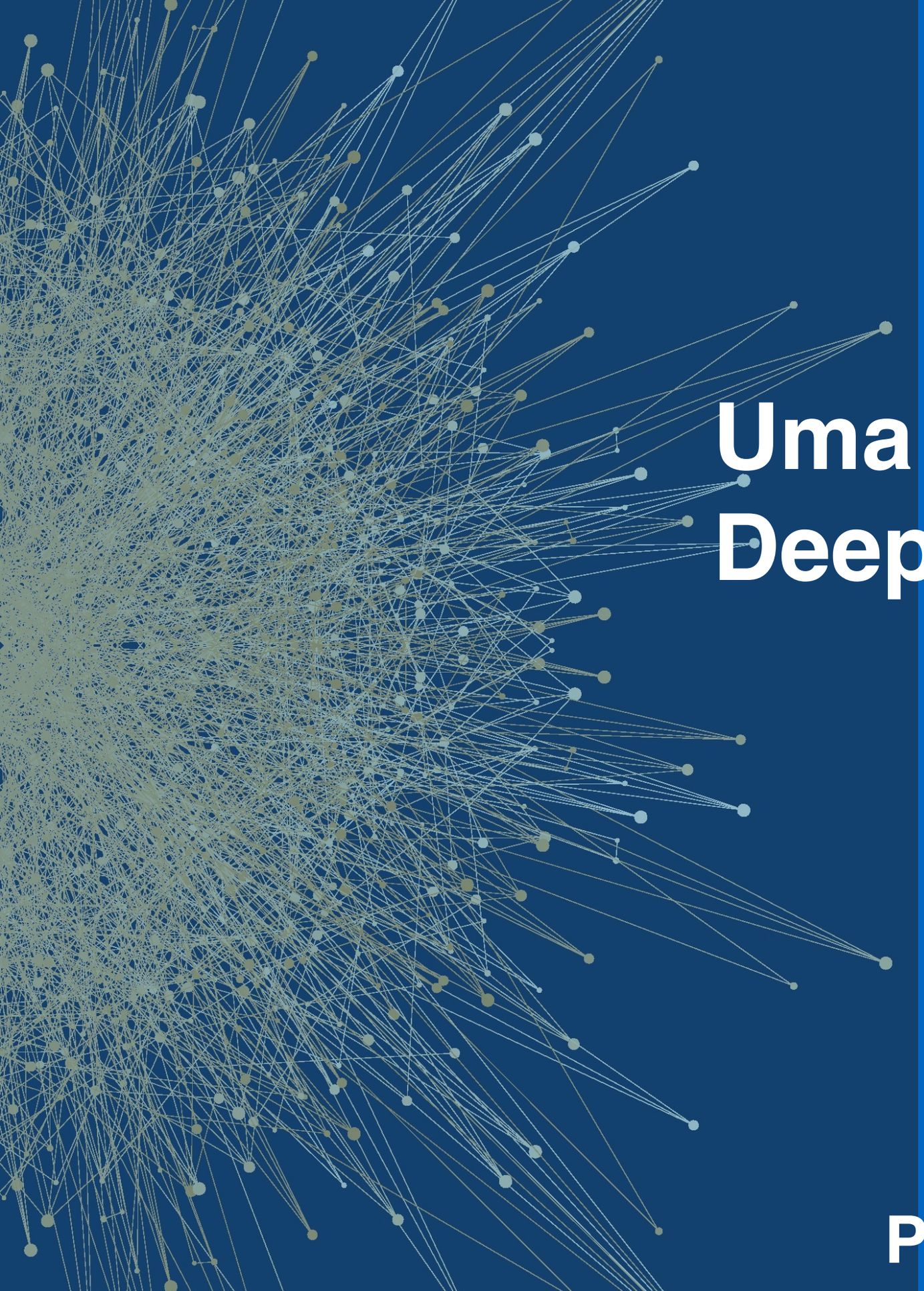
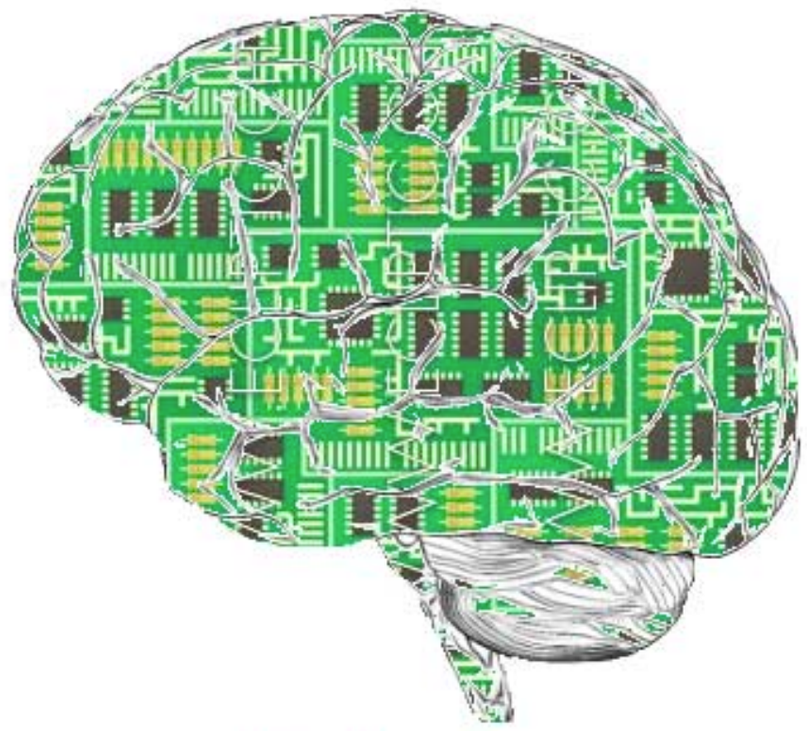


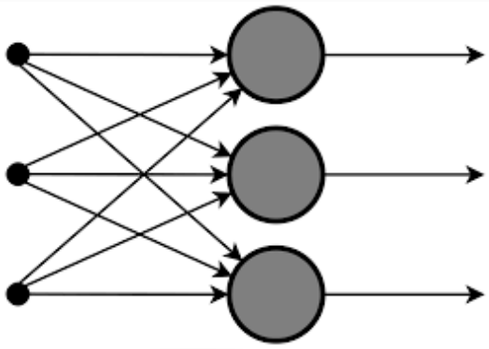
Uma introdução a Deep Learning

Dsc. Nauber Gois
Fametro
Serviço Federal de Processamento de Dados

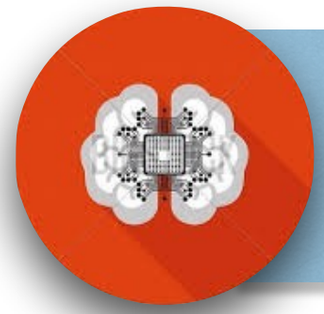


Cognitive Ai

Agenda

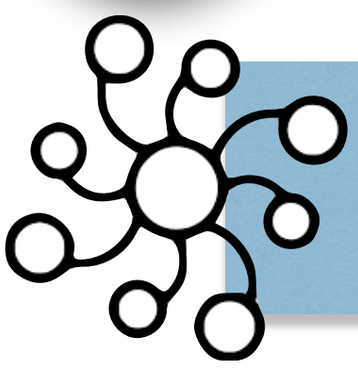


Introdução a Deep Learning

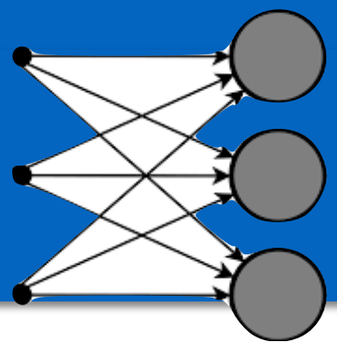


Redes Neurais Convolucionais

Recurrent Neural Networks

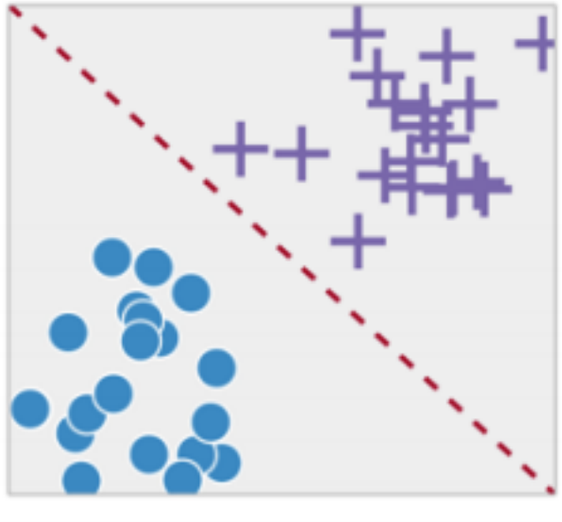


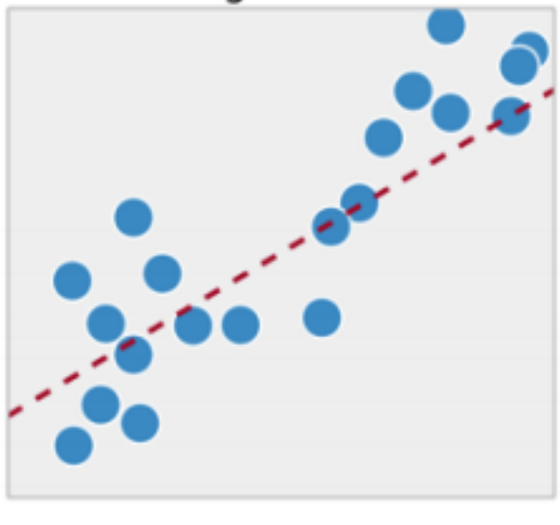
Generative Adversarial Networks

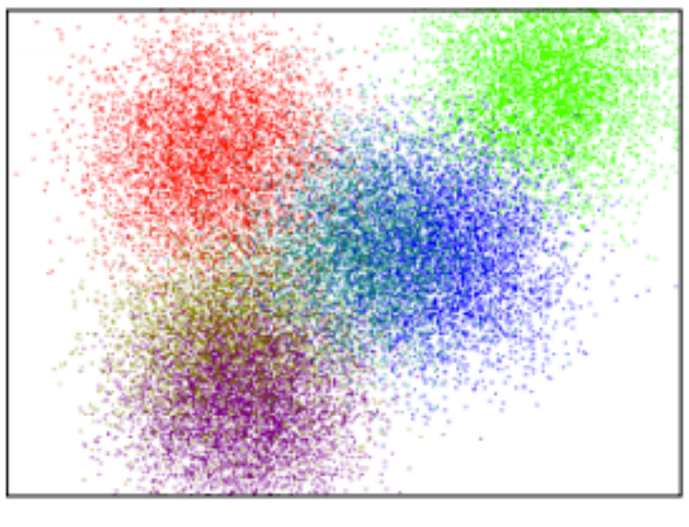


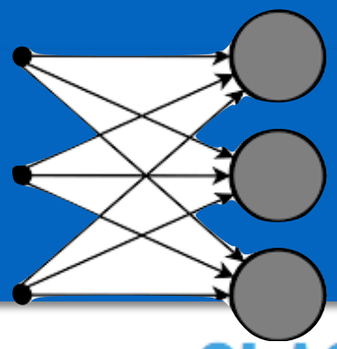
Classificação e Regressão

Identificação de uma área de interesse e o tipo de modelo

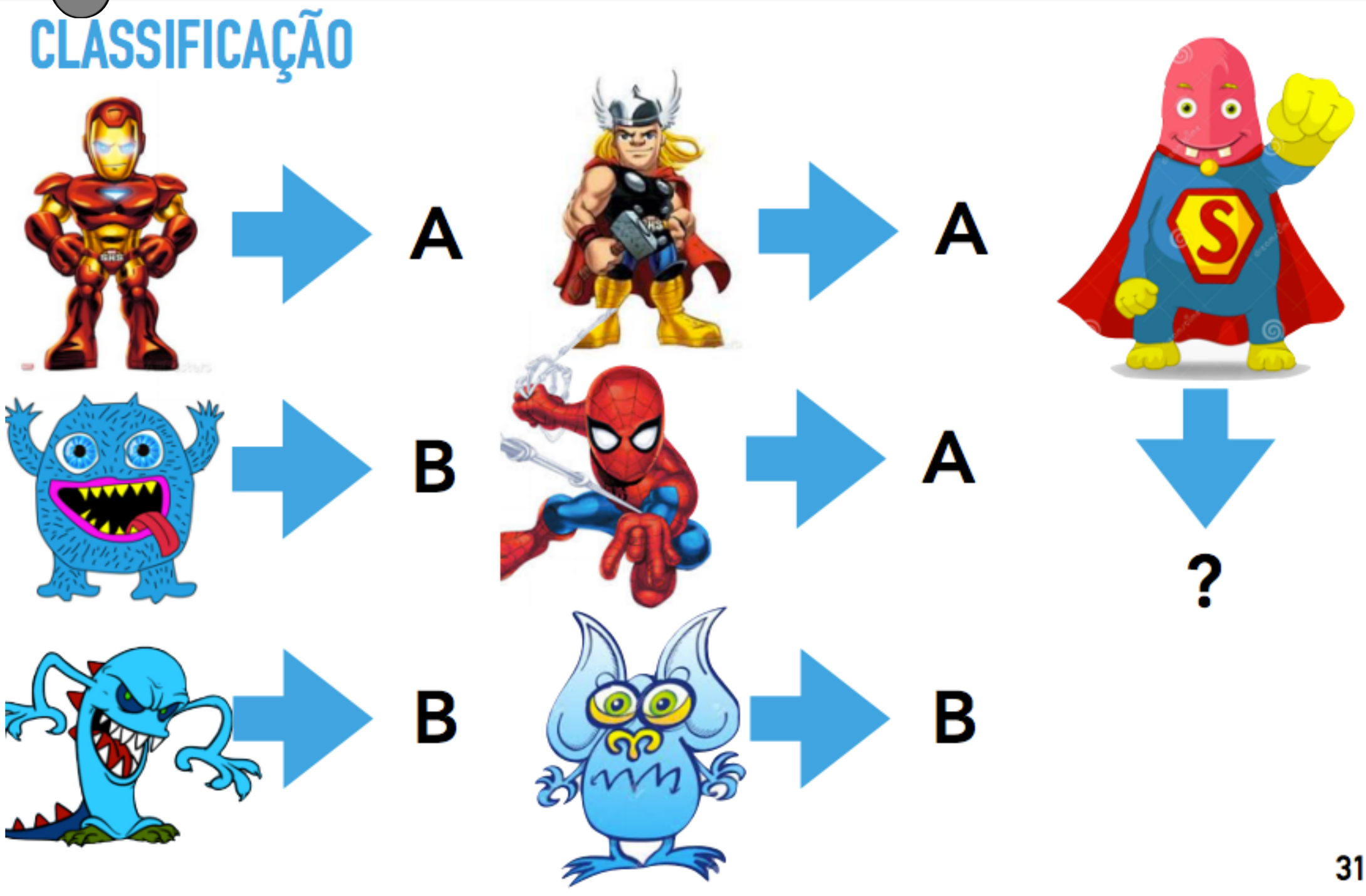


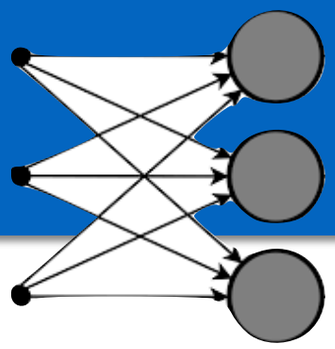






Classificação





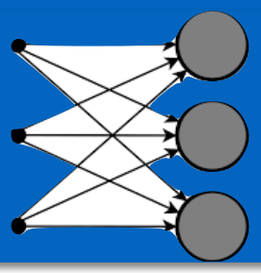
Treinamento

DADOS DE TREINO VS DADOS DE TESTE

- ***Dados de Treino***
  - Usados para treinar um modelo
  - Exemplos

- ***Dados de Teste***
  - Usados para testar a performance do modelo
  - Dados de validação.

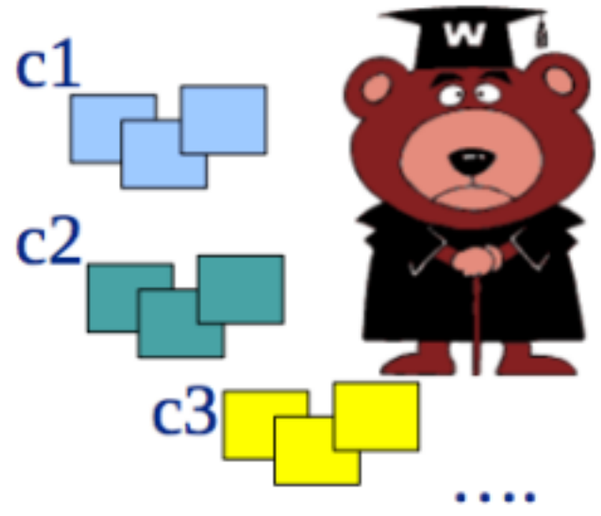
e.g. facial gender classification



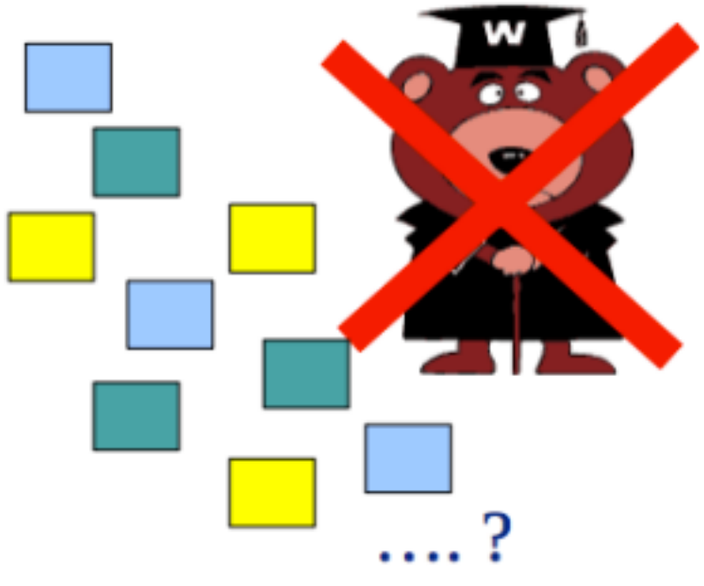
Aprendizado Supervisionado vs Não Supervisionado

APRENDIZADO SUPERVISIONADO VS NÃO SUPERVISIONADO

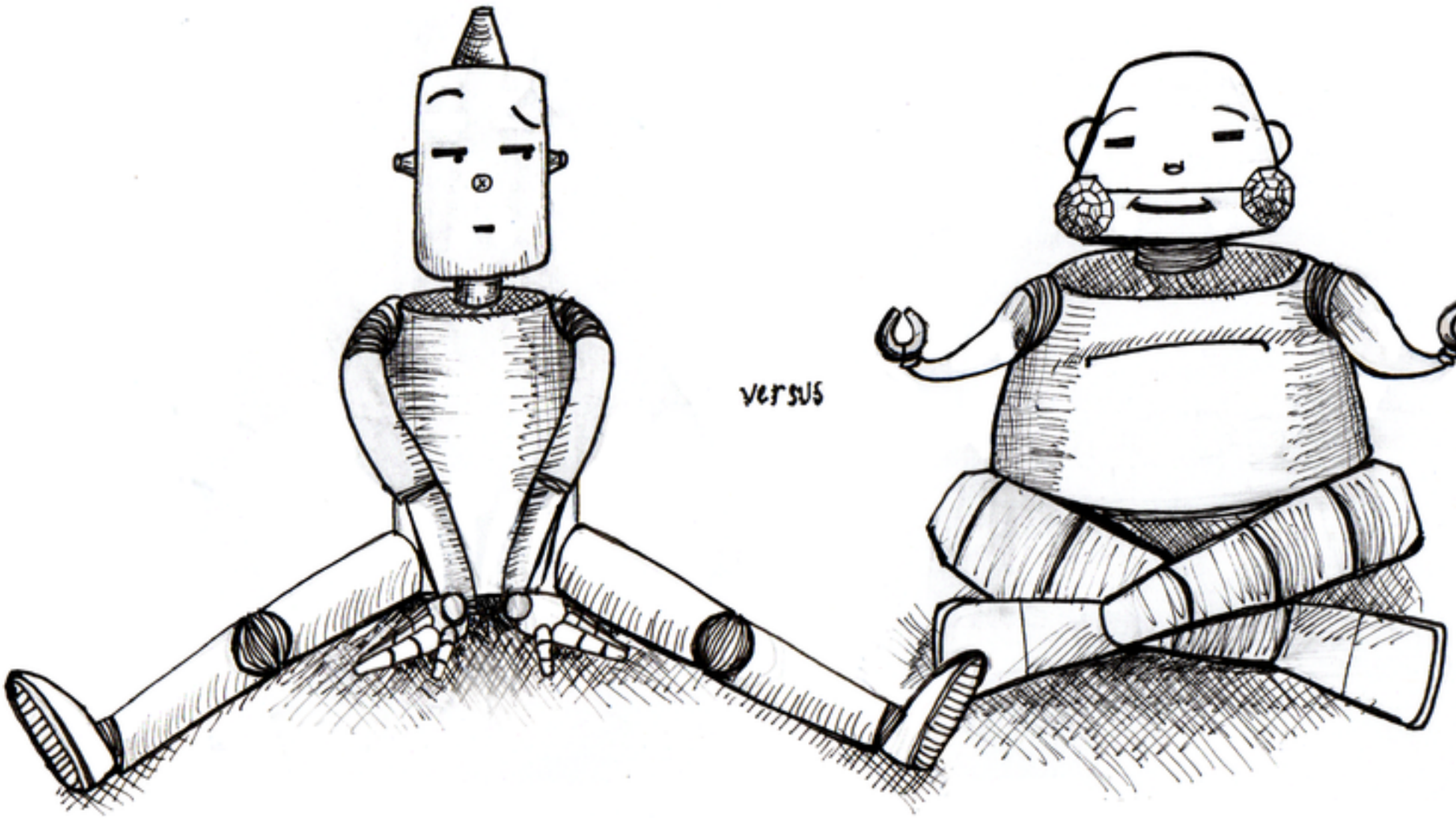
- ***Supervisionado***
  - Conhecimento das entradas e saídas de dados
  - Os dados possuem um label
  - O objetivo é predizer a classe ou o label do dado



- ***Não Supervisionado***
  - Sem conhecimento prévio dos dados
  - O objetivo é determinar padrões

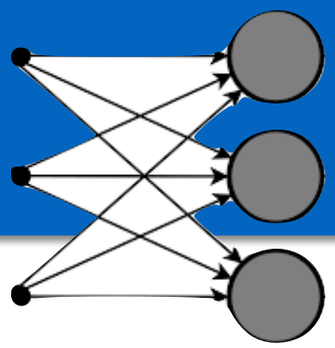


How can developments in deep learning make for a better approach to value investing?

