

The background features a complex network of thin grey lines and dots, forming a web-like structure. Scattered throughout are various triangles of different sizes and orientations, some with solid outlines and others with dashed or dotted lines. The overall aesthetic is technical and modern, typical of a presentation on artificial intelligence or data science.

DEEP LEARNING

Introdução a redes profundas

**Diferença entre DL e
RNAs 01**

**Convolutional Neural
Network 02**

**Recurrent Neural
network 03**

TABLE OF CONTENTS

**04 Long Short Term
Memory (LSTM)**

**05 Arquiteturas de
Deep Learning**

06



01 DIFERENÇAS ENTRE DL E RNAs

O QUE É UMA DL?

Inteligência Artificial

Qualquer técnica que permita aos computadores imitar a inteligência humana, usando lógica, árvores de decisão e aprendizado de máquina (incluindo aprendizado profundo)

- Visão Computacional
- Processamento de Linguagem Natural
- Reconhecimento de Fala
- Robótica

Aprendizado de Máquina

Um contribuinte da IA que inclui técnicas de estatística e algoritmos que capacitam máquinas a melhorarem tarefas com experiência.

- Supervisionado
- Não-supervisionado
- Aprendizado Reforçado

Aprendizagem Profunda

Subconjunto do aprendizado de máquina composto de algoritmos que permitem ao software treinar a si mesmo para realizar tarefas, como reconhecimento de voz e imagem, expondo uma grande quantidade de dados à multicamadas de redes neurais.

- Redes Neurais Artificiais
- Redes Neurais Recorrentes
- Redes Neurais Convolucionais

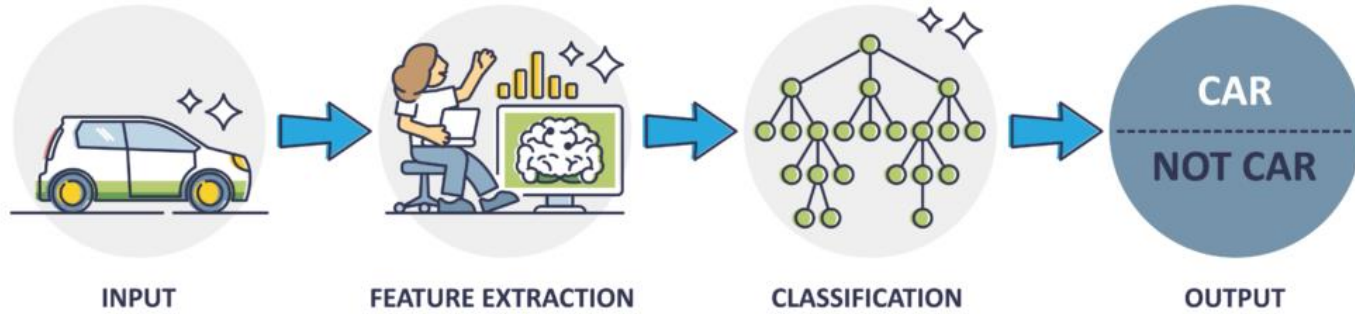
Ciência de Dados

Uma área interdisciplinar, que localiza-se em uma interface entre a **estatística** e a **ciência da computação**, que utiliza o método científico; processos, **algoritmos** e sistemas.

É focada na descoberta de **insights acionáveis** a partir de grandes conjuntos de dados.


BI, Big Data, Data Warehouse, Data Lake, **Analytics**, ...

MACHINE LEARNING




DEEP LEARNING





Deep Learning – Áreas de uso

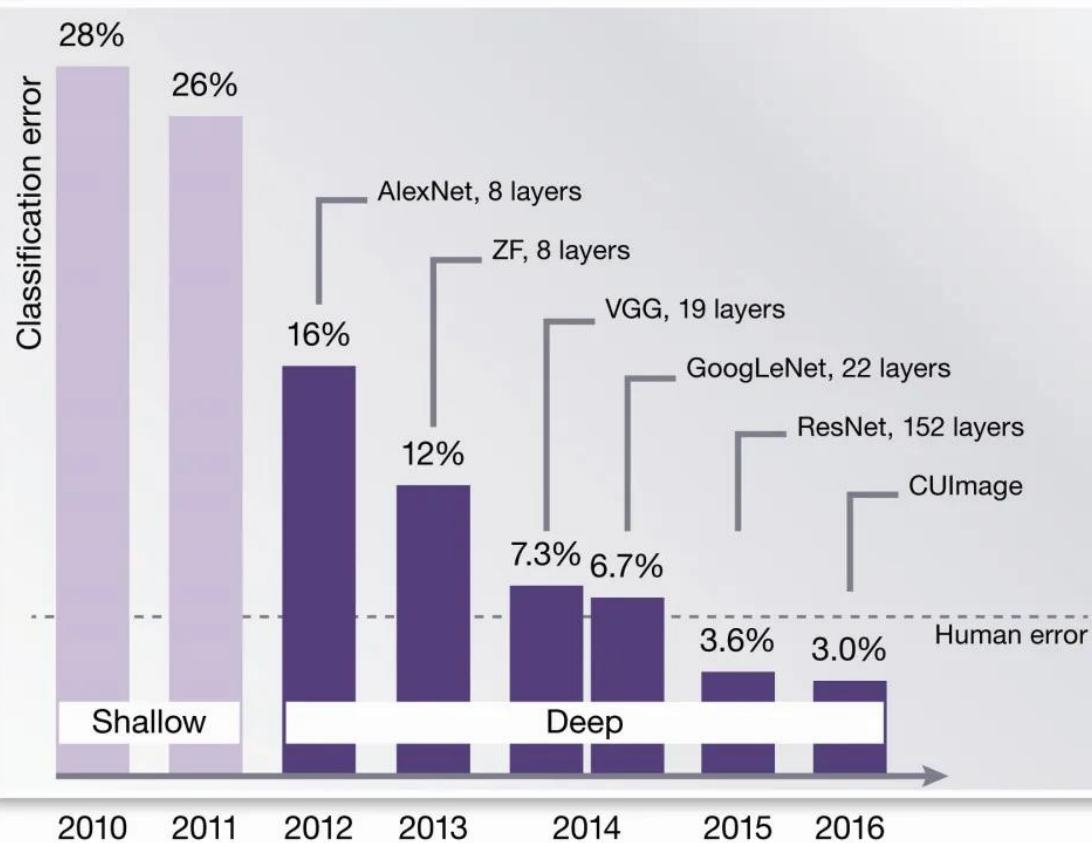
- Reconhecimento e Previsão de Imagem (Manufatura, Medicina, Pesquisa, Agricultura, etc.)
 - Classificação de textos (análise de sentimentos, tradução ou vinculação de entidades contextuais)
 - Automóveis e carros autônomos
 - Assistentes de voz de varejo e populares
 - Tradutores de voz para negócios e viagens
 - Publicidade preditiva
- 



Deep Learning – Algoritmos

- Convolutional Neural Networks (CNNs)
- Recurrent Neural Networks (RNNs)
- Long Short-Term Memory Networks (LSTMs)
- Generative Adversarial Networks (GANs)
- Self Organizing Maps (SOMs)
- Radial Basis Function Networks (RBFNs)
- Autoencoders

