

VAE with Discrete Variables For Semi-Supervised Learning

Author of the task: Cyril Struminsky
Presentation: Nadezhda Chirkova

VAE: reminder

Model for one image:

$$p(x, z | \theta) = p(z)p(x | z, \theta)$$

$$p(z) = \mathcal{N}(z | 0, I) \quad \text{--- Prior over latent code}$$

$$p(x | z, \theta) = \prod_{i=1}^D p_i(z, \theta)^{x_i} (1 - p_i(z, \theta))^{1-x_i}. \quad \text{--- Likelihood of data
(here — Bernoulli distribution)}$$

neural network

x - data (binary image)

z - latent code (continuous d-dim vector)



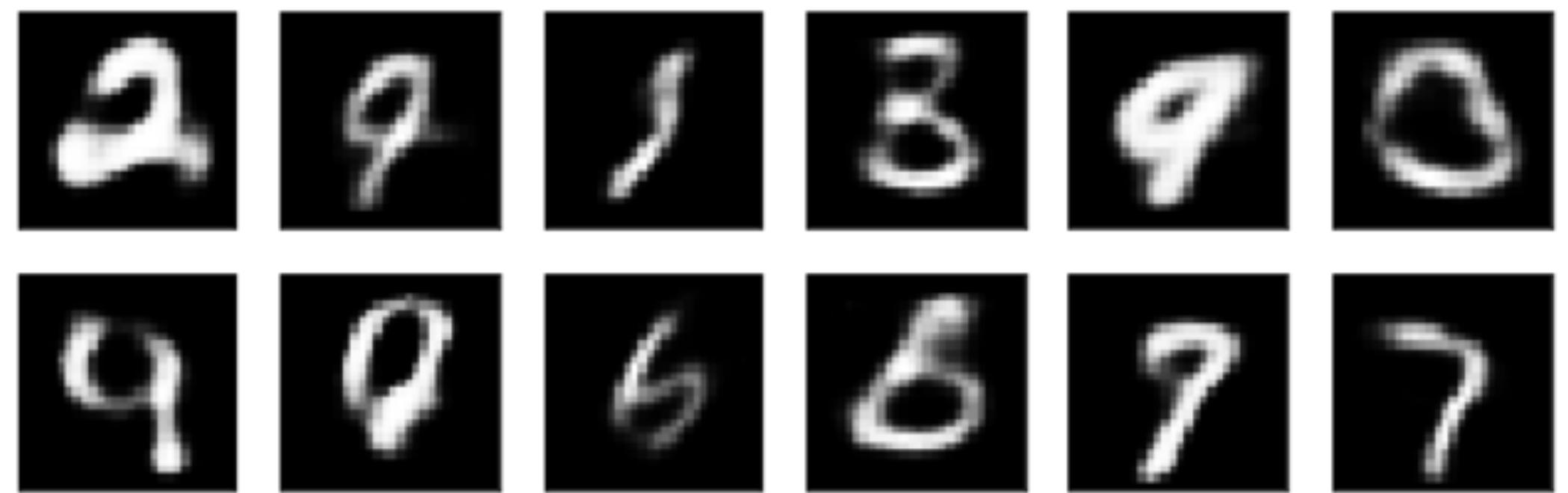
Training data

VAE as a generative model (“testing” stage)

Generate a new image:

$$1. z \sim p(z)$$

$$2. x \sim p(x|z, \theta)$$



Generated images

1. $z \rightarrow NN \rightarrow \{\log p_i(z, \theta)\}_{i=1,\dots,D}$
2. Sample $x_i = 0/1, i = 1, \dots, D$

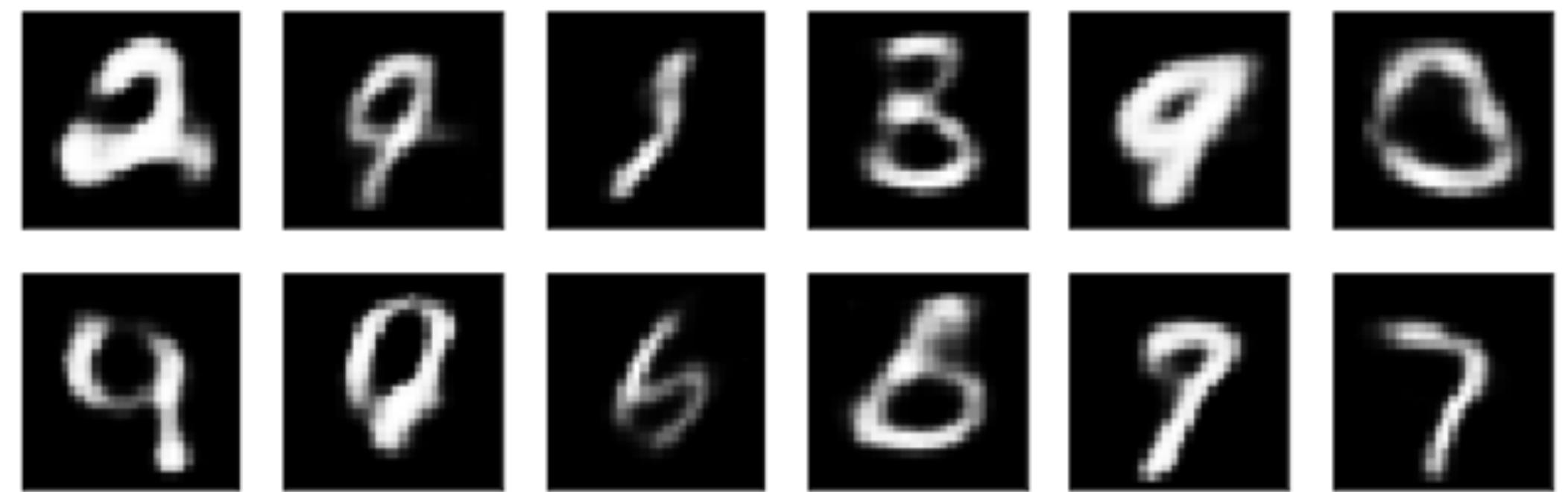
VAE as a generative model (“testing” stage)

Generate a new image:

$$1. z \sim p(z)$$

$$2. x \sim p(x|z, \theta)$$

$\theta?$



Generated images

1. $z \rightarrow NN \rightarrow \{\log p_i(z, \theta)\}_{i=1, \dots, D}$
2. Sample $x_i = 0/1, i = 1, \dots, D$

Training VAE

$$\log p(x) \geq \mathbb{E}_{q(z|x)} \log p(x|z, \theta) - KL(q(z|x)||p(z)) + KL(q(z|x)||p(z|x)) \quad \forall q$$

does not
depend on q

max

\Leftrightarrow

min

$$q(z|x, \phi) \approx p(z|x) \rightarrow$$

$$\mathcal{L}(\theta, \phi) = \sum_{n=1}^N \mathbb{E}_{q(z_n|x_n, \phi)} \log p(x_n|z_n, \theta) - KL(q(z_n|x_n, \phi)||p(z_n)) \rightarrow \max_{\theta, \phi}$$

Training VAE

$$\log p(x) \geq \mathbb{E}_{q(z|x)} \log p(x|z, \theta) - KL(q(z|x)||p(z)) + KL(q(z|x)||p(z|x)) \quad \forall q$$

does not
depend on q

max

\Leftrightarrow

min

$$q(z|x, \phi) \approx p(z|x) \rightarrow$$

$$\mathcal{L}(\theta, \phi) = \sum_{n=1}^N \mathbb{E}_{q(z_n|x_n, \phi)} \log p(x_n|z_n, \theta) - KL(q(z_n|x_n, \phi)||p(z_n)) \rightarrow \max_{\theta, \phi}$$

$$q(z|x, \phi)?$$

Training VAE

$$q(z | x, \phi) = \mathcal{N}(z | \mu(x, \phi), \text{diag}(\sigma^2(x, \phi)))$$

A diagram illustrating a neural network structure. At the bottom center, the text "neural network" is written. Above it, two thin black arrows point towards each other from opposite sides, forming a V-shape that intersects at a single point directly above the text.

