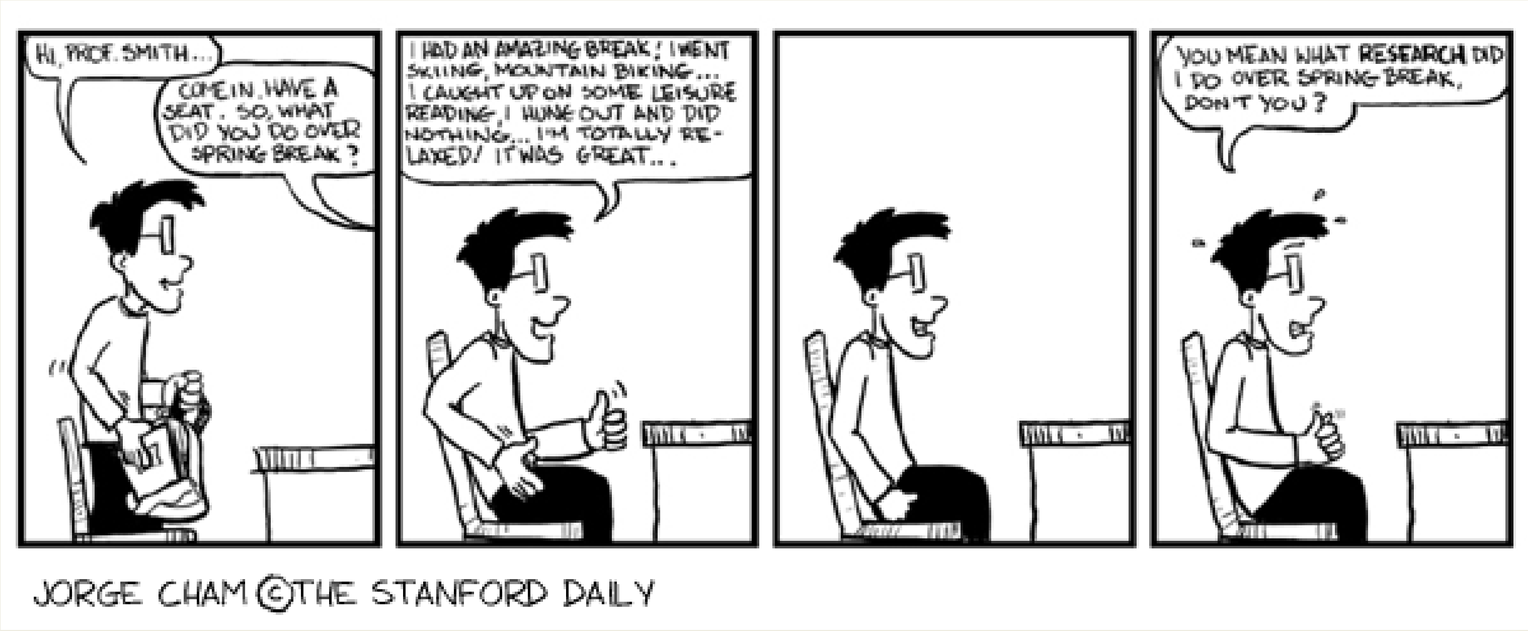
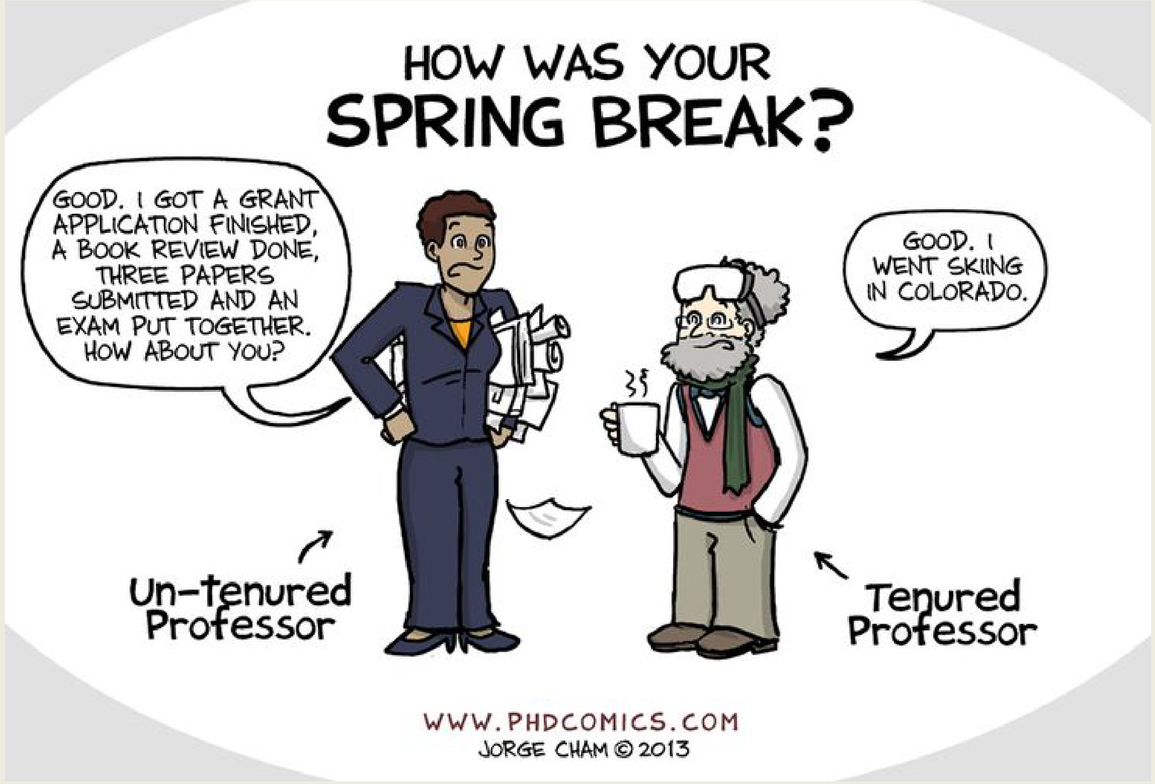
Some Reminders: Spring Break !(?)

- What is Spring Break?
  - Undergrads have fun (take a trip, visit home, etc.)
  - **Ph.D. students have fun (stop worrying about classes/grades and work on research ☺)**
  - Faculty have fun (hide from the world, writing retreat, submit grant proposals)



Some Reminders: Spring Break !(?)

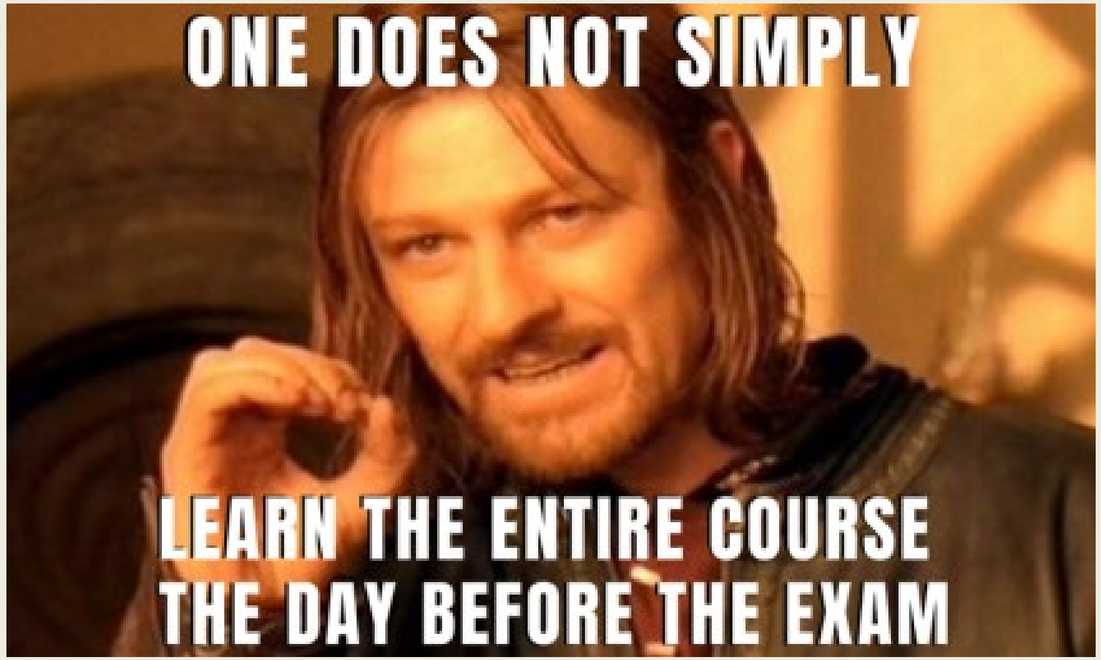
- What is Spring Break?
  - Undergrads have fun (take a trip, visit home, etc.)
  - Ph.D. students have fun (stop worrying about classes/grades and work on research ☺)
  - **Faculty have fun (hide from the world, writing retreat, submit grant proposals)**



Some Reminders: Spring Break !(?)

- What is Spring Break?
  - Undergrads have fun (take a trip, visit home, etc.)
  - Ph.D. students have fun (stop worrying about classes/grades and work on research ☺)
  - Faculty have fun (hide from the world, writing retreat, submit grant proposals)

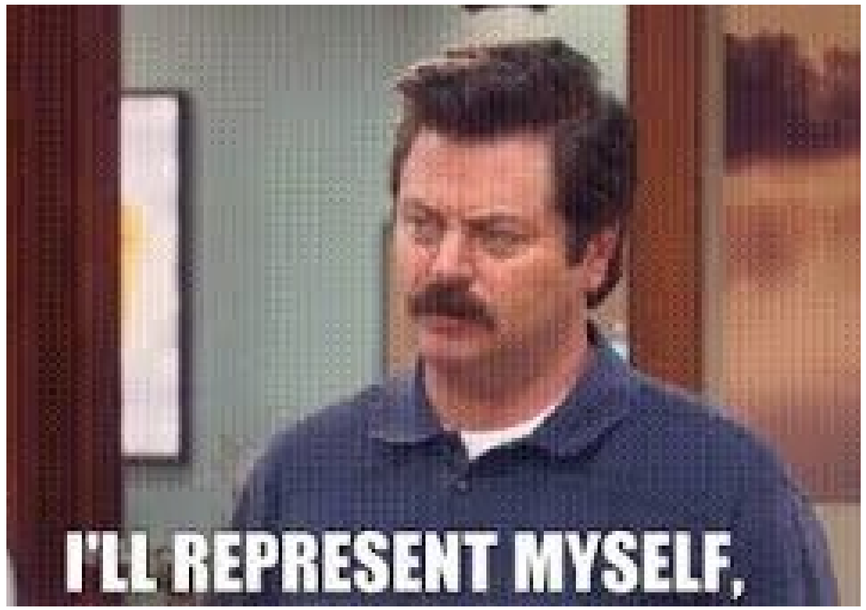
- Midterm Exam is on March 31
  - In this room (ITE 231)
  - During class hours (4:00 PM to 5:00 PM)
  - Syllabus: every lecture until and including 03/26



tejasgokhale.com

CMSC 475/675 Neural Networks

Lecture 7: Representation Learning



UMBC

Some VAE slides are borrowed from Ranjay Krishna

Machine Learning Problems

SUPERVISED LEARNING

- Training time
  - data :

$$\{\mathbf{x}^{(t)}, y^{(t)}\}$$

  - setting :

$$\mathbf{x}^{(t)}, y^{(t)} \dashleftarrow p(\mathbf{x}, y)$$

- Test time
  - data :

$$\{\mathbf{x}^{(t)}, y^{(t)}\}$$

  - setting :

$$\mathbf{x}^{(t)}, y^{(t)} \dashleftarrow p(\mathbf{x}, y)$$

Example

$Input$: $x^{(t)}$ is an image

$Output$: $y^{(t)}$ is an image category

MULTITASK LEARNING

- Training time
  - data :

$$\{\mathbf{x}^{(t)}, y_1^{(t)}, \ldots, y_M^{(t)}\}$$

  - setting :

$$\mathbf{x}^{(t)}, y_1^{(t)}, \ldots, y_M^{(t)} \dashleftarrow p(\mathbf{x}, y_1, \ldots, y_M)$$

- Test time
  - data :

$$\{\mathbf{x}^{(t)}, y_1^{(t)}, \ldots, y_M^{(t)}\}$$

  - setting :

$$\mathbf{x}^{(t)}, y_1^{(t)}, \ldots, y_M^{(t)} \dashleftarrow p(\mathbf{x}, y_1, \ldots, y_M)$$

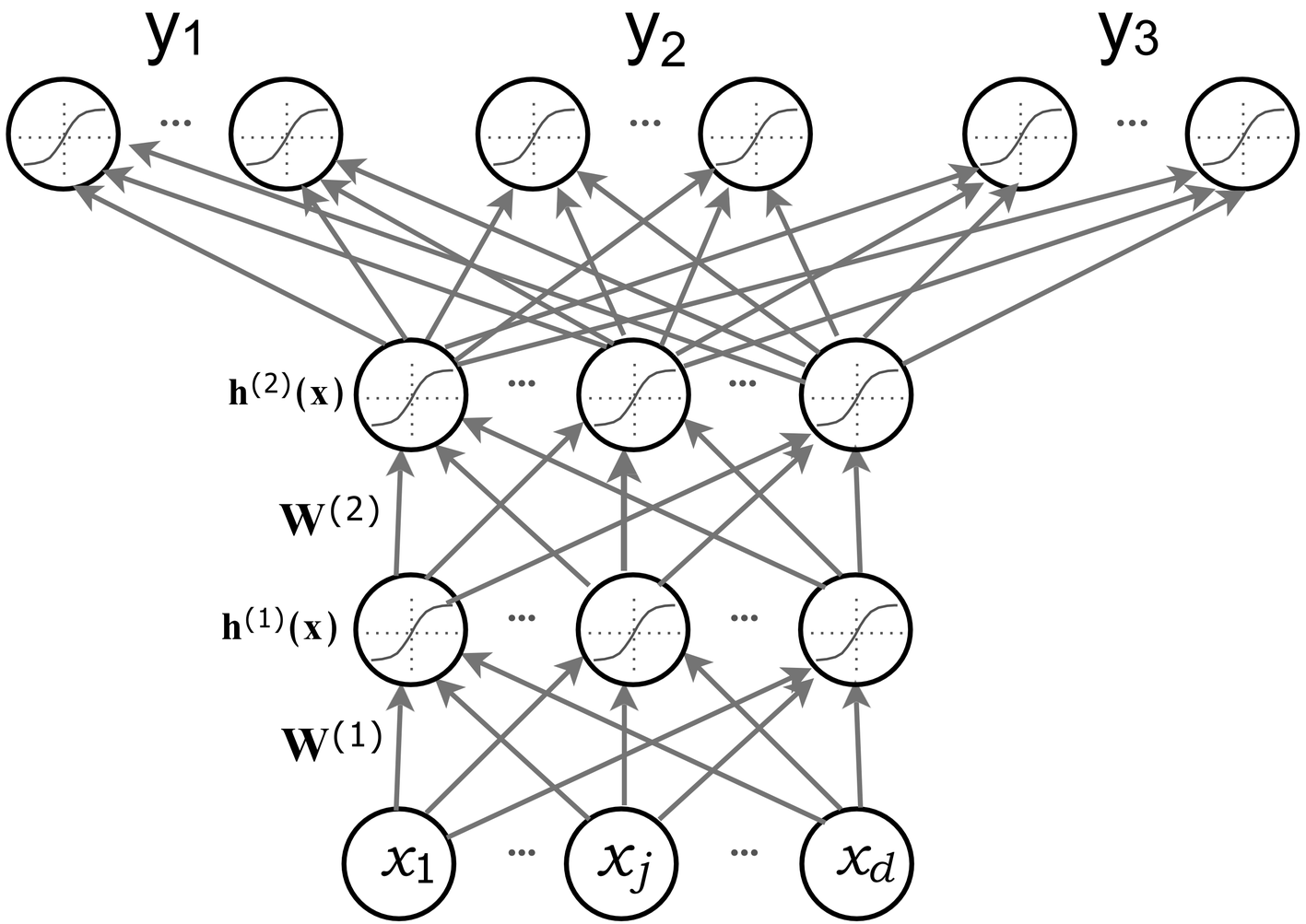
Example

$Input$: $x^{(t)}$ is an image
$Outputs$:

- $y_1^{(t)}$: image category
- $y_1^{(t)}$: object detection
- $y_1^{(t)}$: depth estimation
- $y_1^{(t)}$: semantic segmentation
- …

MULTITASK LEARNING

Topics: multitask learning



DOMAIN ADAPTATION

- Training time
  - data :

$$\{\mathbf{x}^{(t)}, y^{(t)}\}$$

$$\{\overline{\mathbf{x}}^{(t^o)}\}$$

  - setting :

$$\mathbf{x}^{(t)} \dashleftarrow p(\mathbf{x})$$

$$y^{(t)} \dashleftarrow p(y|\mathbf{x}^{(t)})$$

$$\overline{\mathbf{x}}^{(t)} \dashleftarrow q(\mathbf{x}) \nuparrow p(\mathbf{x})$$

- Test time
  - data :

$$\{\overline{\mathbf{x}}^{(t)}, y^{(t)}\}$$

  - setting :

$$\overline{\mathbf{x}}^{(t)} \dashleftarrow q(\mathbf{x})$$

$$y^{(t)} \dashleftarrow p(y|\overline{\mathbf{x}}^{(t)})$$

- Example
  - classify sentiment (positive vs negative)
  - in reviews of different products
  - training on Amazon Reviews but testing on Yelp Reviews

ONE-SHOT LEARNING

- Training time
  - data :

$$\{\mathbf{x}^{(t)}, y^{(t)}\}$$

  - setting :

$$\mathbf{x}^{(t)}, y^{(t)} \dashleftarrow p(\mathbf{x}, y)$$

subject to $y^{(t)} \in \{1, \ldots, C\}$

- Test time
  - data :

$$\{\mathbf{x}^{(t)}, y^{(t)}\}$$

  - setting :

$$\mathbf{x}^{(t)}, y^{(t)} \dashleftarrow p(\mathbf{x}, y)$$

subject to $y^{(t)} \in \{C + 1, \ldots, C + M\}$

  - side information :
    - a single labeled example from each of the M new classes

- Example
  - recognizing a person based on a single picture of him/her

ZERO-SHOT LEARNING

- Training time
  - data :

$$\{\mathbf{x}^{(t)}, y^{(t)}\}$$

  - setting :

$$\mathbf{x}^{(t)}, y^{(t)} \dashleftarrow p(\mathbf{x}, y)$$

    subject to $y^{(t)} \in \{1, \ldots, C\}$
  - side information :
    - description vector z_c of each of the C classes

- Test time
  - data :

$$\{\mathbf{x}^{(t)}, y^{(t)}\}$$

  - setting :

$$\mathbf{x}^{(t)}, y^{(t)} \dashleftarrow p(\mathbf{x}, y)$$

    subject to $y^{(t)} \in \{C+1, \ldots, C+M\}$
  - side information :
    - description vector z_c of each of the new M classes

- Example
  - recognizing an object based on a worded description of it

SEMI-SUPERVISED LEARNING

- Training time
  - data :

$$\{\mathbf{x}^{(t)}, y^{(t)}\}$$

$$\{\mathbf{x}^{(t)}\}$$

  - setting :

$$\mathbf{x}^{(t)}, y^{(t)} \dashleftarrow p(\mathbf{x}, y)$$

$$\mathbf{x}^{(t)} \dashleftarrow p(\mathbf{x})$$

- Test time
  - data :

$$\{\mathbf{x}^{(t)}, y^{(t)}\}$$

  - setting :

$$\mathbf{x}^{(t)}, y^{(t)} \dashleftarrow p(\mathbf{x}, y)$$

UNSUPERVISED LEARNING

- Training time
  - data :

$$\{\mathbf{x}^{(t)}\}$$

  - setting :

$$\mathbf{x}^{(t)} \dashleftarrow p(\mathbf{x})$$

- Test time
  - data :

$$\{\mathbf{x}^{(t)}\}$$

  - setting :

$$\mathbf{x}^{(t)} \dashleftarrow p(\mathbf{x})$$

How can we train models “unsupervised”?

How can we train models “unsupervised”?

This is the focus of representation learning

This is the focus of representation learning

There’s an entire co

ICLR

ICLR has become one of the top CS and Engineering (not just AI) publication although it just started in 2013.

In fact top-10 in ALL OF SCIENCE

Top publications

Categories ▾ English ▾

	Publication	h5-index	h5-median
1.	Nature	488	745
2.	IEEE/CVF Conference on Computer Vision and Pattern Recognition	440	689
3.	The New England Journal of Medicine	434	897
4.	Science	409	633
5.	Nature Communications	375	492
6.	The Lancet	368	678
7.	Neural Information Processing Systems	337	614
8.	Advanced Materials	327	420
9.	Cell	320	482
10.	International Conference on Learning Representations	304	584

How

This is th

There's an entire

ICLR

ICLR has become one of the top CS and Engineering (not just AI) publication although it just started in 2013.

