z},\boldsymbol{x})}\left[\log \frac{p_\theta(\boldsymbol{x}, \boldsymbol{z})}{p_\theta(\boldsymbol{x})p(\boldsymbol{z})} + \log \frac{q_\phi(\boldsymbol{z})q(\boldsymbol{x})}{q_\phi(\boldsymbol{z}, \boldsymbol{x})} + \log \frac{p_\theta(\boldsymbol{x})}{q(\boldsymbol{x})} + \log \frac{p(\boldsymbol{z})}{q_\phi(\boldsymbol{z})}\right]$$

$$= \mathbb{E}_{q_\phi(\boldsymbol{z},\boldsymbol{x})}\Bigg[\underbrace{\log \frac{p_\theta(\boldsymbol{z} \mid \boldsymbol{x})}{p(\boldsymbol{z})}}_{①} \underbrace{- \log \frac{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})}{q_\phi(\boldsymbol{z})}}_{②}\Bigg] \underbrace{- \mathrm{KL}\big(q(\boldsymbol{x}) \,||\, p_\theta(\boldsymbol{x})\big)}_{③} \underbrace{- \mathrm{KL}\big(q_\phi(\boldsymbol{z}) \,||\, p(\boldsymbol{z})\big)}_{④}$$

$$p_\theta(\boldsymbol{z} \mid \boldsymbol{x}) = p_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \;\forall\, \boldsymbol{x} \qquad p_\theta(\boldsymbol{x}) = q(\boldsymbol{x}) \qquad q_\phi(\boldsymbol{z}) = p(\boldsymbol{z})$$

[Hoffman, Johnson, NIPS AABI Workshop 2016], [Chen, Li, Grosse, Duvenaud, Arxiv 2018]
[Esmaeli, Wu, Jain, Bozkurt, Siddharth, Paige, Brooks, Dy, van de Meent, Arxiv 2018]

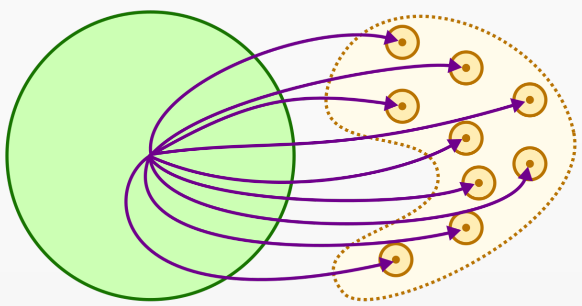
Relaxing Constraints in the Objective

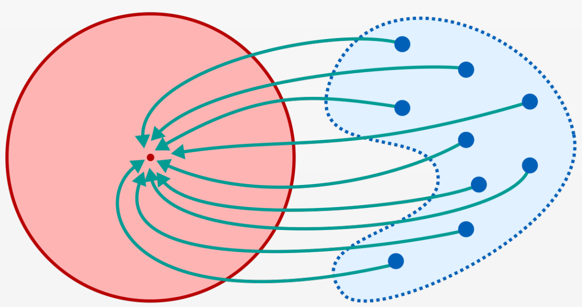
$$\mathcal{L}(\theta, \phi) = \mathbb{E}_{q_\phi(\boldsymbol{z},\boldsymbol{x})}\Bigg[\underbrace{\log \frac{p_\theta(\boldsymbol{z} \mid \boldsymbol{x})}{p(\boldsymbol{z})}}_{①} - \underbrace{\log \frac{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})}{q_\phi(\boldsymbol{z})}}_{②}\Bigg] - \underbrace{\mathrm{KL}(q(\boldsymbol{x}) \,||\, p_\theta(\boldsymbol{x}))}_{③} - \underbrace{\mathrm{KL}\big(q_\phi(\boldsymbol{z}) \,\big|\big|\, p(\boldsymbol{z})\big)}_{④}$$

[Esmaeli, Wu, Jain, Bozkurt, Siddharth, Paige, Brooks, Dy, van de Meent, Arxiv 2018]

Relaxing Constraints in the Objective

$$\mathcal{L}(\theta,\phi) = \mathbb{E}_{q_\phi(z,x)}\Bigg[\underbrace{\log\frac{p_\theta(z \mid x)}{p(z)}}_{①} - \underbrace{\log\frac{q_\phi(z \mid x)}{q_\phi(z)}}_{②}\Bigg] - \underbrace{\mathrm{KL}(q(x) \,||\, p_\theta(x))}_{③} - \underbrace{\mathrm{KL}\big(q_\phi(z) \,\big|\big|\, p(z)\big)}_{④}$$



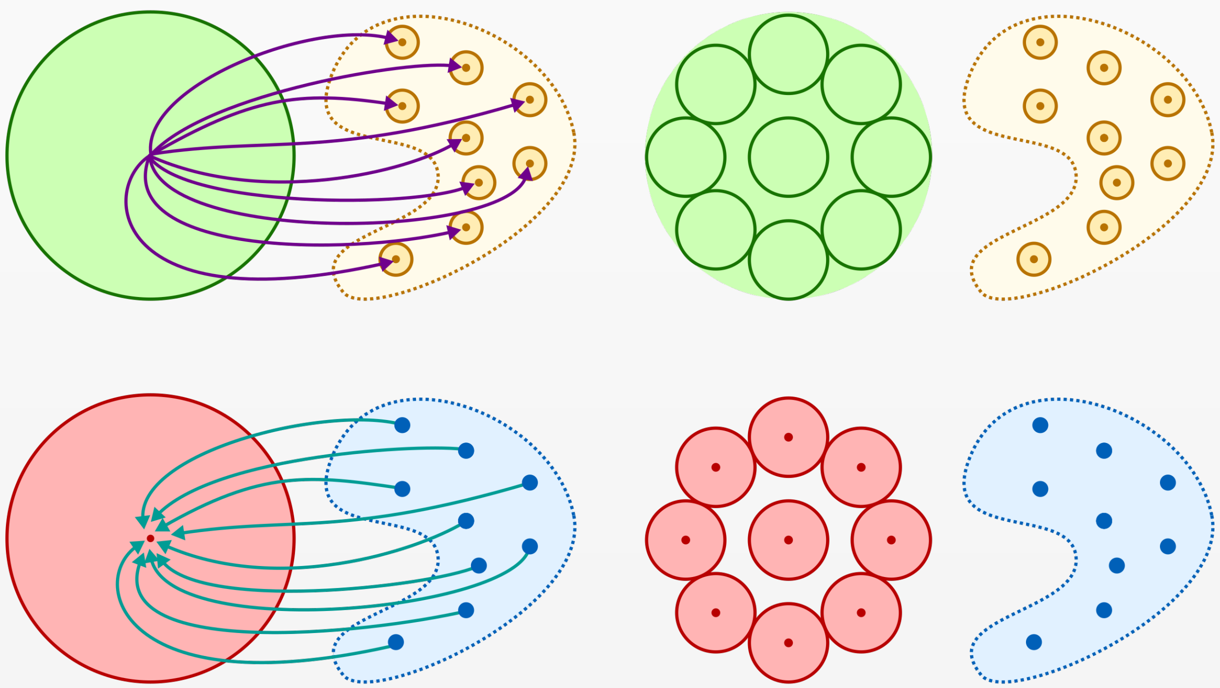


② + ③ + ④

[Esmaeli, Wu, Jain, Bozkurt, Siddharth, Paige, Brooks, Dy, van de Meent, Arxiv 2018]

