
McGill University

ECSE 415

Introduction To Computer Vision

# Final Project


| *Authors:* | *Student Number:* |
|---|---|
| John Wu | 260612056 |
| Frank Ye | 260689448 |
| Tiffany Wang | 260684152 |
| Nathan Clairmonte | 260673075 |
| Ali Shobeiri | 260665549 |


November 30, 2018

# 1 Introduction

Traffic cameras serve as an effective way to survey traffic congestion over time and at a distance. As the popularity of traffic cameras grow, traffic footage has become more abundant. Traffic footage provides a convenient method to collect images as traffic data. Typically, these images in traffic datasets contain different classes of objects, e.g. as cars, trucks, pedestrians, etc. In this report, we will summarize our approach to the tasks of detecting and localizing these classes of objects in the MIO-TCD dataset [1]. Specifically, we will go over our methodology and results for data preprocessing, localization and classification methods. Finally, we will conclude the report tackling localization and classification on the dataset with our deep learning model.

# 2 The Classification Task

The classification challenge requires training a Support Vector Machine (SVM) in addition to another non-deep learning algorithm to classify a set of objects belonging to 11 different categories. The secondary non-deep learning algorithm chosen is logistic regression.

## 2.1 The Dataset

The classification task is performed on the MIO-TCD-Classification Dataset, which contains 648 959 images, with each image containing an object belonging exclusively to one of the 11 categories found in Table 1.

| Category Name | Number of images |
|---|---|
| Articulated truck | 12 933 |
| Bicycle | 2 855 |
| Bus | 12 895 |
| Car | 325 649 |
| Motorcycle | 2 477 |
| Non-motorized vehicle | 2 189 |
| Pedestrian | 7 827 |
| Pickup truck | 63 633 |
| Single unit truck | 6 400 |
| Work van | 12 101 |
| Background | 200 000 |
| Total | 648 959 |

Table 1: MIO-TCD Classification challenge dataset category breakdown

In Figure 1, we show a sample of the training data. In terms of size, the images are all of different dimensions and were directly cropped out from the MIO-TCD-Localization Dataset. The largest single dimension for an image in the dataset is found to be 720x720.

Figure 1: Sample training image - articulated truck

The dataset is evaluated using 10-fold cross validation. This means that for every

iteration, the model is split into 90% training and 10% testing images.

## 2.2 Feature Extraction

In order to compute distinct features on the classification dataset, we made a few design choices in terms of data preprocessing and our feature extraction algorithm. These are outlined in the following section.

### 2.2.1 Data Preprocessing

In order to reduce computation time in training our classification models, all images in the dataset are resized to dimensions 64x64. This decision is a tradeoff we made while trying to balance our computational limitations and memory considerations, as well as our overall model performance (in terms of accuracy, precision, recall, etc.).

In fact, we first experimented with re-sizing images up to the maximum image size found in the dataset which was found to be 720x720 pixels (up-sampling approach). With this method, we quickly found that our training speed significantly decreased due to an increased number of computations and due to memory limitations. Operating on images of this size decreased the number of samples we could train our classifier with at any given time. With this in mind, we then repeated the process, resizing images to a size of 128x128. However, training times were again too slow and memory limits were still being reached.

Finally, we settled on images of size 64x64, which allowed us to use a larger batch size of up to 10 000 images. Hence, we could simultaneously load up a single batch of 10 000 images at once, compute their HoG features, and feed them into our SVM classifier (using the partial_fit parameter in scikit-learn [4], we could run the fit method our model in batches). This strategy lets us train our classifier more efficiently. In the following, we describe our HoG feature extraction approach on the preprocessed images.

### 2.2.2 HoG Features

We first analyze where the differences lie between the samples in the training set to choose an appropriate feature extraction method. Below, we list a few of the differences found between the scenarios in which an image was captured:

- Lighting conditions: some images were captured in broad daylight, while others were captured with very poor lighting conditions.
- Orientation: depending on the position of the camera, the front/back of each vehicle detected would be facing a different direction.
- Scale: the distance at which an object was detected by a camera varied greatly in each sample.
- Aspect Ratio: this is related to the camera's observation point and type of lens used to detect vehicles. Due to variations in these two parameters, the aspect ratio of detected objects of the same class varied between samples.

