

Processamento e Análise de Imagens

Redes Convolucionais e Aprendizado Profundo

Felipe Augusto Lima Reis



PUC Minas

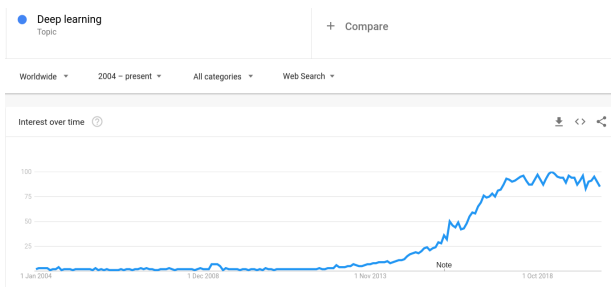
Agenda

- 1 Contexto
- 2 Redes Neurais
- 3 Fundamentos
- 4 Otimizadores
- 5 Métricas
- 6 R. Convolucionais

CONTEXTO

Contexto

- Nos últimos 5 anos, as redes neurais artificiais se tornaram extremamente populares;
- *Deep learning* tornou-se um termo bastante conhecido, especialmente a partir de 2014.



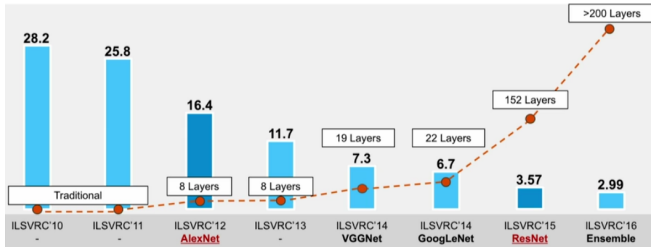
Fonte: [Google Trends, 2020]

Contexto

- Redes Neurais podem ser utilizadas para:
 - Tradução automática;
 - Assistentes virtuais;
 - Processamento de linguagem natural;
 - Visão computacional;
 - Suporte ao diagnóstico médico;
 - Detecção de fraudes;
 - Predição de desastres naturais;
 - Sistemas de recomendações;
 - Veículos autônomos;
 - ...

Contexto

- Um dos eventos responsável por essa popularização das redes neurais foi o desempenho da rede AlexNet na detecção e classificação de objetos no ILSVRC¹ 2012
 - A rede diminuiu o erro das top-5 classes previstas em mais de 10% [Goodfellow et al., 2016] [Krizhevsky et al., 2012].



Fonte: [Sadek Alaoui - SQLML, 2017]

¹ ImageNet Large Scale Visual Recognition Challenge (ILSVRC).

Tensorflow

- Tal desempenho levou a um aumento de pesquisas relacionadas às redes neurais;
- Consequentemente, ferramentas e frameworks começaram a ser desenvolvidos para facilitar o treinamento e o uso das redes
 - Tais ferramentas implementavam algoritmos complexos e forneciam mecanismos simplificados para processamento das redes neurais em GPUs;
 - Diversas ferramentas surgiram, como o Theano, Tensorflow, Caffe, PyTorch e o CNTK.

Tensorflow

- **Tensorflow** é uma plataforma aberta e open-source para criação e implantação de modelos de Aprendizado de Máquina [Abadi et al., 2015]
 - A ferramenta surgiu em 2015 a partir de uma evolução do projeto DistBelieve, desenvolvido pela Google Brain;
 - Foi lançado sob a licença Apache 2.0;



Fonte: [Abadi et al., 2015]

Tensorflow

- O nome Tensorflow está associado a dois conceitos importantes, que podem ser associados ao fluxo de informações em uma rede neural
 - **Tensor**: generalização da noção de escalares, vetores e matrizes;
 - **Dataflow**: paradigma de programação que modela um programa como um grafo direcionado do fluxo de dados entre as operações.

Tensorflow

- Atualmente o Tensorflow está na versão (estável) 2.6.0;
 - A versão 2.0 foi lançada em 2019 e contém muitas modificações em relação à primeira versão;
- Existe ainda uma versão denominada Tensorflow Lite, lançada em 2017 e destinada especificamente aos dispositivos móveis.

Keras

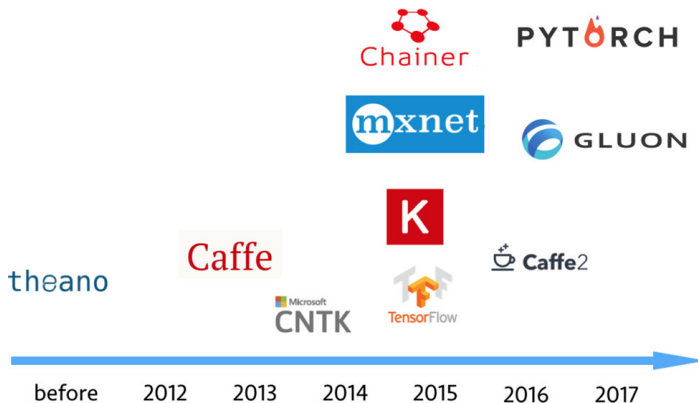
- **Keras** é uma biblioteca de código aberto que fornece uma interface Python para redes neurais artificiais
 - O Keras foi desenvolvido para facilitar o trabalho com ferramentas de aprendizado profundo;
 - Lançado em 2015, suportava back-ends TensorFlow, CNTK, Theano e PlaidML;
 - A partir da versão 2.4, lançada em 2020, o Keras restringiu seu suporte somente ao Tensorflow.



Fonte: [Chollet, 2015]

Outros Frameworks

- A figura abaixo contém uma linha do tempo dos frameworks de aprendizado profundo.



Fonte: [Elshawi et al., 2021]

Outros Frameworks

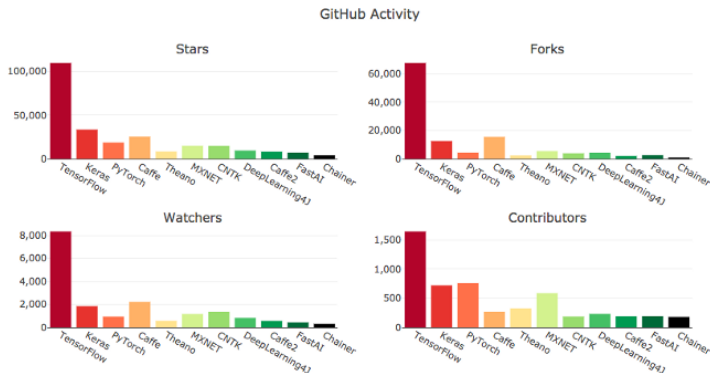
- A tabela abaixo contém uma comparação entre os frameworks.

