Techniques of Artificial Intelligence: Image classification using Convolutional Neural Networks

Charlotte Hendrickx[1], Chris Adam[1] and Olivier Renson[1]

[1] Université Libre de Bruxelles, Masters in Bioinformatics and Modelling

Abstract

Image classification has proven useful in a number of areas like cancer diagnosis, species identification, text recognition,... In this work, we tried to classify images of cats and dogs using a Convolutional Neural Network (CNN). Convolutional Neural Networks are extensively used in computer vision problems and have yielded satisfactory results in various applications. We will first introduce Neural Networks and Convolutional Neural Networks. Then, we will briefly describe the data and the methodology we applied. Finally, we will comment our results and compare them with the results obtained by another group that applied Reinforcement Learning on the same dataset.

Introduction

Neural Networks

Neural networks have been used extensively in AI over the past decades. They have been applied to various topics and enabled breakthroughs in multiple fields.
The structure of a neural network mimics the structure of the human brain. It is composed of multiple layers of "neurons", with a first layer called *input layer* and a final layer called *output layer*. In between them, there can be a varying number of so called *hidden layers*, to which we cannot have a direct access. The neurons in these different layers are connected by weights, that represent the probability of transfer of information from one node to another.
When passed through a neural network, the inputs are multiplied by their respective weights. The outputs are then summed and passed through an activation function. Only when the output is greater than a certain threshold, the information is passed through. The output then becomes the input of another new neuron.

Formally, at each step, each neuron undergoes the following computation (see Figure 1): given x_i being the input of a neuron j coming from neuron i, $w_{i,j}$ being the weight between the neuron i and j, a_j being the activation of neuron j and o_j being a decision rule and θ_j being a threshold:

$$\mathrm{a}_j = \textstyle\sum_i x_i \; w_{i,j} \qquad\qquad o_j = f(a_j) = \begin{cases} 1\; if\; a_j \geq \theta_j \\ 0\; if\; a_j \leq \theta_j \end{cases}$$

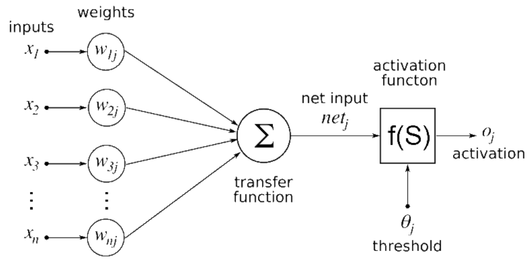

Figure 1: Mathematical representation of a neuron in a neural network. Reproduced from: Wikipedia

During the training phase, we input several images with their labels. We then compute the error, meaning the difference between the output and the true labels of the images. We can then update the weights connecting the different neurons using an algorithm called *error backpropagation* in order to obtain a better performance.

Convolutional Neural Networks

An expansion of classical neural networks (as commented before), used mainly for computer vision applications are the Convolutional Neural Networks (CNN). CNNs are used to capture local features (ex. neighbouring pixels forming a border) and to reduce the complexity of the model.

Like in conventional neural networks, CNNs contain an input layer and an output layer. However, in between it also contains additional layers of two types. The first layer after the input layer is a *convolutional layer* and the last layer is always a *fully connected layer*. In between, there can be various numbers of convolutional layers or *pooling layers*. We will define these terms later on. (see Figure 1).

Convolution Convolution is a mathematical operation that enables to extract the local features of an element (e.g. an

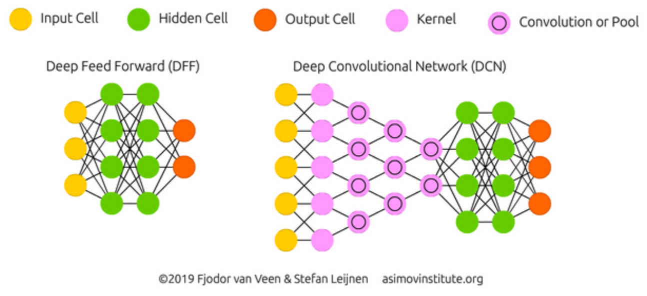
Input Cell
Hidden Cell
Output Cell
Kernel
Convolution or Pool
Deep Feed Forward (DFF)
Deep Convolutional Network (DCN)
©2019 Fjodor van Veen & Stefan Leijnen asimovinstitute.org