Relaxing Constraints in the Objective

$$\mathcal{L}(\theta,\phi) = \mathbb{E}_{q_\phi(z,x)}\Bigg[\underbrace{\log\frac{p_\theta(z \mid x)}{p(z)}}_{①} \underbrace{- \log\frac{q_\phi(z \mid x)}{q_\phi(z)}}_{②}\Bigg] \underbrace{- \mathrm{KL}(q(x) \,||\, p_\theta(x))}_{③} \underbrace{- \mathrm{KL}\big(q_\phi(z) \,||\, p(z)\big)}_{④}$$



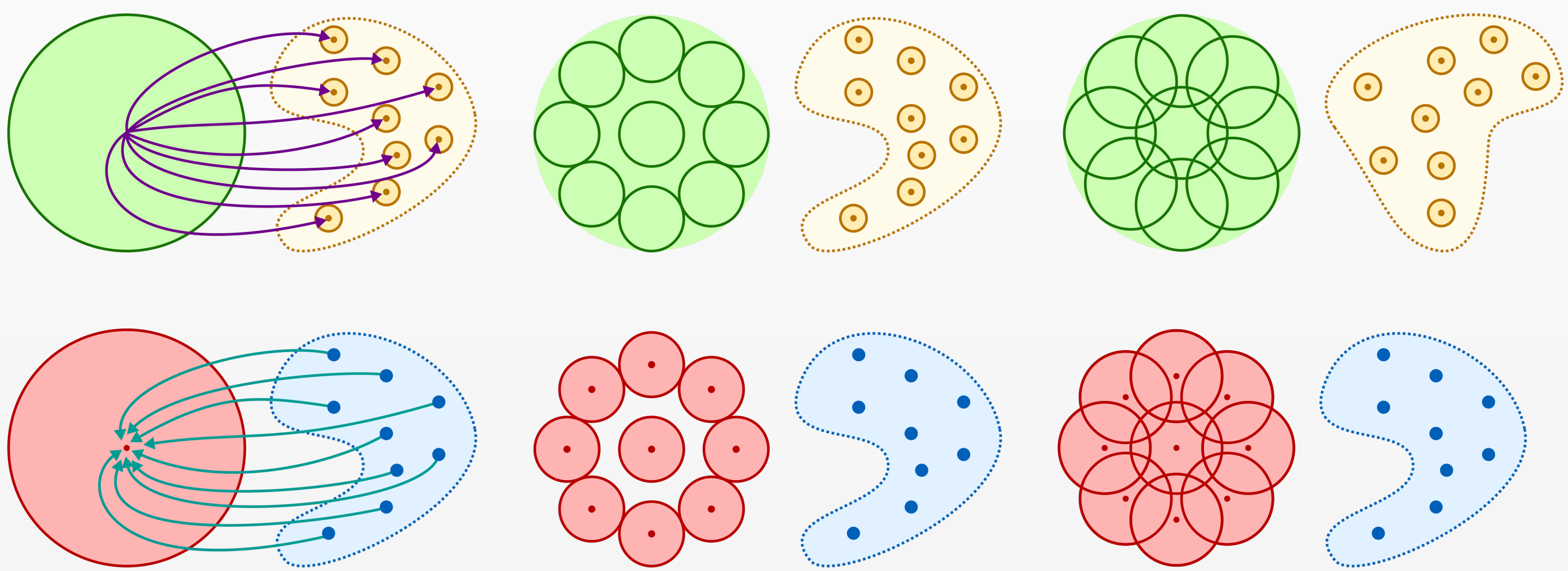
② + ③ + ④

① + ③ + ④

[Esmaeli, Wu, Jain, Bozkurt, Siddharth, Paige, Brooks, Dy, van de Meent, Arxiv 2018]

Relaxing Constraints in the Objective

$$\mathcal{L}(\theta,\phi) = \mathbb{E}_{q_\phi(\boldsymbol{z},\boldsymbol{x})}\left[\underbrace{\log\frac{p_\theta(\boldsymbol{z}\mid \boldsymbol{x})}{p(\boldsymbol{z})}}_{\text{①}} - \underbrace{\log\frac{q_\phi(\boldsymbol{z}\mid \boldsymbol{x})}{q_\phi(\boldsymbol{z})}}_{\text{②}}\right] - \underbrace{\mathrm{KL}(q(\boldsymbol{x})\,||\,p_\theta(\boldsymbol{x}))}_{\text{③}} - \underbrace{\mathrm{KL}\big(q_\phi(\boldsymbol{z})\,\big|\big|\,p(\boldsymbol{z})\big)}_{\text{④}}$$

