CODE MATRIX

SYLLA N'FALY

DEEPLEARNING ARCHITECTURES

FOCUS: CNN & RNN

SYLLA N'FALY | ENG. STUDENT, COMPUTER SCIENCE

CODEMATRIX

10/08/2025

I'M,

SYLLA N'FALY

Engineering student in Computer Science at ENSA Tangier, Morocco

A passionate Data Science enthusiast

in/8sylla 8sylla @8sylla

“Seeing” the progress of deep learning throughout the years

2015
Goodfellow et al.

2018
Karras, Laine, Aila.

2020
MIT Intro to Deep Learning

3 months ago
That is easily the cleanest visual deepfake I've ever seen. It must have taken ages to render, because it just looks flawless.

2 months ago
WOW WOW WOW i am amazed.

5 months ago
THAT INTRO TO THE LECTURE IS SAVAGE!!!

3 months ago
This is the best example of a Course that sells itself.

2020

…creating this 2 minute video required…

2 hours of professional audio

50 hours of HD video

Static, pre-defined script

Over $15K USD of compute

2020

…creating this 2 minute video required…

2 hours of professional audio

50 hours of HD video

Static, pre-defined script

Over $15K USD of compute

2025

…fast forward a few years…

?

OUR PLAN

01 | **The Fondations**

02 | **Computer Vision**

FCNN & CNN

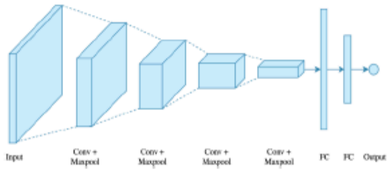
03 | **Deep Sequence Modeling**

RNN, LSTM & GRU

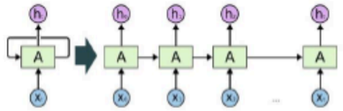
04 | **Q/R**

DEEP LEARNING MODEL ZOO

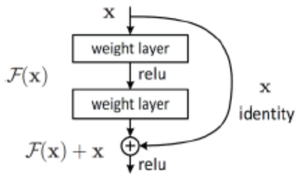
CNN



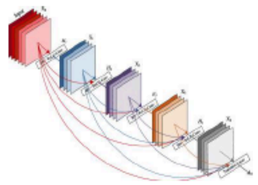
RNN



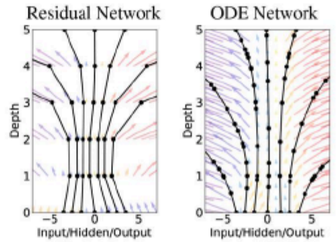
ResNet



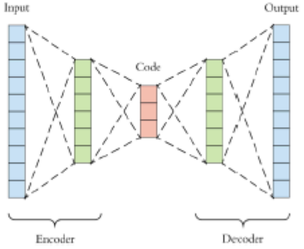
DenseNet



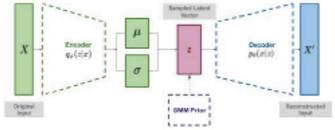
NeuralODE



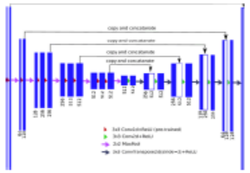
AutoEncoder



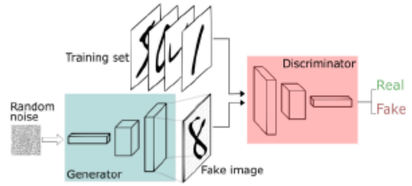
VAE



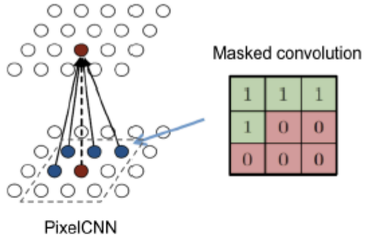
Unet



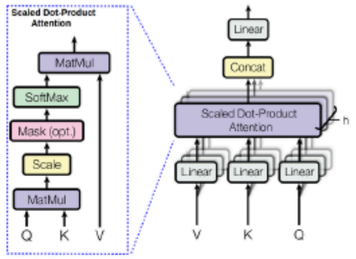
GAN



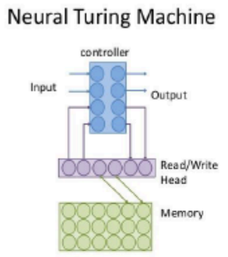
Partial Convolutions



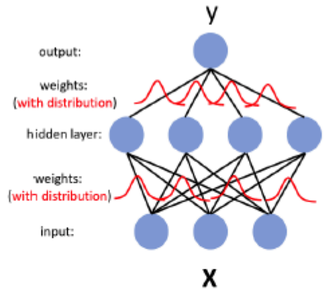
Transformer



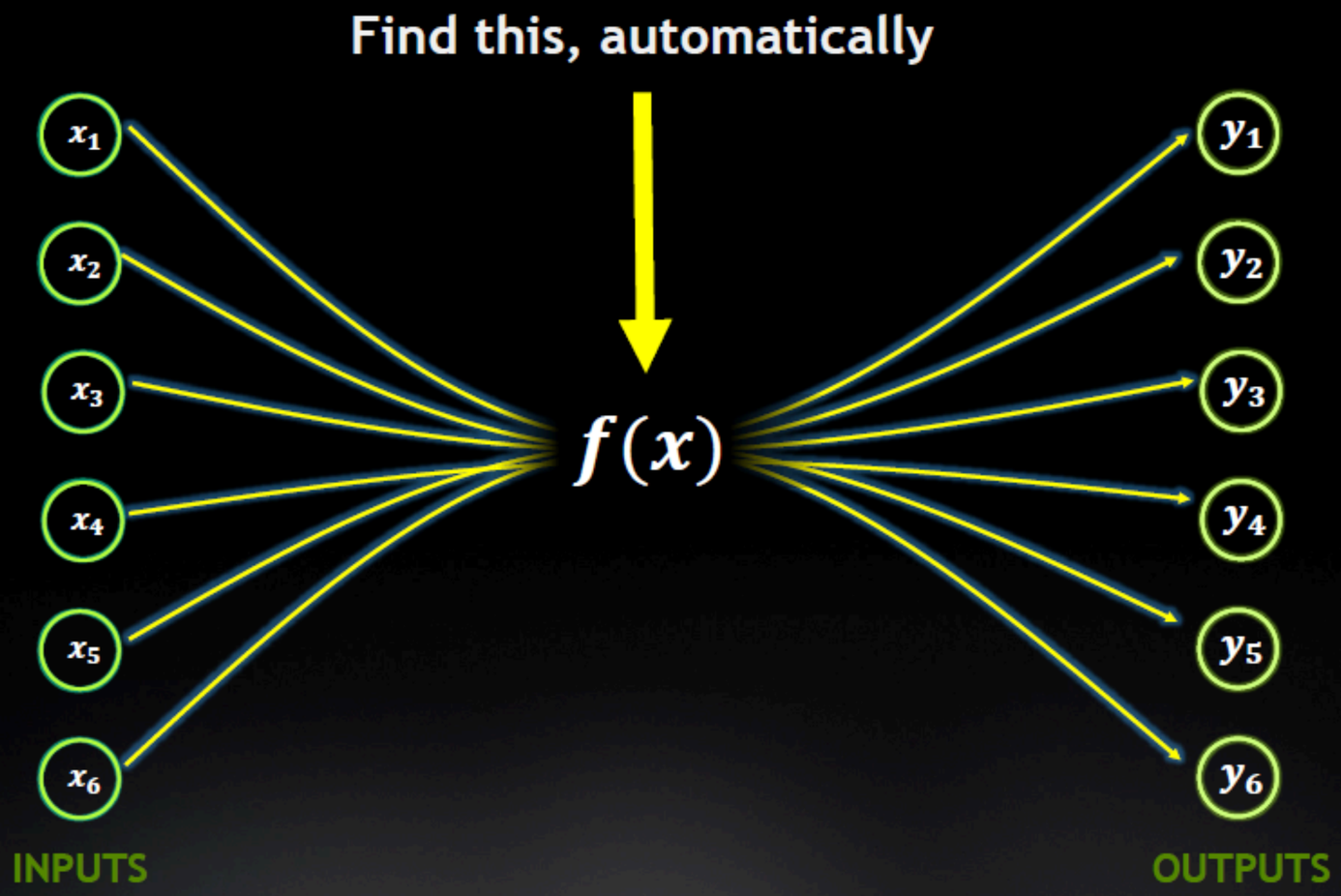
Neural Turing Machine



Bayesian Neural Net

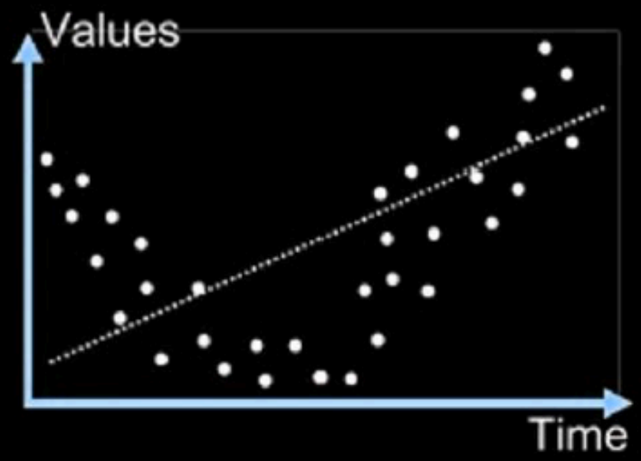


REVERSE-ENGINEER FUNCTIONS FROM EXAMPLES

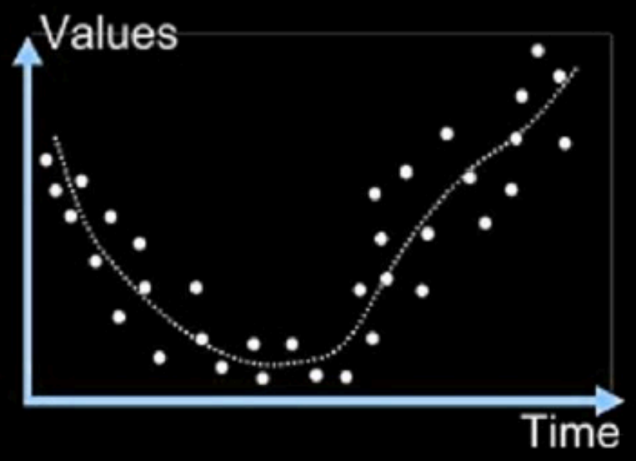


ADJUST MODEL CAPACITY TO FIT YOUR DATA

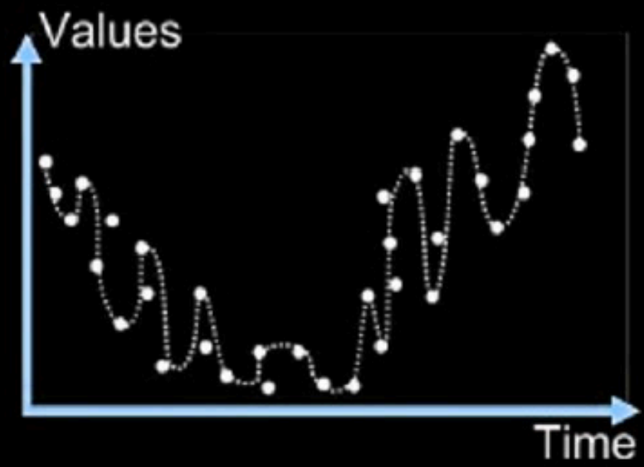
A good model is one that generalizes to new data



