Learning Without Labels

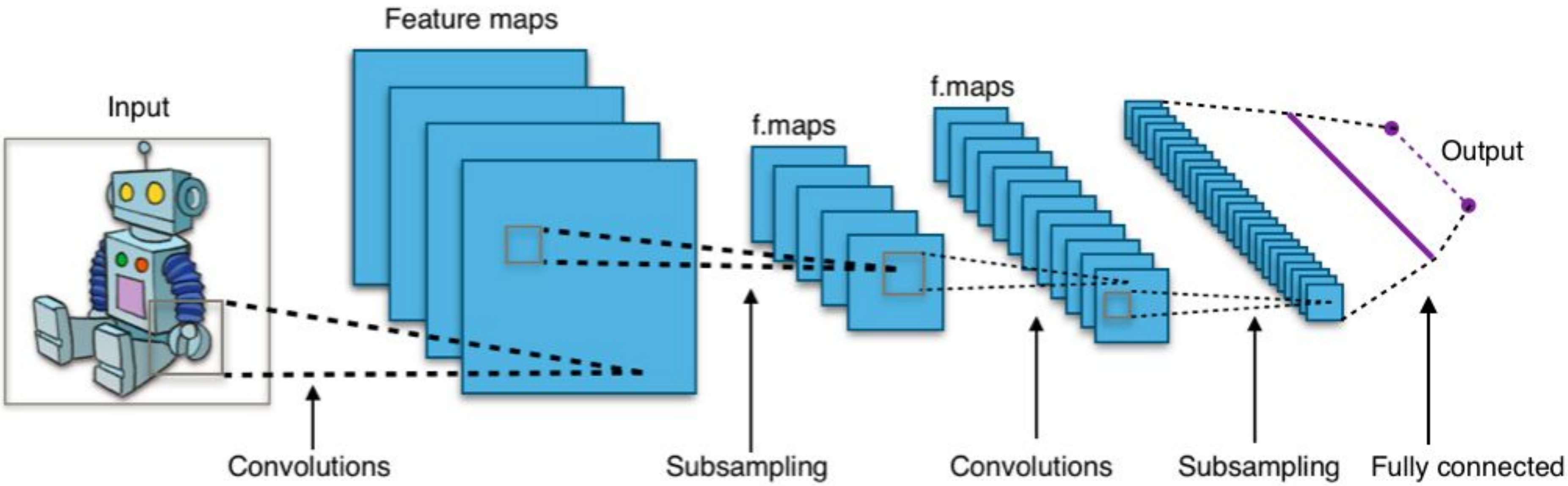
Unsupervised and Weakly Supervised Learning of Deep Models

Presented by Dr. Shazia Akbar
shazia@altislabs.com

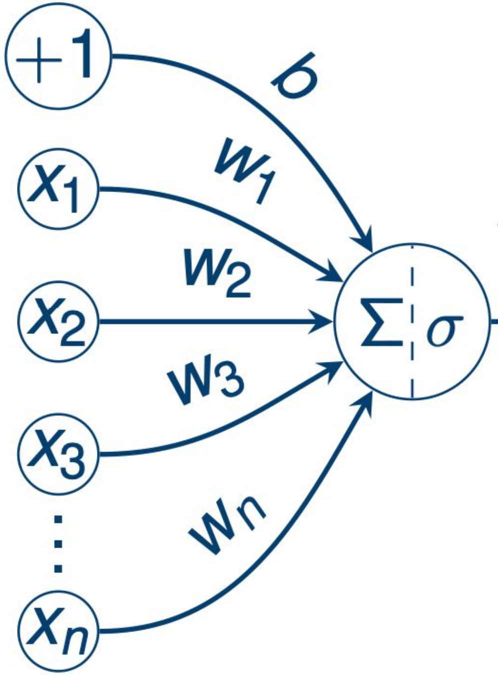
Outline

- Deep learning
- Types of learning
- Unsupervised learning
- Summary
- Conclusion

Deep learning

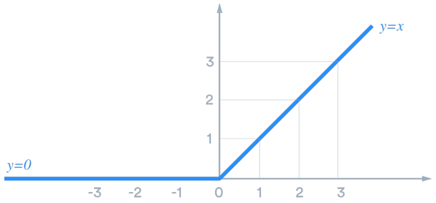


A Neuron



$$h(\vec{x};\vec{w}) = \sigma\left(b + \sum_{i=1}^{n} w_i x_i\right)$$

Activation function



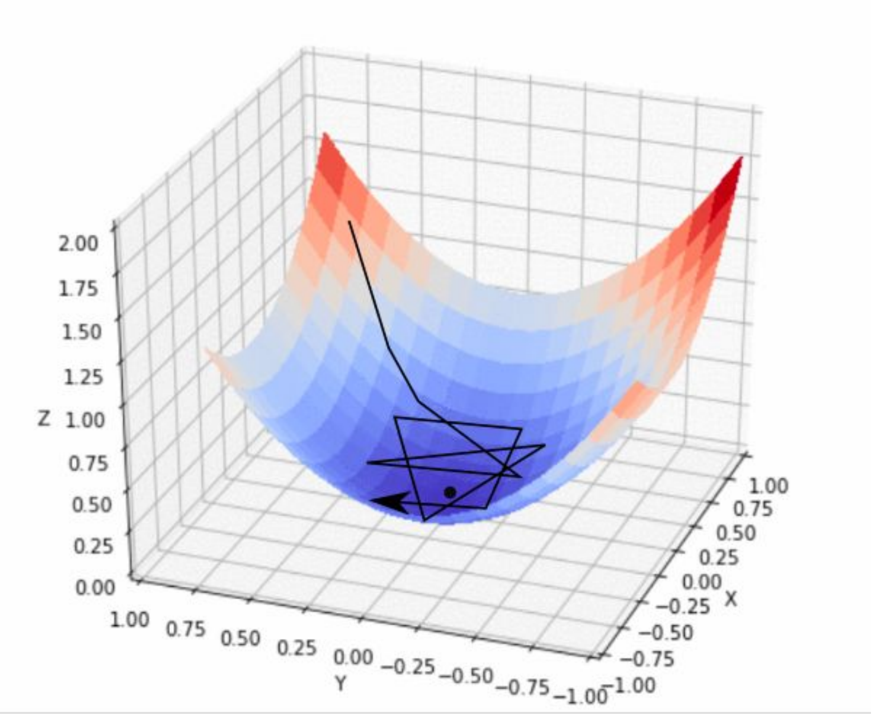
Training a CNN (PyTorch)

```
def train()
    for epoch in range(num_epochs):
        for i, data in enumerate(trainloader, 0):
            inputs, labels = data                         # get batch
            optimizer.zero_grad()
            outputs = net(inputs)                         # forward pass
            loss = criterion(outputs, labels)             # compute loss
            loss.backward()                               # backward pass
            optimizer.step()
```

Updating Gradients

Because of large search space, we need a function to navigate cost towards minimum error

In supervision, need ground truth labels to measure error



Updating Gradients

Stochastic gradient descent:

Updates weights/parameters as

$$\theta = \theta - \underbrace{\alpha}_{\textit{alpha}} \nabla_\alpha \underbrace{J(\theta; x^{(i)}, y^{(i)})}_{\textit{loss function}}$$

Loss Function

Loss function typically needs two inputs:

```
loss = criterion(outputs, labels)        # compute loss
```

For example, widely used categorical cross entropy:

$$-\sum_{c=1}^{M} y_c \log(p_c)$$

Cases when *y* is difficult to gather...

Discovering new biological changes/characteristics to treat diseases

- Knowledge is currently unknown
- Medical expertise is expensive and subjective
- Want to gather this information before death

Anomaly detection

- Definition of abnormal is anything “not normal”

Is a Jaffa Cake a biscuit or a cake?

So why use deep models?

Superior performance. We want to leverage this!