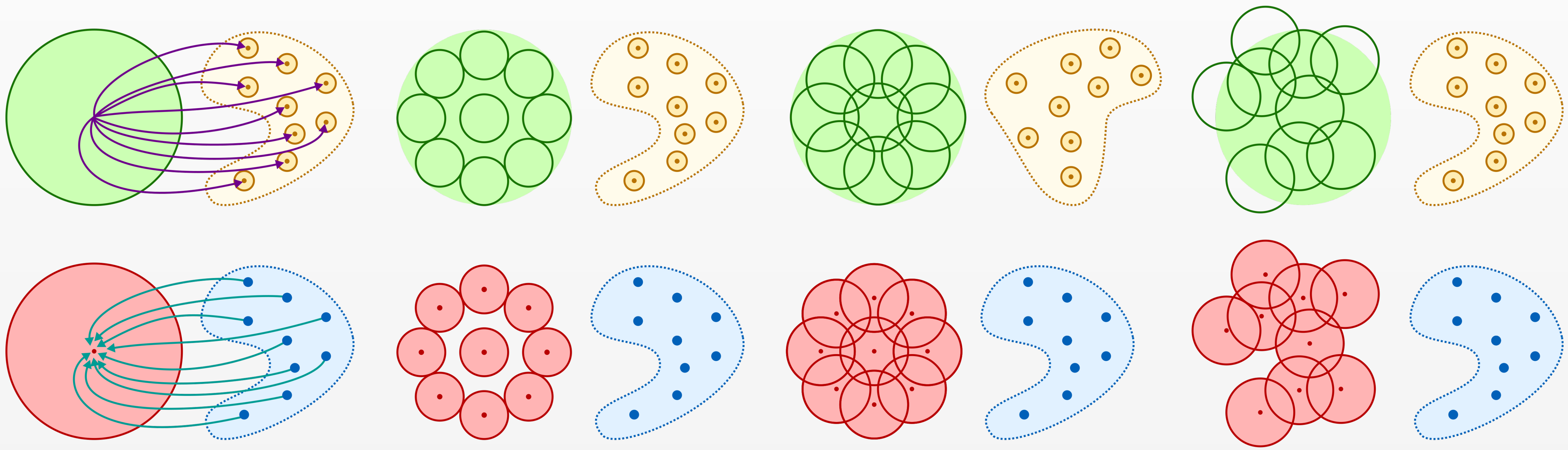


② + ③ + ④ | ① + ③ + ④ | ① + ② + ④

[Esmaeli, Wu, Jain, Bozkurt, Siddharth, Paige, Brooks, Dy, van de Meent, Arxiv 2018]

Relaxing Constraints in the Objective

$$\mathcal{L}(\theta,\phi) = \mathbb{E}_{q_\phi(z,x)}\Bigg[\underbrace{\log \frac{p_\theta(z \mid x)}{p(z)}}_{①} - \underbrace{\log \frac{q_\phi(z \mid x)}{q_\phi(z)}}_{②}\Bigg] - \underbrace{\mathrm{KL}(q(x) \,||\, p_\theta(x))}_{③} - \underbrace{\mathrm{KL}\big(q_\phi(z) \,||\, p(z)\big)}_{④}$$

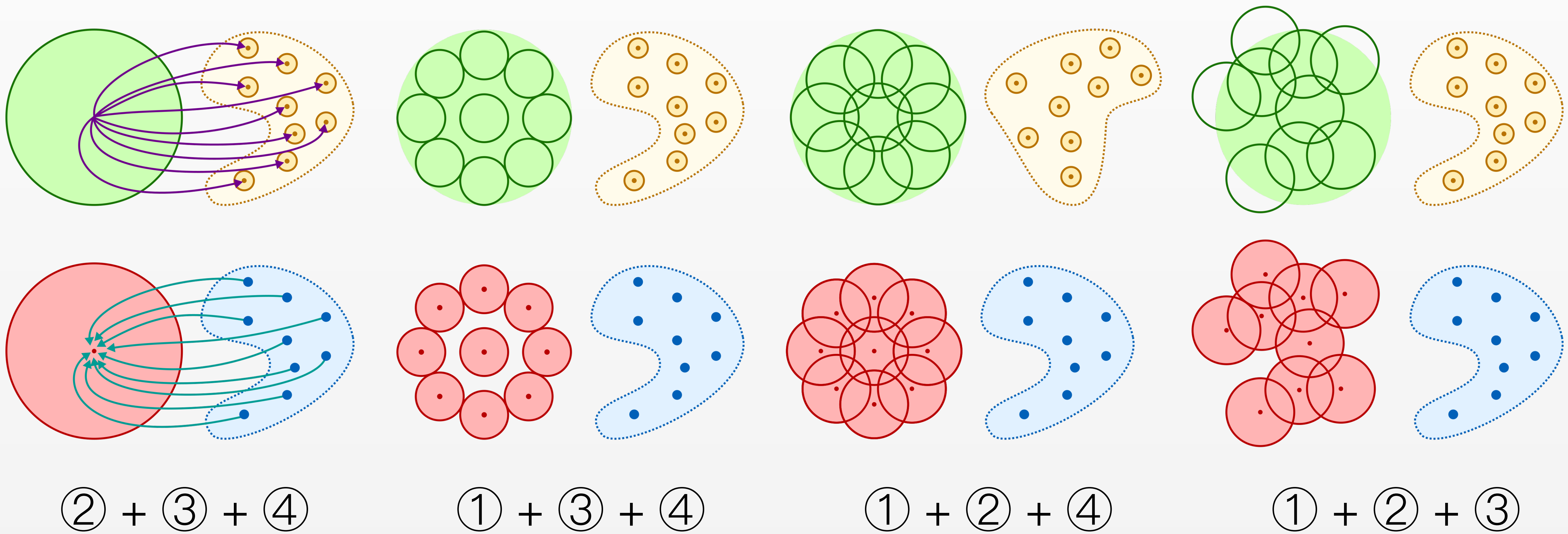


② + ③ + ④ ① + ③ + ④ ① + ② + ④ ① + ② + ③

[Esmaeli, Wu, Jain, Bozkurt, Siddharth, Paige, Brooks, Dy, van de Meent, Arxiv 2018]

Relaxing Constraints in the Objective

$$\mathcal{L}(\theta,\phi) = \mathbb{E}_{q_\phi(z,x)}\Big[\underbrace{\log \frac{p_\theta(z \mid x)}{p(z)}}_{①} \underbrace{- \log \frac{q_\phi(z \mid x)}{q_\phi(z)}}_{②}\Big] \underbrace{- \mathrm{KL}(q(x) \,||\, p_\theta(x))}_{③} \underbrace{- \mathrm{KL}\big(q_\phi(z) \,||\, p(z)\big)}_{④}$$



Idea: Relax ①, ②, and ③ in favor of ④

[Esmaeli, Wu, Jain, Bozkurt, Siddharth, Paige, Brooks, Dy, van de Meent, Arxiv 2018]

Hierarchical Decomposition

$$-\mathrm{KL}\left(q_\phi(\boldsymbol{z}) \,\|\, p(\boldsymbol{z})\right) = -\mathbb{E}_{q_\phi(\boldsymbol{z})}\left[\log \frac{q_\phi(\boldsymbol{z})}{\prod_d q_\phi(\boldsymbol{z}_d)} + \log \frac{\prod_d q_\phi(\boldsymbol{z}_d)}{\prod_d p(\boldsymbol{z}_d)} + \log \frac{\prod_d p(\boldsymbol{z}_d)}{p(\boldsymbol{z})}\right]$$

