An Introduction to Deep Neural Networks

Daniel Arroyo Troyano
Antonia-Irina Boca

March 2022

1 Introduction

Humans have dreamt of creating intelligent machines for centuries. With the rise of digital computing, Artificial Intelligence (AI) has become part of modern science. The goal of AI is to instill computer systems with intellectual capabilities characteristic of the human mind, such as the capacities for abstraction, logic, decision-making, and learning from past experiences. A specific family of AI methods has recently come into focus: deep learning, which is based on artificial neural networks. Neural networks have developed significantly in the last decade, and have become heavily integrated into the fabric of modern society — they power web searches, match social media posts with users' interests, transcribe speech into text and are even used to predict the stock market. The 'deep learning revolution' has been powered by exponentially increasing amounts of publicly available data; substantial advancements in the speed and capacity of computational hardware; and the democratisation of software platforms brought about by the open source movement. However, neural networks are by no means a new phenomenon; the first computational model that can be considered a neural network, the **perceptron**, was developed by Frank Rosenblatt (1958), a psychologist working at Cornell.

2 How does it work?

Rosenblatt simulated the basic principles behind information processing in neurons to build the perceptron. To many researchers working in AI at the time (and today) it seemed sensible to model intelligent machines after the biological systems that support cognition: brains. Consider the basic structure of a neuron: it has multiple branching dendrites, which carry electrochemical (input) signals into the cell. The neuron then adds up all the inputs it receives from other neurons, and if the total sum is greater than a certain threshold level (in humans, that would be a membrane potential value between -50 and -55 millivolts) the neuron fires, sending an (output) signal down its axon. Artificial neural networks, here illustrated by the Multilayer Perceptron (MLP) — the oldest type of neural network still in use today — share fundamental properties with their biological equivalents.

Now, let's say you want to "teach" a simple MLP to recognise pictures of the King's College Chapel.

Imagine you have a set of data points. These can be almost anything, like the colours of a picture's pixels. Let us call this set $\mathbf{S} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n)$, where each $\mathbf{x}_i$ is a data point. One data point can represent an entire picture. Let us then say that this picture has a resolution of 40×40 pixels. Then, $\mathbf{x}_i$ will be a set of 1600 numbers $(p_1, p_2, ..., p_n)$: with each number representing the colour of a pixel. In addition to your set $\mathbf{S}$ you have a set $\mathbf{L} = (t_1, t_2, ..., t_n)$, representing whether picture i, with pixels $\mathbf{x}_i$ in your data set $\mathbf{S}$ is indeed a picture of the King's College chapel. Our only restriction in building a neural network is that we need to encode this using numbers; so let us say that $t_i = 1$ if picture i is a picture of the Chapel and $t_i = 0$ otherwise. Together, $\mathbf{L}$ and $\mathbf{S}$ represent a labelled dataset.

Artificial neural networks consist of a layered assembly of interconnected nodes or 'neurons' designed to loosely mimic the architecture of the brain, with layers of neurons receiving information from previous layers of neurons. The first layer directly reads the input data, typically as a matrix of real values. Input neurons feed into one or more 'hidden' layers of nodes before passing into the final layer, which outputs the network's final prediction based on the input data. This architecture is illustrated in Figure 1. Notice how in the picture, the input layer contains 1600 nodes: one for each pixel. The hidden layer has pre-defined number of $\mathbf{K}$ nodes. Finally, the output layer contains two nodes that represent the "opinion" of the network: y_1 has a numeric value that represents the confidence of the network that the pixels from the input layer represent the King's College Chapel, while y_2 represents its confidence that the pixels are not, in fact, our beloved chapel.

