

Workshop Visão Computacional

1ª semana

- *Introdução à linguagem Python com Anaconda*
- *Prática: Regressão KNN, Random Forrest, Clustering e SVM com Python*

José Humberto Cruvinel

Contato: jose.junior@prof.unibh.br

<https://www.facebook.com/jhcruvinel>

Apresentação do professor

- **José Humberto Cruvinel**
- Formação
 - Graduado em Engenharia Elétrica / Ênfase em Computação (UFMG)
 - Especialista em Análise de Sistemas de Informação (PUCMG)
 - Mestre em Administração Pública / Gestão da Informação (FJP)
 - MBA em Gerenciamento de Projeto de TI / FGV
- Atuação Profissional
 - Analista no Serpro
 - Professor no UNI-BH
- Certificações Profissionais
 - PMP, CFPS, CTFL, ITIL v3, COBIT 4.1, MCP, Scrum Master, DB2 10, Rational Team Concert v3
- Contato e sites:
 - E-mail institucional: jose.junior@prof.unibh.br
 - Facebook / Messenger: <https://www.facebook.com/jhcruvinel>
 - Curriculum Lattes: <http://lattes.cnpq.br/0955831456676522>

Agenda – 1º sábado



1. Apresentação

Programação do Workshop

1º sábado

- Introdução à Redes Neurais e Visão Computacional
- Prática: Introdução ao TensorFlow, MLP e CNN

2º sábado

- Prática: Modelos CNN, Keras e Classificação de objetos
- Detecção de Objetos com Tensorflow

Ferramentas de apoio



UNIBH Workshop Ciência de Dados

- Slides
- Laboratórios

Suporte aos exercícios

Ferramentas que vamos utilizar



Anaconda - <https://www.anaconda.com>

- É uma plataforma que facilita a gestão de ambientes virtuais
- Possui as seguintes características:
 - Mantém cada ambiente isolado, com pacotes de versões diferentes
 - Facilita a instalação e configuração de pacotes
 - Aumenta a produtividade no desenvolvimento
 - Muito utilizada por cientistas de dados



Python - <https://www.python.org>

- Python é uma linguagem de programação que possui as seguintes características:
 - Totalmente livre
 - Código interpretado
 - Multiplataforma
 - Orientada a objetos
 - Sintaxe simples
 - Fácil de aprender
 - Fácil leitura e manutenção
 - Possui modo interativo



Jupyter Notebook - <http://jupyter.org>

- Interface interativa para o desenvolvimento com linguagens de programação
- Características:
 - Permite a visualização da execução comando a comando e suas respectivas saídas, incluindo gráficos
 - Permite salvar um notebook para uso posterior
 - Permite compartilhar notebooks via git
 - Bancada de testes ideal para cientistas de dados



HANDS ON



Laboratório 01

Anaconda e Jupyter Notebook

Tensorflow - <https://www.tensorflow.org>

- TensorFlow é um sistema para criação e treinamento de redes neurais baseado em gráficos de fluxo de dados.
- Os nós do gráfico representam operações matemáticas, enquanto as bordas do gráfico representam os arrays de dados multidimensionais (tensores) que fluem entre eles.
- Essa arquitetura flexível permite que você implante computação para uma ou mais CPUs ou GPUs em uma área de trabalho, servidor ou dispositivo móvel sem reescrever o código.



Cursos complementares gratuitos

<https://www.datascienceacademy.com.br/>



- Inscreva-se gratuitamente nos cursos abaixo:

GRATUITO

[Python Fundamentos para Análise de Dados](#)



Aprenda a programar em Python, uma linguagem poderosa usada para criar aplicações de análise de dados.

GRATUITO

[Inteligência Artificial Fundamentos](#)



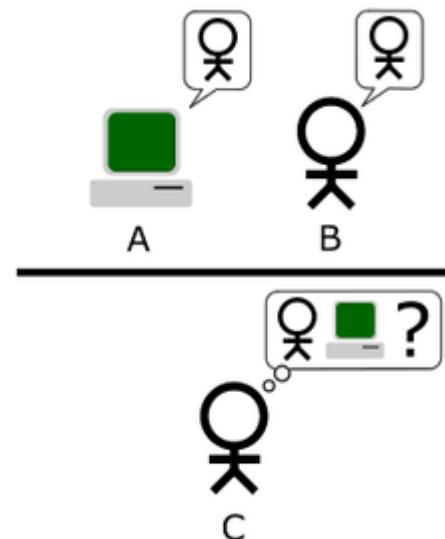
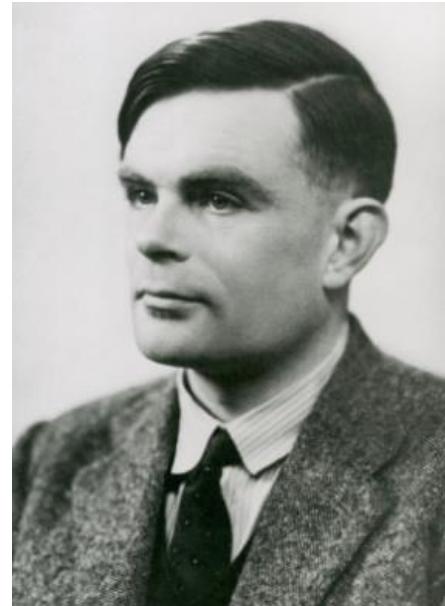
Nosso objetivo com este curso é democratizar os fundamentos de Inteligência Artificial, levando seus principais conceitos a pessoas de todos os níveis e faixas etárias, em todos os cantos do Brasil.

2. Inteligência Artificial

História: Turing

◆ Alan Turing (1950)

- ⊕ Durante a Segunda Guerra Mundial, Turing conseguiu decifrar os códigos alemães criados pela máquina Enigma (ver filme “**O jogo da imitação**”)
- ⊕ Depois da guerra, ele começou a trabalhar na ideia da possibilidade de construir um computador que pensasse. O seu trabalho publicado em 1950, Computing Machinery & Intelligence, foi um dos primeiros trabalhos escritos sobre o assunto → Proposição do “**Teste de Turing**” (Ver filme “Ex Machina”)



Definição formal

➤ I.A. forte:

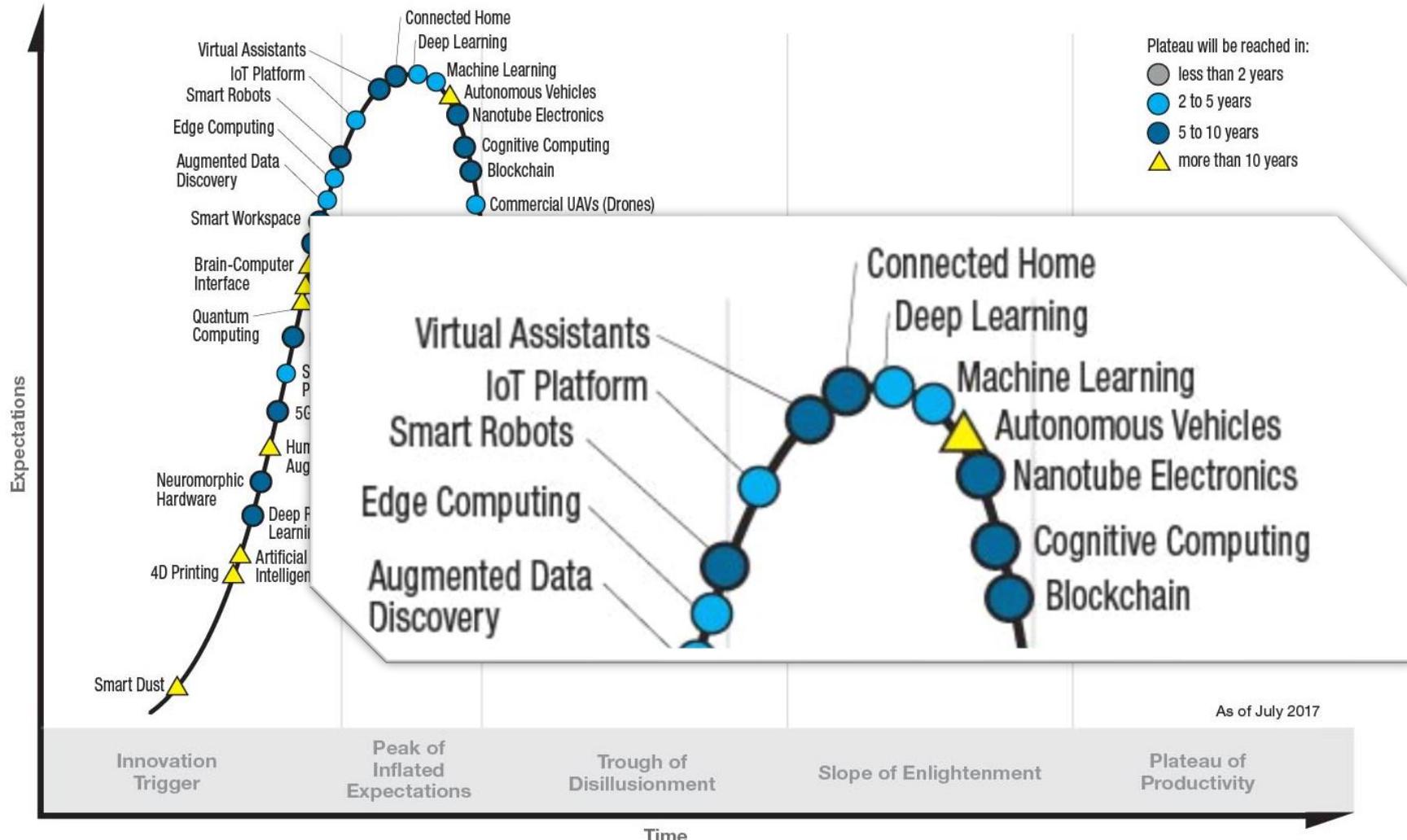
- Os seguidores da IA forte acreditam que, dispondo de um computador com suficiente capacidade de processamento e fornecendo a ele suficiente inteligência, pode-se criar um computador que possa literalmente pensar e ser consciente do mesmo modo que um humano é consciente → **Singularidade**

➤ I.A. fraca:

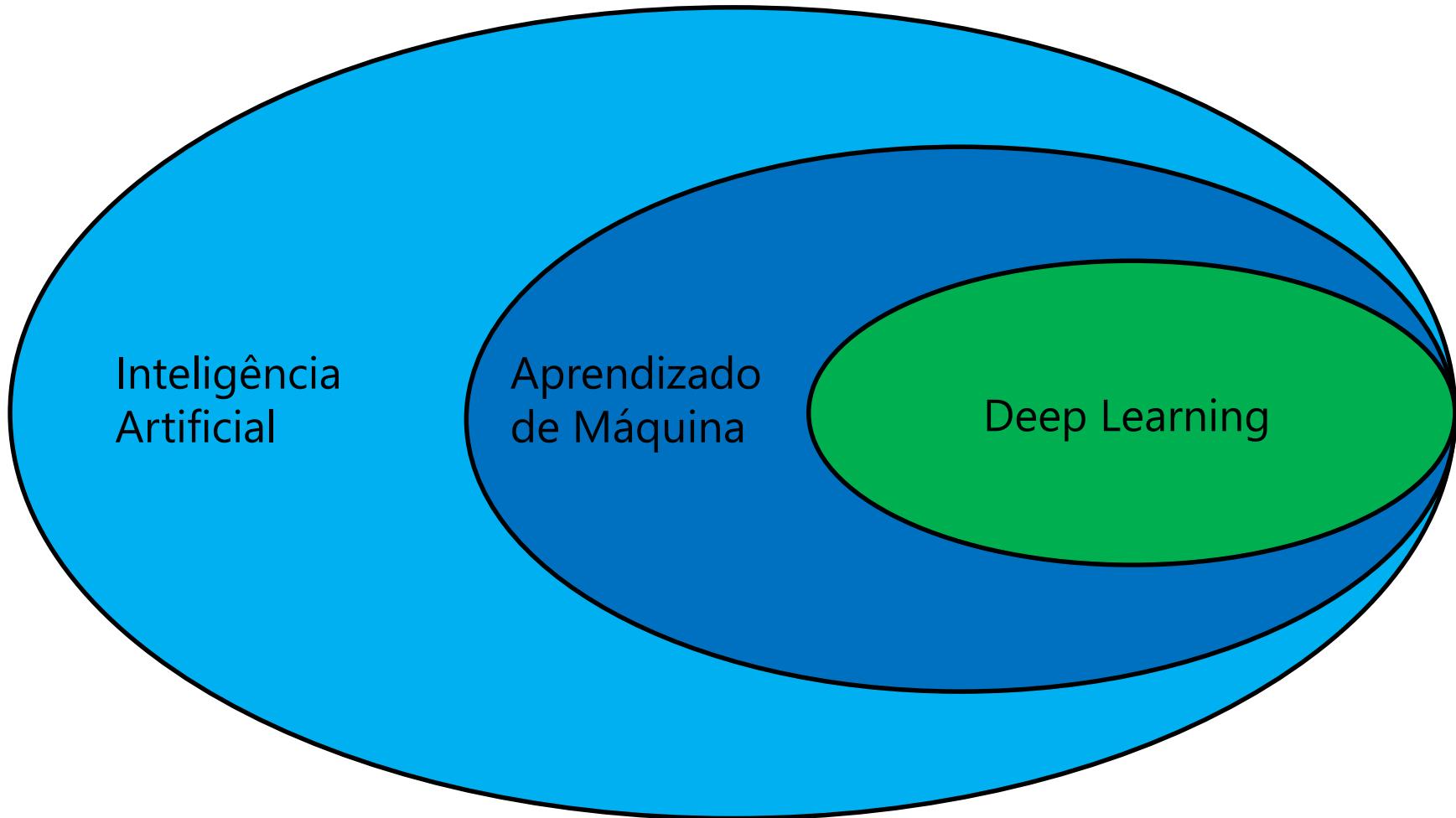
- É simplesmente a visão de que comportamento inteligente pode ser modelado e utilizado por computadores para solucionar problemas complexos → **IA atual**

Hype Cycle da Gartner

Gartner Hype Cycle for Emerging Technologies, 2017



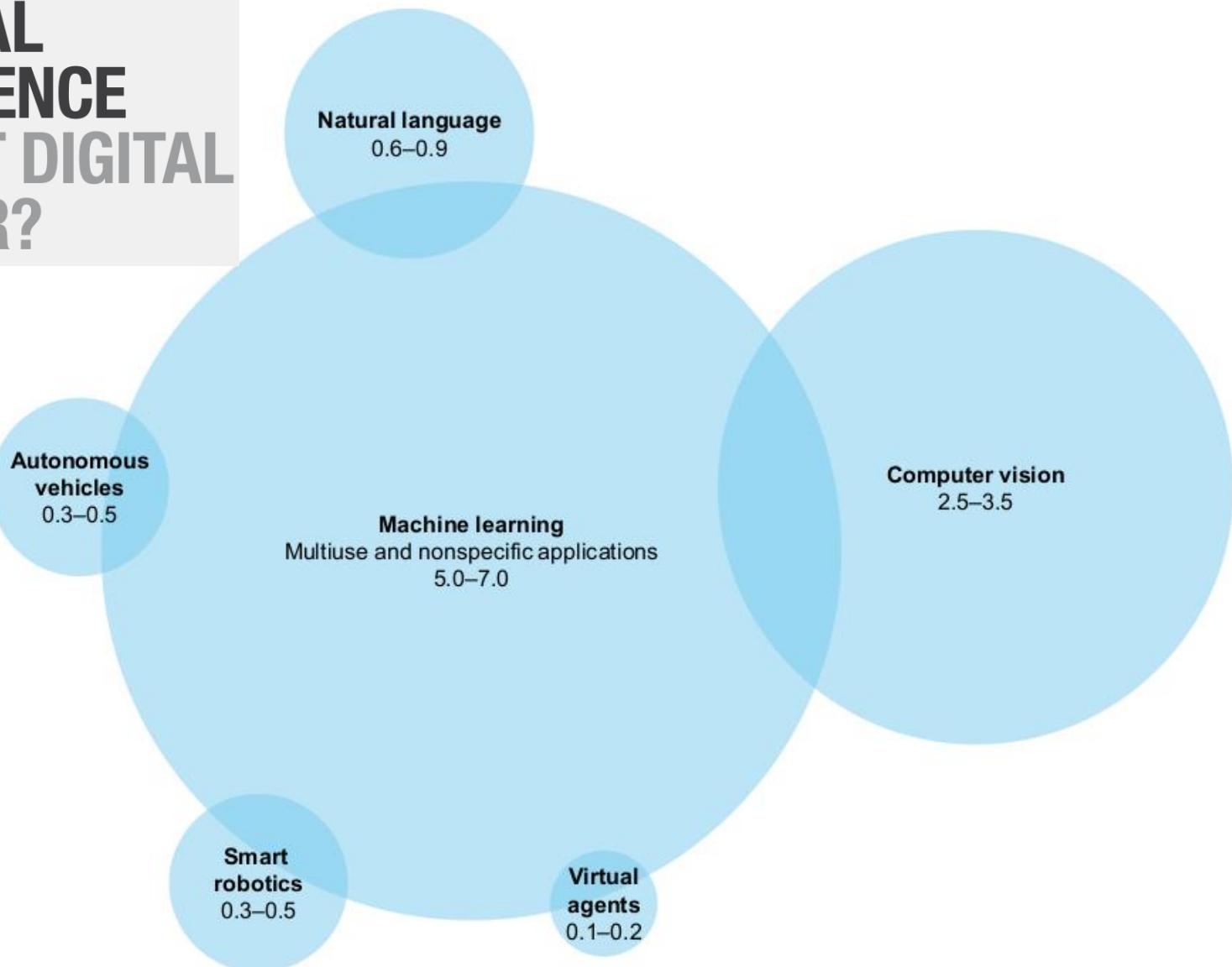
Contextualização



Investimentos em 2016 (U\$ bi)

MCKINSEY GLOBAL INSTITUTE

ARTIFICIAL INTELLIGENCE THE NEXT DIGITAL FRONTIER?



