**Image Classification Project Report**

**Brook Raindle - 19039840**

**Problem Statement**

The goal of this project is to develop a Convolutional neural network (CNN) that is capable of distinguishing the difference between images, which will then classify the images into one of 4 categories: people, buildings, food and landscapes. This adaptation explores the advantages and disadvantages of altering the training data size to see how it effects the models predictions, and to determine if it is a worthwhile process adding more training data.

**Description of the dataset**

Two CNN’s were developed, one with a training size of exactly 375 training images, named ‘Small Data’, and one with around 2000 images, named ‘Large Data’. All images were sourced directly from google images, using a browser extension that downloads all images. Each of the images vary in size. File wise, it ranges from between 10 Kb to 4,500 Kb. Since there is a linear relationship between the amount of images used in the dataset and the size of memory storage required, this is immediately a benefit of using a smaller data set.

A Major disadvantage of sourcing data directly from google includes impure data, images that may be labelled as people, could simply be a logo, or a closeup of a specific part of human, maybe even a diagram or graph, all of these possibilities contribute to a less effective dataset as it trains the model to recognise the wrong features. The benefits of using the Small Dataset, I was able to quickly glance over the data and remove anything impure, however, with a much large dataset, it would have taken me significantly longer to remove all the lower quality data from each set.

**Approach taken**

**Data Preprocessing –** Before transforming the data into the shape readable from the CNN, it is important to clean the data. It was decided that, in order to keep as many variables constant, I would only read in ‘jpg, jpeg’ files. It then tests each image to make sure it is readable. To prepare each image, the ‘Keras’ package was used, more specifically the ‘image\_dataset\_from\_directory’ method, which automatically build a dataset and transforms images to fixed parameters. This project transforms each image to a 256 by 256 pixel, RGB image, with an interpolation of ‘nearest’. Each image is then put through the ‘map()’ method, which scales what originally a 0-255 value, to 0-1, resulting in a float these images have also been separated into shuffled batches of 32, where all the CNN knows is the label attached to the image.

**Data segmentation –** 3 sets of data would be required, training, validation and test data, since I have varying sizes of datasets, the data is segmented using percentages, 70% training data, 20% validation data, and 10% testing data.

**CNN Architecture –** the CNN is a set of Sequential layers of inputs and outputs, in total, this system has 7 layers. 2 Convolutional layers, the first of which has 16 filters, with a kernel size of 3x3, and a ‘Relu’ activation function. The second convolutional layer is the same, except it had 32 filters.

Separating the convolutional Layers, is 2 pooling layers, which reduce the dimensions of the features, which reduces the computational requirements for following layers, 2 of these are used, one after each convolutional layer.

Then, a Flatten layer follows, which restructures the 2 dimensional image into a 1 dimensional array.

Finally, 2 ‘Dense’ layers are added, these layers extract features from the flattened input. The first dense layer uses 256 neurons, with the ‘Relu’ activation function. The second dense layer uses 4 neurons (equivalent to the amount of classes that are being identified) with a soft-max activation function, this is done to convert the extracted features into probabilities relating to the classes.

**Training –** Firstly, the model has to be compiled, this model utilizes the ‘Adam’ optimiser, with a learning rate of 0.001. it also calculates the probabilistic losses using the Sparse Categorical Cross entropy class from Keras, which allows for several classes to be categorized as a result from the training. The performance that is being judges is Accuracy, the metrics parameter.

A Callback variable is also integrated using the Keras ‘ModelCheckpoint()’ method that makes a save file for the model.

Fitting the model using the Keras fit() method , taking my training data segmented earlier, as the training parameter, an epoch value (times it will iterate through the training), in this model, the epochs remain at 25, providing enough time for the CNN to gain a competent understanding of the data. it also requests a callback, which was provided earlier, implementing it this way has benefits as it recalls the ‘ModelCheckpoint()’ method every epoch, so even if the training is interrupted, the most recently run epoch will be saved. And finally validation data to check the accuracy of the training with.

**Performance Analysis**

A table with numbers and text

Description automatically generated

Evaluating this data, the most significantly noticeable change is the data size, 539% larger data set so the model has over 5X as much information to learn from, this comes with many advantages, as it provides the model with more scenarios and combinations of features that can be learnt to recognise and classify different objects. A major downfall of this experiment is that the variation of change in data set sizes ranges very high, to a 4.8X increase, all the way to a 6.7X increase, this could cause an uneven bias in the training results, fixing this issue and making the increase linear between classes would improve this experiment and give more definite results towards the effectiveness of varying data sizes.

Another significant impact it has had is on the time elapsed during the training phase, a 428% increase in time taken could be hugely impactful in a larger scale experiment, where instead of using thousands of images, it uses data sets millions big, the time taken to train the model might not prove worthwhile, and at first glance, you could argue that has occurred in this experiment as the Validation accuracy reaches 1.00 off the Small Data, but only reaches 0.84 from the Large Data. However, this is actually not the case, as with the small data, there isn’t enough for the model to learn from, so it thinks its certain its correct all of the time, whereas a model that is provided with 5.4X as many variations of features, has learned more, but finds more coinciding similarities between classes. This causes the accuracy to fall, which isn’t always a bad thing, as the model was still able to recognise a higher portion of features to the small data model. Evidence of this was found when evaluating the model with Test data. the Small data model would predict images and only get 2/4 correct, often confusing classes with each other, However the Large Data model had a higher portion of correct answers often reaching (3-4)/4 correct classifications.

To conclude, if you are working against time, and have limited computational power, it could be worth using a smaller data set, as they still provide relatively successful results, but in a fraction of the time of the Large Datasets. However if it is accuracy you are looking for, Larger data sets provide a more robust, reliable prediction from the model trained, it just requires more time and a more powerful GPU / TPU.

**List of constants:**

Hardware -

* GPU – Intel UHD Graphics 620 Integrated.
* RAM - 16GB RAM
* CPU – Intel i7-8550U
* OS – Windows 10

Programming –

* Batch size = 32
* Epochs = 25
* Optimizer = ‘Adam’, Learning rate = 0.001
* Losses = Sparse Categorical Cross Entropy
* Extension types = ‘JPEG’,’JPG’

**Imports:**

* TensorFlow
* OS
* Cv2
* ImgHDR
* NumPy
* matplotlib
* Keras (several smaller imports from the Keras module)

**Image Sources:**

All images were sourced directly from Google, as written in the dataset description, they may include duplicates between ‘Small Data’ and ‘Large Data’ providers of images to the google page largely contains the likes of ‘Trip Advisor’, ‘Shutterstock’, ‘The Guardian’ and several other image based websites.

**Learning resources:**

<https://www.youtube.com/watch?v=jztwpsIzEGc&t=3610s> – Nicholas Renotte CNN Image classifier tutorial, Youtube.

<https://keras.io/api> - Keras API containing information on all Keras methods and features.