Training VAE

$$q(z | x, \phi) = \mathcal{N}(z | \mu(x, \phi), \text{diag}(\sigma^2(x, \phi)))$$

$$\mathcal{L}(\theta, \phi) = \sum_{n=1}^N \mathbb{E}_{q(z_n | x_n, \phi)} \log p(x_n | z_n, \theta) - KL(q(z_n | x_n, \phi) || p(z_n)) \rightarrow \max_{\theta, \phi}$$

$$\mathcal{L}(\theta, \phi) = \sum_n \mathbb{E}_{q(z_n | x_n, \phi)} [\log p(x_n | z_n, \theta) + \log p(z_n) - \log q(z_n | x_n, \phi)] \rightarrow \max_{\theta, \phi}$$

?

Bernoulli pdf

Gaussian pdf

Gaussian pdf

Training VAE

$$q(z | x, \phi) = \mathcal{N}(z | \mu(x, \phi), \text{diag}(\sigma^2(x, \phi)))$$

$$\mathcal{L}(\theta, \phi) = \sum_{n=1}^N \mathbb{E}_{q(z_n | x_n, \phi)} \log p(x_n | z_n, \theta) - KL(q(z_n | x_n, \phi) || p(z_n)) \rightarrow \max_{\theta, \phi}$$

$$\mathcal{L}(\theta, \phi) = \sum_n \mathbb{E}_{q(z_n | x_n, \phi)} [\log p(x_n | z_n, \theta) + \log p(z_n) - \log q(z_n | x_n, \phi)] \rightarrow \max_{\theta, \phi}$$

? Bernoulli pdf Gaussian pdf Gaussian pdf

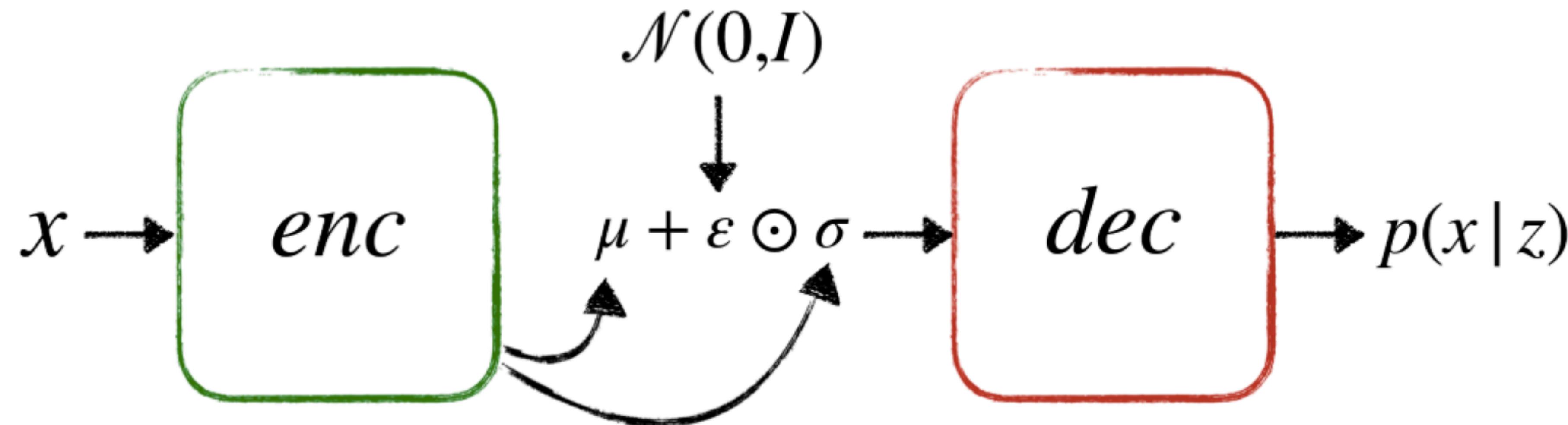
Sample mini-batch + Monte-Carlo sample + Reparametrization trick:

$$\mathbb{E}[\dots] \approx \log p(x | z_0, \theta) + \log p(z_0) - \log q(z_0 | x, \phi)$$

$$z_0 = \mu(x, \phi) + \sigma(x, \phi) \odot \varepsilon_0$$

$$\varepsilon_0 \sim \mathcal{N}(0, I)$$

VAE: computational graph



VAE: code: defining encoder and decoder

```
d, nh, D = 32, 100, 28 * 28

enc = nn.Sequential(
    nn.Linear(D, nh),
    nn.ReLU(),
    nn.Linear(nh, nh),
    nn.ReLU(),
    nn.Linear(nh, 2 * d))

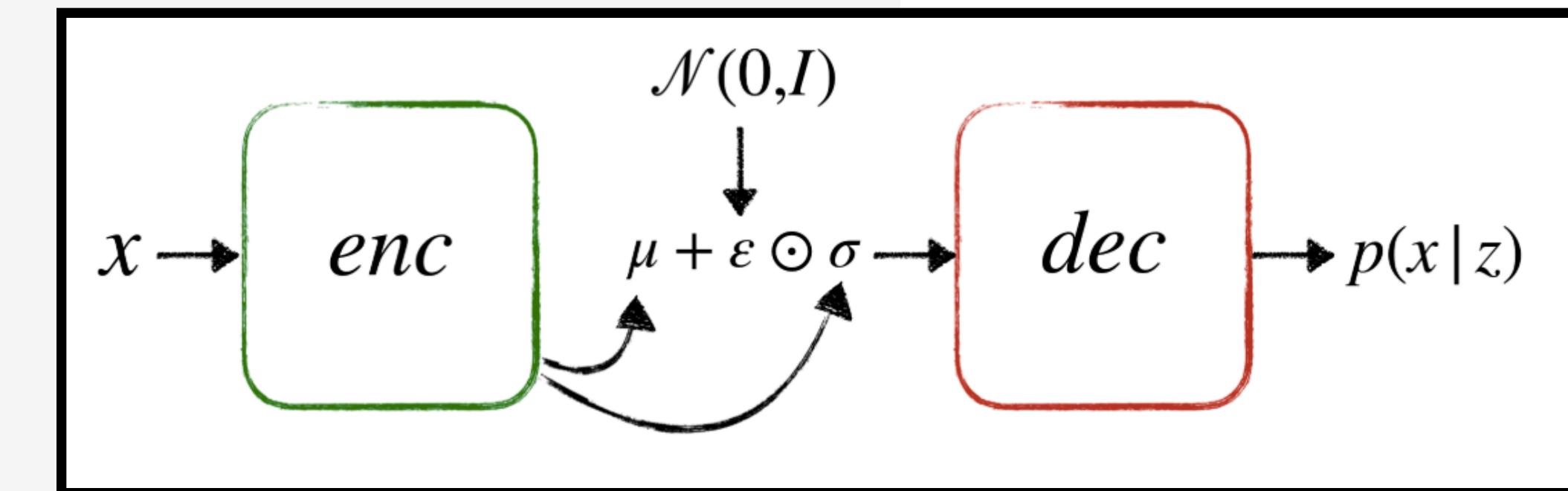
dec = nn.Sequential(
    nn.Linear(d, nh),
    nn.ReLU(),
    nn.Linear(nh, nh),
    nn.ReLU(),
    nn.Linear(nh, D)).to(device)

enc = enc.to(device)
dec = dec.to(device)
```

VAE: code: computing loss

```
def loss_vae(x, encoder, decoder):
    """
    returns
    1. the average value of negative
       ELBO across the minibatch x
    2. and the output of the decoder
    """
    batch_size = x.size(0)
    encoder_output = encoder(x)
    pz = Independent(Normal(loc=torch.zeros(batch_size, d).to(device),
                            scale=torch.ones(batch_size, d).to(device)),
                      reinterpreted_batch_ndims=1)
    qz_x = Independent(Normal(loc=encoder_output[:, :d],
                               scale=torch.exp(encoder_output[:, d:])),
                        reinterpreted_batch_ndims=1)

    z = qz_x.rsample() # reparametrization inside!
    decoder_output = decoder(z)
    px_z = Independent(Bernoulli(logits=decoder_output),
                        reinterpreted_batch_ndims=1)
    loss = -(px_z.log_prob(x) + pz.log_prob(z) - qz_x.log_prob(z)).mean()
    return loss, decoder_output
```