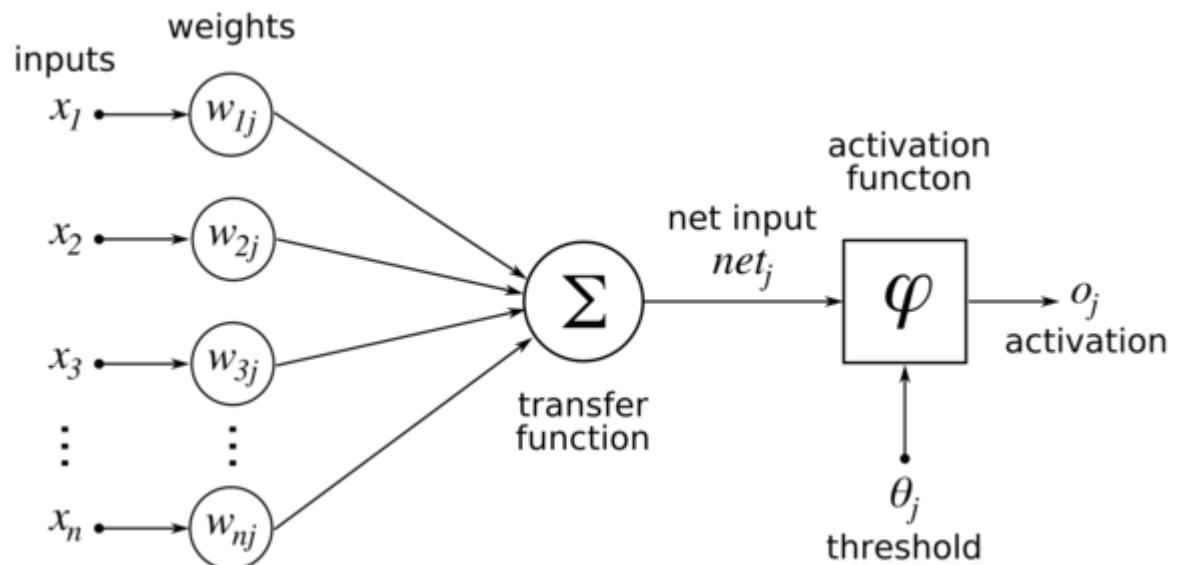
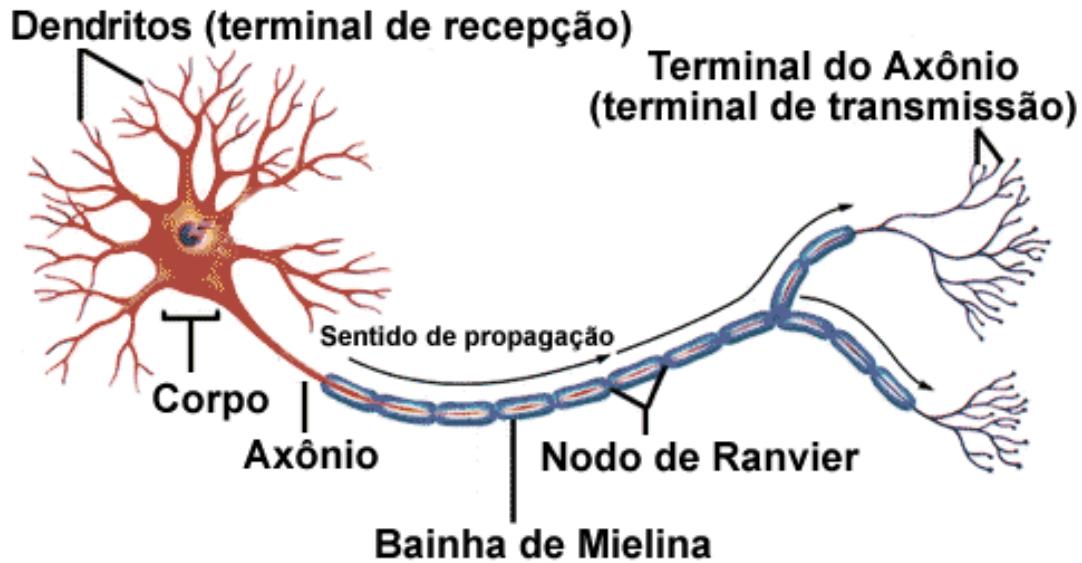
4ª Revolução Tecnológica

- ◆ Novas possibilidades estão surgindo:
 - ⊕ Atividades **preditivas podem ser substituídas** pela máquina (ex: diagnóstico médico e consultoria jurídica)
 - ⊕ **Aumento da acurácia** em atividades que dependem de humanos (ex: reconhecimento facial)
 - ⊕ **Grande volume de informações** podem ser utilizadas no treinamento contínuo de agentes inteligentes (ex: Watson)
 - ⊕ **Atendimento personalizado** de forma automática (chatbots)
 - ⊕ É possível **aumentar o valor agregado** em soluções existentes e criação de novos produtos que facilitam o dia a dia!

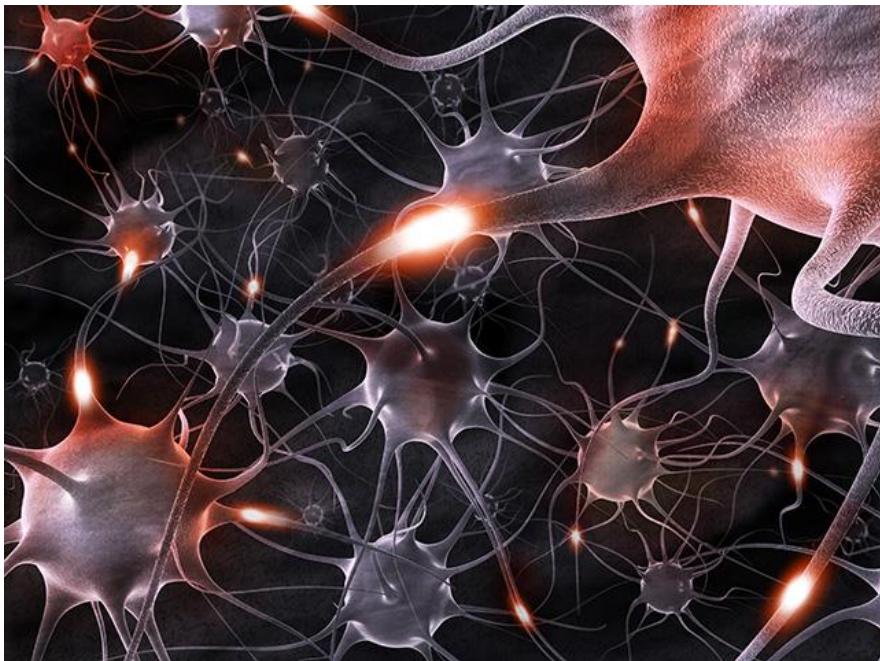
3. Redes Neurais

Neurônio Artificial: Perceptron

- Neurônio biológico
- Neurônio artificial



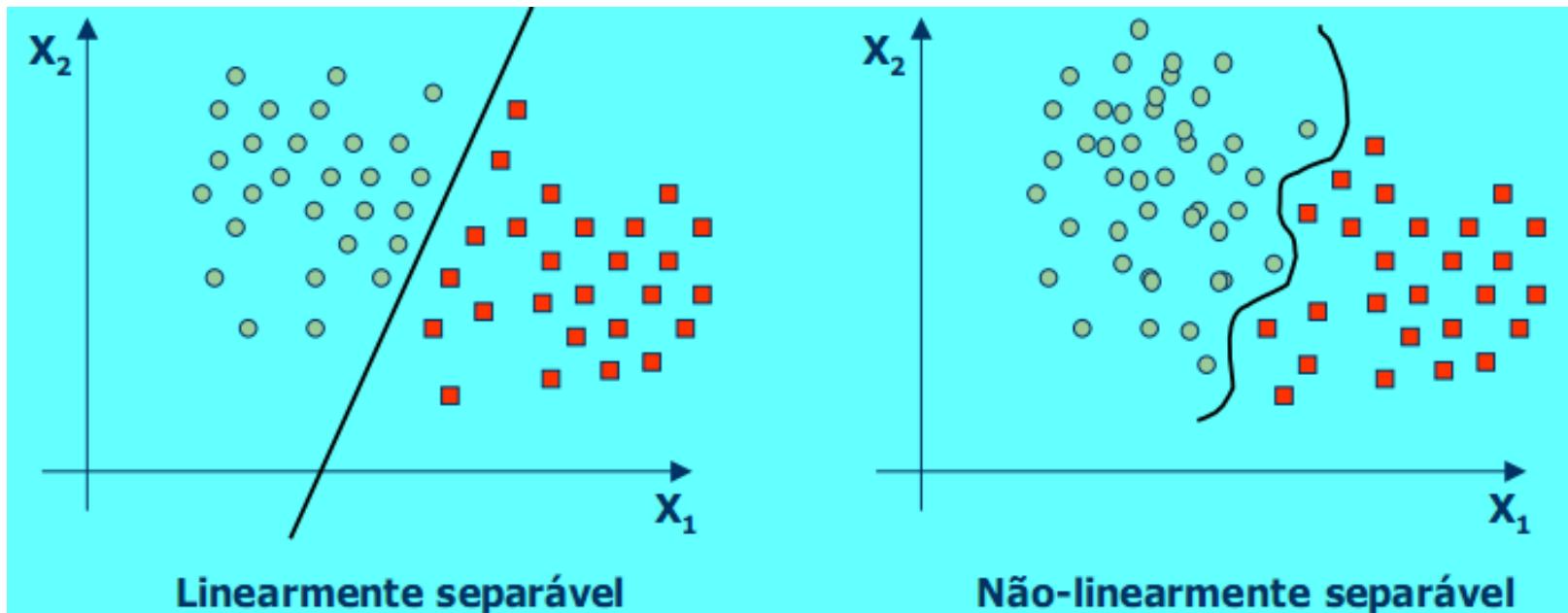
Analogia com o cérebro real



- Quantidade de neurônios ~
5 x quantidade de árvores
na floresta Amazônica
- Quantidade de conexões ~
quantidade de folhas...

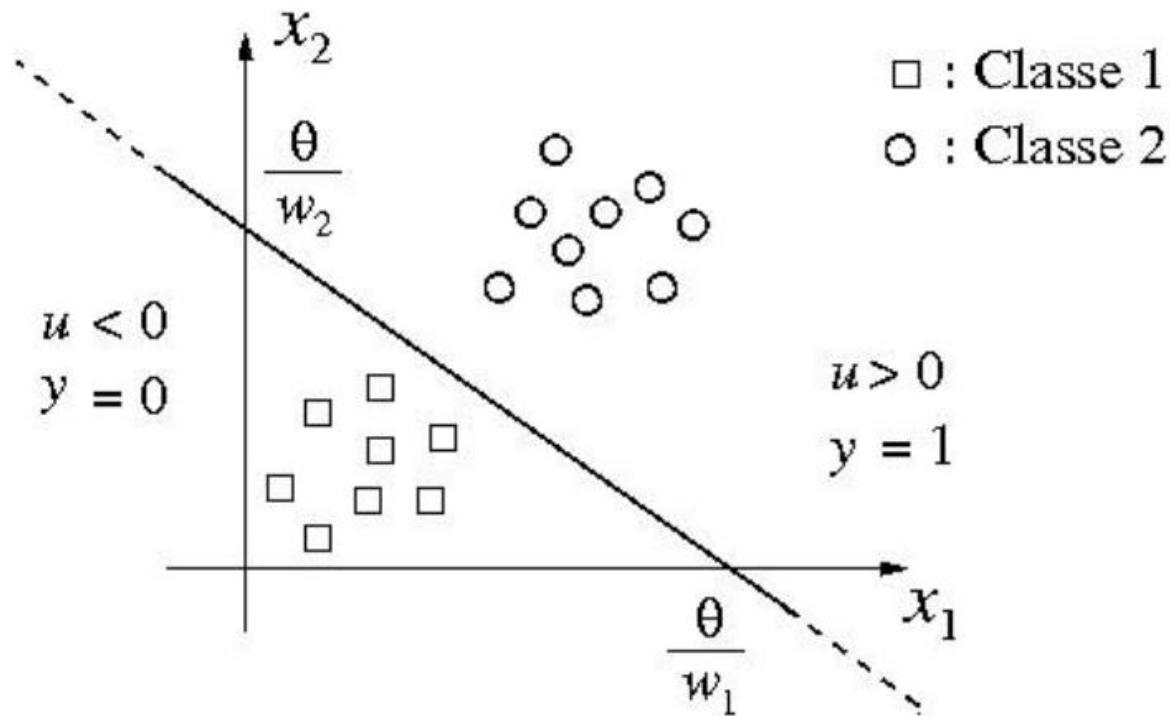
Limitações do Perceptron

- As duas classes devem ser linearmente separáveis
- Não admite mais de uma camada de pesos ajustáveis
- Aprendizado nem sempre ocorre



Limitações do Perceptron

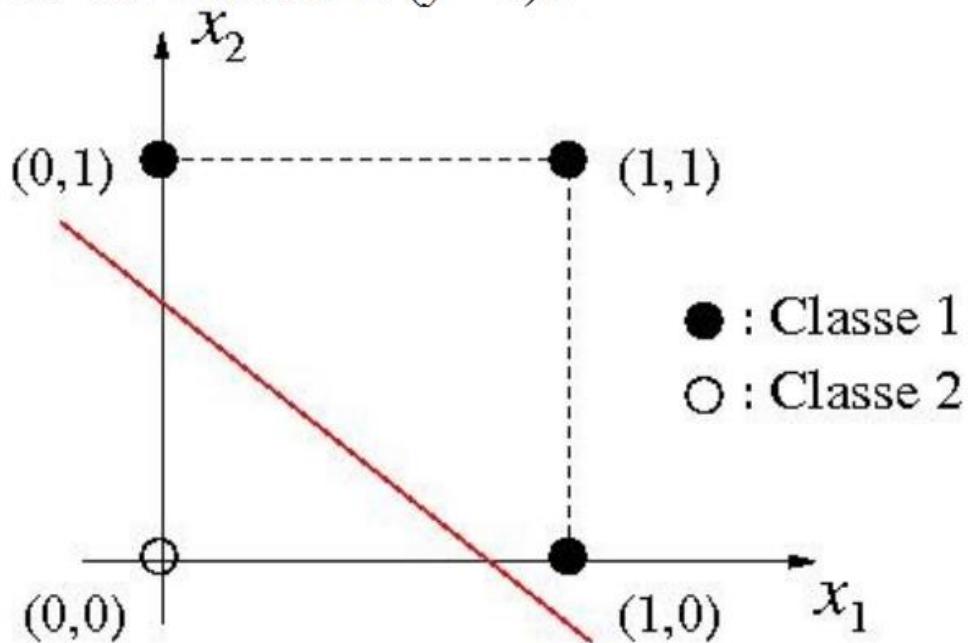
Assim, um neurônio pode ser usado para separar com eficiência duas classes que estejam bem isoladas uma da outra.



Limitações do Perceptron

Exemplo 1 (cont.): É possível encontrar uma reta que separe os pontos da Classe 1 ($y=1$) dos da Classe 2 ($y=0$)?

Resposta: SIM!



Obs: Na verdade, é possível encontrar *infinitas* retas que separam as duas classes!

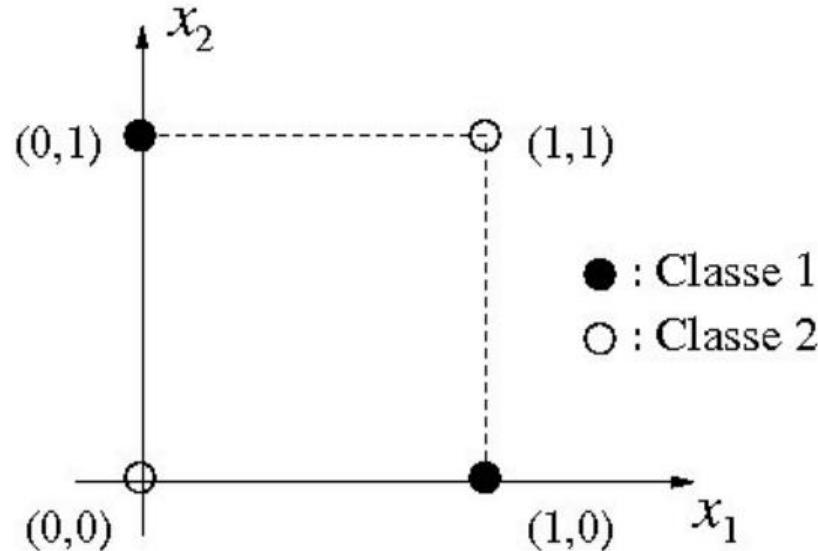
Limitações do Perceptron

Exemplo: É possível implementar a porta XOR com 1 neurônio?

