STAT 453: Introduction to Deep Learning and Generative Models

Sebastian Raschka

http://stat.wisc.edu/~sraschka/teaching

Lecture 17

Introduction to Variational Autoencoders

Lecture Overview

1. Variational Autoencoder Overview
2. Sampling from a Variational Autoencoder
3. The Log-Var Trick
4. The Variational Autoencoder Loss Function
5. A Variational Autoencoder for Handwritten Digits in PyTorch
6. A Variational Autoencoder for Face Images in PyTorch
7. VAEs and Latent Space Arithmetic
8. VAE Latent Space Arithmetic in PyTorch -- Making People Smile

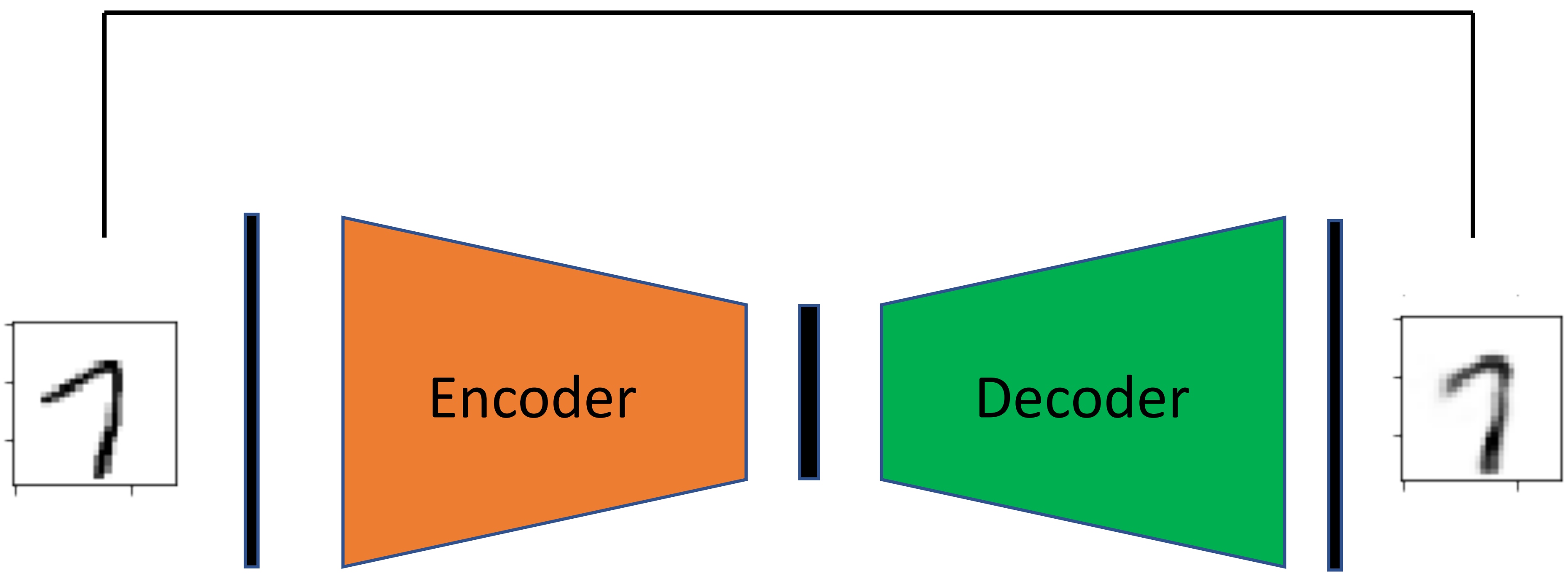
Autoencoders vs. Variational Autoencoders

1. **Variational Autoencoder Overview**
2. Sampling from a Variational Autoencoder
3. The Log-Var Trick
4. The Variational Autoencoder Loss Function
5. A Variational Autoencoder for Handwritten Digits in PyTorch
6. A Variational Autoencoder for Face Images in PyTorch
7. VAEs and Latent Space Arithmetic
8. VAE Latent Space Arithmetic in PyTorch -- Making People Smile

Recap: A Regular Autoencoder

Minimize squared error loss:

$$\mathcal{L} = ||\mathbf{x} - Dec(Enc(\mathbf{x}))||_2^2$$

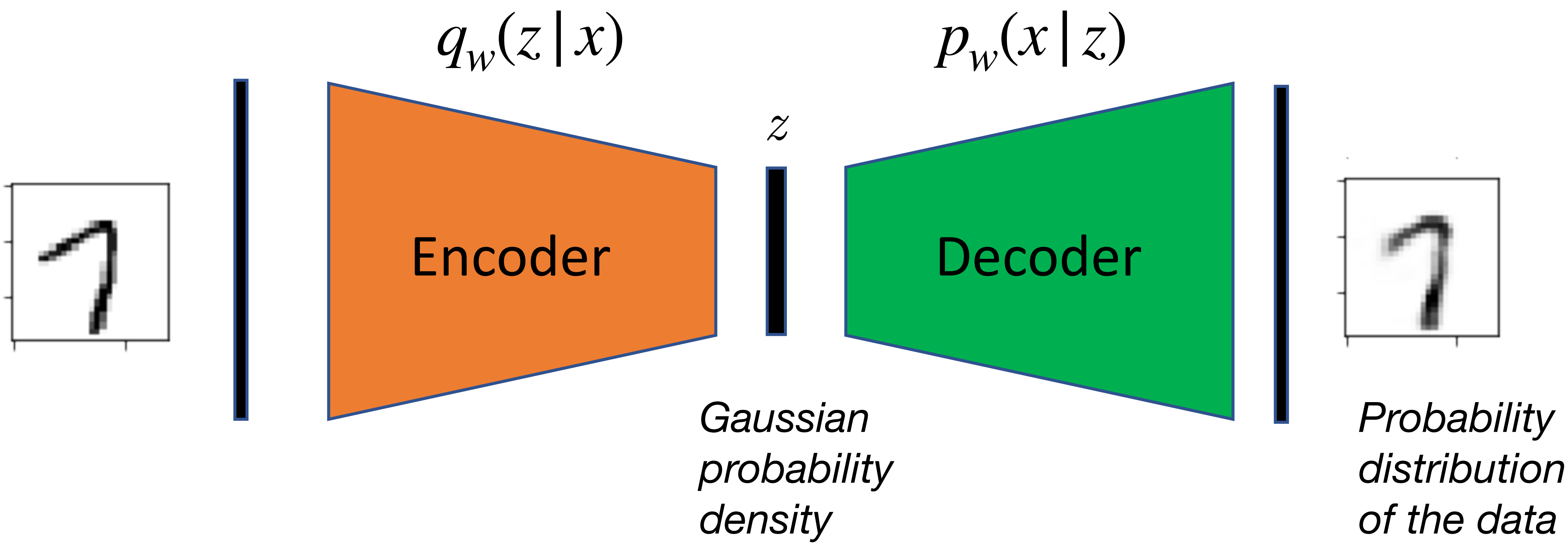


Variational Autoencoder

$$\mathcal{L} = -\mathbb{E}_{z \sim q_w(z \mid x^{[i]})} \left[\log p_w \left(x^{[i]} \mid z\right)\right] + \mathbf{KL}\left(q_w\left(z \mid x^{[i]}\right) \| p(z)\right)$$

Expected neg. log likelihood term; wrt to encoder distribution

Kullback-Leibler divergence term where $p(z) = \mathcal{N}\left(\mu = 0, \sigma^2 = 1\right)$



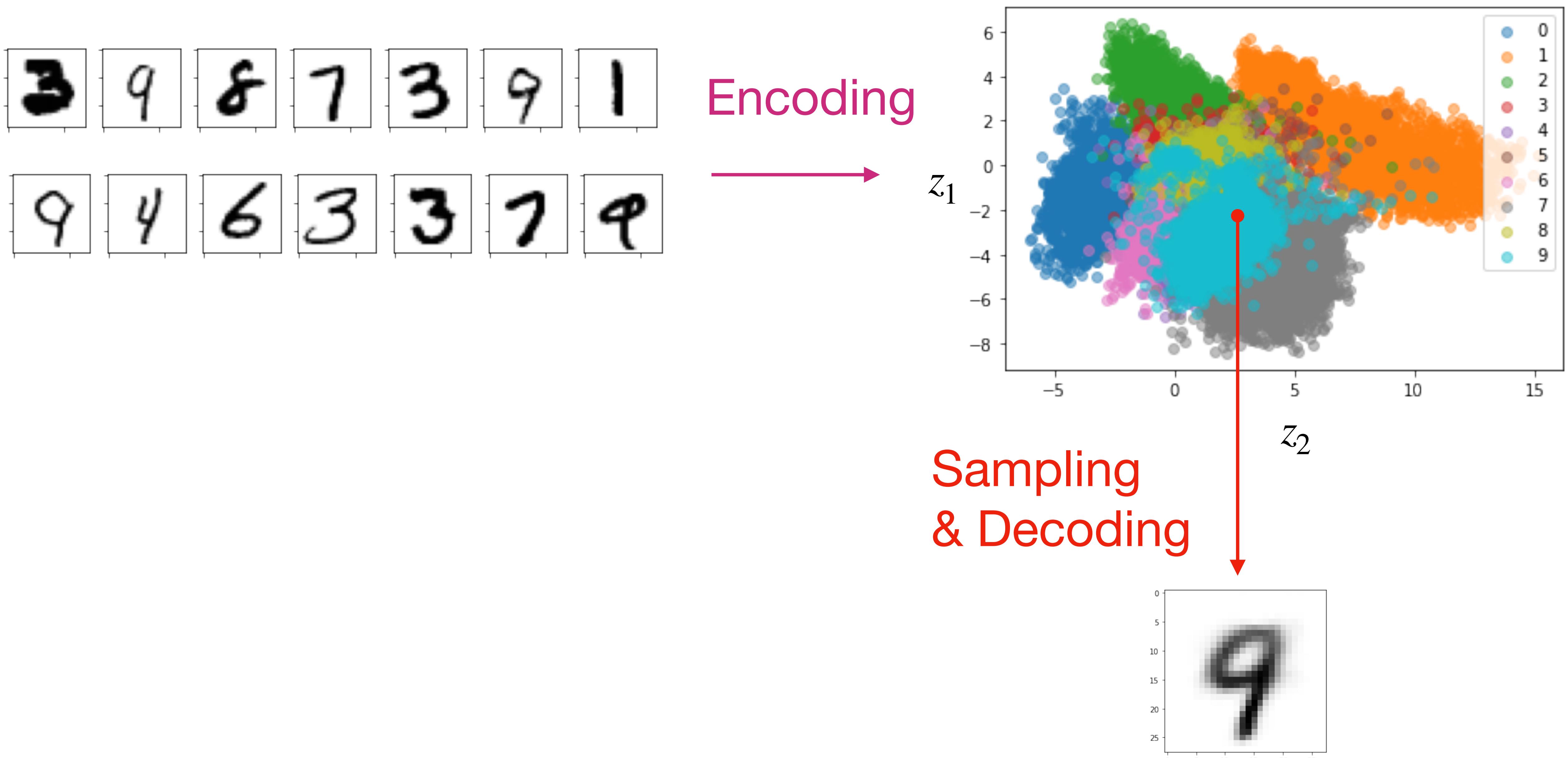
Kingma, D. P., & Welling, M. (2013). Auto-encoding Variational Bayes. *arXiv preprint arXiv:1312.6114*. https://arxiv.org/abs/1312.6114