In fact top-10 in ALL OF SCIENCE

International Conference on Learning Representations

h5-index:304 h5-median:584

#2 Artificial Intelligence
#4 Engineering & Computer Science

Title / Author	Cited by
An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. A Dosovitskiy, L Beyer, A Kolesnikov, D Weissenborn, X Zhai, ... ICLR	38519
Decoupled Weight Decay Regularization. I Loshchilov, F Hutter ICLR (Poster)	18046
Measuring and Improving the Use of Graph Information in Graph Neural Networks. Y Hou, J Zhang, J Cheng, K Ma, RTB Ma, H Chen, MC Yang ICLR	7849
ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. Z Lan, M Chen, S Goodman, K Gimpel, P Sharma, R Soricut ICLR	7127
Large Scale GAN Training for High Fidelity Natural Image Synthesis. A Brock, J Donahue, K Simonyan ICLR	5675
DARTS: Differentiable Architecture Search. H Liu, K Simonyan, Y Yang ICLR (Poster)	4888
LoRA: Low-Rank Adaptation of Large Language Models. EJ Hu, Y Shen, P Wallis, Z Allen-Zhu, Y Li, S Wang, L Wang, W Chen ICLR	4881
Deformable DETR: Deformable Transformers for End-to-End Object Detection. X Zhu, W Su, L Lu, B Li, X Wang, J Dai ICLR	4444
BERTScore: Evaluating Text Generation with BERT. T Zhang, V Kishore, F Wu, KQ Weinberger, Y Artzi ICLR	4143
ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. K Clark, MT Luong, QV Le, CD Manning ICLR	3903

English

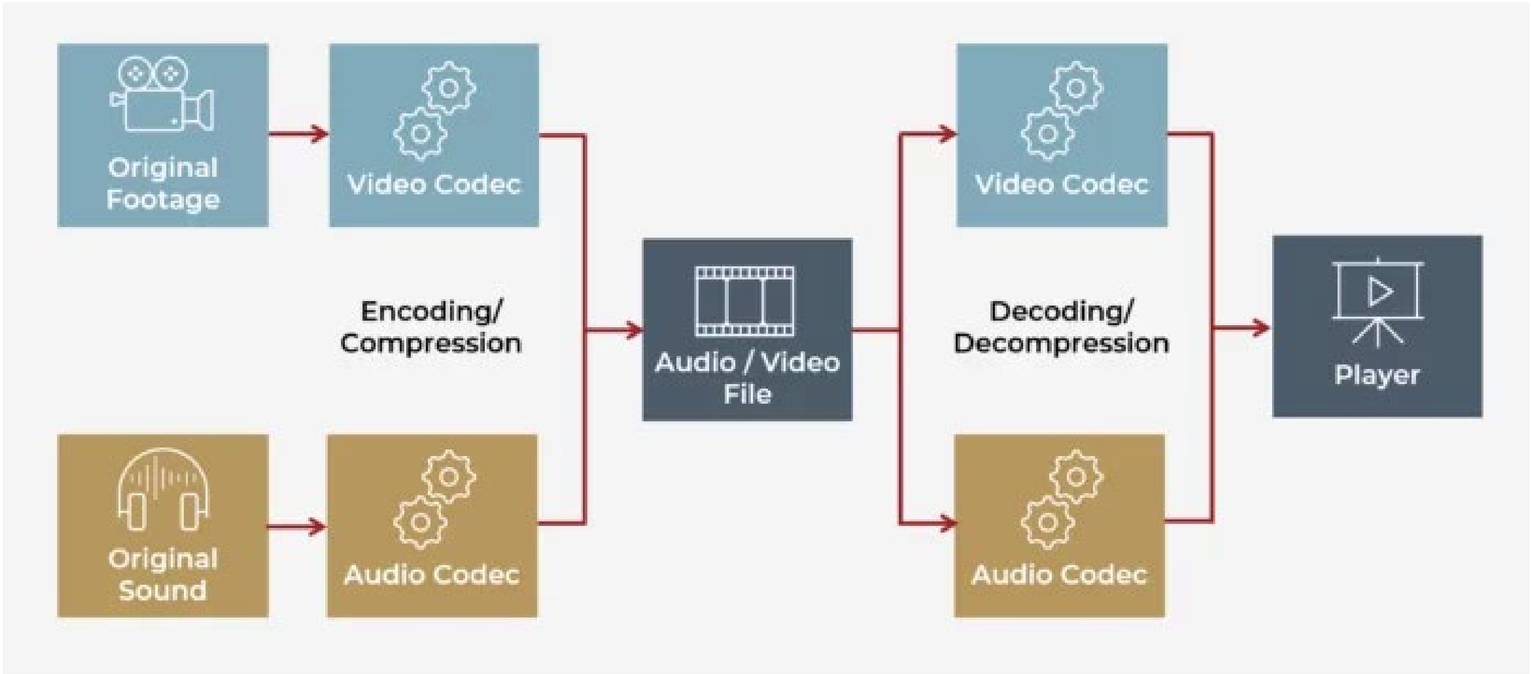
ndex	h5-median
38	745
40	689
34	897
09	633
75	492
68	678
37	614
27	420
20	482
04	584

Warning

I might use the terms “latent”, “embedding”, “representation”, “feature” interchangeably.

Motivation (kind of): Compression

- The idea is similar to compression (signal processing) or hashing (data structures):
  - **encode** an image into a smaller vector s.t. you can **decode** it back to its original form
    - Example: images, audio, video are stored in a compressed form on your computer using compression algorithms like JPEG, MP3, MPEG etc. The computer has software to decode it back so that you can view it (everytime you "open" a JPEG file to view an image, the decoder runs and converts code to RGB)



Motivation (kind of): Compression

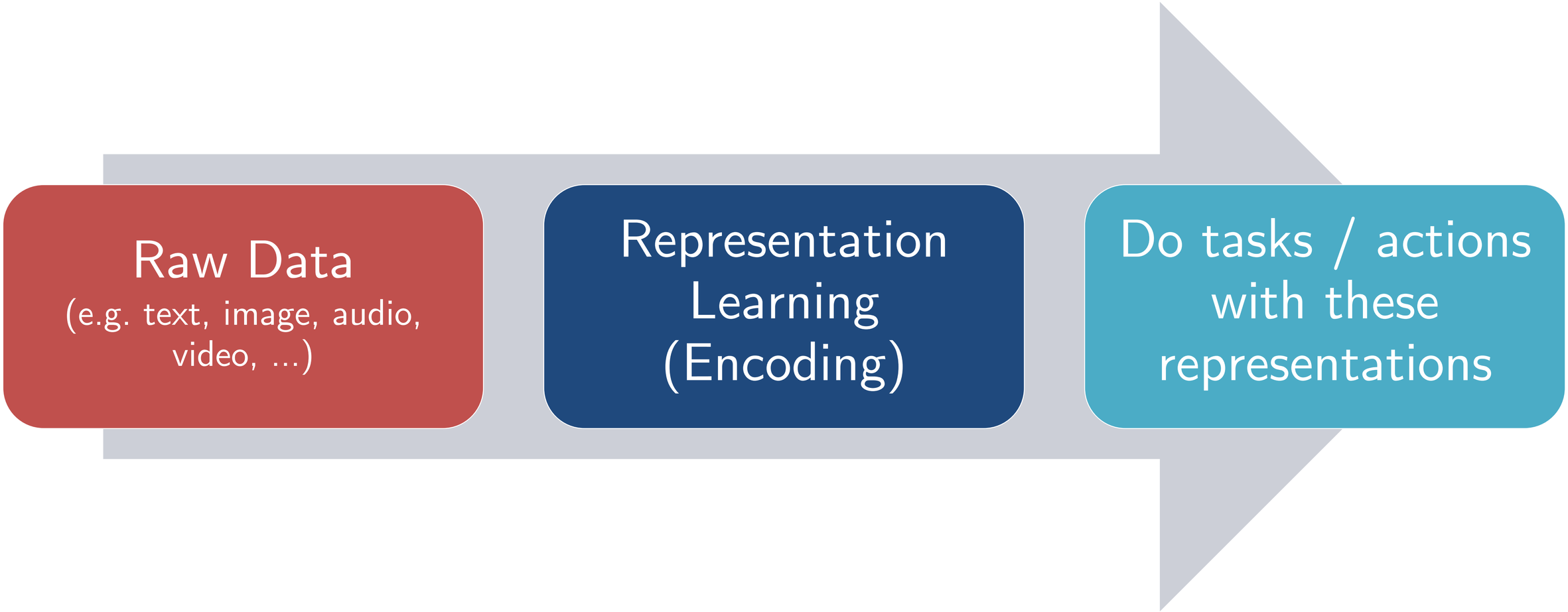
- The idea is similar to compression (signal processing) or hashing (information theory):
  - encode an image into a smaller vector s.t. you can **decode** it back to its original form
    - Example: images, audio, video are stored in a compressed form on your computer using compression algorithms like JPEG, MP3, MPEG etc. The computer has software to decode it back so that you can view it (everytime you "open" a JPEG file to view an image, the decoder runs and converts code to RGB)

- **Representation Learning**
  ~ convert inputs automatically into "codes" (called representations/embeddings / features) s.t. the representations are:
  - Useful for downstream tasks (e.g. classification, regression, ...)
  - "explain the data" and are "meaningful"

- Main difference: "meaningful" representation spaces to do "tasks"

  (The goal for compression is only efficient storage — not data classification/clustering etc.)

Representation Learning Paradigm



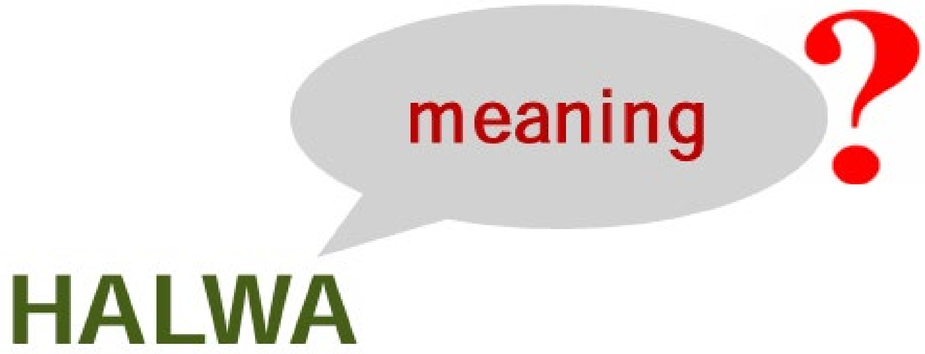
Typical Goals for Representations

Similar representations for similar concepts

Source: Pankaj Gupta (LMU Munich)

Typical Goals for Representations

A Semblance of "Context" should be encoded ...



If you know the answer,
don't share it with the class yet.

People from lands between Greece and India
might know the answer ...

Source: Pankaj Gupta (LMU Munich)

Typical Goals for Representations

A Semblance of "Context" should be encoded ...

Source: Pankaj Gupta (LMU Munich)

Typical Goals for Representations

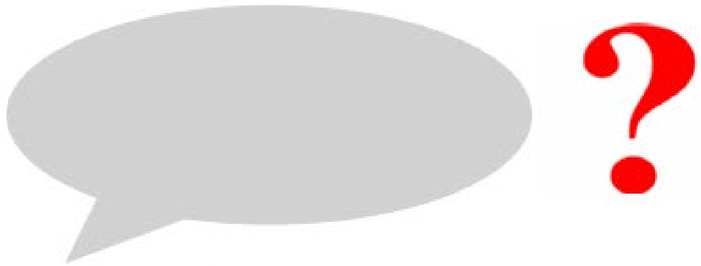
A Semblance of "Context" should be encoded ...

If you speak Marathi, this word has two meanings depending on context

Halwa (1): a food item — *derived from: Farsi*
Halwa (2): (an instruction to) move (something) — *derived from: Sanskrit*

Typical Goals for Representations

A Semblance of "Context" should be encoded ...

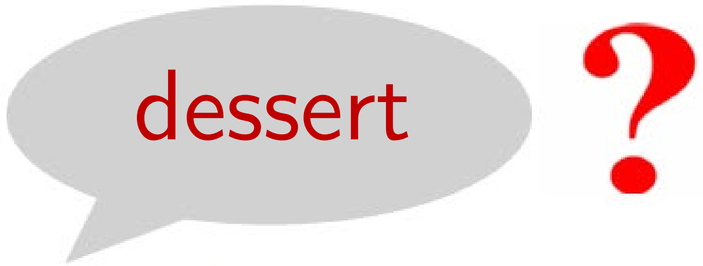


Is it a good idea to *eat* HALWA after a *meal* ?

Source: Pankaj Gupta (LMU Munich)

Typical Goals for Representations

A Semblance of "Context" should be encoded ...



Is it a good idea to *eat* HALWA after a *meal* ?

Source: Pankaj Gupta (LMU Munich)

Typical Goals for Representations

A Semblance of "Context" should be encoded ...

Oh no! I forgot to put sugar in the HALWA

Source: Pankaj Gupta (LMU Munich)

Typical Goals for Representations

Parts, properties, attributes, ontology ?

- “bird” has “wing”, “beak”, “feathers”
- “bird” can “fly”
- “bird” is under category “animal”
- “bird” has subcategories “eagle”, “peacock”, “sparrow”, “seagull”, “pigeon”

Source: Pankaj Gupta (LMU Munich)

Representation Learning is a Philosophy for Learning

Key assumptions in this philosophy:

- You can convert a high-dimensional input space into a low-dimensional representation space
  - Example: RGB images → 100 dim vectors
- A good representation space will have a "structure"
  - Example: Similarity, Symmetry, Relations will be easy to understand
  - Why? So that we can do arithmetic in representation space to do tasks
- Representations can be learned from data
- Representations can be leveraged for doing tasks

Parallel Work in Cog.Sci.

Trends in Cognitive Sciences
CellPress
Volume 28, Issue 9, September 2024, Pages 844-856

Review

Why concepts are (probably) vectors

Steven T. Piantadosi [1,2], Dyana C.Y. Muller [2], Joshua S. Rule [1], Karthikeya Kaushik [1], Mark Gorenstein [2], Elena R. Leib [1], Emily Sanford [1]

For decades, cognitive scientists have debated what kind of representation might characterize human concepts. Whatever the format of the representation, it must allow for the computation of varied properties, including similarities, features, categories, definitions, and relations. It must also support the development of theories, *ad hoc* categories, and knowledge of procedures. Here, we discuss why vector-based representations provide a compelling account that can meet all these needs while being plausibly encoded into neural architectures. This view has become especially promising with recent advances in both large language models and vector symbolic architectures. These innovations show how vectors can handle many properties traditionally thought to be out of reach for neural models, including compositionality, definitions, structures, and symbolic computational processes.

Highlights

Modern language models and vector-symbolic architectures show that vector-based models are capable of handling the compositional, structured, and symbolic properties required for human concepts.

Vectors are also able to handle key phenomena from the psychology, including computation of features and similarities, reasoning about relations and analogies, and representation of theories.

Language models show how vector representation of word semantics and sentences can interface between concepts and language, as seen in definitional theories of concepts or *ad hoc* concepts.

The idea of Church encoding, from logic, allows us to understand how meaning can arise in vector-based or symbolic systems.

By combining these recent computational results with classic findings in psychology, vector-based models provide a compelling account of human conceptual representation.

Representation Learning is a Philosophy for Learning

Key assumptions in this philosophy:

- You can convert a high-dimensional input space into a low-dimensional representation space
  - Example: RGB images → 100 dim vectors
- A good representation space will have a "structure"
  - Example: Similarity, Symmetry, Relations will be easy to understand
  - Why? So that we can do arithmetic in representation space to do tasks
- Representations can be learned from data
- Representations can be leveraged for doing tasks

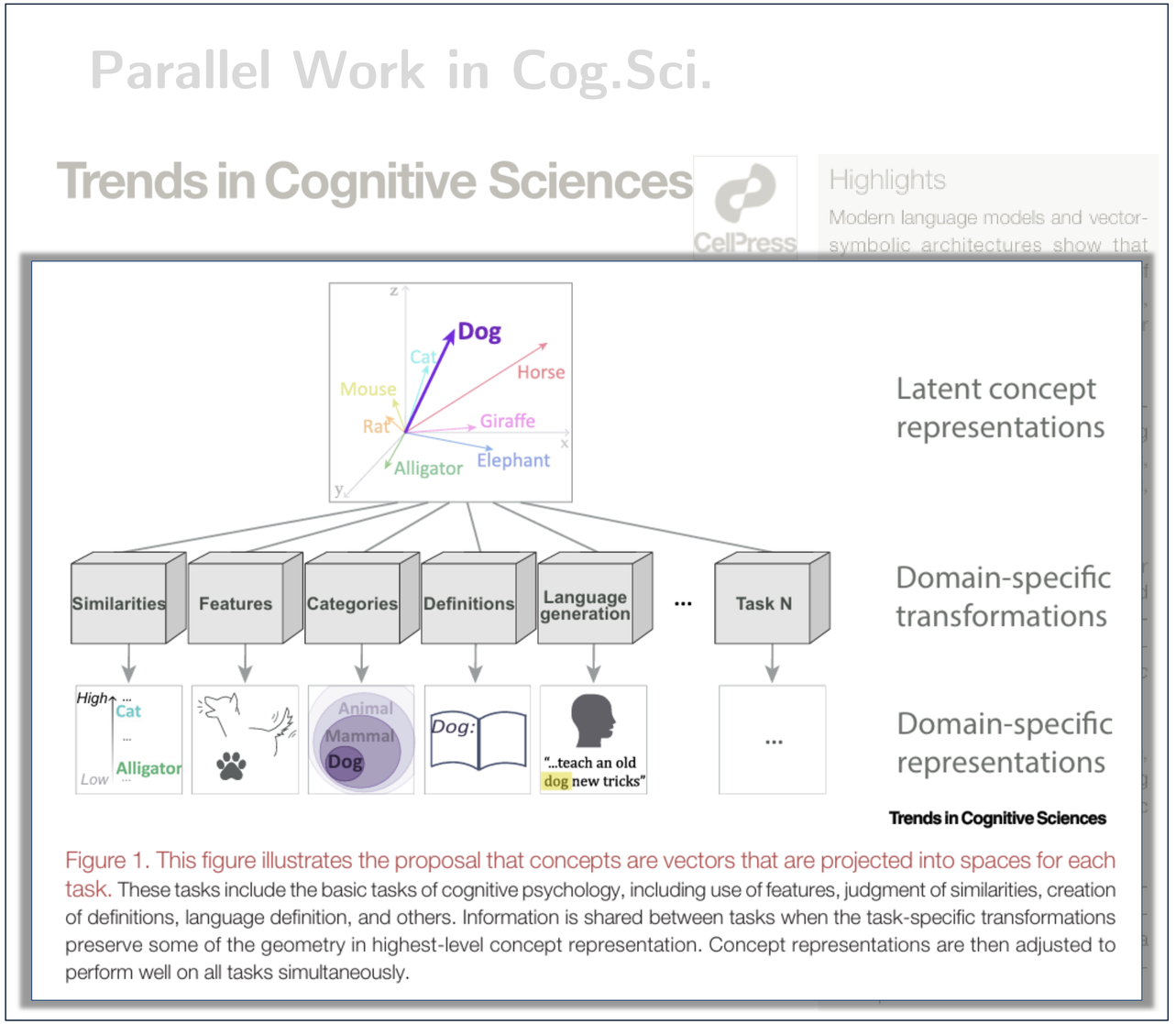


Figure 1. This figure illustrates the proposal that concepts are vectors that are projected into spaces for each task. These tasks include the basic tasks of cognitive psychology, including use of features, judgment of similarities, creation of definitions, language definition, and others. Information is shared between tasks when the task-specific transformations preserve some of the geometry in highest-level concept representation. Concept representations are then adjusted to perform well on all tasks simultaneously.

Ok whatever. Tell us how it works ...

