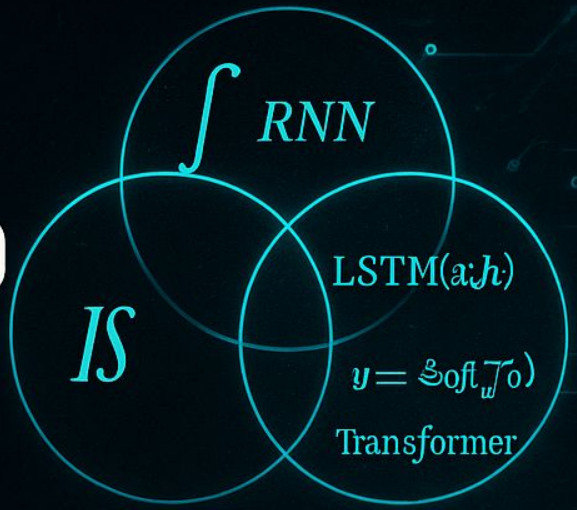
INTRODUCTION TO DEEP LEARNING

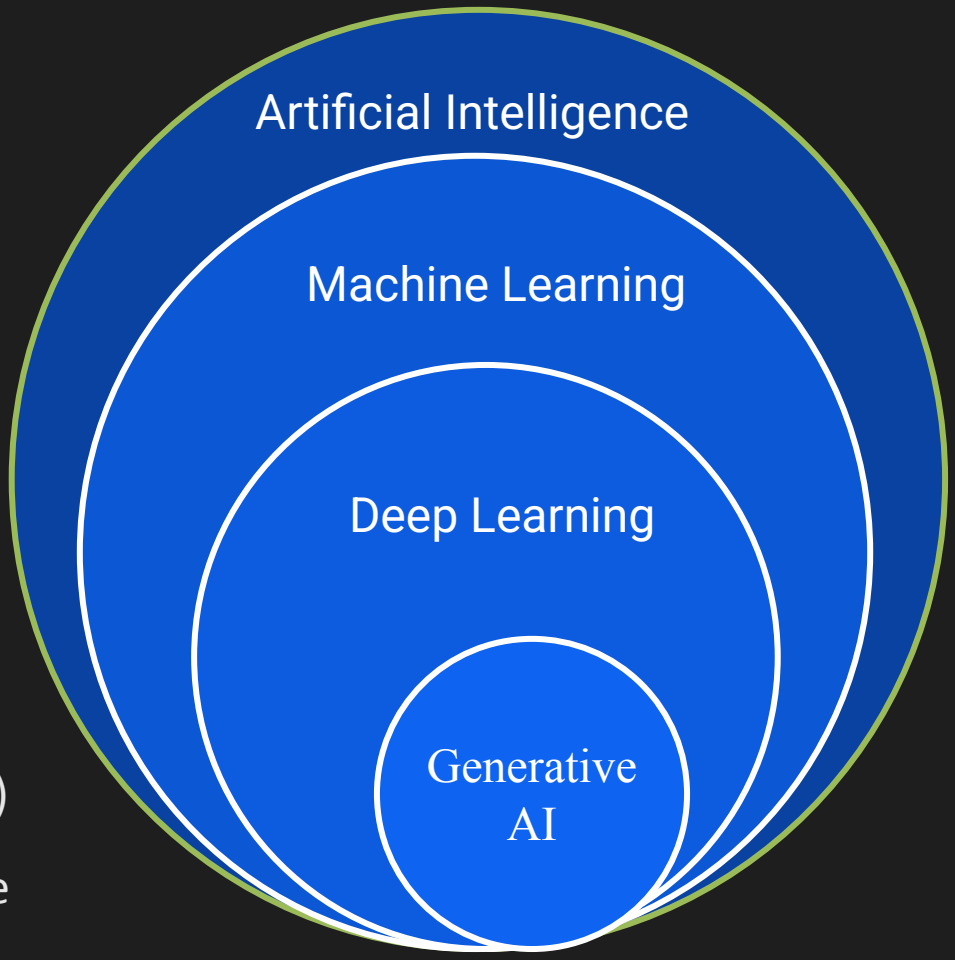
FOUNDATIONS FOR NLP



- Foundations of Deep Learning
- How it connects to AI, ML, and Generative AI
- Why Deep Learning is crucial for NLP
- Setting the stage for LSTMs, RNNs, and Transformers

AI, ML, DL, and Generative AI

Artificial Intelligence (AI): creating systems capable of performing tasks that normally require human intelligence.
Machine Learning (ML): a subset of AI that enables systems to learn patterns from data and improve their performance on a task over time without being explicitly programmed.
Deep Learning (DL): subfield of ML that uses artificial neural networks with many layers (deep networks) to automatically learn complex patterns in large datasets.
Generative AI: Models that generate new data (like text, images, audio, code) that resembles human-created data. It often uses deep learning techniques like transformers and GANs (Generative Adversarial Networks).



The Machine Learning Pipeline

1) **Input Data:**

Each data sample can be: {X}, {X,Y}, {X,Y, Model})

2) **Preprocessing :**

Missing values, noise, and inconsistencies

Normalize/standardize features

Convert categorical variables Y (e.g., one-hot encoding)

Split into training/ validation/ test sets

Creating samples for data augmentation

3) **Choose Model Architecture:**

SVM, Neural Networks Decision Trees

A function f() which relates X→Y_hat

Ensemble models if problem

4) **Define Loss Function:**

Disparity between Y & Y_hat

5)**Train Model via Optimization:**

Update trainable parameters of f() based on Loss so Y_hat→ Y

6)Validate & Evaluate

Four Pillars of Machine Learning

Data: Processed data for training (each sample is {X,y} & y is continuous)

Model (Hypothesis): Function mapping input to output

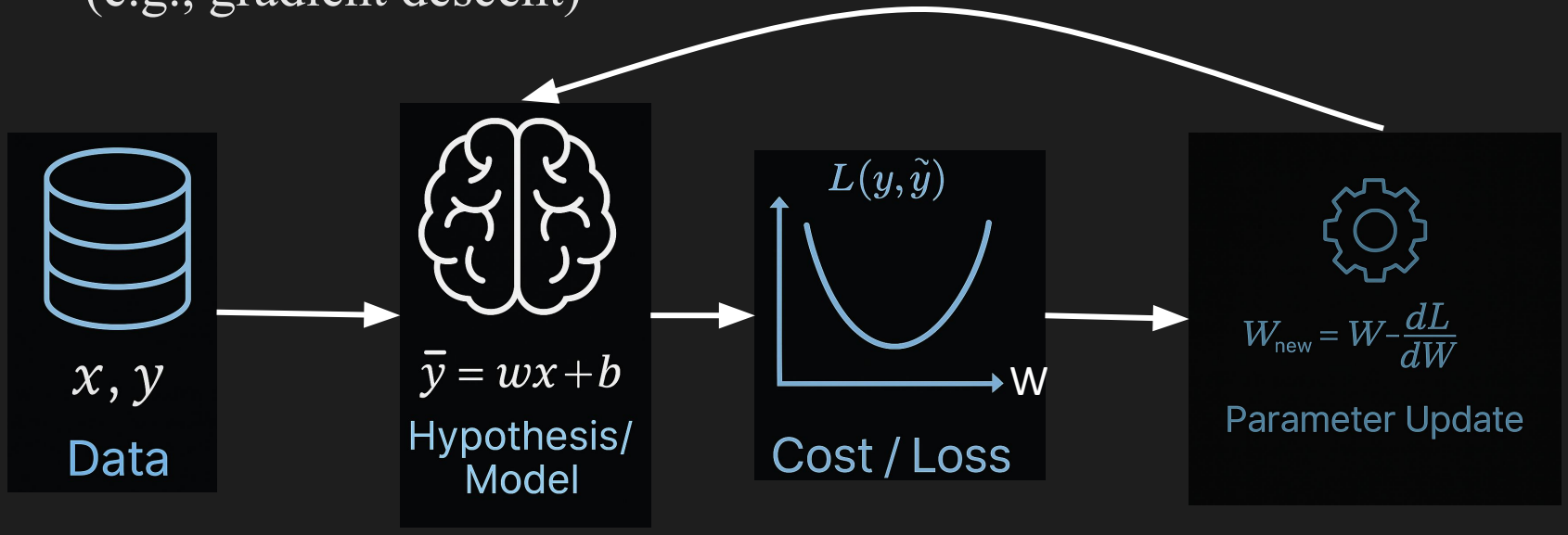
y_hat = f(X) = WX+w0= w1x1+w2x2 +.... +w0

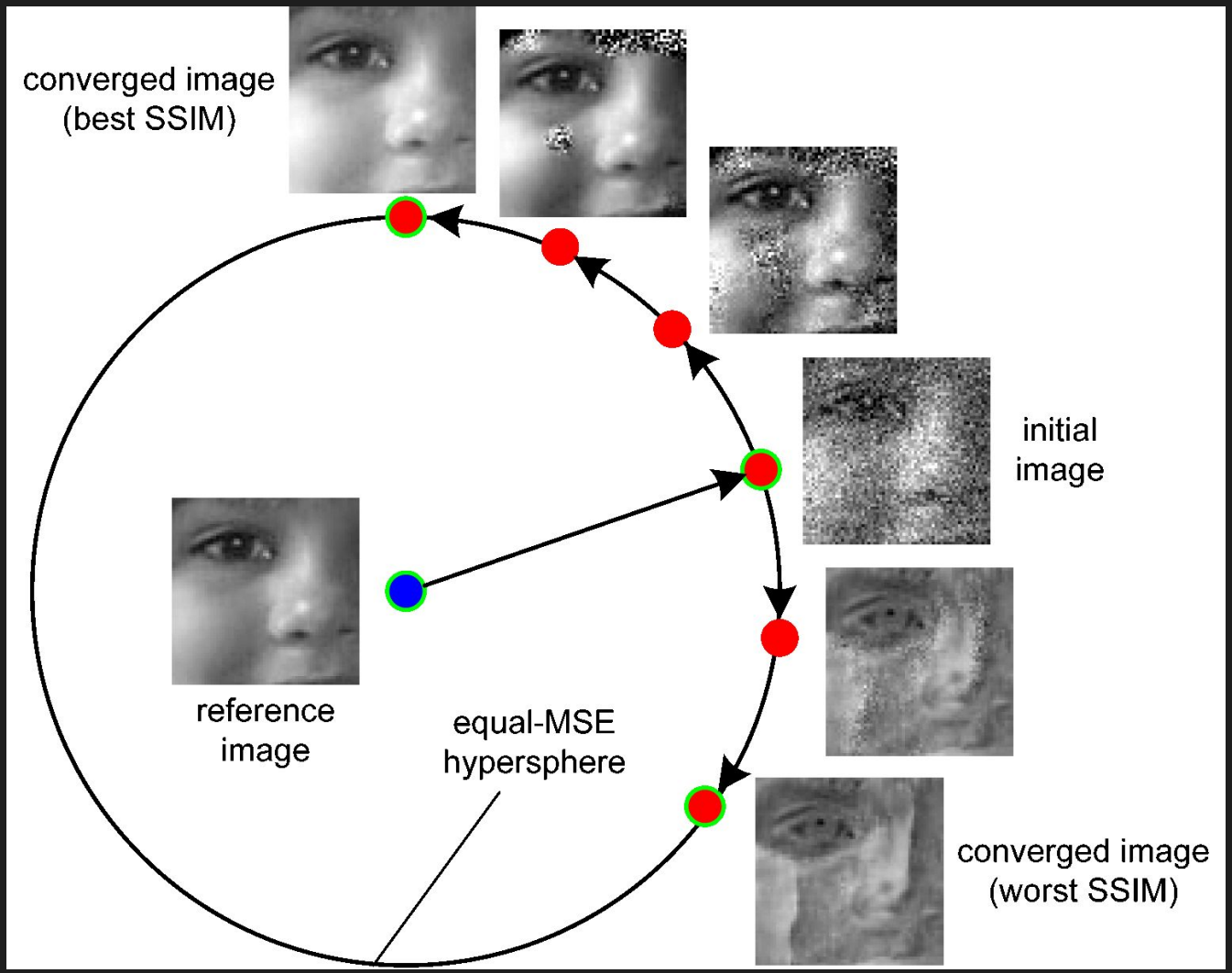
Loss Function: Measures model error

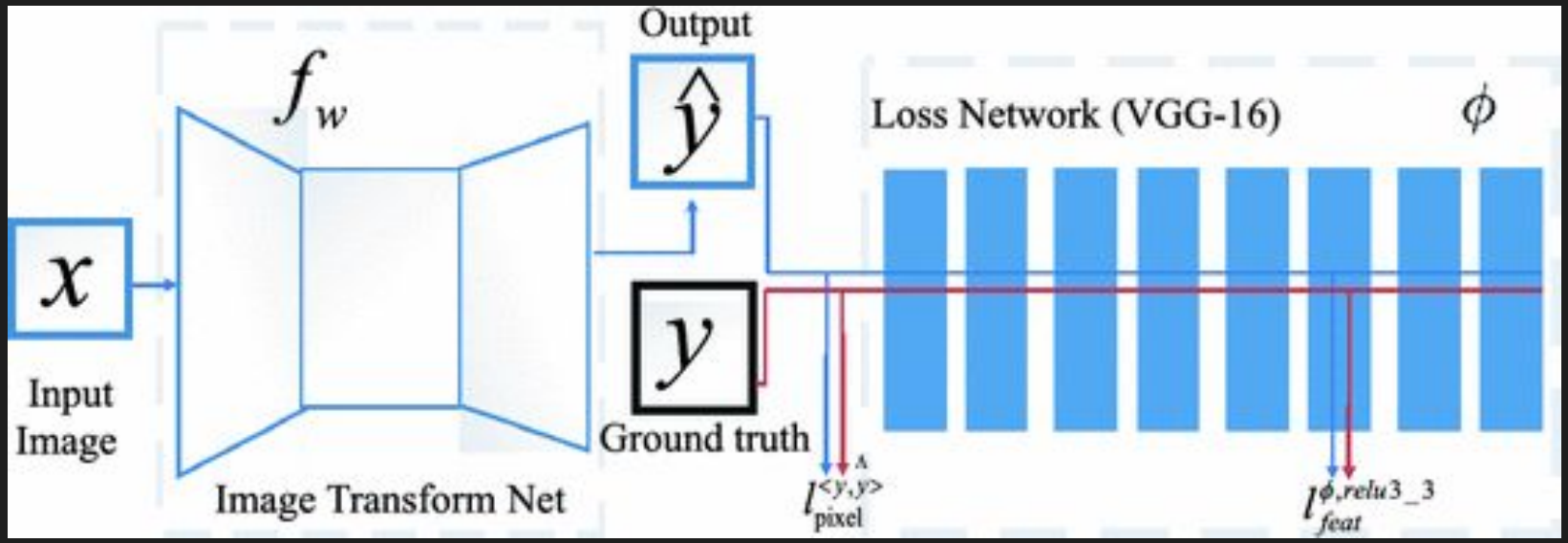
Regression loss function ex MSE

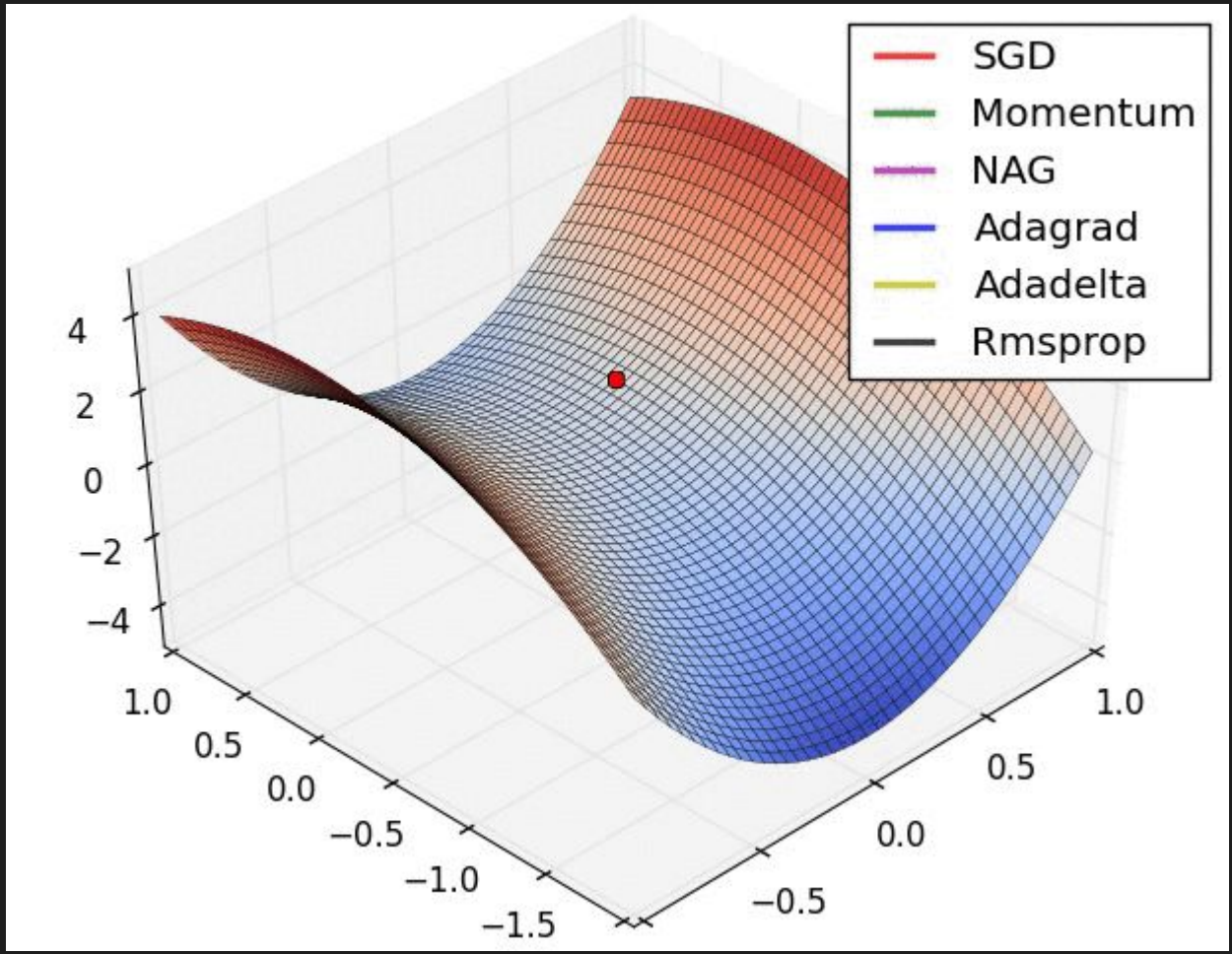
Learning Rule: Updates model parameters (e.g., gradient descent)

$$w := w - \eta \cdot \frac{dL}{dw}$$




converged image
(best SSIM)
initial
image
reference
image
equal-MSE
hypersphere
converged image
(worst SSIM)


Output
f_w
$\hat{y}$
Loss Network (VGG-16)
ϕ
x
y
Input Image
Ground truth
Image Transform Net
$l^{<\hat{y},y>}_{pixel}$
$l^{\phi,relu3\_3}_{feat}$


SGD
Momentum
NAG
Adagrad
Adadelta
Rmsprop
4
2
0
−2
−4
1.0
0.5
0.0
−0.5
−1.0
−1.5
1.0
0.5
0.0
−0.5

\hat{y} = \sigma\left(
\mathbf{w}^\top \mathbf{x} + b
\right)

Classical ML Models

Linear Regression → Logistic Regression

$$f(x) = w_1 x + w_0$$

$$f(x) = \sigma(w_1^T + b_1) = \frac{1}{1+e^{-(w_x+w_0)}}$$

Neural Networks

L2 Regularization

$$\text{Cost} = \underbrace{\sum_{i=0}^{N}(y_i - \sum_{j=0}^{M} x_{ij}W_j)^2}_{\text{Loss function}} + \underbrace{\lambda \sum_{j=0}^{M} W_j^2}_{\text{Regularization Term}}$$

$$\mathcal{L}(\mathbf{w}, b) = -\frac{1}{N}\sum_{i=1}^{N}\left[y^{(i)}\log(\hat{y}^{(i)}) + (1-y^{(i)})\log(1-\hat{y}^{(i)})\right] + \lambda\|\mathbf{w}\|_2^2$$

$$\mathcal{L}(\mathbf{w}, b) = \frac{1}{N}\sum_{i=1}^{N}\max(0, 1 - y^{(i)} f(\mathbf{x}^{(i)})) + \lambda\|\mathbf{w}\|_2^2$$

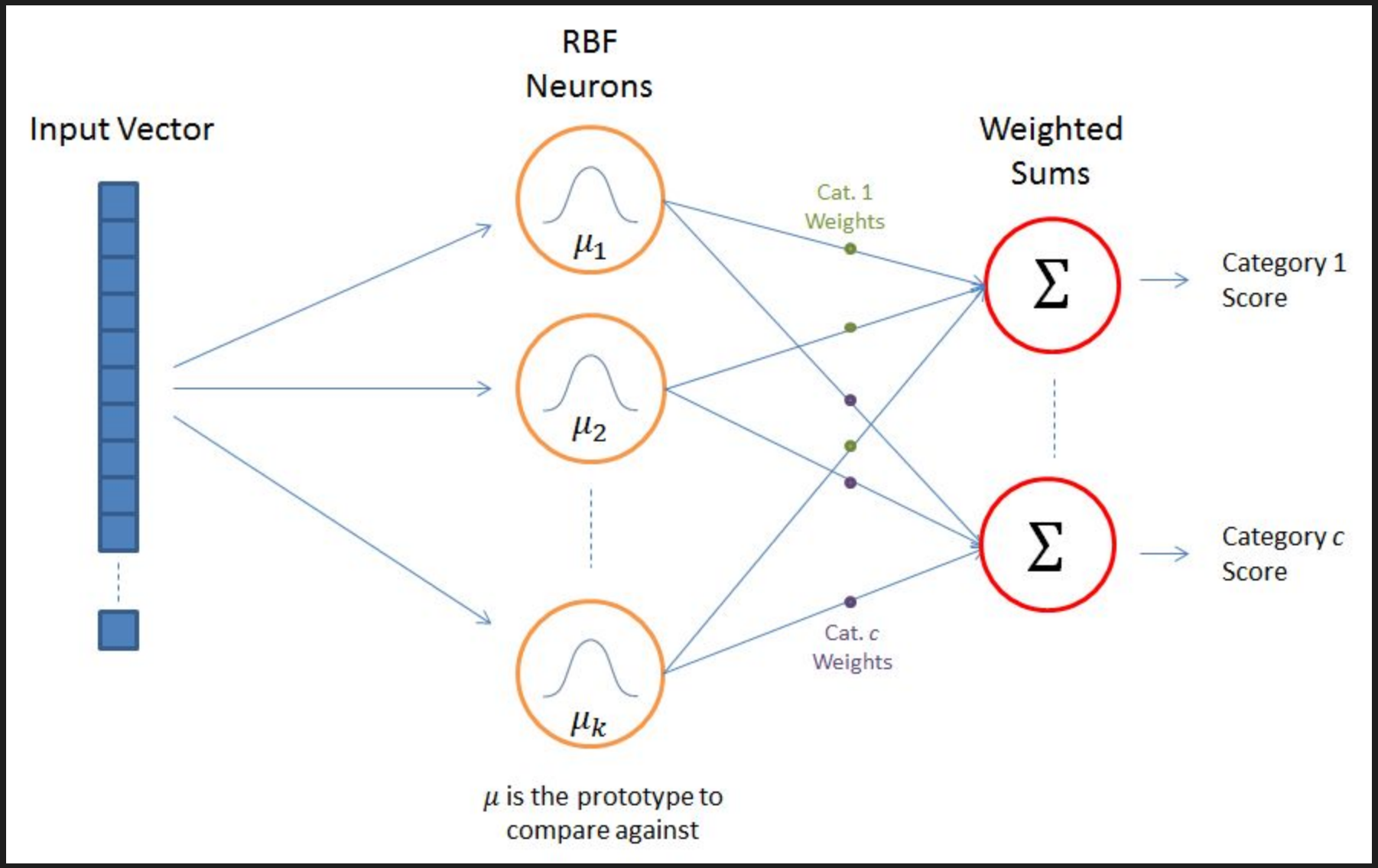
Support Vector Machines

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N}\alpha_i\left[y^{(i)}(\mathbf{w}^\top\mathbf{x}^{(i)} + b) - 1\right]$$

$$\mathcal{L}(\mathbf{w}, b) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\max(0, 1 - y^{(i)}(\mathbf{w}^\top\mathbf{x}^{(i)} + b))$$

