

DEEP LEARNING

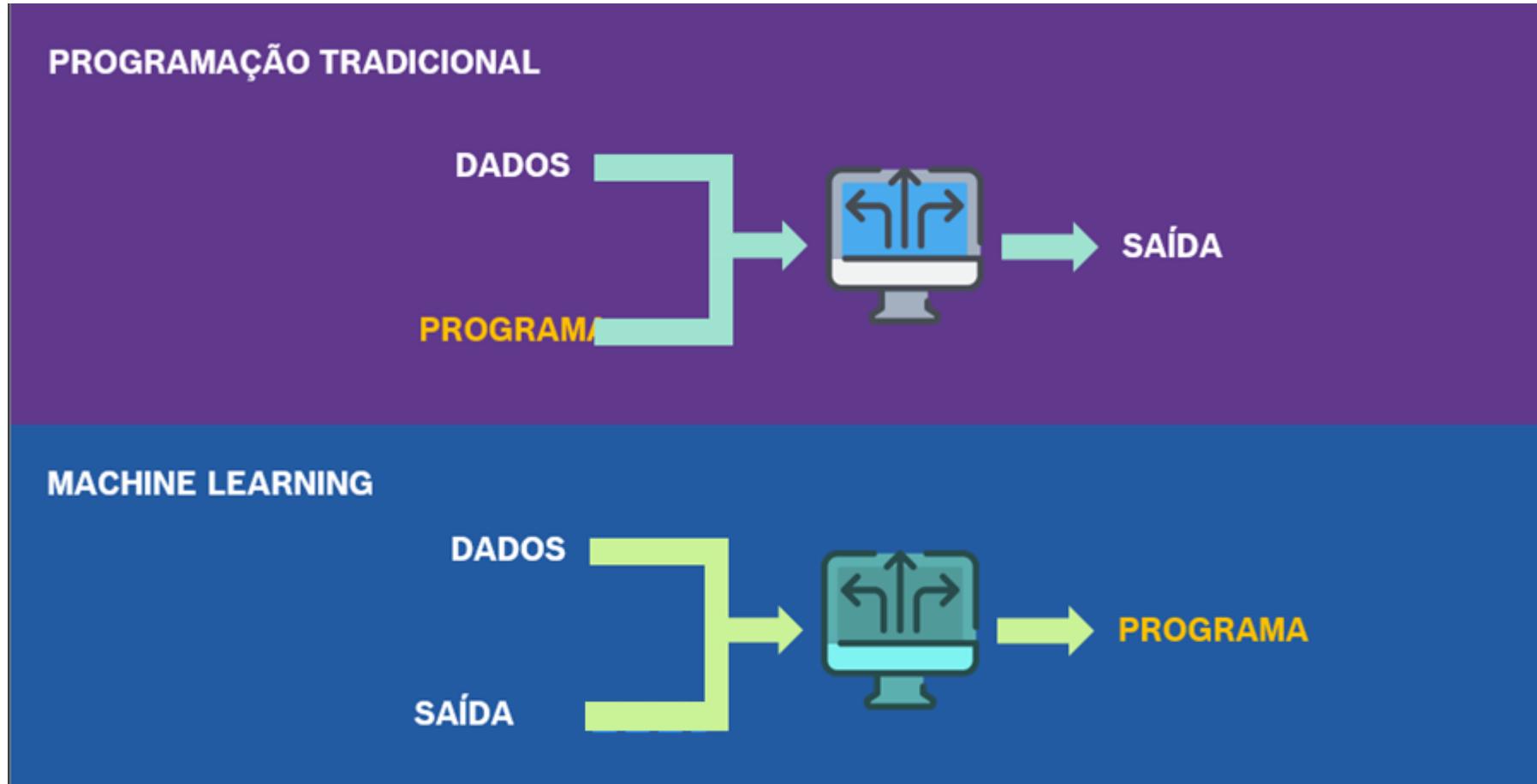
PT-1

Agenda

- **Introdução**
- **Classificação**
- **CNN**
- **Frameworks for Deep Learning (Tensor Flow, Keras)**
- **Data augmentation**
- **Batch Generator**
- **TensorBoard**

INTRODUÇÃO

Introdução Machine Learning



Introdução Machine Learning

“Um programa de computador se diz aprender por experiência E em respeito a uma tarefa T e alguma medida de performance P, se a performance em T, medida por P, melhorar com experiência E.”

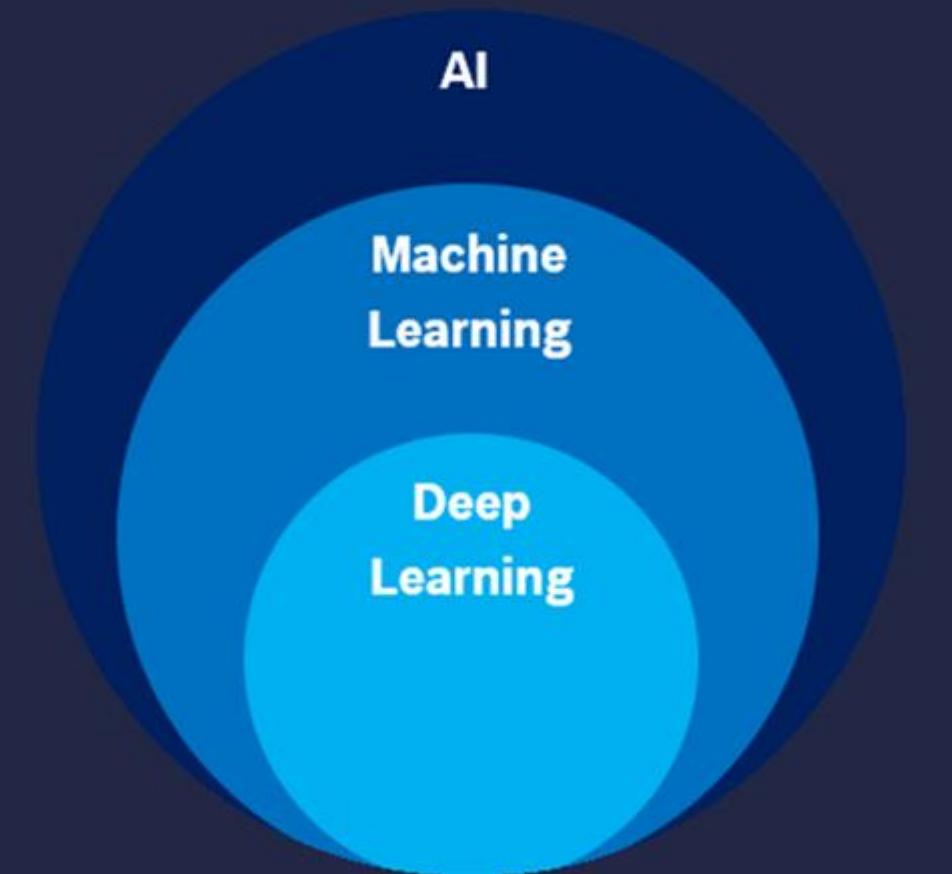
(Tom Mitchell, 1997)

Introdução Machine Learning

**APRENDIZADO DE
MÁQUINA**

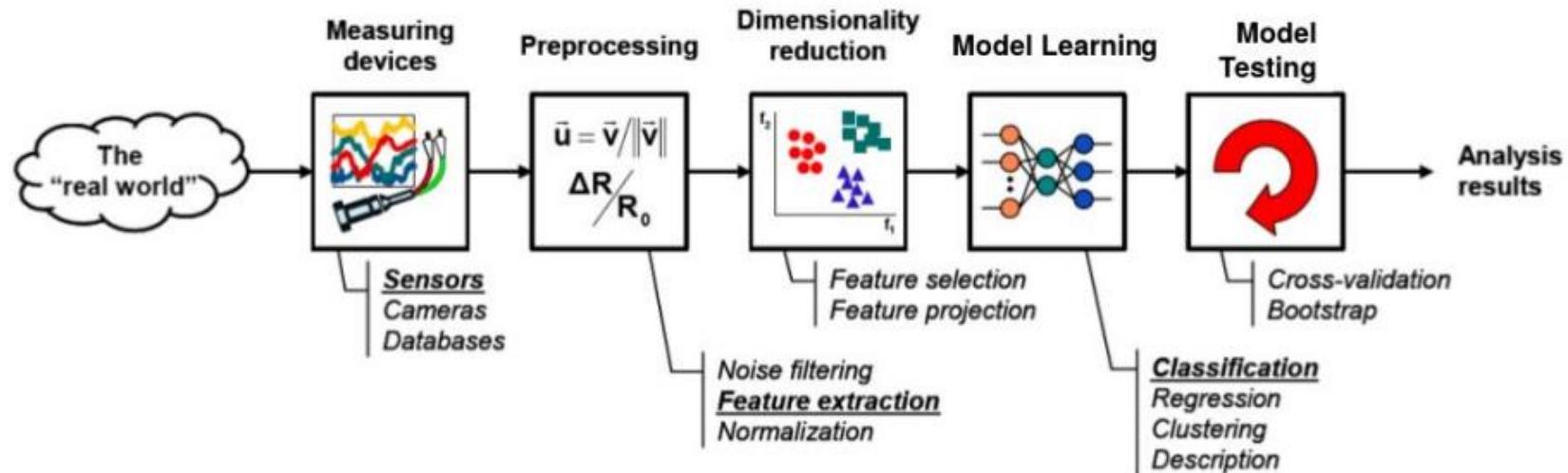
=

INTELIGÊNCIA ARTIFICIAL?



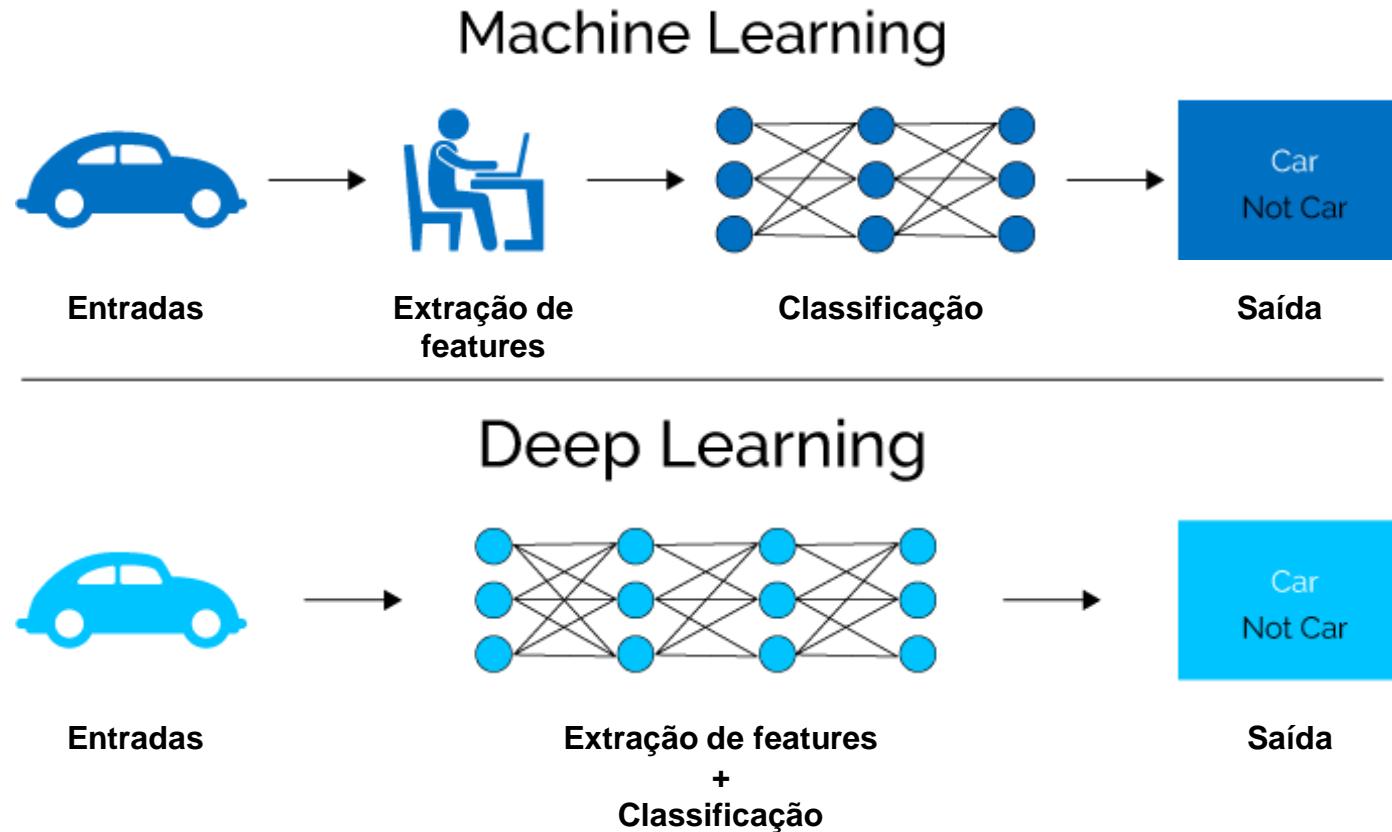
Introdução Machine Learning

O Processo de Aprendizado



Introdução

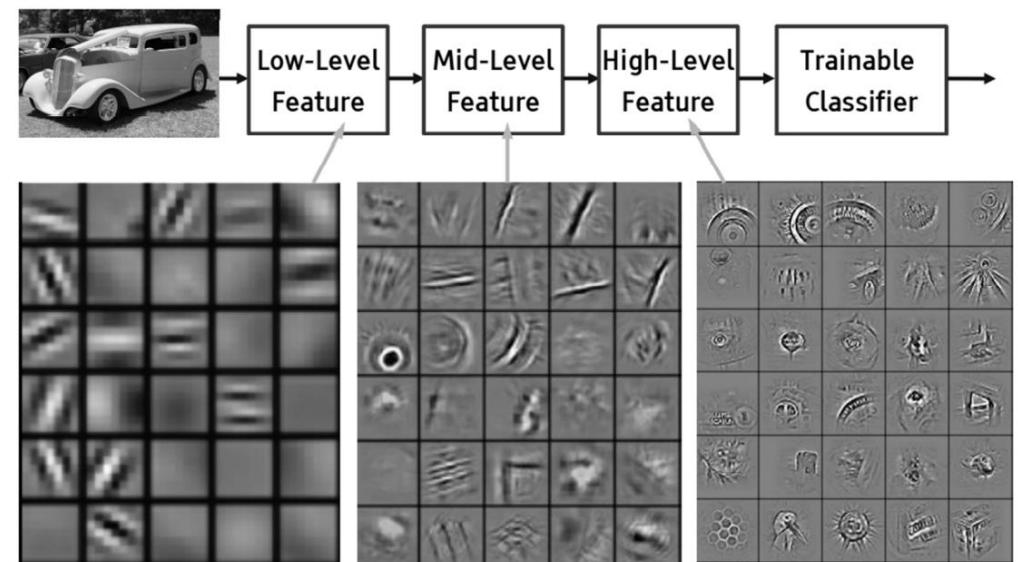
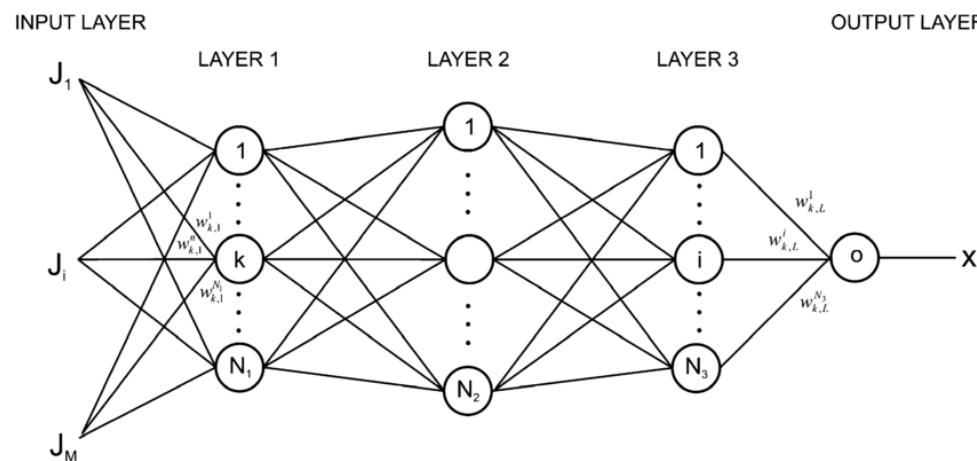
Machine Learning vs Deep Learning



Introdução

Machine Learning vs Deep Learning

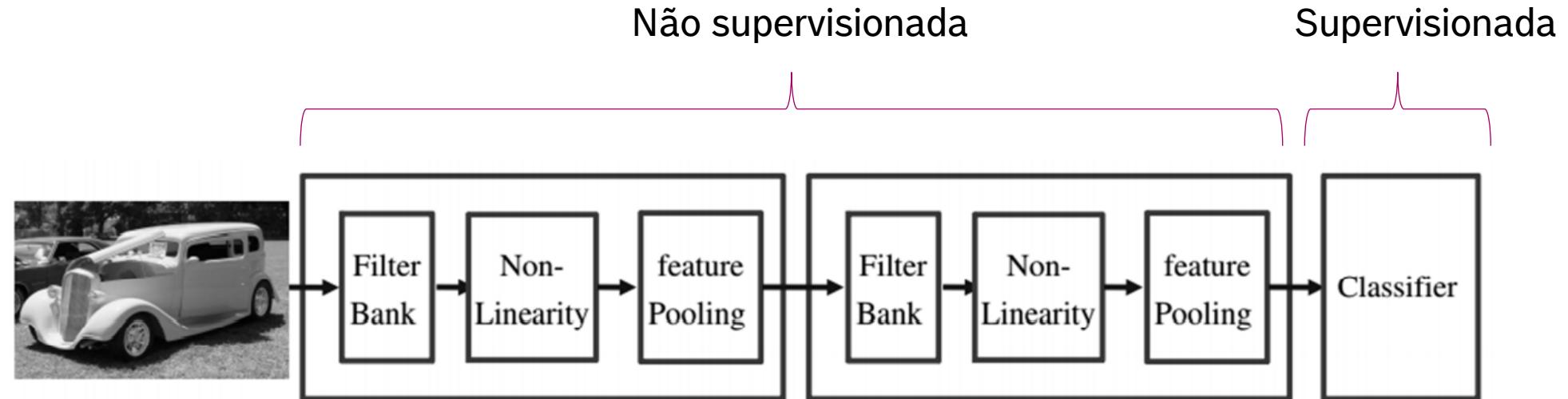
Machine Learning vs Deep Learning



Introdução

Machine Learning vs Deep Learning

Machine Learning vs Deep Learning



- As camadas de extração de características são treinadas de modo não-supervisionado de modo a descobrir características gerais do espaço de entrada
- As características finais alimentam uma camada supervisionada
- A rede inteira é então ajustada de modo supervisionado