Generating New Data

1. Variational Autoencoder Overview
2. **Sampling from a Variational Autoencoder**
3. The Log-Var Trick
4. The Variational Autoencoder Loss Function
5. A Variational Autoencoder for Handwritten Digits in PyTorch
6. A Variational Autoencoder for Face Images in PyTorch
7. VAEs and Latent Space Arithmetic
8. VAE Latent Space Arithmetic in PyTorch -- Making People Smile

Using Regular Autoencoders for Sampling

Previous Lecture:

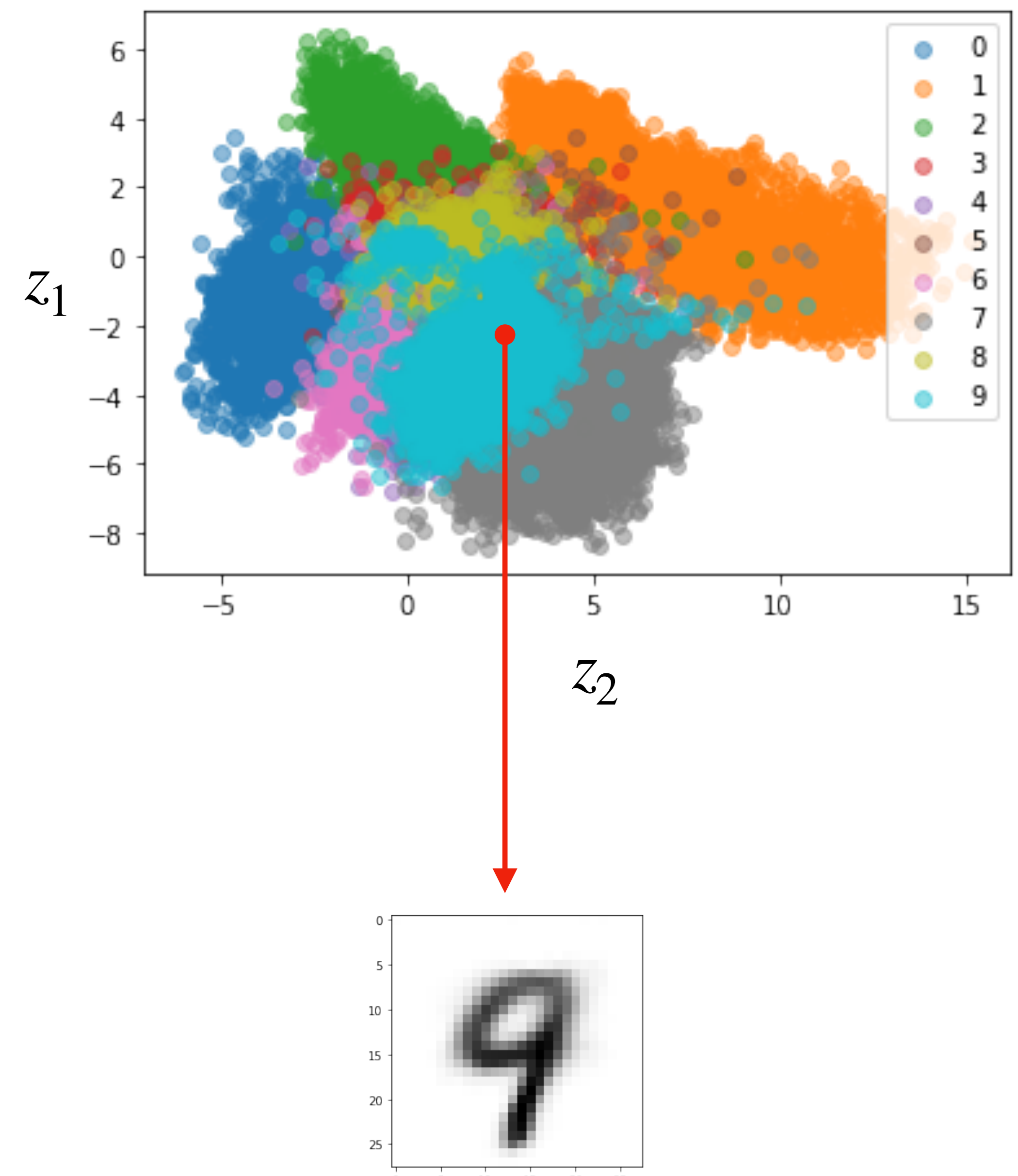


Using Regular Autoencoders for Sampling

Previous Lecture:

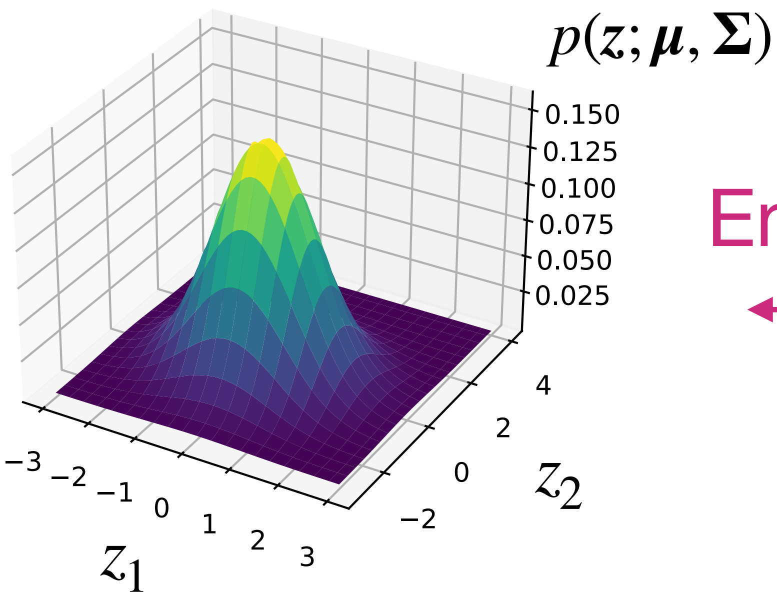
Challenge: regular autoencoders are difficult to sample from, because