\hat{y} = \sigma\left(
\mathbf{w}^\top \mathbf{x} + b
\right)

Classical ML Models: Kernel SVM mimicked with RBF neural network



Decision Trees & Random Forests

\hat{y} = \sigma\left(
\mathbf{w}^\top \mathbf{x} + b
\right)

Classical ML Models

Decision Trees → Softmax activation based neural networks

Entropy Based

$$H(p) = -\sum_{k=1}^{K} p_k \log(p_k)$$

$$IG(X) = H(\text{par}) - \sum_{j} \frac{N_j}{N} H(\text{child}_j)$$

Gini Impurity

$$G(p) = 1 - \sum_{k=1}^{K} p_k^2$$

Neural Networks

$$\mathcal{L}_{\text{CE}} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} y_k^{(i)} \log(\hat{y}_k^{(i)})$$

$$\mathcal{L}_{\text{Gini}} = \frac{1}{N} \sum_{i=1}^{N} \left(1 - \sum_{k=1}^{K} \hat{y}_k^{(i)^2} \right)$$

\hat{y} = \sigma\left(
\mathbf{w}^\top \mathbf{x} + b
\right)

Classical ML Models

Eigen vectors⟶ Autoencoders with linear activation

Clustering —> Self organizing maps

Convolution ⟶ Convolutional neural networks

Dictionary learning ⟶ Encoder decoder architecture with regularizer

Probabilistic Graphical models —> Graph neural networks

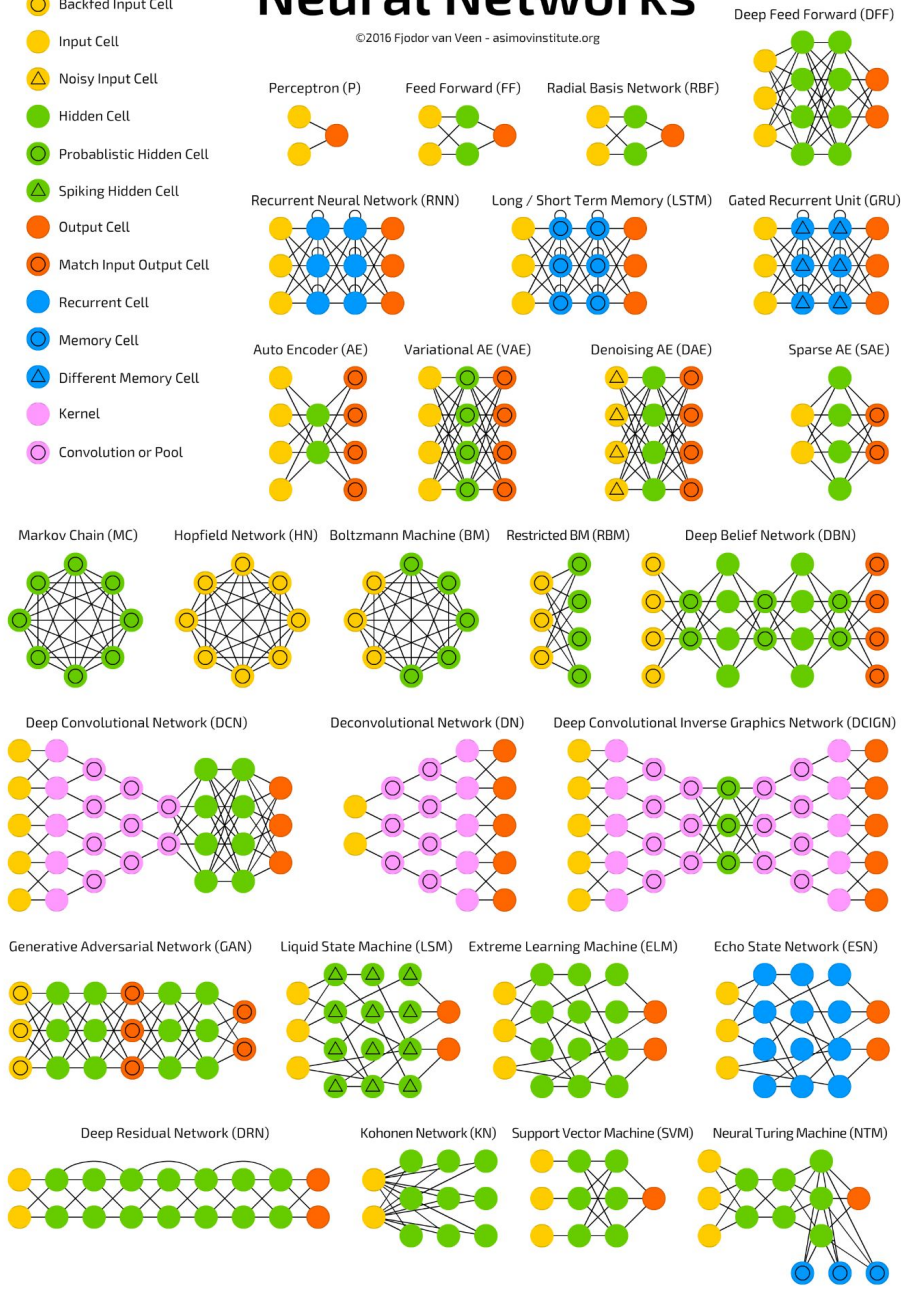
Feature extraction ⟶ Multi layer perceptron

Conditioanal random fields⟶ Energy based neural networks (RNN using mean field thoery)

A mostly complete chart of

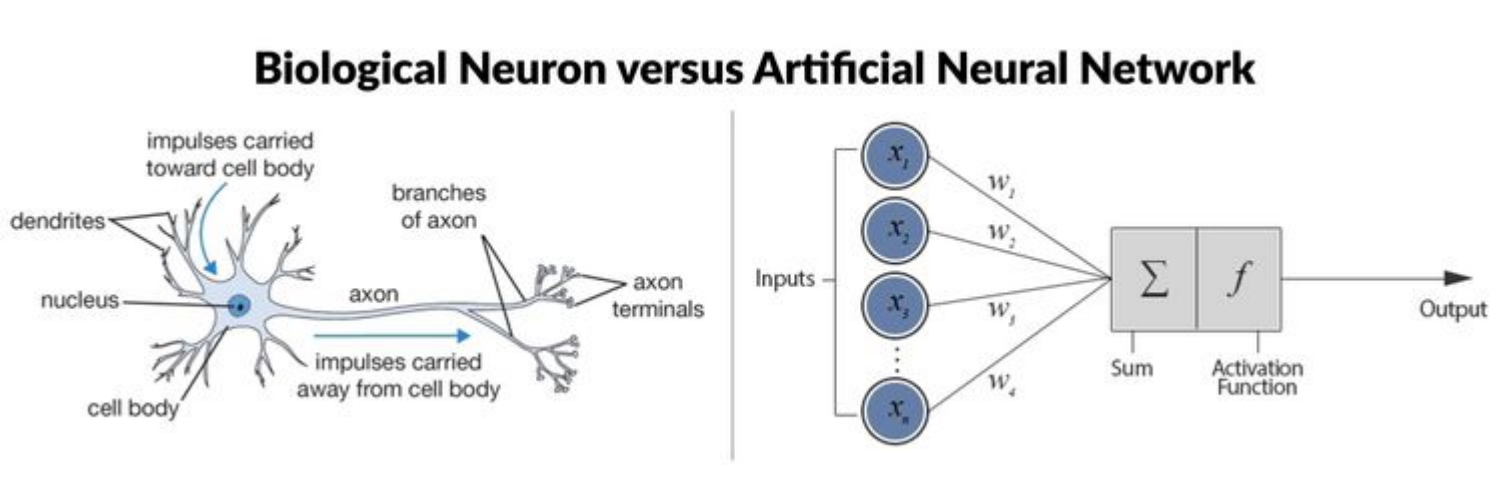
Neural Networks

- Backfed Input Cell
- Input Cell
- Noisy Input Cell
- Hidden Cell
- Probablistic Hidden Cell
- Spiking Hidden Cell
- Output Cell
- Match Input Output Cell
- Recurrent Cell
- Memory Cell
- Different Memory Cell
- Kernel
- Convolution or Pool



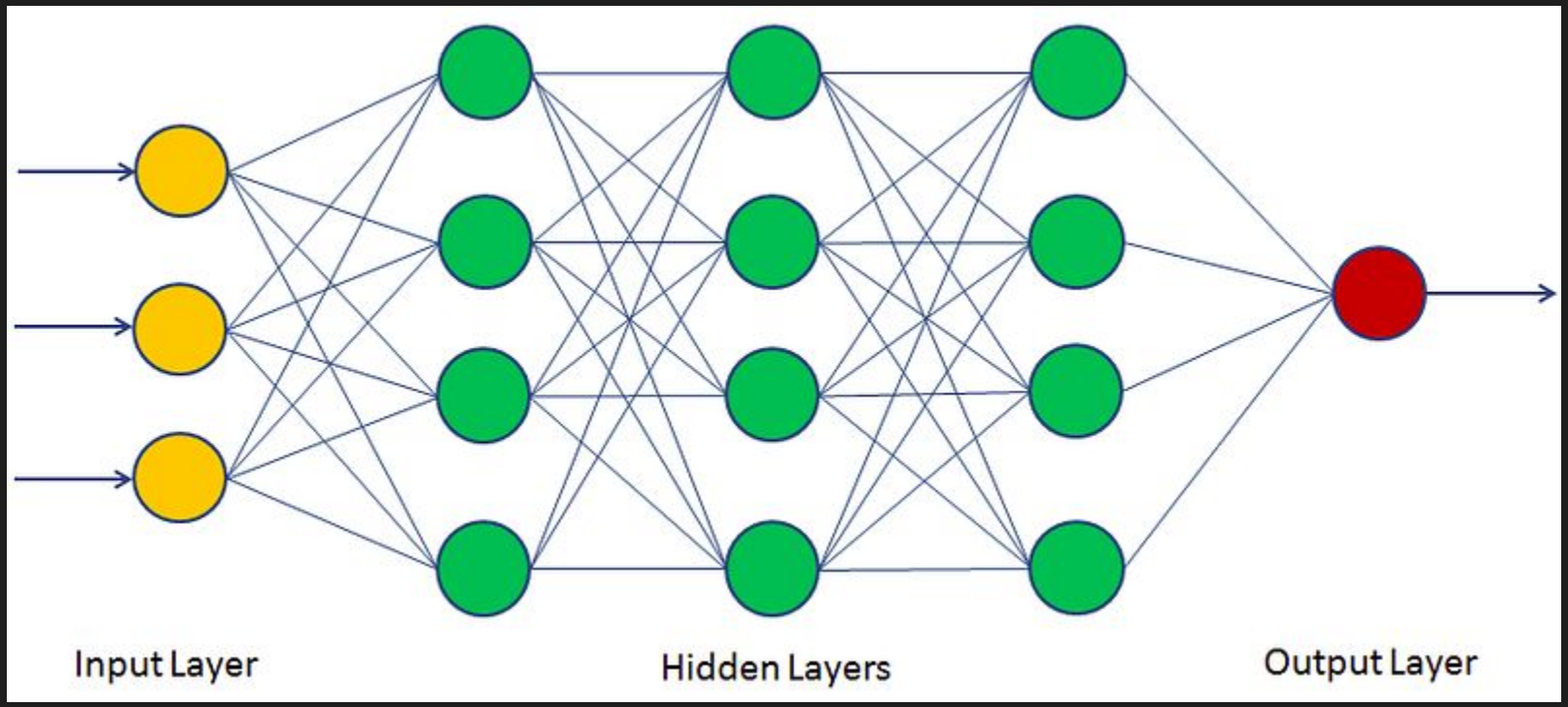
What is a Neural Network?

Inspired by biological neurons
Composed of layers of neurons
Used to approximate complex functions



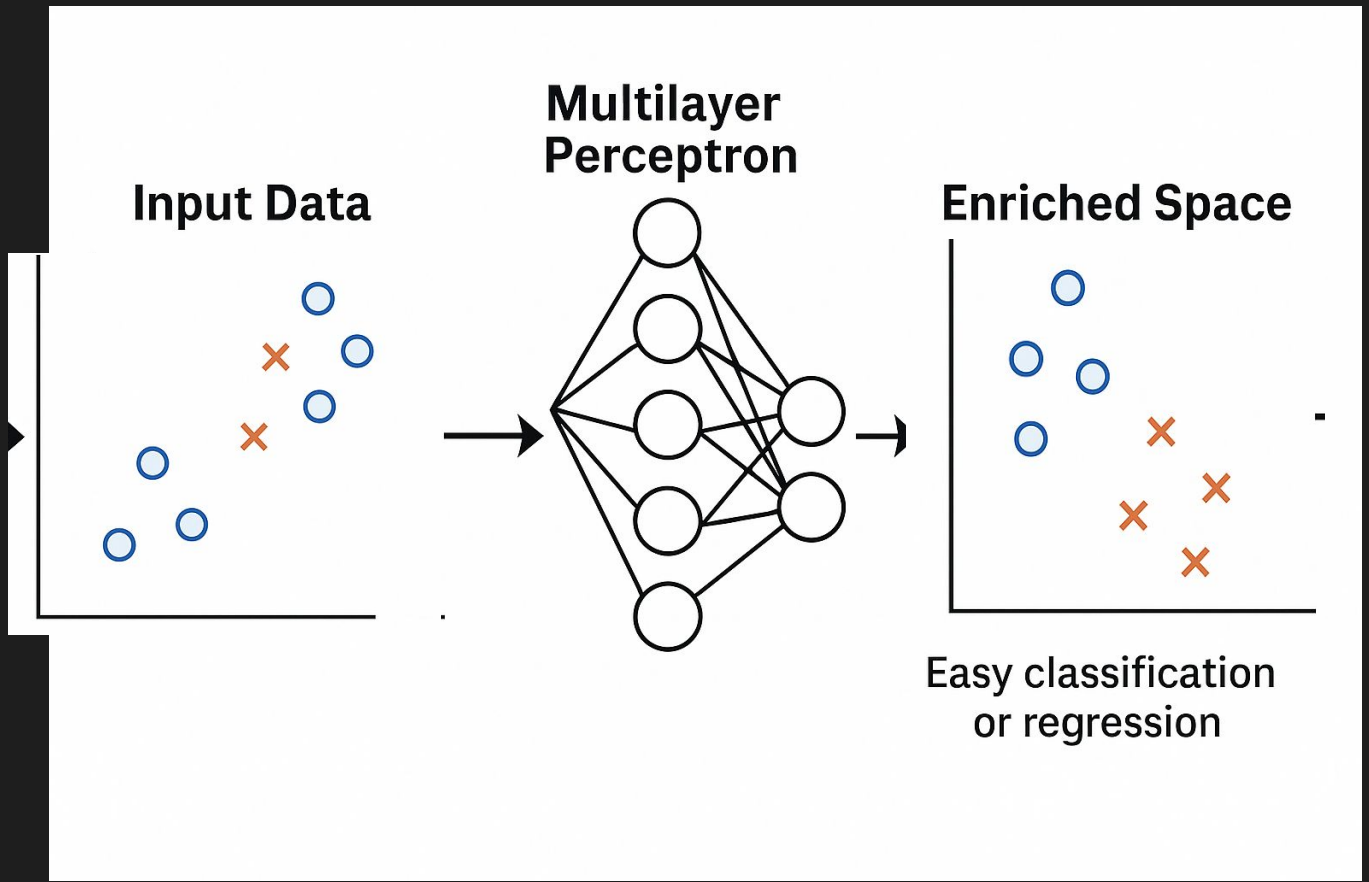
What is a Neural Network?

Inspired by biological neurons
Composed of layers of neurons
Used to approximate complex functions F()--> g(h(q()))



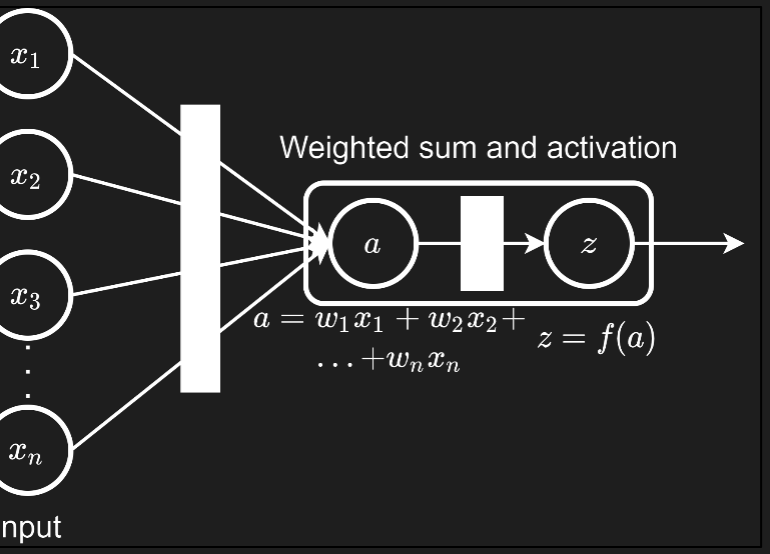
What is a Neural Network?

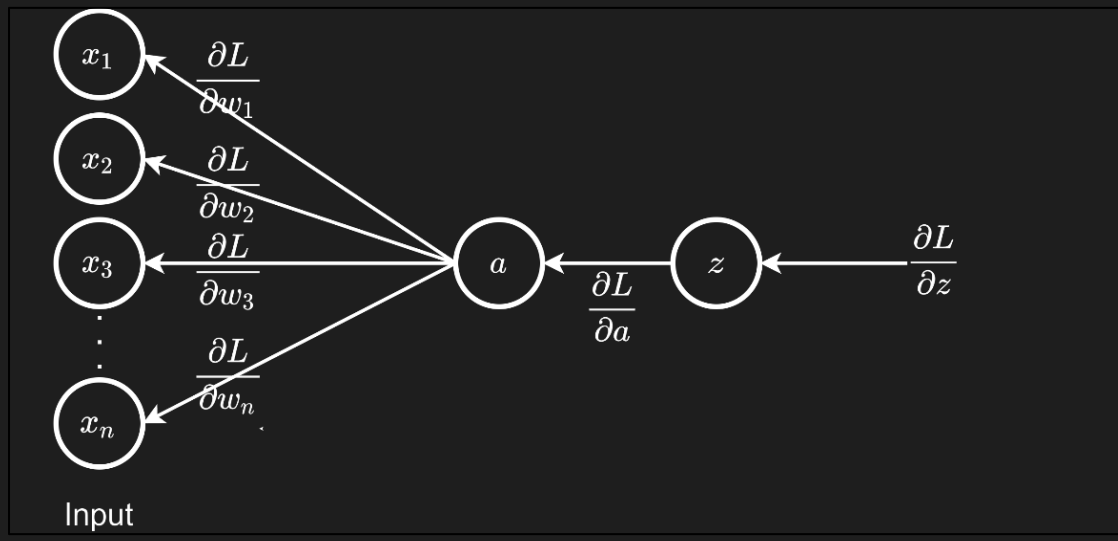
Inspired by biological neurons
Composed of layers of neurons
Used to approximate complex functions



Single Neuron: Equation and Intuition

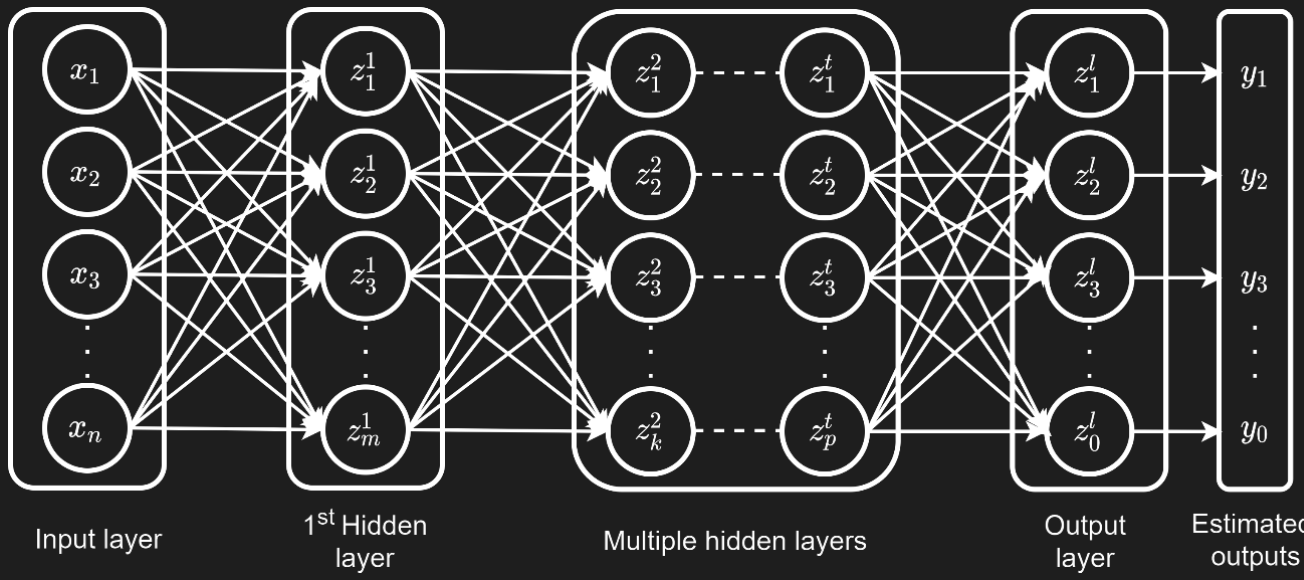
y = activation(w·x + b)
Projects input features to a new space
Acts as a basic computational unit





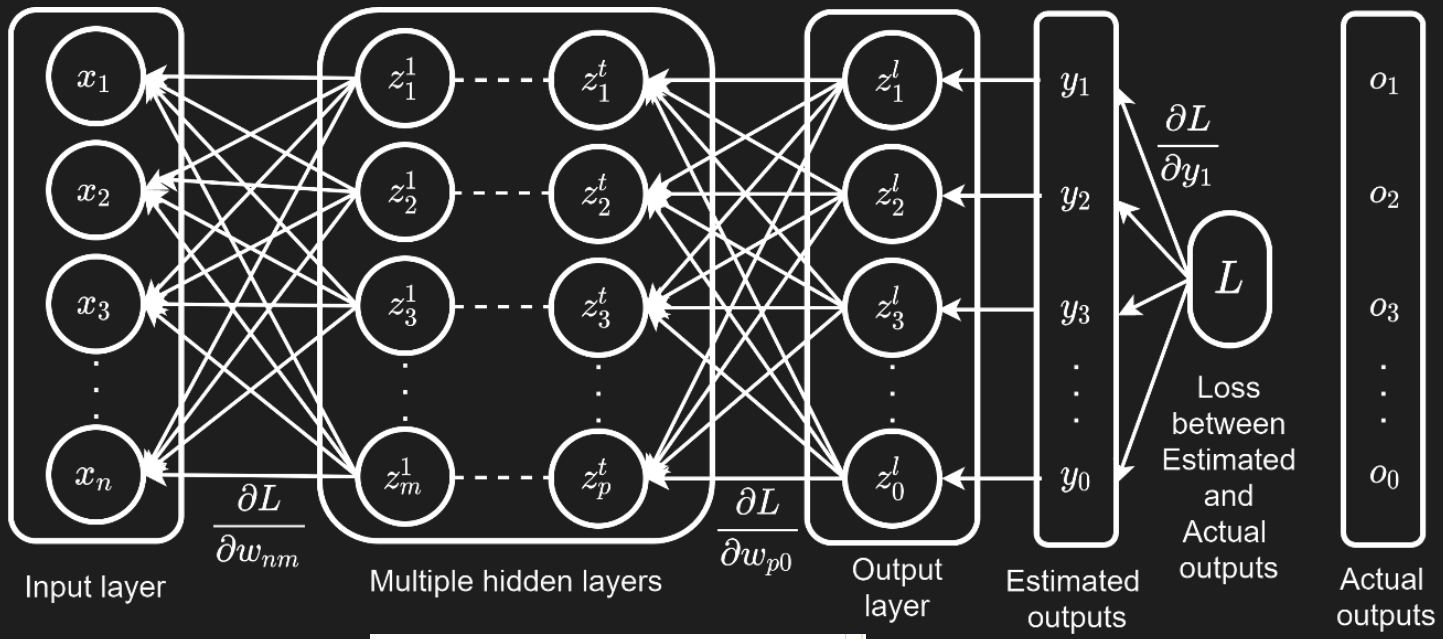
Layer of Neurons = Feature Transformation

