Introdução à Inteligência Artificial

Módulo 04: Aprendizagem Profunda (Deep Learning - DL)

Professor: Dr. João Paulo R. R. Leite

UNIFEI Softex MCTI FUTURO FUTURO DO TRABALHO, TRABALHO DO FUTURO GOVERNO FEDERAL BRASIL UNIÃO E RECONSTRUÇÃO

Sumário

Tipos de Redes Profundas

UNIFEI Softex MCTI FUTURO

FUTURO DO TRABALHO, TRABALHO DO FUTURO

GOVERNO FEDERAL BRASIL UNIÃO E RECONSTRUÇÃO

Objetivo principal

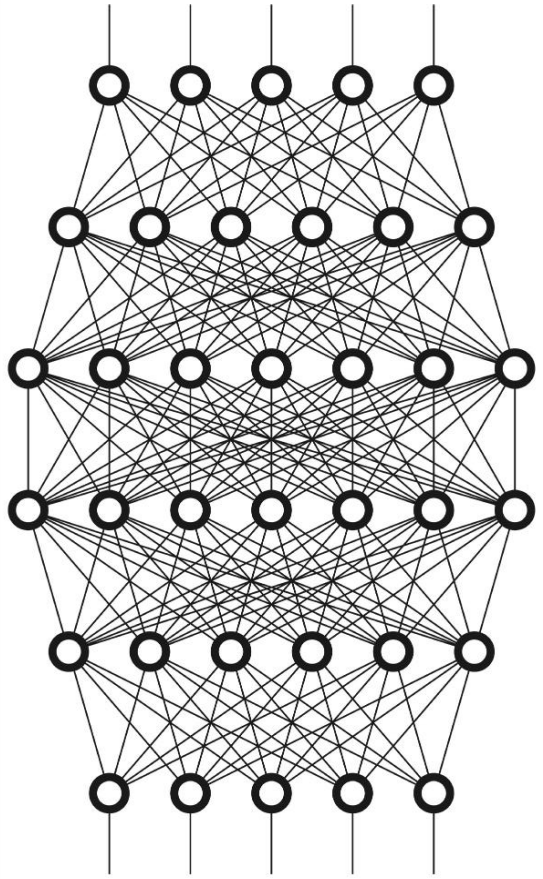
- O **objetivo** desta aula é introduzir superficialmente os tipos de redes neurais artificiais profundas, mostrando suas características, cenários em que podem ser utilizadas e conceitos envolvidos.

Introdução

Aprendizado Profundo, ou ***deep learning***, é baseado na utilização de redes neurais artificiais profundas, ou seja, aquelas **redes que possuem várias camadas**.

Desde o início do desenvolvimento das redes multicamadas, e do algoritmo de *backpropagation*, **muitos modelos diferentes foram propostos**, cada um projetado para a solução de um tipo de problema. Alguns deles se sobressaíram, demonstrando **desempenho superior** em campos específicos da ciência e tornando-se referências no mundo de IA.

Iremos conhecer alguns deles hoje.



Introdução

Vamos tomar como exemplo um modelo neural que toma uma imagem como entrada e a transforma sucessivamente com o objetivo de **identificar o valor do dígito representado** nela.

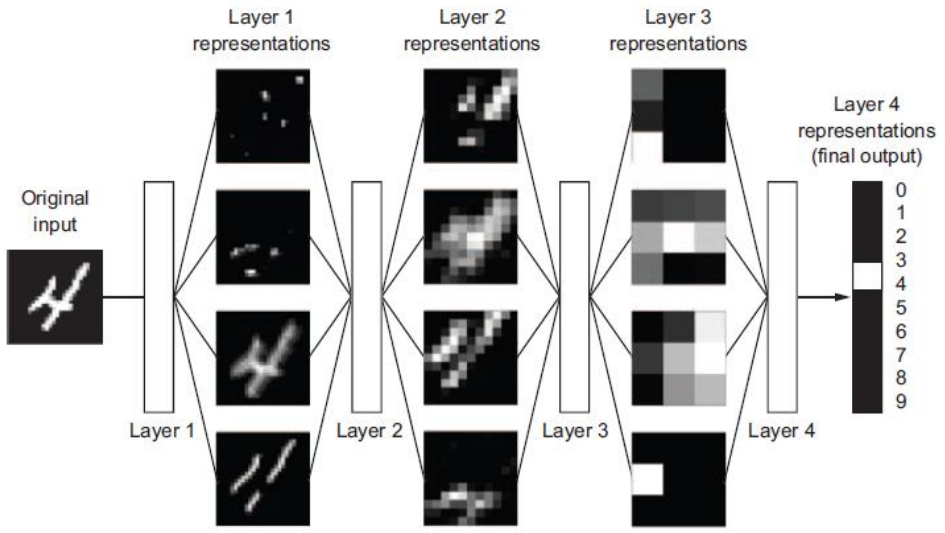


Imagem do livro “Deep learning with Python” (CHOLLET, François).

Veja que a rede “transforma” a imagem em representações que são cada vez mais distantes da imagem original, e **cada vez mais informativas sobre o resultado final**. Ela é capaz de obter a cada passo detalhes que caracterizam as peculiaridades do dígito 4.

A rede neural profunda funciona como uma operação em que a informação é “**destilada**” através de vários estágios, passando por filtros, aprendendo a representação dos dados e saindo “purificada” do outro lado.

Introdução

A essa altura, já sabemos que *deep learning* é geralmente sobre **mapear entradas** (uma imagem, por exemplo) **para saídas correspondentes** (como o rótulo “gato”), e isso é feito a partir da observação de um número grande de exemplos de entrada e respectivas saída.