1. oddly shaped distribution, hard to sample in a balanced way
2. distribution not centered at (0, 0)
3. distribution not necessarily continuous
   (hard to see here in 2D, but a big problem in higher dimensional latent spaces)

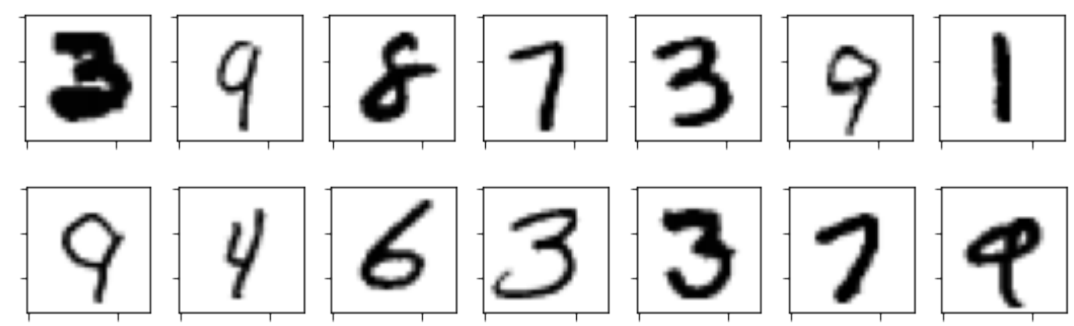


Using Variational Autoencoders (VAEs) for Sampling

This Lecture:



Encoding



d-dimensional probability density for multivariate Gaussian

$$p(z; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^n |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(z-\boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (z-\boldsymbol{\mu})\right)$$

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

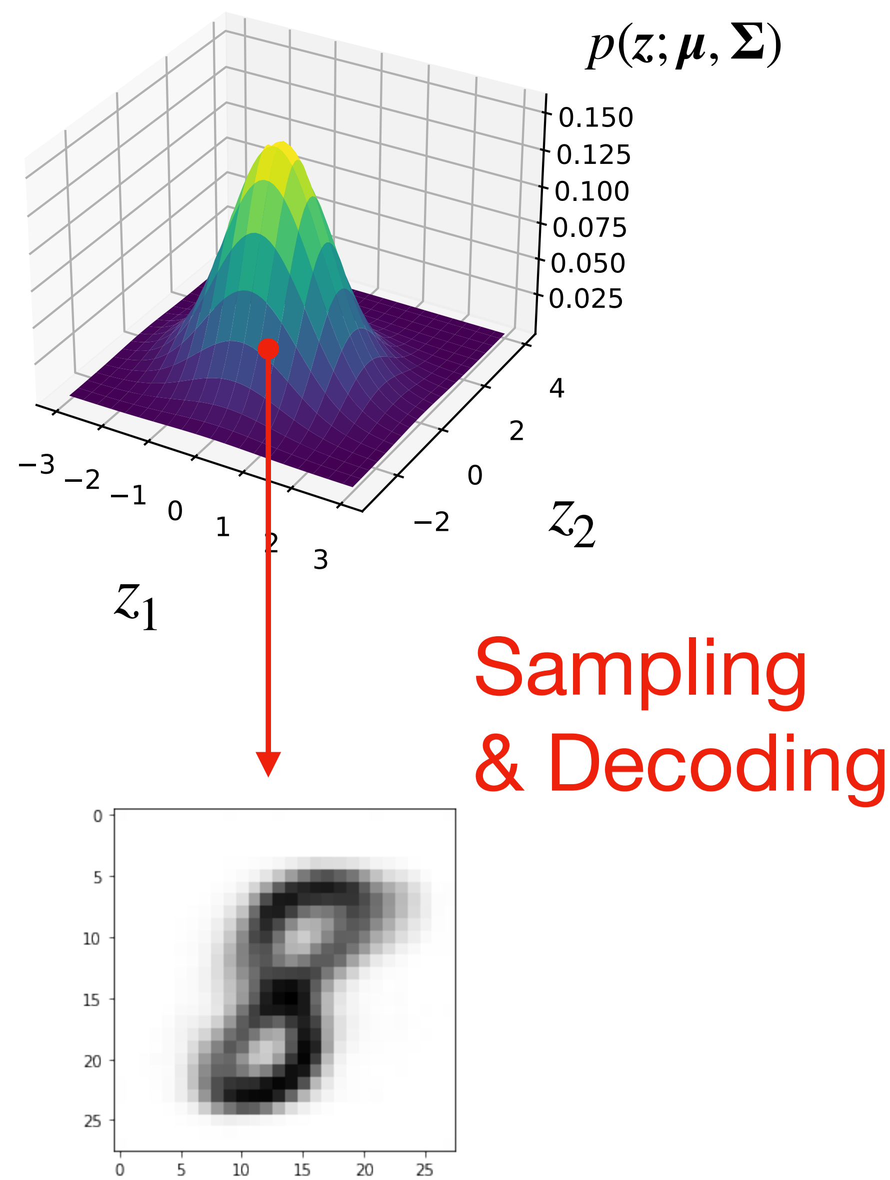
with $\boldsymbol{z} = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}, \boldsymbol{\mu} = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \quad \boldsymbol{\Sigma} = \begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix}$