Based on the different sources of potential noise above, we proceed to compute HoG features on our preprocessed data. HoG feature descriptors have been used for over a decade in many object detection methods (including facial detection), so we decided to investigate the performance feeding HoG features into our two classification methods. In addition, this feature extraction method is able to deal with the majority of variations listed above, while being easily implemented in OpenCV.

In fact, while computing HoG feature descriptors, although gradient magnitudes are affected by changes in illumination, the orientation of gradients is not. Hence, a normalization factor can render HoG features robust to changes in illumination. In terms of scale and aspect, since we re-size all samples to a size of 64x64, we simply have to apply the same re-sizing transformation at test time to obtain similar HoG features for objects of the same class. As for aspect ratio, although we do distort objects by forcing them to be of dimension 64x64 in the preprocessing phase, we distort the aspect ratio in a uniform fashion. Although this re-sizing is mostly done out of convenience and for reducing computation time, if we uniformly distort aspect ratio for objects of the same class, then we should learn similar HoG features for them as well. This leaves the problem of orientation that is not explicitly dealt with using our feature extraction method, but we have noticed that most cameras have similar observation points. Hence, we can capture a good range of slightly rotated gradients regardless.

Sample HoG results with the selected parameters are shown in Appendix A.

### 2.2.3 HoG Hyper parameter tuning

The block size determines the amount of significance allocated to local pixels. A larger block size helps suppressing local illumination changes. The cell size determines the scale of the spacial information extracted. The number of orientation corresponds to the number of bins, and number of feature vectors. These three

parameters are fine tuned to get the best feature extraction results. The feature extraction hyper parameter tuning was done over 10% of the dataset, sampled at randomly without replacement to ensure the maintenance of the class balance. Once extracted, the features are trained in a SVM classifier, and we selected the parameters that gave us the highest prediction F1 score.

The best parameters were:

| Orientation | Block size | Cell size |
|---|---|---|
| 8 | 5 | 4 |

## 2.3 Classification Methodology

### 2.3.1 K-Fold Cross Validation

Cross validation is an evaluation technique often used on predictive models. It operates by partitioning a dataset into a training set to train the model, and a test set to evaluate the model.

In k-fold cross validation, the original dataset is randomly partitioned into k subsamples of the same size. Of these, one sub-sample is left as the test set while the other k-1 sub-samples are used to train the model. The cross validation process is then repeated k times, each time using a different of the k sub-samples as the test set. The k results obtained from this process can then be averaged to produce a single estimation value of the model evaluation. This method is advantageous because every sample in the original dataset is used for both training and testing, and is used for testing exactly once only [2].

For classification problems, it is common to use k-fold cross validation in which the number of folds is selected such that each fold (sub-sample) contains approximately the same class weights [3].

### 2.3.2 Classifier Hyperparameter Tuning

The hyperparameter tuning was done using the built-in scikit-learn GridSearchCV function [4]. We used a predefined train and test set for each search to ensure consistency in the data.

We found that the best parameters are the defaults onces. We then used them during training. The results of both SVM and Logistic Regression classifiers are found in the follow sections.

### 2.3.3 Support Vector Machines (SVM)

SVM is a discriminative classifier that separates the data using hyperplanes in a kernelized space and was used for our first classifier. We used the SGD Classifier from scikit-learn [4] to complete this part of the project. The SGDClassifier implements regularized linear models, one of which is a linear SVM with stochastic gradient descent (SGD) learning. The gradient of the loss is estimated each sample at a time and the model is updated along the way. In this manner, SGDClassifier allows for online learning, or rather, allows us to fit the classifier on batches of the data at a time. This is useful in our use case as our dataset is large and memory limitations is a consideration.