Deep Learning – ILSVRC





Deep Learning - Frameworks



- TensorFlow

- Keras

- PyTorch

- Theano

- Caffe

- Deeplearning4j

- MXNet

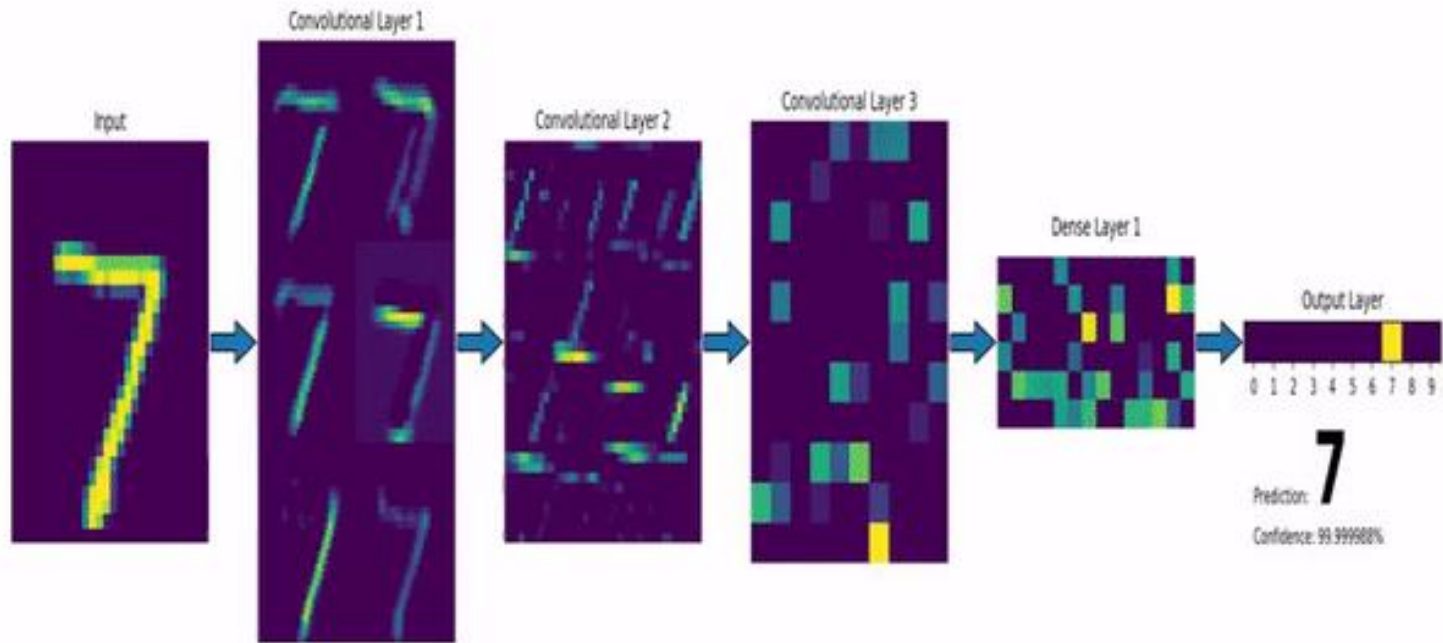
- Chainer



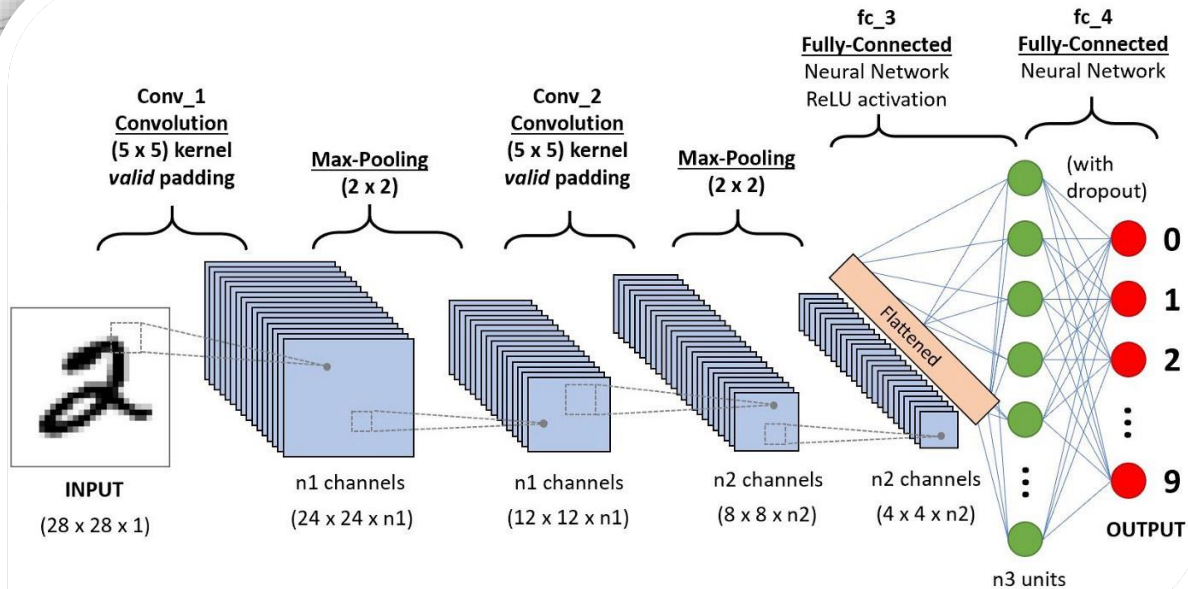
Convolutional Neural Networks (CNN)

02

CNN – Convolutional neural Networks



CNN - Arquitetura



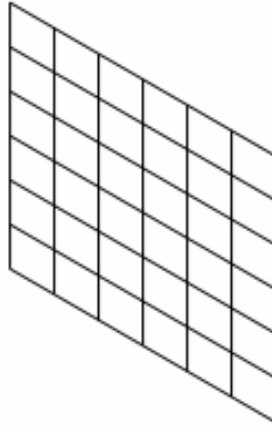
- Arquitetura de Deep Learning → Aprende diretamente dos dados.
- CNNs são particularmente úteis para encontrar padrões em imagens para reconhecer objetos.

CNN - Kernel or Filter or Feature Detectors

output

0	-1	0
-1	5	-1
0	-1	0

7	6	5	5	6	7
6	4	3	3	4	6
5	3	2	2	3	5
5	3	2	2	3	5
6	4	3	3	4	6
7	6	5	5	6	7



input

- Kernel nada mais é do que um filtro que é usado para extrair os recursos das imagens.
- <https://setosa.io/ev/image-kernels/>

$$Kernel = [i - k] + 1$$

Onde:

$I \rightarrow$ Tamanho da entrada

$K \rightarrow$ Tamanho do Kernel

CNN - Stride

- Stride é um parâmetro do filtro da rede neural que modifica a quantidade de movimento sobre a imagem ou vídeo.

$$Stride = [i - k/s] + 1$$

Onde:

I → Tamanho da entrada

K → Tamanho do Kernel

S → Stride

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

7x7

*

3	4	4
1	0	2
-1	0	3

3x3

=

91		

Stride = 2

CNN - Padding

- Refere-se ao número de pixels adicionados a uma imagem quando ela está sendo processada pelo kernel de uma CNN

$$Stride = [i - k + 2p/s] + 1$$

Onde:

I → Tamanho da entrada

K → Tamanho do Kernel

S → Stride

P → Padding

0 ₂	0 ₀	0 ₁	0	0	0	0
0 ₁	2 ₀	2 ₀	3	3	3	0
0 ₀	0 ₁	1 ₁	3	0	3	0
0	2	3	0	1	3	0
0	3	3	2	1	2	0
0	3	3	0	2	3	0
0	0	0	0	0	0	0

1	6	5
7	10	9
7	10	8

CNN - Pooling

1	3	2	1
2	9	1	1
1	3	2	3
5	6	1	2

4x4



9	2
6	3

2x2

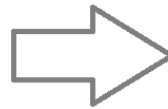
Hyperparameters:
 $f = 2$
 $s = 2$

- Pooling em redes neurais convolucionais é uma técnica para generalizar características extraídas por filtros convolucionais e ajudar a rede a reconhecer características independentes de sua localização na imagem.

CNN - Flatten

- O Flatten, ou achatamento é usado para converter todas as matrizes bidimensionais resultantes de mapas de recursos agrupados em um único vetor linear longo e contínuo.
- A matriz achatada é alimentada como entrada para a camada totalmente conectada para classificar a imagem.

1	1	0
4	2	1
0	2	1



1
1
0
4
2
1
0
2
1


Camadas usadas para construção de uma CNN

Camada
totalmente
conectada (FC)


As redes neurais convolucionais são diferenciadas de outras redes neurais por seu desempenho superior com entradas de sinal de imagem, fala ou áudio. Eles têm três tipos principais de camadas, que são

Camada
convolucional

Camada de
pooling



Camadas usadas para construção de uma CNN



Camada Convulacional

- Essa camada é a primeira camada usada para extrair os vários recursos das imagens de entrada. Nesta camada, usamos um filtro ou método Kernel para extrair recursos da imagem de entrada.

- O objetivo principal desta camada é diminuir o tamanho do features map.
- Isso é realizado diminuindo as conexões entre as camadas e operando independentemente em cada mapa de recursos.

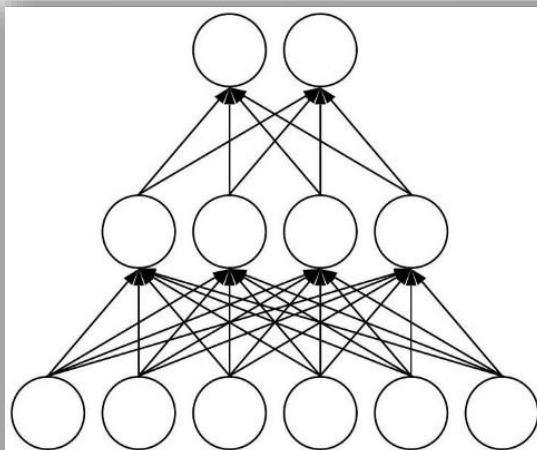
Camada Pooling

Camada Fully-Connected

- A camada Totalmente Conectada (FC) consiste nos pesos e tendências junto com os neurônios e é usada para conectar os neurônios entre duas camadas diferentes.
- Essas camadas geralmente são colocadas antes da camada de saída e formam as últimas camadas de uma Arquitetura CNN.

CNN - Dropout

- A camada Dropout é uma máscara que anula a contribuição de alguns neurônios para a próxima camada e deixa inalterados todos os outros.

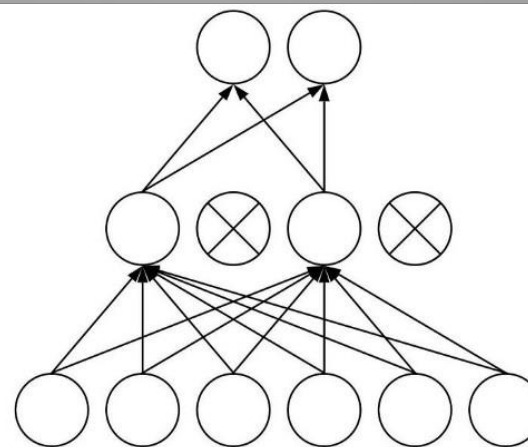


Classification

Hidden layer

Input layer

Without Dropout



Classification

Dropout on
hidden layer

Input layer

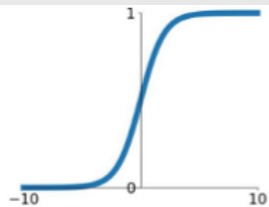
With Dropout

CNN – Função de Ativação

- Uma função de ativação decide se um neurônio deve ser ativado ou não. Isso significa que ele decidirá se a entrada do neurônio para a rede é importante ou não no processo de previsão.
- Existem várias funções de ativação comumente usadas, como as funções ReLU, Softmax, tanH e Sigmoid. Cada uma dessas funções tem um uso específico.

Sigmoid

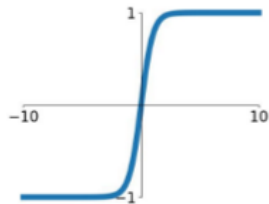
$$\sigma(x) = \frac{1}{1+e^{-x}}$$



Para uma classificação binária no modelo CNN

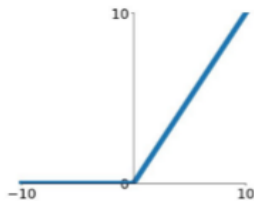
CNN – Função de Ativação

tanh
 $\tanh(x)$



A função tanh é muito semelhante à função sigmóide. A única diferença é que ele é simétrico em torno da origem. O intervalo de valores, neste caso, é de -1 a 1.

ReLU
 $\max(0, x)$



a principal vantagem de usar a função ReLU sobre outras funções de ativação é que ela não ativa todos os neurônios ao mesmo tempo.

$$\frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}$$

Ele é usado em regressão logística multinomial e é frequentemente usado como a última função de ativação de uma rede neural para normalizar a saída de uma rede para uma distribuição de probabilidade sobre as classes de saída previstas.