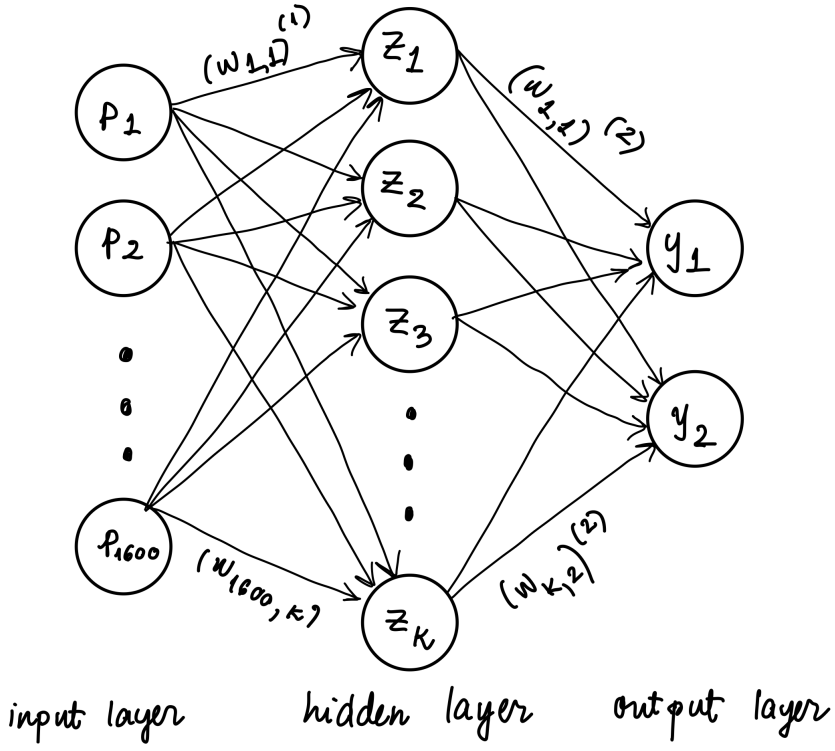

Figure 1: MLP with 1 hidden layer

The visual representation exhibits the dependencies between different layers. We

can quantify the impact of each node p_i on every other node z_j using a weight, $w_{ij}^{(1)}$. Each connection between neurons carries an associated weight as illustrated in our figure. Weights are values that represent the relative importance of a connection. In our network, they are the internal parameters of the model that are learned during training. They can be thought of as knobs that define the input-output function of the network, and a typical deep learning algorithm can have hundreds of millions of them. Similarly, in the brain, different connections between neurons (or synapses, in biological jargon) have different strengths. In calculating the sum of its inputs, a given neuron assigns more weight to the inputs from stronger connections. Crucially, it is believed that adjustments to the strength of synapses is a key part of the process of learning.

At first, in neural networks, these weights are randomly initialised. As you can guess, the verdict for the first picture will be randomly chosen between y_1 and y_2. However, we can quantify the error behind the network's decision by defining an error function. In literature, this is called a **loss function**. The greater its value, the farther away the verdict of the network strays from the known ground truth — here provided by the labelled dataset comprised of **L** and **S**.

Having quantified this error, what we have is fundamentally an optimisation problem: we have a function evaluated using some weights **w**, and we want to modify these weights such that the function is minimised. An entire field of mathematics is concerned with researching such techniques which allow us to perform function optimisation, and the functions that we employ allow us to use derivatives to reach our goal.

Updating the weights, then, consists of two stages: first, we compute the derivative of the error function with respect to each weight to get the function gradient information that we need. This computation follows a revolutionary algorithm, called **backpropagation** (Rumelhart et al., 1986) that can calculate the derivative of each weight *backwards* by starting with the output layer and going in reverse through the network, updating one layer at a time. The second stage carries out the actual update of the weights. The simplest of the existing techniques that perform this update is called **gradient descent**. This adjustment happens after seeing each example and comparing the prediction of the network to its label in the set **L**.

Finally, we have the architecture of a MLP, the *forwards* and *backwards* propagation algorithms, and the optimisation technique. The only thing that is missing is a good set of pictures of the King's College Chapel.

3 Training data, testing data

A neural network's ability to predict is dependent on the data that it receives in order to train its weights. Imagine what happens when you show a child a picture of the King's College Chapel: maybe they have never seen a chapel before, but they will immediately be able to identify it as a building, that it is surrounded by a couple of trees, that its foundations are on the ground and that the blue (or grey, according to the weather) upper part of the picture represents the sky. All these come from our brain's ability to break down information into its component structural parts, from our exposure to the real world and from our ability to comprehend language.

The child may immediately understand that "King's College" is a proper noun and therefore it is the *name* of this particular *chapel*.

Neural networks do not possess these abilities. This is why, while being trained for a particular task, they must be presented with all the context that they need, otherwise they may not be able to generalise. Think of it like this: if you show this network *only* pictures of the King's College Chapel, and later show them a picture of the St John's College Chapel, they might (to our horror) confuse them. This happens because the training data was not refined enough to underline the difference between our chapel and other chapels.