[Esmaeli, Wu, Jain, Bozkurt, Siddharth, Paige, Brooks, Dy, van de Meent, Arxiv 2018]

Hierarchical Decomposition

$$-\mathrm{KL}\big(q_\phi(\boldsymbol{z})\,\|\,p(\boldsymbol{z})\big) = -\mathbb{E}_{q_\phi(\boldsymbol{z})}\left[\log \frac{q_\phi(\boldsymbol{z})}{\prod_d q_\phi(\boldsymbol{z}_d)} + \log \frac{\prod_d q_\phi(\boldsymbol{z}_d)}{\prod_d p(\boldsymbol{z}_d)} + \log \frac{\prod_d p(\boldsymbol{z}_d)}{p(\boldsymbol{z})}\right]$$

$$= \underbrace{\mathbb{E}_{q_\phi(\boldsymbol{z})}\left[\log \frac{p(\boldsymbol{z})}{\prod_d p(\boldsymbol{z}_d)} - \log \frac{q_\phi(\boldsymbol{z})}{\prod_d q_\phi(\boldsymbol{z}_d)}\right]}_{\text{Ⓐ}} - \sum_d \underbrace{\mathrm{KL}\big(q_\phi(\boldsymbol{z}_d)\,\|\,p(\boldsymbol{z}_d)\big)}_{\text{Ⓑ}}.$$

[Esmaeli, Wu, Jain, Bozkurt, Siddharth, Paige, Brooks, Dy, van de Meent, Arxiv 2018]

Hierarchical Decomposition

$$-\mathrm{KL}\left(q_\phi(\boldsymbol{z}) \,\|\, p(\boldsymbol{z})\right) = -\mathbb{E}_{q_\phi(\boldsymbol{z})}\left[\log \frac{q_\phi(\boldsymbol{z})}{\prod_d q_\phi(\boldsymbol{z}_d)} + \log \frac{\prod_d q_\phi(\boldsymbol{z}_d)}{\prod_d p(\boldsymbol{z}_d)} + \log \frac{\prod_d p(\boldsymbol{z}_d)}{p(\boldsymbol{z})}\right]$$

$$= \underbrace{\mathbb{E}_{q_\phi(\boldsymbol{z})}\left[\log \frac{p(\boldsymbol{z})}{\prod_d p(\boldsymbol{z}_d)} - \log \frac{q_\phi(\boldsymbol{z})}{\prod_d q_\phi(\boldsymbol{z}_d)}\right]}_{\text{Ⓐ}} - \sum_d \underbrace{\mathrm{KL}\left(q_\phi(\boldsymbol{z}_d) \,\|\, p(\boldsymbol{z}_d)\right)}_{\text{Ⓑ}}.$$

Marginals should be identical

[Esmaeli, Wu, Jain, Bozkurt, Siddharth, Paige, Brooks, Dy, van de Meent, Arxiv 2018]

Hierarchical Decomposition

$$-\mathrm{KL}\left(q_\phi(\boldsymbol{z}) \,\|\, p(\boldsymbol{z})\right) = -\mathbb{E}_{q_\phi(\boldsymbol{z})}\left[\log \frac{q_\phi(\boldsymbol{z})}{\prod_d q_\phi(\boldsymbol{z}_d)} + \log \frac{\prod_d q_\phi(\boldsymbol{z}_d)}{\prod_d p(\boldsymbol{z}_d)} + \log \frac{\prod_d p(\boldsymbol{z}_d)}{p(\boldsymbol{z})}\right]$$

$$= \underbrace{\mathbb{E}_{q_\phi(\boldsymbol{z})}\left[\log \frac{p(\boldsymbol{z})}{\prod_d p(\boldsymbol{z}_d)} - \log \frac{q_\phi(\boldsymbol{z})}{\prod_d q_\phi(\boldsymbol{z}_d)}\right]}_{\text{Ⓐ}} - \sum_d \underbrace{\mathrm{KL}\left(q_\phi(\boldsymbol{z}_d) \,\|\, p(\boldsymbol{z}_d)\right)}_{\text{Ⓑ}}.$$

Ⓐ Correlations between variables should be identical

Ⓑ Marginals should be identical

[Esmaeli, Wu, Jain, Bozkurt, Siddharth, Paige, Brooks, Dy, van de Meent, Arxiv 2018]

Hierarchical Decomposition

$$-\mathrm{KL}\big(q_\phi(\mathbf{z})\,\|\,p(\mathbf{z})\big) = -\mathbb{E}_{q_\phi(\mathbf{z})}\left[\log\frac{q_\phi(\mathbf{z})}{\prod_d q_\phi(\mathbf{z}_d)} + \log\frac{\prod_d q_\phi(\mathbf{z}_d)}{\prod_d p(\mathbf{z}_d)} + \log\frac{\prod_d p(\mathbf{z}_d)}{p(\mathbf{z})}\right]$$

$$= \underbrace{\mathbb{E}_{q_\phi(\mathbf{z})}\left[\log\frac{p(\mathbf{z})}{\prod_d p(\mathbf{z}_d)} - \log\frac{q_\phi(\mathbf{z})}{\prod_d q_\phi(\mathbf{z}_d)}\right]}_{\text{Ⓐ}} - \sum_d \underbrace{\mathrm{KL}\big(q_\phi(\mathbf{z}_d)\,\|\,p(\mathbf{z}_d)\big)}_{\text{Ⓑ}}.$$

Ⓐ Correlations between variables should be identical

Ⓑ Marginals should be identical

Disentanglement: $p(\mathbf{z}) = \prod_d p(\mathbf{z}_d)$ $\quad q_\phi(\mathbf{z}) = \prod_d q_\phi(\mathbf{z}_d)$

[Esmaeli, Wu, Jain, Bozkurt, Siddharth, Paige, Brooks, Dy, van de Meent, Arxiv 2018]

Hierarchical Decomposition

$$-\mathrm{KL}\left(q_\phi(\boldsymbol{z}) \,\|\, p(\boldsymbol{z})\right) = -\mathbb{E}_{q_\phi(\boldsymbol{z})}\left[\log \frac{q_\phi(\boldsymbol{z})}{\prod_d q_\phi(\boldsymbol{z}_d)} + \log \frac{\prod_d q_\phi(\boldsymbol{z}_d)}{\prod_d p(\boldsymbol{z}_d)} + \log \frac{\prod_d p(\boldsymbol{z}_d)}{p(\boldsymbol{z})}\right]$$

$$= \underbrace{\mathbb{E}_{q_\phi(\boldsymbol{z})}\left[\log \frac{p(\boldsymbol{z})}{\prod_d p(\boldsymbol{z}_d)} - \log \frac{q_\phi(\boldsymbol{z})}{\prod_d q_\phi(\boldsymbol{z}_d)}\right]}_{\text{Ⓐ}} - \sum_d \underbrace{\mathrm{KL}\left(q_\phi(\boldsymbol{z}_d) \,\|\, p(\boldsymbol{z}_d)\right)}_{\text{Ⓑ}}.$$