| K Fold | Accuracy |
|---|---|
| 1 | 0.7955 |
| 2 | 0.7941 |
| 3 | 0.8037 |
| 4 | 0.7878 |
| 5 | 0.7998 |
| 6 | 0.7933 |
| 7 | 0.8060 |
| 8 | 0.8046 |
| 9 | 0.8052 |
| 10 | 0.8011 |
| Average | 0.7990 |
| Std. Dev | 0.006126 |

Table 2: K-Fold accuracy, average and standard deviation for SVM

The results given in Table 2 highlight the performance our SVM classifier when trained on different k fold partitions of the dataset. Our average accuracy is found to be 0.799 with a fairly low standard deviation of 0.006126. This means that for all the folds considered, our classifier is able to accurately predict the label 79.9% of the time. Furthermore, we can see in Table 3 that our classifier scores fairly high in regards to precision and recall. This means that our classifier has relatively few false positives and false negatives. The results of our precision and recall are in line with our accuracy, as they indicate an overall good classifier performance. In general, precision is able to measure the number of false positives and recall is able to measure the number of false negatives.

| Avg. Precision | Precision Std. Dev | Avg. Recall | Recall Std. Dev |
|---|---|---|---|
| 0.785 | 0.00849 | 0.799 | 0.00738 |

Table 3: Average Precision and Std. Dev, Average Recall and Recall Std. Dev for SVM

From Figure 2 we can see that our classifier performs well for objects that are distinct. This can be seen from our confusion matrix as the background and bus classes have very high scores. These two classes are distinct and do not resemble any other class even after resizing. However, our classifier struggles on images that are similar to one another. For example, looking at the car label, we see that the classifier continually misclassifies a work van as a car, this also happens very often for pickup trucks and single unit trucks. This error can be explained through our preprocessing. As we resized the images to 64x64 we lost a lot of dimensionality information for the images. This means that although a truck might be longer and taller than a typical car, after being resized down, it matches very closely with a car and thus would have a similar HoG. This result could explainwhy we have such low scores for work van and single unit trucks.

