UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE MATEMÁTICA, ESTATÍSTICA E COMPUTAÇÃO CIENTÍFICA
DEPARTAMENTO DE MATEMÁTICA APLICADA

HUGO RICARDO RIBEIRO MATAROZZI

Redes neurais e imagens: um estudo sobre redes convolucionais

Campinas
17/07/2022

HUGO RICARDO RIBEIRO MATAROZZI

Redes neurais e imagens: um estudo sobre redes convolucionais

Monografia apresentada ao Instituto de Matemática, Estatística e Computação Científica da Universidade Estadual de Campinas como parte dos requisitos para obtenção de créditos na disciplina Projeto Supervisionado, sob a orientação do(a) Prof. João Batista Florindo.

Resumo

O presente trabalho é um estudo em relação à redes neurais, com o objetivo de apresentar o conceito de rede neural convolucional, muito usada atualmente principalmente em problemas que utilizam imagens. Foi feita uma apresentação e abordagem teórica de alguns dos principais conceitos presentes em uma rede neural artificial, tanto usual quanto convolucional, resultando na diferenciação destas. Isso permite concluir as vantagens de utilizar uma arquitetura ou outra para cada problema, e como a rede neural convolucional tem, de fato, melhores ferramentes para o trabalho com imagens.

Abstract

This paper is a study about neural networks, and its goal is to introduce the concept of convolutional neural network, highly used nowadays mainly in problems that use images. An introduction was made, as well as a theoretic description of some of the main concepts present in an artificial neural network, both usual and convolutional, distinguishing both. This allows us to conclude the advantages of, given a problem, choosing an architecture or another, and how convolutional neural networks have, indeed, proper tools for working with images.

Conteúdo

Lista de Figuras

1 Introdução

O aprendizado de máquina se mostrou extremamente relevante nos últimos anos, quando consideradas suas aplicações em diferentes indústrias. No caso, por mais que aplicações de modelos mais simples bastassem para a resolução de diversos problemas, algumas situações exigem o uso de modelos mais complexos que têm, como uma alternativa, o uso de redes neurais artificiais.

O que antes tinha um componente analógico pôde, com avanços tecnológicos relativamente recentes, ser expandido para uma modelagem totalmente digital com aumento de complexidade e performance e, assim, se popularizar com diversas arquiteturas para a aplicação em diferentes áreas de conhecimento.

As redes neurais, junto com ideias vindas do tratamento de imagens e do aumento de capacidade de processamento e armazenagem de dados, puderam ter uma arquitetura especializada no tema na forma de redes neurais convolucionais.

A visão computacional é um assunto relevante para problemas que vão desde classificação de imagens até detecção de ações, reconhecimento facial ou até o funcionamento de carros autonômos.

2 Redes Neurais

Primeiramente é necessário abordar redes neurais simples, com arquiteturas mais "diretas", de forma a compreender suas limitações ao trabalhar com imagens e, assim, valorizar as soluções trazidas pelas redes convolucionais.

2.1 Estrutura

As redes neurais artificiais surgem inspiradas pelas redes neurais do cérebro humano, e foram muito usadas nas décadas de 80 e 90, recuperando sua relevância mais recentemente dados os avanços tecnológicos.

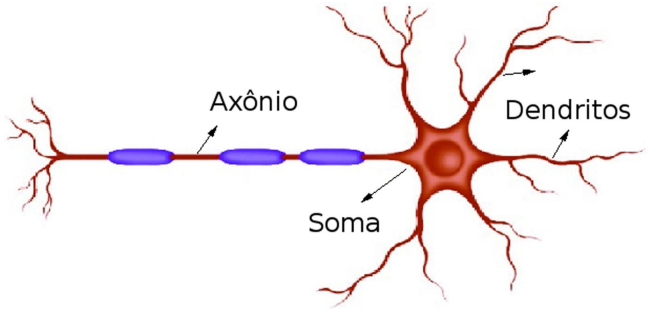


Figura 1: Representação de um neurônio

Fonte: https://www.scielo.br/j/rbef/a/n4Q49fBXdH9NvKT4X9ZjcHD/?lang=pt

Em um neurônio, os dendritos são responsáveis pela entrada de sinais enquanto o axônio, pela saída. Dessa maneira, uma célula de uma rede neural artifical (neuron), inspirada nesta estrutura, pode ser dividida em três partes: camada de entrada, camada(s) intermediária(s)/escondida(s) e camada de saída.