Types of Modeling (Probabilistic Interpretation)

Data: x; **Label: y**

"cat"

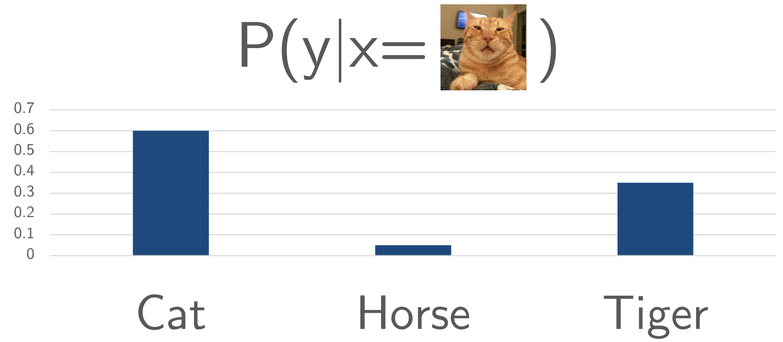
Density Function: $p(x)$

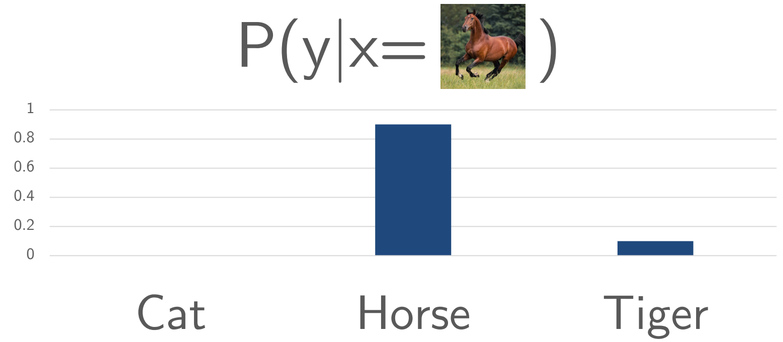
$$\int_X p(x)\,dx = 1$$

(probabilities of all inputs sum to 1)

Discriminative Model

Learn Prob. Dist. $P(y|x)$





$$\forall x, \sum_c P(y = c|x) = 1$$

Types of Modeling (Probabilistic Interpretation)

Data: x; **Label: y**

"cat"

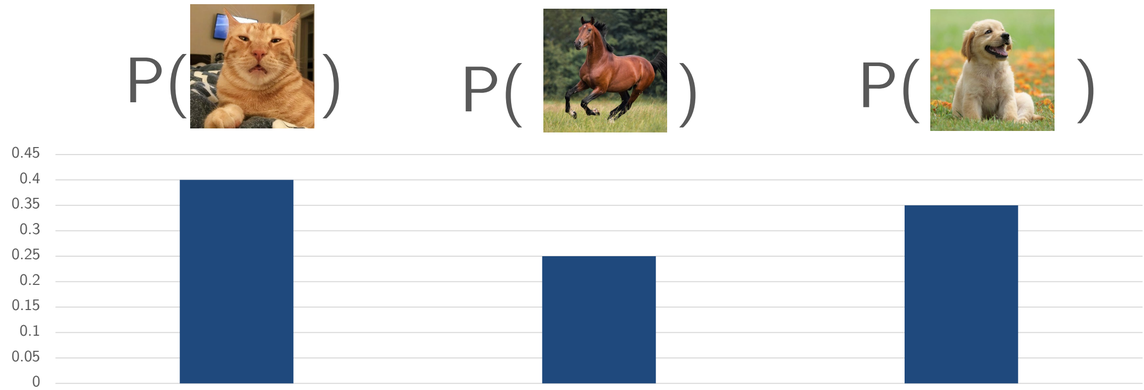
Density Function: $p(x)$

$$\int_X p(x)\,dx = 1$$

(probabilities of all inputs sum to 1)

Generative Model

Learn Marginal Prob. Dist. $P(x)$



Conditional Generative Model

Learn conditional probability $P(x|y)$

$$P(x \mid y) = \frac{P(y \mid x)}{P(y)} P(x)$$

Conditional Generative Model

Discriminative Model

Prior over labels

(Unconditional) Generative Model

Types of Modeling (Probabilistic Interpretation)

Data: x; **Label: y**

"cat"

Density Function: $p(x)$

$$\int_X p(x)\,dx = 1$$

(probabilities of all inputs sum to 1)

- **Discriminative Model**

  Learn Prob. Dist. $P(y|x)$

- **Generative Model**

  Learn Marginal Prob. Dist. $P(x)$

- **Conditional Generative Model**

  Learn conditional probability $P(x|y)$

Types of Modeling (Probabilistic Interpretation)

APPLICATIONS

Classification, Regression, Representation Learning (with labels)

- **Discriminative Model**

  Learn Prob. Dist. $P(y|x)$

- **Generative Model**

  Learn Marginal Prob. Dist. $P(x)$

- **Conditional Generative Model**

  Learn conditional probability $P(x|y)$

Types of Modeling (Probabilistic Interpretation)

APPLICATIONS

- **Discriminative Model**

  Learn Prob. Dist. $P(y|x)$

- **Generative Model**

  Learn Marginal Prob. Dist. $P(x)$

Data Generation
Outlier Detection
Representation Learning
(without labels)

- **Conditional Generative Model**

  Learn conditional probability $P(x|y)$

Types of Modeling (Probabilistic Interpretation)

APPLICATIONS

- **Discriminative Model**

  Learn Prob. Dist. $P(y|x)$

- **Generative Model**

  Learn Marginal Prob. Dist. $P(x)$

- **Conditional Generative Model**

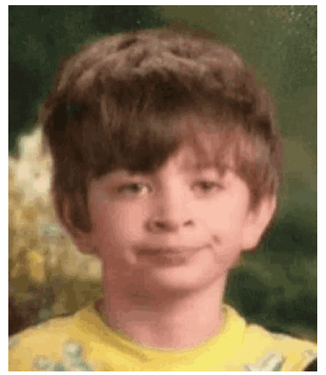
  Learn conditional probability $P(x|y)$

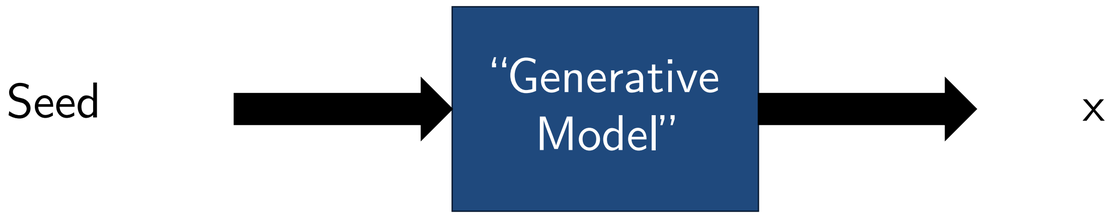
Machine Translation
Text-to-image generation

(pretty much every "GenAI" product you see is a conditional generative model)

Generative Models

- What's a Generative Model?
  - A model for the probability distribution of data x — P(x)
  - A model that can be used to "generate" data — marketing term "genAI"





- Generative Models can be *learned*
  - *You are given some observed data X* — *(e.g. face images)*
  - *You choose a function (e.g. neural network) to model* $P(x;\theta)$ *using parameters* θ
  - *You estimate* θ *s.t.* $P(x;\theta)$ *best fits the observations X*

Generative Models

- Generative Models can be *learned*
  - *You estimate* θ *s.t.* $P(x;\theta)$ **best fits** *the observations* X

- **'Best fit"** in what sense?
  - *Maximum Likelihood*

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\, P(x;\theta)$$

- How to model the distribution of high dimensional data?

$$P(x) = \int_z P(x,z)\, dz = \int_z P(z)P(x|z)$$

  - $P_\theta(z)$ and $P_\theta(x|z)$ can be factorized

$$P_\theta(x|z) = P_\theta(x^1 \ldots, x^D\ |z) = \prod_i P_\theta(x^i|z)$$

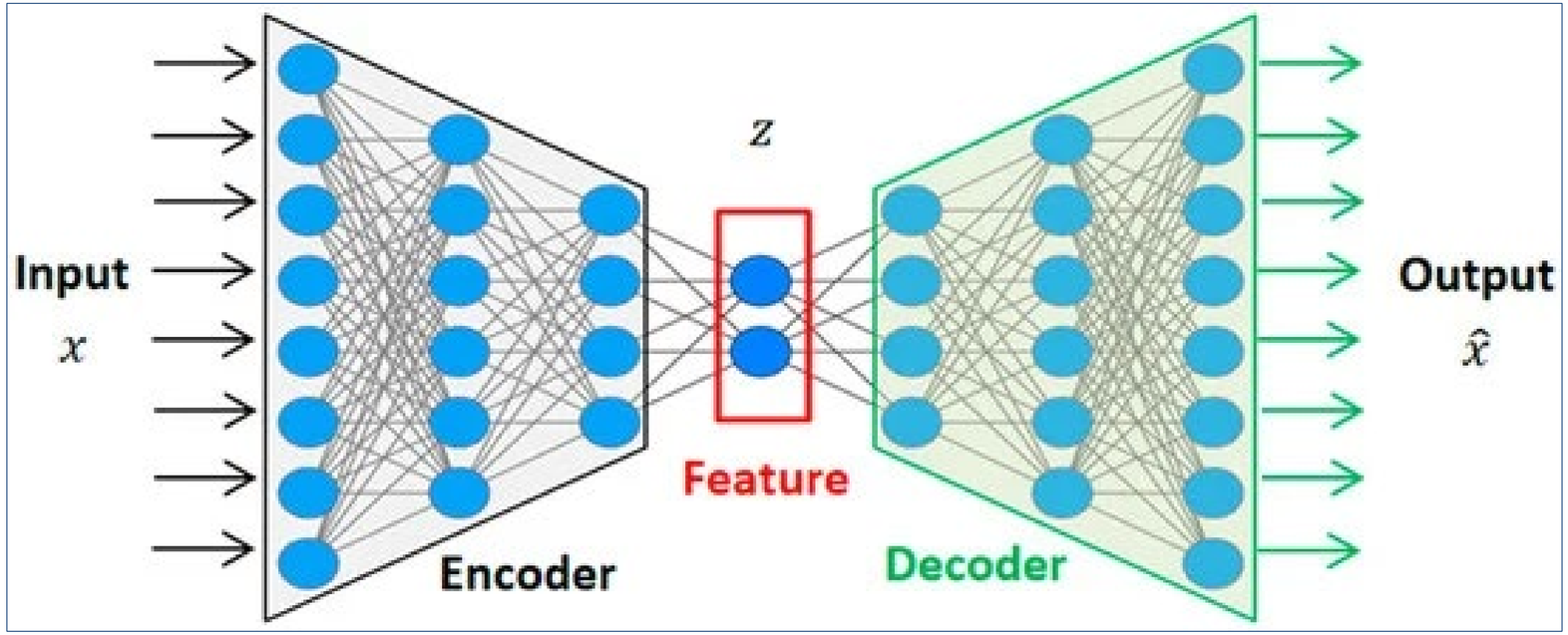
$$\theta^* = \underset{\theta}{\operatorname{argmax}}\, P_\theta(x) = \underset{\theta}{\operatorname{argmax}} \prod_i P_\theta(x^i|z) = \underset{\theta}{\operatorname{argmax}}\ \log \sum_i P_\theta(x^i|z)$$

Ok whatever. Tell us how it works ...

Let's start simple ...

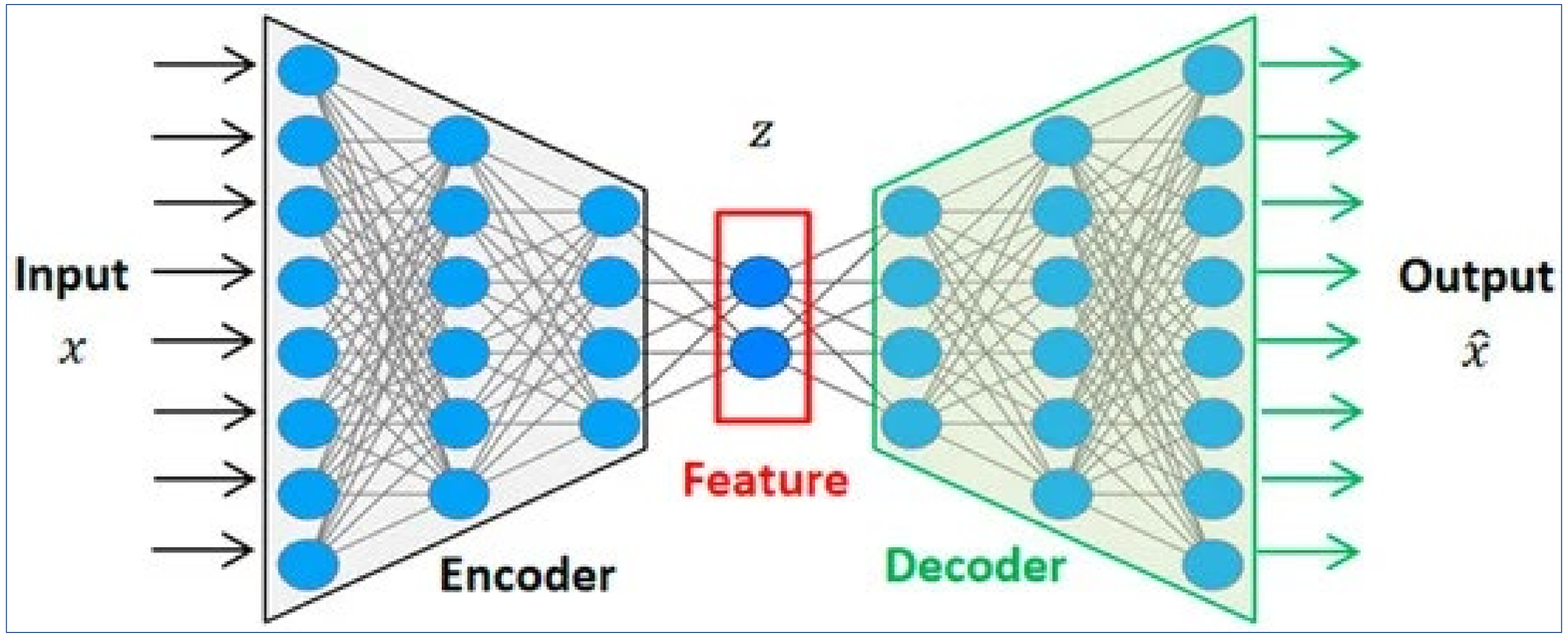
The idea of an “Auto-encoder”

- NN trained to reproduce the input $\hat{x} = F(x)$
- $F()$ is a composition of two functions: encoder $E()$ and decoder $D()$
  - Embedding / Feature / Latent $z = E(x)$
  - Output $\hat{x} = D(z) = D\big(E(x)\big)$



How would you train an autoencoder?

Loss Function?



Autoencoder: Loss Function

- The objective is to minimize the "distance" between x and $\hat{x}$
  - If $d(x, \hat{x}) = 0$ then we get perfect reconstruction

- Mean squared error!

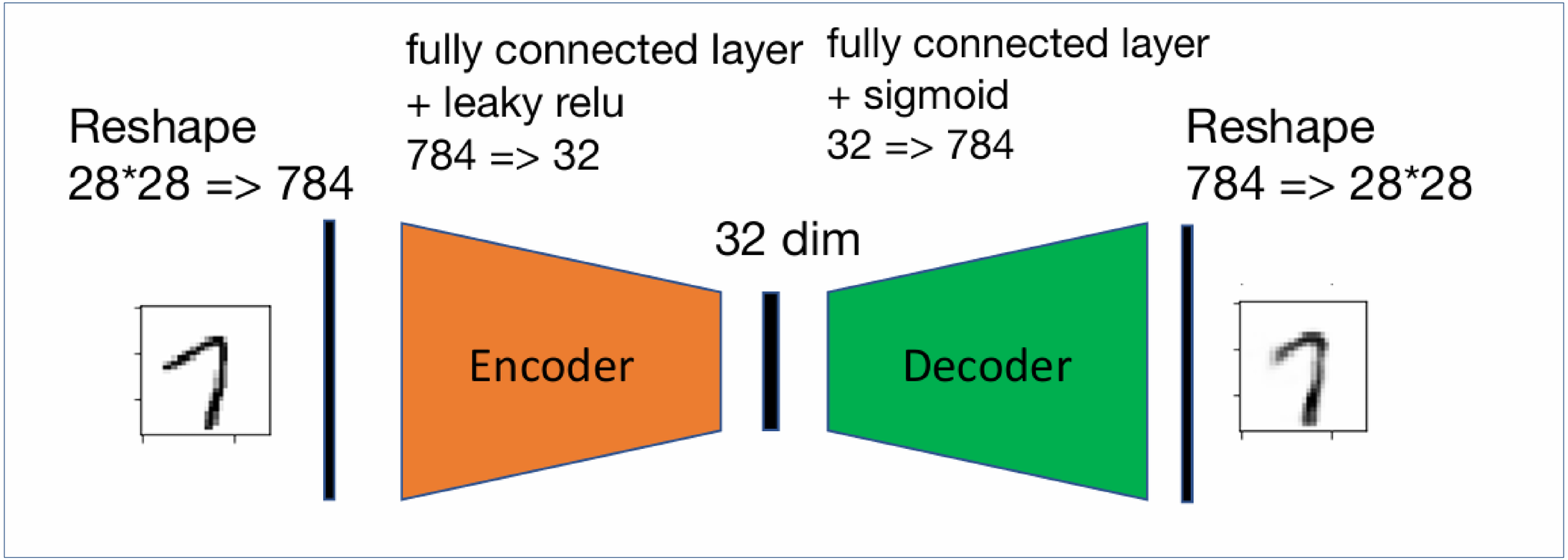
$$l(f(\mathbf{x})) = \frac{1}{2}\sum_k (\hat{x}_k - x_k)^2$$

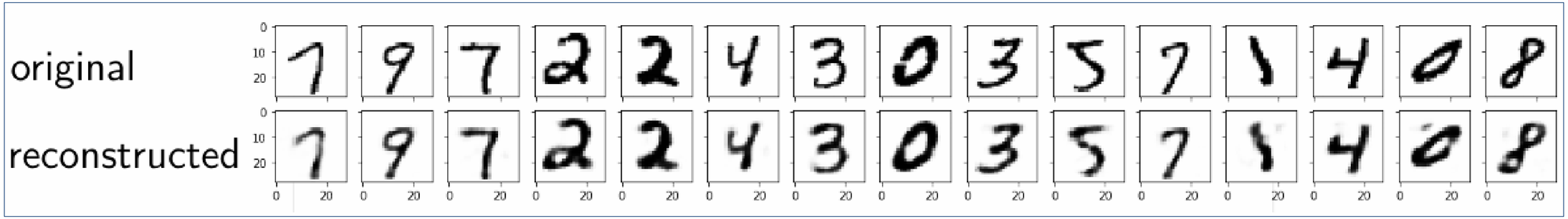
- Cross Entropy (for binary inputs)

$$l(f(\mathbf{x})) = -\sum_k \left(x_k \log(\hat{x}_k) + (1 - x_k)\log(1 - \hat{x}_k)\right)$$