Introdução

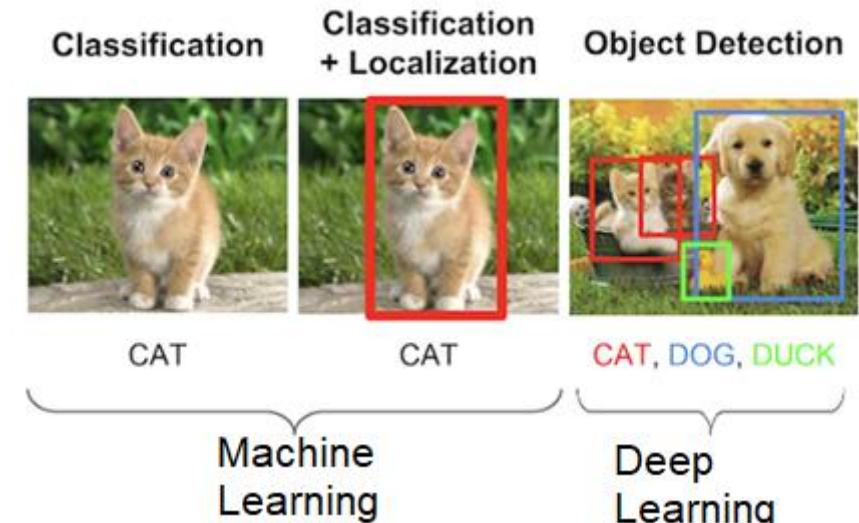
Machine Learning vs Deep Learning

Machine Learning vs Deep Learning

- Intervenção Humana (Ex : Feature Engineering)
- Hardware
- Tempo de Processamento
- Abordagem
- Aplicações

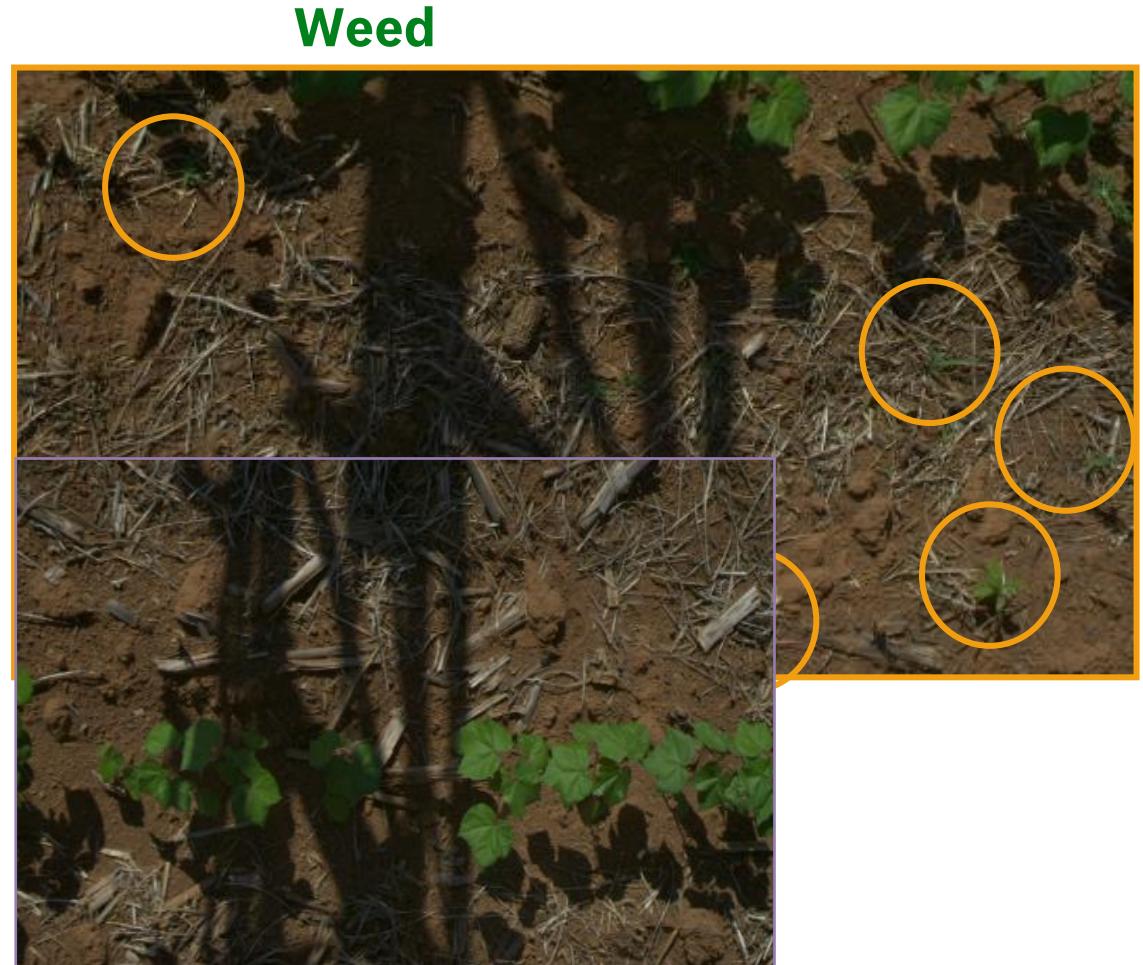
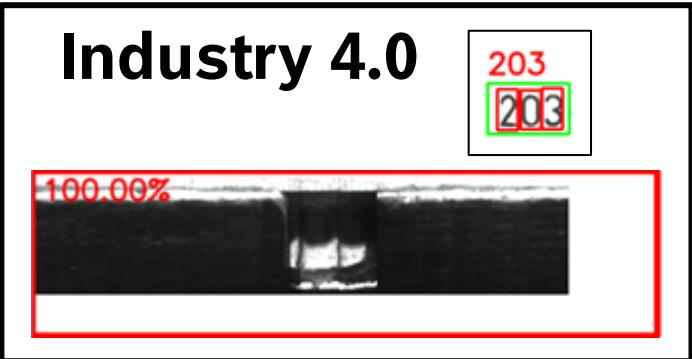
Machine Learning : previsão de valores, identificação de spam em e-mails, sistemas de recomendação para plataformas de streaming, etc...

Deep Learning : Detecção de Objetos, Carros autônomos, Segmentação de objetos, Reconhecimento Facial, etc...



Introdução

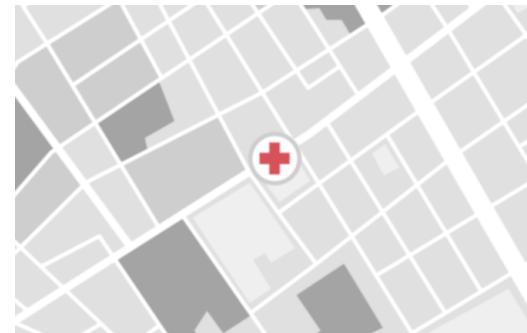
Deep Learning - Aplicações



Aplicações Reais

Deep Learning - Aplicações

- Mobilidade / logística - Veículos autônomos, otimização de rotas, compartilhamento de carros, etc.

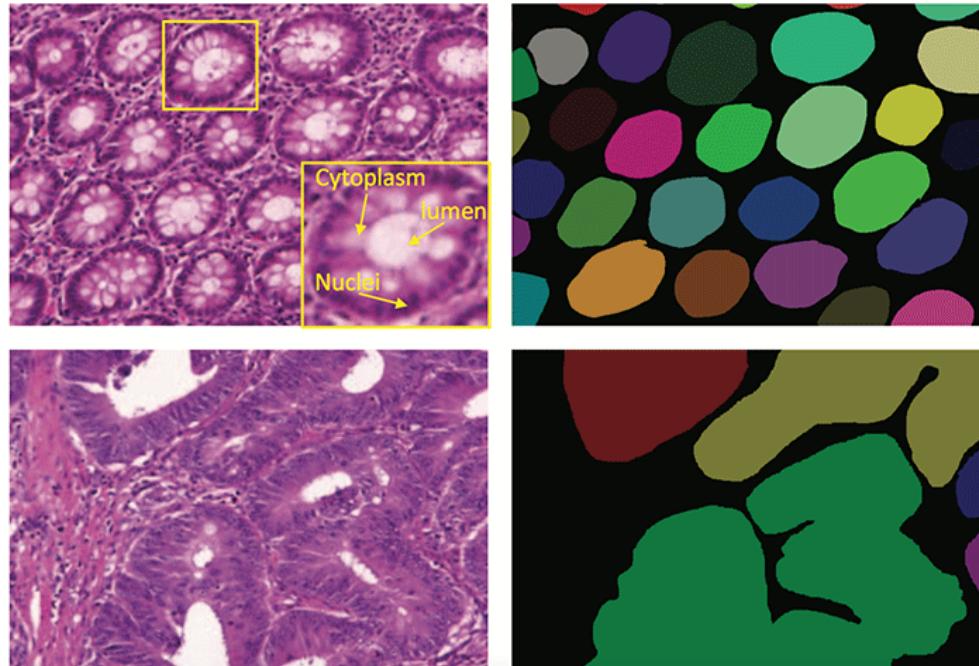


Introdução

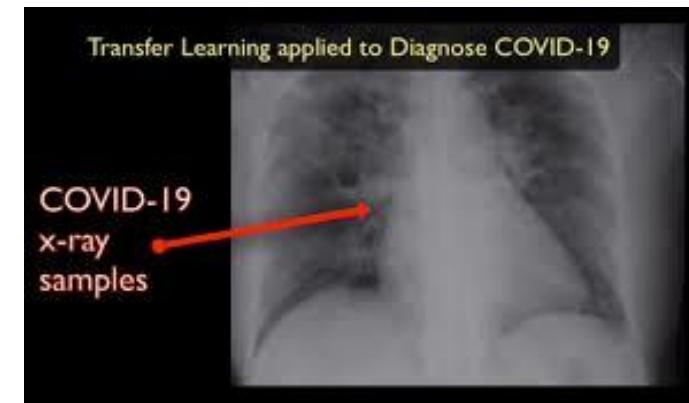
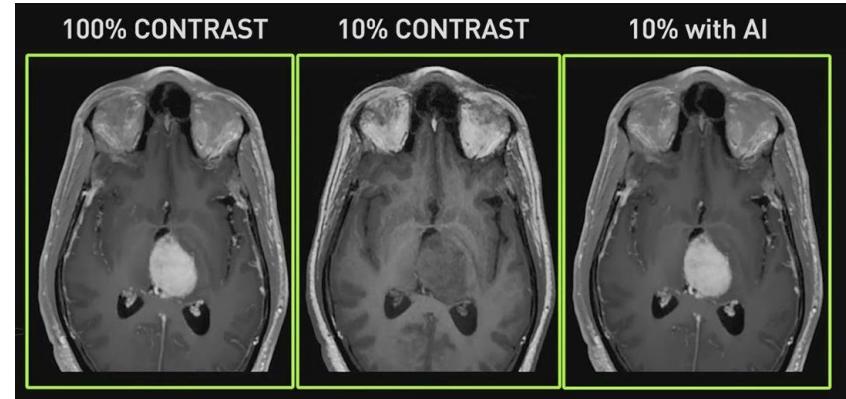
Deep Learning - Aplicações

- Saúde - diagnósticos, previsão de sinais, detecção de anomalias, etc.

GPU-Powered Deep Learning Being Used to Speed Colon Cancer Diagnosis



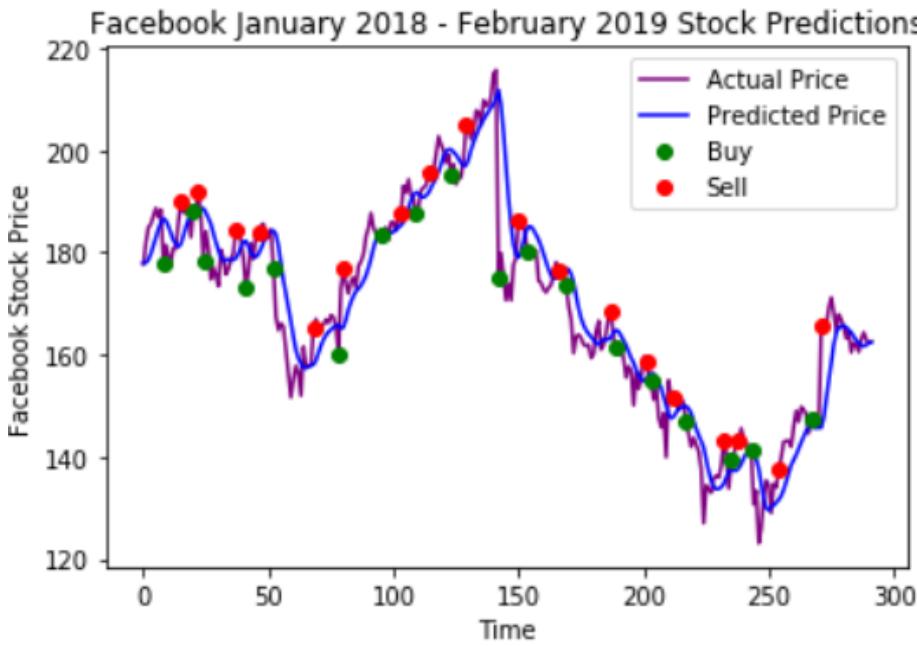
<https://blogs.nvidia.com/blog/2015/12/23/deep-learning-cancer/>



Introdução

Deep Learning - Aplicações

- Finanças - detecção de fraude, previsão de estoque, estimativa de risco, etc.



Introdução

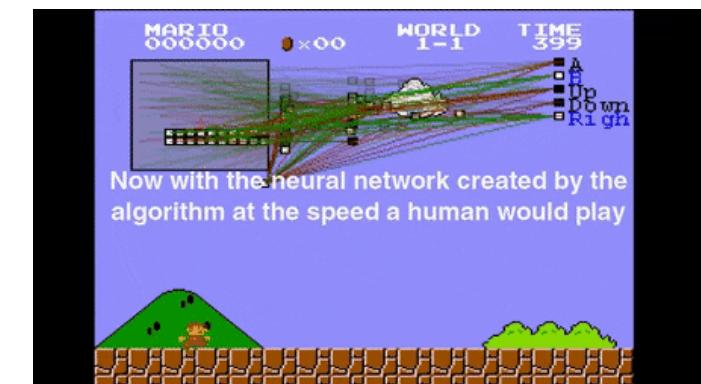
Deep Learning - Aplicações

- Entretenimento - gamificação inteligente, classificação musical, animação em filmes, etc. ([DLSS Nvidia](#))



massive

SketchAI



Introdução

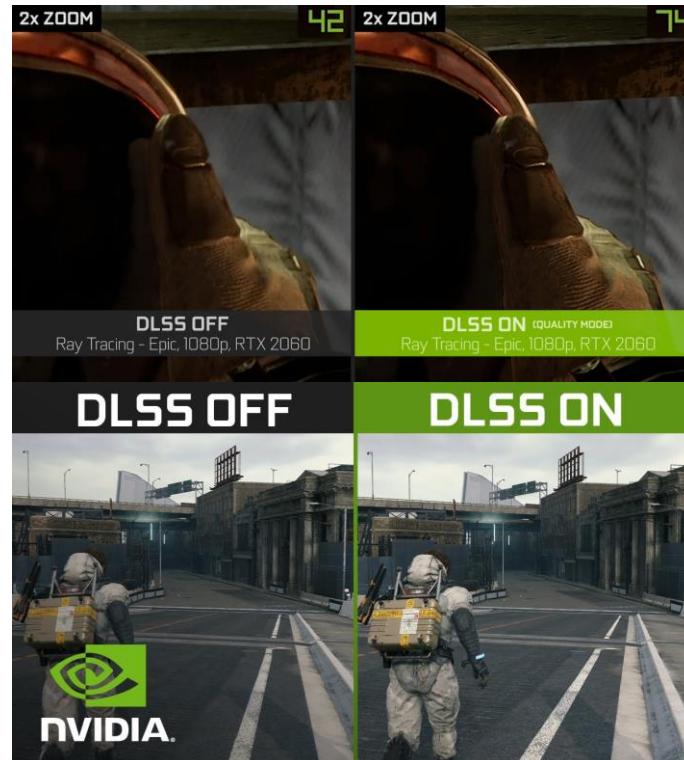
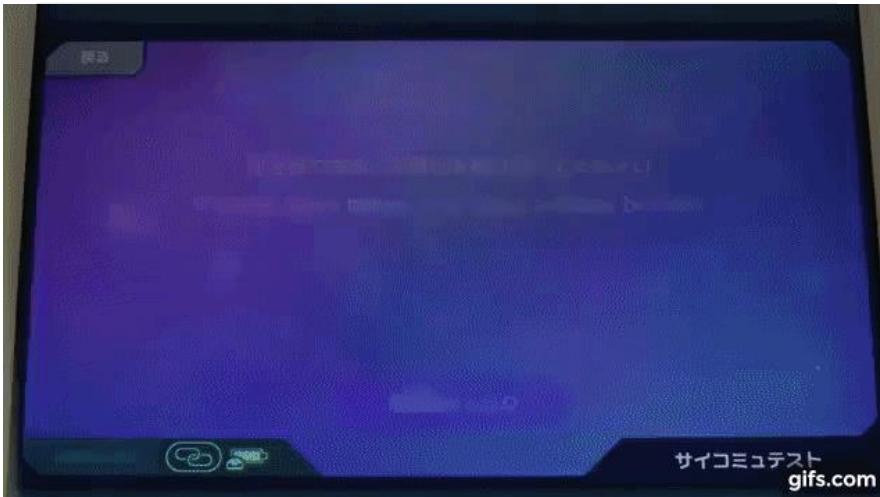
Deep Learning - Aplicações

- Entretenimento - gamificação inteligente, classificação musical, animação em filmes, etc. ([DLSS Nvidia](#))

Japanese Scientists Create Mind Control Tech for Gundam Robot

We're one step closer to life-sized Gundam robots piloted with mind control, just like in the anime.

By Chris Young
November 06, 2020

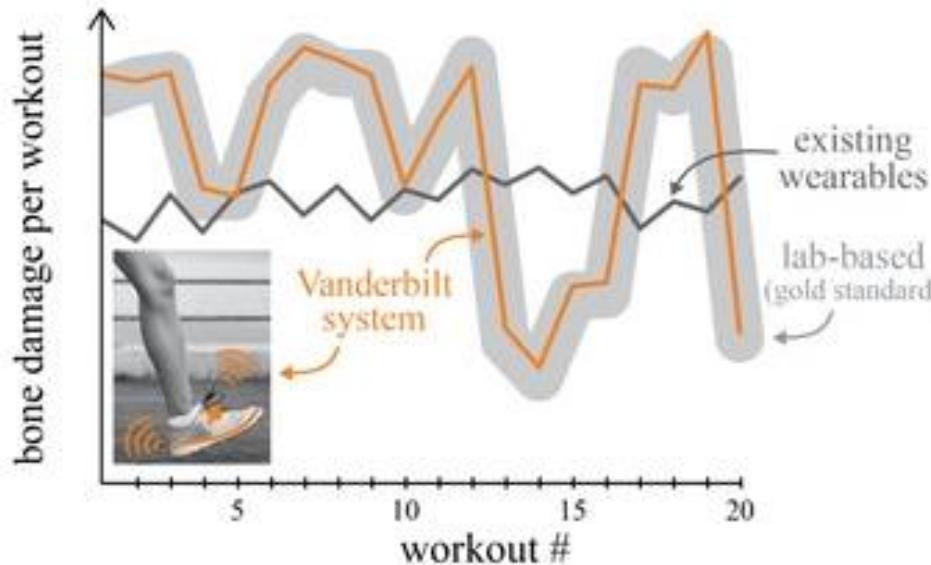


Introdução

Deep Learning - Aplicações

- Esportes - fitness wearables, análise estatística de jogos, etc.