Figure 2: Comparison of a deep neural network (left) and a convolutional neural network (right). Adapted from: The Asimov Institute

image) and thereby reduced the dimmensionality of our input. It takes an input I and a kernel K (an array of parameters acting as a filter) as arguments and outputs a feature map $S(i,j)$. With m and n being the number of rows and columns of the input, equation 1 expressed the general idea of a convolution:

$$S(i,j) = (I \cdot K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{1}$$

Graphically, we could understand a convolution operation as shown in Figure 3. The input is shown in green, the kernel are the red numbers. We slide the kernel over the input and multiply them. We then take the sum (or another function like the mean), which gives the output of the convolution, called the *feature map*. The kernel slides first across the columns and then along the rows. Each kernel extracts different features from the input by checking if they are present at that location.
This convolution step has a number of advantages compared to traditional neural networks:

[label=213]In neural networks, each output neuron interacts with all input neurons of the following layer, which corresponds to a matrix multiplication. In CNN however, a neuron is only connected to a few neurons of the next layer, often those that lie close to him. This results of the convolution operation, where the number of connected neurons in the previous layer depend on the size of the kernel. We thus have to store less parameters, because the multiplication doesn't occur between such large matrices. In CNN, the kernel is used at every position of the input. We do not need to learn separate sets of parameters for every location (like in neural networks), but only have to store one set.

This enables to reduce the number of dimensions and thus of parameters to compute at each step.

Multiple convolutions with different kernels are applied in parallel to produce a set of activations, each extracting a different type of feature.

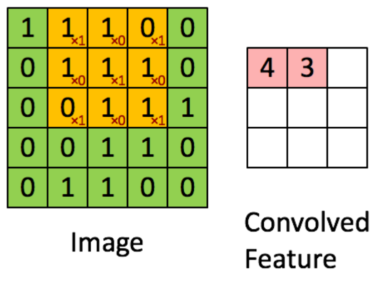

Figure 3: Convolution operation. Reproduced from: Stackexchange

Pooling After applying the convolution function, we get a feature map. However, this can still be very large. We can then apply a pooling function (= downsampling), which aggregates statistics of these features at different locations. For example, in *max-pooling*, we take the maximal value of a set of the feature map. (See Figure 4)

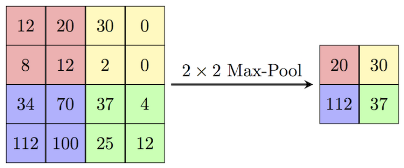

Figure 4: Example of a max-pooling operation. Reproduced from Computerscience Wiki

This pooling operation has two main advantages: it enables to reduce the dimmensionality and to avoid overfitting. Because the pooling only retains the maximum (or mean) of the feature map, it thus becomes invariant to small translations of the input.

Fully connected layer The final layer of the Convolutional Neural Network is always a fully connected layer. This layer performs the final classification step based on the features extracted by the previous layers.

Material and methods

In this section, we will present our implementation of the Convolutional Neural Network.

Data

Because different groups of students chose the Dog-Cat classification problem, we had to define a uniform benchmark dataset in order to compare the results. We used a Kaggle dataset composed of 10,000 images: 4,000 images of each class used as training set and 1,000 images of each class used as testing set. However, we put all the images together and made a new train-test split with the same proportions.

References

Andrew, N., Jiquan, N., Chuan Yu, F., Yifan, M., Caroline, S., Adam, C., Andrew, M., Awni, H., Brody, H., Tao, W., and Sameep, T. Deep learning tutorial. http://ufldl.stanford.edu/tutorial/supervised/Pooling/.

Education, I. C. Convolutional neural networks. https://www.ibm.com/cloud/learn/convolutional-neural-networks.

Education, I. C. Neural networks. https://www.ibm.com/cloud/learn/neural-networks.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Van Veen, F. Leijnen, S. (2019). The neural network zoo. https://www.asimovinstitute.org/neural-network-zoo.