Representação do Problema (Função XOR)

Porta XOR

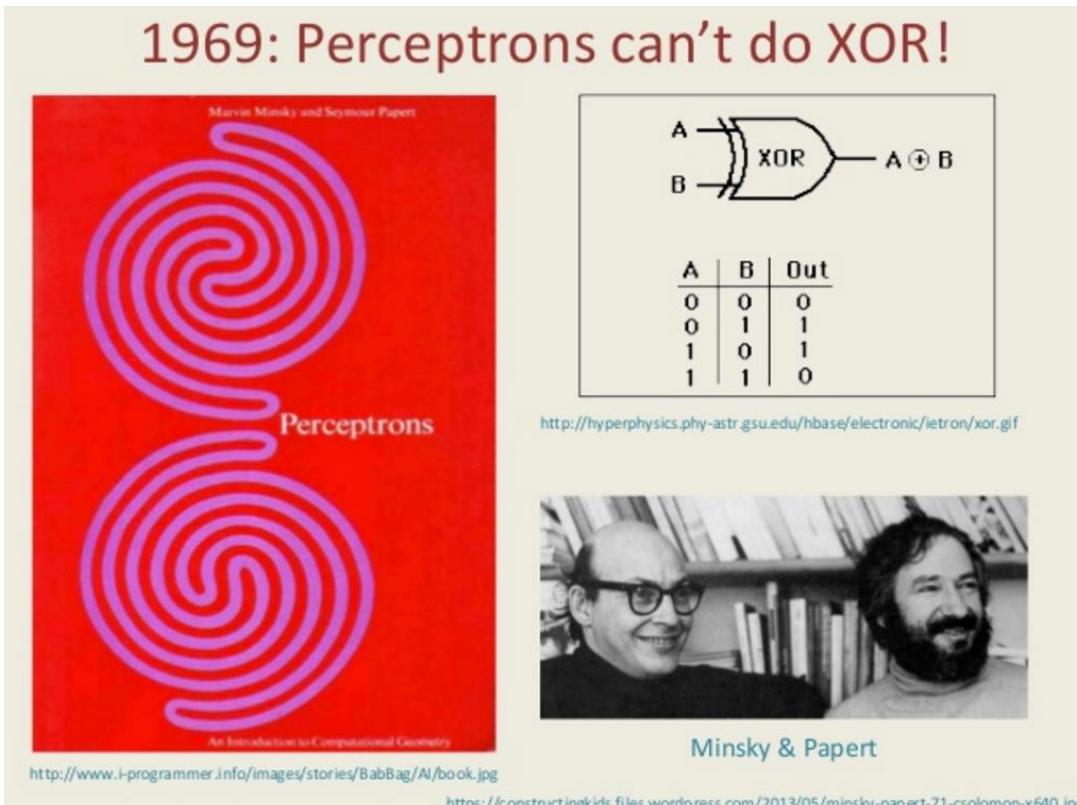
x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0



Resposta: Não!

Histórico: otimismo e AI Winter

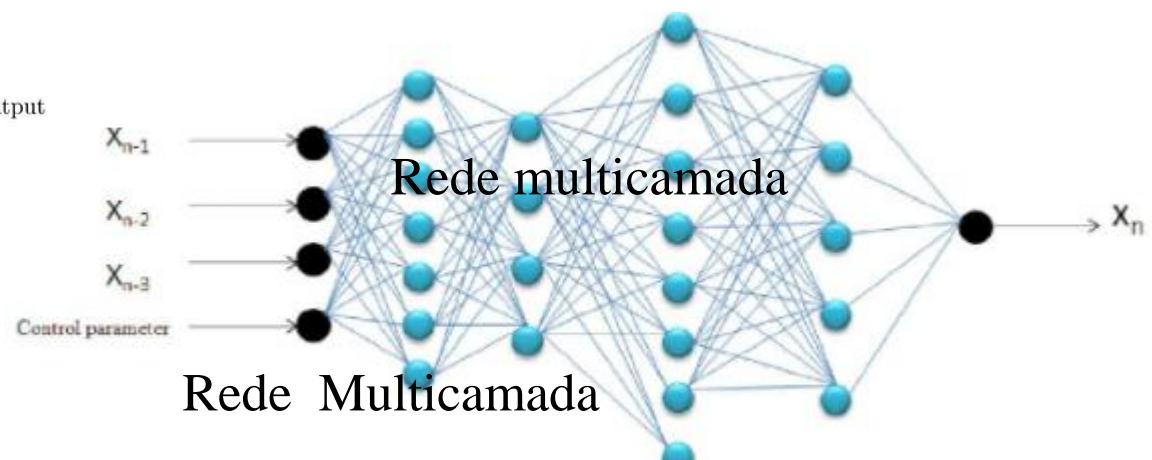
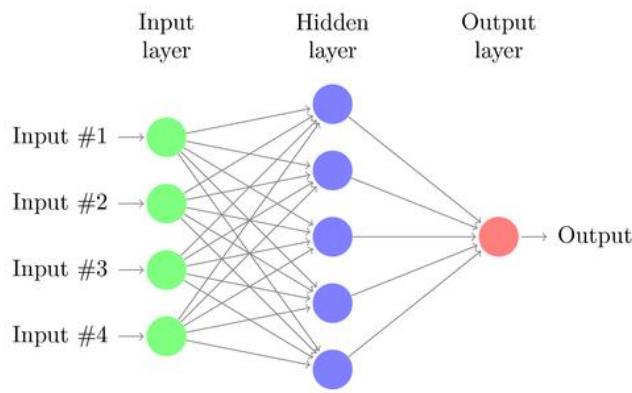
- ◆ Na década de 50 houve grande **otimismo** em relação à I.A. Surgiram os primeiros chatbots baseados em regras e a linguagem LISP.
- ◆ Dos anos 60 ao 70 → otimismo com IA foi perdido e substituído pelo realismo após obra de Minsky e Papert (**A.I. Winter** - “Inverno da I.A.”)



Histórico: Retomada nos anos 90



- Geoffrey Hinton propôs mecanismo de treinamento de Redes Neurais Multicamadas (**MLP** – Multilayer perceptron) denominado backpropagation learning
- Desenvolvimento do **Deep learning** (aprendizado de máquina)

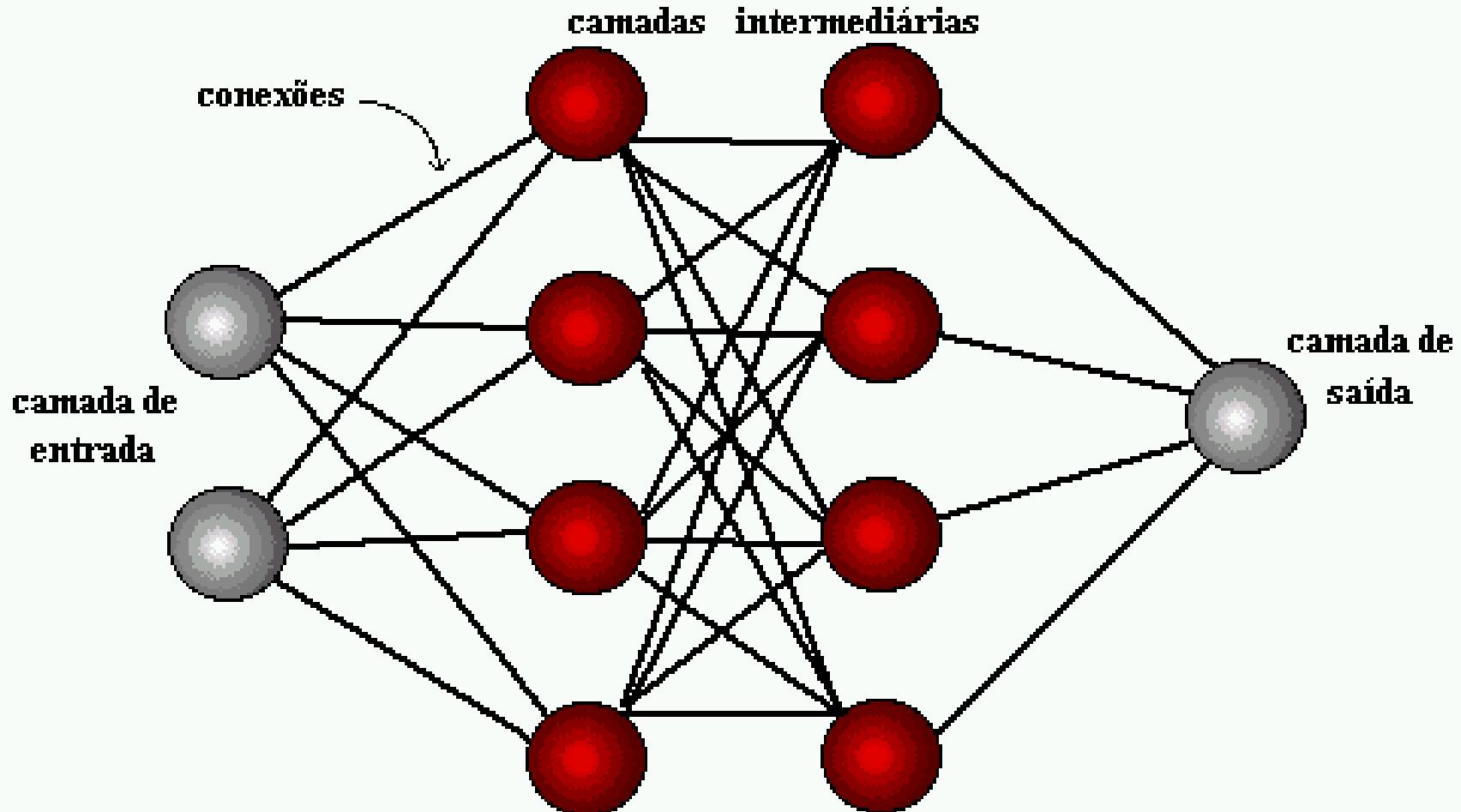


Redes profundas (deep learning)

Structure	Description of decision regions	Exclusive-OR problem	Classes with meshed regions	General region shapes
Single layer	Half plane bounded by hyperplane			
Two layer	Arbitrary (complexity limited by number of hidden units)			
Three layer	Arbitrary (complexity limited by number of hidden units)			

Modelo de Rede Neural Multicamada

MultiLayer Perceptron (MLP)



Organização em camadas

- Usualmente as camadas são classificadas em três grupos:

Camada de Entrada:

- onde os padrões são apresentados à rede;

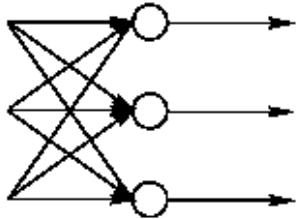
Camadas Intermediárias ou Escondidas:

- onde é feita a maior parte do processamento, através das conexões ponderadas; podem ser consideradas como extratoras de características;

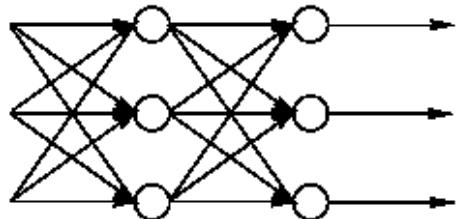
Camada de Saída:

- onde o resultado final é concluído e apresentado.

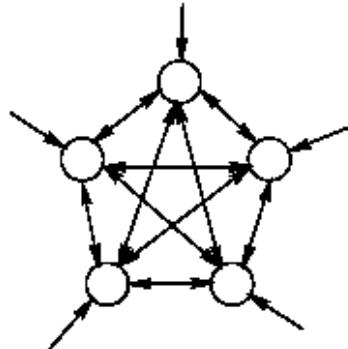
Diferentes arquiteturas de Redes MLP



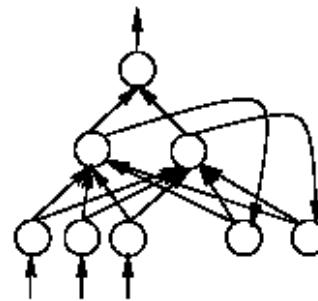
a) Single-Layer Perceptron



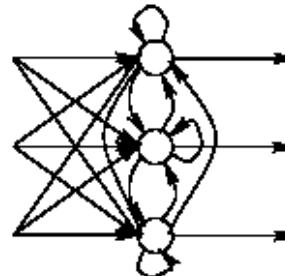
b) Multi-Layer Perceptron



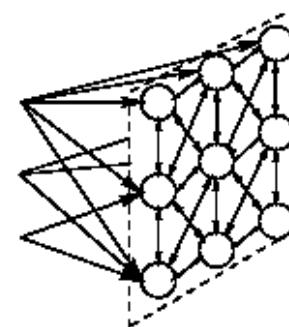
c) Hopfield network



d) Elman recurrent network

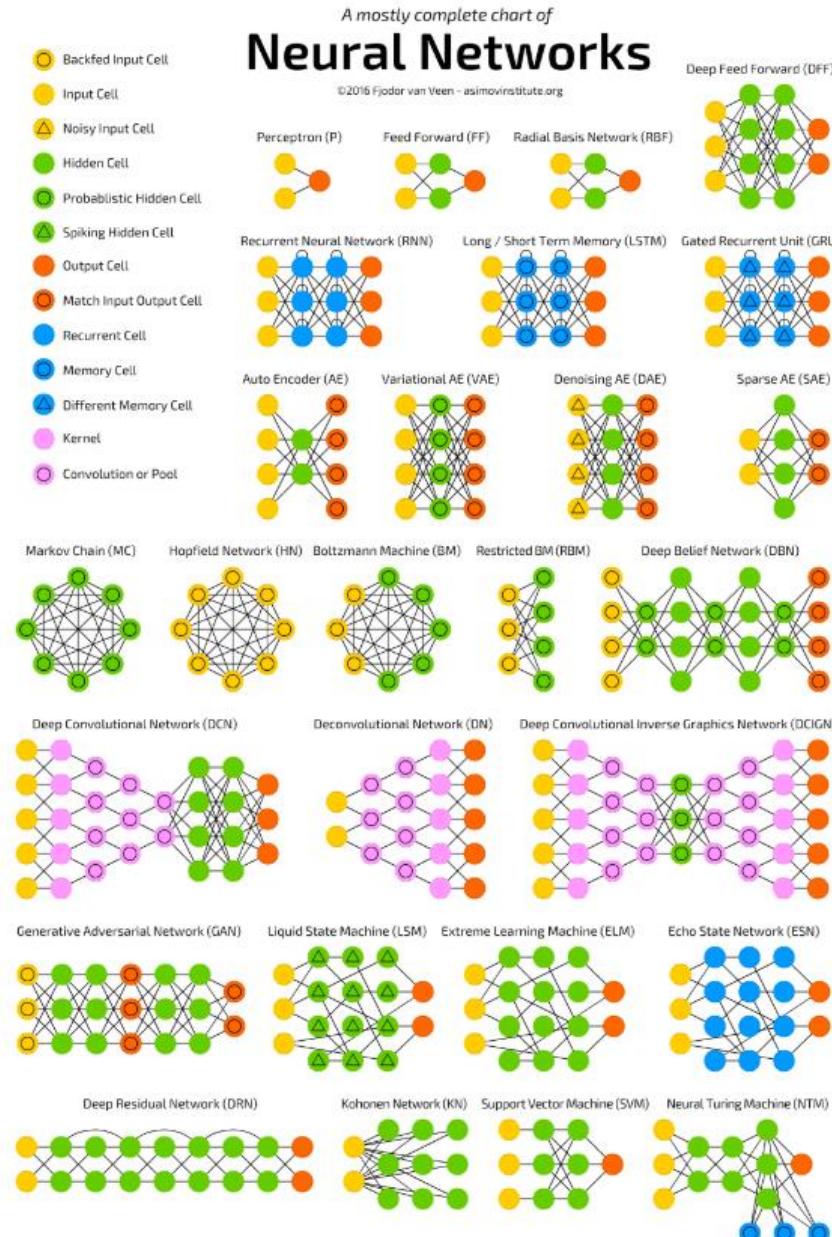


e) Competitive networks



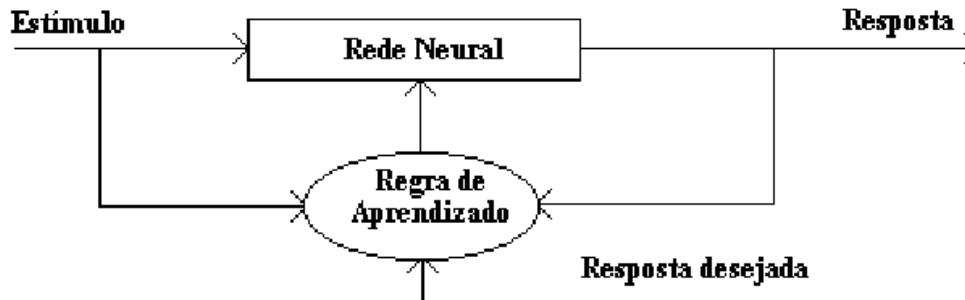
f) Self-Organizing Maps

<http://www.asimovinstitute.org/neural-network-zoo/>

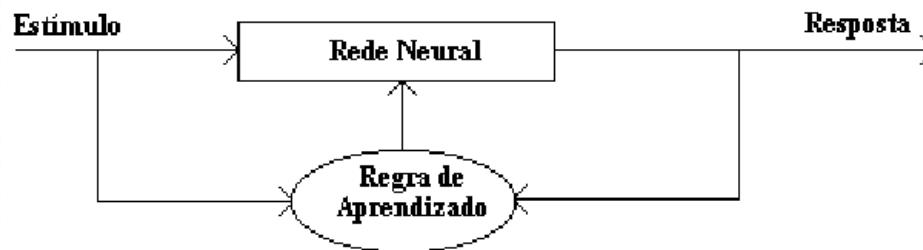


Algoritmo de Aprendizado

- **Aprendizado Supervisionado**, quando é utilizado um agente externo que indica à rede a resposta desejada para o padrão de entrada;



- **Aprendizado Não Supervisionado** (auto-organização), quando não existe uma agente externo indicando a resposta desejada para os padrões de entrada;



