Object Recognition Using Neural Networks: Review and Investigation

Viktoria Plemakova
Institute of Computer Science
University of Tartu
viktoria.plemakova@ut.ee

***Abstract*—A Convolutional Neural Network (CNN) is a type of neural network that is frequently used for tasks concerning image or object recognition and classification but also for various other problems. Firstly, this paper gives an overview of basic neural network structure and then an introduction to the architecture of CNNs and its usage with images. Lastly, author describes her initial implementation of CNN.**

I. Introduction

First research and implementations related to artificial neural networks (ANN) date way back. ANNs emulate the human brain and its neural network and use it to learn and understand data, just like humans or animals. There has been a lot of improvement since then and neural networks are now widely used in machine learning showing outstanding results in various fields, such as computer vision or natural language processing.

Convolutional Neural Networks (CNNs) are a type of ANNs that have recently become more important in image recognition and image vision fields. For example, CNNs have been used in ImageNet Large Scale Visual Recognition Challenge [1]. Although traditional ANNs can be used for image processing, CNNs have proven to be more suitable for such tasks mainly due to better computational complexity [2]. This paper will explain the use and structure of ANNs and more specifically CNNs since further implementation is based on that type of neural network.

II. Neural Networks

An Artificial Neural Network (ANN) is a computational model that was inspired by the nervous system of the human brain and its ability to process and transmit information. Neural networks are widely used in machine learning to solve problems concerning computer vision, natural language processing, pattern and speech recognition.

A. Artificial neuron

A neuron is the processing unit of a neural network. It takes n inputs, $x_1, x_2, ..., x_n$, and computes the output y. Each input is assigned a weight, $w_1, w_2, ..., w_n$, that shows input's significance. Weights are used in output computation. The output is computed by first finding the weighted sum of inputs and then applying an activation function to it:

$$y = f(z) \text{ where } z = \sum_{i=1}^{n} w_i x_i \tag{1}$$

See Figure 1 for a graphical representation of such artificial neuron.

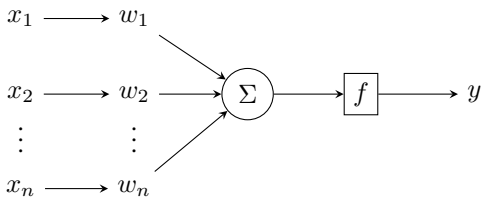

Fig. 1: Artificial neuron structure

B. Activation functions

Activation functions are used to introduce non-linearity to computations. There are several activation functions that can be used with neural networks. Some of more common ones are [3]:

1) Step function is the simplest activation function that outputs either 1 or 0.

$$f(z) = \begin{cases} 1, & \text{if } z \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

2) Sigmoid function $\sigma(z)$ outputs a value between 0 and 1. For very small values the output is close to 0 and for very large values the output becomes close to 1.

$$f(z) = \frac{1}{1 + e^{-z}} \tag{3}$$

3) Hyperbolic tangent function $tanh(z)$ is similar to the sigmoid function but its output falls into a range between -1 and 1.

$$f(z) = tanh(z) = \frac{2}{1 + e^{-2z}} - 1 \tag{4}$$

4) ReLU or Rectified Linear Unit is a function that is widely used with convolutional and deep neural networks. All negative values become zero after applying this function.

$$f(z) = max(0, z) \tag{5}$$

C. Neural network architecture

Neural networks are composed of several artificial neurons that are interconnected and arranged in layers. A neural network always has an input and output layer consisting of input and output neurons respectively. All layers in between input and output are called hidden layers. When neurons from one layer are connected to every neuron in the previous layer, we have a fully-connected layer (see Figure 2).

We can also distinguish between different types of ANNs depending on how the information is passed between neurons. The most typical case is a feedforward neural network, also called a multilayer perceptron (MLP). Here the output from one layer is passed to the next layer, and none of the information is passed back at any stage. Feedback is possible with recurrent neural networks. [4]

The output of the ith neuron in the lth layer can be computed similarly as in subsection II-A

$$y_i^l = f(z_i^l) \text{ where } z_i^l = \sum_{j=1}^{n^{l-1}} w_{ij}^l x_j^{l-1} \tag{6}$$

where we sum over neurons in the layer $(l-1)$ and each ith neuron depends on outputs in layer $(l-1)$ [5].

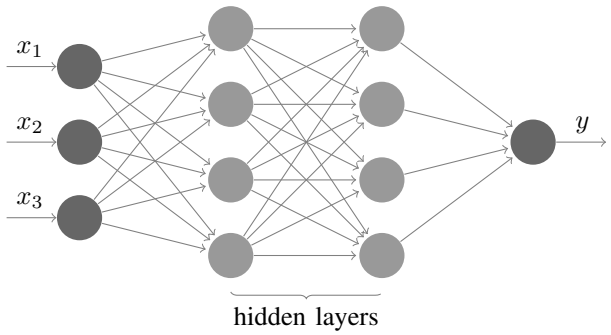

Fig. 2: A neural network with 3 inputs, two hidden layers and one output. This neural network contains fully-connected layers.