Transforms input vector into hidden representation
Linear transformation followed by non-linearity
Helps in learning more abstract features



Layer of Neurons = Feature Transformation

Transforms input vector into hidden representation
Linear transformation followed by non-linearity
Helps in learning more abstract features

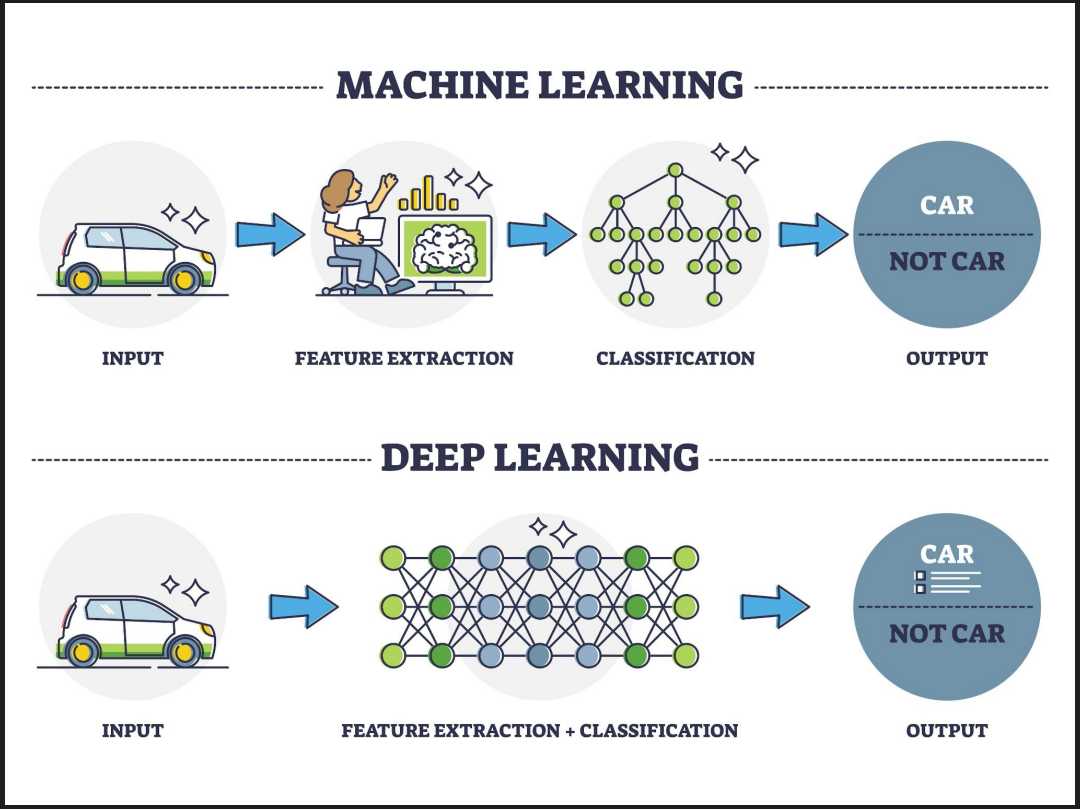


$$\frac{\partial L}{\partial w_{p0}} = \frac{\partial L}{\partial y_0} \cdot \frac{\partial y_0}{\partial w_{p0}}$$

$$= \frac{\partial L}{\partial y_0} \cdot \frac{\partial y_0}{\partial z_0^l} \cdot \frac{\partial z_0^l}{\partial w_{p0}}$$

Deep Neural Networks (DNNs)

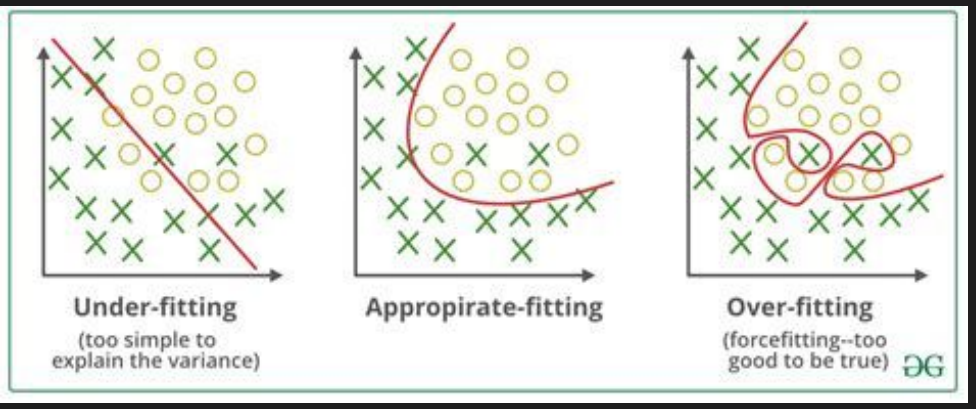
Multiple layers = hierarchical representation or absttraction
Successive transformation of information
Enables learning of complex patterns

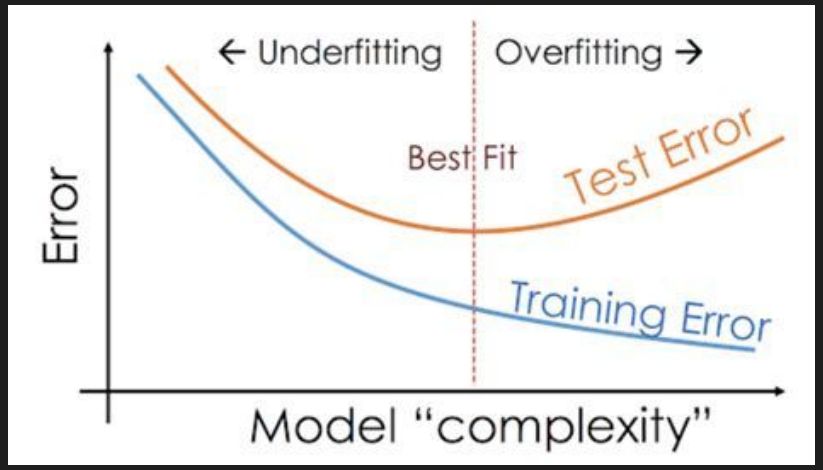


Feedforward Network Structure

Input ⟶ Hidden Layers ⟶ Output
Activation: Sigmoid, Tanh
Loss: MSE, Cross-Entropy
Error back propagation via Chain rule to update weights
One-way data flow (no cycles)

Problems Underfitting vs Overfitting
Solution: Increase # parameters Vs adding regularizer





Issue with Deeper Architecures

Forward pass
Sigmoid Activation at each layer
$a1 = sig(w1x+b1) \rightarrow [0,1]$
$z2=W2a1+b2$
$a2 = sig(z2) \rightarrow [0,1]$ …. say 100 layers

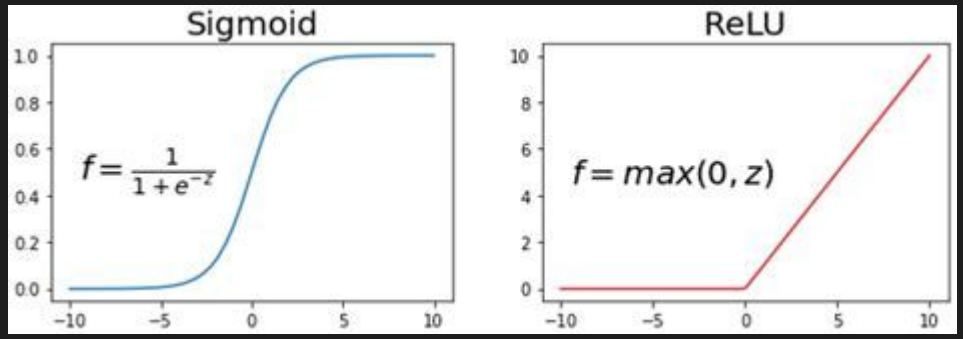
Backward pass
$da_1/dz_1 = a_1(1 - a_1)$ -----> < 0.5
$da_2/dz_2 = a_2(1 - a_2)$ -----> < 0.5
$da_3/dz_3 = a_3(1 - a_3)$ -----> < 0.5

$dL/dW_1 = dL/da_{100} \cdot \underline{da_{100}/dz_{100}} \cdot dz_{100}/da_{99} \cdot \underline{da_{99}/dz_{99}} \cdot dz_{99}/da_{98} \cdot \ldots \cdot dz_2/da_1 \cdot \underline{da_1/dz_1} \cdot dz_1/dW_1$

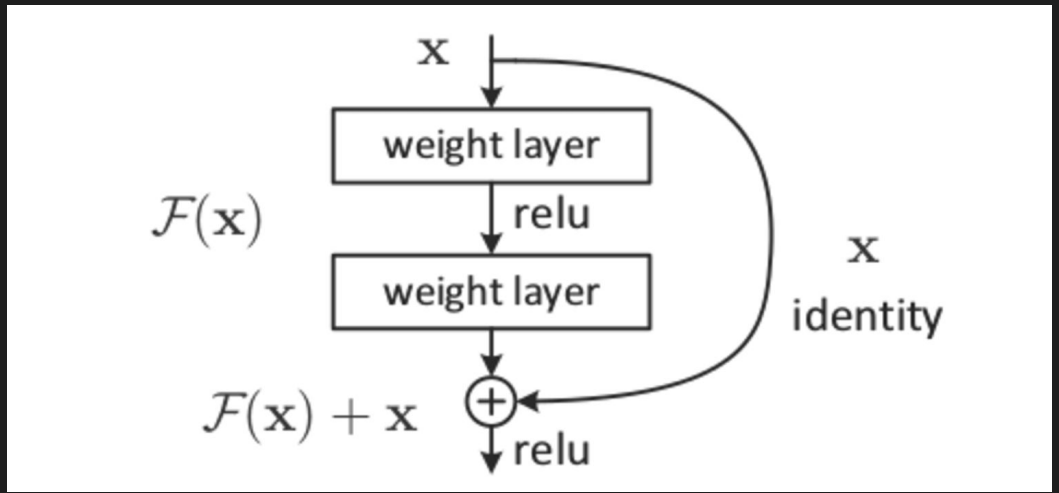
So gradient is being squashed at every layer

Modifications in Deeper Architecures

ReLu Activation instead of Sigmoid



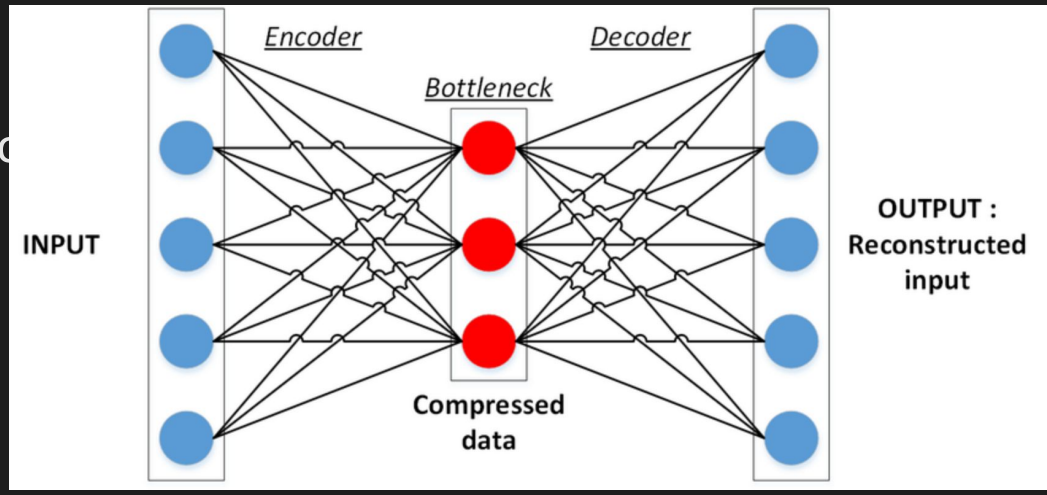
Skip or Residual Connections



Issue with Deeper Architecures

Deep Architecture —> More # learnable parameters
More # learnable parameters – > Large amount of data

But Large amount of Labled data is expensive
Solution: Make use of Unlabelled Data
Conditions:
1) Y=X;
2) Decoder weights is Transpose o
Encoder Weights

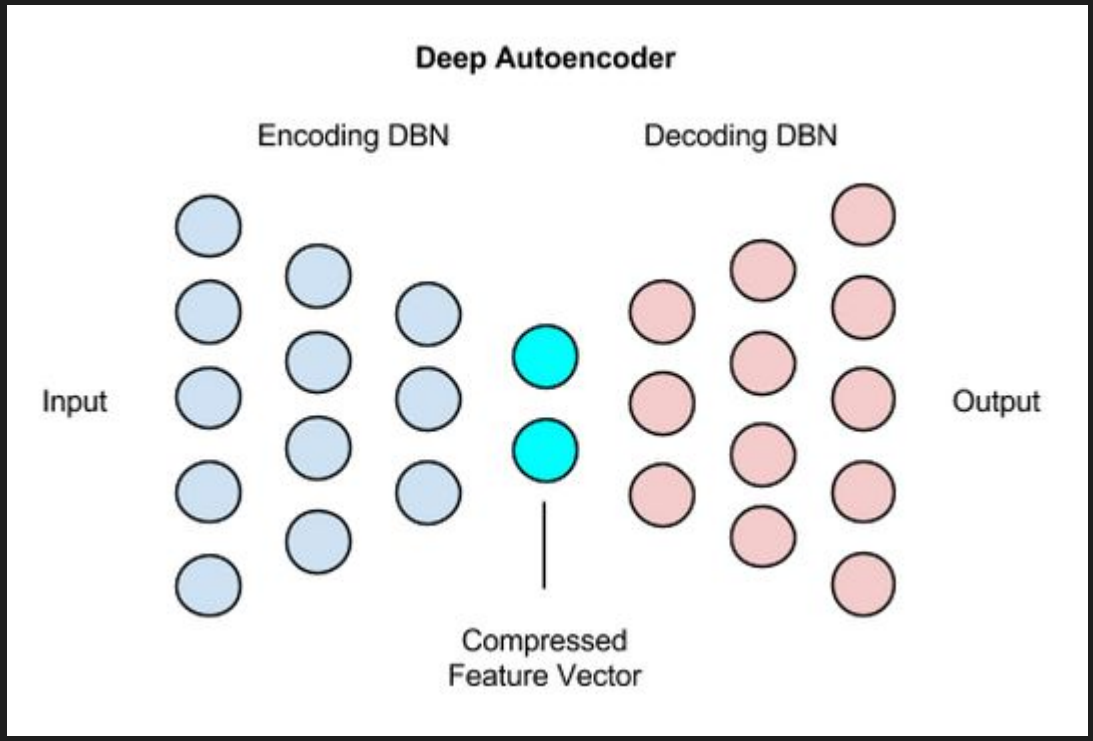


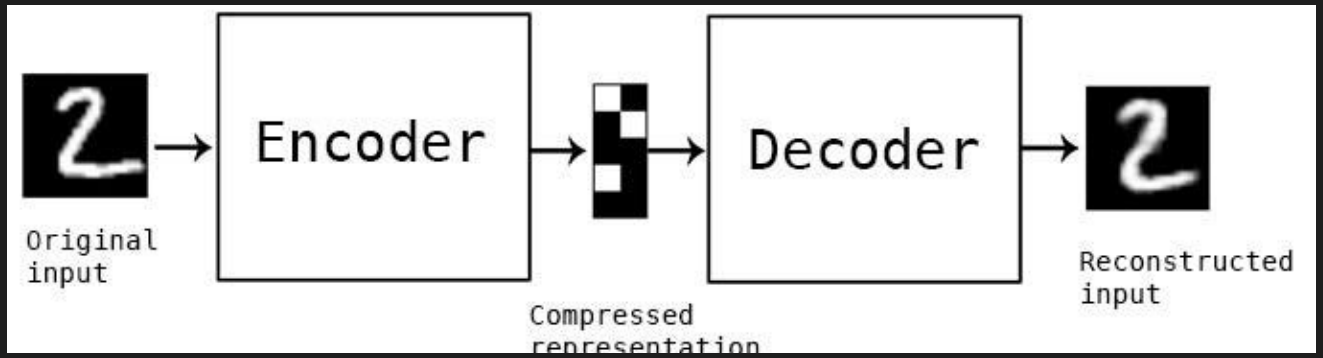
Y_hat=Decoder(Encoder(X))
Y_hat= Decoder (WX)
Y_hat= W' (WX)-->X_hat

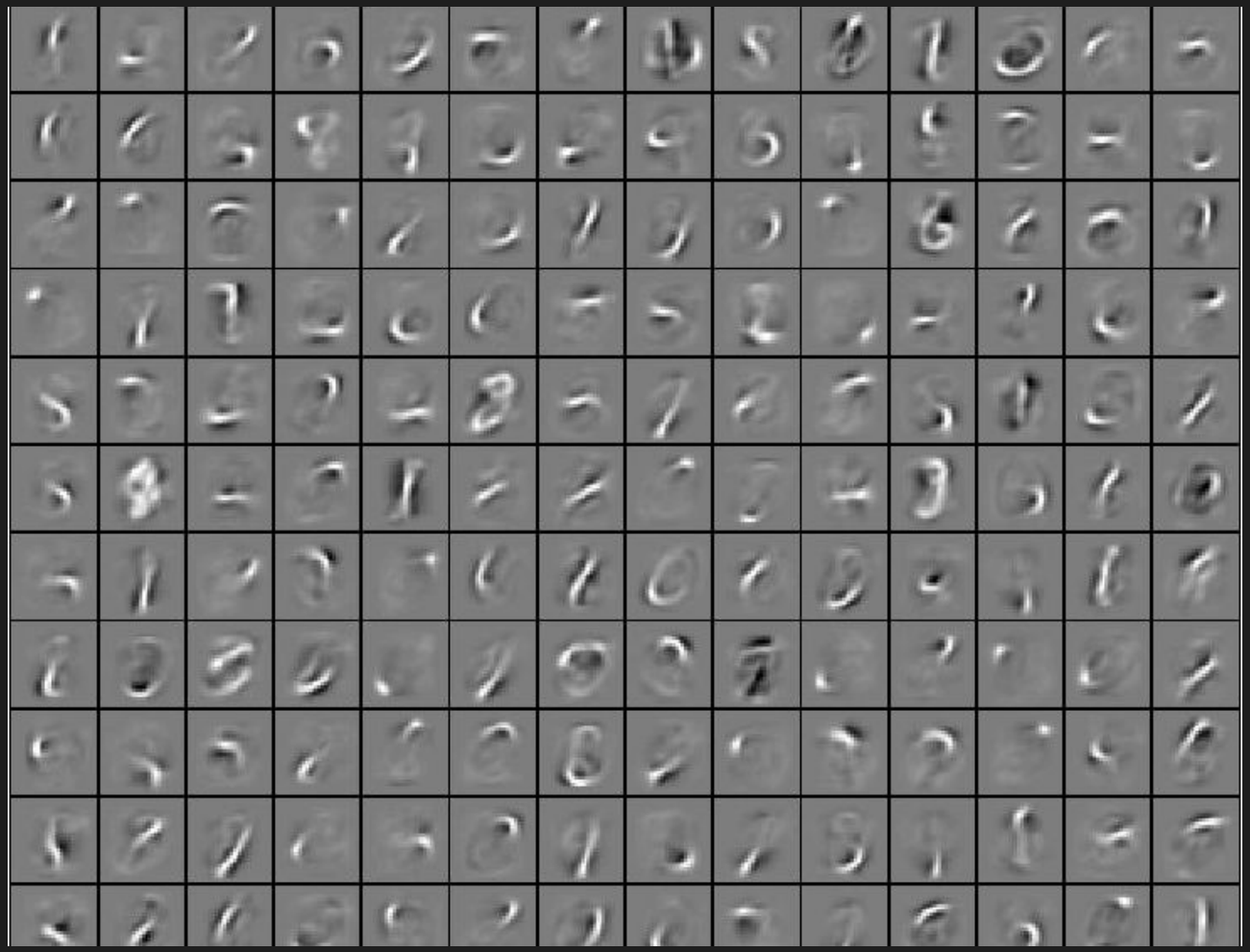
Loss = MSE(X_hat,Y) = MSE(X_hat,X)
Effectively
X_hat = W'WX—> Learn W are analogous to eigen vectors
X_hat = X Only if W'W = Identity

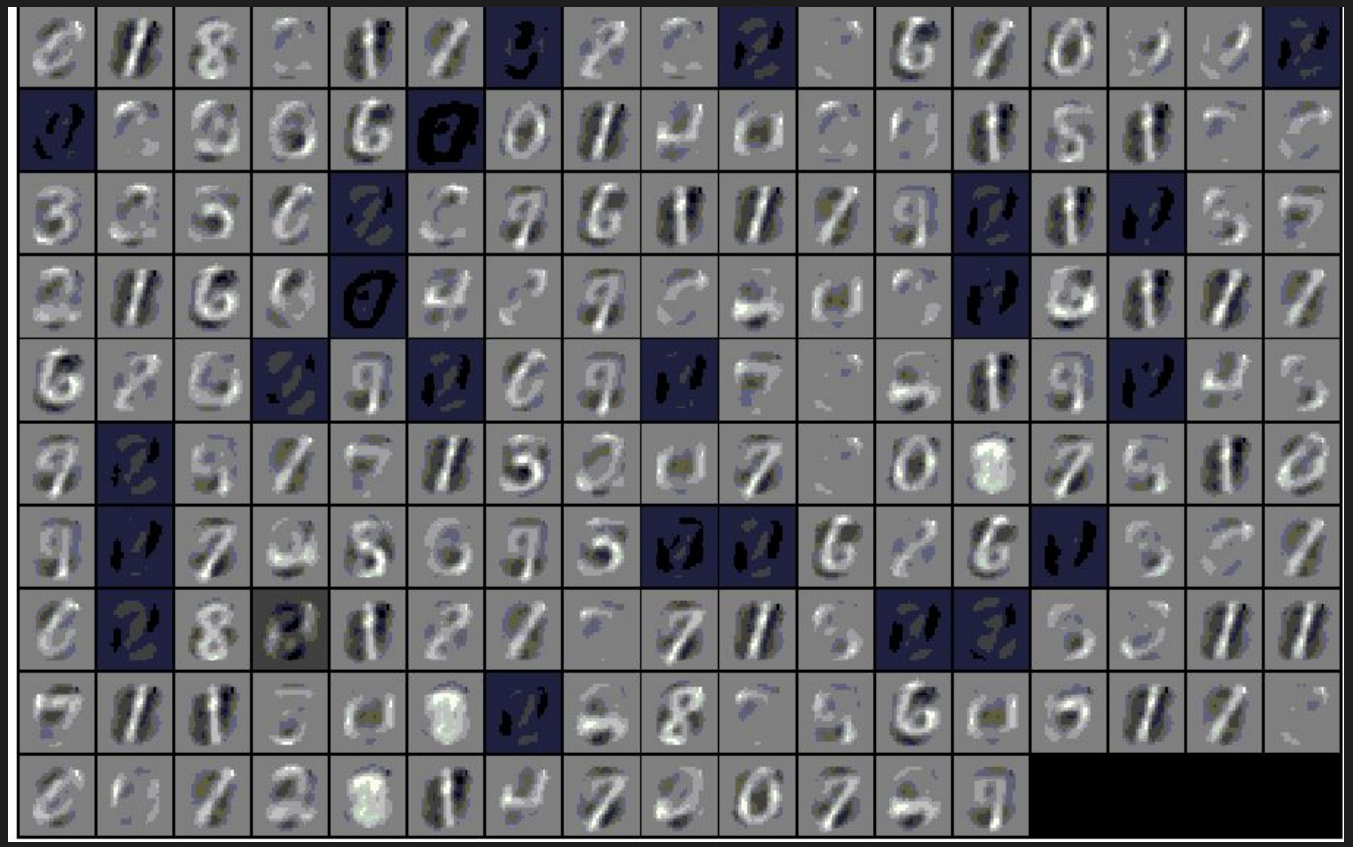
Deeper Architecures Effectively Learn Features

