CHAPTER

10

Convolutional neural networks

Walter Hugo Lopez Pinaya[1,2], *Sandra Vieira*[1], *Rafael Garcia-Dias*[1], *Andrea Mechelli*[1]

[1] Department of Psychosis Studies, Institute of Psychiatry, Psychology & Neuroscience, King's College London, London, United Kingdom; [2] Centre of Mathematics, Computation, and Cognition, Universidade Federal do ABC, Santo André, São Paulo, Brazil

10.1 Introduction

Since the beginning of deep learning, few ideas have generated as much impact as convolutional neural networks (CNNs or ConvNets; Fig. 10.1). Inspired by the experiments of David Hubel and Torsten Wiesel on the visual cortex (Hubel & Wiesel, 1962), these networks were introduced in the 1990s, where the first models were developed to recognize handwritten digits and showed promising performance (LeCun, Bottou, Bengio, & Haffner, 1998). However, the great potential of these networks only captured the attention of the wider research community in 2012, during the "ImageNet Large Scale Visual Recognition Challenge" event (also referred to as "ImageNet competition") (Russakovsky et al., 2015). In that year, Alex Krizhevsky from the University of Toronto pioneered the application of this type of network to a complex challenge involving the classification of 1.2 million high-resolution images across 1000 categories (e.g., bee, toucan, beagle, etc.). The CNN model, called "AlexNet" (Krizhevsky, Sutskever, & Hinton, 2012), achieved first place with an error rate of 16.4%. The second best entry performed significantly worse with an error rate of 26.2%. With this result, AlexNet single-handedly put deep learning on the map and led to a paradigm shift in the field of computer vision. Since then, CNN research has grown exponentially. In just 3 years, researchers progressed from the 8-layer AlexNet to the 152-layer "residual networks" (He, Zhang, Ren, & Sun, 2016), dropping the error rate of the ImageNet challenge to less than 4%, reflecting the increasing sophistication of CNN structures.

https://doi.org/10.1016/B978-0-12-815739-8.00010-9

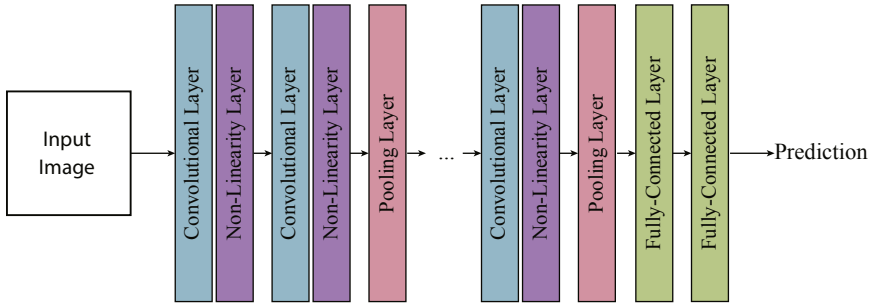

FIGURE 10.1 Example of the structure of a convolutional neural network (CNN). A CNN is composed of different types of layers, such as convolutional layers, nonlinearity layers, pooling layers, and fully connected layers. These layers can be stacked in several ways, some examples are shown later in this chapter.

In addition to showing lower error rates in handwritten digit recognition and object classification, CNNs have excelled in others computer vision tasks, such as human action recognition (Ji, Xu, Yang, & Yu, 2013), object localization (He, Zhang, Ren, & Sun, 2015; Ren, He, Girshick, & Sun, 2015), detecting pedestrians (Angelova, Krizhevsky, Vanhoucke, Ogale, & Ferguson, 2015), and face recognition (Schroff, Kalenichenko, & Philbin, 2015). CNNs have subsequently shown to be effective for natural language and speech processing, achieving excellent results in sentence classification (Kim, 2014), sentence modeling (Kalchbrenner, Grefenstette, & Blunsom, 2014), and speech recognition (Abdel-Hamid et al., 2014). These deep neural networks were also successfully applied to reinforcement learning, hitting the headlines when the AlphaGo model was able to beat the best players in the world at the ancient Chinese game of Go (Silver et al., 2016). More recently, there has been a surge in the application of CNNs in the medical field, including, for example, pneumonia detection (Rajpurkar et al., 2017), retinal disease identification (De Fauw et al., 2018), and classification of skin cancer (Esteva et al., 2017). Today, the successful application of CNNs across a wide range of applications is one of the key reasons for the increasing popularity of deep learning.

The secret of the success of the CNN lies on a carefully designed architecture with a semantic understanding of the domain at hand. In other words, this type of network was created to take into account the particular characteristics that we find in the elements/patterns which constitute the input data. CNNs were initially designed to be applied to images. Therefore, they encode the following key properties into the architecture:

- *Local connectivity:* Images are mainly composed of spatially located elements (e.g., objects, animals, textures, pieces of objects, etc.).

Therefore, the neurons of a network do not need to connect to all the units of the input to find interesting patterns. Instead, the neurons of a CNN are only connected to a small number of units in a spatially localized region of the input—called receptive field. This allows neurons to focus on local features rather than global features.

- *Spatial invariance:* Images can have elements in different positions without altering its semantic content, for example, a picture of a cat will still be a picture of a cat regardless of where the cat is in the image. So, the appropriate neural network to deal with images needs to learn to be translation invariant. In other words, the network needs to produce similar output values from similar input patterns regardless of their location. CNNs implement this property by sharing parameters among different neurons.
- *Hierarchical features:* Patterns in images can usually be decomposed in a hierarchy of features, with low-level features (e.g., ears, eyes, nose) that can be grouped to create high-level features (e.g., faces, people). By using multiple layers, CNN can automatically extract and learn this hierarchy of features for pattern recognition.

When we combine all of these properties, the resulting neural network is highly efficient to implement, making it suitable for dealing with complex images.