Requires little to no domain knowledge (discovery is doable)

- Learn features automatically

Active field

Types of Learning

Supervised

Unsupervised

Weakly supervised

Types of Learning

Supervised

Unsupervised

Weakly supervised

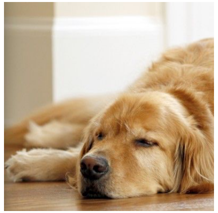

cat


cat


cat


dog


dog


dog


horse


horse


horse

Types of Learning

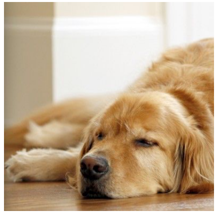
Supervised

Unsupervised

Weakly supervised





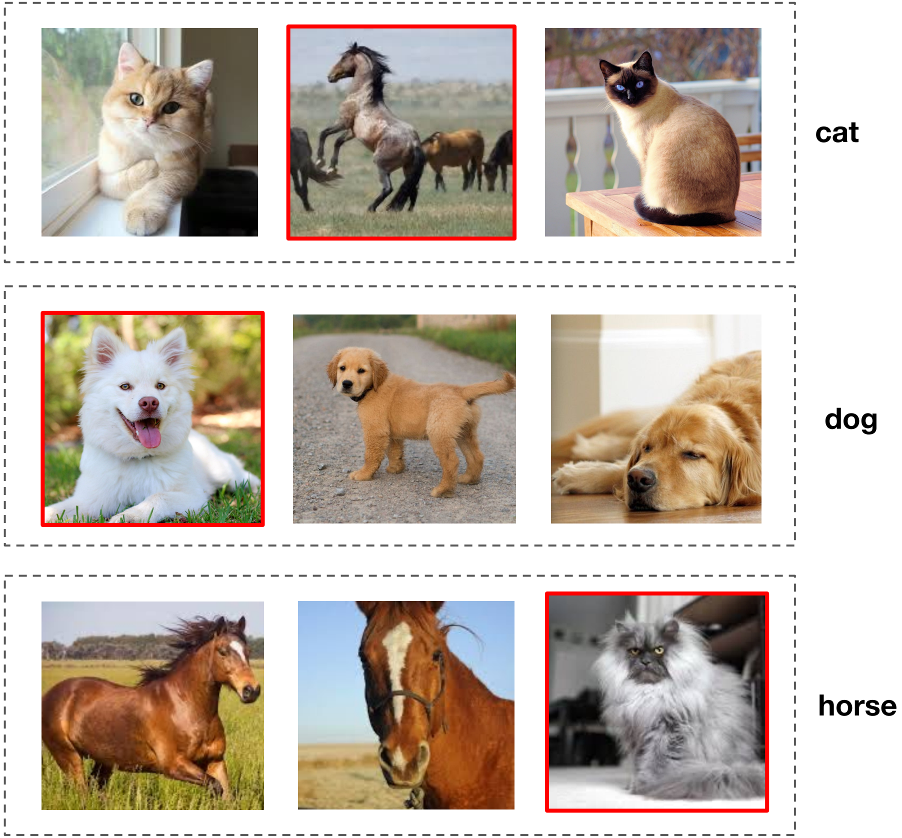




Types of Learning

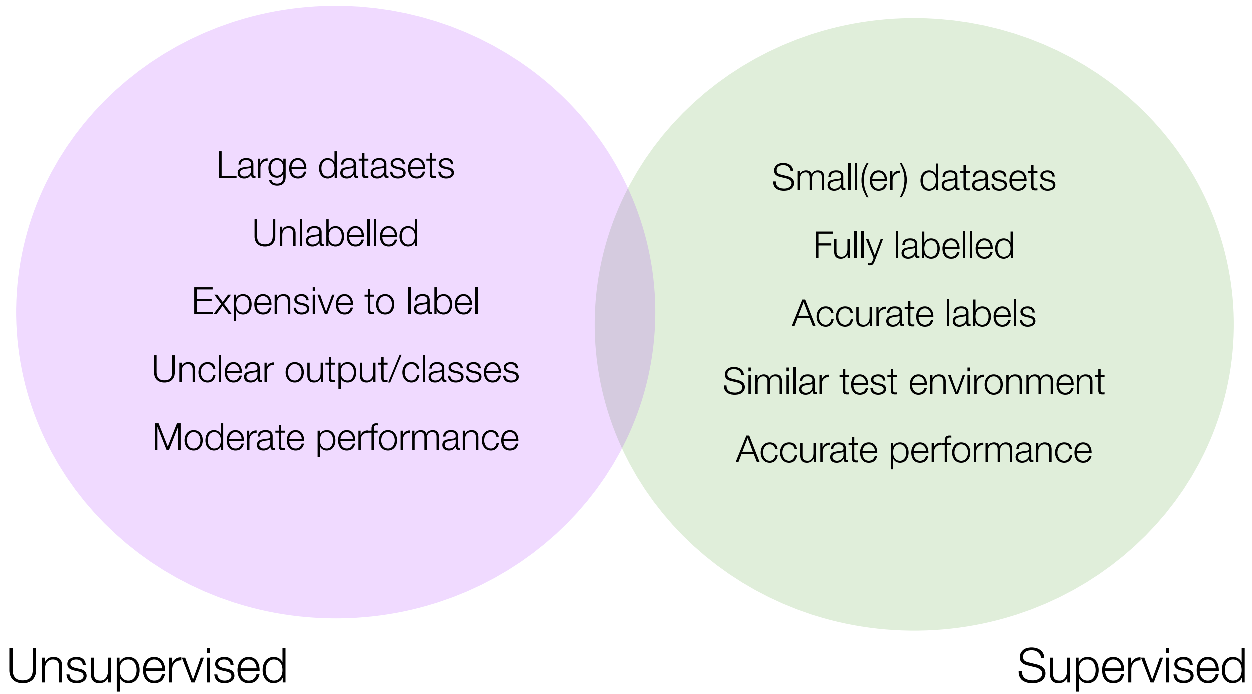
Supervised

Unsupervised

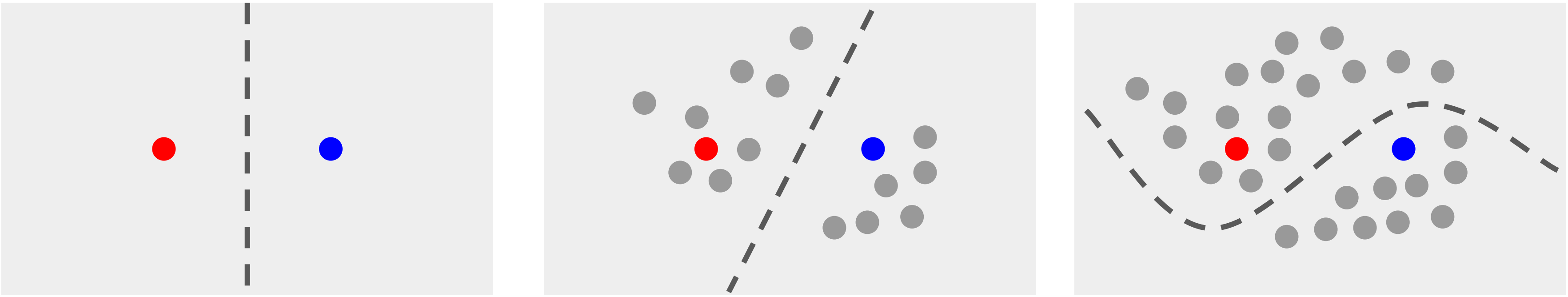
Weakly supervised



When to use what...



When unlabeled data can help...



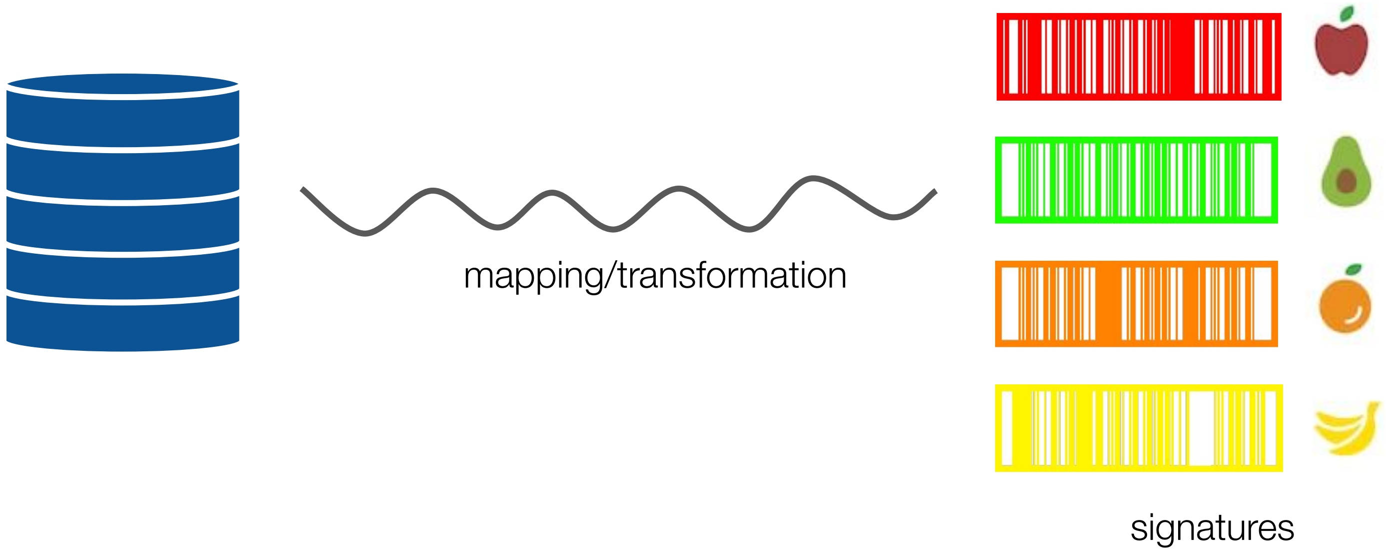
Some labeled data | Some more unlabeled data | Even more unlabeled data

