Redes Neurais Convolucionais

1st Giovanna Carreira Marinho
Departamento de Matemática e Computação (DMC)
FCT - Universidade Estadual Paulista (UNESP)
Presidente Prudente, São Paulo, Brasil
g.marinho@unesp.br

2nd Paulo Henrique Silva Peres
Departamento de Matemática e Computação (DMC)
FCT - Universidade Estadual Paulista (UNESP)
Presidente Prudente, São Paulo, Brasil
paulo.h.peres@unesp.br

I. Introdução

Em 1958, um estudo sobre a estrutura do córtex visual foi realizado por David Hubel e Torsten Wiesel. Nesse estudo, foi mostrado que muitos neurônios dessa região possuem um pequeno campo receptivo local, significando que essas estruturas reagem apenas a estímulos visuais localizados em uma região limitada do campo visual. Juntos, os campos receptivos de diferentes neurônios podem se sobrepor para cobrir todo o campo visual. Já em 1980, um pesquisador inspirado no estudo anterior se tornou o pioneiro no desenvolvimento das Redes Neurais Convolucionais (ou *Convolutional Neural Networks* - CNN), Kunihiko Fukushima com a rede chamada Neocognitron. Seu design hierárquico e multicamadas permitiu o aprendizado de padrões visuais, como o reconhecimento de caracteres manuscritos. Yann LeCun e outros pesquisadores introduziram, em 1998, uma arquitetura muito famosa: LeNet-5, que apresentou novos elementos: camadas convolucionais e camadas de *pooling*.

Devido ao aumento do poder computacional, da disponibilidade de dados para treinamento e de ferramentas poderosas de programação para o treinamento de redes profundas, as CNNs conseguiram obter desempenhos surpreendentes em tarefas visuais complexas. Além de suas aplicações em carros autônomos, sistemas de classificação de vídeos, diagnóstico em imagens, entre outras, elas também são aplicadas em tarefas de reconhecimento de voz e processamento de linguagem natural.

II. Definição

As CNNs são redes neurais que aplicam operação de convolução entre seus pesos e as camadas anteriores, sendo utilizadas principalmente com imagens (mas não se restringem à isso). Por meio da "leitura" de uma imagem de entrada, ela atribui as devidas importâncias aos elementos da imagem (ou seja, os pesos e vieses que são calculados durante o treinamento) e é capaz de diferenciar um ao outro. A principal vantagem de uma CNN, é que seu pré-processamento necessário é muito menor em comparação com outros algoritmos de classificação. Como sua arquitetura é inspirada na organização do córtex visual, cada neurônio responde a estímulos apenas em uma região restrita do campo receptivo (ou campo visual), sendo a área visual total composta por uma coleção de campos receptivos [1]–[3].

Uma camada convolucional é composta por filtros que, durante o processo de treinamento, passam pela imagem de entrada (ou na camada anterior) realizando o processo de convolução (com as intensidades dos pixeis) e gerando os chamados mapa de características (*feature map* ou *activation map*, em inglês). As unidades/neurônios em um *feature map* recebem como entradas apenas de uma pequena sub-região da imagem (ou camada anterior) e todas essas unidades são restritas a compartilhar os mesmos valores de pesos. Essas unidades podem ser interpretadas como detectores de características, dessa forma, todas as unidades de um *feature map* detectam o mesmo padrão, porém em diferentes localizações da imagem de entrada. Como muitas vezes é necessário detectar várias características para construir um modelo eficaz, geralmente haverão vários *feature maps* na camada convolucional, cada um com seu próprio conjunto de parâmetros de pesos e viés (ou *bias*). A Fig. 1 apresenta um diagrama para explicação dessa arquitetura.

Nem todos os pixeis da entrada são conectados com um neurônio oculto. As conexões são feitas entre pequenas regiões da imagem de entrada (o chamado campo receptivo local) com um respectivo neurônio oculto. O campo receptivo local se desliza por toda a imagem, se conectando ao neurônio oculto. Isso permite que a rede se concentre em características de alto nível (como bordas e cores) nas primeiras camadas e as reúna em características de menor nível nas próximas camadas ocultas.

Os pesos de um neurônio podem ser representados como uma pequena imagem do mesmo tamanho do campo receptivo. Os filtros (ou também *kernels* convolucionais) são exemplos de conjuntos de pesos. Uma camada cheia de neurônios que compõem o mesmo filtro, gera um mapa de características (que destaca as regiões em uma imagem que mais ativam o filtro). Durante o treinamento, a camada convolucional aprende automaticamente os filtros necessários para realizar uma determinada tarefa. Os pesos (que definem o filtro/*kernel* e geram esse *feature map*) são chamados de pesos compartilhados. Na prática, uma camada convolucional possui vários filtros e gera um mapa de características por filtro. Dessa forma, muitas representações tratam esse resultado em 3 dimensões, como apresentado no diagrama da Fig. 1.