O papel de cada camada nesse processo é armazenado em **parâmetros** conhecidos como “pesos” (*weights)*, que são valores numéricos associados às conexões. O processo de aprendizado da rede, portanto, consiste na **busca pelo conjunto de valores de pesos** em todas as camadas da rede que a tornam capaz de mapear corretamente as entradas para as saídas esperadas.

O ajuste desse dos pesos visando um conjunto ótimo é realizado iterativamente através do cálculo da função de perda (*loss function)* e da retropropagação de erro (**backpropagation**). Em uma rede profunda, pode haver milhões de valores de parâmetros. Seu ajuste demanda **conjuntos de dados grandes** e de boa qualidade e alta capacidade de **hardware**.

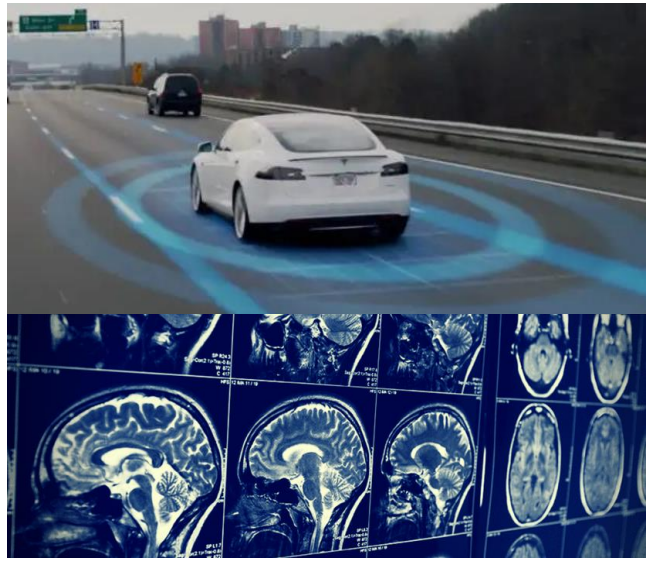
Redes Neurais Convolucionais (CNN)

As Redes Neurais Convolucionais (CNN, de *Convolutional Neural Networks*) são redes *feedforward* voltadas especialmente para o processamento de imagens e visão computacional. São utilizadas na classificação automática de imagens (médicas, industriais, etc.), carros autônomos, reconhecimento facial, e muitas outras coisas. O nome "convolucional" é derivado da operação matemática chamada **convolução**.

Diferentes das redes que vimos até agora, compostas apenas por camadas densamente conectadas, a CNN possui camadas de três tipos: **convolucionais**, de ***pooling***, e totalmente conectadas ***(dense)***.

Devido às suas particularidades, a CNN é capaz de extrair características a partir dos dados puros *(raw),* o que significa que é possível inserir uma imagem diretamente como entrada da rede, ao invés de dados pré-processados a partir da imagem. Economiza o passo de *feature engineering*.

Também podem ser utilizadas em outros cenários, mas é menos comum.



Redes Neurais Convolucionais (CNN)

David H. Hubel e Torsten Wiesel realizaram uma série de experimentos nos anos de 1958 e 1959, através dos quais foi possível apresentar uma boa ideia sobre o **funcionamento do córtex visual de gatos** e, depois, macacos (Receberam o prêmio Nobel em 1981 pelo seu trabalho).

Eles mostraram que os neurônios do córtex visual reagem a estímulos visuais localizados em uma **região limitada do campo visual**, e não à imagem como um todo (campo receptivo local). Os campos receptivos locais de diferentes neurônios podem se sobrepor, e **juntos eles compõe o campo visual completo**.

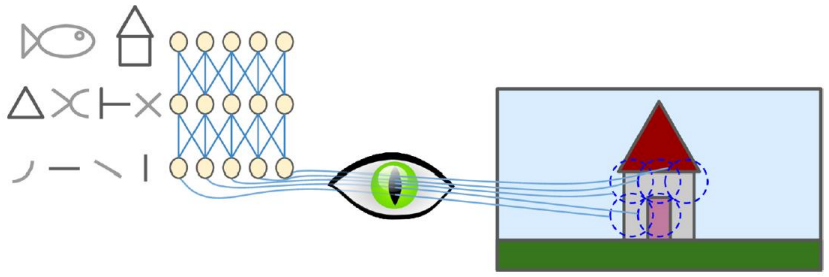


Imagem do livro "Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow." (GÉRON, Aurélien).

Mostraram ainda que alguns neurônios possuem um campo receptivo maior, e reagem a padrões mais complexos. Isso levou à ideia de que alguns **neurônios processam a combinação das saídas de outros neurônios com funções mais simples**. Essa poderosa arquitetura é capaz de detectar todo tipo de padrões complexos em que qualquer parte do campo visual.

Redes Neurais Convolucionais (CNN)

A pergunta que não quer calar: mas porque não podemos simplesmente utilizar uma **rede neural profunda totalmente conectada** como já conhecemos para reconhecer imagens?

Ainda que isso seja viável para imagens pequenas, imagine um banco de dados composto por imagens de 100 x 100 pixels. Cada imagem teria 10.000 pixels. Para uma rede com uma primeira camada oculta com “apenas” 1000 neurônios (o que já prejudica a quantidade de informação transmitida´), teríamos um total de **10 milhões de conexões** apenas na primeira etapa.

Isso significa uma **quantidade de computação excessivamente grande** e um treinamento muito complexo, sem garantias de bom resultado.