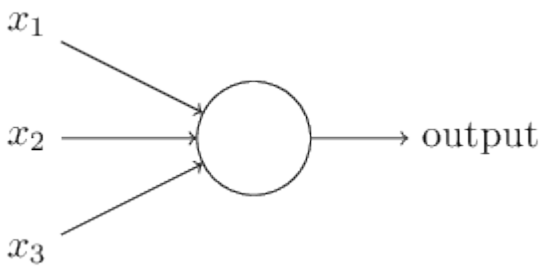


Figura 2: Representação de uma célula simples de rede neural

Fonte: http://neuralnetworksanddeeplearning.com/chap1.html

Uma rede neural artificial simples é uma conexão de unidades como a representada. Além disso, cada unidade realiza uma operação não-linear, por meio da chamada função de ativação, em uma combinação linear dos valores de entrada. Isso é feito pois, dessa maneira, as diferentes saídas das células se tornam linearmente independentes e podem assim ser combinadas sem carregar informações redundantes.

Matematicamente, para n entradas x_i e respectivos pesos w_i e função de ativação $g(.)$, podemos denotar o valor de saída a por:

$$a = g(w_0 x_0 + \sum_{i=1}^{n} w_i x_i) = g(\sum_{i=0}^{n} w_i x_i) = g(z)$$

onde $x_0 = 1$ é o chamado *bias*, com w_0 seu respectivo peso.

Como mencionado, uma rede neural é composta por diversas unidades como a descrita, formando uma ou mais camadas. Isso permite um grande número de combinações de valores linearmente independentes a partir de um conjunto de dados, algo que na prática facilita a realização de operações complexas de forma simples.

Tomemos como exemplo (inspirado por Ng [2021]) a operação XOR (operação ou exclusivo). Usando como função de ativação a função sigmoide logística $g(z) = \frac{1}{1+e^{-z}}$ podemos criar as operações elementares AND, OR e NOT:

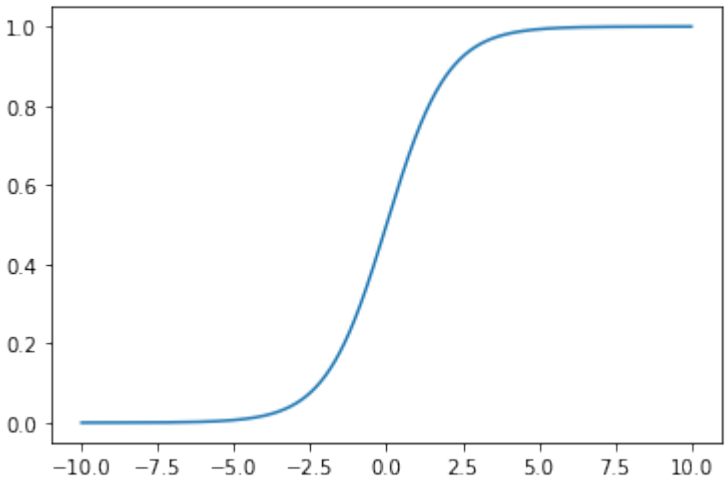
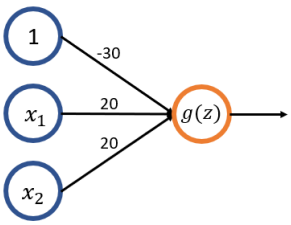
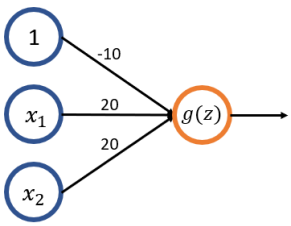


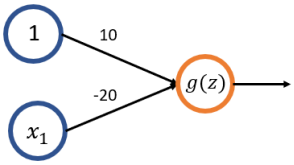
Figura 3: Gráfico da função sigmóide logistica



(a) Rede que aplica função AND



(b) Rede que aplica função OR



(c) Rede que aplica função NOT

Figura 4: Redes referentes às funções AND, OR e NOT

E combiná-las para criar uma rede que calcula a operação XOR:

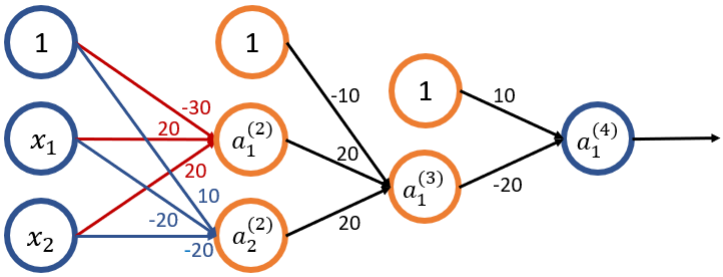


Figura 5: Rede que aplica função XOR

Note que, nesse caso, os pesos foram dados e estes caracterizaram as operações que cada célula realiza. No contexto de aprendizado de máquina, porém, os pesos não são previamente conhecidos e são exatamente os parâmetros a serem aprendidos e ajustados.