	TensorFlow	Keras	PyTorch	MXNet	Theano	Chainer
Release date	2016	2015	2017	2015	2010	2015
Core language	C++	Python, R	C++, Python	C++	C++	Python
API	C++, Python	Python	Python	C++, Python, R Scala, Clojure Javascript, Web-UI	Python	Python
Data parallelism	✓	✓	✓	✓	✓	✓
Model parallelism	✓	✓	✓	✓	✓	✓
Programming paradigm	Imperative	Imperative	Imperative	Imperative declarative	Imperative	Imperative
Fault tolerance	Checkpoint-and -recovery	Checkpoint-and -resume	Checkpoint-and -resume	Checkpoint-and -resume	Checkpoint-and -resume	Checkpoint-and -resume
Multi GPU	✓	✓	✓	✓	✓	✓
Popularity (# stars on Github)	138k	45.8k	34.2k	18.1k	9k	5.2k

Fonte: [Elshawi et al., 2021]

Popularidade dos Frameworks

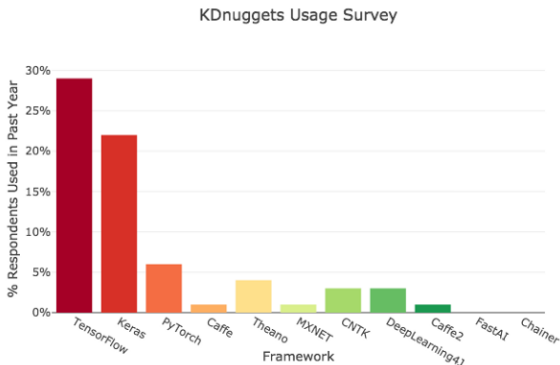
- A popularidade do frameworks para projetos no Github está disponível na imagem abaixo [Hale, 2018].



Fonte: [Hale, 2018]

Popularidade dos Frameworks

- A popularidade do frameworks segundo uma pesquisa do site de data science KDNuggets está disponível abaixo [Piatetsky, 2018] [Hale, 2018].



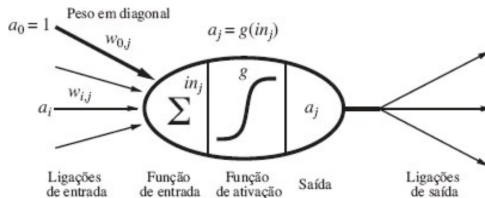
Fonte: [Piatetsky, 2018]

REDES NEURAIS

NEURÔNIOS ARTIFICIAIS

Neurônios artificiais

- Neurônios artificiais correspondem a um modelo matemático simples, desenvolvido McCulloch e Pitts, em 1943 [Russel and Norvig, 2013] [Coppin, 2004];
- Eles disparam uma combinação linear de suas entradas quando algum limiar é excedido [Russel and Norvig, 2013].



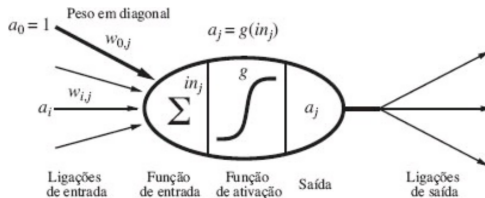
Fonte: [Russel and Norvig, 2013]

Neurônios Artificiais

- O modelo de neurônio proposto por McCulloch e Pitts (1943) pode ser dividido nas seguintes partes:
 - **Entrada:** sinais $\{x_1, x_2, \dots, x_n\}$ oriundos do ambiente externo;
 - **Pesos Sinápticos (w_i):** utilizado para definição de “importância” ou “relevância” das entradas para o neurônio;
 - **Bias (w_b):** entrada extra, utilizada para aumentar o grau de liberdade dos ajustes dos pesos;
 - **Corpo:** soma os produtos das entradas e pesos ($x_i \times w_i$) com o bias (w_b) e aplica a função de ativação;
 - **Função de Ativação:** controla o comportamento do sinal da saída $f(x)$, limitando o intervalo de valores de saída.

Funcionamento de um Neurônio Artificial

- Funcionamento de um neurônio:
 - 1 Cada neurônio possui um sinal de entrada x_i ;
 - 2 Esse neurônio i liga-se a outro neurônio j e é capaz de propagar um valor de ativação x_i ;
 - 3 Cada ligação tem um peso numérico $w_{i,j}$ associado;



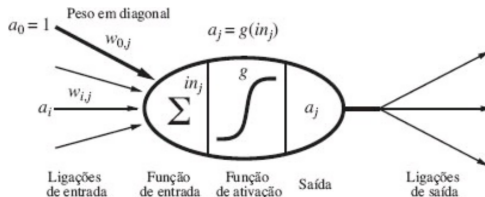
Fonte: [Russel and Norvig, 2013]

Baseado em [da Silva, 2014] e [Russel and Norvig, 2013]

Funcionamento de um Neurônio Artificial

- Funcionamento de um neurônio:
 - ④ O potencial de ativação de um neurônio é dado pela soma ponderada dos sinais de entrada, somado ao bias (w_b);

$$in_j = \sum_{i=1}^n (x_i \cdot w_i) + w_b$$



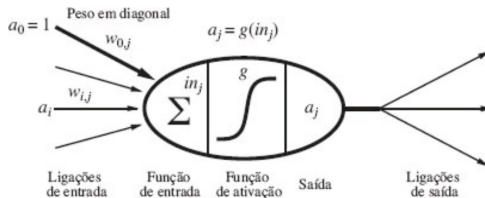
Fonte: [Russel and Norvig, 2013]

Baseado em [da Silva, 2014] e [Russel and Norvig, 2013]

Funcionamento de um Neurônio Artificial

- Funcionamento de um neurônio:
 - ⑤ Em seguida, é aplicada uma função de ativação g a essa soma, com objetivo de limitar o sinal de saída.

