ECSE 415 – Intro to Computer Vision

Final Project Report

Group 13

Antonios Valkanas, Bryan Jay, Ethan Lague, Ezz Aboulezz and Priscilla Ip

2. Classification

Dataset

The training classification images of the MIO-TCD car dataset was the source of all images used to train and evaluate the classifier. The images were read in grayscale and by inspection, the training images were low quality, so a smoothing filter was applied. A simple 3x3 box filter was chosen after testing the accuracy on a linear SVM against a Gaussian filter and median filter. Then, the images were resized to squares of 64x64 because of the feature extraction method described in the following section. Each label consisted of 1,751 images as that was the number of samples available in the smallest category. In order to avoid bias in the classifier, the images used were selected randomly. This distribution of samples was chosen because increasing the number of samples in other labels biases the classifier and would artificially increase the average performance. This will be explained in detail in the Classifier Evaluation section.

Table 1: Number of Samples by label

Label	Number of Samples Used	Number of Available Samples
Articulated Truck	1,751	10,346
Background	1,751	160,000
Bicycle	1,751	2,284
Bus	1,751	10,316
Car	1,751	260,518
Motorcycle	1,751	1,982
Non-motorized Vehicle	1,751	1,751
Pedestrian	1,751	6,262
Pickup Truck	1,751	50,906
Single Unit Truck	1,751	5,120
Work Van	1,751	9,679

Feature Extraction

The feature extraction method used in this project consisted of computing the histogram of oriented gradients (HoG) descriptors for each image followed by principle component analysis to reduce the number of features. The HoG method works by computing the distribution of intensity gradients and depends on the idea that local appearance and shape within an image can be described by the distribution of intensity gradients or edge directions. As there is no compensation for scale, the training images were resized to squares of the same size. The cell size, block size and number of bins were chosen based on a test comparing the precision on a grid of parameters. The results are shown in the table below. Therefore, the chosen parameters were: a cell size of (4,4), a block size of (4,4) and 8 bins. With the selected parameters, there were 2048 components so PCA was used to remove components containing little information and to greatly reduce runtime. There was a trade-off between the number of training images used, runtime, performance and number of components used. In the end, 508 components were chosen that represent over 98% of the total variance explained. This can be explained in the variance curve illustrated in Figure 1. Initially, approximately 1400 components were used in order to keep 99.9% of total variance explained but the runtime was doubled, and the performance was not significantly better.

Table 2: Precision value when tested against cell and block size and number of bins

Cell size:	2	2	4	4	4	6	6	6	2
Block size:	2	4	2	4	6	4	6	2	6
Nb Bins	Precision								
6	0.47	0.55	0.54	0.61	0.6	0.58	0.54	0.57	0.59
8	0.5	0.55	0.54	0.63	0.61	0.6	0.55	0.58	0.6
10	0.53	0.57	0.54	0.63	0.62	0.59	0.56	0.58	0.6
12	0.53	0.57	0.55	0.63	0.62	0.58	0.56	0.58	0.6
14	0.54	0.57	0.54	0.62	0.62	0.59	0.56	0.57	0.6
16	0.56	0.57	0.55	0.55	0.55	0.55	0.55	0.57	0.6
18	0.57	0.57	0.56	0.56	0.56	0.56	0.56	0.58	0.61
20	0.56	0.57	0.55	0.55	0.55	0.55	0.55	0.58	0.6