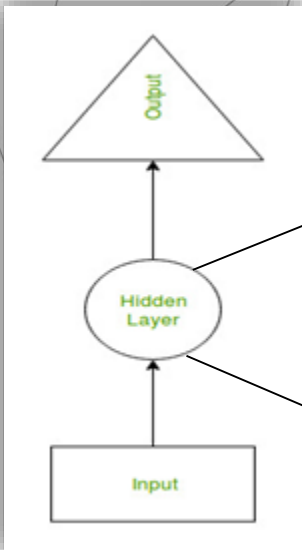




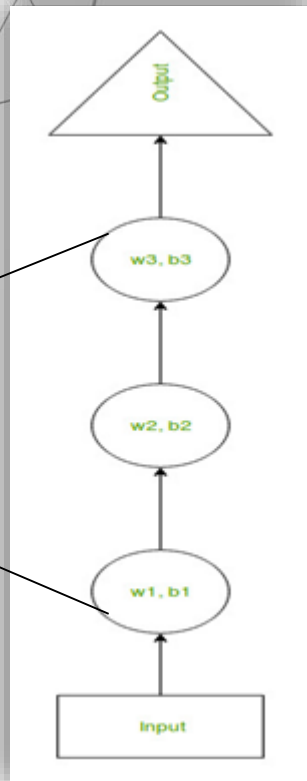
03

Recurrent Neural Network

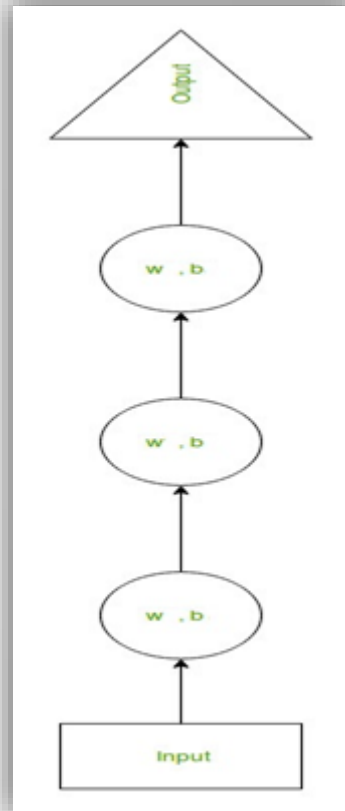
RNN-0 que é?



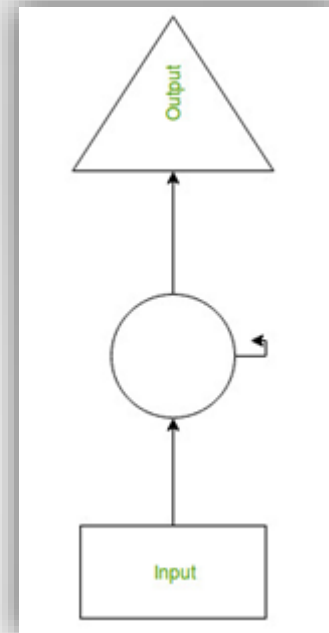
MLP Simples



MLP Simples – Pesos e bias diferentes

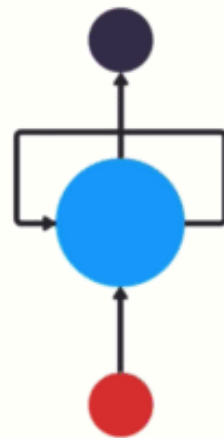
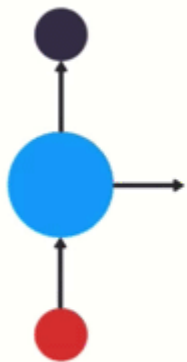


MLP Simples – Pesos e bias iguais

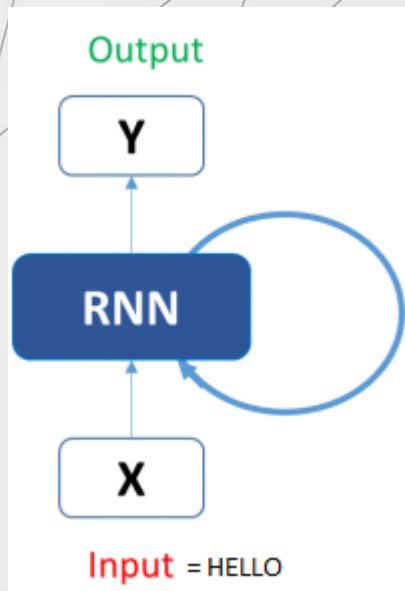


MLP Simples – relação da entrada atual com a entrada anterior

RNN-0 que é?



RNN-Mais detalhada



Estado atual:

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

Onde:

$h_t \rightarrow$ Estado Atual

$h_{t-1} \rightarrow$ Estado Anterior

$x_t \rightarrow$ Estado de Entrada

Função de Ativação:

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

Onde:

$W_{hh} \rightarrow$ peso do neurônio recorrente

$W_{xh} \rightarrow$ Peso do Neurônio de entrada

Estado da saída:

$$y_t = W_{hy}h_t$$

Onde:

$y_t \rightarrow$ Saída

$W_{hy} \rightarrow$ Peso da Camada de Saída

1. x_t é fornecido à rede

2. Em seguida, calculamos seu estado atual usando uma combinação da entrada atual e o estado anterior, ou seja, calculamos h_t

3. O h_t atual se torna h_{t-1} para o próximo passo de tempo

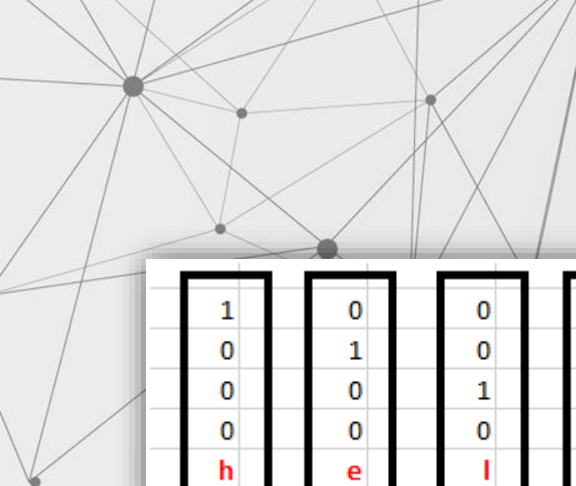
4. Podemos dar tantos passos de tempo quanto o problema exigir e combinar as informações de todos os estados anteriores

5. Uma vez que todas as etapas de tempo são concluídas, o estado atual final é usado para calcular a saída y_t

6. A saída é então comparada com a saída real e o erro é gerado

7. O erro é então retropropagado para a rede para atualizar os pesos.

RNN-Mais detalhada



1	0	0	0
0	1	0	0
0	0	1	1
0	0	0	0
h	e	l	l

Entradas codificadas em one-hot-encoding

wxh			
0.287027	0.84606	0.572392	0.486813
0.902874	0.871522	0.691079	0.18998
0.537524	0.09224	0.558159	0.491528

Pesos inicializados aleatoriamente

Passo 1:

- Letra "h" → Calcular $W_{xh}x_t$



wxh											
0.287027	0.84606	0.572392	0.486813	×	<table><tr><td>1</td></tr><tr><td>0</td></tr><tr><td>0</td></tr><tr><td>0</td></tr><tr><td>h</td></tr></table>	1	0	0	0	h	=
1											
0											
0											
0											
h											
0.902874	0.871522	0.691079	0.18998								
0.537524	0.09224	0.558159	0.491528								
						<table><tr><td>0.287027</td></tr><tr><td>0.902874</td></tr><tr><td>0.537524</td></tr></table>	0.287027	0.902874	0.537524		
0.287027											
0.902874											
0.537524											

RNN-Mais detalhada

Passo 2:

- Neurônio Recorrente $\rightarrow W_{hh}h_{t-1} + W_{xh}x_t$
- Para a letra “h”, o estado anterior é $[0,0,0]$, pois não há letra anterior a ele.

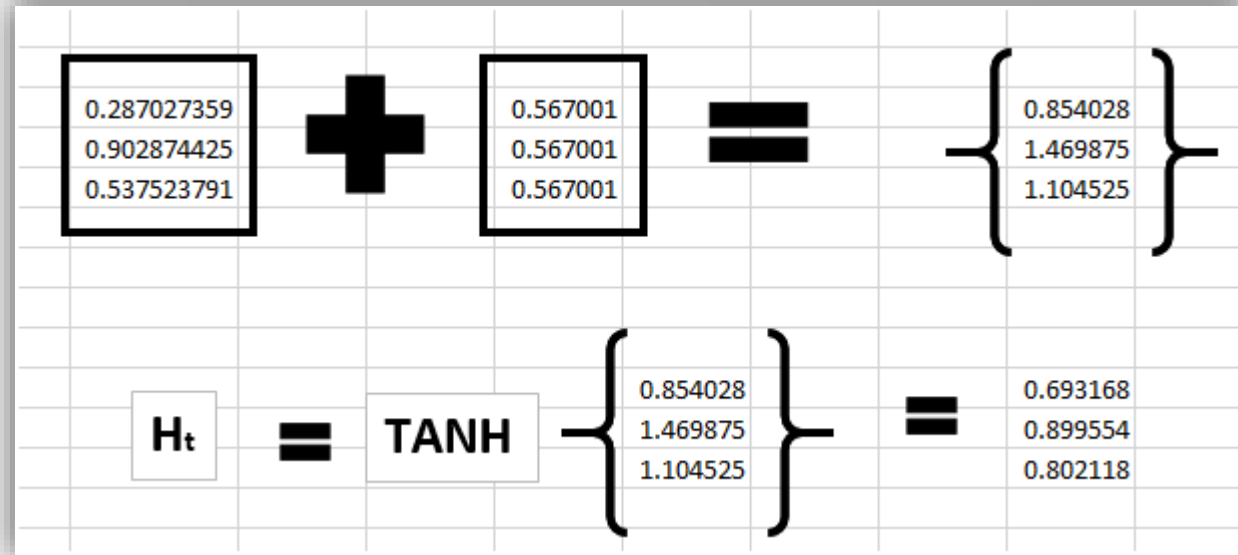
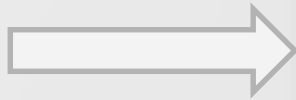
	Weight(w _{hh})	bias					
	0.427043	0.567001		×		=	
					0		0.567001
					0		0.567001
					0		0.567001
					h _{t-1}		

RNN-Mais detalhada

Passo 3:

- Obter o estado atual

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$



RNN-Mais detalhada



Passo 4:

- A letra “e” é fornecida a rede.
- A saída processada de h_t , agora se torna h_{t-1} , enquanto o e codificado one-hot é x_t .
- Calcula-se o estado atual h_t .

$W_{hh} * H_{t-1} + \text{Bias}$	=	0.427043	×	<table border="1"><tr><td>0.69316804</td></tr><tr><td>0.89955366</td></tr><tr><td>0.8021184</td></tr></table>	0.69316804	0.89955366	0.8021184	+	<table border="1"><tr><td>0.567001</td></tr></table>	0.567001	=	<table border="1"><tr><td>0.863013</td></tr><tr><td>0.951149</td></tr><tr><td>0.90954</td></tr></table>	0.863013	0.951149	0.90954
0.69316804															
0.89955366															
0.8021184															
0.567001															
0.863013															
0.951149															
0.90954															

w _{xh}				×	<table border="1"><tr><td>0</td></tr><tr><td>1</td></tr><tr><td>0</td></tr><tr><td>0</td></tr><tr><td>e</td></tr></table>	0	1	0	0	e	=	<table border="1"><tr><td>0.84606</td></tr><tr><td>0.871522</td></tr><tr><td>0.09224</td></tr></table>	0.84606	0.871522	0.09224
0															
1															
0															
0															
e															
0.84606															
0.871522															
0.09224															
0.287027359	0.84606	0.572392	0.486813												
0.902874425	0.871522	0.691079	0.18998												
0.537523791	0.09224	0.558159	0.491528												





RNN-Mais detalhada

Passo 5:

- Calculando h_t para letra “e”.
- isso se tornaria h_{t-1} para o próximo estado e o neurônio recorrente usaria isso junto com o novo caractere para prever o próximo.

