Neural networks and their basic components

Introduction

In the previous article "Introducing Deep Learning for Computer Vision" we explored the world of image classifiers, deep learning and neural networks from a foundational level and this article serves as a continuation on the fundamentals built on the previous article. In this article, we will look into neural networks in a much more detailed manner and build further on our foundations. Our goal at the end of this article is to help you create an image classification model using the MNIST dataset.

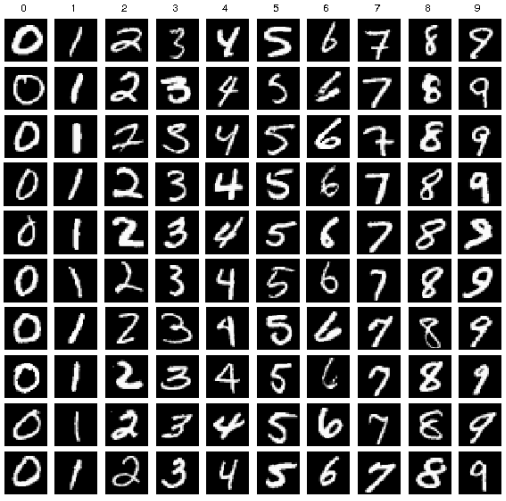
By the end of this article, you'll not only be able to understand what neural networks are but also be able to implement some key components such as activation functions, loss functions and optimization algorithms. We focus more on a hands-on approach for this article with the help of our Google Colab where we will be working on the MNIST dataset, allowing you to understand how these elements work together to train a model which can detect images.

Introducing Key Datasets in Deep Learning

Before we start building our neural network, it's important to familiarise ourselves with some of the most popular and commonly used datasets in deep learning. These datasets have played a crucial role in shaping the field of computer vision and are often used to test and benchmark model performance

1. MNIST

The MNIST dataset [Modified National Institute of Standards and Technology] consists of 70,000 images of handwritten digits (0-9). These images are 28x28 grayscale pixels, making it a simple yet powerful dataset for starting off and learning about deep learning.