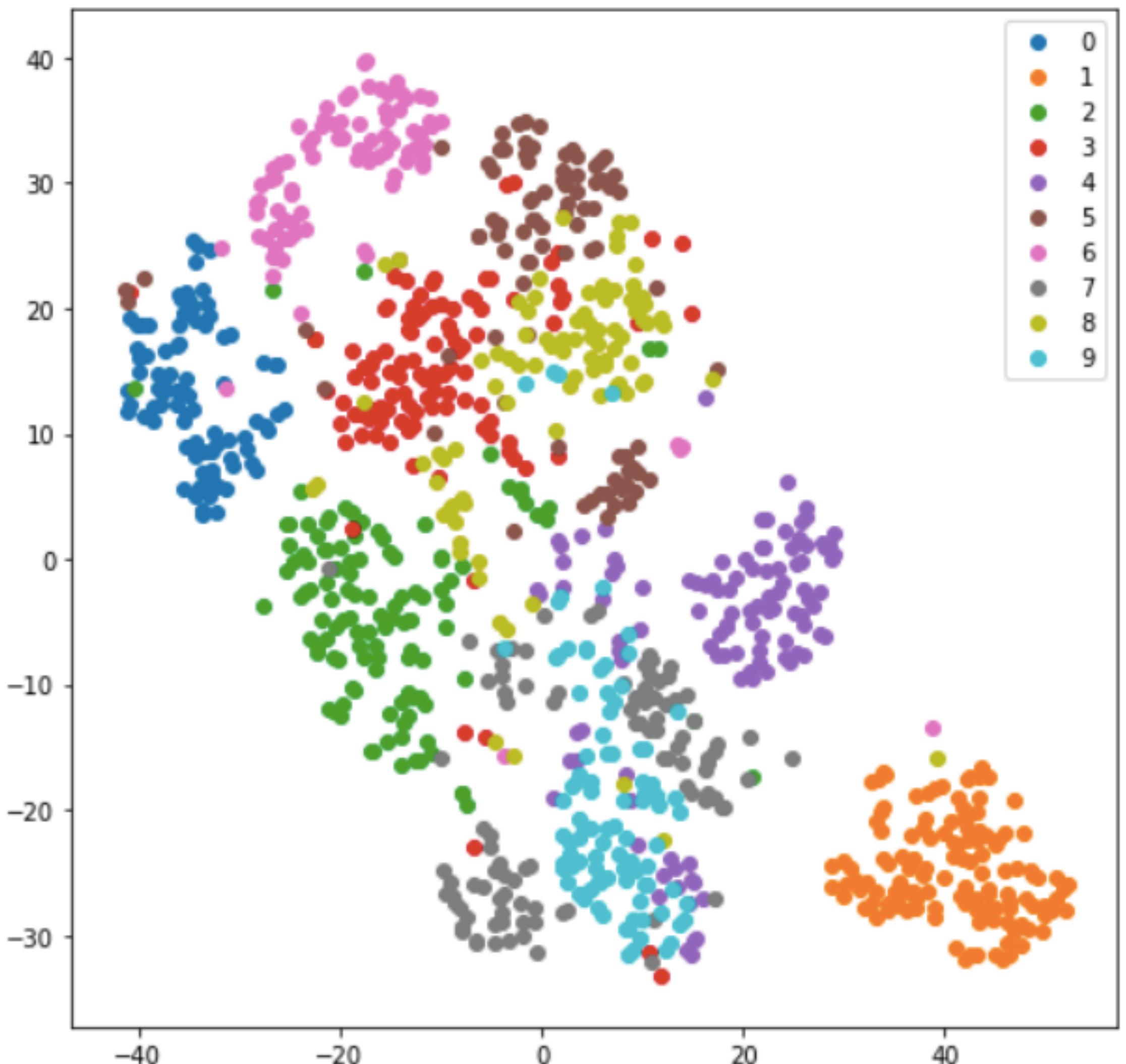


torch.distributions
used to:
1) compute pdf
2) sample

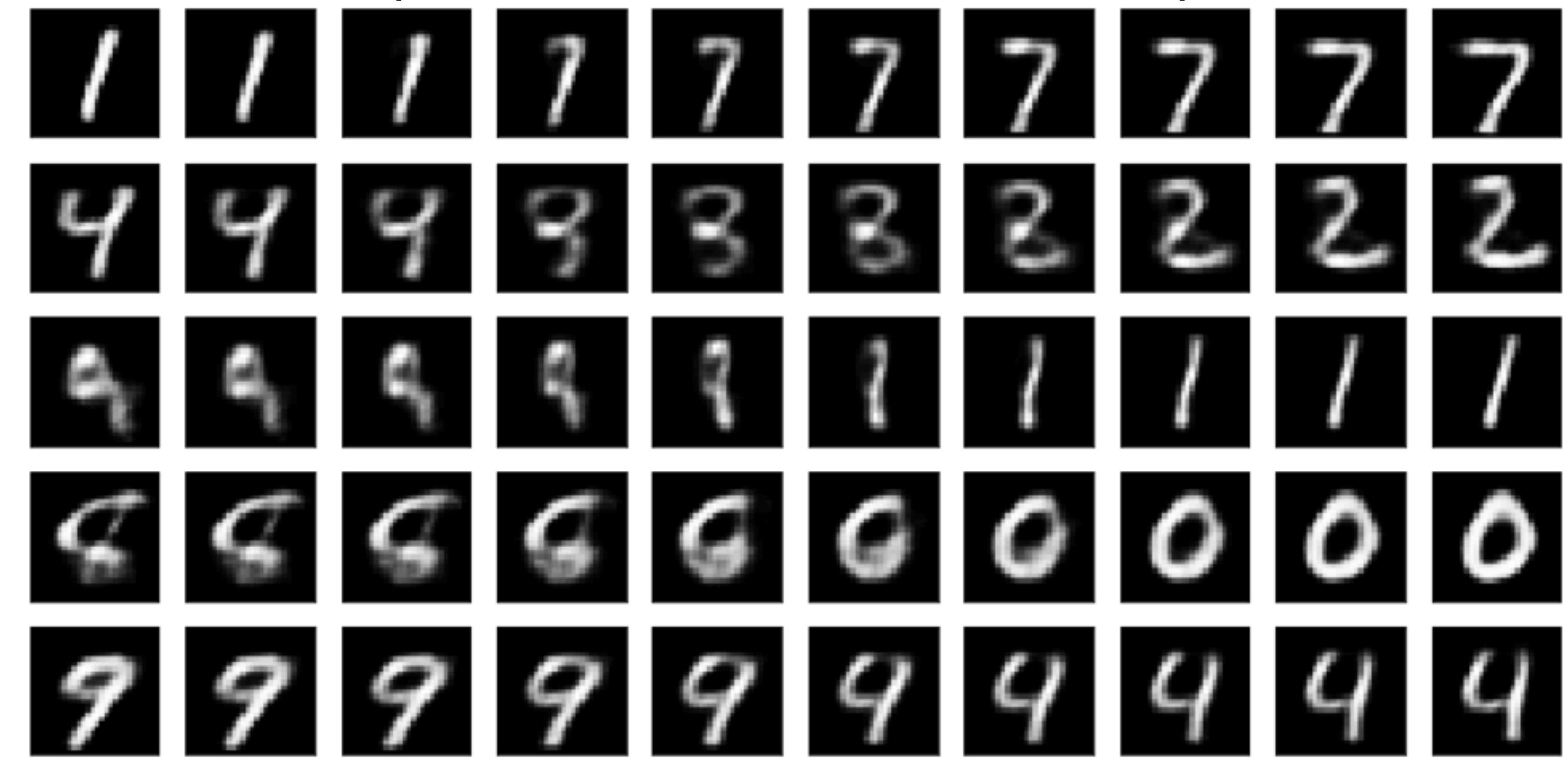
```
from torch.distributions
import Normal,
Bernoulli,
Independent
```

VAE: visualization

tSNE of the latent code mean values



interpolation in the latent space



VAE for semi-supervised learning

Semi-supervised Learning with Deep Generative Models

Diederik P. Kingma*, **Danilo J. Rezende[†]**, **Shakir Mohamed[†]**, **Max Welling***

*Machine Learning Group, Univ. of Amsterdam, {D.P.Kingma, M.Welling}@uva.nl

[†]Google Deepmind, {danilor, shakir}@google.com

CATEGORICAL REPARAMETERIZATION WITH GUMBEL-SOFTMAX

Eric Jang
Google Brain
ejang@google.com

Shixiang Gu*
University of Cambridge
MPI Tübingen
sg717@cam.ac.uk

Ben Poole*
Stanford University
poole@cs.stanford.edu

VAE for semi-supervised learning

Semi-supervised learning: classification when only a small subset of the observations have corresponding class labels.