Deep Autoencoder




Encoder
Decoder
Original input
Compressed representation
Reconstructed input





Training Workflow of Encoder Decoder is same as ML

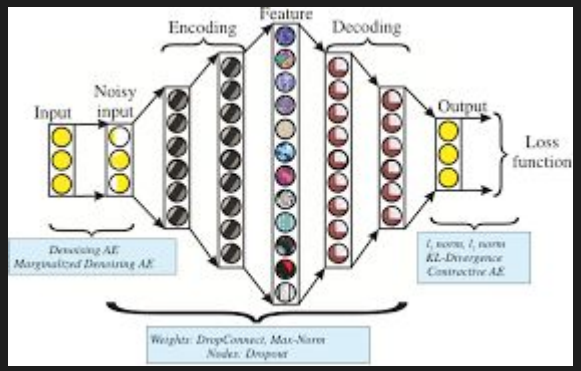
Forward pass → Compute loss
Backward pass (backpropagation)
Update weights using optimizer

Post training **Encoder** acts a **Feature Extractor**
1) Encoder + Regressor OR 2) Encoder + Classifier

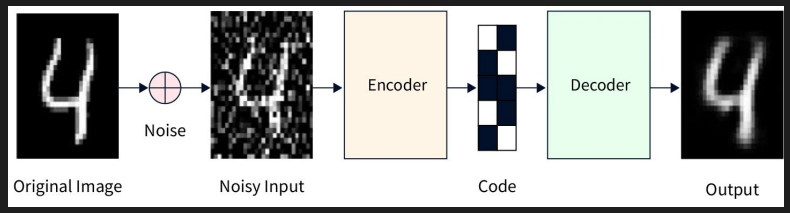
Post training **Decoder** acts a Generator

Variants of Auto encoders

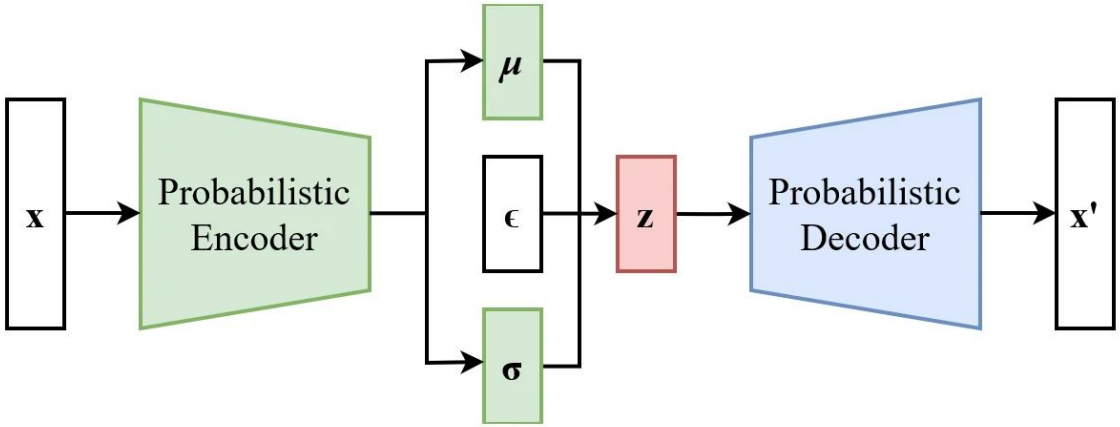
Sparse Autoencoder



Denoising autoencoder

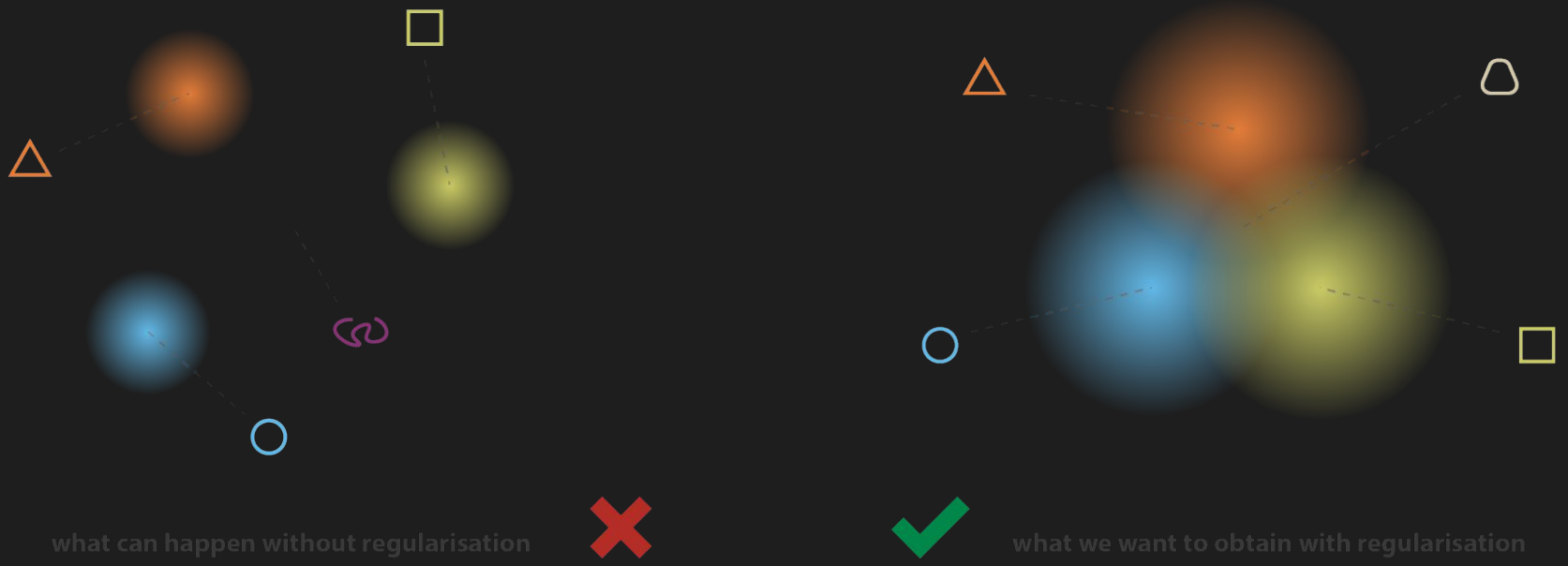


Variational Autoencoder ….