Ⓐ Correlations between variables should be identical

Ⓑ Marginals should be identical

Disentanglement: $p(\boldsymbol{z}) = \prod_d p(\boldsymbol{z}_d) \quad q_\phi(\boldsymbol{z}) = \prod_d q_\phi(\boldsymbol{z}_d)$

$$\text{Ⓑ} = \underbrace{\mathbb{E}_{q_\phi(\boldsymbol{z}_d)}\left[\log \frac{p(\boldsymbol{z}_d)}{\prod_e p(\boldsymbol{z}_{d,e})} - \log \frac{q_\phi(\boldsymbol{z}_d)}{\prod_e q_\phi(\boldsymbol{z}_{d,e})}\right]}_{\text{ⓘ}} - \sum_e \underbrace{\mathrm{KL}\left(q_\phi(\boldsymbol{z}_{d,e}) \,\|\, p(\boldsymbol{z}_{d,e})\right)}_{\text{ⓘⓘ}}$$

(Can continue decomposition for any number of levels)

[Esmaeli, Wu, Jain, Bozkurt, Siddharth, Paige, Brooks, Dy, van de Meent, Arxiv 2018]

Generalizations of VAE objectives

Hierarchically Factorized Variational Autoencoders

$$\mathcal{L}(\theta, \phi) = \text{①} + \text{③} + \text{ⓘⓘ} + \alpha\,\text{②} + \beta\,\text{Ⓐ} + \gamma\,\text{ⓘ}$$

Generalizations of VAE objectives

Hierarchically Factorized Variational Autoencoders

$$\mathscr{L}(\theta,\phi) = \textcircled{1} + \textcircled{3} + \textcircled{\scriptsize ii} + \alpha\,\textcircled{2} + \beta\,\textcircled{\scriptsize A} + \gamma\,\textcircled{\scriptsize i}$$

Paper	**Objective**
Kingma and Welling [2013], Rezende et al. [2014]	$\textcircled{1} + \textcircled{2} + \textcircled{3} + \textcircled{4}$
Higgins et al. [2016]	$\textcircled{1} + \textcircled{3} + \beta\,(\textcircled{2} + \textcircled{4})$
Kumar et al. [2017]	$\textcircled{1} + \textcircled{2} + \textcircled{3} + \lambda\,\textcircled{4}$
Zhao et al. [2017]	$\textcircled{1} + \textcircled{3} + \lambda\,\textcircled{4}$
Gao et al. [2018]	$\textcircled{1} + \textcircled{2} + \textcircled{3} + \textcircled{4} - \lambda\,\textcircled{2}^{a}$
Achille and Soatto [2018]	$\textcircled{1} + \textcircled{3} + \beta\,\textcircled{2} + \gamma\,\textcircled{\scriptsize A}^{*}$
Kim and Mnih [2018],Chen et al. [2018]	$\textcircled{1} + \textcircled{2} + \textcircled{3} + \textcircled{\scriptsize B} + \beta\,\textcircled{\scriptsize A}^{*}$
HFVAE (this paper)	$\textcircled{1} + \textcircled{3} + \textcircled{\scriptsize ii} + \alpha\,\textcircled{2} + \beta\,\textcircled{\scriptsize A} + \gamma\,\textcircled{\scriptsize i}$

Generalizations of VAE objectives

Hierarchically Factorized Variational Autoencoders

$$\mathscr{L}(\theta,\phi) = \text{①} + \text{③} + \text{ⓘⓘ} + \alpha\,\text{②} + \beta\,\text{Ⓐ} + \gamma\,\text{ⓘ}$$

Paper	**Objective**
Kingma and Welling [2013], Rezende et al. [2014]	① + ② + ③ + ④
Higgins et al. [2016]	① + ③ + β (② + ④)
Kumar et al. [2017]	① + ② + ③ + λ ④
Zhao et al. [2017]	① + ③ + λ ④
Gao et al. [2018]	① + ② + ③ + ④ − λ ②[a]
Achille and Soatto [2018]	① + ③ + β ② + γ Ⓐ*
Kim and Mnih [2018],Chen et al. [2018]	① + ② + ③ + Ⓑ + β Ⓐ*
HFVAE (this paper)	① + ③ + ⓘⓘ + α ② + β Ⓐ + γ ⓘ

Generalizations of VAE objectives

Hierarchically Factorized Variational Autoencoders

$$\mathscr{L}(\theta, \phi) = \textcircled{1} + \textcircled{3} + \textcircled{ii} + \alpha\,\textcircled{2} + \beta\,\textcircled{A} + \gamma\,\textcircled{i}$$