Redes Neurais Convolucionais (CNN)

A parte mais importante da CNN é a **camada convolucional**: neurônios na primeira camada convolucional **não estão conectados** a todos os pixels da imagem original, mas apenas a pixels de seus respectivos **campos receptivos**. Os neurônios da segunda camada convolucional estão conectados apenas a pequenos retângulos provenientes da camada anterior. E assim sucessivamente.

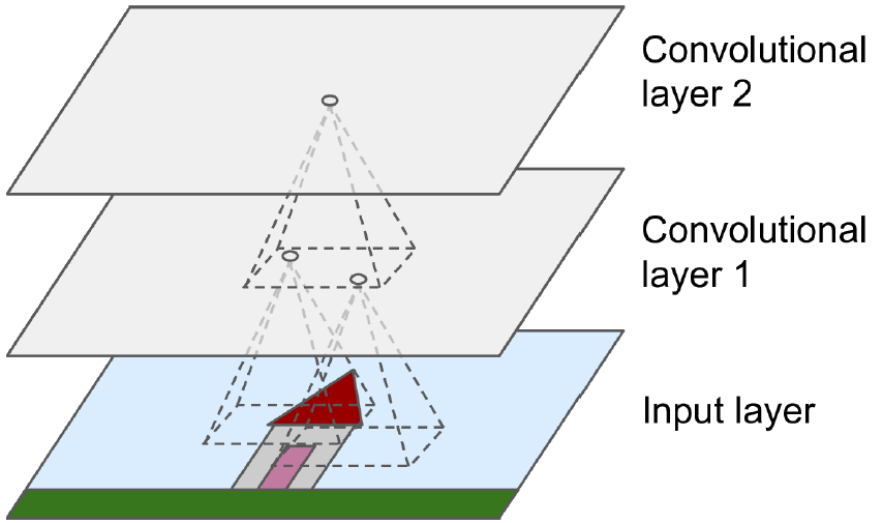


Imagem do livro "Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow." (GÉRON, Aurélien).

Essa arquitetura permite que a rede se concentre em **características simples e de baixo nível** na primeira camada oculta, que são combinados na próxima camada em **características de nível mais alto**, e assim por diante, até a camada final, responsável pela classificação.

Portanto, camadas totalmente conectadas aprendem padrões globais do dado de entrada, enquanto **camadas convolucionais aprendem padrões locais**.

Redes Neurais Convolucionais (CNN)

Convolução é a operação matemática de **somatório do produto** entre duas funções ao longo da região em que se sobrepõem. No caso das imagens, essa soma de produtos é realizada entre uma região da **imagem** e uma **matriz filtro** (chamada de máscara ou *kernel*, menor do que a imagem), resultando em uma nova imagem que contém informações de interesse da imagem original (chamada da *feature map*) . Dependendo do filtro, características diferentes são obtidas, como a suavização, deslocamento, contraste, bordas, texturas, remoção de ruídos, etc.

Durante a convolução, **o *kernel* vai se deslocando ao longo da imagem**, como uma janela móvel , e vai multiplicando e somando os valores sobrepostos. O passo com que o filtro caminha é chamado de *stride*.

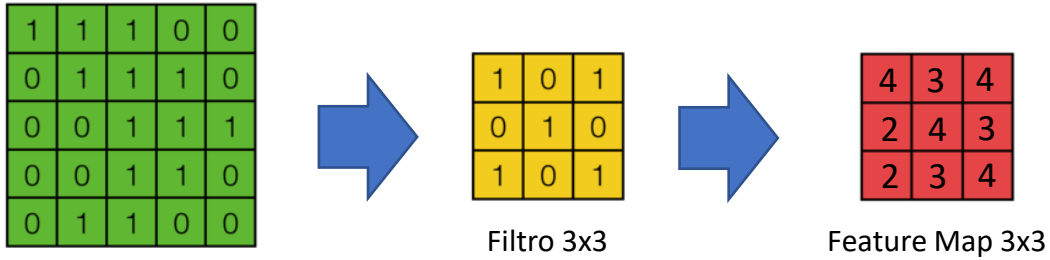
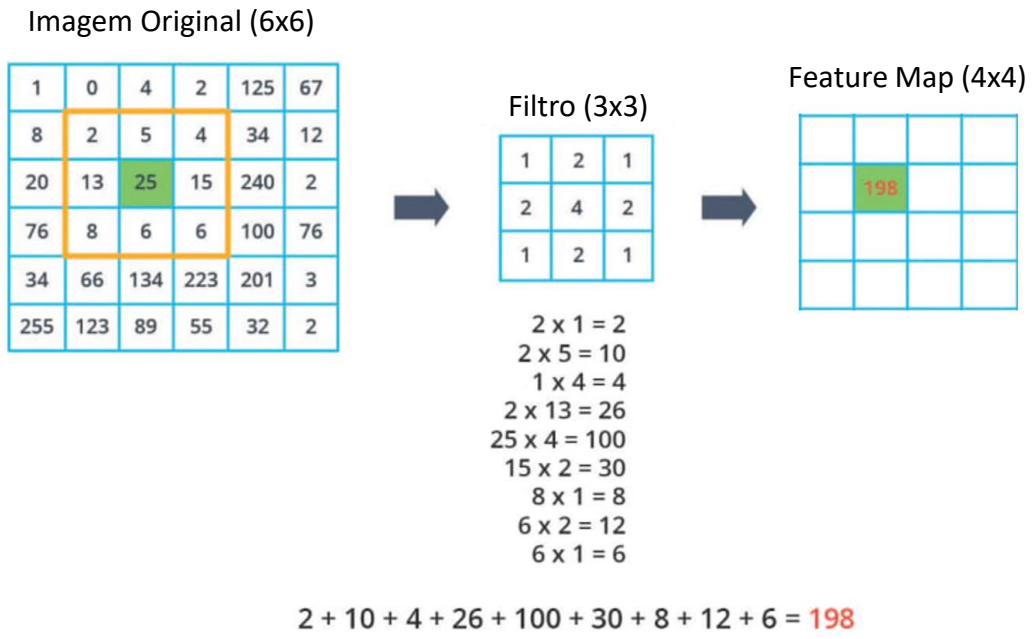