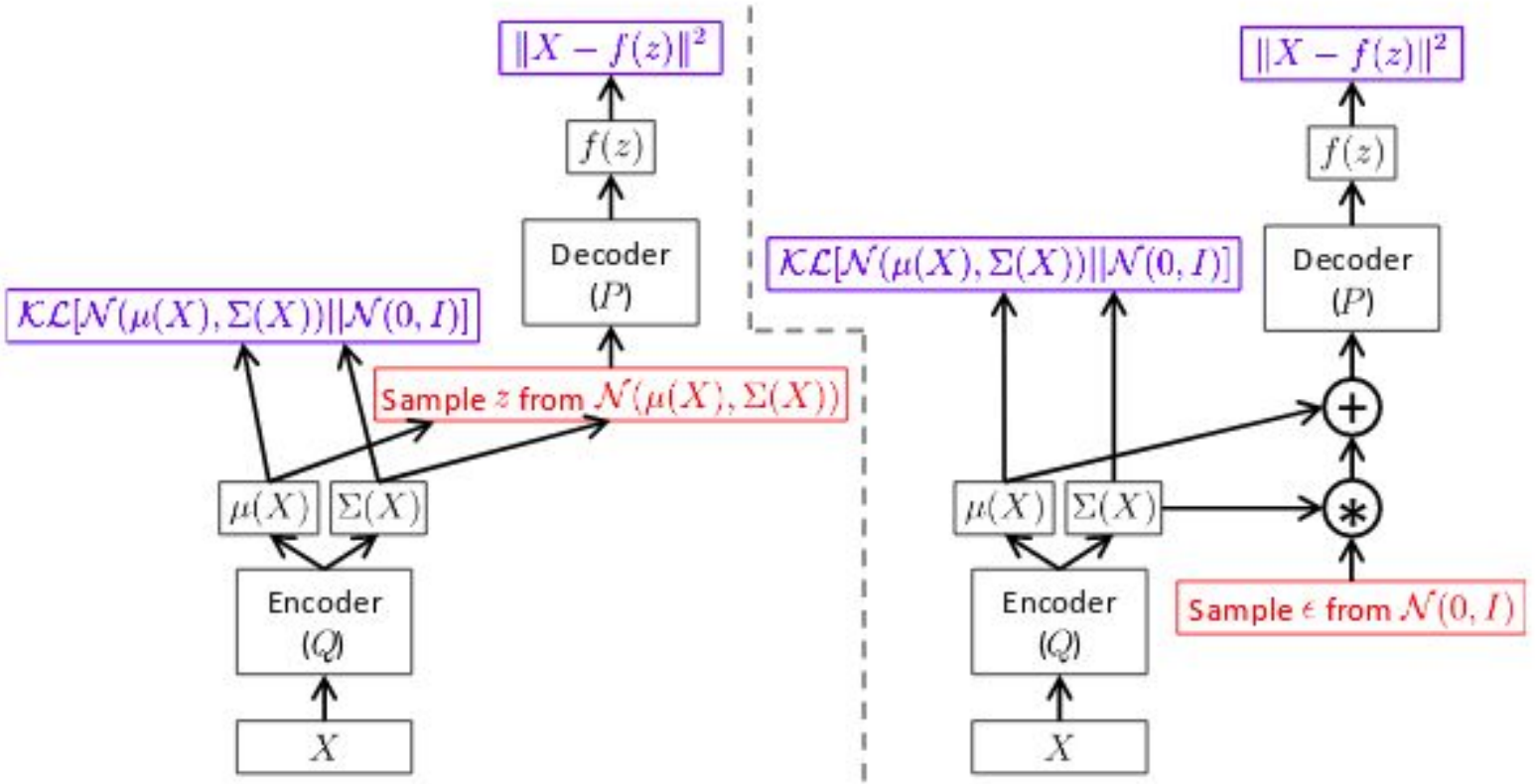


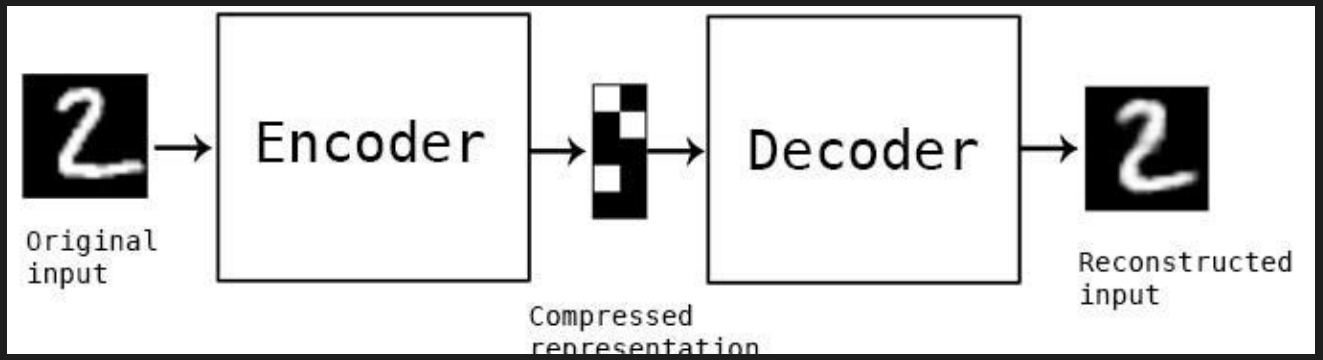
Why VAE

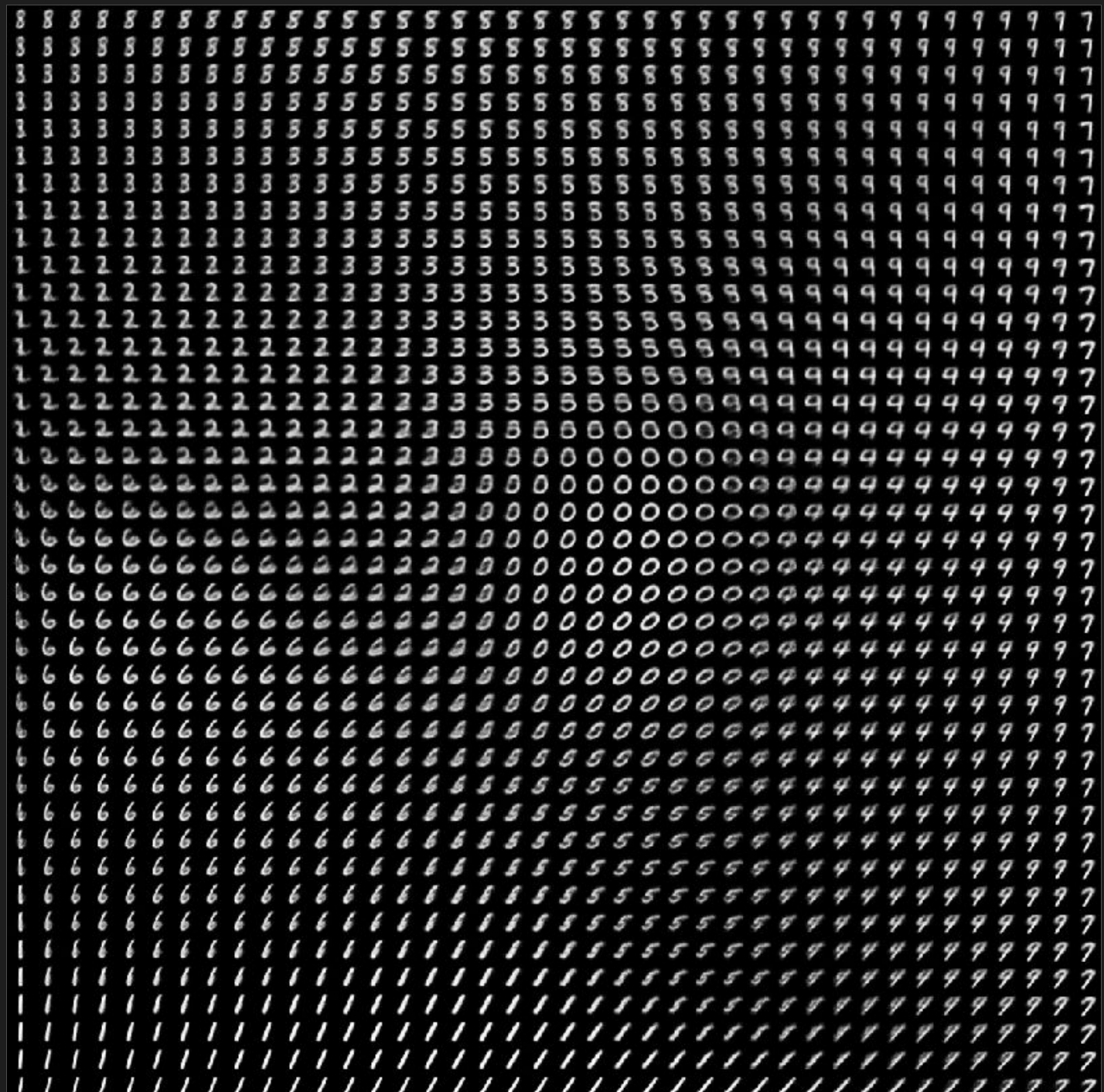
Voids (Discontinuity) in Latent Space

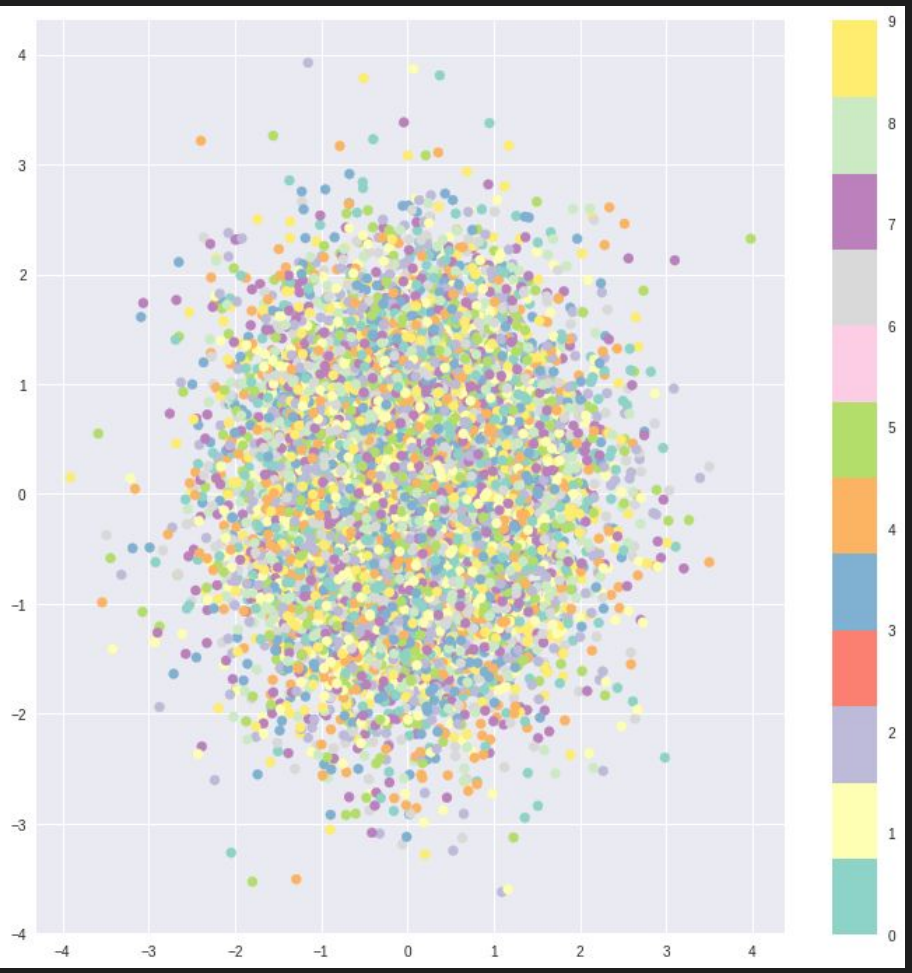


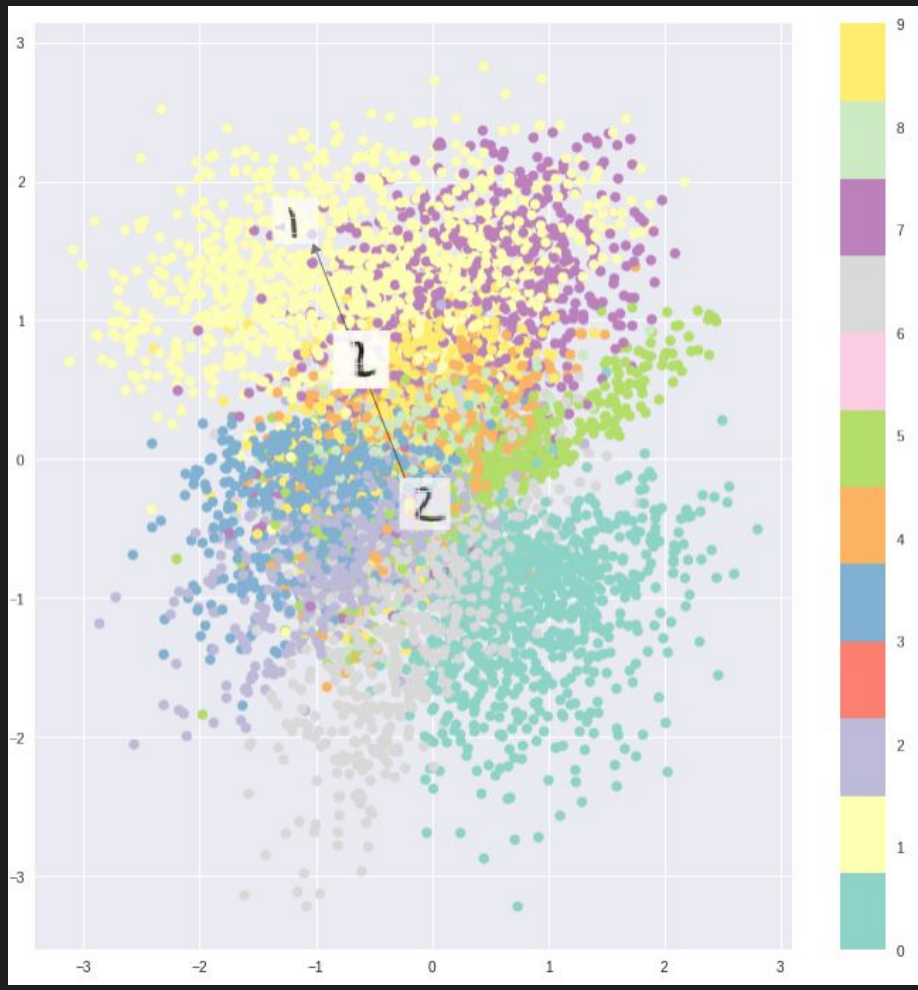
VAE Via reparameterization Trick




Encoder
Decoder
Original input
Compressed representation
Reconstructed input

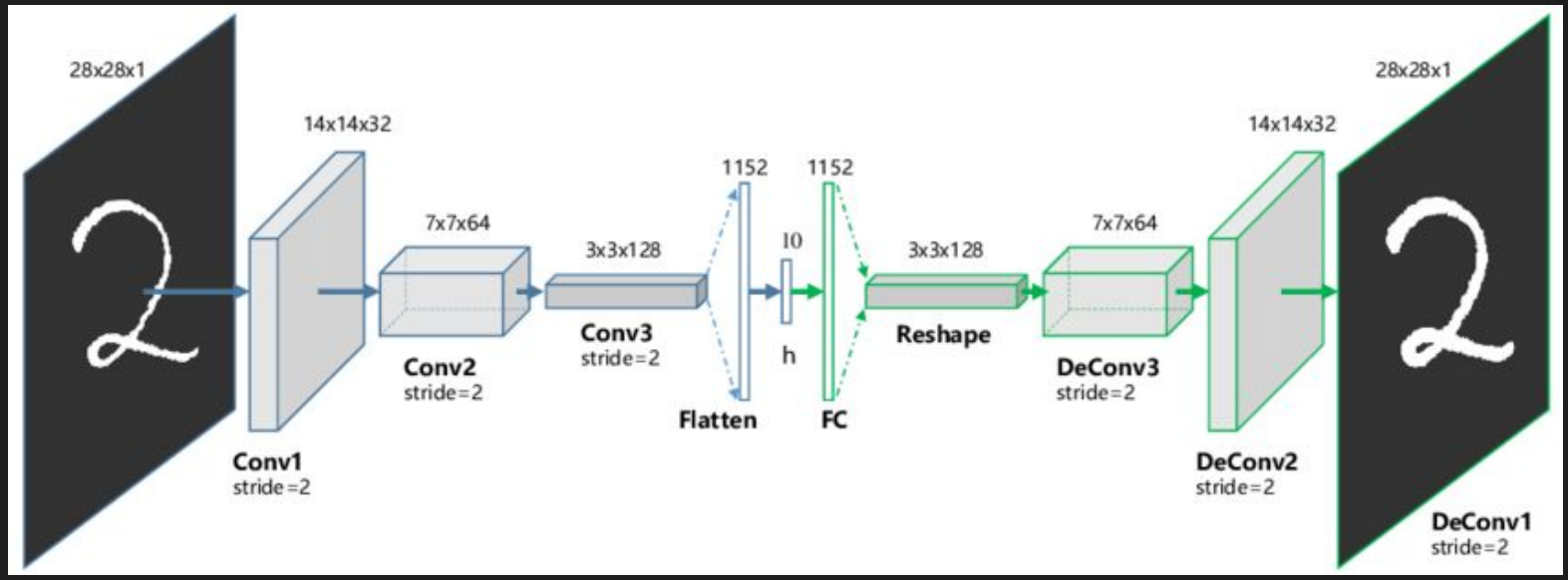



4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3
4
9
8
7
6
5
4
3
2
1
0

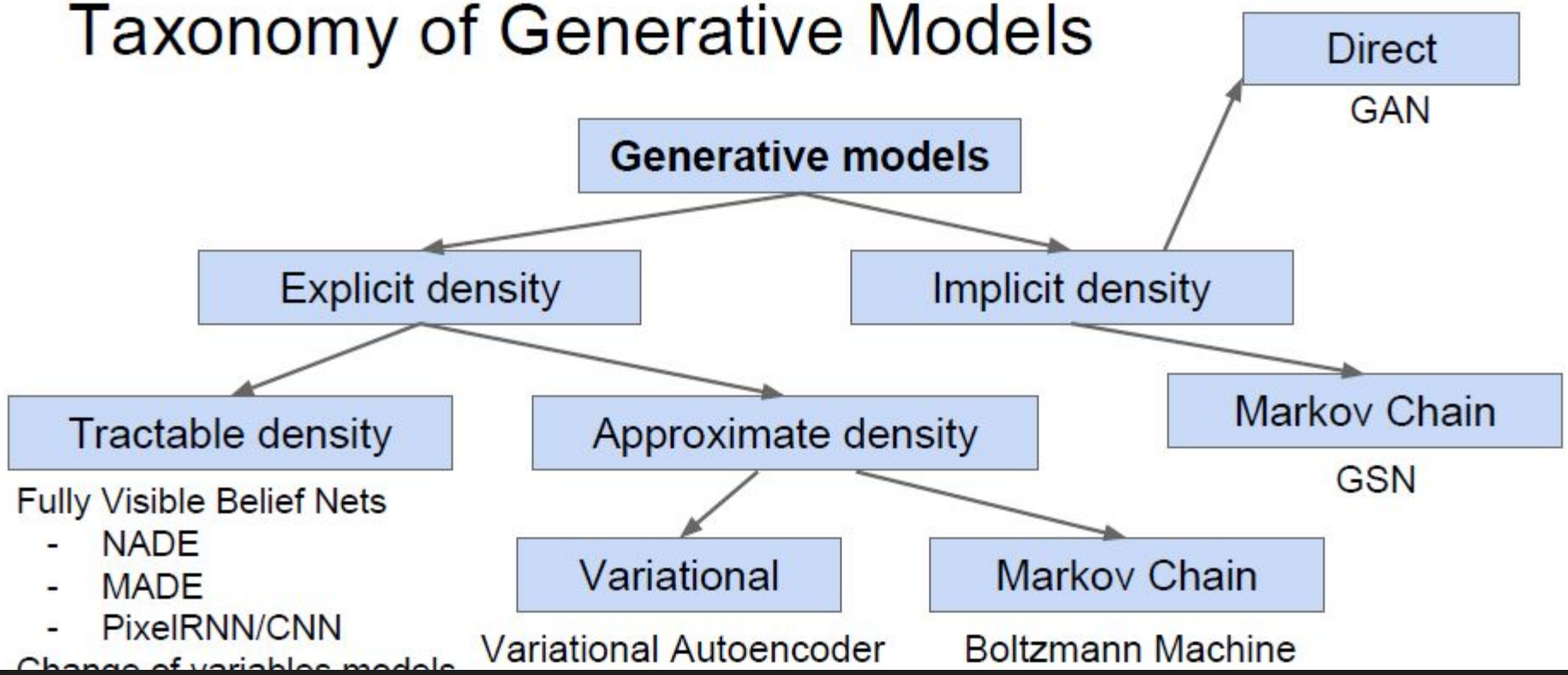

3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3
1
2
2
9
8
7
6
5
4
3
2
1
0

Variants of Auto encoder Basic units

Convolutional Autoencoder



Taxonomy of Generative Models



GAN: Generative adversarial Network

Train jointly in **minimax game**

Discriminator outputs likelihood in (0,1) of real image

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log \underbrace{D_{\theta_d}(x)}_{\text{Discriminator output for real data x}} + \mathbb{E}_{z \sim p(z)} \log(1 - \underbrace{D_{\theta_d}(G_{\theta_g}(z))}_{\text{Discriminator output for generated fake data G(z)}}) \right]$$

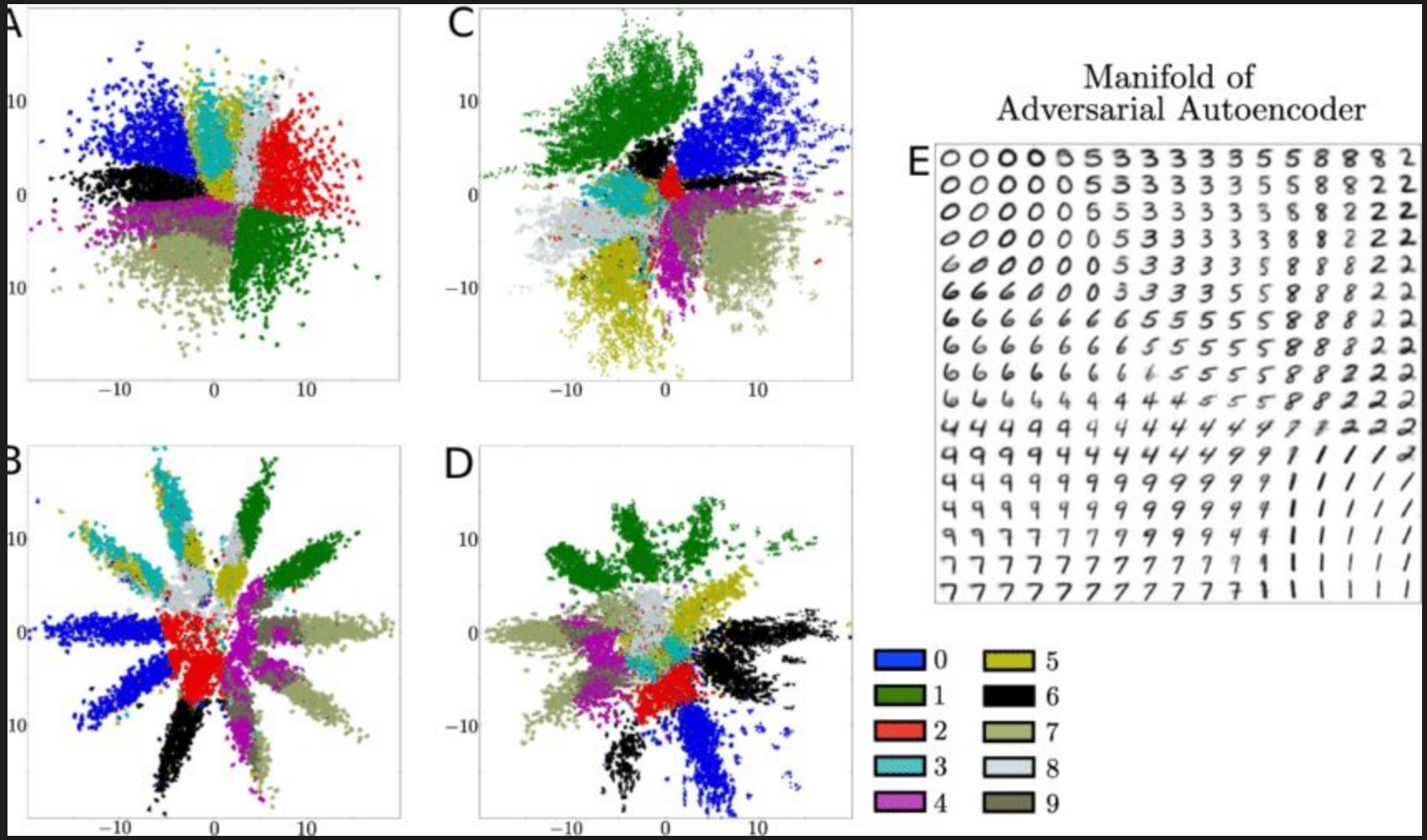
Alternate between:

1. **Gradient ascent** on discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

2. **Gradient descent** on generator

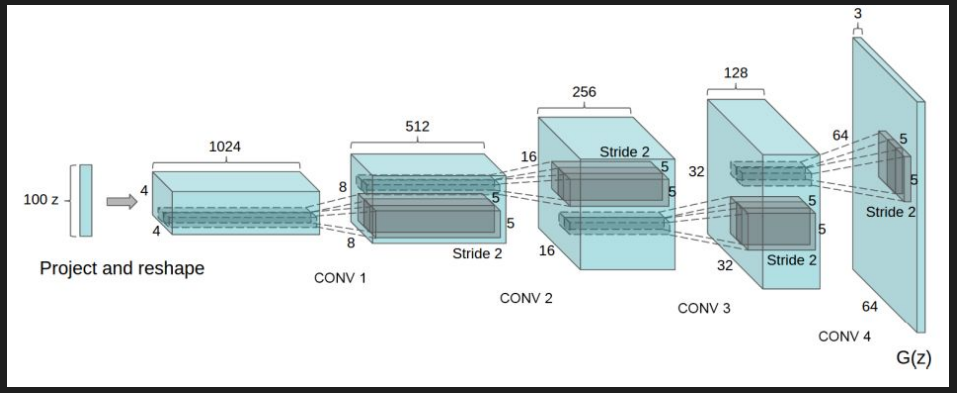
$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

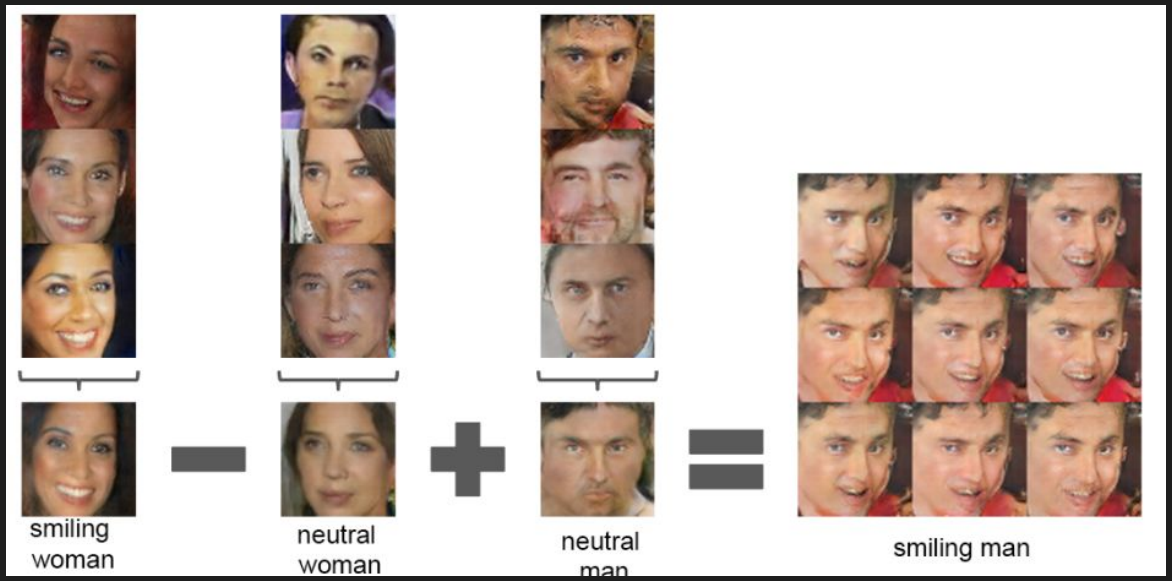


Comparison of adversarial and variational autoencoder on MNIST. The hidden code z of the hold-out images for an adversarial autoencoder fit to (a) a 2-D Gaussian and (b) a mixture of 10 2-D Gaussians. Each color represents the associated label. Same for variational autoencoder with (c) a 2-D gaussian and (d) a mixture of 10 2-D Gaussians. (e) Images generated by uniformly sampling the Gaussian percentiles along each hidden code dimension z in the 2-D Gaussian adversarial autoencoder.

Convolutional GAN

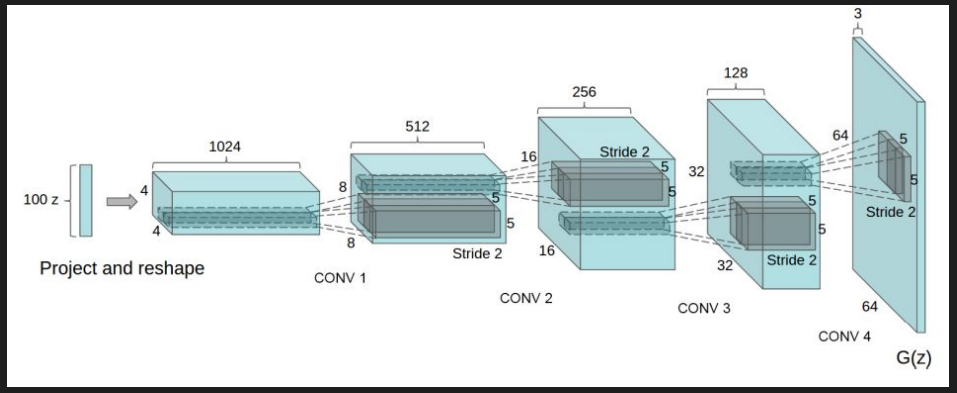
Convolutional GAN with deep CNN

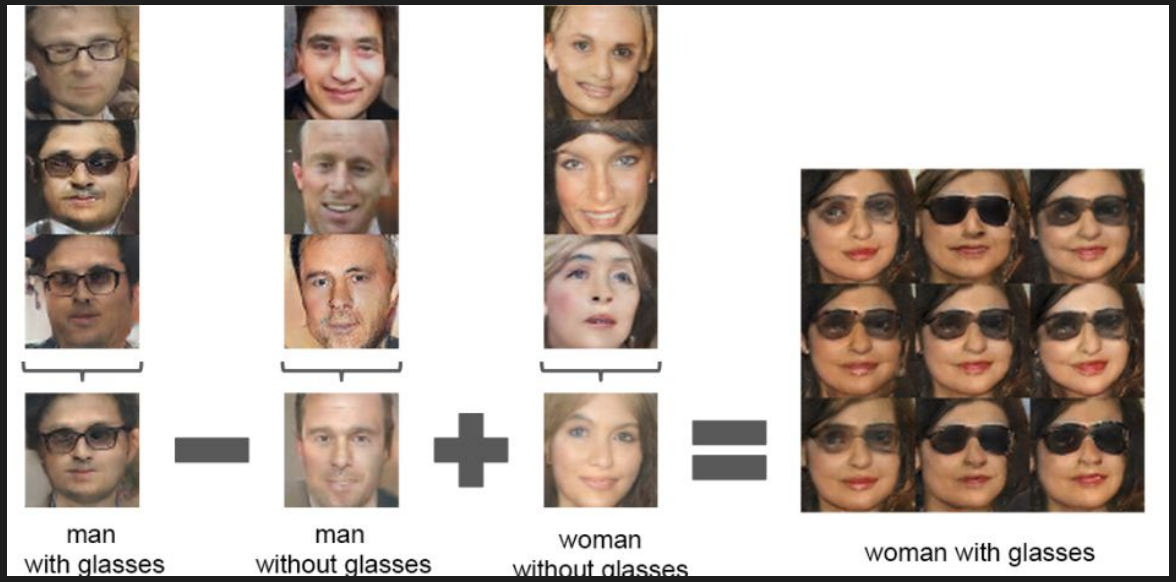




Convolutional GAN

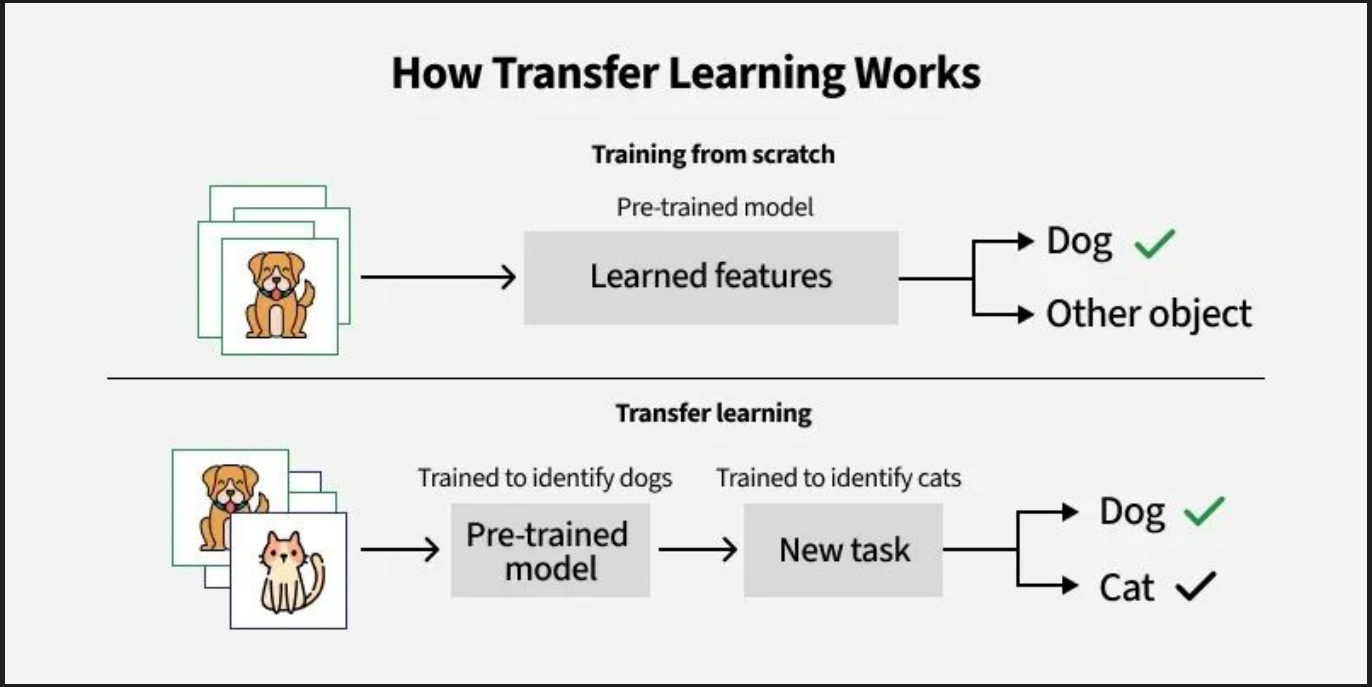
Convolutional GAN with deep CNN



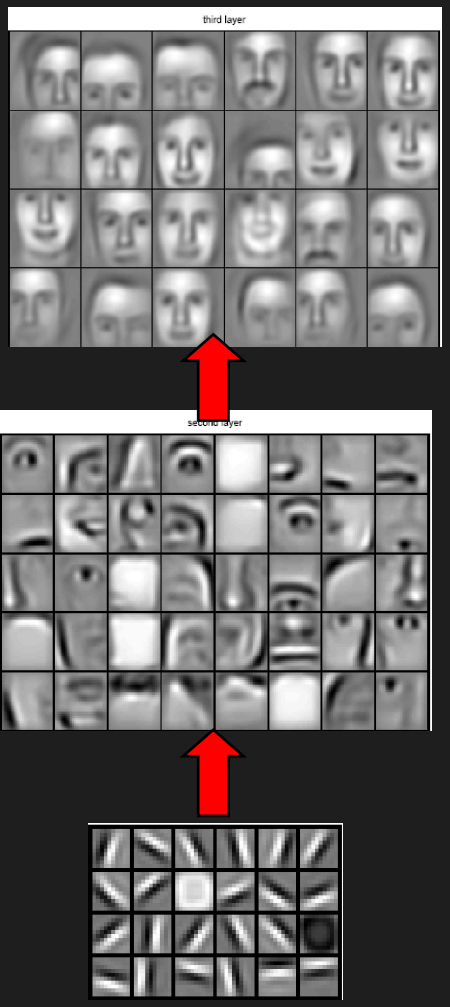


Deep Learning inception & Rise

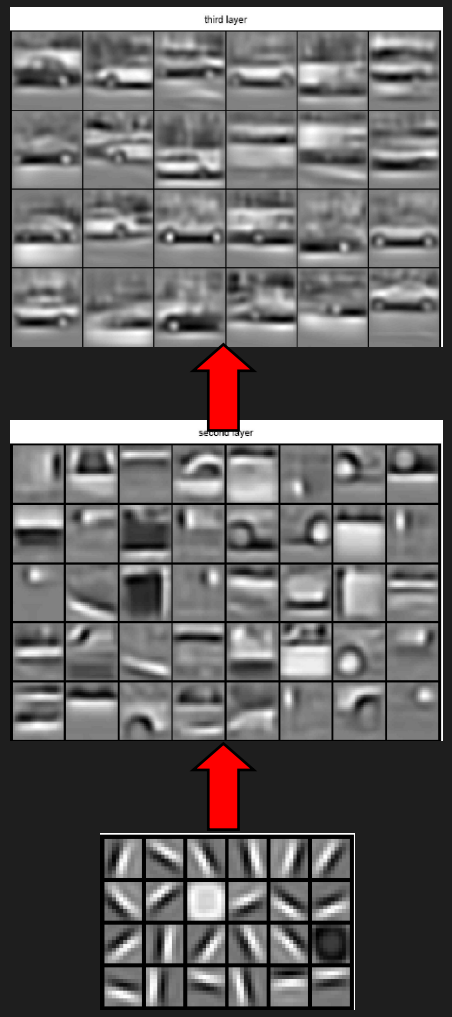
RBM and auto encodersat iception
ImageNET fueled with data
Exponential growth in DL models for Computer vision
CNN, Lenet, VGGNEt, InceptionNet, ResNet, etc
AE evolved to Encoder Decoder Architectures Where Y!=X
Transfer Learning