Em cada *feature map*, existe um neurônio por pixel e todos os neurônios dentro de um *feature map* compartilham os mesmos parâmetros (pesos e bias). Os neurônios em diferentes

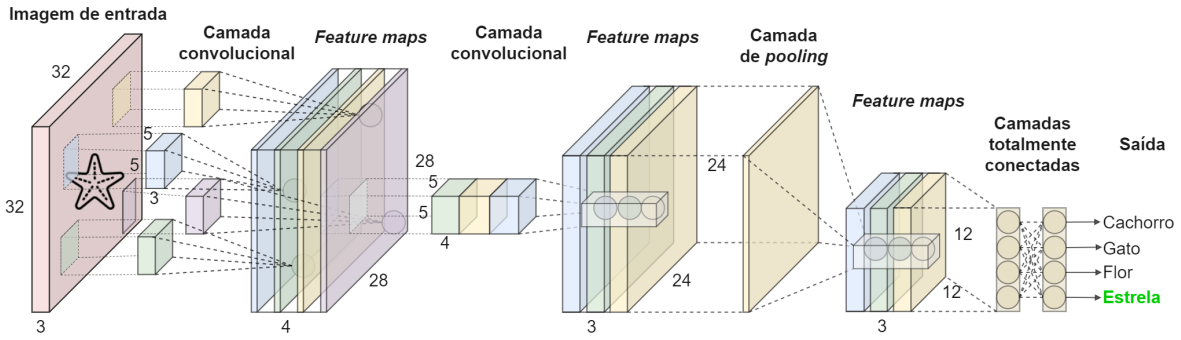

Fig. 1. Diagrama de uma CNN. Produzida pelos autores.

feature maps usam diferentes parâmetros. O campo receptivo associado a um neurônio se estende aos *feature maps* das camadas anteriores.

Considerando os conceitos anteriormente apresentados, podemos definir que o neurônio localizado na linha i e coluna j do *feature map* k de uma dada camada convolucional l é conectado com as saídas dos neurônios da camada anterior $l-1$, localizado nas linhas $i*s_h$ a $i*s_h+f_h-1$ e colunas $j*s_w$ a $j*s_w+f_w-1$, em todos os mapas de recursos na camada $l-1$. Ou seja, todos os neurônios localizados na mesma linha i e coluna j e mapas de recursos diferentes estão conectados às saídas dos mesmos neurônios da camada anterior. Assim, a saída $z_{i,j,k}$ de um neurônio localizado na linha i coluna j e *feature map* k em uma camada convolucional l é dada por:

$$z_{i,j,k} = b_k + \sum_{u=0}^{f_h-1} \sum_{v=0}^{f_w-1} \sum_{k'=0}^{f_{n'}-1} x_{i',j',k'} \times w_{u,v,k',k} \qquad (1)$$

com $i' = i \times s_h + u$ e $j' = j \times s_w + v$. Onde $\times$ representa o processo de convolução; s_h e s_w representam *stride* (deslocamento de um campo receptivo) horizontal e vertical, respectivamente; f_h e f_w a altura e largura de um campo receptivo; $f_{n'}$ o número de *feature maps* na camada anterior $(l-1)$; $x_{i',j',k'}$ saída do neurônio localizado na linha i', coluna j', *feature map* k' da camada $l-1$; b_k o termo de bias para o *feature map* k (na camada l); $w_{u,v,k',k}$ o peso entre um neurônio do *feature map* k da camada l e sua entrada localizada na linha u, coluna v e *feature map* k'. Para que uma camada tenha as mesmas dimensões da camada anterior, é comum colocar zeros em volta da entrada (o chamado zero *padding*). Esse processo é apresentado na Fig. 2.