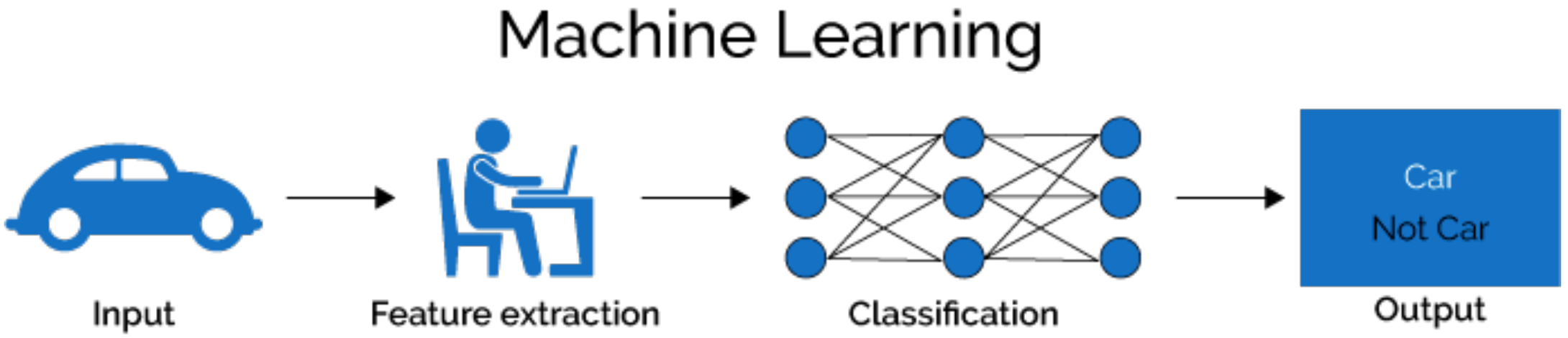


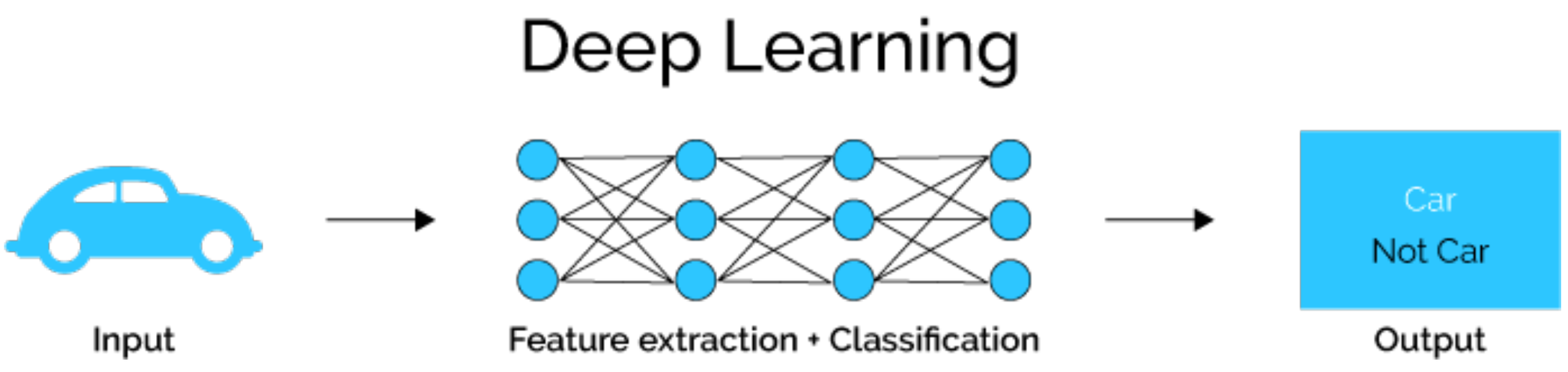
MACHINE LEARNING

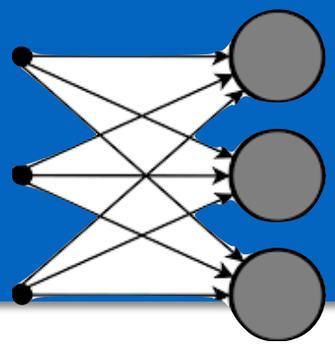
DEEP LEARNING



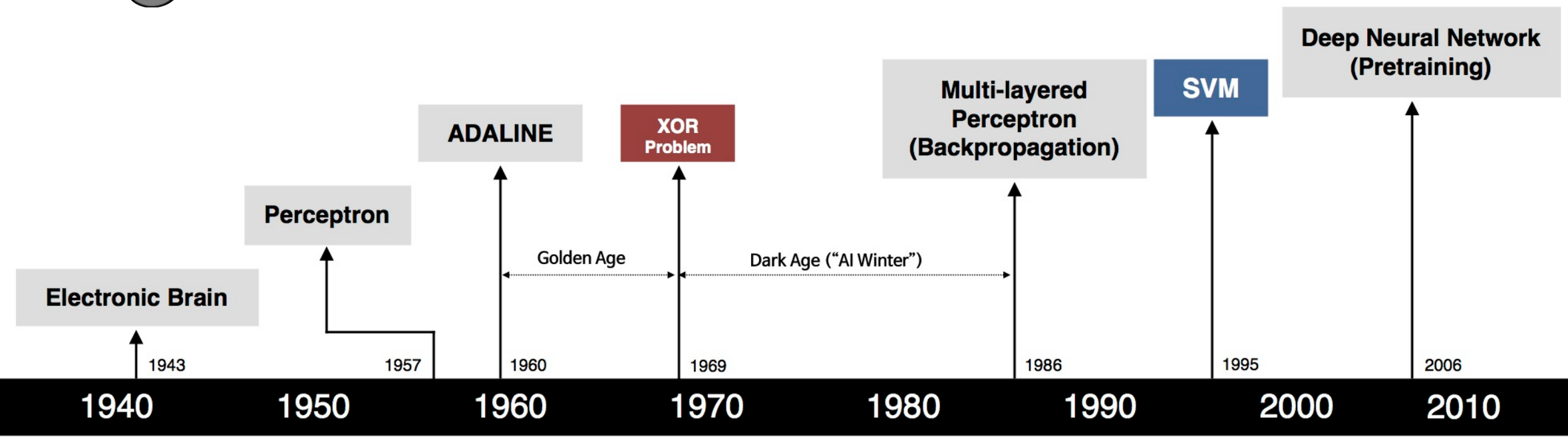
O que é Deep Learning?







Histórico



S. McCulloch – W. Pitts

F. Rosenblatt

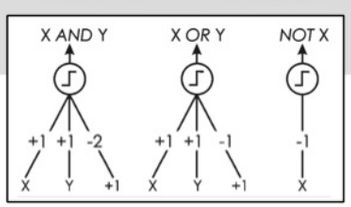
B. Widrow – M. Hoff

M. Minsky – S. Papert

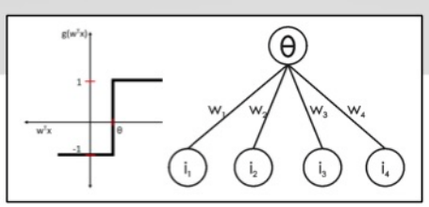
D. Rumelhart – G. Hinton – R. Wiliams

V. Vapnik – C. Cortes

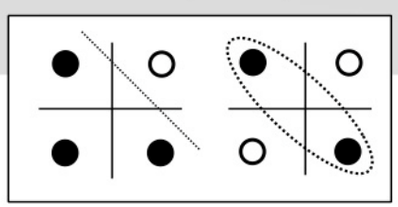
G. Hinton – S. Ruslan



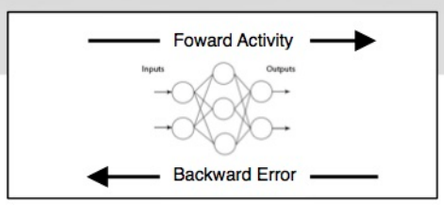
- Adjustable Weights
- Weights are not Learned



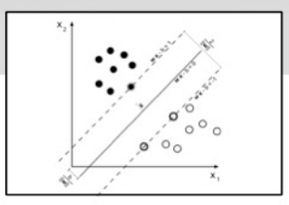
- Learnable Weights and Threshold



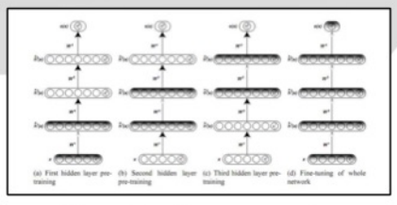
- XOR Problem



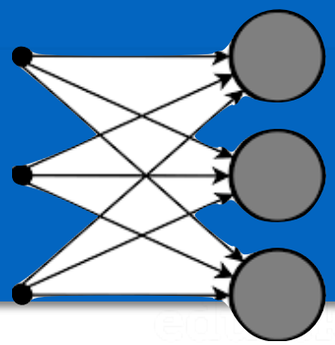
- Solution to nonlinearly separable problems
- Big computation, local optima and overfitting



- Limitations of learning prior knowledge
- Kernel function: Human Intervention



- Hierarchical feature Learning



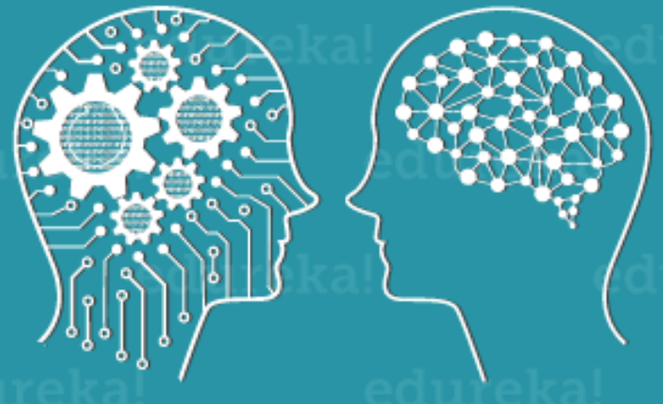
Histórico

ARTIFICIAL INTELLIGENCE

Engineering of making Intelligent Machines and Programs

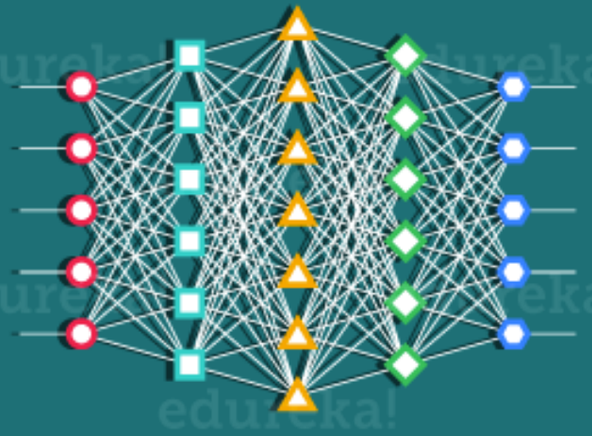
MACHINE LEARNING

Ability to learn without being explicitly programmed

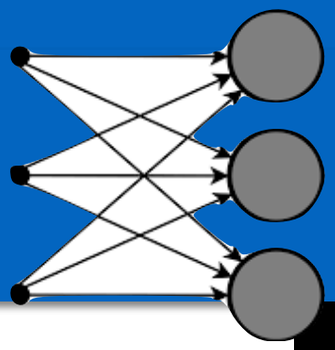


DEEP LEARNING

Learning based on Deep Neural Network



1950's | 1960's | 1970's | 1980's | 1990's | 2000's | 2006's | 2010's | 2012's | 2017's

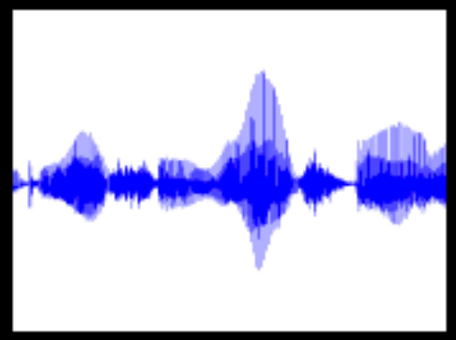


Onde utilizar Deep Learning

Images

Label image

Audio



Speech recognition

Text

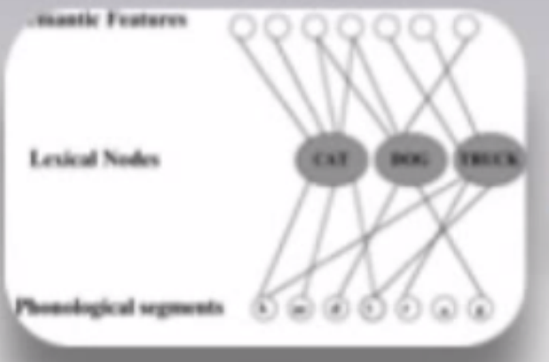
Web search

Application areas

Images

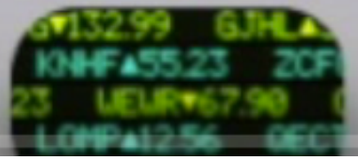
Jennifer Aniston

Speech

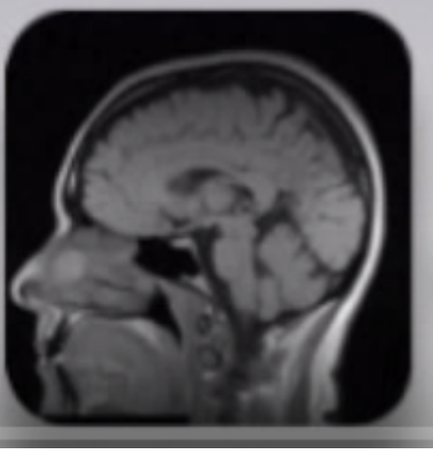


"This is what I'm saying"

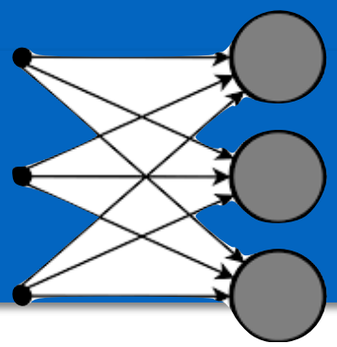
Network security risks



Trading signals

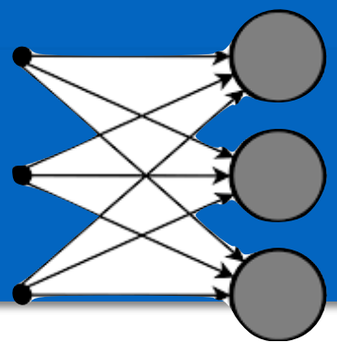


Medical diagnostics

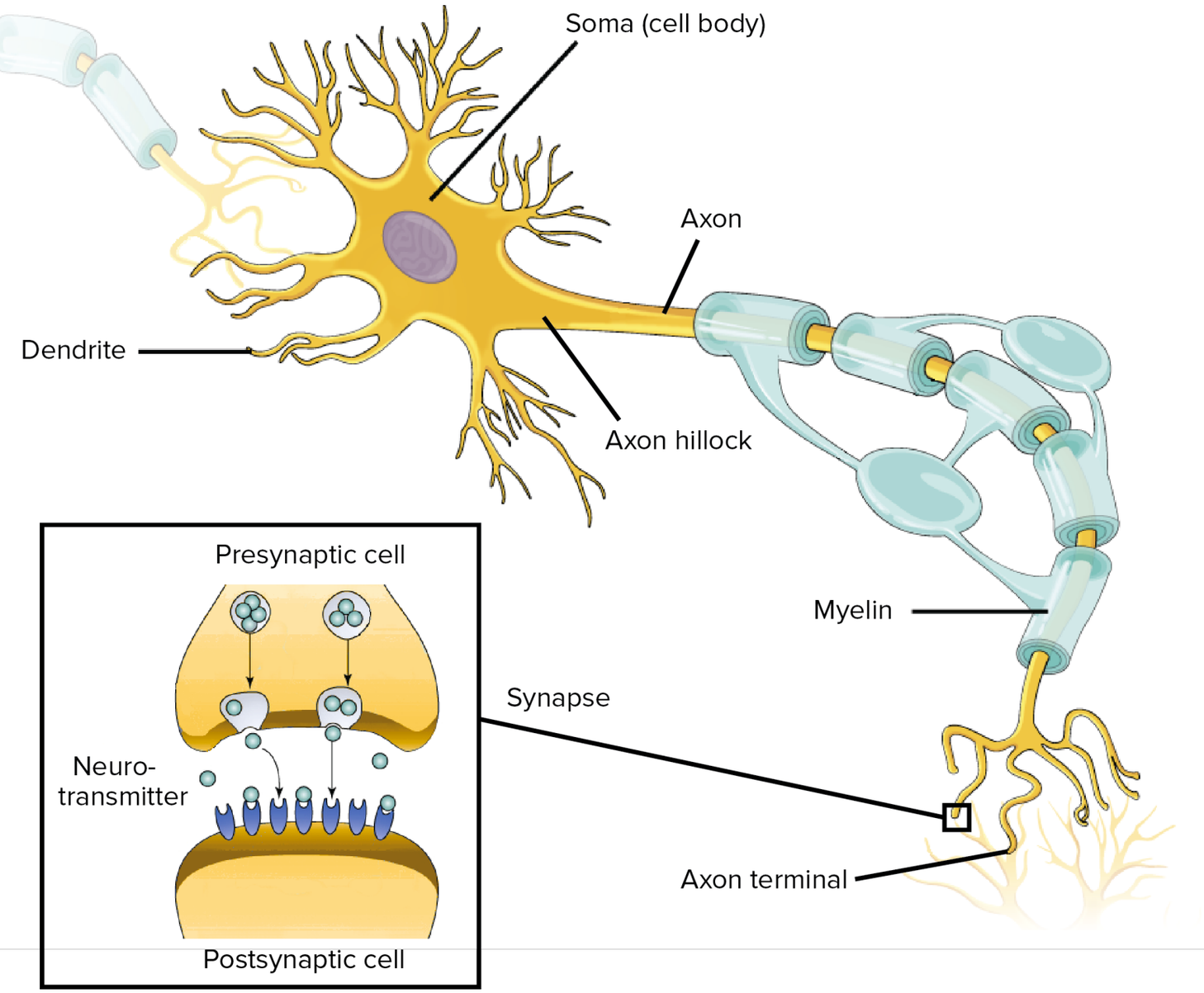


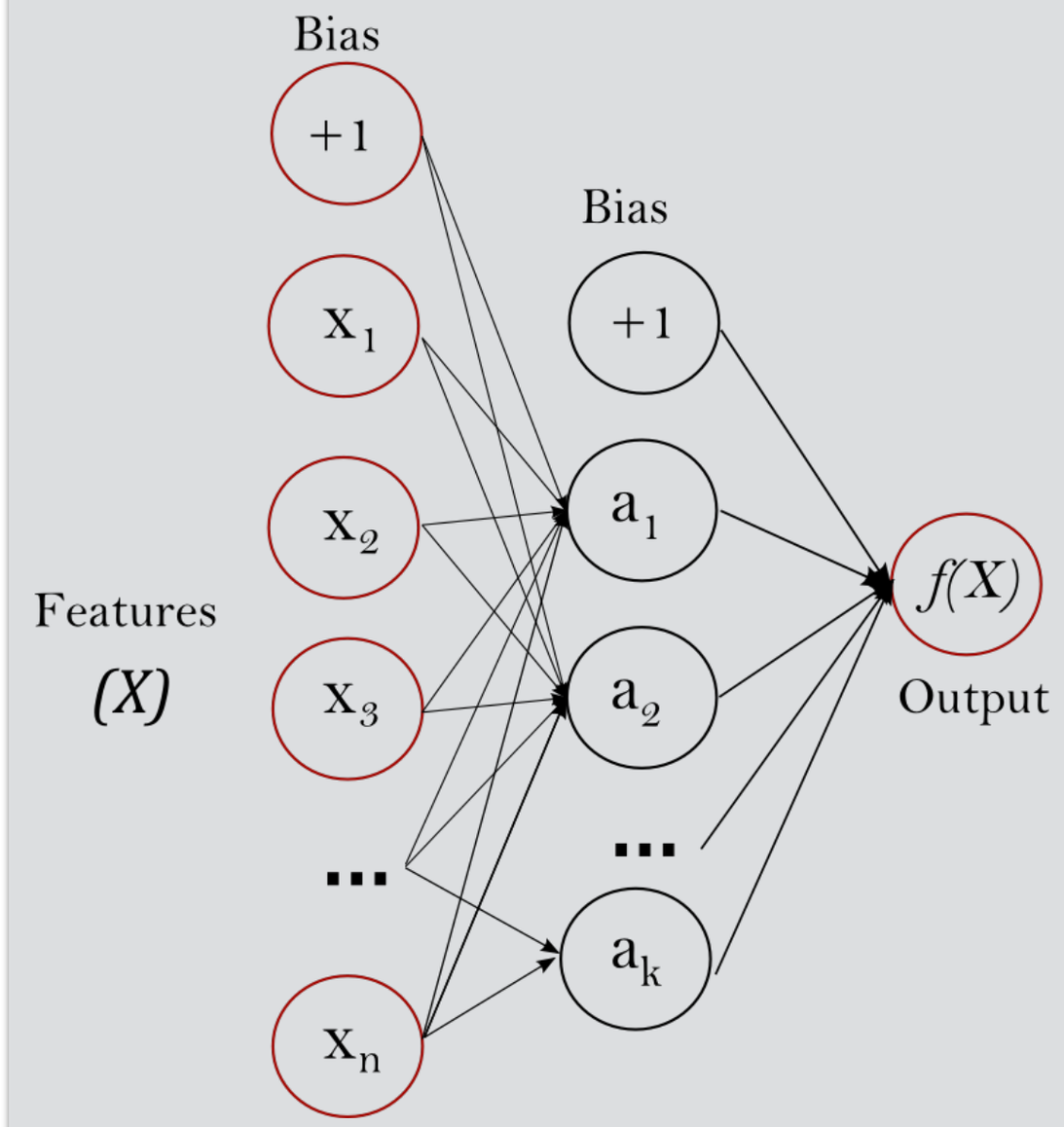
Ferramentas

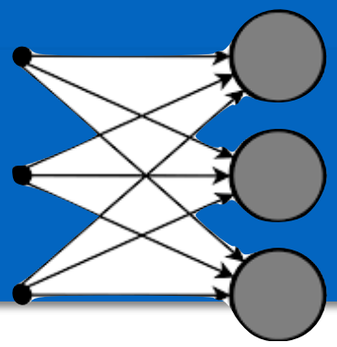
TensorFlow é uma biblioteca de software de código aberto para aprendizagem em máquina em uma variedade de tarefas. É um sistema para construir e treinar redes neurais para detectar e decifrar padrões e correlações, análogo ao (e não o mesmo) aprendizado e raciocínio humano