VAE for semi-supervised learning

Model for one image:

$$p(x, y, z) = p(x | y, z)p(z)p(y)$$

$$p(y) = \text{Cat}(y | \pi_0), \pi_0 = (1/10, \dots, 1/10)$$

$$p(z) = \mathcal{N}(z | 0, I)$$

$$p(x | y, z) = \prod_{i=1}^D p_i(y, z)^{x_i} (1 - p_i(y, z))^{1-x_i}$$

←
neural network

Priors over class label
and latent code

Likelihood of data
(Bernoulli distribution)

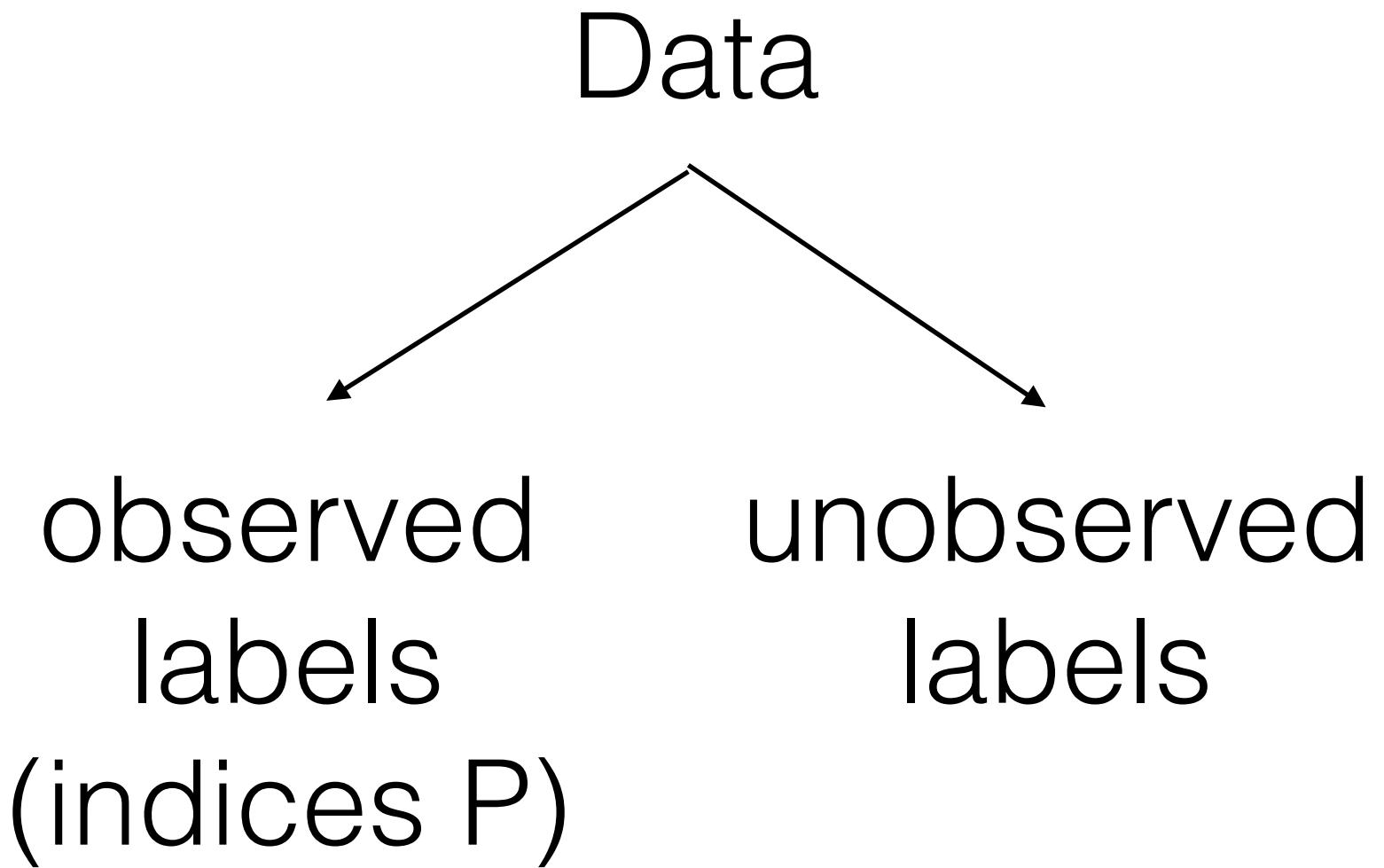
* We now drop parameters in conditioning since formulas are already very impressive

SS-VAE as a generative model

Generate a new image:

1. $z \sim p(z)$
2. $y \sim p(y)$
3. $x \sim p(x|y, z)$

Training SS-VAE



$$LogLikelihood(X, y) = \sum_{i \notin P} \log p(x_i) + \sum_{i \in P} \log p(x_i, y_i)$$

Training SS-VAE

$$\text{LogLikelihood}(X, y) = \sum_{i \notin P} \log p(x_i) + \sum_{i \in P} \log p(x_i, y_i)$$

For observed labels (similar to VAE):

$$q(z | y, x) = \mathcal{N}(z | \mu_\phi(x, y), \text{diag } \sigma_\phi^2(y, x))$$

$$\begin{aligned} \log p(x, y) &= \log \mathbb{E}_{p(z)} p(x, y | z) \geq \\ &\mathbb{E}_{q(z|y,x)} \log \frac{p(x, y | z)p(z)}{q(z | y, x)} \end{aligned}$$

For unobserved labels:

$$q(y, z | x) = q(y | x)q(z | y, x)$$

$$q(y | x) = \text{Cat}(y | \pi(x))$$

$$q(z | y, x) = \mathcal{N}(z | \mu_\phi(x, y), \text{diag } \sigma_\phi^2(y, x))$$

$$\log p(x) = \log \mathbb{E}_{p(y)} \mathbb{E}_{p(z|y)} \log p(x | z, y) \geq$$

$$\mathbb{E}_{q(y|x)} \mathbb{E}_{q(z|y,x)} \log \frac{p(x, y | z)p(z)}{q(z | y, x)q(y | x)}$$

Training SS-VAE

Final objective:

$$\mathcal{L}(X, y) = \sum_{i \in P} \mathbb{E}_{q(z_i | y_i, x_i)} \log \frac{p(x_i, y_i | z_i)p(z_i)}{q(z_i | y_i, x_i)} + \sum_{i \notin P} \mathbb{E}_{q(y_i | x_i)} \mathbb{E}_{q(z_i | y_i, x_i)} \log \frac{p(x_i, y_i | z_i)p(z_i)}{q(z_i | y_i, x_i)q(y_i | x_i)}$$

The diagram illustrates the components of the SS-VAE loss function. The first term, involving samples from Gaussian distributions using the reparametrization trick, is shown with arrows pointing down to the terms $\mathbb{E}_{q(z_i | y_i, x_i)}$ and $p(z_i)$. The second term, involving samples from categorical distributions, is shown with an arrow pointing down to the term $\mathbb{E}_{q(y_i | x_i)}$. A large vertical arrow points down to the third term, which is marked with a red question mark, indicating that the sampling mechanism for this term is not explicitly specified.