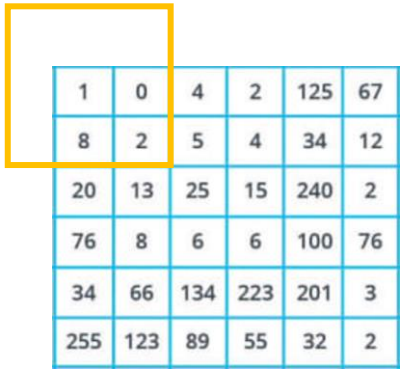
Imagem 5x5 Filtro 3x3 Feature Map 3x3

Redes Neurais Convolucionais (CNN)



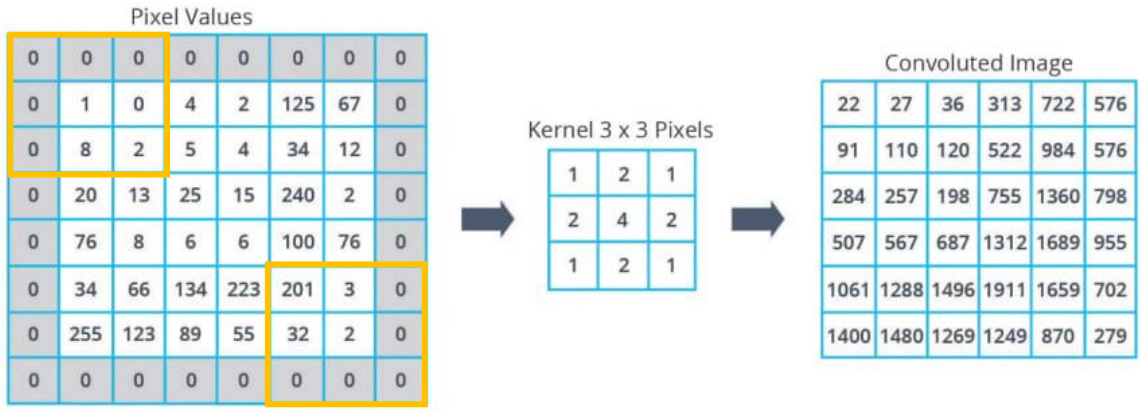
A imagem ao lado mostra **o resultado de um passo da convolução**. Esse passo precisa ser feito para cada elemento do *feature map*. No exemplo, temos um *stride* de 1, ou seja, a janela caminha de um em um.

Repare que os cantos não são processados adequadamente. Nos **cantos da imagem**, não conseguimos aplicar o filtro. Veja:



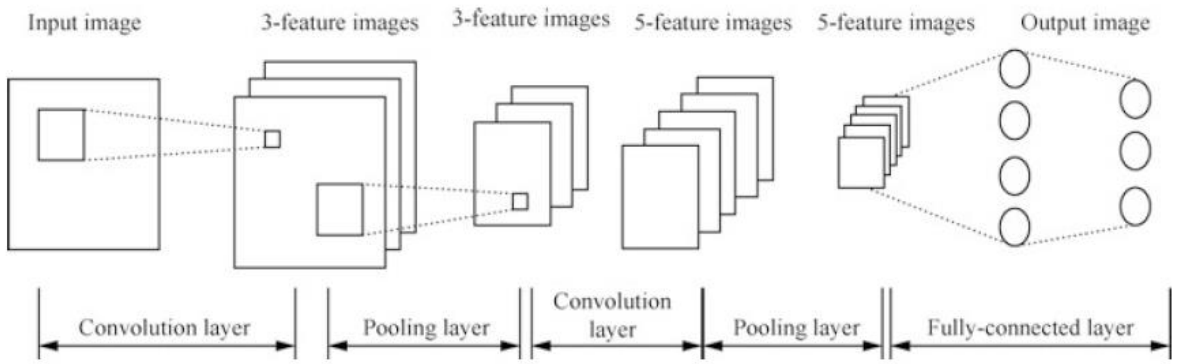
Redes Neurais Convolucionais (CNN)

Para resolver este problema, podemos expandir a imagem em todas as direções, preenchendo as linhas e colunas adicionais com o valor zero. Com isso, toda a imagem serve como entrada para a convolução, e nada se perde no processo. Essa técnica é chamada de *zero padding*.



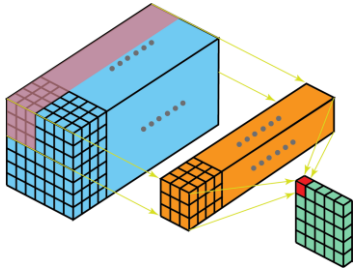
Na rede convolucional, **os parâmetros do modelo são os coeficientes dos filtros** que serão aplicados à imagem a cada camada (na imagem acima: 1,2,1,2,4. etc.). Eles são os **pesos dos neurônios**, que são ajustados durante do treinamento, até que os filtros sejam capazes de extrair exatamente as características da imagem (*features*) que sejam importantes para a tarefa que estamos tentando realizar (classificação).

