

EN.601.482/682 Deep Learning

Generative Models

(Variational) Autoencoders and Disentanglement

Mathias Unberath, PhD

Assistant Professor

Dept of Computer Science

Johns Hopkins University

Reminder

Acquiring data is relatively easy ...

... but annotating data is comparably difficult.

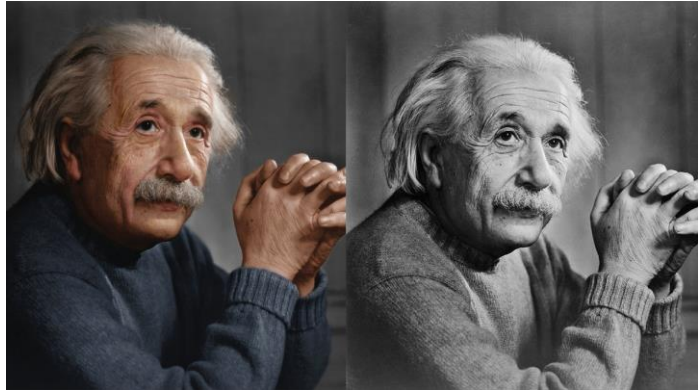
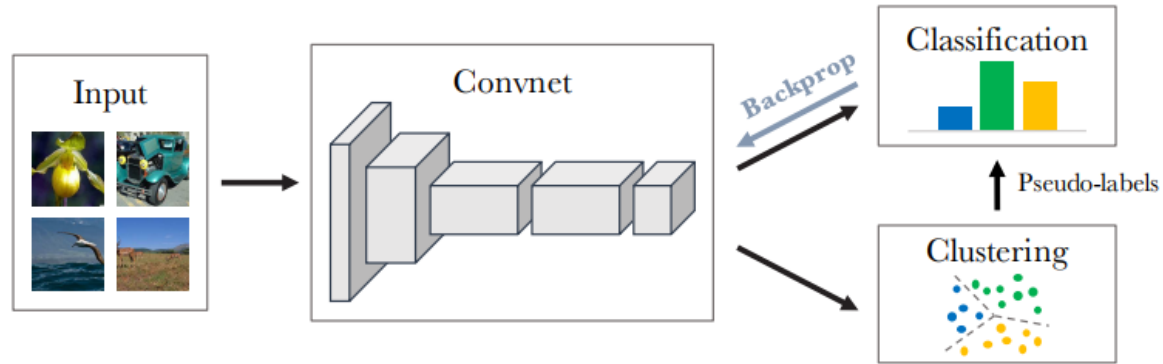
$$\min_{\theta, W} \frac{1}{N} \sum_i L(g_W(f_\theta(x_i)), y_i)$$

→ Learn generalizable, semantic representations of samples solely from data.

- Input x_i with label y_i (**Caveat:** y is derived from data itself)
- f_θ computes high-level representations of x_i
- g_W is a classifier that maps representations to labels

→ Task: Find parameters Θ, W such that loss is optimal

Reminder



Reminder

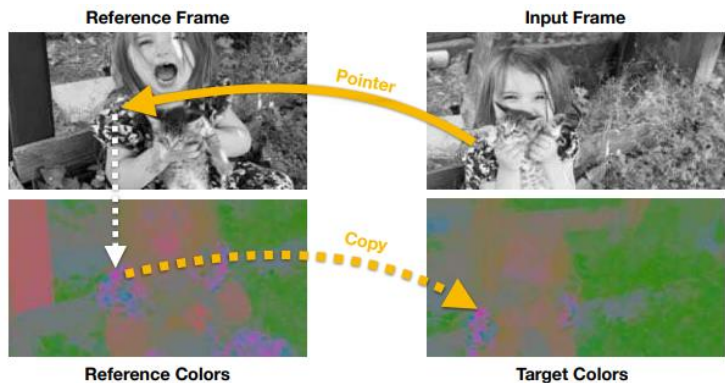


Fig. 1. Self-supervised Tracking: We capitalize on large amounts of unlabeled video to learn a self-supervised model for tracking. The model learns to predict the target colors for a gray-scale input frame by pointing to a colorful reference frame, and copying the color channels. Although we train without ground-truth labels, experiments and visualizations suggest that tracking emerges automatically in this model.

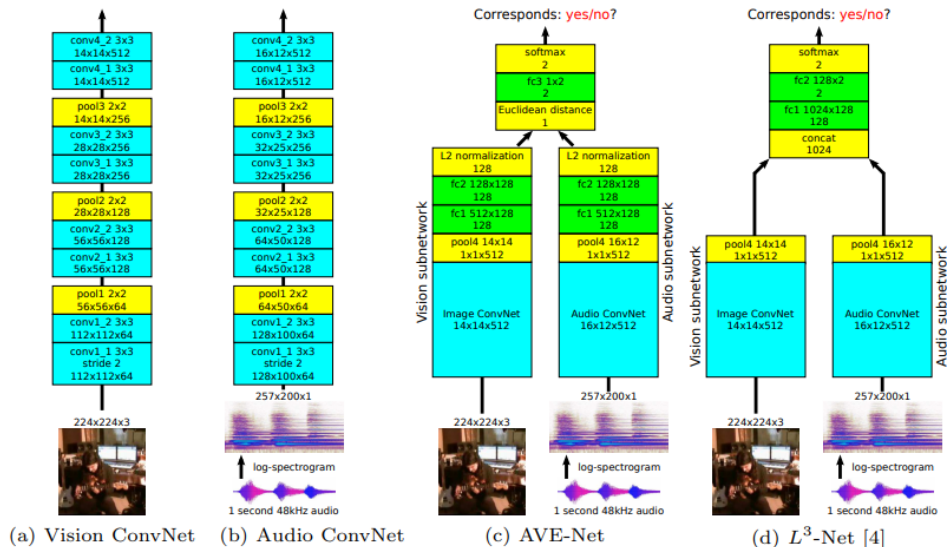


Fig. 2. ConvNet architectures. Each blocks represents a single layer with text providing more information – first row: layer name and optional kernel size, second row: output feature map size. Each convolutional layer is followed by batch normalization [25] and a ReLU nonlinearity, and the first fully connected layer (fc1) is followed by ReLU. All pool layers perform max pooling and their strides are equal to the kernel sizes. (a) and (b) show the vision and audio ConvNets which perform initial feature extraction from the image and audio inputs, respectively. (c) Our AVE-Net is designed to produce aligned vision and audio embeddings as the only information, a single scalar, used to decide whether the two inputs corresponds is the Euclidean distance between the embeddings. (d) In contrast, the L³-Net [4] architecture combines the two modalities by concatenation and a couple of fully connected layers which produce the corresponds or not classification scores.



Reminder

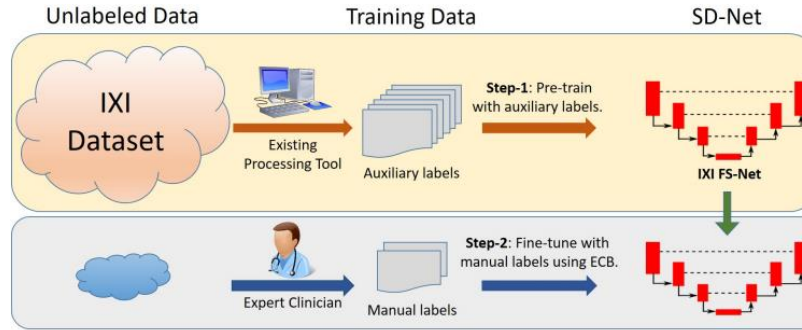
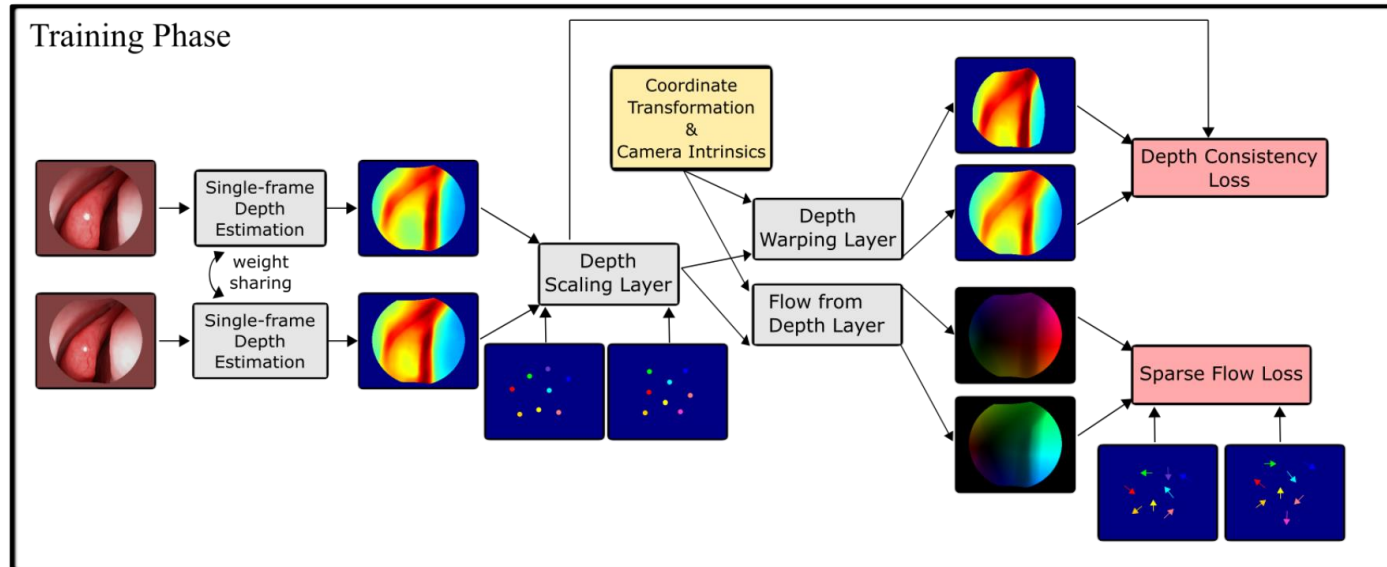


Fig. 1: Illustration of the different steps involved in training of F-CNNs with surplus auxiliary labeled data and limited manually labeled data.



Goal for Today

So far

Image in \rightarrow Encoding (latent representation) \rightarrow Classification out

Now

Random (latent?) representation in \rightarrow Image out

How can we generate samples?

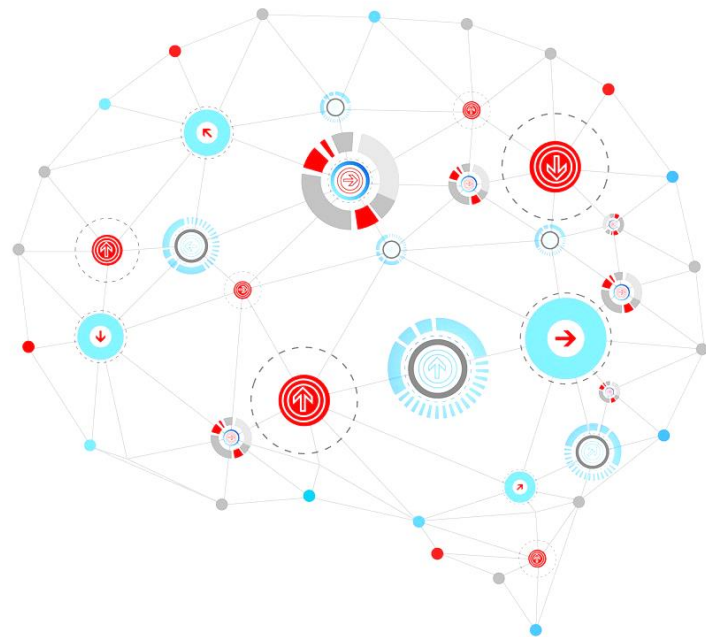
This will be the goal for the two next lectures!

Today's Lecture

The Autoencoder

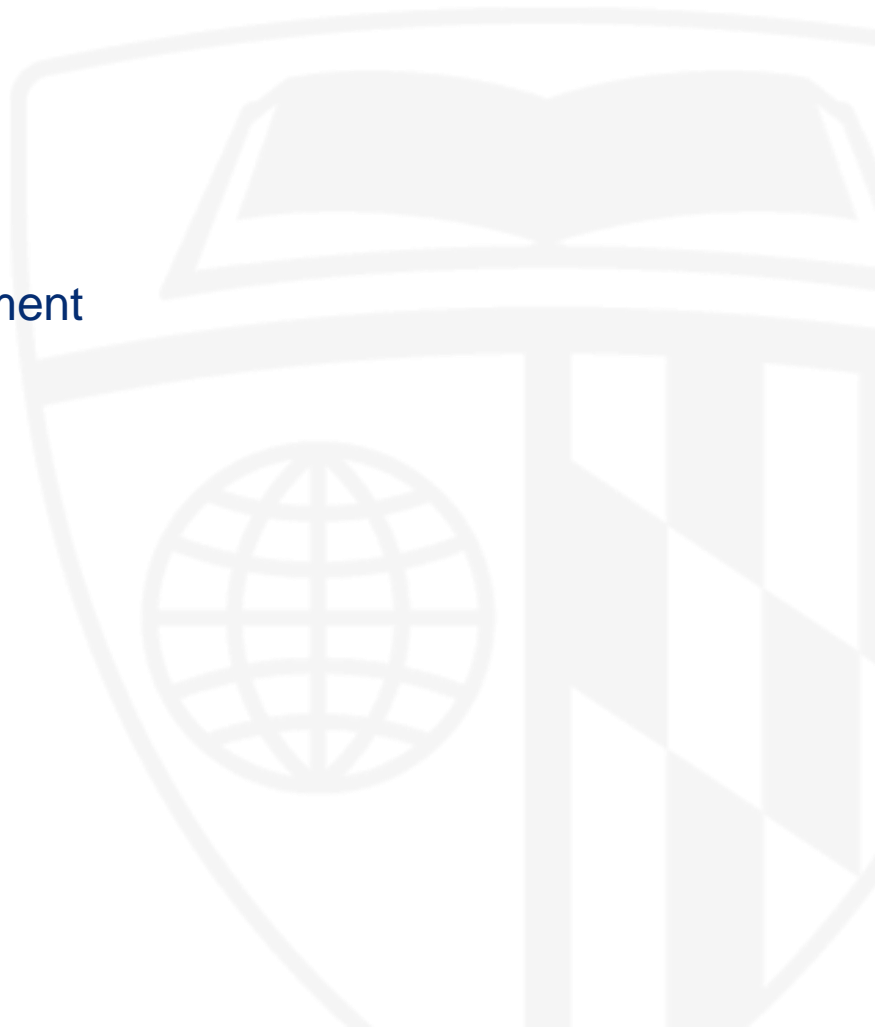
Variational Autoencoder

Disentanglement



(Variational) Autoencoders and Disentanglement

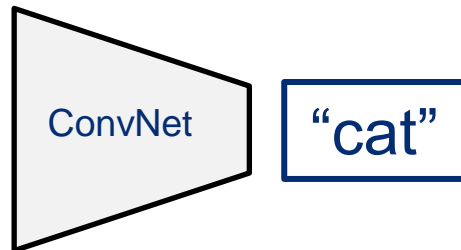
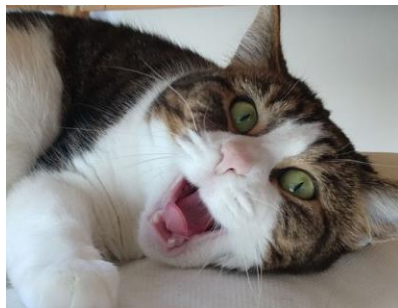
Autoencoder



Learning Meaningful Feature Representations from Data

The image classification problem

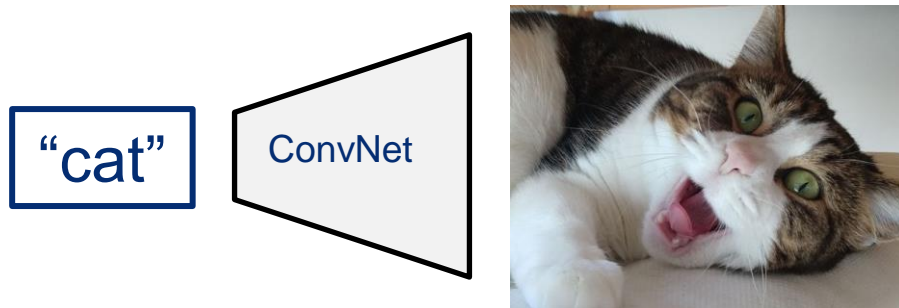
- Given: A large set of labeled images
- Task: Assign label to image



Learning Meaningful Feature Representations from Data

The image generation problem

- Q: How can we achieve this?