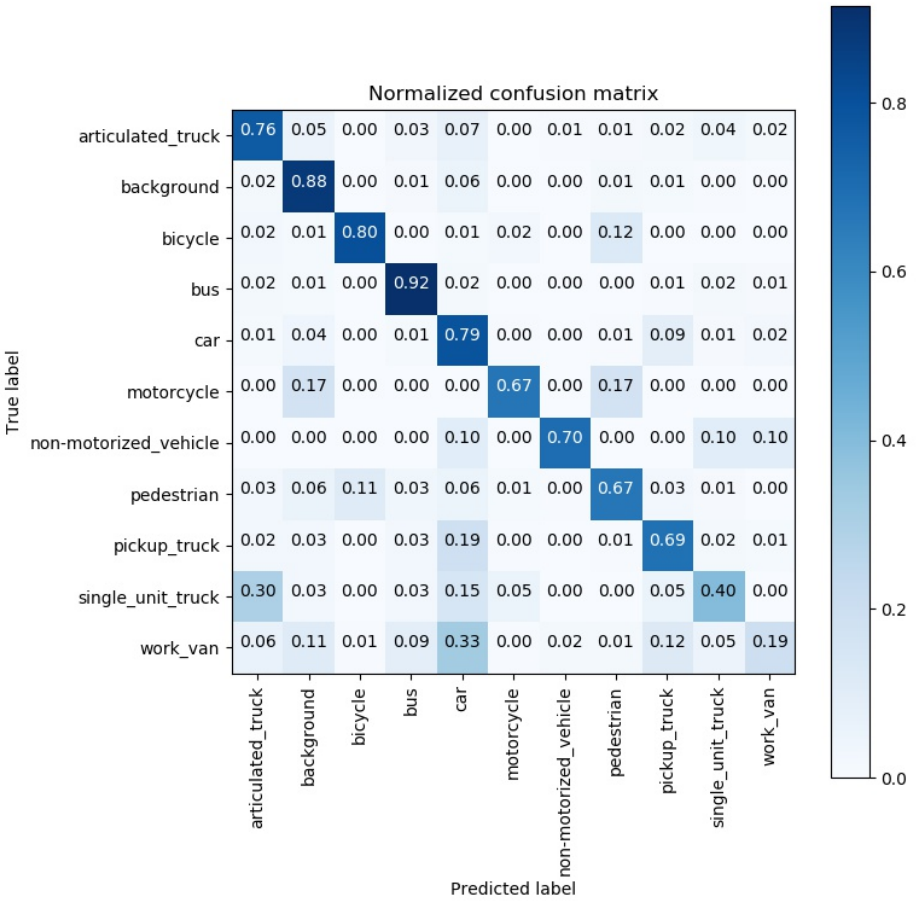



Figure 2: SVM Confusion matrix from k = 3

### 2.3.4 Logistic Regression

Logistic regression is a discriminative classifier that is used typically on binary classification tasks but can also be extended to multiclass cases as well. Logistic regression is based off the sigmoid function,

$$\sigma\theta(x) = \frac{1}{1 + e^{-\theta^T x}} \tag{1}$$

Logistic regression uses the sigmoid function to transform its output into a probability that can predict the probability of a class, for a multiclass problem logistic regression is run on each class and a probability is generated for each class. This

approach is called one vs the rest as one class is compared against all the remaining classes in the dataset. The class with highest probability is passed along as the output label.

We implemented logistic regression using SGD Classifier from scikit-learn [4]. The SGDClassifier implements regularized linear models, that can be used for online learning. One of the available linear models is a logistic regression. We use this class as the dataset is large and memory limitations is a consideration.

| K Fold | Accuracy |
|---|---|
| 1 | 0.797 |
| 2 | 0.791 |
| 3 | 0.809 |
| 4 | 0.804 |
| 5 | 0.804 |
| 6 | 0.806 |
| 7 | 0.802 |
| 8 | 0.803 |
| 9 | 0.804 |
| 10 | 0.805 |
| Average | 0.803 |
| Std. Dev | 0.004839 |

Table 4: K Fold accuracy, average and standard deviation for Logistic Regression

The results given in Table 4 highlight the performance our logistic regression classifier when trained different k fold partitions of the dataset. Our average accuracy is found to be 0.803 with a very low standard deviation of 0.004839. This much lower than std. dev. value seen in Table 2 for the SVM. For all the folds considered, our classifier is able to accurately predict the label 80.3% of the time. Furthermore, we can see in Table 5 that our classifier scores fairly high in regards to precision and recall. This means that our classifier has relatively few false positives and false negatives. The results of our precision and recall are in line with our accuracy, as they indicate an overall good classifier performance. In general, precision is able to measure the number of false positives and recall is able to measure the number of false negatives.

| Avg. Precision | Precision Std. Dev | Avg. Recall | Recall Std. Dev |
|---|---|---|---|
| 0.803 | 0.00667 | 0.803 | 0.006749 |

Table 5: Logistic Regression

From Figure 9 we can see that similar to our SVM, our logistic classifier performs well for objects that are distinct. Our confusion matrix is also very similar to the confusion matrix observed for the SVM. This can be seen from our confusion matrix as the background and bus classes have very high scores. As we noticed before, the classifier seems to struggle when classifying images that are not distinct when they are resized down to 64x64. The worst performing classes are non motorized vehicle, motorcycle and work van. For each of these classes, it is theorized that they perform poorly as they are small portions of the dataset relative to other classes as seen in Table 1 and because rescaling the images down probably further reduces the ability of the classifier to distinguish between these more obscure classes and their more common counterparts.

### 2.3.5 Classifier Evaluation

In general, we noticed that our SVM seemed to have better performance when identifying non-distinct values. This can be seen by directly comparing Figure 2 and Figure 9. However, for the classes our logistic regression performed well in, it outperformed the SVM. Due to this, the logistic regression was able to attain a greater accuracy, precision and recall than the SVM.

