Deep learning

Axel-Jan Rousseau,[a,b] Melvin Geubbelmans,[a,b] Tomasz Burzykowski,[a,b,c] and Dirk Valkenborg[a,b]
Hasselt, Belgium, and Bialystok, Poland

Over the past decade, deep learning (DL) methods have rapidly become the state of the art when it comes to automated classification and segmentation of biomedical images. In this article, we provide an overview of convolutional neural networks (CNNs), the most important type of model for image analysis.

Although ideas on and the theory of CNNs have been around for decades, CNNs started gaining popularity quickly from about 2012 onward. Because of fast CNN implementations on graphics processing units, a type of a parallel processor, computing power has now become available to train and use these models. CNNs started winning image classification contests by large margins. For example, the 2012 ImageNet challenge (a large challenging contest in which millions of images had to be classified into 1000 classes) was won by Alex Krizhevsky, whose CNN model achieved an accuracy of 83.6%, whereas the best model based on classic computer vision methods the year before had an accuracy of 74.3%. By 2015, the accuracy of the best model had increased to 96.4%. These days, DL has become ubiquitous in our lives and can be found in applications ranging from self-driving cars to silly selfie filters on our phones.

BUILDING BLOCKS OF CONVOLUTIONAL NETWORKS

Similar to other DLs discussed in a previous article of this series on ML,[1] CNN architectures consist of many simple layers, each performing nonlinear transformations from input to output. This hierarchy of layers learns progressively more complex and meaningful attributes, enabling the model to learn complex interactions in the data and model complex functions. For example, on the first layers, the model will learn simple attributes such as edges and colors; intermediate layers will combine these to recognize shapes; and the deepest layers will further combine this output to learn complex representations of the images.

In principle, a multilayer perceptron with many hidden layers, which was briefly discussed in a previous article of this series on ML,[1] is a DL model and could be applied to images. However, this approach does not scale well: for a 300 × 300 pixels color image, a single neuron in the first fully connected hidden layer will require 270,000 weights (300 × 300 pixels times 3 intensities of red, green, and blue), and a hidden layer typically consists of many neurons. A deep network built in this manner will use an enormous number of weights and will be computationally too expensive to learn and use. In addition, a model with such a large number of parameters will be prone to overfitting.[1]

A solution is to make use of convolutional layers, the main building block of CNNs. Unlike the fully connected layers in a multilayer perceptron that learn global patterns, the neurons in a convolutional layer make use of the 2-dimensional structure of images: pixels that are close together are related and belong together. Therefore, the neurons are only connected to a small area from the input (typically between 3 × 3 and 11 × 11 pixels), limiting it to learning local patterns and thus reducing the number of weights.

Another property of images is that objects or features are not in a fixed location. They can occur anywhere in the image. Therefore, the neurons are reused on each input location: attributes such as edges or textures learned in one location can be useful anywhere. As the Figure illustrates, the convolutional layer scans its input and produces a result at each location. This forms a 2-dimensional "activation map" showing where and by how much a neuron is activated and, thus, where the attributes were found. The next convolutional layer then uses these results as its starting point and can then combine the previous attributes and learn more complex ones. In this way, CNN learns a hierarchy of patterns of increasing complexity.

[a]Center for Statistics, Hasselt University, Hasselt, Belgium;
[b]Data Science Institute, Hasselt University, Hasselt, Belgium;
[c]Department of Biostatistics and Medical Informatics, Medical University of Bialystok, Bialystok, Poland.
This research received funding from the Flemish Government under the "Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen" program.
All authors have completed and submitted the ICMJE Form for Disclosure of Potential Conflicts of Interest, and none were reported.
Address correspondence to: Dirk Valkenborg, Data Science Institute, Hasselt University, Agoralaan 1, Bldg D, B-3590 Diepenbeek, Belgium; e-mail, dirk.valkenborg@uhasselt.be.
Am J Orthod Dentofacial Orthop 2024;165:369-71
0889-5406/$36.00

https://doi.org/10.1016/j.ajodo.2023.12.003

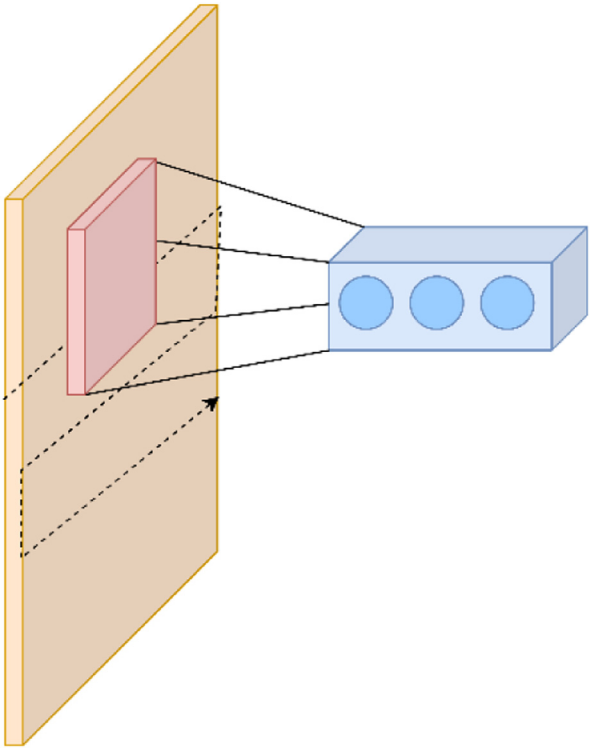

Fig. Illustration of locally connected neurons in a convolutional layer (*blue*). Neurons in the convolutional layer are locally connected to a patch of the input. This patch is moved over the input 1 pixel at a time, producing a result for each location.