Approaches to Unsupervised Learning

1. Data compression
2. Generative modeling
    a. Disentanglement
3. Clustering
4. (Predictive Networks)

Data embedding

Projecting high-dimensional space into a low space, whilst preserving “important” information



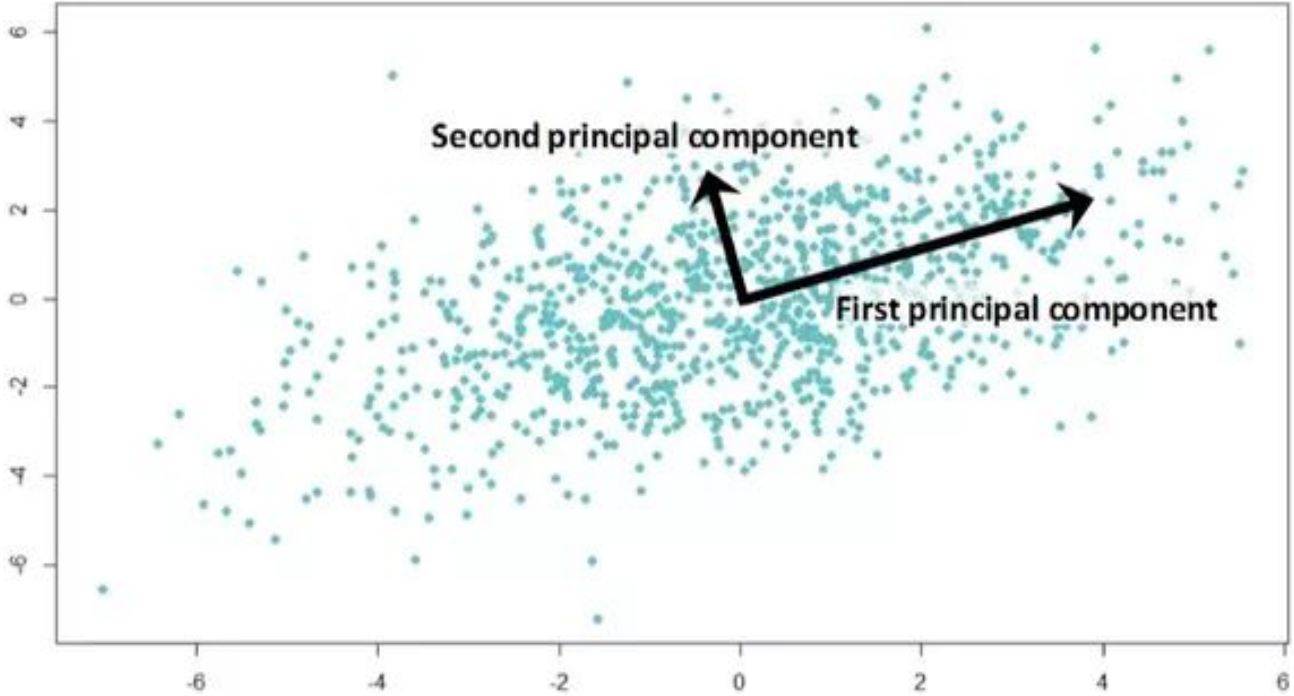
PCA

Projecting *n*-dimensional input data to *m* orthogonal axes

Preserve most important information

- Always choose one of the axes to have high variance (principal component)
- Constrain other ones to be orthogonal
    - Linearly independent variables

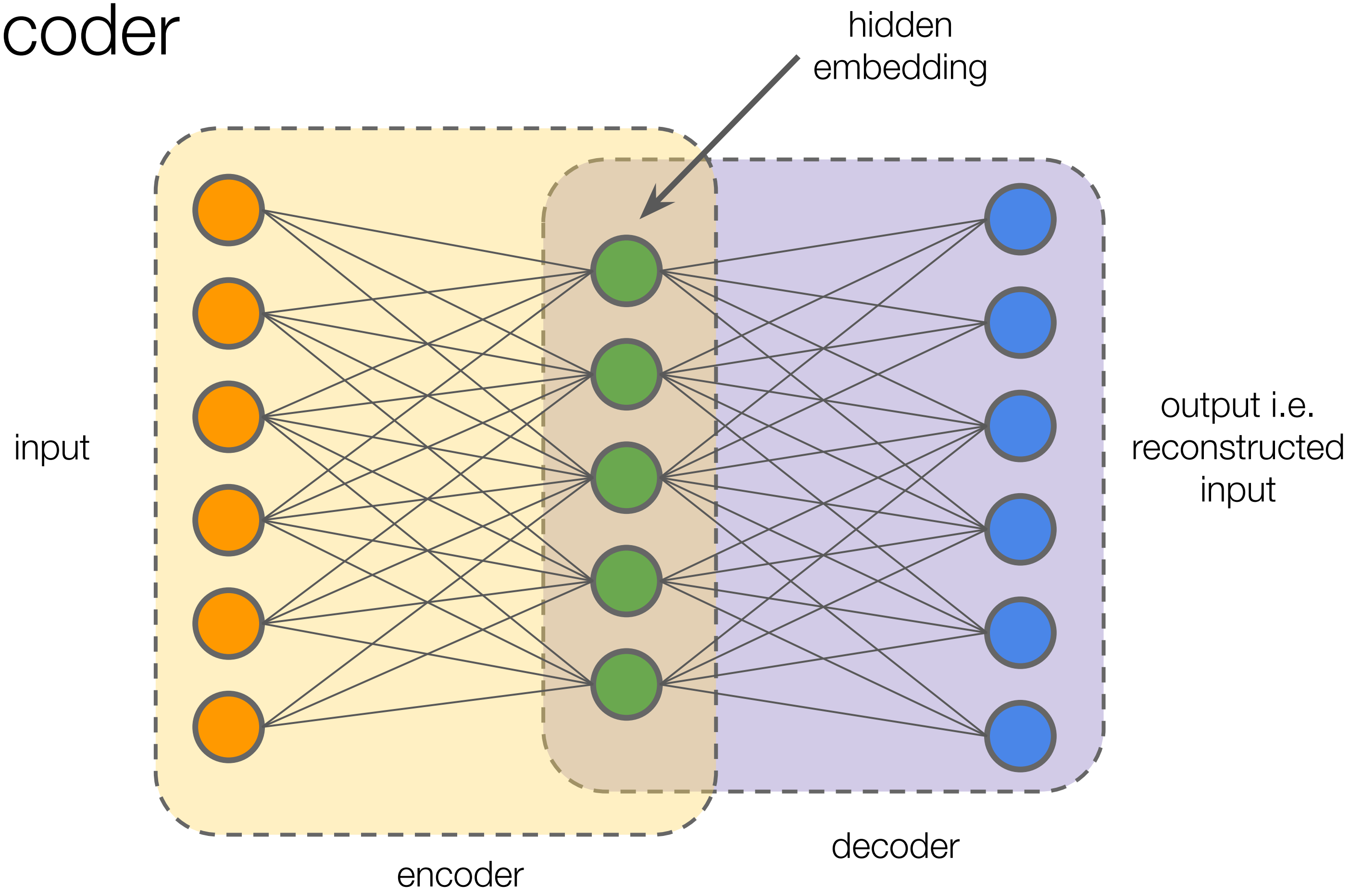
PCA



https://medium.com/analytics-vidhya/principal-component-analysis-pca-with-code-on-mnist-dataset-da7de0d07c22

Deep Unsupervised Learning

Autoencoder



Autoencoder

Cost = reconstruction error so we only need data!

$$\mathcal{L}(\vec{x}') = \|\vec{x}' - \vec{x}\|^2$$

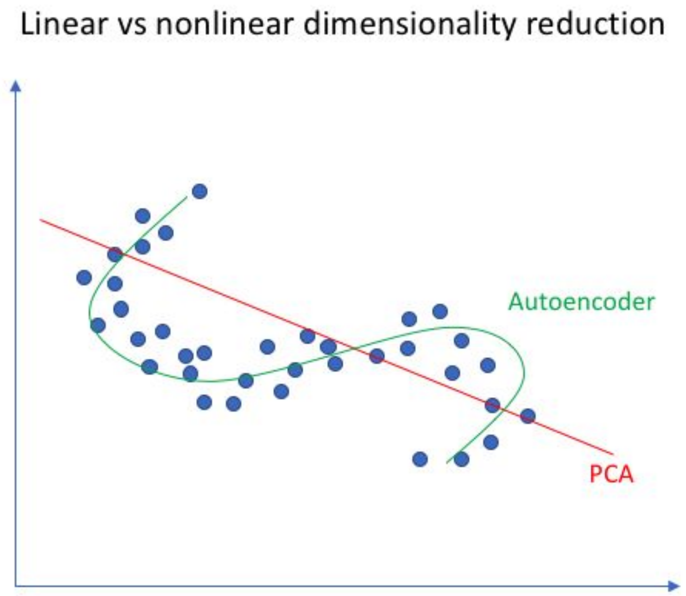
Learned in a fully automated manner but very similar to PCA

By adding more layers, we can introduce depth and non-linearity...