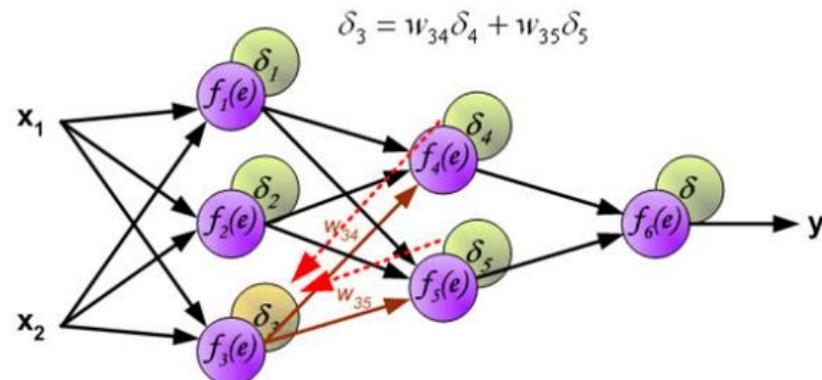
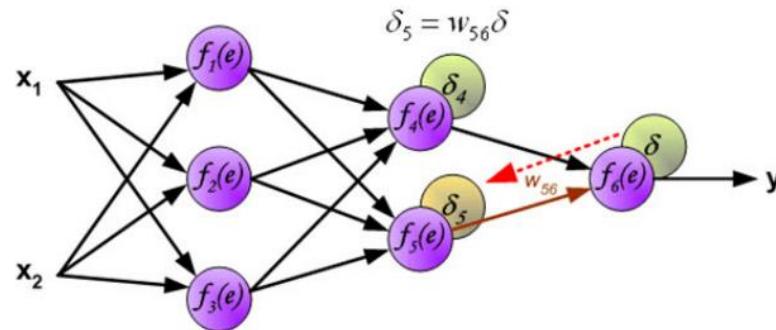
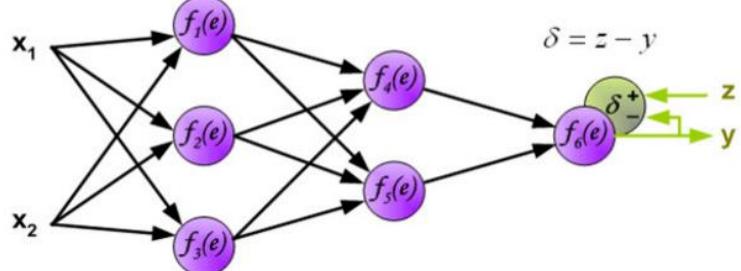
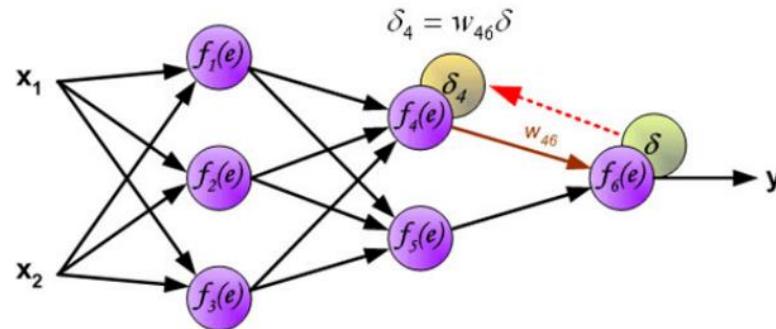
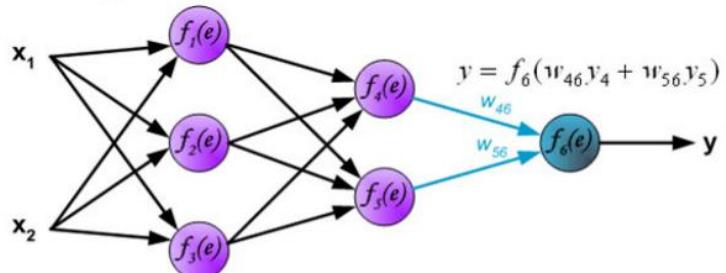
- **Reforço**, quando um crítico externo avalia a resposta fornecida pela rede.

Backpropagation

- Durante o treinamento com o algoritmo backpropagation, a rede opera em uma sequência de dois passos:
- **Primeiro:**
 - um padrão é apresentado à camada de entrada da rede. A atividade resultante flui através da rede, camada por camada, até que a resposta seja produzida pela camada de saída.
- **Segundo:**
 - a saída obtida é comparada à saída desejada para esse padrão particular. Se esta não estiver correta, o erro é calculado. O erro é propagado a partir da camada de saída até a camada de entrada, e os pesos das conexões das unidades das camadas internas vão sendo modificados conforme o erro é retropropagado.

- Desvantagens do algoritmo de aprendizagem *backpropagation*:
- ❖ Normalmente o tempo de processamento é elevado
 - ❖ A arquitetura da rede deve ser fixada *a priori*

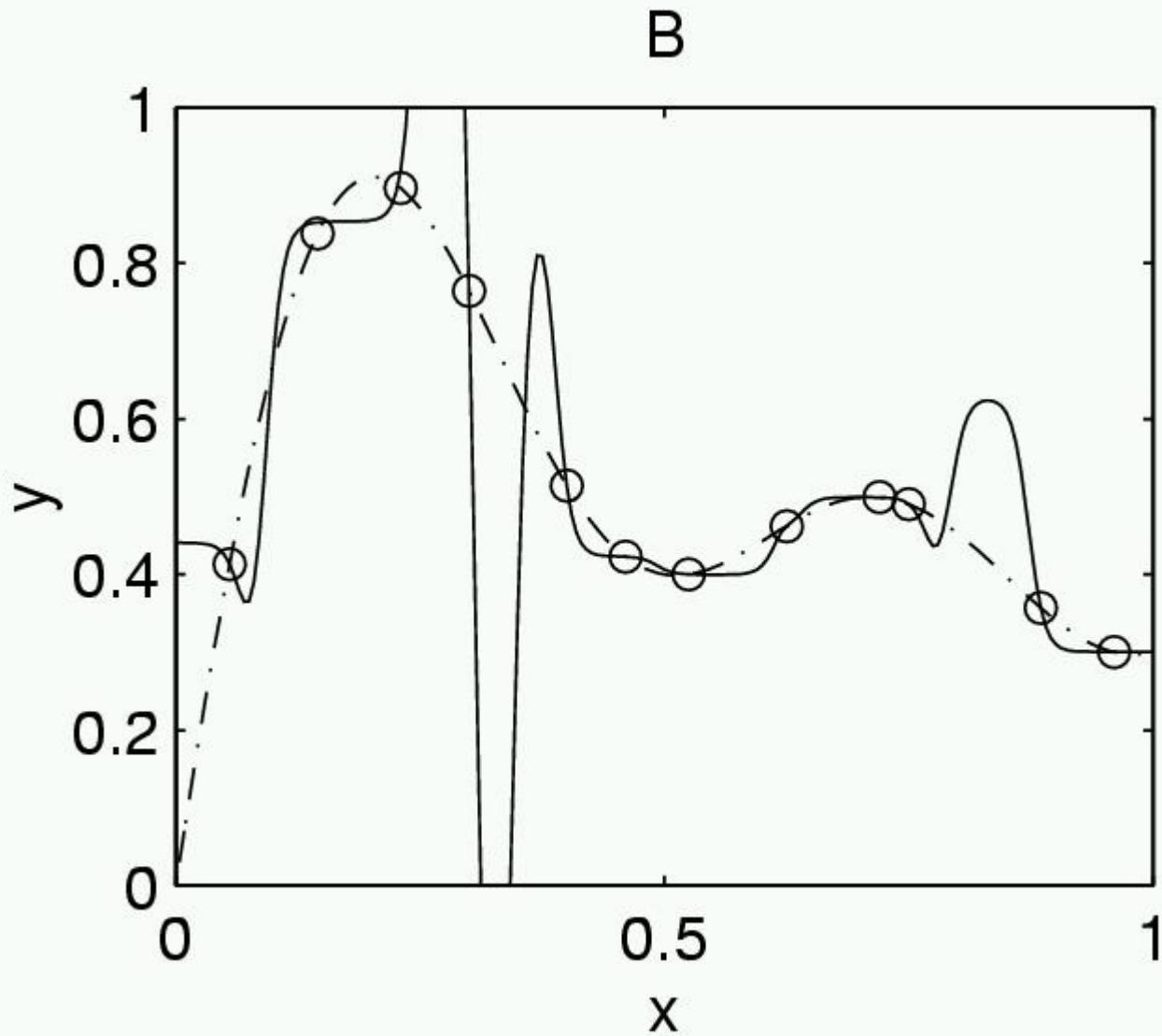
Backpropagation



Definição da Arquitetura da rede

- A quantidade de neurônios na camada de entrada e saída é dada pelo problema a ser abordado.
- No entanto, a quantidade de neurônios nas camadas de processamento são características do projeto.
Aumentando-se o número de neurônios na camada escondida aumenta-se a capacidade de mapeamento não-linear da rede.

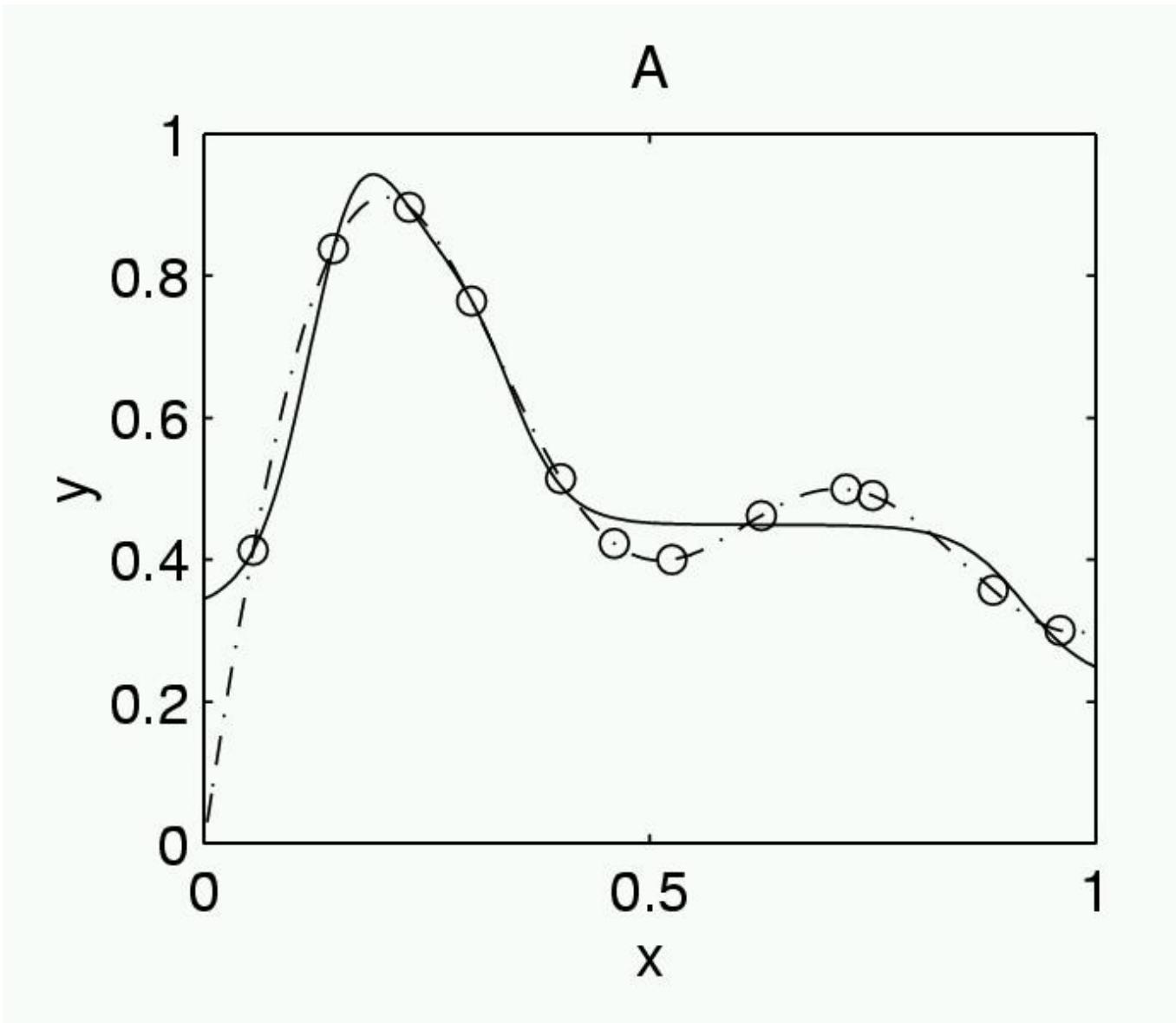
Primeiro treinamento



Underfitting

- Por outro lado, uma rede com poucos neurônios na camada escondida pode não ser capaz de realizar o mapeamento desejado, o que é denominado de *underfitting*.
- O *underfitting* também pode ser causado quando o treinamento é interrompido de forma prematura.

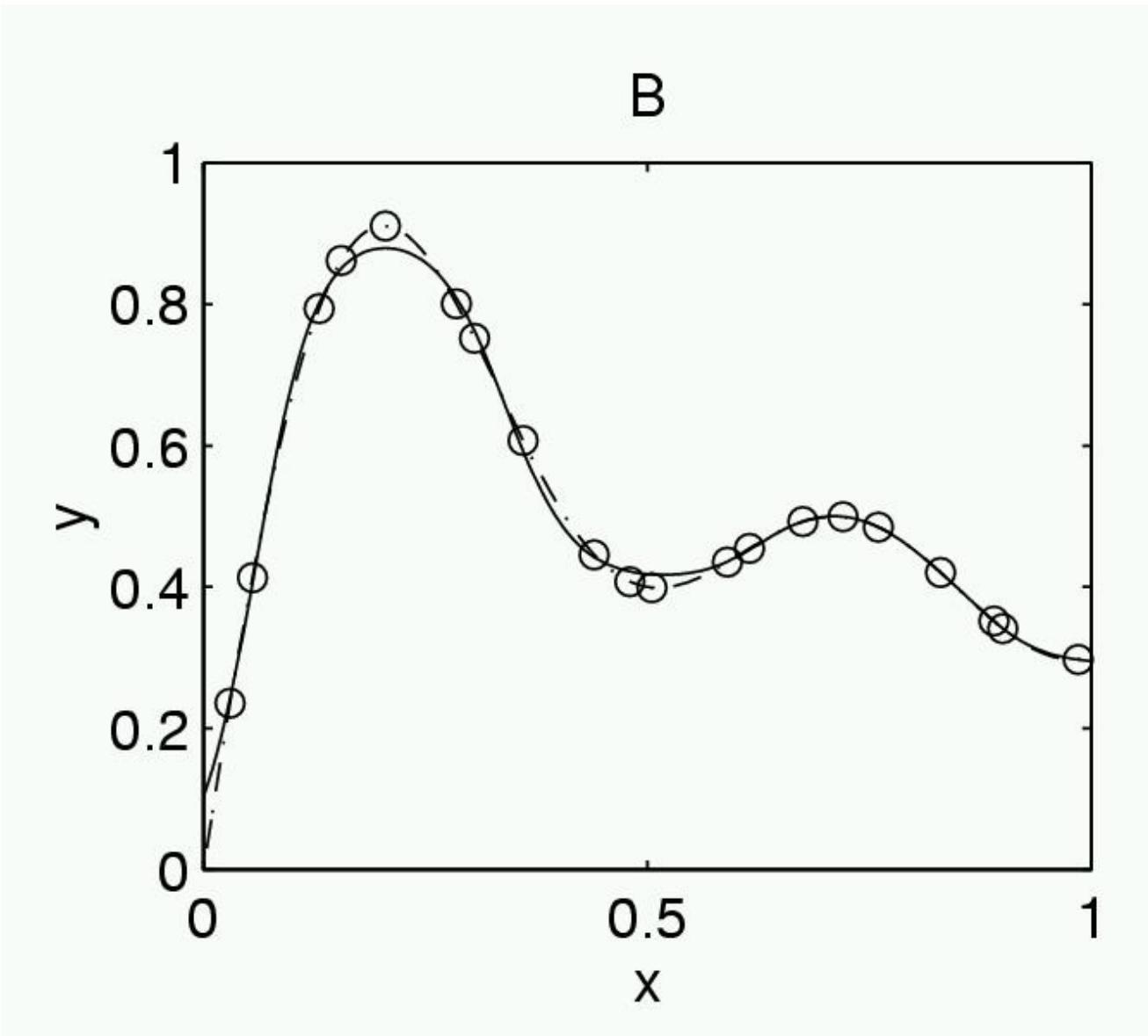
Underfitting



Overfitting

- No entanto, quando o número de neurônios na camada escondida muito grande, o modelo pode se sobreajustar aos dados, na presença de ruído nas amostras de treinamento.
- Diz-se que a rede está sujeito ao sobre-treinamento (***overfitting***), ou seja, quando o modelo estatístico se ajusta em demasiado ao conjunto de dados/amostra.
- O overfitting pode ocorrer também quando o treinamento alcança uma quantidade de épocas / passos muito elevado.
- Uma alternativa é utilizar uma técnica ao final do treinamento conhecida como ***dropout***, que remove aleatoriamente alguns neurônios da rede.

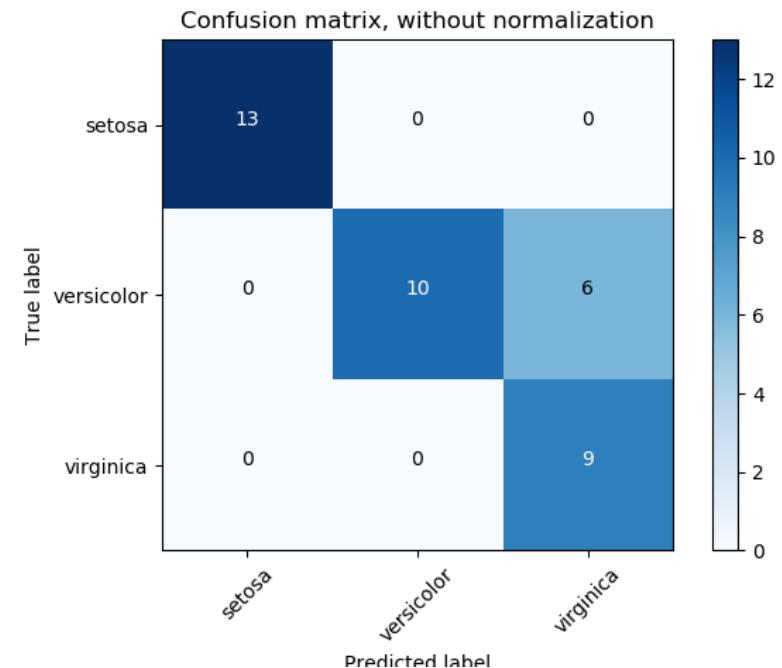
Overfitting



Matriz de Confusão (Confusion Matrix)

- Mostra a quantidade de valores corretos e errados:

		Predicted class	
		<i>P</i>	<i>N</i>
<i>P</i>	True	False	
	Positives (TP)	Negatives (FN)	
<i>N</i>	False	True	
	Positives (FP)	Negatives (TN)	



Medindo a acurácia de uma Rede Neural

- Considere a **Confusion Matrix**

		Predicted		
		Positive	Negative	
Actual True	TP	FN		
	FP	TN		

- **Precision (Precisão):**

- Proporção de positivos que foram classificados corretamente: **TP/(TP + FP)**
- Ou seja, está tendo muito falso positivo?