Abstração Biólogica

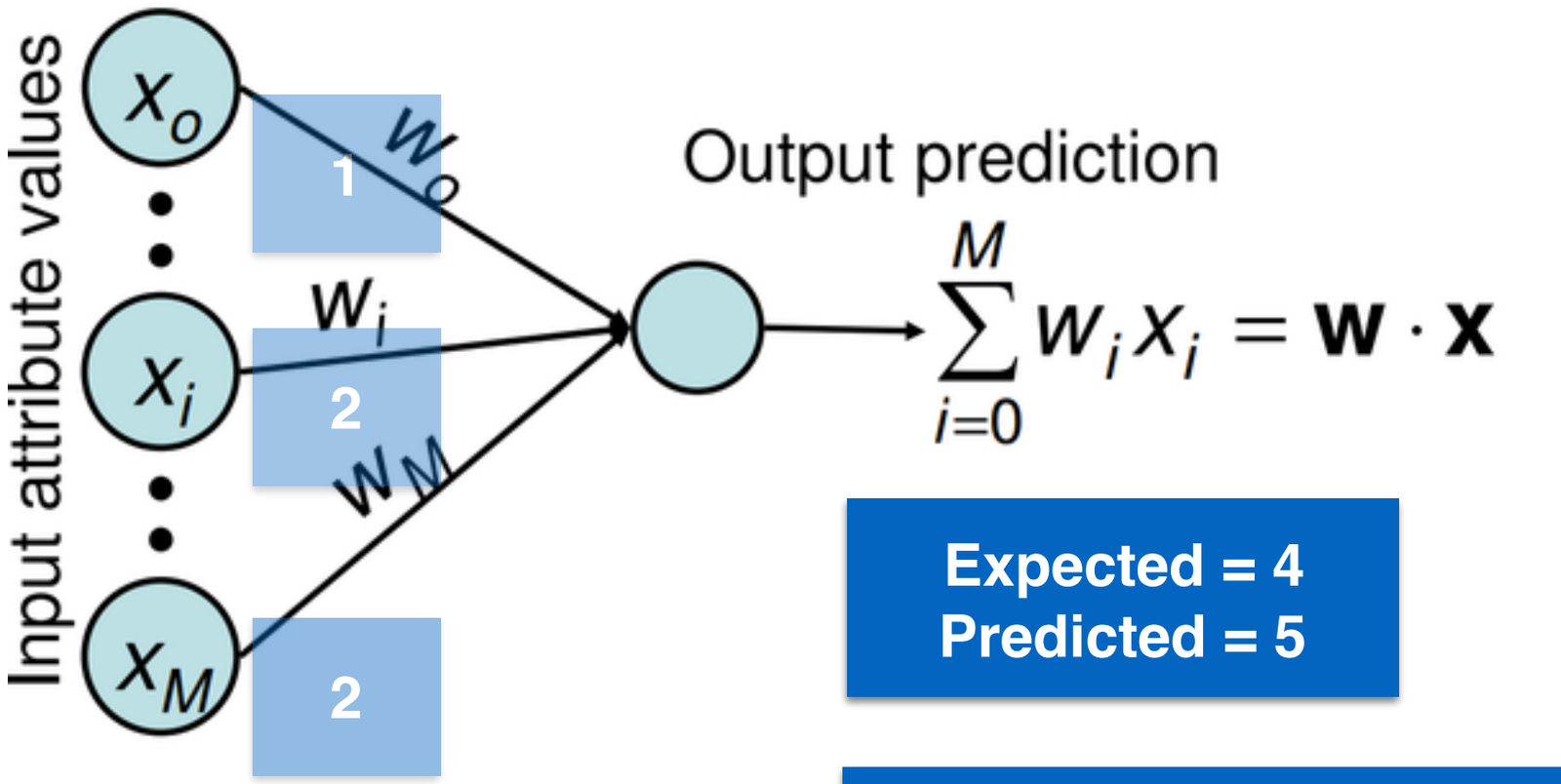






Perceptrons

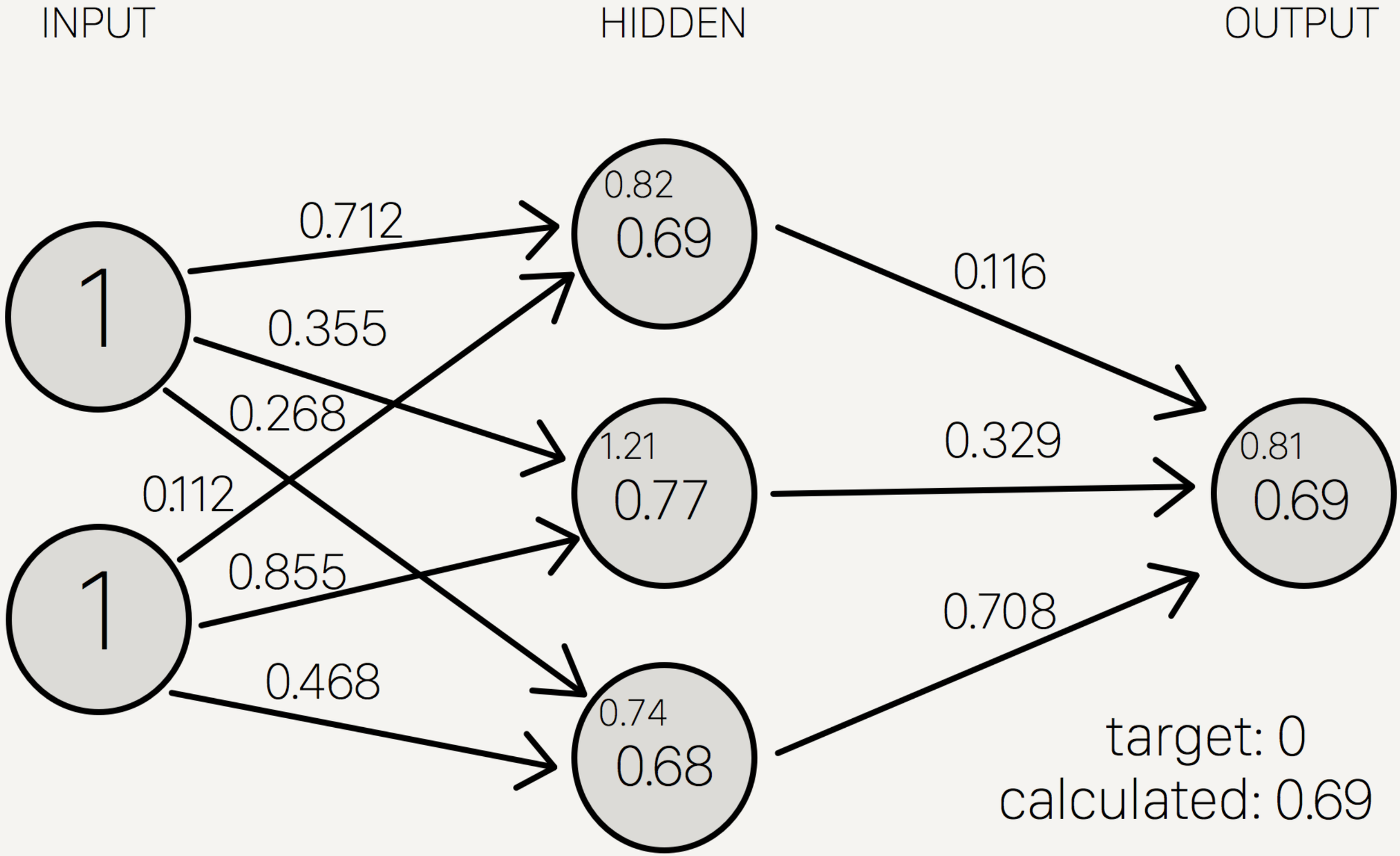
Neural Network: *Linear Perceptron*

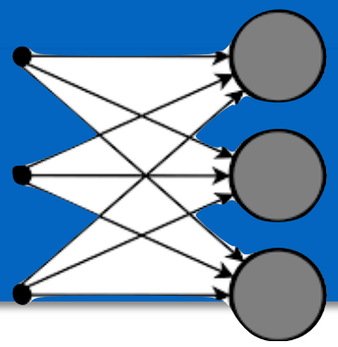


Expected = 4
Predicted = 5

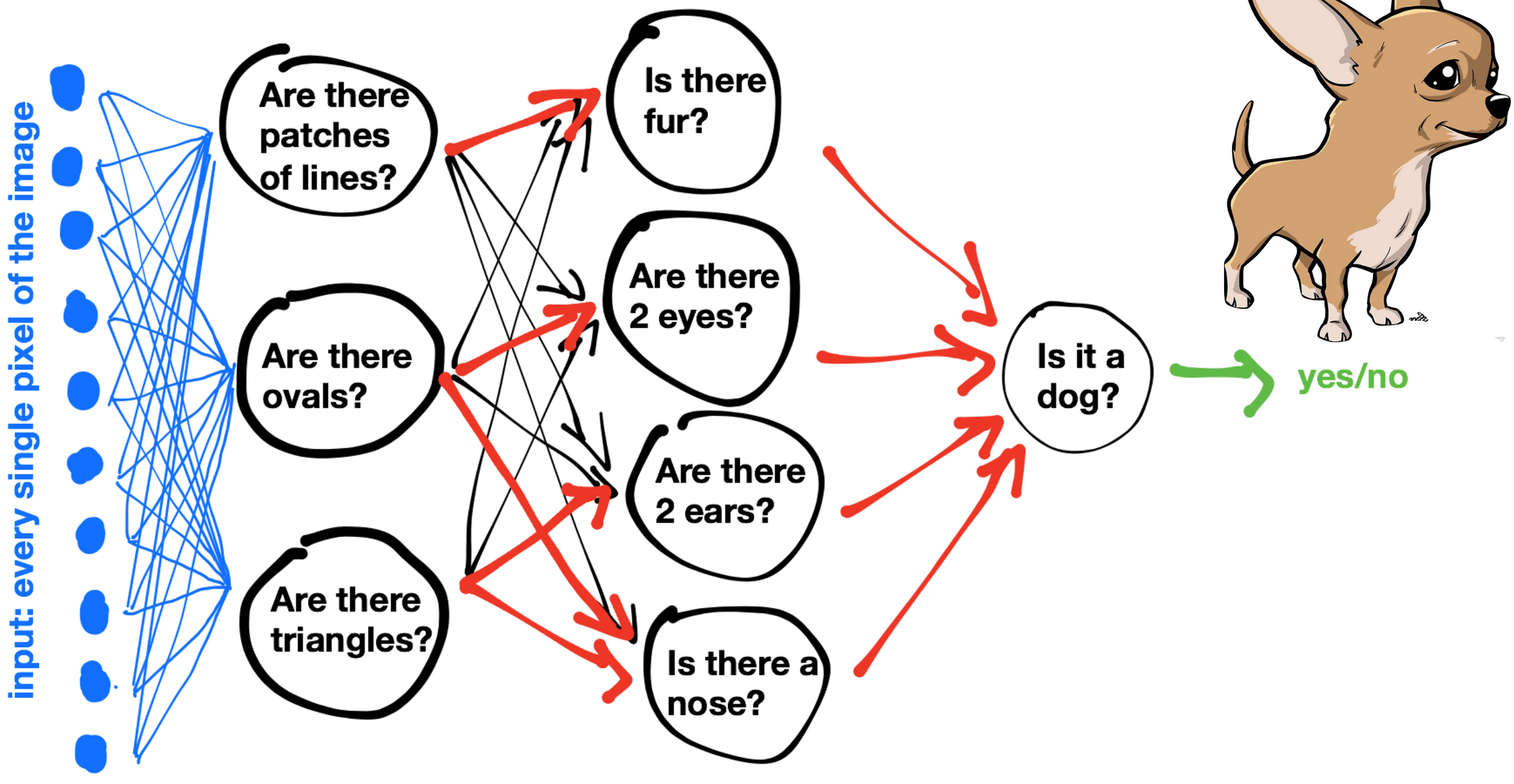
Erro = predicted value - expected value

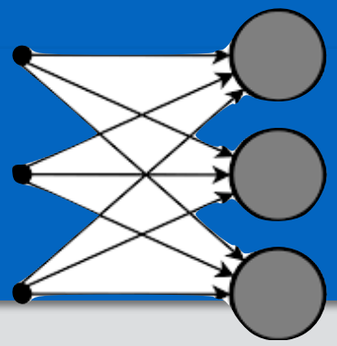
O perceptron é um tipo de rede neural artificial inventada em 1957 no Cornell Aeronautical Laboratory por Frank Rosenblatt. Ele pode ser visto como o tipo mais simples de rede neural feedforward: um classificador linear.


INPUT
HIDDEN
OUTPUT
1
1
0.712
0.355
0.268
0.112
0.855
0.468
0.82
0.69
1.21
0.77
0.74
0.68
0.116
0.329
0.708
0.81
0.69
target: 0
calculated: 0.69

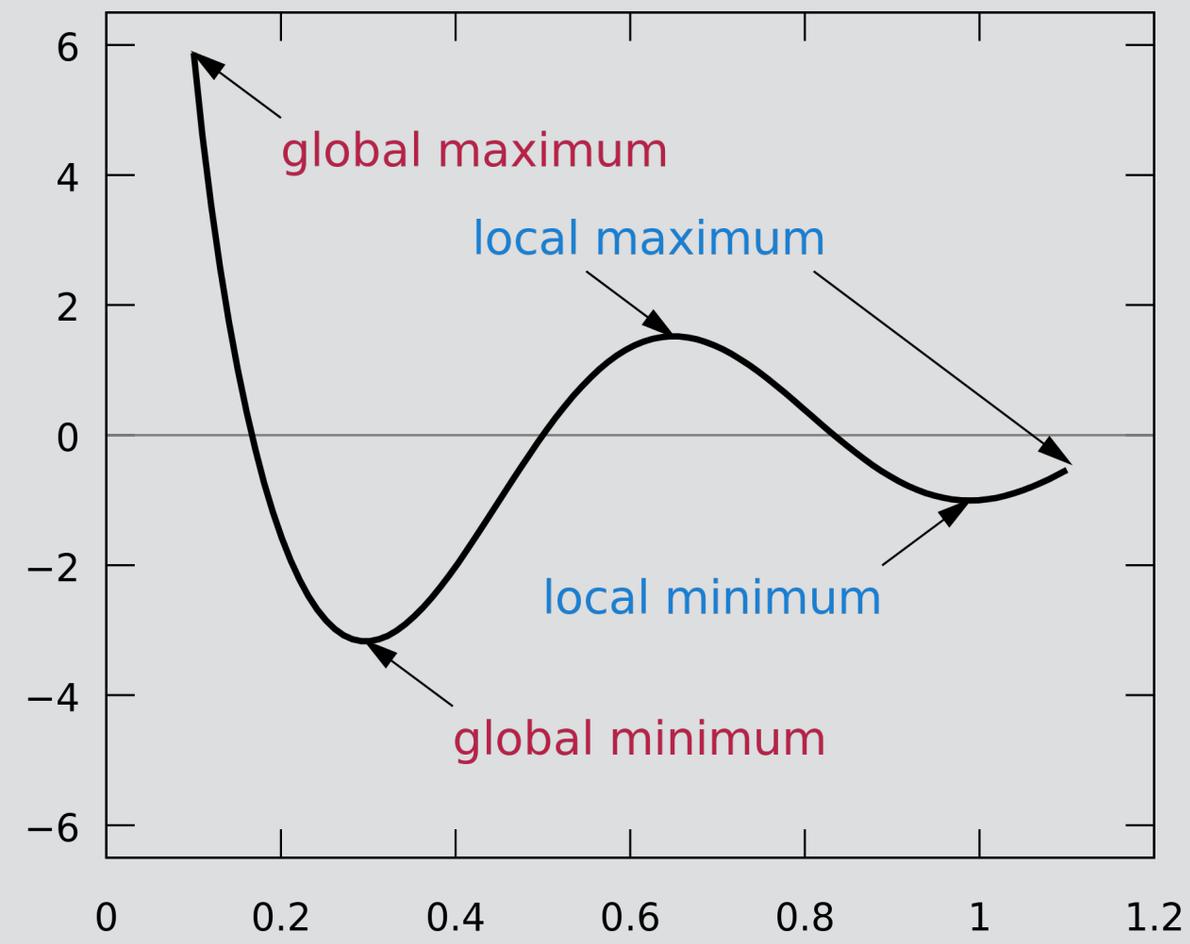


Backpropagation

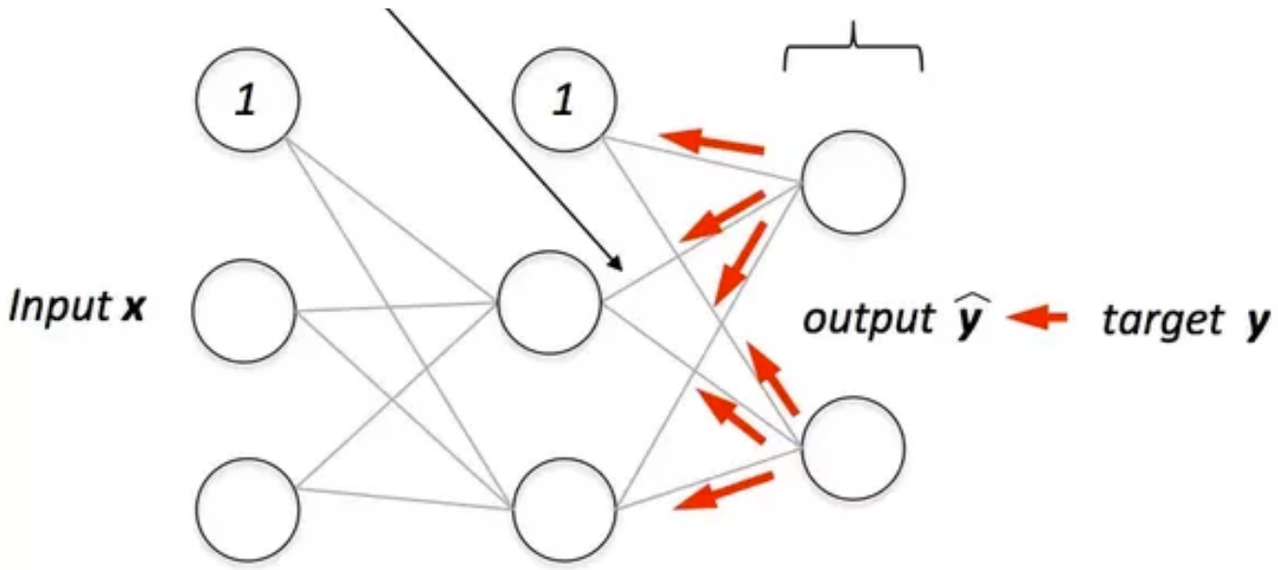


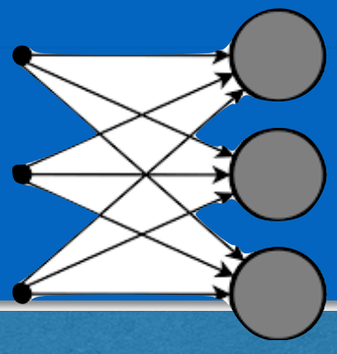


Backpropagation e Gradiente Descendente



peso (w) = peso (w) + taxa_aprendizado * (esperado - predito) * x



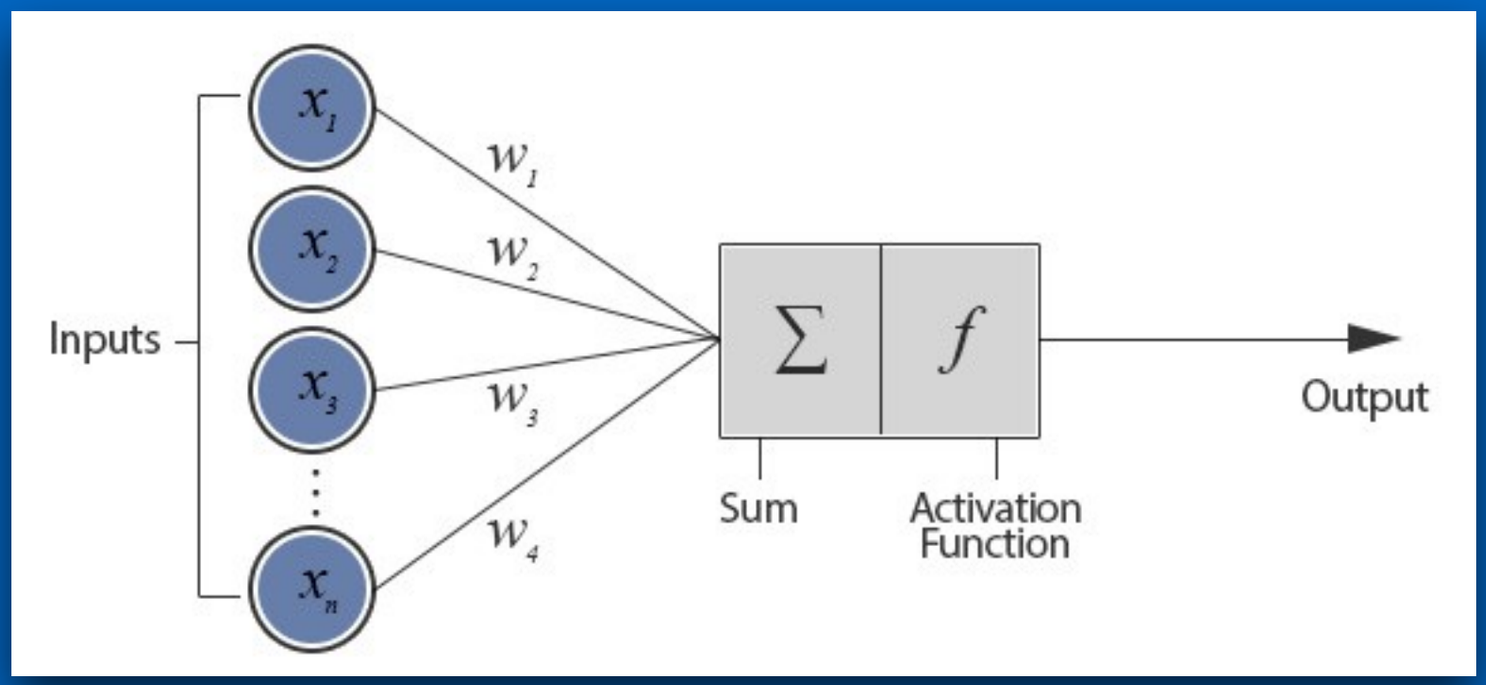


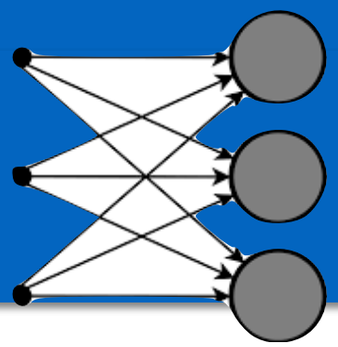
Função de Ativação

Nas redes neurais biologicamente inspirados, a função de ativação é geralmente uma abstração que representa a taxa de potencial de ação na célula.

x1w1 + x2w2 = (0.6 x 0.5) + (1 x 0.8) = 1.1

Input 1 (x1) = 0.6
Input 2 (x2) = 1.0
Weight 1 (w1) = 0.5
Weight 2 (w2) = 0.8
Threshold = 1.0



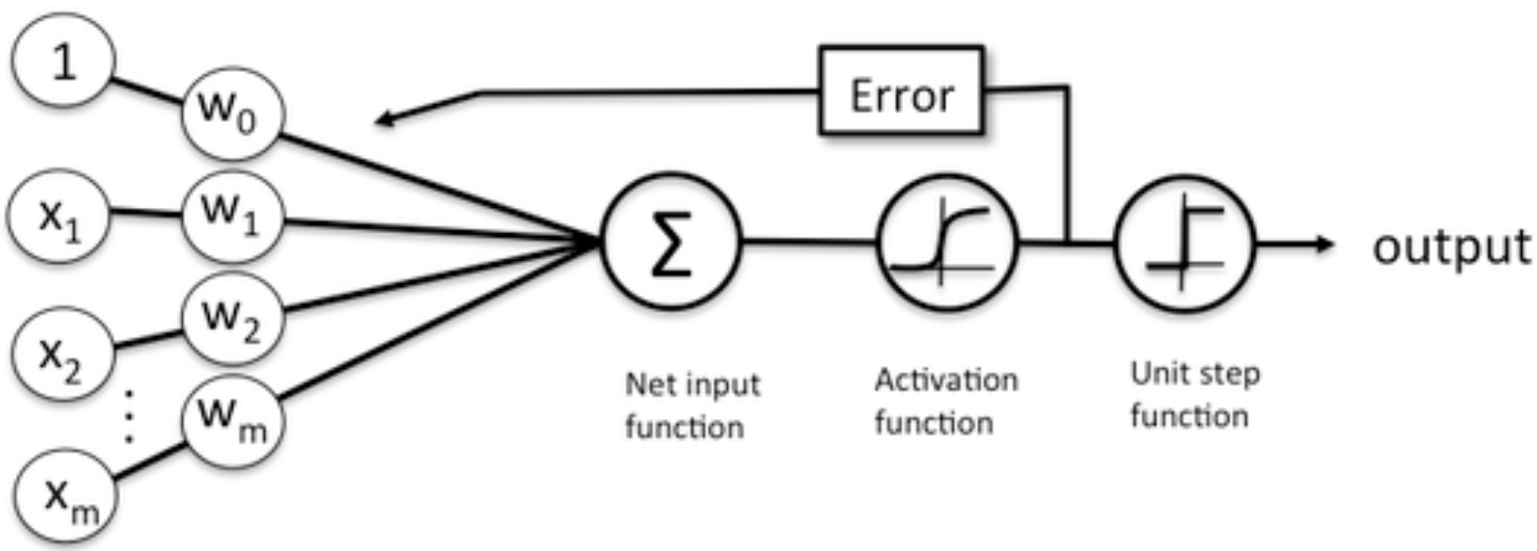


Função de Ativação

Uma função de activação serve como um limite,

ativacao = sum(weight_i * x_i) + bias

predicao = 1.0 if ativacao >= 0.0 else 0.0



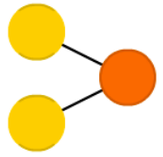
Schematic of a logistic regression classifier.