Learning Meaningful Feature Representations from Data

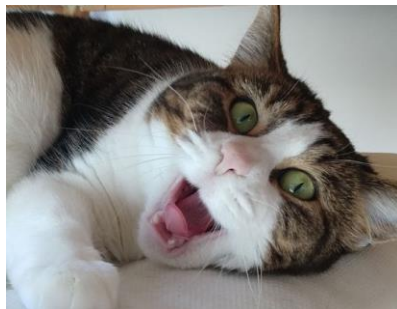
A first step: Autoencoder

- Learning feature representations from unlabeled data
- Underlying idea: Learn to reproduce inputs!

$$\theta = \arg \min_{\hat{\theta}} d[x, h(x, \hat{\theta})]$$

We now use θ for parameters rather than W for consistency with literature

Q: Isn't this trivial?



$f(x, \theta)$



[Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P. A. \(2008, July\). Extracting and composing robust features with denoising autoencoders. ICML.](#)
[Vincent, P. et al. \(2010\). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. JMLR.](#)

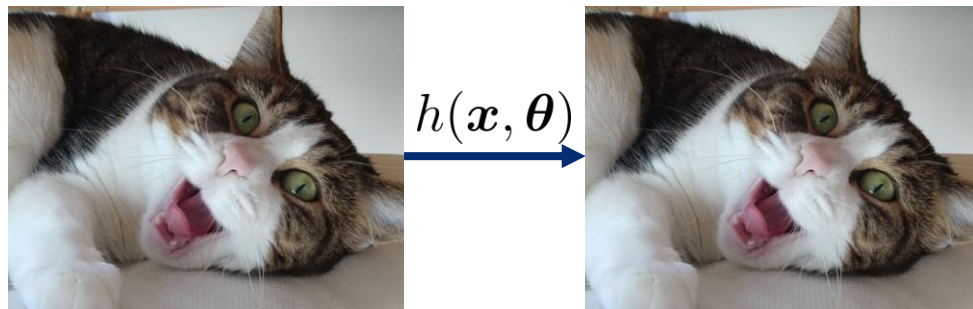
Learning Meaningful Feature Representations from Data

A first step: Autoencoder

- Obviously, the identity mapping does just this. So we need constraints!
- Assumption: Variation in our data is of much lower dimension

$$\theta = \arg \min_{\hat{\theta}} d[\mathbf{x}, h(\mathbf{x}, \hat{\theta})]$$

- Push data through bottleneck!



[Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P. A. \(2008, July\). Extracting and composing robust features with denoising autoencoders. ICML.](#)
[Vincent, P. et al. \(2010\). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. JMLR.](#)

Learning Meaningful Feature Representations from Data

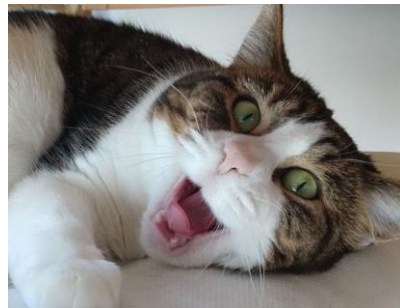
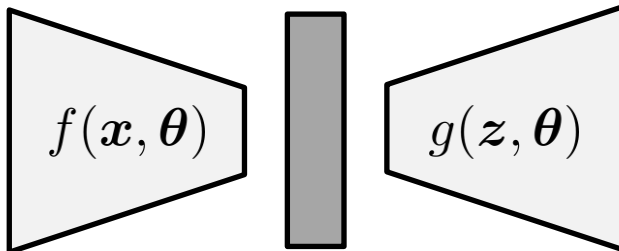
A first step: Autoencoder

- “Encoder”: $f()$ → Encodes input in a lower dimensional latent representation
- “Decoder”: $g()$ → Reconstructs input from latent representation

$$\theta = \arg \min_{\hat{\theta}} d[x, g(f(x, \hat{\theta}), \hat{\theta})]$$

Slight abuse of notation:
 θ now used for both $f()$ and $g()$

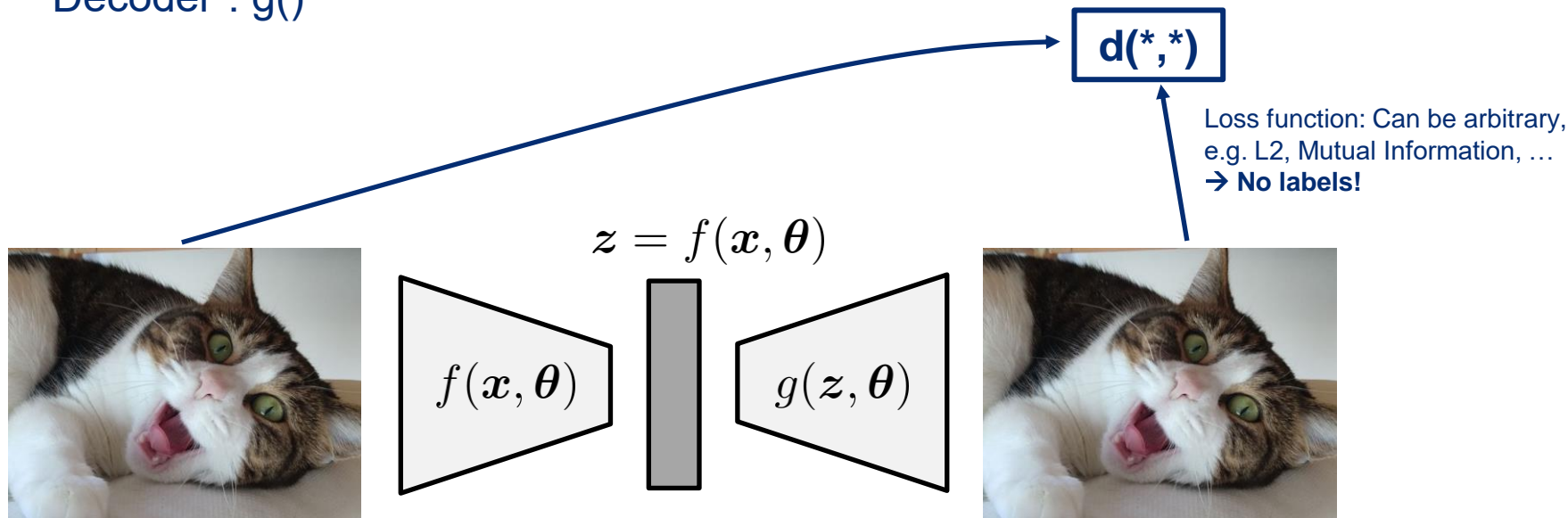
$$z = f(x, \theta)$$



Learning Meaningful Feature Representations from Data

A first step: Autoencoder $\theta = \arg \min_{\hat{\theta}} d[x, g(f(x, \hat{\theta}), \hat{\theta})]$

- “Encoder”: $f()$
- “Decoder”: $g()$

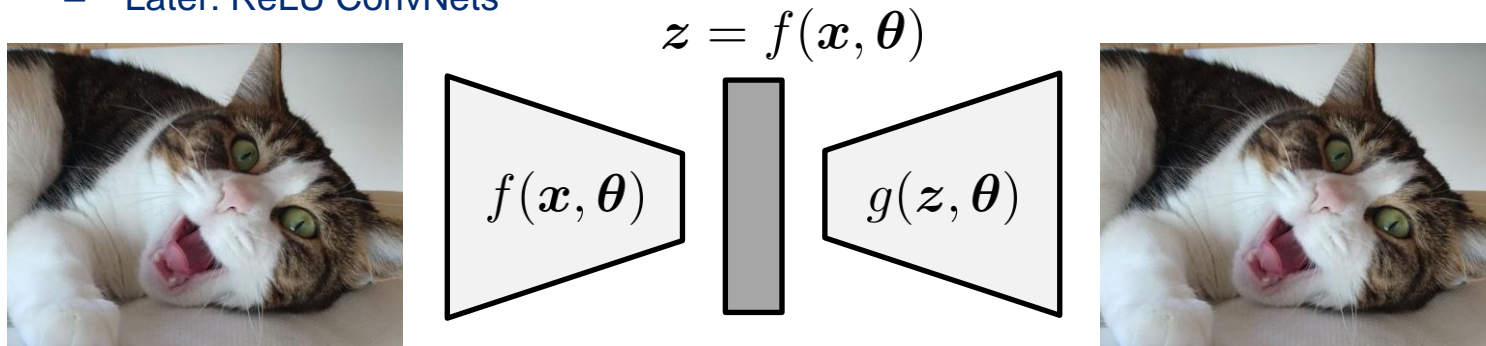


[Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P. A. \(2008, July\). Extracting and composing robust features with denoising autoencoders. ICML.](#)
[Vincent, P. et al. \(2010\). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. JMLR.](#)

Learning Meaningful Feature Representations from Data

A first step: Autoencoder $\theta = \arg \min_{\hat{\theta}} d[x, g(f(x, \hat{\theta}), \hat{\theta})]$

- “Encoder”: $f()$
- “Decoder”: $g()$
- Architecture
 - First: Single fully connected + non-linearity
 - Later: Stacked fully connected
 - Later: ReLU ConvNets

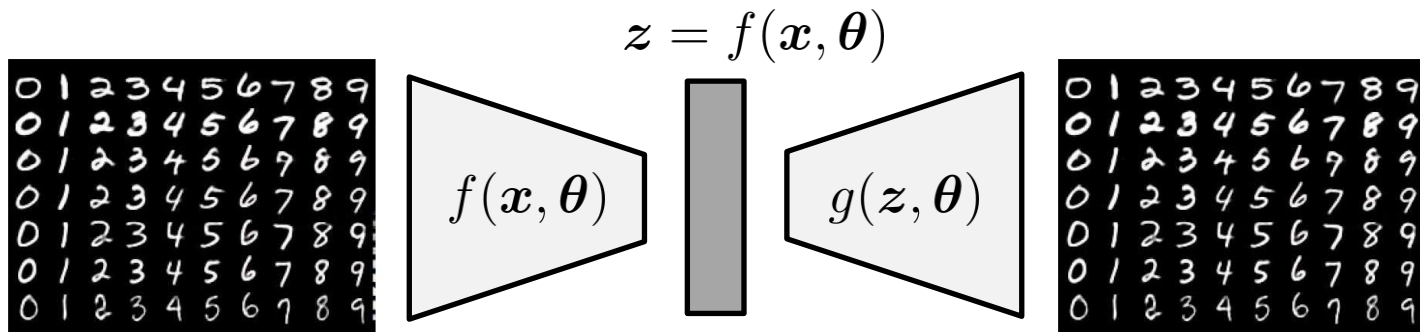


[Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P. A. \(2008, July\). Extracting and composing robust features with denoising autoencoders. ICML.](#)
[Vincent, P. et al. \(2010\). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. JMLR.](#)

Learning Meaningful Feature Representations from Data

A first step: Autoencoder

- Interesting observation (easy case)
Let's consider a stacked AE with 3 hidden layers on MNIST data
- Not convolutions yet, but perceptrons (→ remember, rows are templates!)

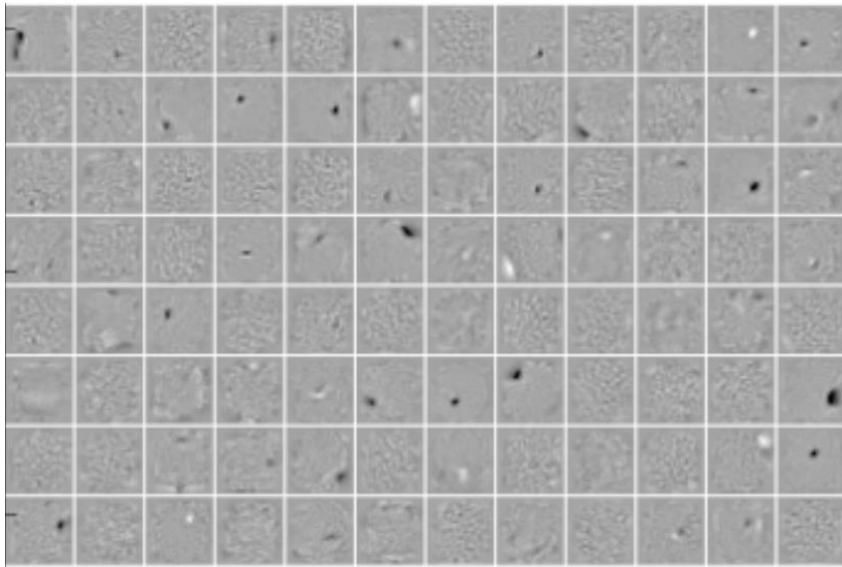


[Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P. A. \(2008, July\). Extracting and composing robust features with denoising autoencoders. ICML.](#)
[Vincent, P. et al. \(2010\). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. JMLR.](#)

Learning Meaningful Feature Representations from Data

A first step: Autoencoder

- Let's visualize the templates of the first layer
- Very noisy! Despite bottleneck, network tries to learn identity! Overfitting!



Q: What now?

If learning identity is a problem, then let's create a problem where the identity mapping is not the solution!

[Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P. A. \(2008, July\). Extracting and composing robust features with denoising autoencoders. ICML.](#)
[Vincent, P. et al. \(2010\). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. JMLR.](#)

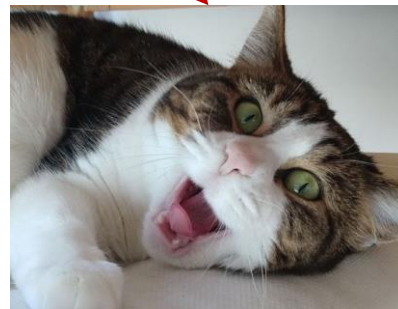
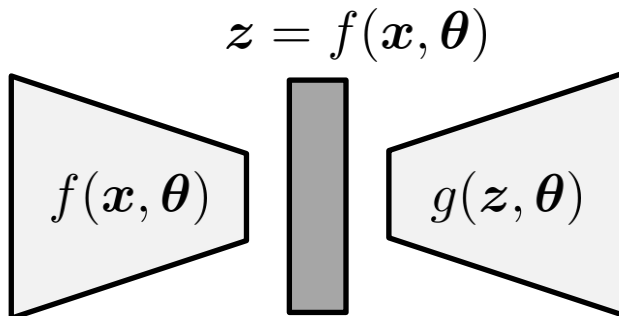
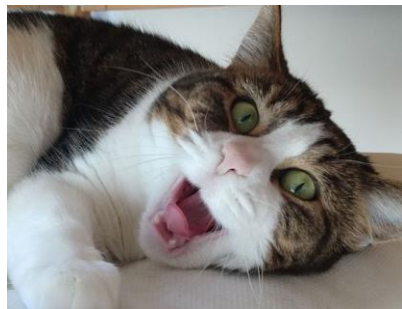
Learning Meaningful Feature Representations from Data

Denoising Autoencoder

- “Encoder”: $f()$
- “Decoder”: $g()$

$$\theta = \arg \min_{\hat{\theta}} d[\boxed{x}, g(f(\tilde{x}, \hat{\theta}), \hat{\theta})]$$

We still do not have/want labels!
So output must remain the same: x



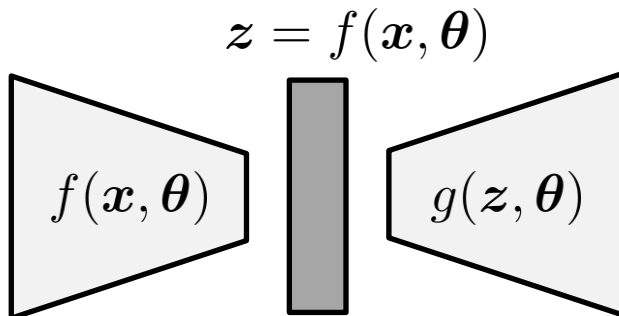
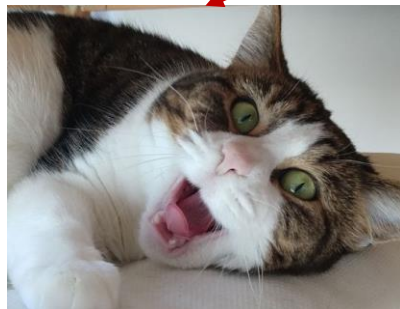
Learning Meaningful Feature Representations from Data

Denoising Autoencoder

- “Encoder”: $f()$
- “Decoder”: $g()$
- **Q: What is \tilde{x} ?**

$$\theta = \arg \min_{\hat{\theta}} d[x, g(f(\tilde{x}, \hat{\theta}), \hat{\theta})]$$

Input must change!
This was just x before!



[Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P. A. \(2008, July\). Extracting and composing robust features with denoising autoencoders. ICML.](#)
[Vincent, P. et al. \(2010\). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. JMLR.](#)

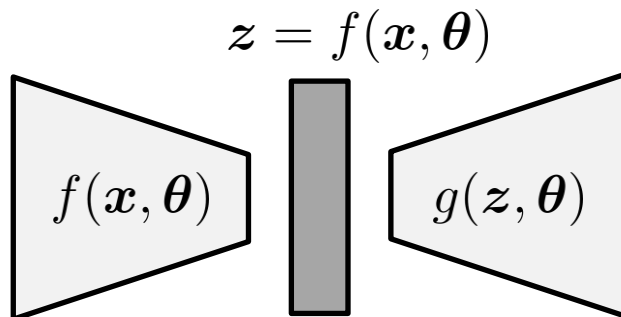
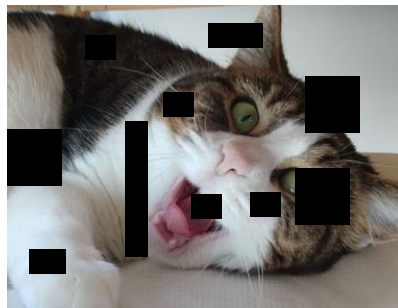
Learning Meaningful Feature Representations from Data

Denoising Autoencoder

$$\theta = \arg \min_{\hat{\theta}} d[x, g(f(\tilde{x}, \hat{\theta}), \hat{\theta})]$$

- “Encoder”: $f()$
- “Decoder”: $g()$
- **Q: What is \tilde{x} ?**

→ Heavily corrupted versions of x ! Corruption: Zero-ing, noise, etc...



[Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P. A. \(2008, July\). Extracting and composing robust features with denoising autoencoders. ICML.](#)
[Vincent, P. et al. \(2010\). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. JMLR.](#)

Learning Meaningful Feature Representations from Data

Denoising Autoencoder

$$\theta = \arg \min_{\hat{\theta}} d[x, g(f(\tilde{x}, \hat{\theta}), \hat{\theta})]$$

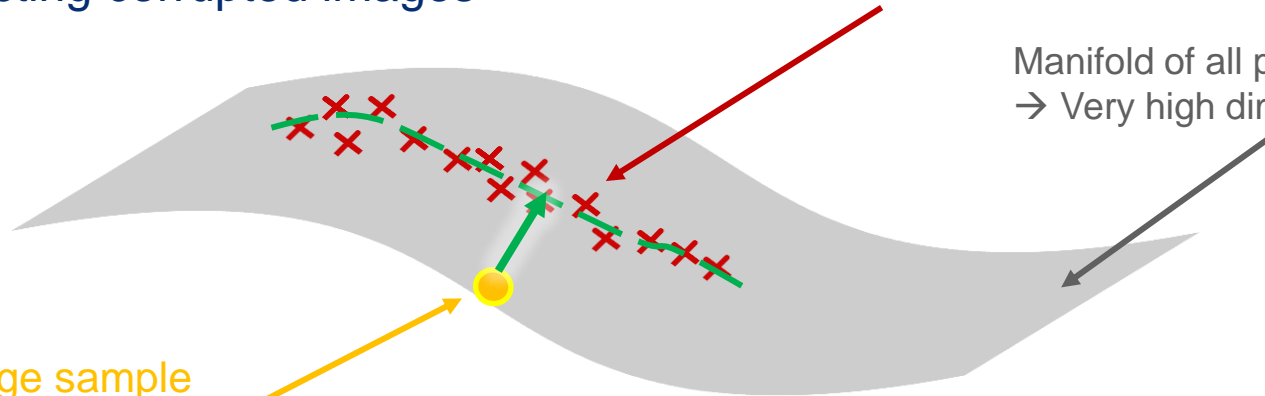
- “Encoder”: $f()$
- “Decoder”: $g()$

→ Reconstructing corrupted images

Valid image sample
→ Hypothesis: Valid images
cluster on a lower dim. manifold

Manifold of all possible images
→ Very high dimensional

Corrupted image sample
→ Autoencoder projects onto
“valid Image manifold”

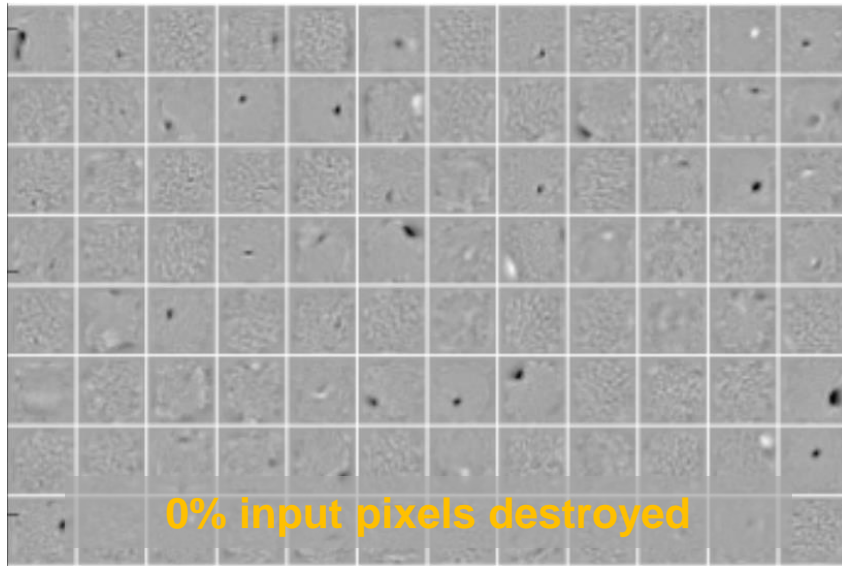


In this picture, the latent representation z can be interpreted as “coordinate system”.

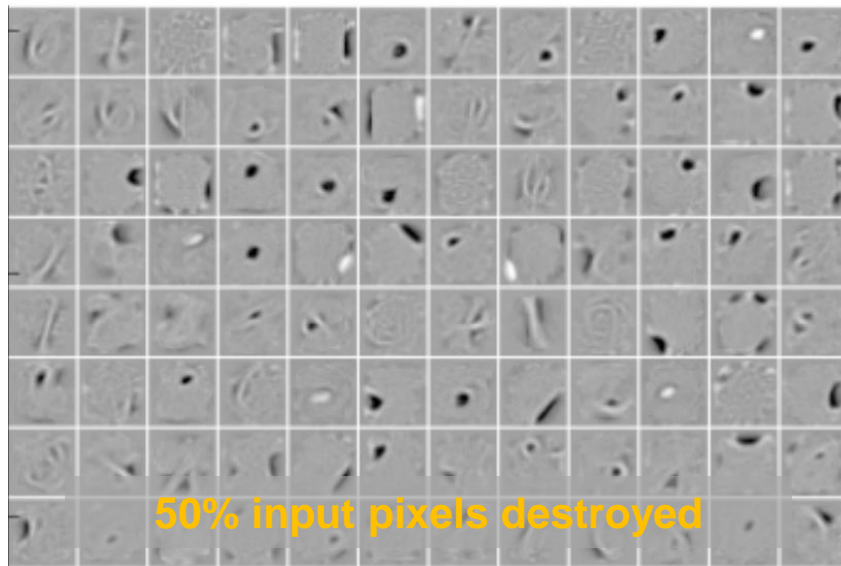
Learning Meaningful Feature Representations from Data

Denoising Autoencoder

- Let's visualize the templates of the first layer
- Corrupting input images yields much more robust features



0% input pixels destroyed



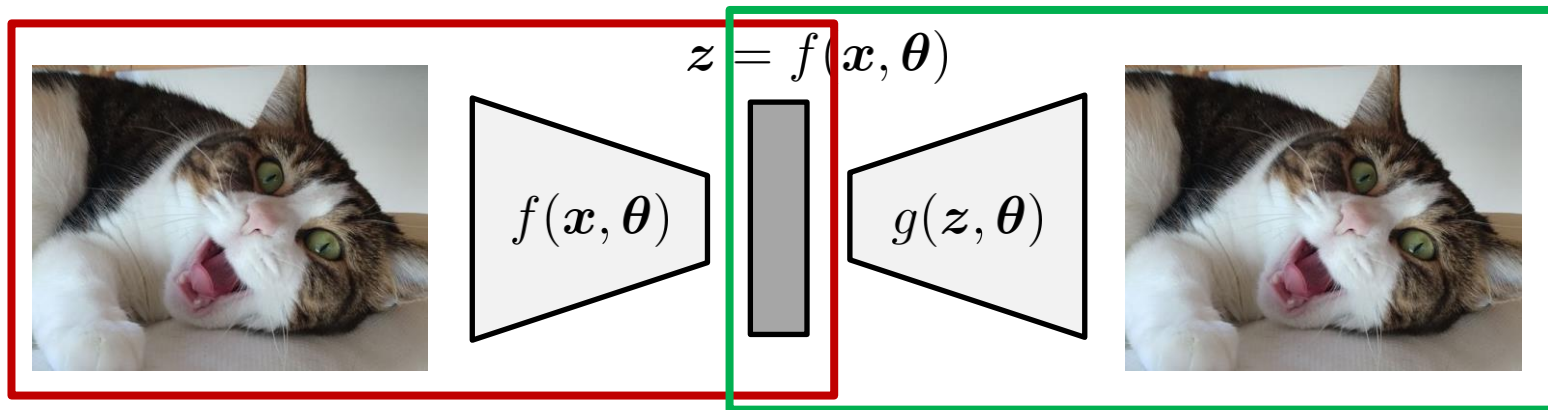
50% input pixels destroyed

[Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P. A. \(2008, July\). Extracting and composing robust features with denoising autoencoders. ICML.](#)
[Vincent, P. et al. \(2010\). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. JMLR.](#)

Learning Meaningful Feature Representations from Data

After training: Two sub-modules!

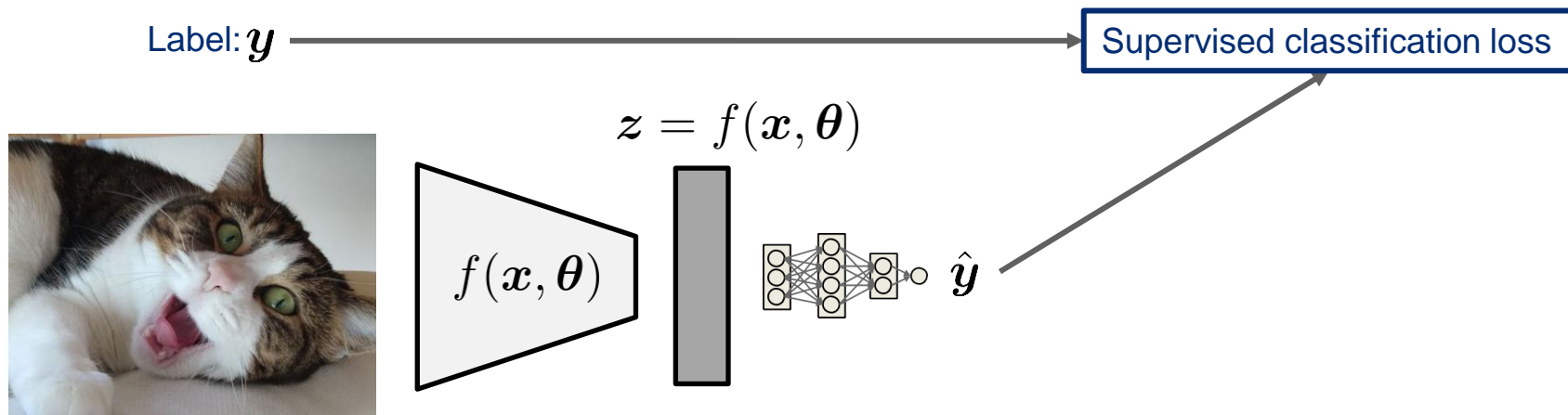
- Pre-trained classification network
- Image generator



Learning Meaningful Feature Representations from Data

After training: Two sub-modules!