For both our classifiers, we were able to score highly for accuracy, percision and for recall. For us, since we have a relatively imbalanced dataset, we would find that percision and recall are better measures for our classifier performance. In general, if classes are imbalanced, accuracy is not a good measure of classifier performance. For example, if we tackle the task of detecting fraudulent credit card transactions, we might have a dataset with 97% of samples containing legitimate transactions and 3% fraudulent samples. Some classification algorithms tend to show a bias for the majority class, treating the minority class as noise. Hypothetically, if all fraudulent transactions are misclassified, and all legitimate transactions are well classified, it might be tempting to claim that 97% accuracy means that the classifier performance is very good. However, that would defeat the purpose of fraud detection, as none of the fraudulent observations are well classified.

Precision would measure the number of correctly identified observations as positive out of total items identified as positive. The formula is shown below, and in the

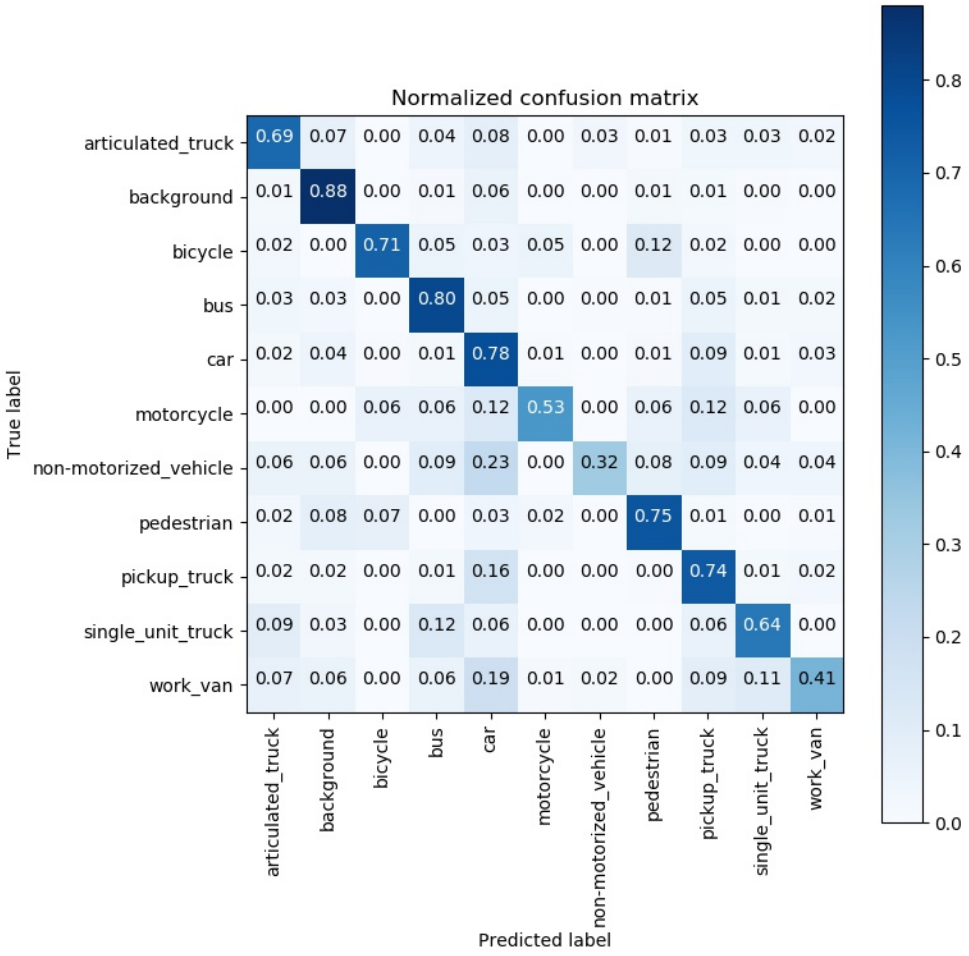



Figure 3: Logistic Regression Confusion matrix from k = 5

situation above we would get 0% precision (assuming that fraudulent samples are of the positive class).

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

Similarly, recall would measure the number of correctly identified samples as positive out of total actual positives. The formula is given below, and we would get a recall of 0% for the case described above.

