Neural Networks
Deep Learning

Silas Hoffmann, inf103088

December 10, 2019

Contents

List of Figures

1 Introduction

Notes for the introduction to neural networks presented by **3blue1brown**. Watch the whole series here: . This series describes the strategy to recognize handwritten numbers.

2 Chapter 1 - But what is a neural network?

2.1 Structural overview

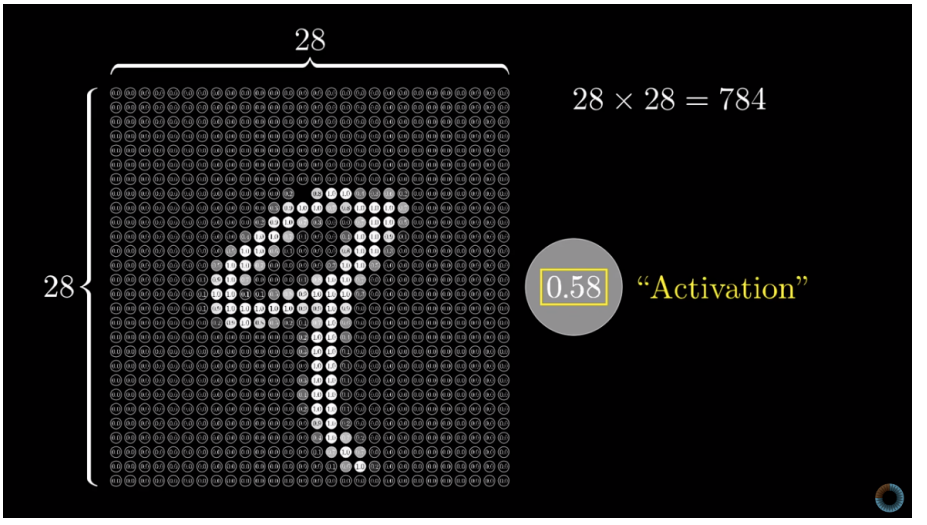

Figure 2.1: Image interpreted as first layer of neural network

Each pixel from the image is represented by a so called *neuron* (see image 2.1). These neurons possess a value called **Activiation** which contains a float between 0 and 1. This value specifies the brightness of this the particular pixel. This activation value also can be interpreted as how *lit up* this particular neuron actually is, 1 meaning 100 % in this context.

As each number is analyzed the top layer gives its prediction to a given input via Activation of the ten most right nodes (see image 2.1). The one with the highest activation states what the system thinks the correct number might be. The layers between the top and the bottom layer are called **hidden layers**.