UNDERFIT



GOOD FIT

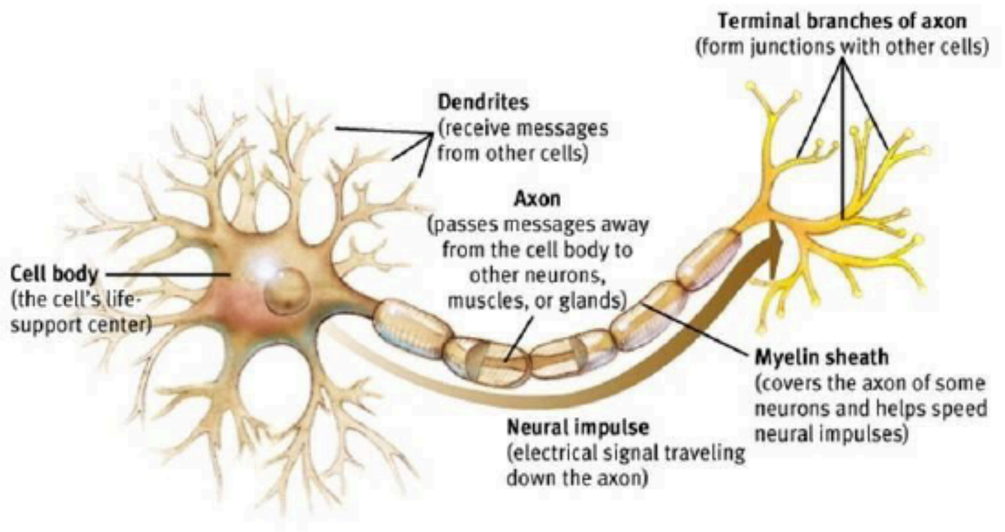


OVER FIT

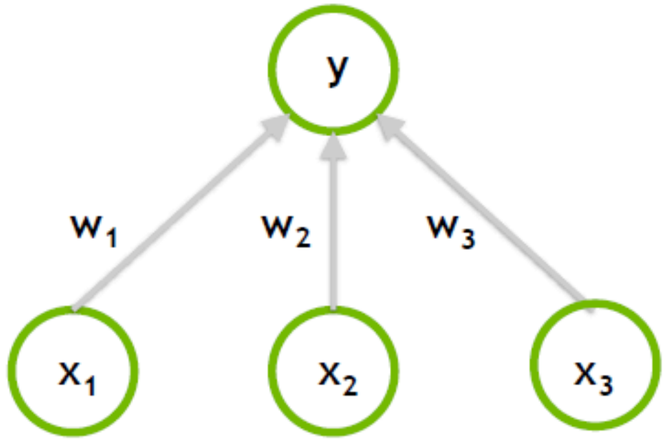
ARTIFICIAL NEURONS

Are simple equations with a set of adjustable parameters

Biological neuron



Artificial neuron

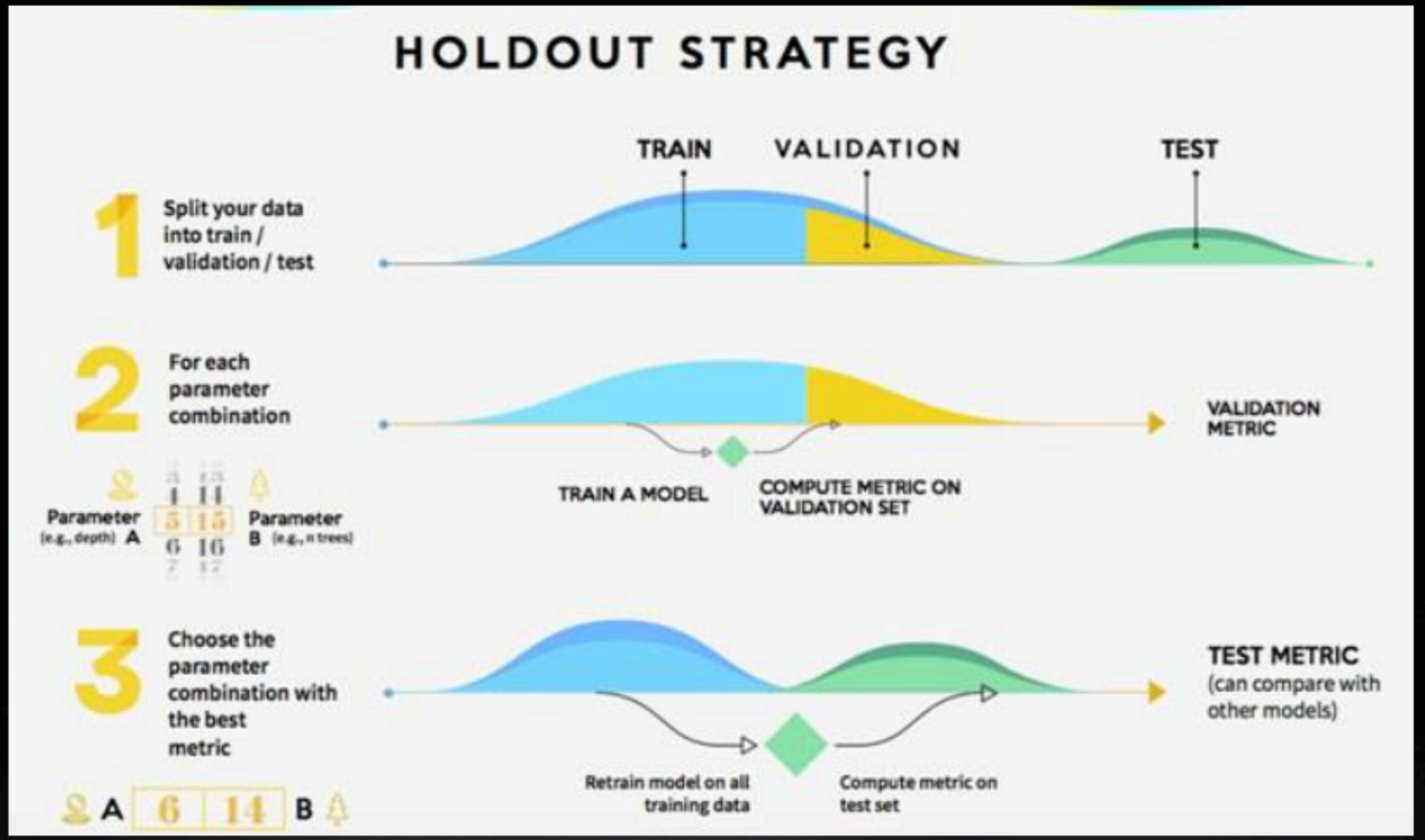


$$y=f(w_1x_1+w_2x_2+w_3x_3)$$

https://towardsdatascience.com/the-differences-between-artificial-and-biological-neural-networks-a8b46db828b7

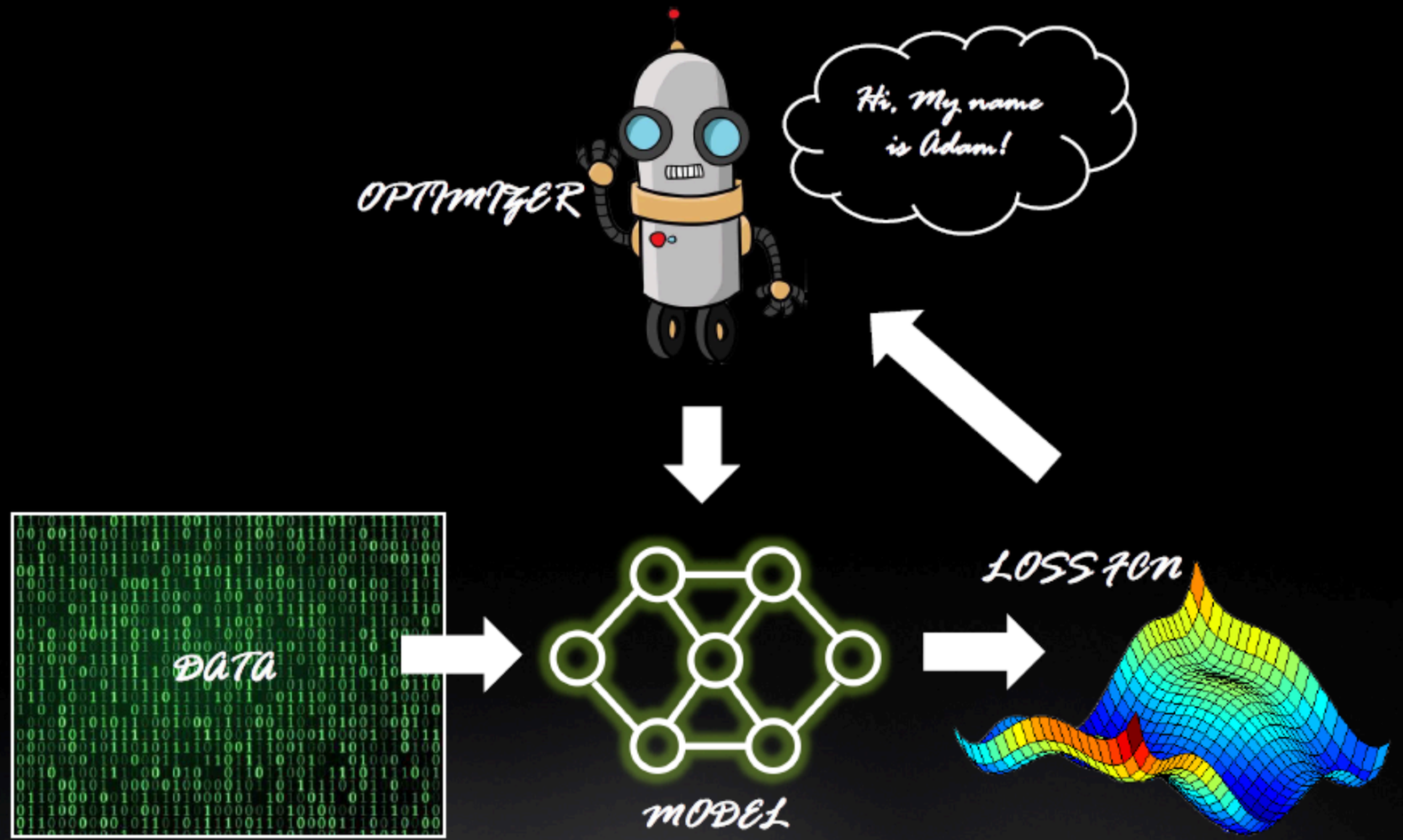
KEEP TRAINING AND TEST DATA SEPERATE

KEEP TEST, TRAINING, AND VALIDATION DATA SEPERATE



TRAINING

DATA, MODEL, LOSS, AND OPTIMIZER

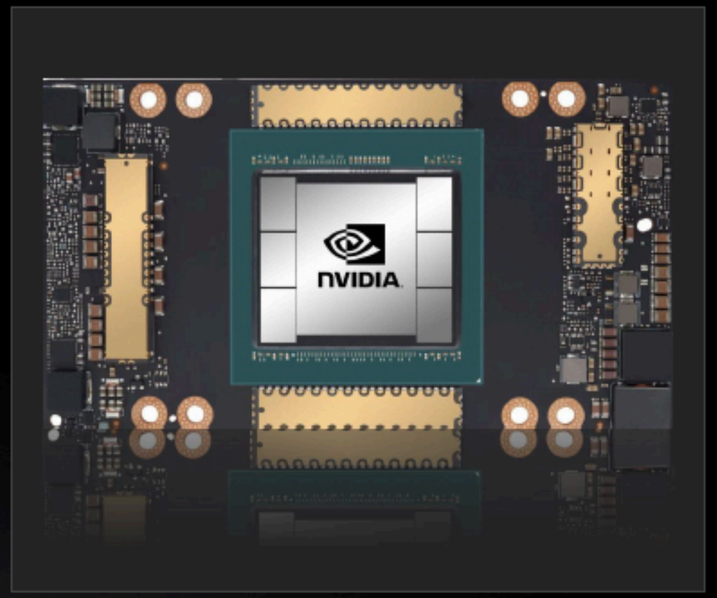


WHAT YOU NEED TO MAKE IT WORK

You need three main ingredients (and some skill)

LARGE QUANTITIES OF DATA

ML FRAMEWORK



GPU ACCELERATOR

Why Now?

Neural Networks date back decades, so why the dominance?