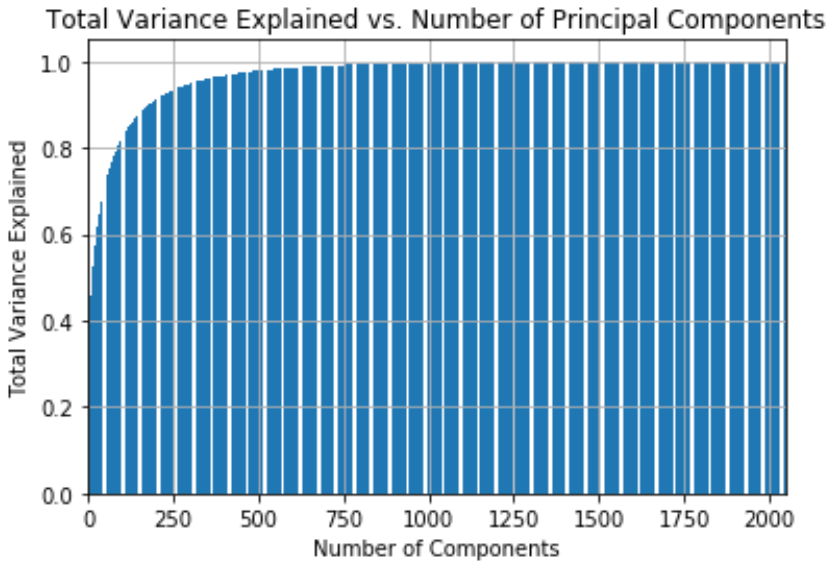

Figure 1: Total Variance Explained vs. Number of Principal Components

Model Selection

The models considered for the classifier were the from the sklearn library and included: LinearSVC, LogisticRegression and KNeighborsClassifier. With initial testing using default parameters of the models, it was found that the KNeighborsClassifier performed better on average but came with a hefty price tag in terms of runtime. After considering the tight deadline of the project, it was determined that performing a grid search to tune the parameters and performing cross validation on the model would not be feasible however, it could be interesting to look into in the future. Therefore, the LinearSVC and LogisticRegression models were chosen.

Using the GridSearchCV method from the sklearn library, several values (1, 10, 100, 1000) of the ‘C’ parameter of the models were tested to find the highest performing. For LinearSVC, the best performing value was 1 and for LogisticRegression, it was 10.

K-fold Cross-Validation

One of the key performance evaluation metrics used was k-fold cross-validation with 10 folds. The method works by splitting the data into k disjoint subsets where the model is run k times such that each fold is the validation set once. The model is trained on the remaining k-1 subsets and the desired performance metrics from each fold calculated. The bar in Figure 2 represents the total amount of data. If 10 folds are used, the data is divided as shown. In the fifth fold, for example, the model is trained on Subsets 1-4 and 6-10 and then used to predict the labels of the data in Subset 5. As the model is a classifier, the division of data into subsets is stratified such that the percentage of samples in each class are preserved. From k-fold cross-validation, the accuracy, precision and recall were obtained from the validation subsets and are presented in the next section.

Figure 2: k-fold cross-validation example graphic

Classifier Evaluation

The performance metrics obtained from k-fold cross-validation are displayed in Table 3. The average accuracy, precision and recall are consistent in the testing. This is because the number of samples used to train and test the model were the same. As the number of samples per class vary greatly in this dataset, it can be argued that the precision and recall are more representative of the performance than the accuracy. This is due to the fact that if the entire dataset was used, the test samples would largely be made up of car and background images. These would likely be predicted correctly since there was also a large amount of training data for those classes and the test samples from the other classes would often get mistaken for car or background images. This is seen in the confusion matrix of the CNN classifier.

The confusion matrices from the LinearSVC and LogisticRegression classifiers are shown in Figures 2 and 3. The class that is the most difficult for both classifiers is the non-motorized vehicle. They are often confused with buses. Looking at the actual images, this makes sense because many of the non-motorized vehicles are trailers or other long, rectangular vehicles that have the same general shape as buses. The most commonly misidentified label for LinearSVC is a single-unit truck mistaken for an articulated mistaken at 16%. For LogisticRegression, it is bicycles mistaken for motorcycles also at 16%. Again, these are mistakes that can be expected with the small amount of training data and the similarity between those particular classes. These values would likely improve given more training instances.

Table 3: Performance metrics from k-fold cross-validation

Model	Average Accuracy	Accuracy Std. Dev.	Average Precision	Average Recall
LinearSVC	62.51%	1.17%	62.37%	62.49%
LogisticRegression	63.13%	1.21%	63.11%	63.11%