- **Recall (Recuperação):**

- Taxa de positivos verdadeiros: **TP/(TP + FN)**
- Ou seja, está tendo muito falso negativo?

- **F1 Score:**

- Ela é a média harmônica dessas 2 métricas e é calculada como tal:
- **F1 = 2 (Precision x Recall) / (Precision + Recall)**

4. Visão Computacional

Visão Computacional

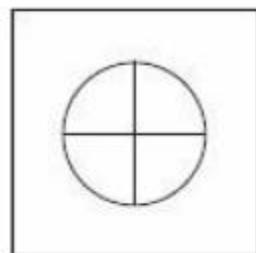
- Para o ser humano é fácil identificar uma laranja



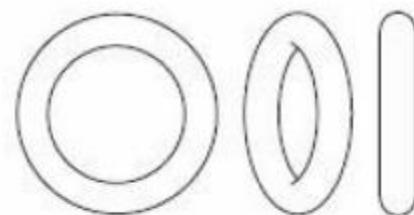
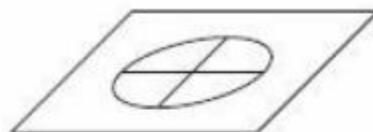
- Mas isso é difícil para o computador, pois ele enxerga apenas uma sequência de pixels de um arquivo.
- Veja a seguir várias dificuldades relacionadas à visão computacional

Dificuldades relacionadas à variação da aparência

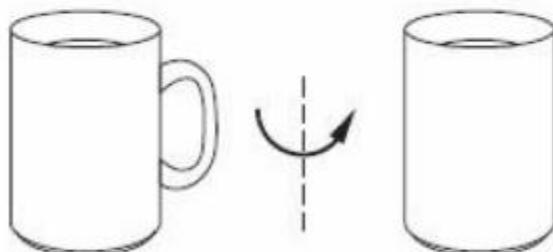
- **Encurtamento** devido à inclinação
- Posição do observador muda drasticamente o **aspecto**.
- A **occlusão** esconde partes sobrepostas ou escondidas pelo próprio objeto: **auto-occlusão**
- Alguns objetos podem **deformar** drasticamente.



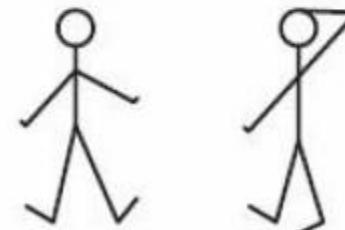
Encurtamento



Aspecto



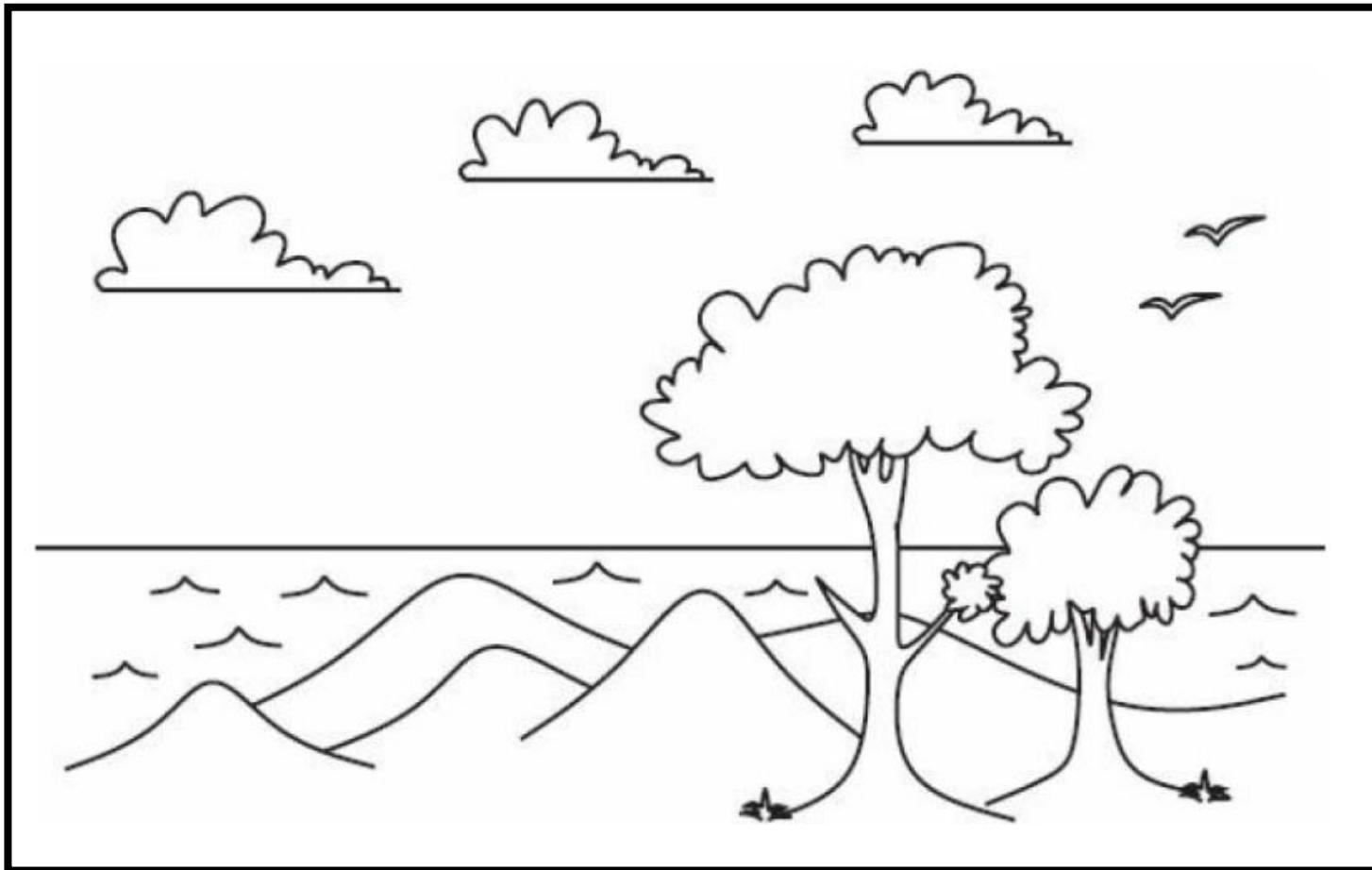
Occlusão



Deformação

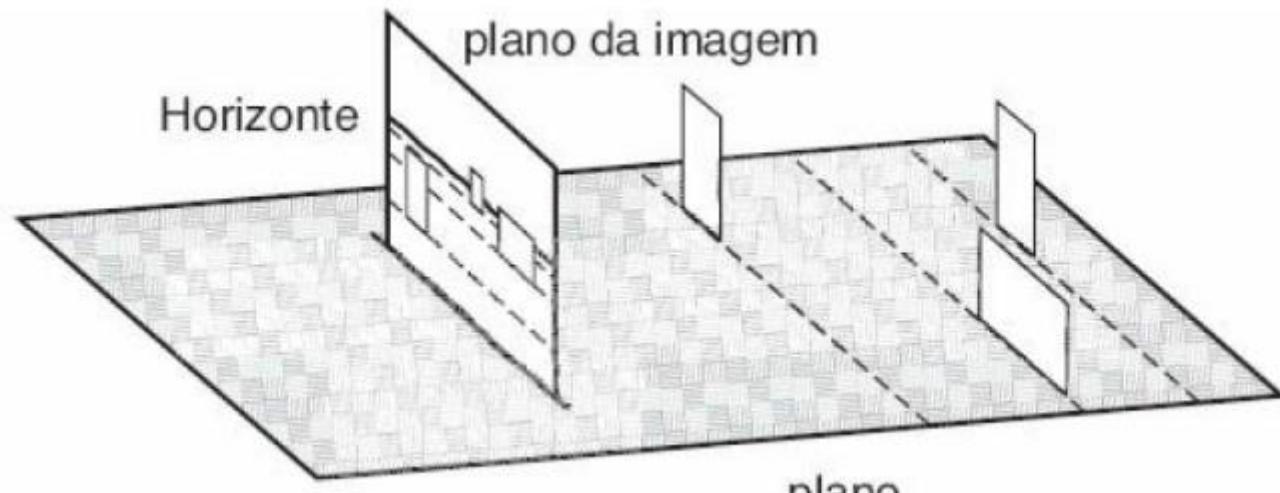
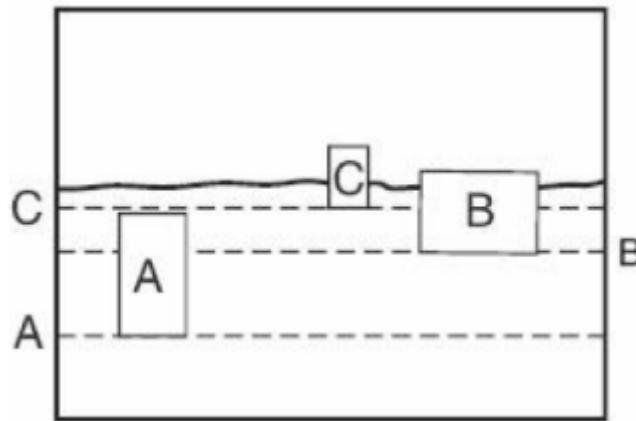
Percepção tridimensional

- Para o ser humano é fácil ter uma percepção vívida sobre o layout e formas tridimensionais a partir de uma desenho de linha. Para o computador isso não é natural.



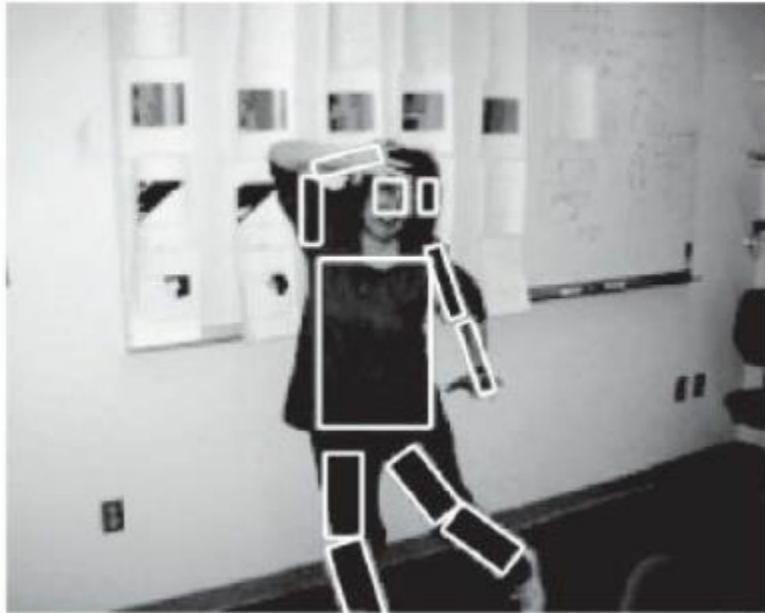
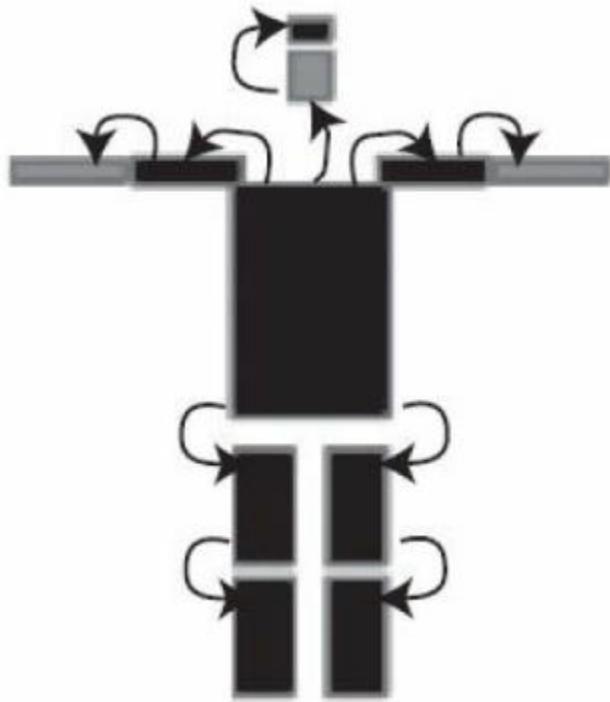
Percepção de profundidade e tamanho do objeto

- Nós conseguimos identificar a profundidade e o tamanho dos objetos devido à posição em relação ao plano do chão. Mas como uma máquina pode identificar isso?

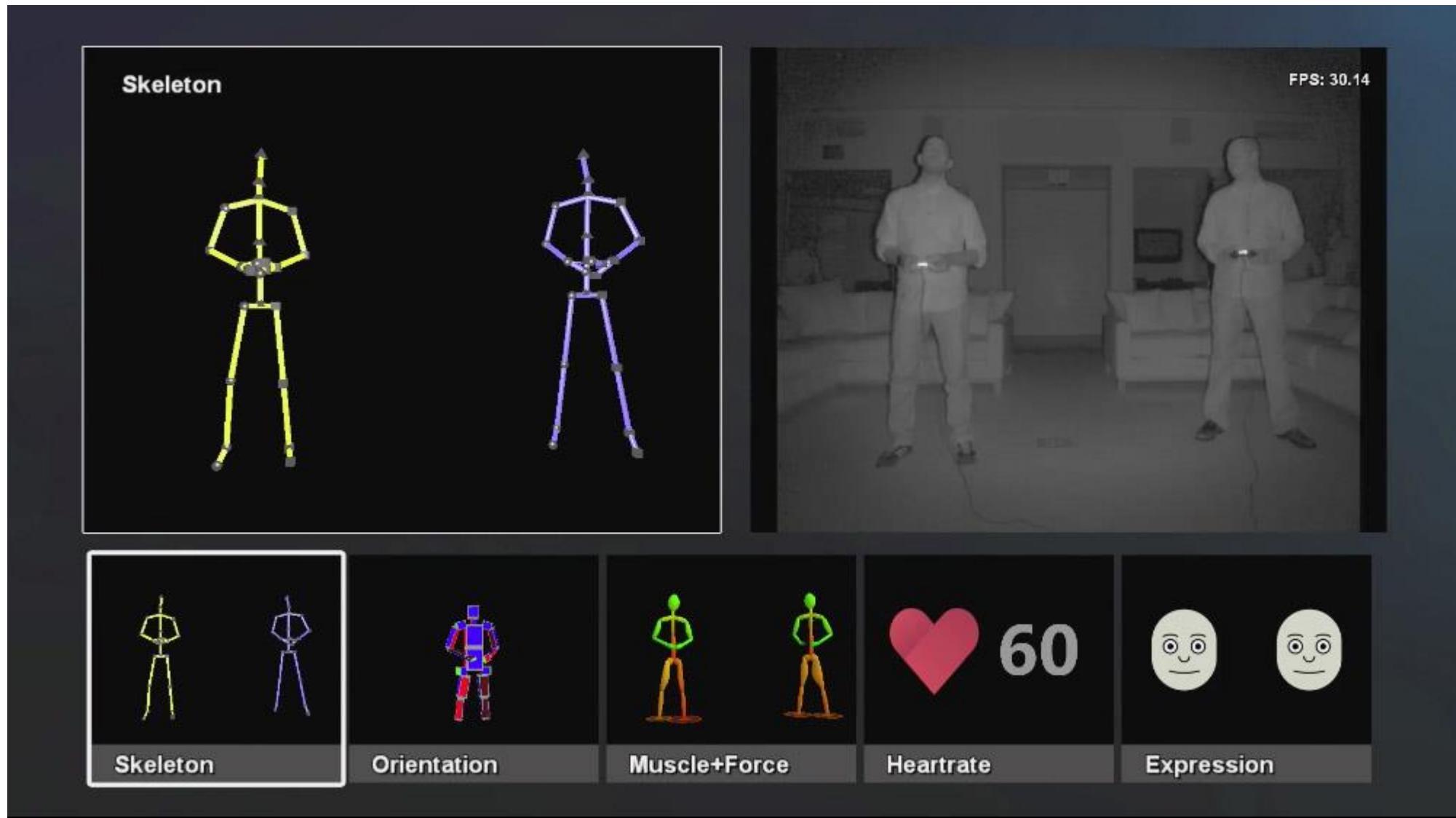


Geometria dos objetos

- Para objetos que se deformam, como uma pessoa, é comum criar uma geometria deformável que possa ser identificada no objeto.