Além das camadas convolucionais, as CNNs possuem camadas de agrupamento, ou de *pooling*. Seu objetivo principal é "encolher", ou seja, sub-amostrar a imagem de entrada para reduzir a carga computacional, o uso de memória e o número de parâmetros (evitando o risco de *overfitting*). Cada neurônio de uma camada de *pooling* é conectado com às saídas de um número limitado de neurônios da camada anterior, localizados dentro de um pequeno campo receptivo. A diferença de um neurônio dessa camada, é que ele não tem pesos, sua função

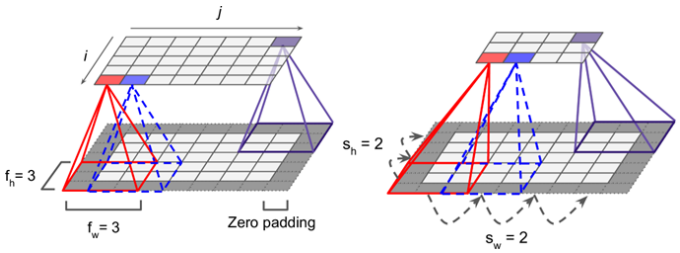

Fig. 2. Em ambos os exemplos, a imagem de entrada possui dimensão de 5 x 7 pixeis, zero *padding* e o campo receptivo local tem dimensão de 3 x 3 pixeis. No da esquerda, o campo está sendo movido 1 pixel por vez, resultando em uma dimensão de 5 x 7 neurônios na próxima camada. No da direita, ao mover o campo 2 pixeis por vez, a próxima terá 3 x 4 neurônios. Adaptada de [4].

é agregar as entradas usando uma função (média, máximo, mínimo, por exemplo).

Em uma arquitetura prática, são utilizados vários pares de camadas convolucionais e *pooling*. Pode existir vários *feature maps* em uma determinada camada convolucional para cada plano de unidades na camada de *pooling* anterior, de modo que a redução gradual na resolução espacial seja compensada por um número crescente de características. A camada final da rede normalmente é uma camada densa e adaptável.

III. Estudo de Caso

O exemplo a seguir (disponibilizado completamente nesse link) utiliza uma CNN pré-treinada VGG16 para detectar qual animal está em uma imagem. O modelo, oferecido pela biblioteca Keras [5], foi criado por pesquisadores na universidade de Oxford e consiste em uma arquitetura que apresentou ótimo rendimento na atividade de classificação do conjunto de dados ImageNet (composto por milhões de imagens que possuem objetos que pertencem a 1000 categorias). Sua arquitetura, apresentada na Fig. 3, espera como entrada uma imagem de dimensão 224x224x3, em seguida, existem 5 blocos de camadas convolucionais (com filtros de tamanho 3x3) empilhados sobre camadas de *pooling* (que possuem *stride* 2). Os dois primeiros blocos possuem duas camadas convolucionais (sendo que cada camada de cada bloco, respectivamente, aplicam 64 e 128 filtros, sobre a imagem e gera *features maps* de dimensão 224x224x64 e 112x112x128)