2.2 Breve história

O primeiro modelo lógico de rede neural foi realizado em 1943 pelo neurofisiologista Warren McCulloch e o matemático Walter Pitts (McCulloch and Pitts [1943]), fortemente estimulado pelo comportamento do cérebro e pelo "estímulo"que neurônios dentro de uma rede recebiam, vistos como respostas para simples proposições lógicas. O modelo foi criado então com premissas e deenvolvimentos matemáticos, e estes deram início ao estudo que chegaria a se popularizar tanto hoje.

Posteriormente, em 1958, Frank Rosenblatt (Rosenblatt [1958]) criou o *perceptron*, famoso primeiro algoritmo que de fato utilizava o modelo de redes neurais, previamente proposto e discutido, para resolução de problemas.

Sua arquitetura, simples, continha somente duas camadas, sendo somente uma camada escondida. É o caso, por exemplo, da rede representada na figura 2.

2.3 Aplicando aprendizado de máquina

Como mencionado, o aprendizado de máquina entra no contexto de redes neurais quando os pesos das combinações lineares entre cada camada são aprendidos, e não dados. Para isso, buscamos estruturar e resolver um problema de otimização. Temos as seguintes etapas e elementos:

2.3.1 *Forward propagation*

O *forward propagation* é a "execução"da rede neural, desde a camada de entrada até a de saída, assim como visto no exemplo da função XOR. No contexto de aprendizado de máquinas, é relevante tanto para avaliar a qualidade das saídas (i.e. quão bem os pesos estão sendo"aprendidos") quanto para efetuar predições após o treinamento de fato.

2.3.2 *Cost function*

A *cost function* (ou *loss function*) é a função objetivo do problema de otimização a ser minimizada. Para cada problema a ser resolvido e método de otimização a ser usado, uma diferente função é pensada como função de custo, tendo em mente a simplicidade de implementação, existência de mínimos locais e velocidade/eficiência, entre outros aspectos.

2.3.3 *Backpropagation*

O *backpropagation* (Rumelhart et al. [1986]) é o algoritmo que, de fato, permite o aprendizado dos pesos. De forma contrária ao *forward propagation*, que percorre a rede da entrada até a saída, o *backpropagation* percorre a rede da saída até a entrada, visando ajustar os pesos a partir do valor da *cost function*.

Pensado com base no algoritmo de otimização do gradiente de descida, são usadas derivadas parciais (com uso da regra da cadeia para cada camada) de forma a avaliar a sensibilidade do resultado às variações dos pesos e fazer os ajustes apropriados a cada iteração.

A rede é treinada e avaliada a cada iteração, de forma a ser verificado se esta é aderente aos dados já conhecidos (de treino) mas também generaliza para casos desconhecidos. Isso é realizado através de diferentes métricas, o que informa se são necessários ajustes nos parâmetros da rede ou mais iterações para treinamento.

3 Redes Neurais Convolucionais

Abordamos agora as redes neurais convolucionais, que levam esse nome a partir da operação de convolução (ou melhor, relação cruzada) que é realizada em uma camada da rede. Muito útil para grandes quantidades de dados, se tornou relevante com o aumento de capacidade computacional para trabalhar com grandes imagens.

3.1 Motivação

Para uma rede neural artificial usual, com camada de entrada e duas camadas escondidas de 10 nós cada, e uma camada de saída com somente um nó, temos da ordem de 100 conexões com seus pesos w. Ao considerar, porém, uma imagem de 1024×1024 e somente uma camada escondida de 1000 nós, nosso número de conexões já passa de 10^9.