Training SS-VAE

Final objective:

$$\mathcal{L}(X, y) = \sum_{i \in P} \mathbb{E}_{q(z_i | y_i, x_i)} \log \frac{p(x_i, y_i | z_i)p(z_i)}{q(z_i | y_i, x_i)} + \sum_{i \notin P} \mathbb{E}_{q(y_i | x_i)} \mathbb{E}_{q(z_i | y_i, x_i)} \log \frac{p(x_i, y_i | z_i)p(z_i)}{q(z_i | y_i, x_i)q(y_i | x_i)}$$

sample from Gaussian distr: reparam. trick

known pdfs

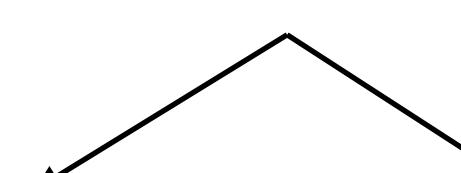
sample from Gaussian distr: reparam. trick

known pdfs

sample from categorical distr.:
Gumbel-Softmax trick

without trick:
|Y| forward passes

with trick:
1 forward pass



Training SS-VAE: important practical aspect

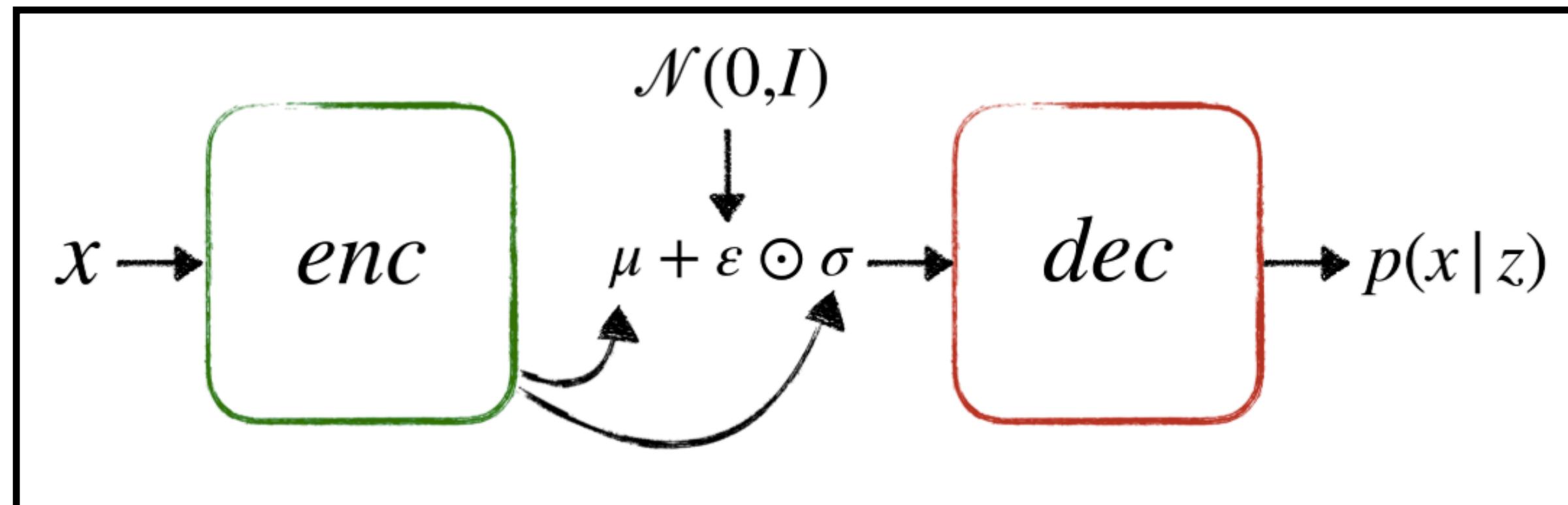
ELBO maximization does not lead to any semantics in latent variables y .

We are going to restrict variational approximations $q(y \mid x)$ to the ones that correctly classify observation x on fully-observed variables (x_i, y_i) . As in the original paper, we will add a cross-entropy regularizer to the objective with weight α :

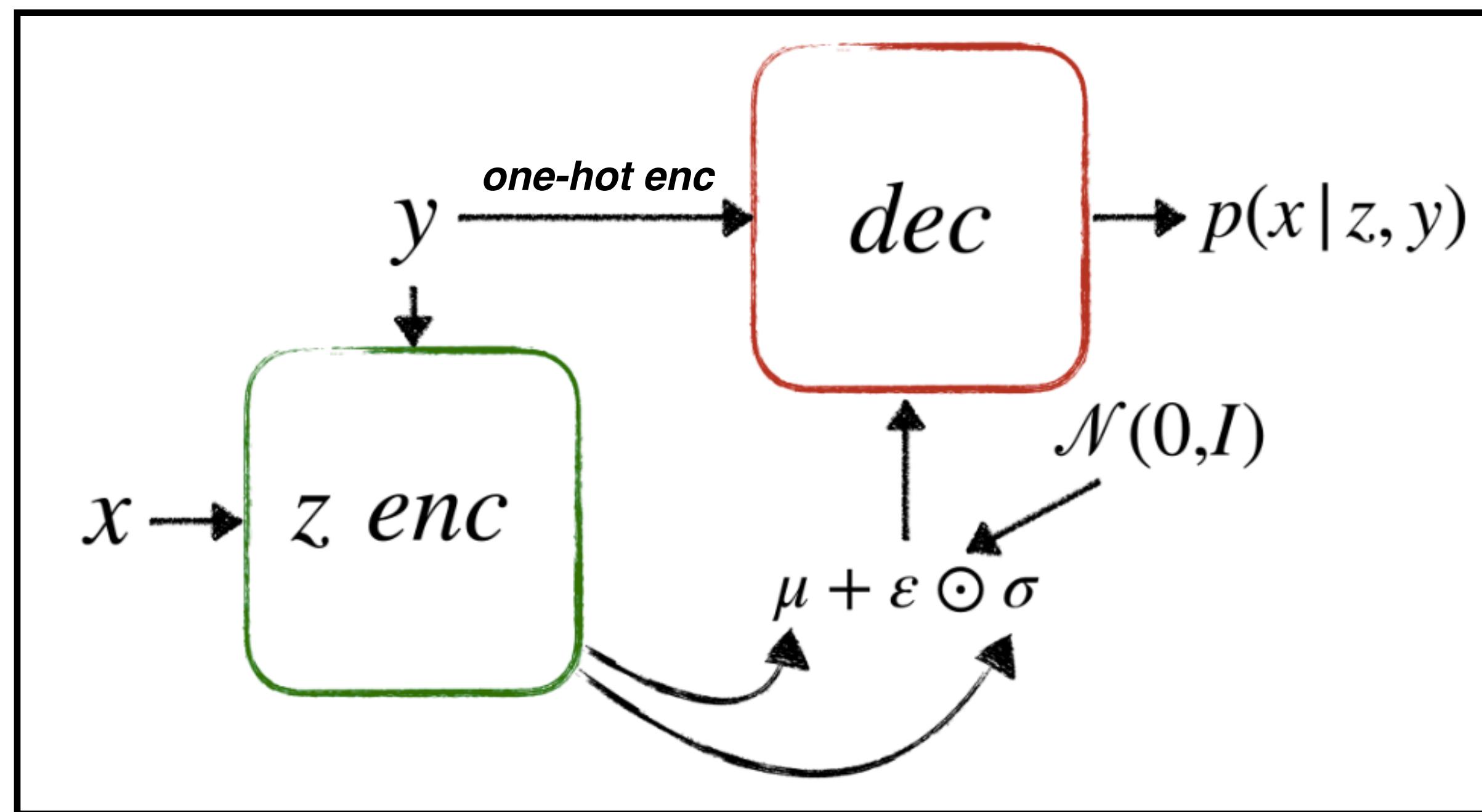
$$\frac{1}{|P|} \sum_{i \in P} y_i^T \log q(y \mid x).$$