- Pre-trained classification network
 - Remove generator
 - Append FC layer(s)
 - Jointly train new parameters and refine encoder

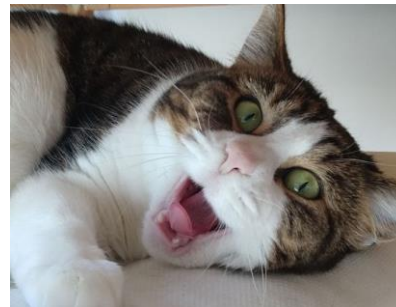
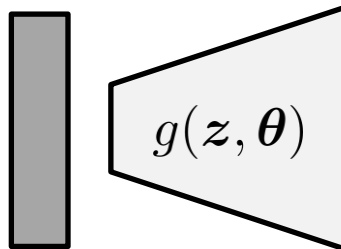


Learning Meaningful Feature Representations from Data

After training: Two sub-modules!

- Image generator
 - Remove encoder
 - Randomly generate latent vector z
 - Generate output sample
 - ??
 - Profit

$$z = f(x, \theta)$$



Learning Meaningful Feature Representations from Data

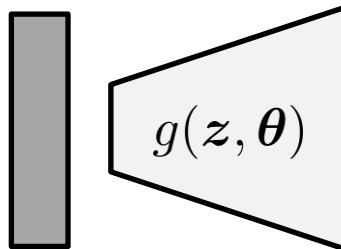
After training: Two sub-modules!

- Image generator
 - Remove encoder
 - Randomly generate latent vector z
 - Generate output sample
 - ??
 - Profit

Bad news! This does not work very well.

Q: Why?

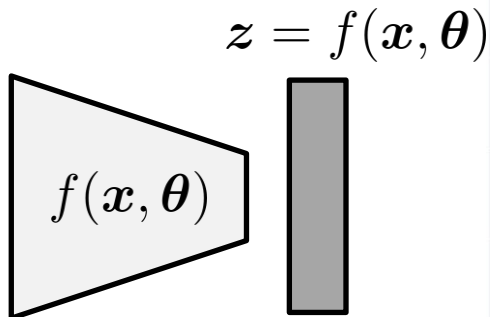
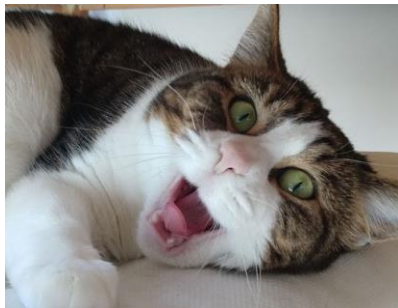
$$z = f(x, \theta)$$



Learning Meaningful Feature Representations from Data

After training: Two sub-modules!

- Image generator
 - Remove encoder
 - Randomly generate latent vector z
 - Generate output sample
 - ??
 - Profit



Q: Why?

Embedding shows clusters

→ Makes sense: Distinct encoding for distinct image appearance

But: Embedding is not continuous

→ Also makes sense, because we did not care

Problematic for generative models.

→ What happens **here**? Never seen for decoding!

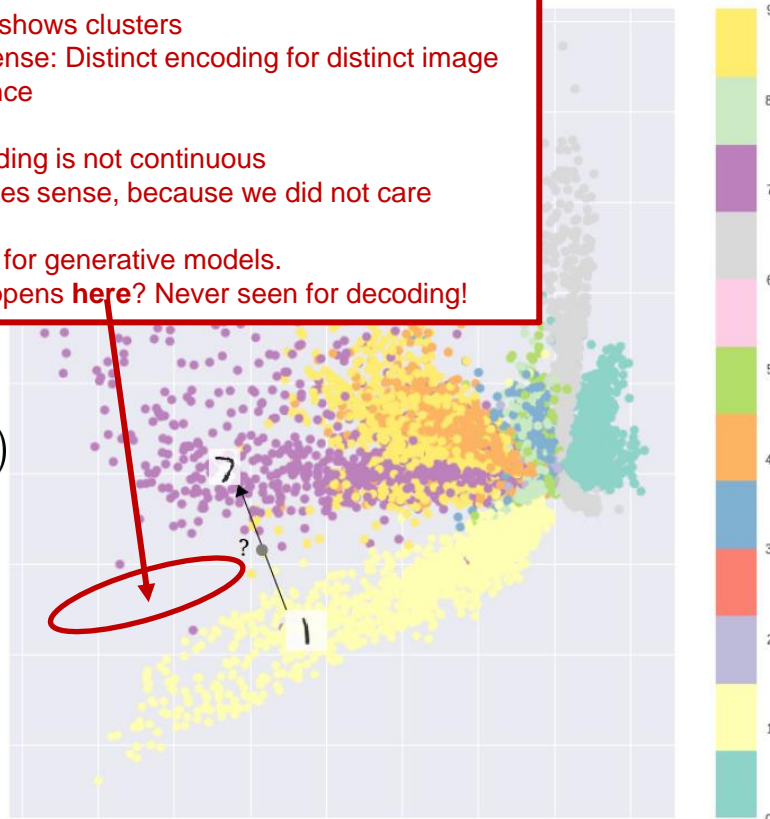


Image from [this blog post](#).



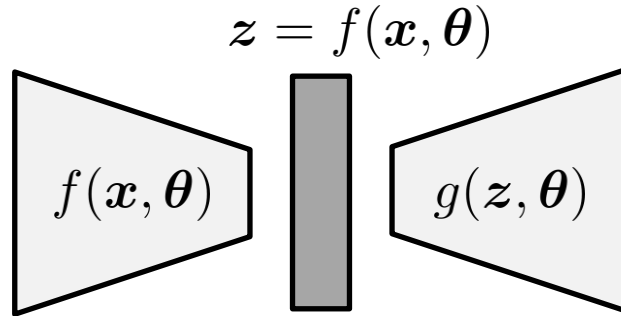
(Variational) Autoencoders and Disentanglement

Variational Autoencoder



Variational Autoencoders: A Probabilistic Spin

The autoencoder

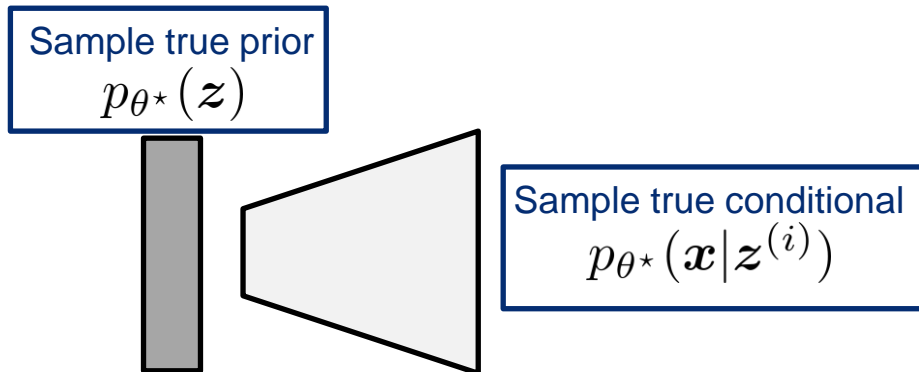


→ We want a generative model, so let's forget about encoder for now

Variational Autoencoders: A Probabilistic Spin

Assume training data $\{\mathbf{x}^{(i)}\}$, $i = 1, \dots, N$ is generated from latent representation \mathbf{z}

Intuitively: \mathbf{z} are latent factors of \mathbf{x} ,
e.g. color of car, amount of smile, etc.

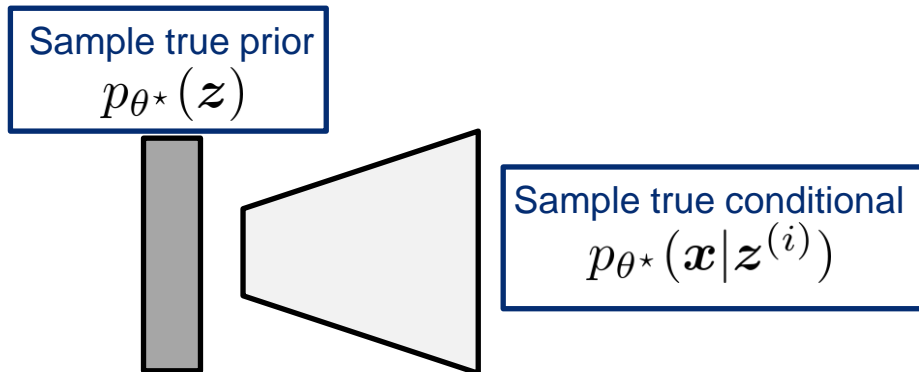


Variational Autoencoders: A Probabilistic Spin

Assume training data $\{\mathbf{x}^{(i)}\}$, $i = 1, \dots, N$ is generated from latent representation \mathbf{z}

Estimate parameters: θ^*

Q: How to represent this model?



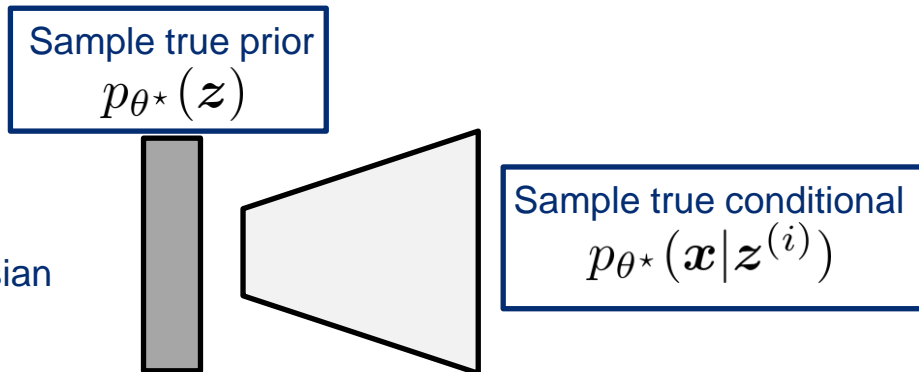
Variational Autoencoders: A Probabilistic Spin

Assume training data $\{\mathbf{x}^{(i)}\}$, $i = 1, \dots, N$ is generated from latent representation \mathbf{z}

Estimate parameters: θ^*

Q: How to represent this model?

Prior $p(\mathbf{z})$ should be simple, e.g. Gaussian
Reasonable for latent attributes



But $p(\mathbf{x}|\mathbf{z})$ is complex \rightarrow Generates image based on latent representation

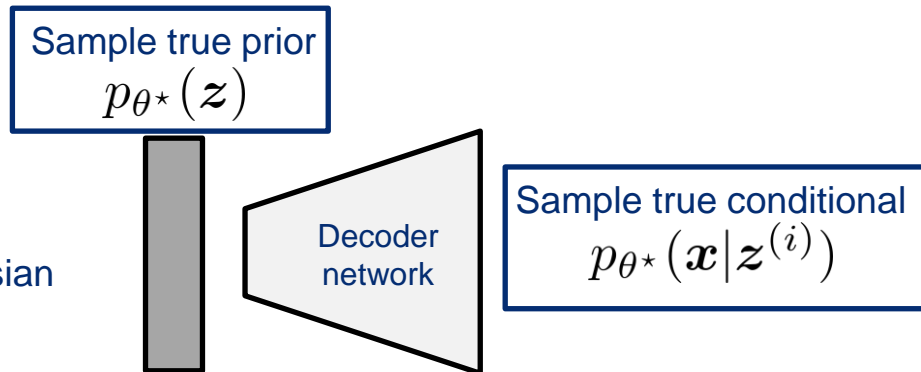
Variational Autoencoders: A Probabilistic Spin

Assume training data $\{\mathbf{x}^{(i)}\}$, $i = 1, \dots, N$ is generated from latent representation \mathbf{z}

Estimate parameters: θ^*

Q: How to represent this model?

Prior $p(\mathbf{z})$ should be simple, e.g. Gaussian
Reasonable for latent attributes



But $p(\mathbf{x}|\mathbf{z})$ is complex \rightarrow Generates image based on latent representation

\rightarrow **Model with neural network**

Variational Autoencoders: A Probabilistic Spin

Assume training data $\{\mathbf{x}^{(i)}\}$, $i = 1, \dots, N$ is generated from latent representation \mathbf{z}

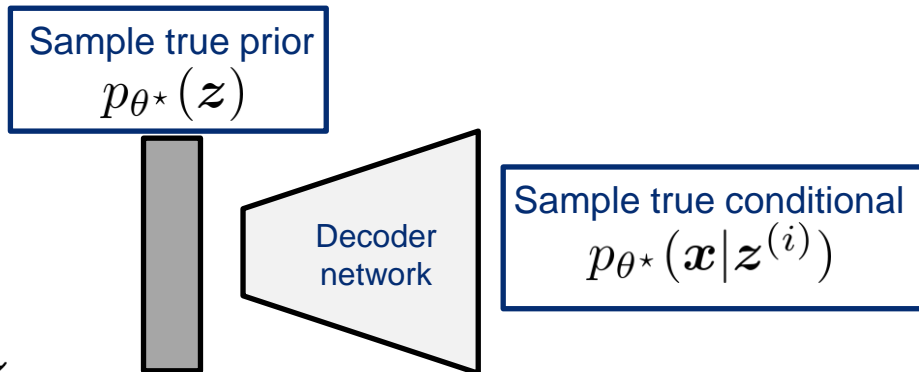
Estimate parameters: θ^*

Q: How to train this model?

Learn parameters that maximize likelihood of training data!

$$p_{\theta}(\mathbf{x}) = \int p_{\theta}(\mathbf{z}) p_{\theta}(\mathbf{x}|\mathbf{z}) d\mathbf{z}$$

Q: Any problem with this?



Variational Autoencoders: A Probabilistic Spin

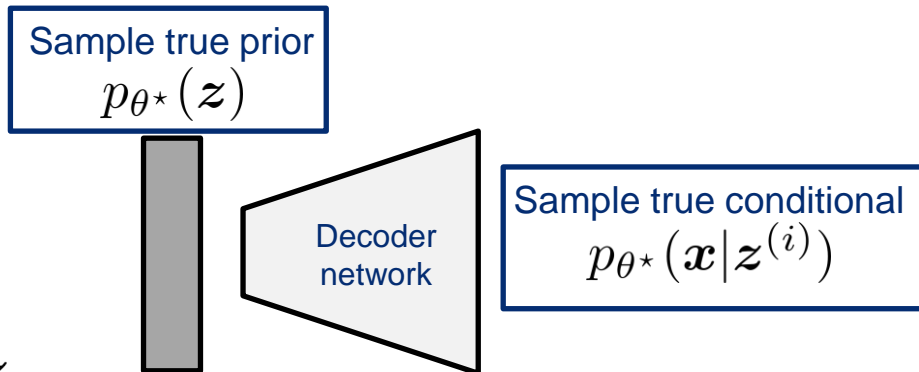
Assume training data $\{\mathbf{x}^{(i)}\}$, $i = 1, \dots, N$ is generated from latent representation \mathbf{z}

Estimate parameters: θ^*

Q: How to train this model?