1952 Stochastic Gradient Descent

1958 Perceptron
- Learnable Weights

1986 Backpropagation
- Multi-Layer Perceptron

1995 Deep Convolutional NN
- Digit Recognition

1. Big Data

- Larger Datasets
- Easier Collection & Storage

IMAGENET

WIKIPEDIA
The Free Encyclopedia

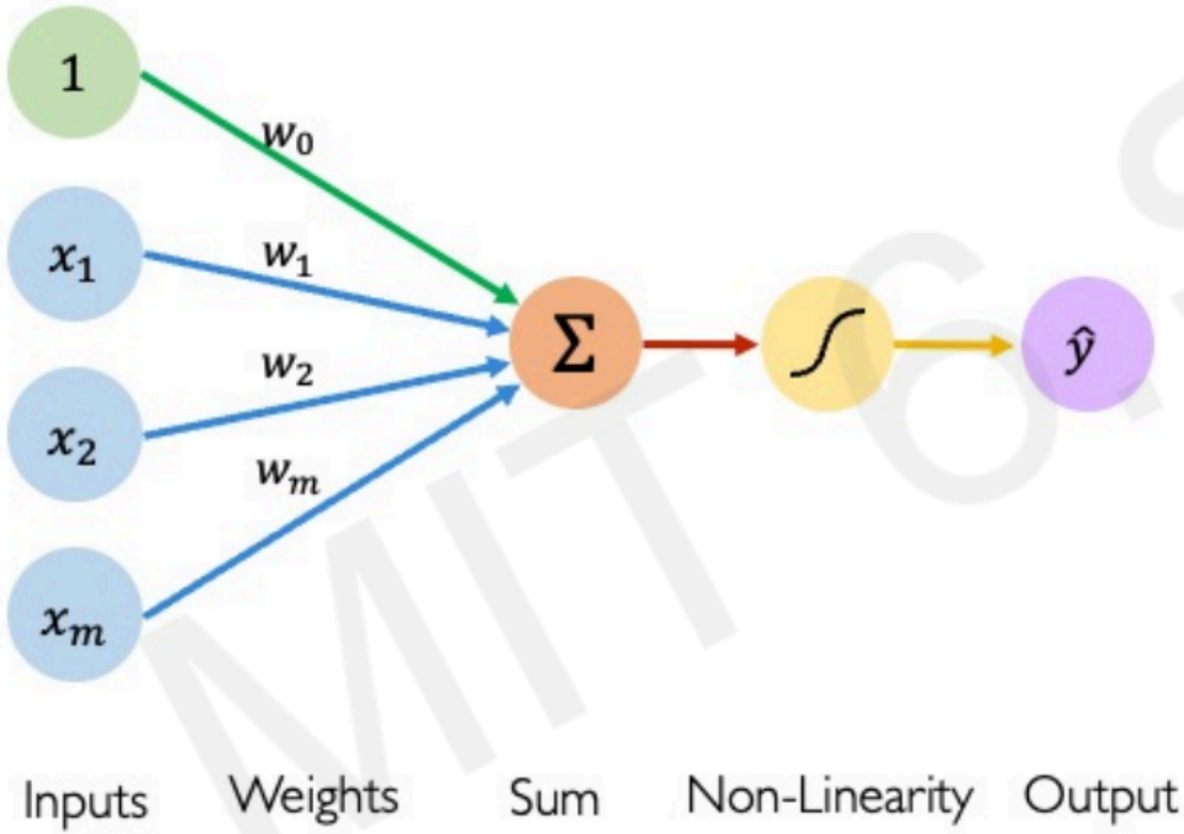
2. Hardware

- Graphics Processing Units (GPUs)
- Massively Parallelizable

3. Software

- Improved Techniques
- New Models
- Toolboxes

The Perceptron: Forward Propagation

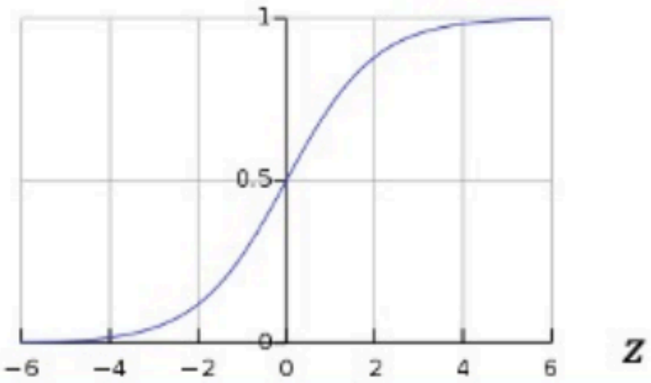


Activation Functions

$$\hat{y} = g\left(w_0 + \boldsymbol{X}^T \boldsymbol{W} \right)$$

- Example: sigmoid function

$$g(z) = \sigma(z) = \frac{1}{1 + e^{-z}}$$

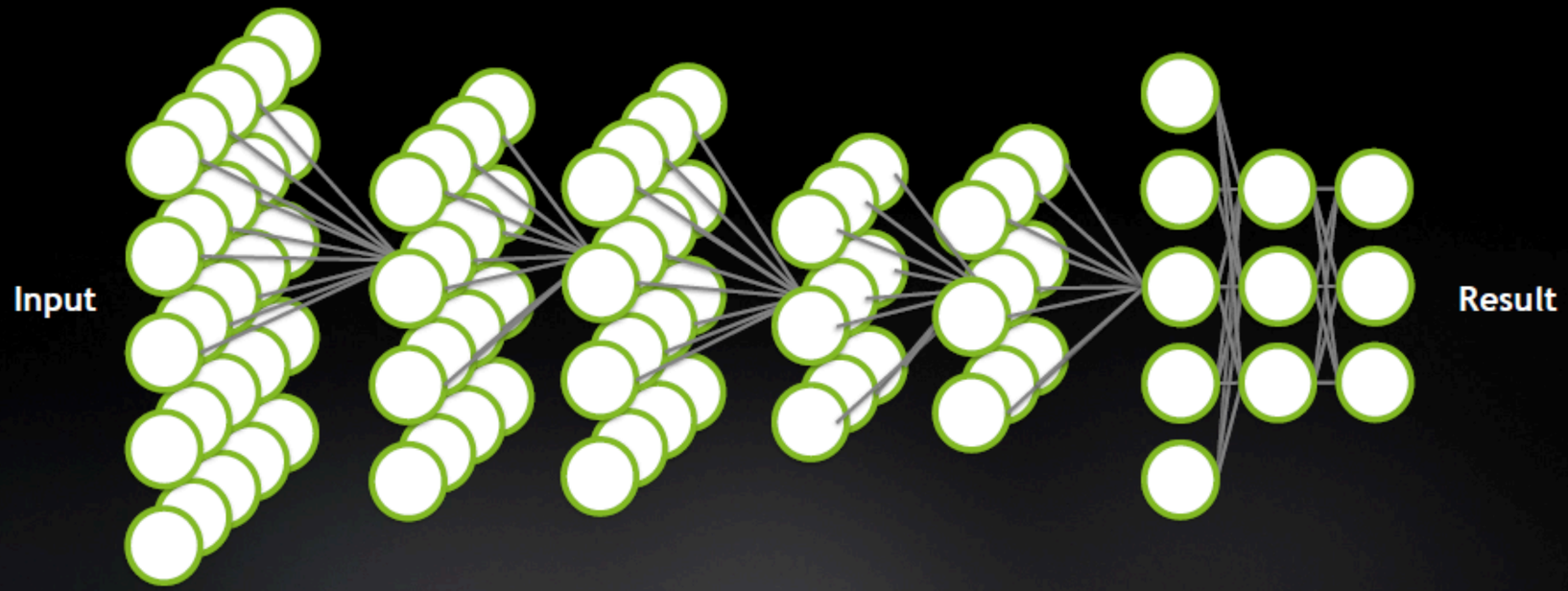


COMPUTER VISION

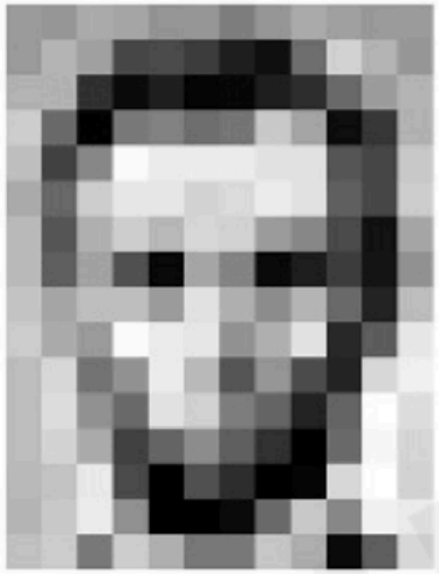
FCNN & CNN

DEEPER NEURAL NETWORKS

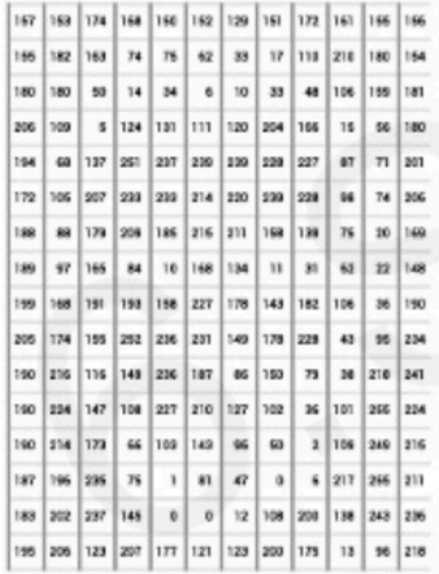
More layers allows for more levels of abstraction



Tasks in Computer Vision



Input Image

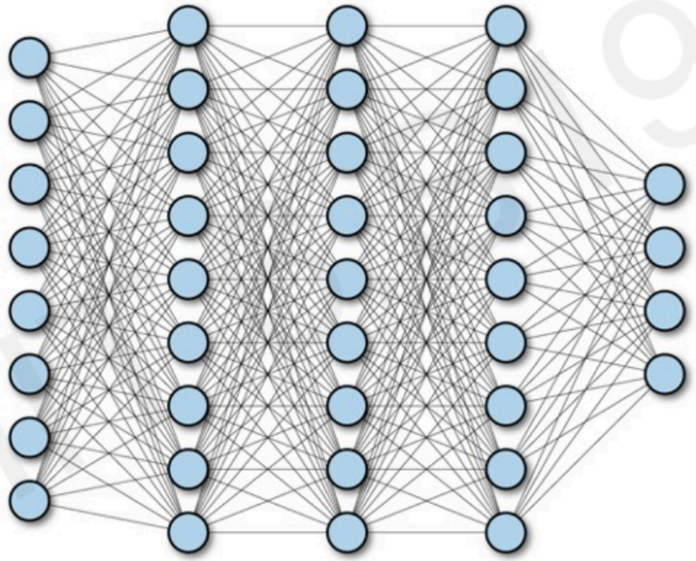


Pixel Representation

classification

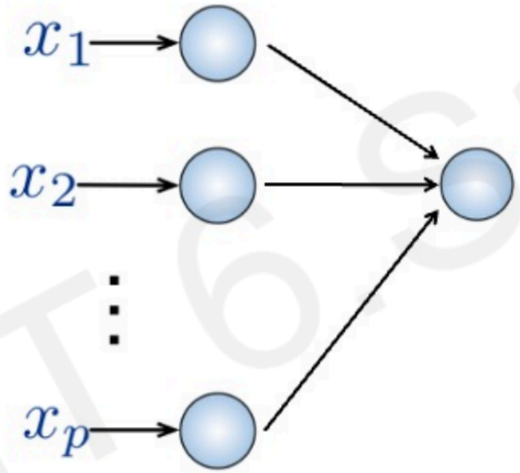
Lincoln	0.8
Washington	0.1
Jefferson	0.05
Obama	0.05

Fully Connected Neural Network



Input:

- 2D image
- Vector of pixel values



Fully Connected:

- Connect neuron in hidden layer to all neurons in input layer
- No spatial information!
- And many, many parameters!