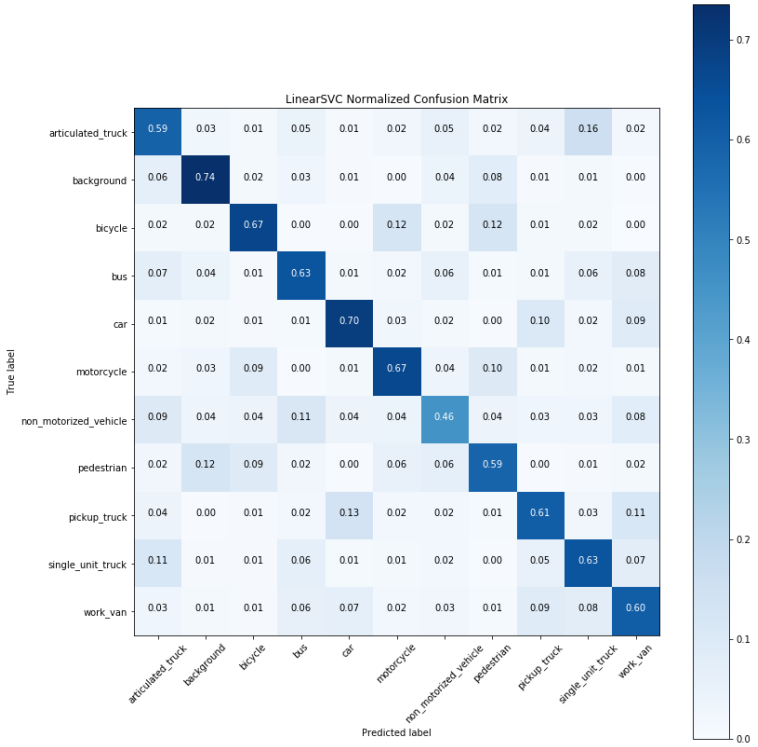

Figure 2: Confusion Matrix of LinearSVC Predictions from Test Set

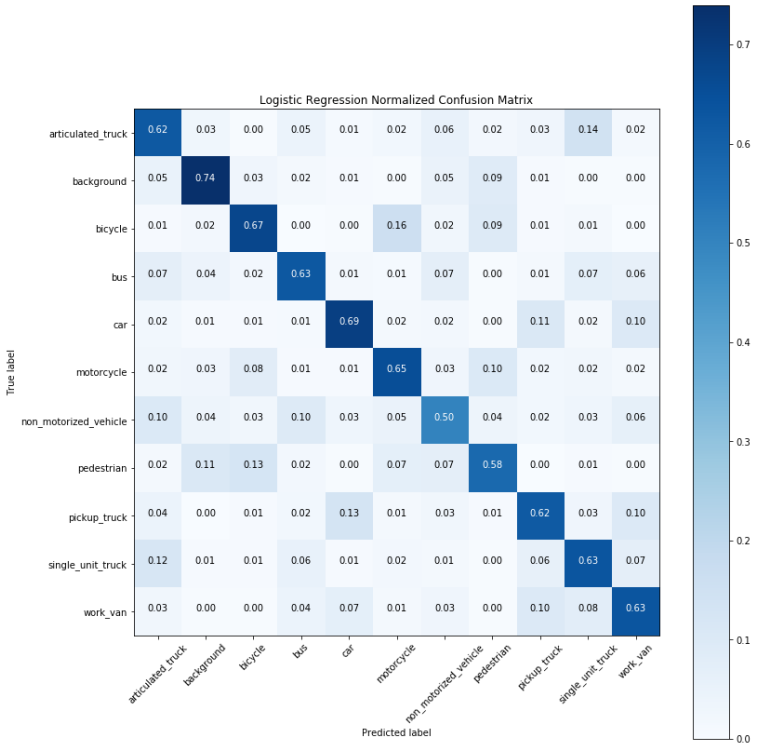

Figure 3: Confusion Matrix of LogisticRegression Predictions from Test Set

3. Localization and Classification

Dataset

The training localization images of the MIO-TCD car dataset was the source of all images used to evaluate the localizer. As done with the classification dataset, the images were converted to grayscale and filtered using a 3x3 box filter. 100 sample images were used to test and evaluate the localizer. The size of the bounding box outputs varied as explained in the following section.

Localization Approach

The main idea behind the approach was to extract windows from the images and use the CNN classifier described in the next section in order to determine whether the window contain an object. As the objects vary in size within an image, multiple window sizes were needed. This was implemented by resizing the window to a fraction of the image dimensions between runs. Four window sizes were used at 1/2, 1/4, 1/6 and 1/12 of the original image size. The step sizes between windows were also adjusted based on the image size accordingly. Although the dimensions of the objects vary, the windows were squares for simplicity and consistency. Varying the dimensions of the window would introduce many more false positives as the current implementation as the current implementation already has many. After all the windows with their corresponding coordinates were found, each window would be classified using the CNN to check whether it is an object or part of the background.