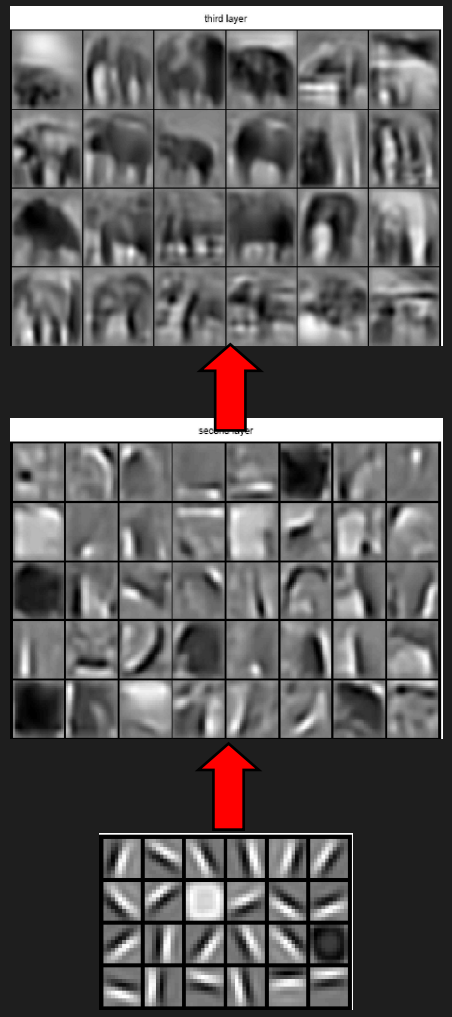
Faces



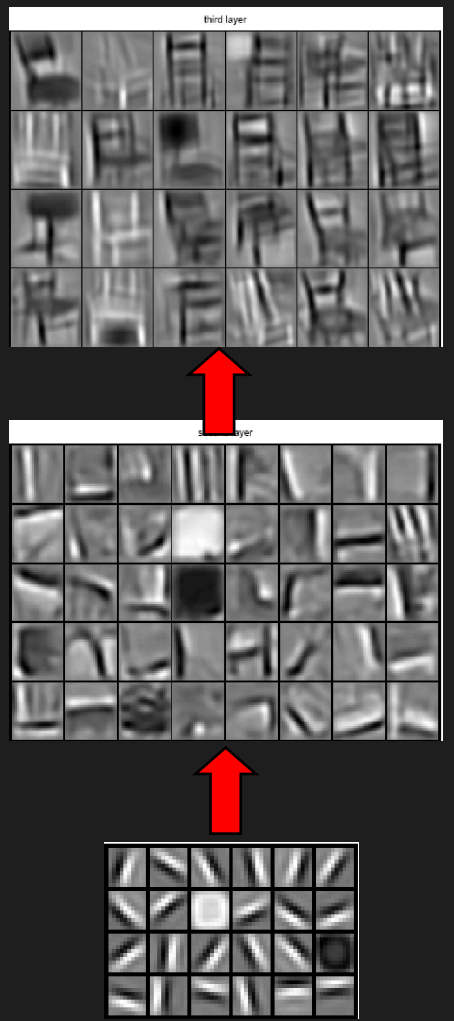
Cars



Elephants

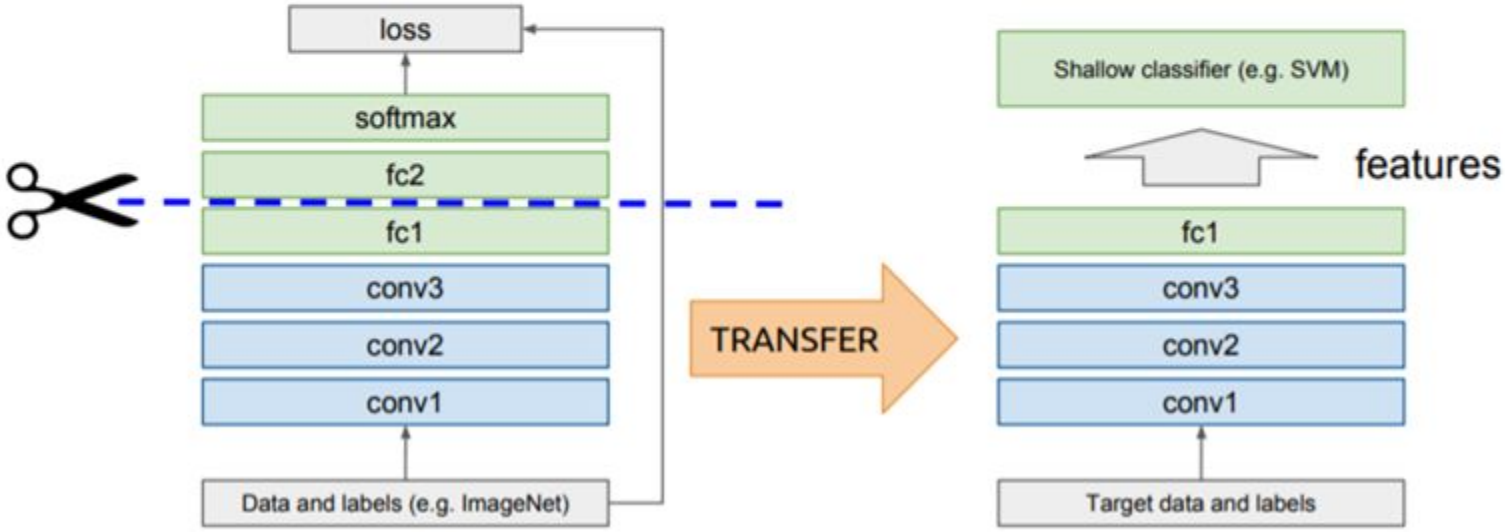


Chairs

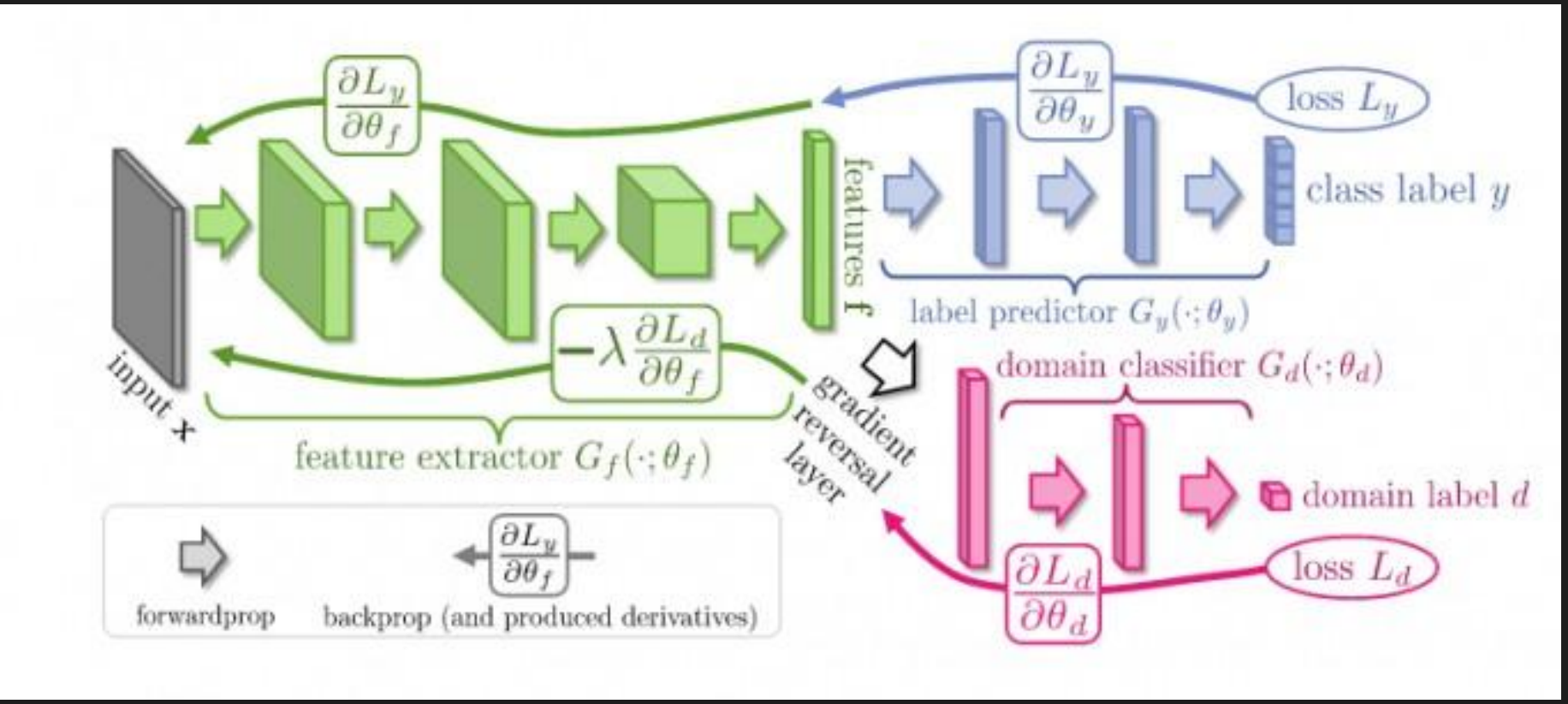


Idea: use outputs of one or more layers of a network trained on a different task as generic feature detectors. Train a new shallow model on these features.

Assumes that $D_S = D_T$

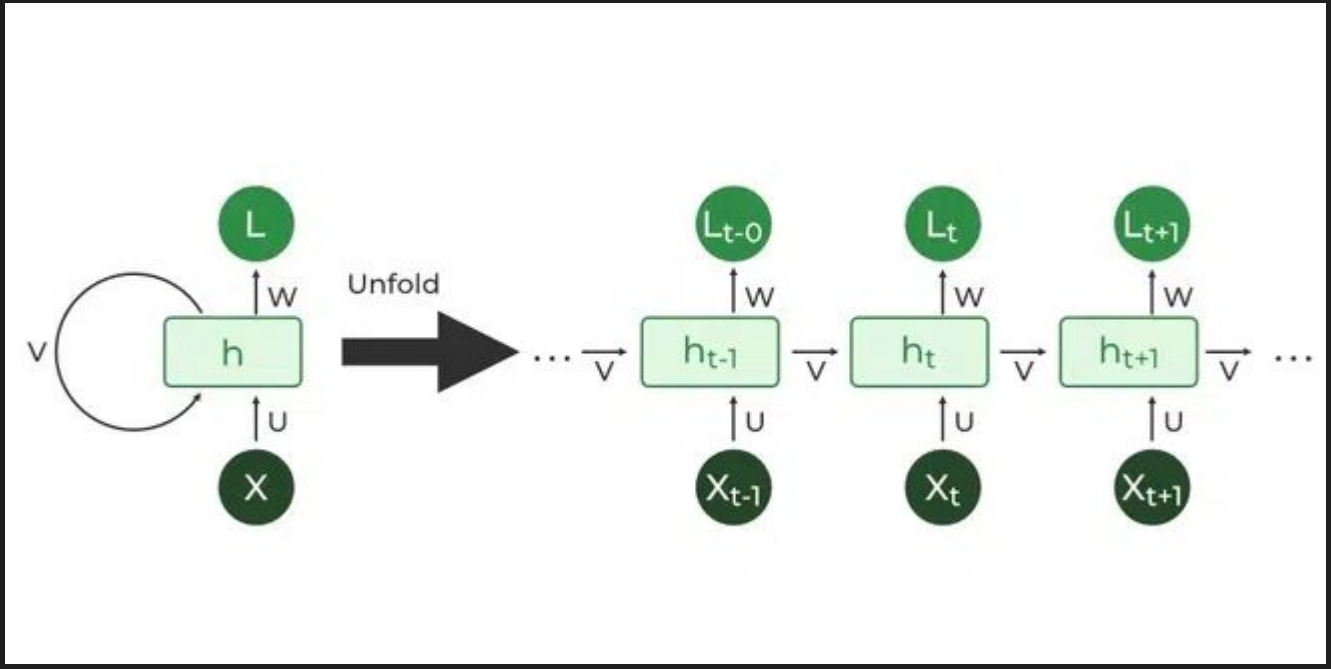


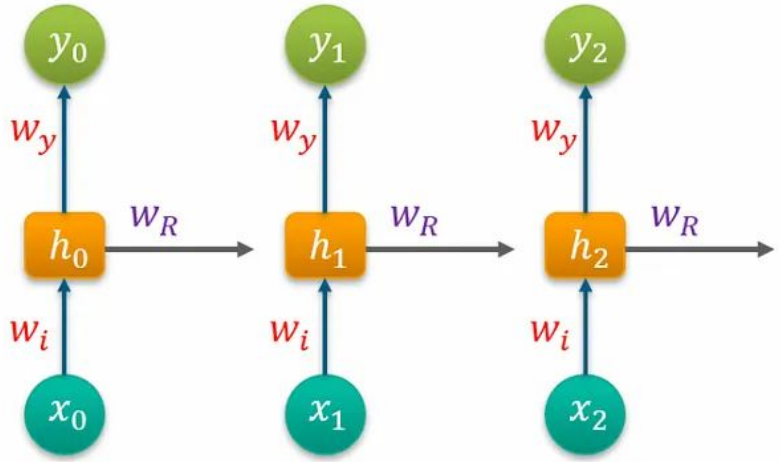
Domain-Adversarial Training of NN



Sequence models Catch up in Deep Learning

CNN a generalized version ANN is good with non sequential data
Forced to turn to RNN for temporal
CNN+LSTM or CNN +RNN: where CNN extracts local features and RNN/LSTM extracting sequential relationsships



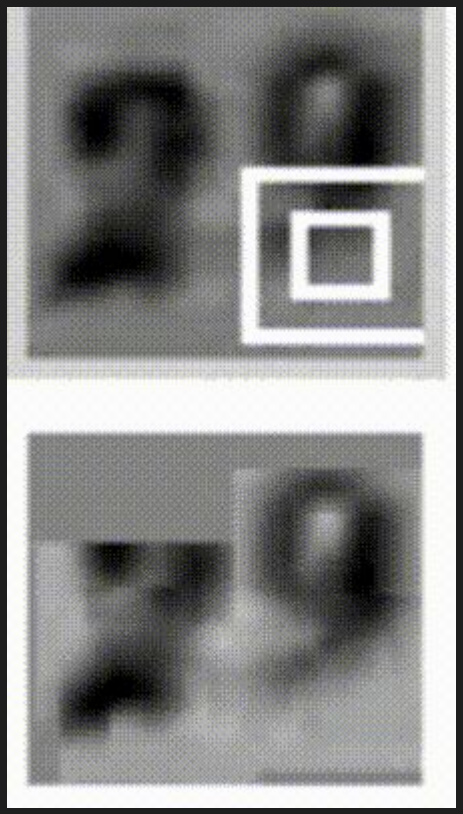


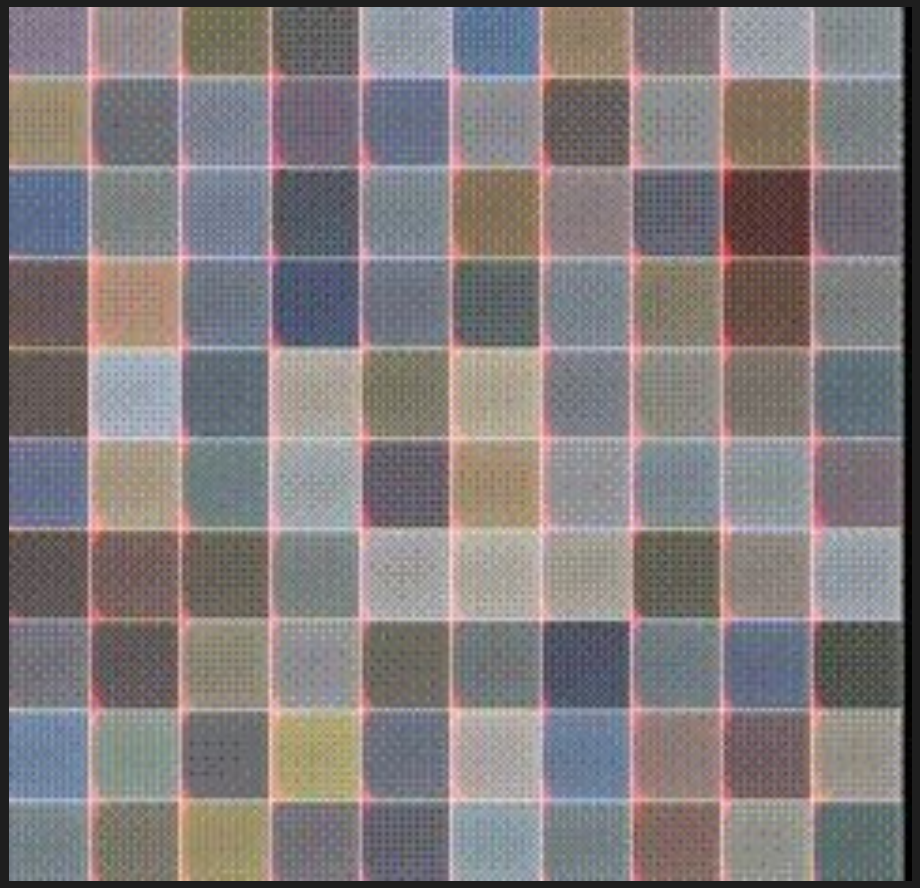
$$h^{(t)} = g_h\,(w_i x^{(t)} + w_R h^{(t-1)} + b_h)$$

$$y^{(t)} = g_y\,(w_y h^{(t)} + b_y)$$

Consider **'w'** to be the **weight matrix** and **'b'** being the **bias:**

At time **t=0,** input is **'x0'** and the task is to figure out what is **'h0'.** Substituting **t=0** in the **equation** and obtaining the function **h(t) value.** Next, the value of **'y0'** is found out using the **previously calculated values** when applied to the **new formula.**

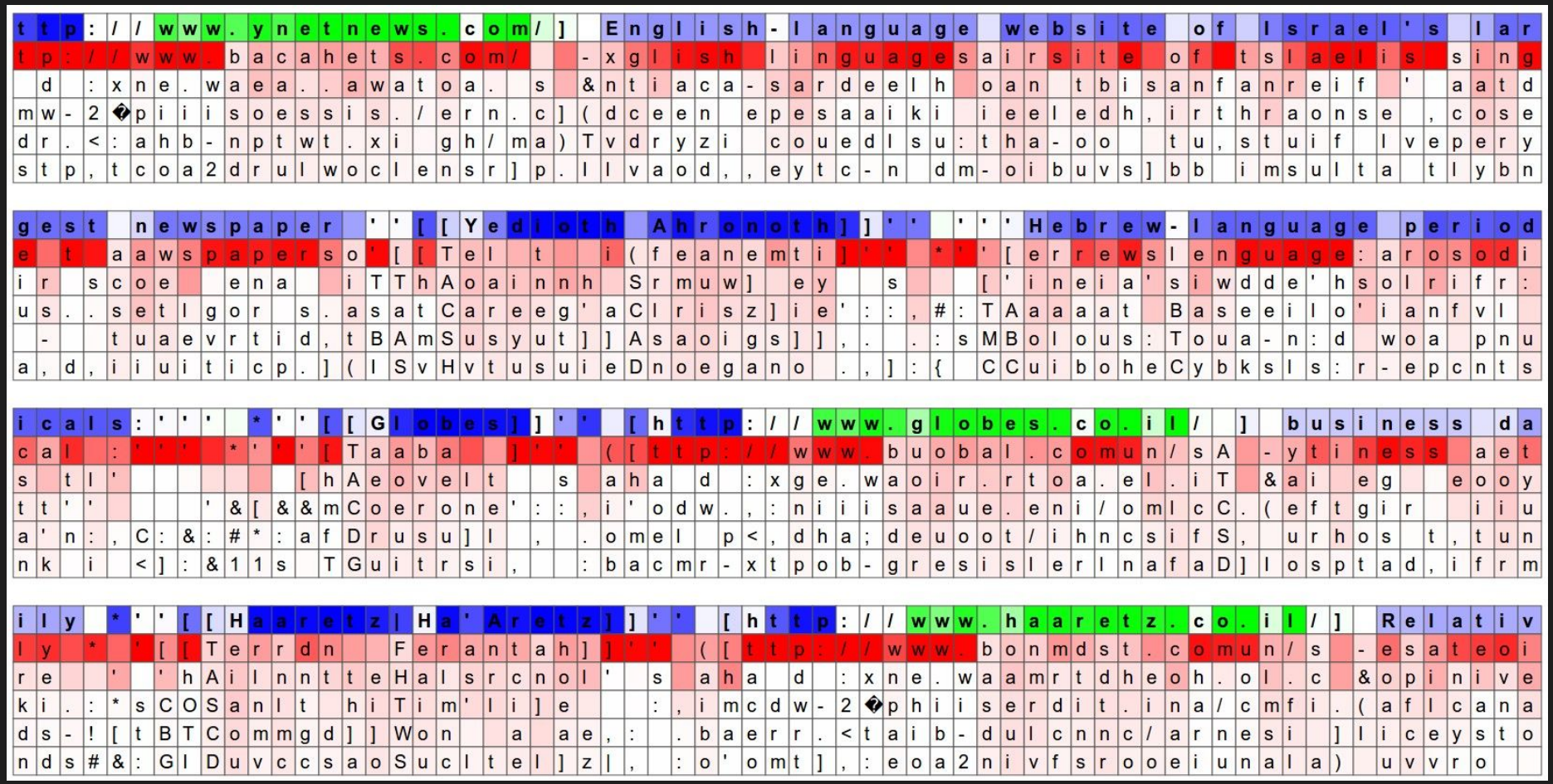




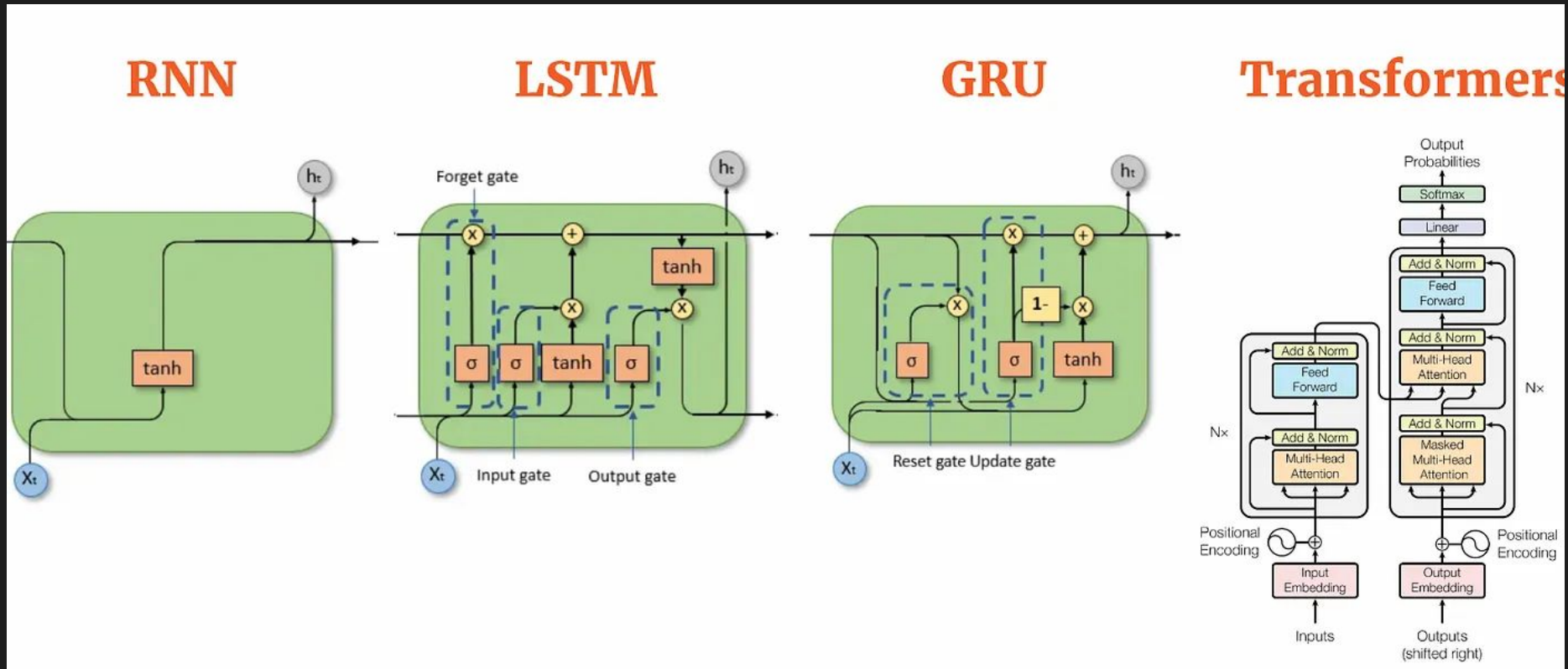
On the left, an algorithm learns a recurrent network policy that steers its attention around an image; In particular, it learns to read out house numbers from left to right (Ba et al.). On the right, a recurrent network *generates* images of digits by learning to sequentially add color to a canvas (Gregor et al.):

Sequence modeling Using RNN

Multiple RNN & Next Character Prediction



The neuron highlighted in this image seems to get very excited about URLs and turns off outside of the URLs. The LSTM is likely using this



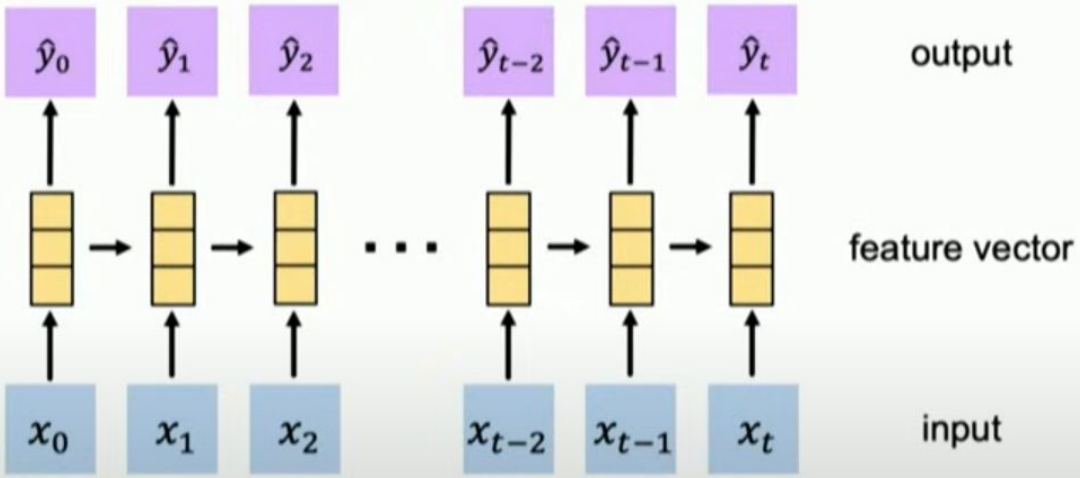
Title: Comparing different Sequence models: RNN, LSTM, GRU, and Transformers

Source: Colah's blog, and Attention paper. Compiled by AIML.com Research

Goal of Sequence Modeling

Desired Capabilities

- Continuous stream
- Parallelization
- Long memory



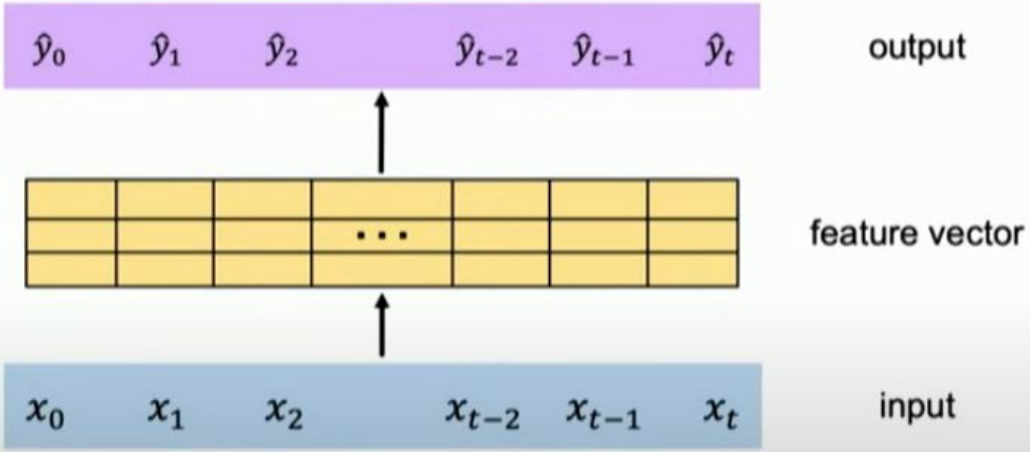
Goal of Sequence Modeling

Idea 1: Feed everything into dense network

- ✓ No recurrence
- ✗ Not scalable
- ✗ No order
- ✗ No long memory

Idea: Identify and attend to what's important

Can we eliminate the need for recurrence entirely?