H_t	=	TANH	{	<table border="1"><tr><td>0.863013</td></tr><tr><td>0.951149</td></tr><tr><td>0.90954</td></tr></table>	0.863013	0.951149	0.90954	+	<table border="1"><tr><td>0.84606</td></tr><tr><td>0.871522</td></tr><tr><td>0.09224</td></tr></table>	0.84606	0.871522	0.09224	}	=	<table border="1"><tr><td>0.93653372</td></tr><tr><td>0.94910403</td></tr><tr><td>0.76234056</td></tr></table>	0.93653372	0.94910403	0.76234056
0.863013																		
0.951149																		
0.90954																		
0.84606																		
0.871522																		
0.09224																		
0.93653372																		
0.94910403																		
0.76234056																		

- Em cada estado, a rede neural recorrente também produziria a saída

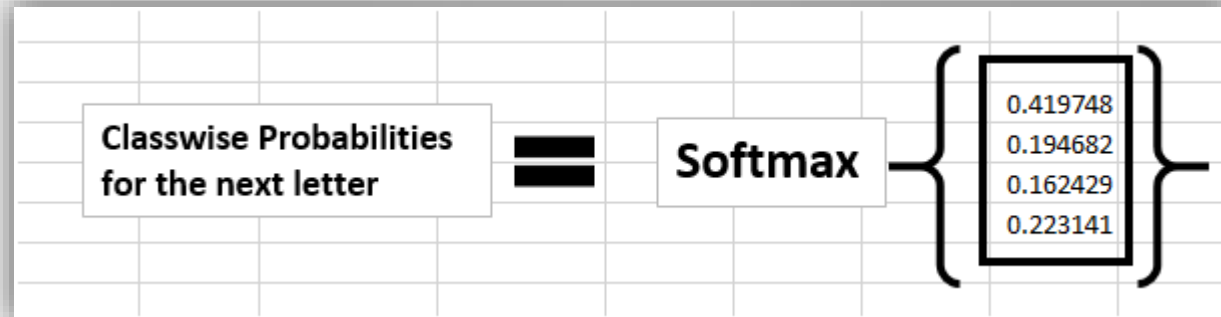
why				Ht		yt
0.37168	0.974829459	0.830034886		0.936534		1.90607732
0.39141	0.282585823	0.659835709		0.949104		1.13779113
0.64985	0.09821557	0.334287084		0.762341		0.95666016
0.91266	0.32581642	0.144630018				1.27422602

RNN-Mais detalhada

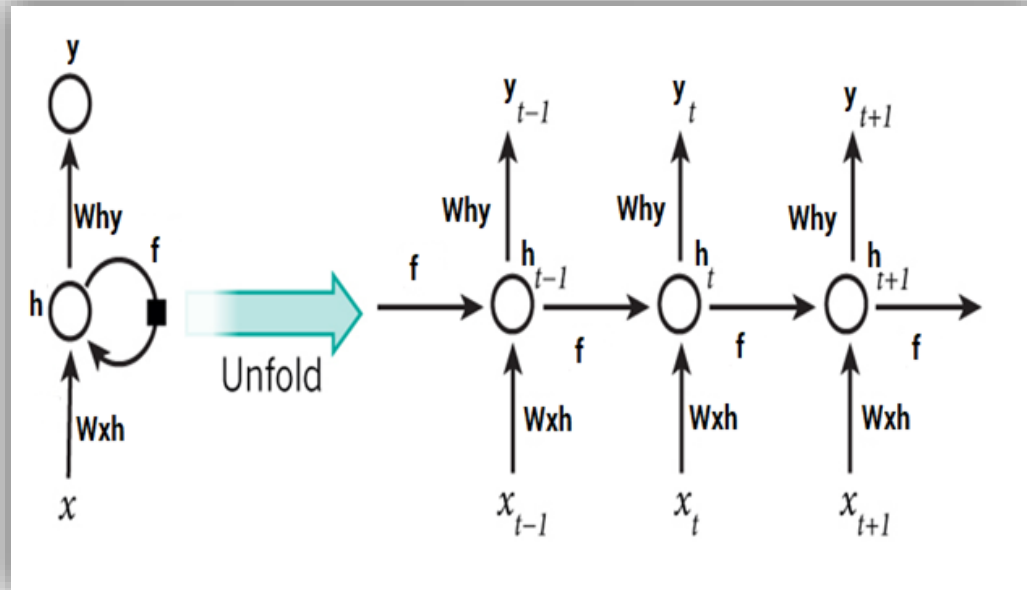


Passo 7:

- A probabilidade de uma determinada letra do vocabulário pode ser calculada aplicando a função softmax.



RNN - BackPropagation



- $y_t \rightarrow$ Valor previsto
- $\bar{y}_t \rightarrow$ Valor Real
- erro total é apenas a soma dos erros em cada etapa de tempo

RNN - BackPropagation

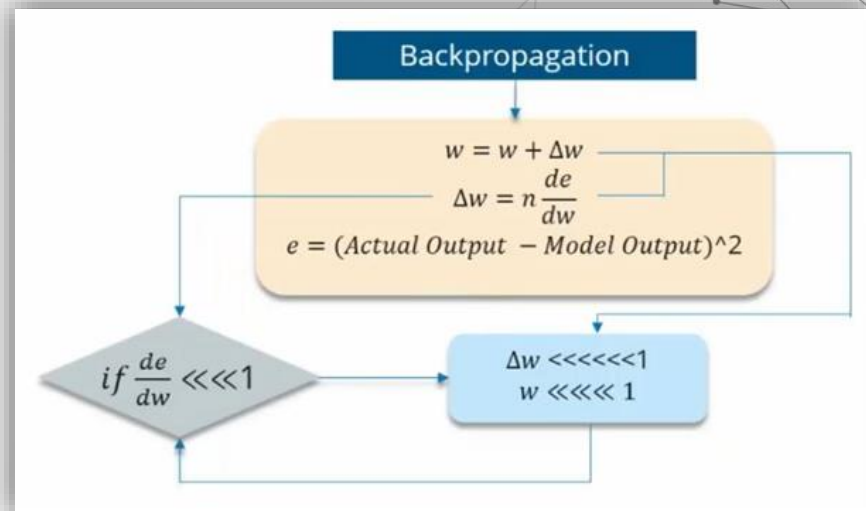
- O erro é calculado primeiro usando a saída atual e a saída real
- Lembre-se de que a rede é desenrolada para todas as etapas de tempo
- Para a rede desenrolada, sabemos:
 1. O gradiente é calculado para cada passo de tempo em relação ao peso referente
 2. O peso é o mesmo para todos os passos de tempo
 3. Os gradientes podem ser combinados para todos os passos de tempo, então atualizados tanto para o neurônio recorrente quanto para as camadas densas

RNN - Vanishing & Exploding Gradient Problem

- Redes neurais recorrentes usam um **algoritmo de retropropagação** para treinamento, **mas é aplicado** para cada **timestamp (tempo de passo)**. É comumente conhecido como **Back-propagation Through Time (BTT)**.
- Existem **alguns problemas** com a retropropagação, como:
 1. Vanishing Gradient
 2. Exploding Gradient

RNN - Vanishing & Exploding Gradient Problem

- O problema do **Vanishing Gradient** é causado pelo fato de que, as RNNs não tem a capacidade de propagar informações úteis do gradiente das camadas mais extremas do modelo, de volta para as camadas próximas à extremidade de entrada.
- Ou seja, enquanto o **algoritmo de retropropagação avança para trás** (camada de saída → camada de entrada), os **gradientes tendem a diminuir**, tornando-se cada vez menores até se **aproximarem de zero**. Isso acaba deixando os pesos das camadas iniciais ou inferiores praticamente inalterados. Nesta situação, o gradiente descendente nunca acaba convergindo para o ótimo.
- Isso implica que os gradientes são minúsculos, o que faria com que o aprendizado fosse muito lento.



RNN - Vanishing & Exploding Gradient Problem

Como você sabe se o seu modelo está sofrendo com o problema do Vanishing?

- Os parâmetros das camadas superiores mudam bastante, enquanto os parâmetros das camadas inferiores mudam pouco (ou não mudam nada).
- Os pesos do modelo podem se tornar 0 durante o treinamento.
- O modelo aprende em um ritmo particularmente lento e o treinamento pode estagnar em uma fase muito inicial após apenas algumas iterações.

RNN - Vanishing & Exploding Gradient Problem

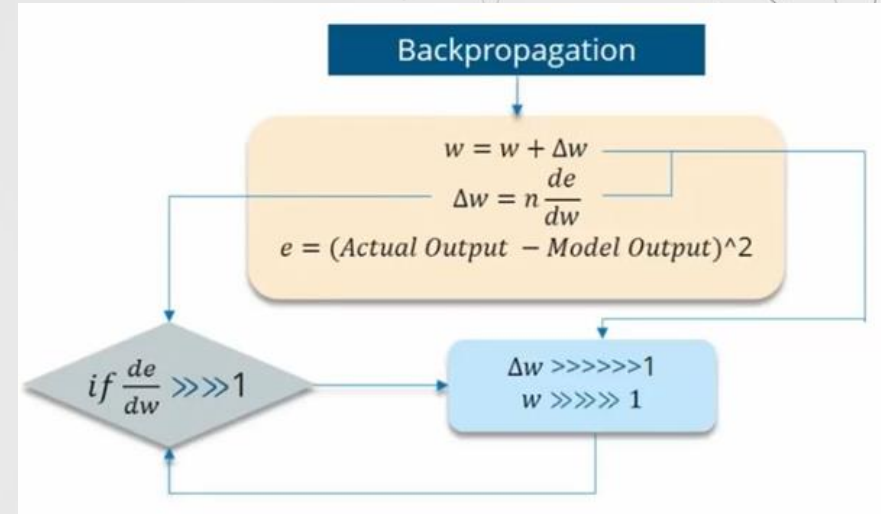
Por que o problema do Vanishing é significativo?

- O problema do Vanishing faz com que os gradientes encolham. Mas, se um gradiente for pequeno, não será possível atualizar efetivamente os pesos e vieses das camadas iniciais a cada sessão de treinamento.
- Essas camadas iniciais são vitais para reconhecer os elementos centrais dos dados de entrada, portanto, se seus pesos e vieses não forem atualizados adequadamente, é possível que toda a rede seja imprecisa.