Sampling from a VAE

$$z = \boldsymbol{\mu} + \boldsymbol{\sigma} \cdot \boldsymbol{\epsilon}$$

Where $\boldsymbol{\sigma}^2 = \begin{pmatrix} \sigma_1^2 \\ \sigma_2^2 \end{pmatrix}$

$$\epsilon_1, \epsilon_2 \sim N(0,1)$$



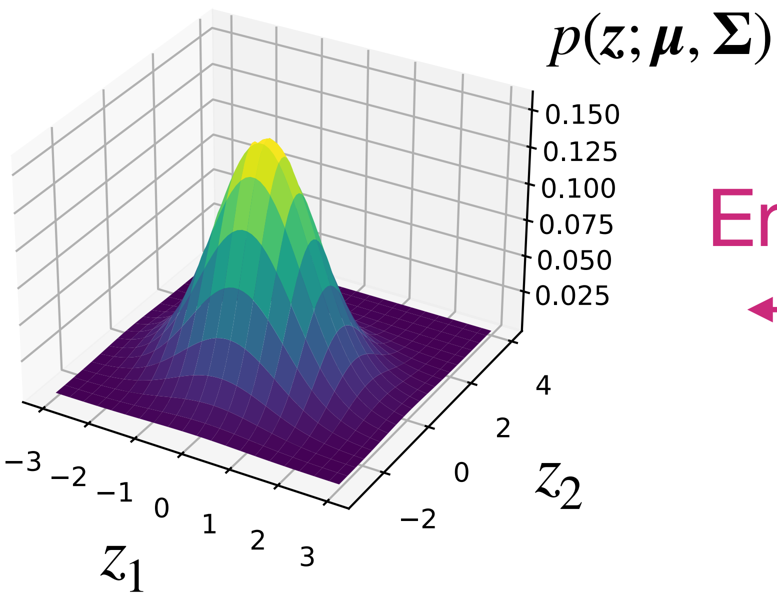
- VAE's assume a diagonal covariance matrix (no interaction between the features).
- Thus, we only need a mean and a variance vector, no covariance matrix

How Can We Use Backropagation with a Probability Distribution?

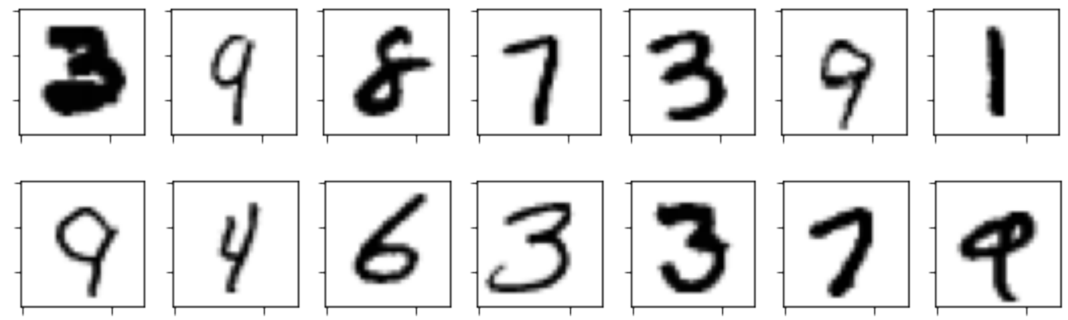
1. Variational Autoencoder Overview
2. Sampling from a Variational Autoencoder
3. **The Log-Var Trick**
4. The Variational Autoencoder Loss Function
5. A Variational Autoencoder for Handwritten Digits in PyTorch
6. A Variational Autoencoder for Face Images in PyTorch
7. VAEs and Latent Space Arithmetic
8. VAE Latent Space Arithmetic in PyTorch -- Making People Smile

Using Variational Autoencoders (VAEs) for Sampling

This Lecture:



Encoding



d-dimensional probability density for multivariate Gaussian

$$p(z; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^n |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(z-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(z-\boldsymbol{\mu})\right)$$

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

with $\boldsymbol{z} = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}, \boldsymbol{\mu} = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \quad \boldsymbol{\Sigma} = \begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix}$

Sampling from a VAE

$$z = \boldsymbol{\mu} + \boldsymbol{\sigma} \cdot \boldsymbol{\epsilon}$$

Sampled from standard multivariate normal distribution in each forward pass

Where $\boldsymbol{\sigma}^2 = \begin{pmatrix} \sigma_1^2 \\ \sigma_2^2 \end{pmatrix}$

$$\epsilon_1, \epsilon_2 \sim N(0,1)$$

But why ϵ? Continuous distribution; VAE must ensure that points in neighborhood encode the same image so that when decoding they produce the same image

Think of these as parameter vectors included in training & backpropagation

Sampling from a VAE -- The Log-Var Trick

Instead of using a variance vector, $\boldsymbol{\sigma}^2 = \begin{pmatrix} \sigma_1^2 \\ \sigma_2^2 \end{pmatrix}$

we use the
log-var vector
to allow for positive and negative values: $\log(\boldsymbol{\sigma}^2)$

Why can we do this?

$$\log(\sigma^2) = 2 \cdot \log(\sigma)$$

$$\log(\sigma^2)/2 = \log(\sigma)$$

$$\sigma = e^{\log(\sigma^2)/2}$$

So, when we sample the points, we can do

$$z = \mu + e^{\log(\sigma^2)/2} \cdot \epsilon$$

Combining Two Objectives

1. Variational Autoencoder Overview
2. Sampling from a Variational Autoencoder
3. The Log-Var Trick
4. **The Variational Autoencoder Loss Function**
5. A Variational Autoencoder for Handwritten Digits in PyTorch
6. A Variational Autoencoder for Face Images in PyTorch
7. VAEs and Latent Space Arithmetic
8. VAE Latent Space Arithmetic in PyTorch -- Making People Smile

Variational Autoencoder

Minimizes ELBO (Evidence lower bound), consisting of KL term and reconstruction loss