Autoencoder

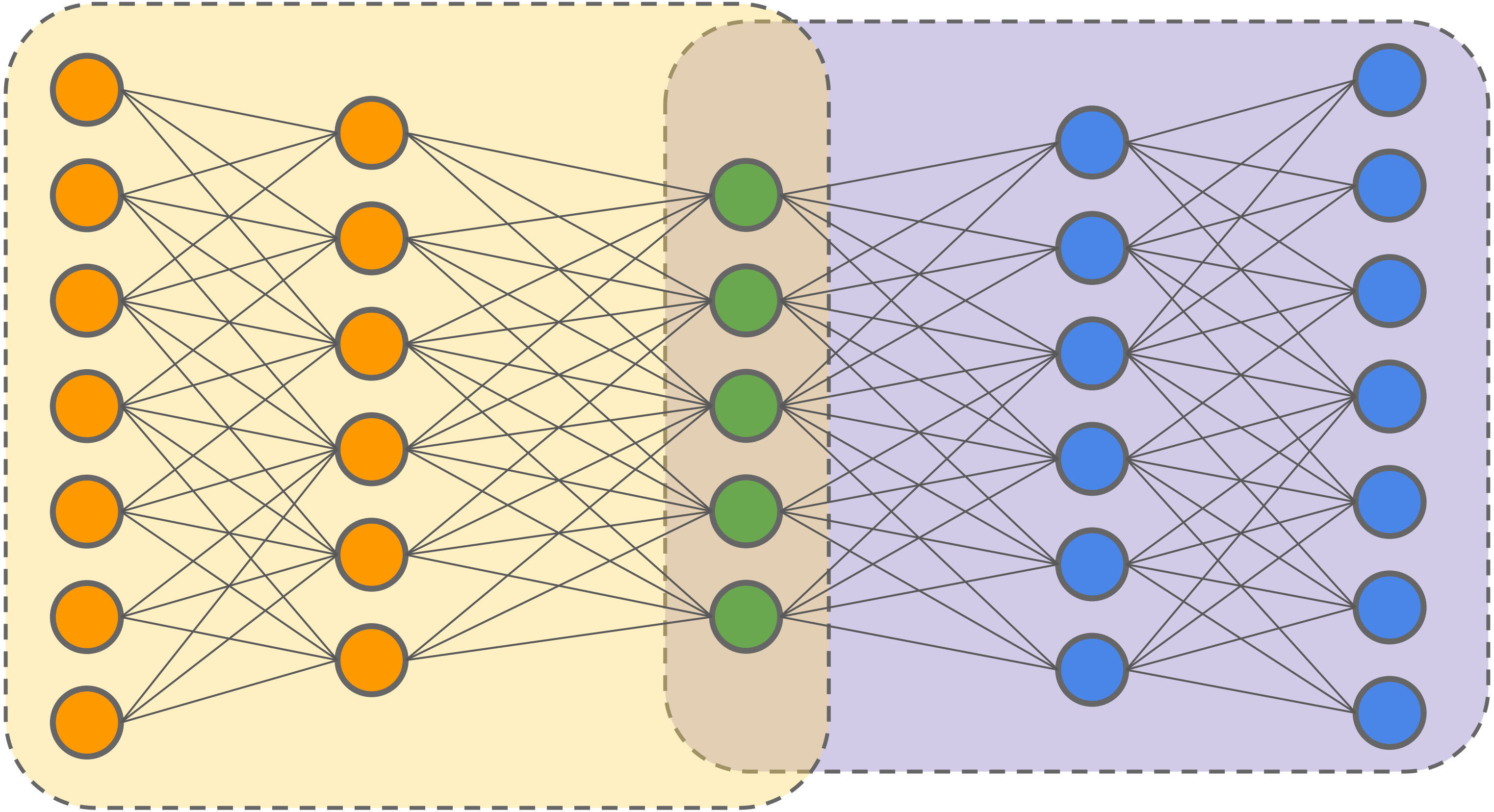
The difference between PCA and AEs is that AEs are capable of learning non-linear manifolds

Stronger mapping function suitable for raw image data e.g.



Deep Autoencoders

+ **ReLU (conv)**
+ **Logistics sigmoid (output)**



Deep Autoencoder

Bring back categorical cross-entropy

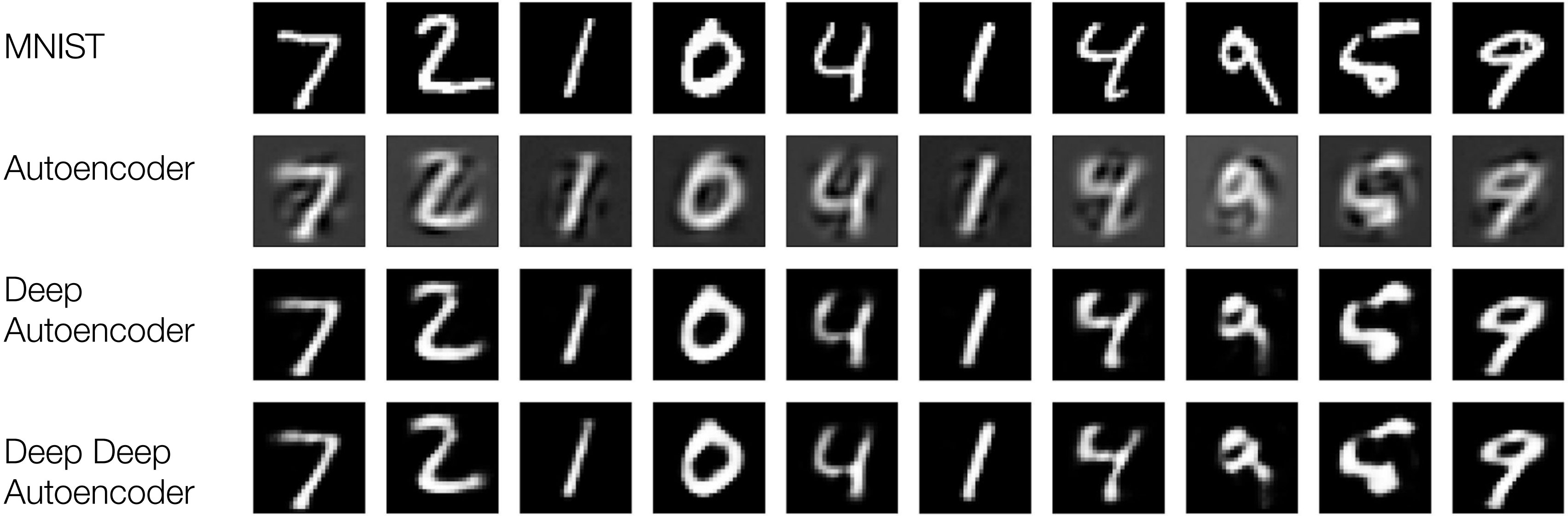
$$-\sum_{j=1}^{M}\sum_{i=1}^{N} y_{i,j}\log(p_{i,j})$$