Pesquisadores da Universidade de Vanderbilt apresentaram uma nova abordagem para o design de fitness wearables que utilizam modelos físicos e machine learning. Esta técnica pode ajudar a diminuir esforço nas juntas, resolvendo problemas para ambos atletas amadores e profissionais



Aplicações Reais

- Varejo - publicidade inteligente, preços dinâmicos etc.



Amazon now has 200,000
robots working in its
warehouses

Aplicações Reais

- Agricultura/Pecuária – detecção de ervas daninhas, Pesagem de Gado com câmeras, etc.



Aplicações Reais

- Militar

November 14, 2020 | Topic: Security | Blog Brand: The Buzz | Tags: U.S. Army, Robotics, Missiles, UAVs, Robots

The U.S. Army Wants Heavy Robots Armed with Missiles

This is how America could fight the next great war.

by Kris Osborn



November 14, 2020 | Topic: Security | Region: Europe | Blog Brand: The Buzz | Tags: United Kingdom, British Army, Robotics, Innovation, Autonomous Weapons

The British Army Could Soon be Manned by Robots

“I suspect we could have an army of 120,000, of which 30,000 might be robots.”

by Peter Suciu

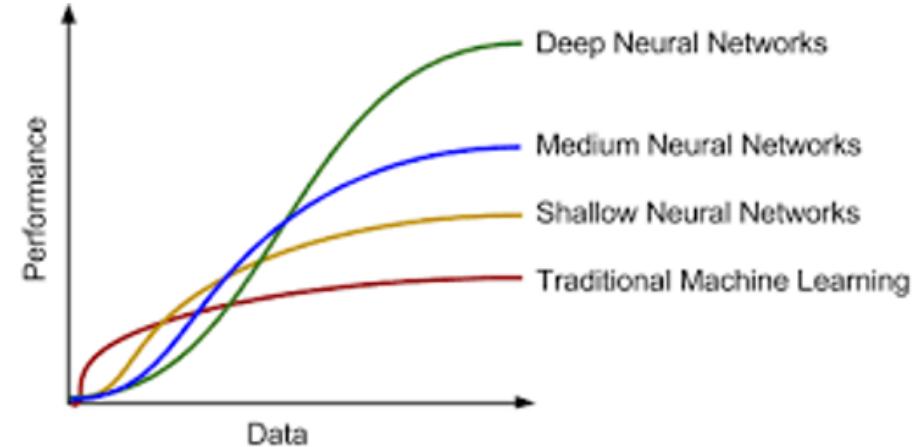


Introdução

Deep Learning - Limitações

- Grande volume de dados

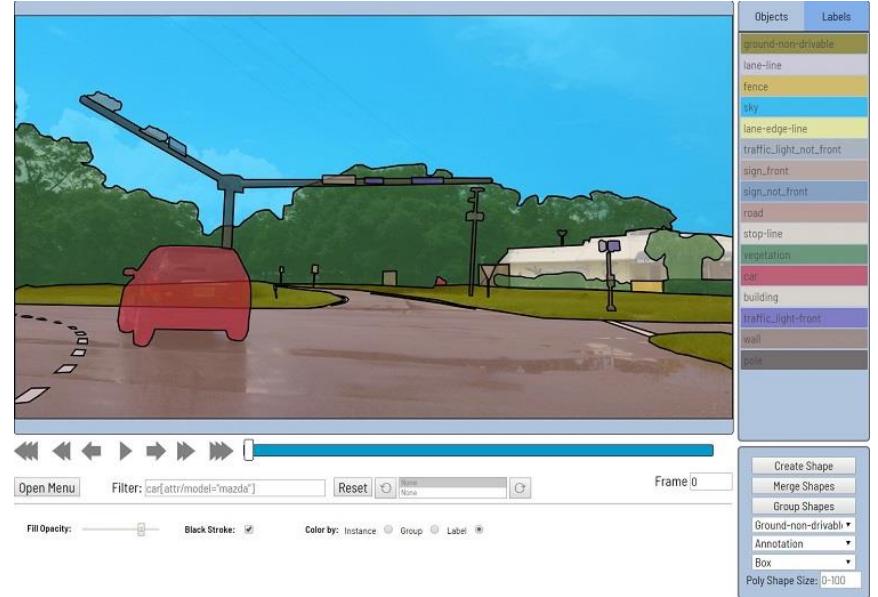
Visão Computacional : Para classificação de imagens utilizando deep learning, a regra é utilizar 1000 imagens por classe, este número pode diminuir drasticamente caso sejam utilizados models pré-treinados.



Introdução

Deep Learning - Limitações

- Ainda é necessária anotação dos dados



Data Annotators Needed for 1 Week Contract

Data Annotator

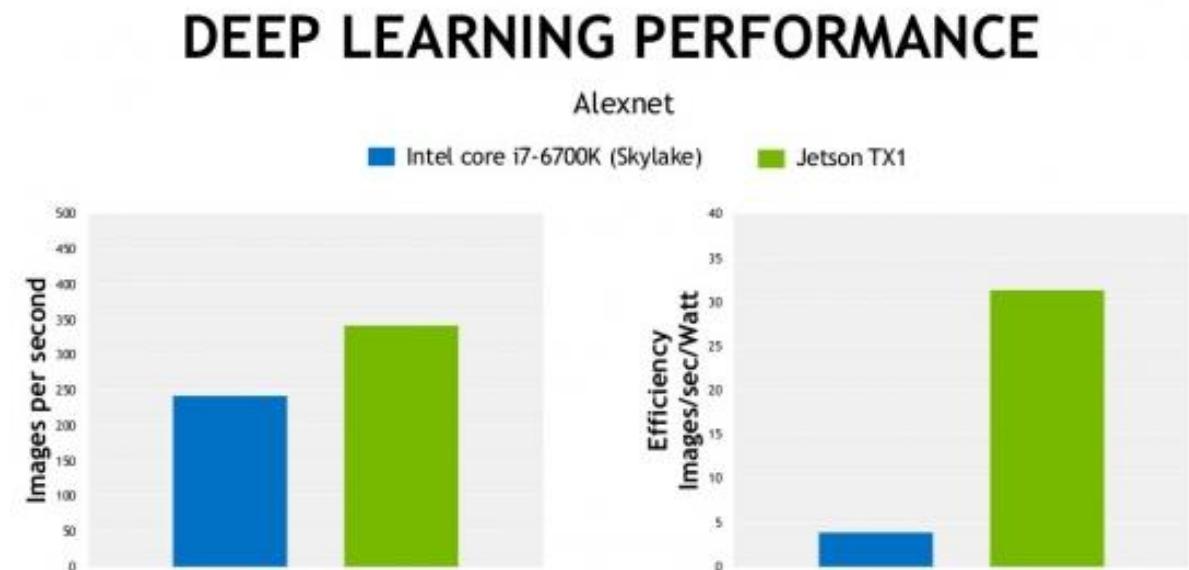
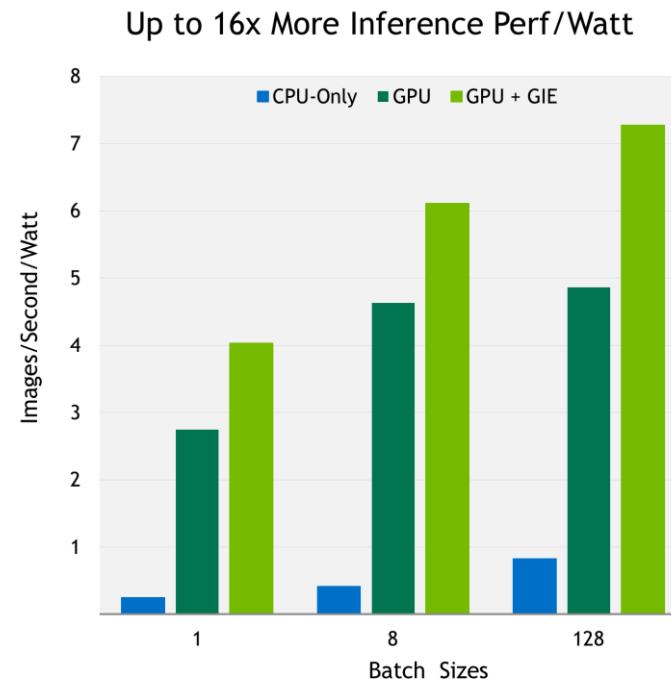
As a Data Annotator at [REDACTED] you will help train our novel deep learning models to better recognize deepfakes, shallowfakes and all sorts of different altered and synthetic media. You will...

Data Annotation Specialist

Introdução

Deep Learning - Limitações

- Alto custo computacional / Tempo de treinamento

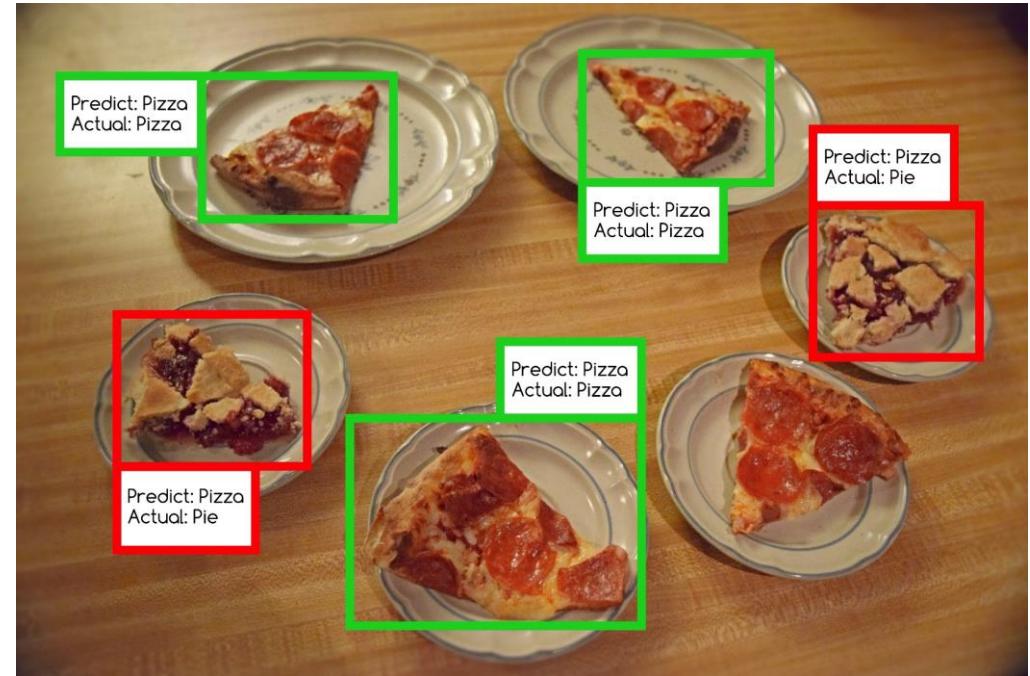
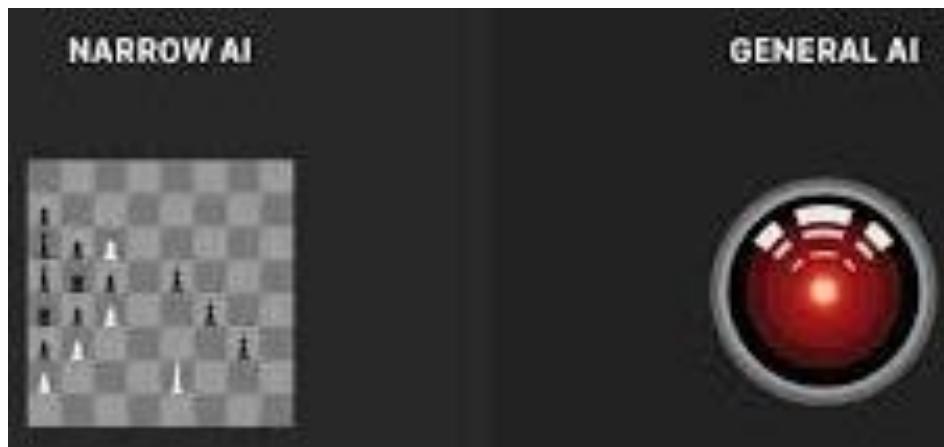


Introdução

Deep Learning - Limitações

- Conhecimento específico (Narrow AI)

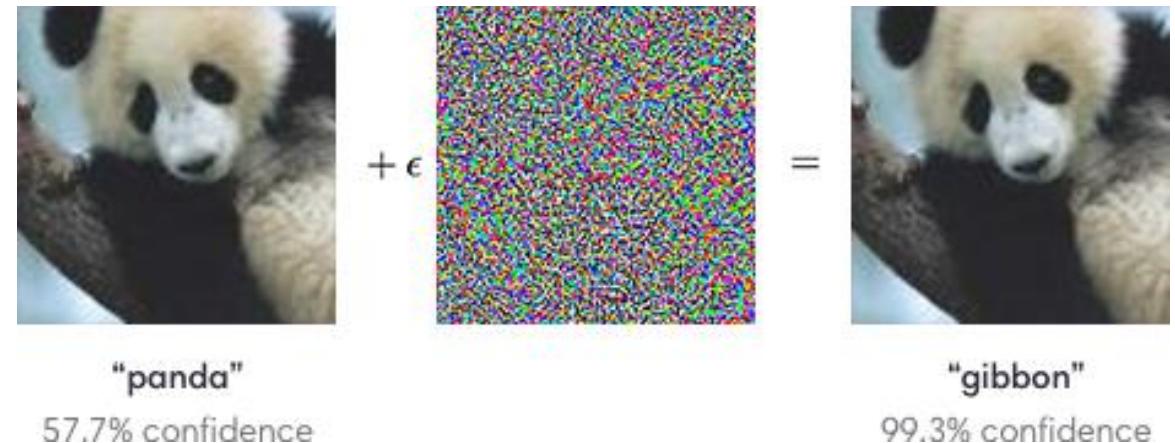
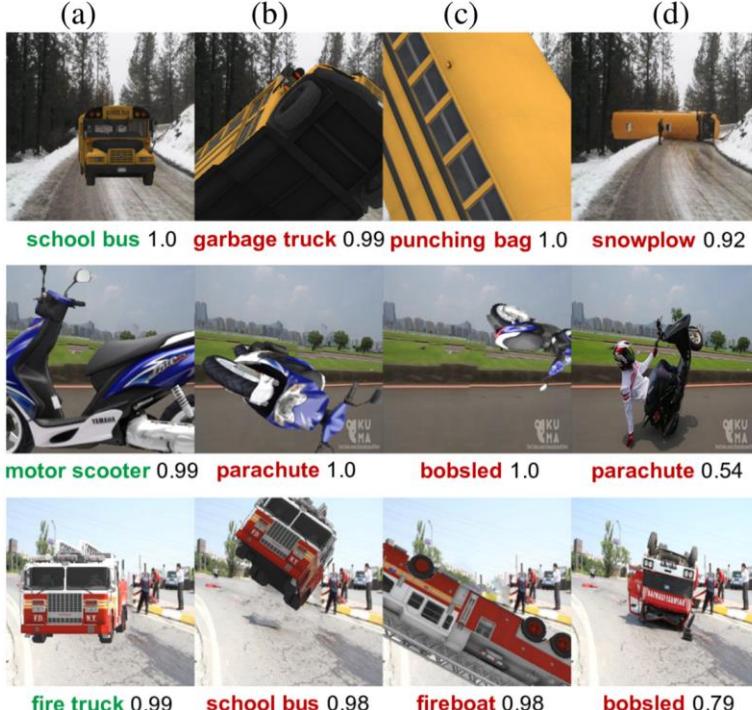
“Deep learning não sabe reagir a situações insólitas”



Introdução

Deep Learning - Limitações

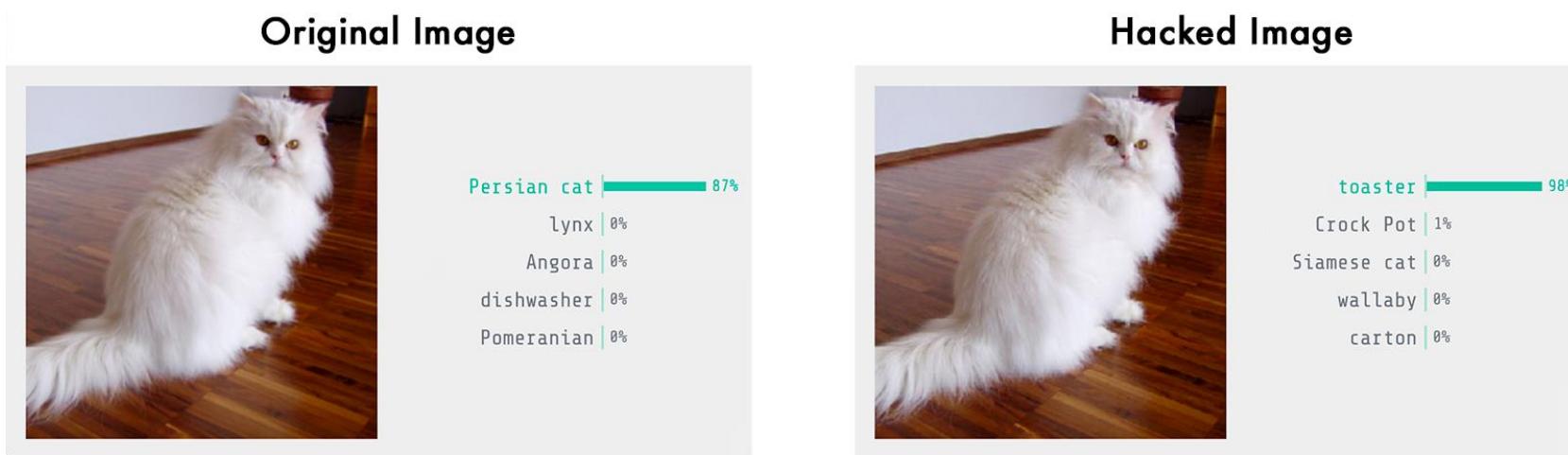
- Sensível a ruídos (luminosidade, oclusão, etc...)



Introdução

Deep Learning - Limitações

- Sensível a ruídos (luminosidade, oclusão, etc...)



Introdução

Deep Learning - Limitações

- Opacidade (BlackBox)



Introdução

Machine Learning vs Deep Learning

- Qual abordagem devo usar?
- Quais critérios devo levar em consideração?
 - Custo Computacional?
 - Tempo de Projeto?
 - Volume de Dados?
 - Qualidade dos Dados?
 - Performance do Modelo acima de tudo?

CLASSIFICAÇÃO

Classificação

Aprendizado supervisionado

Aprendizado não-supervisionado

Nenhum rótulo é dado ao algoritmo de aprendizagem, deixando-o sozinho para encontrar a estrutura em sua entrada **(descobrindo padrões ocultos nos dados)**.