A mostly complete chart of

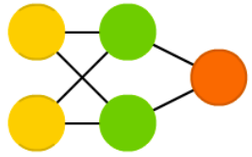
Neural Networks

- 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

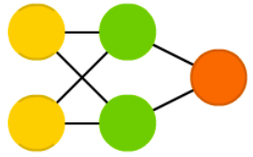
Perceptron (P)



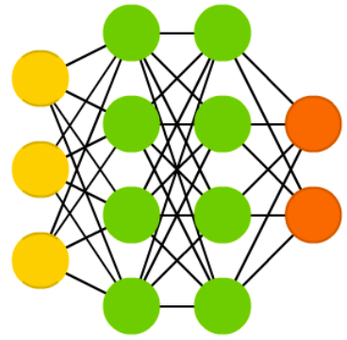
Feed Forward (FF)



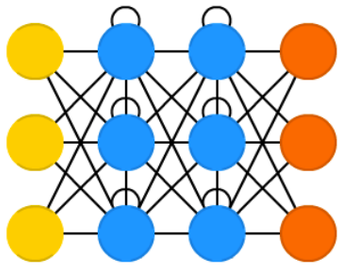
Radial Basis Network (RBF)



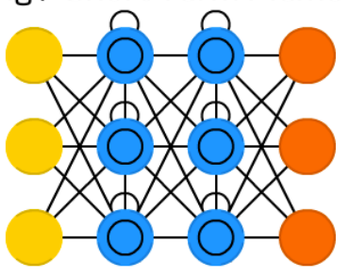
Deep Feed Forward (DFF)



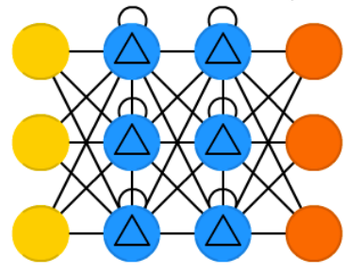
Recurrent Neural Network (RNN)



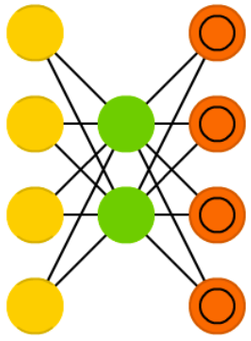
Long / Short Term Memory (LSTM)



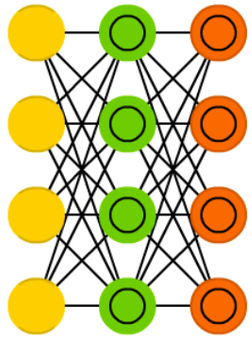
Gated Recurrent Unit (GRU)



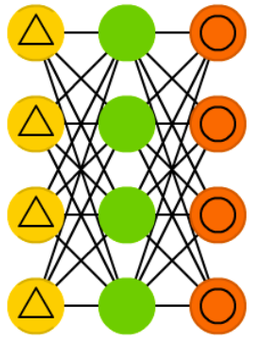
Auto Encoder (AE)



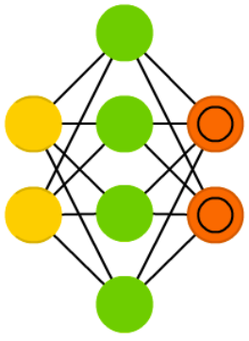
Variational AE (VAE)



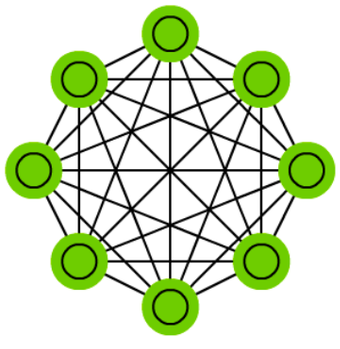
Denoising AE (DAE)



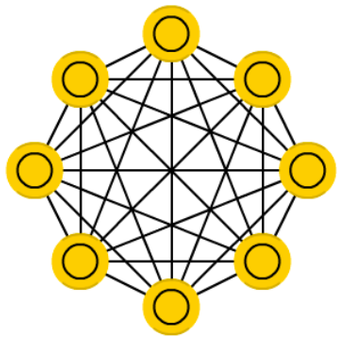
Sparse AE (SAE)



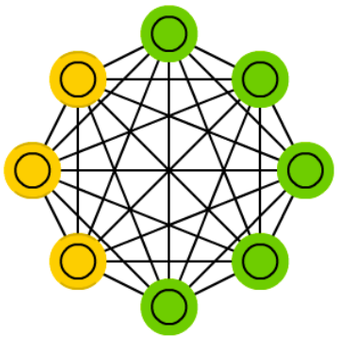
Markov Chain (MC)



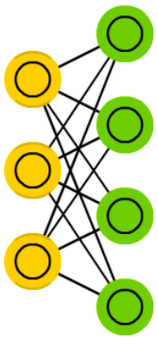
Hopfield Network (HN)



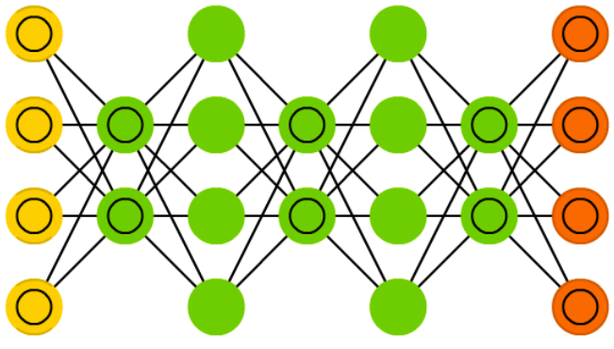
Boltzmann Machine (BM)



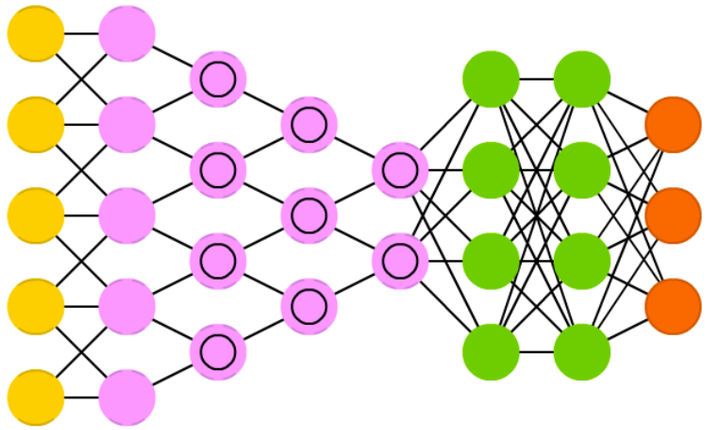
Restricted BM (RBM)



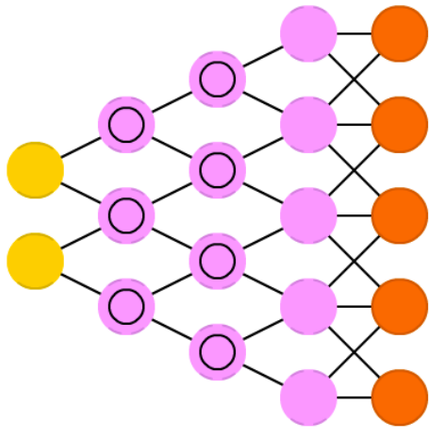
Deep Belief Network (DBN)



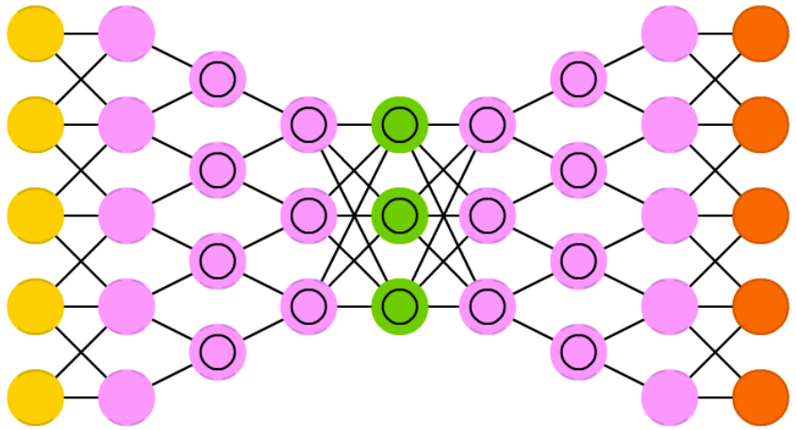
Deep Convolutional Network (DCN)



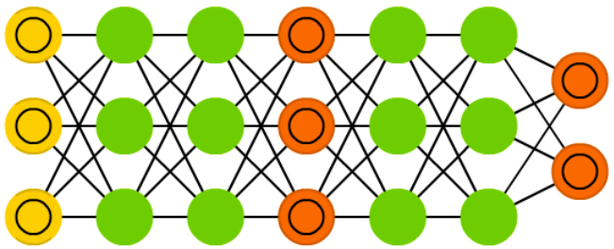
Deconvolutional Network (DN)



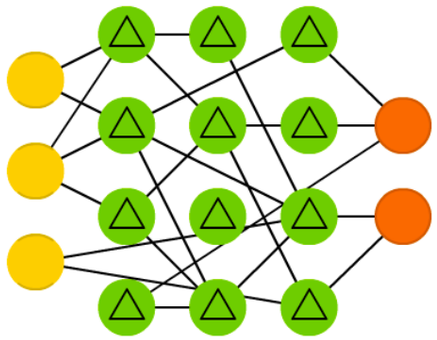
Deep Convolutional Inverse Graphics Network (DCIGN)



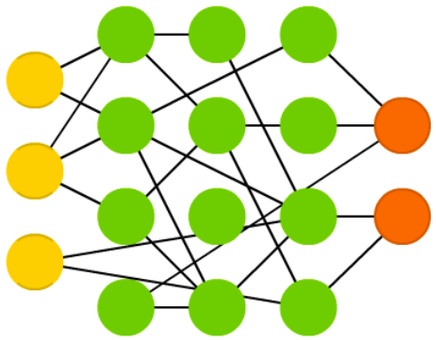
Generative Adversarial Network (GAN)



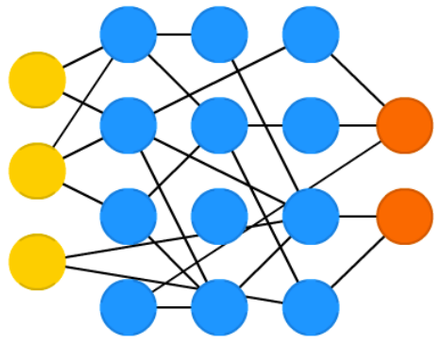
Liquid State Machine (LSM)



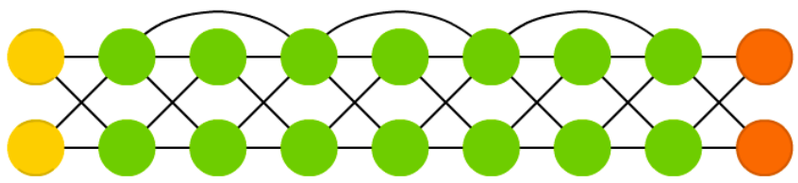
Extreme Learning Machine (ELM)



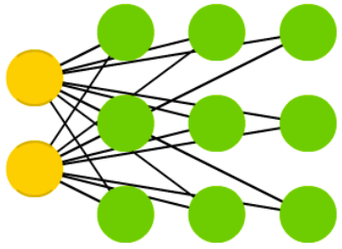
Echo State Network (ESN)



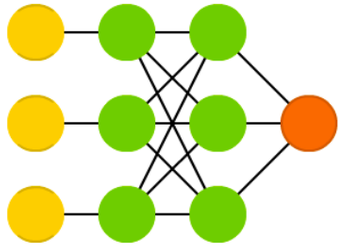
Deep Residual Network (DRN)



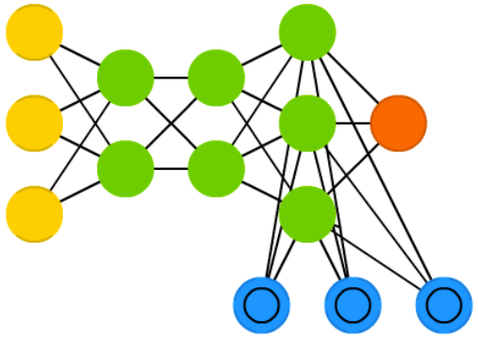
Kohonen Network (KN)

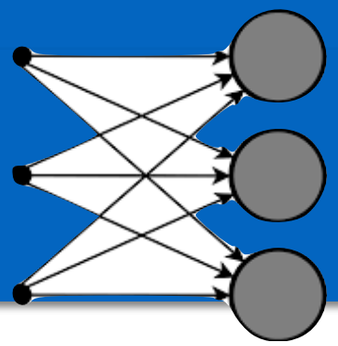


Support Vector Machine (SVM)

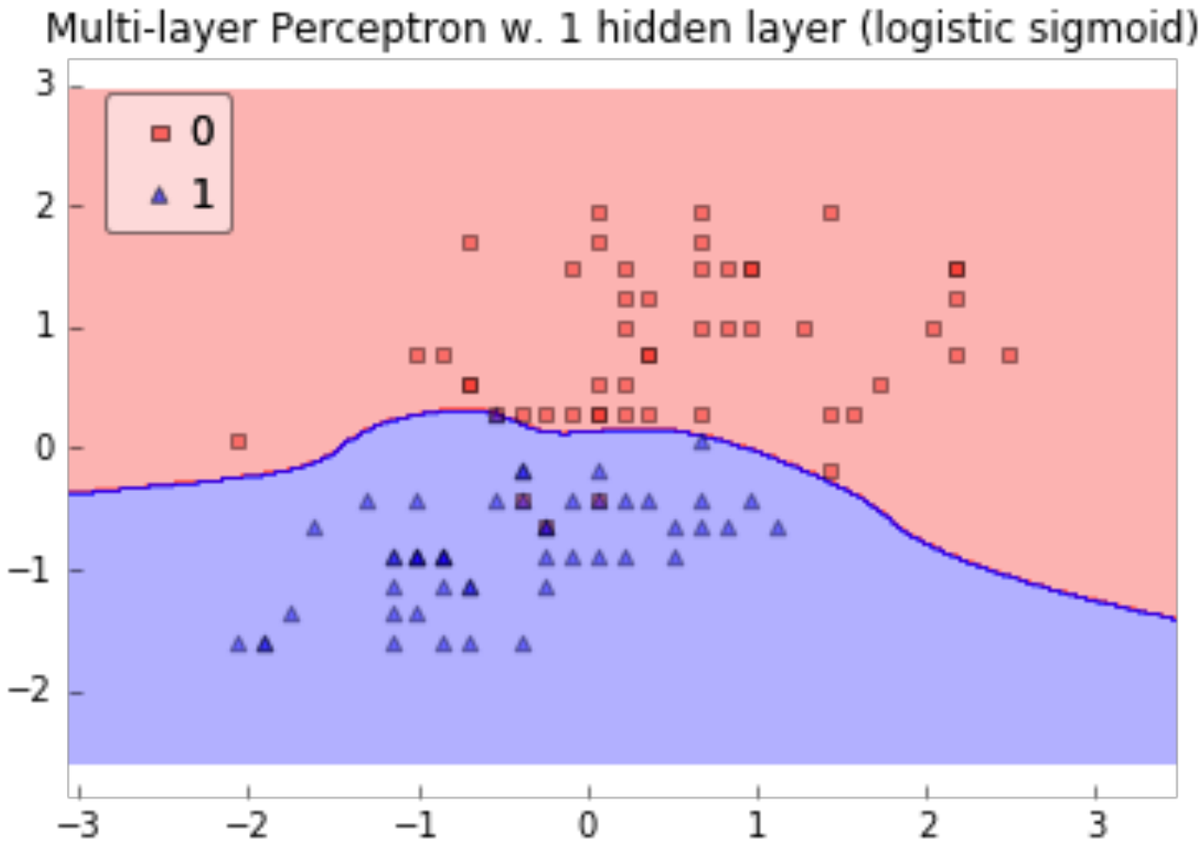


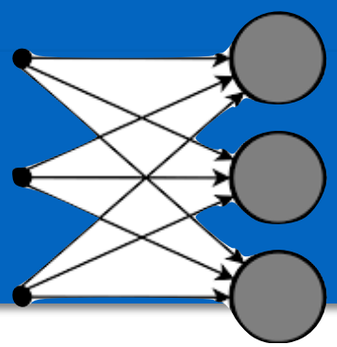
Neural Turing Machine (NTM)





Função de Ativação





Experimento com Redes Neurais

http://playground.tensorflow.org/

DATA

Which dataset do you want to use?

FEATURES

Which properties do you want to feed in?

4 HIDDEN LAYERS

4 neurons | 2 neurons | 2 neurons | 2 neurons

X_1, X_2, X_1^2, X_2^2, X_1X_2, $\sin(X_1)$, $\sin(X_2)$

*This is the output from one **neuron**. Hover to see it larger.*

*The outputs are mixed with varying **weights**, shown by the thickness of the lines.*

OUTPUT

Test loss 0.004

Training loss 0.000

Ratio of training to test data: 20%

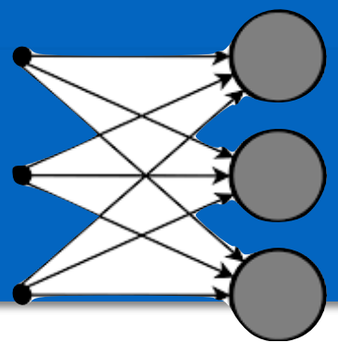
Noise: 0

Batch size: 10

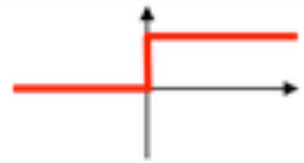
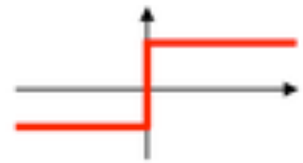
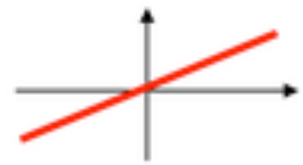
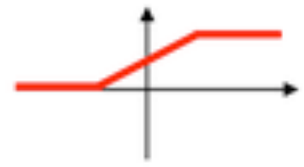
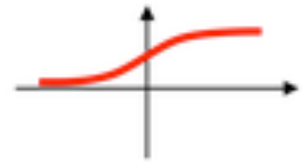
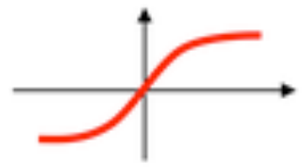
REGENERATE

Colors shows data, neuron and weight values. (-1, 0, 1)

Show test data | Discretize output

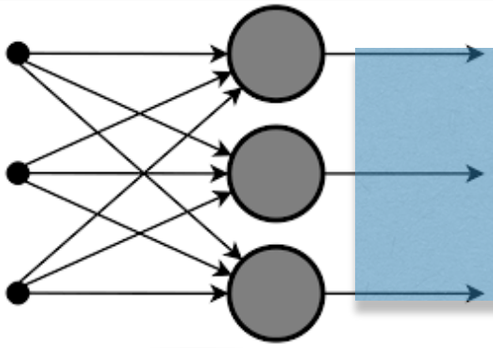


Função de Ativação

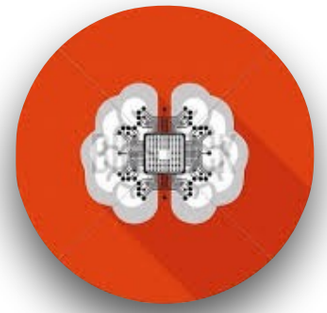
Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multi-layer NN	

CNN RNN GAN

Agenda

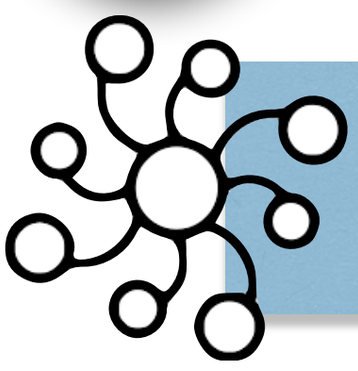


Introdução a Deep Learning



Redes Neurais Convolucionais

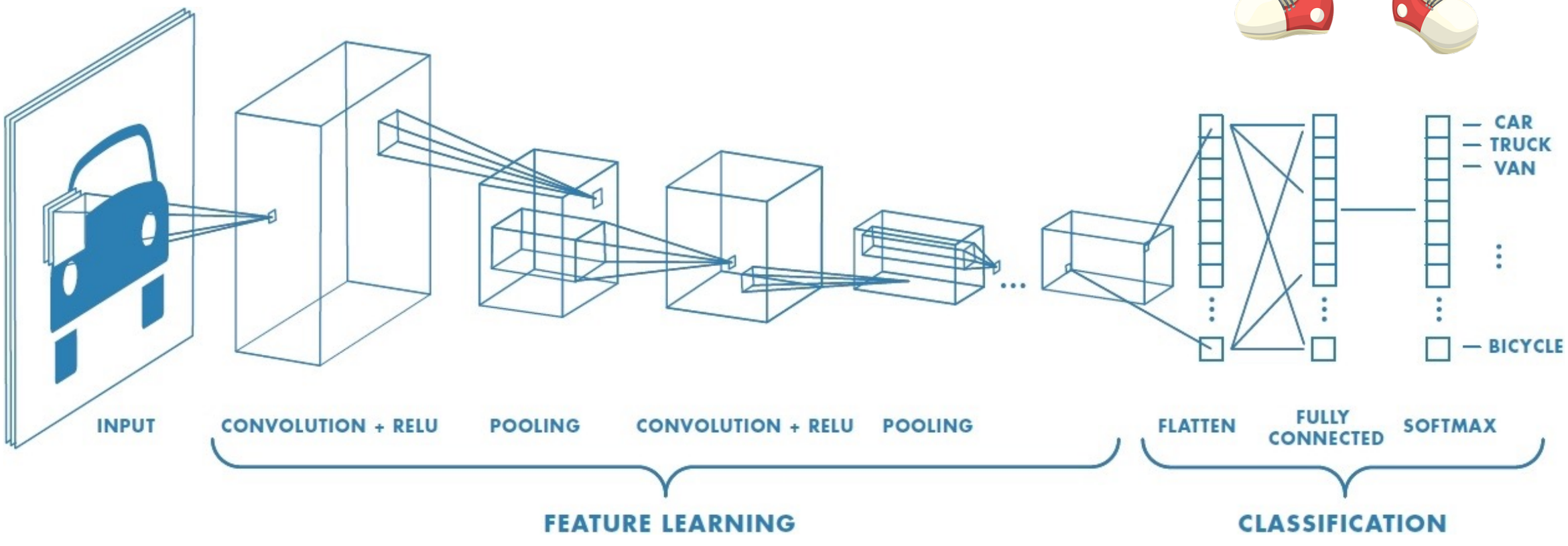
Recurrent Neural Networks



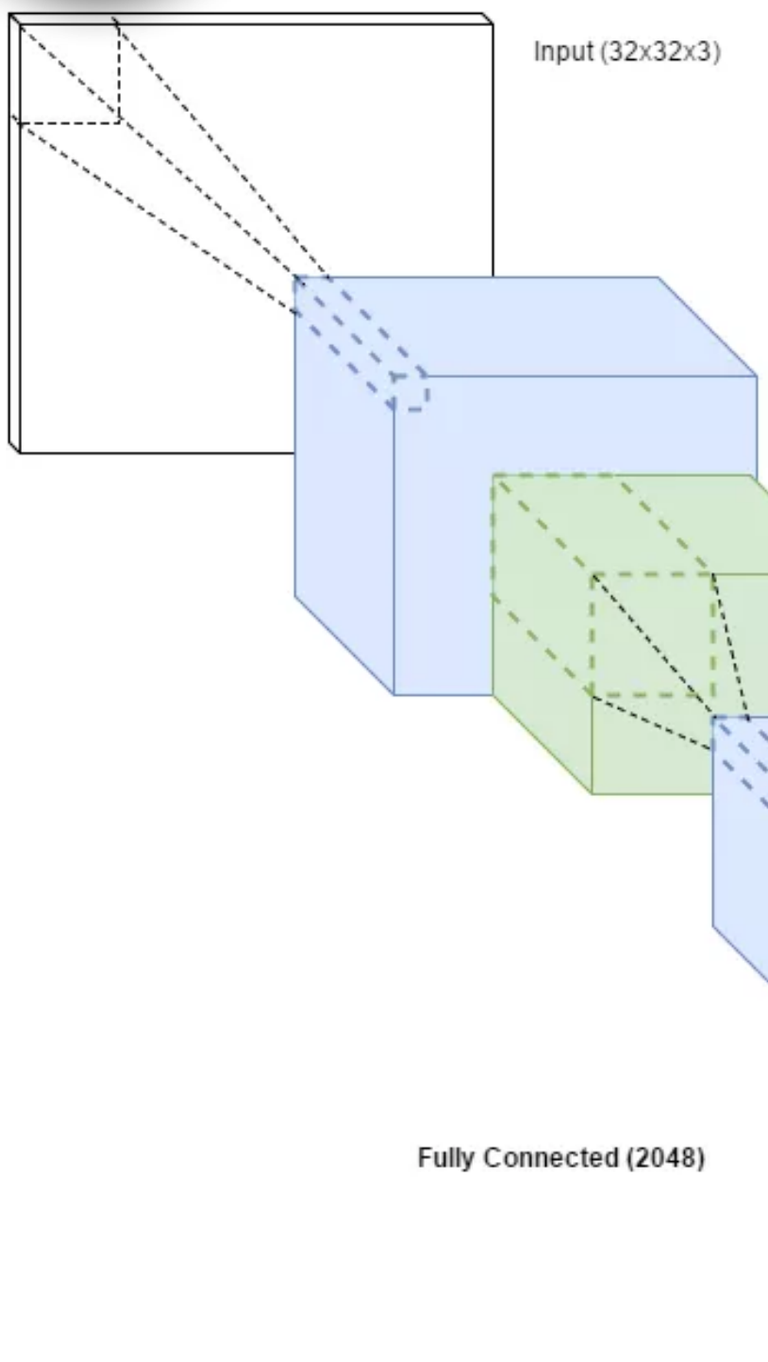
Generative Adversarial Networks

Convolutional Neural Network

Tipo de rede foi inspirada em processos biológicos e são variações de redes multilayer perceptron que são desenhadas o mínimo de pré-processamento possível. Tipo de rede é geralmente usada em reconhecimento de imagens e vídeo.