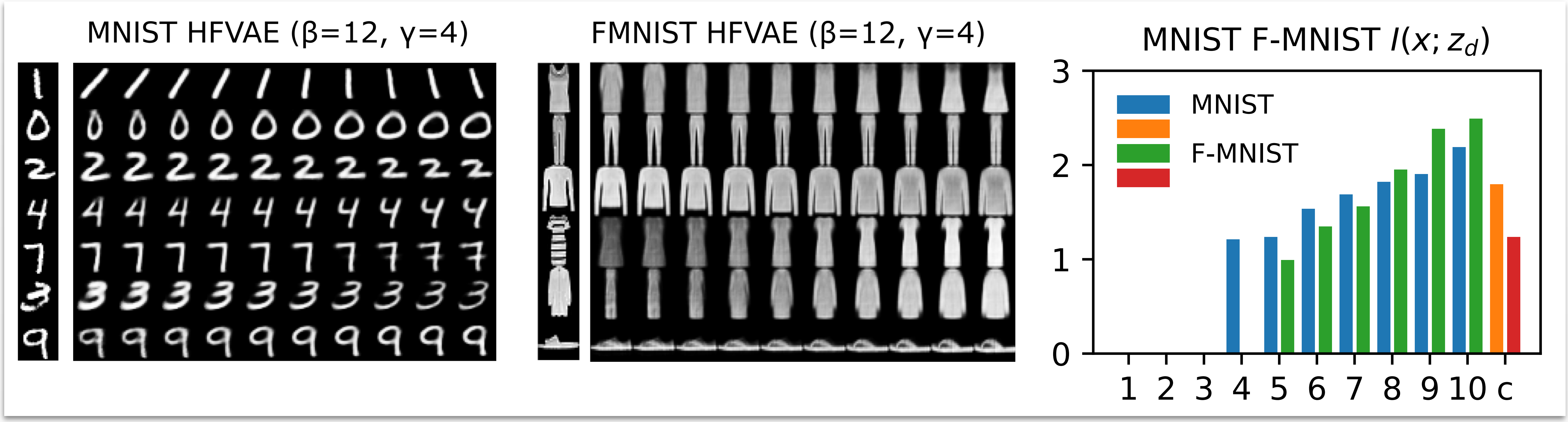
Paper	**Objective**
Kingma and Welling [2013], Rezende et al. [2014]	$\textcircled{1} + \textcircled{2} + \textcircled{3} + \textcircled{4}$
Higgins et al. [2016]	$\textcircled{1} + \textcircled{3} + \beta\,(\textcircled{2} + \textcircled{4})$
Kumar et al. [2017]	$\textcircled{1} + \textcircled{2} + \textcircled{3} + \lambda\,\textcircled{4}$
Zhao et al. [2017]	$\textcircled{1} + \textcircled{3} + \lambda\,\textcircled{4}$
Gao et al. [2018]	$\textcircled{1} + \textcircled{2} + \textcircled{3} + \textcircled{4} - \lambda\,\textcircled{2}^{a}$
Achille and Soatto [2018]	$\textcircled{1} + \textcircled{3} + \beta\,\textcircled{2} + \gamma\,\textcircled{A}^{*}$
Kim and Mnih [2018],Chen et al. [2018]	$\textcircled{1} + \textcircled{2} + \textcircled{3} + \textcircled{B} + \beta\,\textcircled{A}^{*}$
HFVAE (this paper)	$\textcircled{1} + \textcircled{3} + \textcircled{ii} + \alpha\,\textcircled{2} + \beta\,\textcircled{A} + \gamma\,\textcircled{i}$

Results: CelebA

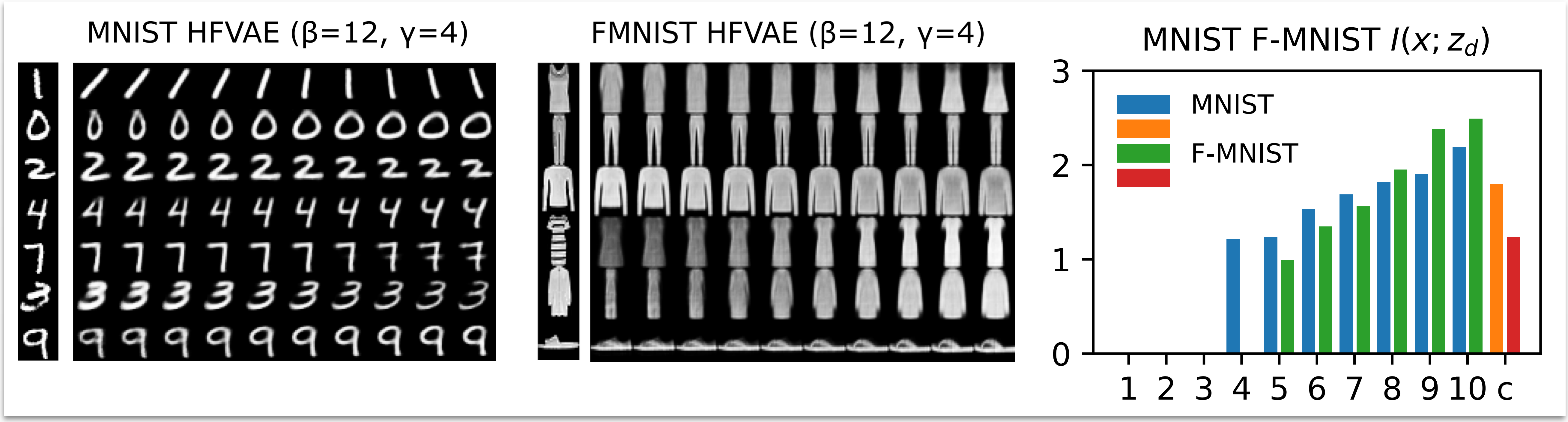
HFVAE and β-VAE are qualitatively similar

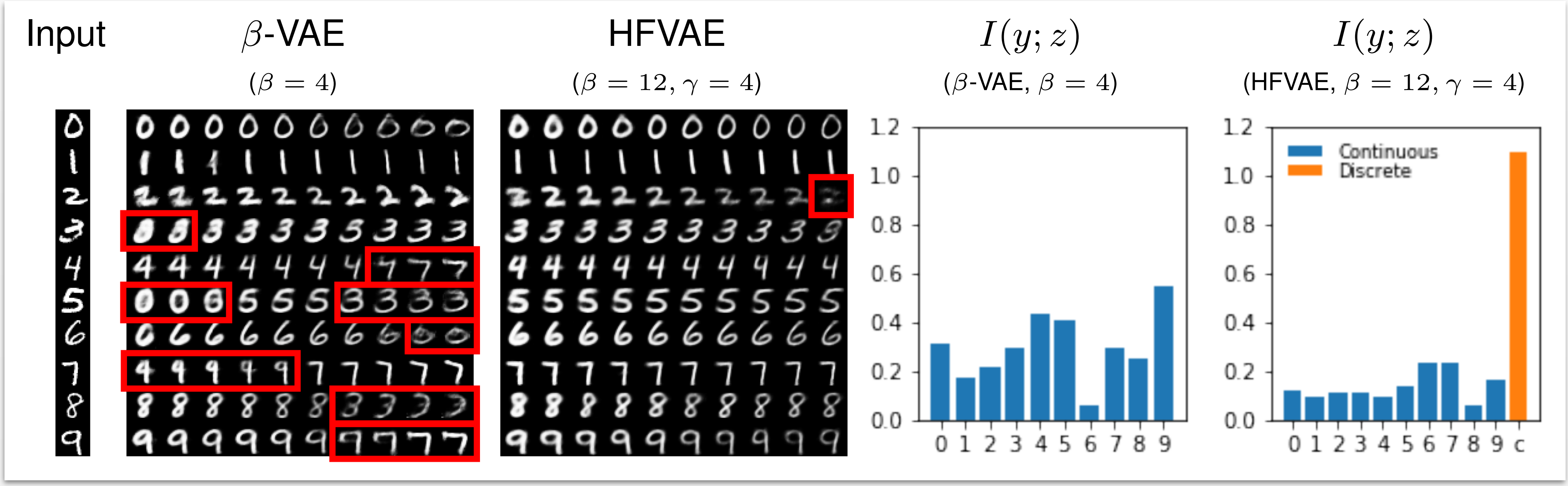
Both objectives learn (some) interpretable features