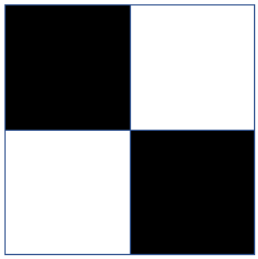
Para trabalhar com imagens cada vez maiores (e com conjuntos cada vez maiores de imagens), a capacidade computacional se torna um gargalo ao considerar a arquitetura padrão de uma rede neural. É necessário, então, pensar em como aproveitar a estrutura da imagem para simplificar a rede assim como melhorar os resultados do aprendizado.

3.2 Operações com imagens - Convolução

A estrutura entre os valores em uma imagem permitem com que realizemos operações de forma a otimizar a informação contida nos dados, para o propósito de realizar predições com redes neurais.

A operação de convolução, no contexto de imagens, é o nome dado para a operação de relação cruzada em processamento de sinais. É aplicado um filtro na imagem, de forma que algumas características se sobressaiam e outras se suavizem. No caso padrão, dada uma imagem de tamanho $n \times n$ e um filtro de tamanho $m \times m$, $m < n$, a aplicação da convolução é dada pela soma do produto ponto-a-ponto entre o valores do filtro e da imagem, deslocando o filtro apropriadamente.

Um simples exemplo (inspirado por Ng [2022]) é, para uma imagem de um pedaço $2x2$ de um tabuleiro de xadrez, que pode ser representada por:



(a) Imagem xadrez

10	10	10	10	0	0	0	0
10	10	10	10	0	0	0	0
10	10	10	10	0	0	0	0
10	10	10	10	0	0	0	0
0	0	0	0	10	10	10	10
0	0	0	0	10	10	10	10
0	0	0	0	10	10	10	10
0	0	0	0	10	10	10	10

(b) Representação

Figura 6: Imagem de xadrez e representação correspondente

Podemos aplicar os filtros de bordas verticais e horizontais, representados respectivamente por:

1	0	−1
1	0	−1
1	0	−1

(a) Filtro vertical

1	1	1
0	0	0
-1	-1	−1

(b) Filtro hoizontal

Figura 7: Filtros vertical e horizontal

Ao aplicar o filtro vertical, o primeiro valor será obtido ao sobrepor o filtro à imagem, começando pelo canto superior esquerdo. Assim, o valor será dado por:

$$1 \times 10 + 0 \times 10 + (-1)10 + 1 \times 10 + 0 \times 10 + (-1)10 + 1 \times 10 + 0 \times 10 + (-1)10 = 0$$

De fato, é esperado o valor nulo dado que não há nenhuma linha vertical. No terceiro caso, porém, dando dois passos à direita, obtemos:

$$1 \times 10 + 0 \times 10 + (-1) \times 0 + 1 \times 10 + 0 \times 10 + (-1) \times 0 + 1 \times 10 + 0 \times 10 + (-1) \times 0 = 30$$

O resultado final de aplicar a convolução da imagem com ambos os filtros é:

Note que os filtros capturaram os bordos que de fato se propunham a capturar. Além disso, houve a redução de dimensionalidade: uma imagem 8×8 foi reduzida, após a convolução com um filtro 3×3, para uma imagem de dimensão 6×6. Isso pode ser alterado, mas é algo relevante na forma com que redes neurais convolucionais operam.

0	0	30	30	0	0
0	0	30	30	0	0
0	0	10	10	0	0
0	0	−10	−10	0	0
0	0	−30	−30	0	0
0	0	−30	−30	0	0

(a) Convolução com filtro vertical

0	0	0	0	0	0
0	0	0	0	0	0
30	30	10	-10	−30	−30
30	30	10	−10	−30	−30
0	0	0	0	0	0
0	0	0	0	0	0

(b) Convolução com filtro horizontal

Figura 8: Resultado das convoluções com filtros vertical e horizontal

A convolução tem alguns parâmetros a serem considerados:

3.2.1 *Padding*

O *padding* consiste em preencher com zeros ao redor da imagem de forma a aumentar as dimensões do resultado da convolução com um filtro. Tomando exemplo anterior e realizando uma camada de *padding* obtemos:

0	0	0	0	0	0	0	0	0	0
0	10	10	10	10	0	0	0	0	0
0	10	10	10	10	0	0	0	0	0
0	10	10	10	10	0	0	0	0	0
0	10	10	10	10	0	0	0	0	0
0	0	0	0	0	10	10	10	10	0
0	0	0	0	0	10	10	10	10	0
0	0	0	0	0	10	10	10	10	0
0	0	0	0	0	10	10	10	10	0
0	0	0	0	0	0	0	0	0	0

(a) Exemplo com *padding*

−20	0	0	20	20	0	0	0
−30	0	0	30	30	0	0	0
−30	0	0	30	30	0	0	0
−20	0	0	10	10	0	0	10
−10	0	0	−10	-10	0	0	20
0	0	0	-30	-30	0	0	30
0	0	0	−30	−30	0	0	30
0	0	0	−20	−20	0	0	20

(b) Convolução com filtro vertical

Figura 9: Resultado da convolução com filtro vertical com *padding*

Note como o *padding* distorceu um pouco nesse resultado final, como preço por retornar informação das bordas e por manter o tamanho original.