In the following sections, we provide a comprehensive introduction to CNNs. We then present the essential concepts behind the different types of layers and how to connect these layers to create popular CNN architectures. Finally, we describe and discuss exemplary applications of CNN to brain disorders.

10.2 Method description

CNNs are similar to traditional deep neural networks (DNNs) presented in Chapter 9: they are made up of layers of neurons that have learnable weights and biases. Each neuron receives some inputs, calculates a dot product, and optionally follows it with a nonlinear function. This process is repeated layer by layer until the output layer, where the network's prediction is generated. However, DNNs have an important limitation: they do not scale well in terms of computational resources. As seen in Chapter 9, traditional neural networks use fully connected layers, where each neuron is connected to every neuron in the previous layer. This can easily lead to a network with thousands of parameters to be estimated. This type of configuration is for general purpose, i.e., it makes no assumptions about the properties of the input variables. For this reason, it tends to be expensive in terms of memory and computation.

For example, let us consider a particular kind of data that DNNs do not scale well to: natural images. Images can be represented as a two-dimensional (2D) grid structure. Each grid point corresponds to a pixel. In the RGB color scheme, each pixel has three values of intensity, each corresponding to a primary color: red, green, and blue. If the image has a spatial dimension of 100×100 pixels, then the overall number of variables is $100 \times 100 \times 3 = 30{,}000$. In this case, a single fully connected neuron would have 30,000 weighted connections. This amount might still be manageable, but it does not scale well if we want to add more neurons. And in deep learning, we want to have a lot of them over multiple layers. As our network begins to grow, the full connectivity starts to become wasteful, and the massive number of parameters can quickly lead to overfitting. This limitation is particularly critical in brain disorders research, where data tend to have a very high dimensionality (e.g., genetic or neuroimaging data).

So, what is different about CNNs? These networks are specifically designed to work with grid-structured inputs that have strong spatial dependencies in local regions of the grid. The most obvious example is the natural image. Natural images exhibit spatial dependencies where adjacent pixels often have similar color values. Therefore, the features within an image have dependencies among one another based on spatial distances. Besides 2D images, there are other examples of similar grid-structured data, such as

- Signals and sequences, including sequence of words (one-dimensional or 1D data)
- Audio spectrograms (2D data)
- Videos and volumetric images, such as computed tomography and magnetic resonance imaging (MRI) data (three-dimensional or 3D data)
- Volumetric images over time, such as functional MRI data (four-dimensional or 4D data)

10.2.1 Convolutional neural network layers

A CNN can be composed of several building blocks. In the next sections, we present four of them: convolutional layer, nonlinearity layer, pooling layer, and fully connected layer. For simplicity, we consider the case where the layers are processing 2D input data.

10.2.1.1 Convolutional layer

The essential component of a CNN is the convolutional layer which does most of the computational heavy lifting. In a few words:

The convolutional layer contains a set of *filters*, and its function is to perform a *convolution operation* between these filters and the input of the layer to create *feature maps*.

Let us analyze this sentence. First, a *filter* is a grid of discrete numbers and, usually, is square-shaped. Its parameters (i.e., the numbers in the grid) mainly store a single template/pattern. This pattern is what the filter will detect in the layer's input (like a feature detector). For example, a filter made of discrete numbers that form a "cat pattern" will be able to detect the presence of a cat in the input.

The *convolution operation* is the key step that allows CNNs to be spatial invariant. The convolution operation will position the filter over the left upper section of the image. It will perform an element-wise product between the filter's parameters and the matching grid in the input, followed by summing the result to get a single value; in other words, it will perform a dot product (Fig. 10.2). The resulting single value indicates the presence or absence of the filter's single template/pattern in this specific section of the image (for example, the presence or absence of a cat in our earlier example). Next, the convolution operation will slide the filter to the right and calculate the dot product in this new position. This filter sliding is implemented from left to right and top to bottom across the input and allows the application of the filter at each position of the image (Fig. 10.3). Mathematically, the *combination of filter sliding and dot products* can be defined as a convolution operation.

Finally, the *feature map* stores the result of the convolution operation (the results of many dot products). This storage is done in a spatial grid structure, which maintains the spatial relationships among the inputted grid. This property of the feature map is essential because the convolution operations of the next layers are critically dependent on these spatial relationships.

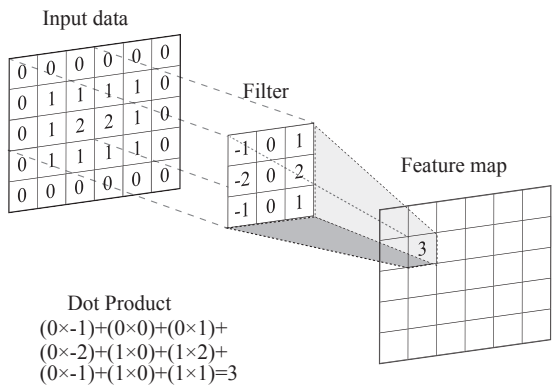

FIGURE 10.2 Calculation of a single dot product. The calculation of the dot product involves an element-wise multiplication between the convolutional filter and the matching grid in the input data. The resulting values are summed obtaining a single number that is stored in the central pixel in the feature map.