Convolutional Neural Network

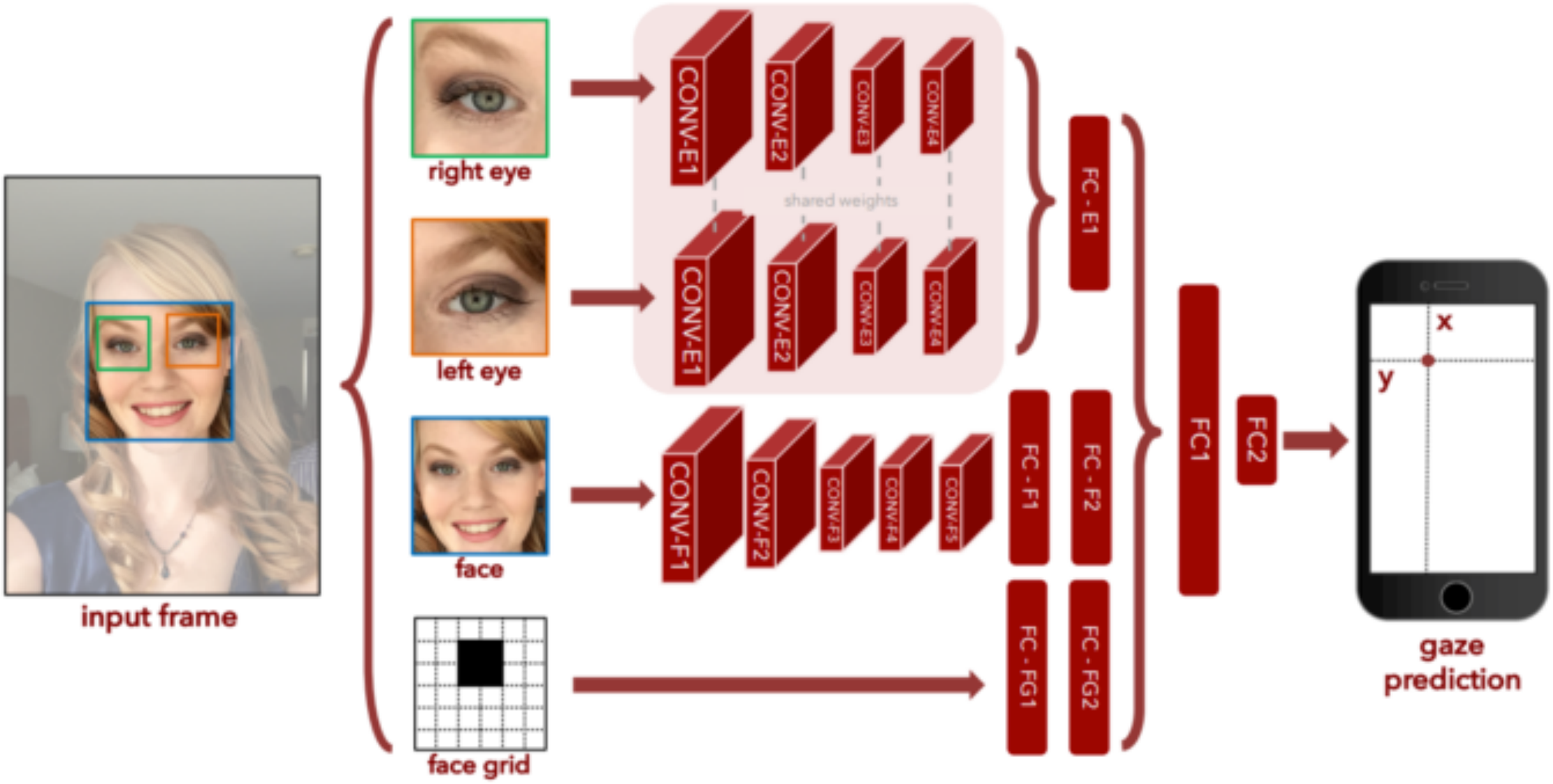


Convolução

Pooling

Convolução

Pooling


input frame
right eye
left eye
face
face grid
CONV-E1
CONV-E2
CONV-E3
CONV-E4
shared weights
CONV-F1
CONV-F2
CONV-F3
CONV-F4
CONV-F5
FC - E1
FC - F1
FC - F2
FC - FG1
FC - FG2
FC1
FC2
x
y
gaze
prediction

Convolução

$$\begin{pmatrix} 0 & 1 & 2 \\ 2 & 2 & 0 \\ 0 & 1 & 2 \end{pmatrix}$$

Kernel

3_0	3_1	2_2	1	0
0_2	0_2	1_0	3	1
3_0	1_1	2_2	2	3
2	0	0	2	2
2	0	0	0	1

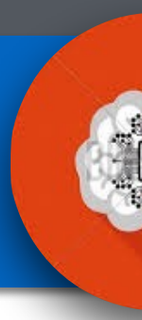
12	12	17
10	17	19
9	6	14

Input Vector

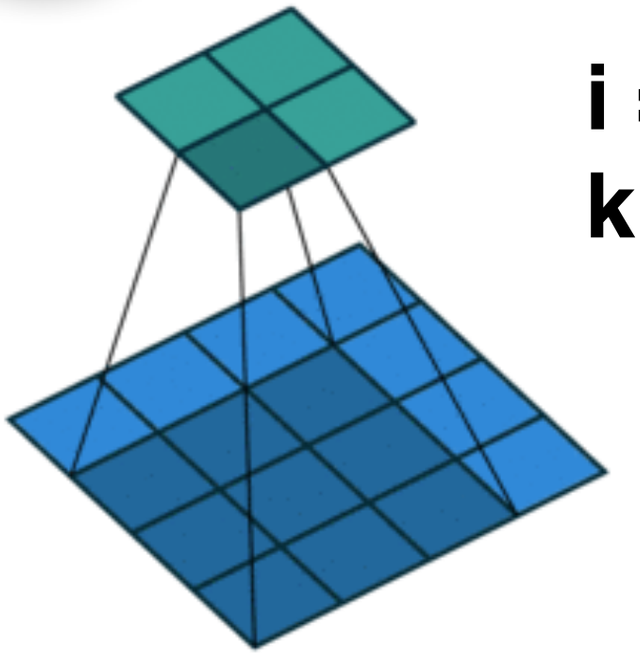
Padding

3	3_0	2_1	1_2	0
0	0_2	1_2	3_0	1
3	1_0	2_1	2_2	3
2	0	0	2	2
2	0	0	0	1

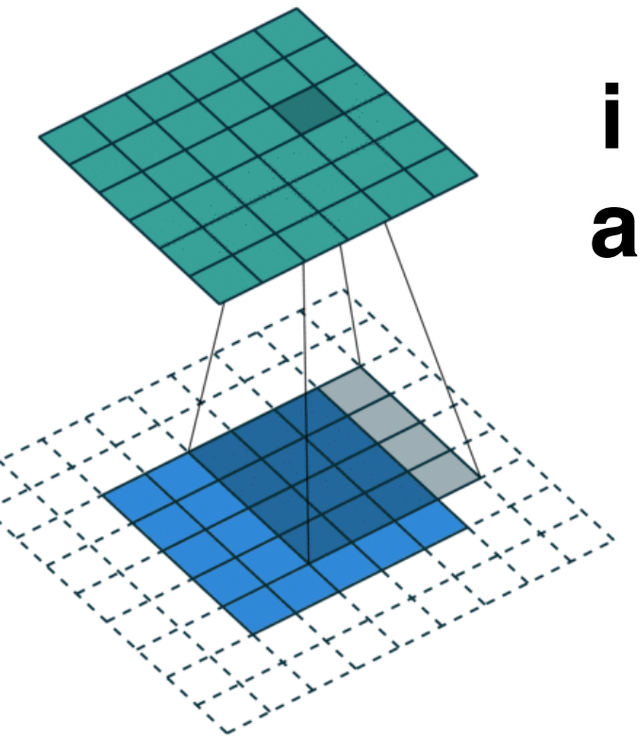
12	12	17
10	17	19
9	6	14



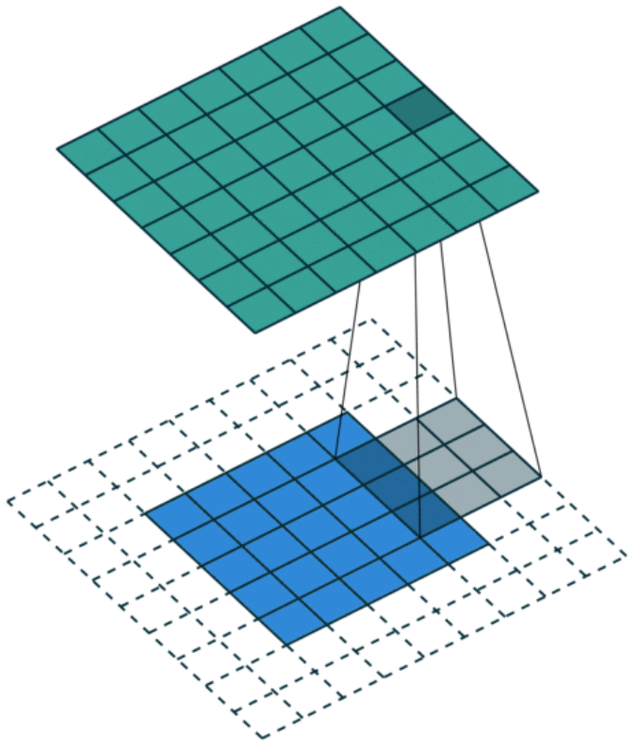
Convolução



i = 4 and k = 3

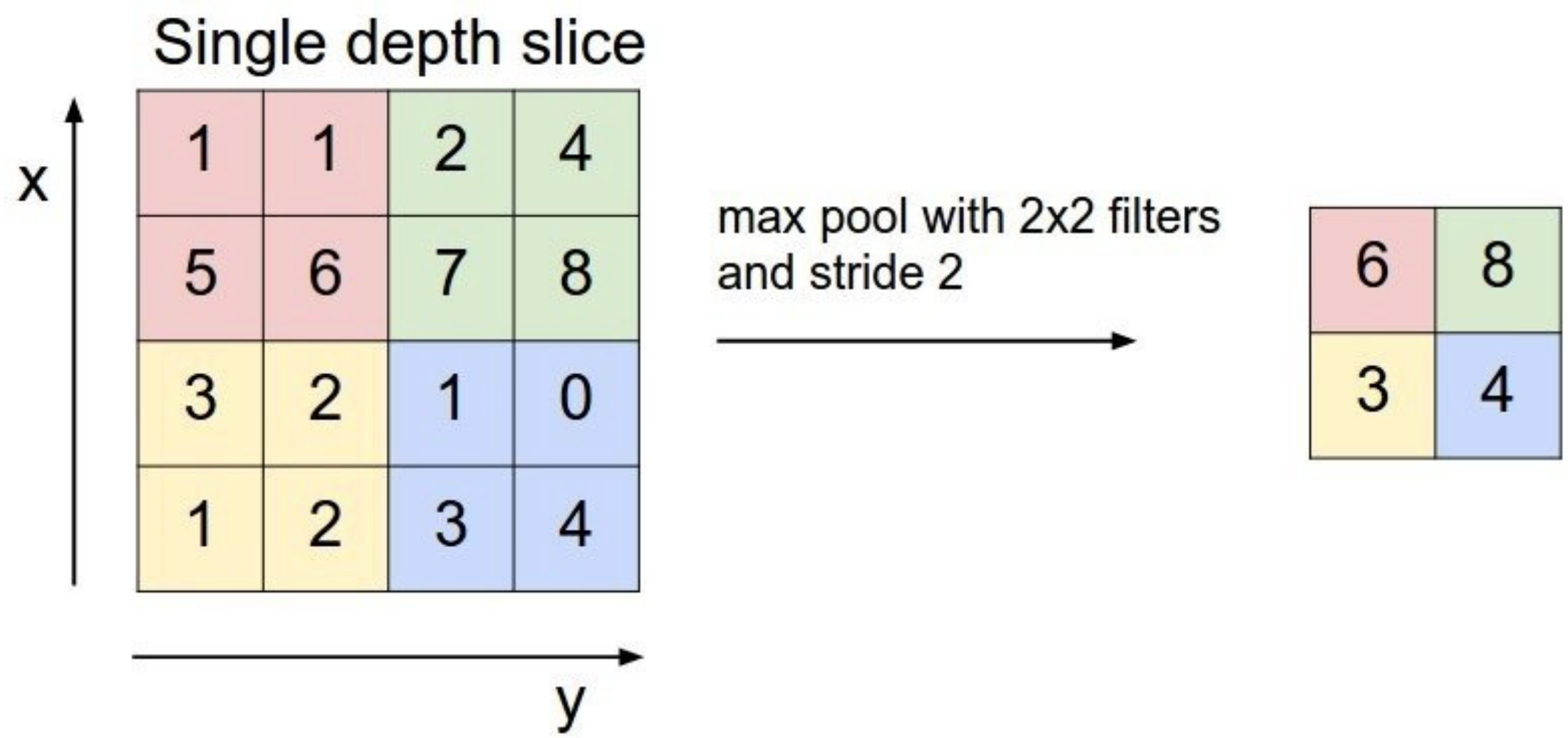


i = 5, k = 4 and p = 2:

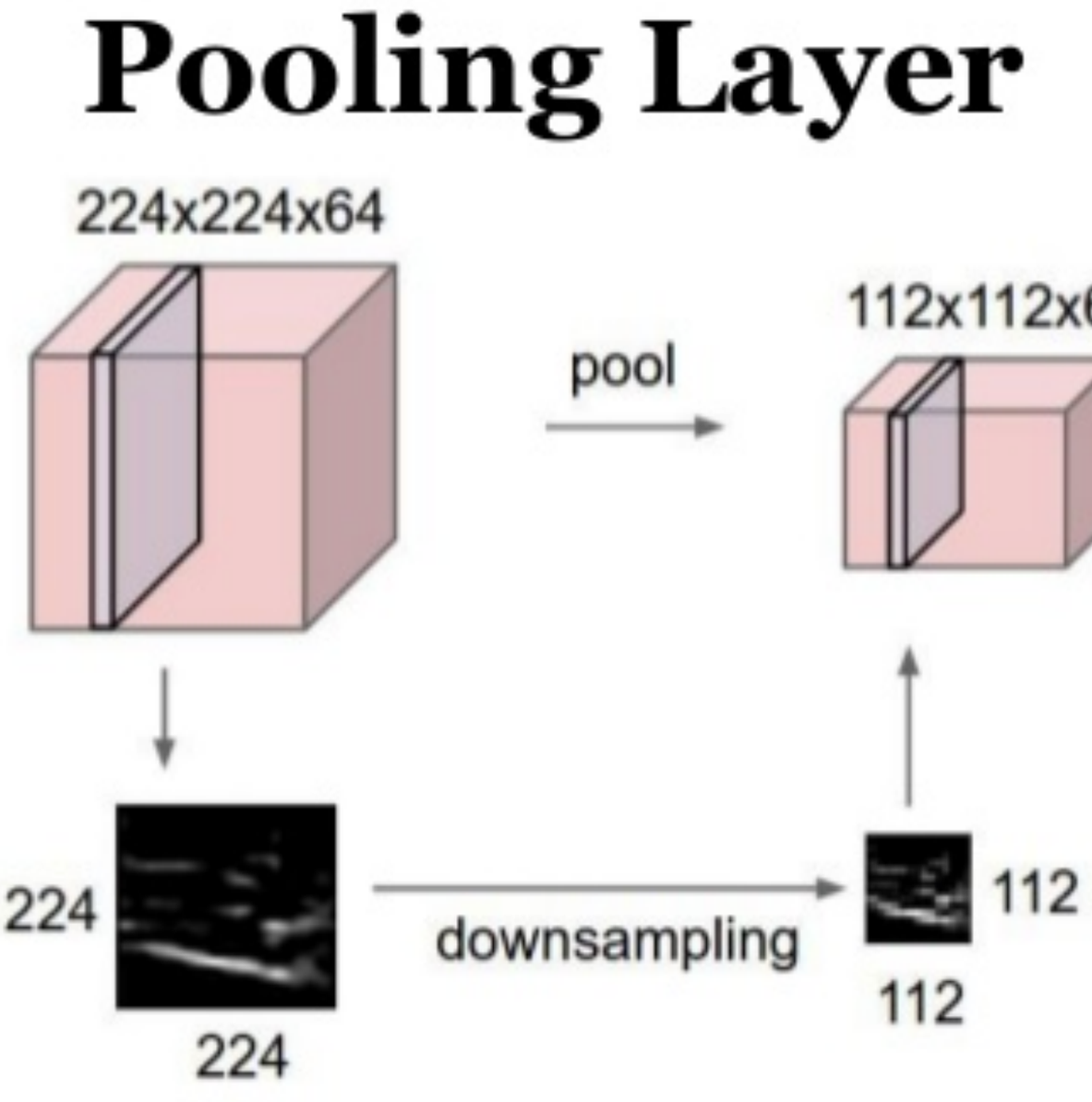


i = 5, k = 3 and p = 2:

Pooling

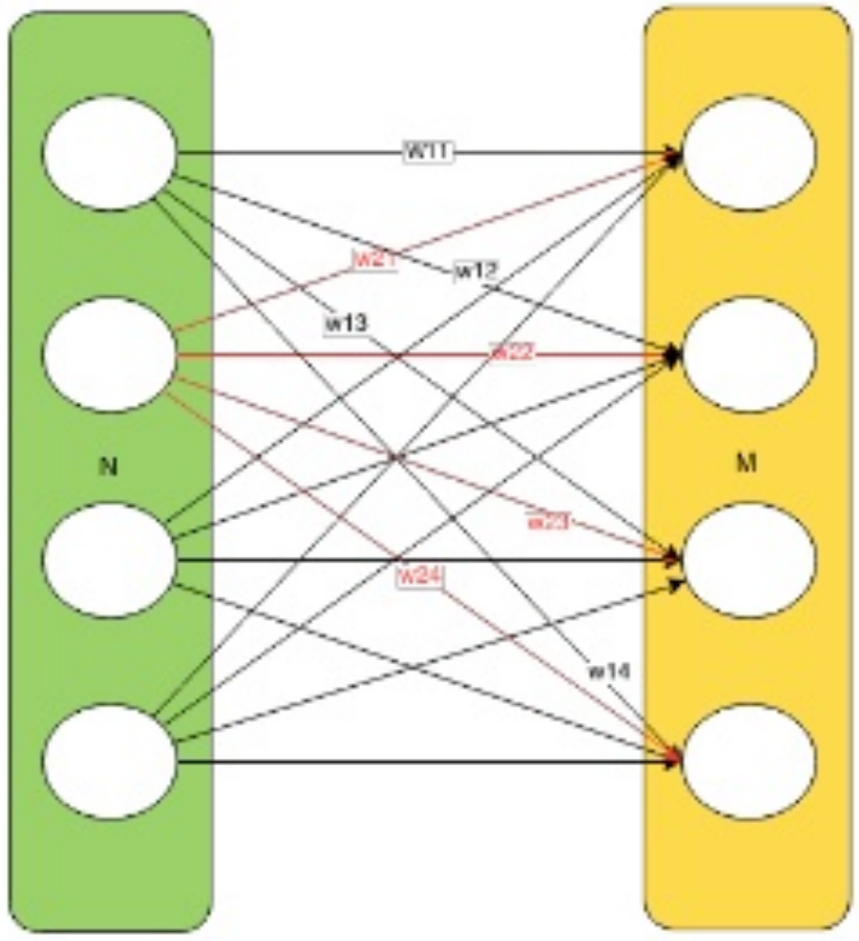

Single depth slice
x
y
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
max pool with 2x2 filters
and stride 2
6 8
3 4