$$x_j = g \left(\sum_{i=1}^n (x_i \cdot w_i) + w_b \right)$$

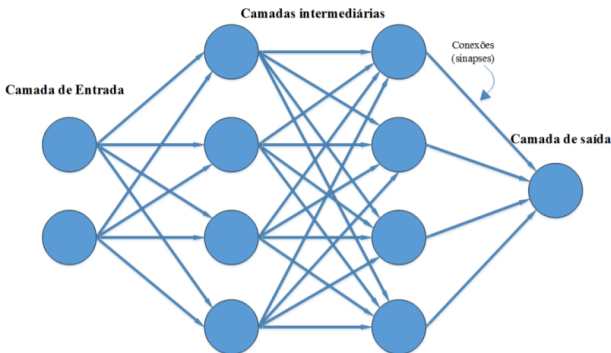


Fonte: [Russel and Norvig, 2013]

Baseado em [da Silva, 2014] e [Russel and Norvig, 2013]

Redes neurais artificiais

- Uma rede neural é apenas uma coleção de neurônios
 - Suas propriedades são determinadas pela topologia e pelos próprios neurônios [Russel and Norvig, 2013].



Fonte: [Antonelli and Neitzel, 2015]

REDES NEURAI

Redes Neurais

- Redes neurais são agrupamentos de neurônios artificiais
 - Cada um dos neurônios provê um mecanismo simples de processamento;
 - No contexto de redes, neurônios são chamados de nós.
- Redes podem ser arrançadas em camadas (*layers*)
 - A primeira camada, de entrada de dados, é chamada de **camada de entrada**;
 - As m camadas intermediárias são chamadas de **camadas intermediárias ou ocultas**;
 - A última camada, de saída da rede, é chamada de **camada de saída**;

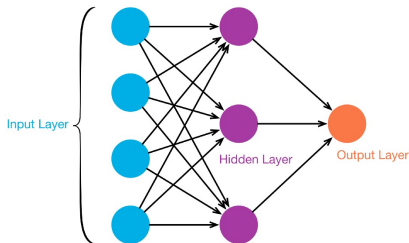
Funcionamento das Redes Neurais

- Funcionamento básico das redes neurais:
 - ① Neurônios na camada de entrada recebem entradas para serem classificadas;
 - ② As entradas causam o disparo de alguns neurônios (retornam valor acima de um determinado limiar, de acordo com a função de ativação);
 - ③ Neurônios das camadas iniciais transmitem o sinal para os neurônios das camadas seguintes;
 - ④ O processo se repete até o final da rede, quando a rede retorna um ou múltiplos valores de saída².

² Múltiplos valores de saída podem ser utilizados, por exemplo, no processamento de imagens, onde cada saída corresponde a um *pixel* de resultado.

Classificação das Redes Neurais

- Redes de Múltiplas Camadas
 - Redes mais utilizadas, com múltiplas camadas intermediárias;
 - Utilizadas para classificação de padrões, otimização, robótica, controle de processos, etc.
 - Possuem somente uma camada de entrada, m camadas intermediárias e uma camada de saída;



Fonte: [Nahua Kang - Towards Data Science, 2017]

Treinamento de Redes Neurais

- O treinamento das redes neurais são divididos em 2 fases
 - Fase forward
 - Ativações dos neurônios (valores \times pesos ativados) são propagadas da entrada para saída;
 - Fase backward
 - A erro/perda entre o valor produzido pela rede e o valor correto é propagado para trás, a fim de modificar pesos e valores de *bias*;
- As fase de propagação para frente e para trás são dependentes [Zhang et al., 2020].

FUNDAMENTOS

TIPOS DE APRENDIZADO

Classificação quanto a Supervisão

- **Aprendizado supervisionado**
 - O algoritmo aprende utilizando dados de treinamento pré-rotulados;
 - O algoritmo de treinamento supervisiona a diferença entre as saídas em relação às respostas esperadas;
- **Aprendizado não supervisionado**
 - O algoritmo aprende a partir de dados não rotulados;
 - Útil em situações onde dados precisam ser classificados ou *clusterizados* em classes ainda indefinidas;

Classificação quanto a Supervisão

- **Aprendizado semi supervisionado**
 - Intermediário entre o aprendizado supervisionado e o não supervisionado;
 - São fornecidas informações incompletas para treinamento, cabendo ao *software* a decisão sobre o conteúdo não rotulado.
- **Aprendizado por reforço**
 - Treinamento destinado à interação com ambientes dinâmicos, com objetivo de desempenhar uma ação;
 - O algoritmo recebe uma recompensa (ou premiação), caso atinja o objetivo, ou uma punição, caso falhe;
 - O objetivo é obter o maior valor de recompensa possível.

Classificação quanto a Supervisão

- **Aprendizado evolucionário**
 - A evolução biológica é definida como processo de aprendizado;
 - Utiliza-se o conceito de *fitness* (adaptação ao ambiente);
 - Organismos mais adaptados sobrevivem, enquanto os menos adaptados perecem [Marsland, 2014].

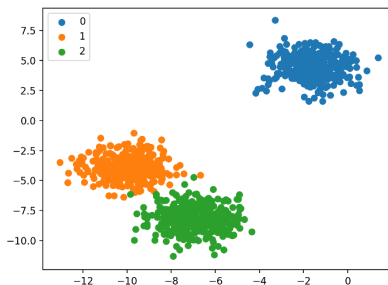
CLASSIFICAÇÃO X REGRESSÃO

Definição

- **Classificação:** rótulos t_i são definidos por uma quantidade limitada de valores discretos
 - Cada elemento pertence somente a uma classe;
 - O conjunto de classes cobre todo o espaço de saídas possível.
- **Regressão:** rótulos são definidos como valores contínuos
 - O objetivo da regressão é que a curva/reta se aproxime o máximo possível de todos os pontos;
 - A regressão pode ser um problema de aproximação ou interpolação de funções [Marsland, 2014].

Classificação

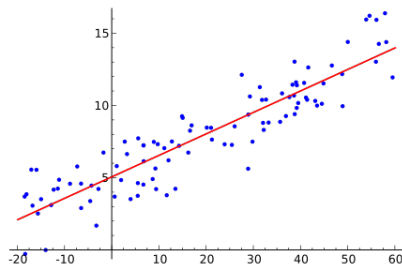
- O problema de **classificação** consiste em tomar um conjunto de elementos e decidir a qual das N classes eles pertencem
 - A classificação é feita com base em um treinamento prévio [Marsland, 2014].