- For both cases, gradient is very simple: $\nabla_x L(f(x), x) = \hat{x} - x$

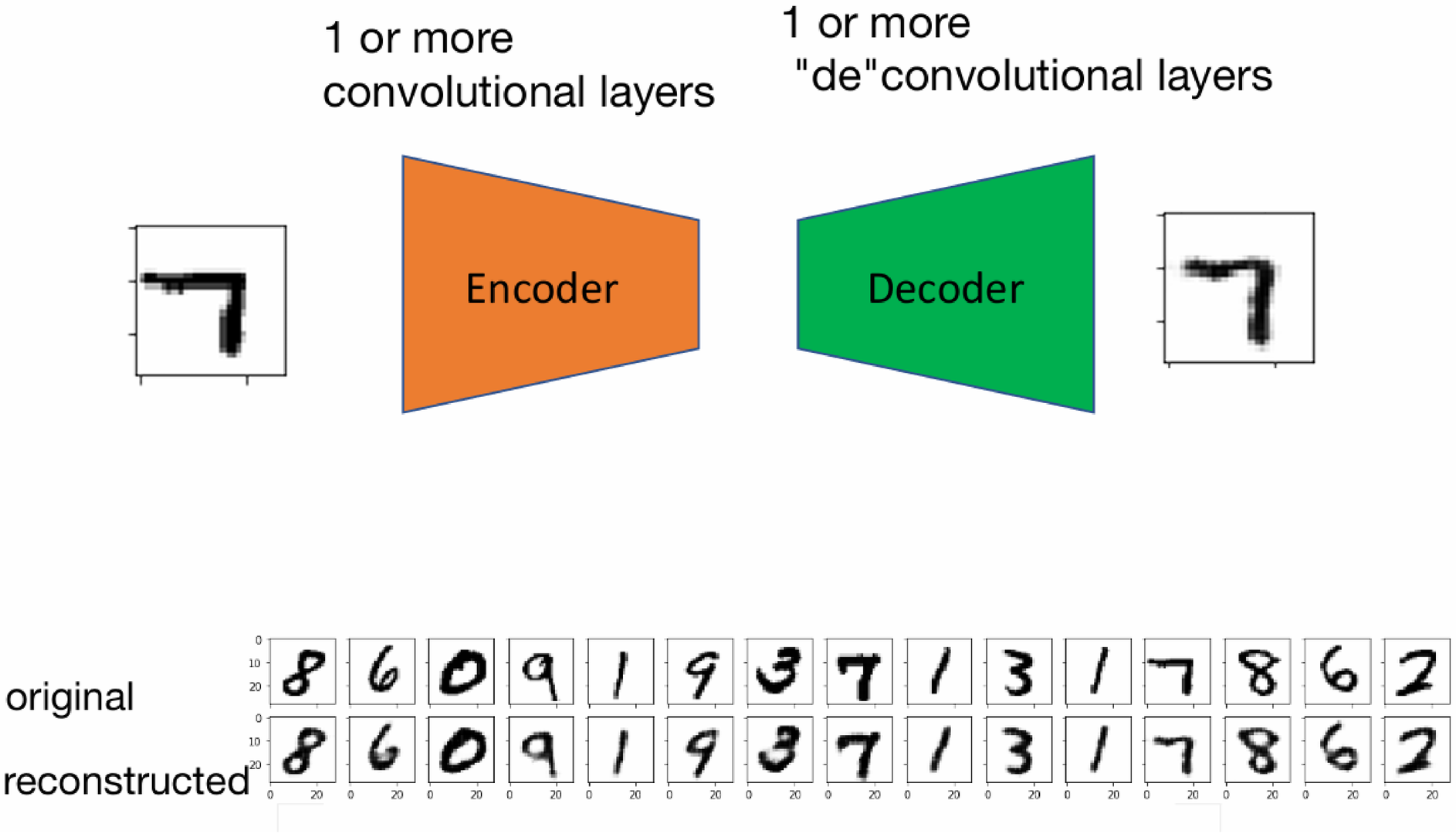
Autoencoder: Simple Example





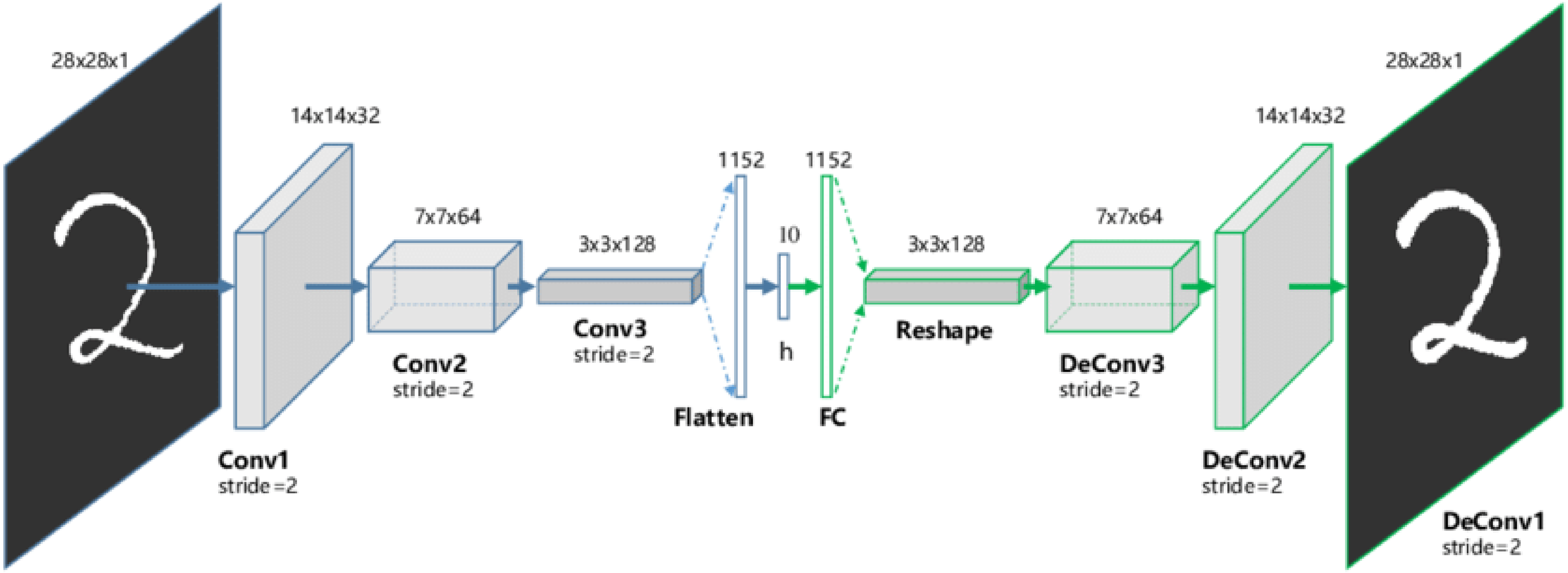
Source: Sebastian Raschka

Convolutional Autoencoder



Source: Sebastian Raschka

Convolutional Autoencoder

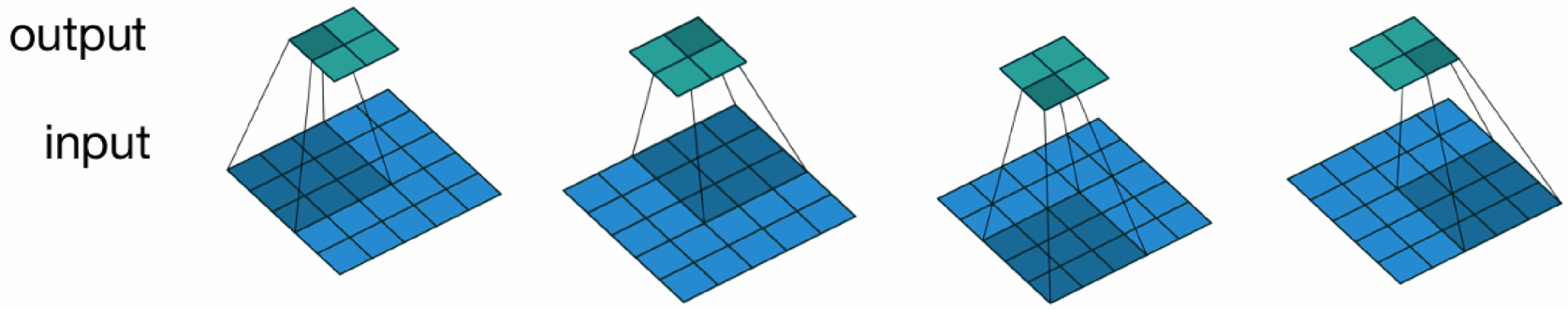


Convolutional Autoencoder:
Expand Dimensions? Transposed Convolution!

- The decoder needs to "expand dimensions"
  - Convert a small feature z into a large input x
- Use transposed convolution! A.K.A. fractionally stride convolution
  - Often (incorrectly) called "de"convolution
  - This is an incorrect term because mathematically "deconvolution" is "inverse of convolution"

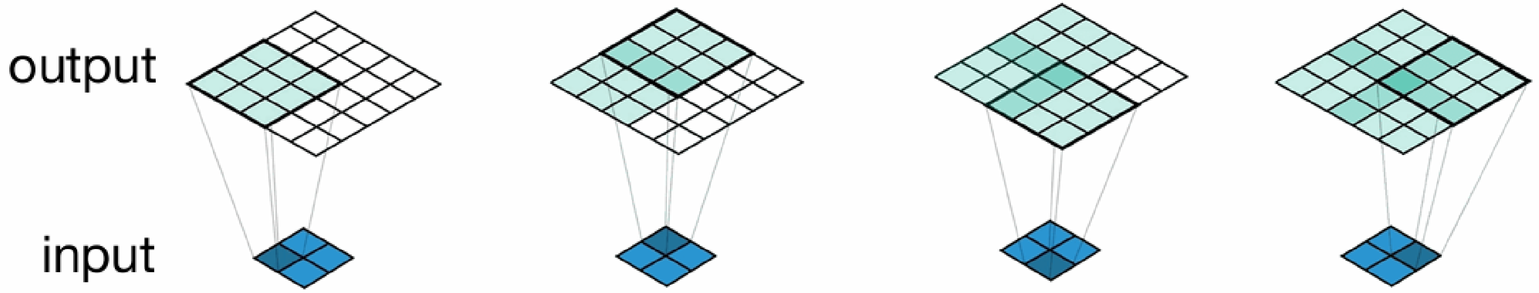
Convolutional Autoencoder:
Expand Dimensions? Transposed Convolution!

Regular Convolution:



Dumoulin, Vincent, and Francesco Visin. "A guide to convolution arithmetic for deep learning." *arXiv preprint arXiv:1603.07285* (2016).

Transposed Convolution (stride = 2)



Transposed Conv in PyTorch

```
import torch
```

```
torch.manual_seed(123)
a = torch.rand(4).view(1, 1, 2, 2)

conv_t = torch.nn.ConvTranspose2d(in_channels=1,
                                  out_channels=1,
                                  kernel_size=(3, 3),
                                  padding=0,
                                  stride=1)

# output = s(n-1)+k-2p = 1*(2-1)+3-2*0 = 4
conv_t(a)
```

```
tensor([[[[-0.2863, -0.2766, -0.1478, -0.3274],
          [-0.3522, -0.5356, -0.1591, -0.2911],
          [-0.3054, -0.4644, -0.3286, -0.2444],
          [-0.2332, -0.2557, -0.1876, -0.3970]]]],
       grad_fn=<ThnnConvTranspose2DBackward>)
```

$$\text{output} = s(n-1) + k - 2p$$

```
torch.manual_seed(123)
a = torch.rand(16).view(1, 1, 4, 4)

conv_t = torch.nn.ConvTranspose2d(in_channels=1,
                                  out_channels=1,
                                  kernel_size=(3, 3),
                                  padding=0,
                                  stride=1)

# output = s(n-1)+k-2p = 1*(4-1)+3-2*0 = 6
conv_t(a).size()
```

```
torch.Size([1, 1, 6, 6])
```

```
torch.manual_seed(123)
a = torch.rand(64).view(1, 1, 8, 8)

conv_t = torch.nn.ConvTranspose2d(in_channels=1,
                                  out_channels=1,
                                  kernel_size=(3, 3),
                                  padding=0,
                                  stride=1)

# output = s(n-1)+k-2p = 1*(8-1)+3-2*0 = 10
conv_t(a).size()
```

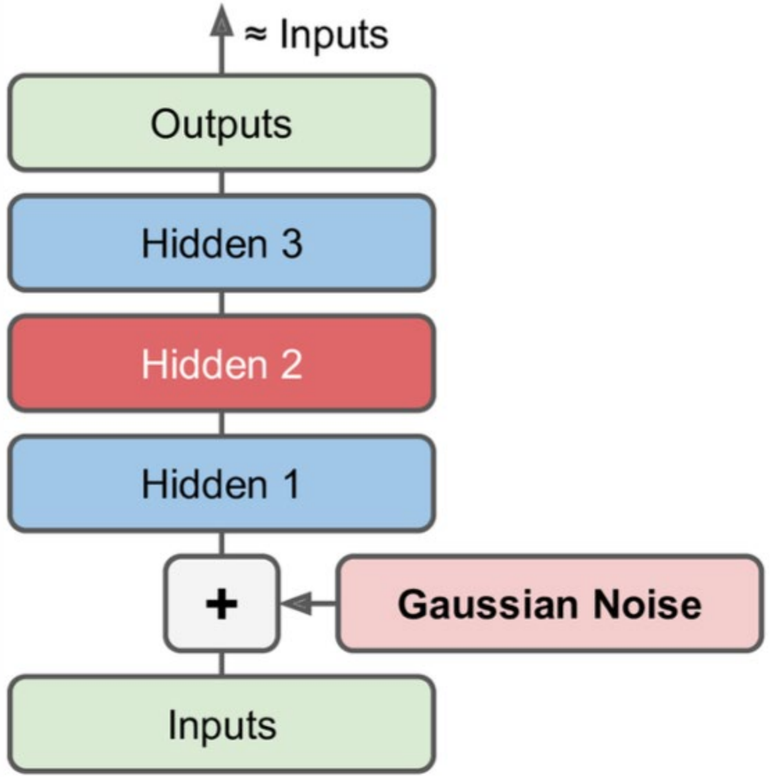
```
torch.Size([1, 1, 10, 10])
```

Denoising Autoencoder

- The input is "noisy" $\tilde{x}$. The expected output is a clean image (denoised image)

- Noise Examples:
  - Gaussian: $\tilde{x} = x + z; \quad z \sim N(0, \sigma^2 I)$
  - Masking: Zero-out some of the components of x (for images, make some pixels 0)
    - Can be random masks
    - Can be square masks

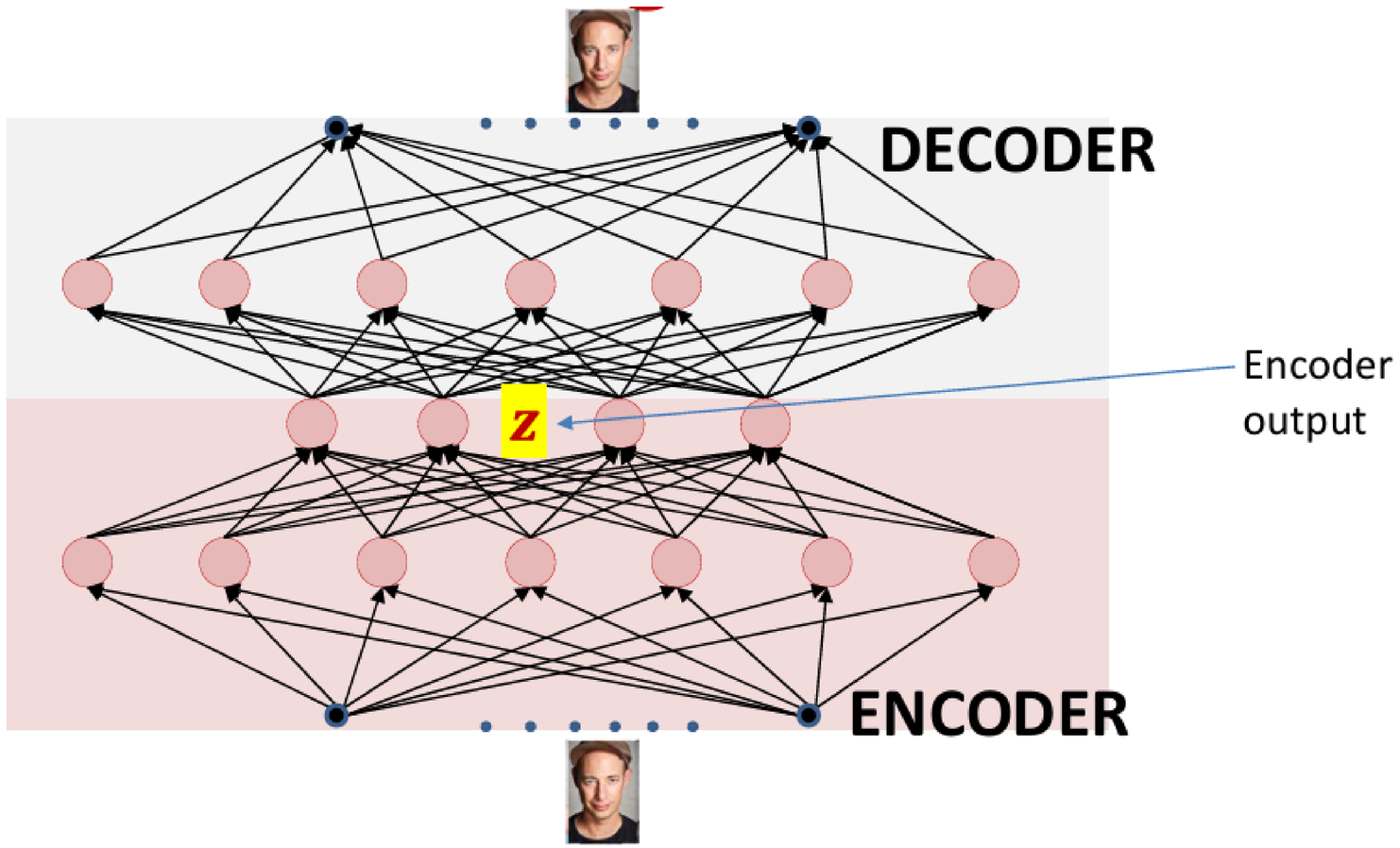
- Adding noise makes representations more robust
  - Expect $D(E((\tilde{x})) = x$ for all z



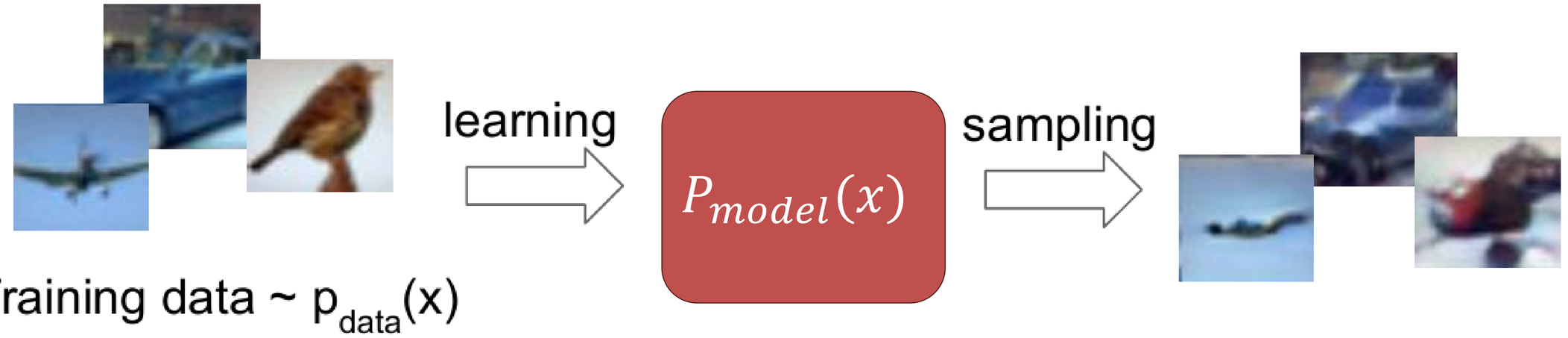
Vincent, Larochelle, Bengio, Manzagol. ICML 2008

Example: Face Auto-Encoder

Once trained, what can you do with this model?



With Generative Models, there are 2 objectives:



Objectives:

1. Learn $p_{model}(x)$ that approximates $p_{data}(x)$
2. Sampling new x from $p_{model}(x)$

figure adapted from Ranjay Krishna

An Auto-Encoder is a Generative Model

Probabilistic Interpretation:

- Encoder $E()$ estimates $P_{\theta_E}(z|x)$
- Decoder $D()$ estimates $P_{\theta_D}(x|z)$
- The marginal $P(x) = \int_z P(x,z)\,dz = \int_z P(z)P(x|z)$

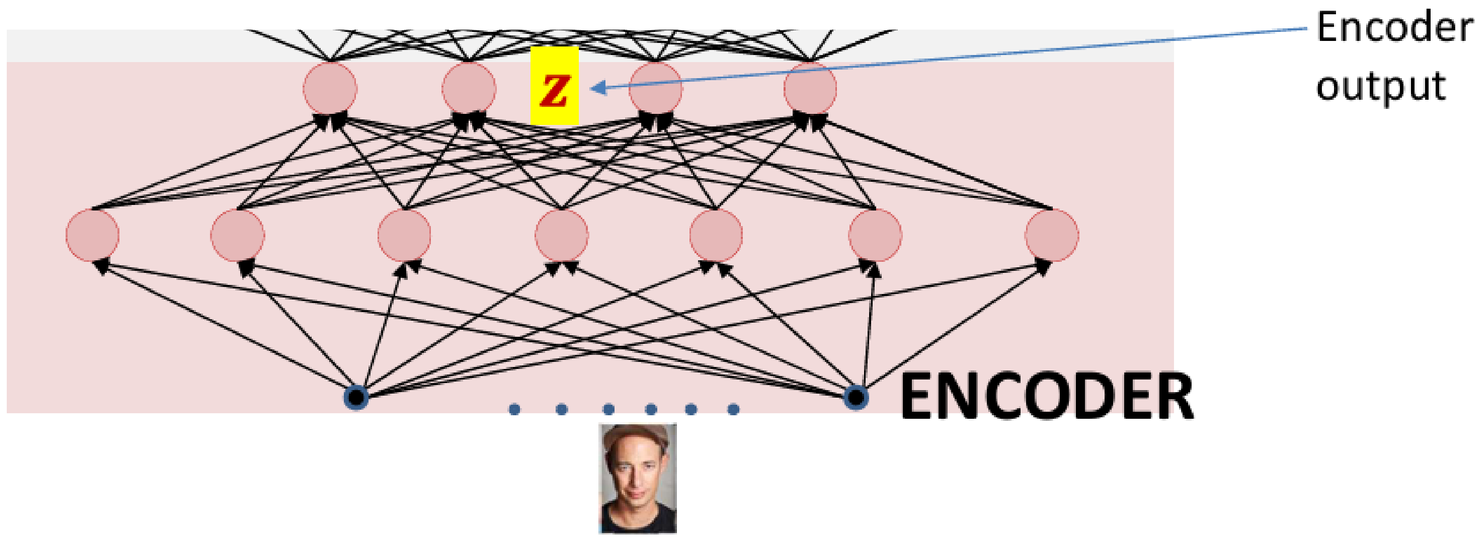
  Bayes/Chain Rule ...
- Once the AE is "trained"
  - you get a generative model that generates "x" given a latent code "z"
  - A conditional generative model takes an additional input "y" $P(x|z,y)$
    - E.g. generating images from text y=text, x = image
    - *More on this later ...*

Example: Face Auto-Encoder

Once trained, what can you do with this model?