SS-VAE: computational graph

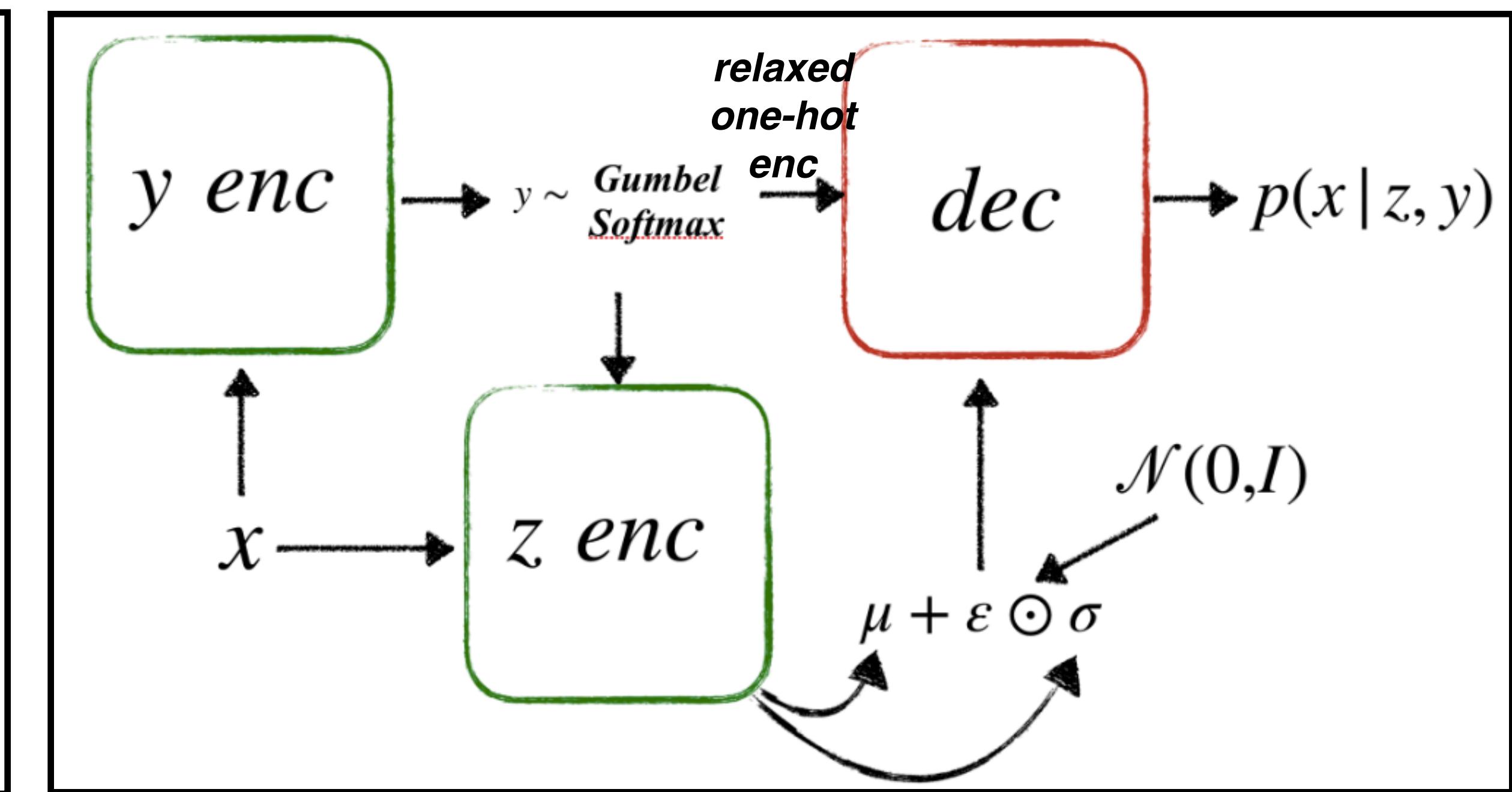
For vanilla VAE:



For observed labels (similar to VAE):



For unobserved labels:



SS-VAE: code: preprocessing data

1. Remove 95% of labels from the training set.
2. The observed labels have a standard one-hot encoding.
3. The unobserved labels are represented by all-zero ten dimensional vectors.

```
data = MNIST(root='.', download=True, train=True)
new_train_labels = torch.zeros(60000, 10)
observed = np.random.choice(60000, 3000)
new_train_labels[observed] = torch.eye(10)[data.targets][observed]
train_data = TensorDataset(data.data.view(-1, 28 * 28).float() / 255,
                           new_train_labels)

data = MNIST(root='.', download=True, train=False)
test_data = TensorDataset(data.data.view(-1, 28 * 28).float() / 255,
                          data.targets)
```

SS-VAE: code: defining encoders and decoder

```
n_classes, d, nh, D = 10, 32, 500, 28 * 28
default_T = torch.tensor(0.6, device=device)

yz_dec = nn.Sequential(
    nn.Linear(n_classes + d, nh),
    nn.ReLU(),
    nn.Linear(nh, D))

y_enc = nn.Sequential(
    nn.Linear(D, nh),
    nn.ReLU(),
    nn.Linear(nh, n_classes))

z_enc = nn.Sequential(
    nn.Linear(n_classes + D, nh),
    nn.ReLU(),
    nn.Linear(nh, 2 * d)
)

yz_dec = yz_dec.to(device)
y_enc = y_enc.to(device)
z_enc = z_enc.to(device)
```

SS-VAE: code: computing loss

```
def loss(x, y, y_encoder, z_encoder, decoder, T=default_T, alpha=32.):
    #TODO
    """
    NOTE:
        hyperparameter alpha was tuned for the implementation that computed
        the mean of elbo terms and sum of cross-entropy terms over the observed
        datapoints in the batch
    """

    return elbo + alpha * cross_entropy
```

In the modified training set the observed labels have a standard one-hot encoding and the unobserved labels are represented by all-zero ten dimensional vectors.

To compute the mask for observed labels you can compute

```
y_is_observed = y.sum(1, keepdim=True)
```

The function has to

1. sample y from $q(y | x)$
2. sample z from $q(z | x, y)$
3. compute the evidence lower bound for observed and unobserved variables
4. compute the cross_entropy regularizer with weight alpha for object with observed labels
5. return the sum of two losses

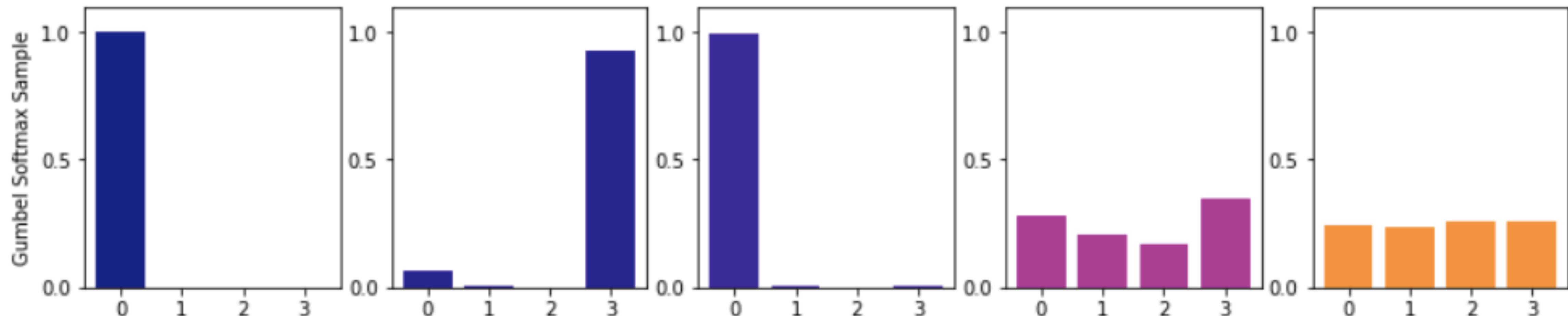
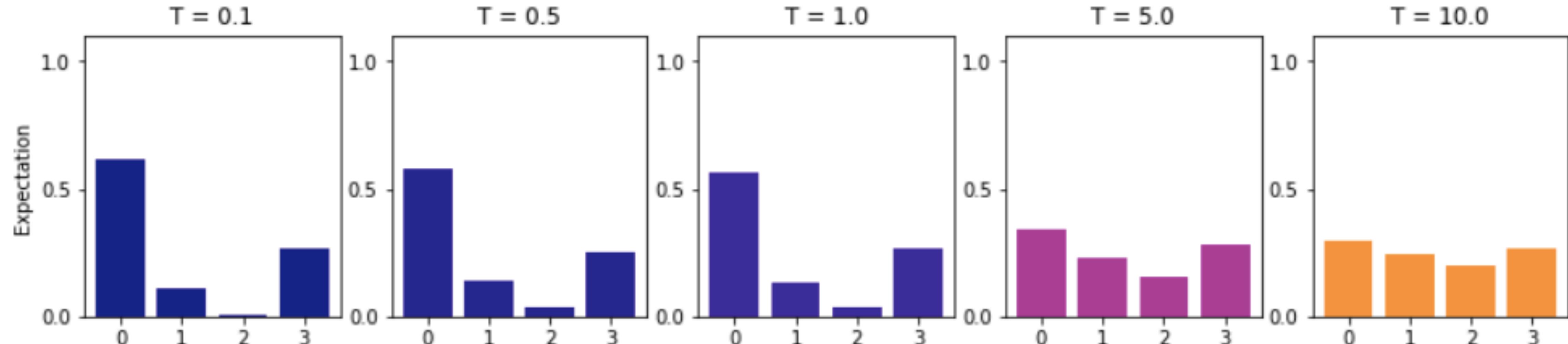
"""