Fonte: [Jason Brownlee - Machine Learning Mastery, 2020]

Regressão

- **Regressão** é um processo estatístico que busca ajustar uma função matemática para descrever uma curva [Marsland, 2014].



Fonte: [Wikipedia contributors, 2020d]

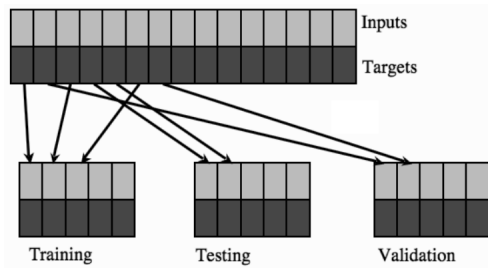
TREINO, VALIDAÇÃO E TESTE

Conjuntos de Treino, Validação e Teste

- Para treinamento de um algoritmo de aprendizado é apropriado utilizar 3 conjuntos:
 - **Conjunto de Treino**: destinado efetivamente ao treinamento do algoritmo;
 - **Conjunto de Validação**: destinado à avaliação de resultados durante a fase de treinamento;
 - **Conjunto de Testes**: destinado apenas à avaliação de resultados finais.

Proporção entre conjuntos

- Tipicamente utiliza-se a proporção 50:25:25, correspondente a treino, validação e testes;
- Para casos em que há poucos dados, recomenda-se a divisão na proporção 60:20:20 [Marsland, 2014].



Fonte: [Marsland, 2014]

Recomendação - Conjunto Testes

- **Importante:** O conjunto de testes deve ser utilizado somente após a conclusão de todo o treinamento
 - Não deve ser utilizado durante a fase de treinamento;
 - Deve, ainda, ser utilizado uma única vez.

OTIMIZADORES

Otimizadores

- Otimizadores correspondem aos métodos numéricos utilizados para minimização do erro esperado
 - O objetivo é reduzir a diferença entre a saída predita e a saída esperada;
 - Para isso, devemos minimizar a perda, definindo valores para os pesos da rede;
 - O método mais simples para isso é o *Gradient Descent (GD)*³, porém esse método tem um custo muito alto
 - Dessa forma o *Gradient Descent* é muito pouco usado.

³Também chamado de *Batch Gradient Descent* (BDG) ou *Vanilla Gradient Descent* (VGD).

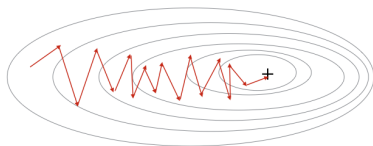
Stochastic Gradient Descent (SGD)

- O método do **gradiente estocástico (SGD)** corresponde a uma alteração no método *Gradient Descent*
 - O algoritmo não analisa todos os pontos a serem otimizados, apenas amostras dos dados;
 - O algoritmo varre as amostras e, calcula os valores de gradiente para cada delas;
 - O algoritmo toma a direção de maior declividade dentro das amostras;
 - Essa modificação reduz consideravelmente o custo computacional de cálculo dos gradientes [Ruder, 2016] [Hansen, 2019].

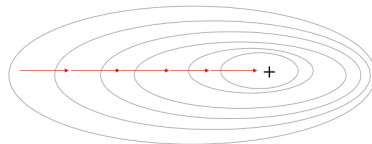
Stochastic Gradient Descent (SGD)

- Como o SGD analisa apenas amostras aleatórias dos dados, seu comportamento é mais instável que o algoritmo GD;
- No entanto, a redução de custo computacional compensa os efeitos colaterais [Ruder, 2016] [Yakout, 2021].

Stochastic Gradient Descent



Gradient Descent



Fonte: [Yakout, 2021]

Momentum

- **Momentum** é um método para aceleração do otimizadores quando as superfícies da curva convergem mais abruptamente em uma direção do que em outra;
- Seu funcionamento é análogo ao momento em física e auxilia o SGD a manter uma direção fixa [Ruder, 2016].



(a) SGD without momentum



(b) SGD with momentum

Fonte: [Ruder, 2016]

Link: [animação de um SGD com momentum](#) [Hansen, 2019].

Outros Otimizadores

- Além dos métodos previamente citados (mais simples), destacam-se também na literatura:
 - **NAG**: acrônimo de Nesterov Accelerated Gradient, tenta dar ao Momentum um tipo de presciência, de forma a dar saltos grandes no início da execução e evitando saltos à medida em que o algoritmo aproxima-se do objetivo [Ruder, 2016];
 - **AdaGrad**: adapta a taxa de aprendizado aos parâmetros, realizando atualizações maiores para parâmetros pouco frequentes e atualizações menores para parâmetros frequentes [Ruder, 2016].

Link 1: [animação de otimizadores para redes neurais](#) [Duong, 2015].

Link 2: [animação de otimizadores para redes neurais](#) [Ruder, 2016] [Radford, 2014].

Link 3: [animação de otimizadores para redes neurais](#) [Radford, 2014].

Outros Otimizadores

- Outros otimizadores:
 - **Adadelta**: melhoria do AdaGrad que busca reduzir o decrescimento agressivo e monotônico da taxa de aprendizado, ao reduzir o acúmulo de gradientes passados a uma janela de tempo fixa [Ruder, 2016];
 - **RSMprop**: método não publicado proposto por Geoffrey Hinton e semelhante ao Adadelta (desenvolvidos separadamente);
 - **Adam**: acrônimo de Adaptive Moment Estimation, consiste em um método adaptativo para ajuste das taxas de aprendizado, podendo ser considerado uma evolução dos métodos Adadelta/RMSprop e Momentum [Ruder, 2016].

Link 4: [animação de otimizadores para redes neurais](#) [Hansen, 2019] [Rahman, 2017].

Link 5: [animação de otimizadores para redes neurais](#) [Hansen, 2019] [Rahman, 2017].

