**Machine Learning Report**

**Project 1 - Classify Images of Road and Traffic Signs**

Group Name: Dan & Ethan

Daniel Schellekens: 3900792

Ethan Rayner: s3902240

In order to best solve the classification of images of road signs into both type and shape, we employed a number of different machine learning techniques in order to find the one that is most appropriate. To begin with, we decided to set our target accuracy to **(something)**%. We are able justify using accuracy to measure performance since our data has no label imbalance and the task falls under multi-class classification.

We began by splitting our data into two halves for training and validation.

**Approach 1 - Multilayer Perceptron**

Our initial approach was to create an algorithm utilising multilayer perceptron (MLP). To begin with, we set up the layers that we will be using for our algorithm using tensorflow’s sequential API. By flattening our input layer (the 28x28 images) into a two-dimensional array, we are then able to pass this to three dense layers, the first and second layers contain 112 neurons, the third layer returns an array of length equal to the number of output classes we had (5 for shapes and 16 for types). We then compiled our model with the **‘adam or SGD’** optimiser and the tensorflow categorical cross entropy loss function, measuring the model by categorical accuracy (how often predictions match one-hot labels). We were able to use tensorflow’s inbuilt imageDataGenerator to load our data to memory in ‘batches’ efficiently and then use fit\_generator() from keras to train our model. We decided on an epoch value of 50 due to this being roughly where the model began to level off.

**Approach 2 - Convolutional Neural Network (CNN)**

Our next approach was to use a convolutional neural network with a VGG architecture. As the name suggests this approach uses convolution and pooling layers to help our model learn to classify images by extracting their features. Once again we compiled our model with the ‘adam’ optimiser and the tensorflow categorical cross entropy loss function, measuring the model by categorical accuracy. From here we flattened the layers and passed them into two dense layers. The first layer was made up of 112 neurons and the second layer returns an array equal to our number of output classes (5 for shapes and 16 for types). Once again, we used tensorflow’s inbuilt imageDataGenerator to load our data to memory in ‘batches’ efficiently and then use fit\_generator() from keras to train our model. It was then a matter of deciding on a number of epochs, we decided on an epoch value of 30 as this was where the model began to level off for both our training and validation data.