Math 521 - Final Project

Felix Alcantara
Emily Conway
Pablo Gomez

May 8, 2018

Abstract

This project will consist on creating different classification methods for images of cats and dogs. We are given 160 images, in which, 80 are dog images and 80 are cat images. This paper will refer to three classification methods; Convolutional Neural Networks, Linear Discriminant Analysis, and Emily's method. We are going to analyze each result and make a conclusion based on it.

1 Introduction

2 Mathematical Background

2.1 Convolutional Neural Network

Convolutional Neural Network (CNN or Convet) is a structure used in the area of **Deep Learning**. A biological explanation to CNNs has made possible thanks to T.B Hubel and T.N. Wiesel. In their research, they proposed an explanation of how animals (mammals) see (visually speaking) the world around them, which lead to an architecture construction of neurons in the brain. This idea inspired engineers to create an architecture similar in computer vision.

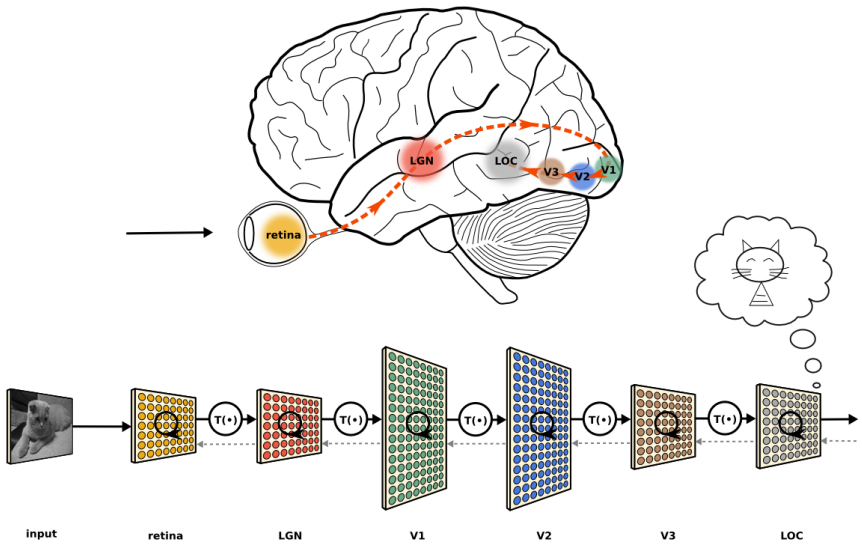

Figure 1: Neural Network from the human prospective

How does CNN works:
CNNs are biologically-inspired programming algorithm that allows a computer from observational data. In other words, Convolutional Neural Networks are made to imitate the process made by the visual cortex on mammalians (including humans). The CNN structure consist sending an image through different layers in order to obtain a classification. Each layer has an assigned mathematical operation that let us collect some of the data needed for the classification. The filters used for the Convolutional Neural Network for this project are **kernel filter, pooling filter** and **fully connected (FC) filter**.
CNN architecture: The whole CNN architecture and the work done by each of its layers will be explained in the following steps

1. Preparing the images:
Recall that, for this project, we have a collection of cat and dog images. Each image has a matrix representation of pixel values [0,255].
2. Convolution:
The formal definition of convolution in the continuous form as:

$$(f \star h)(t) = \int_{-\infty}^{\infty} f(\tau) \cdot h(t-\tau) d\tau = \int_{-\infty}^{\infty} h(\tau) \cdot f(t-\tau) d\tau$$

We know that *correlation* (in image processing applications) is the process of moving a filter mask (f) over a signal (s) or image and computing the sum of products at each location and how *convolution* works similarly except that the filter is rotated 180°. We have also studied the interplay between the convolution with filter and the Fourier series of the filter; where

$$(f \star h)(t) = \mathcal{F}^{-1}\{\sqrt{2} F(\mu) H(\mu)\}$$

In convolutional Neural Network terminology, the convolution operation is the same as the convolution operation from image processing:

$$(x \star w)(t) = \int_{-\infty}^{\infty} x(\tau) \cdot w(t-\tau) d\tau$$

Where the first argument to the convolution is referred as our image **input** and the second argument as our **kernel**. The output is referred as the **feature map**.

Since we use convolution on more than one axis at the time, we can use the discrete convolution in 2D to define spatial filtering in manipulating images:

$$S(i,j) = (I \star K)(i,j) = \sum_m \sum_n I(m,n) K(i-m, j-n)$$

the variable I represents our two-dimensional image as an input, and K is our two-dimensional kernel. A kernel (also known as filter) is a smaller size matrix, compared to the input (image) matrix, which consists of real value entries. Kernels are convolved with the input by using the dot product and the result forms a single entry in the feature (activation) map. The total number of entries in the activation map are based on these three values: **depth, stride**, and **zero-padding**.