MÉTRICAS DE DESEMPENHO

Métricas de Desempenho

- Além de avaliar a quantidade de dados necessários para testes do algoritmo, é necessário utilizar métricas que indiquem se o resultado é bom ou não [Marsland, 2014];
- Esta seção tem como objetivo descrever as principais métricas de avaliação de desempenho de algoritmos para conjuntos de dados balanceados.

Matriz de Confusão

- A decisão tomada por um classificador pode ser representada por uma estrutura conhecida como matriz de confusão ou tabela de contingência.

		Valor Predito	
		Sim	Não
Real	Sim	Verdadeiro Positivo (TP)	Falso Negativo (FN)
	Não	Falso Positivo (FP)	Verdadeiro Negativo (TN)

Fonte: [Nogare, 2020]

Outputs			
	C_1	C_2	C_3
C_1	5	1	0
C_2	1	4	1
C_3	2	0	4

Fonte: [Marsland, 2014]

Matriz de Confusão

- Segundo [Davis and Goadrich, 2006], matrizes de confusão podem identificar 4 tipos de informação:
 - Verdadeiros Positivos (TP)
 - Valores corretamente classificados como positivos;
 - Falsos Negativos (FN)
 - Valores incorretamente classificados como negativos;
 - Falsos Positivos (FP)
 - Valores incorretamente classificados como positivos;
 - Verdadeiros Negativos (TN)
 - Valores corretamente classificados como negativos.

Métricas de Classificação Binária

- A partir da classificação dos resultados, podem ser geradas as seguintes métricas [Fawcett, 2006]:

- Taxa de Falsos Positivos (FPR)

- Quantidade de valores incorretamente classificados como positivos, FP , sobre o total de valores negativos, N ;

$$FPR = \frac{FP}{N}$$

- Taxa de Falsos Negativos (FNR)

- Quantidade de valores incorretamente classificados como negativos, FN , sobre o total de valores positivos, P .

$$FNR = \frac{FN}{P}$$

Métricas de Classificação Binária

- A partir das classificação dos resultados, podem ser geradas as seguinte métricas [Fawcett, 2006]:

- Taxa de Verdadeiros Positivos (TPR)

- Quantidade de valores corretamente classificados como positivos, TP , sobre o total de valores positivos, P ;

$$TPR = \frac{TP}{P}$$

- Taxa de Verdadeiros Negativos (TNR)

- Quantidade de valores corretamente classificados como negativos, TN , sobre o total de valores negativos, N .

$$TNR = \frac{TN}{N}$$

Acurácia

- Podemos considerar as seguintes métricas de acurácia:
 - Acurácia
 - Quantidade de verdadeiros Positivos, TP , somado à quantidade verdadeiros negativos, TN , dividido pelo número de amostras [Fawcett, 2006];

$$acurácia = \frac{TP + TN}{P + N}$$

- Taxa de Erro⁴
 - Número de valores positivos e negativos classificados incorretamente sobre soma total de valores [Guo et al., 2012].

$$ER = \frac{FP + FN}{P + N}$$

⁴ Error Rate (ER).

Acurácia

- **Importante:** Para uso da métrica acurácia, devemos assumir, implicitamente, a existência da mesma quantidade de exemplos positivos e negativos
 - Esse conjunto de dados é conhecido como balanceado [Marsland, 2014].
- Ex.: Considere o algoritmo abaixo, para detecção de uma doença X, onde somente 1% dos indivíduos analisados estão efetivamente doentes
 - Esse algoritmo possui acurácia de 99%, no entanto, é incapaz de identificar um único paciente doente.

```
bool pacienteEstaDoente(){  
    |   return False;  
}
```

Fonte: Próprio autor

Precisão e Revocação

- Precisão⁵ e Revocação⁶ são métricas de classificação binária frequentemente usadas em Aprendizado de Máquinas;
 - Precisão (*precision*)
 - Mede a proporção de eventos de modelo que são verdadeiros;
 - Revocação (*recall*)
 - Mede a proporção de eventos que ocorrem no domínio que são capturados pelos modelos [Fawcett, 2006].

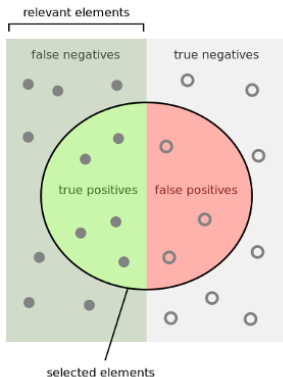
$$precisão = \frac{TP}{TP + FP}$$

$$revocação = \frac{TP}{TP + FN}$$

⁵ Denominado também por Valor Predito Positivo.

⁶ Denominado também por Sensibilidade.

Precisão e Revocação



How many selected items are relevant?

$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

How many relevant items are selected?

$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

Fonte: [Wikipedia contributors, 2020b]

F-measure

- Para uma matriz de confusão simples, a *F-measure*⁷, é uma média harmônica entre a precisão e a revocação;
 - Busca balancear a importância de precisão e revocação;
 - Quando ambas possuem valores próximos, a métrica equivale a aproximadamente a média das duas medidas;

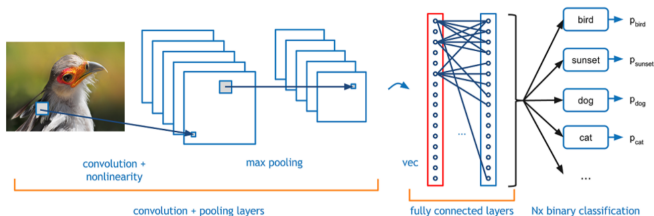
$$F\text{-measure} = \frac{2 \cdot \text{precisão} \cdot \text{revocação}}{\text{precisão} + \text{revocação}}$$

⁷ Também chamada de *F-score* ou *F₁ measure* [Marsland, 2014].

REDES CONVOLUCIONAIS

Redes Convolucionais

- Redes convolucionais (CNNs) são compostas por camadas com convoluções, operações de *pooling* e funções de ativação ReLU (nas camadas intermediárias);
- Dependendo do objetivo, podem ser utilizadas camadas totalmente conectadas no final da rede [Li et al., 2021].