Learn parameters that maximize likelihood of training data!

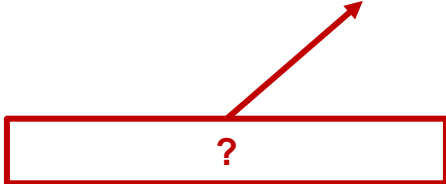
$$p_{\theta}(\mathbf{x}) = \int p_{\theta}(\mathbf{z}) p_{\theta}(\mathbf{x}|\mathbf{z}) d\mathbf{z}$$



Q: Any problem with this?

→ Intractable!

Data Likelihood and Intractability

$$p_{\theta}(\mathbf{x}) = \int \boxed{p_{\theta}(\mathbf{z})} p_{\theta}(\mathbf{x}|\mathbf{z}) \mathrm{d}\mathbf{z}$$


Data Likelihood and Intractability

$$p_{\theta}(\mathbf{x}) = \int \boxed{p_{\theta}(\mathbf{z})} \boxed{p_{\theta}(\mathbf{x}|\mathbf{z})} d\mathbf{z}$$

Diagram illustrating the equation $p_{\theta}(\mathbf{x}) = \int p_{\theta}(\mathbf{z}) p_{\theta}(\mathbf{x}|\mathbf{z}) d\mathbf{z}$. A green box highlights $p_{\theta}(\mathbf{z})$, with a green arrow pointing to it from a box labeled "Simple Gaussian prior". A red box highlights $p_{\theta}(\mathbf{x}|\mathbf{z})$, with a red arrow pointing to it from a box labeled "?".

Data Likelihood and Intractability

$$p_{\theta}(\mathbf{x}) = \int p_{\theta}(\mathbf{z}) p_{\theta}(\mathbf{x}|\mathbf{z}) d\mathbf{z}$$

Diagram illustrating the components of the equation:

- Simple Gaussian prior** points to $p_{\theta}(\mathbf{z})$.
- Decoder network** points to $p_{\theta}(\mathbf{x}|\mathbf{z})$.

Data Likelihood and Intractability

Intractable to compute
 $p(x|z)$ for **every** z

Decoder network

$$p_{\theta}(x) = \int p_{\theta}(z) p_{\theta}(x|z) dz$$

Simple Gaussian prior

Data Likelihood and Intractability

$$p_{\theta}(\mathbf{x}) = \int p_{\theta}(\mathbf{z}) p_{\theta}(\mathbf{x}|\mathbf{z}) d\mathbf{z}$$

Posterior density is also intractable!

$$p_{\theta}(\mathbf{z}|\mathbf{x}) = p_{\theta}(\mathbf{x}|\mathbf{z}) p_{\theta}(\mathbf{z}) / p_{\theta}(\mathbf{x})$$

Data Likelihood and Intractability

$$p_{\theta}(\mathbf{x}) = \int p_{\theta}(\mathbf{z}) p_{\theta}(\mathbf{x}|\mathbf{z}) d\mathbf{z}$$

Posterior density is also intractable!

$$p_{\theta}(\mathbf{z}|\mathbf{x}) = p_{\theta}(\mathbf{x}|\mathbf{z}) p_{\theta}(\mathbf{z}) / p_{\theta}(\mathbf{x})$$

Intractable data likelihood

Variational Autoencoders

Data likelihood: $p_{\theta}(\mathbf{x}) = \int p_{\theta}(\mathbf{z}) p_{\theta}(\mathbf{x}|\mathbf{z}) \mathrm{d}\mathbf{z}$

Posterior density: $p_{\theta}(\mathbf{z}|\mathbf{x}) = p_{\theta}(\mathbf{x}|\mathbf{z}) p_{\theta}(\mathbf{z}) / p_{\theta}(\mathbf{x})$

Decoder network models $p_{\theta}(\mathbf{x}|\mathbf{z})$

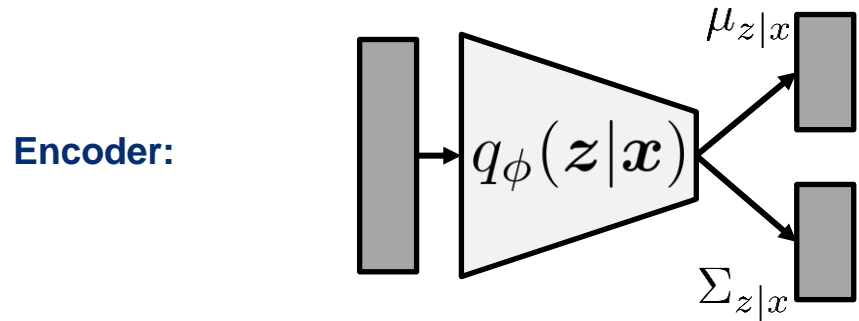
→ Encoder network $q_{\phi}(\mathbf{z}|\mathbf{x})$ to approximate $p_{\theta}(\mathbf{z}|\mathbf{x})$

→ Allows derivation of a lower bound on the data likelihood that is **tractable**
(can be optimized)

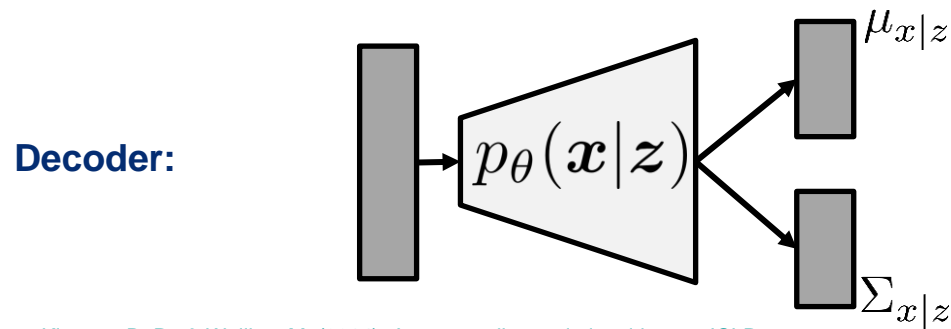


Variational Autoencoders

Probabilistic generation: Encoder and decoder are probabilistic!



Mean and covariance (diagonal) of $\mathbf{z}|\mathbf{x}$

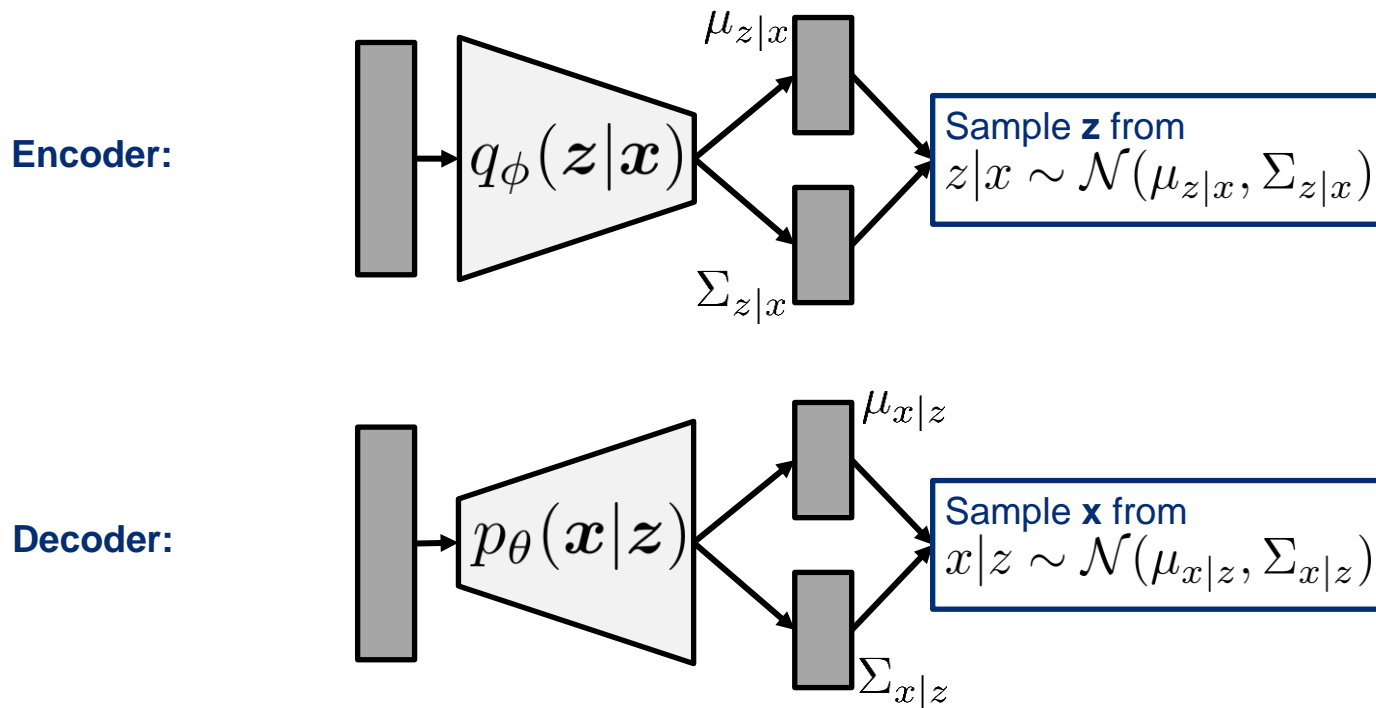


Mean and covariance (diagonal) of $\mathbf{x}|\mathbf{z}$

[Kingma, D. P., & Welling, M. \(2014\). Auto-encoding variational bayes. ICLR.](#)
[Doersch, C. \(2016\). Tutorial on variational autoencoders. arXiv preprint arXiv:1606.05908.](#)

Variational Autoencoders

Probabilistic generation: Encoder and decoder are probabilistic!



[Kingma, D. P., & Welling, M. \(2014\). Auto-encoding variational bayes. ICLR.](#)
[Doersch, C. \(2016\). Tutorial on variational autoencoders. arXiv preprint arXiv:1606.05908.](#)

Variational Autoencoders

Deriving the log data likelihood

$$\log p_{\theta}(\mathbf{x}^{(i)}) = \mathbf{E}_{\mathbf{z} \sim q_{\phi}(\mathbf{z}|\mathbf{x}^{(i)})} \left[\log p_{\theta}(\mathbf{x}^{(i)}) \right] \quad (p_{\theta}(\mathbf{x}^{(i)})) \text{ does not depend on } \mathbf{z}$$


Expectation w.r.t. \mathbf{z}

→ can be done since $p(\mathbf{x})$ does not depend on \mathbf{z} , but will come in handy later

Variational Autoencoders

Deriving the log data likelihood

$$\begin{aligned}\log p_{\theta}(\mathbf{x}^{(i)}) &= \mathbf{E}_{\mathbf{z} \sim q_{\phi}(\mathbf{z}|\mathbf{x}^{(i)})} \left[\log p_{\theta}(\mathbf{x}^{(i)}) \right] \quad (p_{\theta}(\mathbf{x}^{(i)}) \text{ does not depend on } \mathbf{z}) \\ &= \mathbf{E}_{\mathbf{z}} \left[\log \frac{p_{\theta}(\mathbf{x}^{(i)}|\mathbf{z}) p_{\theta}(\mathbf{z})}{p_{\theta}(\mathbf{z}|\mathbf{x}^{(i)})} \right] \quad (\text{Bayes' rule})\end{aligned}$$


$$p(A|B) = \frac{p(B|A) p(A)}{p(B)} \rightarrow p(B) = \frac{p(B|A) p(A)}{p(A|B)}$$

Variational Autoencoders

Deriving the log data likelihood

$$\begin{aligned}\log p_{\theta}(\mathbf{x}^{(i)}) &= \mathbf{E}_{\mathbf{z} \sim q_{\phi}(\mathbf{z}|\mathbf{x}^{(i)})} \left[\log p_{\theta}(\mathbf{x}^{(i)}) \right] \quad (p_{\theta}(\mathbf{x}^{(i)})) \text{ does not depend on } \mathbf{z} \\ &= \mathbf{E}_{\mathbf{z}} \left[\log \frac{p_{\theta}(\mathbf{x}^{(i)}|\mathbf{z}) p_{\theta}(\mathbf{z})}{p_{\theta}(\mathbf{z}|\mathbf{x}^{(i)})} \right] \quad (\text{Bayes' rule}) \\ &= \mathbf{E}_{\mathbf{z}} \left[\log \frac{p_{\theta}(\mathbf{x}^{(i)}|\mathbf{z}) p_{\theta}(\mathbf{z})}{p_{\theta}(\mathbf{z}|\mathbf{x}^{(i)})} \frac{q_{\phi}(\mathbf{z}|\mathbf{x}^{(i)})}{q_{\phi}(\mathbf{z}|\mathbf{x}^{(i)})} \right] \quad (\text{Multiply with constant})\end{aligned}$$

Variational Autoencoders

Deriving the log data likelihood

$$\begin{aligned}\log p_{\theta}(\mathbf{x}^{(i)}) &= \mathbf{E}_{\mathbf{z} \sim q_{\phi}(\mathbf{z}|\mathbf{x}^{(i)})} \left[\log p_{\theta}(\mathbf{x}^{(i)}) \right] \quad (p_{\theta}(\mathbf{x}^{(i)}) \text{ does not depend on } \mathbf{z}) \\&= \mathbf{E}_{\mathbf{z}} \left[\log \frac{p_{\theta}(\mathbf{x}^{(i)}|\mathbf{z}) p_{\theta}(\mathbf{z})}{p_{\theta}(\mathbf{z}|\mathbf{x}^{(i)})} \right] \quad (\text{Bayes' rule}) \\&= \mathbf{E}_{\mathbf{z}} \left[\log \frac{p_{\theta}(\mathbf{x}^{(i)}|\mathbf{z}) p_{\theta}(\mathbf{z})}{p_{\theta}(\mathbf{z}|\mathbf{x}^{(i)})} \frac{q_{\phi}(\mathbf{z}|\mathbf{x}^{(i)})}{q_{\phi}(\mathbf{z}|\mathbf{x}^{(i)})} \right] \quad (\text{Multiply with constant}) \\&= \mathbf{E}_{\mathbf{z}} \left[\log p_{\theta}(\mathbf{x}^{(i)}|\mathbf{z}) \right] - \mathbf{E}_{\mathbf{z}} \left[\log \frac{q_{\phi}(\mathbf{z}|\mathbf{x}^{(i)})}{p_{\theta}(\mathbf{z})} \right] + \mathbf{E}_{\mathbf{z}} \left[\log \frac{q_{\phi}(\mathbf{z}|\mathbf{x}^{(i)})}{p_{\theta}(\mathbf{z}|\mathbf{x}^{(i)})} \right] \quad (\text{Logarithms})\end{aligned}$$