Depth: refers to the output volume from the dot product between the filters and the input; The depth is based in the number of filters we would like to use (each filter will try to learn something about the input).

Stride: Refers to the amount of shifting assigned to each filter (starting from top left to right, or downwards when the boundary of the matrix is reached). This process is repeated until the whole input image has been processed. instead of connecting all neurons to all possible pixels, we specify

a region that in the input that has the same size as the filter (called **receptive field**). The result of the filters pointing at the same receptive field will be stored in the **column depth**.

zero-padding: Addition of zeros outside the input volume so that way the convolution process ends up with the same number of outputs as the same number of inputs. It is said that, if we don't use padding, the information at the borders will be lost after each Convolutional layer, which will reduce the size of the volumes as well as the performance. However, the *Krizhevsky et al.* architecture that won the ImageNet Challenge in 2012 had no zero-padding.

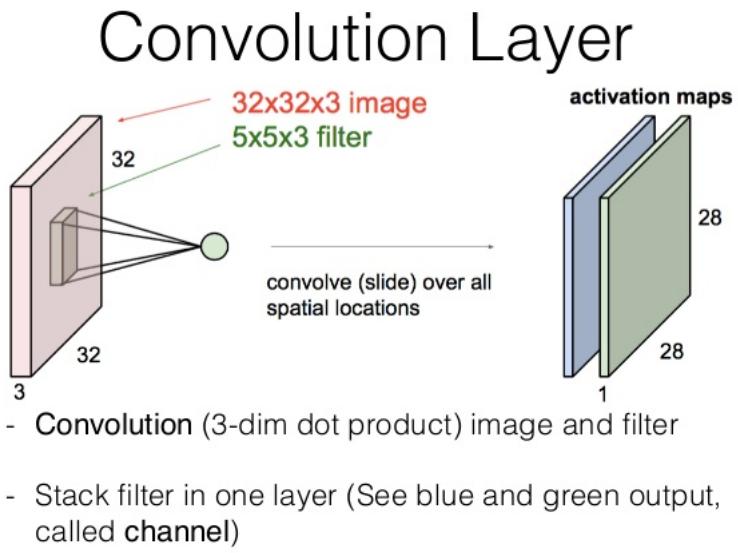

Figure 2: Convolution layer process

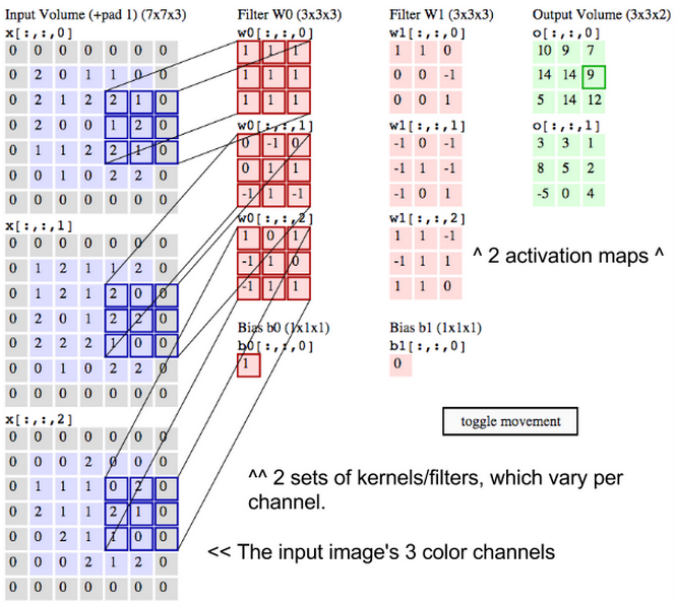

Figure 3: Convolution Process with padding and strides

Calculating the size of the output: Here is the following formula to calculate the the volume of the output; suppose that the input image has a volume of of $W_1 \times H_1 \times D_1$, then we are required to choose the following hyper-parameters:

$$\begin{aligned} \text{Number of filters} &: K \\ \text{The spatial extent (filter size)} &: F \\ \text{Stride} &: S \\ \text{The amount of zero-padding} &: P \end{aligned}$$

Which will give us the output volume $W_2 \times H_2 \times D_2$, where

$$W_2 = \frac{W_1 - F + 2P}{F} + 1$$

$$H_2 = \frac{H_1 - F + 2P}{F} + 1 \quad \text{(i.e. width and height are computed equally by symmetry)}$$

$$D_2 = K.$$

A common setting of these hyper-parameters is usually: $F = 3$, $S = 1$, $P = 1$, But this choice is under research and different approaches have been made since each Convolutional Network structure is different.