Fonte: [Florindo, 2018]

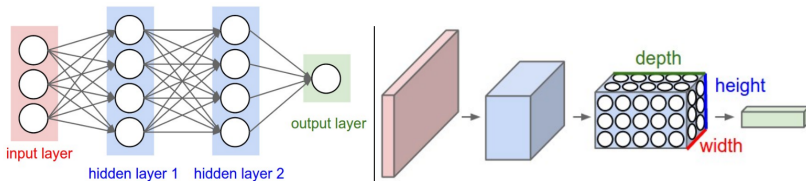
Redes Convolucionais

- Em geral, as redes convolucionais trabalham com imagens
 - Com isso, a arquitetura pode conter algumas restrições não existentes em redes tradicionais;
- Ao contrário de redes convencionais, redes convolucionais possuem neurônios organizados em 3 dimensões:
 - As dimensões são: largura, altura e profundidade⁸;
 - Essa característica é especialmente útil no processamento de imagens;
 - Imagens possuem altura, largura e, em geral, 3 canais de cores [Li et al., 2021].

⁸Tradução do inglês: *width*, *height* e *depth*.

Redes Convolucionais

- Ao contrário das redes totalmente conectadas:
 - Os neurônios em uma camada são conectados apenas a uma pequena região da camada anterior;
 - A rede transforma a entrada 3D em um “volume” 3D de ativações de neurônios;
 - No final de algumas redes existem camadas de achatamento (*flatten*) e/ou camadas totalmente conectadas, para produzir previsões [Li et al., 2021].



Fonte: [Li et al., 2021]

CONVOLUÇÃO

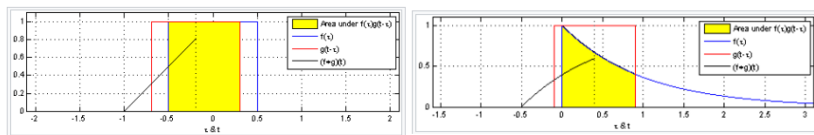
Operação de Convolução

- A convolução é a operação mais importante em uma rede convolucional;
- Consiste em um operador linear para duas funções f e g ;
- Resulta em um terceiro valor s , que mede a soma do produto ao longo da região subentendida pela superposição de g , quando deslocada sobre uma função f [Weisstein, 2018].

$$(f * g)(t) = s(t) = \int_{-\infty}^{\infty} f(u) \cdot g(t - u) du$$

Operação de Convolução

- A convolução pode ser entendida como a forma com que um sistema opera sobre um sinal de entrada [Smith, 1997].



Fonte: Adaptado de [Wikipedia contributors, 2020a]

Link: [animação de uma convolução 1](#) [Wikipedia contributors, 2020a]

Link: [animação de uma convolução 2](#) [Wikipedia contributors, 2020a]

Operação de Convolução

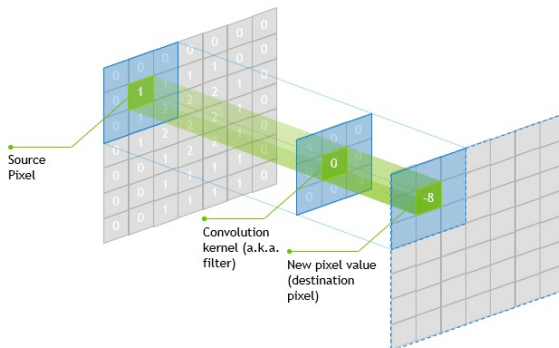
- Na terminologia de redes convolucionais, a função f corresponde a **entrada**, g representa o **kernel** e a saída s corresponde ao mapa de saída (**feature map**);
- Segundo [Goodfellow et al., 2016], as convoluções são utilizadas para aprimorar resultados nos seguintes campos:
 - Interações esparsas: possibilitam o armazenamento de características importantes com baixo número de parâmetros;
 - Compartilhamento de parâmetros: possibilita armazenamento apenas de alguns valores, que geram os demais pesos da rede;
 - Representações equivariantes: alterações realizadas na entrada produzem efeitos correspondentes nas saídas.

Camada Convolutiva

- Principal camada das redes convolucionais, responsável pela maior parte dos cálculos [Li et al., 2021];
- Convoluções utilizam múltiplos filtros (*kernels*), que podem ter diferentes tamanhos (ex.: $5 \times 5 \times 3$, $32 \times 32 \times 3$, etc);
- Durante o *forward pass*, é feita a convolução de cada filtro na largura e altura do volume de entrada;
- Em seguida, são calculados os produtos escalares entre as entradas do filtro e a entrada em qualquer posição [Li et al., 2021].

Camada Convolucional

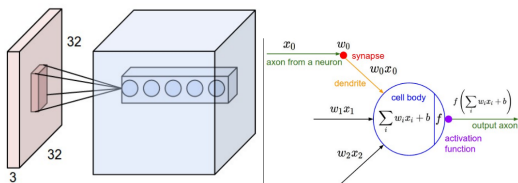
- Os filtros são “deslizados” sobre largura e altura do volume;



Fonte: [Martin, 2018]

Camada Convolutiva

- Produz-se um mapa de ativação bidimensional que fornece as respostas desse filtro em cada posição espacial;
- A rede aprende quais filtros ativar a partir das entradas;
- Os filtros ativados são empilhados na camada de profundidade para produção do volume de saída [Li et al., 2021].

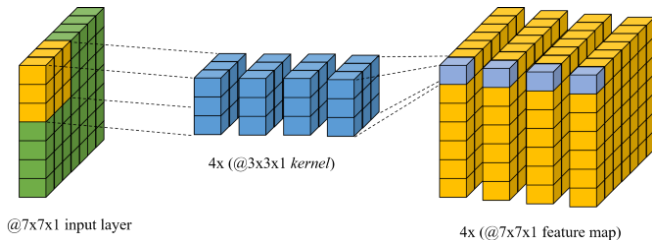


Fonte: [Li et al., 2021]

Link: [Animação com o funcionamento de uma convolução](#) [Li et al., 2021].

Camada Convolutiva

- A aplicação de múltiplos filtros (*kernels*) convolucionais dá origem a uma profundidade diferente na camada seguinte [Li et al., 2021].



Fonte: [Yunus, 2020]

Link: [Animação com o funcionamento de uma convolução](#) [Li et al., 2021].