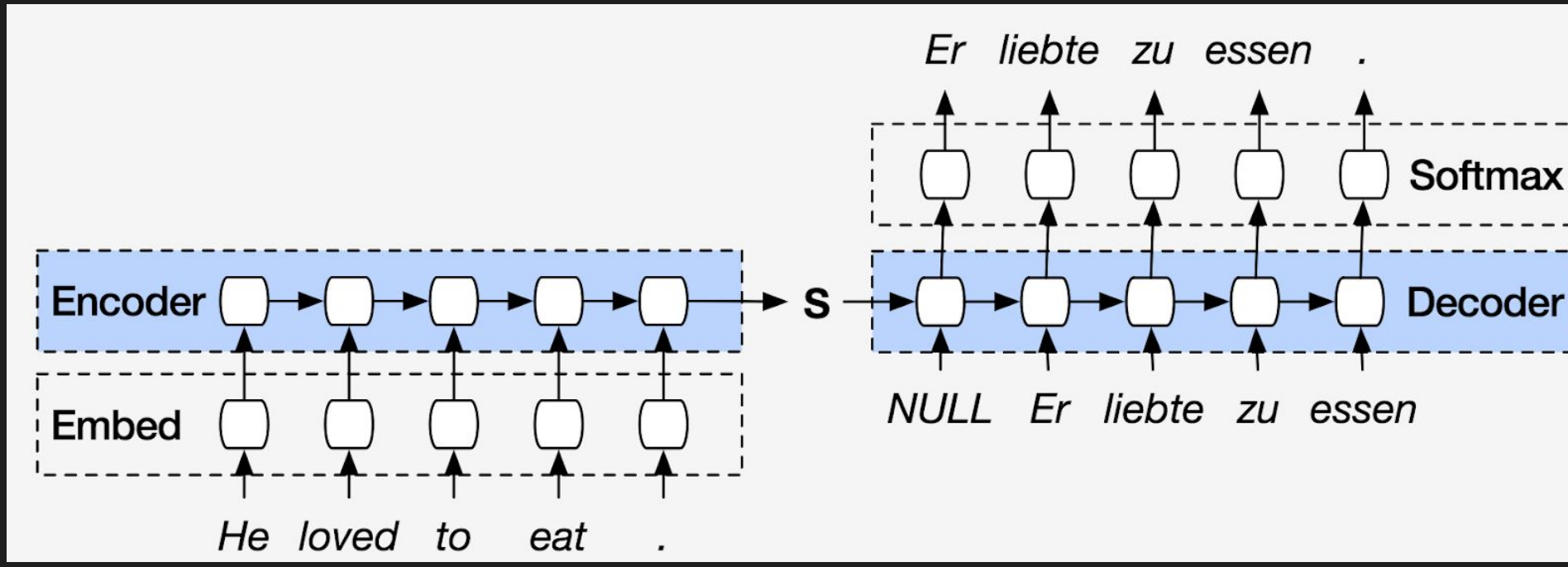


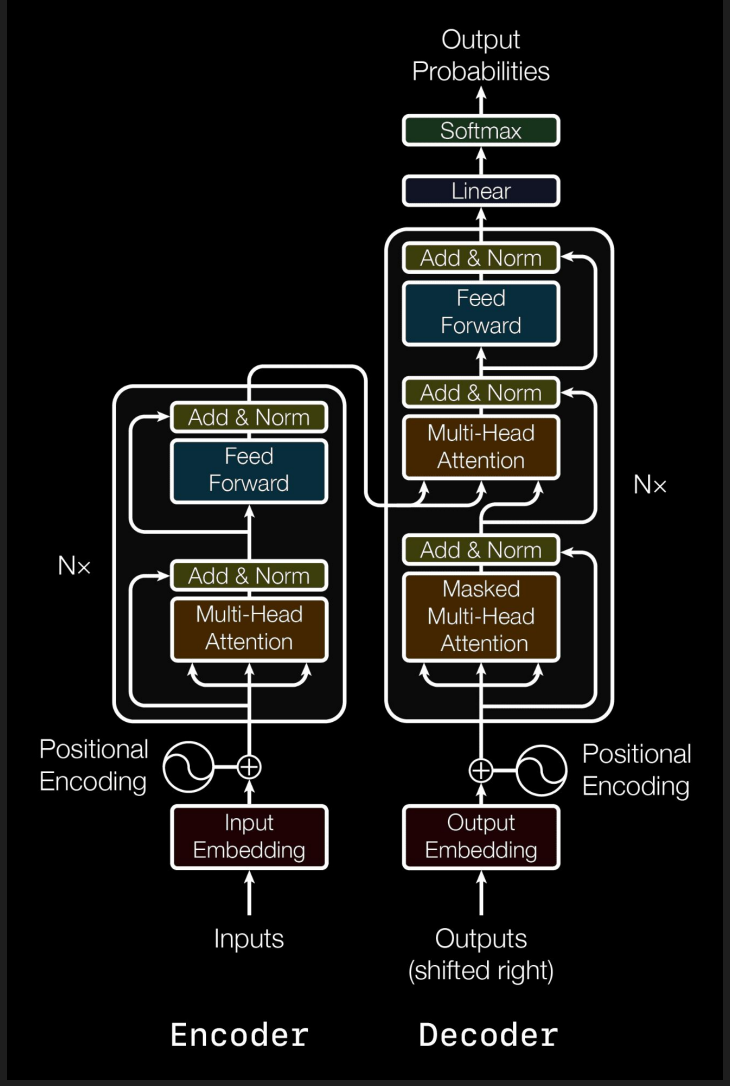
Intuition Behind Self-Attention

Attending to the most important parts of an input.

1. Identify which parts to attend to
2. Extract the features with high attention

Transformers: Attention is all you need




Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)
Encoder
Decoder

Understanding Attention with Search

YouTube

deep learning

Query (Q)

GIANT SEA TURTLES • AMAZING CORAL REEF FISH • 12 HOURS of THE BEST RELAX MUSIC

5.4M views • 4 years ago

Cat Trumpet

Enjoy 3 hours of giant sea turtles. This video features amazing coral reef fish and relaxing. Ideal for sleep, study and ...

MIT 6.S191 (2020): Introduction to Deep Learning

1M views • 1 year ago

Alexander Amini

MIT Introduction to Deep Learning 6.S191: Lecture 1 Foundations of Deep Learning Lecturer: Amini January 2020 For ...

CC

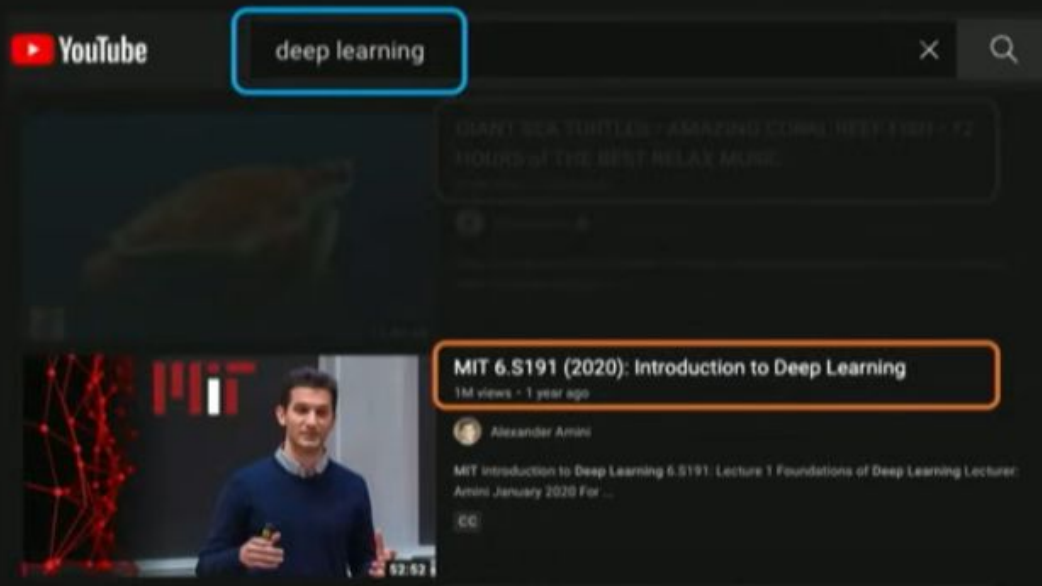
The Kobe Bryant Fadeaway Shot

235K views • 1 year ago

NBAtv

Postup #Footwork #PartOne No copyright infringement is intended, all audio and video clips property of their ...

Understanding Attention with Search



Query (Q)

Key (K_1)

Key (K_2)

Key (K_3)

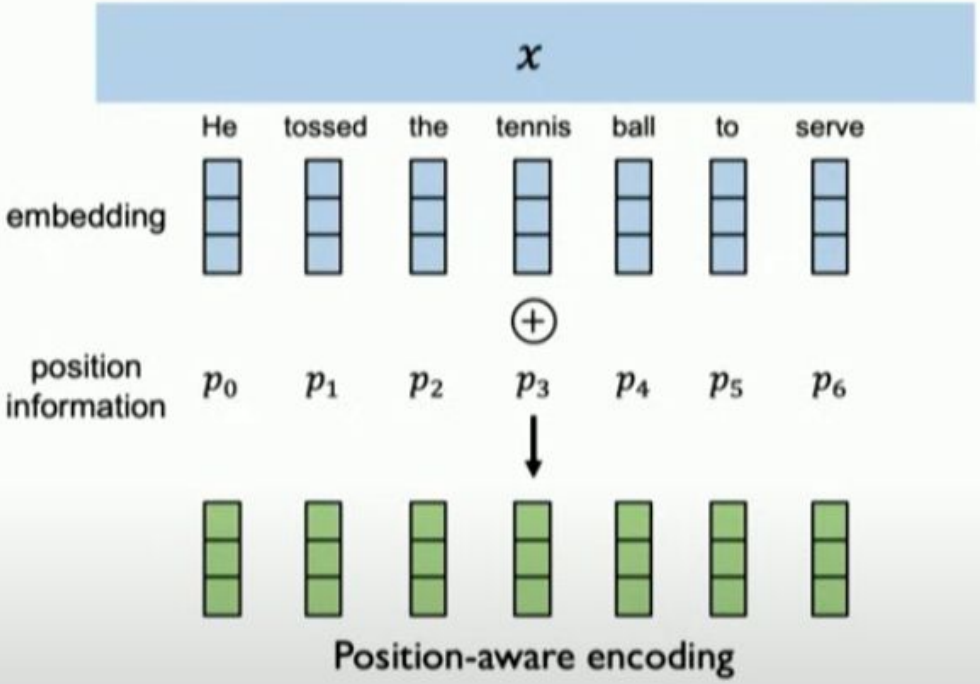
How similar is the key to the query?

1. **Compute attention mask:** how similar is each key to the desired query?

Learning Self-Attention with Neural Networks

Goal: identify and attend to most important features in input.

1. Encode **position** information
2. Extract query, key, value for search
3. Compute attention weighting
4. Extract features with high attention

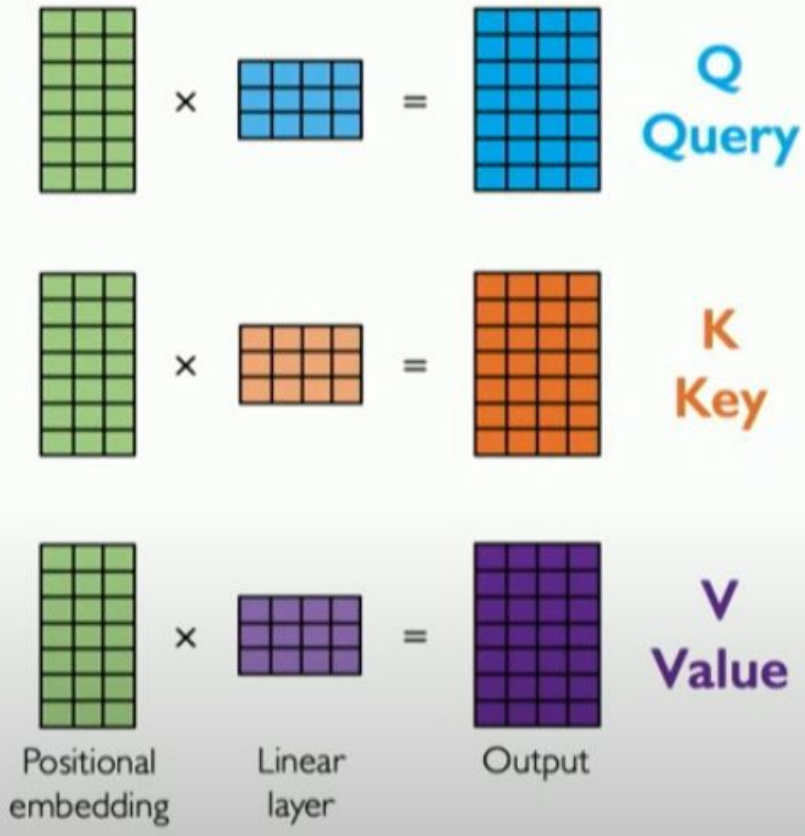


Data is fed in all at once! Need to encode position information to understand order.

Learning Self-Attention with Neural Networks

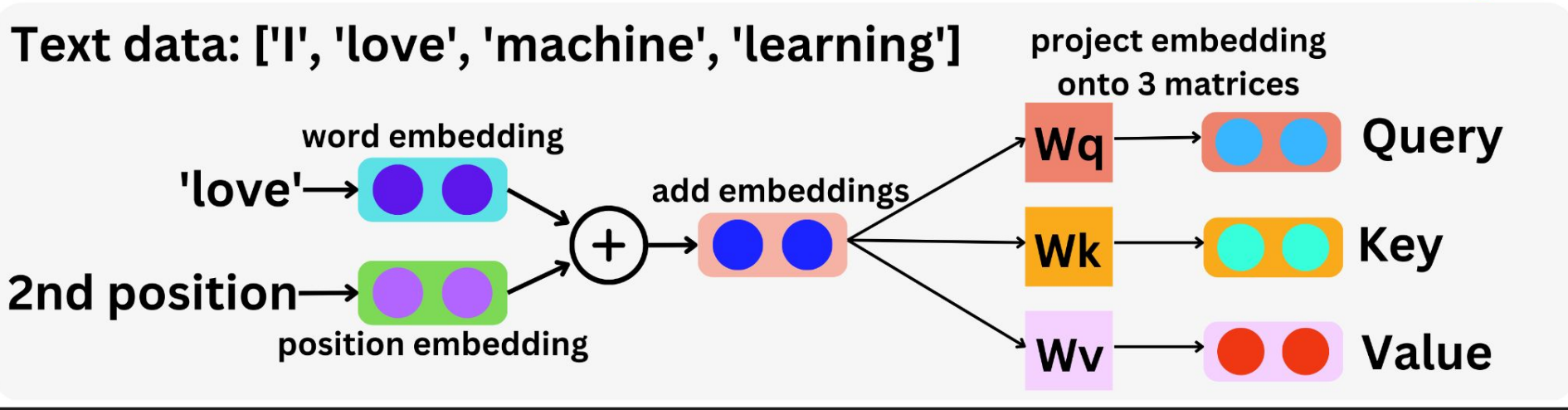
Goal: identify and attend to most important features in input.

1. Encode **position** information
2. Extract **query, key, value** for search
3. Compute attention weighting
4. Extract features with high attention



Step 1: Create the Query, Key, Value

TheAiEdge.io



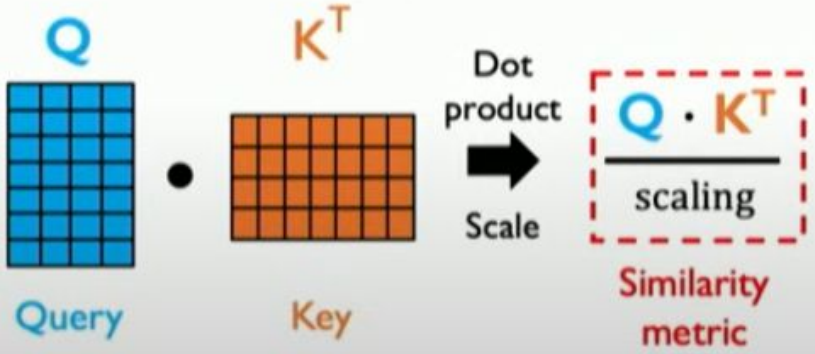
Learning Self-Attention with Neural Networks

Goal: identify and attend to most important features in input.

1. Encode **position** information
2. Extract **query, key, value** for search
3. Compute **attention weighting**
4. Extract features with high attention

Attention score: compute pairwise similarity between each query and key

How to compute similarity between two sets of features?



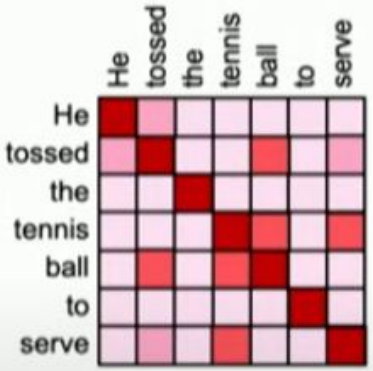
Also known as the "cosine similarity"

Learning Self-Attention with Neural Networks

Goal: identify and attend to most important features in input.

1. Encode **position** information
2. Extract **query, key, value** for search
3. Compute **attention weighting**
4. Extract features with high attention

Attention weighting: where to attend to! How similar is the key to the query?



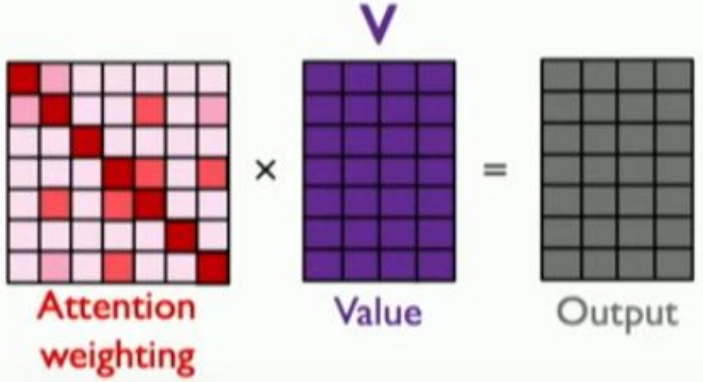
$$softmax\left(\frac{Q \cdot K^T}{scaling}\right)$$

Learning Self-Attention with Neural Networks

Goal: identify and attend to most important features in input.

1. Encode **position** information
2. Extract **query, key, value** for search
3. Compute **attention weighting**
4. Extract **features with high attention**

Last step: self-attend to extract features



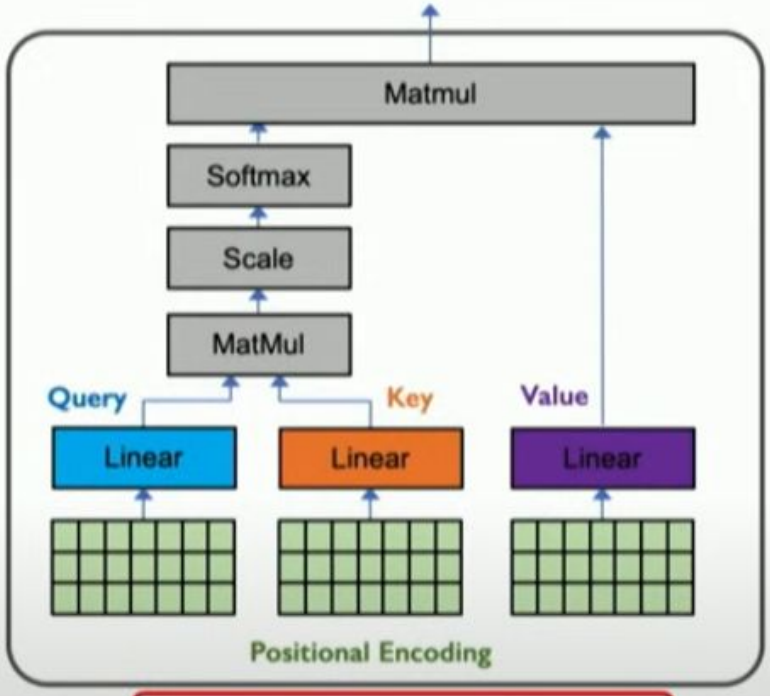
$$softmax\left(\frac{Q \cdot K^T}{scaling}\right) \cdot V = A(Q,K,V)$$

Learning Self-Attention with Neural Networks

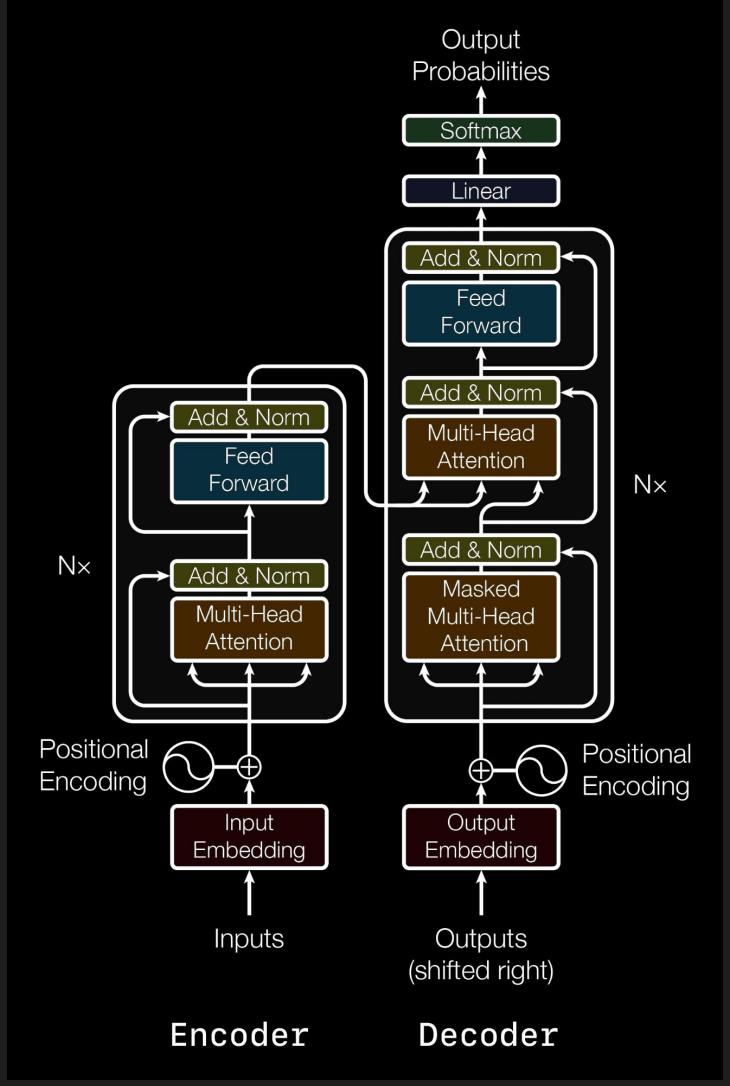
Goal: identify and attend to most important features in input.