Aprendizado por reforço

Algoritmo para **agir em processos de decisão desconhecidos e não especificados, guiados apenas por recompensas e punições.**

Aprendizado supervisionado

O computador é apresentado com exemplos de entradas e suas saídas desejadas, fornecidas por um "professor", e **o objetivo é aprender uma regra geral que mapeia entradas em saídas.**

Classificação

REDUÇÃO DE DIMENSIONALIDADE

ele simplifica as entradas mapeando-as em um espaço dimensional inferior (LDA, PCA, codificadores automáticos, detecção de compressão, etc.)

CLUSTERIZAÇÃO

a **tarefa de agrupar** um conjunto de objetos de tal forma que os objetos no mesmo grupo (**cluster**) sejam mais semelhantes entre si do que em grupos diferentes

REGRESSÃO

As saídas são **contínuas em vez de discretas**. O problema de prever uma saída de quantidade contínua para uma determinada entrada

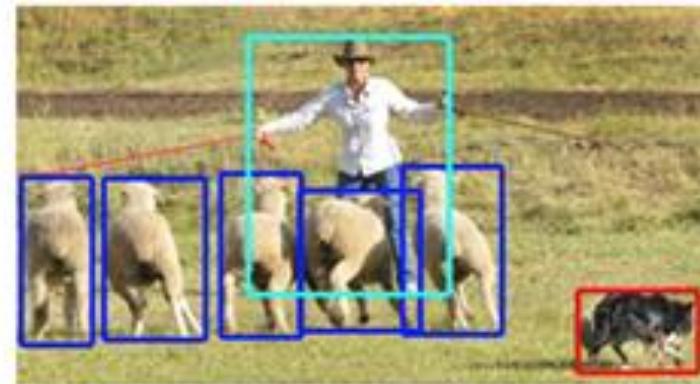
CLASSIFICAÇÃO

o processo de previsão da classe de dados. As classes às vezes são chamadas de alvos / rótulos ou categorias

Classificação



(a) classification

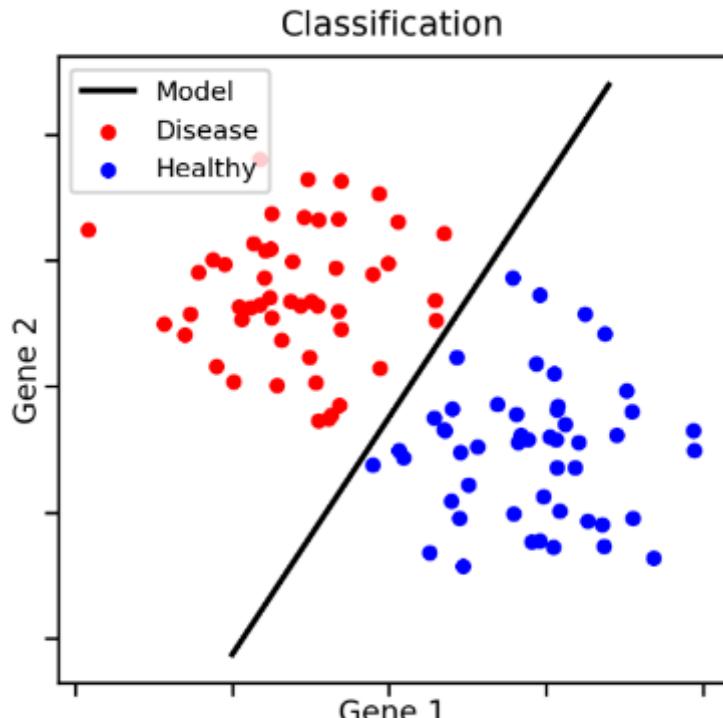


(b) detection

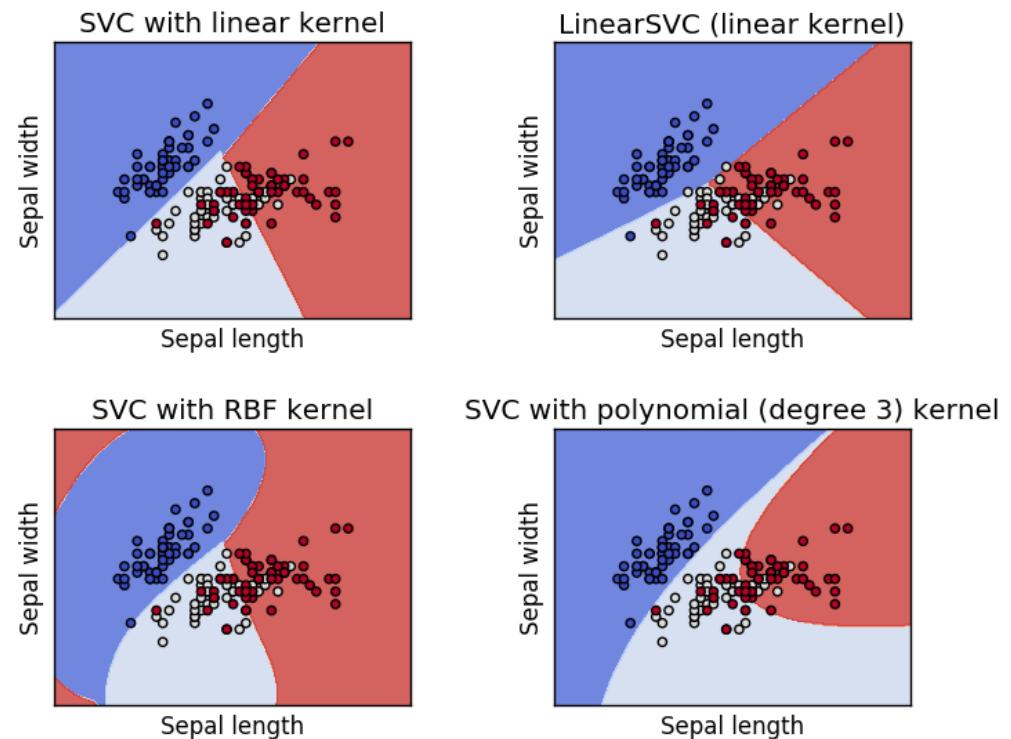


(c) segmentation

Classificação



<https://bit.ly/2MYXmaR>



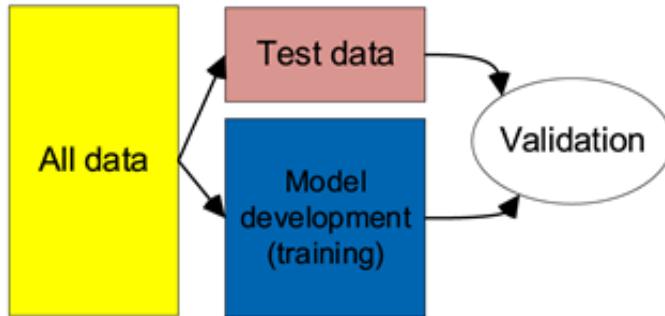
<https://bit.ly/2A8dTEu>

Classificação Exemplo

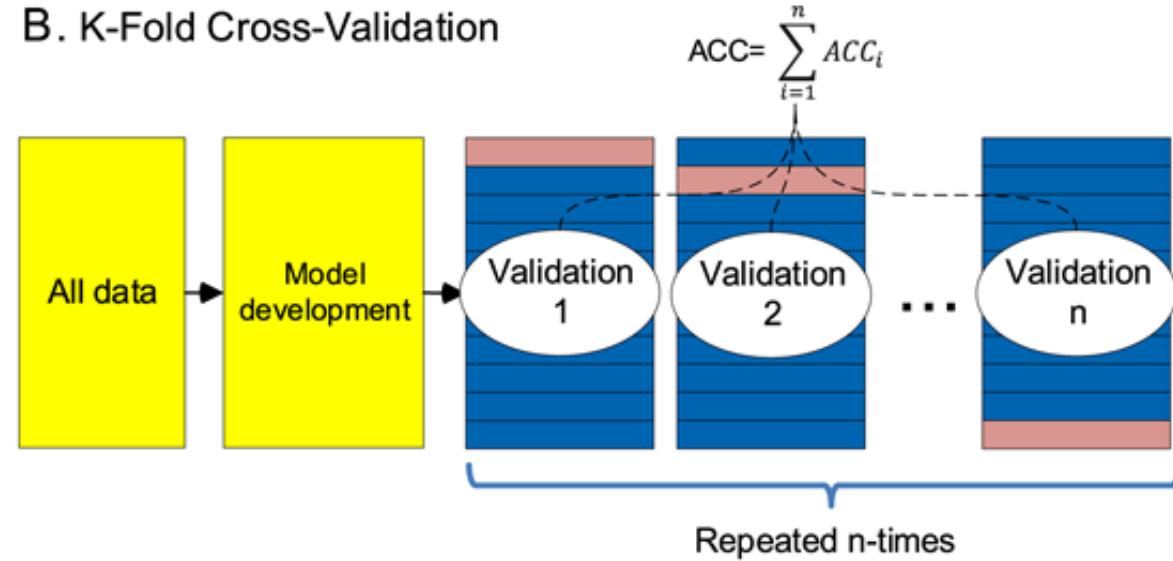
Classificação

Divisão de Dataset

A. Train/Test Split



B. K-Fold Cross-Validation



Classificação Métricas de Desempenho

- Consiste no conjunto de métricas utilizadas para avaliar o desempenho de um algoritmo.
- A avaliação de desempenho, é uma etapa muito importante, dependendo do problema e do algoritmo utilizado pode ser uma tarefa difícil, e deve ser feita tomando o devido cuidado.
- É recomendado, em problemas mais complexos (a até mesmo nos mais simples), a utilização de mais de uma métrica de desempenho, a fim de ter um resultado mais fiel ao comportamento do algoritmo (ou algoritmos) em questão.
- Métricas diferentes podem representar noções diferentes para o mesmo resultado, a fim de, facilitar o processo de tomada de decisão.

Classificação Métricas de Desempenho

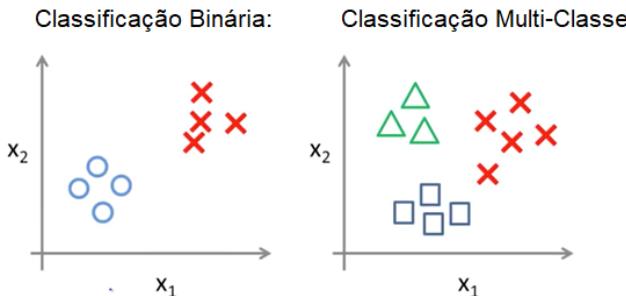
As métricas mais comuns utilizadas na avaliação e comparação de algoritmos:

- Métricas de Erro.
- Custo Computacional.
- Convergência.

Classificação Métricas de Desempenho - Erro

As métricas de erro consistem na avaliação do resultado final do algoritmo em relação a uma dada solução, e a mesma deve ser selecionada de acordo com a característica do problema.

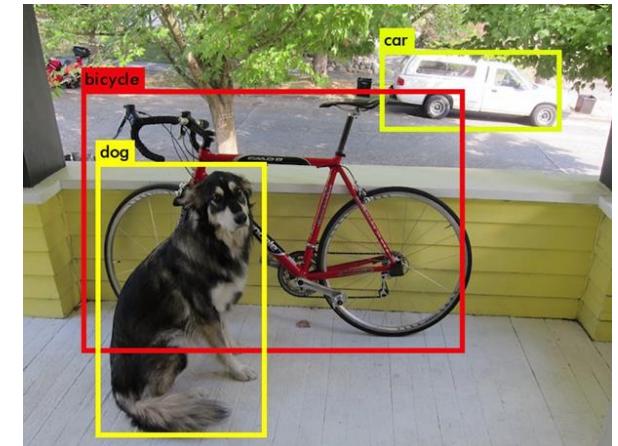
Problemas de Classificação



Problemas de Regressão



Detecção de Objetos



Classificação Métricas de Desempenho - Erro

As métricas de erro consistem na avaliação do resultado final do algoritmo em relação a uma dada solução, e a mesma deve ser selecionada de acordo com a característica do problema.

Problemas de Classificação

- Matriz de Confusão
- Acurácia
- Precisão
- Recall
- F1

Problemas de Regressão

- R^2
- MSE/RMSE
- MAE

Detecção de Objetos

- MAP

Classificação

Métricas de Desempenho - Erro

- **Matriz de Confusão**, é uma matriz que apresenta o número de observações reais e preditas pelo algoritmo.

		PREDITO	
		Classe A	Classe B
VERDADEIRO	Classe A	VP	FN
	Classe B	FP	VN

TP (True Positive/Verdadeiro Positivo) : Classe A é classificada como Classe A.

FP (False Positive/Falso Positivo) : Classe B é classificada como Classe A.

FN (False Negative/Falso Negativo) : Classe A é classificada como Classe B.

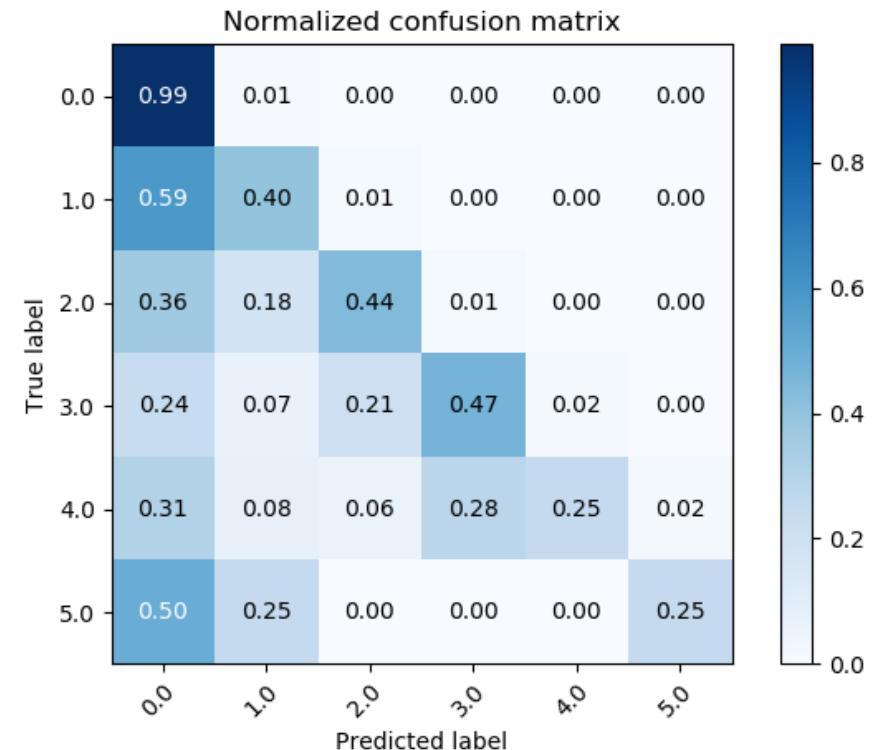
TN (True Negative/Verdadeiro Negativo) : Classe B é classificada como Classe B.

Classificação

Métricas de Desempenho - Erro

- **Matriz de Confusão**, é uma matriz que apresenta o número de observações reais e preditas pelo algoritmo.

Real		Previsão					
		Yes	No				
Yes	Yes	50 (TP)	40 (FN)				
	No	10 (FP)	500 (TN)				



Classificação

Métricas de Desempenho - Erro

- **Acurácia** ou taxa de acerto é a métrica mais intuitiva, se resume à razão do número de observações corretamente preditas sob o número do total de observações. É uma ótima métrica para avaliar modelos nos quais os dados possuem classes balanceadas, ou seja o mesmo número de observações por classe.

$$\text{Acurácia} = \frac{TP + TN}{TP + FP + TN + FN}$$

- **Precisão** é a taxa de observações classificadas corretamente como positivas em relação a todas as observações classificadas como positivas. Uma alta precisão se relaciona com uma baixa taxa de falsos positivos (FP).

$$Precisão = \frac{TP}{TP + FP}$$

Classificação

Métricas de Desempenho - Erro

- **Recall** (Sensitividade) se remete à proporção de observações corretamente classificadas como positivas em relação a todas as observações positivas que poderiam ter sido feitas, ou seja, também levam em consideração observações positivas marcadas como negativas (FN). Um Baixo valor de Recall se remete a um alto número de falsos negativos.

$$\text{Recall} = \frac{TP}{TP + FN}$$

- **F1-Score** é a média harmônica entre **Precisão** e **Recall**. Essa métrica leva em consideração ambos falsos negativos e falsos positivo. É uma métrica mais utilizada que acurácia, principalmente se possui dados com classes desbalanceadas, ou seja, quantidades diferentes de observações por classe.

$$F1 = \frac{2 * (\text{Recall} * \text{Precisão})}{\text{Recall} + \text{Precisão}}$$

Classificação

Métricas de Desempenho - Erro

Em resumo das métricas de erro de classificação:

- **Acurácia** mede a taxa de acerto do modelo, com classes balanceadas.
- **Precisão** indica a proporção de itens da classe classificados corretamente (TP) entre as observações marcadas da mesma classe (TP + FP).
- **Recall** indica a proporção de itens da classe classificados corretamente (TP) em relação a eles mesmos (TP) mais os itens da classe classificados incorretamente (FN).
- **F1-Score** média harmônica de **Precisão** e **Recall**. Leva em consideração FN e FP. Mais utilizado que **Acurácia** para dados com classes desbalanceadas.

Classificação

Métricas de Desempenho – Custo Computacional

- A métrica de desempenho de **custo computacional** é uma das métricas mais simples de avaliar o desempenho de um algoritmo.
- Leva em consideração o tempo em que o algoritmo leva na hora de processar os dados.
- Algoritmos mais complexos precisam de mais tempo para realizarem cálculos, portanto o custo computacional de um algoritmo destes tende a ser mais alto do que de um algoritmo mais simples.
- Em alguns casos, dependendo do tipo de dado à ser processado (Ex: imagens), mesmo algoritmos mais simples podem consumir uma quantidade absurda de processamento deixando inviável sua utilização em máquinas com hardwares de baixo custo.
- Custo Computacional sozinho não é uma boa métrica de avaliação, porém combinado a uma boa métrica de erro (**Acurácia, RMSE, MAP, ...**) pode ser um bom critério de desempate.

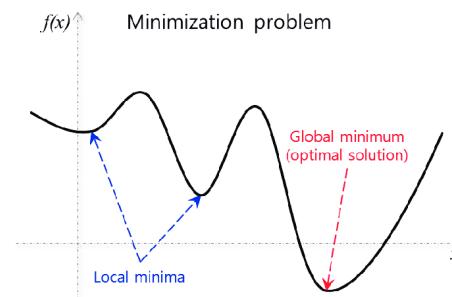
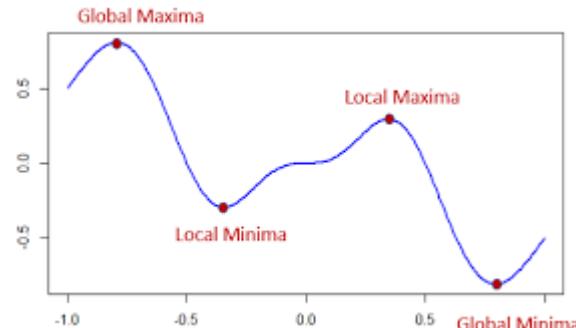
Classificação

Métricas de Desempenho - Convergência

- A métrica de desempenho de **convergência**, diferente das outras métricas mencionadas anteriormente, é a única métrica que só pode ser utilizada durante a fase de treinamento de um algoritmo.
- É pouco mencionada, porém muito importante na hora de escolher um algoritmo, mais do que uma métrica comparativa e critério de desempate, pode funcionar como indicador de que o algoritmo em questão não é adequado para os dados processados.
- Um dos parâmetros utilizados durante a fase de treinamento é o número máximo de iterações em que o algoritmo irá processar, às vezes chamado de *epochs* ao trabalhar com **Redes Neurais**.