RNN - Vanishing & Exploding Gradient Problem

Exploding Gradient:

- Ocorre quando grandes gradientes de erro se acumulam e resultam em atualizações muito grandes nos pesos.
- Os gradientes são usados durante o treinamento para atualizar os pesos da rede, mas normalmente esse processo funciona melhor quando essas atualizações são pequenas e controladas.
- Quando as magnitudes dos gradientes se acumulam, é provável que ocorra uma rede instável, o que pode causar resultados ruins de previsão ou até mesmo um modelo que não relata nada de útil.



RNN - Vanishing & Exploding Gradient Problem

Como você sabe se o seu modelo está sofrendo com o problema do Exploding?

- O modelo é instável, resultando em grandes mudanças na perda de atualização para atualização.
- A perda/pesos do modelo vai para NaN durante o treinamento.
- Os pesos do modelo rapidamente se tornam muito grandes durante o treinamento.
- Os valores do gradiente de erro estão consistentemente acima de 1,0 para cada nó e camada durante o treinamento.

RNN - Vanishing & Exploding Gradient Problem

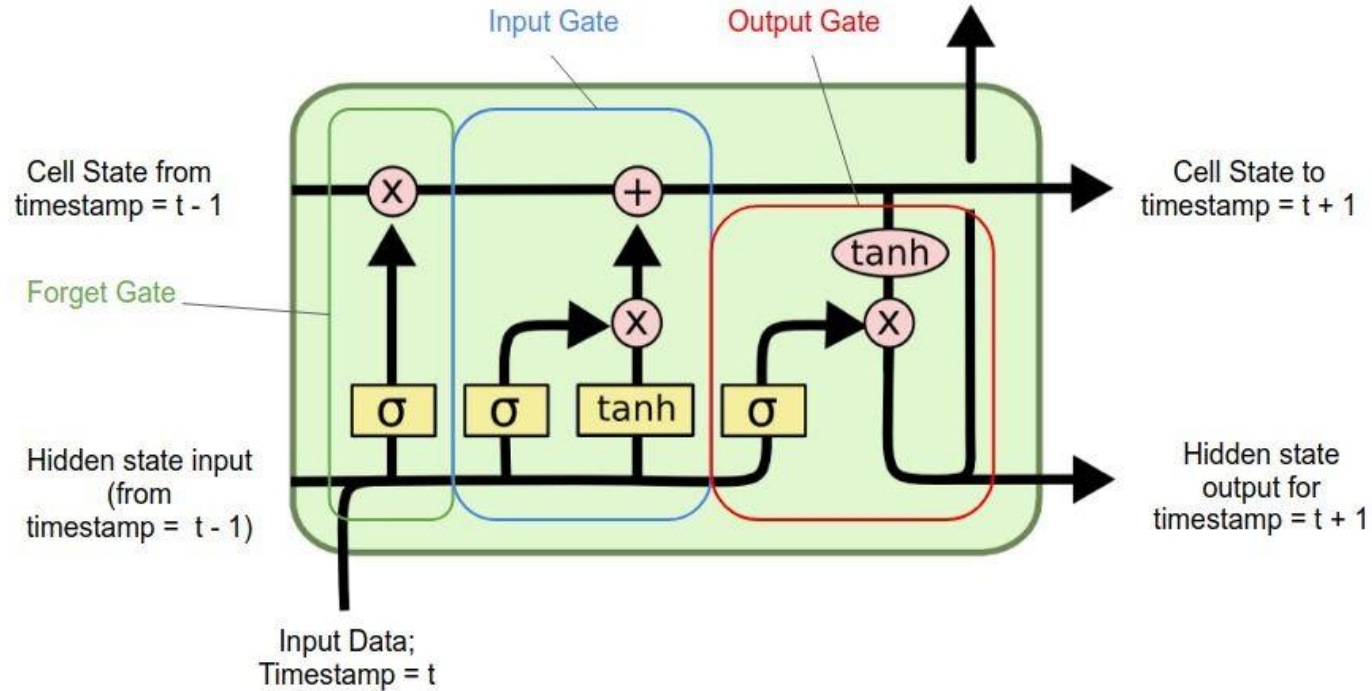
Como você supera o problema do gradiente de fuga?

- Residual neural networks (ResNets)
- Multi-level hierarchy
- **Long short term memory (LSTM)**
- ReLU

04 LONG SHORT TERM MEMORY



LSTM – Estrutura



LSTM – Funcionamento

Passo 1 Forget Gate:

- Responsável por identificar as informações que não são necessárias e serão descartadas do estado da célula.

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$$

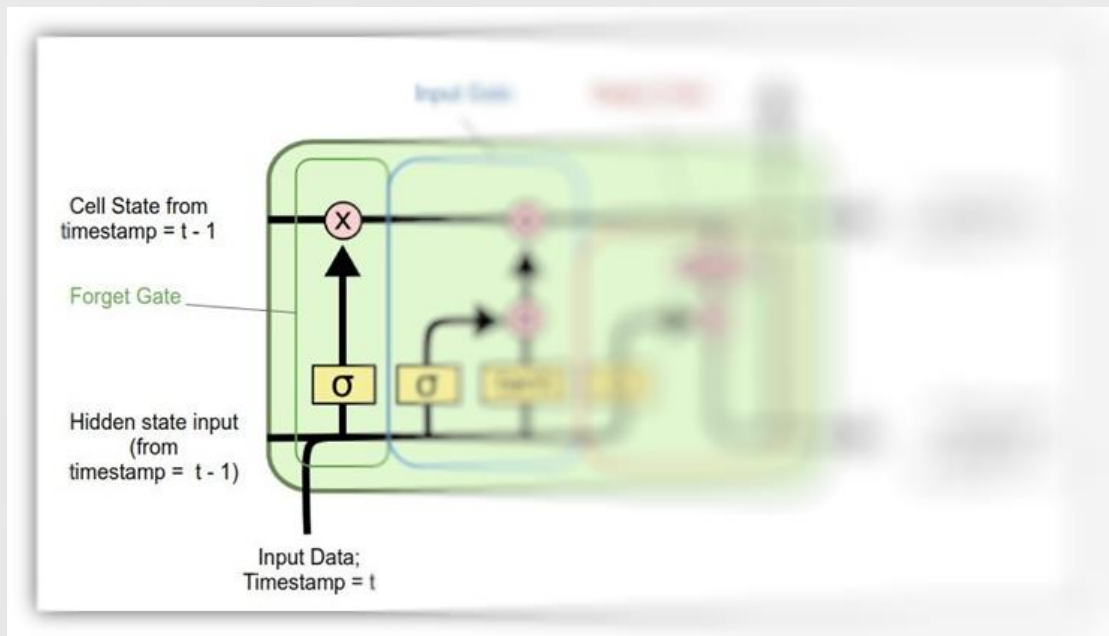
Onde:

$W_f \rightarrow$ Peso

$h_{t-1} \rightarrow$ Saída do tempo anterior

$x_t \rightarrow$ Nova entrada

$b_f \rightarrow$ bias



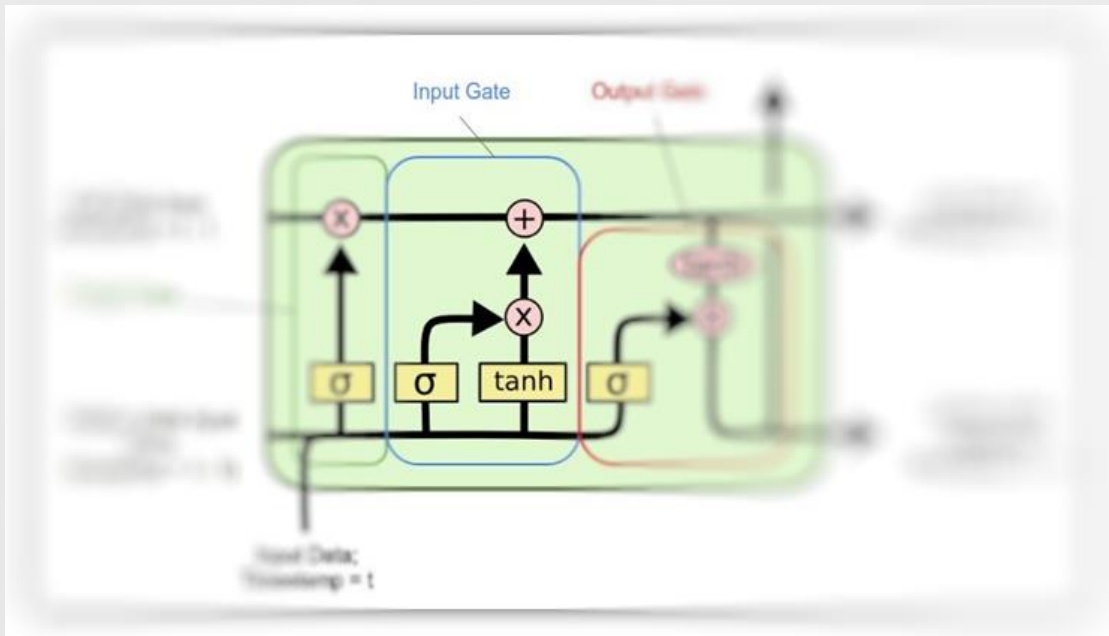
LSTM – Funcionamento

Passo 2 Input Gate:

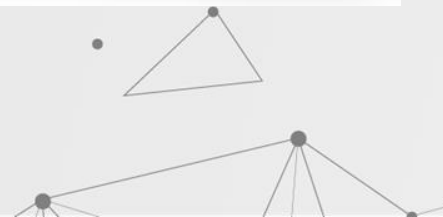
- O próximo passo é decidir quais novas informações vamos armazenar no estado da célula

$$\begin{aligned} i_t &= \sigma(W_i[h_{t-1}, x_t] + b_i) \\ c_t &= \tanh(W_c[h_{t-1}, x_t] + b_c) \end{aligned}$$

- Uma camada sigmóide chamada.
- A camada tanh cria um vetor de novos valores candidatos que podem ser adicionados ao estado.



A entrada do **timestamp** anterior e a nova entrada são passadas através de uma função sigmóide que dá o valor $i(t)$. Este valor é então multiplicado por $c(t)$ e então adicionado ao estado da célula.