Another problem we might encounter during training is that the network *memorises* data points rather than learning a function that generalises the concepts presented in the data. Let's say you have 30 pictures of the chapel, all taken during sunny days (when the chapel is the prettiest, of course). Now, there's a likely scenario in which you give your network a picture of the chapel on a rainy day, it will be less confident, to the point where it classifies your picture incorrectly. This is called **overfitting**, and the first step towards avoiding this is to separate all the data that you have into a training set and a testing set. The performance of your network will be analysed only in respect with the latter. If the predictive accuracy of your training set is high, while the prediction accuracy of your testing set is low, then you are overfitting. The purpose of neural network is to be able to predict, with high accuracy, outcomes given before unseen data points; and for that, good quality data is the most important ingredient.

4 Big and Biased Data

Did you notice how Google Translate suddenly improved in 2015? That was the year Google started using neural networks for its translation service. The past years have seen an explosion in the usage of neural networks for all types of tasks: from predicting stock market prices to facial recognition services and playing Go (Google's *AlphaGo* defeated Lee Sedol, a Go world champion, in 2016). There is no doubt that there is a strong connection between the efficacy of neural networks and "Big Data". Social media platforms especially have fuelled this phenomenon, by systematically gathering labelled data from their users. If you have ever tagged a friend in a Facebook picture, identified a picture to prove you are not a robot on a website, reacted and commented to a post, or asked Siri to set an alarm, your actions and data may be part of the labelled set a neural network uses to train its weights.

It is important to note that datasets are not value-neutral, but rather prone to bias because data collected by people will inherently reflect human biases. For instance, in most genomic databases non-European populations are heavily underrepresented (Bentley et al., 2020), leading to an embedded bias in the data, which the algorithms then reflect in their predictions. In turn, this leads to misdiagnoses, uneven success rates in personalised medicine and poorer patient care for underserved groups. At the end of the day, these networks are only as good as the data they receive.

5 Model interpretability

After the process of mathematical optimisation that the millions of network parameters undergo, the result is an input-output function that reflects the real, physical system being modelled with state-of-the-art accuracy. But we are often more interested in **how** the network arrived at such a good prediction. For example, last year, in what was arguably the most ground-breaking application of AI in science to date, the London-based AI research group DeepMind released *AlphaFold2*, a program capable of predicting protein structures based on amino acid sequences with astounding accuracy, providing a solution to a 50-year-old grand challenge in biology. However, *AlphaFold2* has done little to advance our understanding of the protein folding process.

The problem is that deep neural networks are not designed to highlight interpretable relationships in data or to guide the formulation of mechanistic hypotheses; their intricate internal structures are considered to be 'black boxes', not based on human logic or corresponding to the actual system being modelled.

Consider a rough analogy between neural networks and the human brain. If it were possible to see through someone's head and watch some subset of the brain's billions of neurons firing, it is unlikely that any insight would be gained into that person's thoughts, or the 'rules' they used to arrive at a given decision. However, our brains have given rise to language, which allows humans to explain their thoughts with words and phrases. A neural network does not possess this ability (yet): its weights do not stand for particular concepts, they are often redundant, and show a non-linear relationship to the output. This makes it challenging to understand how different characteristics are used to classify the data or what role any particular set of neurons is playing in the classification process. As Ludwig Wittgenstein aptly phrased it in his *Last Writings on the Philosophy of Psychology* (1949): "nothing seems more possible to me than that people some day will come to the definite opinion that there is no copy in the nervous system which corresponds to a particular thought, or a particular idea, or memory".

6 Conclusion

We are currently witnessing an extraordinary technological advancement. Our computers no longer merely perform menial tasks, such as arithmetic, communication and entertainment, but are indeed starting to heavily influence what we think, say and do. Nonetheless, neural networks ought to be regarded as a tool that augments rather than replaces human capabilities. Appropriately trained human oversight is necessary for crucial checks such as ethical and validation tests, in the absence of which neural networks cannot fulfil their functions. If we are to live in a world that aspires towards equity, we should not replace the processes of human decision making with automated ones.

References

Amy R. Bentley, Shawneequa L. Callier, and Charles N. Rotimi. 2020. https://doi.org/10.1038/s41525-019-0111-x Evaluating the promise of inclusion of african ancestry populations in genomics. *npj Genomic Medicine*.

Frank Rosenblatt. 1958. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65 6:386–408.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1986. https://doi.org/10.1038/323533a0 Learning Representations by Back-propagating Errors. *Nature*, 323(6088):533–536.