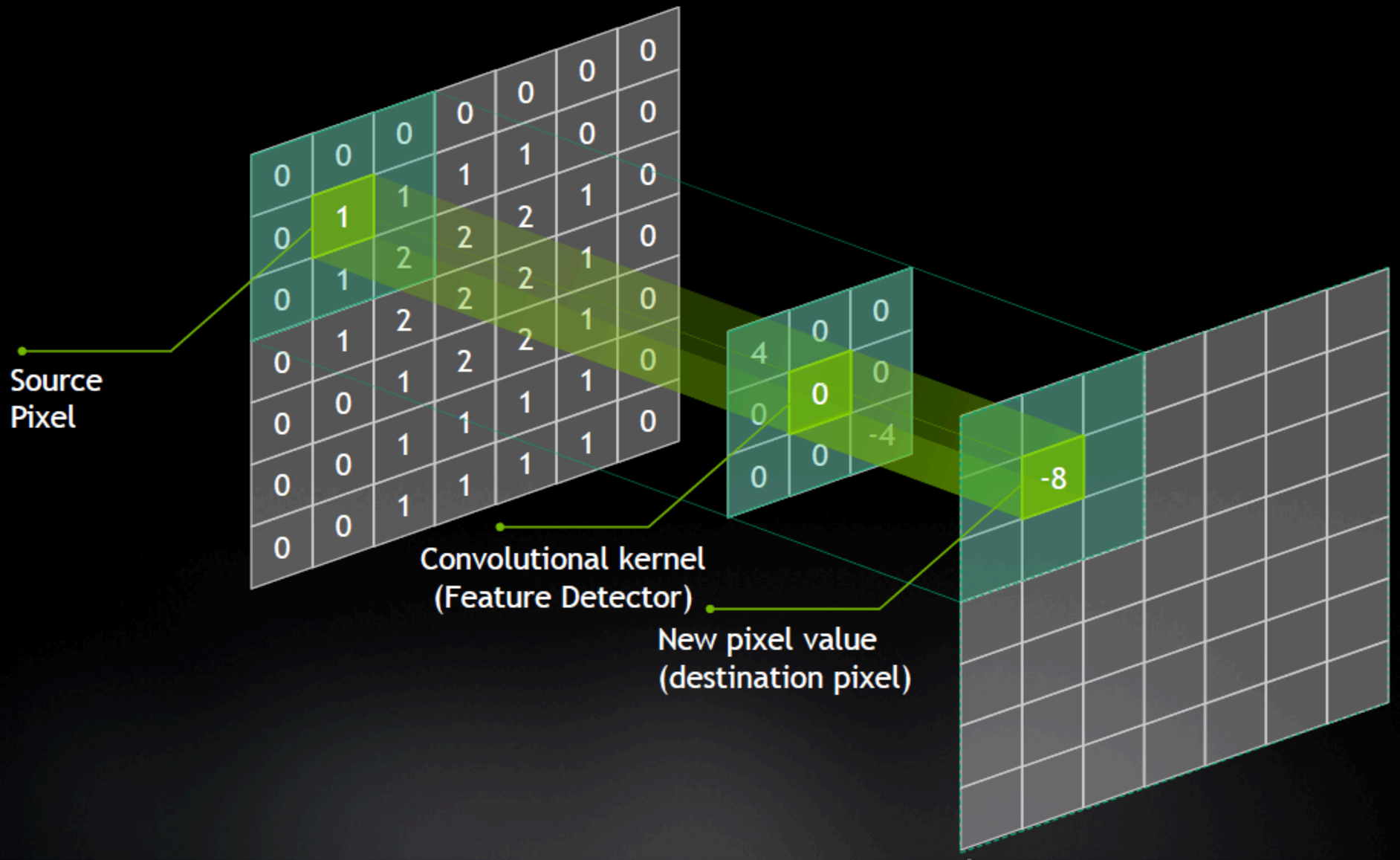
How can we use **spatial structure** in the input to inform the architecture of the network?

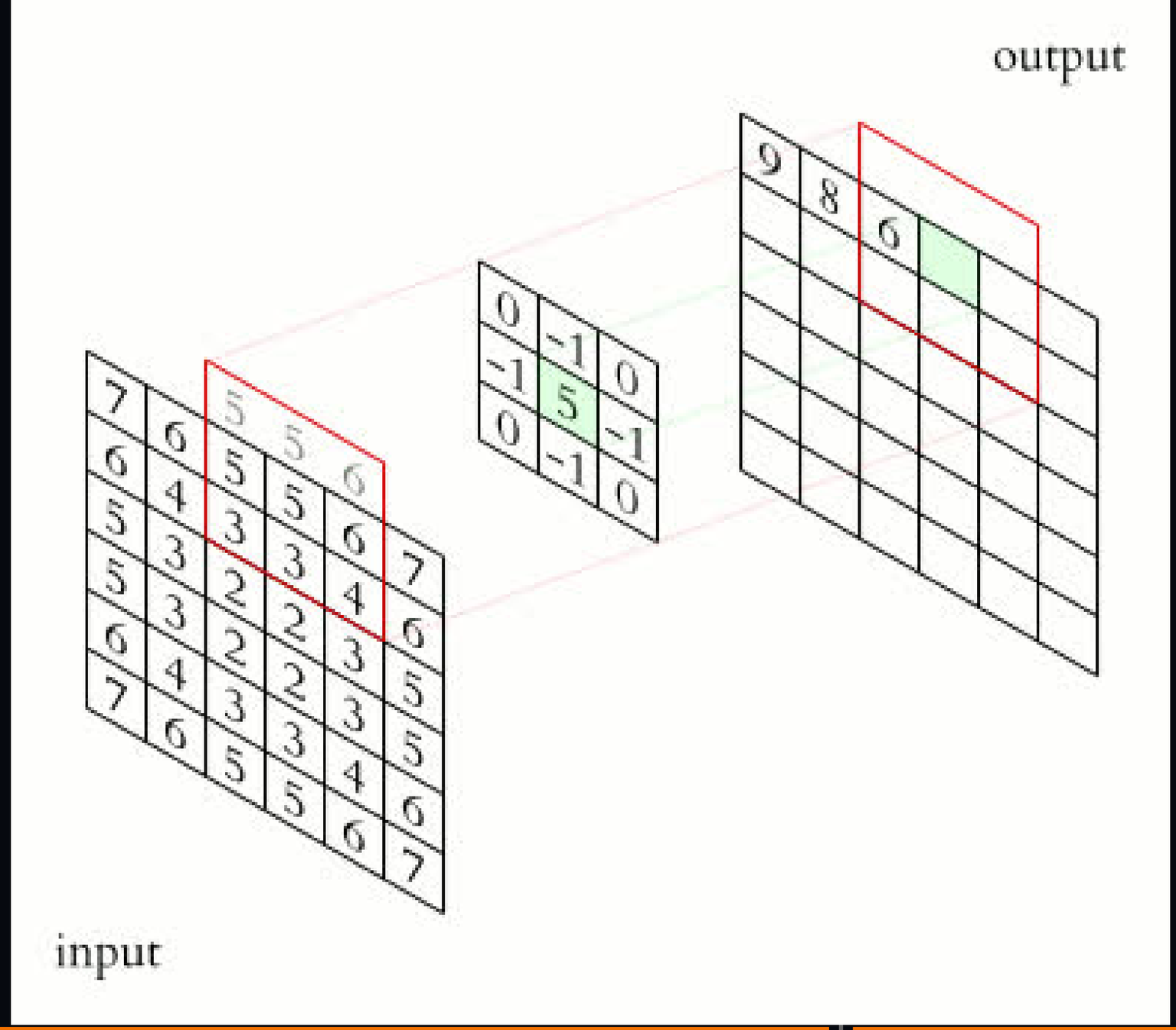
THE

CONVOLUTION

WHAT IS A CONVOLUTION?

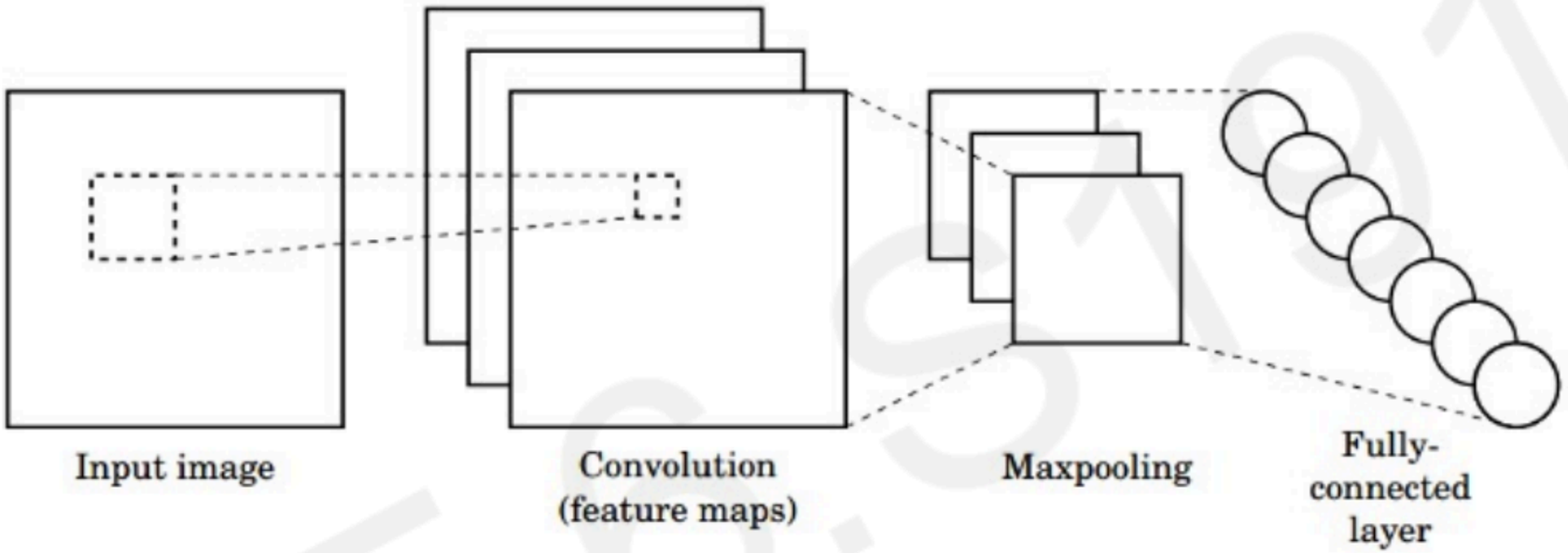
A small matrix transformation, applied at each point of the image





https://en.m.wikipedia.org/wiki/File:2D_Convolution_Animation.gif

CNNs for Classification



`tf.keras.layers.Conv2D` — `torch.nn.Conv2d`

1. **Convolution**: Apply filters to generate feature maps.

`tf.keras.activations.*` — `torch.nn.ReLU,…`

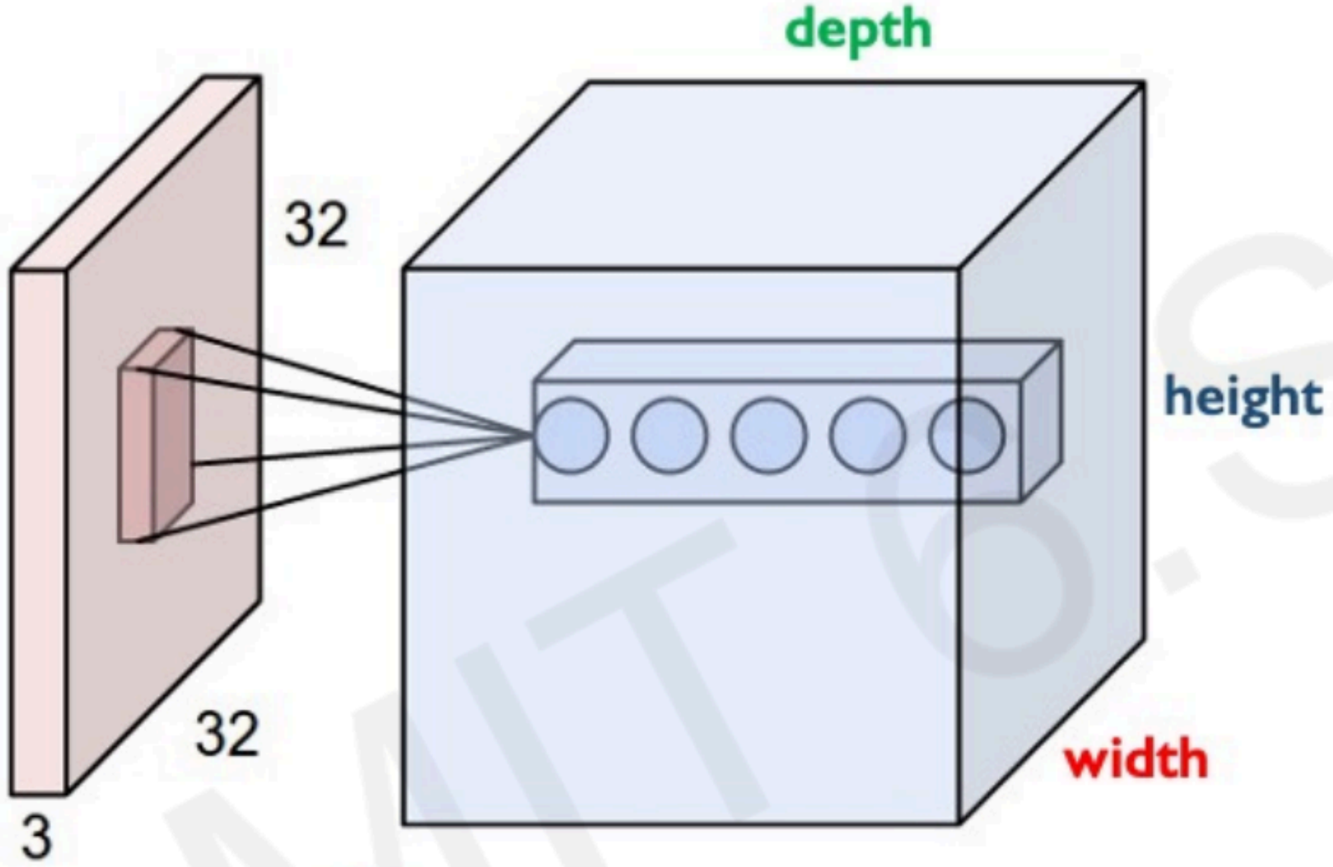
2. **Non-linearity**: Often ReLU.