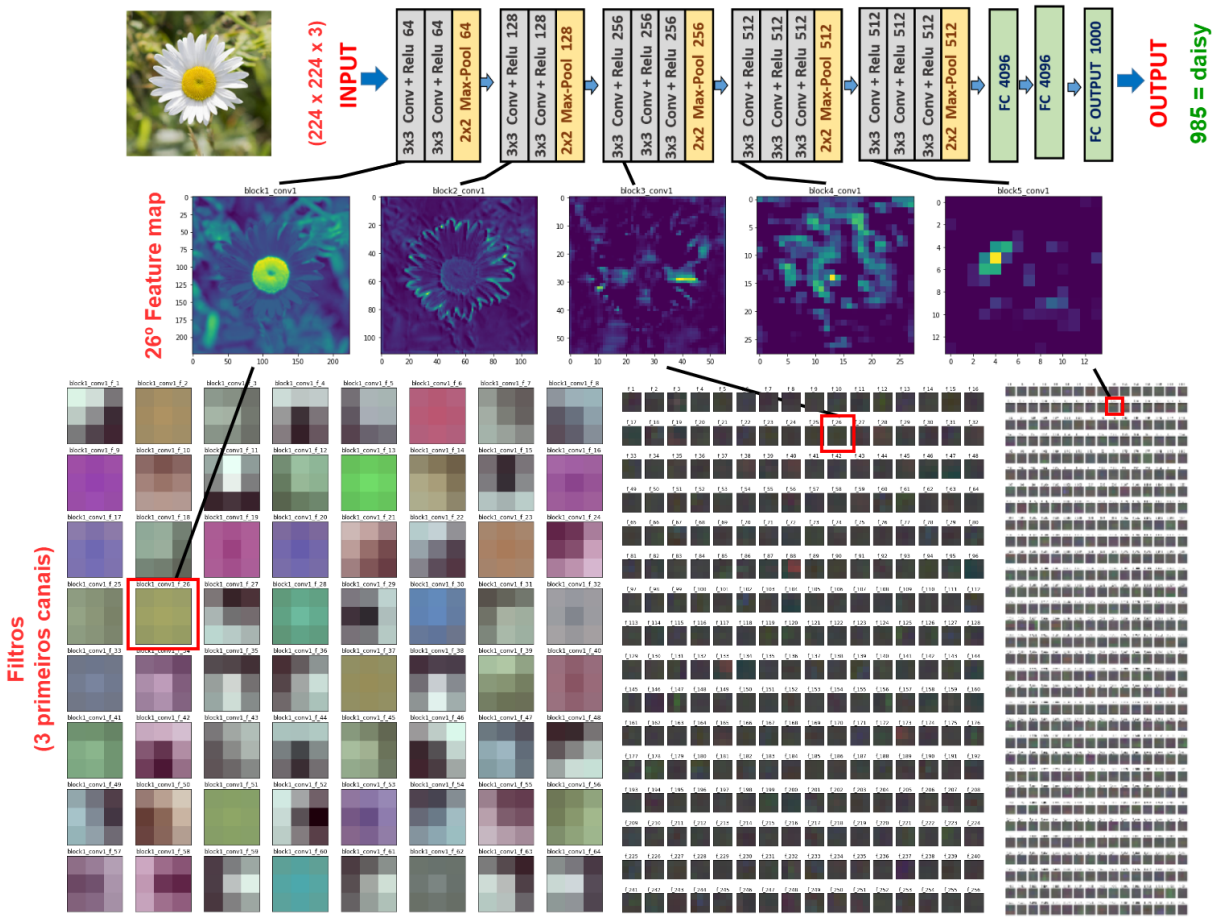

Fig. 3. *Feature maps* e filtros de algumas camadas convolucionais da VGG16. Produzida pelos autores.

e uma camada *max pooling* (gerando dados de dimensão 112x112x64 e 56x56x128, respectivamente). Os três próximos blocos são compostos por três camadas convolucionais (cada uma com 256, 512 e 512 filtros e gerando *feature maps* de tamanho 56x56x256, 28x28x512 e 14x14x512, respectivamente) e uma camada de *max pooling* (resultando em uma dimensão 28x28x256, 14x14x512 e 7x7x512). Após os blocos, o resultado é comprimido em uma dimensão menor (1x25088) a partir de uma camada *flatten*. Por fim, três camadas densas são colocadas ao final da arquitetura, sendo que as duas primeiras transformam o resultado em dimensão 1x4096 e última 1x1000, indicando o resultado da classificação.

Como o modelo já foi previamente treinado no conjunto de dados ImageNet, esse exemplo não realiza o treinamento da rede, utiliza os filtros/pesos já disponíveis. Ao escolher uma imagem para saber qual classe o objeto pertencente à ela faz parte, passamos a imagem pela rede e, ao final, a classe com maior probabilidade dos neurônios da última camada da rede é a classe correspondente. A Fig. 3 mostra alguns resultados intermediários desse processo.

Analisando o 26º *feature map* da primeira camada convolucional de cada bloco (esse índice foi escolhido aleatoriamente entre a quantidade em comum das camadas), observa-se que nos primeiros blocos existe uma maior retenção da informação, onde cores e contornos da imagem são mais evidentes. Ao passar pelas outras camadas, o *feature map* fica menos semelhante a imagem original, contendo menos informação sobre a imagem e mais sobre a classe correspondente (ou algo que irá facilitar esse processo). No que se refere aos filtros, ao passar da rede, sua profundidade aumenta de modo a detectar padrões mais complexos (nesse caso, apenas os 3 primeiros canais estão sendo exibidos). Ao final da rede, aplicando uma função *argmax* no resultado da previsão, a classe com maior probabilidade é informada e, nesse caso, houve um acerto da rede ao objeto pertencente a imagem (uma margarida).

REFERENCES

[1] C. Aggarwal, *Neural Networks and Deep Learning: A Textbook*. Springer International Publishing, 2018.

[2] S. Skansi, *Introduction to Deep Learning: From Logical Calculus to Artificial Intelligence*. Undergraduate Topics in Computer Science, Springer International Publishing, 2018.

[3] C. Bishop, *Pattern Recognition and Machine Learning*. Information Science and Statistics, Springer, 2006.

[4] A. Géron and a. O. M. C. Safari, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition*. O'Reilly Media, Incorporated, 2019.

[5] "Keras: the python deep learning api." https://keras.io/. Accessed: 2023-02-05.