3.2.2 *Stride*

O *stride*, ou tamanho do passo, é quanto que nosso filtro se desloca ao caminhar para os lados. É feito para diminuir ainda mais a imagem resultante, e com um custo menor de cálculo. Informações podem se perder, porém, se a diferença entre o tamanho da imagem e do filtro não for um múltiplo do valor de *stride* escolhido.

Naturalmente, segue nosso exemplo para $stride = 2$

0	30	0
0	10	0
0	-30	0

Figura 10: Resultado da convolução com filtro vertical de $stride = 2$

Note que obtivemos um resultado simples, pequeno e com boa parte da informação que buscávamos em relação à borda vertical.

Ambos os parâmetros apresentados, *padding* e *stride*, alteram o tamanho da imagem de saída. Podemos calcular esse tamanho por:

$$dim(n, p, f, s) = \left\lfloor \frac{n + 2p - f}{s} + 1 \right\rfloor$$

onde n é a dimensão da imagem, p o número de camadas para *padding*, f a dimensão do filtro e s o tamanho do *stride*.

3.3 Operações com imagens - Pooling

O *pooling* é uma operação de baixo custo, sem parâmetros a serem aprendidos, que visa diminuir a granularidade da imagem e, assim, diminuir sua dimensão. De forma semelhante à convolução, dado um filtro de tamanho $n \times n$ não aplicamos a soma ponderada pelos pesos, mas sim outra função, aos valores da imagem. Usando como exemplo a função máximo (*max pooling*) em nossa imagem, para $f = s = 2$, obtemos:

De fato, nossa imagem não precisava de toda a definição inicial, dada sua simplicidade. Mas esse não é sempre o caso: o *pooling* tem baixo custo e simplifica nossa

10	10	0	0
10	10	0	0
0	0	10	10
0	0	10	10

Figura 11: Resultado de *max pooling* de dimensão $f = 2$ e *stride* $= 2$

imagem, mas potencialmente com perda de informação relevante.

3.4 Arquiteturas de redes neurais convolucionais

3.4.1 Montando a rede neural convolucional

Após entender as operações que nos permitem o processamento de imagens, podemos considerá-las no contexto de uma rede neural. Nela, chamamos de totalmente conectada a camada que conhecemos de uma rede neural usual, e adicionamos ainda dois tipos de camadas: a convolucional e a de *pooling*.

A camada totalmente conectada tem, para serem aprendidos através da otimização, os pesos entre os nós. A camada convolucional, por outro lado, tem como valores w a serem aprendidos os filtros que são aplicados nas imagens. Para a quebra de linearidade, também é aplicada uma função de ativação nos valores obtidos ao realizar a convolução. Por fim, a camada de *pooling* não tem valores a serem aprendidos.

Uma rede neural convolucional, dessa maneira, geralmente é estruturada da seguinte forma: uma série de camadas convolucionais e de pooling que aplicam diversos filtros e, para cada filtro, criam uma imagem na camada seguinte.

Essas imagens podem ser representadas como uma imagem de diversos canais, um para cada resultado da convolução por um filtro. Assim, uma imagem vai sendo, a cada camada, transformada em outra de dimensões menores e com mais canais, até que estes canais componham os valores de entrada de uma camada totalmente conectada, onde a rede passa a se comportar como uma rede neural usual.

Em outras palavras, a rede convolucional adiciona camadas de processamento de imagem que extraem caracteristicas a serem inseridas em uma rede neural usual.