Results: MNIST and FMNIST

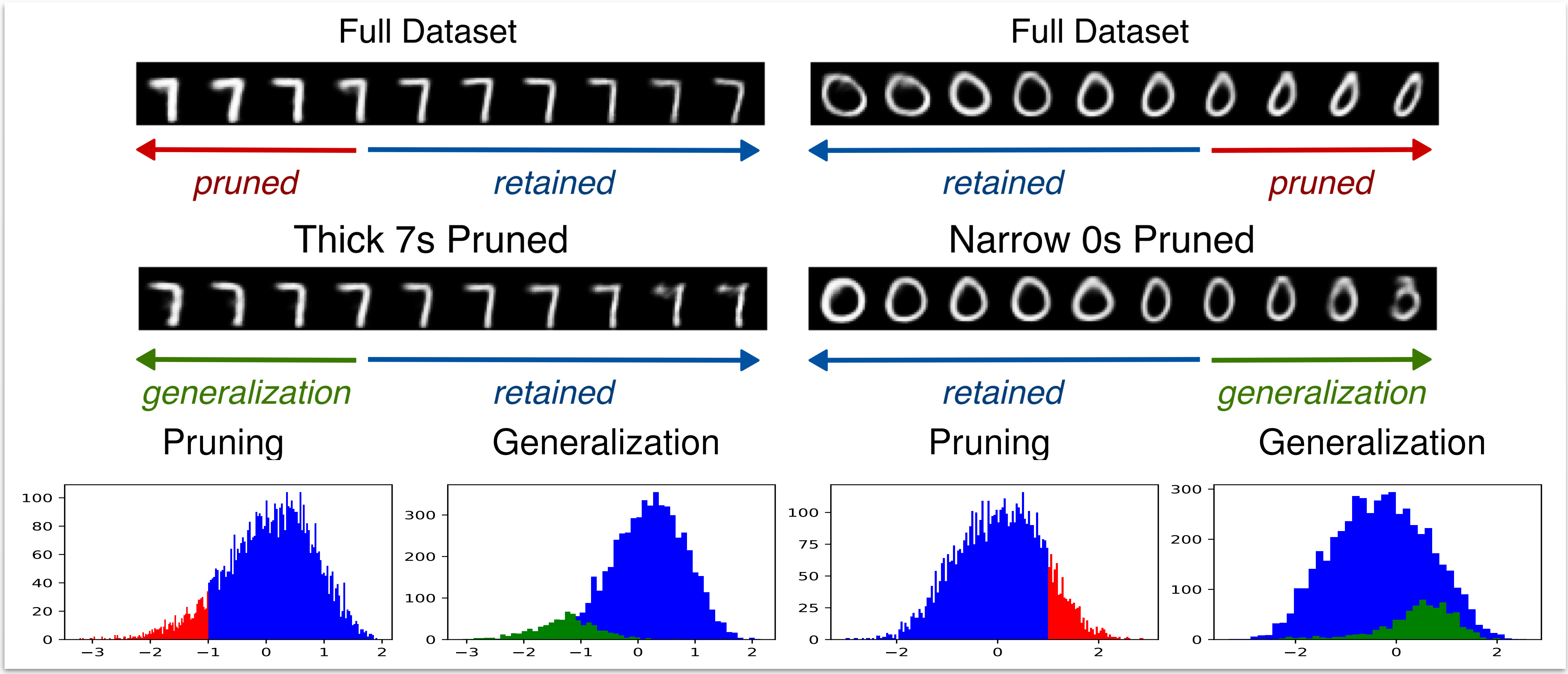


Results: MNIST and FMNIST



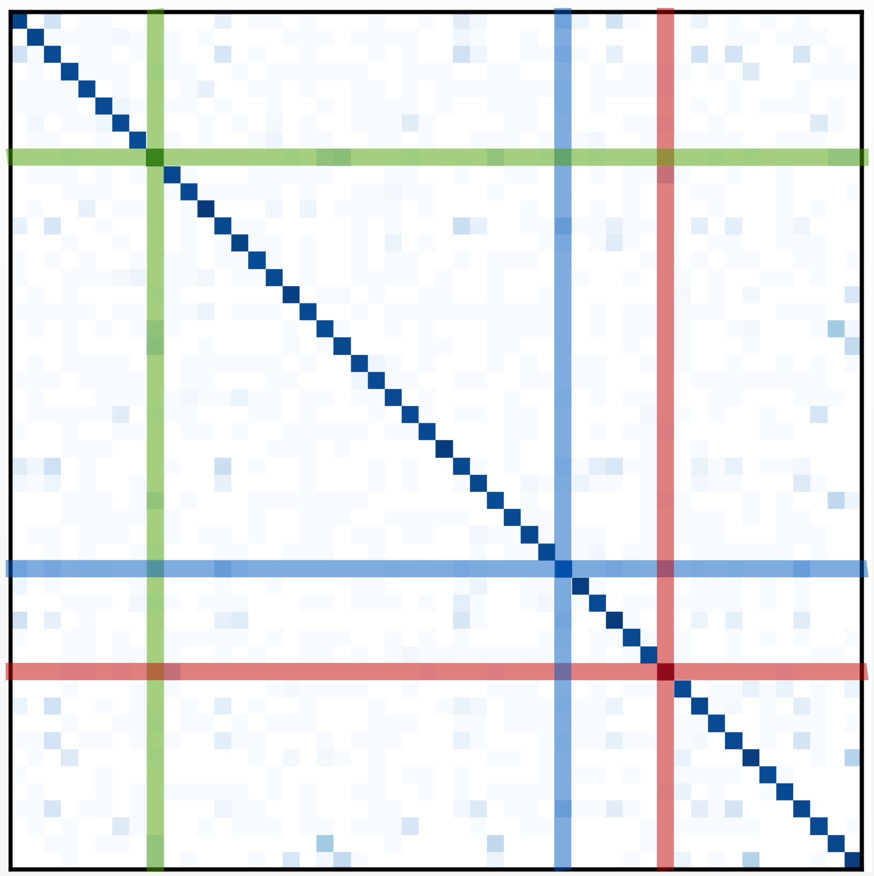


Results: Generalization



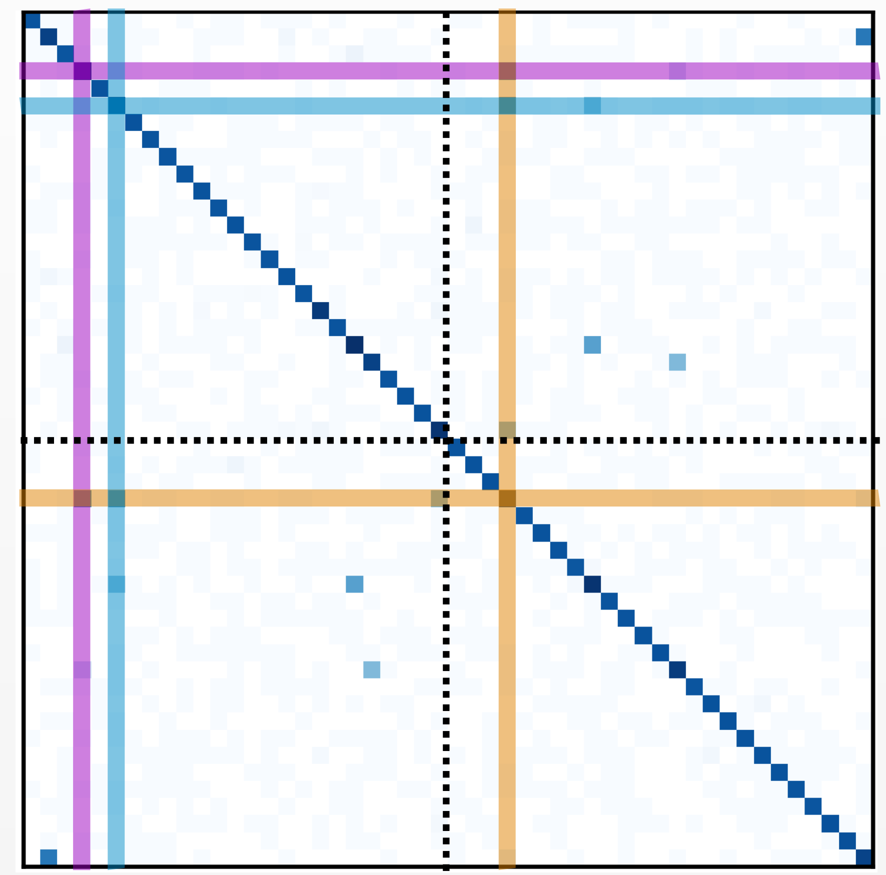
Bonus: Learning Correlations Between Topics

VAE (50 topics)



Top Words	NPMI
jesus, scripture, god, christ, bible, sin, christian, doctrine, faith, church	0.41
team, game, player, score, league, play, leafs, hockey, season, nhl	0.37
scsi, ide, controller, drive, bus, subject, lines, organization, card, problem	0.14

HFVAE (25 + 25 topics)



Top 3 Most Correlated Topics	NPMI
armenian, turk, civilian, turkish, soldier, kill, extermination, armenia, israel, jew	0.40
jesus, belief, god, christian, christ, faith, scripture, moral, truth, sin	0.35
gun, crime, people, law, defense, government, criminal, shoot, assault, fire	0.26

Deep Learning + Probabilistic Programming

Deep Learning

Probabilistic Programming

Deep Learning + Probabilistic Programming

Deep Learning

Discriminative (Data → Features)

Probabilistic Programming

Generative (Variables → Data)

Deep Learning + Probabilistic Programming

Deep Learning

Discriminative
(Data → Features)

Large Data,
Low Uncertainty

Probabilistic Programming

Generative
(Variables → Data)

Small Data,
High Uncertainty

Deep Learning + Probabilistic Programming

Deep Learning	Probabilistic Programming
Discriminative (Data → Features)	Generative (Variables → Data)
Large Data, Low Uncertainty	Small Data, High Uncertainty
Stochastic Gradient Descent	Many Inference Methods

Deep Learning + Probabilistic Programming

Deep Learning	Probabilistic Programming
Discriminative (Data → Features)	Generative (Variables → Data)
Large Data, Low Uncertainty	Small Data, High Uncertainty
Stochastic Gradient Descent	Many Inference Methods
Representation Learning	Model-based Reasoning

Deep Learning + Probabilistic Programming

Deep Learning	Integrated Approaches	Probabilistic Programming