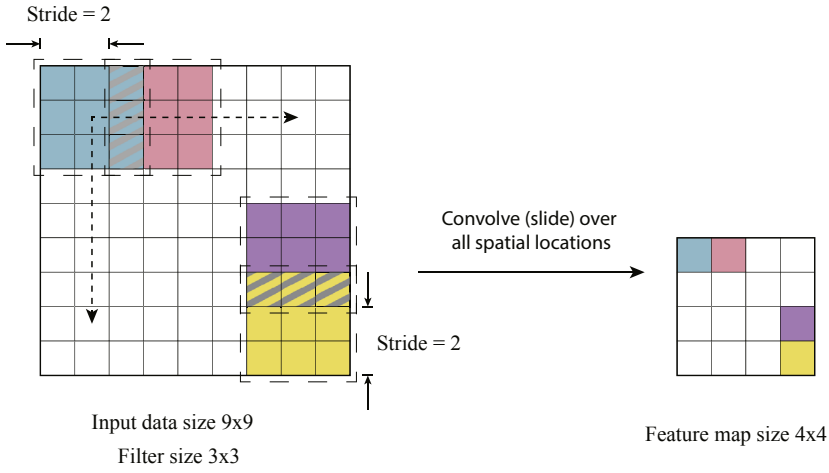

FIGURE 10.3 The convolution operation. We start with the filter in the top left corner of the image and calculate the dot product between this filter and the input data; the resulting single value is added to the corresponding position in the feature map. Then, we slide the filter one position to the right (each position is shown with a different color) and perform the same operation; again, the resulting single value is added to the corresponding position in the feature map. And so, successively, we go through the entire space of the input layer, from left to right and top to bottom.

However, the above definition of the convolutional layer does not make any reference to the neural network's neurons (Chapter 9). How do neurons relate to the notions of filter, convolution operation, and feature map?

We can imagine that each element in the feature map is associated with a neuron that looks at only a small window of the input (i.e., receptive field). The neuron weights are analogous to the filter parameters, and the value of the feature map is the result of its dot product with the inputs. Every neuron associated with an element in a feature map is connected to a fixed region of the previous layer in such a way that the whole input is covered ("local connectivity"). Within the same feature map, each neuron shares the same weights values, based on the assumption that a feature is equally likely to occur at every input position ("spatial invariance"). As a result of this design, the whole image is scanned for the same pattern.

As we saw earlier, each filter stores only one pattern. Scanning the image for only one pattern would, therefore, be likely to result in a very limited network. To address this limitation, a convolutional layer needs to have several filters, each one producing a single 2D feature map. After obtaining the different feature maps, we stack all of them together and that becomes the final output of the convolutional layer—a 3D volume with all feature maps (Fig. 10.4).

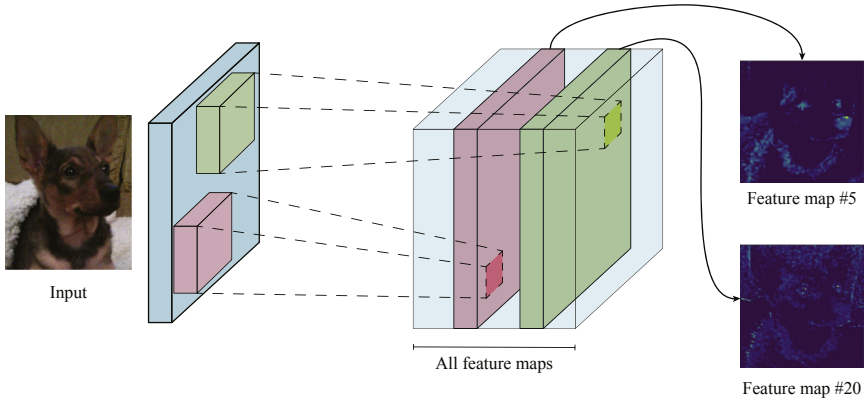

FIGURE 10.4 Stacked feature maps. In this example, the convolution operation is applied to the input image multiple times using different filters (green and red) (gray and light gray in print version). Each filter creates a different feature map that detects different patterns on the image. The resulting feature maps are then stacked together to form the layer's output.

Similar to a DNN, the parameters of each filter (the shared neuron's weights) are learned during the training phase. This learning procedure involves a random initialization of the filter parameters at the start, which are then tuned in many iterations using a gradient descent method (more details about neural networks training in Chapter 9).

So far, we considered the input data of the layer as a single 2D matrix. This input structure would be able to represent a grayscale image. However, images usually do not have a single channel color. In RGB images, data are described as three 2D matrices. Also, if the layer's input is the feature maps from the previous layer, it will have many 2D matrices, one for each feature map. In these cases, the filter will process the whole number of matrices at the same time. They will operate on the entire volume of feature maps (or color channels), having a 3D format. For example, when trying to classify the picture of a face, we might use features such as the nose, eyes, ears, and mouth. These features would be represented in a specific layer of the network in four feature maps, one for eyes, one for noses, one for mouths, and one for ears. We know that a particular location contains a face if the corresponding positions in the inputted feature maps include the appropriate features (two eyes, a nose, a mouth, and two ears). In other words, to make decisions about the existence of a face, the convolutional layer must combine evidence over multiple feature maps with a single filter for searching for the presence of all of these elements.

Due to these characteristics, the convolutional layer has different hyperparameters compared with the layers of DNN. These are the filter count, filter size, stride, and padding.

10.2.1.1.1 Filter count

The number of filters determines the number of feature detectors. This hyperparameter is the most variable one among layers and, usually, is set to a power of 2, between 32 and 512. Using more filters results in a more powerful neural network, but increases the risk overfitting because of the higher number of parameters to estimate.

10.2.1.1.2 Filter size

The size (height and width) of the filter defines its spatial extent. We typically use small filters with 3×3 grids, but 5×5 or 7×7 is also used. The use of small filters provides two key benefits: (i) the number of learnable parameters is significantly reduced; and (ii) it ensures that distinctive patterns are learned from local regions. Note that when we stack many convolutional layers on top of each other, the "effective receptive field" of each layer becomes a function of the receptive fields of all the previous convolutional layers. For example, if we stack two convolutional layers, each of filter size 3×3 and 3×3, the receptive field of the second layer would be 3×3, but its effective receptive field to the input image would be 5×5. A natural consequence is that features in later layers capture larger spatial regions.