$$-\sum_{i=1}^{N} x_i \log(p_i) + (1 - x_i)\log(1 - p_i)$$

reconstructed *x*

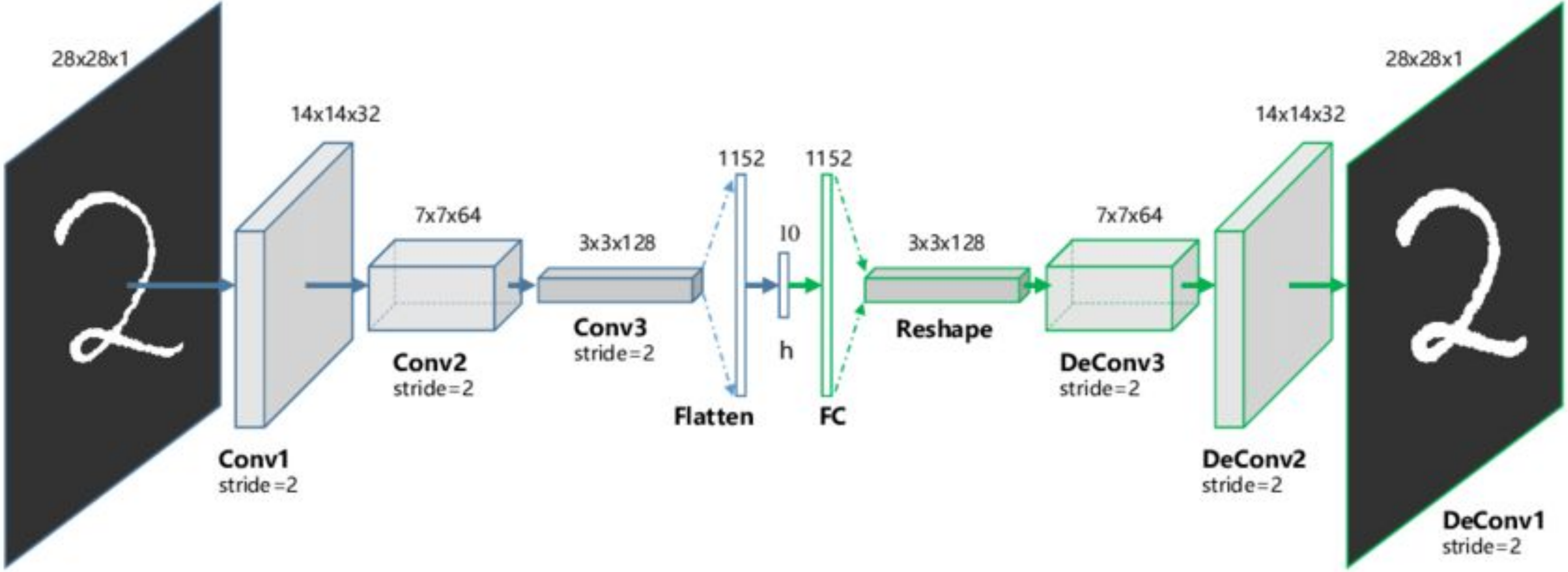
```python
criterion = nn.MSELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate, weight_decay=1e-5)

for epoch in range(num_epochs):
    for data in dataloader:
        img, _ = data
        ...
        # ===================forward=====================
        output = model(img)
        loss = criterion(output, img)
        # ===================backward====================
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
```

Comparing Autoencoders

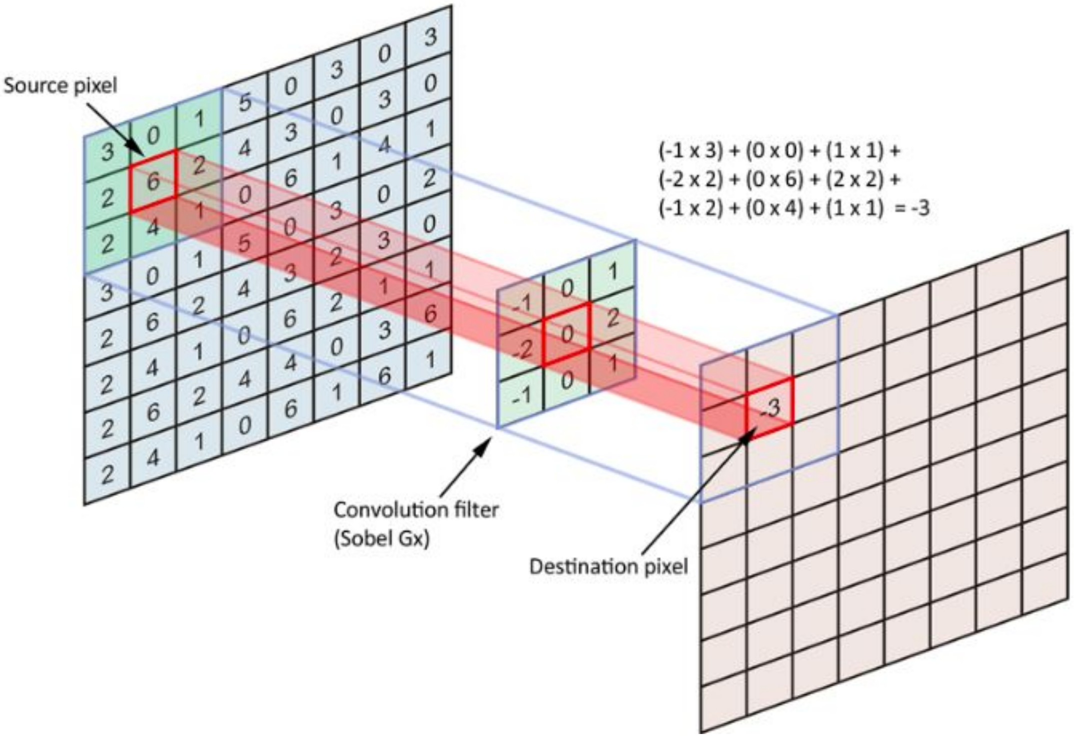


https://www.cl.cam.ac.uk/~pv273/slides/UCLSlides.pdf

Convolutional Autoencoders



Convolution Layers



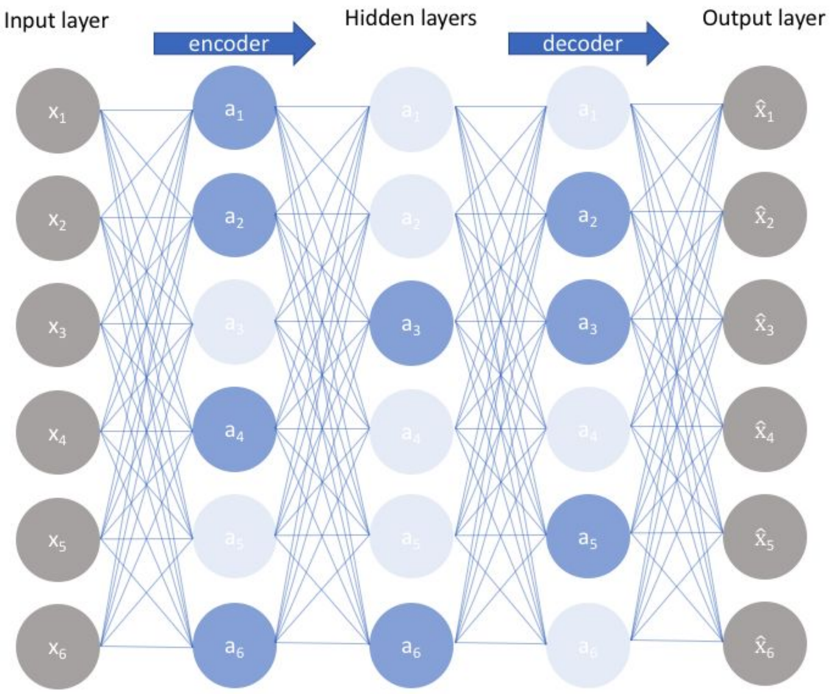
https://www.freecodecamp.org/news/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050/

Sparse Autoencoder

Forces the model to be sparse by switching off activations

Reduces chance of overfitting

Add another term in loss function to penalize excessive activations (L1, L2, KL)



Other applications of autoencoders

Denoising:

- Denoising Adverserial Autoencoders, Creswell and Bharath, https://arxiv.org/pdf/1703.01220.pdf

Image Inpainting

- Semantic Image Inpainting with Deep Generative Models, Yeh *et al,* CVPR 2017
- Context Encoders: Feature Learning by Inpainting, Pathak *et al*, CVPR 2016

Information Retrieval (hashing functions)

- Semantic Hashing, https://www.cs.utoronto.ca/~rsalakhu/papers/semantic_final.pdf

Problem with Autoencoders

Autoencoder is solely trained to encode and decode with as few loss as possible, no matter how the latent space is organised

- Prone to overfitting as a result
  - Particular deep and complex AEs
  - Be careful when modeling and training!

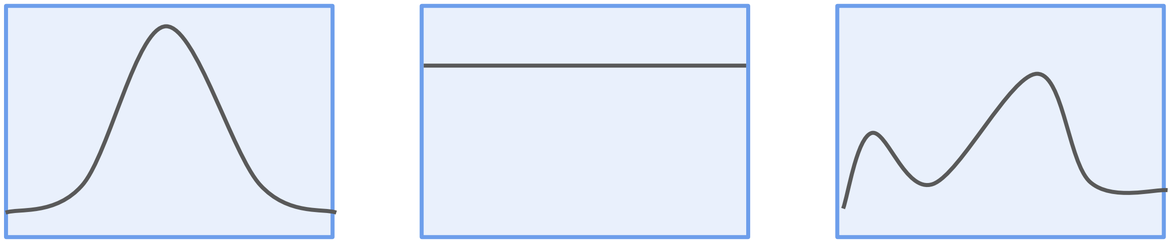
Sparse AEs help to mitigate this

But there is an alternative...