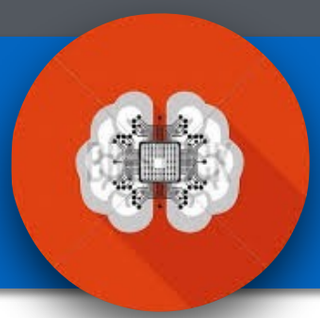
Pooling


Pooling Layer
224x224x64
pool
112x112x64
224
224
downsampling
112
112

Rede Neural

Fully-connected layer





Casos de Sucesso

Case studies

LeNet : The first successful applications of CNN

AlexNet: The first work that popularized CNN in Computer Vision

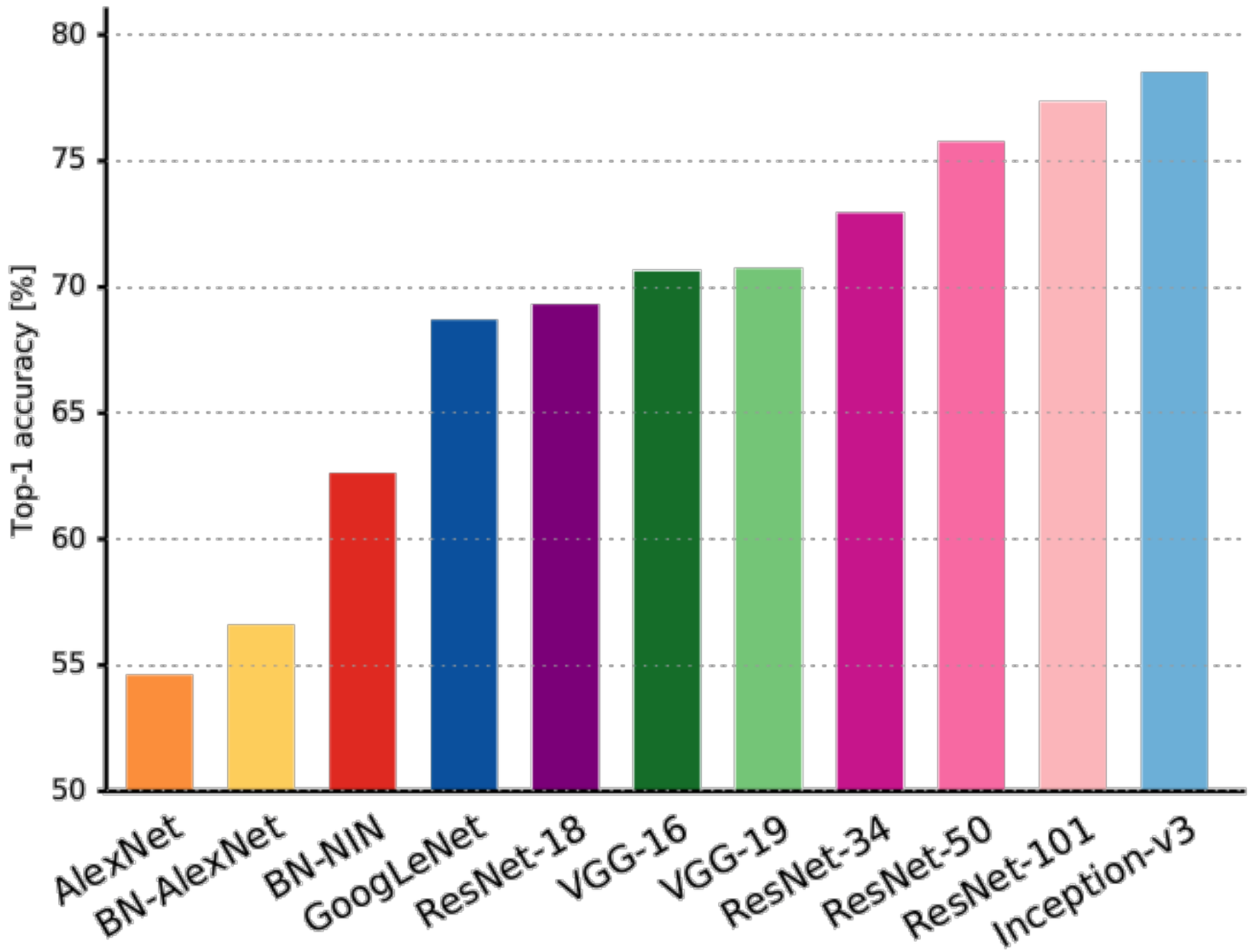
ZF Net: The **ILSVRC** 2013 winner

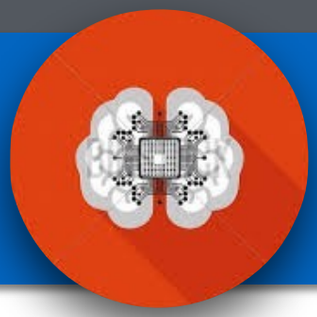
GoogLeNet: The **ILSVRC** 2014 winner

VGGNet: The runner-up in **ILSVRC** 2014

ResNet: The winner of **ILSVRC** 2015

Redes Treinadas

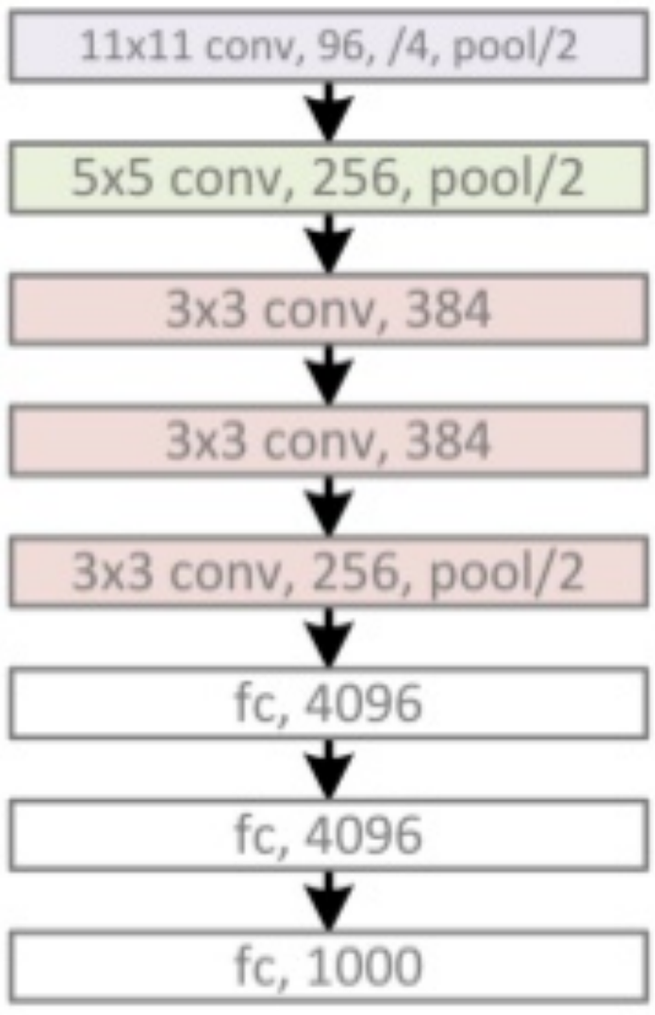




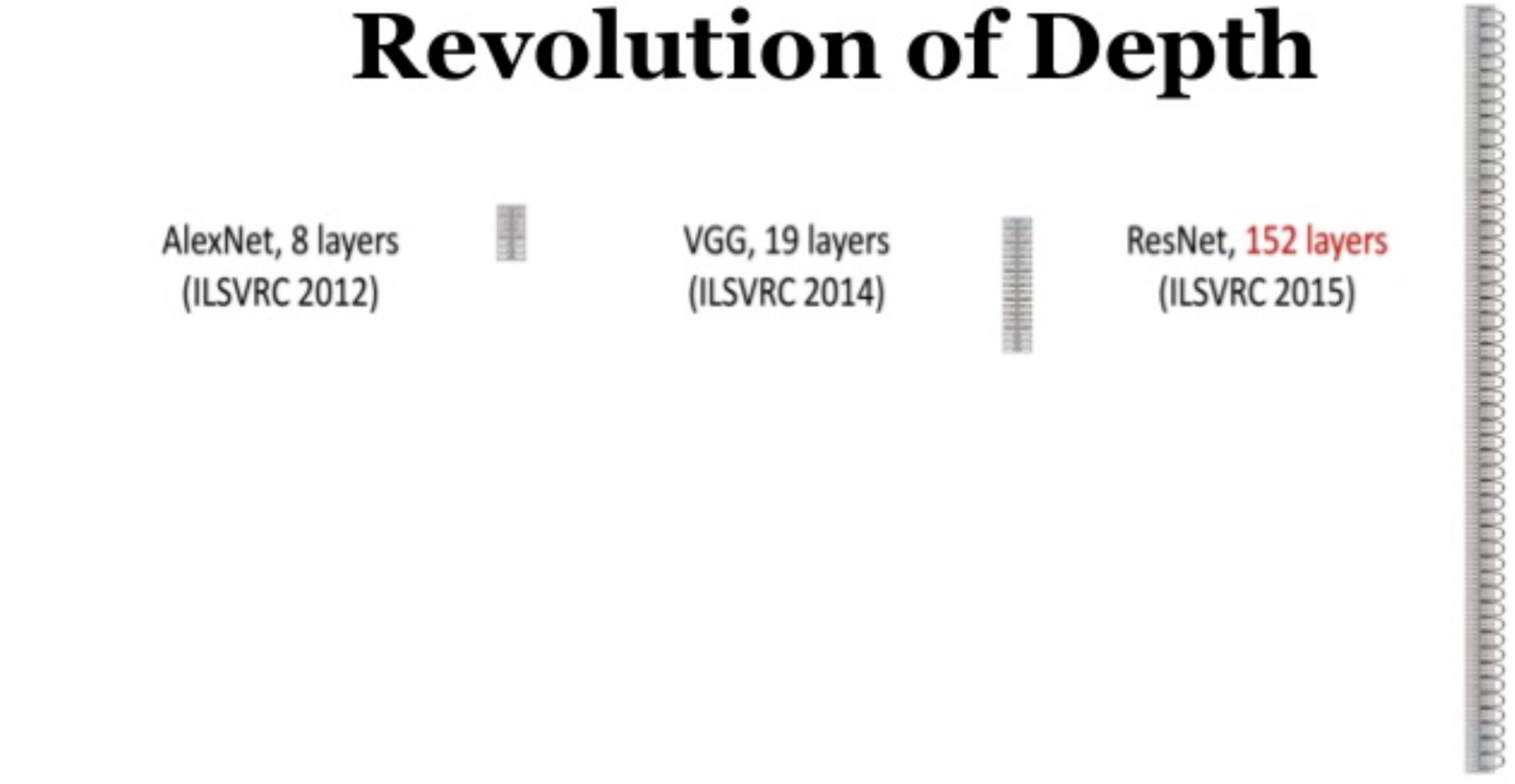
Redes Treinadas

AlexNet

AlexNet, 8 layers
(ILSVRC 2012)



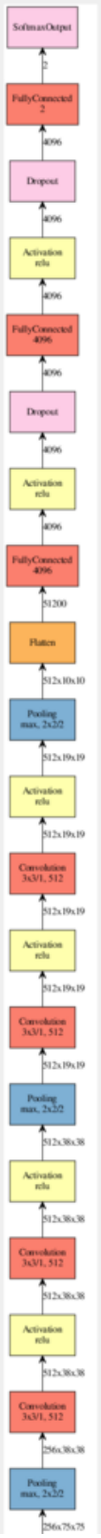
Redes Treinadas



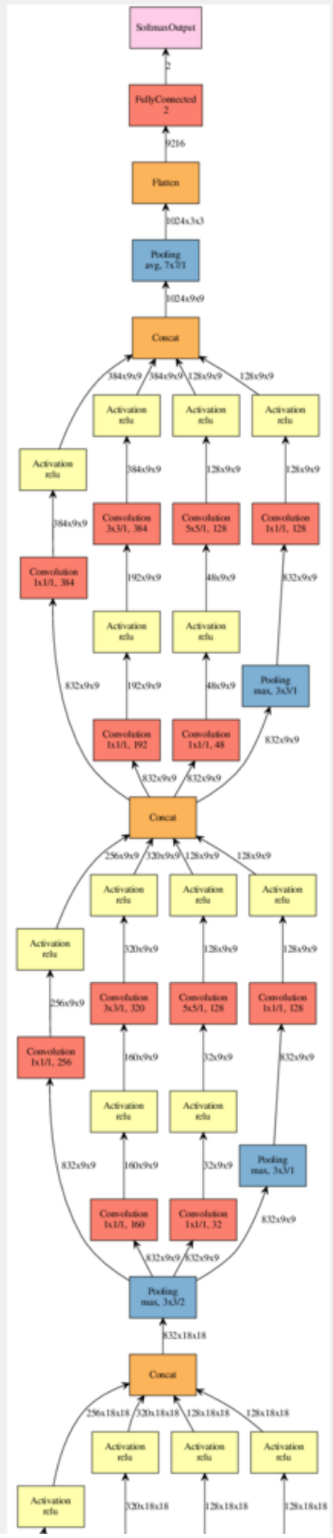
LeNet

AlexNet

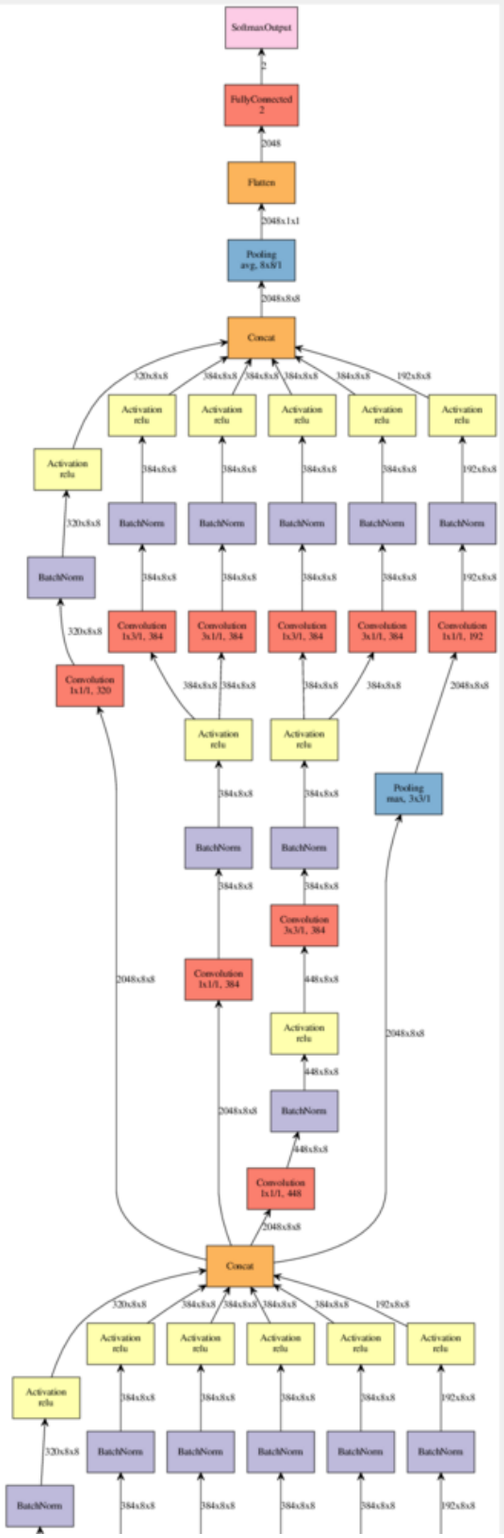
VGG



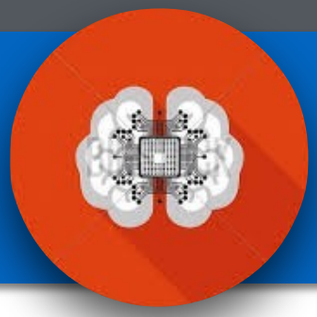
GoogLeNet



Inception V3



Inception BN



Datasets

Datasets Benchmark

MNIST Handwritten digits – 60000 Training + 10000 Test Data

Google House Numbers from street view - 600,000 digit images

CIFAR-10 60000 32x32 colour images in 10 classes

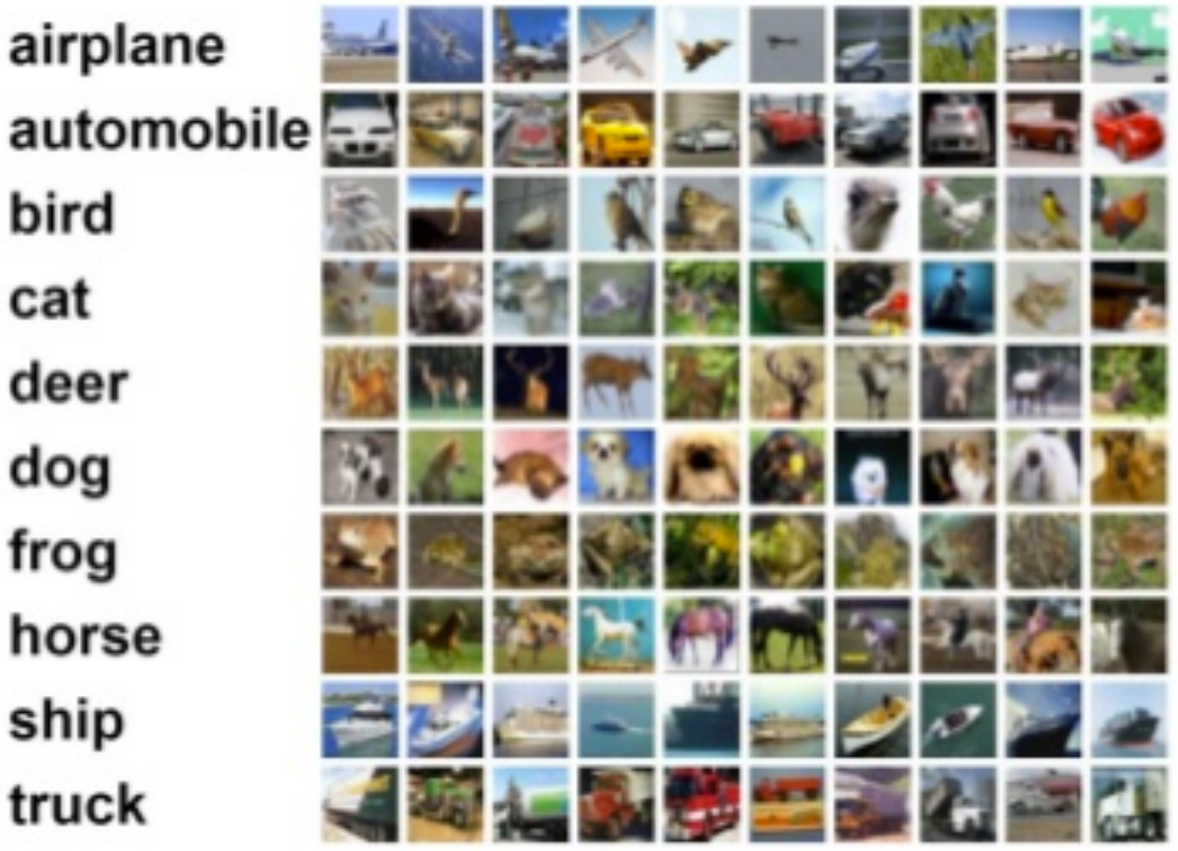
IMAGENET >150 GB

Tiny Images 80 Million tiny images

Flickr Data 100 Million Yahoo dataset

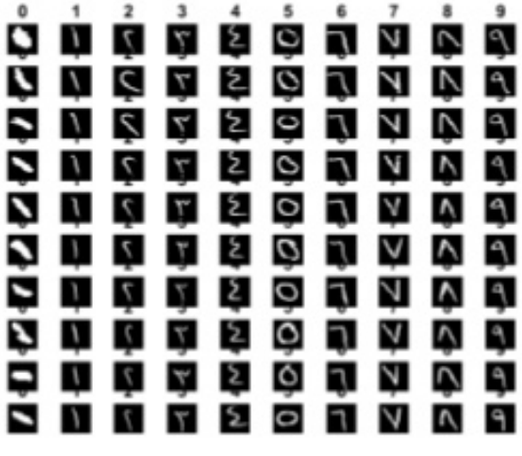
MNIST Dataset

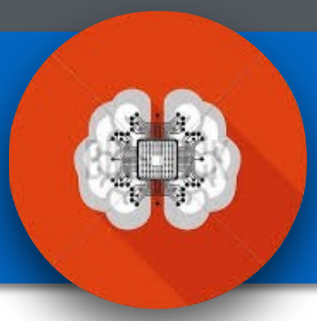
CIFAR-10 Dataset



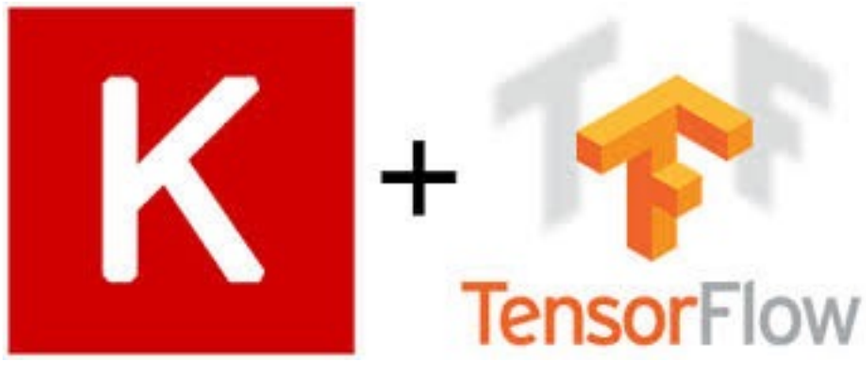
Arabic Handwritten Digits Database

- By Sherif Abdelazeem, Ezzat El-Sherif
- 70,000 digits





Implementação no Keras



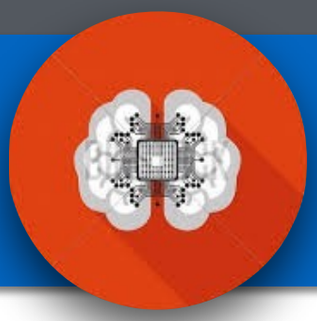

```
model = Sequential()

model.add(Convolution2D(12, 5, 5, activation = 'relu', input_shape=in_shape,
init='he_normal'))

model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Convolution2D(25, 5, 5, activation = 'relu', init='he_normal'))

model.add(MaxPooling2D(pool_size=(2, 2)))
```



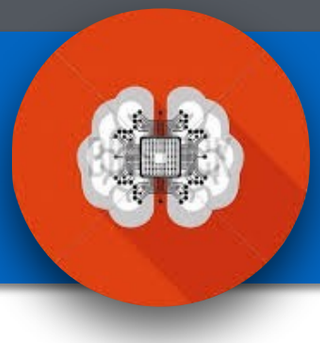
Implementação no TensorFlow

```
# Convolution Layer with 32 filters and a kernel size of 5
conv1 = tf.layers.conv2d(x, 32, 5, activation=tf.nn.relu)
# Max Pooling (down-sampling) with strides of 2 and kernel size of 2
conv1 = tf.layers.max_pooling2d(conv1, 2, 2)

# Flatten the data to a 1-D vector for the fully connected layer
fc1 = tf.contrib.layers.flatten(conv2)

# Fully connected layer (in tf contrib folder for now)
fc1 = tf.layers.dense(fc1, 1024)
# Apply Dropout (if is_training is False, dropout is not applied)
fc1 = tf.layers.dropout(fc1, rate=dropout, training=is_training)

# Output layer, class prediction
out = tf.layers.dense(fc1, n_classes)
```



Exemplos

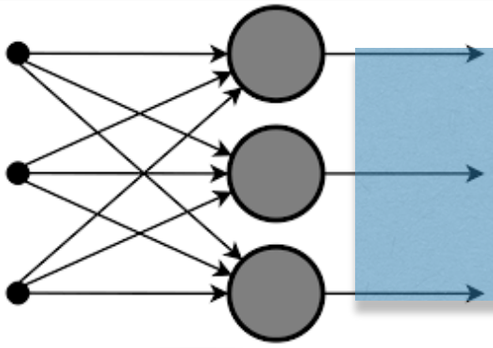
https://github.com/hiteshvaidya/Star-wars-classifier

https://github.com/rdcolema/keras-image-classification/blob/master/cats_n_dogs_BN.ipynb

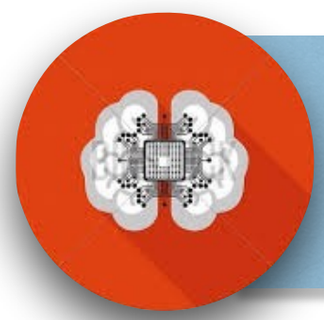
https://github.com/rajshah4/image_keras/blob/master/notebook.ipynb

CNN RNN GAN

Agenda

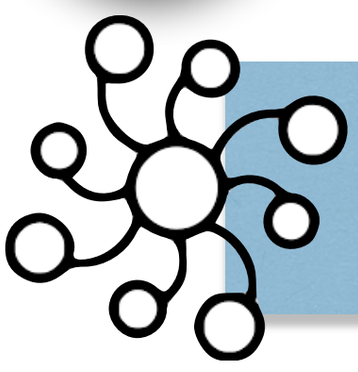


Introdução a Deep Learning



Redes Neurais Convolucionais

Recurrent Neural Networks



Generative Adversarial Networks

Recurrent Neural Networks

ANALISANDO TEXTO

Passados quase três meses do final dos Jogos, o Comitê **Rio-2016** ainda deve reembolso a **8.000 torcedores** que utilizaram sua plataforma online para revender ingressos.

A entidade reduziu o contingente de consumidores a quem devia pagamentos, que chegou a 140 mil em 19 de outubro, data até a qual prometeu quitar os débitos. Mas ainda não deu fim ao problema.