Profundidade, *Stride* e *Zero-padding*

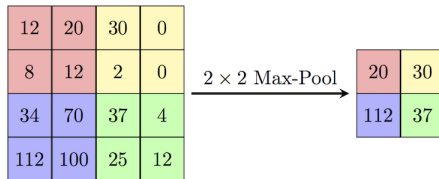
- O tamanho do volume, é definido por 3 parâmetros:
 - *Profundidade (depth)*: hiperparâmetro correspondente ao número de *kernels* que serão utilizados para criação de mapas de características
 - Cada filtro pode aprender características diferentes, como cores, texturas e variações de tonalidade;
 - *Stride*: taxa de “deslizamento” dos *kernels*, em *pixels*;
 - *Zero-padding*⁹: corresponde ao à quantidade de zeros adicionados às bordas da imagem para melhor ajuste de tamanho (*pad*) [Li et al., 2021] [IBM, 2020].

⁹Frequentemente denominado apenas como *padding*, pode ser traduzido como preenchimento de zeros.

POOLING

Pooling

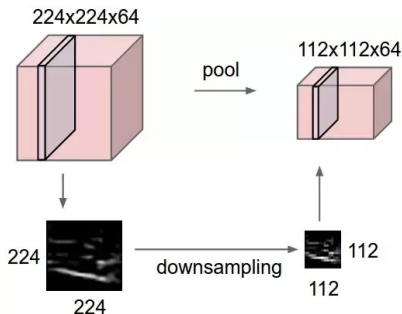
- A função de *pooling* é utilizada para prover informação estatística a respeito das saídas próximas;
- Tem como objetivo maximizar as vantagens da saída e tornar-se invariante a pequenos ruídos;
- Podem também ser utilizadas para redução da quantidade de neurônios entre camadas da rede, agrupando resultados na camada seguinte [Goodfellow et al., 2016].



Fonte: [Computer Science Wiki, 2020]

Pooling

- A operação de *pooling* é executada em cada camada do volume, promovendo redução de dimensionalidade;



Fonte: [Computer Science Wiki, 2020]

Pooling

- Os tipos de pooling mais comuns são:
 - **Max-pooling:** utiliza operação de máximo no filtro de *pooling*
 - Tipo mais utilizado de *pooling*, presente em redes como a VGGNet e AlexNet;
 - Capaz de selecionar os *pixels* mais ativados localmente, descartando *pixels* com valores baixos;
 - Capaz de representar uma região, por meio das informações mais importantes;
 - **Avg-pooling:** utiliza a operação de média no filtro de *pooling*
 - Capaz de representar uma região, por meio de informações médias;
 - Menos comum que *max-pooling*, mas presente na rede ResNet.

FUNÇÃO DE ATIVAÇÃO RELU

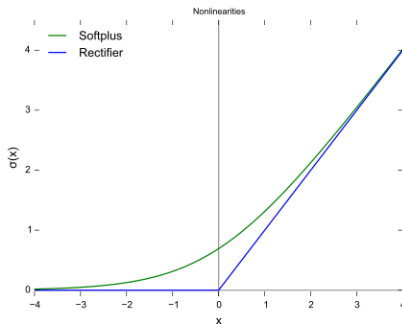
ReLU

- ReLU: *Rectified linear unit* (Unidade linear retificada);
- Função de ativação empregada em redes neurais convolucionais, principalmente em camadas intermediárias;
- Vantagens:
 - Melhor propagação de gradientes: diminuição da quantidade de gradientes que tendem a zero devido ao acúmulo de camadas e pequenos valores, quando comparado a função sigmoide;
 - Invariante a escala: $\max(0, ax) = a \max(0, x)$ para $a \geq 0$;
 - Ativação esparsa: em redes inicializadas aleatoriamente, somente 50% das camadas intermediárias são ativadas;

ReLU

- A função ReLU é definida formalmente como:

$$f(x) = \max(0, x)$$



Fonte: [Wikipedia contributors, 2020c]

Referências I



Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015).
TensorFlow: Large-scale machine learning on heterogeneous systems.
Software available from tensorflow.org.



Antonelli, G. and Neitzel, I. (2015).

Aplicação de redes neurais artificiais na indústria de fios de algodão: Determinação do Índice de fibras imaturas.

Revista Gestão Industrial, 11.



Chollet, F. (2015).

keras.

<https://github.com/fchollet/keras>.

[Online]; acessado em 17 de outubro de 2021. Disponível em <https://keras.io/>.



Computer Science Wiki (2020).

Max-pooling / pooling.

[Online]; acessado em 08 de Setembro de 2020. Disponível em:

https://computersciencewiki.org/index.php/Max-pooling/_Pooling.



Coppin, B. (2004).

Artificial intelligence illuminated.

Jones and Bartlett illuminated series. Jones and Bartlett Publishers, 1 edition.

Referências II



da Silva, D. M. (2014).
Inteligência Artificial - Slides de Aula.
IFMG - Instituto Federal de Minas Gerais, Campus Formiga.



Davis, J. and Goadrich, M. (2006).
The relationship between precision-recall and roc curves.
In Proceedings of the 23rd International Conference on Machine Learning, ICML '06, pages 233–240, New York, NY, USA. ACM.



Duong, H. (2015).
Gradient descent and variants - convergence rate summary.
[Online]; acessado em 03 de Setembro de 2020. Disponível em:
<http://hduongtrong.github.io/2015/11/23/coordinate-descent/>.



Elshawi, R., Wahab, A., Barnawi, A., and Sakr, S. (2021).
Dlbench: a comprehensive experimental evaluation of deep learning frameworks.
Cluster Computing, 24(3):2017–2038.



Fawcett, T. (2006).
An introduction to roc analysis.
Pattern Recognition Letters, 27(8):861 – 874.
ROC Analysis in Pattern Recognition.

Referências III



Florindo, J. a. B. (2018).

Redes neurais convolucionais.

[Online]; acessado em 08 de Setembro de 2020. Disponível em:

<https://www.ime.unicamp.br/~jbflorindo/Teaching/2018/MT530/T10.pdf>.



Goodfellow, I., Bengio, Y., and Courville, A. (2016).

Deep Learning.

MIT Press.

[Online]; acessado em 17 de outubro de 2021. Disponível em: <http://www.deeplearningbook.org>.



Google Trends (2020).

Deep learning.

[Online]; acessado em 25 de Agosto de 2020. Disponível em:

<https://trends.google.com.br/trends/explore?date=all&q=deep%20learning>.