10.2.1.1.3 Stride size

The stride size indicates the number of pixels in which the filter window moves. Its value is usually 1, meaning that the filter slides pixel by pixel. However, we can increase the stride size if we want the filter to slide with a larger interval resulting in less overlap between the receptive fields. This alteration makes the resulting feature map smaller (Fig. 10.5).

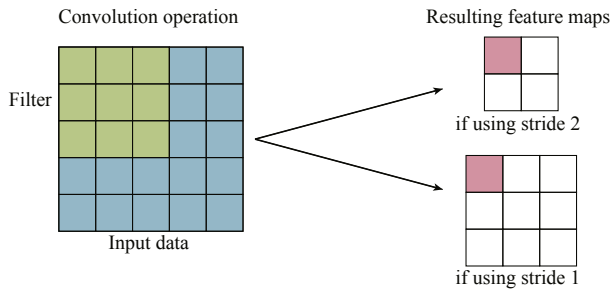

FIGURE 10.5 Impact of stride size on the feature map. The stride size refers to the number of pixels in which the filter window moves. When it increases, the filter slides with a larger interval resulting in less overlap between the receptive fields and smaller feature map.

10.2.1.1.4 Padding

Sometimes it is beneficial to pad the input data with zero-value pixels around the border. This pad prevents our feature map from shrinking during the convolution operation because the central pixel of the filter can now be positioned in the border pixel of the input image (Fig. 10.6). This avoids a collapse of the output feature dimensions, thereby allowing us to design deeper networks.

10.2.1.2 Nonlinearity layer

Similar to other neural networks, CNNs involve the use of a nonlinear activation function after the computation of the convolutional layer's operations. Usually, this nonlinear function is defined inside the convolutional layer. However, sometimes nonlinear transformations are implemented as an independent layer to allow more flexibility in the network architecture. Among the possible nonlinearities, the ReLU function is the most popular. The use of this function allows us to train deep CNNs much faster than using other activation functions, such as tanh or sigmoid. The reason is that the tanh and sigmoid functions saturate at very high or very low values, making the gradient of the function very close to zero, which slows down the gradient descent optimization. On the other hand, the ReLU function's gradient is not close to zero for any positive values, helping the optimization to converge faster.

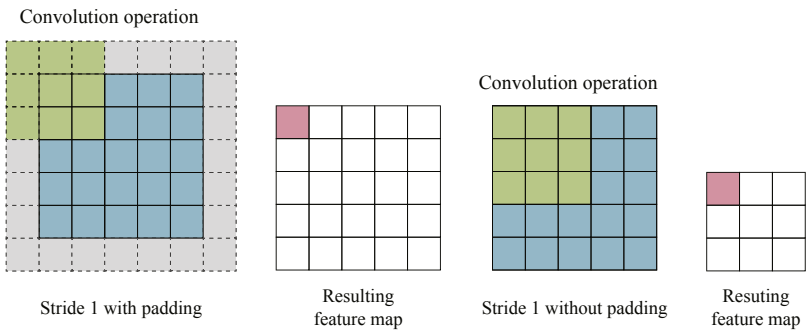

FIGURE 10.6 The effect of using padding (gray area) around the layer's input. (Left): Padding the input data with zero-value pixels around the border allows the central pixel of the filter to be positioned in the border pixel of the input image; this prevents the feature map from shrinking during the convolution operation. (Right): Without padding the input data with zero-value pixels around the border, the central pixel of the filter is no longer positioned in the border pixel of the input image; this causes the feature map to shrink during the convolution operation.

10.2.1.3 Pooling layer

The purpose of the pooling layer is to reduce the spatial size of the representation captured by the convolutional layer. It mainly simplifies the information collected and creates a condensed version of the same information. The most common form of pooling is the max *pooling* (Fig. 10.7). The max pooling layer slides a window over its input and takes the max value in the window, discarding all other values. Similar to a convolutional layer, we specify the window size (analogous to the filter size) and stride.

10.2.1.4 Fully connected layer

Usually, the last layers of a CNN are fully connected layers. These layers are the same to the ones in a DNN. Their main function is to perform classification on the features detected and extracted by the series of convolutional layers and pooling layers. To be inputted in the fully connected layers, the features maps are flattened into a single 1D vector.

10.2.2 Convolutional neural network architecture

Our next question is how do we combine the different layers to make a functional CNN? As mentioned in Chapter 9, neural networks do not have a well-defined process to determine the ideal structure; instead, this requires a significant amount of experimentation. However, a good starting point is to use the principles that have been shown useful in the same or other research fields, within the existing literature. In this section, we illustrate how to combine the different layers using a widespread neural network as an example: the Visual Geometry Group (VGG) model.

The VGG model is a network that can be applied to image recognition. This network was a runner-up in the ImageNet classification challenge

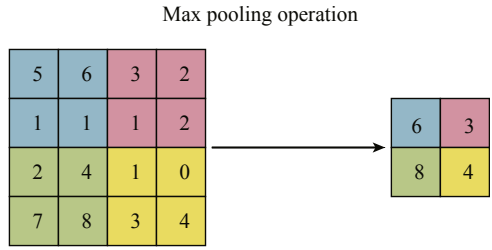

FIGURE 10.7 The max pooling layer downsamples the feature map while keeping the important information. Each color denotes a different filter position. The idea behind max pooling is that it can be more informative to look at the maximal presence of a pattern than its average presence. In this example, we are using a max pooling with 2×2 window and stride 2.

from researchers at Oxford's Visual Geometry Group, hence the name. One of its variants has 16 layers and is called VGG16. Usually, we start the development of our network with a VGG-based architecture because of its remarkable simplicity (Fig. 10.8).

