

Comparing the Efficiency of a Neural Network

An EPQ By Sam Laister



June 30, 2020

Contents

[Small Introduction 2](#_Toc44448598)

[What will I be comparing? 2](#_Toc44448599)

[MNIST Database 2](#_Toc44448600)

[EMNIST Database 2](#_Toc44448601)

[Recourses Used (Credits) 3](#_Toc44448602)

Comparing the Efficiency of a Neural Network

# Small Introduction

Artificial Intelligence has always interested me. However, it’s a very time-consuming thing, making one. So, for my EPQ, I decided it was the perfect opportunity to actual have a go and code one from scratch. Since I take A Level Maths at college, I assumed I wouldn’t be in much over my head, which was sort of right. However, this was a lot more challenging and time-consuming than I originally though, and I didn’t think I’d make it to this point, with a finished product, at one point.

# What will I be comparing?

By the end of this, I hope to test multiple things; how well my neural network is at image detection. My initial idea was to include learning inputs for a videogame, e.g. Mario. While this would be possible to do with the product I have created, I don’t understand enough about emulation required to pull it off in time for the deadline. It also wouldn’t make sense in context of the test since detecting a handwritten 2 isn’t comparable with how long it could beat a simple level. But know that the maths and code required all boils down to exactly what I’ve created, so in the future it would be possible

## MNIST Database

So, for this test I have decided to compare detecting a number (0-9), a letter (a-z), and a simple drawing. In theory, these should all follow the same principles, just with different databases for it to learn from. The database for the numbers will be MNIST. From the website:

```The MNIST database of handwritten digits, available from this page, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image.

It is a good database for people who want to try learning techniques and pattern recognition methods on real-world data while spending minimal efforts on pre-processing and formatting.```

So, the general idea:

1. Feed the neural network each one of the 60,000 premade handwritten images
2. Feed the neural network a testing image that it hasn’t previously seen and chart the result.

I’m going to be collecting the data in excel and decide what numbers it’s best and worst at guessing. As well as comparing the average percentage it gets correct compared to the letters and simple drawings.

<http://yann.lecun.com/exdb/mnist/>

## EMNIST Database

To be completed… A handwritten database I haven’t investigated yet

<https://www.nist.gov/itl/products-and-services/emnist-dataset>

# MNIST Database Testing

## Sample of a Random 50

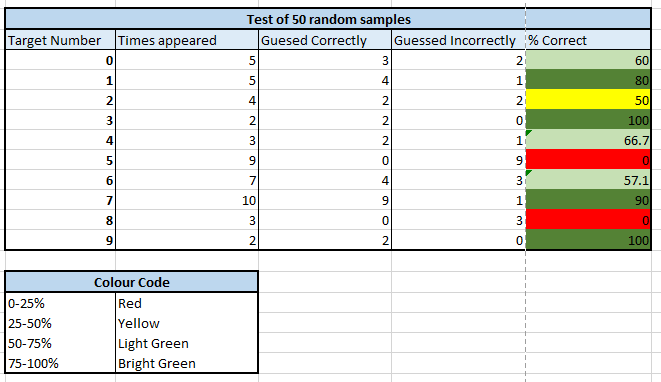


Figure - Source: My own testing

Some interesting notes:

* For some reason, the neural network really struggles to guess 8 and 5 correctly, with getting 0 correct out of 50 attempts
* The neural network got 9 and 3 100% of times, however I wouldn’t say this is reliable since 9 and 3 only got randomly picked twice.
* Out of all the data, I’d say the neural network is reliably best at guessing 7. Since out of all 10 times it was picked, it correctly guessed 9. This surprised me since 7 is very similar to 1 and I expected the neural network to confuse the two.
* I expected the neural network to correctly guess 0 and 3 since they are very distinct numbers. However, I assume it thought 8 was 0 as well as 6 since they both contain circles which could be mistaken for a 0.

## Test of each Number 5 Times

# Recourses Used (Credits)

A massive thank you to these recourses, since I they helped immensely.

|  |  |
| --- | --- |
| What it is | Direct link |
| An archive of a livestream by the university professor who runs The Coding Train YouTube channel. | [The Coding Train’s – Live Stream #120, MNIST and Pendulums](https://www.youtube.com/watch?v=KhogNPC24eI&t=5557s) |
| A video demonstrating how a neural network can solve XOR and why a simple perceptron can not | [The Coding Train’s – Coding Challenge #92: XOR Problem](https://www.youtube.com/watch?v=188B6k_F9jU) |
| This is my main recourse used in developing the neural network. It’s an 8.5-hour long series where he goes through the maths, and code required for this project. And while I wish it was as simple as copying the code, I decided to challenge myself and code mine in a different language. The outcome is very similar however. | [The Coding Train’s – Neural Network – The Nature of Code Series](https://www.youtube.com/watch?v=XJ7HLz9VYz0&list=PLRqwX-V7Uu6aCibgK1PTWWu9by6XFdCfh) |
| Great help with understanding the theory behind it | [3Blue1Brown - But What is a Neural Network? | Deep learning, chapter 1](https://www.youtube.com/watch?v=aircAruvnKk) |
| ^ | [3Blue1Brown - Gradient descent, how neural networks learn | Deep learning, chapter 2](https://www.youtube.com/watch?v=IHZwWFHWa-w) |
| This is the first recourse I bought to ensure I stuck with the project. It’s a book I bough on my kindle which I gradually read through quarantine. It’s split into 3 sections; the theory, a worked example, and using it. The worked example was written in a programming language called Python, since mine is written in C# it didn’t help massively but gave me the general workflow for how it should work. The theory section is what inspired The Coding Train’s series, so it was crucial for me to read it | [Tariq Rashid - Make your own Neural Network](https://www.amazon.com/Make-Your-Own-Neural-Network-ebook/dp/B01EER4Z4G/ref=as_li_ss_tl?ie=UTF8&qid=1498492463&sr=8-1&keywords=make+your+own+neural+network&linkCode=sl1&tag=natureofcode-20&linkId=0d10fdc485d6452bb7fc2b62ab4ffd31) |
| The MNIST Database, used for the handwritten numbers | <http://yann.lecun.com/exdb/mnist/> |
| The EMNIST Database, used for the handwritten letters | <https://www.nist.gov/itl/products-and-services/emnist-dataset> |