Generative Models

Generative models learn a probability distribution representative of the data itself

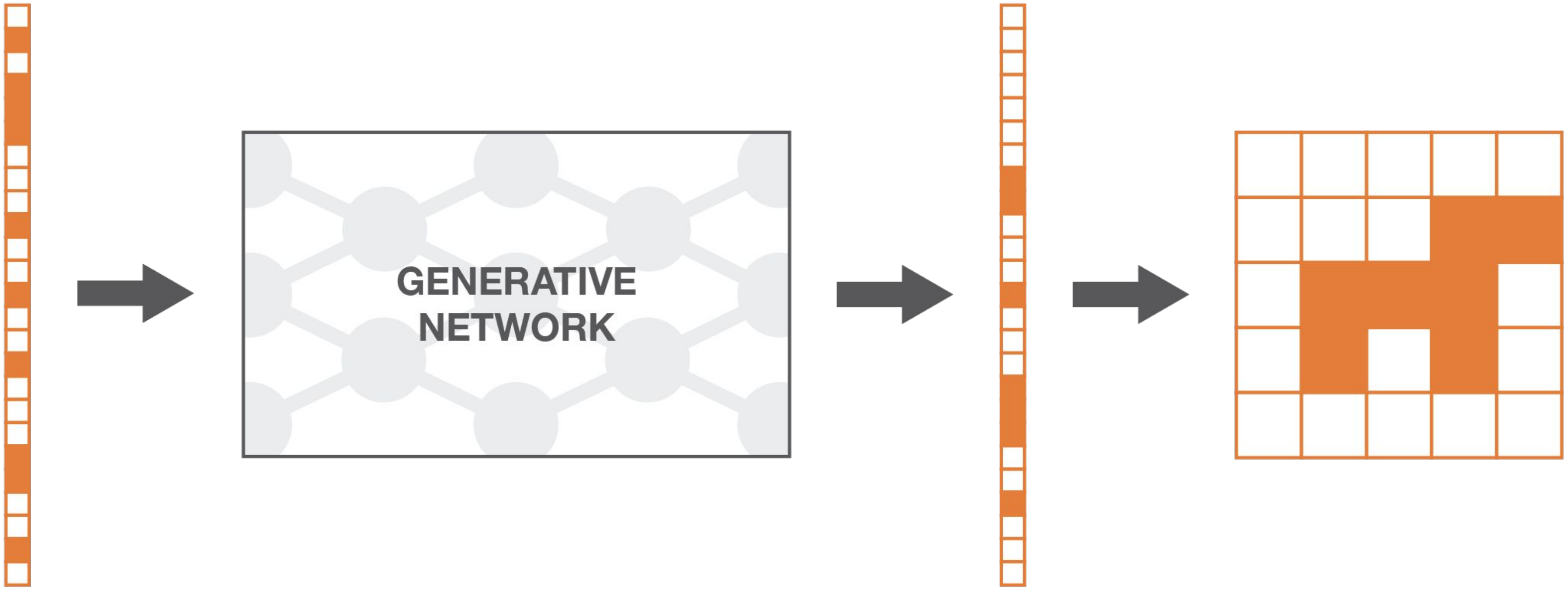
- Ability to generate new data points
- Forms of this probability distribution:



Gaussian   Uniform   Complex

Generative Models

When given a random variable, a well-calibrated generative model should be able to recreate a new data point



Variational Autoencoder

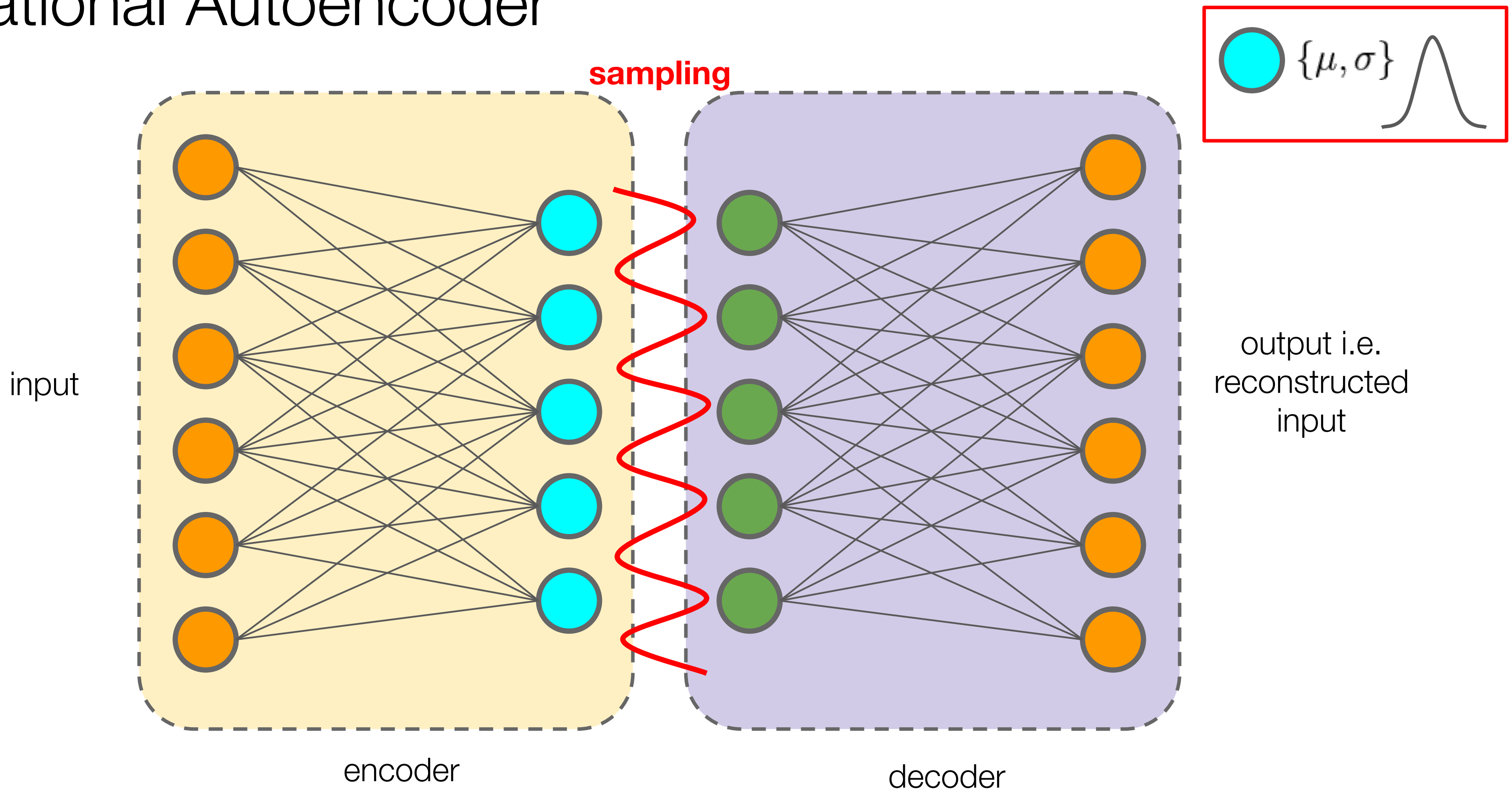
Similar to autoencoder but learns a latent space with some degree of variation - ideal for generating samples!

Latent space is composed of Gaussian representations

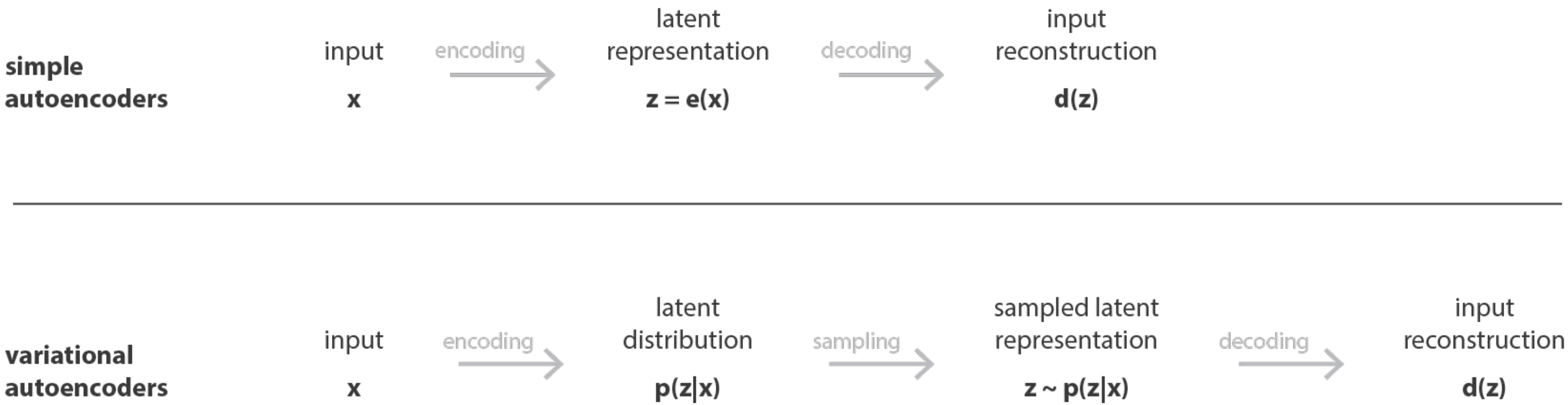
Great for highly variable data which is not fully captured in training set

So how does it work...