The VGG can be separated into two parts. The *feature extraction part* of the model comprises of pairs of convolutional and ReLU layers, followed by a single pooling layer. This pattern is repeated many times, thus reducing the spatial dimensions of the data. In this pipeline, several filters are applied along the network, extracting the presence of features with different levels of abstraction ("hierarchical features"). The *classification part* of the model contains the most complex features. Here there is a transition to fully connected layers which learn global patterns in the total input space. The global information present in the last convolutional layer is finally used to perform the classification per se. Usually, we begin by building a small network comprising of just few layers. If the training process does not show signs of overfitting, we start to increase the number of layers and filters: the more convolutional layers we have, the more complex features our neural network will learn to identify.

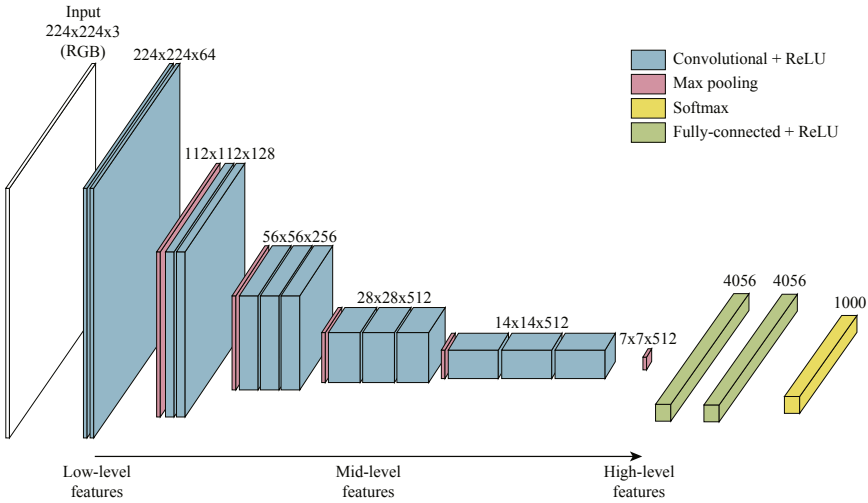

FIGURE 10.8 A schematic of the VGG16 model. This model only uses convolutional filters with a size of 3×3 throughout the feature extraction part. The text over each convolutional layer indicates the width of feature map, the height of feature map, and the number of filters. One characteristic of the network is that the number of filters increases (horizontal dimension) when we reduce the spatial dimension of the feature maps. The text over the fully connected layers indicates the number of neurons. An output layer with 1000 neurons with softmax nonlinearity computes the probability of the input image belonging to each of the 1000 categories included in the ImageNet competition.

10.3 Applications to brain disorders

The success of CNNs has inspired medical researchers to investigate the potential of CNNs using several types of medical data. The first applications of CNNs in brain disorders research focused on the analyses of neuroimaging data such as structural MRI (Gupta, Ayhan, & Maida, 2013; Payan & Montana, 2015; Zikic, Ioannou, Brown, & Criminisi, 2014). These applications were the natural extension of existing approaches for processing images that had been used successfully in the field of computer vision. Over the years, the applications of CNNs in brain disorders have become more and more sophisticated; this is underlined, for example, by the use of models with an increasing number of layers and models that use structures where layers are not only linearly arranged one after the other but also divide into different branches and merge later. More recently, applications of CNNs in brain disorders research have extended to other data modalities, such as electroencephalogram (EEG) (Acharya, Oh, Hagiwara, Tan, & Adeli, 2018; Truong et al., 2018), speech (He & Cao, 2018), and facial appearance (Zhu, Shang, Shao, & Guo, 2018). In this section, we present some of these applications.

10.3.1 Prediction of Alzheimer's disease

Alzheimer's disease (AD) is the most common form of dementia and is symptomatically characterized by impairments in memory, thinking, and behavior (Rathore, Habes, Iftikhar, Shacklett, & Davatzikos, 2017). Long before the manifestation of these symptoms, however, microscopic changes related to cell death take place (Frisoni, Fox, Jack Jr, Scheltens, & Thompson, 2010). Radiologically, this neurodegeneration is the hallmark of AD, starting in the temporal lobe and then spreading across the brain. However, because normal aging is also characterized by brain atrophy, distinguishing normal age-related atrophy from AD-mediated atrophy can be a difficult task, even for experienced radiologists.

Over the past few years, more and more machine learning models have been used to distinguish between healthy subjects and AD patients. This surge has been motivated by the increasing interest in developing artificial intelligence–based decision support systems that could help diagnose and treat people with this illness (Rathore et al., 2017). The majority of studies have used CNNs to identify patients with AD based on structural neuroimaging brain data. In one of the earliest studies, for example, Payan and Montana (2015) employed this type of network to classify subjects into three classes: healthy controls, mild cognitive impairment (the prodromal phase of AD), and AD. The authors used T1-weighted MRI images of 2265 subjects. After preprocessing, each

image resulted in a final 3D matrix with dimension $68 \times 95 \times 79$ (consisting of over half a million voxels). The CNN used in this study had a simple structure. It was composed of a convolutional layer, a max pool layer, a fully connected layer, and an output layer (also fully connected). However, the training process was more complex. The authors did not use randomly initialized convolutional filters (recall that the weights of the network are typically set to a random value at the beginning of training—see Chapter 9). Instead, they took a two-stage approach. First, they pretrained the convolutional filters using an unsupervised convolutional autoencoder (see Chapter 11). Second, they initialized the convolutional filters with the pretrained parameters and then trained the CNN to classify the subject between the categories.

The authors were particularly interested in comparing the performance of 2D and 3D CNNs to predict AD. Results showed that the network with 3D inputs performed better (89% accuracy) compared to the network with 2D inputs. The authors concluded that local 3D patterns in the image were more useful for their machine learning task than the local 2D patterns, a finding that has been replicated in subsequent medical imaging studies (de Brebisson & Montana, 2015; Deniz et al., 2018).