`tf.keras.layers.MaxPool2D` — `torch.nn.MaxPool2d`

3. **Pooling**: Downsampling operation on each feature map.

Train model with image data.
Learn weights of filters in convolutional layers.

CNNs: Spatial Arrangement of Output Volume



Layer Dimensions:

$h \times w \times d$

where h and w are spatial dimensions
d (depth) = number of filters

Stride:

Filter step size

Receptive Field:

Locations in input image that a node is path connected to

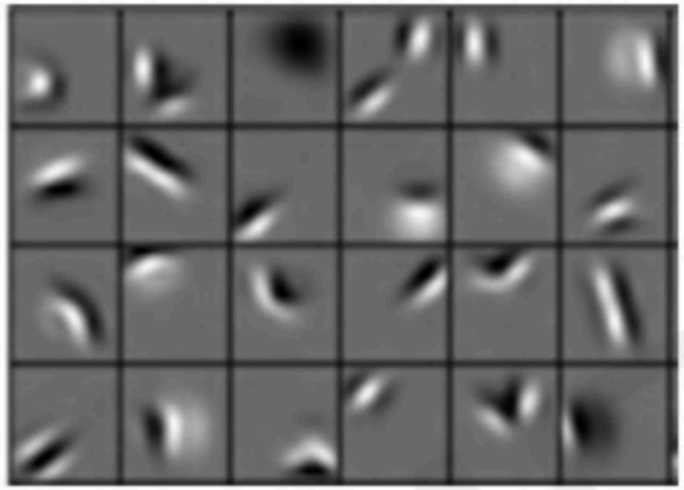
```
tf.keras.layers.Conv2D( filters=d, kernel_size=(h,w), strides=s )
```

```
torch.nn.Conv2d( in_channels=3, out_channels=d, kernel_size=(h,w), stride=s )
```

Need to specify input dimensionality!

Representation Learning in Deep CNNs

Low level features



Edges, dark spots

Conv Layer 1

Mid level features

Eyes, ears, nose

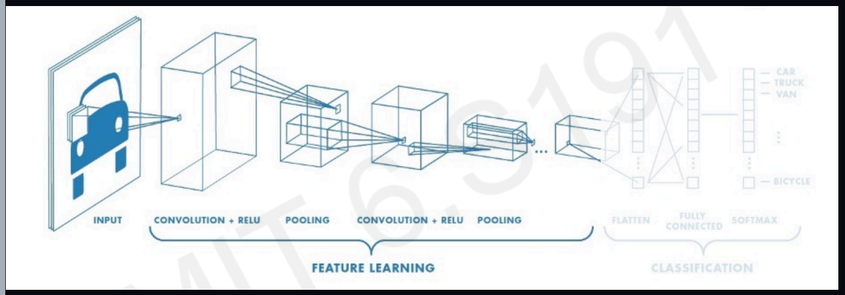
Conv Layer 2

High level features

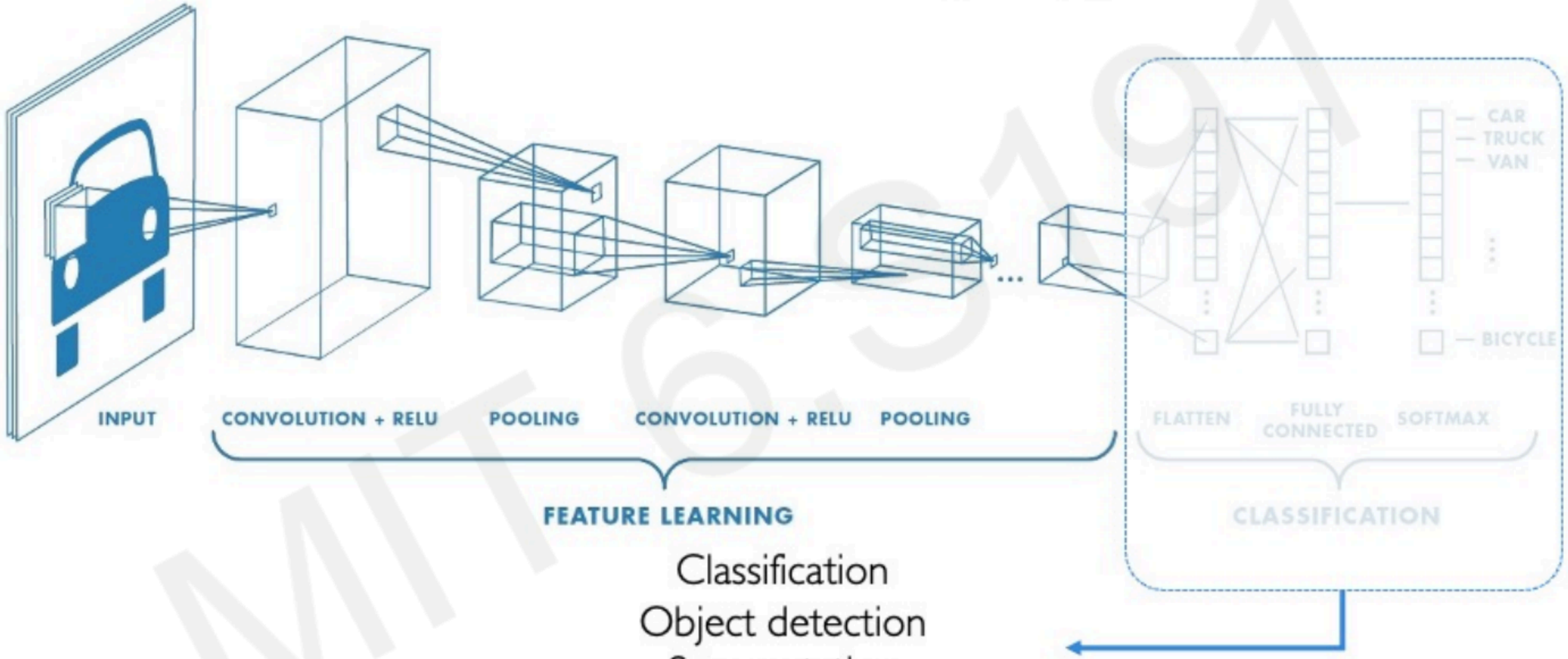
Facial structure

Conv Layer 3

```python
model_cnn = Sequential([
  # 32 filtres pour apprendre 32 motifs différents.
  Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=input_shape),

  # On résume l'information
  MaxPooling2D(pool_size=(2, 2)),

# On utilise 64 filtres pour capturer plus de complexité.
  Conv2D(64, kernel_size=(3, 3), activation='relu'),
  MaxPooling2D(pool_size=(2, 2)),

  # On transforme notre carte 2D de features en un vecteur 1D.
  Flatten(),

  # Il "éteint" aléatoirement 50% des neurones pendant l'entraînement.
  Dropout(0.5),

  # 'softmax' transforme les sorties en une distribution de probabilité.
  Dense(10, activation='softmax')
])

# Affichons le plan de notre architecture
model_cnn.summary()
```

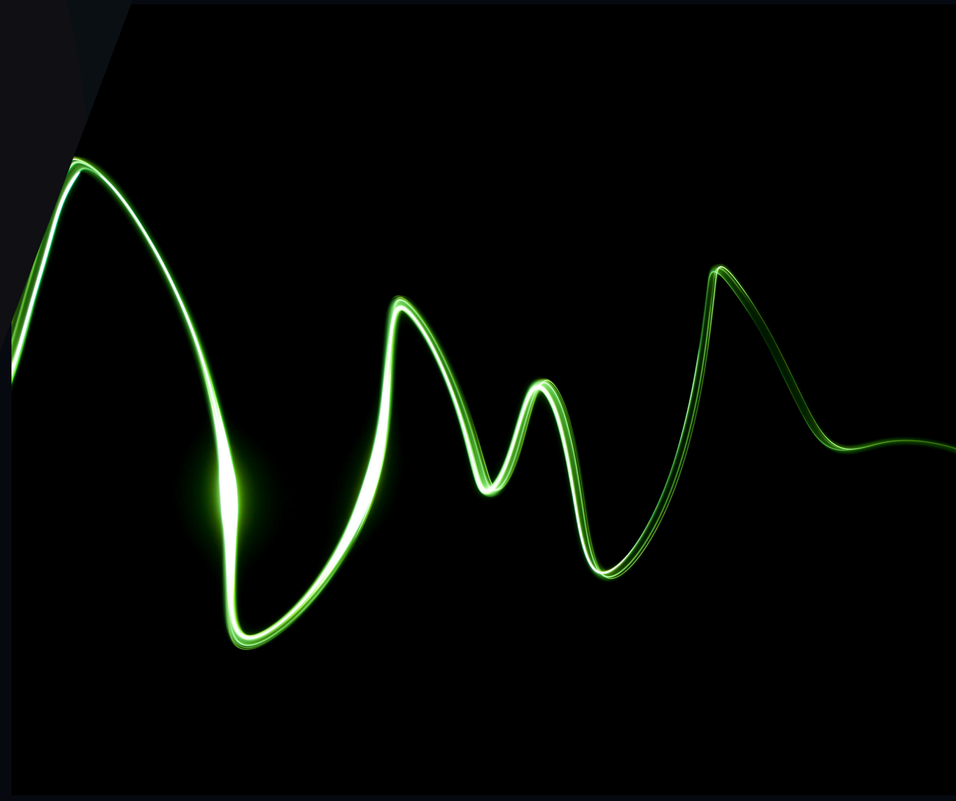


An Architecture for Many Applications



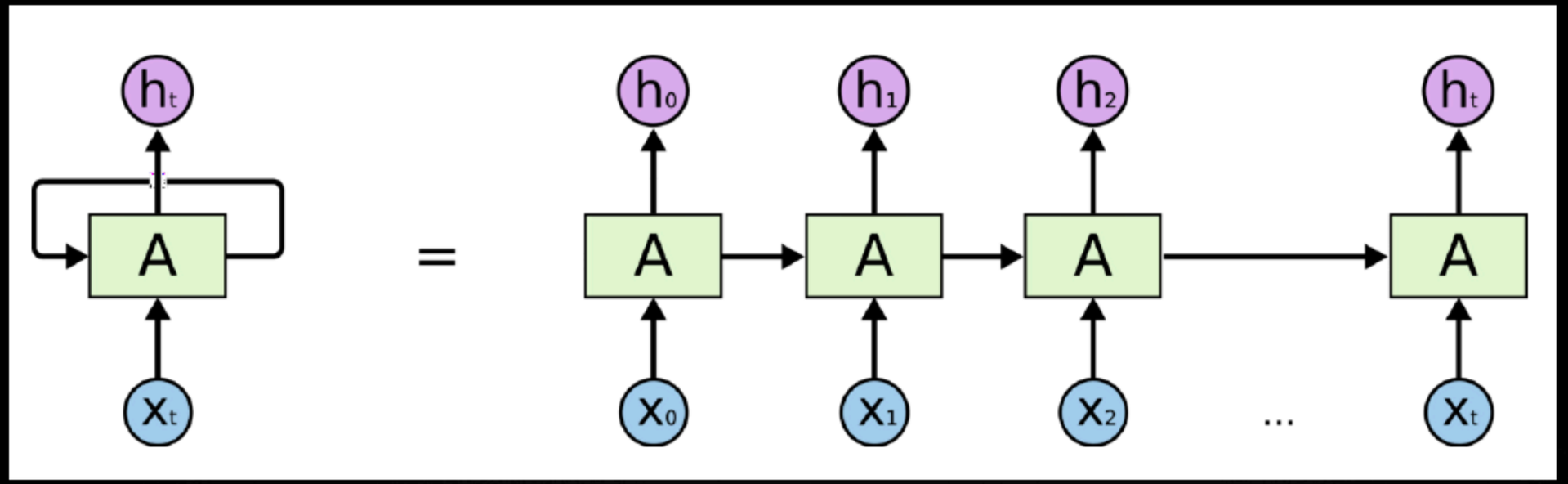
Classification
Object detection
Segmentation
Probabilistic control

DEEP SEQUENCE MODELING
RNN, LSTM & GRU



RECURRENT NEURAL NETS

(1986) Neural networks for sequences

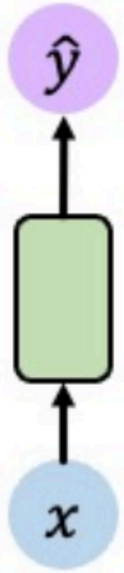


RNN, Unrolled over time

https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

Sequence Modeling Applications

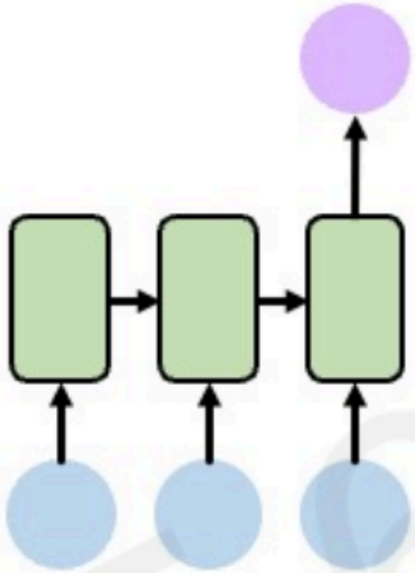


One to One

Binary Classification

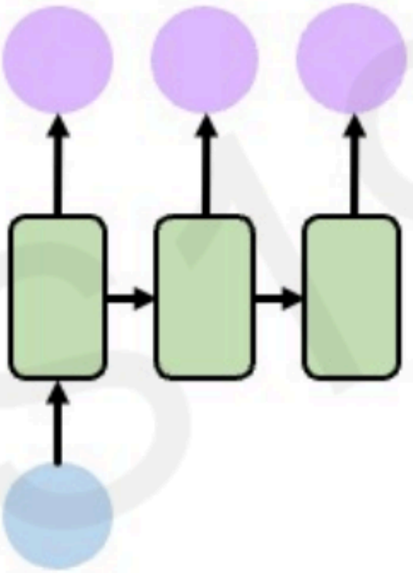
"Will I pass this class?"