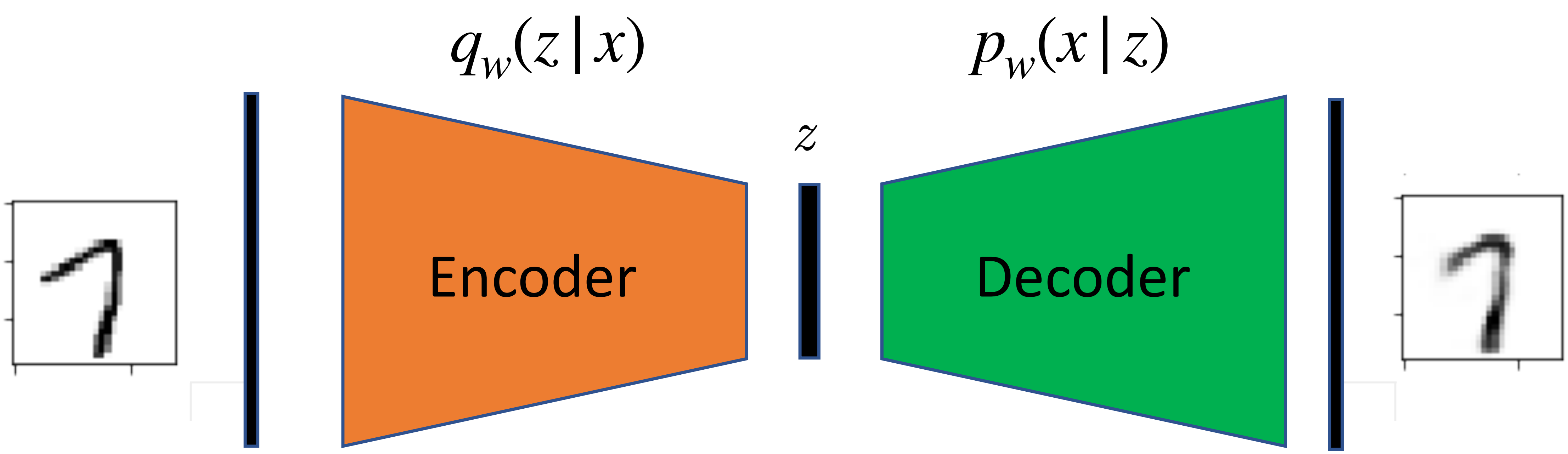
If you assume $p_w(x \mid z)$ follows multivariate-Bernoulli, use cross entropy;
if you assume it follows normal distribution, use MSE

MSE is same as cross-entropy between the empirical distribution and a Gaussian model
(Reference: Deep Learning book by Goodfellow et al., pg. 132)

$$\mathcal{L} = -\mathbb{E}_{z \sim q_w(z \mid x^{[i]})} \left[\log p_w \left(x^{[i]} \mid z \right) \right] + \mathbf{KL} \left(q_w \left(z \mid x^{[i]} \right) \| p(z) \right)$$

Expected neg. log likelihood term; wrt to encoder distribution

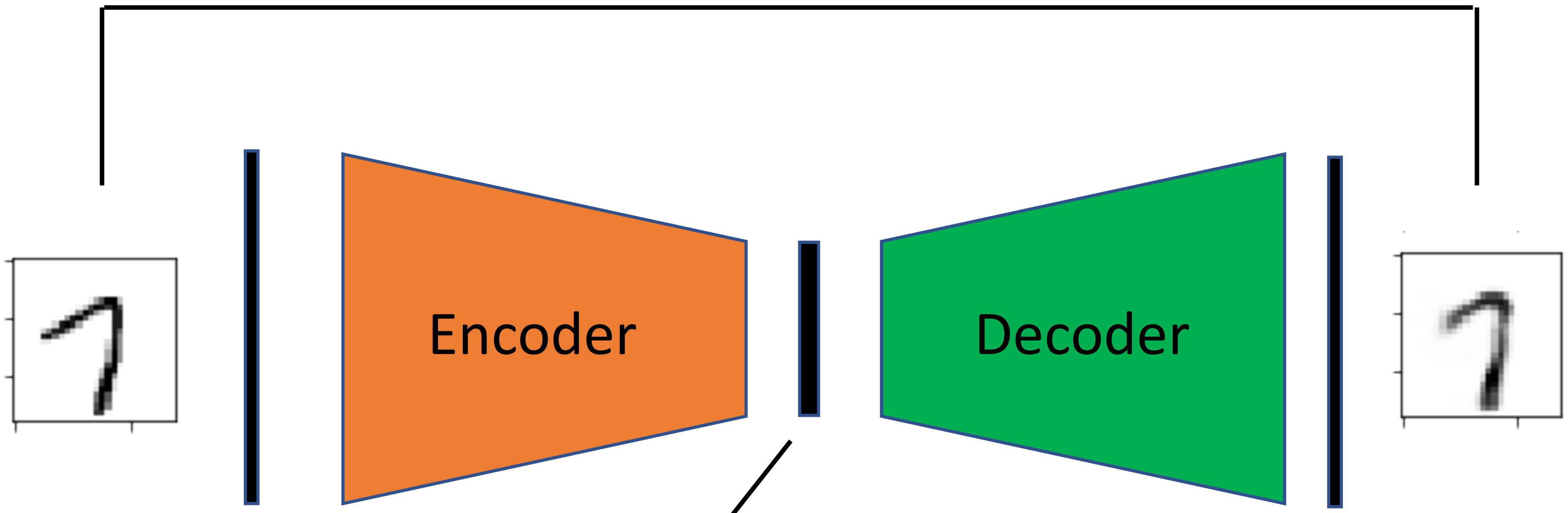
Kullback-Leibler divergence term
where $p(z) = \mathcal{N} \left(\mu = 0, \sigma^2 = 1 \right)$