Over the years, CNNs applied to neuroimaging data have become more complex. For example, Khvostikov, Aderghal, Benois-Pineau, Krylov, and Catheline (2018) developed a CNN that incorporated multimodal data, an increasingly popular approach for AD classification. Using structural MRI and diffusion tension imaging data from the hippocampus, the authors compared the performance of each single modality against the performance of the two combined modalities. In their approach, they used a structure called "siamese network," with many input branches which eventually merged to enable the fusion of data from different modalities (Fig. 10.9).

The multimodal approach achieved the best performance, with an accuracy of over 95%, in the classification between healthy controls and AD groups. Despite the small dataset of 48 AD patients and 58 healthy controls, this study showed the promising potential of applying neural networks to 3D neuroimaging data and fusing different modalities.

10.3.2 Brain tumor segmentation

Structural MRI is one of the most common tools to assess brain tumors. Using this technique, the tumor can be segmented for the purpose of diagnostic and prognostic evaluation as well as to measure growth rate. However, the development of automated techniques for tumor segmentation is technically challenging. Brain tumors can appear anywhere in the brain in almost any shape and size. This high variability prohibits the use

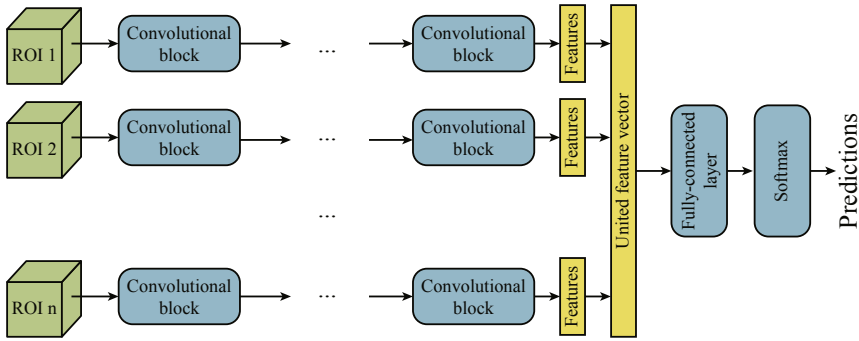

FIGURE 10.9 The multimodal architecture from Khvostikov et al. (2018). Each inputted region of interest (ROI) corresponds to the hippocampus, which was measured in the left and right hemispheres using two modalities (structural magnetic resonance imaging and diffusion tension imaging). *Adapted from Khvostikov, A., Aderghal, K., Benois-Pineau, J., Krylov, A., & Catheline, G. (2018). 3D CNN-based classification using sMRI and MD-DTI images for Alzheimer disease studies.* ArXiv *Preprint ArXiv:1801.05968.*

of strong priors on shape and location that are essential components in the segmentation of other anatomical structures.

To evaluate the state of the art in this segmentation task, Menze et al. (2015) have organized the Multimodal Brain Tumor Image Segmentation Benchmark (known as BRATS) challenge. This initiative utilizes preoperative MRI scans and focuses on the segmentation of heterogeneous brain tumors, namely gliomas. Their dataset is updated yearly and includes over 450 scans of low- and high-grade glioma patients, with repeated manual tumor delineations by several human experts. The BRATS challenge also includes the prediction of the patient's overall survival to explore the potential clinical relevance of this automated task.

Many methods using CNNs have been proposed in the BRATS challenge. For example, Pereira, Pinto, Alves, and Silva (2016) used 2D axial slices of the structural MRI with different contrasts as channels. This CNN used as input a patch of 33×33 pixels extracted from the MRI slices. Then, it predicted the category of the central pixel among five possible kinds of tissues: normal tissue, necrosis, edema, nonenhancing, and enhancing tumor. This approach showed a sensitivity of 86% on BRATS (2013 edition test set).

In Havaei et al. (2017), the authors explored the CNN's flexibility. First, their model used patches as inputs, but the network had two different input layers with different scales. The first input had a patch size of 65×65, which was processed by convolutional and pooling layers resulting in 33×33 features maps that were then concatenated with the second input, a 33×33 patch of the same central pixel. Second, their network did not have the typical linear structure of the network, where

the different layers are arranged one after the other in one single branch; instead, the layers were split into branches and then merged. This approach revealed a sensitivity of 84% on BRATS (2013 edition test set).

In Kamnitsas et al. (2017), the authors focused on developing a more reliable system by bringing together seven different CNN models. They configured and trained these models in diverse ways to introduce high variance between them. They employed three different kinds of CNNs: two multiscale 3D networks, three fully convolutional networks (Long, Shelhamer, & Darrell, 2015), and two known as U-Net (Ronneberger, Fischer, & Brox, 2015). All these architectures were created especially for the segmentation tasks. The outputs from all architectures were then combined into an ensemble of models. Using an ensemble approach, their system was robust to unpredictable failures of independent components. Their method won the first position on the BRATS challenge that year, with a sensitivity of 78% (2017 edition validation set), showing how the flexibility of CNNs and variance in an ensemble of techniques can be a promising approach for effective tumor segmentation.

10.3.3 Seizure detection

As mentioned earlier, CNNs can be applied to biosignals with the format of 1D time series. To achieve this, it is necessary to remove one of the dimensions of the 2D convolutional filters. With this approach, we can make the CNN invariant along the time dimension.