Discriminative (Data → Features)		Generative (Variables → Data)
Large Data, Low Uncertainty		Small Data, High Uncertainty
Stochastic Gradient Descent		Many Inference Methods
Representation Learning		Model-based Reasoning

Deep Learning + Probabilistic Programming

Deep Learning	Integrated Approaches	Probabilistic Programming
Discriminative (Data → Features)		Generative (Variables → Data)
Large Data, Low Uncertainty	Decision Making	Small Data, High Uncertainty
Stochastic Gradient Descent		Many Inference Methods
Representation Learning		Model-based Reasoning

Deep Learning + Probabilistic Programming

Deep Learning	Integrated Approaches	Probabilistic Programming
Discriminative (Data → Features)	Autoencoders	Generative (Variables → Data)
Large Data, Low Uncertainty	Decision Making	Small Data, High Uncertainty
Stochastic Gradient Descent		Many Inference Methods
Representation Learning		Model-based Reasoning

Deep Learning + Probabilistic Programming

Deep Learning	Integrated Approaches	Probabilistic Programming
Discriminative (Data → Features)	Autoencoders	Generative (Variables → Data)
Large Data, Low Uncertainty	Decision Making	Small Data, High Uncertainty
Stochastic Gradient Descent	Variational Inference	Many Inference Methods
Representation Learning		Model-based Reasoning

Thank You!

Northeastern University — Oxford — ATI


Babak Esmaeli


Hao Wu


Sarthak Jain


Alican Bozkurt


Siddharth N.


Brooks Paige

Papers

Structured Disentangled Representations
B. Esmaeili, H. Wu, S. Jain, A. Bozkurt, N. Siddharth, B. Paige, D. H. Brooks, J. Dy, J.-W. van de Meent
ArXiv [https://arxiv.org/abs/1804.02086]

Learning disentangled representations with semi-supervised deep generative models
N. Siddharth, B. Paige, J.-W. van de Meent, A. Desmaison, F. Wood, N.D. Goodman, P. Kohli, P.H.S. Torr
NIPS 2017 [https://bit.ly/probtorch-nips-2017]

Code

https://github.com/probtorch/probtorch