Student → Pass?



Many to One

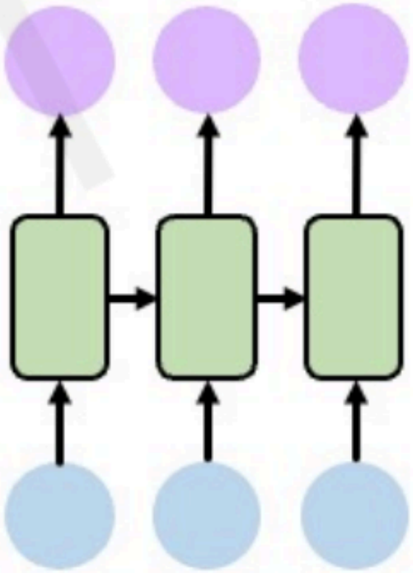
Sentiment Classification



One to Many

Image Captioning

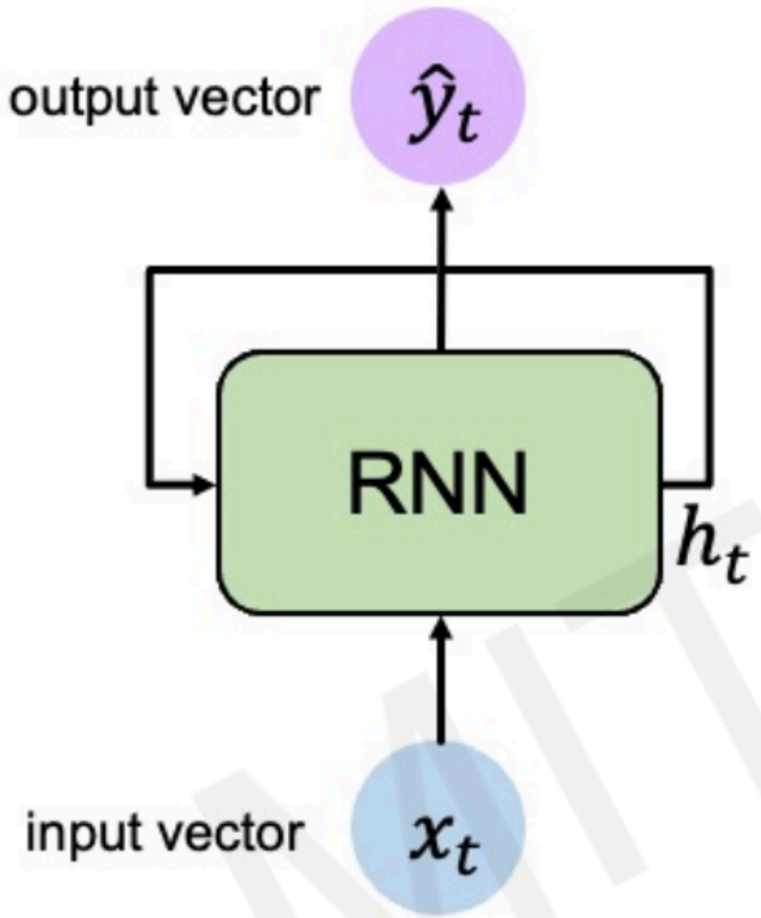
"A baseball player throws a ball"



Many to Many

Machine Translation

Recurrent Neural Networks (RNNs)



Apply a **recurrence relation** at every time step to process a sequence:

$$h_t = f_W(x_t, h_{t-1})$$

cell state

function with weights W

input

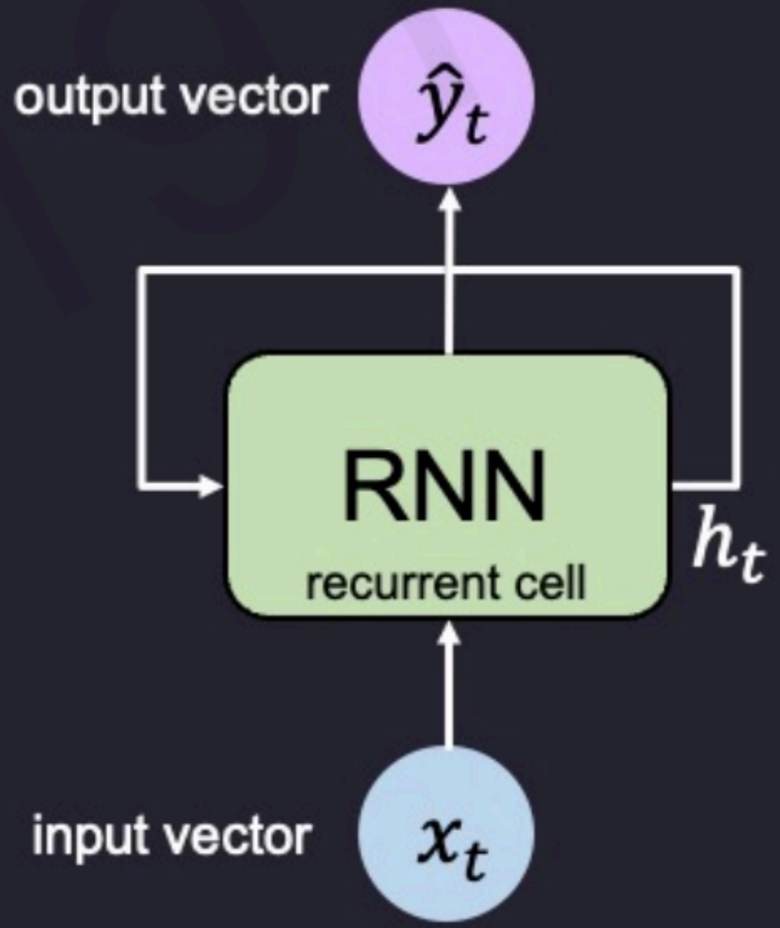
old state

Note: the same function and set of parameters are used at every time step

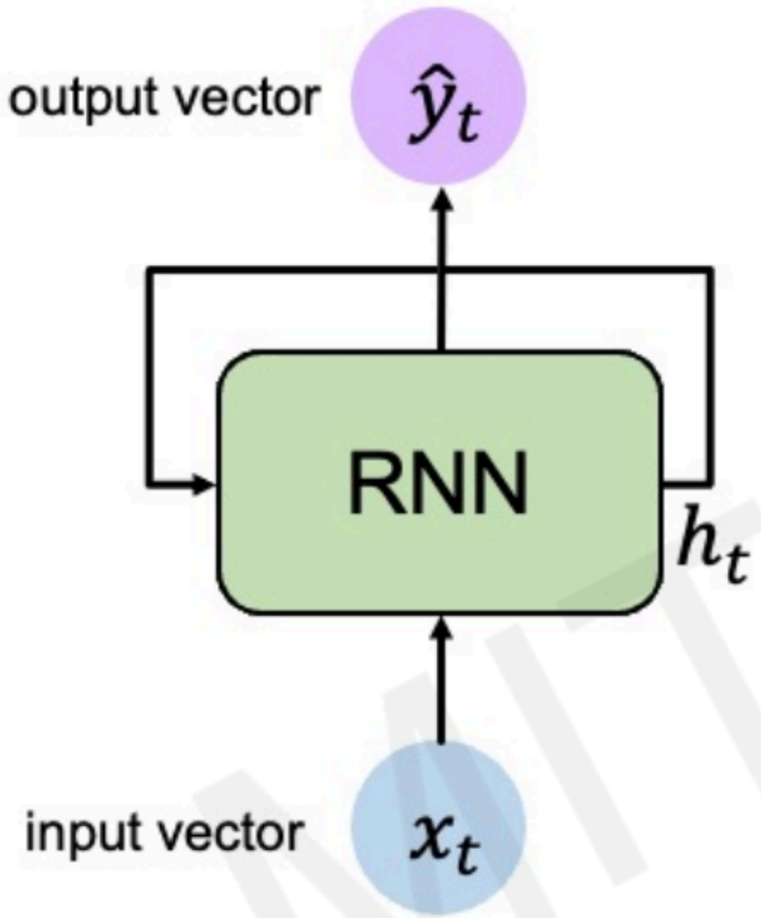
RNNs have a **state**, h_t, that is updated **at each time step** as a sequence is processed

RNN Intuition

```
my_rnn = RNN()
hidden_state = [0, 0, 0, 0]

sentence = ["I", "love", "recurrent", "neural"]

for word in sentence:
    prediction, hidden_state = my_rnn(word, hidden_state)

next_word_prediction = prediction
# >>> "networks!"
```