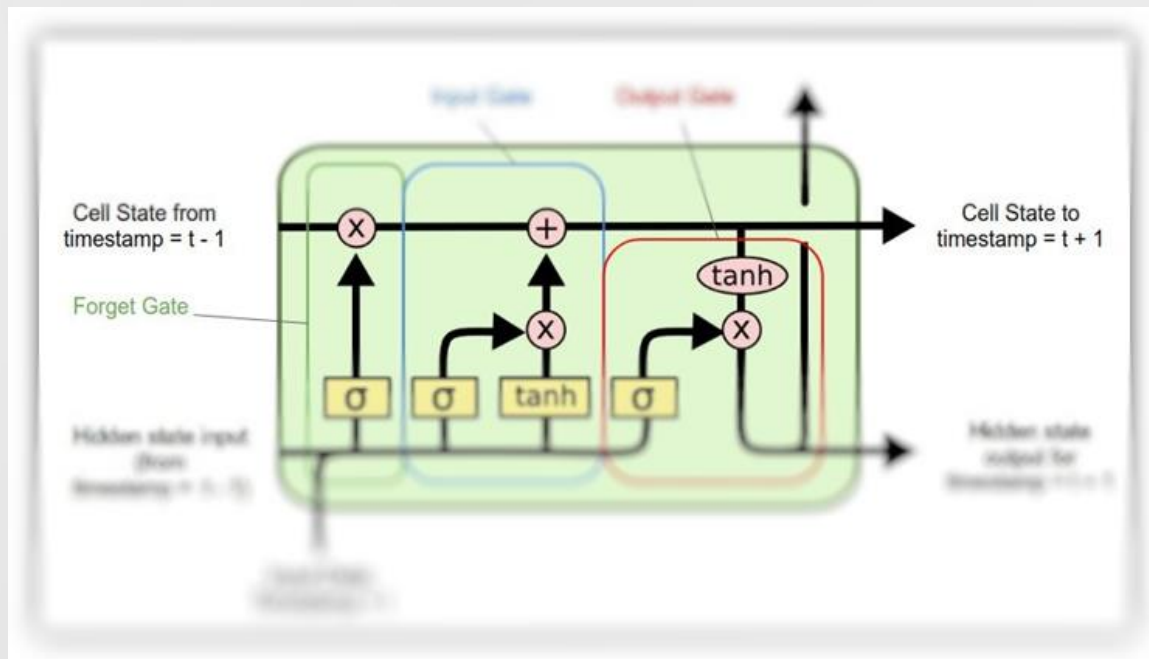
LSTM – Funcionamento

Passo 3 Input Gate:

- É atualizado o antigo estado da célula c_{t-1} , para o novo estado da célula c_t .
- Primeiro, multiplicamos o estado antigo (c_{t-1}) por f_t , esquecendo as coisas que foi decidido deixar para trás anteriormente.

$$c_t = f_t * c_{t-1} + i_t * c_t$$

- Foi adicionado $i_t * c_t$. Esses são os novos valores para atualizar cada valor de estado.
- Na segunda etapa, foi os dados que só são necessários nessa fase.
- Na terceira etapa, foi realmente implementada



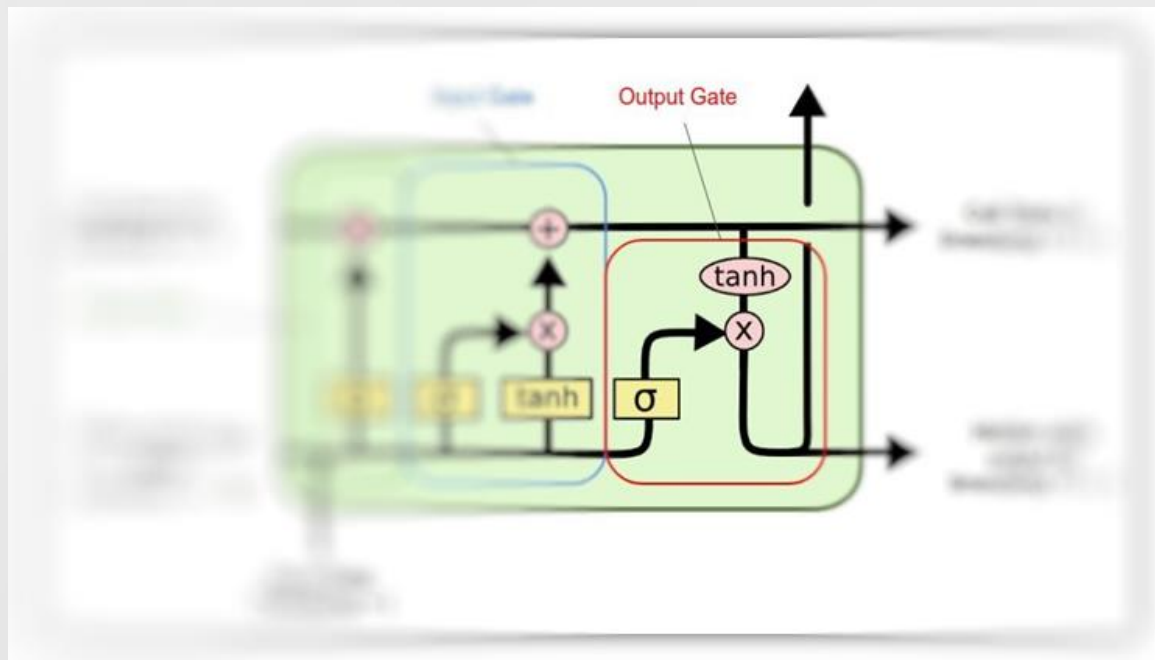
LSTM – Funcionamento

Passo 4 Output Gate:

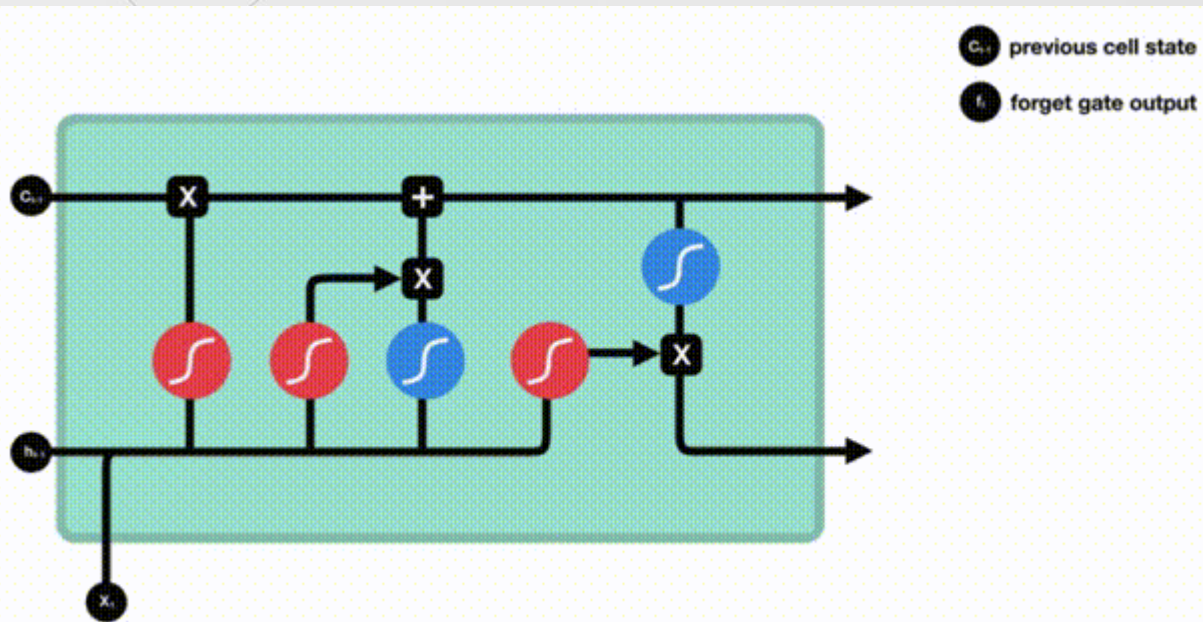
Vamos executar uma camada sigmoid para decidir quais partes do estado da célula vai produzir. Em seguida, colocamos o estado da célula em \tanh . Mais a frente, multiplicamos pela saída da porta σ , de modo que é produzido apenas as partes que atualizáveis.

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$
$$h_t = o_t * \tanh(c_t)$$

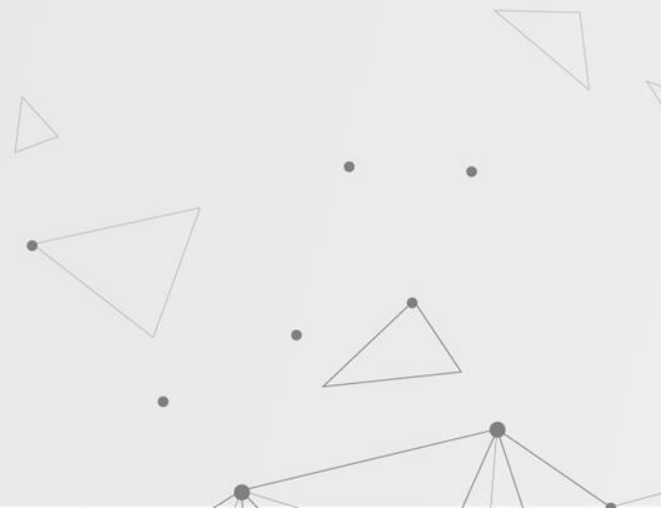
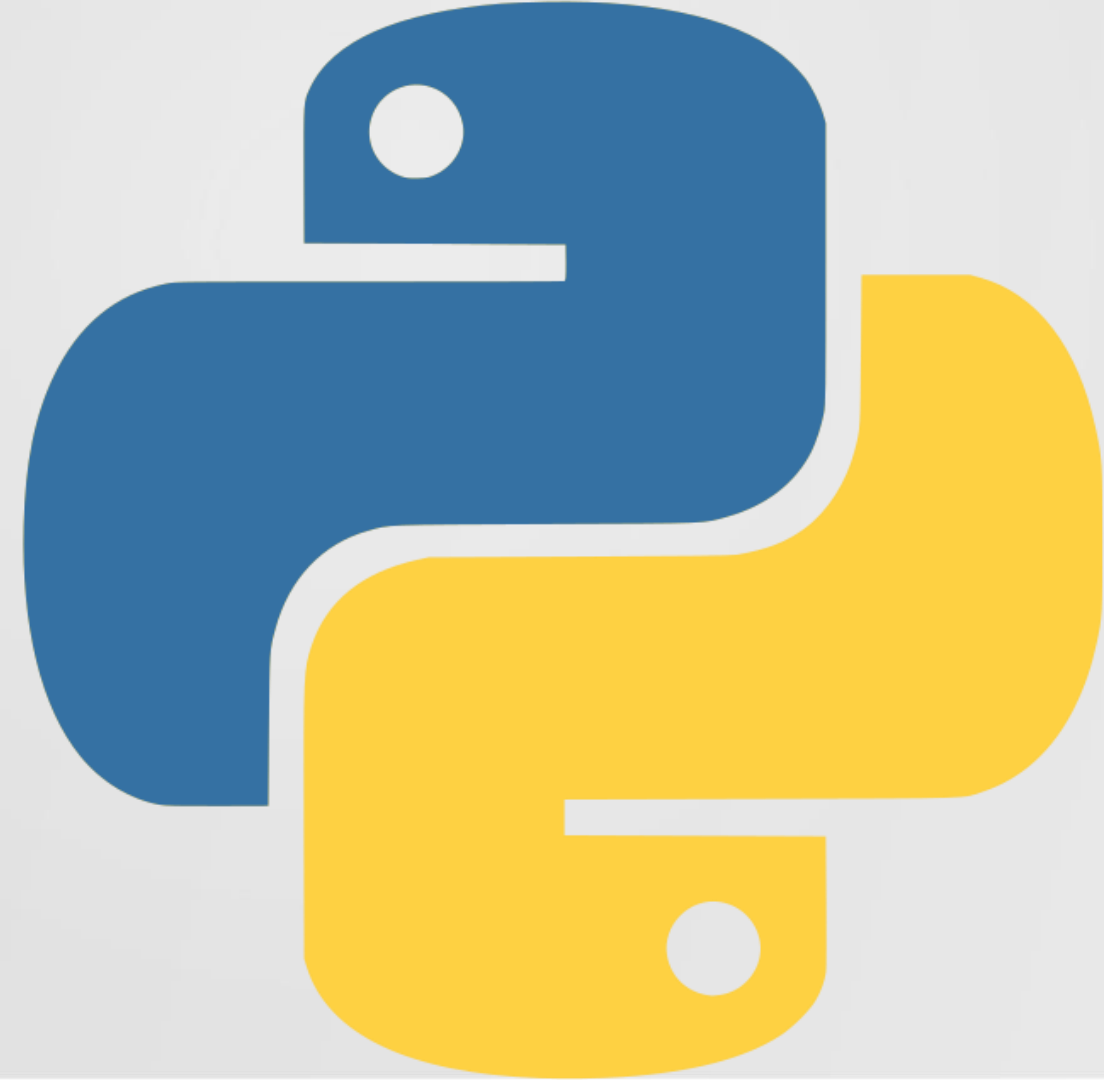
- O cálculo nesta etapa é bastante simples, o que eventualmente leva à saída.
- No entanto, a saída consiste apenas nas saídas que foram decididas a serem transportadas nas etapas anteriores e não em todas as saídas de uma só vez.