Redes Neurais Convolucionais (CNN)

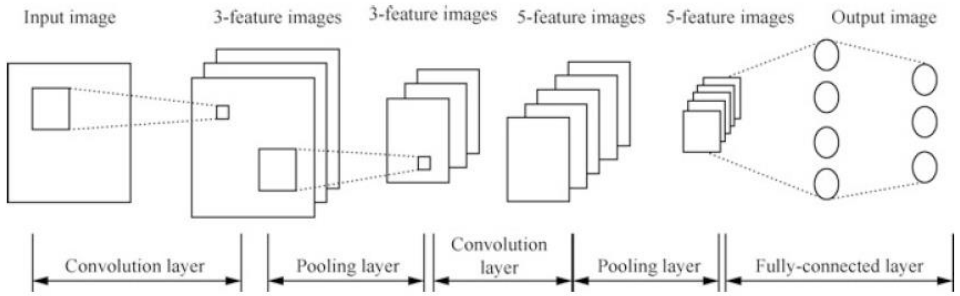


A imagem acima representa **uma rede neural convolucional**. Inicialmente, uma imagem serve como entrada da rede e passa por uma camada convolucional. **Cada camada convolucional pode ter mais de um filtro**, cada um obtendo diferentes *features* dos dados. Nesse caso, são 3 filtros de mesmo tamanho mas independentes, que formam uma matriz tridimensional como saída. Pode haver muito mais filtros por camada, como 16, 64, etc. Os **filtros funcionam como neurônios**, e seus **parâmetros são os pesos** ajustados pelo algoritmo de *backpropagation*. Depois da convolução, ainda na camada convolucional, os *feature maps* precisam ser "ativados" a partir de uma **função de ativação**. Geralmente, é utilizada a **ReLU**.

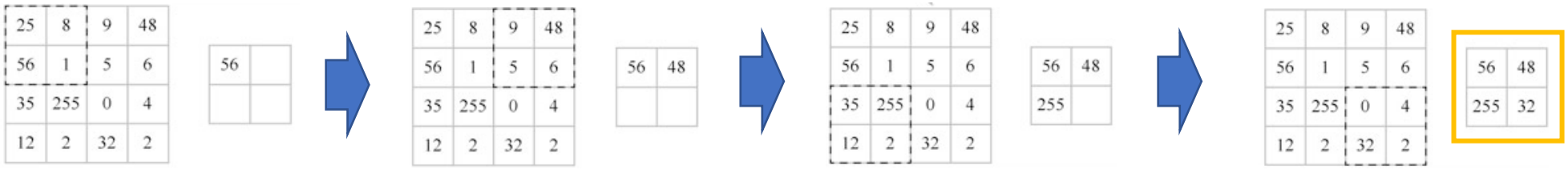
O **número de canais do filtro** precisa ser o mesmo da imagem ou *feature map* em que ele é aplicado (por ex. 3 x 3 x 5, na segunda camada).



Redes Neurais Convolucionais (CNN)



A seguir, os *feature maps* passam por um processo de redução de dimensionalidade na **camada de *pooling***. Os dois tipos de processo mais utilizados são o *max pooling* e o *average pooling*. Seu papel é basicamente reduzir a intensidade da computação e prevenir a ocorrência de *overfitting*. Além disso, é capaz de normalizar imagens para uma determinada dimensão necessária para a computação, tornando a rede capaz de trabalhar com imagens de qualquer tamanho.

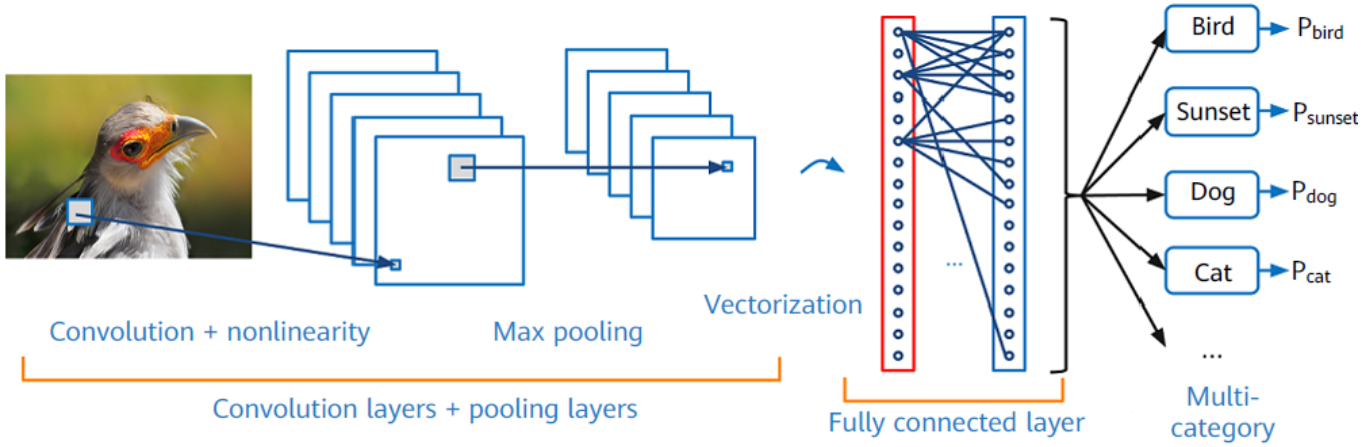


Processo de ***Max Pooling*** com *pool window size* e stride iguais a 2. *Average Pooling* funciona de maneira análoga, mas tira a media ao invés de pegar o valor máximo.

Redes Neurais Convolucionais (CNN)

Por fim, após várias camadas de convolução e *pooling*, **camadas totalmente conectadas** servem como saída da CNN. Para isso, o último *feature map* obtido é projetado como um vetor de tamanho fixo (etapa geralmente chamada de *flatten*) e submetido a uma ou mais camadas totalmente conectadas, geralmente também com função de ativação **ReLU**.