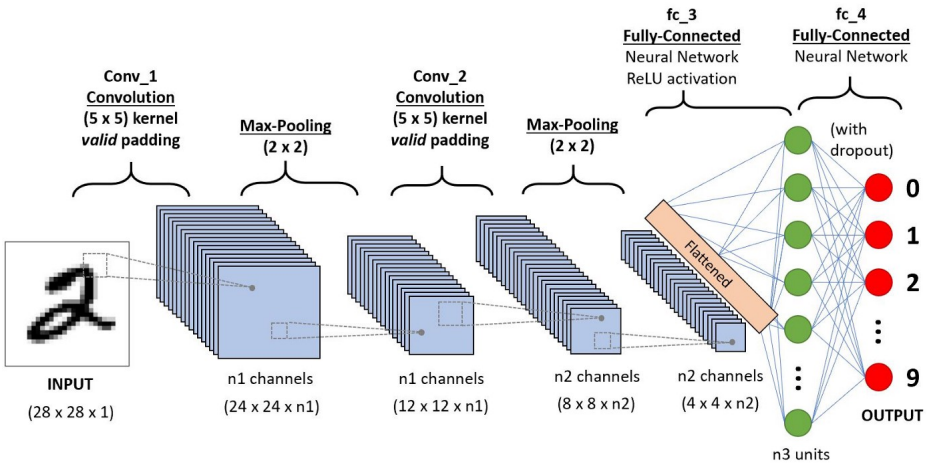


Figura 12: Representação de uma rede neural convolucional

Fonte: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

A partir destas camadas, diversas redes neurais podem e são criadas, resultantes de ajustes empíricos na resolução de problemas. Alguns exemplos são a LeNet - 5 (Lecun et al. [1998]), composta por dois pares de camadas convolucionais e *pooling* para, depois, conectar em duas camadas totalmente conectadas; AlexNet (Krizhevsky et al. [2012]), que realiza dois pares de camadas de convolução e *pooling* para, depois, realizar três camadas seguidas de convolução com *padding*, seguida de uma de *pooling* e, por fim, três totalmente conectadas; ou até a VGG - 16 (Simonyan and Zisserman [2014]), profunda, com 16 camadas no total.

3.4.2 Vantagens

Após observar como que é realizado o tratamento das imagens e a inserção das operações na arquitetura da rede neural, podemos considerar vantagens das redes convolucionais:

Pensando em nosso caso motivador, de uma imagem de 1024×1024, se temos uma camada convolucional de 10 filtros de dimensão 3×3, temos da ordem de 100 parâmetros a aprender, contra 1000 do caso anterior. Se incluirmos *stride* > 1 e, após isso, passarmos por uma camada de *pooling*, teremos realizado um pré-processamento relativamente barato computacionalmente, e terminaremos com informações extraídas da

imagem na forma de menos nós de entrada para a próxima camada.

Além disso, um filtro que extrai informações da imagem é invariante a translações. Como exemplo, pensemos no caso de um filtro de bordas verticais: não importa a região da imagem, ele consegue extrair essas características sem precisar de ajustes. Isso permite com que os pesos aprendidos na camada de convolução sejam utilizados para diversos valores de entrada, diferente de uma rede usual que pode capturar padrões em uma região mas não em outra.

Por fim, há um menor número de conexões: enquanto em uma camada totalmente conectada há ligação entre todos os nós das camadas, na rede convolucional há somente a influência dos valores capturados no filtro: para um filtro 3×3, somente 9 valores de entrada são usados para calcular um valor da próxima camada. Diminuir o número de conexões permite maior velocidade na hora de treinamento e predição.

4 Conclusão

O aumento da dimensão na camada de entrada é um fator limitante ao considerar imagens cada vez maiores, quando se trata de uma rede neural artificial. Operações com imagens, que visam filtrar as informações relevantes e capturar a estrutura entre os dados (como, por exemplo, contornos), são soluções simples e efetivas para melhorar a eficiência e o desempenho da rede.

As redes neurais convolucionais, assim, têm em sua arquitetura ferramentas que permitem o tratamento da imagem e extração de características, ao trabalhar também com a relação entre os valores de entrada. São, consequentemente, promissoras para a área de visão computacional sendo, de fato, presentes em suas aplicações.

Referências

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60:84—-90, 2012. doi: 10.1145/3065386.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278—-2324, 1998. doi: 10.1109/5.726791.

W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115—-133, 1943. doi: 10.1007/BF02478259.

A. Ng. Machine learning by stanford university. Online Course, 2021.

A. Ng. Convolutional neural networks by deeplearning.ai. Online Course, 2022.

F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386—-408, 1958. doi: 10.1037/h0042519.

D. Rumelhart, Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986. doi: 10.1038/323533a0.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv.1409.1556*, 2014. doi: https://doi.org/10.48550/arXiv.1409.1556.