The Variational Autoencoder Loss Function

1) Minimize squared error loss: (ensures good reconstruction)

$$\mathcal{L}_1 = ||\mathbf{x} - Dec(Enc(\mathbf{x}))||_2^2 = \sum_{i=1}^{d} (x_i - x_i')^2$$



2) Minimize KL divergence:

(ensures latent space is continuous and standard normal distributed)

$$\mathcal{L}_2 = D_{KL}\left[N(\mu, \sigma) \| N(0,1)\right] = -\frac{1}{2} \sum \left(1 + \log\left(\sigma^2\right) - \mu^2 - \sigma^2\right)$$

Overall loss: $\mathcal{L} = \alpha \cdot \mathcal{L}_1 + \mathcal{L}_2$

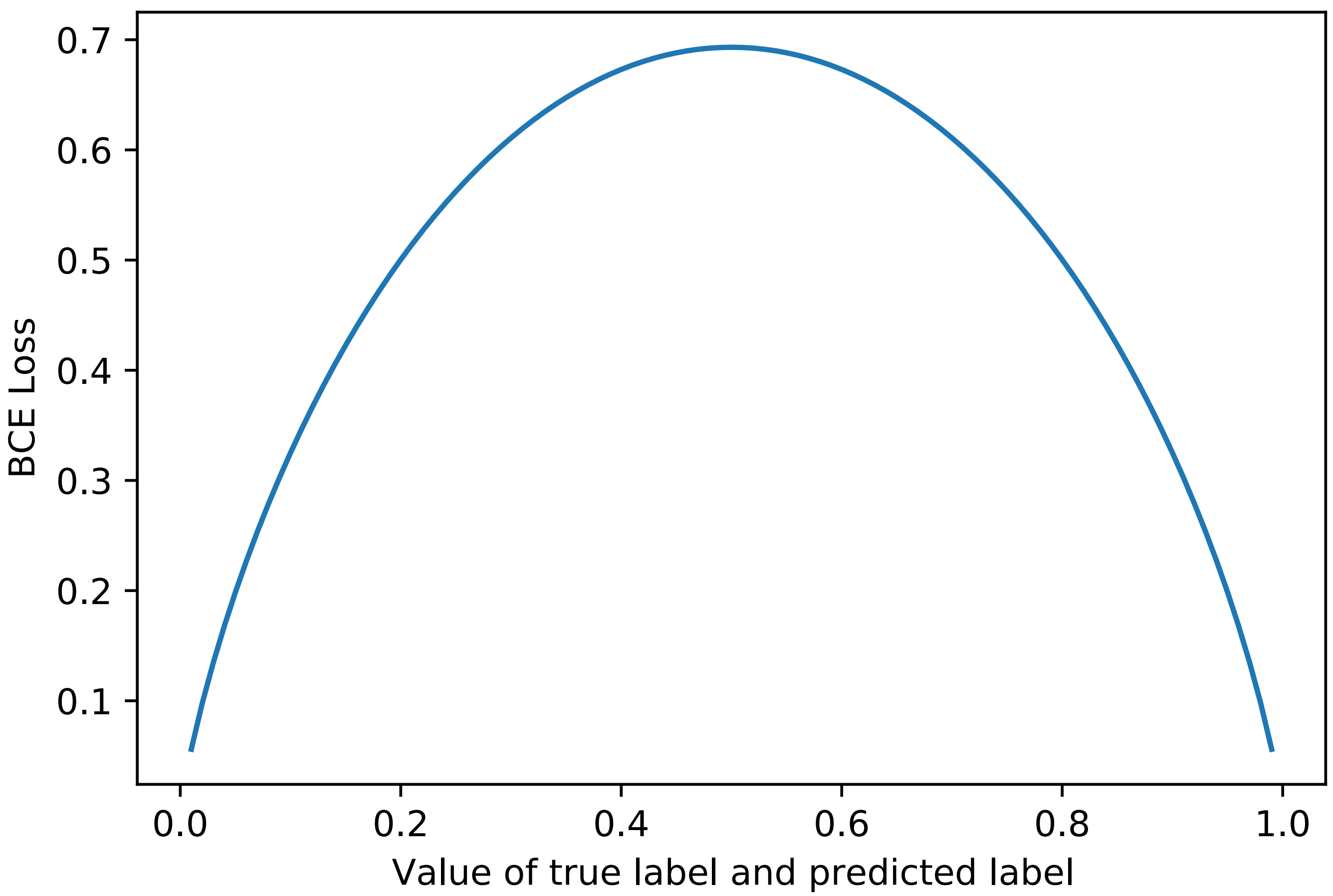
Binary Cross Entropy vs MSE

Cross Entropy is not symmetric:

$$H(p, q) = -\sum_{x \in \mathcal{X}} p(x) \cdot \log q(x)$$

pixel in x

pixel in x'



-0.8 * log(0.7) = 0.285340
-0.8 * log(0.9) = 0.0842884

-0.2 * log(0.1) = 0.460517
-0.2 * log(0.3) = 0.240795

KL Loss Derivation

The encoder distribution is $q(z|x) = \mathcal{N}(z|\mu(x), \Sigma(x))$ where $\Sigma = \text{diag}(\sigma_1^2, \ldots, \sigma_n^2)$.

The latent prior is given by $p(z) = \mathcal{N}(0, I)$.

Both are multivariate Gaussians of dimension n, for which in general the KL divergence is:

$$\mathfrak{D}_{\text{KL}}[p_1 \mid\mid p_2] = \frac{1}{2}\left[\log \frac{|\Sigma_2|}{|\Sigma_1|} - n + \text{tr}\{\Sigma_2^{-1}\Sigma_1\} + (\mu_2 - \mu_1)^T \Sigma_2^{-1}(\mu_2 - \mu_1)\right]$$

where $p_1 = \mathcal{N}(\mu_1, \Sigma_1)$ and $p_2 = \mathcal{N}(\mu_2, \Sigma_2)$.