A última camada é basicamente um **classificador**, baseado nas informações obtidas nas camadas anteriores e combinando finalmente as *features* locais em *features* globais. A função de ativação **Softmax** é a mais utilizada, por ser adequada a problemas de classificação com várias classes.

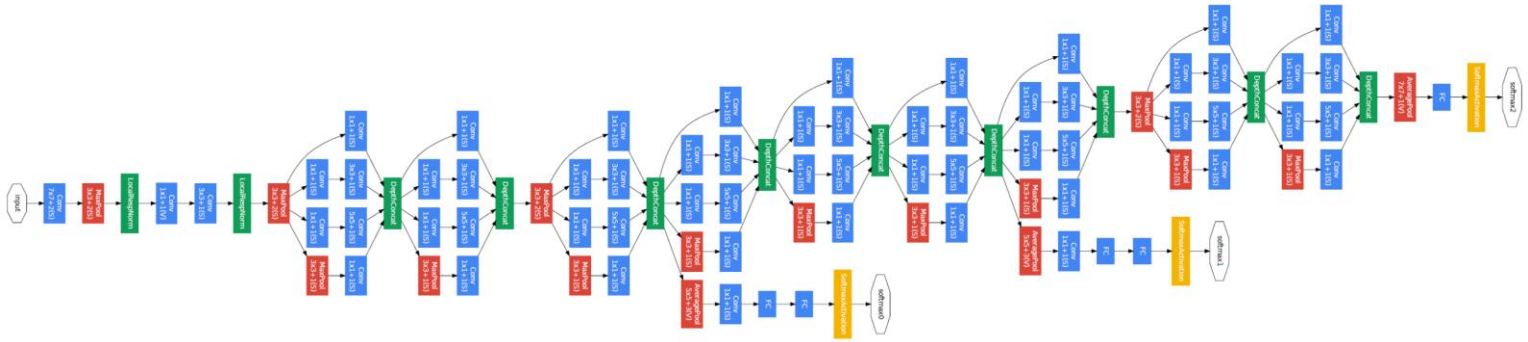


Redes Neurais Convolucionais (CNN)

Existem várias **arquiteturas de redes convolucionais** disponíveis que foram fundamentais na construção de algoritmos que alimentam e devem alimentar o campo de estudos da Inteligência Artificial como um todo no futuro próximo. São arquiteturas validadas, muitas vezes ganhadoras de concursos mundiais de classificação de imagens.

Alguns deles estão listados abaixo:

1. LeNet
2. AlexNet
3. VGGNet
4. GoogLeNet
5. ResNet
6. ZFNet



Redes Neurais Recorrentes (RNN)

As Redes Neurais Recorrentes (RNN, de *Recurrent Neural Networks*) são redes capazes de capturar a dinâmica de uma **informação serializada**, armazenando informação sobre o **contexto** desses dados na sequência em que estão inseridos. Por este motivo, ela pode ser expandida para a dimensão do tempo, ou seja, é **capaz de processar dados que vão evoluindo em uma contexto temporal**, como vídeo, áudio e texto.

As RNN são utilizadas em **dados temporais**, como a previsão de preços na bolsa de valores ou o clima para a próxima semana. Também pode prever **trajetórias** de carros e pedestres em sistemas de carros autônomos, evitando acidentes. Pode classificar **textos e documentos** de acordo com seu conteúdo, indicando o sentimento do autor ou sua intenção (NLP).

Os humanos não começam a pensar do zero a cada segundo. Ao assistir essa aula, você entende cada palavra com base em sua compreensão pré-adquirida. Você não joga tudo fora e começa a pensar de novo a cada palavra que você lê. Seus pensamentos têm persistência. As redes neurais tradicionais não podem fazer isso, mas as RNN podem.

Redes Neurais Recorrentes (RNN)

As redes que vimos até agora são *feedforward*, ou seja, a informação flui apenas em um sentido: da camada de entrada em direção à camada de saída. A RNN parece se assemelha às redes feedforward, mas possui algumas **conexões que apontam para trás**, como um *feedback*. Veja uma representação simples:

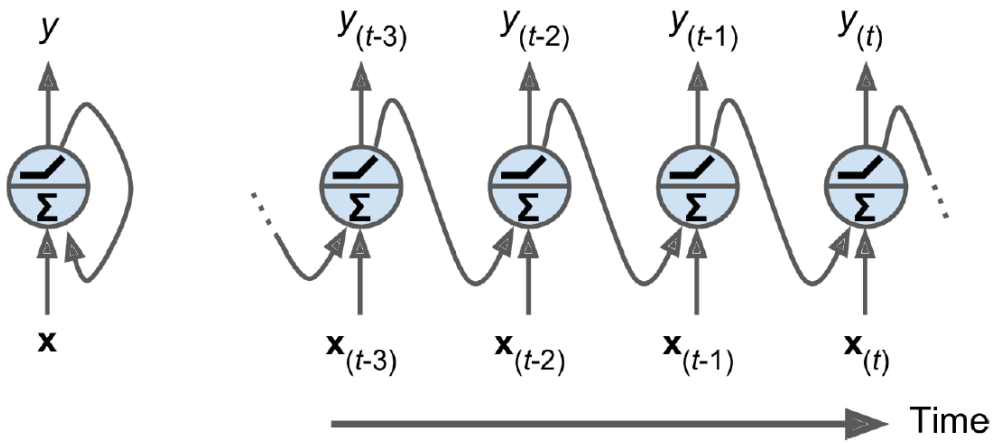


Imagem do livro "Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow." (GÉRON, Aurélien).