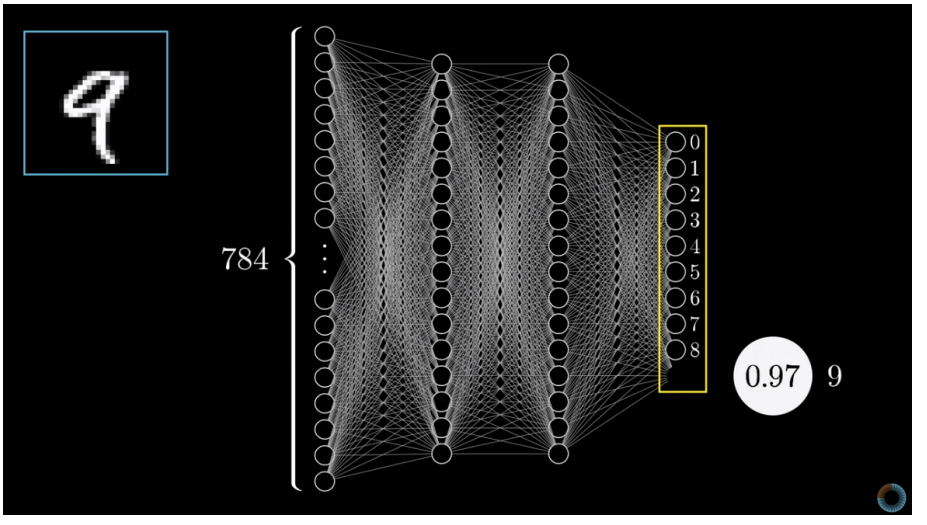

Figure 2.2: Resulting top layer

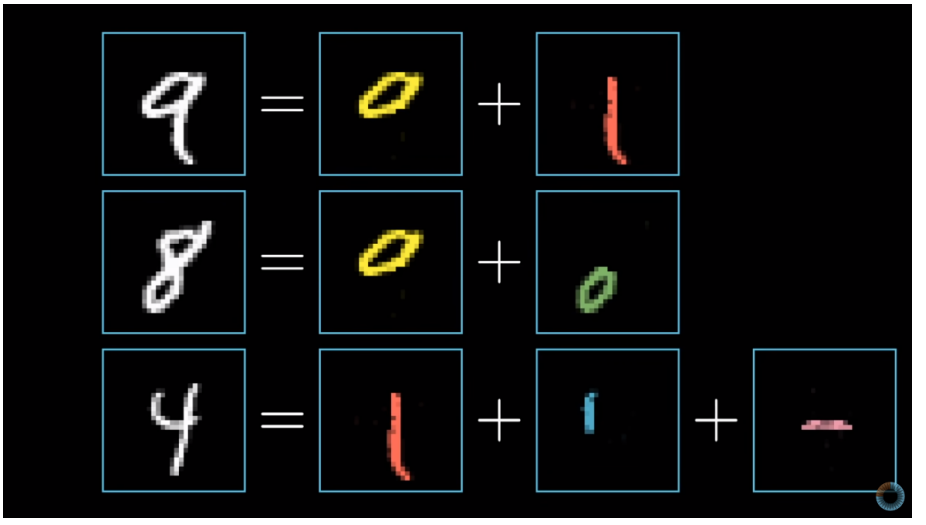

Figure 2.3: Image interpreted as first layer of neural network

The number of layers is not fixed and can be set to any number appropriate to the problem the system should solve. In this example it is useful to two hidden layers because of how the system works. A handwritten digit can be subdivided into different components (see image 2.1). E.g. the number 9 can be divided into a top loop and a bottom line. These different steps will be analyzed by the various layers from the network. Maybe some digits share specific components and those nodes can be *reused* (see image 2.1). The idea is that any digit with a *loopy* pattern towards the top sends of the specified neuron. This information is then used to fire up neurons in the next layer based on the idea that at least some kind of pattern already fits the input data.

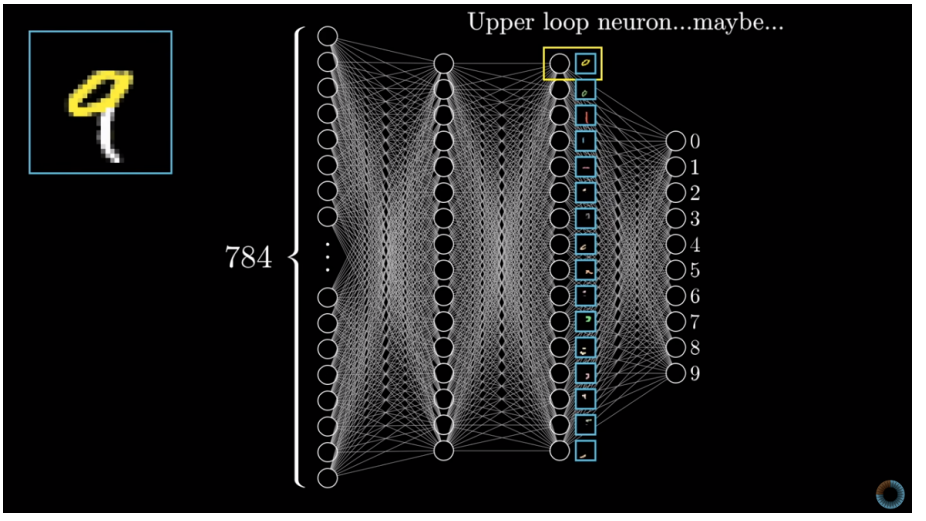

Figure 2.4: Recognizing more complex patterns via abstraction

But to be capable of recognizing such a complex patterns as a loop, the system has to recognize all the various subcomponents of a loop. Those mainly consist out of small edges which in total form a loop from a wider perspective. This may be the task of the second layer while the third layer is responsible for recognizing those more complex

parts (see image 2.1).