III. Convolutional Neural Networks

A Convolutional Neural Network (CNN) is a type of multilayer ANN which takes images as its input. The name of this neural network comes from the usage of an operation called convolution. CNNs are especially used in image recognition and classification since CNNs imitate how we see things.

A CNN usually includes three types of special layers: convolution, pooling and fully-connected layer.

A. Convolution layer

Convolution is an operation that we perform on an image to extract features, such as edges or textures. In this context, an image is a two-dimensional $n \times m$ array of pixel values from 0 to 255 — black and white respectively.

A kernel or filter K is used to do the convolution on the image. It is another two-dimensional $n \times m$ array of values that we consider as weights. More than one filter can be used to perform convolution so that multiple features can be extracted.

The output of the convolution layer is a feature map. The convolution process starts from the top-left corner of the image and is performed by sliding the kernel over the entire image. Each region that we slide over is called a receptive field [5]. Each receptive field is connected with one neuron in the feature map (see Figure 3) and uses the same weights [5]. Shared weights mean that we are trying to detect same features in different parts of the image [5].

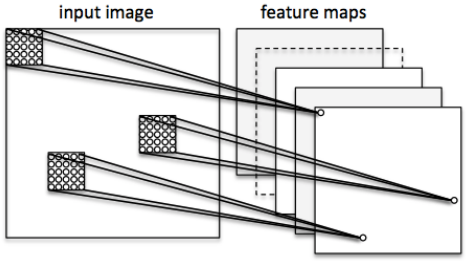

Fig. 3: Convolution layer [6].

At each region element wise multiplication is performed between the kernel and the region of the image and then products are summed up. The resulting value in each region is the corresponding pixel in the feature map. In the case of a two-dimensional input I and kernel K we can express convolution in the following way [4]:

$$\begin{aligned} Z_{ij} &= (I * K)_{ij} \\ &= \sum_m \sum_n I_{i-m,j-n} K_{mn} \end{aligned} \tag{7}$$

An example of image convolution can be seen in Figure 4. The grayscale image of a cat in Figure 4a is convolved with the following an edge detection kernel of size 3×3:

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}.$$

(a) Original image [7] in grayscale.

(b) Image 4a after convolution.

Fig. 4: Example of image convolution.

Aside from the kernel size we specify how many pixels at a time we move along the image. This number is called the

stride length and typically it is set to 1. The larger the stride, the smaller the output [3].

B. Pooling layer

A pooling or down-sampling layer commonly follows each convolution layer and takes feature maps as its input. In case of multiple feature maps pooling is applied to each feature map separately. The purpose of pooling layers is to simplify the information coming from convolution layers [5] but preserving important parts of the feature map at the same time [8]. There are different types of pooling, such as max, sum, L2 or average pooling [8], among which max-pooling is the most common approach.

The feature map is divided into non-overlapping regions (receptive fields) of size $n \times n$ (imagine sliding over the image as in convolution) and one of the previously mentioned pooling procedures is then applied to each region. The filter size is commonly 2×2 and the stride size 2. For example, in the case of max-pooling, the largest value of each receptive field is the output. A sample result of 2×2 max-pooling on image can be seen in Figure 5.

(a) Image 4b. (b) Image 5a after max-pooling.

Fig. 5: Example of max-pooling on convolved image.

C. Fully-connected layer

One or more final layers are fully-connected. Like mentioned in II-C, a layer is fully connected if it has connections to every neuron in the previous layer. At this stage previously two-dimensional outputs are converted into a one-dimensional feature vector. The purpose of the fully-connected layer is to use previously learned features for final classification.

IV. Implementation

For the seminar a partial implementation of CNN in Python was done, namely the convolution and pooling layers, based on the theoretical information from [5] and [3]. Currently the solution supports one kernel in each convolution layer but there are no other major structural differences from the explanation in section III.

V. Conclusion

This paper has given an overview of architecture and components of artificial neural networks as well as convolutional neural networks. In addition, the implementation of convolutiona neural network in Python was started. Future work will include developing and the implementation further and conducting training and testing with a dataset for vehicle detection.

References

[1] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[2] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," *CoRR*, vol. abs/1511.08458, 2015. [Online]. Available: http://arxiv.org/abs/1511.08458

[3] A. Karpathy, "Cs231n convolutional neural networks for visual recognition," 2017. [Online]. Available: http://cs231n.github.io/

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

[5] M. A. Nielsen, *Neural Networks and Deep Learning*. Determination Press, 2015, http://neuralnetworksanddeeplearning.com/.

[6] S. Raschka, *Python Machine Learning*. Birmingham, UK: Packt Publishing, 2015.

[7] Alexas_Fotos, "Cat picture," 2016. [Online]. Available: https://pixabay.com/en/cat-face-close-view-eyes-portrait-1334970/

[8] Y.-L. Boureau, J. Ponce, and Y. Lecun, "A theoretical analysis of feature pooling in visual recognition," in *27TH INTERNATIONAL CONFERENCE ON MACHINE LEARNING, HAIFA, ISRAEL*, 2010.