Variational Autoencoder



Variational Autoencoder



Loss function in VAE

Made up off two components:

$$\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z}))$$

Negative log-likelihood **Regularizer**

Also known as ELBO

Loss function in VAE

Kullback–Leibler divergence

- Measures how much two probability distribution diverge from one another
- Assuming a Gaussian distribution

True distribution

$$D_{KL}(p||q) = \sum_{i=1}^{N} p(x_i) log(\frac{p(x_i)}{q(x_i)})$$

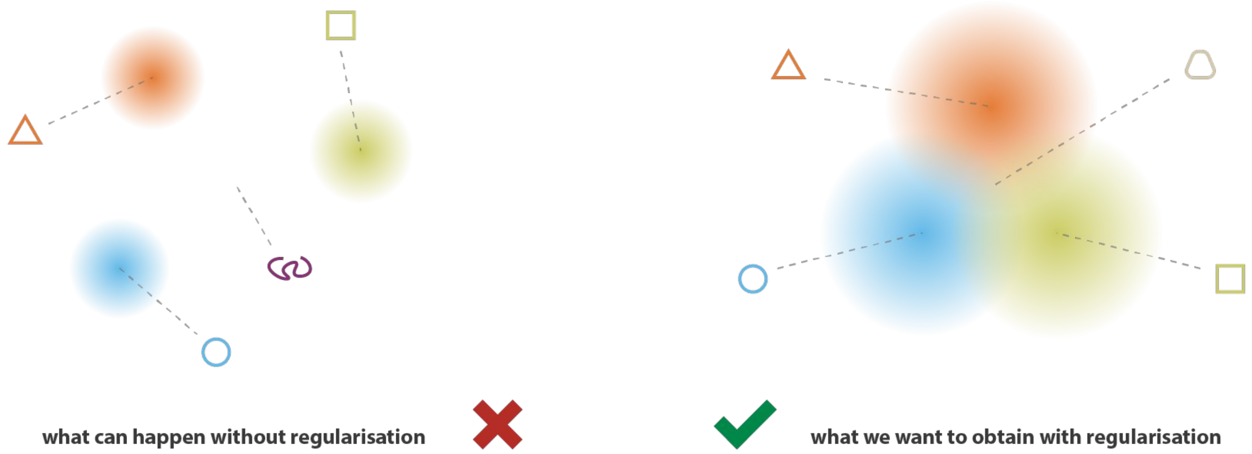
Approximate distribution

$\downarrow$ KL = better match between two distribution

Loss function in VAE

Penalizes clusters which falls away from centre of latent space

- Terrible for clustering, but great for generative modeling



https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

Disentanglement

“if you’re modeling pictures of people, then someone’s clothing is independent of their height, whereas the length of their left leg is strongly dependent on the length of their right leg. The goal of disentangled features can be most easily understood as wanting to use each dimension of your latent z code to encode one and only one of these underlying independent factors of variation.”

https://towardsdatascience.com/what-a-disentangled-net-we-weave-representation-learning-in-vaes-pt-1-9e5dbc205bd1

Disentanglement

Benefits:

- You can test your models whilst varying one feature
  - E.g. Driving simulations: change the weather conditions and subsequently test how well out self-driving car can adapt
- Can adapt our models to only change one property
  - E.g. Recreate another person who is taller

Beta VAE

Enforcing a higher weight on our VAE regularizer

$$\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \boxed{\beta} D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z}))$$

When Beta = 1, same as VAE

When Beta > 1, limit the representation capacity of latent space (z)

Other VAEs

VQ-VAE and VQ-VAE-2

- Vector Quantised-Variational AutoEncoder; van den Oord, et al. 2017
- Latent space = latent discrete codebook
- Codebook can be of any length and “height”

TD-VAE

- Temporal Difference VAE; Gregor et al., 2019
- Works with sequential data
- Based on Markov Chain Model

GANs

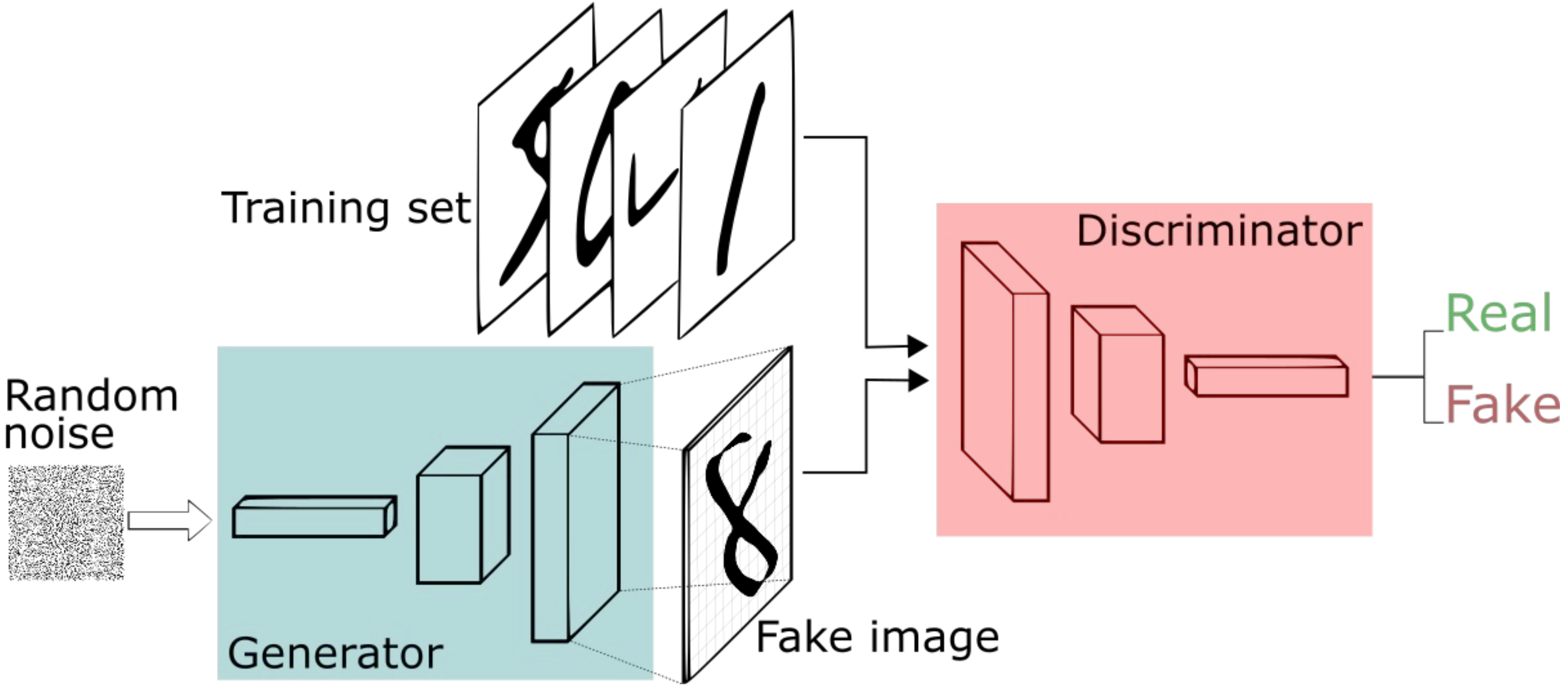
Fake or Real? (labels)

By imposing this restriction (in the discriminator) we can build an architecture which can recreate images which are lifelike.