3. Pooling layer:
The pooling layer performs a function to reduce the spatial dimension of the input (only the width and the height), as well as reducing the computational complexity of our model. There are different types of pooling; Max pooling, L2 pooling, or average pooling. The most common type of pooling is the Max pooling, since it takes the highest value of a pixel around the filter. The size of a pooling filter is usually a 2×2 matrix with a stride value of 2; In the picture above, the Max

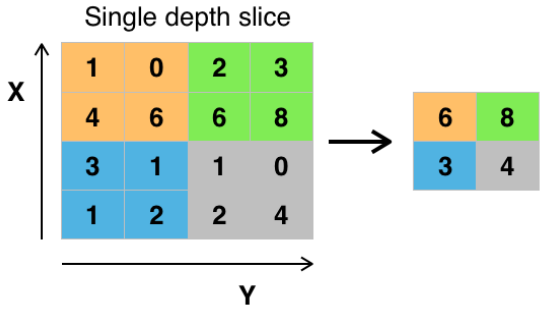

Figure 4: 2×2 Max polling filter with stride 2

Pooling operation will be taking the highest (pixel) value over 4 numbers. If we want to speak more generally, if we have an input volume of size $W_1 \times H_1 \times D_1$, with the following hyper-parameters

$$\begin{aligned} \text{Spatial extent} &: F \\ \text{Stride} &: S \end{aligned}$$

will produce an output volume of $W_2 \times H_2 \times D_2$, where

$$W_2 = \frac{W_1 - F}{S} + 1$$

$$H_2 = \frac{H_1 - F}{S} + 1 \quad \text{(not common to use padding in the Pooling layer)}$$

$$D_2 = D_1.$$

The common hyper-parameters for pooling are $F = 2$, $S = 2$. Anything with higher value could be too destructive.

4. Normalization Layer (ReLU in this case):
The ReLU (Rectified Linear Unit) normalization layer applies an element wise activation function $max(0, x)$ which turns negative values into zeros (i.e. thresholding at zero). This layer does not change the size of the input volume and does not have any hyper-parameters.

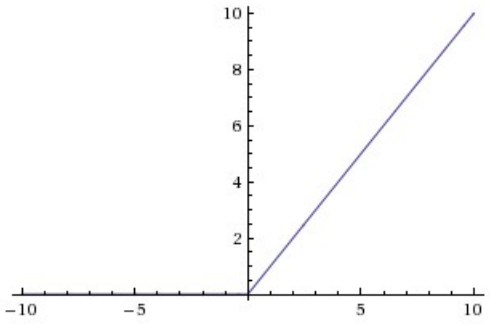

Figure 5: ReLU activation function

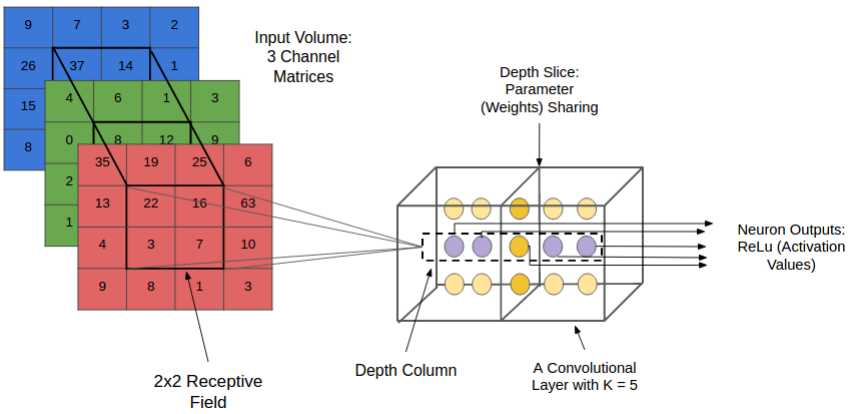

Figure 6: Application of the ReLU layer

5. Fully Connected (FC) layer:

Fully connected layers connect every neuron in one layer to every neuron in another layer. The last fully-connected layer uses a soft-max activation function for classifying the generated features of the input image into various classes based on the training dataset.

3 References

Jen-Mei Chang. *Matrix Methods for Geometric Data Analysis and Pattern Recognition.* California, 2014.