Variational Autoencoders

Deriving the log data likelihood

$$\begin{aligned}\log p_{\theta}(\mathbf{x}^{(i)}) &= \mathbf{E}_{\mathbf{z} \sim q_{\phi}(\mathbf{z}|\mathbf{x}^{(i)})} \left[\log p_{\theta}(\mathbf{x}^{(i)}) \right] \quad (p_{\theta}(\mathbf{x}^{(i)}) \text{ does not depend on } \mathbf{z}) \\&= \mathbf{E}_{\mathbf{z}} \left[\log \frac{p_{\theta}(\mathbf{x}^{(i)}|\mathbf{z}) p_{\theta}(\mathbf{z})}{p_{\theta}(\mathbf{z}|\mathbf{x}^{(i)})} \right] \quad (\text{Bayes' rule}) \\&= \mathbf{E}_{\mathbf{z}} \left[\log \frac{p_{\theta}(\mathbf{x}^{(i)}|\mathbf{z}) p_{\theta}(\mathbf{z})}{p_{\theta}(\mathbf{z}|\mathbf{x}^{(i)})} \frac{q_{\phi}(\mathbf{z}|\mathbf{x}^{(i)})}{q_{\phi}(\mathbf{z}|\mathbf{x}^{(i)})} \right] \quad (\text{Multiply with constant}) \\&= \mathbf{E}_{\mathbf{z}} \left[\log p_{\theta}(\mathbf{x}^{(i)}|\mathbf{z}) \right] - \mathbf{E}_{\mathbf{z}} \left[\log \frac{q_{\phi}(\mathbf{z}|\mathbf{x}^{(i)})}{p_{\theta}(\mathbf{z})} \right] + \mathbf{E}_{\mathbf{z}} \left[\log \frac{q_{\phi}(\mathbf{z}|\mathbf{x}^{(i)})}{p_{\theta}(\mathbf{z}|\mathbf{x}^{(i)})} \right] \quad (\text{Logarithms}) \\&= \mathbf{E}_{\mathbf{z}} \left[\log p_{\theta}(\mathbf{x}^{(i)}|\mathbf{z}) \right] - D_{\text{KL}}(q_{\phi}(\mathbf{z}|\mathbf{x}^{(i)}), p_{\theta}(\mathbf{z})) + D_{\text{KL}}(q_{\phi}(\mathbf{z}|\mathbf{x}^{(i)}), p_{\theta}(\mathbf{z}|\mathbf{x}^{(i)}))\end{aligned}$$

$$D_{\text{KL}}(q, p) = \sum_{\mathbf{z}} q(\mathbf{z}) \log \frac{q(\mathbf{z})}{p(\mathbf{z})} = \mathbf{E}_{\mathbf{z}} \left(\log \frac{q(\mathbf{z})}{p(\mathbf{z})} \right)$$

Expectation w.r.t. \mathbf{z} let's us write these KL terms

Variational Autoencoders

Deriving the log data likelihood

$$\begin{aligned}\log p_{\theta}(\mathbf{x}^{(i)}) &= \mathbf{E}_{\mathbf{z} \sim q_{\phi}(\mathbf{z}|\mathbf{x}^{(i)})} \left[\log p_{\theta}(\mathbf{x}^{(i)}) \right] \quad (p_{\theta}(\mathbf{x}^{(i)}) \text{ does not depend on } \mathbf{z}) \\ &= \mathbf{E}_{\mathbf{z}} \left[\log \frac{p_{\theta}(\mathbf{x}^{(i)}|\mathbf{z}) p_{\theta}(\mathbf{z})}{p_{\theta}(\mathbf{z}|\mathbf{x}^{(i)})} \right] \quad (\text{Bayes' rule}) \\ &= \mathbf{E}_{\mathbf{z}} \left[\log \frac{p_{\theta}(\mathbf{x}^{(i)}|\mathbf{z}) p_{\theta}(\mathbf{z})}{p_{\theta}(\mathbf{z}|\mathbf{x}^{(i)})} \frac{q_{\phi}(\mathbf{z}|\mathbf{x}^{(i)})}{q_{\phi}(\mathbf{z}|\mathbf{x}^{(i)})} \right] \quad (\text{Multiply with constant}) \\ &= \mathbf{E}_{\mathbf{z}} \left[\log p_{\theta}(\mathbf{x}^{(i)}|\mathbf{z}) \right] - \mathbf{E}_{\mathbf{z}} \left[\log \frac{q_{\phi}(\mathbf{z}|\mathbf{x}^{(i)})}{p_{\theta}(\mathbf{z})} \right] + \mathbf{E}_{\mathbf{z}} \left[\log \frac{q_{\phi}(\mathbf{z}|\mathbf{x}^{(i)})}{p_{\theta}(\mathbf{z}|\mathbf{x}^{(i)})} \right] \quad (\text{Logarithms}) \\ &= \mathbf{E}_{\mathbf{z}} \left[\log p_{\theta}(\mathbf{x}^{(i)}|\mathbf{z}) \right] - D_{\text{KL}}(q_{\phi}(\mathbf{z}|\mathbf{x}^{(i)}), p_{\theta}(\mathbf{z})) + D_{\text{KL}}(q_{\phi}(\mathbf{z}|\mathbf{x}^{(i)}), p_{\theta}(\mathbf{z}|\mathbf{x}^{(i)}))\end{aligned}$$

Decoder network, estimates this term through sampling. (Differentiable via re-parametrization, see paper)

KL between encoder and z-prior →
Two Gaussians, closed-form solution

$p(\mathbf{z}|\mathbf{x})$ intractable, this cannot be computed. But $\text{KL} \geq 0$

Variational Autoencoders

Deriving the log data likelihood

$$\log p_{\theta}(\mathbf{x}^{(i)}) = \underbrace{\mathbf{E}_{\mathbf{z}} \left[\log p_{\theta}(\mathbf{x}^{(i)} | \mathbf{z}) \right] - D_{\text{KL}}(q_{\phi}(\mathbf{z} | \mathbf{x}^{(i)}), p_{\theta}(\mathbf{z}))}_{\mathcal{L}(\mathbf{x}^{(i)}, \theta, \phi)} + \underbrace{D_{\text{KL}}(q_{\phi}(\mathbf{z} | \mathbf{x}^{(i)}), p_{\theta}(\mathbf{z} | \mathbf{x}^{(i)}))}_{\geq 0}$$

$$\log p_{\theta}(\mathbf{x}^{(i)}) \geq \mathcal{L}(\mathbf{x}^{(i)}, \theta, \phi)$$

Evidence Lower Bound (“ELBO”)

$$\theta^*, \phi^* = \arg \max_{\theta, \phi} \mathcal{L}(\mathbf{x}^{(i)}, \theta, \phi)$$

Training: Maximize lower bound

Variational Autoencoders

Deriving the log data likelihood

$$\log p_{\theta}(\mathbf{x}^{(i)}) = \underbrace{\mathbf{E}_{\mathbf{z}} \left[\log p_{\theta}(\mathbf{x}^{(i)} | \mathbf{z}) \right] - D_{\text{KL}}(q_{\phi}(\mathbf{z} | \mathbf{x}^{(i)}), p_{\theta}(\mathbf{z}))}_{\mathcal{L}(\mathbf{x}^{(i)}, \theta, \phi)} + \underbrace{D_{\text{KL}}(q_{\phi}(\mathbf{z} | \mathbf{x}^{(i)}), p_{\theta}(\mathbf{z} | \mathbf{x}^{(i)}))}_{\geq 0}$$

Reconstruct input data:
Make $p(\mathbf{x}|\mathbf{z})$ high!

Make approx. posterior close
to prior (Gaussian)
Make this KL small!

$\log p_{\theta}(\mathbf{x}^{(i)}) \geq \mathcal{L}(\mathbf{x}^{(i)}, \theta, \phi)$
Evidence Lower Bound (“ELBO”)

$\theta^*, \phi^* = \arg \max_{\theta, \phi} \mathcal{L}(\mathbf{x}^{(i)}, \theta, \phi)$
Training: Maximize lower bound

Variational Autoencoders

Maximizing ELBO $\underbrace{E_z \left[\log p_\theta(\mathbf{x}^{(i)}|z) \right] - D_{\text{KL}}(q_\phi(z|\mathbf{x}^{(i)}), p_\theta(z))}_{\mathcal{L}(\mathbf{x}^{(i)}, \theta, \phi)}$

→ Compute bound for a minibatch of input data

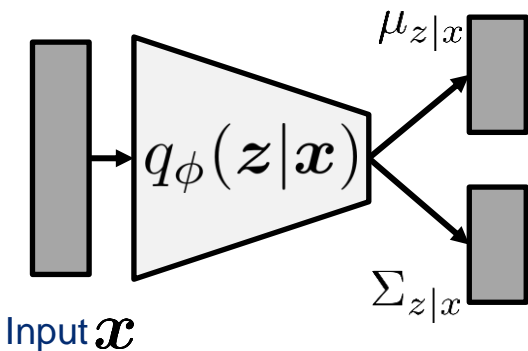


Input \mathbf{x}

Variational Autoencoders

Maximizing ELBO
$$\underbrace{E_z \left[\log p_\theta(\mathbf{x}^{(i)}|z) \right] - D_{\text{KL}}(q_\phi(z|\mathbf{x}^{(i)}), p_\theta(z))}_{\mathcal{L}(\mathbf{x}^{(i)}, \theta, \phi)}$$

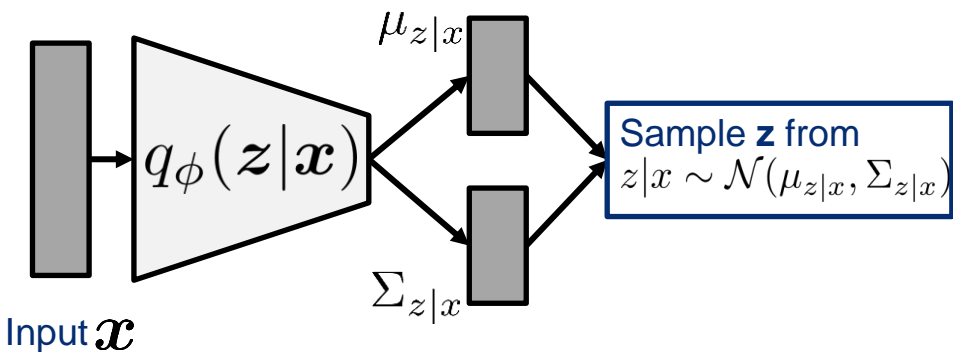
Approx. posterior close to prior



Variational Autoencoders

Maximizing ELBO
$$\underbrace{E_z \left[\log p_\theta(\mathbf{x}^{(i)}|z) \right] - D_{\text{KL}}(q_\phi(z|\mathbf{x}^{(i)}), p_\theta(z))}_{\mathcal{L}(\mathbf{x}^{(i)}, \theta, \phi)}$$

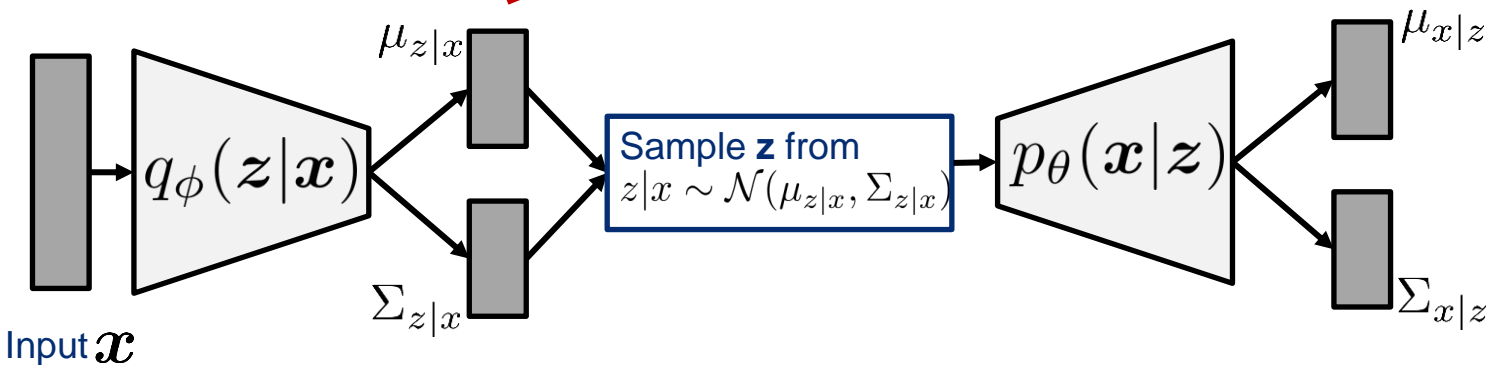
Approx. posterior close to prior



Variational Autoencoders

Maximizing ELBO
$$\underbrace{E_z \left[\log p_\theta(\mathbf{x}^{(i)}|z) \right] - D_{\text{KL}}(q_\phi(z|\mathbf{x}^{(i)}), p_\theta(z))}_{\mathcal{L}(\mathbf{x}^{(i)}, \theta, \phi)}$$

Approx. posterior close to prior

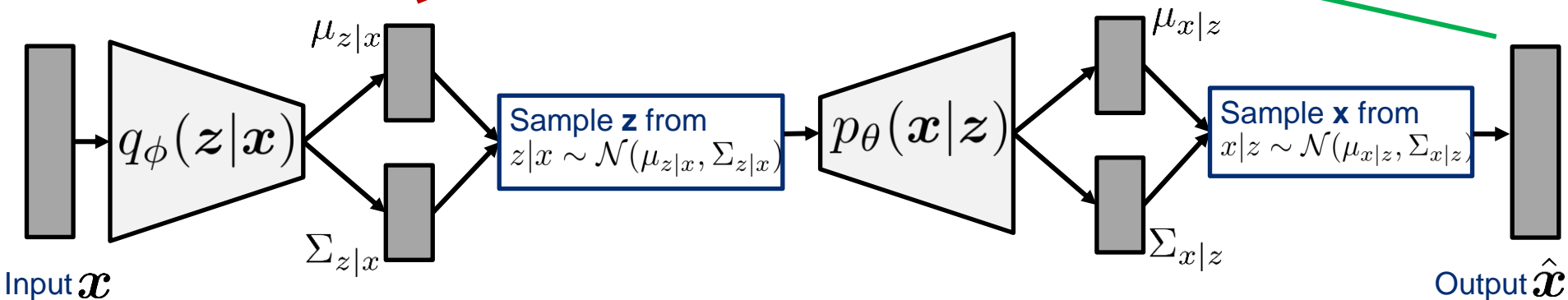


Variational Autoencoders

Maximizing ELBO
$$\underbrace{E_z \left[\log p_\theta(\mathbf{x}^{(i)}|z) \right] - D_{\text{KL}}(q_\phi(z|\mathbf{x}^{(i)}), p_\theta(z))}_{\mathcal{L}(\mathbf{x}^{(i)}; \theta, \phi)}$$

Approx. posterior close to prior

Max. likelihood of input being reconstructed!