Two components:

1. Generator: creates new images based on knowledge learned in NN
2. Discriminator: Predicts which images are generated/real

GANs

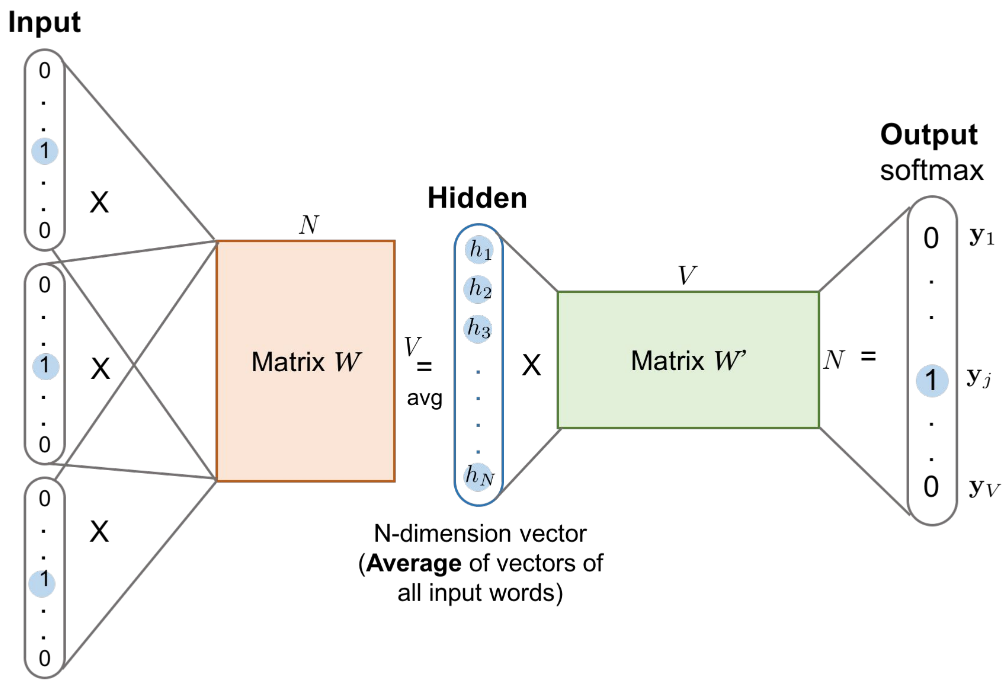

Training set
Discriminator
Real
Fake
Random noise
Generator
Fake image

What about text?

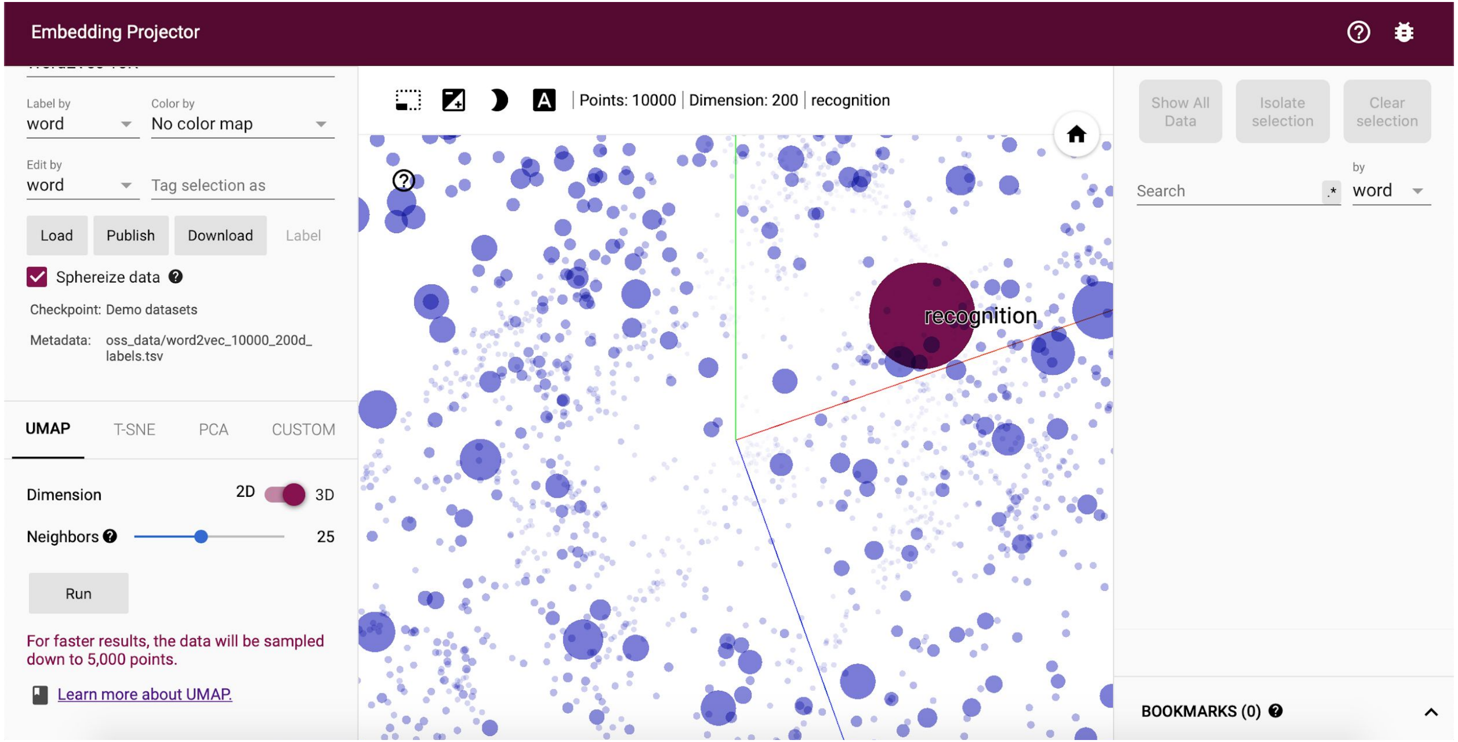
Natural Language Processing

Data embedding is commonly used for text and have led to models like Word2Vec

Skip Gram Model:



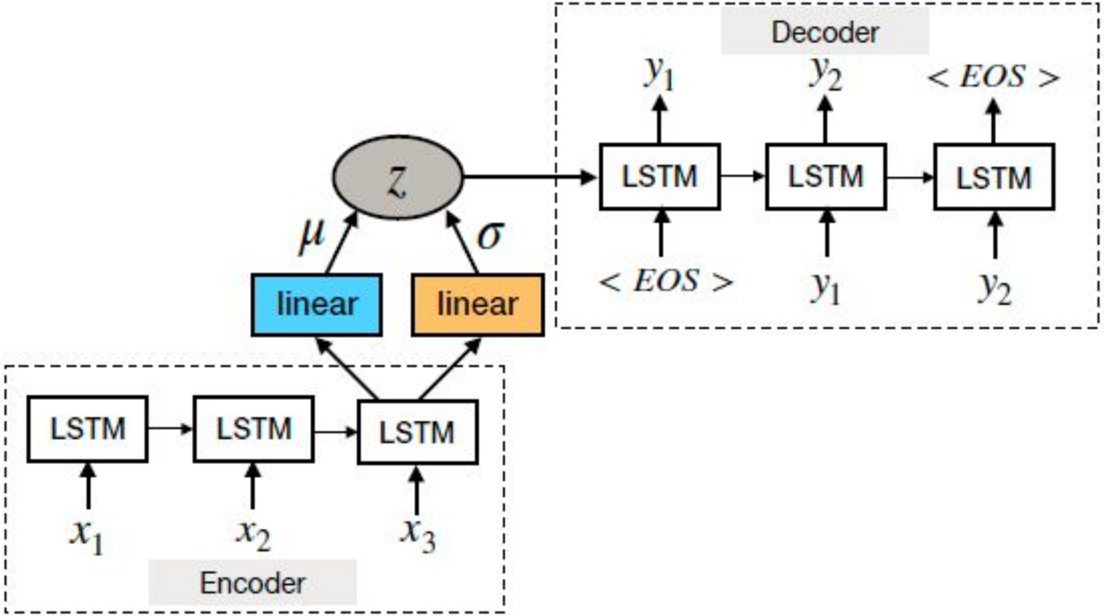
Natural Language Processing



http://projector.tensorflow.org

Natural Language Processing

Generative models inspired by VAEs



Generating Sentences from a Continuous Space, Bowan et al

Autoencoder

Pros	Cons
Simple	Each layer is trained greedy
Stack multiple layers	No global optimization
Intuitive	Reconstruction may be the ideal metric for learning

Generative Models (VAE, GAN)

Pros	Cons
Global training	Hard to train: conversion problem
Learning meaningful representation of data	More computationally expensive than AE
Better performance that AE	Slightly more parameters to learn, increasing complexity

Next week

Continue exploring more advanced unsupervised learning techniques

Deep Clustering

Practical Session

Quick recap of this material

Bring your laptops!

We will be coding in CoLab to build a VAE