(1) Encode images into vectors

(*throw away the decoder ...*)



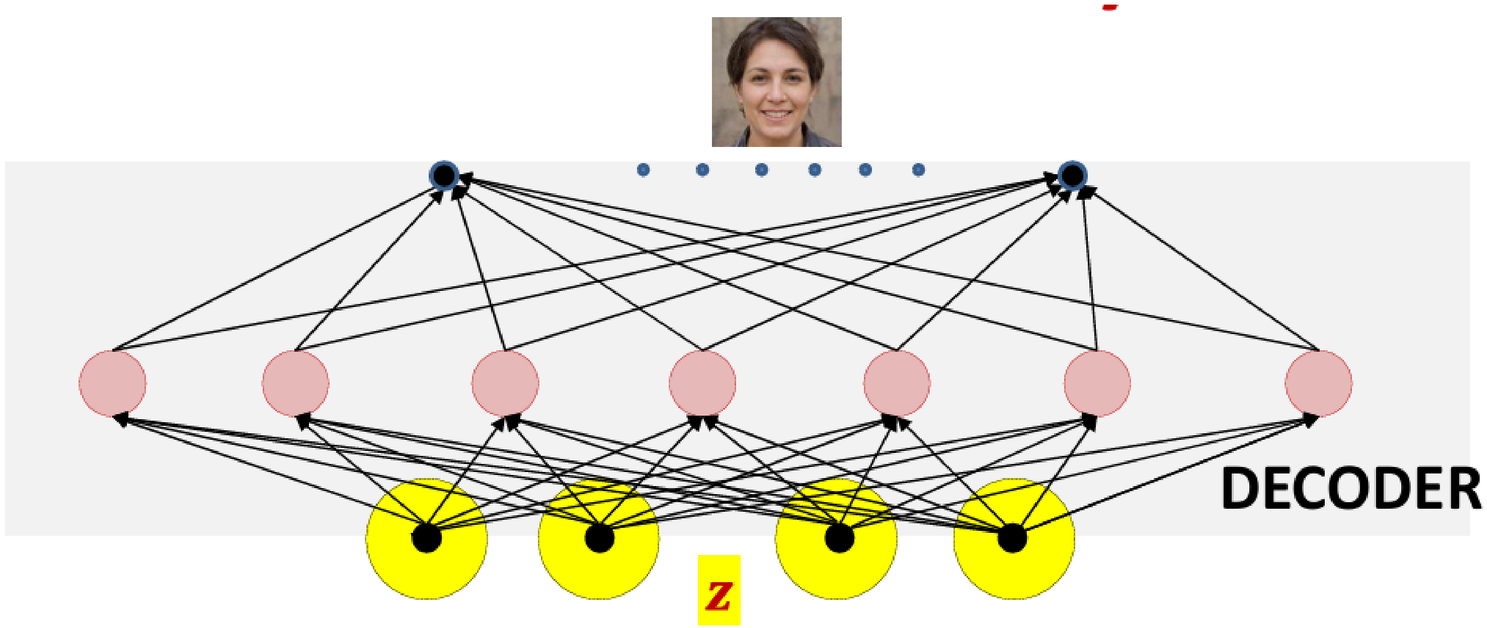
Example: Face Auto-Encoder

Once trained, what can you do with this model?

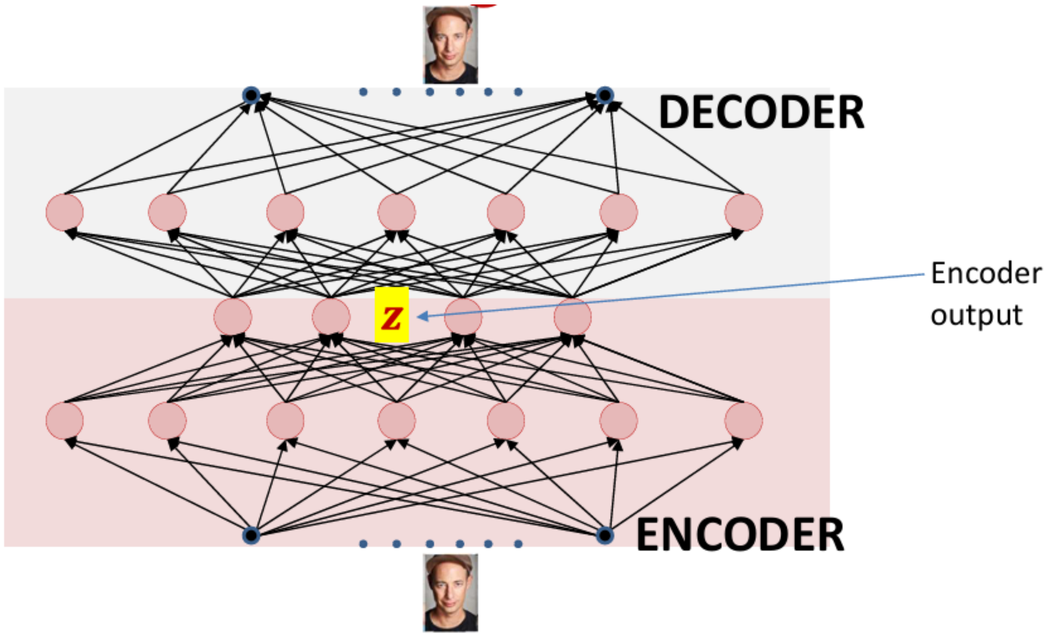
(1) Encode images into vectors

(2) Generate new faces ...

(*throw away the encoder*)



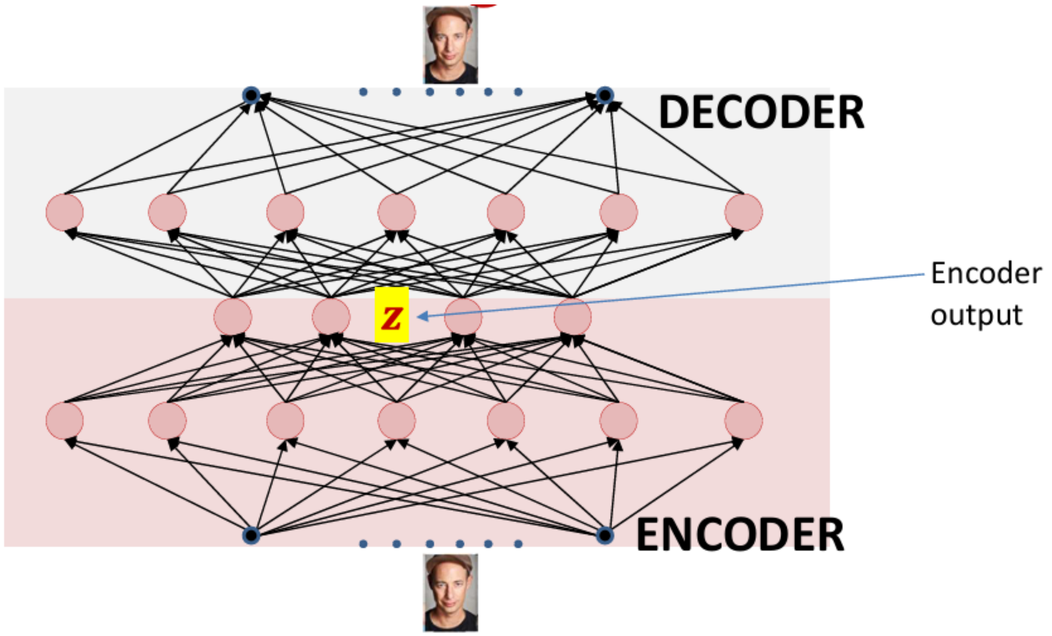
Types of Autoencoders



So far, we have not enforced any "structure" on the latents z

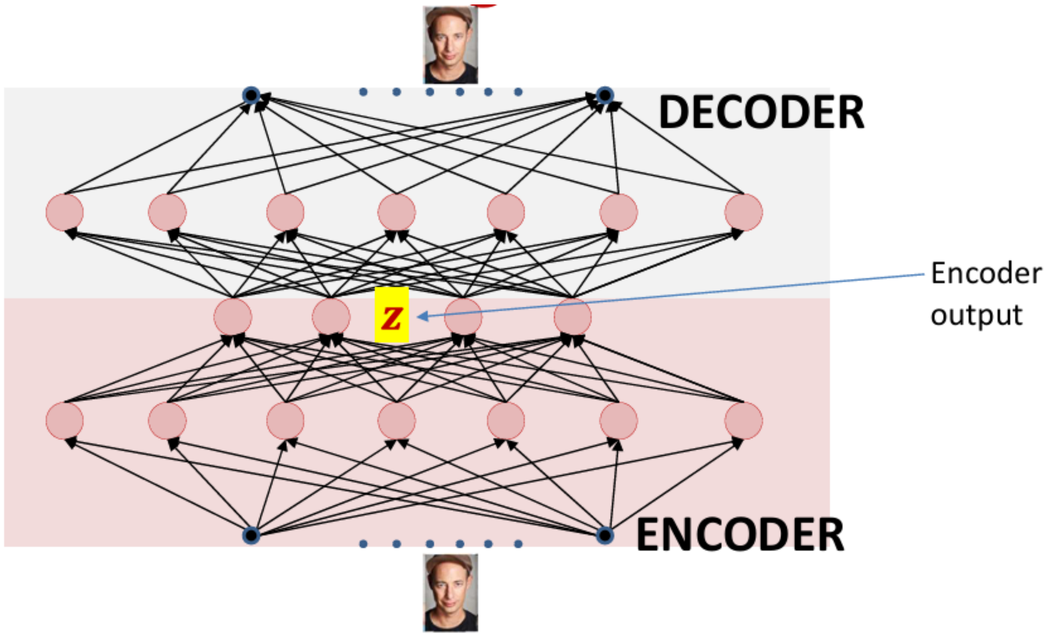
But a structure is desirable

(Remember our motivations / goals for representation learning)



So far, we have not enforced any “structure” on the latents z

We can’t generate ***new images*** from D() if we don’t understand the z-space

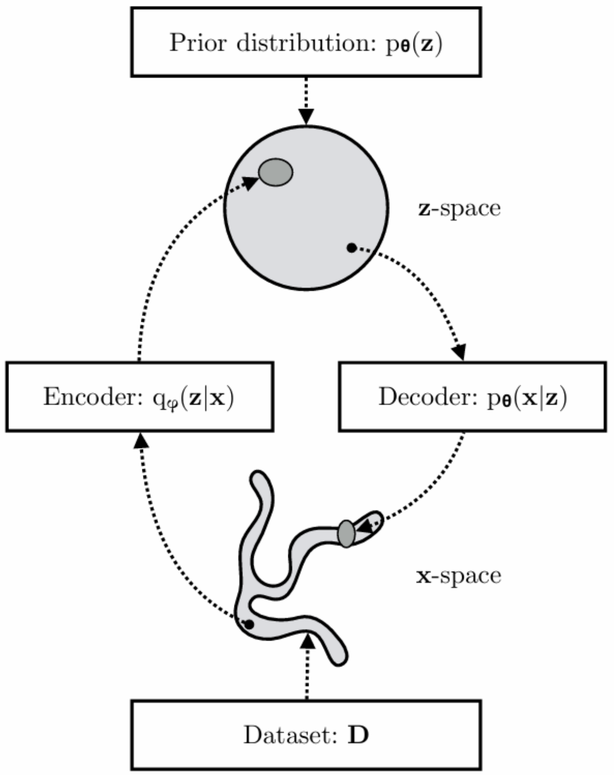


So far, we have not enforced any "structure" on the latents z

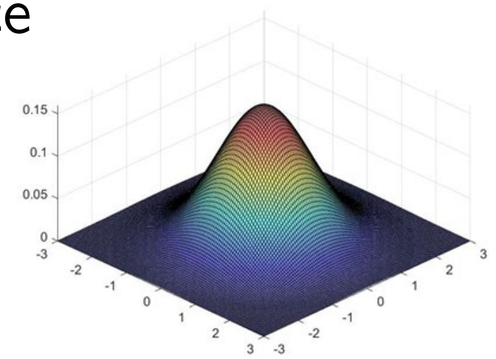
We can't generate ***new images*** from D() if we don't understand the z-space

For example, if I ask you to generate a "face with beard, glasses, brown hair" which z would you choose?

VAE: Variational Autoencoder

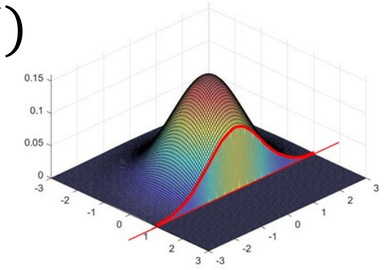
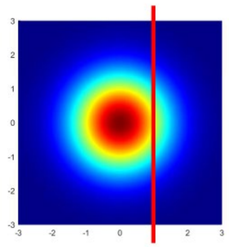



- Force a "prior" distribution on the latent space
  - Example: Gaussian $N(0, I)$

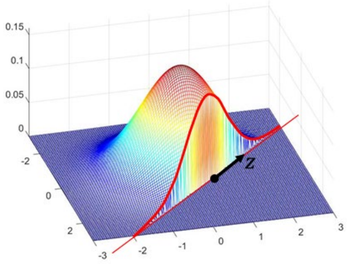


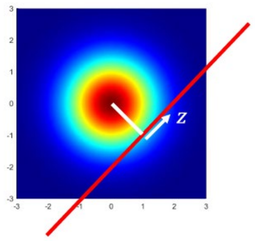
- Gaussians are nice because they are perfectly symmetrical in every dimension
  - Isotropic (covariance matrix is identity I)
  - Dimensions are independent,

    i.e. $P(z_1|z_2) = P(z_1) = N(0, I)\ \forall z_1, z_2$

  - Property holds for any linear combination of z elements
    - i.e. $P(z_1|az_2 + bz_3) = N(0, I)$





Kingma and Welling. Auto-Encoding Variational Bayes. ICLR 2014

Variational Autoencoders

Probabilistic spin on autoencoders - will let us sample from the model to generate data!

Assume training data $\{x^{(i)}\}_{i=1}^{N}$ is generated from the distribution of unobserved (latent) representation **z**

Variational Autoencoders

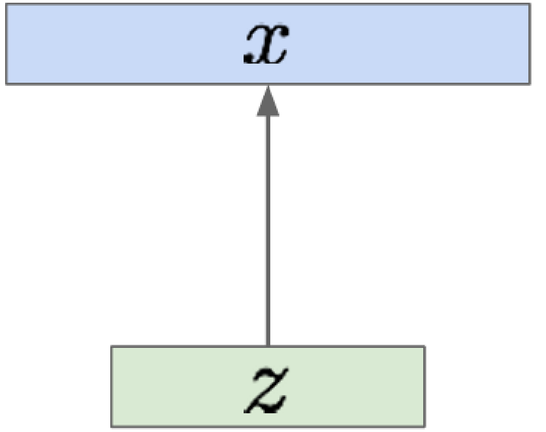
We want to estimate the parameters θ^* given training real data x.

Sample from true conditional

$p_{\theta^*}(x \mid z^{(i)})$

Sample from true prior

$z^{(i)} \sim p_{\theta^*}(z)$



How should we represent this model?

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014

These slides on VAE are adpated from Ranjay Krishna

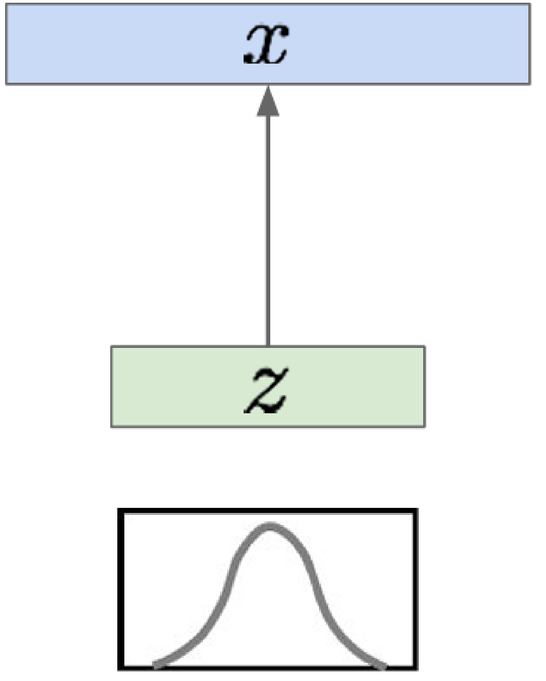
Variational Autoencoders

Sample from true conditional

$p_{\theta^*}(x \mid z^{(i)})$

Sample from true prior

$z^{(i)} \sim p_{\theta^*}(z)$



We want to estimate the parameters θ^* given training real data x.

How should we represent this model?

Choose prior p(z) to be simple, e.g. Gaussian. Reasonable for latent attributes, e.g. pose, how much smile.

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

These slides on VAE are adapted from Ranjay Krishna

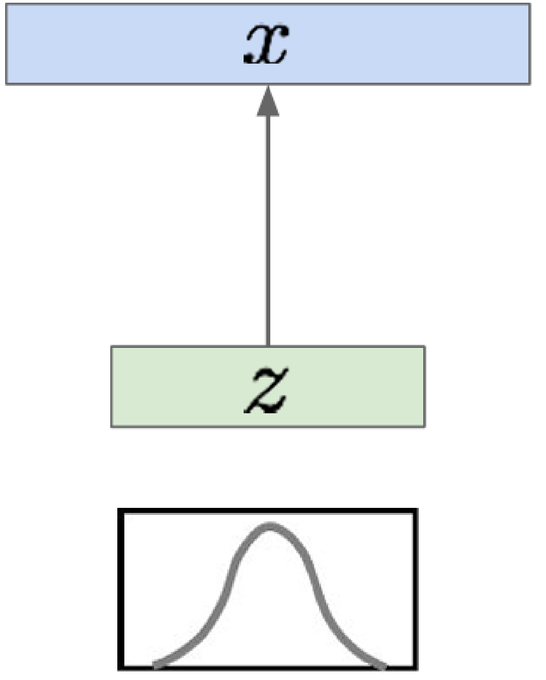
Variational Autoencoders

Sample from true conditional

$p_{\theta^*}(x \mid z^{(i)})$

Sample from true prior

$z^{(i)} \sim p_{\theta^*}(z)$



We want to estimate the parameters θ^* given training real data x.

How should we represent this model?

Choose prior p(z) to be simple, e.g. Gaussian. Reasonable for latent attributes, e.g. pose, how much smile.

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

These slides on VAE are adapted from Ranjay Krishna

Variational Autoencoders

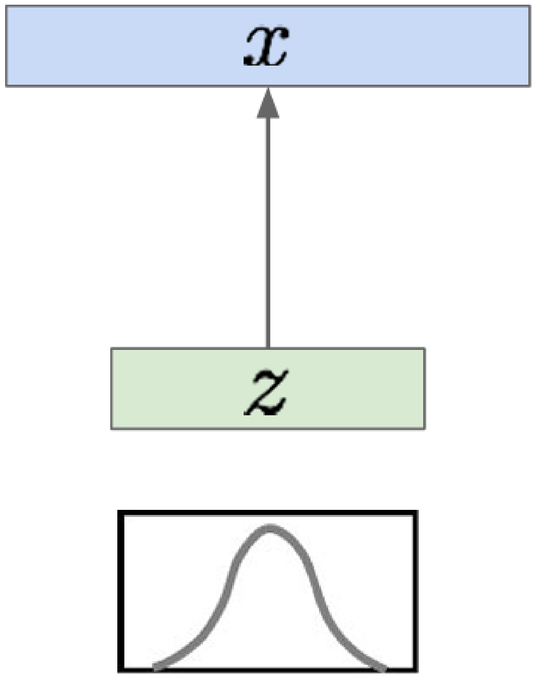
Decoder must be probabilistic:
Decoder inputs z, outputs mean $\mu_{x|z}$ and (diagonal) covariance $\sum_{x|z}$

Sample from true conditional

$p_{\theta^*}(x \mid z^{(i)})$

Sample from true prior

$z^{(i)} \sim p_{\theta^*}(z)$



We want to estimate the parameters θ^* given training real data x.

How should we represent this model?

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Variational Autoencoders

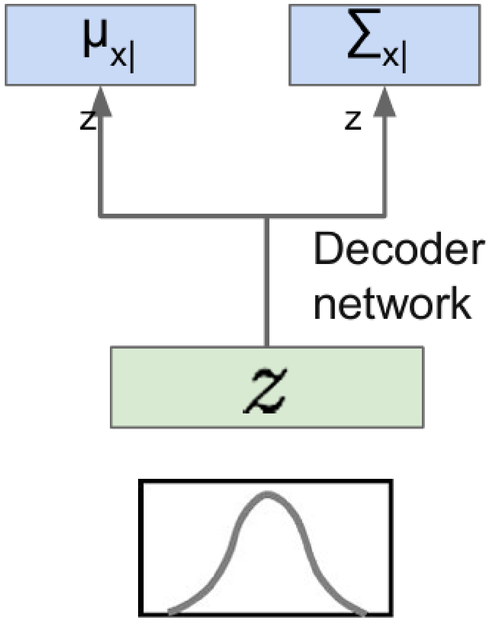
Decoder must be probabilistic:
Decoder inputs z, outputs mean $\mu_{x|z}$ and (diagonal) covariance $\sum_{x|z}$

Sample from true conditional

$p_{\theta^*}(x \mid z^{(i)})$

Sample from true prior

$z^{(i)} \sim p_{\theta^*}(z)$



We want to estimate the parameters θ^* given training real data x.

How should we represent this model?

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

These slides on VAE are adapted from Ranjay Krishna

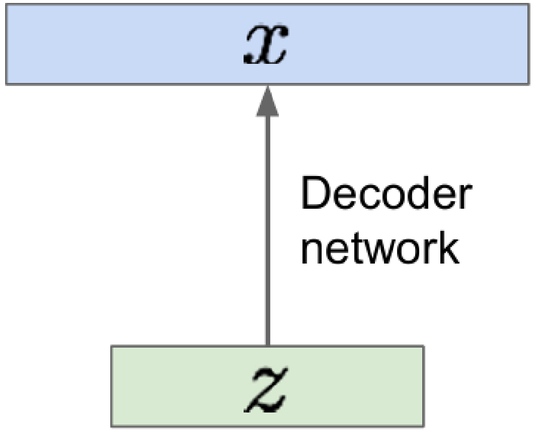
Variational Autoencoders

Sample from true conditional

$p_{\theta^*}(x \mid z^{(i)})$

Sample from true prior

$z^{(i)} \sim p_{\theta^*}(z)$



We want to estimate the parameters θ^* given training real data x.

How to train the model?

Learn model parameters to maximize likelihood of training data

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

Q: What is the problem with this?