Again, the number of layers is totally arbitrary. It might even be usefull to divide those smaller lines from layer 2 into smaller components, then there might be another layer. Or maybe the second layer is redundant altogether because the system can recognize those complex pattern directly. It always depends on the problem that the network should solve to decide how many layers are actually usefull.

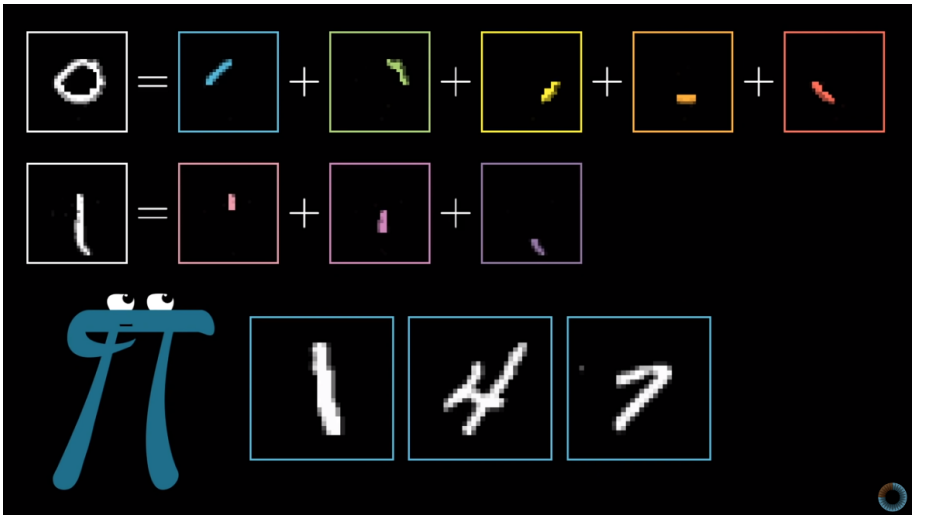

Figure 2.5: Examples of simple shapes forming complex digits

In a nutshell the different layers in a neural network each describe a certain level of complexity. Those lower levels represent simpler coherence than the ones on top of those. In the next picture you can see the which neurons on each layer may fire when confronted with a sample of a handwritten nine.

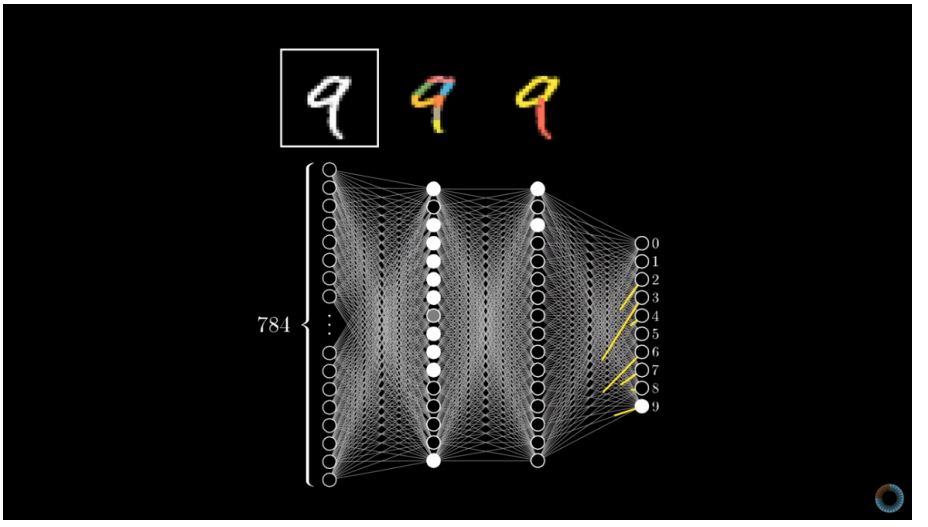

Figure 2.6: Top to bottom overview

2.2 Regognizing patterns

The main purpose of the given example network is to detect edges of the given digits and from these edges determine patterns which in turn result in a identified digit. But this principle of detecting details and forming a bigger picture is also relevant in other fields e.g. autonomous cars or speeach regognition where a given audio samples consists out of waves which form voices which form words etc.

To show how it actually works to detect one of those minor details to from the simples *hidden layer* the tutor shows how to recognize a single line in a specific part of the picture (see image 2.2).