O que a imagem ao lado basicamente quer dizer, é que, a cada momento no tempo, **o neurônio recebe tanto o seu vetor de entrada $x_{(t)}$ e o vetor de saída obtido na iteração anterior $y_{(t-1)}$**.

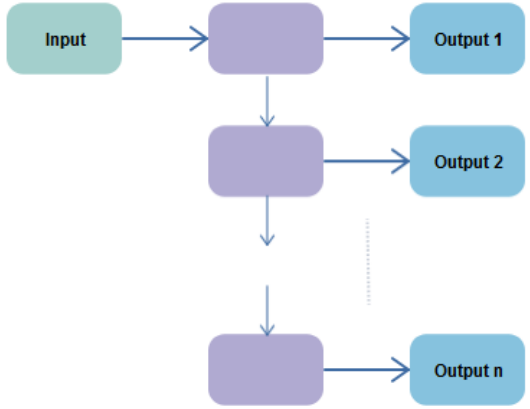
Cada neurônio recorrente possui dois conjuntos de pesos, um para a entrada e outro para as saídas do momento anterior. Esses pesos contêm informação sobre a influência do estado atual e dos estados anteriores para a obtenção da saída da rede. Unidades recorrentes são chamadas de **células de memória**.

Redes Neurais Recorrentes (RNN)

Há diferentes tipos de arquiteturas de **Redes Recorrentes**:

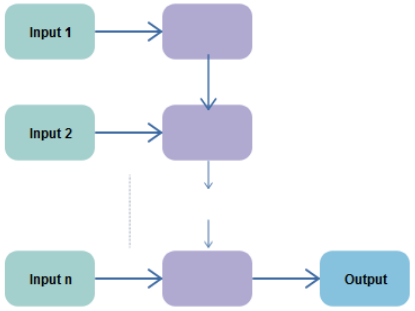
1. **One-to-One**: É o tipo comum de rede neural. Não tem relação com tempo. Para cada entrada, há uma saída.

2. **One-to-Many:** É um tipo de RNN generativa, que retorna uma sequência de dados de saída a partir de uma única entrada. É muito utilizada para geração de música e de legendas para imagens.

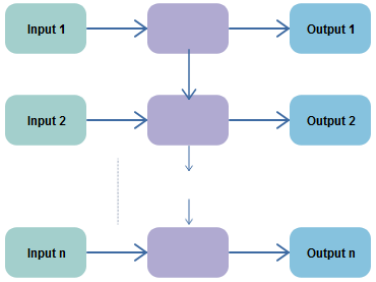


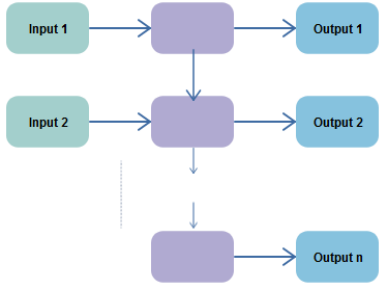
Redes Neurais Recorrentes (RNN)

3. **Many-to-One**: RNN utilizada quando uma única saída é obtida a partir uma sequência de entradas (vídeo, áudio, texto). Muito utilizada em análise de sentimento em textos e reconhecimento de voz.



4. **Many-to-Many:** É utilizada para gerar uma sequência de saídas a partir de uma sequência de entradas. Pode ser utilizada em aplicações de tradução voz-para-texto (e o oposto).





Redes Neurais Recorrentes (RNN)

O treinamento das RNN é realizado através do *backpropagation*, mas em uma variação que considera a evolução no domínio do **tempo** chamada *Backpropagation Through Time* (**BPTT**).

Apesar das vantagens desse tipo de rede, o treinamento das RNN é mais **complexo** e computacionalmente **intenso** do que o de uma rede comum, devido a sua recorrência:

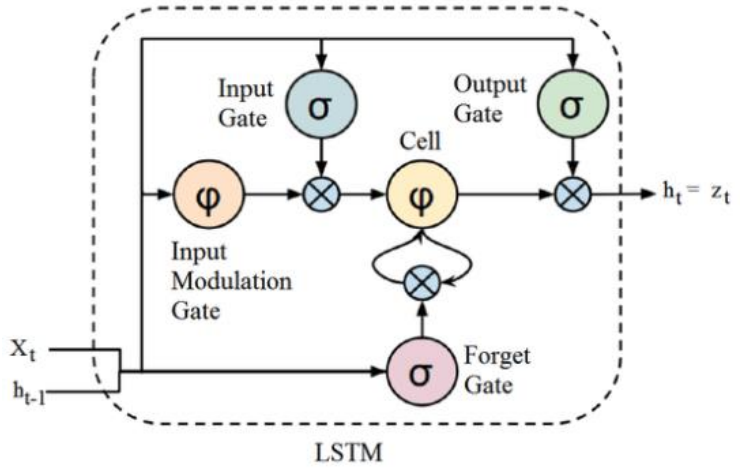
- Há **mais pesos** envolvidos no treinamento (peso da conexão recorrente).
- Para cada célula de memória, há **duas fontes de erros**, relacionadas ao estado atual e ao anterior. Eles são somados para cálculo da função de perda e subsequente ajuste com gradiente.
- Quanto mais longa for a sequência, maior a possibilidade dos problemas de ***vanishing gradient*** ou ***exploding gradient***, em que o gradiente pode ficar pequeno demais para gerar mudanças nos pesos ou grande demais, devido ao acúmulo de erros grandes, causando uma escalada exponencial.