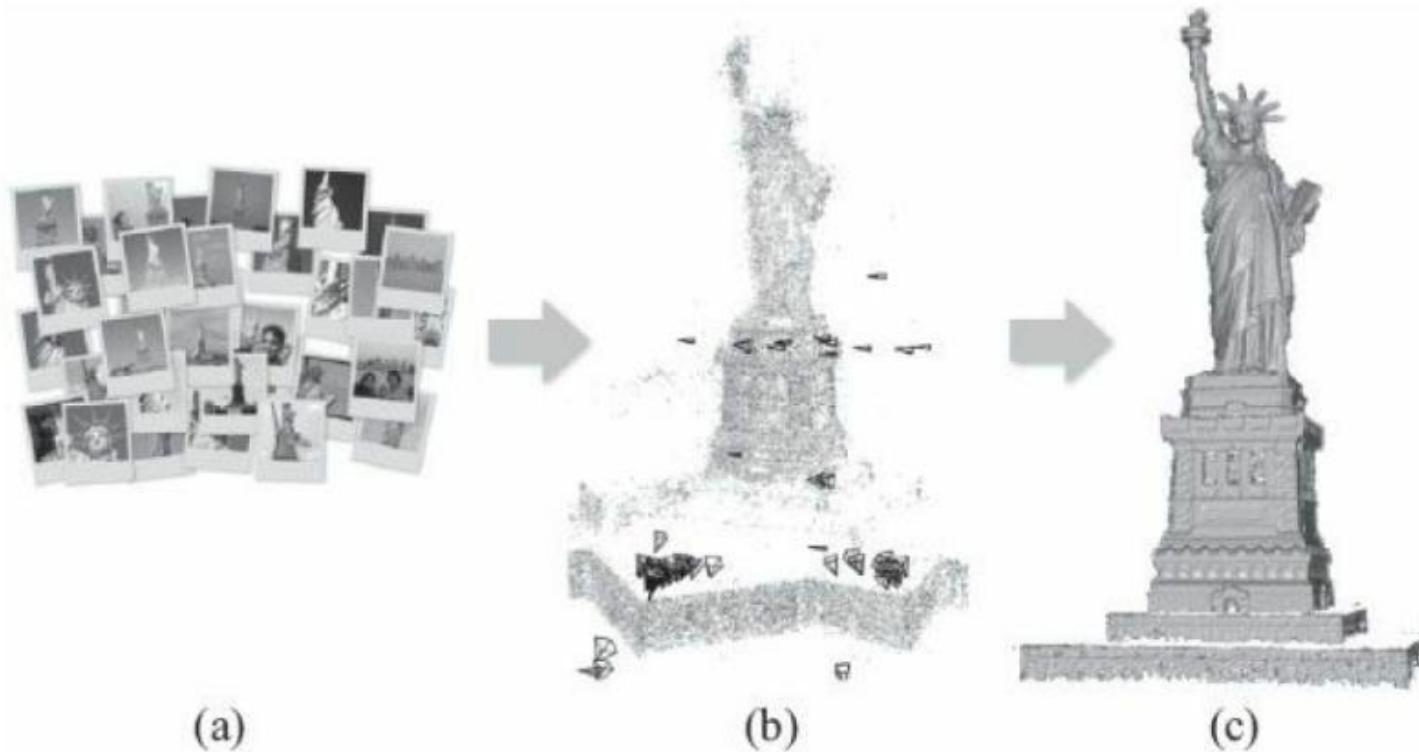


Modelo de uma pessoa no Kinect (XBox)



Criação de modelos 3D a partir de imagens 2D

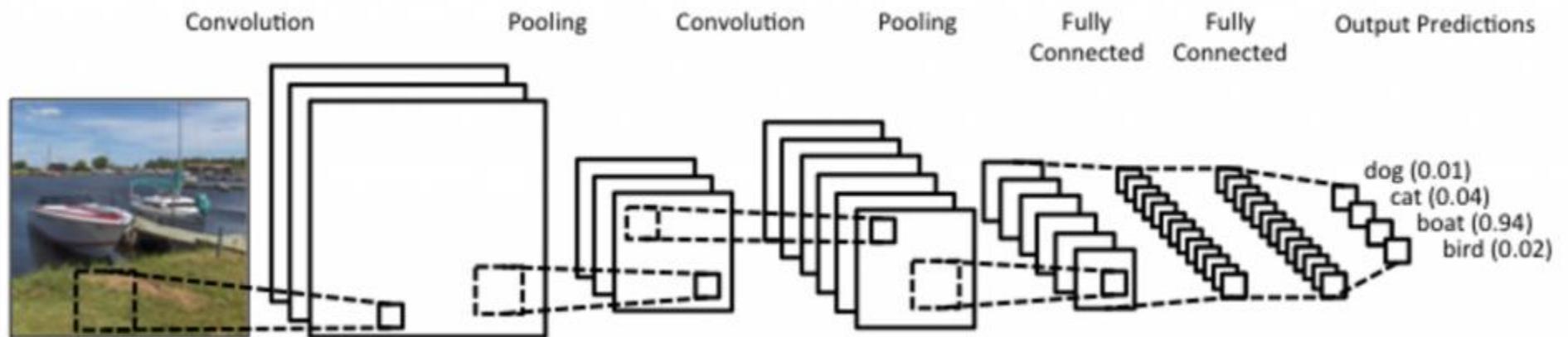
- Um dos maiores desafios consiste em reconstruir um modelo a partir de múltiplas visões, como construir um modelo 3D de uma estátua a partir de diversas fotos



Uma possível solução são as

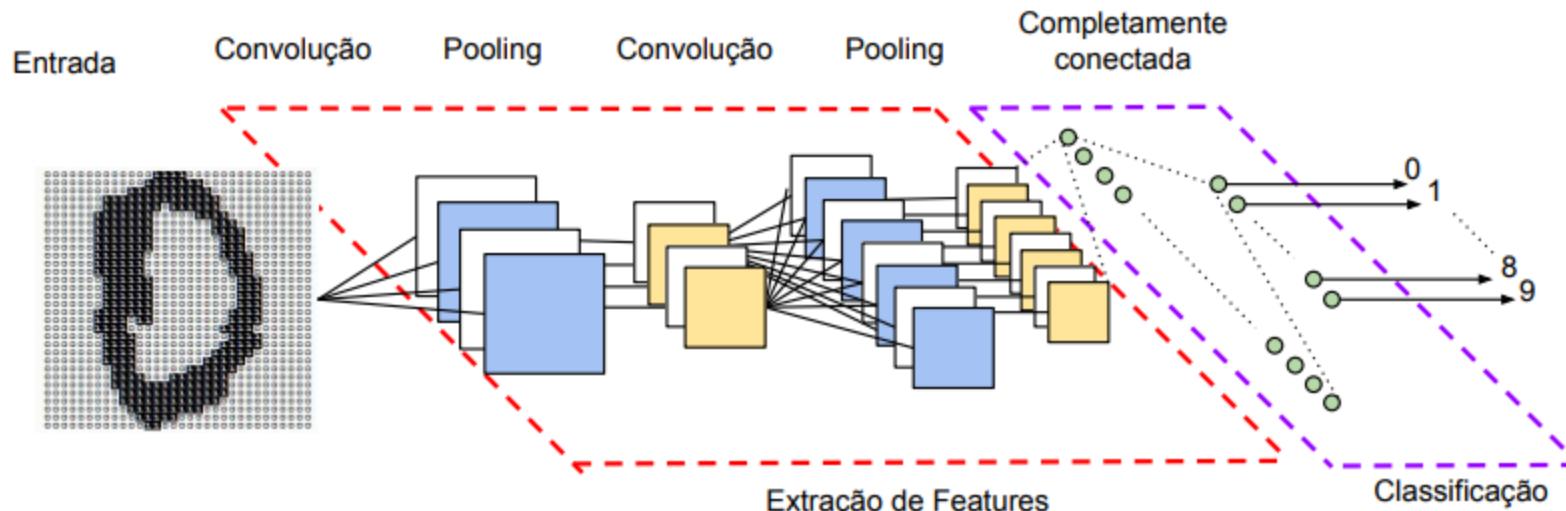
Redes Neurais Convolucionais

Convolution Neural Network - CNN

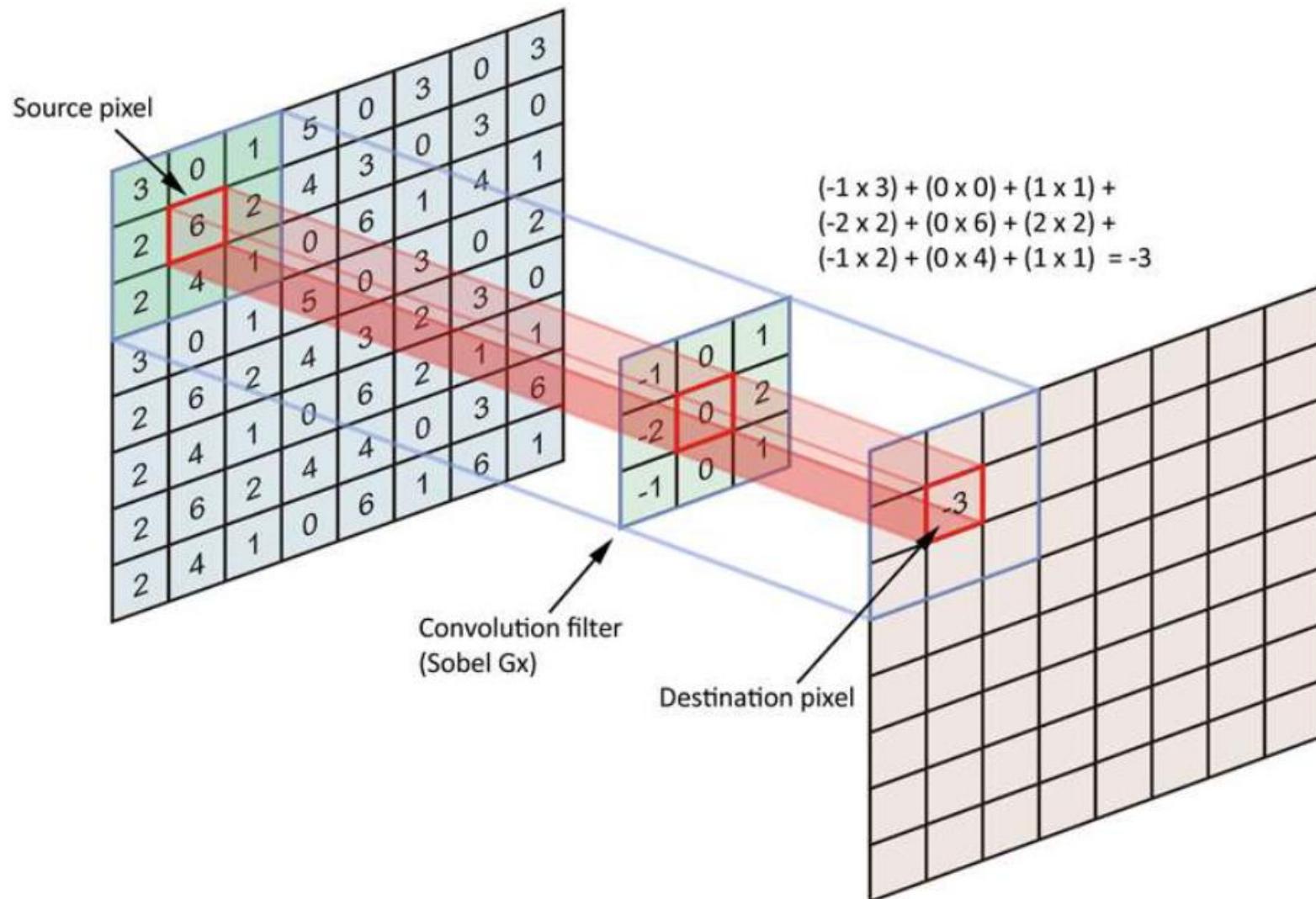


Redes Neurais Convolucionais (CNN)

- Uma Rede Neural Convolucional é uma variação das redes MLP.
- É composta de algumas etapas, conforme a imagem a seguir:
 - **Convolução**, que consiste em extrair características da imagem com a aplicação de filtros (Ex: detecção de borda, nitidez, desfoque, desfoque gaussiano)
 - **Pooling**, que realiza a diminuição do tamanho da imagem para reduzir o tamanho do problema, mas mantendo a relação de proximidade entre os pixels ao longo do processamento da rede.
 - **Classificação** da imagem em categorias conhecidas

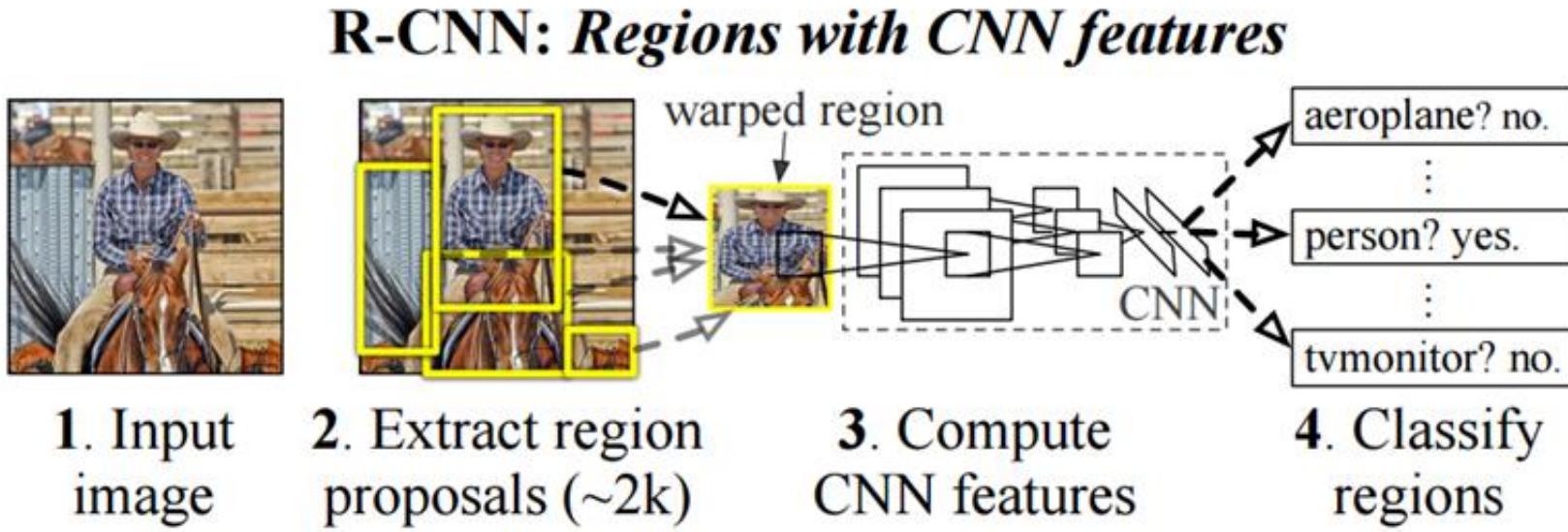


Funcionamento do Pooling



Redes baseadas em Regiões (R-CNN, Fast R-CNN, Faster R-CNN)

- O objetivo deste tipo de rede neural é identificar objetos em imagens.
- Funcionamento: primeiro são propostas diversas regiões aleatórias (de 300 a 2mil); em seguida, cada região proposta é submetida a uma rede neural CNN para classificação de objetos.



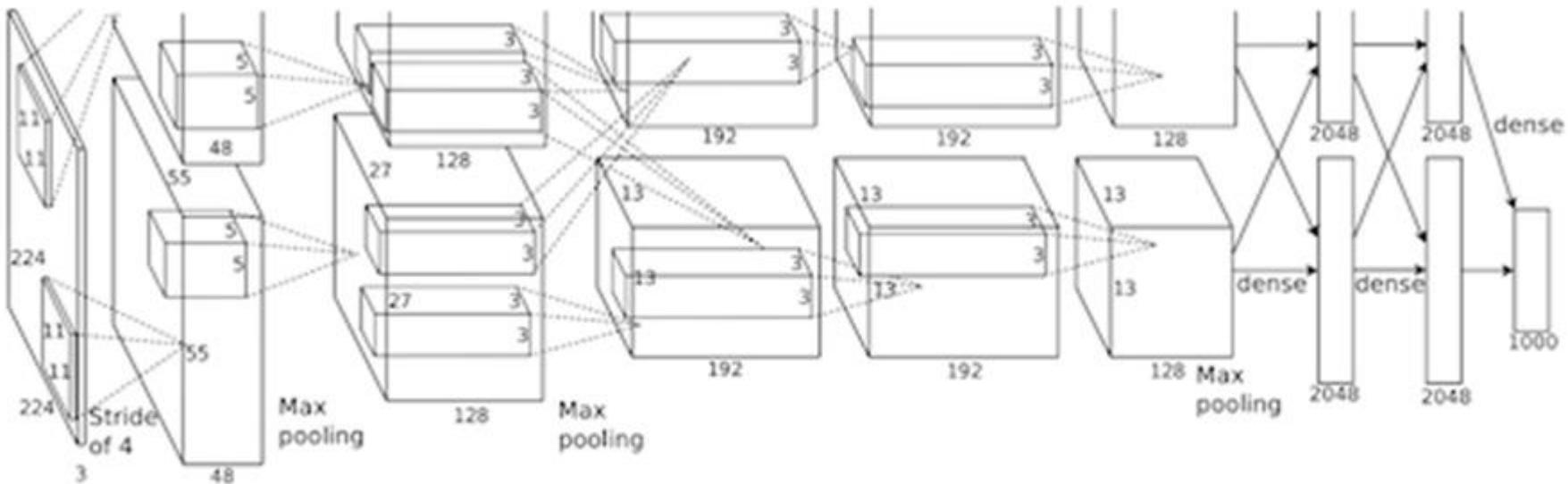
Exemplos de CNNs

- AlexNet (2012)
- ZF Net (2013)
- VGG Net (2014)
- Generative Adversarial Networks (2014)
- Generating Image Descriptions (2014)
- GoogLeNet (2015)
- Microsoft ResNet (2015)
- Spatial Transformer Networks (2015)
- Region Based CNNs (R-CNN - 2013, Fast R-CNN - 2015, Faster R-CNN - 2015)

<https://adeshpande3.github.io/adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html>

AlexNet (2012)