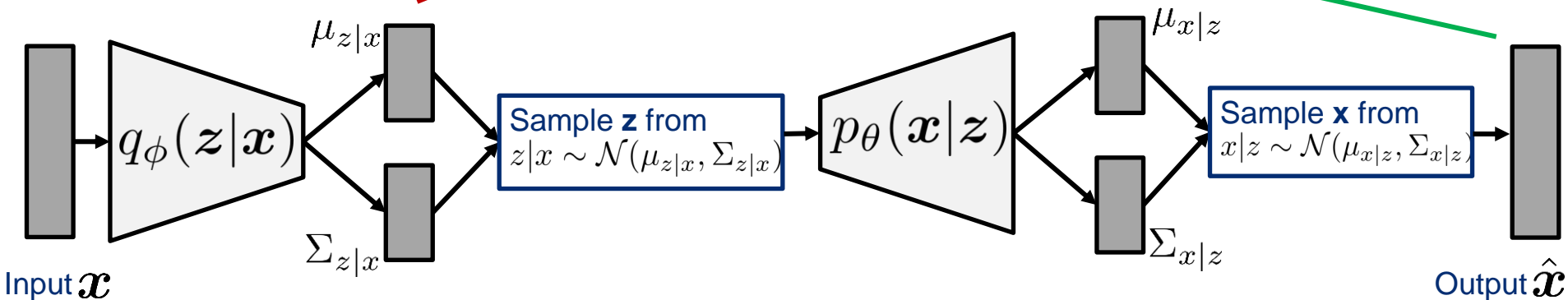


Variational Autoencoders

Maximizing ELBO
$$\underbrace{E_z \left[\log p_\theta(\mathbf{x}^{(i)}|z) \right] - D_{\text{KL}}(q_\phi(z|\mathbf{x}^{(i)}), p_\theta(z))}_{\mathcal{L}(\mathbf{x}^{(i)}; \theta, \phi)}$$

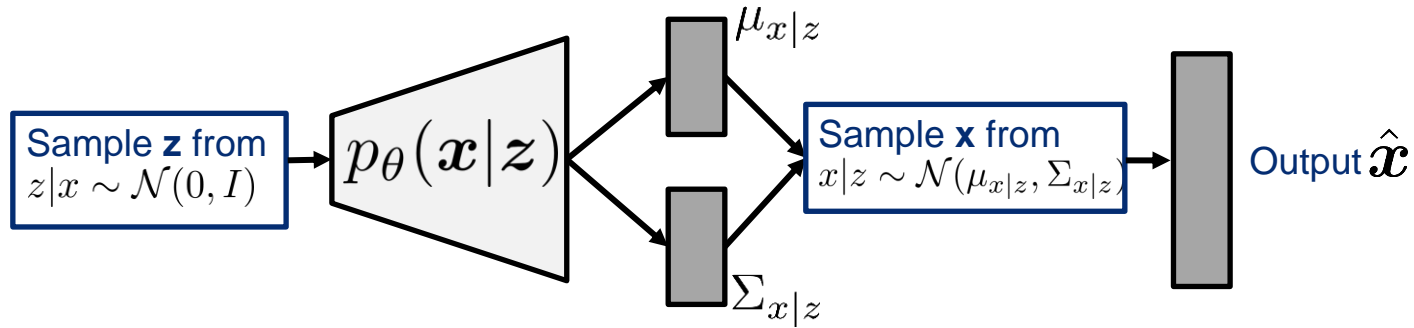
Approx. posterior close to prior

Max. likelihood of input being reconstructed!

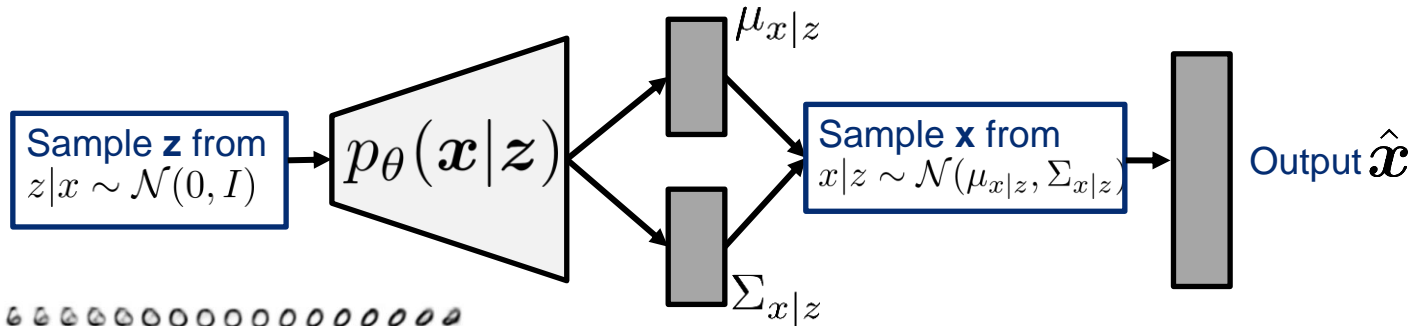


→ For every minibatch, compute forward pass, then back-prop!

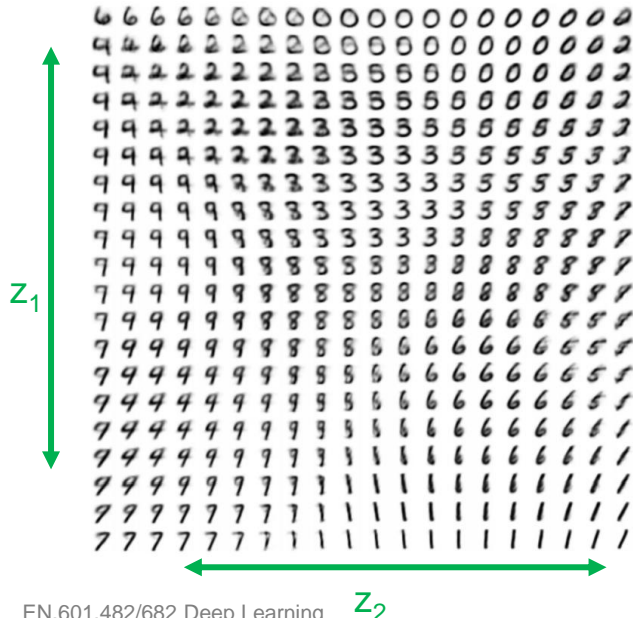
Variational Autoencoders: Generating samples



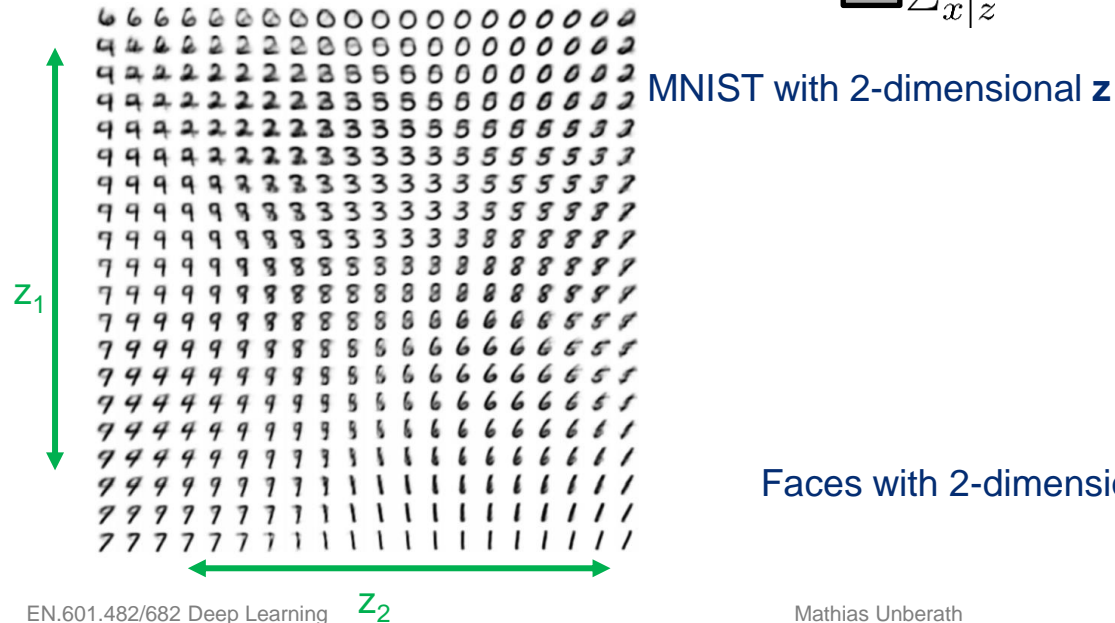
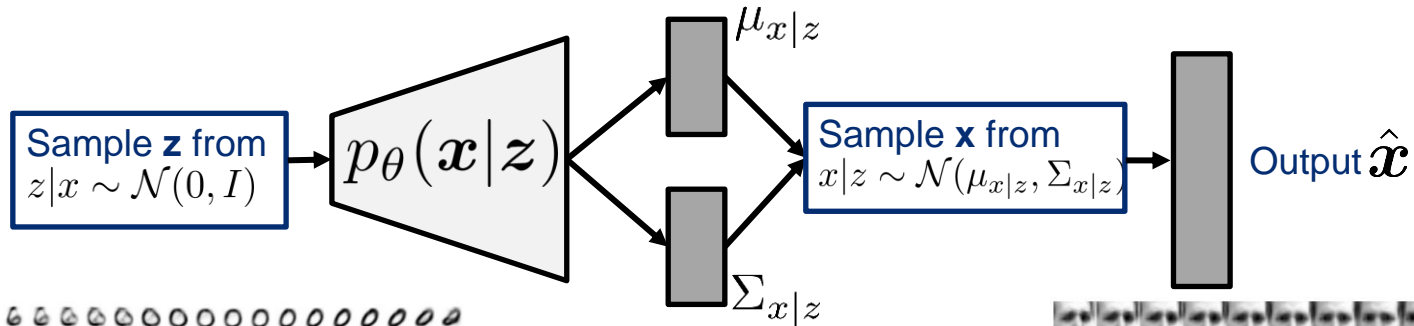
Variational Autoencoders: Generating samples



MNIST with 2-dimensional \mathbf{z}



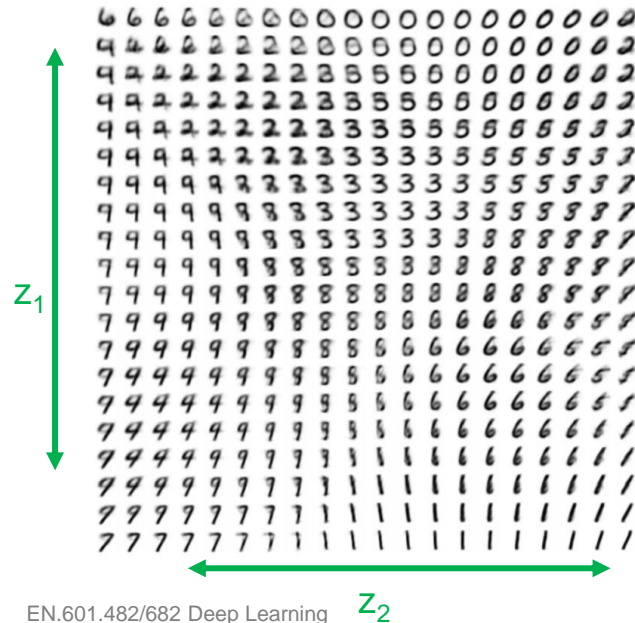
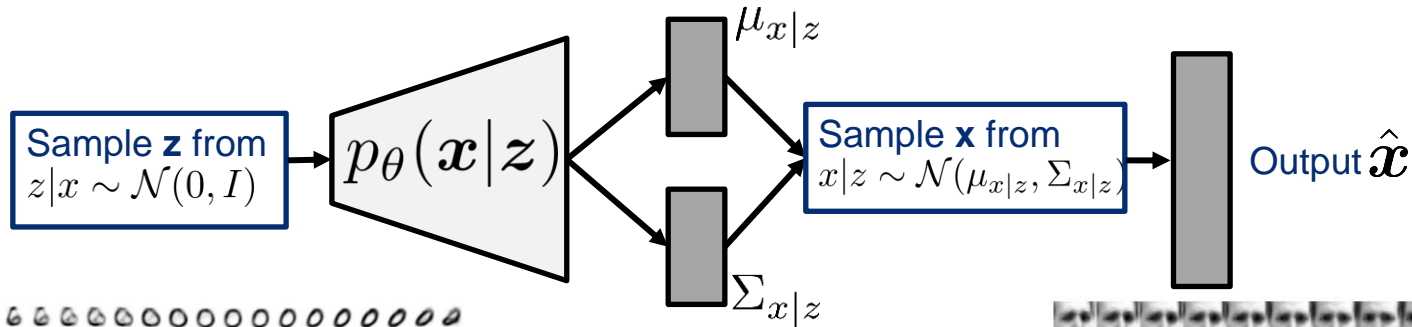
Variational Autoencoders: Generating samples