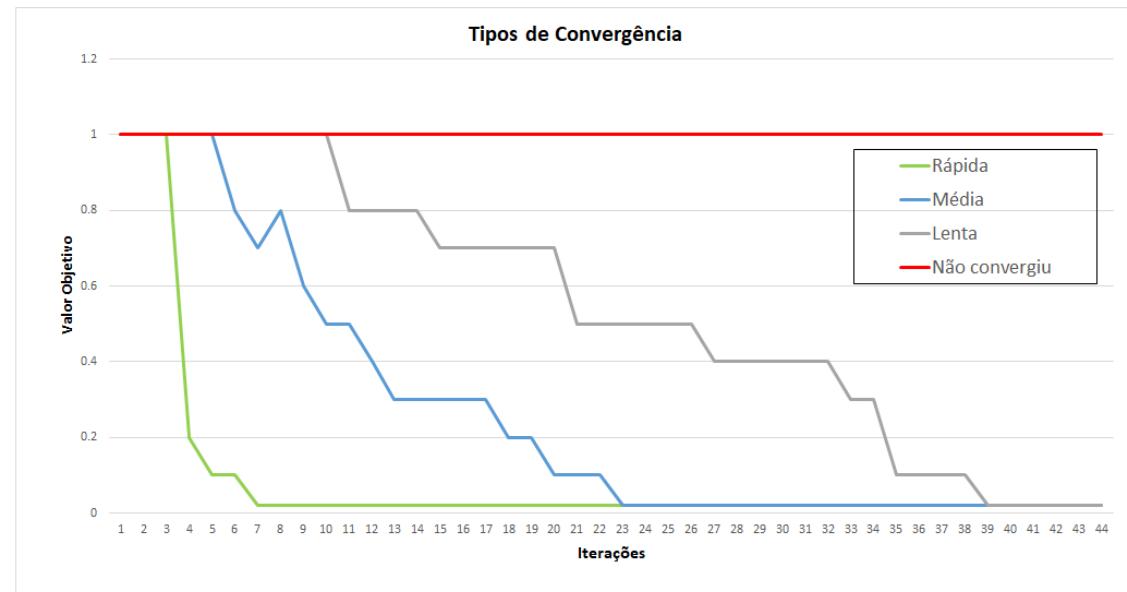
Classificação Métricas de Desempenho - Convergência

- **Solução Local Vs. Solução Global:** Na busca para encontrar o melhor **valor objetivo**, dentro do espaço de possíveis soluções existente para o denominado problema. O algoritmo utilizado pode encontrar **Soluções Locais** ou **Soluções Globais**.
- **Soluções Locais** é o nome dado ao conjunto para possíveis soluções que podem apresentar valores promissores dentro de uma vizinhança de possíveis soluções, porém não correspondem a o melhor **valor objetivo** contido dentro do domínio total de todas as possíveis soluções.
- **Soluções Globais** é o nome dado ao conjunto de soluções que representam o melhor **valor objetivo** de um problema levando em consideração todo o domínio do espaço de busca.



Classificação Métricas de Desempenho - Convergência

- **Curvas de Convergência** são uma representação gráfica do **valor objetivo** encontrado pelo algoritmo no final de cada iteração.
- Indicam a velocidade que o algoritmo leva para atingir (ou não) a convergência. O que pode significar desde um erro de parametrização ou até mesmo inaptidão do algoritmo escolhido.
- Uma rápida convergência, indica uma boa eficiência do algoritmo em lidar com **Soluções Locais**.



Classificação Métricas de Desempenho - Exemplo

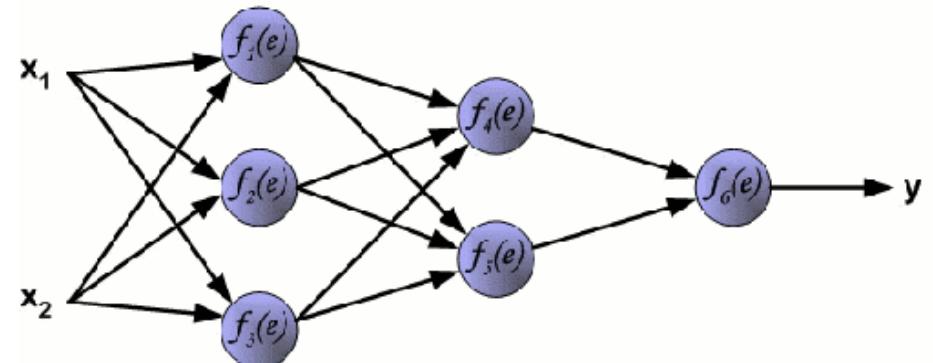
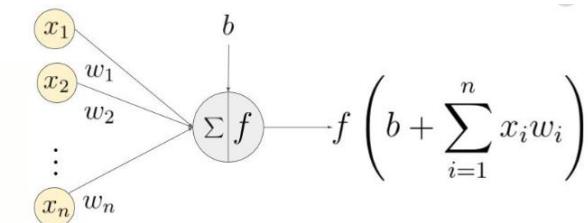
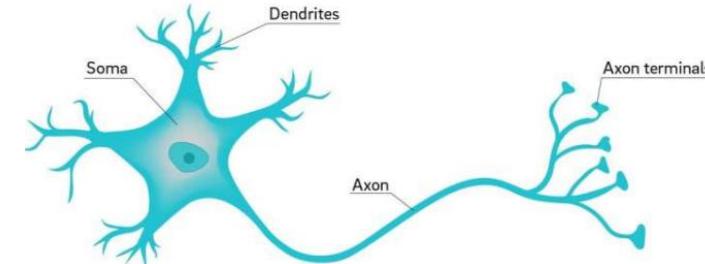
Classificação Métricas de Desempenho - Exercício

CNN

CNN

Artificial Neural Networks (ANN)

ANNs são sistemas de computação que "aprendem" (ou seja, melhoram progressivamente o desempenho) tarefas, considerando exemplos sem programação específica de tarefa.



CNN

Funções de Ativação

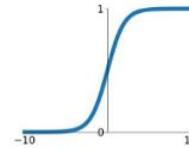
A função de ativação define a saída de um neurônio dado uma entrada ou conjunto de entradas (saída de vários neurônios anteriores). A função de ativação é a imitação da estimulação de um neurônio biológico.

As principais características são:

1. Controle de magnitude dos valores de saída
2. Adição de não linearidade, permitindo o neurônio aprender padrões mais complexos.
3. Vanishing Gradient
4. Outras Funções

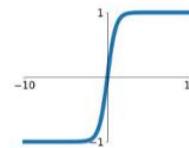
Sigmoid

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



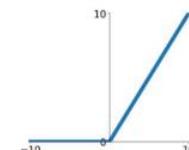
tanh

$$\tanh(x)$$



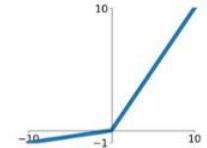
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

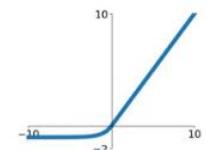


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

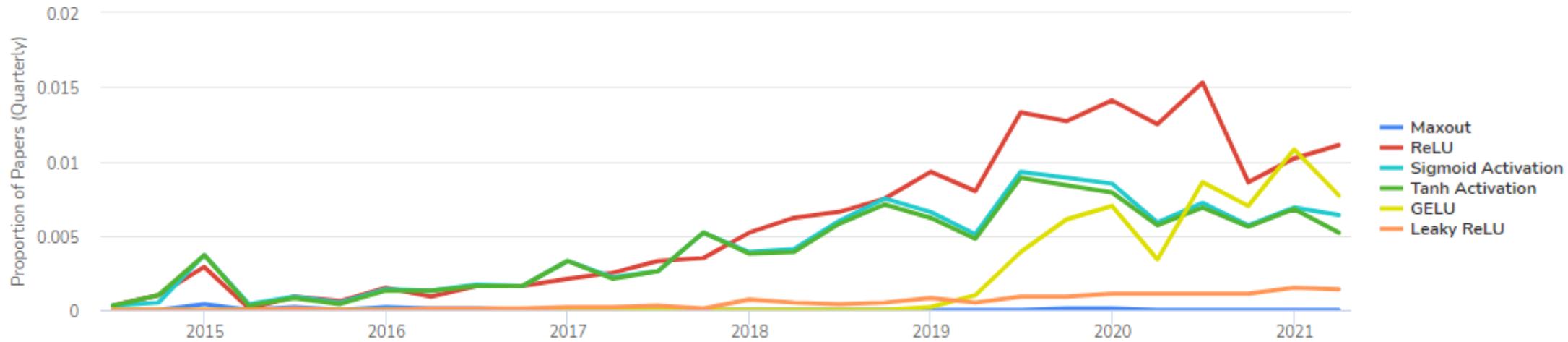
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



CNN

Funções de Ativação

Usage Over Time



CNN

Funções de Ativação Sigmoid Vs Softmax

- Sigmoid
 - Função unidimensional.
 - Comumente utilizada para Classificação Binária.
Pode ser utilizada para Multi-Classe.
 - Varia de [0 a 1] ou [-1 a 1], dependendo da convenção.
 - A soma das probabilidades não precisa ser 1.(MultiClasse)
 - Probabilidades independentes.
 - Ex: Doenças em uma imagem de [Raio-X](#)
- Softmax
 - Função Multivariável.
 - Utilizada apenas para Classificação Multi-Classe.
 - Varia de [0 a 1] por variável
 - A soma das probabilidades vai ser 1.
 - Probabilidades Inter-relacionadas
 - A maior será o label target
 - Não é uma função contínua, é utilizada para mapear as saídas da ultima camada da rede neural em uma distribuição de probabilidade.

$$f(x) = \text{sigmoid}(x) = \frac{1}{1 + e^{-x}}$$

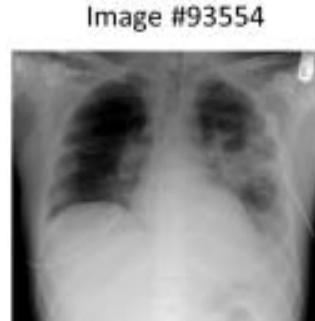


Image #93554

Label #93554

pneumonia	1
cardiomegaly	0
nodule	0
abscess	1
mass	0
pleural thickening	0
hernia	0

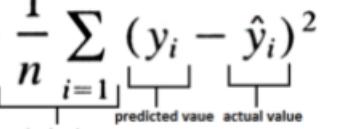
$$\text{softmax}(z_j) = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}} \text{ for } j = 1, \dots, K$$

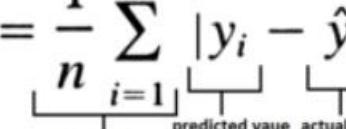
CNN

Loss Functions

- Regressão ou classificação
- Análise do conjunto de dados a ser aplicado
- Afetam diretamente a performance do modelo

Regressão:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$


$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$


Classificação:

$$CE = - \sum_i^C t_i \log(s_i)$$

$$CE = -\log \left(\frac{e^{s_p}}{\sum_j^C e^{s_j}} \right)$$

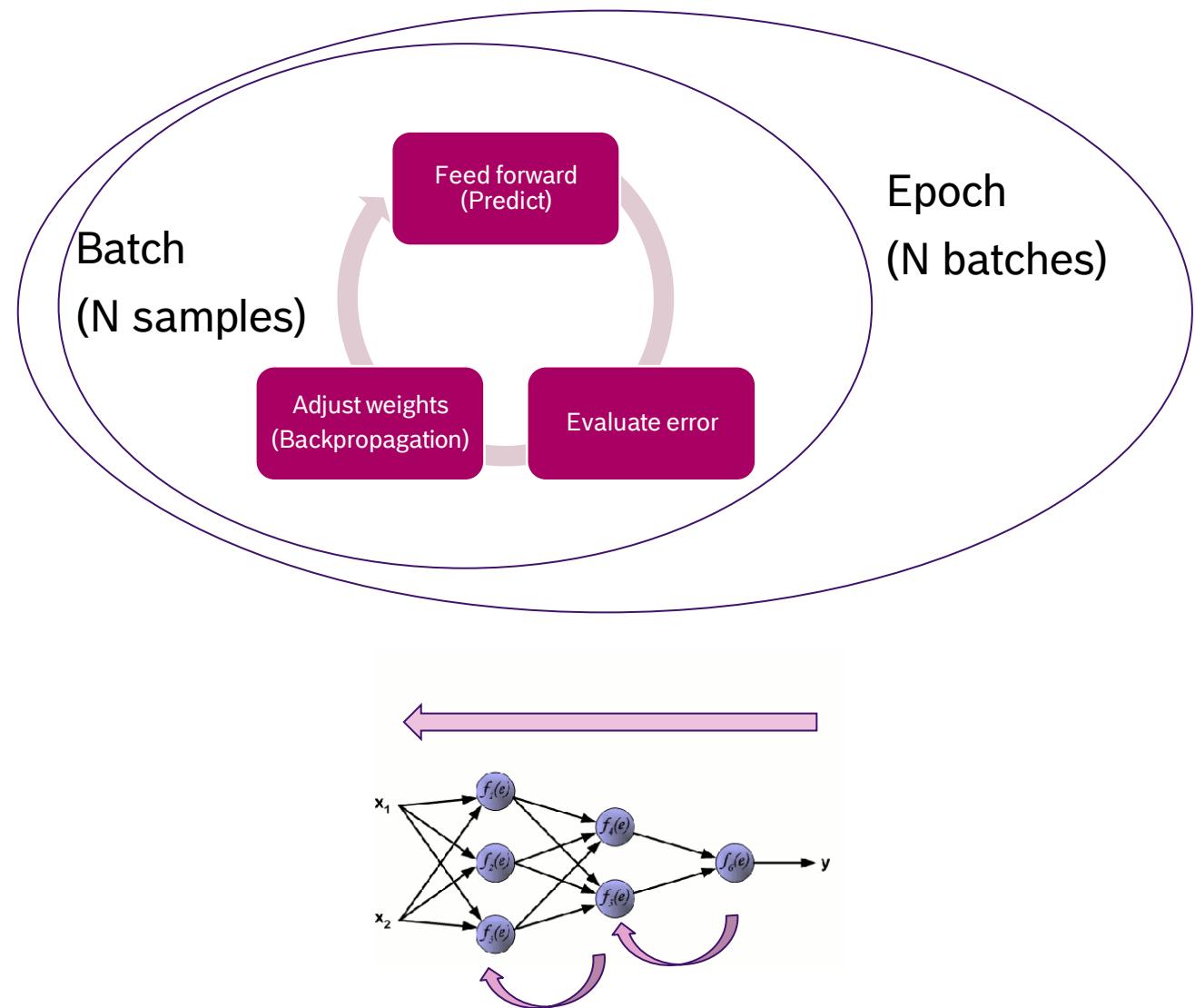
Personalizadas:

$$\begin{aligned} & \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B 1_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] \\ & + \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B 1_{ij}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] \\ & + \sum_{i=0}^{S^2} \sum_{j=0}^B 1_{ij}^{obj} (C_i - \hat{C}_i)^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^B 1_{ij}^{noobj} (C_i - \hat{C}_i)^2 \\ & + \sum_{i=0}^{S^2} 1_i^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2 \end{aligned}$$

CNN

“Backpropagation”

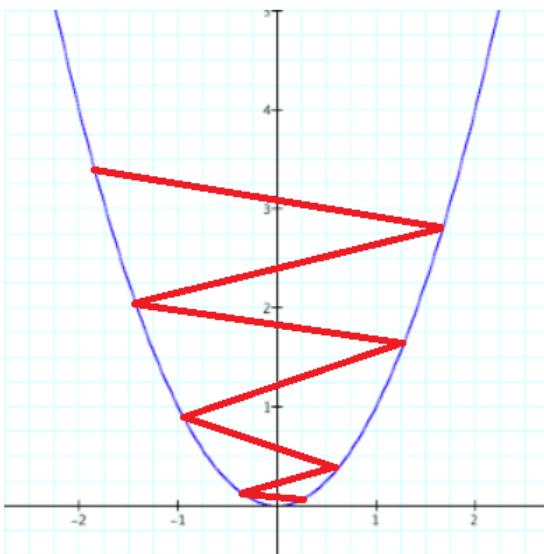
“O termo backpropagation surge do fato que o algoritmo se baseia na retropropagação dos erros para realizar os ajustes de pesos das camadas intermediárias. A maneira de calcular as derivadas parciais do erro de saída em relação a cada um dos pesos da rede é o que caracteriza o backpropagation.”