These slides on VAE are adapted from Ranjay Krishna

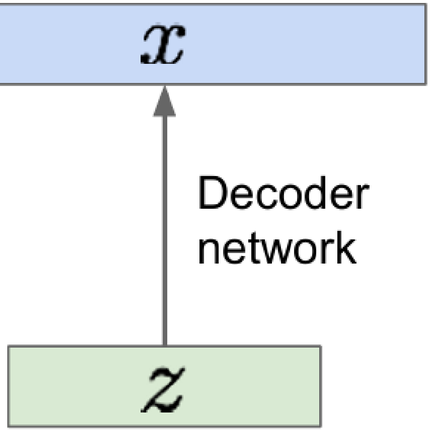
Variational Autoencoders

Sample from true conditional

$p_{\theta^*}(x \mid z^{(i)})$

Sample from true prior

$z^{(i)} \sim p_{\theta^*}(z)$



We want to estimate the parameters θ^* given training real data x.

How to train the model?

Learn model parameters to maximize likelihood of training data

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

Q: What is the problem with this?

Intractable! Impossible to iterate over all z

These slides on VAE are adapted from Ranjay Krishna

Variational Autoencoders: Intractability

Data likelihood: $p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014

Variational Autoencoders: Intractability

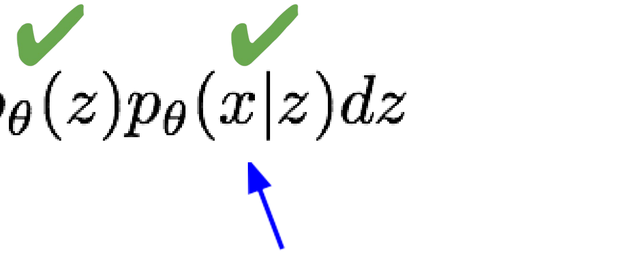
Data likelihood: $p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$

✔

Simple Gaussian prior

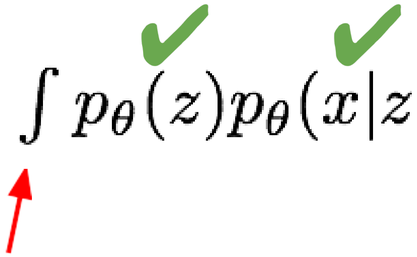
Variational Autoencoders: Intractability

Data likelihood: $p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$



Variational Autoencoders: Intractability

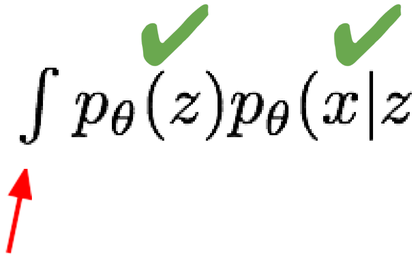
Data likelihood: $p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$



Intractable to compute p(x|z) for every z!

Variational Autoencoders: Intractability

Data likelihood: $p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$



Intractable to compute p(x|z) for every z!

$$\log p(x) \approx \log \frac{1}{k} \sum_{i=1}^{k} p(x|z^{(i)}), \text{where } z^{(i)} \sim p(z)$$

Monte Carlo estimation is too high variance

Variational Autoencoders: Intractability

Data likelihood: $p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$ ✔ ✔

Another idea: $p_\theta(x) = \dfrac{p_\theta(x \mid z) p_\theta(z)}{p_\theta(z \mid x)}$ ← Use Bayes rule

Variational Autoencoders: Intractability

Data likelihood: $p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$ ✔ ✔

Another idea: $p_\theta(x) = \dfrac{p_\theta(x \mid z) p_\theta(z)}{p_\theta(z \mid x)}$

We know how to calculate these

Variational Autoencoders: Intractability

Data likelihood: $p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$

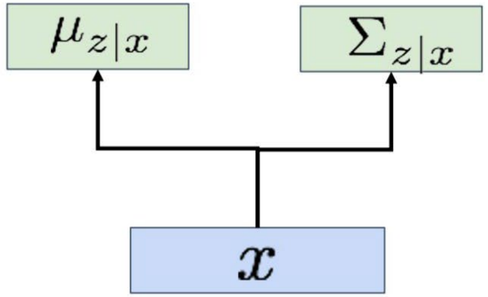
Another idea: $p_\theta(x) = \dfrac{p_\theta(x \mid z) p_\theta(z)}{p_\theta(z \mid x)}$

But how do you calculate this?

Solution: In addition to modeling $p_\theta(x|z)$,
Learn $q_\phi(z|x)$ that approximates the true posterior $p_\theta(z|x)$.

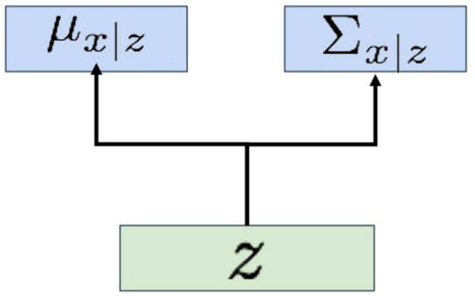
Encoder Network

$$q_\phi(z \mid x) = N(\mu_{z|x}, \Sigma_{z|x})$$



Decoder Network

$$p_\theta(x \mid z) = N(\mu_{x|z}, \Sigma_{x|z})$$



Variational Autoencoders: Intractability

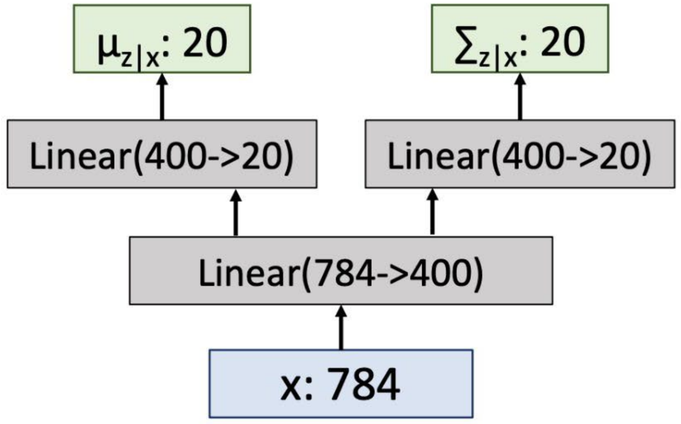
Data likelihood: $p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$

Another idea: $p_\theta(x) = \dfrac{p_\theta(x \mid z) p_\theta(z)}{p_\theta(z \mid x)}$

x: 28x28 image = 784-dim vector
z: 20-dim vector

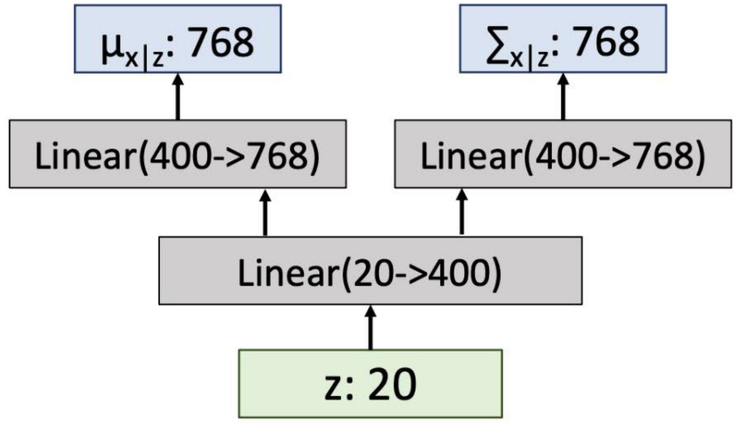
Encoder Network

$q_\phi(z \mid x) = N(\mu_{z|x}, \Sigma_{z|x})$



Decoder Network

$p_\theta(x \mid z) = N(\mu_{x|z}, \Sigma_{x|z})$



Variational Autoencoders

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

Using this approximation, we can derive a lower bound on the data likelihood p(x), making it tractable, therefore, possible to optimize.

Variational Autoencoders

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

Taking expectation wrt. z
(using encoder network) will
come in handy later

Variational Autoencoders

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

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \qquad (\text{Bayes' Rule})$$

Variational Autoencoders

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

Variational Autoencoders

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

Variational Autoencoders

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

The expectation wrt. z (using encoder network) let us write nice KL terms

Variational Autoencoders

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

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \qquad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \qquad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \qquad \text{(Logarithms)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z)) + D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))$$

Decoder network gives $p_\theta(x|z)$, can compute estimate of this term through sampling (need some trick to differentiate through sampling).

Variational Autoencoders

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

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \qquad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \qquad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \qquad \text{(Logarithms)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z)) + D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))$$

Decoder network gives $p_\theta(x|z)$, can compute estimate of this term through sampling (need some trick to differentiate through sampling).

This KL term (between Gaussians for encoder and z prior) has nice closed-form solution!

Variational Autoencoders

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

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \qquad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \qquad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \qquad \text{(Logarithms)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z)) + D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))$$

Decoder network gives $p_\theta(x|z)$, can compute estimate of this term through sampling (need some trick to

This KL term (between Gaussians for encoder and z prior) has nice closed-form

$p_\theta(z|x)$ intractable (saw earlier), can't compute this KL term :(But we know KL divergence always >= 0.

Variational Autoencoders

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

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Logarithms)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z)) + D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))$$

We want to maximize the data likelihood

Decoder network gives $p_\theta(x|z)$, can compute estimate of this term through sampling.

This KL term (between Gaussians for encoder and z prior) has nice closed-form solution!

$p_\theta(z|x)$ intractable (saw earlier), can't compute this KL term :(But we know KL divergence always >= 0.

Variational Autoencoders

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

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Logarithms)}$$

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

We want to maximize the data likelihood

Tractable lower bound which we can take gradient of and optimize! ($p_\theta(x|z)$ differentiable, KL term is differentiable)

Variational Autoencoders

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

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Logarithms)}$$

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

Decoder: reconstruct the input data

Encoder: make approximate posterior distribution close to prior

Tractable lower bound which we can take gradient of and optimize! ($p_\theta(x|z)$ differentiable, KL term differentiable)

Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

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

Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

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

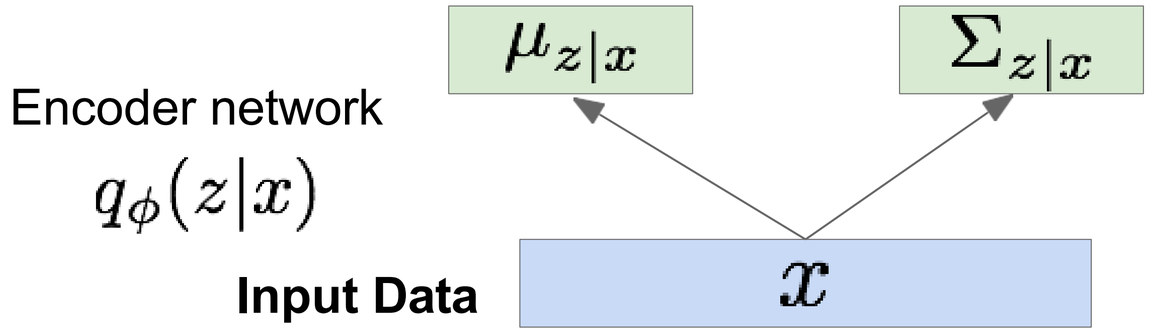
Let's look at computing the KL divergence between the estimated posterior and the prior given some data

Input Data x

Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

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



Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

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

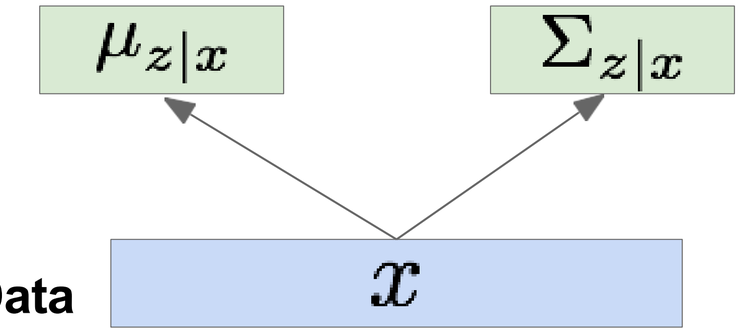
$$D_{KL}(\mathcal{N}(\mu_{z|x}, \Sigma_{z|x})||\mathcal{N}(0, I))$$

This equation has an analytical solution

Make approximate posterior distribution close to prior

Encoder network

$q_\phi(z|x)$



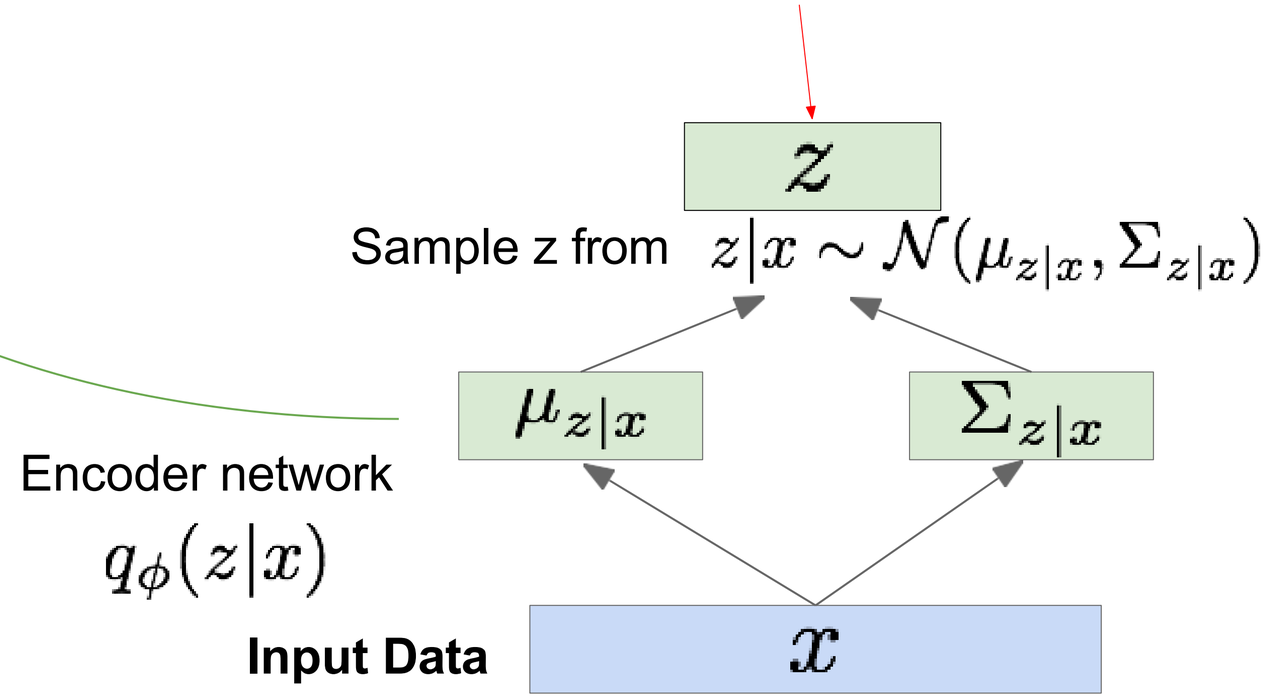
Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

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

Not part of the computation graph!

Make approximate posterior distribution close to prior



Variational Autoencoders

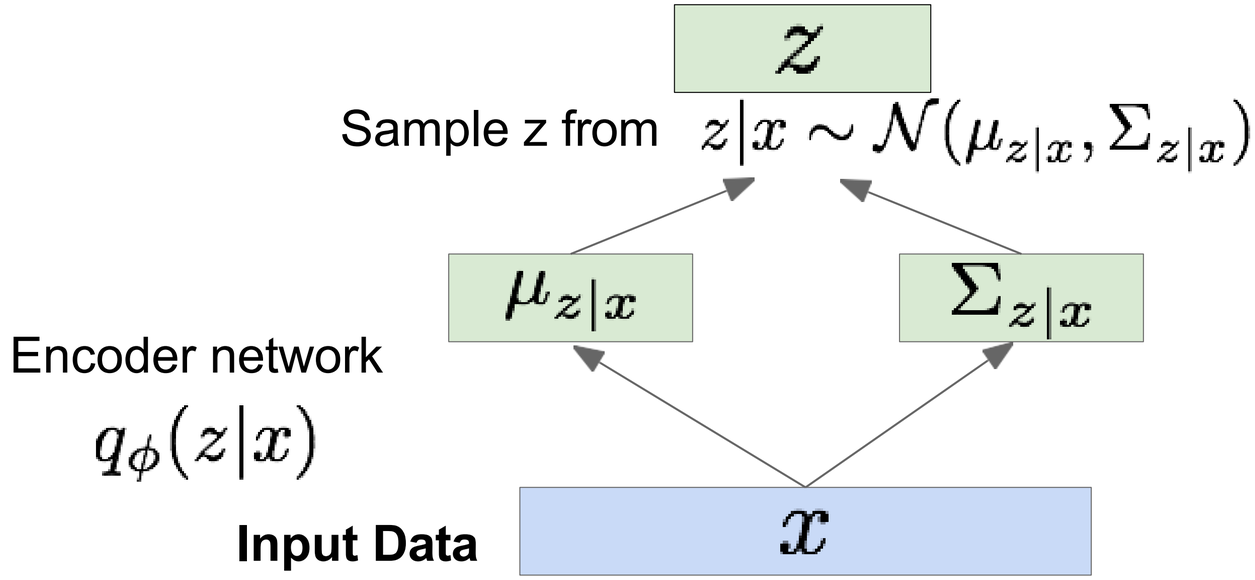
Putting it all together: maximizing the likelihood lower bound

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

Reparameterization trick to make sampling differentiable:

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

$$z = \mu_{z|x} + \epsilon\sigma_{z|x}$$



Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

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

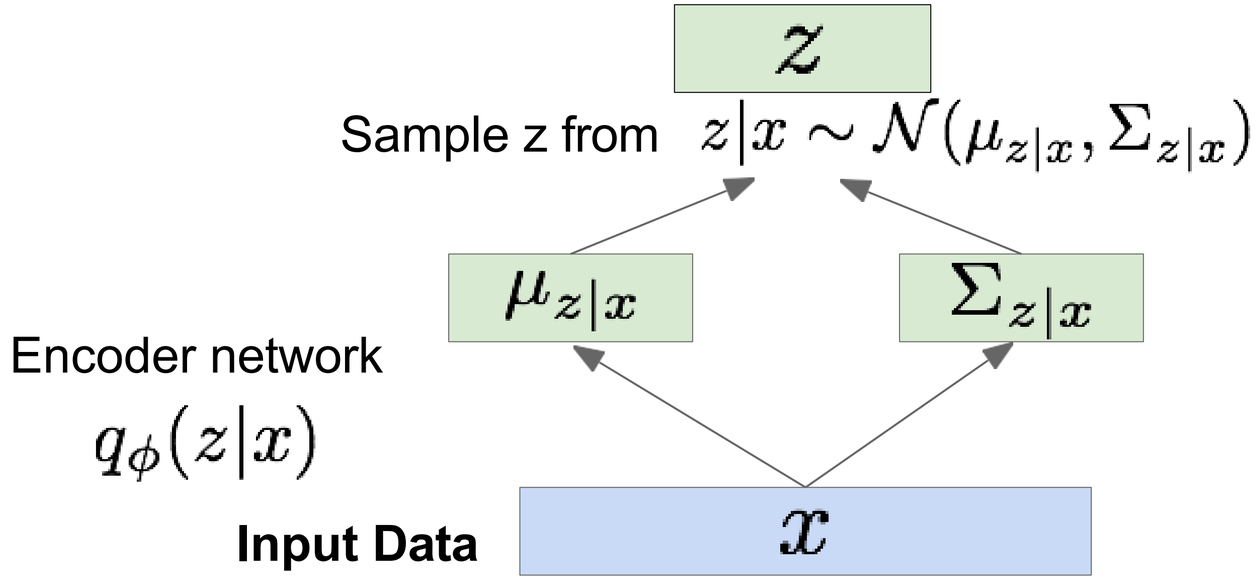
Reparameterization trick to make sampling differentiable:

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

$$z = \mu_{z|x} + \epsilon\sigma_{z|x}$$

Input to the graph

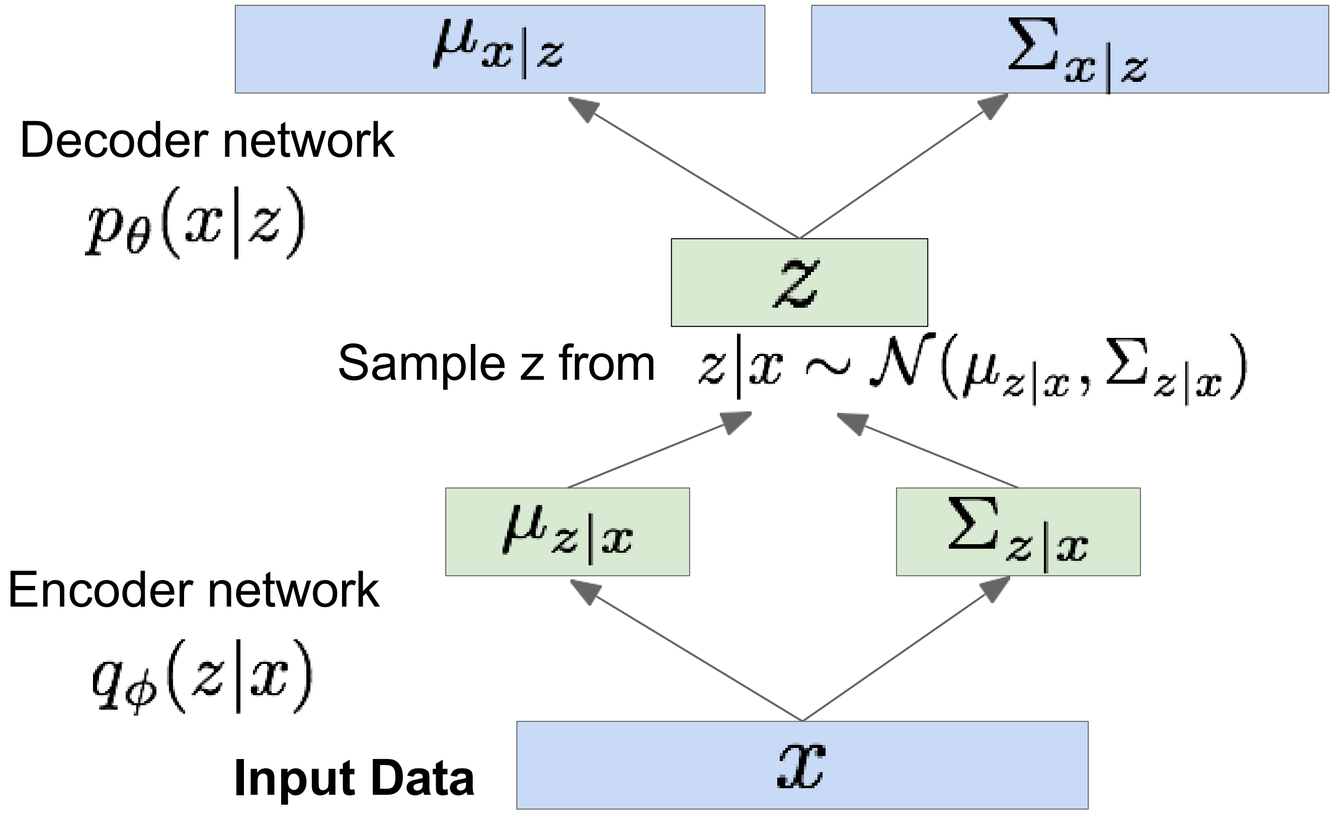
Part of computation graph



Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

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

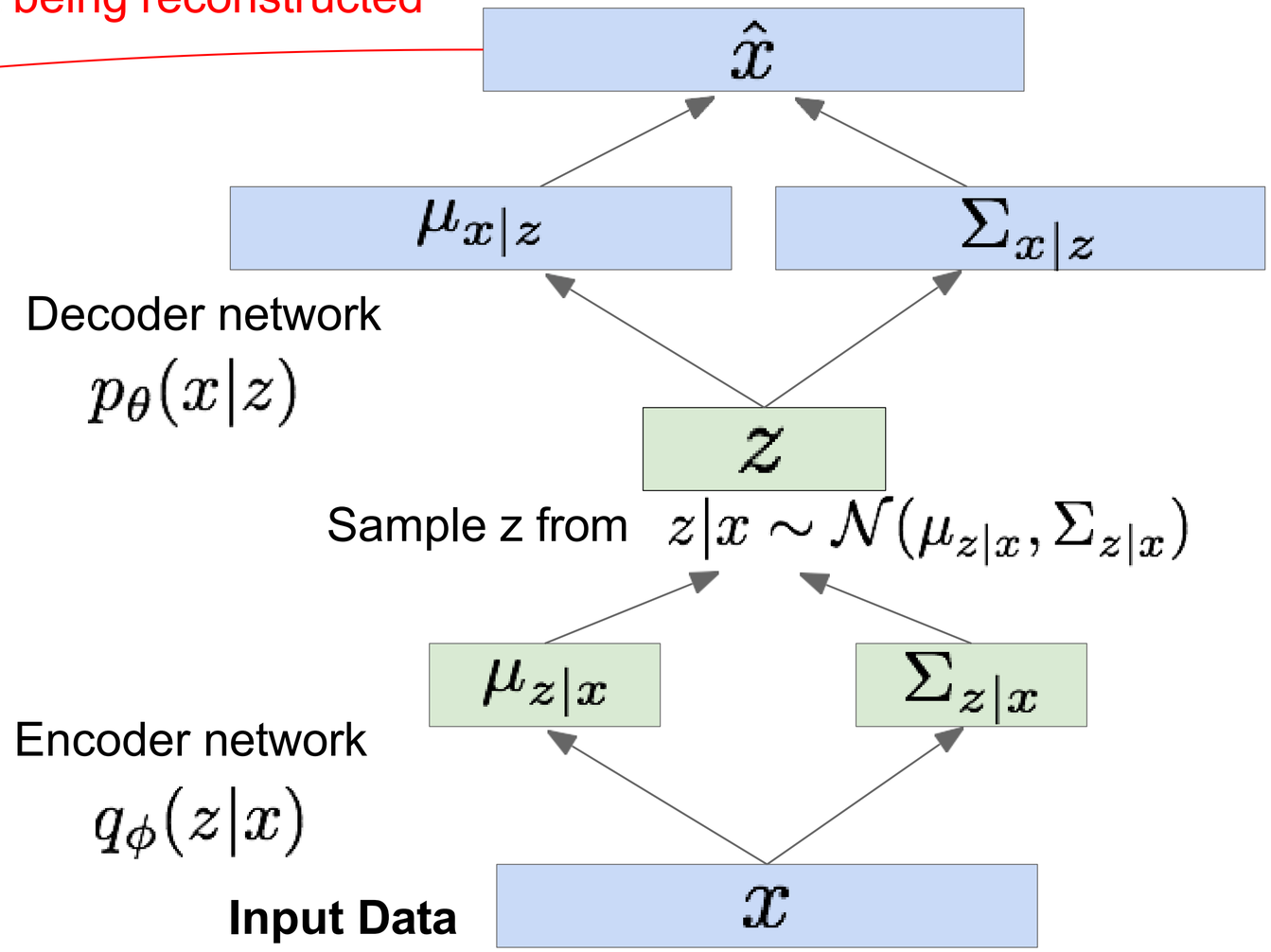


Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

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

Maximize likelihood of original input being reconstructed

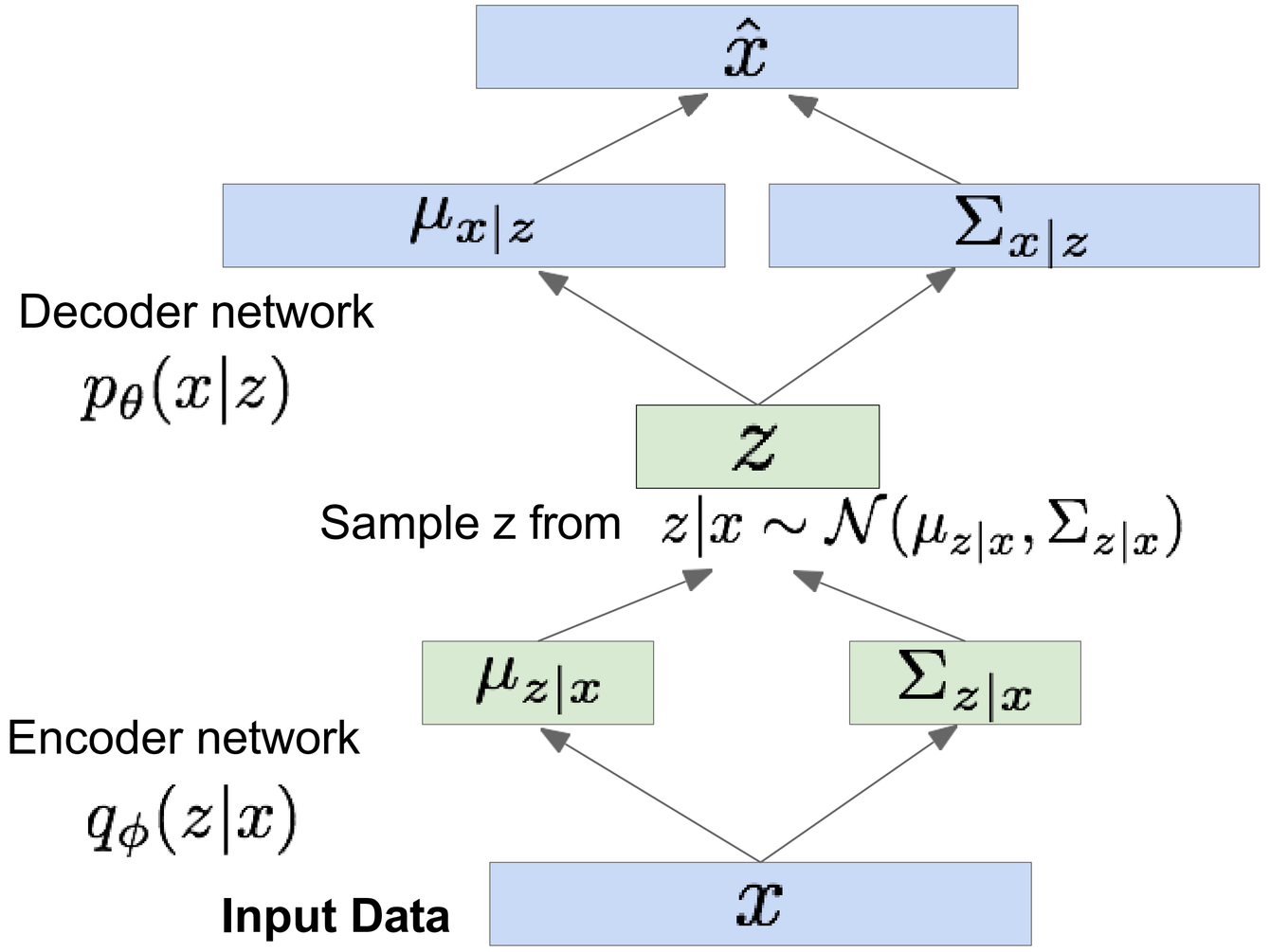


Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

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

For every minibatch of input data: compute this forward pass, and then backprop!



Variational Autoencoders: Generating Data!

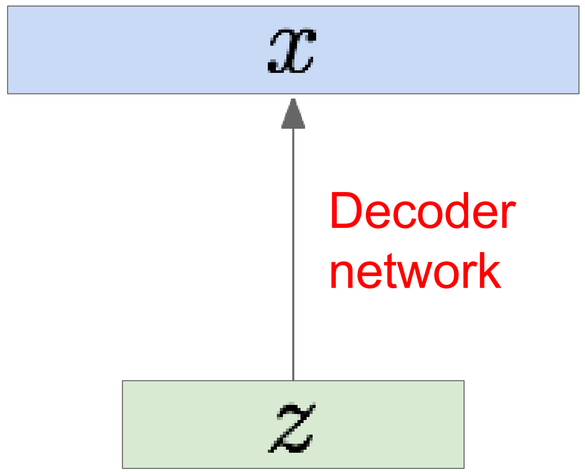
Our assumption about data generation process

Sample from true conditional

$p_{\theta^*}(x \mid z^{(i)})$

Sample from true prior

$z^{(i)} \sim p_{\theta^*}(z)$



Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014

Variational Autoencoders: Generating Data!

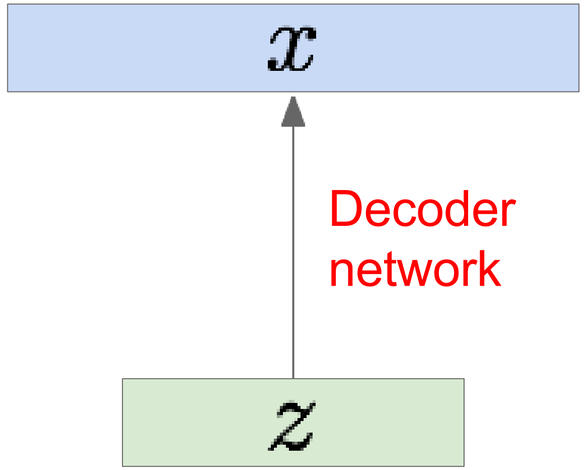
Our assumption about data generation process

Sample from true conditional

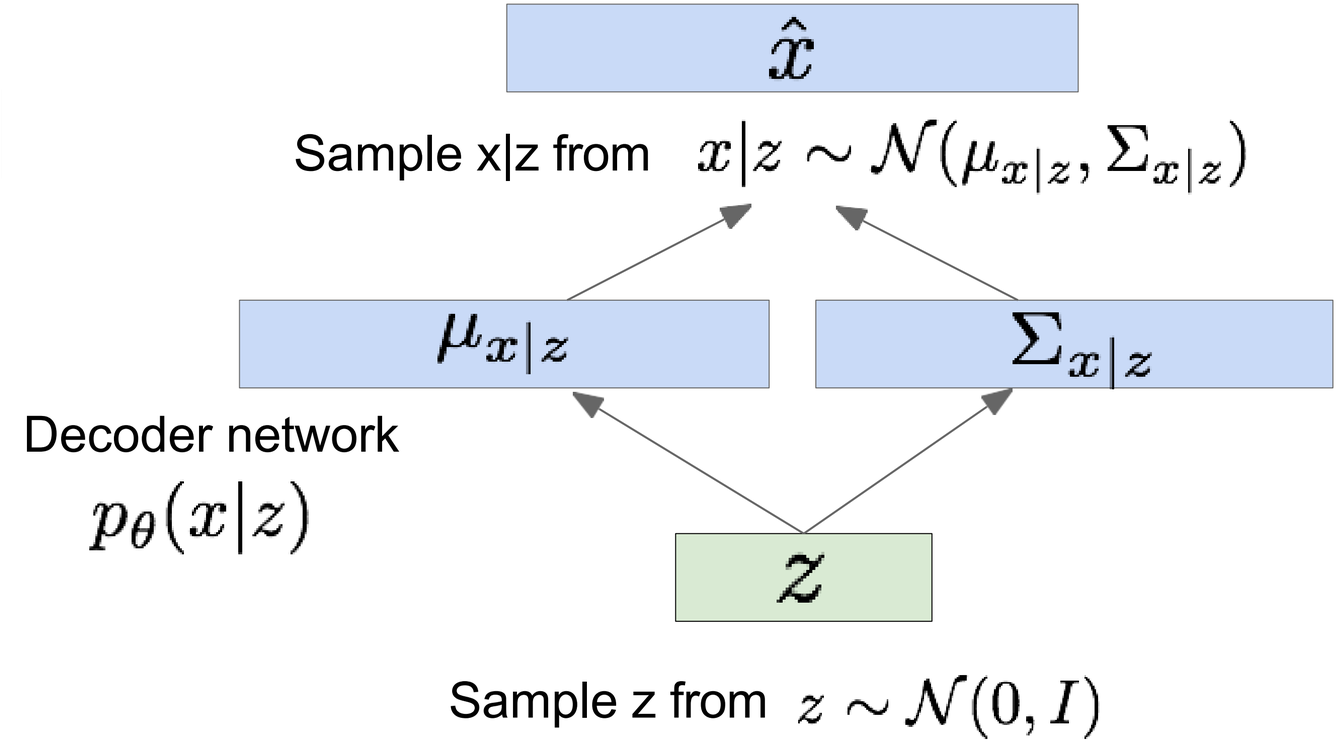
$p_{\theta^*}(x \mid z^{(i)})$

Sample from true prior

$z^{(i)} \sim p_{\theta^*}(z)$



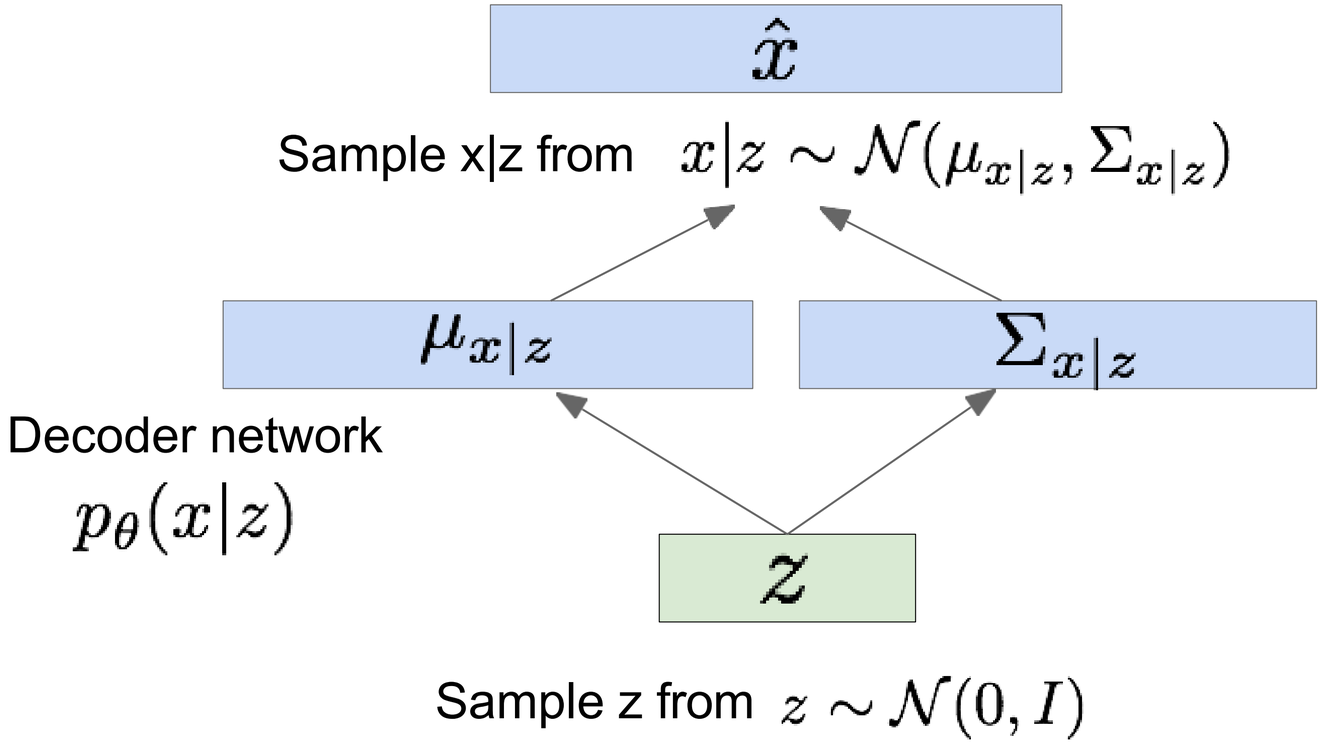
Now given a trained VAE:
use decoder network & sample z from prior!

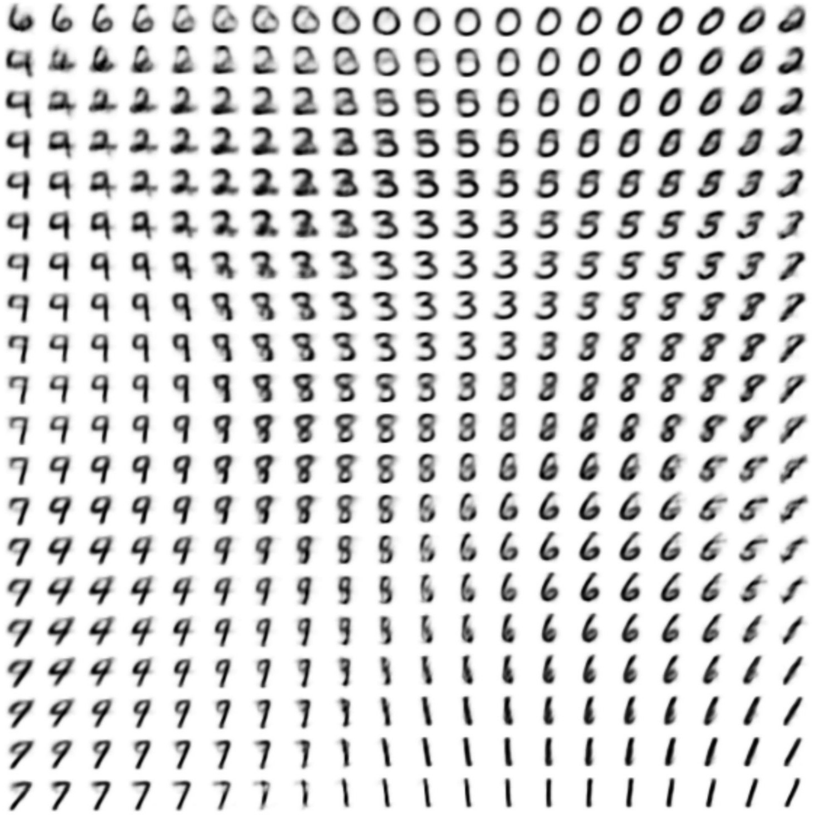


Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014

Variational Autoencoders: Generating Data!

Use decoder network. Now sample z from prior!