A entidade afirmou que tem dificuldades para ressarcir o restante. Alega problemas para encontrar os credores e inconsistência nos dados bancários fornecidos —muitos depósitos não foram completados.

De acordo com o **comitê**, 3.500 pessoas foram procuradas mas não responderam às mensagens eletrônicas, 2.500 até deram retorno, porém as informações repassadas continham algum erro e 2.000 devem receber o reembolso até esta segunda (12), após terem dados checados.

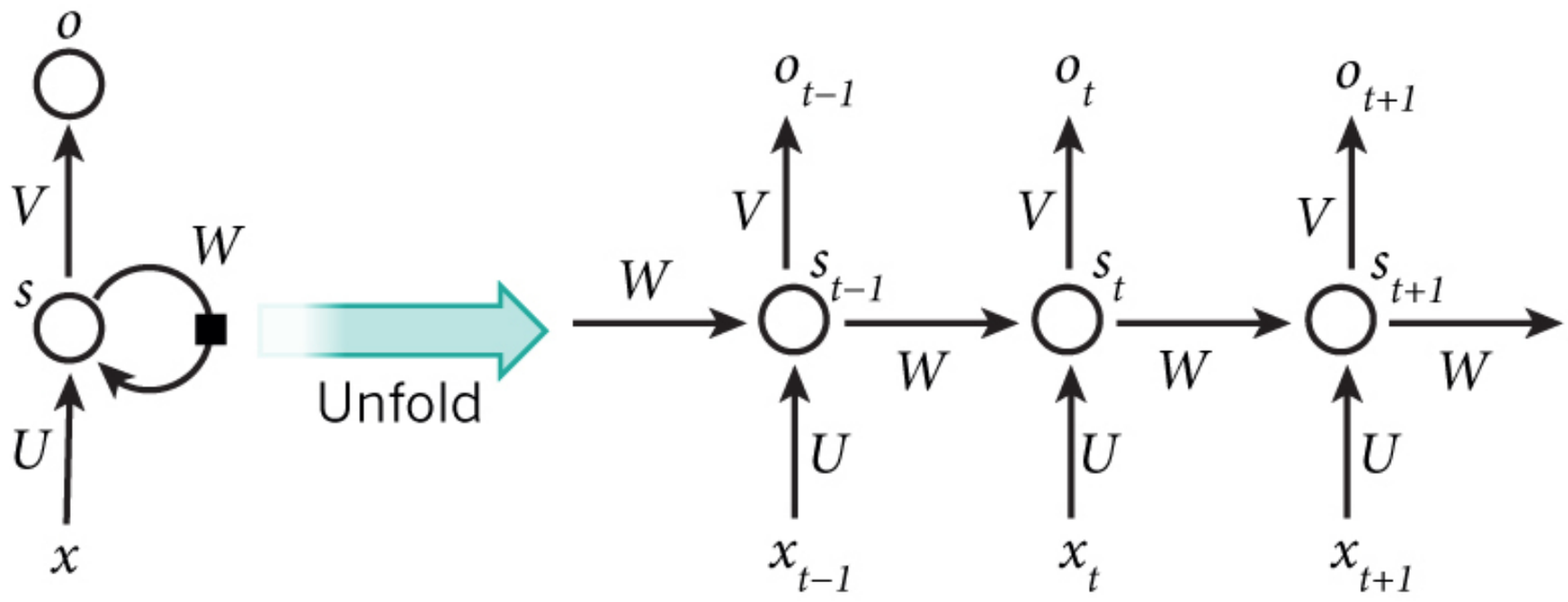
Uma mutação aparentemente insignificante no **DNA** dos ancestrais da humanidade pode ter contribuído para que nosso cérebro alcançasse o tamanho descomunal que tem hoje (três vezes maior que o dos grandes macacos).

Bastou inserir o gene que contém essa mutação em fetos de camundongo para que dobrasse o número de células que dão origem aos neurônios do córtex, a área cerebral mais "nobre".

A **pesquisa,** conduzida por cientistas do **Instituto** Max Planck (Alemanha), é um dos primeiros frutos da tentativa de usar o genoma para entender como a evolução humana se desenrolou. Por enquanto, isso não tem sido fácil –tanto que o gene analisado pelos pesquisadores no novo estudo, designado pela indigesta sigla ARHGAP11B, é o único específico da linhagem humana a ser associado com a proliferação das tais células do córtex cerebral.

Recurrent Neural Networks

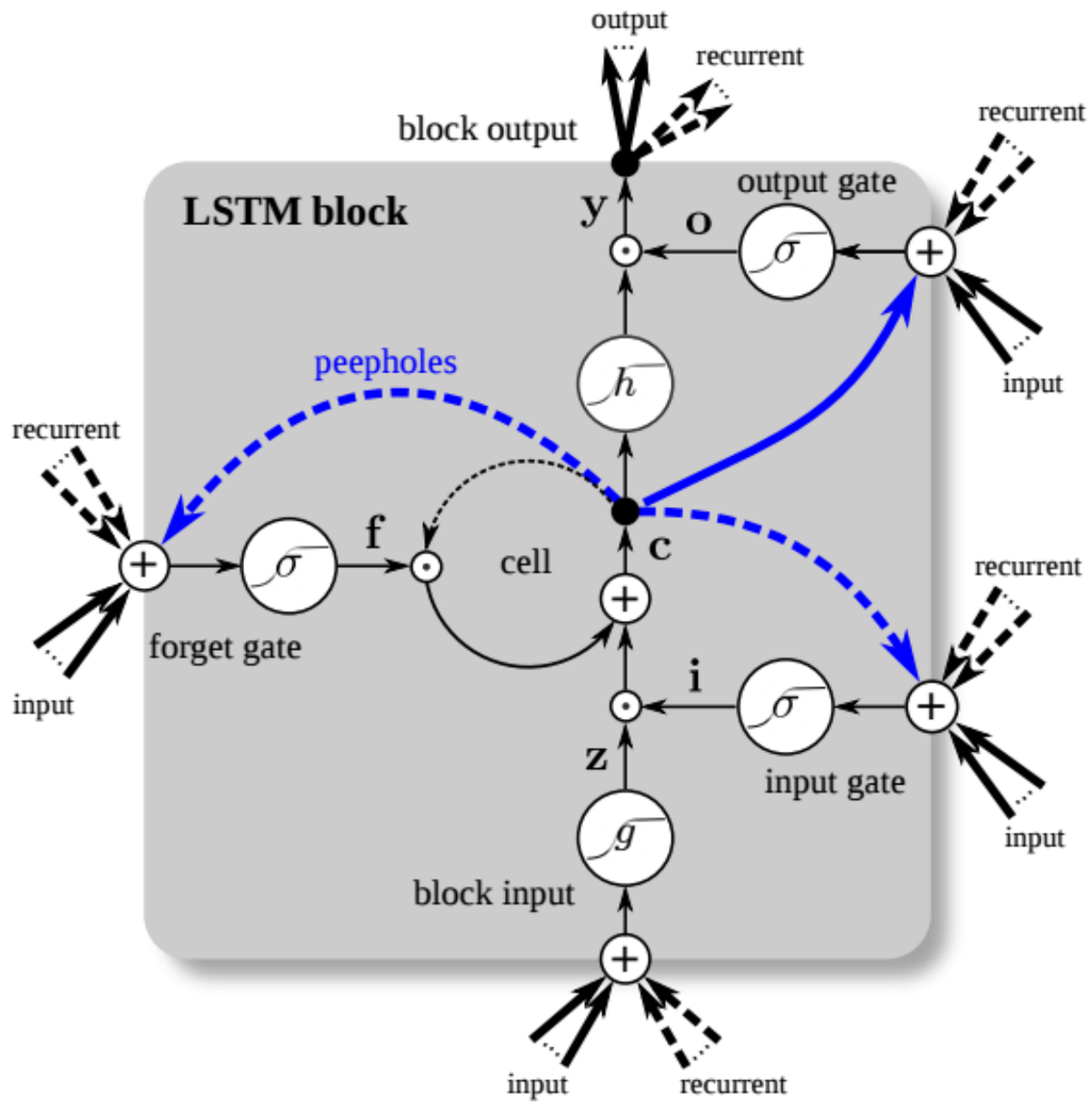
Rnns por trás da idéia é fazer uso da informação sequencial. Em uma rede neural tradicional assumimos que todas as entradas (e saídas) são independentes um do outro. Mas para muitas tarefas que é uma idéia muito ruim.



Longa memória de curto prazo (LSTM)

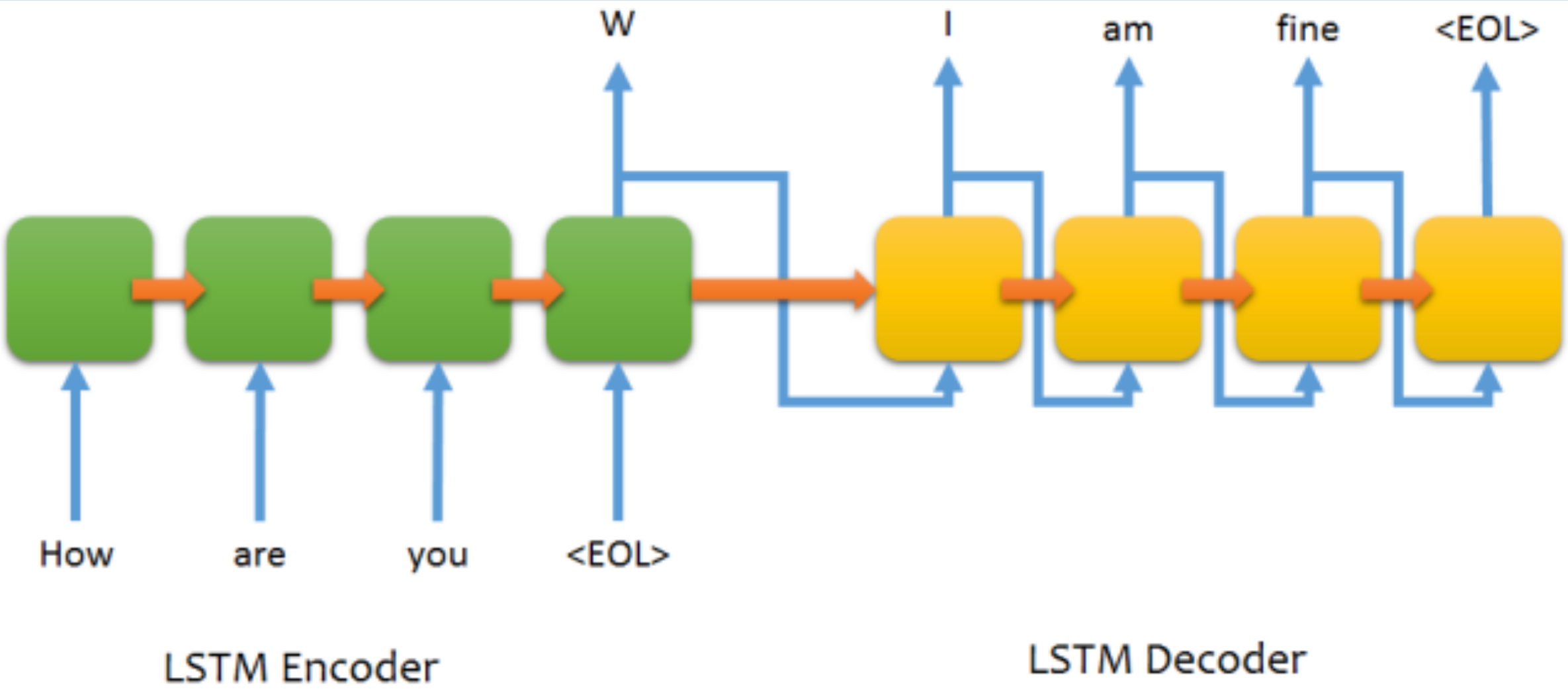
- **Longa memória de curto prazo (LSTM) é uma arquitetura de rede neural recorrente (RNN) que lembra os valores em intervalos arbitrários.**

- **Valores armazenados não são modificados à medida que prossegue aprendizagem.**



Longa memória de curto prazo (LSTM)

Um modelo básico sequência-a-sequência, tal como foi introduzido em Cho et al, 2014 (pdf), consiste de duas rede neural recorrente (rnns): um codificador que processa a entrada e um descodificador que gera a saída.

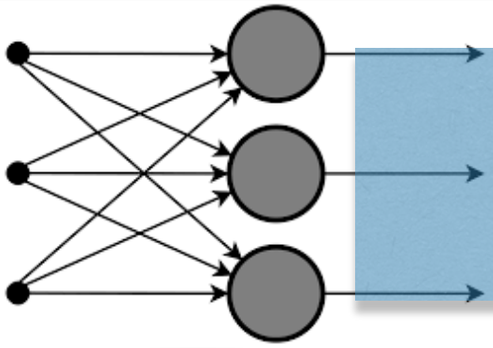


Exemplos

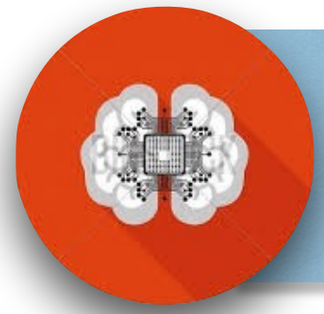
https://github.com/RyanCCollins/deep-learning-nd

CNN RNN GAN

Agenda

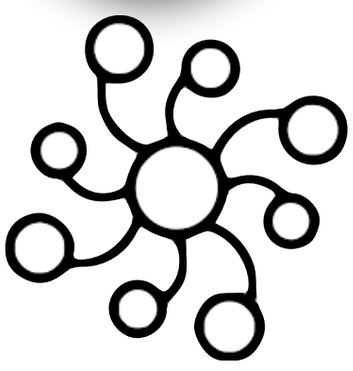


Introdução a Deep Learning



Redes Neurais Convolucionais

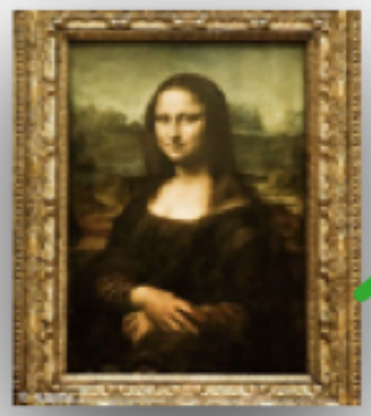
Recurrent Neural Networks



Generative Adversarial Networks

Generative Adversarial Network

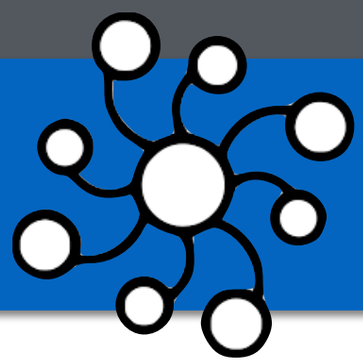
D: Detective



R: Real Data

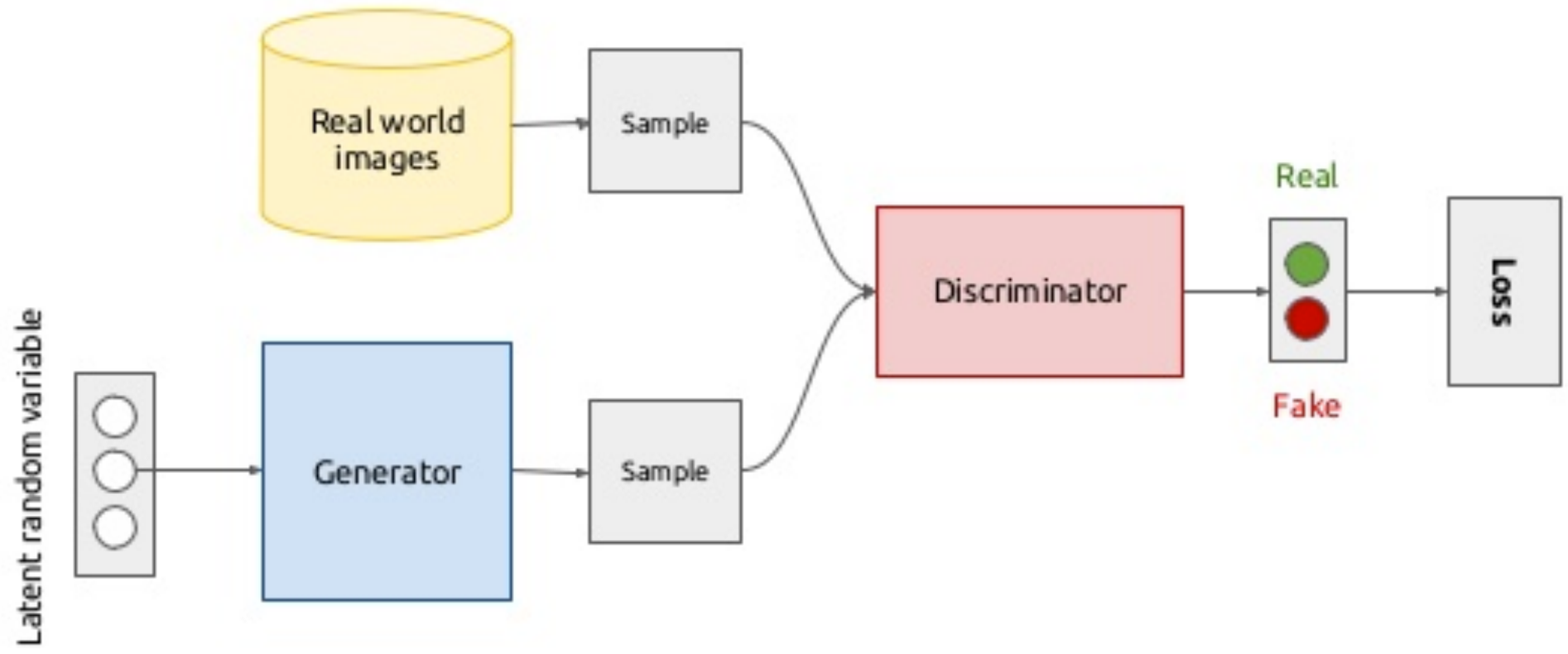
G: Generator (Forger)

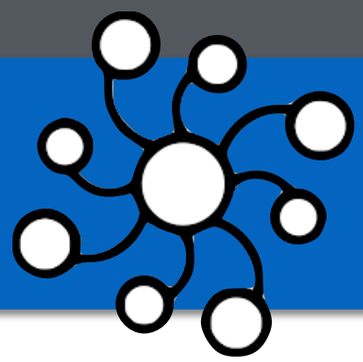
I: Input for Generator



Generative Adversarial Network

Generative adversarial networks (conceptual)