CNN

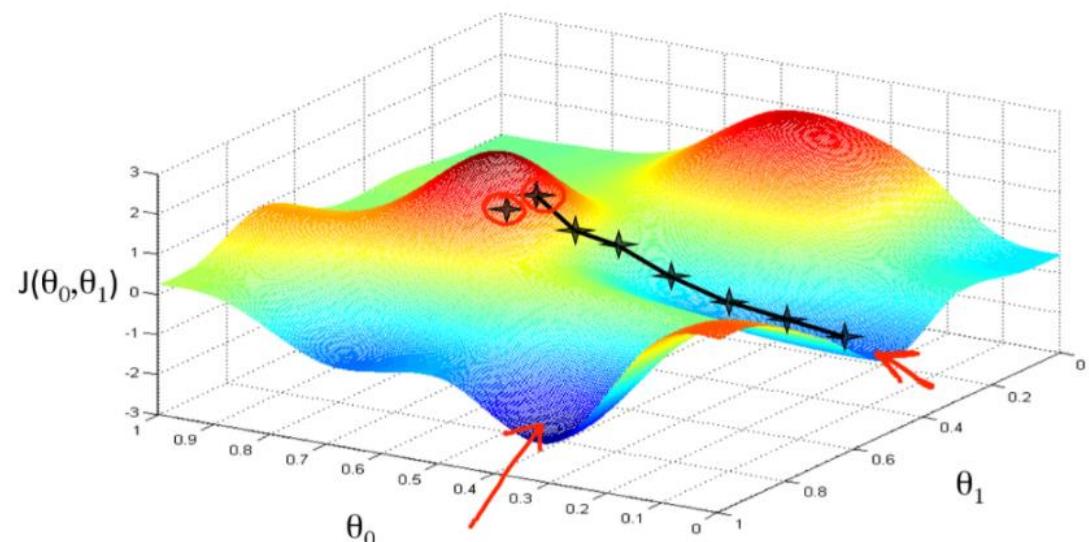
Gradient descent

O gradiente descendente é um otimizador utilizado que auxilia a atualização dos valores dos pesos de uma rede neural, de forma com que faça que o erro da rede diminua com sua recorrência durante o treinamento



$$\Delta w_{ij} = (\eta * \frac{\partial E}{\partial w_{ij}})$$

↑ ↑ ↑
weight learning weight
increment rate gradient

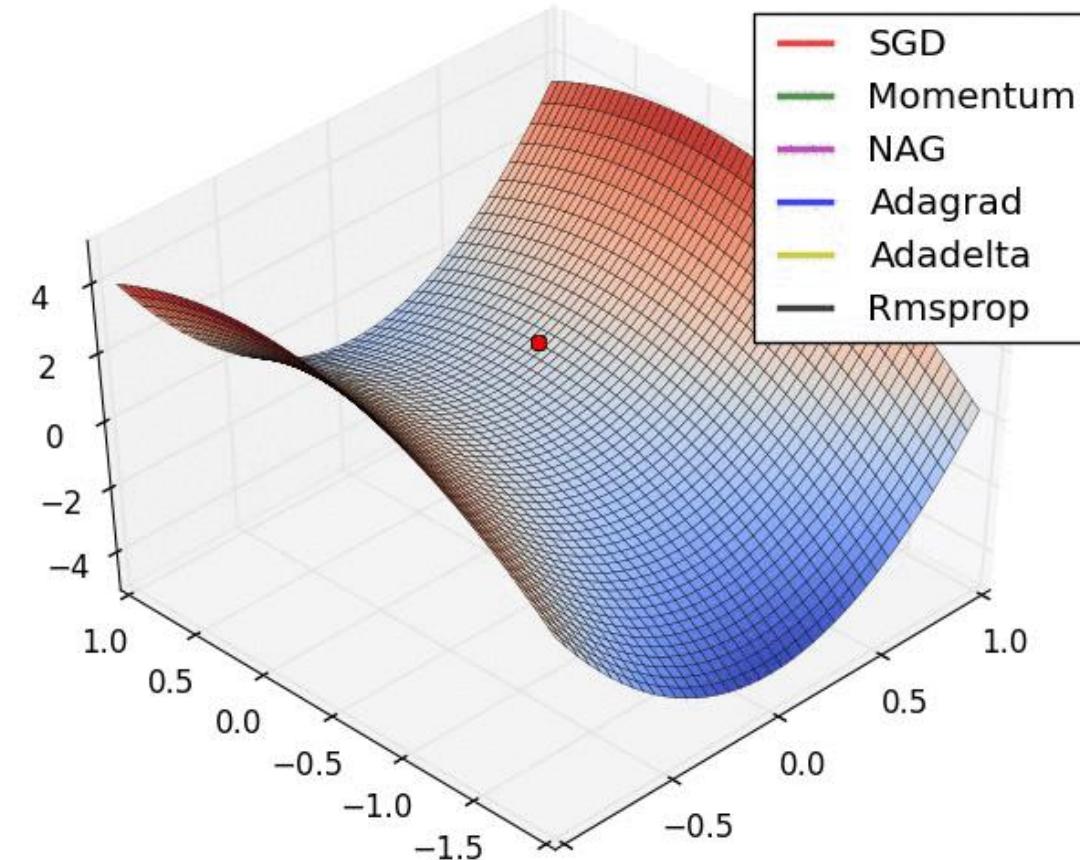


CNN

Otimizadores

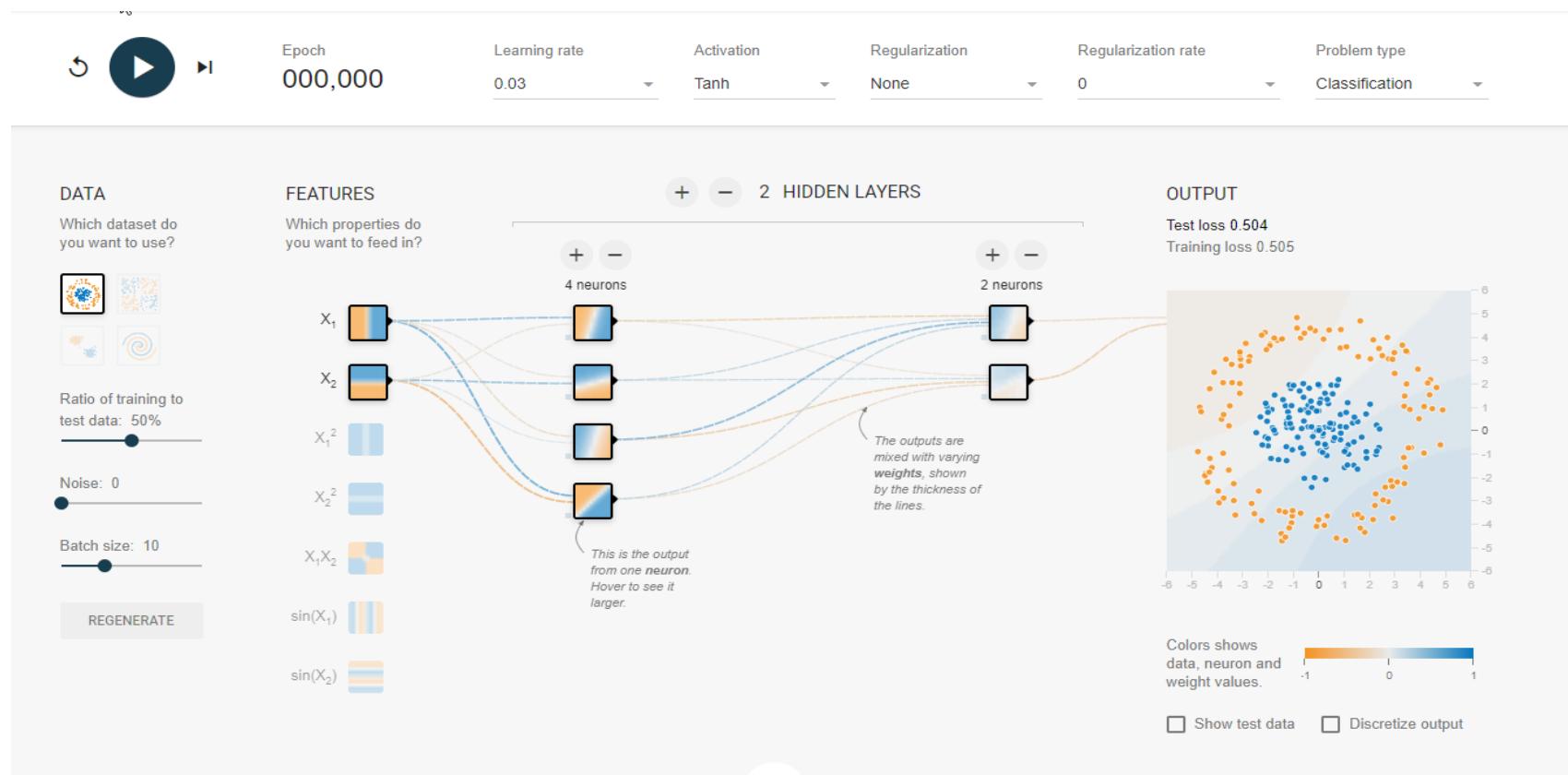
Os otimizadores atualizam os parâmetros de peso para minimizar a função de perda. A função de perda atua como um guia para o otimizador dizendo se ele está se movendo na direção certa para chegar ao o mínimo global.

Existem vários possíveis otimizadores a serem utilizados com as redes neurais.



CNN

Tensorflow Playground



<https://playground.tensorflow.org/>

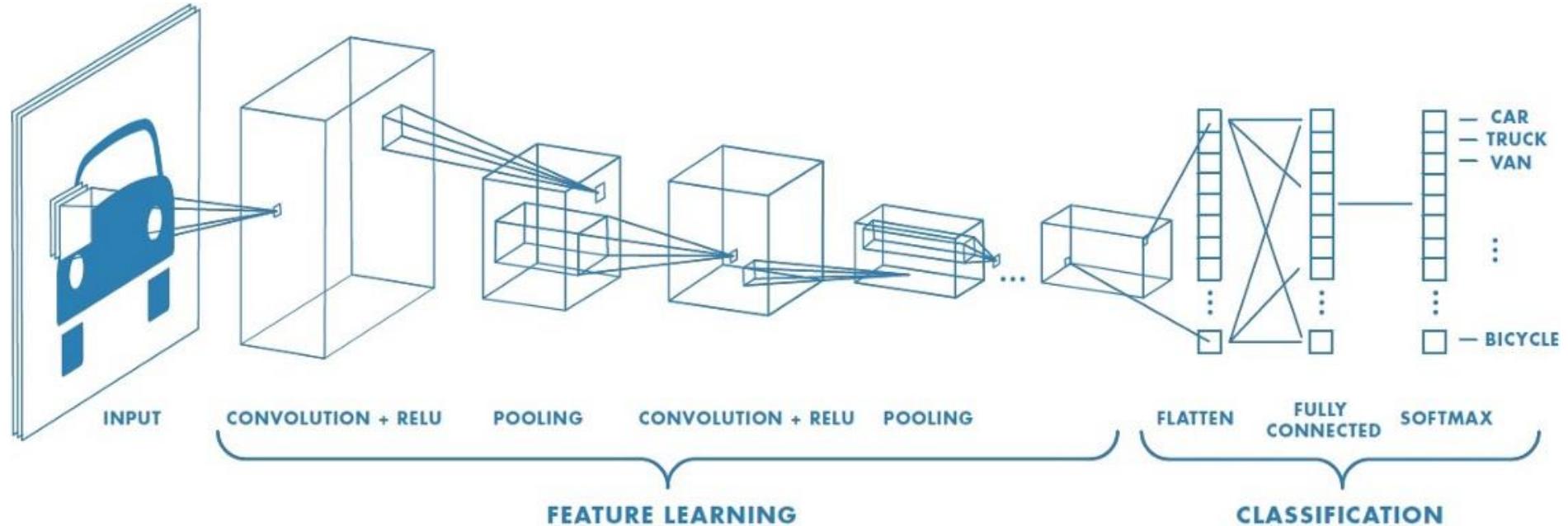
CNN

Hands on classification with sklearn and mnist

Jupyter: HO_Classification_MNIST_ANN

CNN

Convolutional Neural Network



<https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148#:~:text=Convolution%20preserves%20the%20relationship%20between,an%20a%20filter%20or%20kernel.>

CNN Convolução



		165	187	209	58	7
	14	125	233	201	96	159
253	144	120	251	41	147	204
57	100	32	241	23	165	30
208	118	124	27	59	201	79
210	236	105	169	19	218	156
35	178	199	197	4	14	216
115	104	34	111	19	196	
32	69	231	203	74		

Imagen RGB

254	141	200	276	100	220	3	77	224	100	0	220	100	0	0	20	0	0	0	0
27	66	231	79	14	107	140	210	43	209	11	220	69	243	11	238	244	0	33	89
54	45	17	715	879	274	17	41	97	199	55	273	213	47	0	279	78	189	173	959
6	169	105	71	104	160	68	171	37	182	18	147	174	1	140	211	178	189	199	894
370	128	107	100	107	14	249	231	179	13	131	110	274	179	233	139	234	0	324	
270	25	0	169	174	11	11	11	11	50	110	109	0	173	107	183	173	93	87	49
230	29	171	180	180	18	230	279	190	94	261	51	97	131	66	97	198	141	53	174
155	24	54	6	104	100	47	129	274	205	21	193	103	13	12	93	132	206		
320	114	79	170	111	111	0	101	69	103	69	14	173	94	19	1	23	5	133	218
53	249	237	169	80	101	131	94	69	103	111	0	111	28	109	6	249	76	117	145
135	155	220	78	60	327	210	129	110	77	29	40	0	0	214	190	209	116	135	93
102	64	179	192	20	119	37	263	282	111	93	40	177	13	31	179	199	118	204	81
264	110	120	177	79	59	69	109	107	192	42	44	8	175	255	101	11	28	112	
250	69	88	7	93	69	170	180	68	102	18	218	849	277	23	138	252	145		
69	196	224	207	140	22	110	224	94	102	10	83	47	69	242	199	232	111	249	
140	39	101	230	240	175	133	64	37	69	20	235	147	1	91	199	69	93	49	295
251	1	178	150	131	95	97	174	240	192	77	113	223	193	192	93	65	232	63	196
319	180	222	220	69	102	140	70	179	19	19	4	194	279	192	102	63	81	132	206
273	237	185	940	161	101	114	49	74	209	16	95	108	29	217	141	0	188	256	
69	9	17	22	10	210	70	81	70	241	198	119	93	93	220	192	153	242	137	47

Imagen Cinza

<https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148#:~:text=Convolution%20preserves%20the%20relationship%20between,an%20a%20filter%20or%20kernel.>

CNN

Convolução

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0



1	0	1
0	1	0
1	0	1

Convolução

<https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148#:~:text=Convolution%20preserves%20the%20relationship%20between,an%20a%20filter%20or%20kernel.>

CNN

Convolução

The diagram illustrates a convolution operation. On the left, a 5x2 input matrix (green) is multiplied by a 3x3 kernel (yellow). The result is a 3x1 output matrix (black).

Input Matrix:

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Kernel:

1	0	1
0	1	0
1	0	1

Result:

$1*1=1$	$0*1=0$	$1*1=1$
$0*0=0$	$1*1=1$	$0*1=0$
$1*0=0$	$0*0=0$	$1*1=1$

Output:

4		

Convolução

<https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148#:~:text=Convolution%20preserves%20the%20relationship%20between,an%20a%20filter%20or%20kernel.>

CNN

Convolução

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0



1	0	1
0	1	0
1	0	1

=

$1*1=1$	$0*1=0$	$1*0=0$
$0*1=0$	$1*1=1$	$0*1=0$
$1*0=0$	$0*1=0$	$1*1=1$

=

4	3	

Convolução

<https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148#:~:text=Convolution%20preserves%20the%20relationship%20between,an%20a%20filter%20or%20kernel.>

CNN

Convolução

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0



1	0	1
0	1	0
1	0	1

=

1*1=1	0*1=0	1*1=1
0*1=0	1*1=1	0*0=0
1*1=1	0*0=0	1*0=0

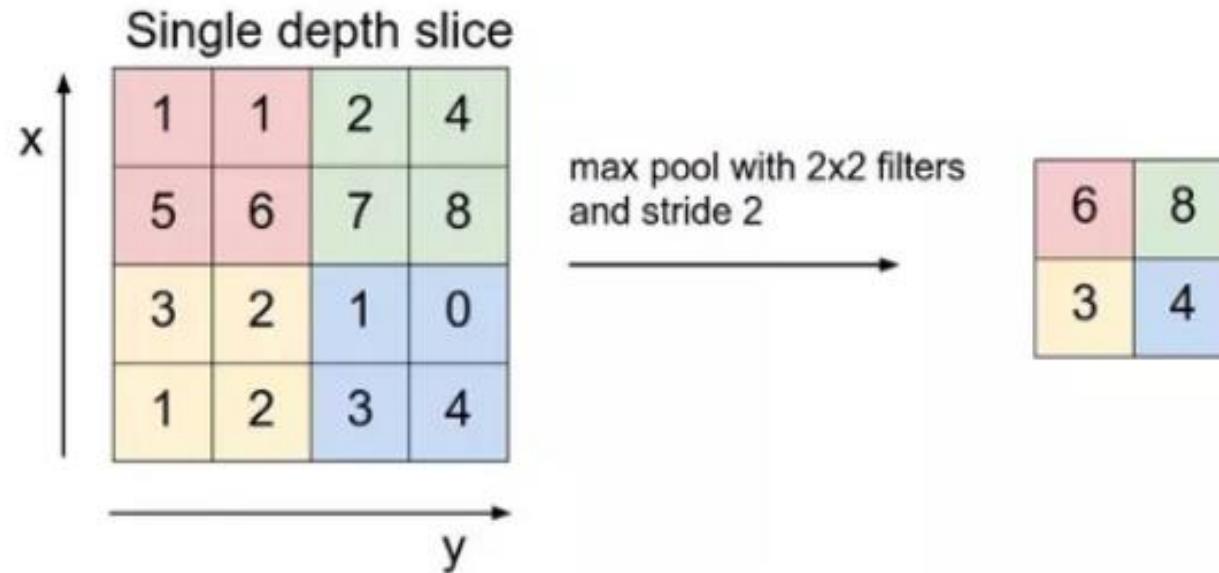
=

4	3	4
2	4	3
2	3	4

Convolução

<https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148#:~:text=Convolution%20preserves%20the%20relationship%20between,an%20a%20filter%20or%20kernel.>

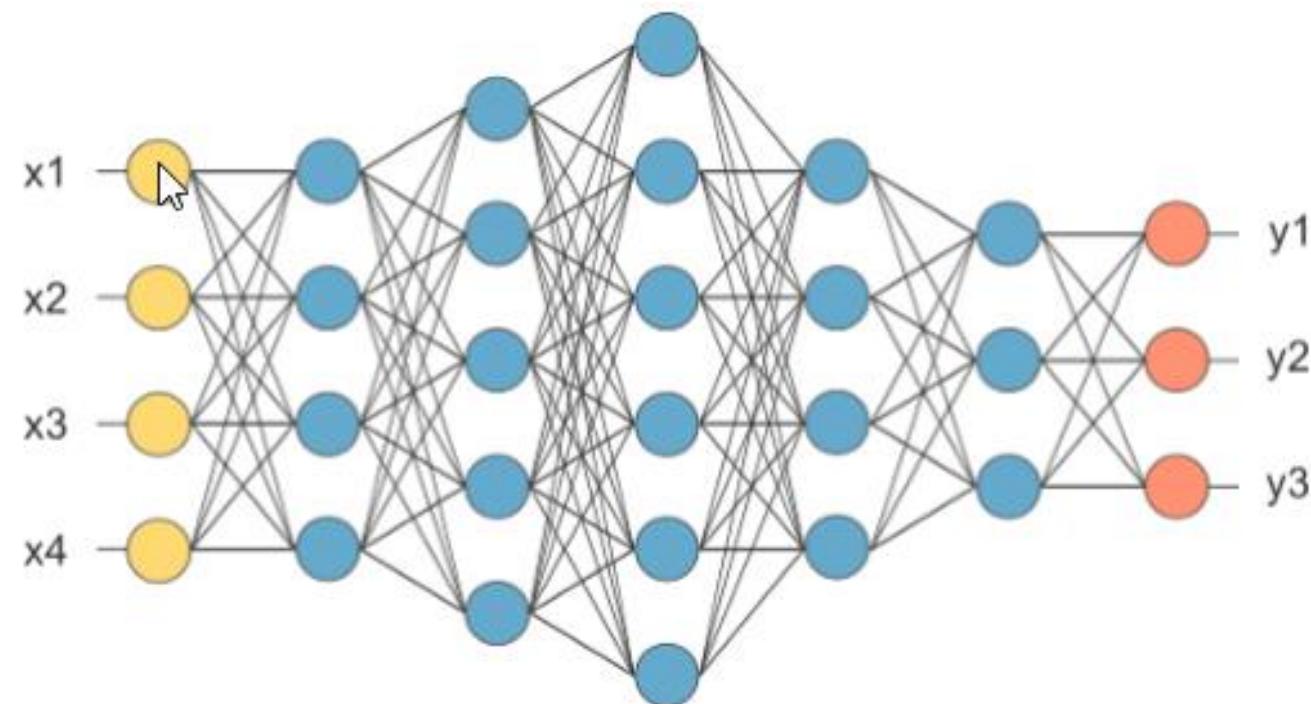
CNN Pooling



<https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148#:~:text=Convolution%20preserves%20the%20relationship%20between,an%20a%20filter%20or%20kernel.>