Faces with 2-dimensional \mathbf{z}



Variational Autoencoders: Generating samples

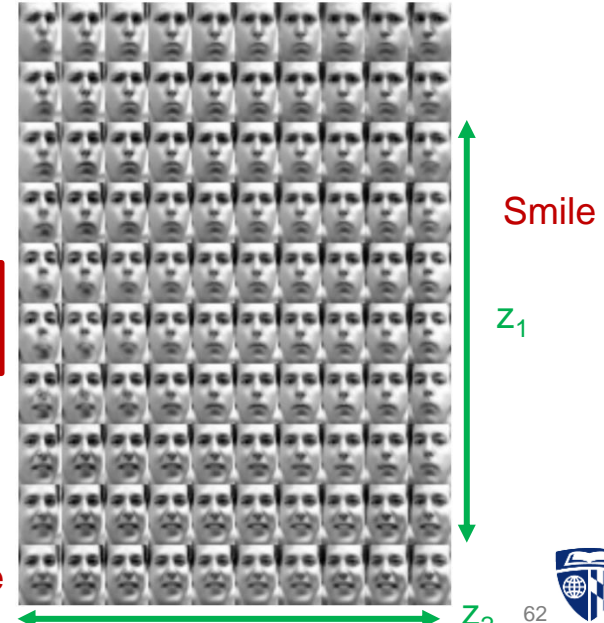


MNIST with 2-dimensional z

Diagonal prior on z
 → Independent latent variables

Faces with 2-dimensional z

Head pose



Embedding

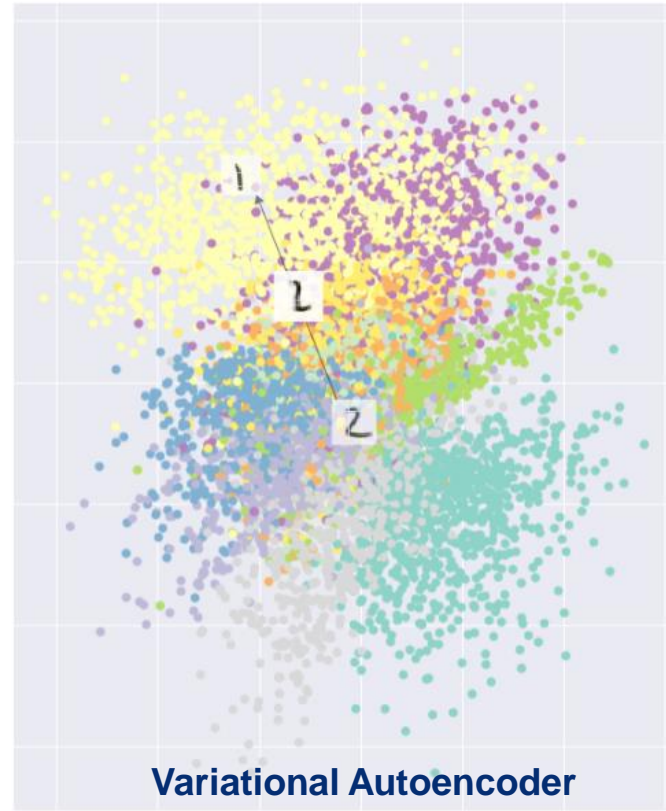
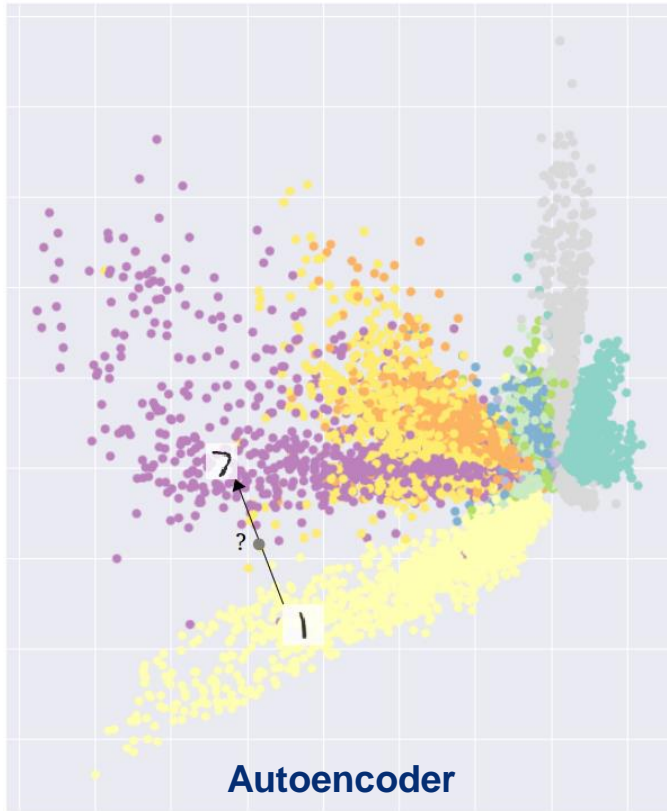


Image from [this blog post](#).

Representative Results



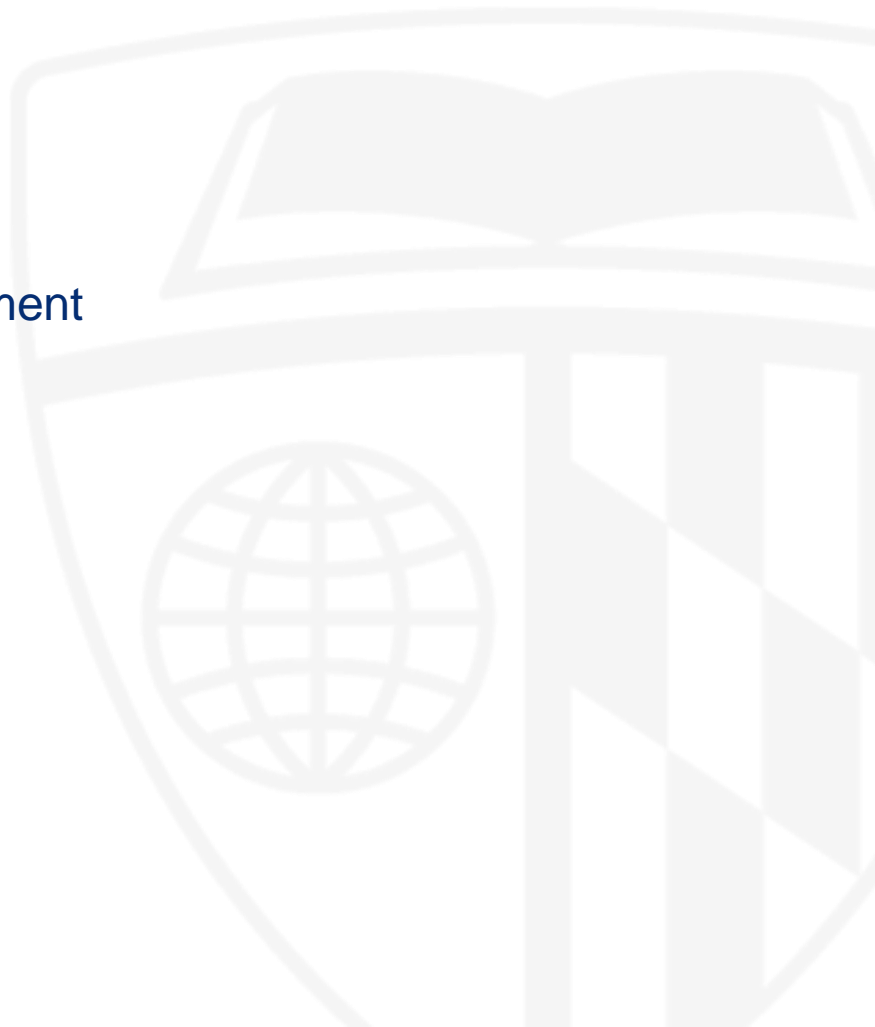
VAE on celebA. Taken from [github](#).

Results tend to be unsharp/blurry
→ GANs (next lecture) have taken over

Allows inference of $q(z|x)$ (encoder)
→ Can be useful for other tasks

(Variational) Autoencoders and Disentanglement

Disentanglement



Background

- VAE: Diagonal prior on $z \rightarrow$ Independent latent variable
- This independence is known as disentanglement
- Why would we want disentangled representations?
 - Generally considered to contain interpretable, semantic information
 - Reflect separate factors of variation in the data
 - Seem to be more robust against “adversarial attacks”
- Difficult to learn such representations without supervision

[Chen, T. Q., Li, X., Grosse, R. B., & Duvenaud, D. K. \(2018\). Isolating sources of disentanglement in variational autoencoders. NeurIPS.](#)

Background

- From VAE to β -VAE

$$\mathcal{L}(\mathbf{x}^{(i)}, \theta, \phi) = \mathbf{E}_{\mathbf{z}} \left[\log p_{\theta}(\mathbf{x}^{(i)} | \mathbf{z}) \right] - \boxed{\beta} D_{\text{KL}}(q_{\phi}(\mathbf{z} | \mathbf{x}^{(i)}), p_{\theta}(\mathbf{z}))$$

- $\beta > 1$ heavily penalizes the representation and enforces disentanglement, but not made explicit why this works

Background

- From VAE to β -VAE

$$\mathcal{L}(\mathbf{x}^{(i)}, \theta, \phi) = \mathbf{E}_{\mathbf{z}} \left[\log p_{\theta}(\mathbf{x}^{(i)} | \mathbf{z}) \right] - \boxed{\beta} D_{\text{KL}}(q_{\phi}(\mathbf{z} | \mathbf{x}^{(i)}), p_{\theta}(\mathbf{z}))$$

- $\beta > 1$ heavily penalizes the representation and enforces disentanglement, but not made explicit why this works
- β -TCVAE
Two quantities are especially important for disentangled representation
 - Mutual information between latent variables and data variable
 - Independence between latent variable
- ELBO can be decomposed to reveal the above components!

[Chen, T. Q., Li, X., Grosse, R. B., & Duvenaud, D. K. \(2018\). Isolating sources of disentanglement in variational autoencoders. NeurIPS.](#)

β -TCVAE

$$\mathbb{E}_{p(n)} \left[\text{KL}(q(z|n) || p(z)) \right] = \underbrace{\text{KL}(q(z, n) || q(z)p(n))}_{\text{(i) Index-Code MI}} + \underbrace{\text{KL}(q(z) || \prod_j q(z_j))}_{\text{(ii) Total Correlation}} + \underbrace{\sum_j \text{KL}(q(z_j) || p(z_j))}_{\text{(iii) Dimension-wise KL}}$$

I. Index-code mutual information (MI)

Enforces MI between data variable and latent variable

II. Total correlation (TC)

Generalization of MI to multiple variables (actually measures dependence)

Penalizing this term forces model to find statistically independent factors

→ Existence of this term is likely the reason for success of β -VAE

III. Dimension-wise KL

Prevents latent representations from deviating too much from prior

[Chen, T. Q., Li, X., Grosse, R. B., & Duvenaud, D. K. \(2018\). Isolating sources of disentanglement in variational autoencoders. NeurIPS.](#)

β -TCVAE

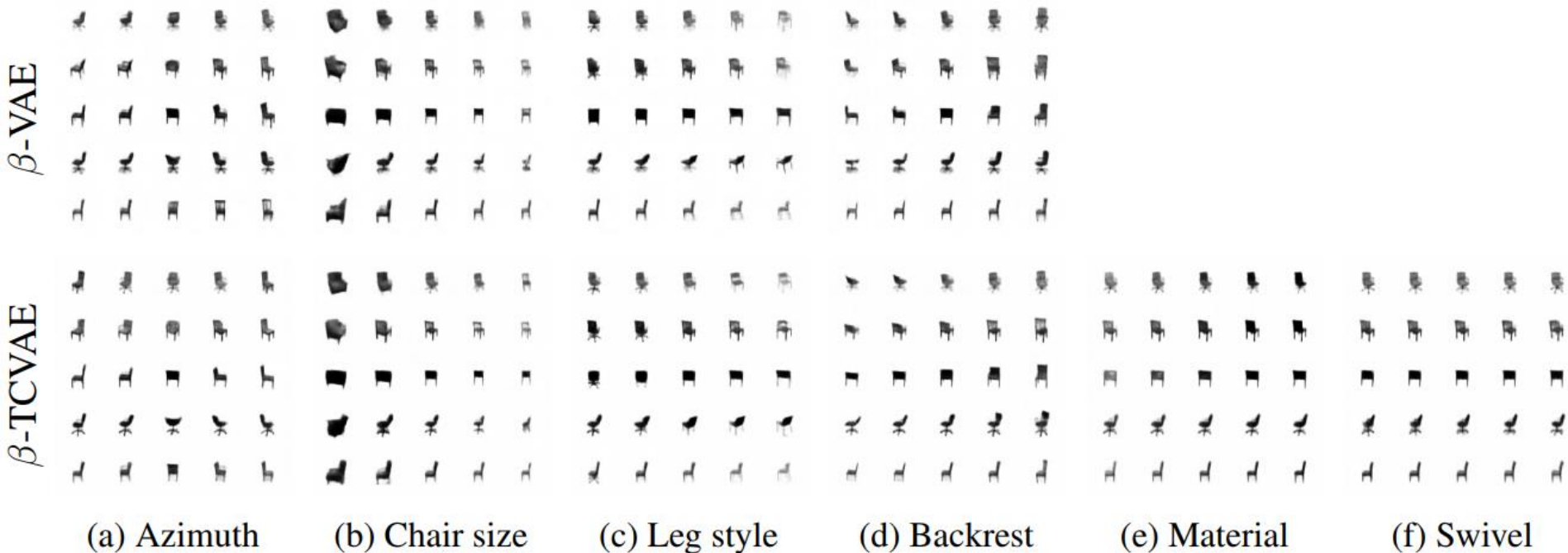


Figure 6: Learned latent variables using β -VAE and β -TCVAE are shown. Traversal range is $(-2, 2)$.

β -TCVAE

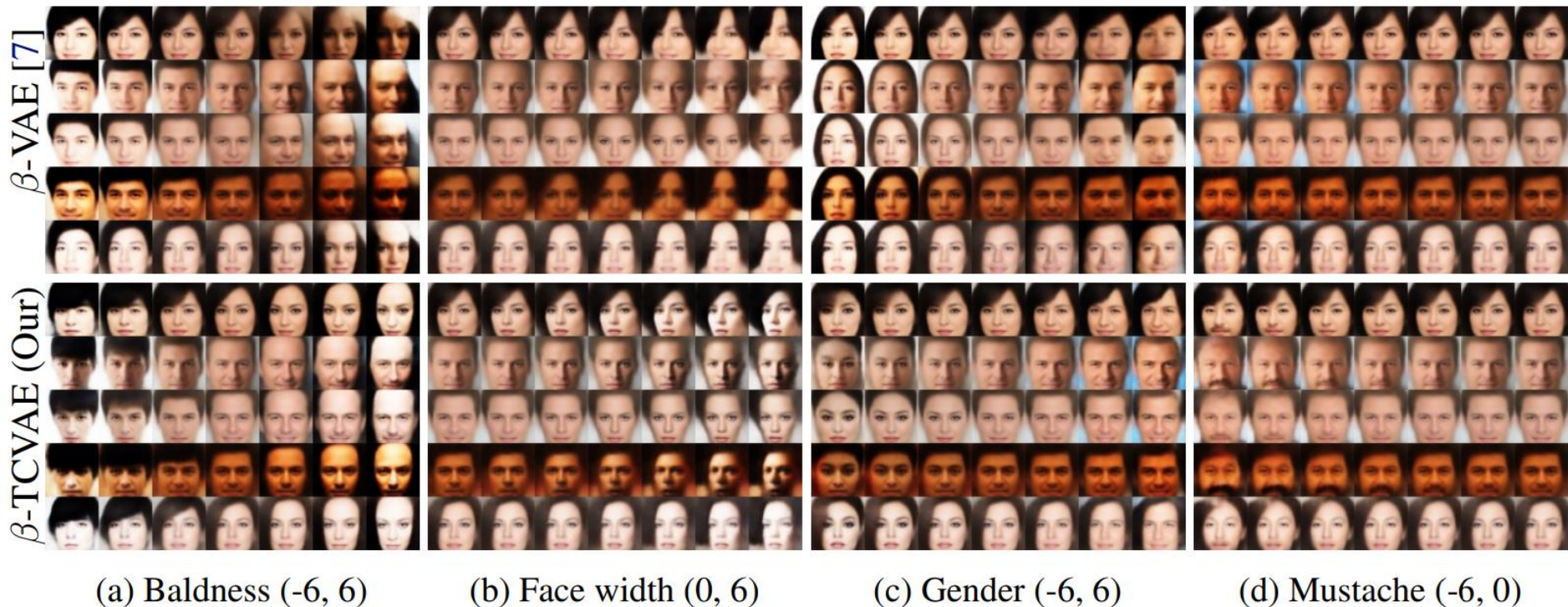


Figure 1: Qualitative comparisons on CelebA. Traversal ranges are shown in parentheses. Some attributes are only manifested in one direction of a latent variable, so we show a one-sided traversal. Most semantically similar variables from a β -VAE are shown for comparison.

(Variational) Autoencoders and Disentanglement

Questions?