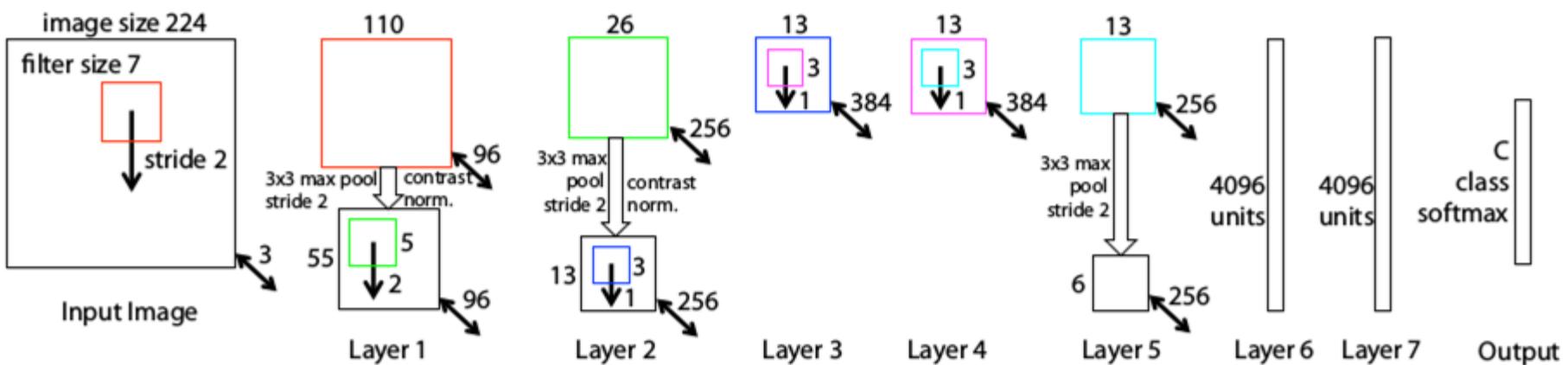
- 7 camadas escondidas, 650K neurônios, 60M parâmetros
- Taxa de erro: 15.4%
- Uso de GPU é 50x mais rápido que utilizando uma CPU
- Treino utilizando ImageNet (~15M imagens e 22K categorias), utilizando 2 GPUs GTX 580 durante seis dias



AlexNet architecture (May look weird because there are two different “streams”. This is because the training process was so computationally expensive that they had to split the training onto 2 GPUs)

ZF Net (2013)

- 7 camadas
- Taxa de erro: 11.2%
- Treinou apenas 1,3M imagens
- Usou uma GTX 580 GPU por 12 dias



ZF Net Architecture

VGG Net (2014)

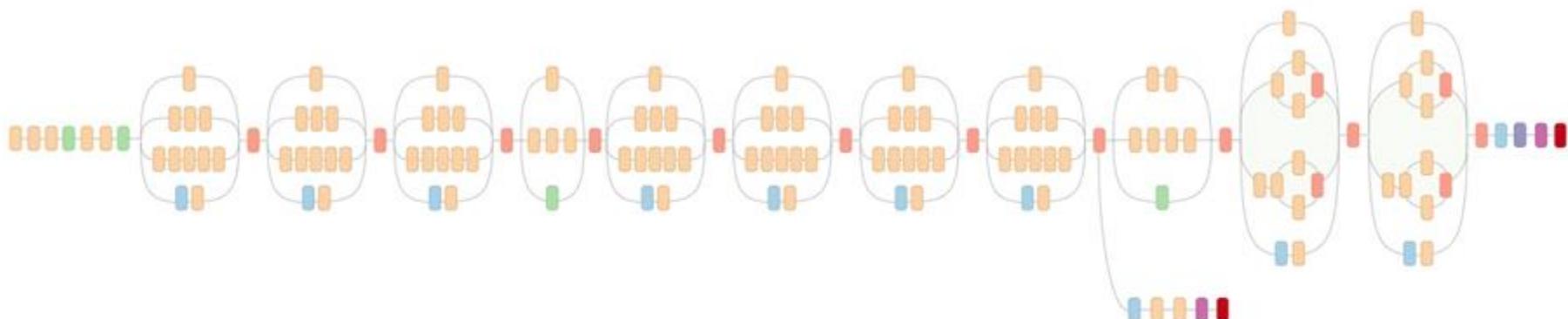
- 19 camadas
- Taxa de erro:
7.3%
- Treinou com 4
Nvidia Titan
Black GPUs por
duas a três
semanas
- Funciona bem
para
reconhecimento e
classificação

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv1-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

The 6 different architectures of VGG Net. Configuration D produced the best results

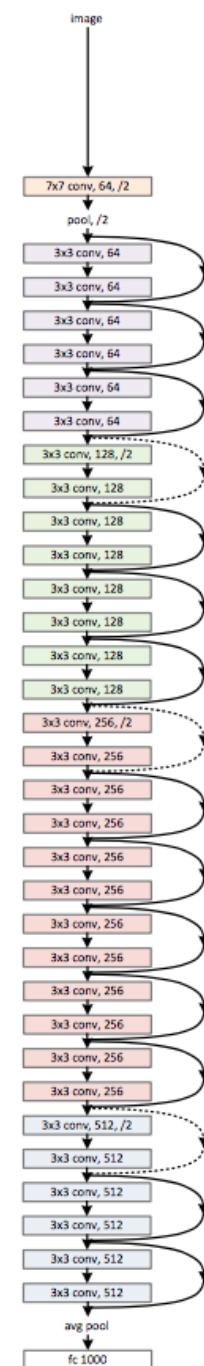
GoogLeNet (2015)

- 100 camadas
- Taxa de erro: 6.7%
- Usa 9 módulos inception (processamento paralelo)
- Usa 12x menos parâmetros que AlexNet
- Utiliza conceitos de R-CNN para detecção
- Treinada com GPUs modernas dentro de uma semana



Legend:
— Convolution
— AvgPool
— MaxPool
— Concat
— Dropout
— Fully connected
— Softmax

Another view of GoogLeNet's architecture.



Microsoft ResNet (2015)

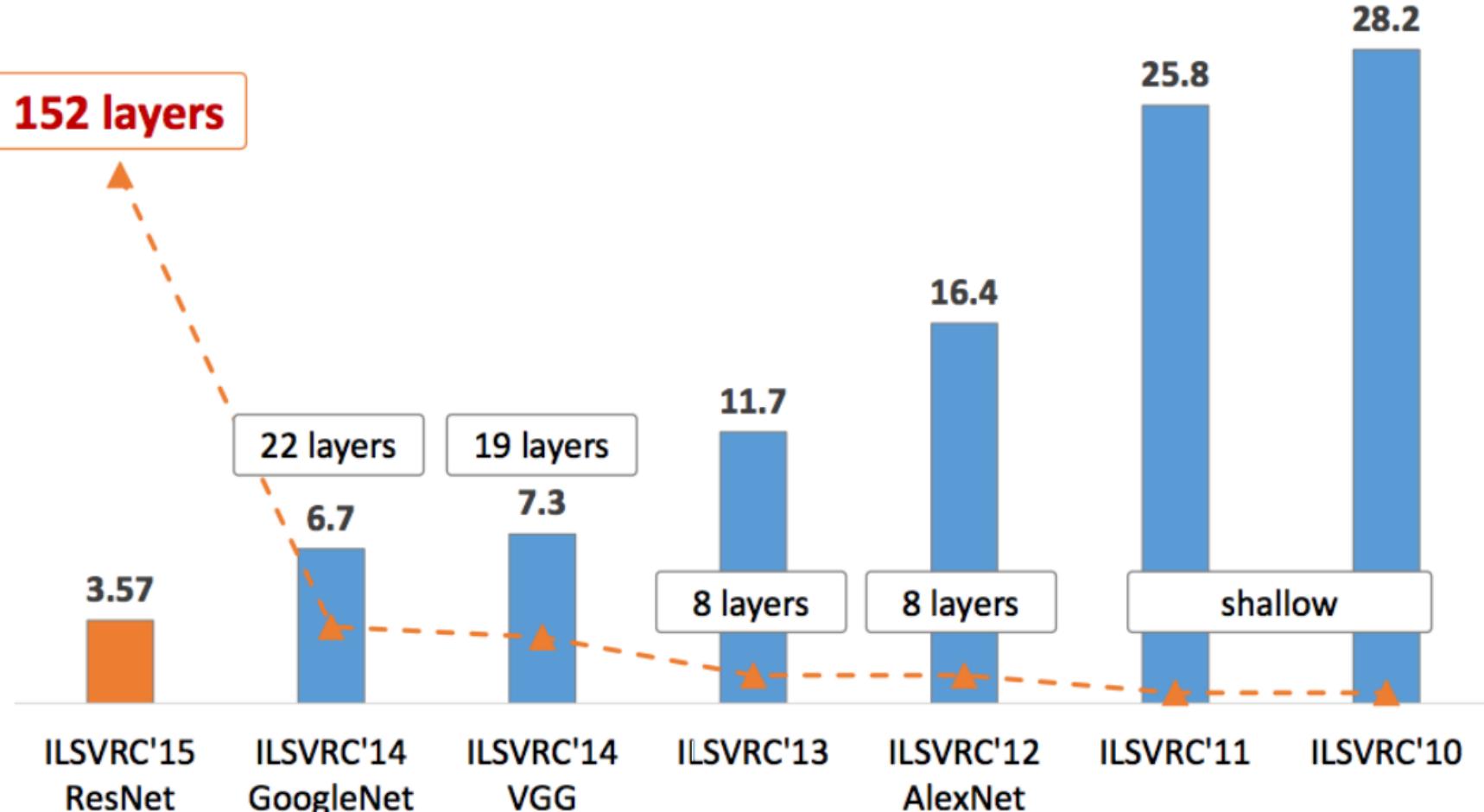
- 152 camadas (ultra deep)
- Taxa de erro: 3.6%
- Treinada com GPUs por duas a três semanas
- Melhor rede até então!
- The group tried a 1202-layer network, but got a lower test accuracy, presumably due to overfitting.

Summary: ILSVRC 2012-2015

Team	Year	Place	Error (top-5)	External data
SuperVision – Toronto (AlexNet, 7 layers)	2012	-	16.4%	no
SuperVision	2012	1st	15.3%	ImageNet 22k
Clarifai – NYU (7 layers)	2013	-	11.7%	no
Clarifai	2013	1st	11.2%	ImageNet 22k
VGG – Oxford (16 layers)	2014	2nd	7.32%	no
GoogLeNet (19 layers)	2014	1st	6.67%	no
ResNet (152 layers)	2015	1st	3.57%	
Human expert*			5.1%	

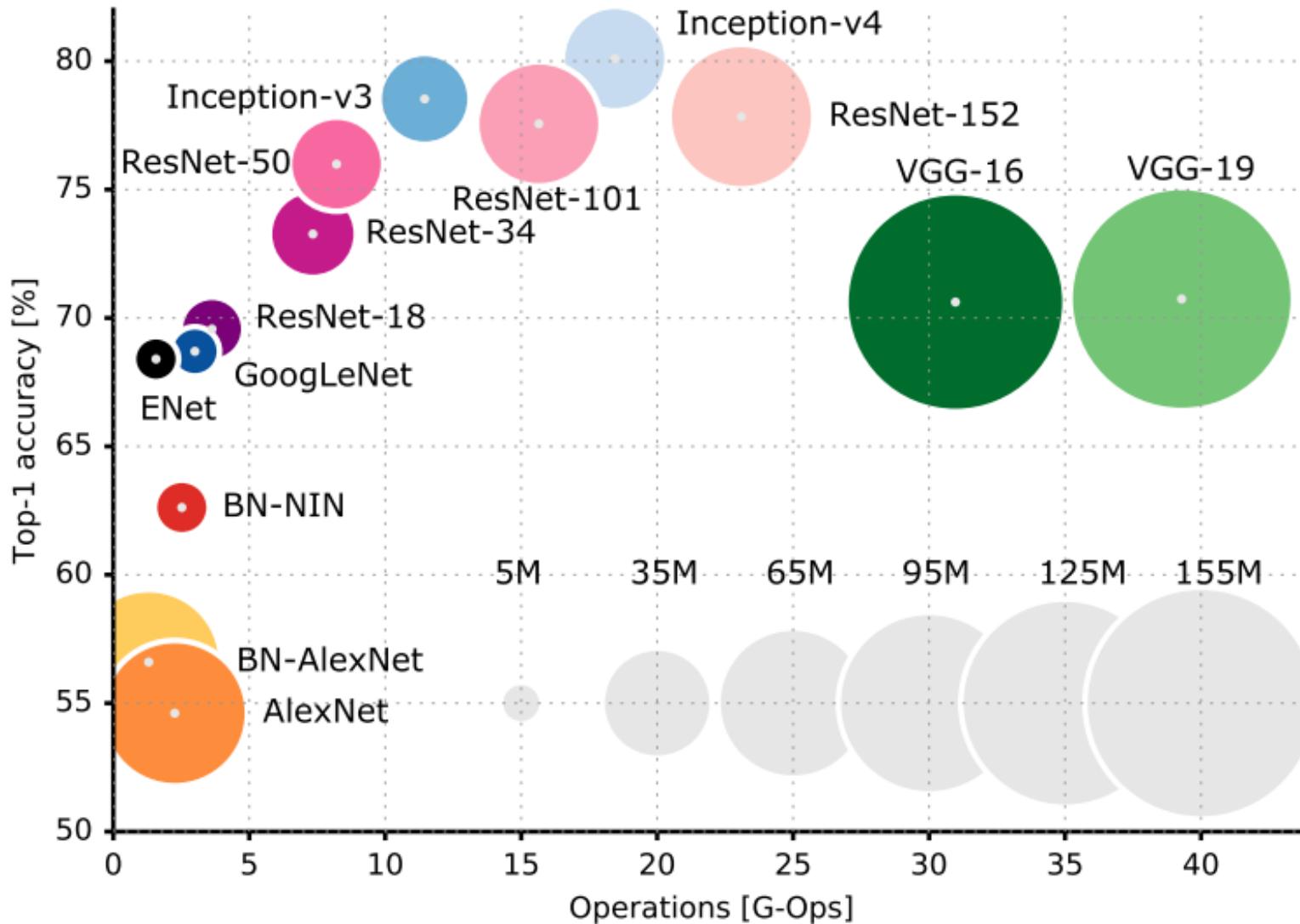
<http://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/>

ImageNet Challenge top-5 error



O tamanho do círculo é
proporcional ao número de
parâmetros da rede

Accuracy vs. efficiency



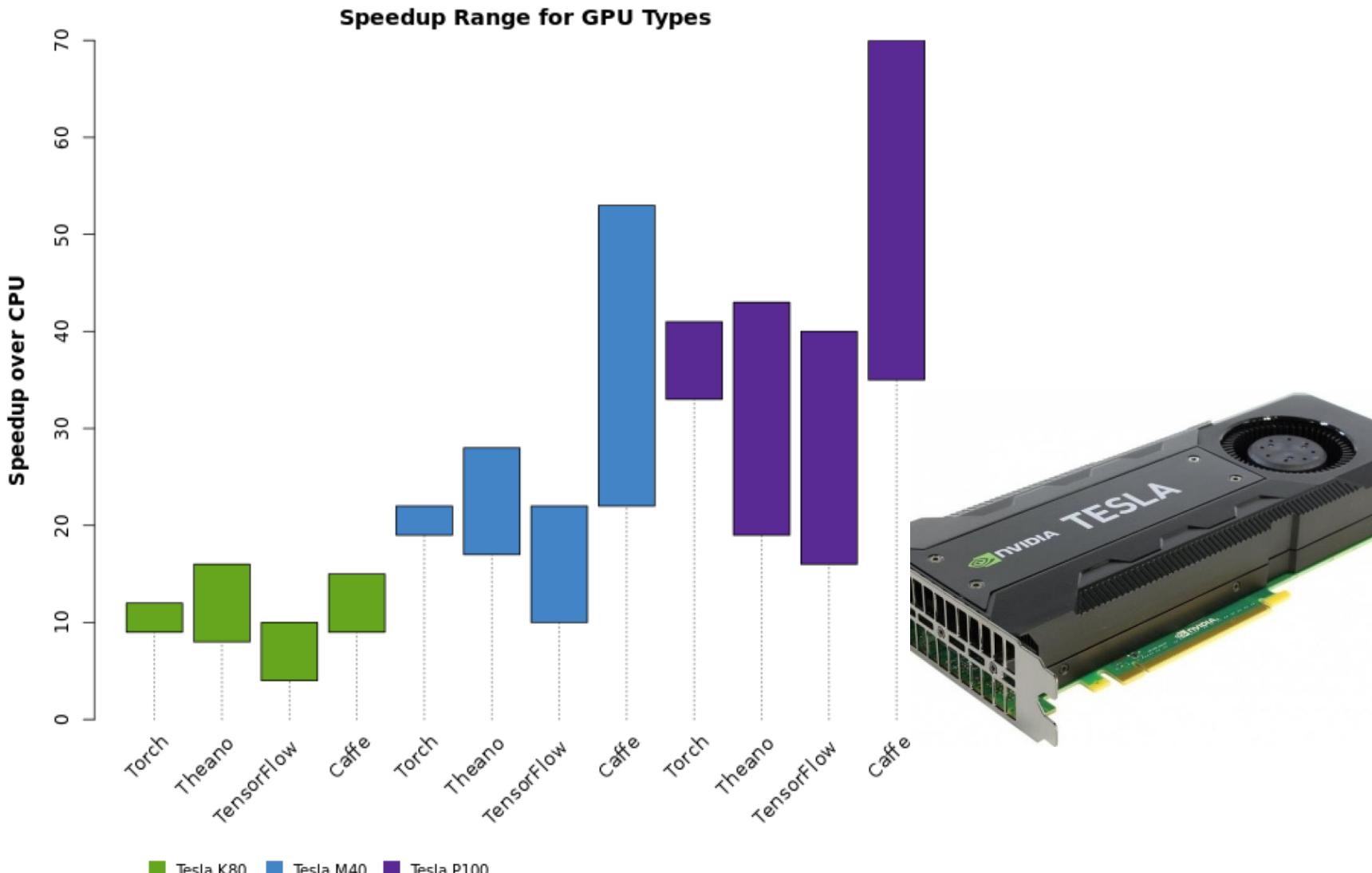
Redes baseadas em Regiões (R-CNN, Fast R-CNN, Faster R-CNN)

- Como esse tipo de processamento pode ser lento, uma vez que muitas regiões são propostas, recomenda-se a utilização placas de vídeo ou GPU (Graphics Processing Unit) para acelerar o processamento, ficando esse tipo de solução conhecida como “Faster R-CNN”



<https://developer.nvidia.com/cuda-gpus>

Placas de vídeo



Sugestão de Hardware para IA

- Corsair Chassis, Carbide Series Air 540 MT ATX 2x3.5 (\$127)



- Intel Processor, Core i7-6850K 3.6G 15MB 140W (\$360)