Discriminator

Input

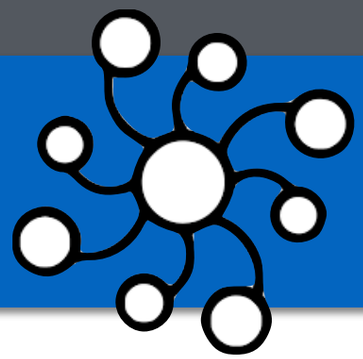
Picture of Money

Deep Convolutional Neural Network

Output

True

Whether or not the picture contains money



Generator

Input

100 numbers

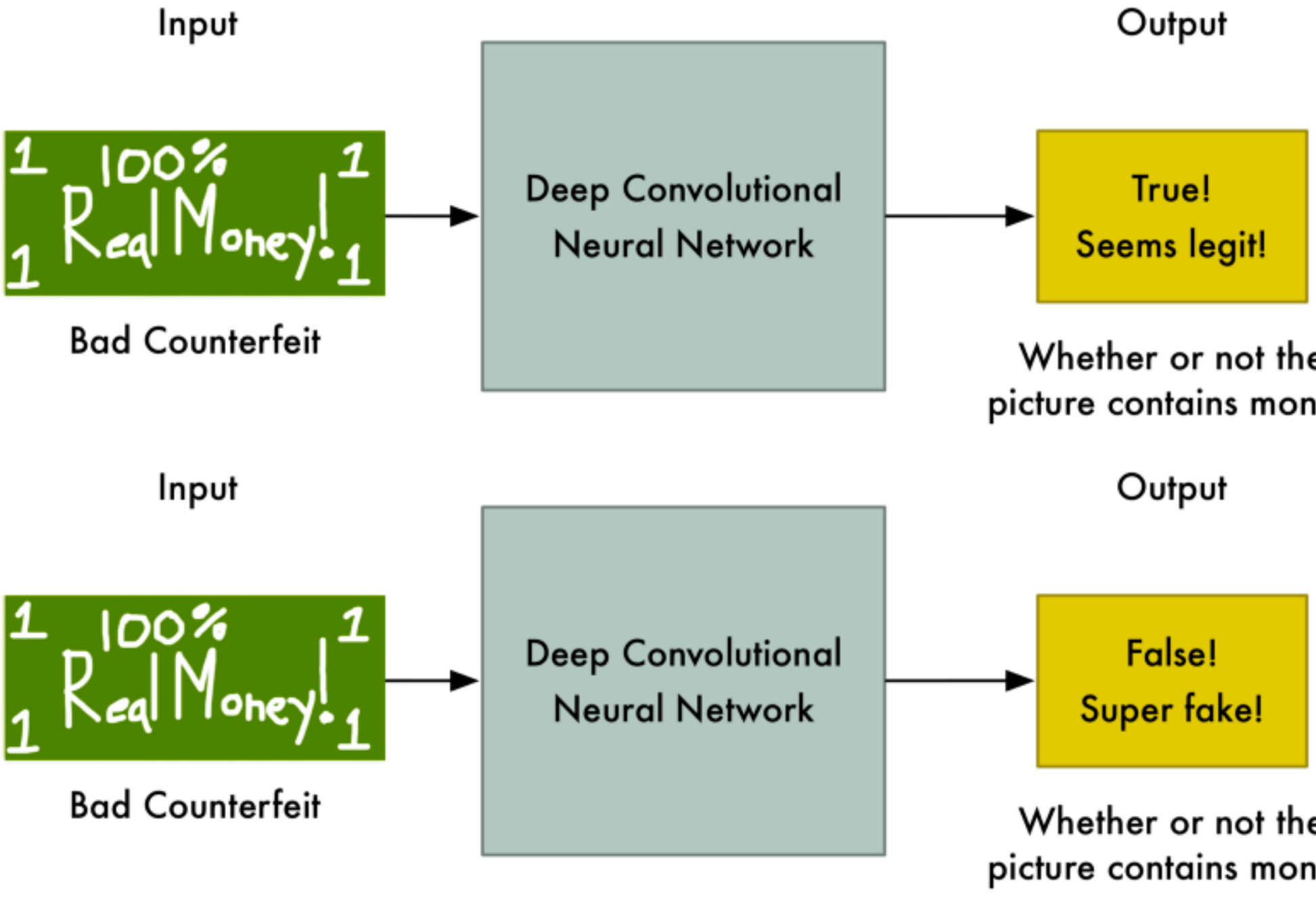
A list of numbers

Reversed Deep Convolutional Neural Network

Output

Picture of Money

Discriminator Evolution



Discriminator Evolution

Input

100 numbers

A list of numbers

Reversed Deep Convolutional Neural Network

Output

Bad Counterfeit

Input

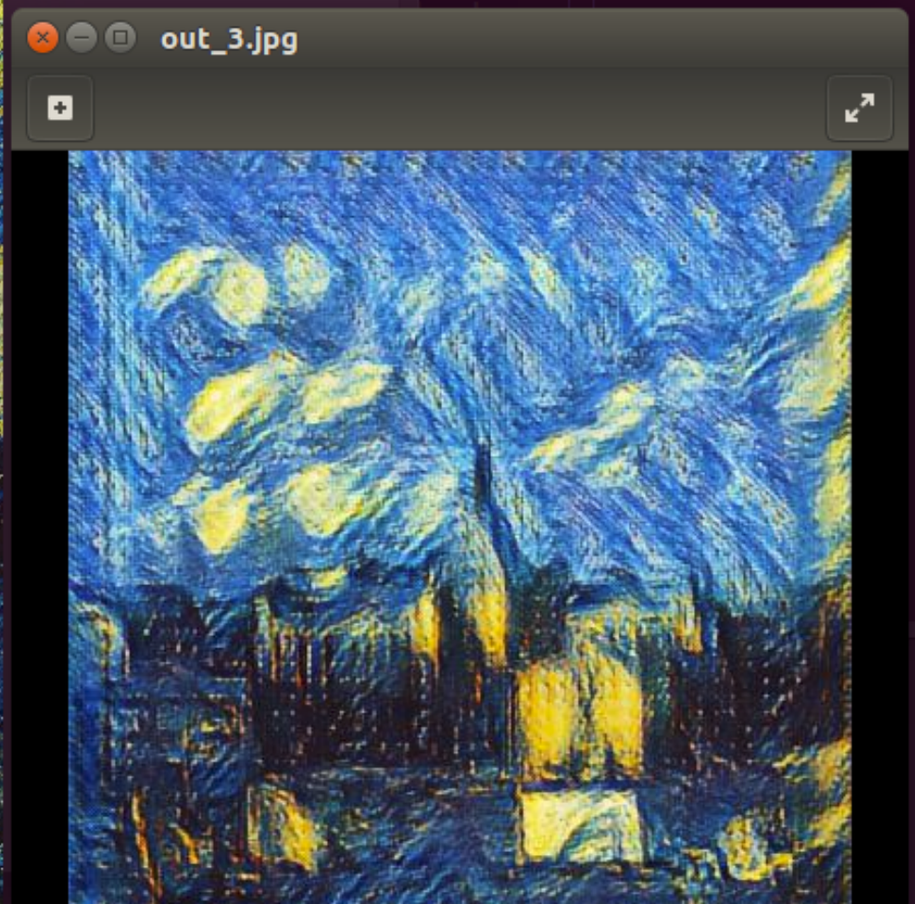
100 numbers

A list of numbers

Reversed Deep Convolutional Neural Network

Output

Slightly Better Counterfeit


out_3.jpg

OOL

line

eraser

INPUT

pix2pix

process

OUTPUT

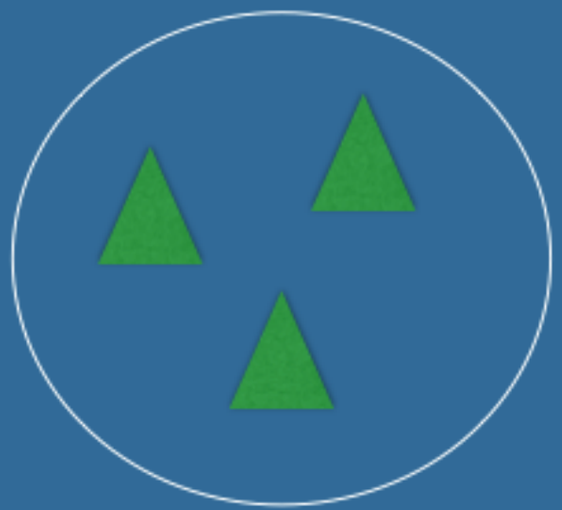
undo

clear

save

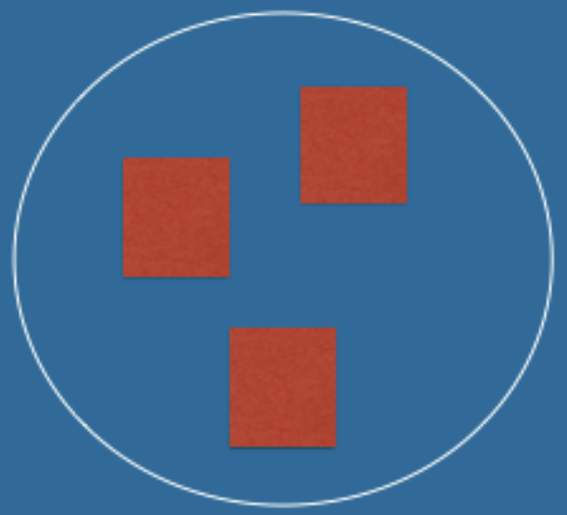
Transfer Learning

Traditional ML



Task / domain A

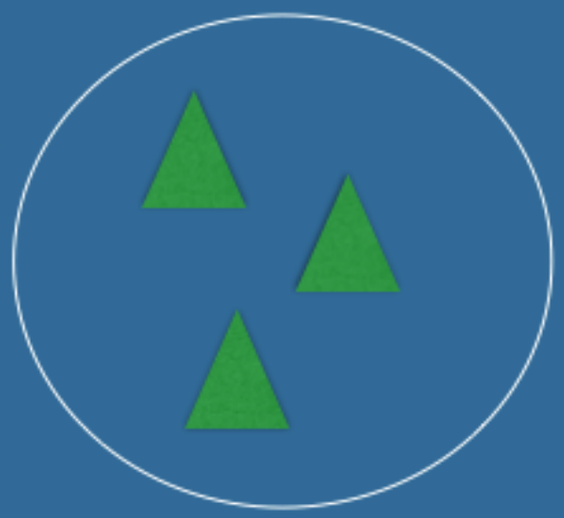
Task / domain B

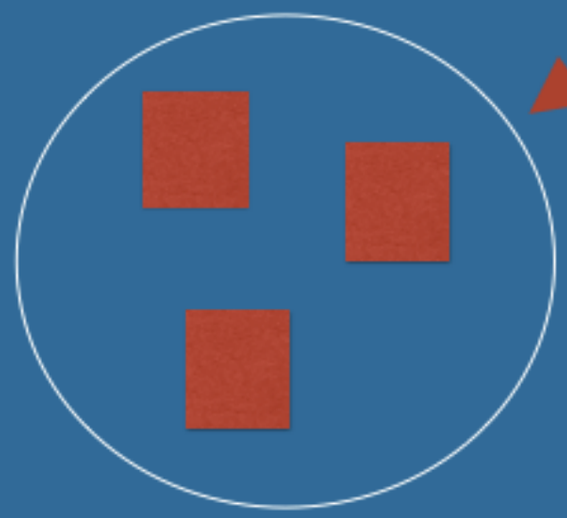


Training and evaluation on the same task or domain.

Model A

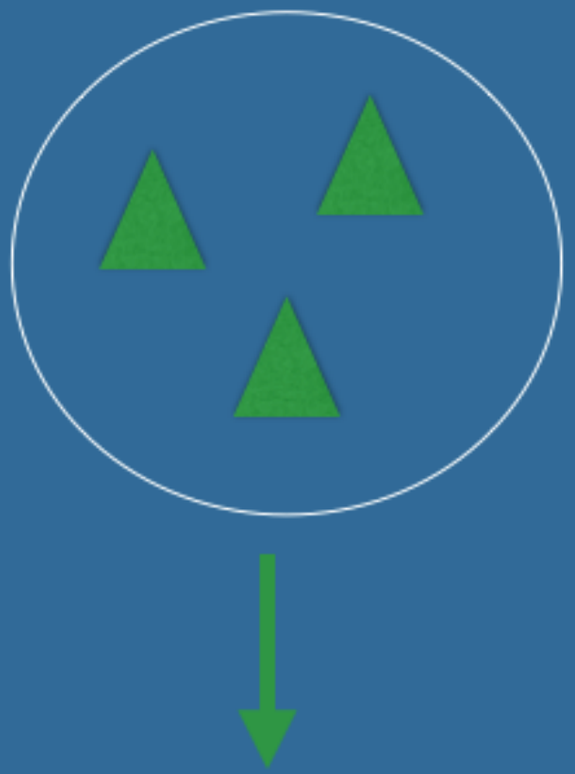
Model B



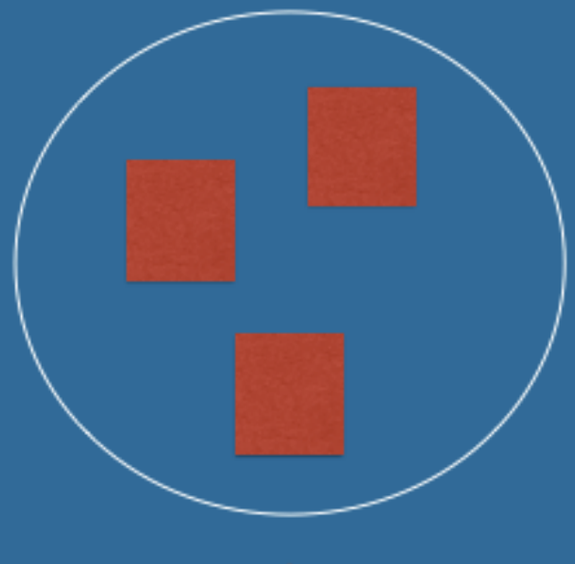


Transfer Learning

Transfer learning



Source task / domain



Target task / domain

Storing knowledge gained solving one problem and applying it to a different but related problem.

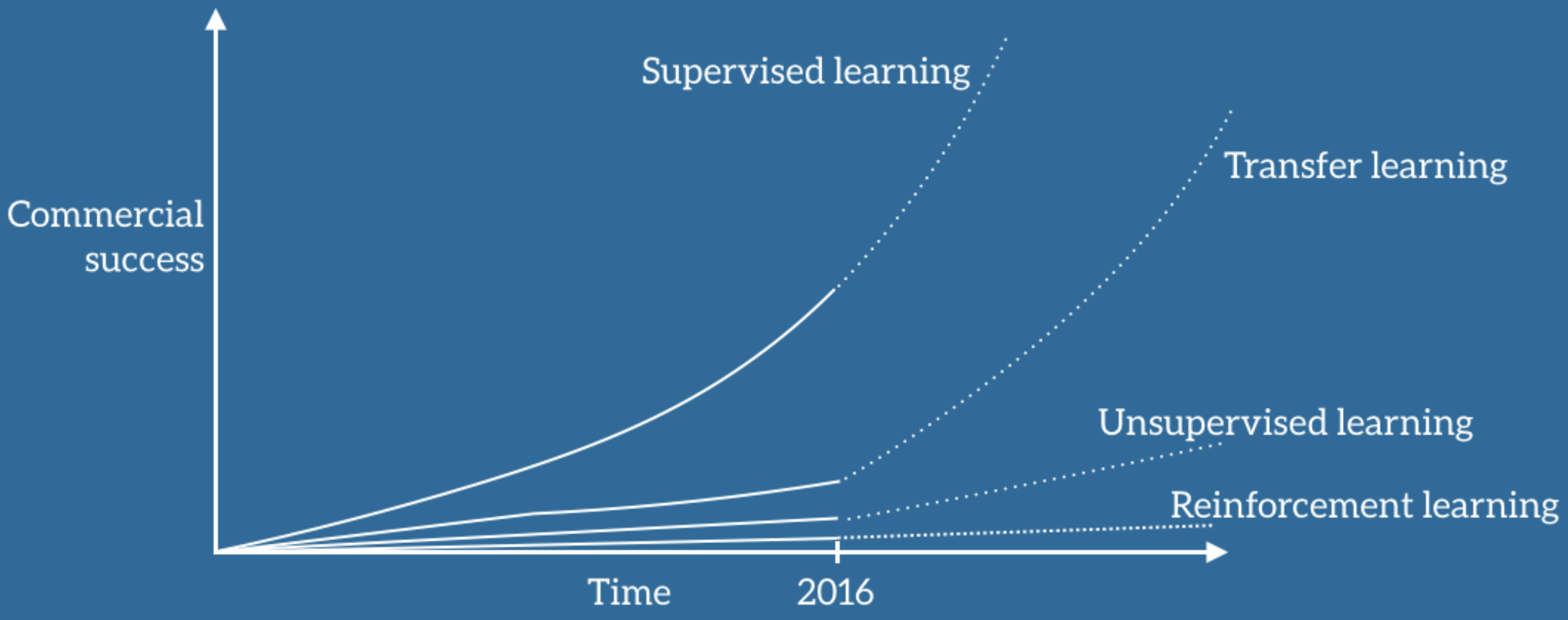
Model

Model

Knowledge

Transfer Learning

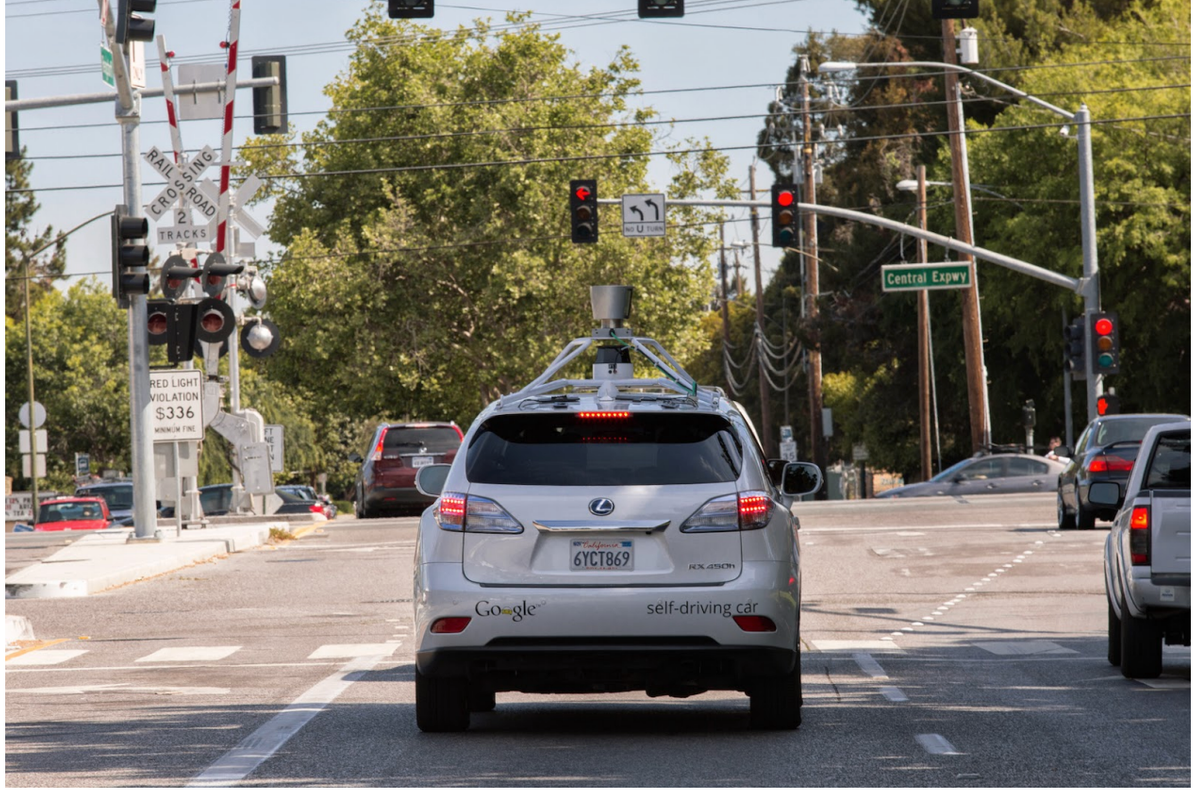
Drivers of ML success in industry

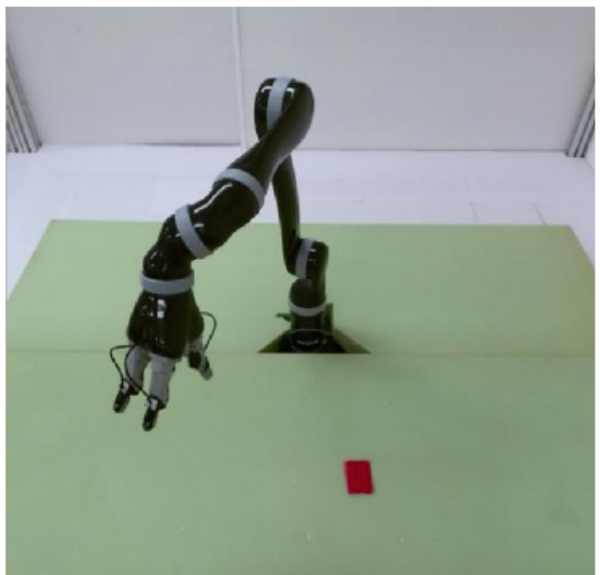


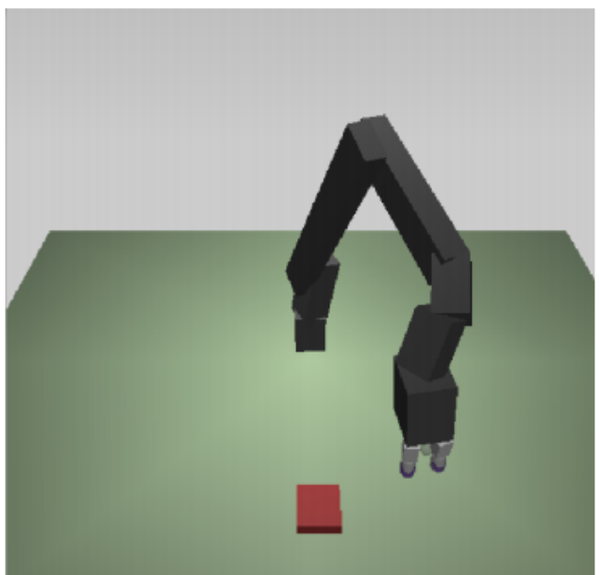
- Andrew Ng, NIPS 2016 tutorial

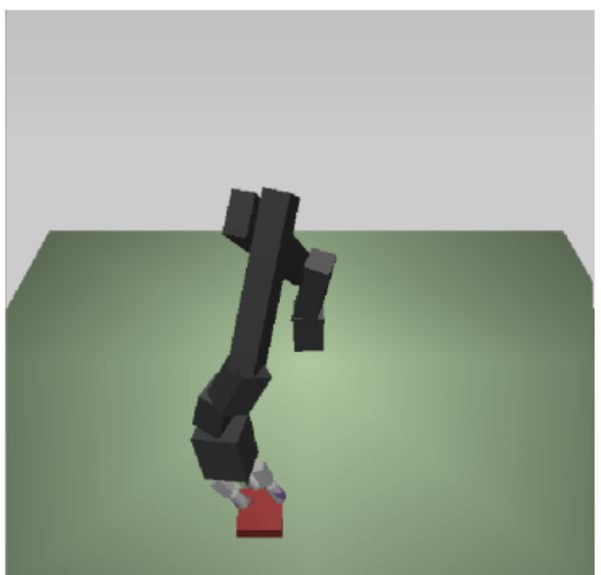
Domain 1

Domain 2

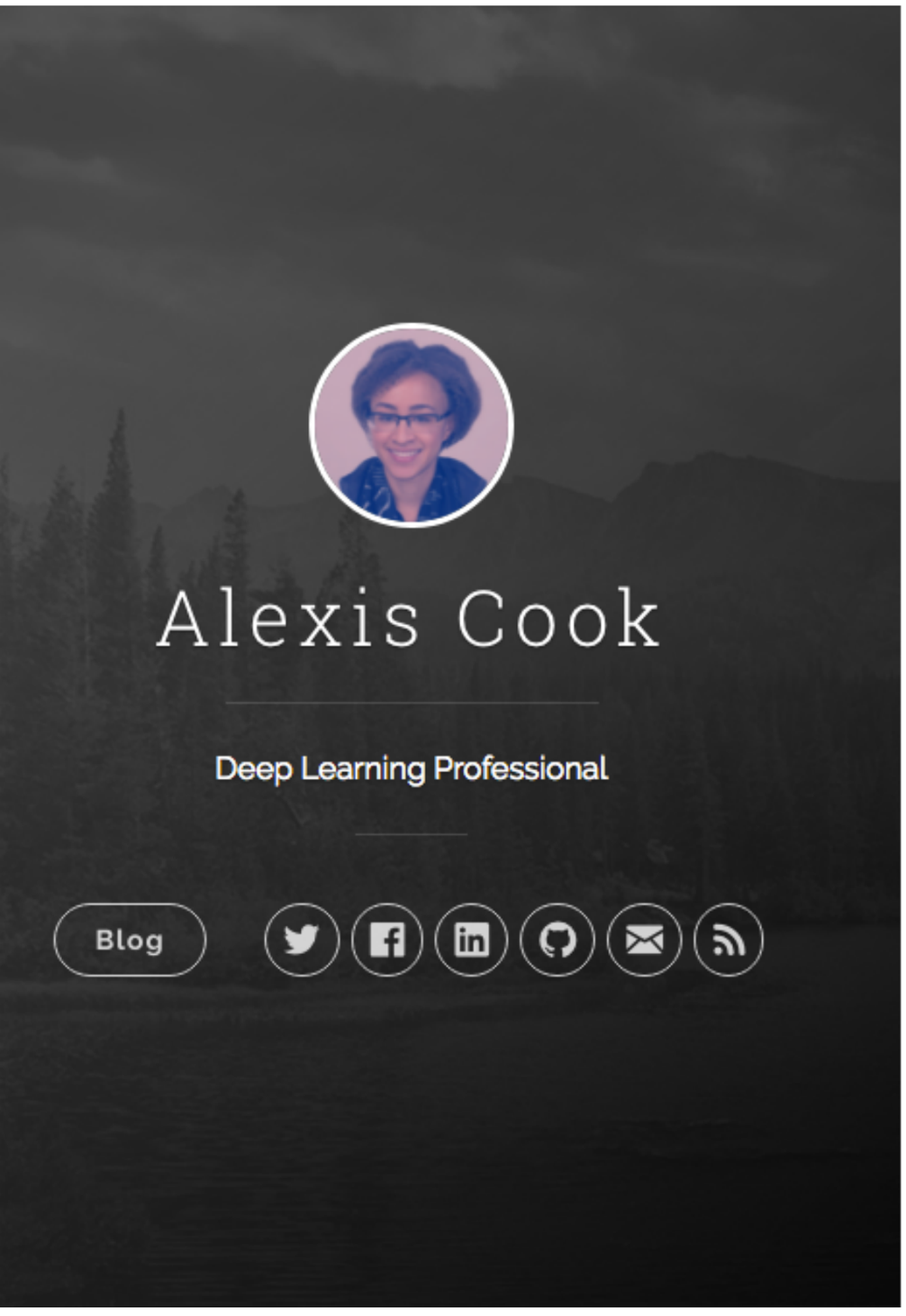








https://alexisbcook.github.io/2017/using-transfer-learning-to-classify-images-with-keras/

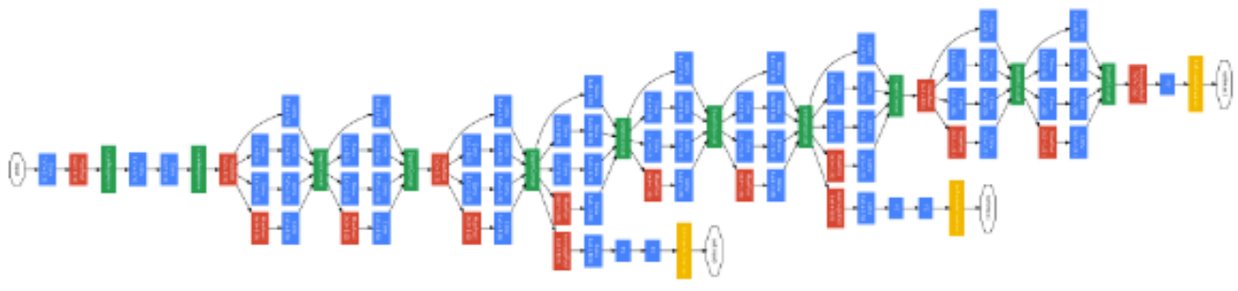



```
from keras.datasets import cifar10
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
```

Extracting the InceptionV3 Bottleneck Features

Instead of building a CNN from scratch, I used **transfer learning** to leverage a pre-trained CNN that has demonstrated state-of-the-art performance in object classification tasks.

Keras makes it very easy to access several pre-trained CNN architectures. I decided to use the InceptionV3 architecture.



After importing the necessary Python class, it's only one line of code to get the model, along with the pre-trained weights.

```
from keras.applications.inception_v3 import InceptionV3
base_model = InceptionV3(weights='imagenet', include_top=True)
```

The pre-trained InceptionV3 architecture is stored in the variable `base_model`.