CNN

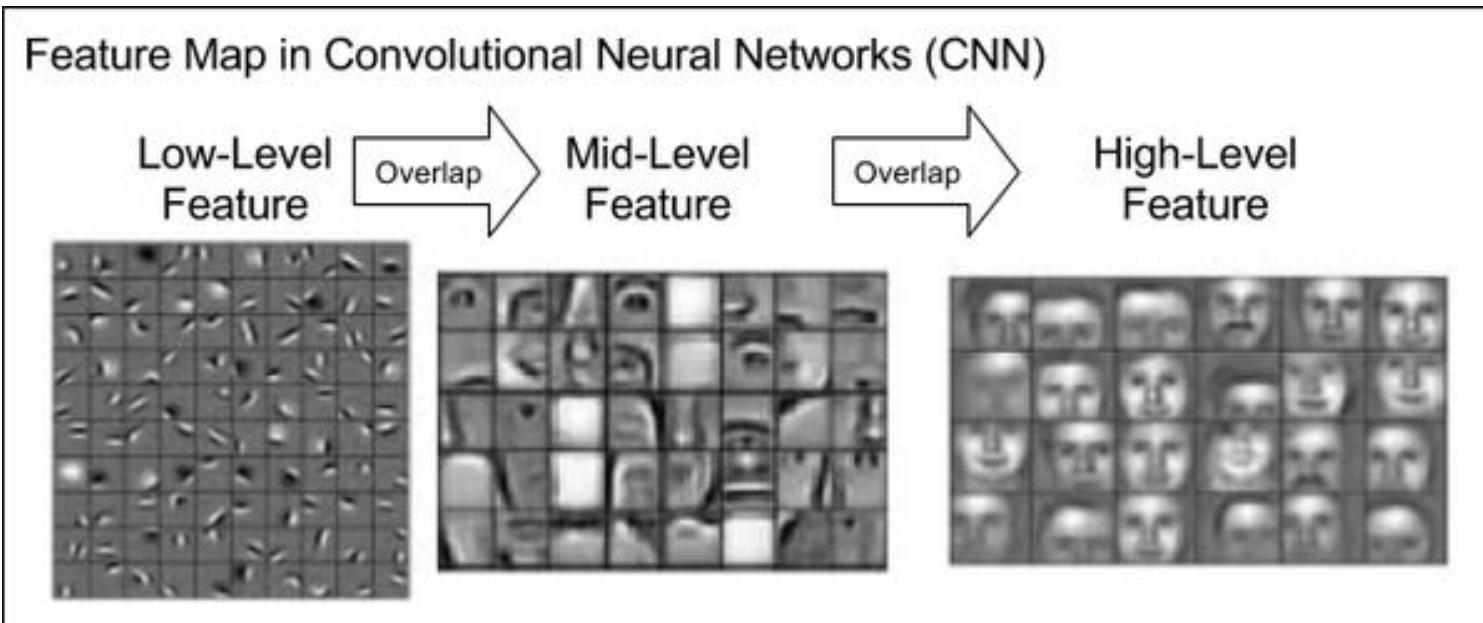
Fully Connected



<https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148#:~:text=Convolution%20preserves%20the%20relationship%20between,an%20a%20filter%20or%20kernel.>

CNN

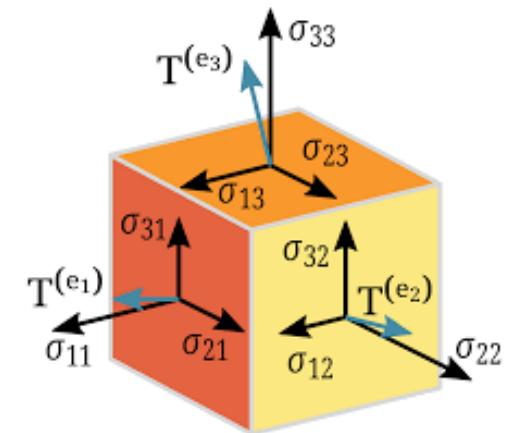
Abstração



FRAMEWORKS PARADEEP LEARNING

Tensorflow Intro

Originalmente desenvolvida pelo Google, Tensorflow é uma Biblioteca OpenSource de cálculo numérico utilizando grafos de fluxo de dados.



Tensorflow

Por que utilizar?

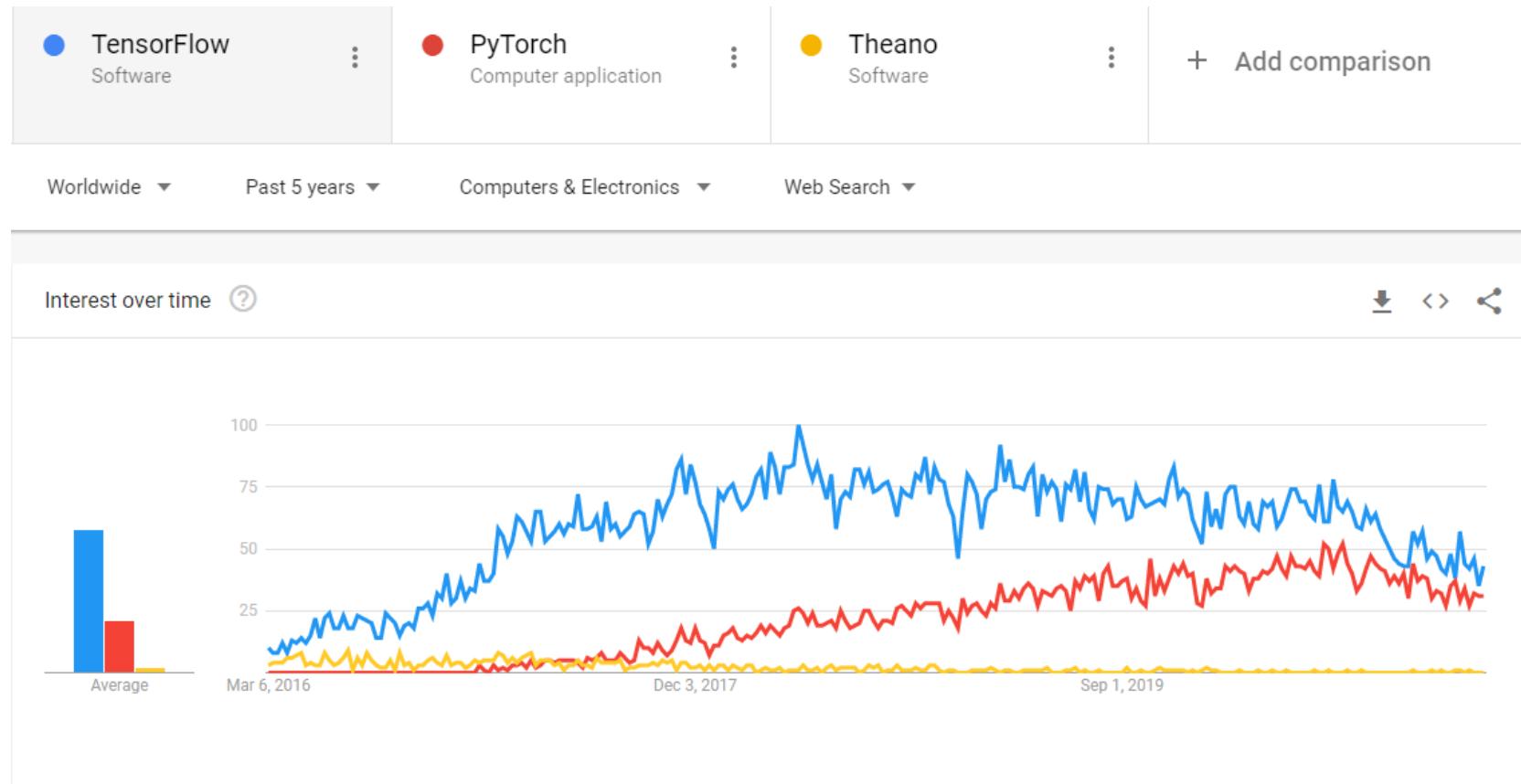
Companies using TensorFlow

[See case studies →](#)



Tensorflow

Por que utilizar?



Tensorflow

Por que utilizar?

Framework	Tensorflow	PyTorch	OMNX
Linguagens	Python, C++	Python, Lua.	Python, Julia, C++, R, or JavaScript
Pros.	<ul style="list-style-type: none">It has a lot of documentation and guidelines;It offers monitoring for training processes of the models and visualization (Tensorboard);It's backed by a large community of devs and tech companies;It provides model serving;It supports distributed training;Tensorflow Lite enables on-device inference with low latency for mobile devices;Tensorflow JS - enables deploying models in JavaScript environments, both frontend and Node.js backend. TensorFlow.js also supports defining models in JavaScript and training them directly in the browser using a Keras-like API.	<ul style="list-style-type: none">The modeling process is simple and transparent thanks to the framework's architectural style;The default define-by-run mode is more like traditional programming, and you can use common debugging tools as pdb, ipdb or PyCharm debugger;It has declarative data parallelism;It features a lot of pretrained models and modular parts that are ready and easy to combine;	<ul style="list-style-type: none">It features advanced GPU support, including multiple GPU mode;It can be run on any device;It has a high-performance imperative API;It offers easy model serving;It's highly scalable;It provides rich support for many programming languages, such as Python, R, Scala, Javascript, and C++, among others;
Cons.	<ul style="list-style-type: none">It has a higher entry threshold for beginners than PyTorch or Keras. Plain Tensorflow is pretty low-level and requires a lot of boilerplate coding, And the default Tensorflow "define and run" mode makes debugging very difficult.	<ul style="list-style-type: none">It lacks model serving in production (Although it will change in the future)It's new and not yet widely known (Has less customers/users at this point)It lacks interfaces for monitoring and visualization such as Tensorboard (As a workaround, you can connect externally to Tensorboard)	<ul style="list-style-type: none">It has a much smaller community behind it compared with Tensorflow;It's not so popular among the research community.

Tensorflow

Por que utilizar?

- Possui integração com Tensor-RT da NVIDIA
- API de Alto nível para treinamento validação e inferência (Keras)
- Otimização de Models (TF model optimization toolkit)
- Servir Modelos para Inferência (Tensorflow Serving)

Tensorflow Intro

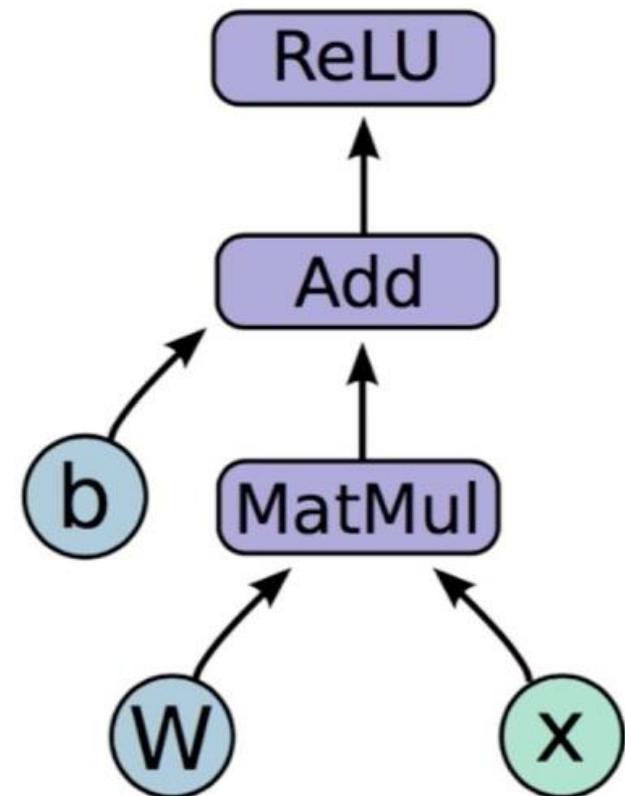
Computação representada por grafos:

- Nós = Operações
- Bordas = Tensores

Programa consiste de 2 fases:

- Construção
- Execução

$$h = \text{ReLU}(Wx + b)$$



Keras

Intro

Api de alto nível utilizada no Tensorflow.

- Keras é built-in a partir do Tensorflow 2.0
- User-friendly, permite desenvolvimento mais rápido de protótipos
- Tem todos os benefícios do Tensorflow. Ex: TF Model optimization tool, TF Serving, etc...



Keras

Comparação TF vs Keras

```
# Treinar um rede neural para classificação do MNIST
from keras.datasets import mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

```
# Keras
import tensorflow as tf
from tensorflow.keras.layers import Input, Dense, Flatten

model = tf.keras.Sequential([
    Flatten(input_shape=(784,)),
    Dense(128, activation='relu'),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')
])

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics = ['accuracy'])

model.summary()

model.fit(x_train.reshape(-1,784), pd.get_dummies(y_train), nb_epoch=15, batch_size=128,
verbose=1)
```

```
# Tensorflow Graph
X = tf.placeholder(dtype=tf.float64)
Y = tf.placeholder(dtype=tf.float64)
num_hidden=128

# Build a hidden layer
W_hidden = tf.Variable(np.random.randn(784, num_hidden))
b_hidden = tf.Variable(np.random.randn(num_hidden))
p_hidden = tf.nn.sigmoid( tf.add(tf.matmul(X, W_hidden), b_hidden) )

# Build another hidden layer
W_hidden2 = tf.Variable(np.random.randn(num_hidden, num_hidden))
b_hidden2 = tf.Variable(np.random.randn(num_hidden))
p_hidden2 = tf.nn.sigmoid( tf.add(tf.matmul(p_hidden, W_hidden2), b_hidden2) )

# Build the output layer
W_output = tf.Variable(np.random.randn(num_hidden, 10))
b_output = tf.Variable(np.random.randn(10))
p_output = tf.nn.softmax( tf.add(tf.matmul(p_hidden2, W_output), b_output) )

loss = tf.reduce_mean(tf.losses.mean_squared_error(
    labels=Y,predictions=p_output))
accuracy=1-tf.sqrt(loss)
minimization_op = tf.train.AdamOptimizer(learning_rate=0.01).minimize(loss)

feed_dict = {
    X: x_train.reshape(-1,784),
    Y: pd.get_dummies(y_train)
}
with tf.Session() as session:
    session.run(tf.global_variables_initializer())

    for step in range(10000):
        J_value = session.run(loss, feed_dict)
        acc = session.run(accuracy, feed_dict)
        if step % 100 == 0:
            print("Step:",step, " Loss:", J_value, " Accuracy:", acc)