Similar to the neurons in a perceptron, each layer of neurons in a CNN is followed by a nonlinear activation function. For CNN, the ReLu activation function is the most used, as it does not saturate as easily as the sigmoid or tanh functions.[1]

The result of a convolutional layer is roughly the same size as its inputs. This leads to several issues. First, even in deeper layers, the more complex attributes will only come from a small part of the original input image, whereas we would like to have a more global overview. Second, a reduction in size reduces the computations needed. Therefore, so-called pooling layers, which serve to scale down the activation maps of attributes, are placed between the convolutional layers. These merge attribution activations that are close together by dividing the intermediate result into a coarser grid and taking the average or maximum value for that grid cell.

Because all the neurons in the network are trained together, they can become too dependent on each other and co-adapt, causing the model to overfit. For example, assume that we want to detect vehicles and our dataset happens to contain many images of black cars. Although the data contain only images and no features such as color, number of wheels, etc, the model may learn a "black color" attribute, and subsequent neurons may learn that it is very important to the "car" class, as discussed in a previous article in this series.[2] To prevent this co-adaptation, the dropout layer method is used.[3] This layer randomly drops connections from neurons during training, removing the attribute that that neuron represents. If the "black" neuron is randomly missing, the network must learn to also rely on other attributes to detect cars.

ARCHITECTURES

The architecture of a CNN refers to the way the building blocks and layers are combined to create the model. Over the years, the way these are connected has become increasingly complex as new generations of architecture build on ideas of the previous generation. AlexNet,[4] which won the 2012 ImageNet challenge, was a straightforward architecture with 8 convolutional layers that learned and extracted the features and patterns from the images, followed by 3 fully connected layers to make the classifications.

A natural question to ask is, how important is the "deep" in DL? The Visual Geometry Group architecture[5] showed that substantial improvements could be made by pushing the depth to 19 layers. It uses a setup similar to AlexNet, but applies convolutional layers that use a smaller area. This reduces the computational complexity, allowing this architecture to be much deeper. GoogLeNet[6] and its subsequent improvement Inceptionv2[7] make use of a network-in-network approach. A building block called the inception module, which contains multiple convolutional layers in parallel with different area sizes that are applied to the same input, is used and thus learns patterns at different scales.

Even when computational power is not a problem, going deeper with CNNs encounters other obstacles. One of them is related to the fact that the error signal that is propagated back into the model to update the weights becomes smaller and smaller for the bottom layers at the start of the model, making learning for these layers very slow or even impossible. This is called the vanishing gradients problem. The winning architecture for the 2015 ImageNet challenge, ResNet,[8] came up with a solution to this issue. Namely, they introduced skip connections, which are connections that bypass a

layer and allow a path for the error signal to propagate back to the bottom layers more easily.

Finding a good architecture for a given task can be a huge undertaking. Luckily, one does not need to reinvent the wheel. Typically, architectures that perform well on ImageNet will give decent results on other image classification tasks. On top of that, "of-the-shelf" architectures often have trained models available so that we can use a technique called transfer learning. The idea is that a CNN trained on a large, diverse dataset such as ImageNet will learn to recognize many different edges, gradients, shapes, textures, etc. Chances are that some of these attributes will also be relevant when learning another task, or at least be a better starting point than a randomly initialized model. Therefore, ResNet (and variants thereof) is still a widely used architecture.

TRAINING

The basics of training a DL model were described in a previous article in this series.[1] However, DL practitioners employ a large toolbox of tricks to train models.[9] Because the datasets used are often too large to fit into a computer's working memory, models are trained on batches of data. The choice of batch size has a large influence on training. Large batch sizes lead to better estimates of the error, allowing for larger updates of the weights and faster convergence. At the later stages of training, however, small updates may be preferred to make finer adjustments to the weights. Therefore, learning rate schedules are used to gradually reduce the learning rate over time.

Training a CNN requires lots of data, which are often not available. To increase the availability of data and prevent overfitting, extra synthetic data can be created, called data augmentation.[10] New versions of an image can be created by using transformations such as zooming, rotating, shearing, cropping, elastic deformation, and color and intensity changes. These transformations are randomly applied during training so that no image is the same as before, preventing the network from memorizing the training samples.

CONCLUSIONS

CNNs have revolutionized the field of computer vision. They use stacks of many simple layers that make use of the properties of images to efficiently compute and learn large amounts of image attributes of varying complexity. Although the layers may be simple, the architectures used are complex and are in continuous development to improve on the state of the art.

REFERENCES

1. Geubbelmans M, Rousseau A-J, Burzykowski T, Valkenborg D. Artificial neural networks and deep learning. Am J Orthod Dentofacial Orthop 2024;165:248-51.
2. Valkenborg D, Geubbelmans M, Rousseau AJ, Burzykowski T. Supervised learning. Am J Orthod Dentofacial Orthop 2023;164:146-9.
3. Srivastava N, Hinton G, Krizhevsky A, Salakhutdinov R. Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research 2014;15:1929-58.
4. Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. Adv Neural Inf Process Syst 2012;25.
5. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 2014.
6. Szegedy C, Liu Wei, Jia Yangqing, Sermanet P, Reed S, Anguelov D, et al. Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2015; p. 1-9.
7. Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z. Rethinking the inception architecture for computer vision. 2016 IEEE Conference on Computer Vision and Pattern Recognition 2016;2818-26.
8. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2016; p. 770-8.
9. He T, Zhang Z, Zhang H, Zhang Z, Xie J, Li M. Bag of tricks for image classification with convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2019; p. 558-67.
10. Shorten C, Khoshgoftaar TM. A survey on image data augmentation for deep learning. J Big Data 2019;6:60.