RNN State Update and Output



Output Vector

$$\hat{y}_t = \boldsymbol{W}_{hy}^{T} h_t$$

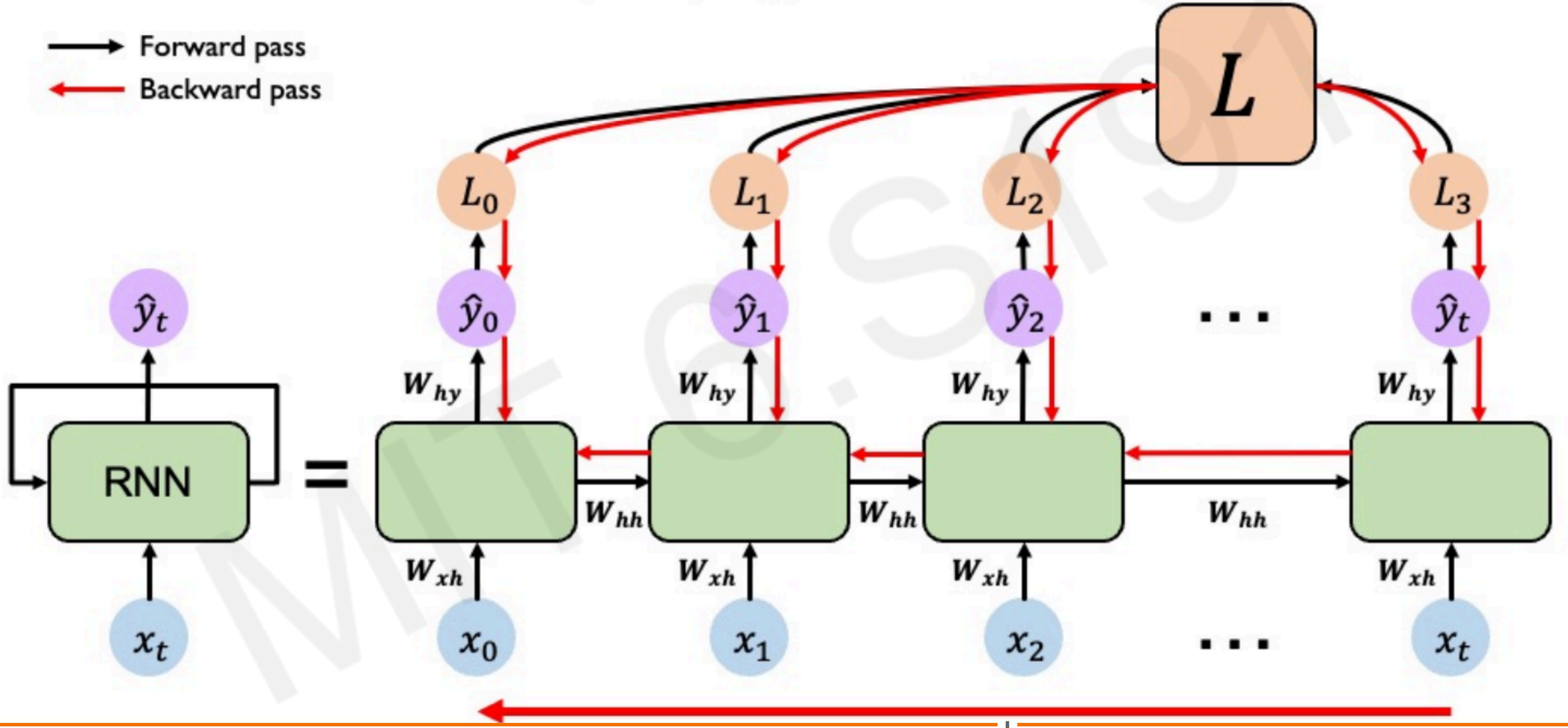
Update Hidden State

$$h_t = \tanh(\boldsymbol{W}_{hh}^{T} h_{t-1} + \boldsymbol{W}_{xh}^{T} x_t)$$

Input Vector

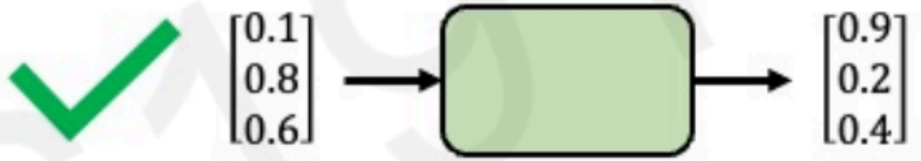
$$x_t$$

RNNs: Backpropagation Through Time



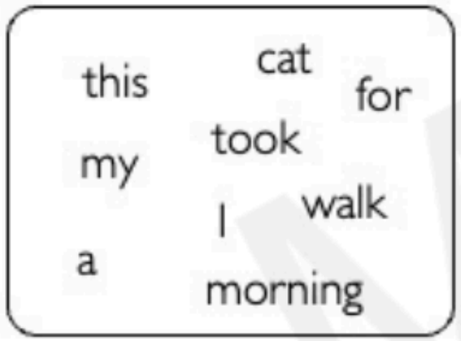
Encoding Language for a Neural Network

Neural networks cannot interpret words

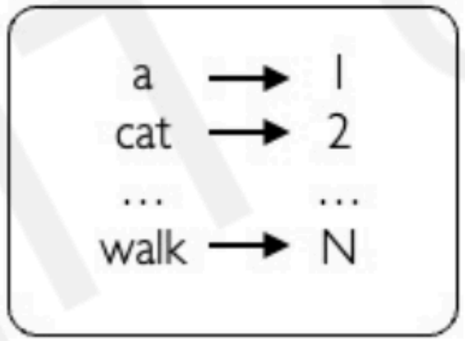


Neural networks require numerical inputs

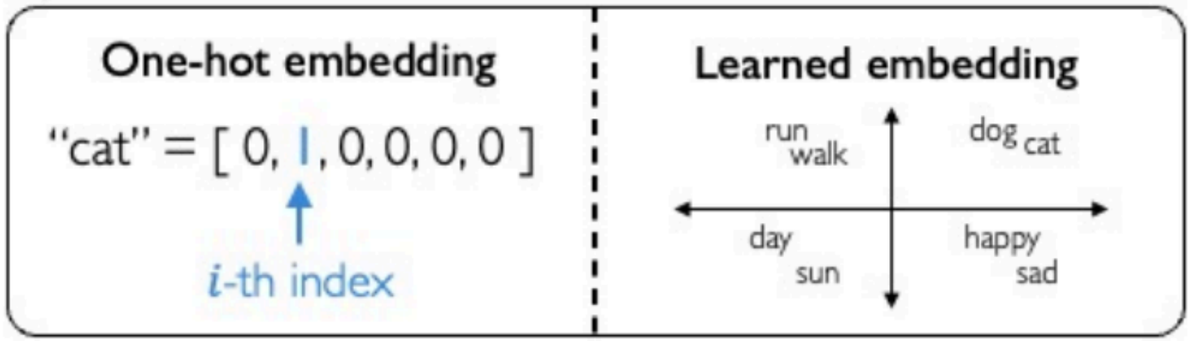
Embedding: transform indexes into a vector of fixed size.



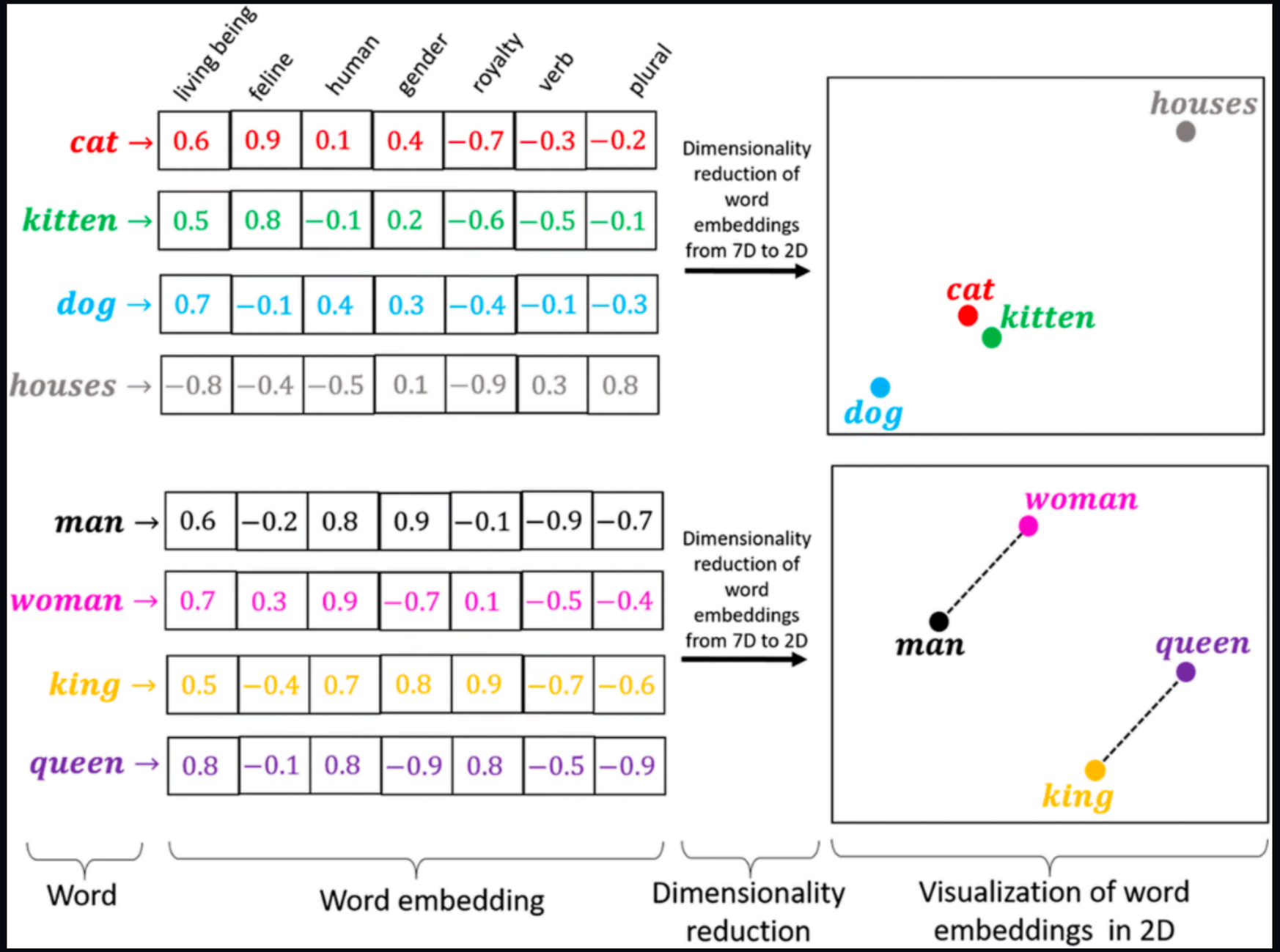
1. Vocabulary:
Corpus of words



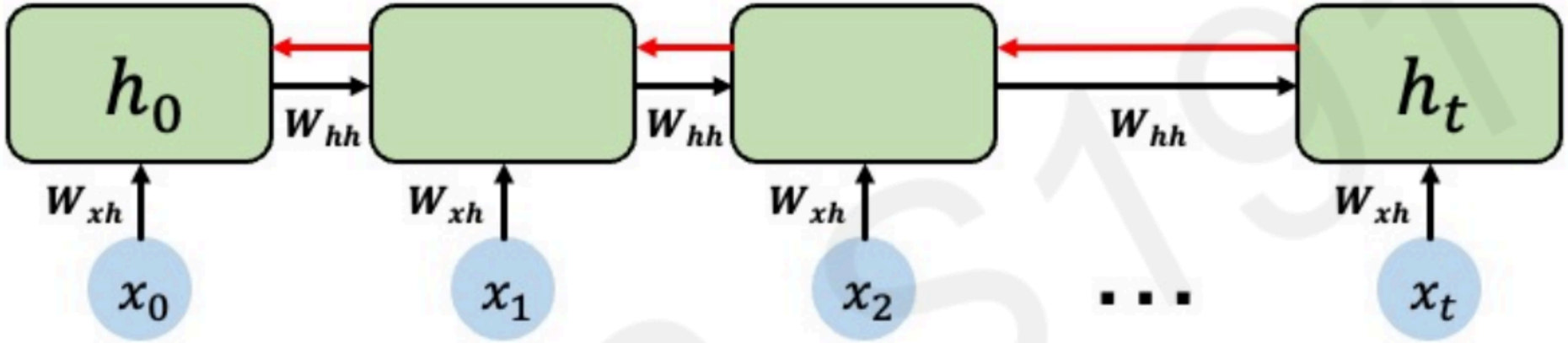
2. Indexing:
Word to index



3. Embedding:
Index to fixed-sized vector


living being
feline
human
gender
royalty
verb
plural
cat → 0.6 0.9 0.1 0.4 −0.7 −0.3 −0.2
kitten → 0.5 0.8 −0.1 0.2 −0.6 −0.5 −0.1
dog → 0.7 −0.1 0.4 0.3 −0.4 −0.1 −0.3
houses → −0.8 −0.4 −0.5 0.1 −0.9 0.3 0.8
Dimensionality reduction of word embeddings from 7D to 2D
houses
cat
kitten
dog
man → 0.6 −0.2 0.8 0.9 −0.1 −0.9 −0.7
woman → 0.7 0.3 0.9 −0.7 0.1 −0.5 −0.4
king → 0.5 −0.4 0.7 0.8 0.9 −0.7 −0.6
queen → 0.8 −0.1 0.8 −0.9 0.8 −0.5 −0.9
Dimensionality reduction of word embeddings from 7D to 2D
woman
man
queen
king
Word
Word embedding
Dimensionality reduction
Visualization of word embeddings in 2D