Cross-Validation Approach

The cross-validation approach taken was similar to the k-fold method described above expect 5 folds were used.

Evaluation and Interpretation of Results

For a specific test image, the DICE coefficients for the predicted vs. true bounding boxes were computed. The number of objects in the image according to the ground truth was 3 and the localizer detected 84 objects. The average DICE coefficient (taking the highest value when comparing each computed box to the ground truth boxes) was 6.83%. This is a very low value and indicates that the localizer is performing extremely poorly. This can also be een in the DICE coefficient plot. The average across folds was merely 5.12% and the average across sets is shown in the table below.

When comparing the performance of the localizer + classifier against the classification data + classifier, there is no competition. The poor performance of the localizer justifies the large gap between performance metrics of the classifier and classification data versus the classifier using localized data. The reduced performance of our classifier can be seen in table 5.

Table 4: DICE coefficients across validation sets

Validation Set	1	2	3	4	5
Avg. DICE Coeff.	6.41%	5.67%	4.79%	4.13%	4.61%
Std. Dev.	9.95%	10.19%	10.54%	8.98%	9.54%

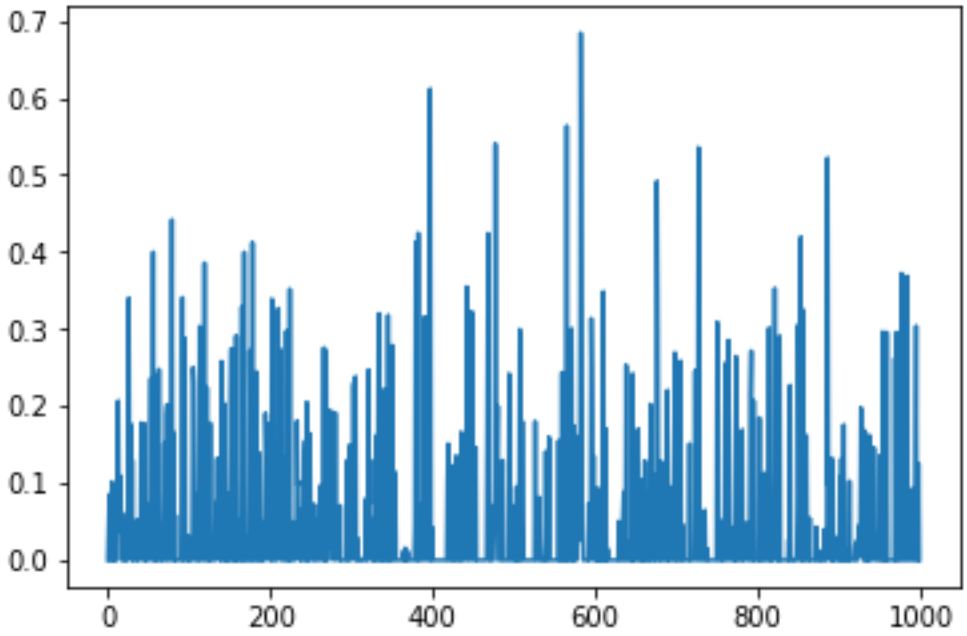

Figure 4: DICE Coefficients across Validation Set

Table 5: DICE coefficients across validation sets

Validation Set	1	2	3	4	5
Avg. Accuracy	64.9%	62.8%	56.8%	58.45%	64.4%
Avg. Recall	45.0%	47.5%	49.1%	43.6%	56.5%
Avg. Precision	32.5%	35.8%	28.9%	28.1%	35.8%

