Machine Learning Engineer Nanodegree

Proposal: State Farm Distracted Driver Detection

1. Domain Background

The domain of this problem is computer vision. Computer vision is a branch of machine learning concerned with the automatic extraction, analysis and understanding of useful information from a single image or a sequence of images (BMVA, n.d.). Computer vision began in the 1960's, when a person named Larry Roberts wrote his PhD thesis on the possibility of extracting 3D details and information from 2D images(T.S. Huang, n.d.). In the 70's, some progress was made on the interpretation of 2d images to 3d images (Hari Narayanan, et al, n.d.). In the 80's, optical character recognition systems that recognize letters, symbols and numbers were used in several industries (Quick history, n.d.). In the 90's, new applications of computer vision were possible as computers became more powerful and common (Quick history, n.d.). In the 2000's, computer vision was used to process large datasets, videos and could understand motion, patterns and predict outcomes (Hari Narayanan, et al, n.d.). This problem interests me because I want to have a better understanding of computer vision classification problems specifically human actions and the different approaches I could use in order to solve it.

2. Problem Statement

In order to improve the alarming statistics states in the project overview section, innovative methods should be tested. One such method would be to develop an algorithm to detect drivers engaging in distracted behaviours by feeding it 2D dashboard camera images. This algorithm can then be used as an API in a device to classify the driver's behaviour by checking if they are driving attentively, wearing their seatbelt and remind them if they are not.

The dataset provided by State Farm consists of images, which means that the most efficient way to tackle the problem at hand is to develop a deep convolutional neural network model and then train it on a training subset of the dataset with the objective to optimize a certain evaluation metric. The model will then be tested on a testing subset of the dataset and evaluated.

2.1. Metrics

The models will be evaluated using the multi-class logarithmic loss between the predicted class of the image and the actual class. The formula of the evaluation metric is:

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} log(p_{ij})$$

Where N is the number of images in the dataset, M is the number of image class labels, log is the natural logarithm, y is 1 if observation i belongs to class j and 0 otherwise, and p is the predicted probability that observation i belongs to class j.

3. Datasets and Inputs

The dataset being considered is the one provided in the Kaggle competition. It contains 2 folders, one which contains the training images and the other which contains the test images. There are approximately 2,500 images for each of the 10 classes in the train folder and 102,149 images for the test data. All images are colored and 640 x 480 in size. The images capture the driver from a side-view dashboard camera. I will make use of the training images to train a model and test images to test the model.

4. Solution Statement

For this computer vision multi-class classification problem, working with Convolutional Neural Networks would be a good idea because CNN's are good at classifying images and are known to give results with high accuracy. Keras application models with their pre-trained weights could reduce the time it takes to train while still yielding good results. But before we do that, pre-processing the images will be necessary. Reducing image size and dividing the images into the RGB channels could make processing of the images more manageable. Once the model is fit, we will need to predict the labels of the test set to determine which of 10 categories each picture belongs to. The results, evaluation metric and benchmark model will then tell us if/where we still need to improve the model.

CNN differs from other neural networks by arranging its neurons in three dimensions (height, width, depth). The main layer types used in CNN are:

1. Convolutional layer: computes the output of a neuron that is connected to the region in the input by applying a dot product between its weights and the region's values.
2. Pooling layer: performs a downsampling operation along the spatial dimensions (i.e. height and width).
3. Fully connected layer: computes the class score.

5. Benchmark Model

The benchmark model for this project was a simple CNN model made up of three blocks each block starts with a convolutional layer applying a relu activation followed by a max-pooling layer. The data is then flattened using a flatten layer followed by a dropout layer followed by a dense layer followed by

another dropout layer and finally another dense layer of 10 outputs corresponding to the number of classes and a softmax activation. The model was compiled using a rmsprop optimizer and a categorical cross-entropy loss function. The model was trained for 5 epochs. The model performed not very well achieving accuracy of 16% .

6. Evaluation Metrics

I will make use of the Multi-class logarithmic loss. This metric is used in many computer vision classification problems because it measures the accuracy of a classifier by penalizing false classifications. It is also a good metric for this problem because in order to calculate log-loss, the classifier must assign a probability to each class rather than yielding the most likely class(). If we know the distribution of response values, we could use this information to improve our model and dataset. In our problem, the accuracy is very important, so we will need the information. Using other metrics such as classification accuracy, only provides us with the accuracy. We need to compare our results to a benchmark, so it is best to stick with the metric used by the Kaggle competition.

7. Project Design

A solution I have thought of is :

- Step 1, input the training data.
- Step 2, use a subset of the data to train.
- Step 3, Reduce and scale images from the dataset to a more manageable size. .
- Step 4, make the CNN architecture but most optimally make use of the keras application and pre-trained model weights : VGG-16.
- Step 5 , use another pre-trained CNN : InceptionV3.
- Step 6 , I will then test with a subset of the test dataset.
- Step 6 , compare the results of the 2 models and consider the best of them.