One useful application of 1D CNNs is the detection of epileptic seizures. Seizures occur due to a sudden change in the brain's activity which results in alteration of awareness often associated with abnormal involuntary movements. Epilepsy is characterized by having two or more unprovoked seizures and affects nearly 50 million people worldwide. Monitoring of brain activity through EEG is the standard technique for the confirmation of the diagnosis. Experts analyze the EEG via direct visual inspection to check for the presence of epileptiform abnormalities that may provide valuable information on the type and etiology of a patient's epilepsy. However, this visual analysis is expensive: analyzing the recordings can be very time-consuming, and it requires scarce highly trained professionals. These limitations have motivated the development of automated approaches to seizure detection.

In one of the first CNN's studies applied to seizure detection, Acharya et al. (2018) used CNNs for the automated detection of three EEG classes: normal, preictal, and seizure. Their dataset was composed of 100 EEG signals per subject, acquired from 5 healthy control and 5 patients. The data of the normal class came from the healthy controls, the preictal category was defined by the signal of epileptic patients when they did not

undergo seizure, and the seizure signal came from the same patients when they were having an epileptic seizure.

Their CNN architecture followed the VGG network's structure and had 13 layers. They used blocks of convolutional layers followed by pooling layers, reducing the spatial dimensions and increasing the number of filters along the network. Finally, fully connected layers were used to generate the output.

The network correctly classified 90% of the normal EEG signals, 88% of preictal signals, and 88% of the seizure class. Even with results below the performance of other state-of-art methods at the time (Bhattacharyya, Pachori, Upadhyay, & Acharya, 2017; Sharma, Pachori, & Acharya, 2017), this approach had the advantage of removing the separate steps of feature extraction and feature selection, both challenging and time-consuming steps in the traditional development of an automated seizure detector.

10.4 Conclusion

The development of CNNs has led to promising findings in the areas of psychiatry and neurology, by enabling the discovery of useful patterns. However, there is still room for improvement. CNNs were a significant breakthrough in computer vision in 2012. This breakthrough was made possible not only by the development of new algorithms and the availability of greater computational power but also by concerted efforts to build large publicly available datasets which could be used to train and test models. At present, despite the increasing number of approaches being developed to improve performance with small datasets (such as transfer learning, pretraining, and one-shot learning), the limited size of existing datasets remains a key challenge for the application of CNNs in psychiatry and neurology. The number of training samples is essential to building complex and powerful models without suffering from overfitting or sample-biased results. This problem is exacerbated in the context of high dimensional data (a problem known as "curse of dimensionality" in the machine learning field). However, most applications in brain disorders have far fewer images (i.e., <1000) compared to other areas of research such as computer vision. Thankfully, there is now an increased awareness of the importance of large open datasets, and a growing number of initiatives are being developed to address this. Large consortiums such as the Alzheimer's Disease Neuroimaging Initiative (Petersen et al., 2010) and BRATS (Menze et al., 2015) were initially developed to facilitate reproducible findings and also brought benefits to machine learning research. For example, the increasing availability of multicenter datasets means it is easier to use independent derivation and validation samples to develop models with greater generalizability to new cohorts and potential for clinical translation.

Another key challenge in the use of CNNs—this time not specific to brain disorders research—is that the tuning of the models requires substantial technical expertise, which in turn limits the accessibility of these models. In response to this challenge, the machine learning community has been developing protocols for facilitating the process of tuning the models' hyperparameters. Ideally, one would want to have an automated method to generate the right architecture for any given task, a process called automated machine learning. Google AutoML API (Real, Aggarwal, Huang, & Le, 2018) (https://cloud.google.com/automl/), Adanet (Cortes, Gonzalvo, Kuznetsov, Mohri, & Yang, 2017) (https://adanet.readthedocs.io/), and Autokeras (Jin, Song, & Hu, 2018) (https://autokeras.com/) are some of the latest libraries that have improved access to AutoML. We expect that these tools will be applied in brain disorders research in the near future to improve the performance of neural network models including CNNs.

10.5 Key points

- CNNs were inspired by experiments on the visual cortex and typically comprise of convolutional, pooling, nonlinearity, and fully connected layers.
- CNNs were designed to deal with grid-structured inputs with strong local dependencies and as such are well-suited for the analysis of biosignals and neuroimages.
- Convolutional layers—the essential component of a CNN—learn local patterns in small windows; in contrast the fully connected layer—typical of traditional neural networks—learns global patterns in the total input space.
- By combining local connectivity, spatial invariance, and hierarchical features, CNNs are highly efficient in reducing the number of parameters.
- In brain disorders research, CNNs have been used to classify between categories, predict clinical and cognitive features, segment specifically tissues, and detect abnormal patterns.

References

Abdel-Hamid, O., Mohamed, A., Jiang, H., Deng, L., Penn, G., & Yu, D. (2014). Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing, 22*(10), 1533–1545.

Acharya, U. R., Oh, S. L., Hagiwara, Y., Tan, J. H., & Adeli, H. (2018). Deep convolutional neural network for the automated detection and diagnosis of seizure using EEG signals. *Computers in Biology and Medicine, 100*, 270–278.

Angelova, A., Krizhevsky, A., Vanhoucke, V., Ogale, A. S., & Ferguson, D. (2015). Real-time pedestrian detection with deep network cascades. *BMVC, 2*(4).

Bhattacharyya, A., Pachori, R., Upadhyay, A., & Acharya, U. (2017). Tunable-Q wavelet transform based multiscale entropy measure for automated classification of epileptic EEG signals. *Applied Sciences, 7*(4), 385.

de Brebisson, A., & Montana, G. (2015). Deep neural networks for anatomical brain segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops* (pp. 20–28).

Cortes, C., Gonzalvo, X., Kuznetsov, V., Mohri, M., & Yang, S. (2017). *Proceedings of the 34th International Conference on Machine Learning, 70*, 874–883.

De Fauw, J., Ledsam, J. R., Romera-Paredes, B., Nikolov, S., Tomasev, N., Blackwell, S., et al. (2018). Clinically applicable deep learning for diagnosis and referral in retinal disease. *Nature Medicine*. https://doi.org/10.1038/s41591-018-0107-6.