4. Deep Learning Bonus

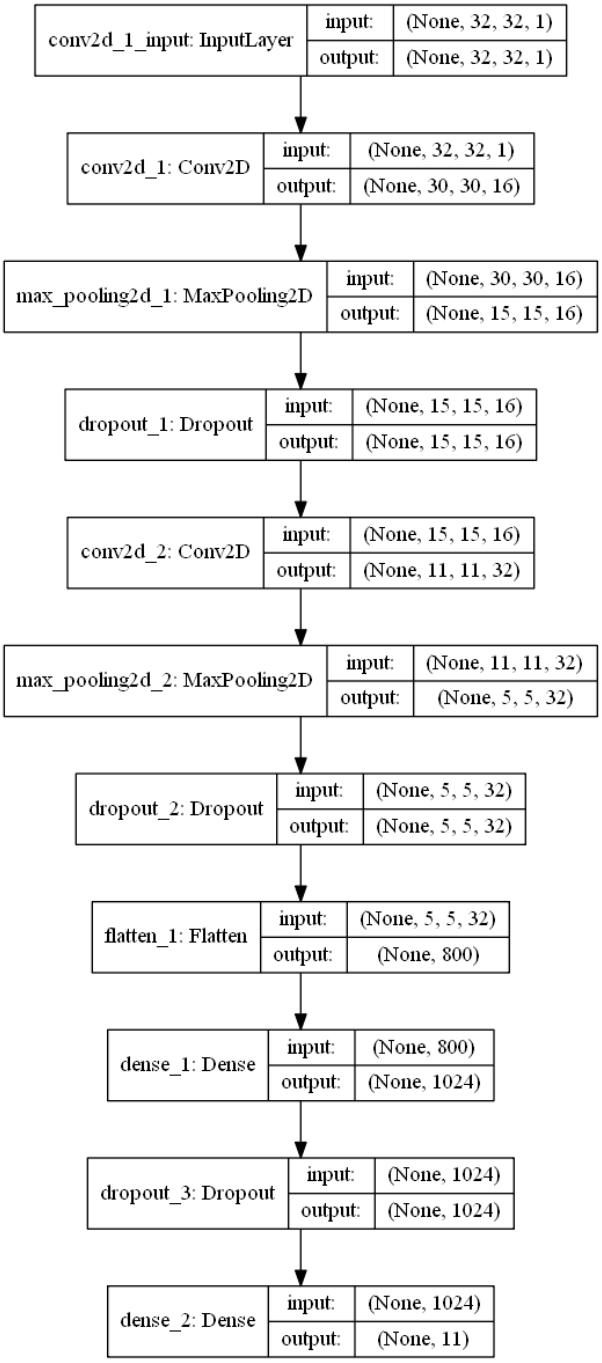

Figure 5: Schematic of Architecture

Evaluation of Performance

The accuracy of predictions on the test set was 91.68% and the confusion matrix is shown below. The results shown in the confusion matrix are to be expected. The classes with lower numbers of samples, have poorer prediction accuracy and are often misidentified as a car which has up to 200 times as many samples. Since the background and car classes have much more

training and test data, the average prediction accuracy is skewed to be much higher than if the test data had an equal number of samples from each class. It would be interesting to see how the classifier would perform given a equally large number of samples from each class however, there is not enough samples in this dataset to try this.