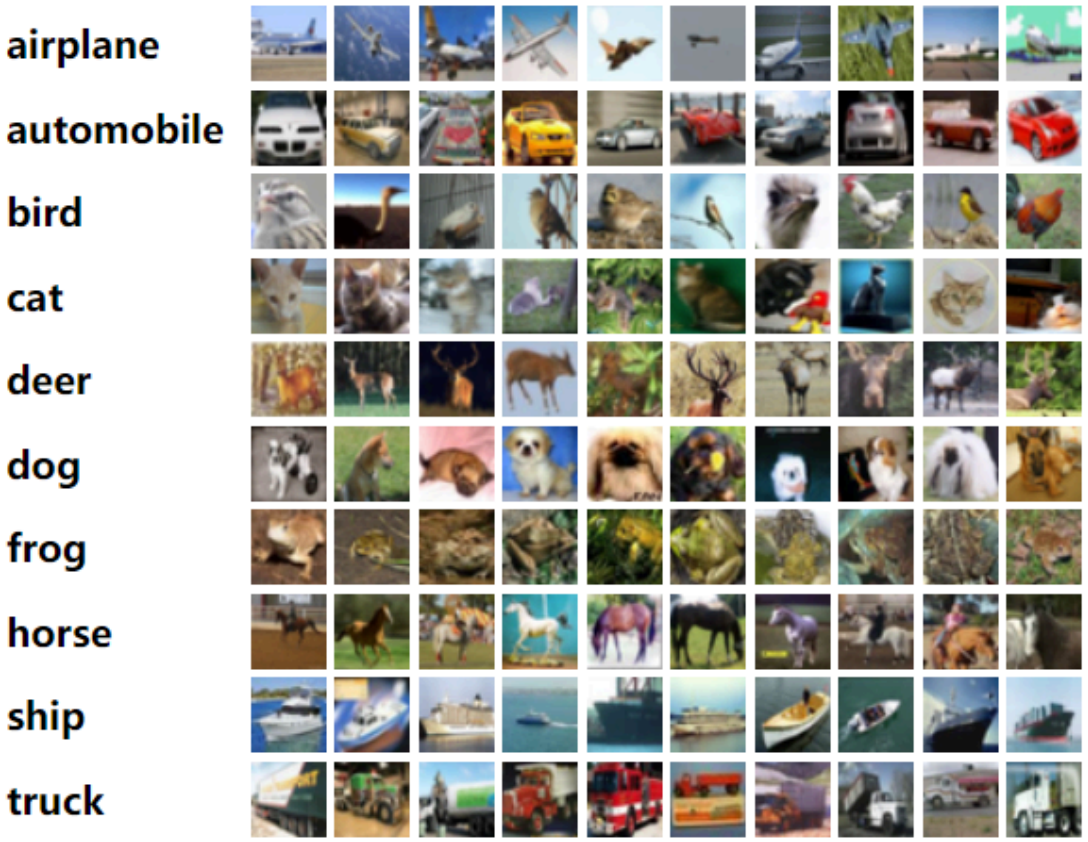


In this article we will focus more on the MNIST dataset and build up on the number classification problem discussed in the previous article. I've stated below the reason why we chose the MNIST dataset for our example.

- **Historical Importance:** MNIST was introduced by Yann LeCun, Corinna Cortes and Christopher Burges in the late 1990s. It has since become the go-to dataset for testing neural network architectures.
- **Simplicity:** it consists of small grayscale images (28x28 pixels)

2. CIFAR-10

The CIFAR-10 dataset consists of 60,000 coloured images of 32x32 pixels. These images are divided into 10 classes (aeroplanes, cars, birds, cats, deers, dogs, frogs, horses, ships and trucks). There are 6000 images per class of which 5000 comprise the training set while 1000 comprise the test set. This dataset was introduced by Alex Krizhevsky, Geoffrey Hinton in 2009. This dataset is a step above in complexity to the MNIST due to its inclusion of colour and a broader range of objects.



3. CIFAR-100

It's similar to the CIFAR-10 but with 100 image classes. Both CIFAR-10 and CIFAR-100 have the same amount of images being 60,000 but since the CIFAR-100 is divided into 100 classes there are only 600 images of each class, 500 images per class in the training set and 100 images in the testing set. CIFAR-100 is significantly more challenging due to its larger variety of categories with fewer images to learn from in each class.

4. ImageNet

ImageNet was introduced in 2009, it contains over 14 million labelled images across thousands of categories. It is one of the largest datasets and has helped drive crazy advances in deep learning.

Data Splitting - Training, Test, and Validation Sets

In the first article *"Introduction to Deep Learning for Computer Vision"* we read about the test set, training set and the validation set. Here I will brief you about the importance of Data Splitting.

Lets understand why we split our data, splitting data ensures that the model that we build will be able to generalize the data. The training set is used to train the model by adjusting weights based on some error that the model computes (This is the "loss") and the model learns the pattern from this dataset. Validation set is used to tune the "hyperparameters" (basically the number of layers in the neural network and learning rate) and monitor the performance of our model. Validation set essentially helps us grade how well our model is being trained by the training set. The Test Set consists of data that is never seen/computed upon by the model. The importance of test set being unvisited is very important since it will help us evaluate our models performance in the real world, this provides an unbiased estimate of how well our model will truly perform.

Now… What happens if we don't split our data? If you don't split your data the model will essentially "memorize" the data instead of learning from the data, this results in extremely high accuracy while testing but very poor performance when applied in the real world. And without a test set we cannot measure how well our model performs and hence it becomes impossible to trust the models predictions in the real world. An easier to understand example for not splitting your data is, imagine you give a student the same set of questions every time, this student might score 100% every time because they are having to answer the same questions, if a different question is provided the student might struggle to provide the correct answer.

Therefore, essential data splitting is essential for evaluating model performance. The general data split often depends upon the data size but an average estimate would be as follows:

- Training 70-80% (usually divided into further sets, we will see this in our upcoming MNIST follow along project)
- Validation: 10-15%
- Test: 10-15%

Google Colab follow along project:

Awesome!!! Now we are ready to work on your first deep learning project! You remember the problem statement we introduced in the first article? Where we tried to understand how a computer is able to differentiate between different handwriting for numbers, here in this project we will implement this very system. Everything in the below project is very beginner friendly with explanations for each step, at the end of this project you will be able to write numbers in your own handwriting and detect them! Have fun with your first neural network project!

MNIST Digit Recognition Project.ipynb