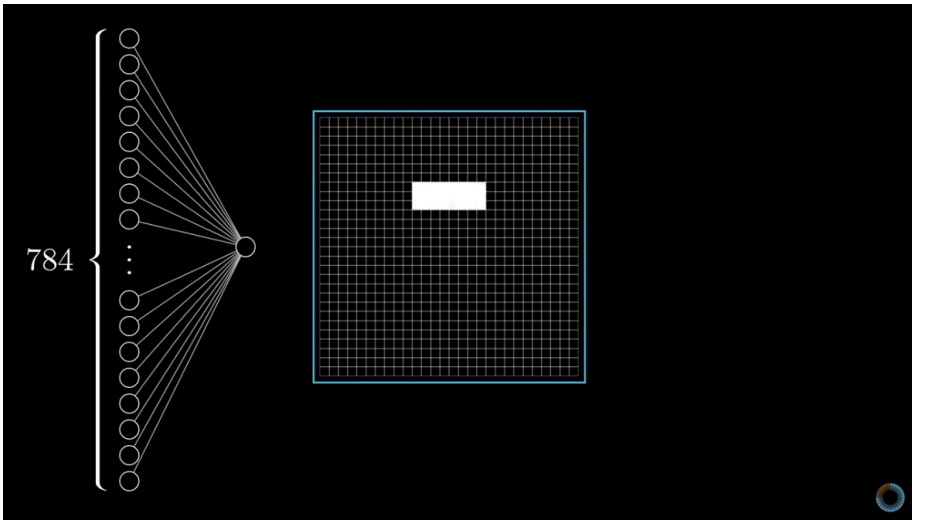

Figure 2.7: Bounding box for detecting specific line

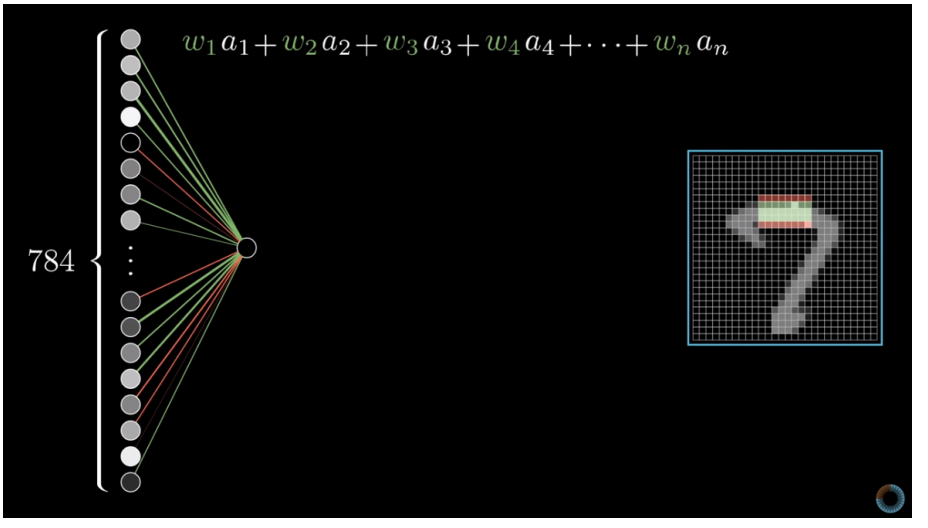

Figure 2.8: Colored weights for specifying outlines

In the example we try to detect the middle part of the digit seven (see image 2.2). To check whether or not the given pixels have a line in the specified part of the image we assign a so called **weight** to each of the connections of nodes on the first layer (direct

image input) to the node responsible for recognizing the depicted part of the image as a line. To do so each connection outside of the bounding box gets assigned a value of zero and the relevant nodes get a value greater zero. Now each weight will be multiplied with the corresponding node value *(brightness of the node)* and all those values will be added. Whenever there is actually a line in the specified box the sum of the values will be higher than if there wasn't a line. To make it state that we actually expect a line which does not touch the upper and lower bound of the box we can also create negative bounds for the outer pixels which in turn extravagates the result even more.

E.g. a black line crosses the otherwise completely blank / white canvas. Then the nodes which are expected to be white multiply by a quite big factor with the small values and the outer pixels touch the bound of the box. This will result in some pretty huge negative numbers because the weight are negative in those places. On the other hand, if you put a perfect line that is not touching the bounds in the box then the sum of all products with the weights will be much larger and the neuron will fire a value much ***brighter***.