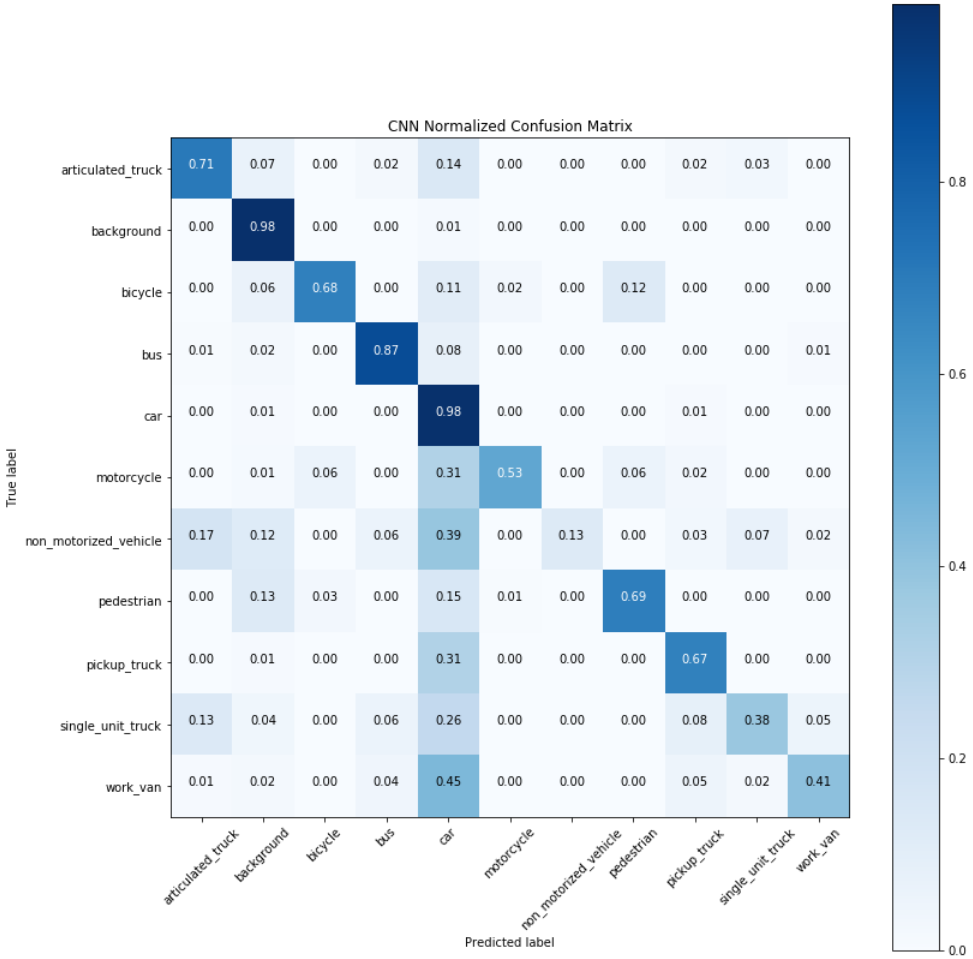

Figure 6: Confusion Matrix of CNN Predictions from Test Set

Description of Validation

While the performance of the network on the validation set was calculated at the end of each epoch, the model does not use that to adjust the weights. However, being able to track the validation loss and accuracy at each epoch allows us to observe how well the model can generalize and if it overfits or not. In this case we found that the model performs better on the validation set than the training set which illustrates exactly that, as given by the figures below.

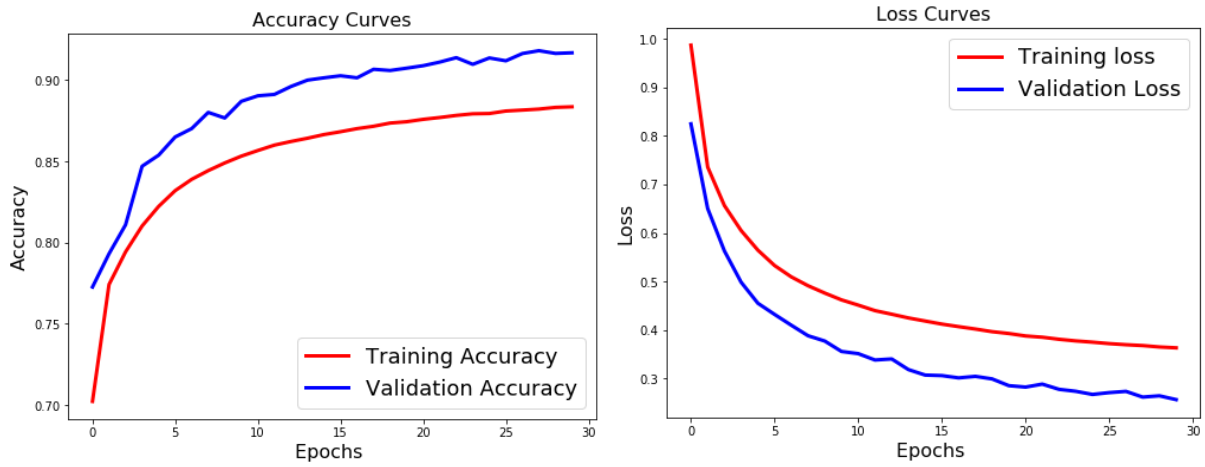

Figure 7: Accuracy and Loss Curves

Comparison with the methods in Section 2

The average prediction accuracy obtained from the CNN is much greater than from the LinearSVC and LogisticRegression models discussed previously. This is likely due to the large number of test images from the background and car classes that inflate the average. The HoG descriptors with PCA used with the non-deep learning methods was not needed with the CNN which made the runtime much quicker and allowed the usage of the entire dataset.