1. Encode **position** information
2. Extract **query, key, value** for search
3. Compute **attention weighting**
4. Extract **features with high attention**

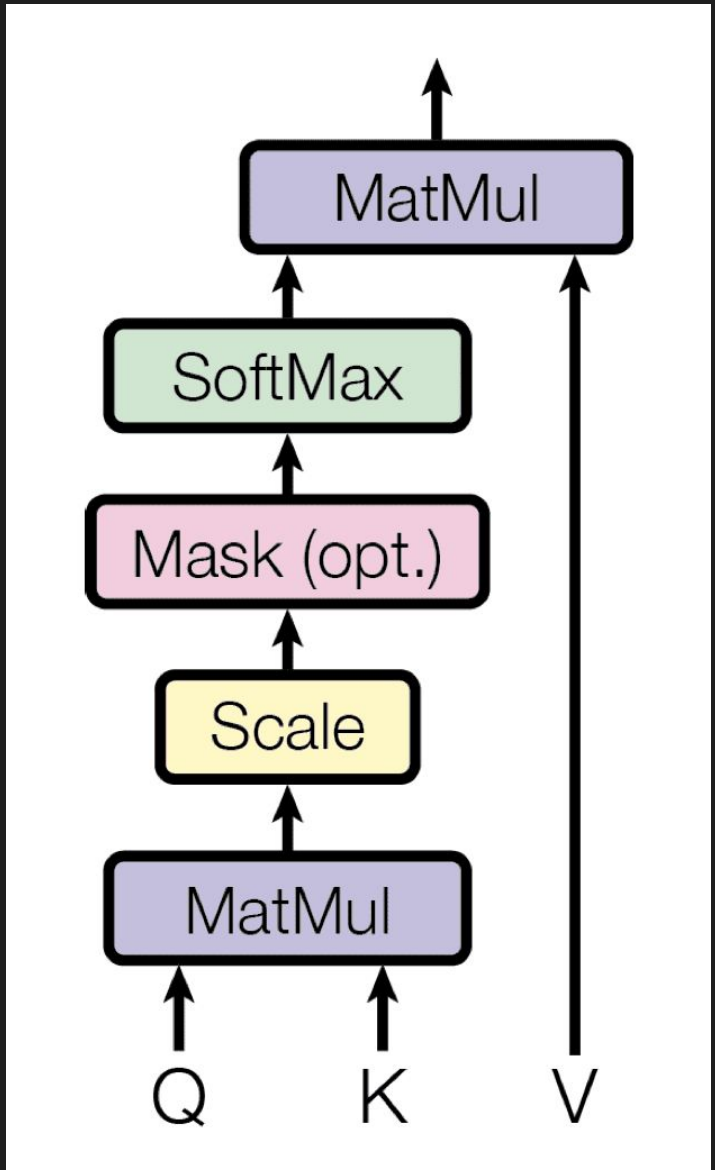
These operations form a self-attention head that can plug into a larger network. Each head attends to a different part of input.



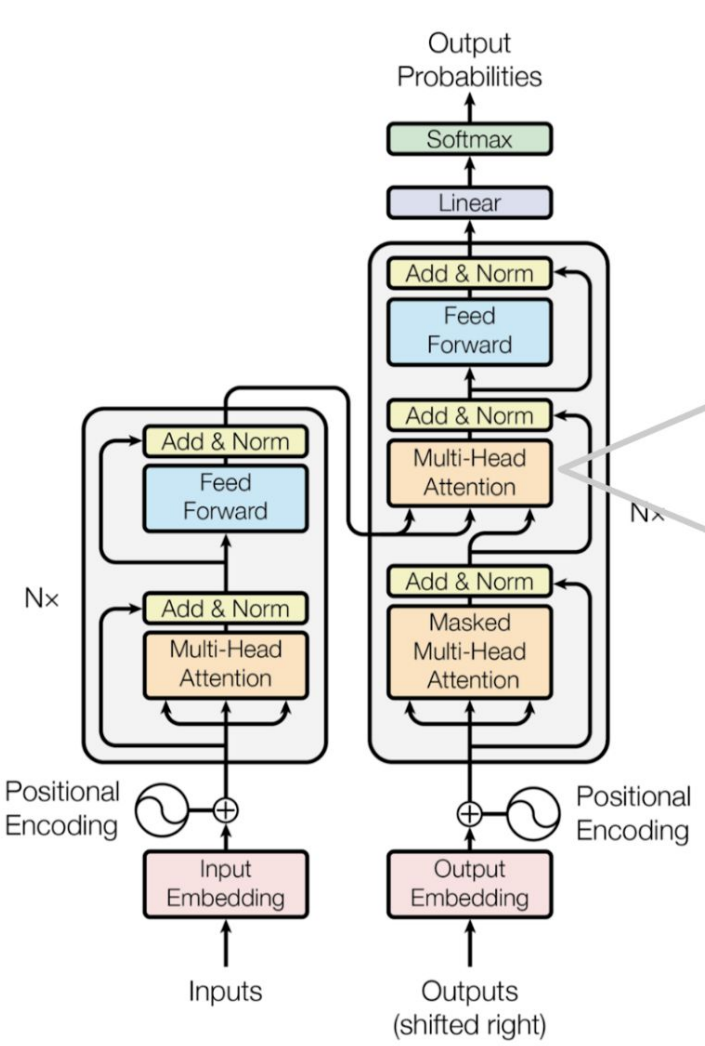
$$softmax\left(\frac{Q \cdot K^T}{scaling}\right) \cdot V$$

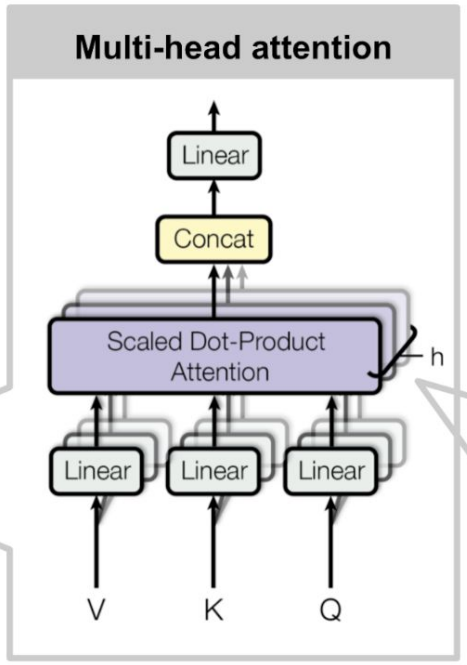

Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)
Encoder
Decoder

Transformer: Attention to context

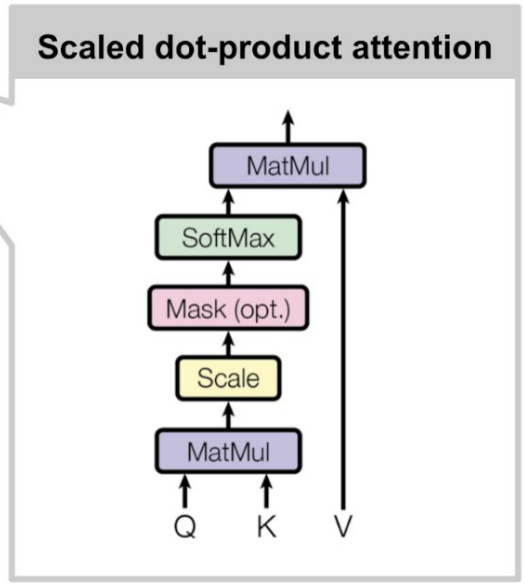


Multi Head attention



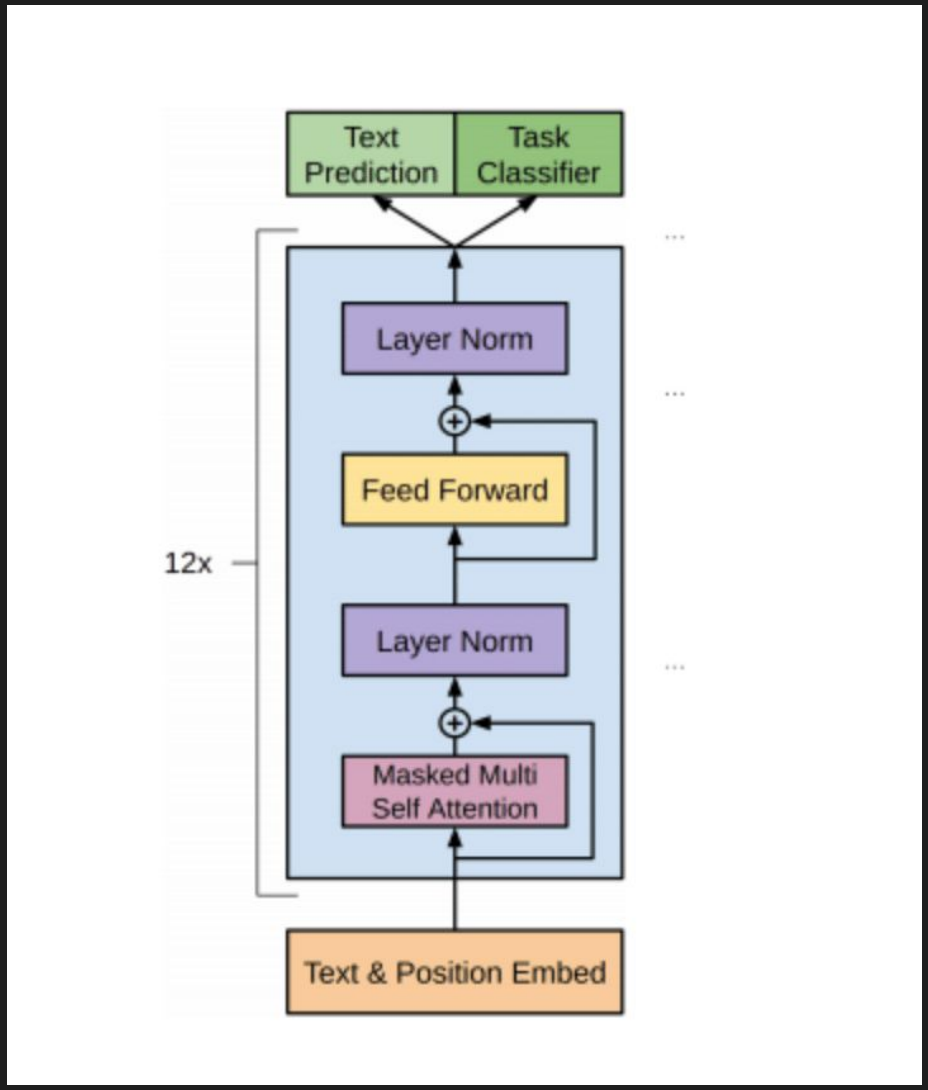


Zoom-In!

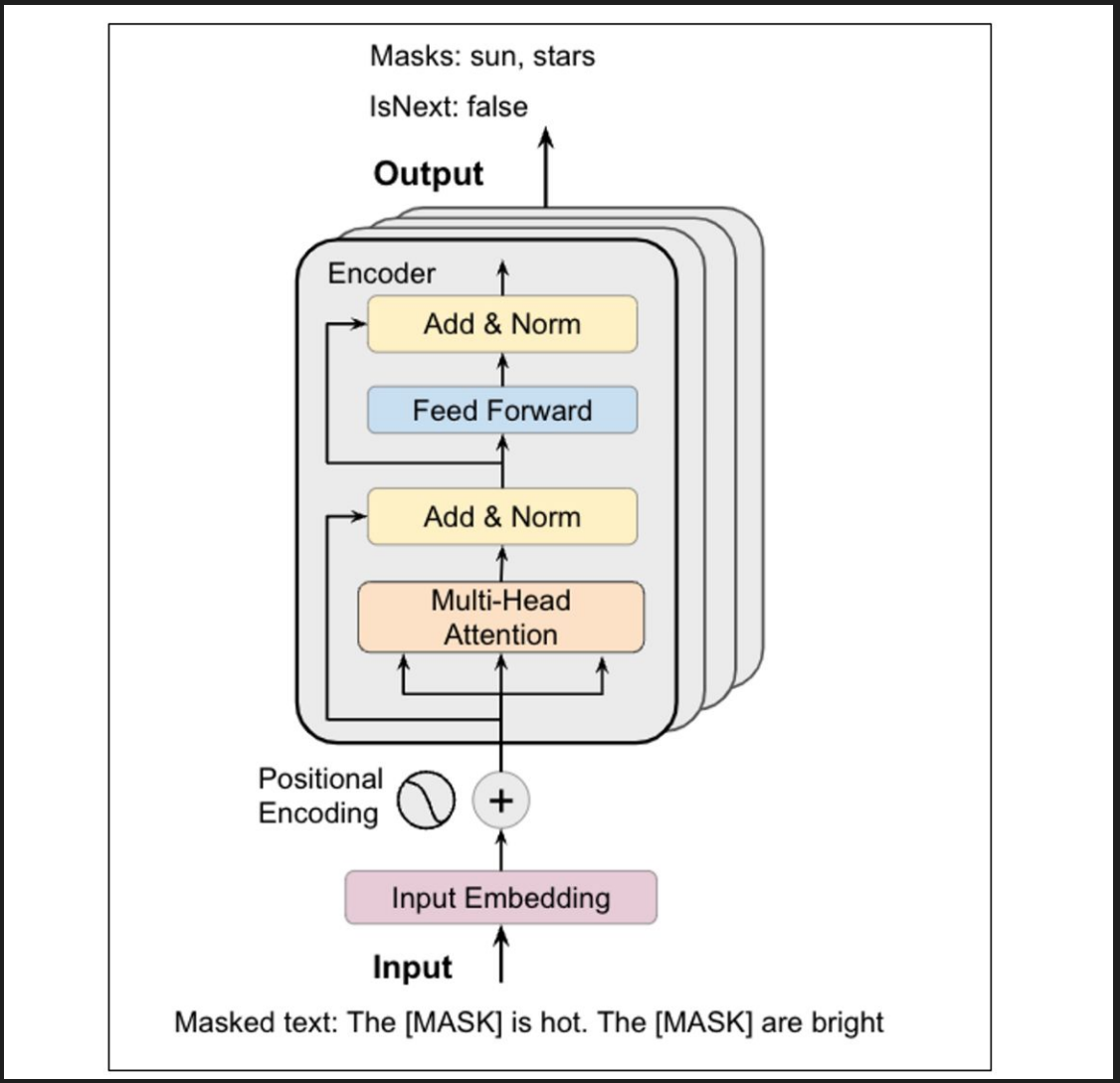


Zoom-In!

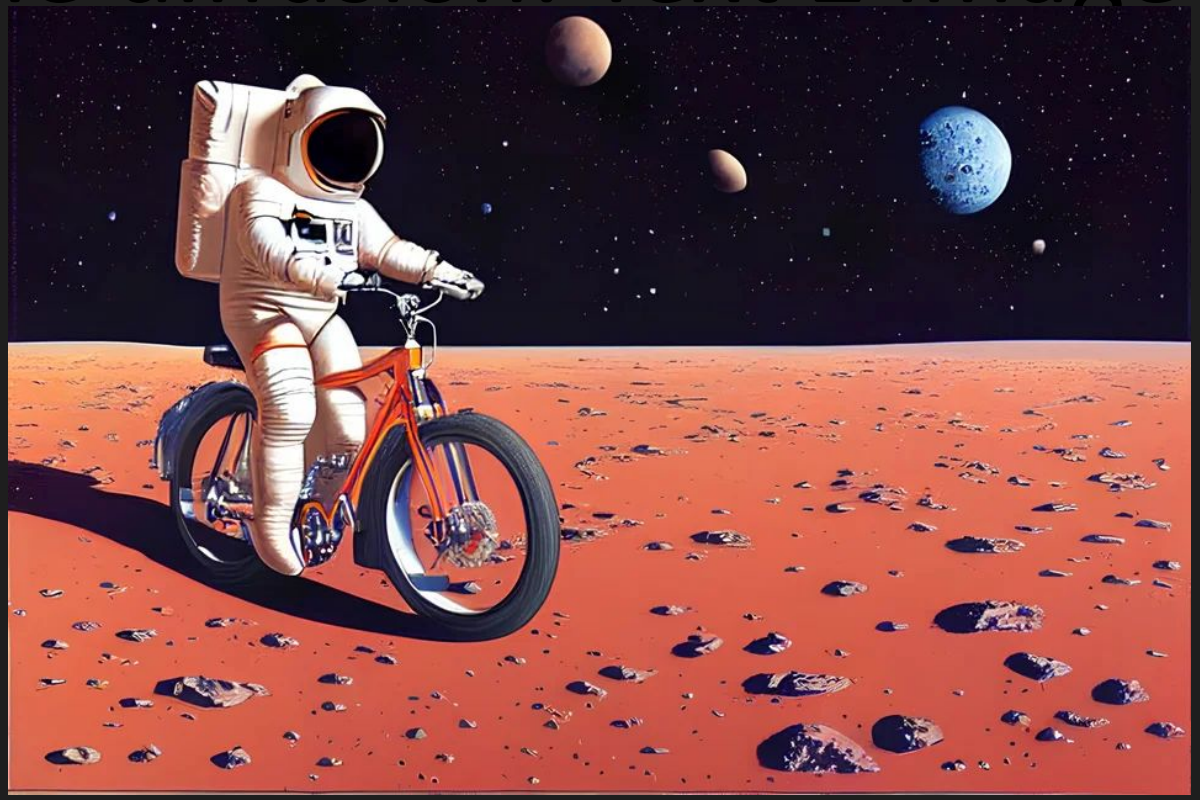
GPT 2 Architecture



Bert Architecture



Stable diffusion: Text 2 image



https://scholar.harvard.edu/files/binxuw/files/stab
fusion_a_tutorial.pdf

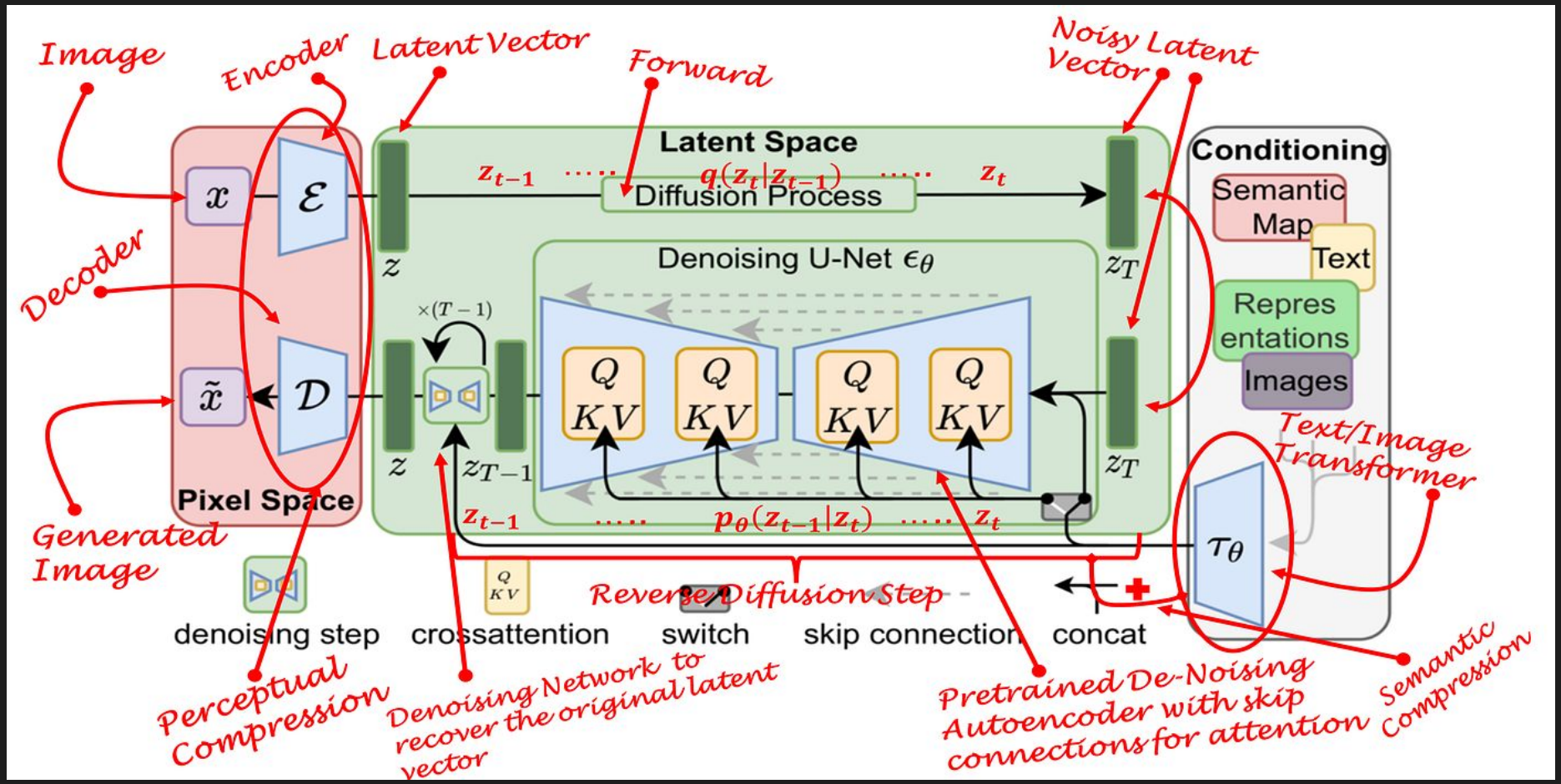


Image
Encoder
Latent Vector
Forward
Noisy Latent Vector
Latent Space
Conditioning
Diffusion Process
Denoising U-Net ϵ_θ
Semantic Map
Text
Representations
Images
Decoder
Pixel Space
Generated Image
Text/Image Transformer
Reverse Diffusion Step
denoising step
crossattention
switch
skip connection
concat
Perceptual Compression
Denoising Network to recover the original latent vector
Pretrained De-Noising Autoencoder with skip connections for attention
Semantic Compression

Bridge to PyTorch Lab

Hands-on with PyTorch for NLP
Build a classifier, language model
Link theory to implementation