Standard RNN Gradient Flow: Vanishing Gradients



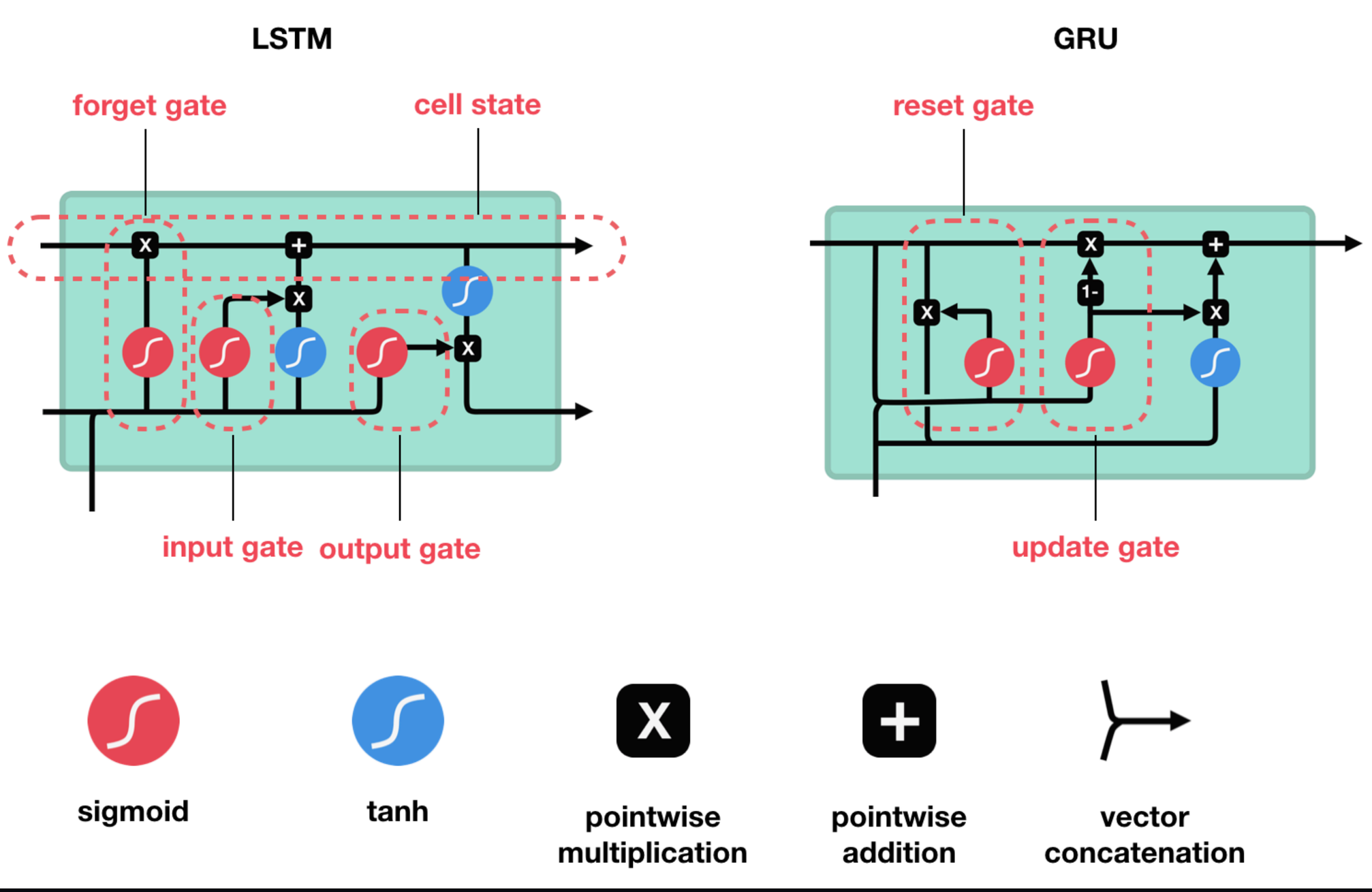
Computing the gradient wrt h_0 involves **many factors of W_{hh} + repeated gradient computation!**

Many values > 1:
exploding gradients

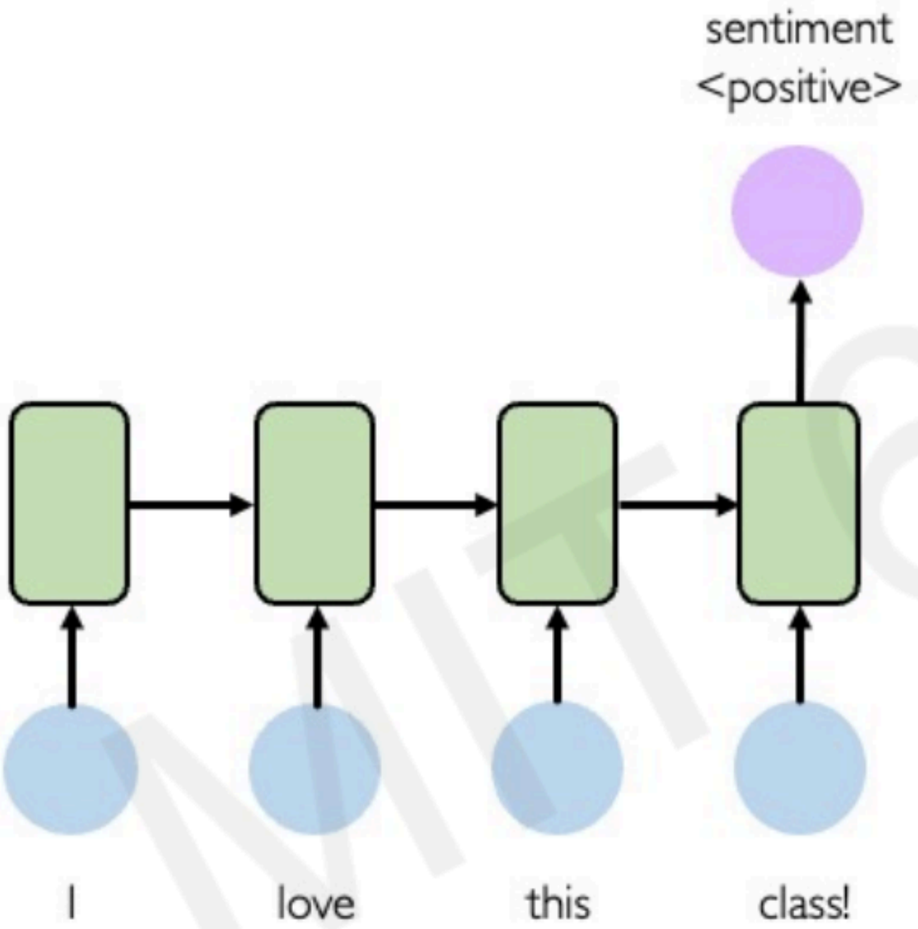
Gradient clipping to scale big gradients

Many values < 1:
vanishing gradients

1. Activation function
2. Weight initialization
3. Network architecture


LSTM
forget gate
cell state
input gate
output gate
GRU
reset gate
update gate
1-
sigmoid
tanh
pointwise multiplication
pointwise addition
vector concatenation

Example Task: Sentiment Classification



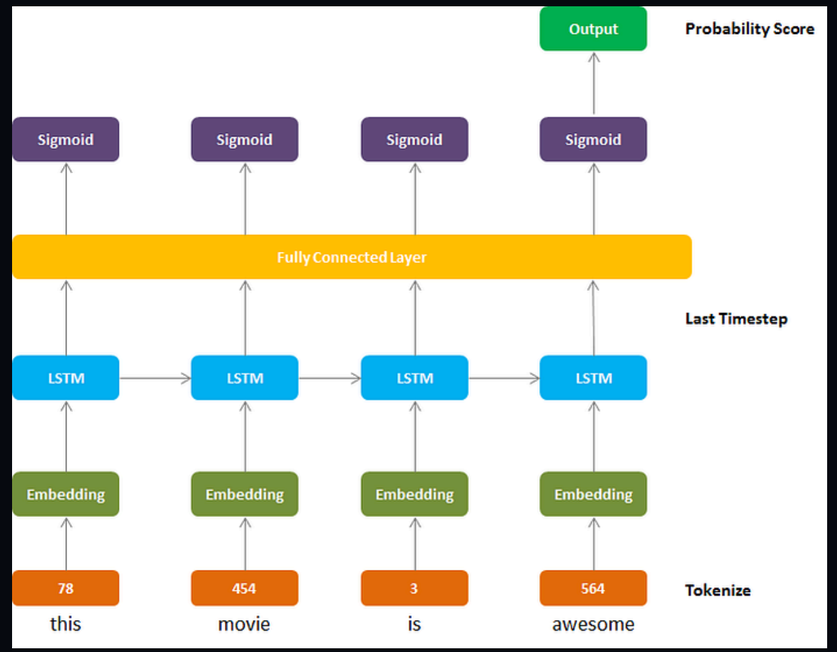
Input: sequence of words

Output: probability of having positive sentiment

```
loss = tf.nn.softmax_cross_entropy_with_logits(y, predicted)
```

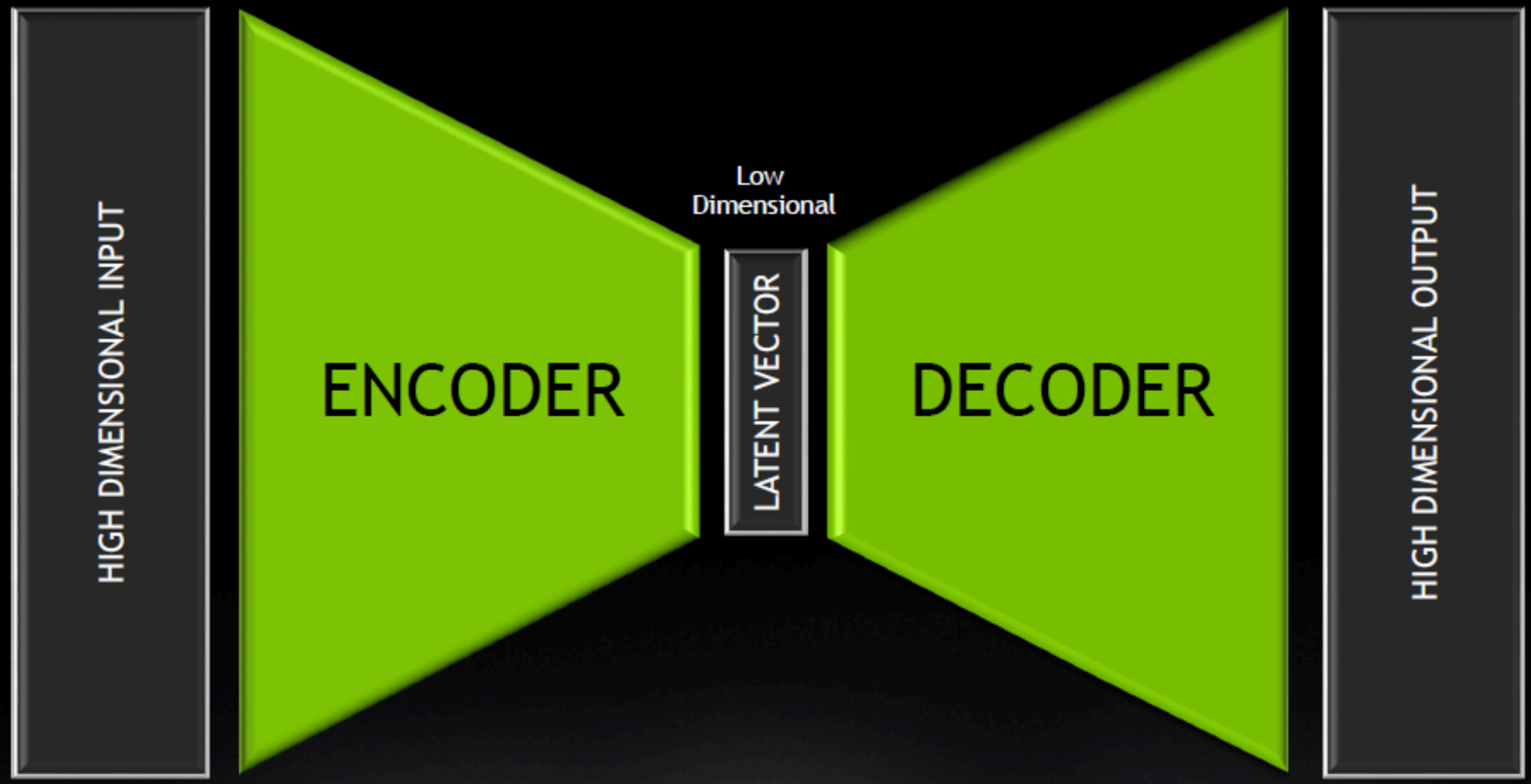
Their customer service

```python
EMBEDDING_DIM = 128
LSTM_UNITS = 64

model_lstm = Sequential([
  # Étape 1: Créer un "dictionnaire de sens" pour chaque mot.
  # input_dim est la taille de notre vocabulaire.
  # output_dim est la taille du vecteur de sens que nous voulons pour chaque mot.
  Embedding(input_dim=VOCAB_SIZE, output_dim=EMBEDDING_DIM, input_length=MAX_LEN),

  # Étape 2: La couche LSTM qui lit la séquence et se souvient du contexte.
  # On l'enrobe dans Bidirectional pour lire la phrase dans les deux sens.
  Bidirectional(LSTM(units=LSTM_UNITS, dropout=0.2, recurrent_dropout=0.2)),

  # Étape 3: Une couche dense pour un traitement final.
  Dense(64, activation='relu'),
  Dropout(0.5), # Un autre Dropout pour la régularisation

  # Étape 4: Le neurone de sortie qui donne la probabilité d'un sentiment positif.
  Dense(1, activation='sigmoid')
])

# Affichons le plan de notre architecture
model_lstm.summary()
```



ENCODERS AND DECODERS

Networks connecting high and low dimensional spaces



THANK YOU

ANY QUESTION?