Guo, C., Mita, S., and McAllester, D. (2012).

Robust road detection and tracking in challenging scenarios based on markov random fields with unsupervised learning.

IEEE Transactions on Intelligent Transportation Systems, 13(3):1338–1354.



Hale, J. (2018).

Deep learning framework power scores 2018.

[Online]; acessado em 17 de outubro de 2021. Disponível em

<https://towardsdatascience.com/deep-learning-framework-power-scores-2018-23607ddf297a>.

Referências IV



Hansen, C. (2019).

Optimizers explained - adam, momentum and stochastic gradient descent.
[Online]; acessado em 27 de Janeiro de 2021. Disponível em:
<https://mlfromscratch.com/optimizers-explained/#/>.



IBM (2020).

Convolutional neural networks.
[Online]; acessado em 26 de Janeiro de 2021. Disponível em:
<https://www.ibm.com/cloud/learn/convolutional-neural-networks>.



Jason Brownlee - Machine Learning Mastery (2020).

4 types of classification tasks in machine learning.
[Online]; acessado em 17 de outubro de 2021. Disponível em
<https://machinelearningmastery.com/types-of-classification-in-machine-learning>.



Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012).

Imagenet classification with deep convolutional neural networks.
In Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1, NIPS'12, pages 1097–1105, Red Hook, NY, USA. Curran Associates Inc.



Li, F.-F., Krishna, R., and Xu, D. (2021).

Convolutional neural networks (cnns / convnets).
[Online]; acessado em 26 de Janeiro de 2021. Disponível em:
<https://cs231n.github.io/convolutional-networks/>.

Referências V



Marsland, S. (2014).

Machine Learning: An Algorithm Perspective.

CRC Press, 2 edition.

Disponível em: <https://homepages.ecs.vuw.ac.nz/~marslast/MLbook.html>.



Martin, S. (2018).

What's the difference between a cnn and an rnn?

[Online]; acessado em 26 de Janeiro de 2021. Disponível em:

<https://blogs.nvidia.com/blog/2018/09/05/whats-the-difference-between-a-cnn-and-an-rnn/>.



Nahua Kang - Towards Data Science (2017).

Multi-layer neural networks with sigmoid functionâ deep learning for rookies.

[Online]; acessado em 01 de Setembro de 2020. Disponível em: [https://towardsdatascience.com/](https://towardsdatascience.com/multi-layer-neural-networks-with-sigmoid-function-deep-learning-for-rookies-2-bf464f09eb7f)

[multi-layer-neural-networks-with-sigmoid-function-deep-learning-for-rookies-2-bf464f09eb7f](https://towardsdatascience.com/multi-layer-neural-networks-with-sigmoid-function-deep-learning-for-rookies-2-bf464f09eb7f).



Nogare, D. (2020).

Performance de machine learning - matriz de confusão.

Disponível em <http://diegonogare.net/2020/04/performance-de-machine-learning-matriz-de-confusao/>.



Piatetsky, G. (2018).

Python eats away at r: Top software for analytics, data science, machine learning in 2018: Trends and analysis.

[Online]; acessado em 17 de outubro de 2021. Disponível em <https://www.kdnuggets.com/2018/05/poll-tools-analytics-data-science-machine-learning-results.html/2>.

Referências VI



Radford, A. (2014).

Visualizing optimization algos.

[Online]; acessado em 27 de Janeiro de 2021. Disponível em: <https://imgur.com/a/Hqolp>.



Rahman, R. (2017).

Visualising stochastic optimisers.

[Online]; acessado em 27 de Janeiro de 2021. Disponível em:

<https://rnr Rahman.com/blog/visualising-stochastic-optimisers/>.



Ruder, S. (2016).

An overview of gradient descent optimization algorithms.

CoRR, abs/1609.04747.



Russel, S. and Norvig, P. (2013).

Inteligência artificial.

Campus - Elsevier, 3 edition.



Sadek Alaoui - SQLML (2017).

Convolutional neural network.

[Online]; acessado em 25 de Agosto de 2020. Disponível em:

<http://sqlml.azurewebsites.net/2017/09/12/convolutional-neural-network/>.



Smith, S. W. (1997).

The Scientist and Engineer's Guide to Digital Signal Processing.

California Technical Publishing, San Diego, CA, USA.

<http://www.dspguide.com>.

Referências VII



Weisstein, E. W. (2018).
Convolution.
<http://mathworld.wolfram.com/Convolution.html>.



Wikipedia contributors (2020a).
Convolution.
[Online]; acessado em 08 de Setembro de 2020. Disponível em:
<https://en.wikipedia.org/wiki/Convolution>.



Wikipedia contributors (2020b).
Precision and recall.
[Online]; acessado em 17 de outubro de 2021. Disponível em
https://en.wikipedia.org/wiki/Precision_and_recall.



Wikipedia contributors (2020c).
Rectifier (neural networks).
[Online]; acessado em 08 de Setembro de 2020. Disponível em:
[https://en.wikipedia.org/wiki/Rectifier_\(neural_networks\)](https://en.wikipedia.org/wiki/Rectifier_(neural_networks)).



Wikipedia contributors (2020d).
Regression analysis.
[Online]; acessado em 17 de outubro de 2021. Disponível em
https://en.wikipedia.org/wiki/Regression_analysis.

Referências VIII



Yakout, A. (2021).

Coursera deep learning specialization by andrew ng.

[Online]; acessado em 27 de Janeiro de 2021. Disponível em:

<https://yakout.io/deeplearning/coursera-deep-learning-course-2-week-2/>.



Yunus, M. (2020).

11 artificial neural network (ann) â part 6 konsep dasar convolutional neural network (cnn).

[Online]; acessado em 26 de Janeiro de 2021. Disponível em: [https://yunusmuhammad007.medium.com/](https://yunusmuhammad007.medium.com/11-artificial-neural-network-ann-part-6-konsep-dasar-convolutional-neural-network-cnn-3cc10fd9cf68)

11-artificial-neural-network-ann-part-6-konsep-dasar-convolutional-neural-network-cnn-3cc10fd9cf68



Zhang, A., Lipton, Z. C., Li, M., and Smola, A. J. (2020).

Dive into Deep Learning.

<https://d2l.ai>.