- MSI X99A GodLike Gaming Carbon LGA 2011-v3 (\$450)



- Kingston 128GB PC4-24000 288-pin DDR4 SDRAM UDIMM Kit (\$1200)



Sugestão de Hardware para IA

- Samsung 1TB 960 EVO PCIe NVMe M.2 Internal Solid State Drive (\$430)
- 2 x Seagate 6TB IronWolf SATA 6Gb 7200RPM 3.5" HDD, 256MB Cache (\$250)
- Corsair Hydro Series H80i v2 CPU Cooler (\$93)
- 4 x EVGA GeForce GTX TITAN X 12GB GAMING (\$799)



Sugestão de Hardware para IA

Item	Valor U\$	Quant	Total
Chassi	\$ 127,00	1	\$ 127,00
CPU i7	\$ 360,00	1	\$ 360,00
Board	\$ 450,00	1	\$ 450,00
RAM	\$ 1.200,00	1	\$ 1.200,00
SSD	\$ 430,00	1	\$ 430,00
HDD	\$ 250,00	2	\$ 500,00
Cooler	\$ 93,00	1	\$ 93,00
GPU	\$ 799,00	4	\$ 3.196,00
	Total Geral		\$ 6.356,00

5. Práticas no Laboratório

Utilizando o Tensorflow com Python

➤ Para desenvolver projetos com o Tensorflow seguimos os passos abaixo:

1. Definimos a arquitetura da rede (**blueprint**), incluindo:

- Variáveis de entrada, saída e pesos
- Camadas da rede (entrada, escondidas, saída)
- Métodos de treinamento e de teste da rede

2. Criamos uma sessão do Tensorflow (**tf.Session**)

3. Executamos com **run()** os métodos criados utilizando a sessão

4. Verificamos o resultado no **Tensorboard**



Conhecendo o TensorFlow

- Execute os Jupyter Notebooks abaixo:

01_TF_Executando.ipynb

02_TF_Placeholder.ipynb

03_TF_Variaveis.ipynb

04_TF_Tensorboard.ipynb

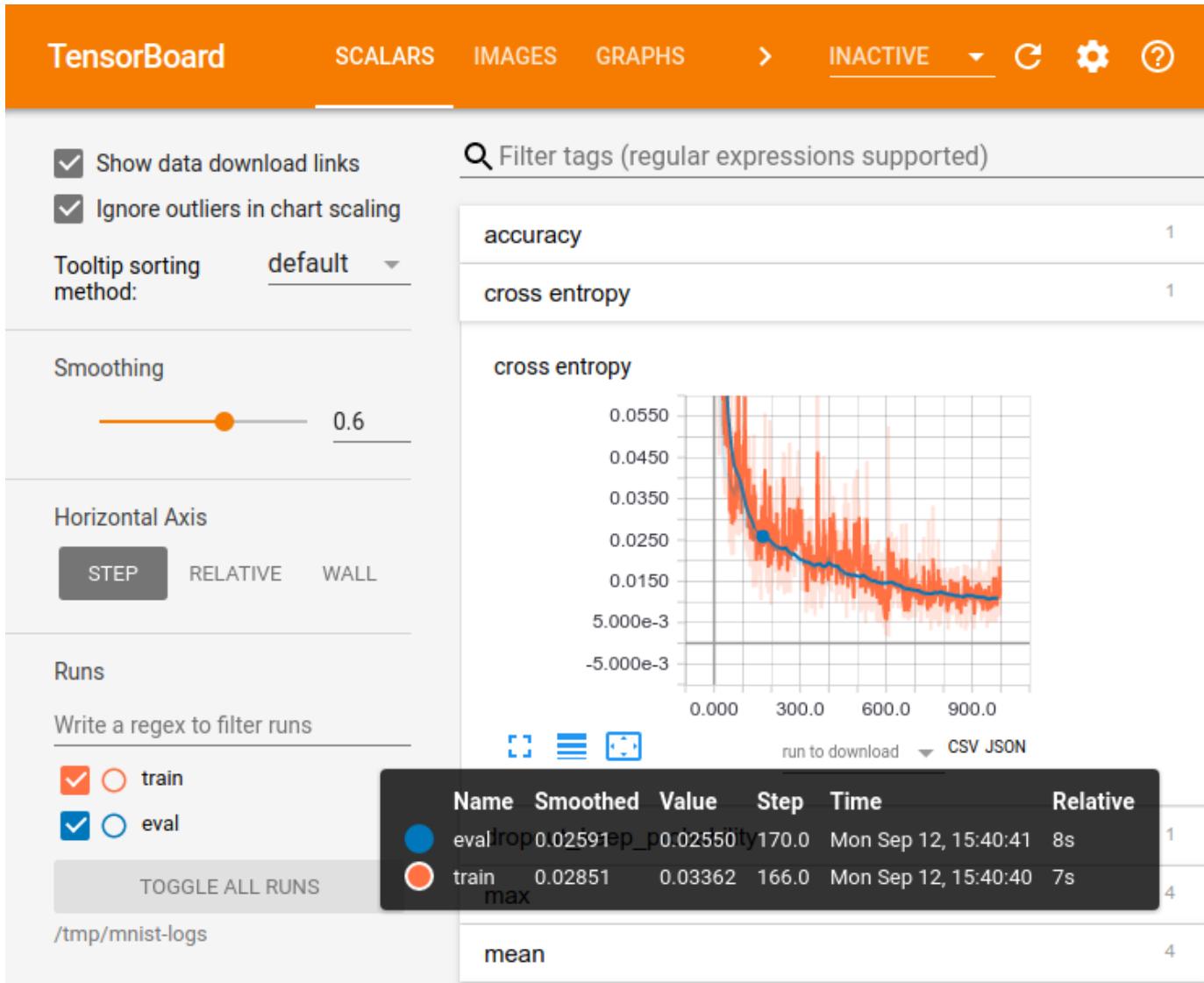
05_TF_Tensores.ipynb

06_TF_Imagens_e_Matrizes.ipynb

Como funciona o TensorFlow

- Para criar e rodar um modelo no TF, é necessário o seguinte:
 1. Dados de treinamento (X e Y)
 2. Variáveis a serem encontradas
 3. Modelo a ser utilizado
 4. Função de custo
 5. Função de Otimização
 6. Rodar o modelo com uma sessão do TensorFlow
 7. Ver o resultado no TensorBoard

TensorBoard



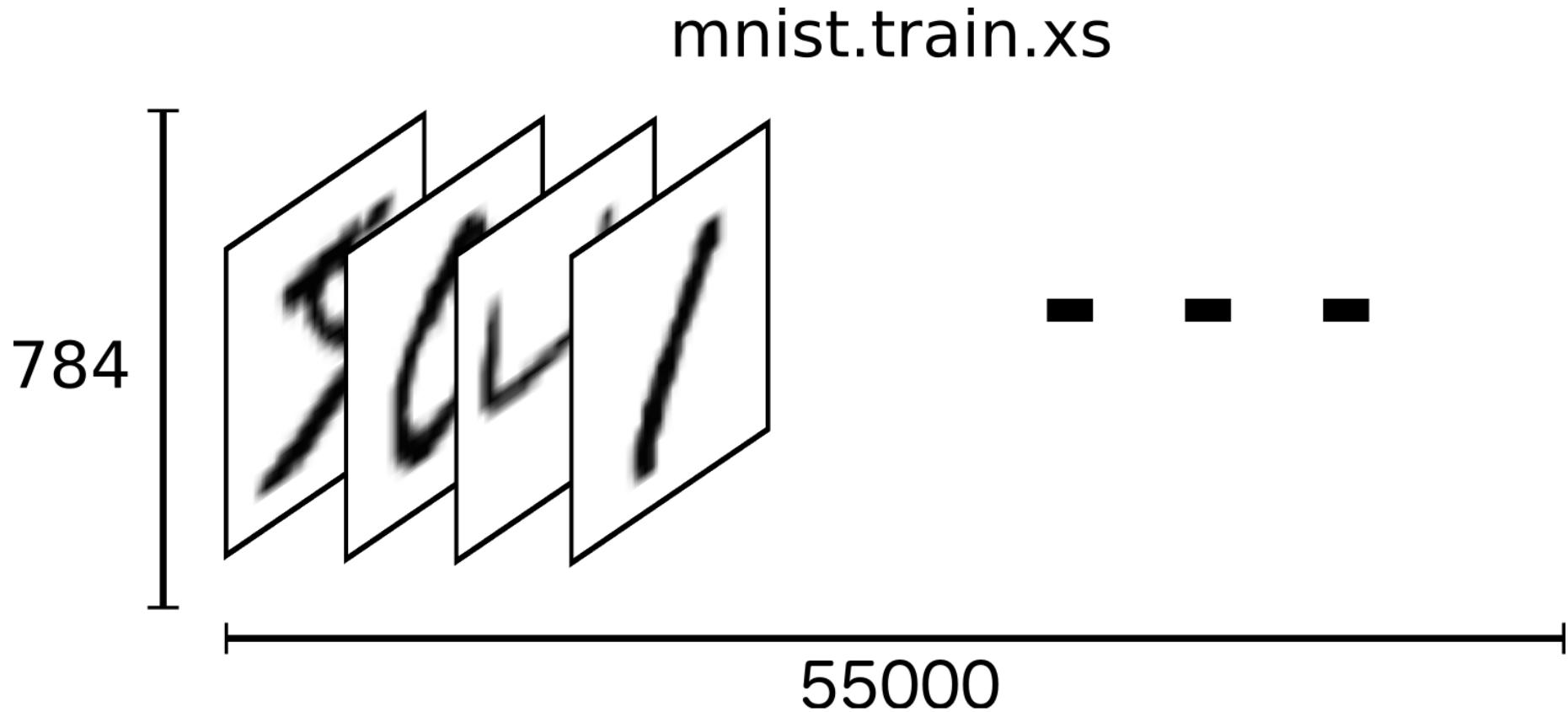
HANDS ON



07_TF_Como_Funciona.ipynb

Base de dados de imagens que vamos utilizar

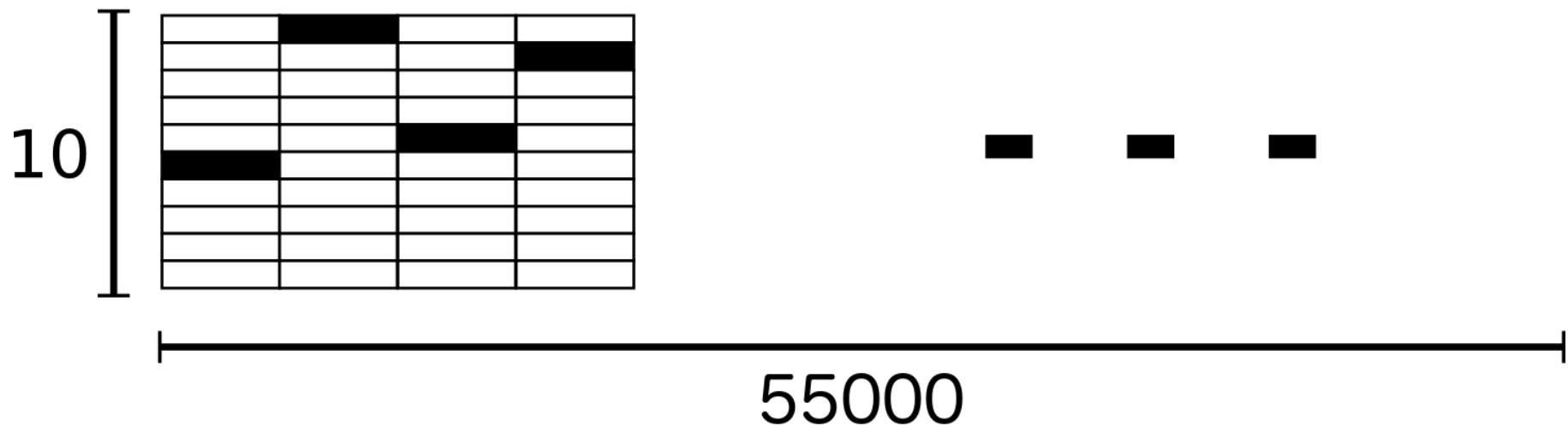
- MNIST é um dataset de dígitos manuscritos com 55 mil exemplos



Base de dados de imagens que vamos utilizar

- Os dígitos são de “0” a “9”, correspondendo portanto a 10 classificações possíveis

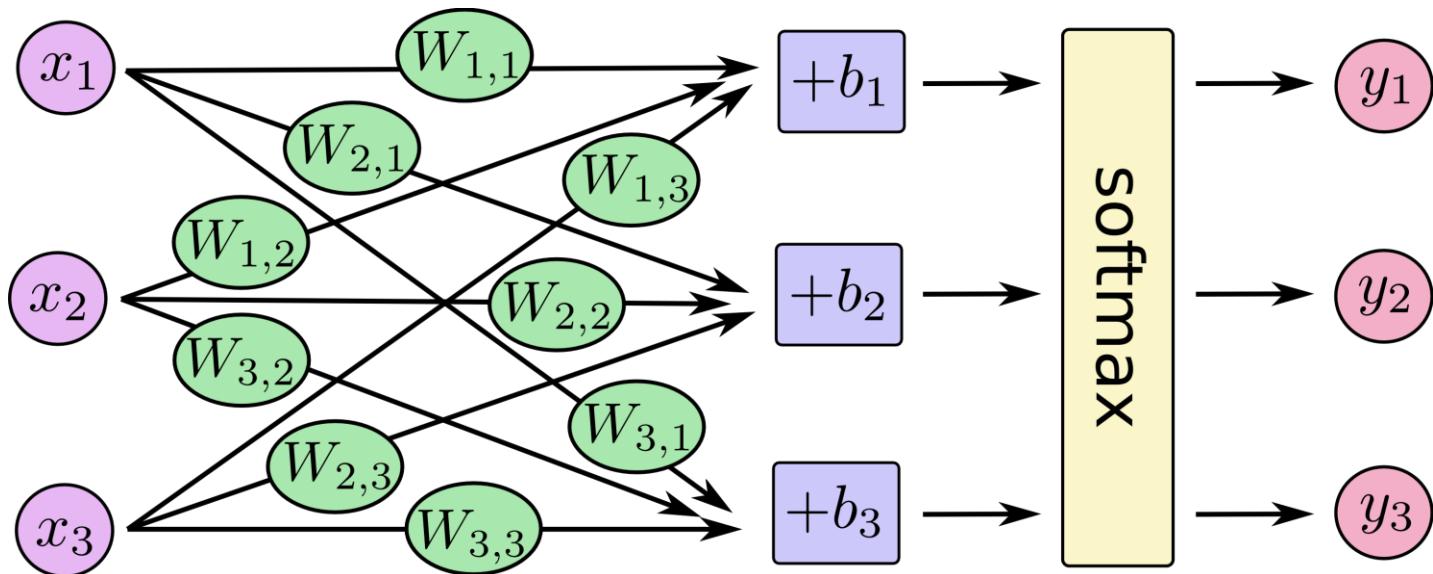
mnist.train.ys



Utilizando Redes Neurais

- Vamos agora utilizar uma rede neural MLP para resolver a classificação de imagens MNIST

- Rede:



- Matriz:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \text{softmax} \left(\begin{bmatrix} W_{1,1} & W_{1,2} & W_{1,3} \\ W_{2,1} & W_{2,2} & W_{2,3} \\ W_{3,1} & W_{3,2} & W_{3,3} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \right)$$

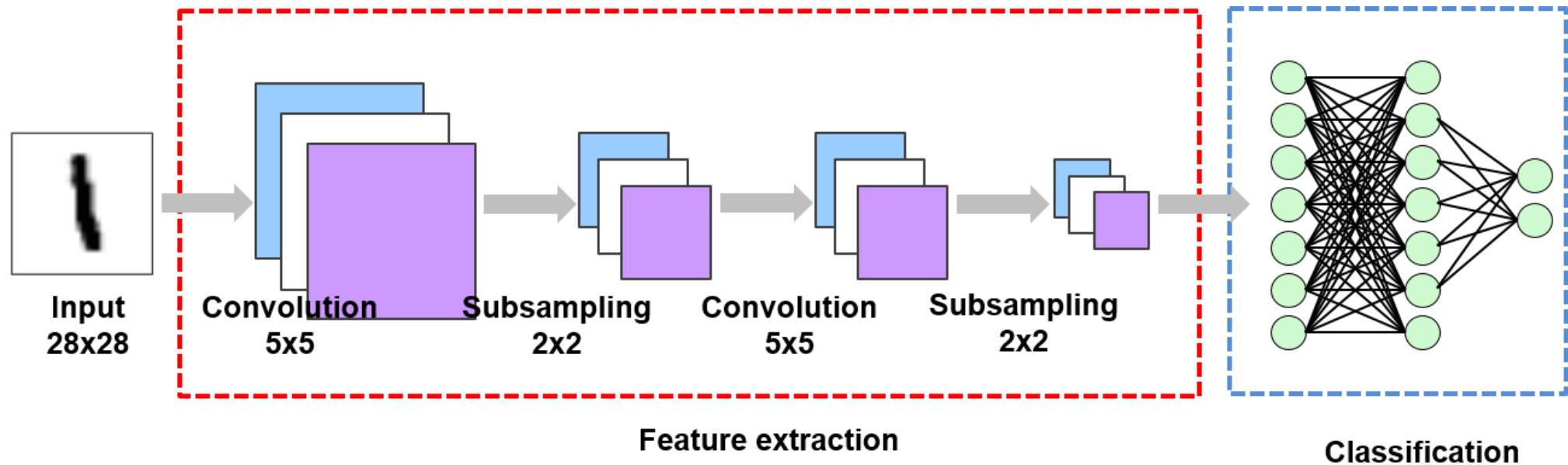
HANDS ON



08 TF MLP MNIST.ipynb

Uso de CNN para classificação MNIST

- Finalmente, vamos começar a utilizar redes neurais convolucionais (CNN)



HANDS ON



09 TF CNN MNIST.ipynb

Obrigado!

José Humberto Cruvinel
Contato: jose.junior@prof.unibh.br
<https://www.facebook.com/jhcruvinel>