            session.run(minimization_op, feed_dict)
pred00 = session.run([p_output], feed_dict={X: x_test.reshape(-1,784)})
```

Keras

Sequential x Functional API

Jupyter: Sequential_vs_API.ipynb

Keras Camadas

Inicialização:

- `Input(shape=(?, ?, ?))`
- `Model(inputs, outputs)`
- `Sequential()`

Keras

Camadas

Extração de características:

- Dense(units, activation, use_bias)
- Conv2D(filters, kernel_size, strides)
- LSTM(units, activation)
- MaxPooling2D/MaxPool2D(pool_size, padding)
- Flatten(input_shape)

Keras Camadas

Ativação:

- ReLU()
- LeakyReLU()
- SoftMax()
- Sigmoid()

Keras Camadas

Complementares:

- GlobalAveragePooling()
- BatchNormalization()
- Dropout(rate)
- Concatenate()([a , b])

Keras Camadas

Customização:

- Lambda(function, output_shape)

Keras

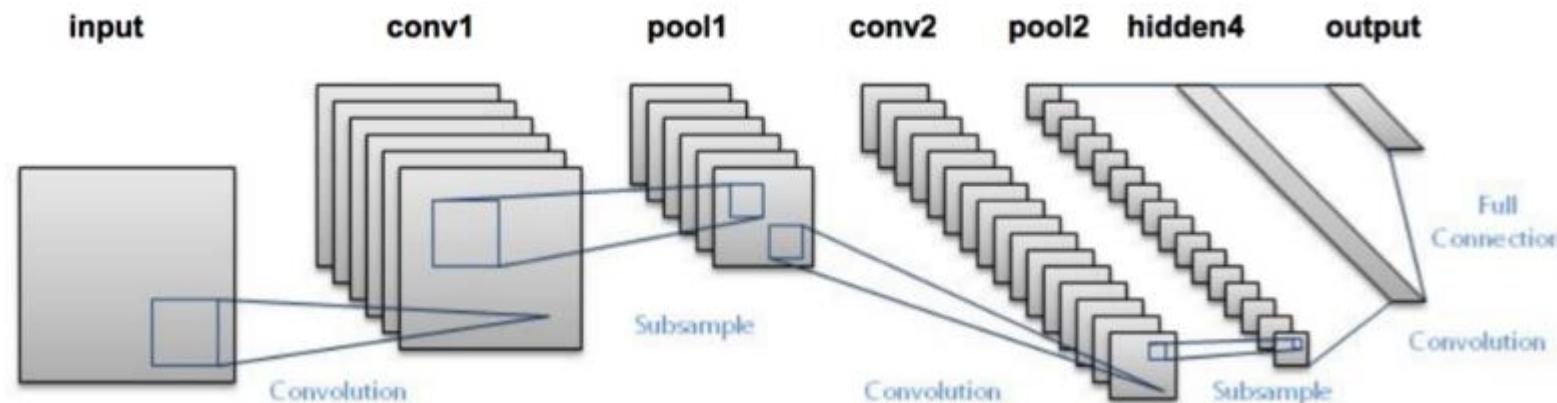
Exemplo : Arquitetura da LeNet para MNIST

Input = (28, 28)

outputs = 10

Filters = 16, 64

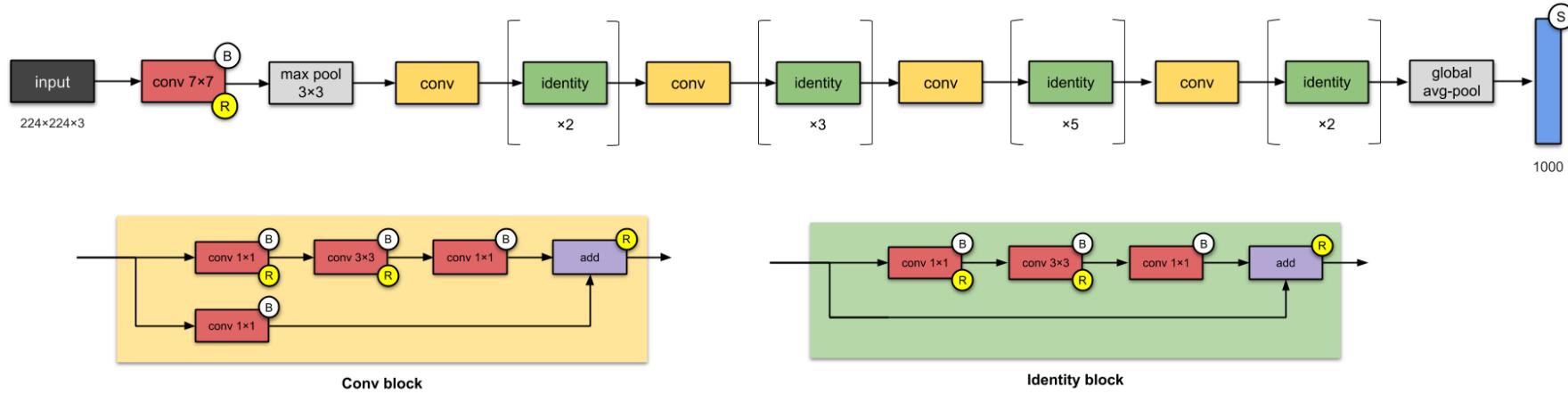
Hidden = ?



Jupyter: Códigos/LeNet.ipynb

Keras

Exercício : Arquitetura ResNet 50



Jupyter: ResNet50.ipynb

Keras

Exercício ANN vs Lenet para KMNIST

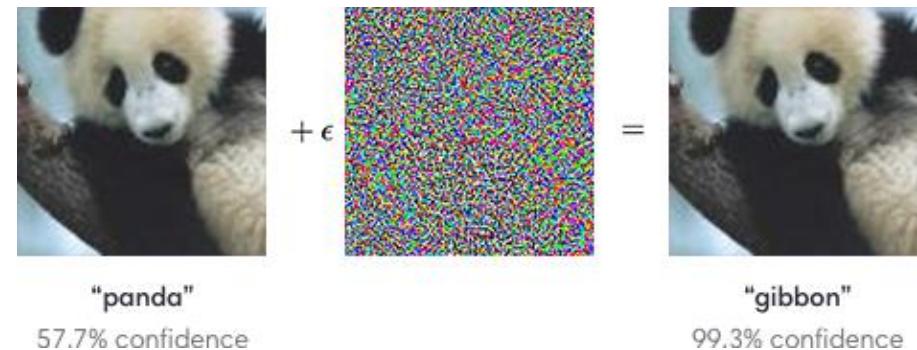
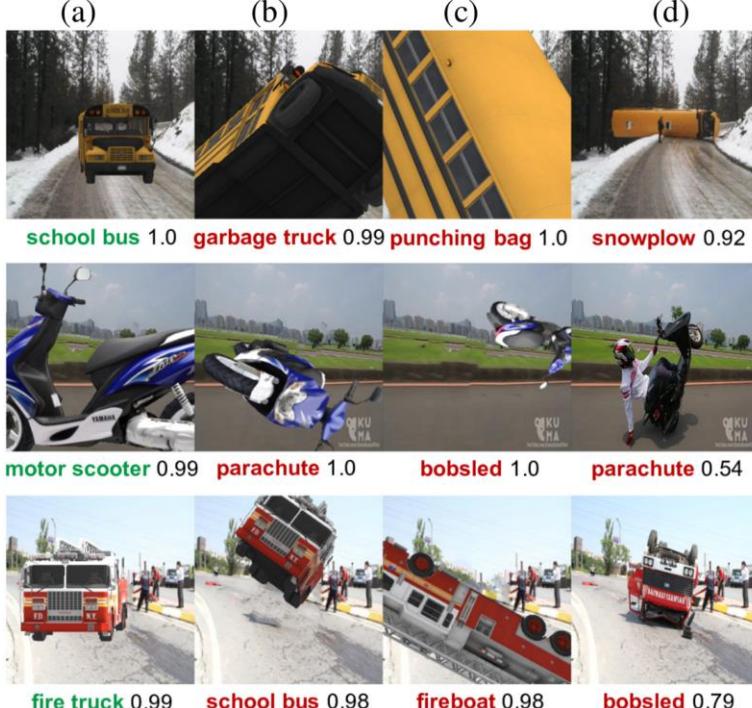
Jupyter: LeNet vs ANN for KMNIST.ipynb

DATA AUGMENTATION

Data Augmentation

Limitações da CNN

- Sensível a ruídos (luminosidade, oclusão, etc...)



Data Augmentation

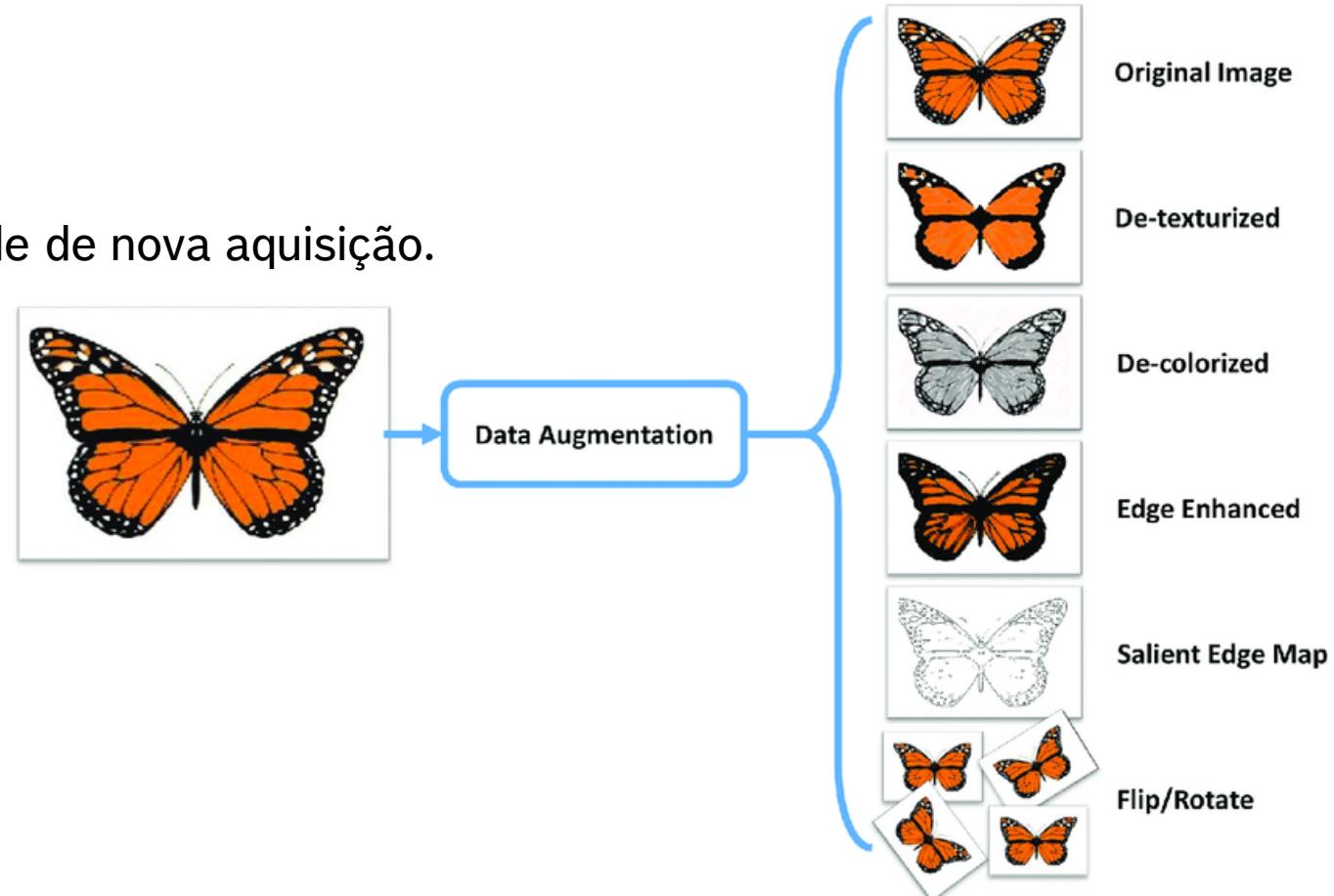
Limitações da CNN

Data Augmentation

Intro

Vantagens

- Gerar mais dados, sem necessidade de nova aquisição.
- Aumenta a Invariância da CNN.
- Diminui Overfitting.

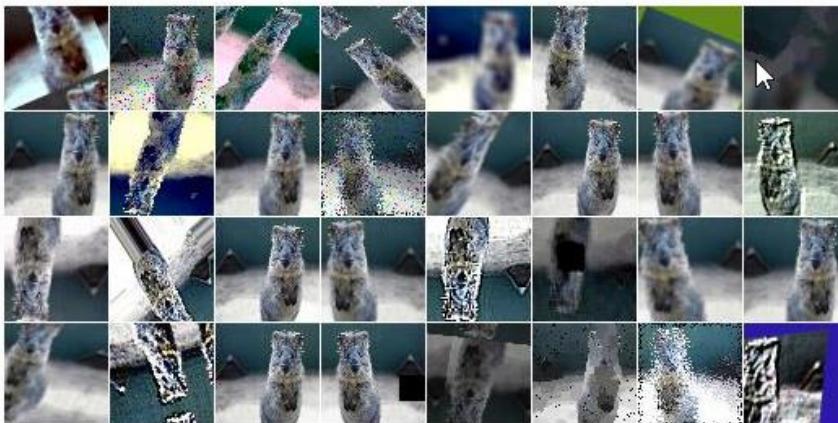


Data Augmentation

imgaug

imgaug

imgaug is a library for image augmentation in machine learning experiments. It supports a wide range of augmentation techniques, allows to easily combine these and to execute them in random order or on multiple CPU cores, has a simple yet powerful stochastic interface and can not only augment images, but also keypoints/landmarks, bounding boxes, heatmaps and segmentation maps.



Example augmentations of a single input image.

- Criar pipelines de transformação complexos
- Suporta Transformação de:
 - Imagens.
 - Landmarks/Keypoints.
 - Bounding Box.
 - Heatmaps.
 - Mapas de Segmentação.
- Transformação em Batchs.
- Funções de visualização e conversão.
- Documentação

Data Augmentation

Imgaug - Exemplo

Jupyter: LeNet vs ANN for KMNIST.ipynb

Data Augmentation

Albumentations

A Albumentations

Do more with less data

Albumentations is a computer vision tool that boosts the performance of deep convolutional neural networks.



- Baseado em Numpy, OpenCV e ImgAug.
- Muito utilizada em competições do Kaggle.
- Mais rápida que outros frameworks.
- Mais de 70 transformações.
- [Documentação](#)

Data Augmentation

Albumentations Comparação

Results for running the benchmark on the first 2000 images from the ImageNet validation set using an Intel Xeon Gold 6140 CPU. All outputs are converted to a contiguous NumPy array with the np.uint8 data type. The table shows how many images per second can be processed on a single core; higher is better. Source : <https://github.com/Albumentations-team/Albumentations#benchmarking-results>

FPS	albumentations 0.5.0	imgaug 0.4.0	torchvision (Pillow- SIMD backend) 0.7.0	keras 2.4.3	augmentor 0.2.8	solt 0.1.9
HorizontalFlip	9909	2821	2267	873	2301	6223
VerticalFlip	4374	2218	1952	4339	1968	3562
Rotate	371	296	163	27	60	345
ShiftScaleRotate	635	437	147	28	-	-
Brightness	2751	1178	419	229	418	2300
Contrast	2756	1213	352	-	348	2305
BrightnessContrast	2738	699	195	-	193	1179
ShiftRGB	2757	1176	-	348	-	-
ShiftHSV	597	284	58	-	-	137

Data Augmentation Albumentations - Exemplo

Jupyter: LeNet vs ANN for KMNIST.ipynb

Data Augmentation

Exercício

Fazer uma Classe de aumento de dados para o MNIST

Jupyter: LeNet vs ANN for KMNIST.ipynb

BATCH GENERATOR

Batch Generator

Intro

- Meu dataset tem mais de 500.000 imagens? Preciso carregar todas na memória?
- Parseamento de anotações precisa ser feito antes de carregar imagens?
- Como fazer um aumento de dataset sem gerar novas imagens no disco?
- Tamanho de imagens é diferente do tamanho de input da rede, preciso transformar todas as imagens antes? Preciso alterar o input da rede? Mas se estou utilizando um modelo pré-treinado?
- Meu modelo utiliza uma entrada de dados mais complexa, preciso criar um dataset utilizando este formato? (Ex: triplet loss)

Batch Generator Keras

- **ImageDataGenerator**
 - Vantagens:
 - Normalização
 - Possui pipeline de transformação integrado
 - Determinar Split de Validação
 - Desvantagens:
 - Entrada de dados limitada
 - Pipeline de transformação limitado
 - Não tem suporte para dados com anotação
- **Sequence**
 - Vantagens
 - Customização de todos os passos do Generator
 - Definição de pipelines de transformação (Ex: albumentations, imgaug, etc...)
 - Entrada de Dados e Anotações mais flexível
 - Possibilita fluxo mais complexos de dados (Ex: Triplet Loss)
 - Desvantagens:
 - Necessita de um Conhecimento maior da ferramenta
 - Tempo de desenvolvimento é maior
 - Mono-thread

Batch Generator

Exemplo Batch Generator para Classificação

Jupyter: LeNet vs ANN for KMNIST.ipynb

Batch Generator

Exercício Batch Generator para Detecção de Objetos

Jupyter: LeNet vs ANN for KMNIST.ipynb

Batch Generator

Tensorflow Dataset API

- **Tensorflow dataset API**

- Executado em Múltiplas threads.
- Mais escalável.
- Código mais Limpo.
- Mais controle sobre como os dados são carregados.

Batch Generator Tensorflow.Data

- **Funções**

- **shuffle**: Ordenar os dados aleatoriamente para remover a correlação entre os dados
- **map**: Aplicar uma função definida pelo usuário à todos os dados ao mesmo tempo.
- **batch** : Estruturar os dados em mini-batches
- **prefetch**: salva os batches cacheados na memória, para serem consumidos instantaneamente.

Batch Generator

Exemplo Batch Generator com tf.data

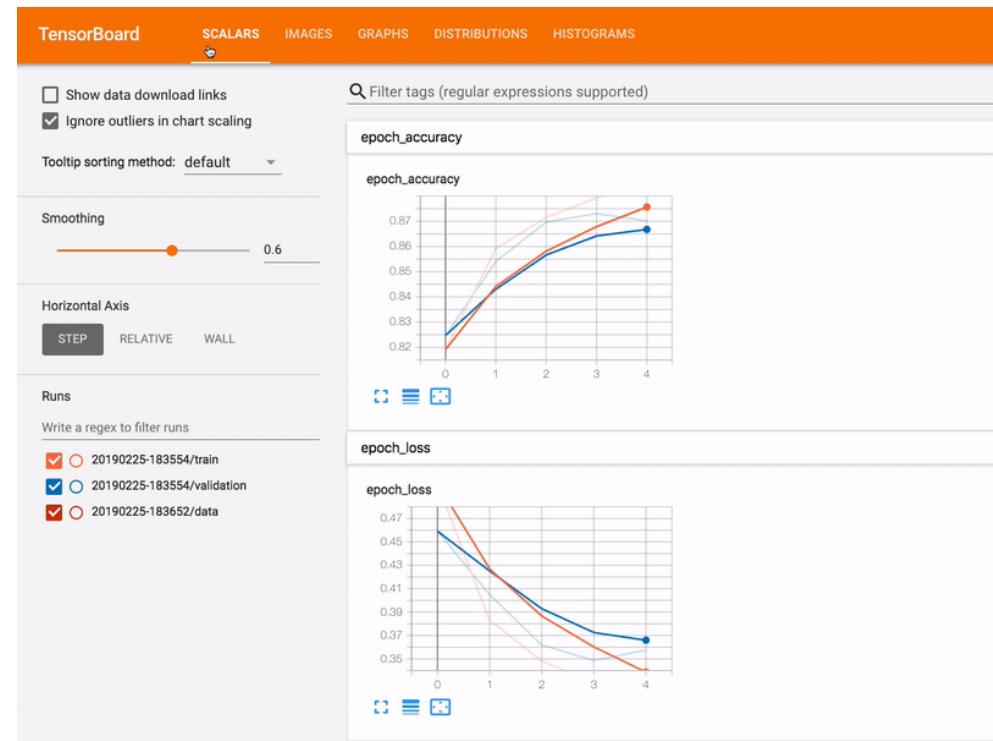
TENSORBOARD

Tensorboard Webserver

O tensorboard é uma ferramenta do tensorflow utilizada para o acompanhamento do treinamento de IA, utilizada até mesmo em frameworks “concorrentes” como o PyTorch

Principais utilidades:

- Baseado em desenvolvimento Web
- Acompanhamento de métricas
- Visualização de grafos
- Visualizações de imagens por épocas



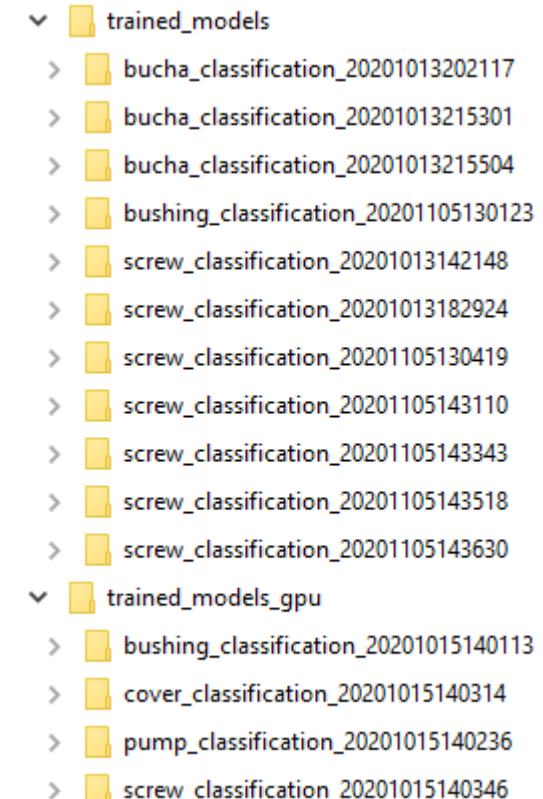
Tensorboard

Como utilizar

- O tensorboard monitora uma pasta em específico e para cada subpasta dentro dessa raiz será considerado um treinamento diferente.
- Ao instalar o tensorflow o tensorboard é instalado automaticamente
- O tensorboard é um executável incluso dentro da pasta Scripts da instalação do python, em geral essa pasta é adicionada junto a variável “path” do seu sistema operacional

Windows PowerShell

```
PS C:\Repositorios\vtwm\integration_beckhoff> tensorboard --logdir= ./ --port=6006
TensorBoard 1.13.1 at http://CA1NB7952:6006 (Press CTRL+C to quit)
```



Tensorboard

Como utilizar

- O keras possui a classe abstrata Callback que executa tarefas no fim de determinadas ações durante o treino.
- O tensorboard possui um callback com o próprio nome de Tensorboard onde implementa o uso apropriado das ferramentas para gerar logs compatíveis.
- Esse callback inclui o grafo do modelos e logs de todas as métricas utilizadas no treinamento

Tensorboard

Como utilizar

- O keras possui a classe abstrata Callback que executa tarefas no fim de determinadas ações durante o treino.
- O tensorboard possui um callback com o próprio nome de Tensorboard onde implementa o uso apropriado das ferramentas para gerar logs compatíveis.
- Esse callback inclui o grafo do modelos e logs de todas as métricas utilizadas no treinamento

```
from tensorflow.keras.callbacks import TensorBoard
```

```
tb = TensorBoard(log_dir='treinamento01', histogram_freq=0, batch_size=batch_size,
                 write_graph=True, write_grads=False, write_images=False, embeddings_freq=0,
                 embeddings_layer_names=None, embeddings_metadata=None)
```

```
model.fit(
    train_generator,
    epochs=epochs,
    validation_data=val_generator,
    callbacks=[check_point_saver_best_loss, check_point_saver_best_acc, check_point_saver, tb, cm_tb])
```