Redes Neurais Recorrentes (RNN)

Para algumas tarefas, esperamos que o modelo seja capaz de **manter a informação por um tempo mais longo**. No entanto, a célula de memória tem capacidade limitada, e a RNN não é capaz de memorizar toda a sequência de dados. Por isso, seria interessante que as células de memória fossem capazes de **selecionar o que irão armazenar**, guardando apenas **informações chaves** sobre os dados e **descartando** outras. Isso pode ser atingido através de redes do tipo Long Short-Term Memory (**LSTM**).

A LSTM é útil para classificar, processar e prever séries temporais com intervalos de tempo de duração desconhecida. A insensibilidade relativa ao comprimento da sequência dá uma vantagem à LSTM em relação a RNN tradicional.

Ela é composta por **blocos LSTM** no lugar das camadas ocultas comuns da RNN. Esses blocos possuem três unidades para computação: *Input gate*, *Forget gate* e *Output gate*. Essas unidades fazem com que a LSTM seja capaz de lembrar ou esquecer seletivamente informação sobre a entrada.



Redes Generativas Adversárias (GAN)

As Redes Adversárias Generativas (**GAN**, de *Generative Adversarial Networks*) são arquiteturas de redes neurais profundas compostas por duas redes colocadas uma contra a outra (daí o nome "adversárias"). Esta é uma das arquiteturas mais recentes (**2014**, *Goodfellow et al.*) e fascinantes em Deep Learning. São capazes de **gerar imagens sintéticas bastante realísticas**, através de um processo de treinamento que força os limites até que as imagens sintéticas sejam estatisticamente **indistinguíveis** das reais. Pode também ser utilizado para geração de texto, melhoria de áudio e imagem, etc.

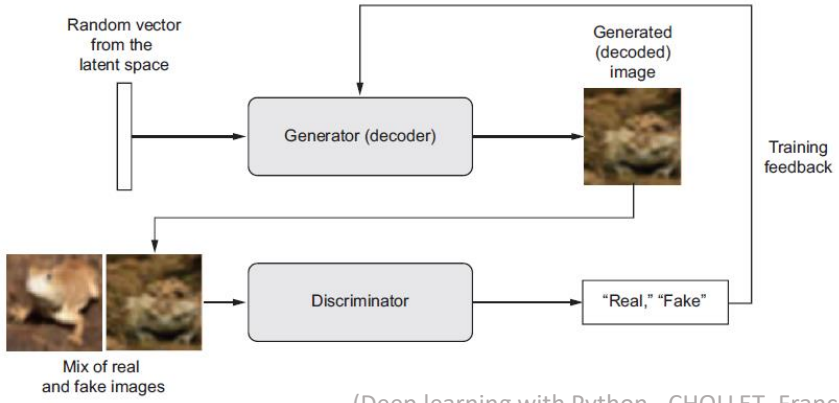
"Uma maneira de entender a GAN é imaginar um falsificador tentando criar uma **pintura falsa** de Picasso. Inicialmente, o falsificador é muito ruim. Ele mistura algumas de suas falsificações com Picassos autênticos e os mostra para um negociante de arte. O negociante de arte faz uma avaliação de autenticidade para cada pintura e dá ao falsificador *feedback* sobre o que faz um Picasso parecer um Picasso. O falsificador volta ao seu estúdio para preparar algumas novas falsificações. Com o passar do tempo, **o falsificador se torna cada vez mais competente em imitar o estilo de Picasso, e o negociante de arte se torna cada vez mais especialista em identificar falsificações**. No final, eles têm em mãos alguns excelentes Picassos falsos."

(Deep learning with Python - CHOLLET, François).

Redes Generativas Adversárias (GAN)

Portanto, o processo adversário da GAN treina um gerador G e um discriminador D. O treinamento é realizado com uma versão do *backpropagation*, e seu ajuste para um bom resultado é notoriamente difícil.

- O **discriminador D** é projetado para determinar se um exemplo é real ou gerado por G. Toma como entrada uma imagem e um valor de probabilidade referente à "autenticidade" dessa imagem. É um modelo neural comum, CNN ou MLP.
- O **gerador G** tem como objetivo gerar um exemplo tão bom que **seja capaz de "enganar" o discriminador D**. Toma como entrada um vetor aleatório (ruído, a partir de uma distribuição normal ou gaussiana) e o decodifica em uma imagem sintética. À medida que o gerador melhora, o desempenho do discriminador piora, por não conseguir facilmente distinguir entre real e falso. Se o gerador for bem-sucedido, o discriminador terá uma precisão de cerca de 50% (**Equilíbrio de Nash**). Na verdade, o discriminador "joga uma moeda" para fazer a previsão. O estado de **convergência** é tênue, ao invés de estável.



(Deep learning with Python - CHOLLET, François)

Redes Generativas Adversárias (GAN)

A polêmica do **Deep Fake**.

Vídeos falsos, como discurso de Obama sobre IA elaborado por Jordan Peele.

https://www.youtube.com/watch?v=cQ54GDm1eL0

E a inserção de Tom Cruise em situações e ambientes em que ele não esteve

https://www.youtube.com/watch?v=iyiOVUbsPcM

Apoio

Este projeto é apoiado pelo Ministério da Ciência, Tecnologia e Inovações, com recursos da Lei nº 8.248, de 23 de outubro de 1991, no âmbito do [PPI-Softex| PNM-Design], coordenado pela Softex.

João Paulo Reus Rodrigues Leite
Universidade Federal de Itajubá
e-mail: joaopaulo@unifei.edu.br

UNIFEI

Softex

MCTI FUTURO
FUTURO DO TRABALHO, TRABALHO DO FUTURO

GOVERNO FEDERAL
BRASIL
UNIÃO E RECONSTRUÇÃO