In the VAE case, $p_1 = q(z|x)$ and $p_2 = p(z)$, so $\mu_1 = \mu$, $\Sigma_1 = \Sigma$, $\mu_2 = \vec{0}$, $\Sigma_2 = I$. Thus:

$$\begin{aligned}
\mathfrak{D}_{\text{KL}}[q(z|x) \mid\mid p(z)] &= \frac{1}{2}\left[\log \frac{|\Sigma_2|}{|\Sigma_1|} - n + \text{tr}\{\Sigma_2^{-1}\Sigma_1\} + (\mu_2 - \mu_1)^T \Sigma_2^{-1}(\mu_2 - \mu_1)\right] \\
&= \frac{1}{2}\left[\log \frac{|I|}{|\Sigma|} - n + \text{tr}\{I^{-1}\Sigma\} + (\vec{0} - \mu)^T I^{-1}(\vec{0} - \mu)\right] \\
&= \frac{1}{2}\left[-\log |\Sigma| - n + \text{tr}\{\Sigma\} + \mu^T \mu\right] \\
&= \frac{1}{2}\left[-\log \prod_i \sigma_i^2 - n + \sum_i \sigma_i^2 + \sum_i \mu_i^2\right] \\
&= \frac{1}{2}\left[-\sum_i \log \sigma_i^2 - n + \sum_i \sigma_i^2 + \sum_i \mu_i^2\right] \\
&= \frac{1}{2}\left[-\sum_i \left(\log \sigma_i^2 + 1\right) + \sum_i \sigma_i^2 + \sum_i \mu_i^2\right]
\end{aligned}$$

Source: https://stats.stackexchange.com/questions/318748/deriving-the-kl-divergence-loss-for-vaes/370048#370048

Implementing Our First Convolutional Variational Autoencoder in PyTorch

1. Variational Autoencoder Overview
2. Sampling from a Variational Autoencoder
3. The Log-Var Trick
4. The Variational Autoencoder Loss Function
5. **A Variational Autoencoder for Handwritten Digits in PyTorch**
6. A Variational Autoencoder for Face Images in PyTorch
7. VAEs and Latent Space Arithmetic
8. VAE Latent Space Arithmetic in PyTorch -- Making People Smile

Variational Autoencoders for Generating New Face Images

A Variational Autoencoder for Face Images

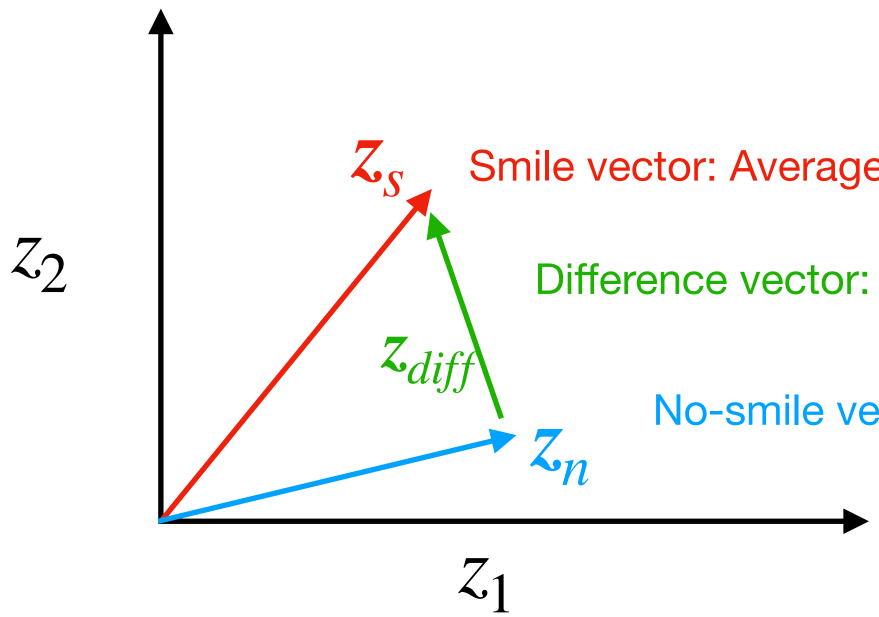
Architectural changes compared to previous MNIST example:

- based on 2016: Deep Feature Consistent Variational Autoencoder (https://ieeexplore.ieee.org/document/7926714)
- 1 -> 3 color channels
- 2 -> 200 latent dim
- BatchNorm, Dropout
- increase reconstruction loss coefficient

Manipulating Images in Latent Space

1. Variational Autoencoder Overview
2. Sampling from a Variational Autoencoder
3. The Log-Var Trick
4. The Variational Autoencoder Loss Function
5. A Variational Autoencoder for Handwritten Digits in PyTorch
6. A Variational Autoencoder for Face Images in PyTorch
7. **VAEs and Latent Space Arithmetic**
8. VAE Latent Space Arithmetic in PyTorch -- Making People Smile

Latent Space Arithmetic



Smile vector: Average embedding of all people with a smile

Difference vector: $z_s - z_n$

No-smile vector: Average embedding of all people without a smile

E.g., we can give a sad person a smile by

- $z_{new} = z_{orig} + \alpha \cdot z_{diff}$

Making People Smile

1. Variational Autoencoder Overview
2. Sampling from a Variational Autoencoder
3. The Log-Var Trick
4. The Variational Autoencoder Loss Function
5. A Variational Autoencoder for Handwritten Digits in PyTorch
6. A Variational Autoencoder for Face Images in PyTorch
7. VAEs and Latent Space Arithmetic
8. **VAE Latent Space Arithmetic in PyTorch -- Making People Smile**