LSTM - Funcionamento



1. Na primeira etapa, descobrimos o que precisava ser descartado.
2. A segunda etapa consistiu em quais novas entradas são adicionadas à rede.
3. O terceiro passo foi combinar as entradas obtidas anteriormente para gerar os novos estados da célula.
4. Por fim, chegamos à saída conforme a necessidade.



05

Alexnet



História da AlexNet

- AlexNet foi projetado principalmente por Alex Krizhevsky. É uma Rede Neural Convolutacional ou CNN.
- **ImageNet Large Scale Visual Recognition Challenge: top-5 de 15,3%.** Isso foi 10,8% menor do que o vice-campeão.
- O principal resultado do artigo original foi que a profundidade do modelo era absolutamente necessária para seu alto desempenho.
- Isso era bastante caro computacionalmente, mas foi viabilizado devido a GPUs durante o treinamento.



AlexNet Arquitetura

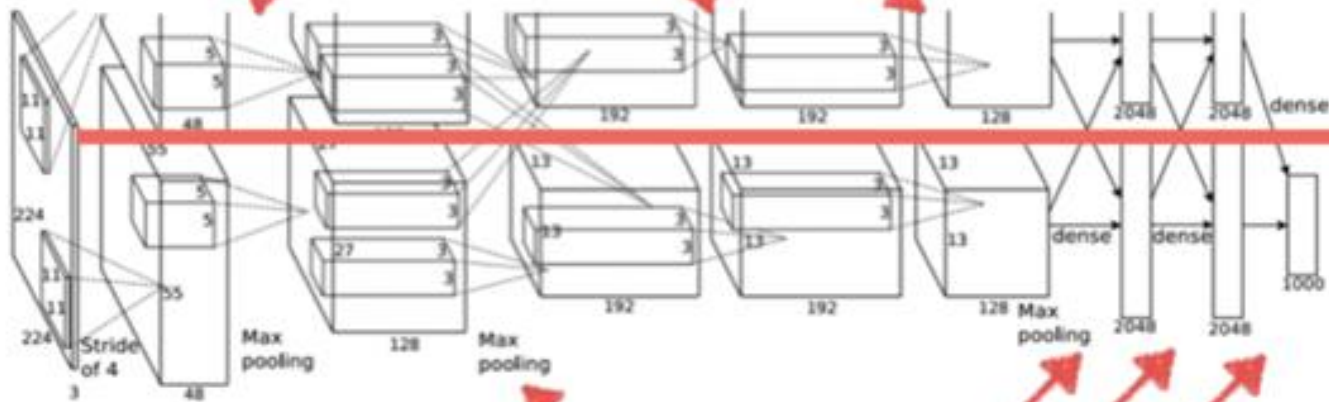
AlexNet foi a primeira rede convolucional que usou GPU para aumentar o desempenho.

1. A arquitetura AlexNet consiste em 5 camadas convolucionais, 3 camadas de pool máximo, 2 camadas de normalização, 2 camadas totalmente conectadas e 1 camada softmax.
2. Cada camada convolucional consiste em filtros convolucionais e uma função de ativação não linear ReLU.
3. As camadas de pooling são usadas para realizar o pooling máximo.
4. O tamanho da entrada é fixo devido à presença de camadas totalmente conectadas.
5. O tamanho da entrada é mencionado na maioria dos lugares como $224 \times 224 \times 3$, mas devido a algum preenchimento que acontece, acaba sendo $227 \times 227 \times 3$.
6. O AlexNet em geral tem 60 milhões de parâmetros.

AlexNet Arquitetura

GPU #1

intra-GPU connections



GPU #2

inter-GPU connections

AlexNet – Mais detalhes

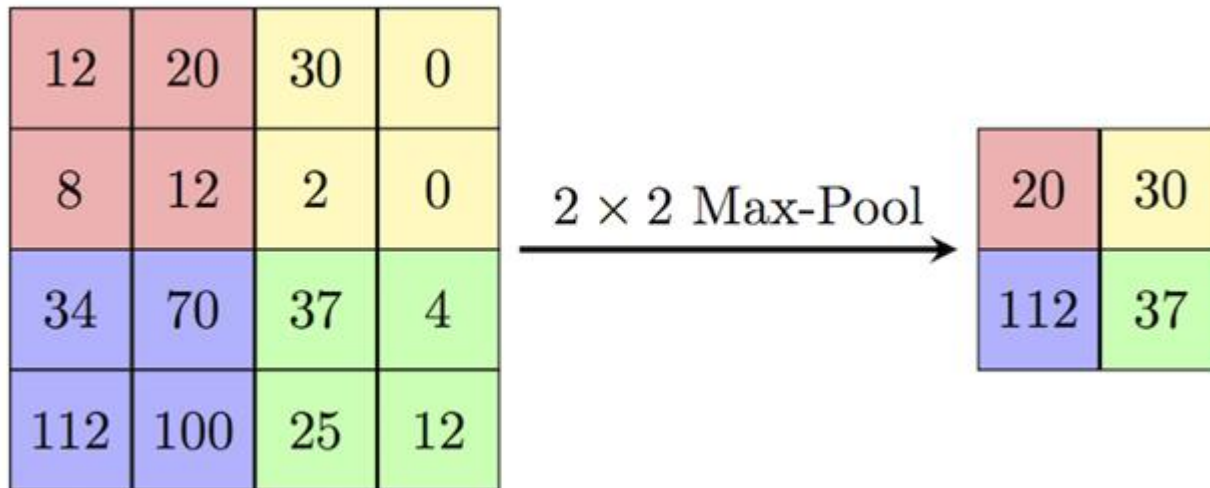
O modelo que venceu a competição possuía detalhes específicos -

1. ReLU é uma função de ativação
2. Camadas de normalização usadas (que não são mais comuns)
3. Tamanho do lote de 128
4. Momentum SGD como algoritmo de aprendizado
5. Aumento de dados usando artifícios
como flipping, jittering, cropping, normalização de cores, etc.
6. Montagem de modelos para obter os melhores resultados.

O AlexNet foi treinado em uma GPU GTX 580 com apenas 3 GB de memória que não cabia em toda a rede. Assim, a rede foi dividida em 2 GPUs, com metade dos neurônios (mapas de recursos) em cada GPU.