Deniz, C. M., Xiang, S., Hallyburton, R. S., Welbeck, A., Babb, J. S., Honig, S., et al. (2018). Segmentation of the proximal femur from MR images using deep convolutional neural networks. *Scientific Reports, 8*(1), 16485.

Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., et al. (2017). Dermatologist-level classification of skin cancer with deep neural networks. *Nature, 542*(7639), 115.

Frisoni, G. B., Fox, N. C., Jack, C. R., Jr., Scheltens, P., & Thompson, P. M. (2010). The clinical use of structural MRI in Alzheimer disease. *Nature Reviews Neurology, 6*(2), 67.

Gupta, A., Ayhan, M., & Maida, A. (2013). Natural image bases to represent neuroimaging data. In *International conference on machine learning* (pp. 987–994).

Havaei, M., Davy, A., Warde-Farley, D., Biard, A., Courville, A., Bengio, Y., et al. (2017). Brain tumor segmentation with deep neural networks. *Medical Image Analysis, 35*, 18–31. https://doi.org/10.1016/j.media.2016.05.004.

He, L., & Cao, C. (2018). Automated depression analysis using convolutional neural networks from speech. *Journal of Biomedical Informatics*.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. *ArXiv*, 1–11. Preprint https://doi.org/10.1109/ICCV.2015.123.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).

Hubel, D. H., & Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology, 160*(1), 106–154.

Jin, H., Song, Q., & Hu, X. (2018). Efficient neural architecture search with network morphism. *ArXiv*. Preprint ArXiv:1806.10282.

Ji, S., Xu, W., Yang, M., & Yu, K. (2013). 3D convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 35*(1), 221–231.

Kalchbrenner, N., Grefenstette, E., & Blunsom, P. (2014). A convolutional neural network for modelling sentences. *ArXiv*. Preprint ArXiv:1404.2188.

Kamnitsas, K., Bai, W., Ferrante, E., McDonagh, S., Sinclair, M., Pawlowski, N., et al. (2017). Ensembles of multiple models and architectures for robust brain tumour segmentation. In *International MICCAI brainlesion workshop* (pp. 450–462). Springer.

Khvostikov, A., Aderghal, K., Benois-Pineau, J., Krylov, A., & Catheline, G. (2018). 3D CNN-based classification using sMRI and MD-DTI images for Alzheimer disease studies. *ArXiv*. Preprint ArXiv:1801.05968.

Kim, Y. (2014). Convolutional neural networks for sentence classification. *ArXiv*. Preprint ArXiv:1408.5882.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097–1105).

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86*(11), 2278–2324.

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3431–3440).

Menze, B. H., Jakab, A., Bauer, S., Kalpathy-Cramer, J., Farahani, K., Kirby, J., et al. (2015). The multimodal brain tumor image segmentation benchmark (BRATS). *IEEE Transactions on Medical Imaging, 34*(10), 1993.

Payan, A., & Montana, G. (2015). Predicting alzheimer's disease: A neuroimaging study with 3D convolutional neural networks. *ArXiv*, 1–9. Preprint ArXiv:1502.02506.

Pereira, S., Pinto, A., Alves, V., & Silva, C. A. (2016). Brain tumor segmentation using convolutional neural networks in MRI images. *IEEE Transactions on Medical Imaging, 35*(5), 1240–1251.

Petersen, R. C., Aisen, P. S., Beckett, L. A., Donohue, M. C., Gamst, A. C., Harvey, D. J., et al. (2010). Alzheimer's disease neuroimaging initiative (ADNI): Clinical characterization. *Neurology, 74*(3), 201–209. https://doi.org/10.1212/WNL.0b013e3181cb3e25.

Rajpurkar, P., Irvin, J., Zhu, K., Yang, B., Mehta, H., Duan, T., et al. (2017). Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning. *ArXiv*. Preprint ArXiv:1711.05225.

Rathore, S., Habes, M., Iftikhar, M. A., Shacklett, A., & Davatzikos, C. (2017). A review on neuroimaging-based classification studies and associated feature extraction methods for Alzheimer's disease and its prodromal stages. *NeuroImage*. https://doi.org/ https://doi.org/10.1016/j.neuroimage.2017.03.057.

Real, E., Aggarwal, A., Huang, Y., & Le, Q. V. (2018). Regularized evolution for image classifier architecture search. *ArXiv*. Preprint ArXiv:1802.01548.

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems* (pp. 91–99).

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention* (pp. 234–241). Springer.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., et al. (2015). ImageNet large scale visual recognition challenge. *International Journal of Computer Vision, 115*(3), 211–252. https://doi.org/10.1007/s11263-015-0816-y.

Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 815–823).

Sharma, M., Pachori, R. B., & Acharya, U. R. (2017). A new approach to characterize epileptic seizures using analytic time-frequency flexible wavelet transform and fractal dimension. *Pattern Recognition Letters, 94*, 172–179.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., et al. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature, 529*(7587), 484.

Truong, N. D., Nguyen, A. D., Kuhlmann, L., Bonyadi, M. R., Yang, J., Ippolito, S., et al. (2018). Convolutional neural networks for seizure prediction using intracranial and scalp electroencephalogram. *Neural Networks, 105*, 104–111.

Zhu, Y., Shang, Y., Shao, Z., & Guo, G. (2018). Automated depression diagnosis based on deep networks to encode facial appearance and dynamics. *IEEE Transactions on Affective Computing, 9*(4), 578–584.

Zikic, D., Ioannou, Y., Brown, M., & Criminisi, A. (2014). Segmentation of brain tumor tissues with convolutional neural networks. *Proceedings MICCAI-BRATS*, 36–39.