Gumbel softmax trick: example

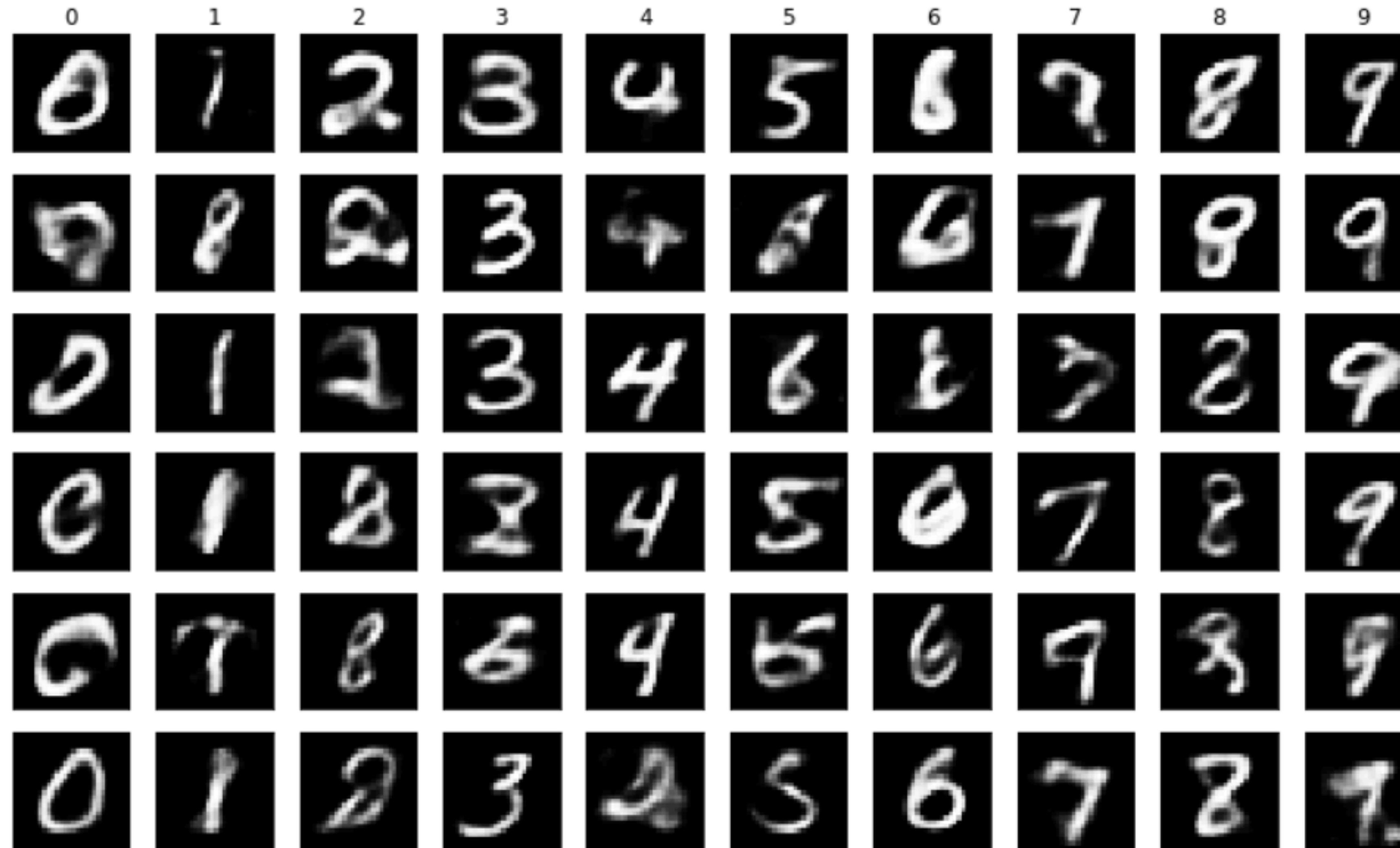
```
from torch.distributions.relaxed_categorical import RelaxedOneHotCategorical
n_classes = 4
logits = torch.randn(1, n_classes)
print('Probs: ', torch.nn.functional.softmax(logits, 1).squeeze().numpy())
temperatures = [0.1, 0.5, 1., 5., 10.]
M = 128

for n, t in enumerate(temperatures):
    dist = RelaxedOneHotCategorical(t, logits=logits)
    mean = torch.zeros_like(logits)
    for _ in range(M):
        mean += dist.sample() / M
    sample = dist.sample()
```