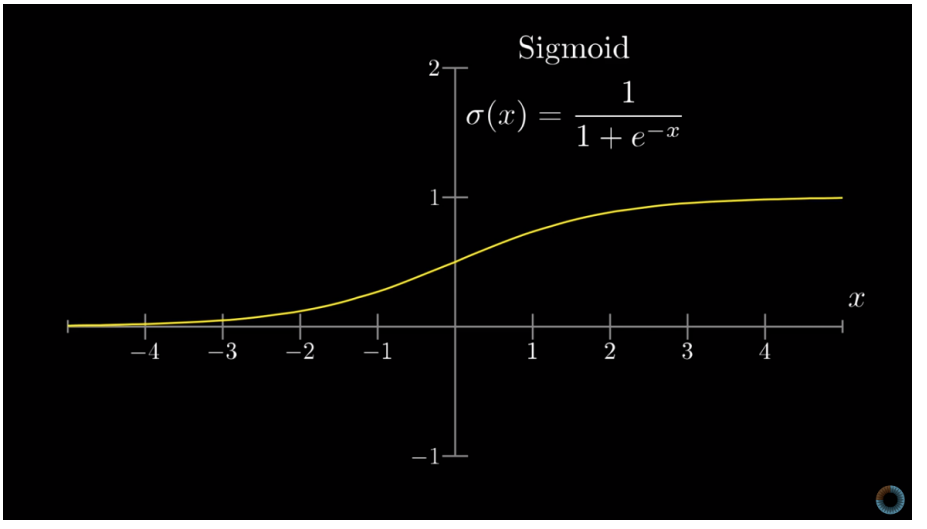

Figure 2.9: Sigmoid function

Since the calculated sum of the product with the weights may very well be much larger than 1 or smaller than 0, the sum will be squished into the the interval 0 to 1 via some kind of function. Often a function called the **sigmoid function** is used which creates a value near zero for negative values and a value near 1 for high values (see image 2.2). It is important to limit the output to this interval because it will serve as the **activision** of the node when the node is used in the next step / layer of the network.

A very important summary: *The activation of a neuron is basically a measure of how positive the relevant weighted sum is.*

Another important tool is the so called **bias** which acts a sort of lower bound for the activation of the neuron. Say you want the neuron to only detect very thick and bright lines. Then you need this bias to to say something like *only light up when weighted sum is greater than 10* (see image 2.2). This can be achived by subtracting the bias inside the sigmoid function.

2.2

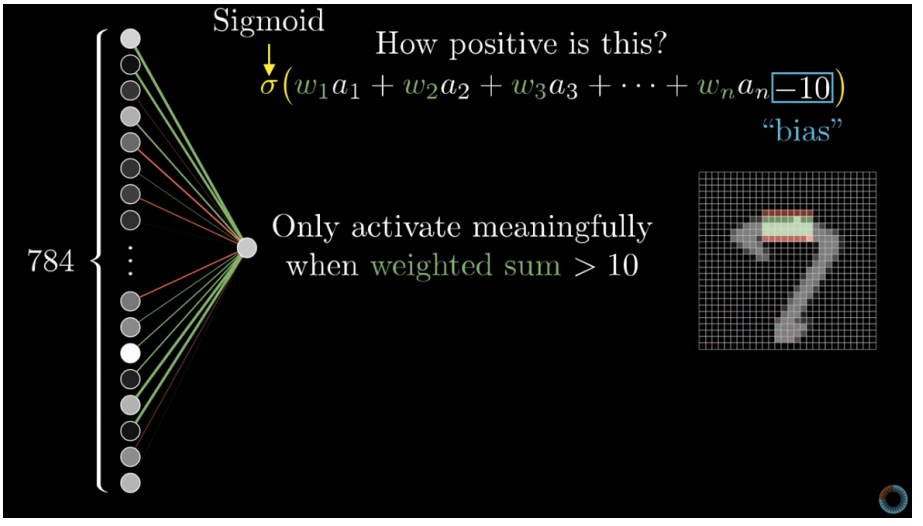

Figure 2.10: Activation displayed with bias and sigmoid function