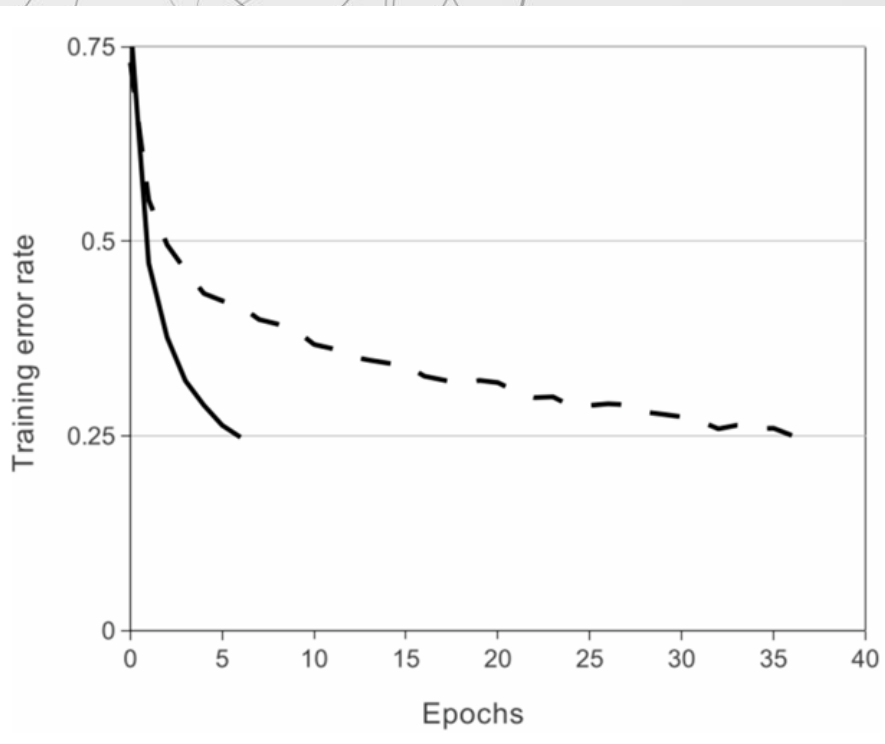


AlexNet Key Features



Overlapping Max Pooling

AlexNet Key Features

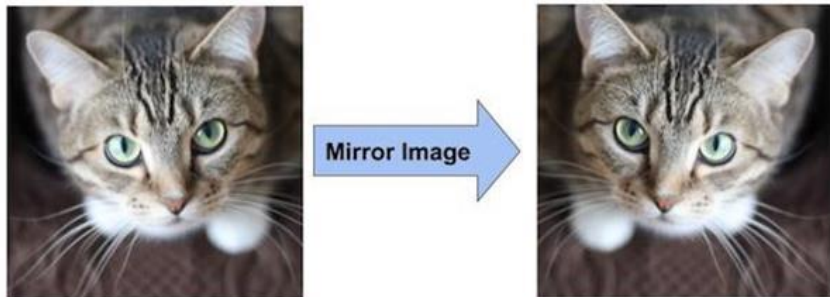


ReLU Nonlinearity

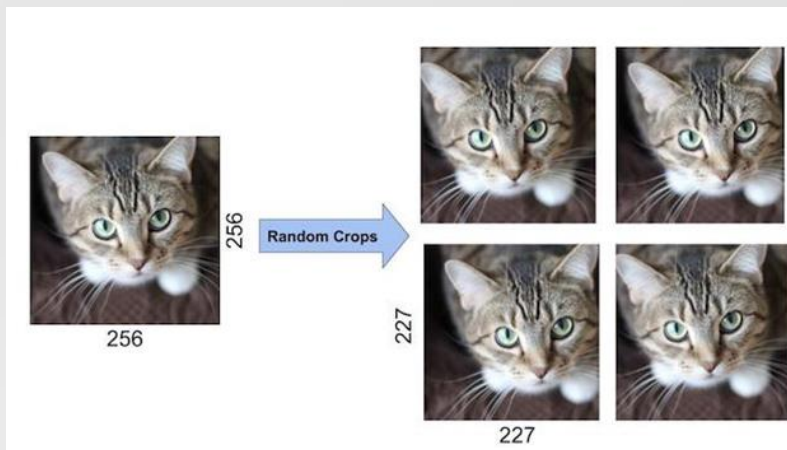
Usando a não-linearidade do ReLU, o AlexNet nos mostra que CNNs profundas podem ser treinadas muito mais rapidamente com a ajuda de funções de ativação de saturação, como Tanh ou Sigmoid.

A figura mostra que com a ajuda de ReLUs, AlexNet pode atingir uma taxa de erro de treinamento de 25%. Isso é seis vezes mais rápido do que uma rede equivalente que usa tanh(curva pontilhada).

AlexNet Key Features

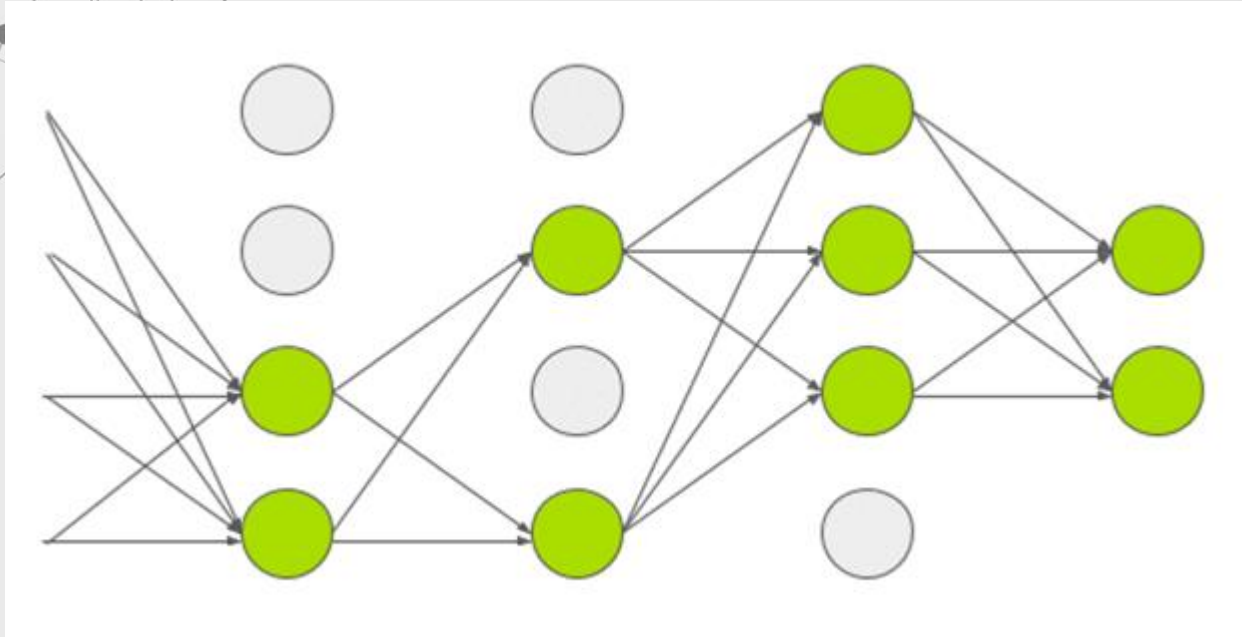


Data Augmentation by Mirroring

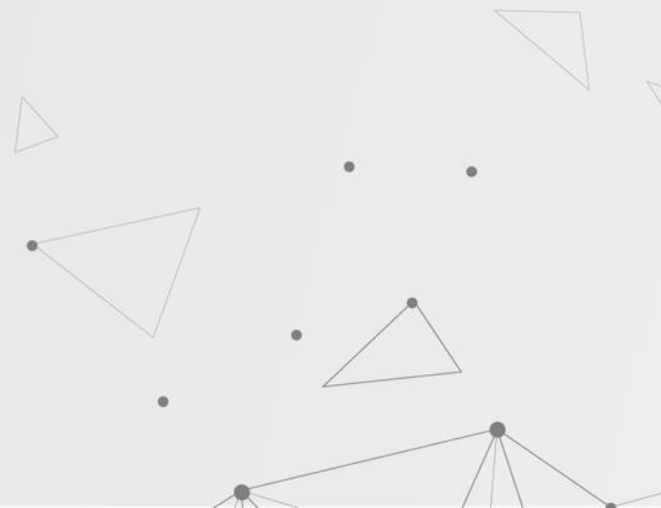
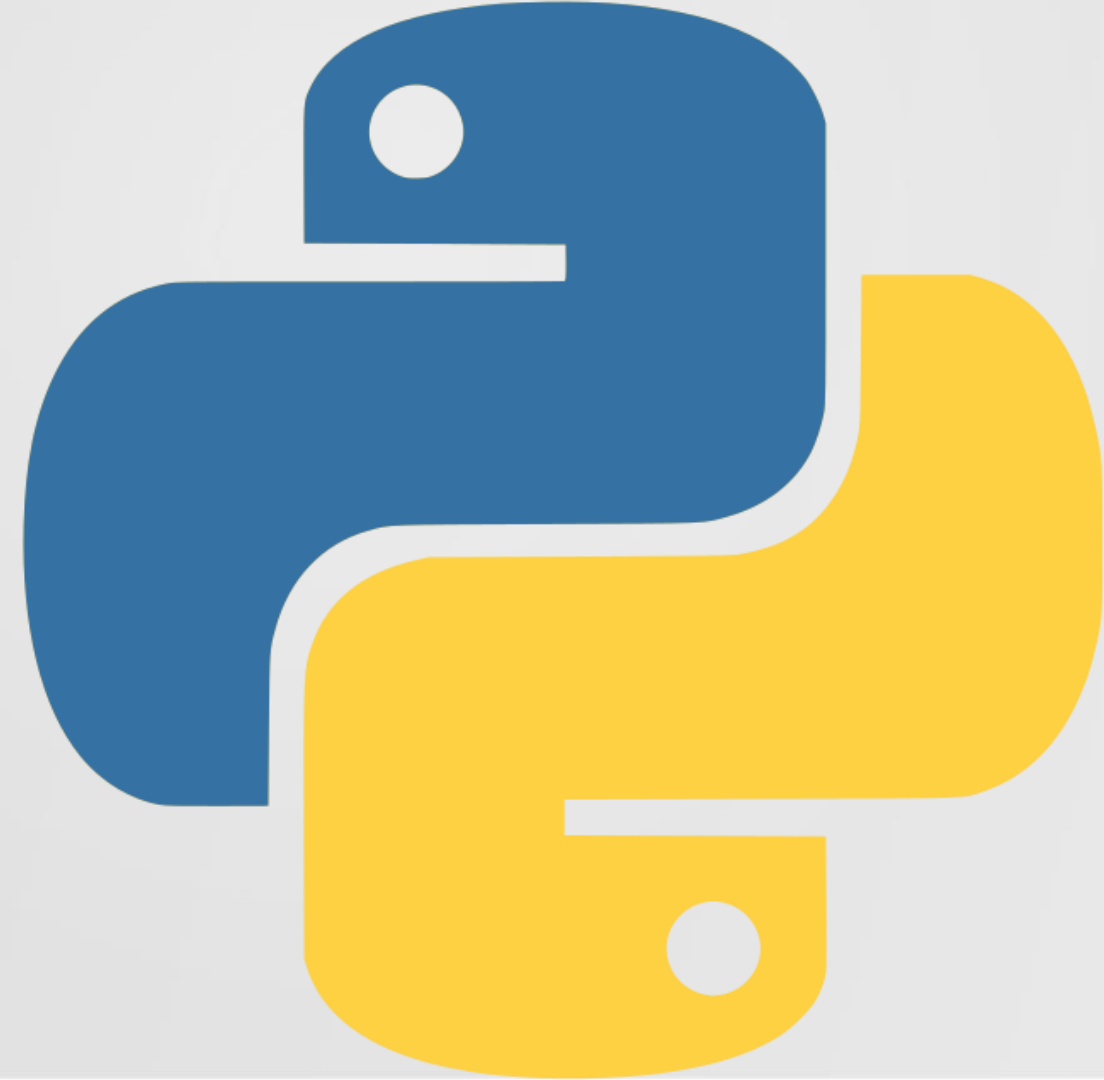


Data Augmentation by Random Crops

AlexNet Key Features



Dropout





Trabalho

- Desenvolver um classificador para solucionar algum problema nas áreas:

Manufatura, Medicina, Pesquisa, Agricultura, Pecuária, Astronomia, Telecomunicações, etc.

Metodologia: Apenas o notebook contendo o problema, a solução e o programa.

Tecnologia: Python

Banco de dados: Kaggle, Imaginet, Próprio, CAVE, GENOME, Open Image Dataset ...

Entrega: 08/11/2022