Gumbel softmax trick: example

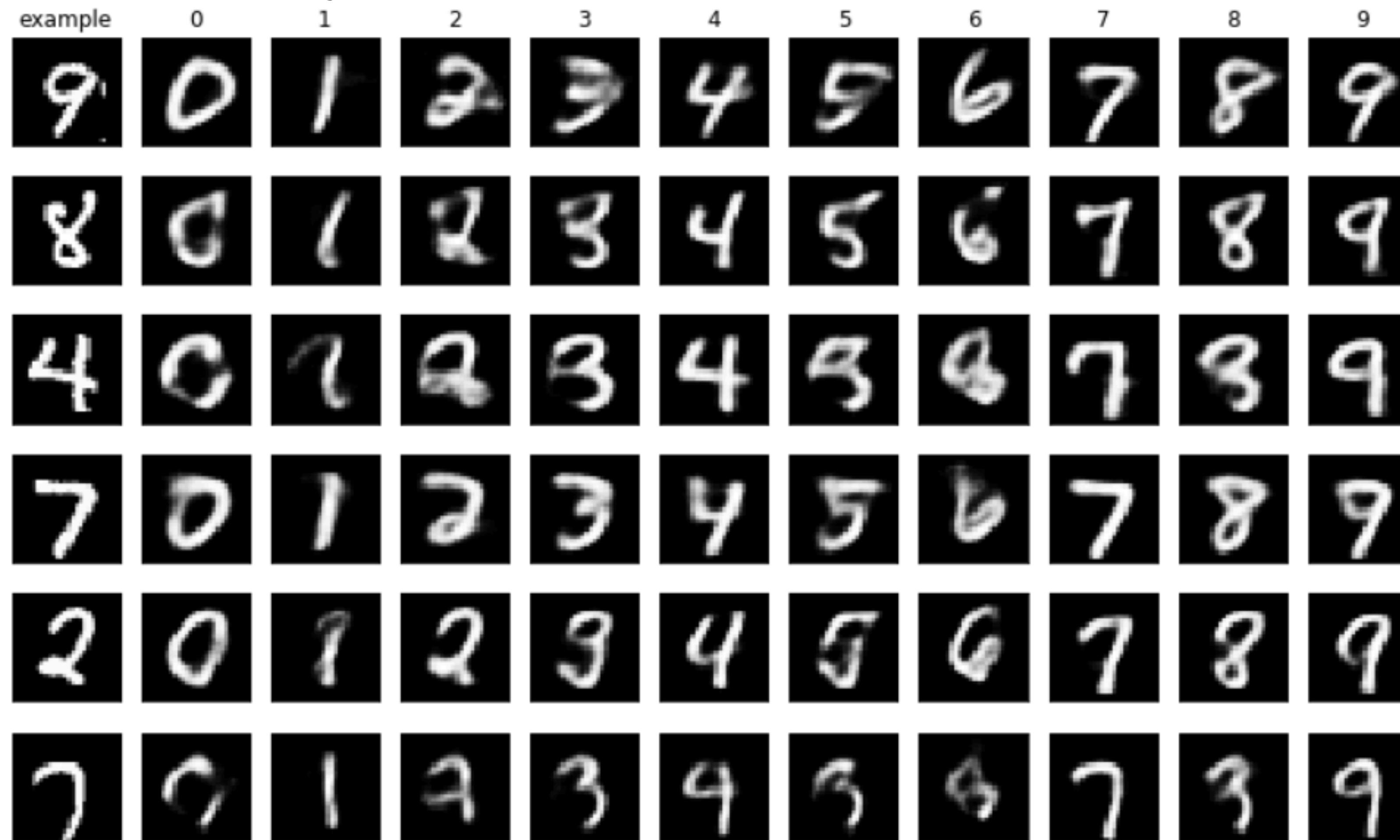


Results: samples

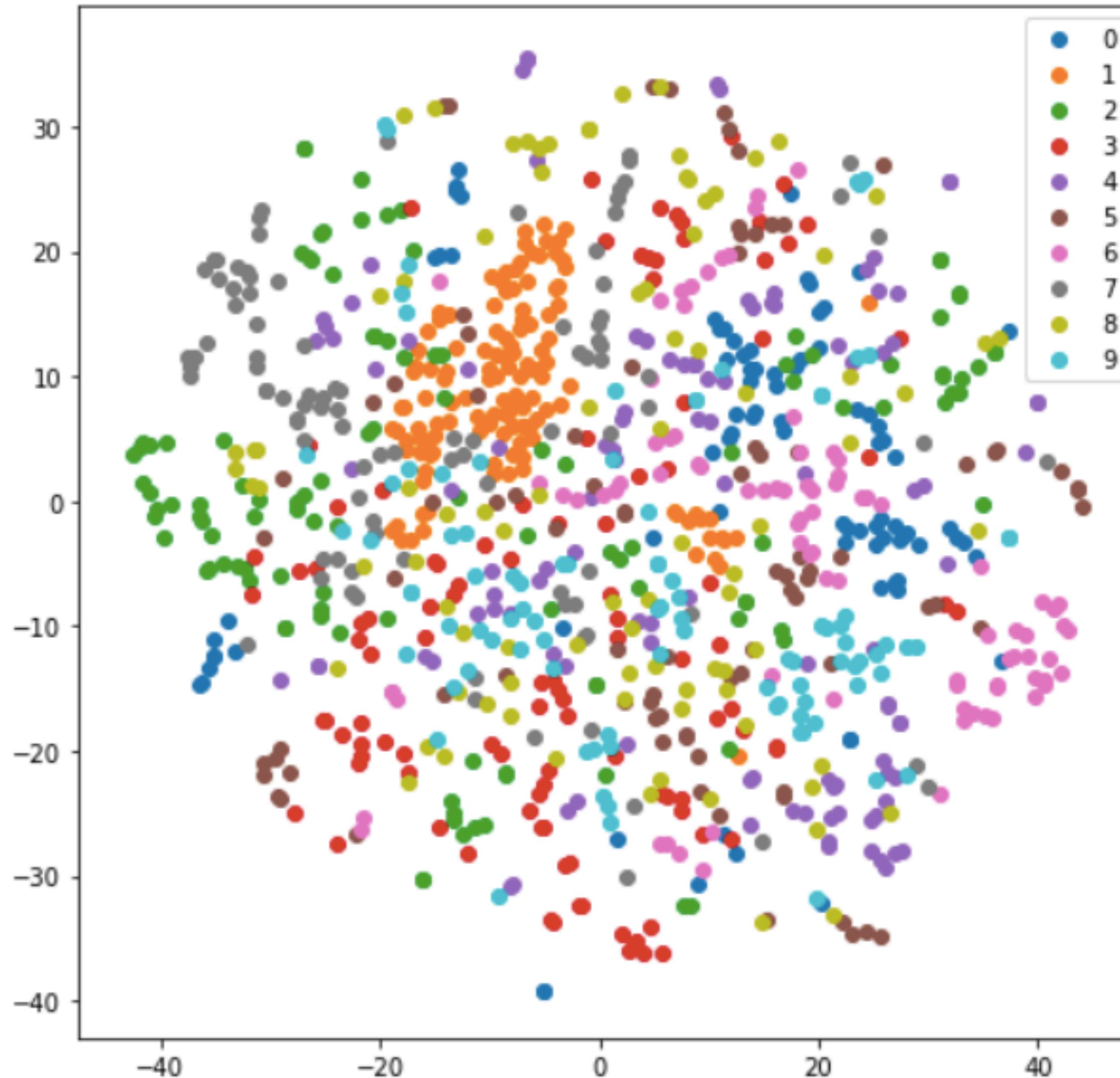


Results: "style transfer"

Infer latent representation z of a given digit x and then generate from $p(x | z, y)$ for different y .



Results: tSNE of the latent space

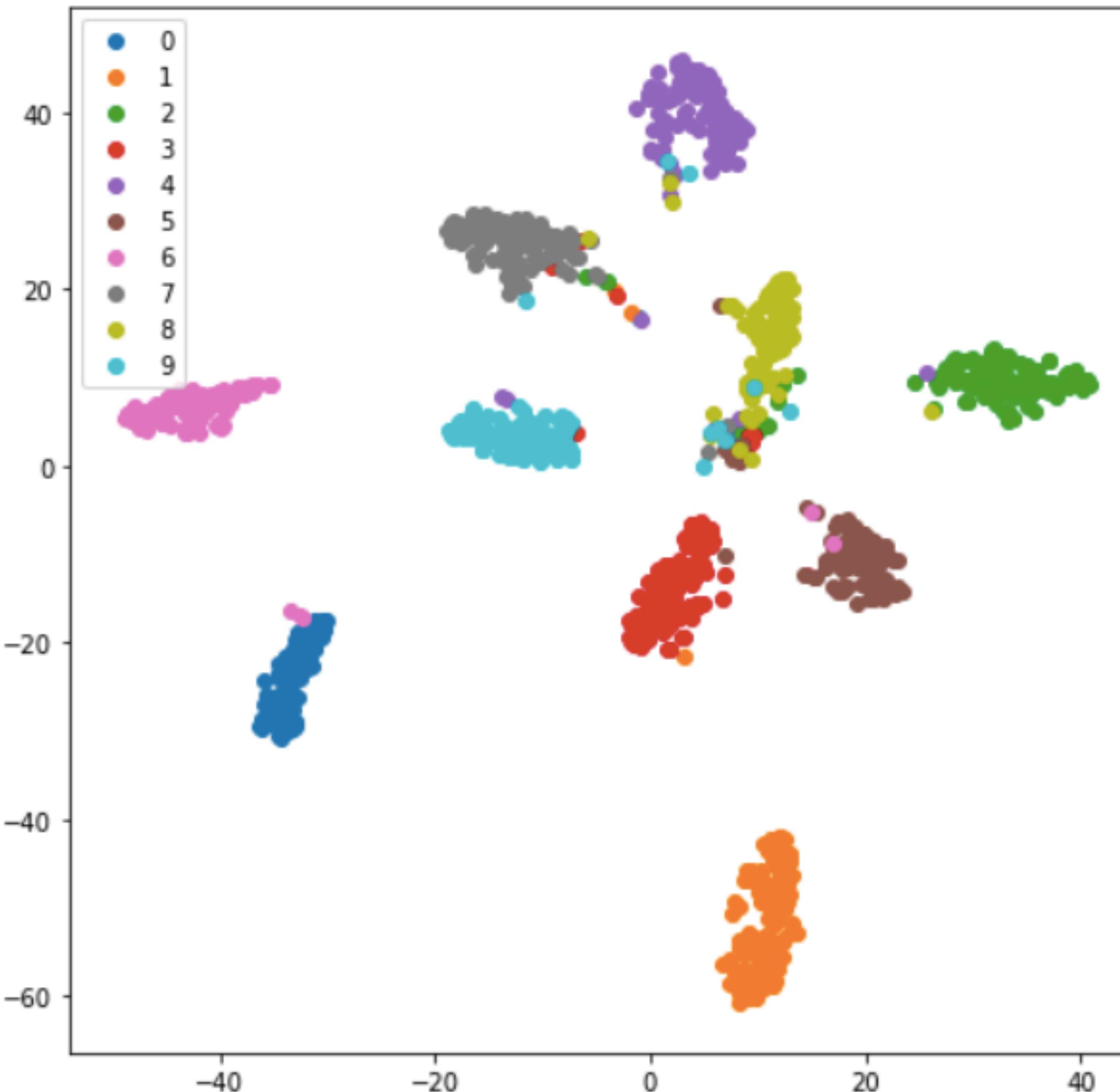


tSNE for $q(z | x, y)$ mean values

Gaussian distr.

Why mixed?

Results: tSNE of the class logits



tSNE for $q(y | x)$ logits