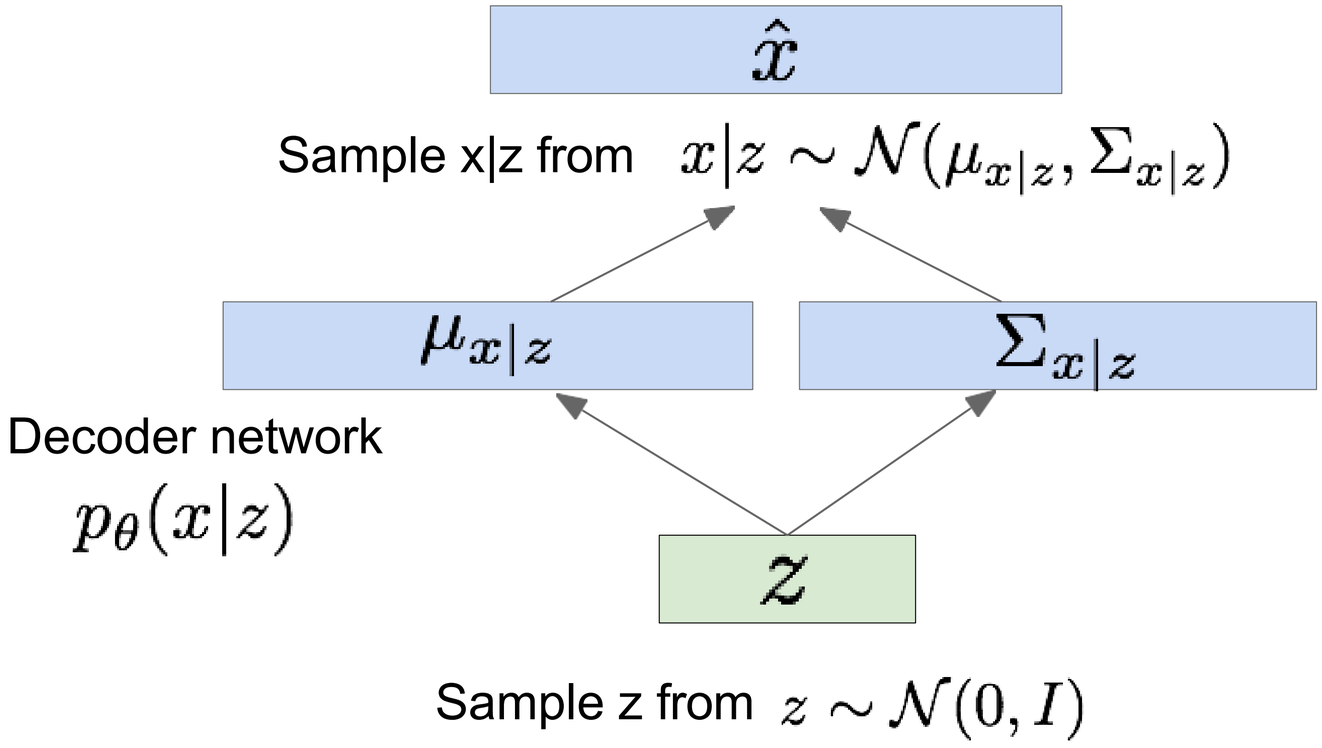




Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014

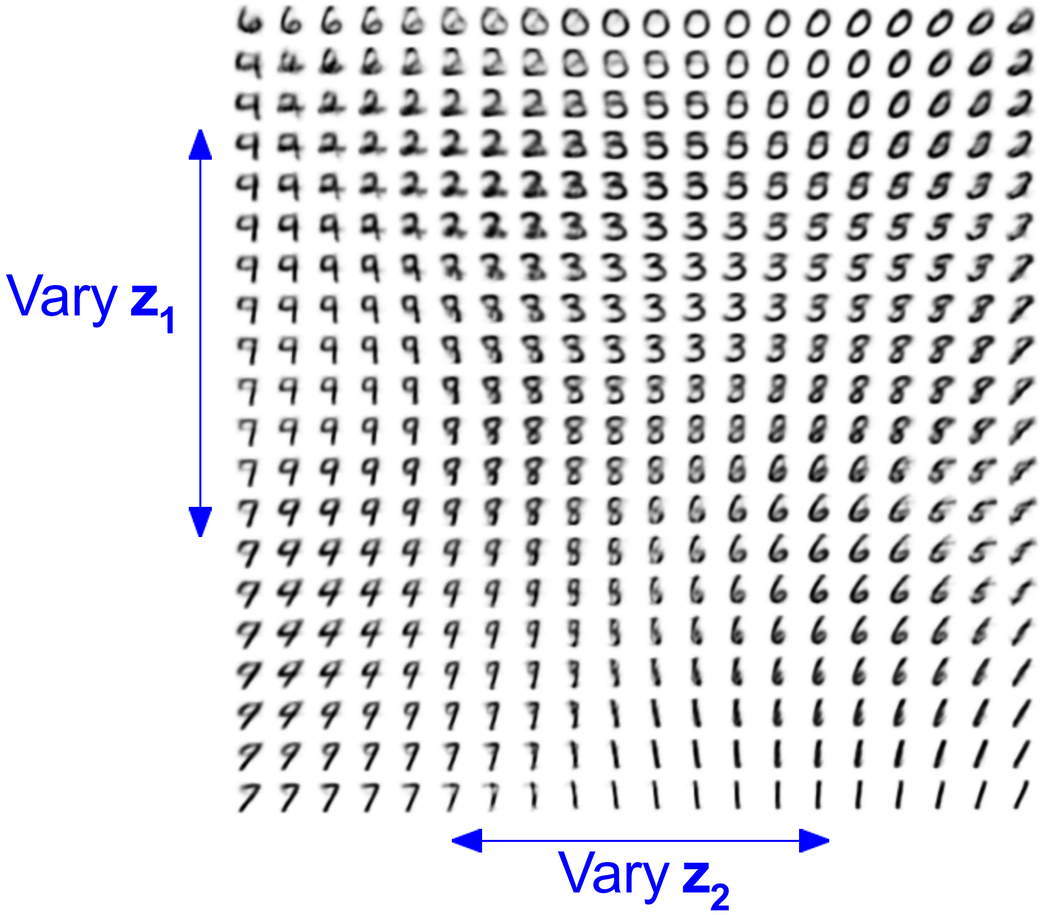
Variational Autoencoders: Generating Data!

Use decoder network. Now sample z from prior!



Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014

Data manifold for 2-d **z**



Variational Autoencoders: Generating Data!

Diagonal prior on **z**
=> independent
latent variables

Different
dimensions of **z**
encode
interpretable factors
of variation

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014

Variational Autoencoders: Generating Data!

Diagonal prior on **z**
=> independent
latent variables

Different
dimensions of **z**
encode
interpretable factors
of variation

Also good feature representation that can be computed using $q_\phi(z|x)$!

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014

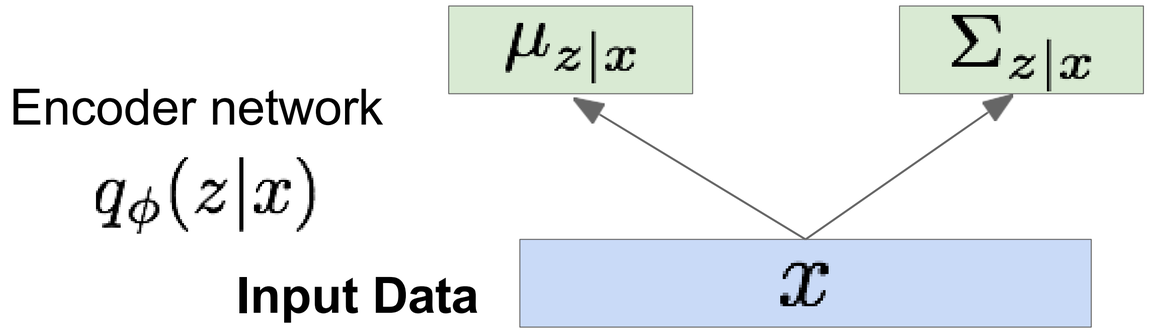
Variational Autoencoders: Generating Data!

32x32 CIFAR-10

Labeled Faces in the Wild

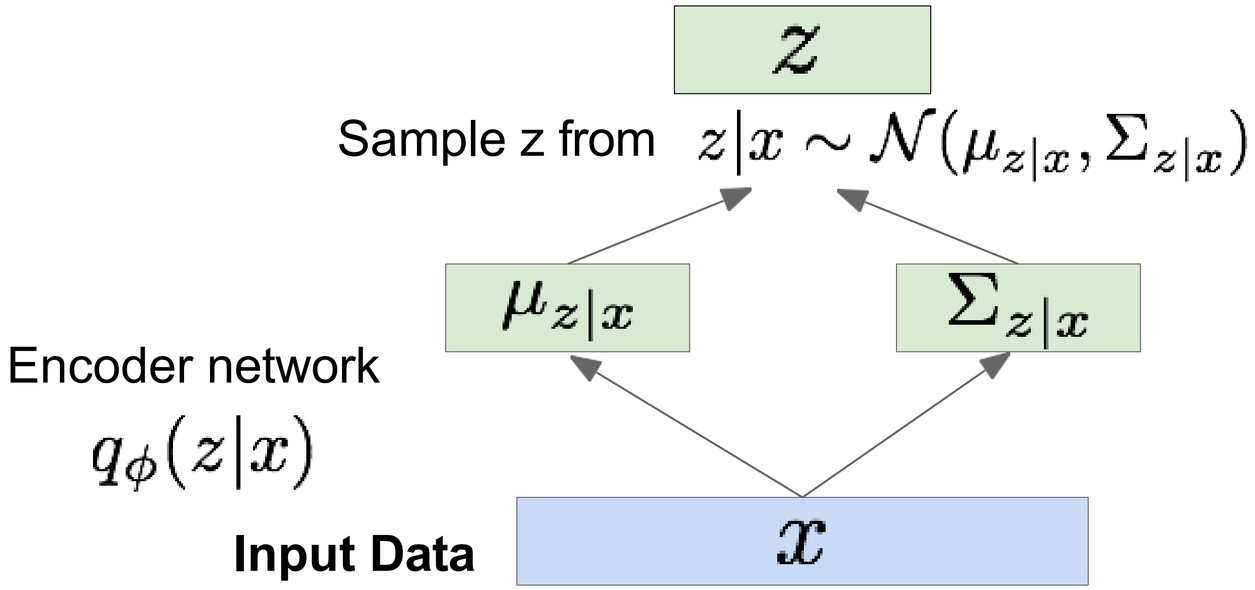
Editing images with VAEs

1. Run input data through encoder to get a distribution over latent codes



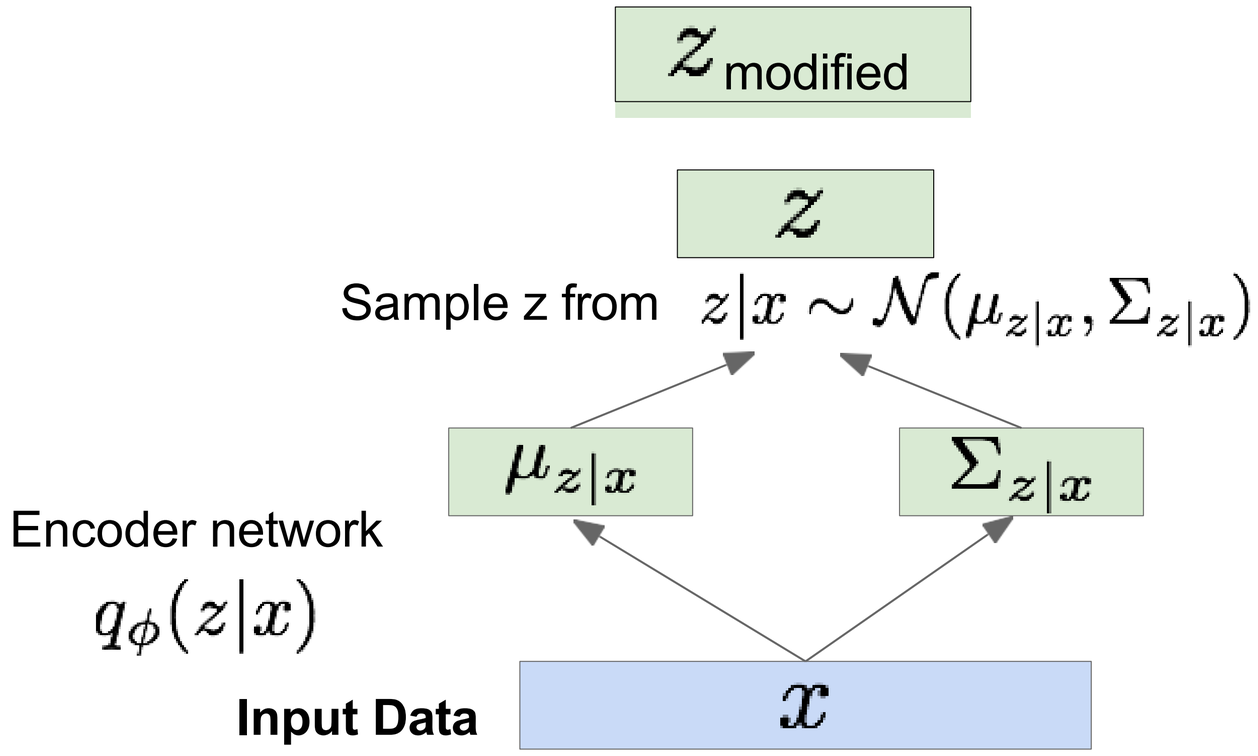
Editing images with VAEs

1. Run input data through encoder to get a distribution over latent codes
2. Sample code z from encoder output



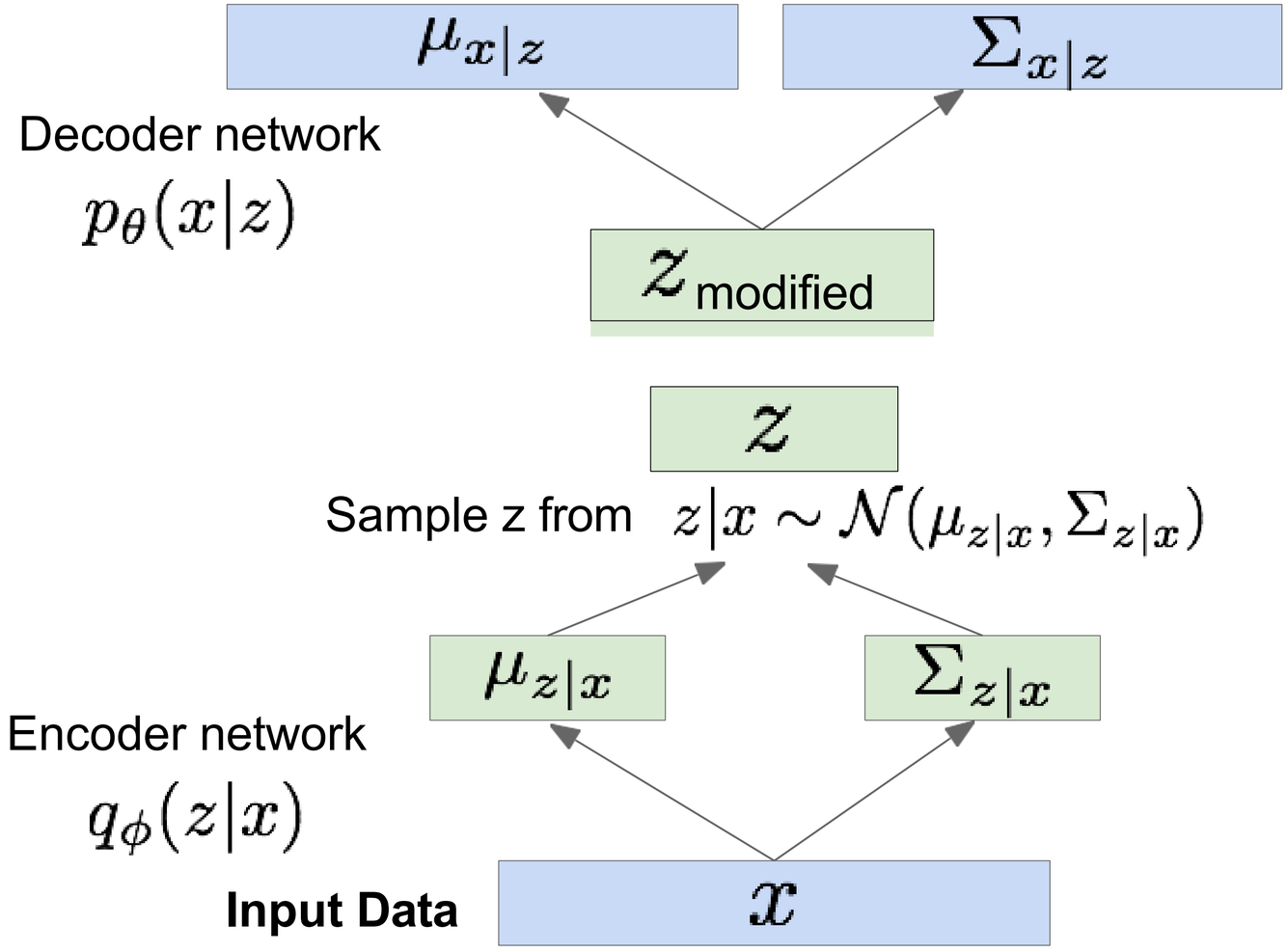
Editing images with VAEs

1. Run input data through encoder to get a distribution over latent codes
2. Sample code z from encoder output
3. Modify some dimensions of sampled code



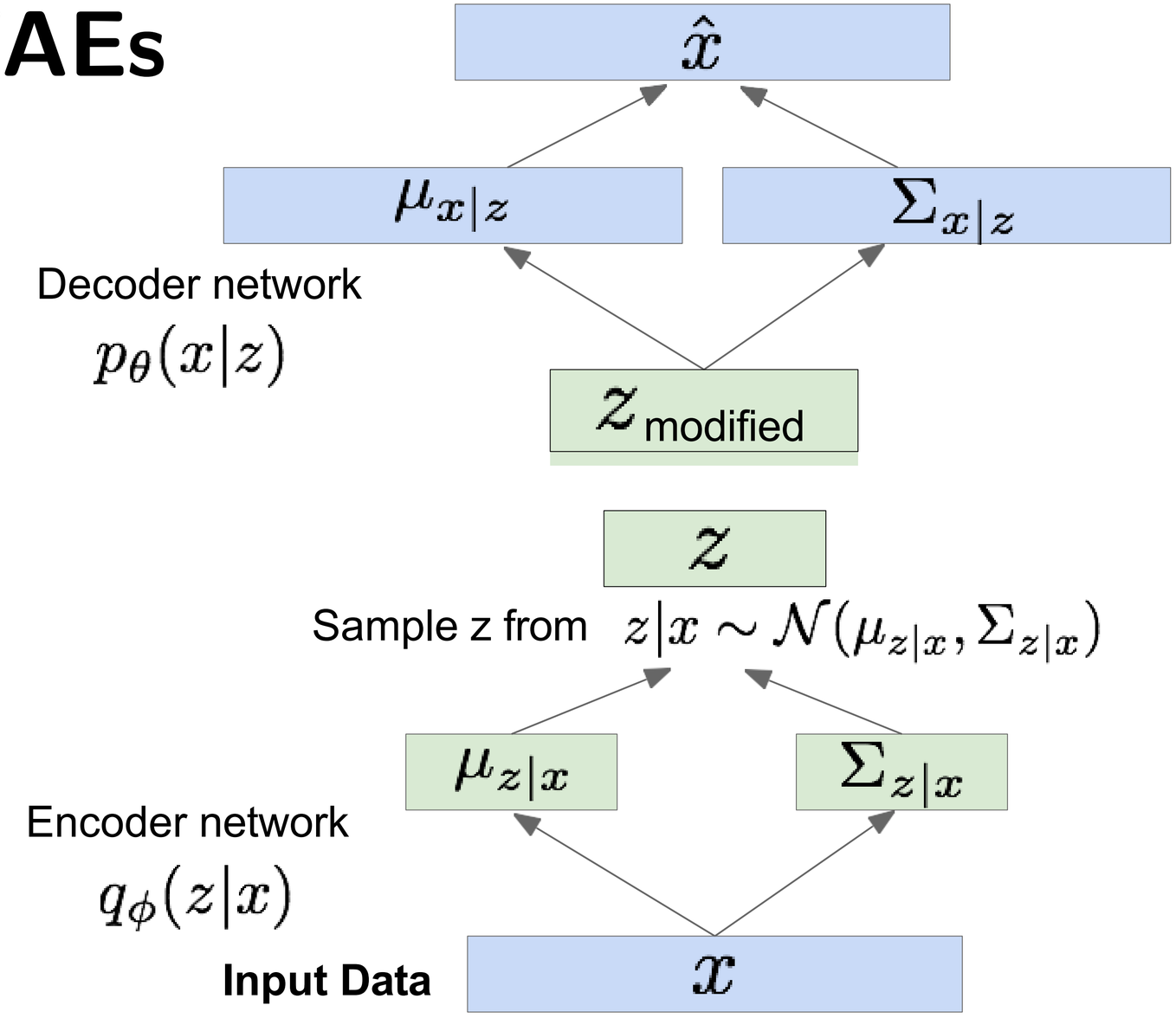
Editing images with VAEs

1. Run input data through encoder to get a distribution over latent codes
2. Sample code z from encoder output
3. Modify some dimensions of sampled code
4. Run modified z through decoder to get a distribution over data sample



Editing images with VAEs

1. Run input data through encoder to get a distribution over latent codes
2. Sample code z from encoder output
3. Modify some dimensions of sampled code
4. Run modified z through decoder to get a distribution over data sample
5. Sample new data from (4)



Editing images with VAEs