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

Where

- TP = True Positive
- FP = False Positive
- FN = False Negative

# 3 Localization and Classificatrion

The localization challenge requires training a model that can localize the coordinates of objects in a photo and then classify it using the classifier trained in the previous section. The same SVM classifier as above was used but was retrained using the new class labels and cropped out images from the localization dataset.

## 3.1 Dataset

The localization challenge is completed on the the MIO-TCD-Localization Dataset. [1]. There are a total of 137 743 training images that contain one or more objects in the following categories:

| Category Name | Number of Appearances |
|---|---|
| Articulated truck | 9 301 |
| Bicycle | 2 260 |
| Bus | 10 598 |
| Car | 233 497 |
| Motorcycle | 1 837 |
| Motorized vehicle | 25 845 |
| Non-motorized vehicle | 2 350 |
| Pedestrian | 7 128 |
| Pickup truck | 44 283 |
| Single unit truck | 5 741 |
| Work van | 8 709 |
| Total | 351 549 |

Table 6: MIO-TED Localization challenge dataset category breakdown

All images vary in size and are accompanied by a ground truth list of detected objects and their labels. Unlike the classification challenge dataset, the labels do not include the *background* label, while adding a *motorized vehicle* label.

## 3.2 Using YOLO as Our Localization Algorithm

For our localization, a YOLOv3 (You Only Look Once V3) [5] implementation was used. YOLOv3 is a state-of-the-art of deep learning algorithm which uses a single neural network to predict both bounding boxes and class probabilities for objects in an image, in a single evaluation. This allows the prediction to be evaluated with global context from the entire image at test time. Using the given labelled data, custom weights were trained on the YOLOv3 architecture. The exact implementation will be further discussed in the bonus section for using deep learning approaches.

Since it was required to use the classifier in the previous section for predicting the labels of the object, only the localization feature of the YOLO model will be used for this challenge. However, the same YOLOv3 model will be used for the deep learning component (both localization and classification functions will be used).

## 3.3 DICE Coefficient

Another metric used for evaluating the YOLOv3 model was the DICE coefficient for the predicted vs. true bounding boxes. The DICE coefficient is a metric that compares the similarity between two datasets. The DICE coefficient of two sets $X$ and $Y$ can be defined as follows:

$$DICE = \frac{2|X \cap Y|}{|X|+|Y|} \tag{4}$$

When being applied to boolean data, using the definition of true positive (TP), false positive (FP), and false negative (FN), it can be written as follows:

$$DICE = \frac{2TP}{2TP + FP + FN} \tag{5}$$

For 25 000 iterations, the DICE coefficient for the predicted vs. true bounding boxes was computed to be 0.7113. The distribution of DICE coefficients over the validation sets was also computed and is shown in Table 7. We motivate our table displaying our DICE coefficient over the number of training iterations in the following section.

| Number of Iterations | DICE Coefficient |
|---|---|
| 5 000 | 0.5736 |
| 10 000 | 0.6674 |
| 15 000 | 0.6706 |
| 20 000 | 0.7611 |
| 25 000 | 0.7113 |

Table 7: Distribution of DICE coefficients

## 3.4 DICE Coefficient versus Number of Iterations

In order to evaluate our localizer, we use the following approach to k cross-fold validation. This alternative method is based off the inherent difference in the number of times a training sample is viewed by our neural network (YOLOv3) during the training phase. In fact, using a batch size of 64 per iteration, we end up viewing all the training samples multiple times over many iterations (1.6 million samples viewed in total). Hence, we choose to compute the DICE coefficient over every 5000 training iterations.

## 3.5 Defining Detection

The ground truth bounding boxes were relative to the original image dimensions, and therefore had variable sizes for their heights and widths. Thus, the ground truth bounding boxes were scaled in order to make them compatible with the scaled YOLOv3 training images. The maximum number of overlapping pixels between every ground truth bounding box and every predicted bounding box, as shown in Figure 4. If the number of overlapping pixels is greater than 70% of the number of pixels in the ground truth bounding box, then the predicted bounded box is considered a true positive.

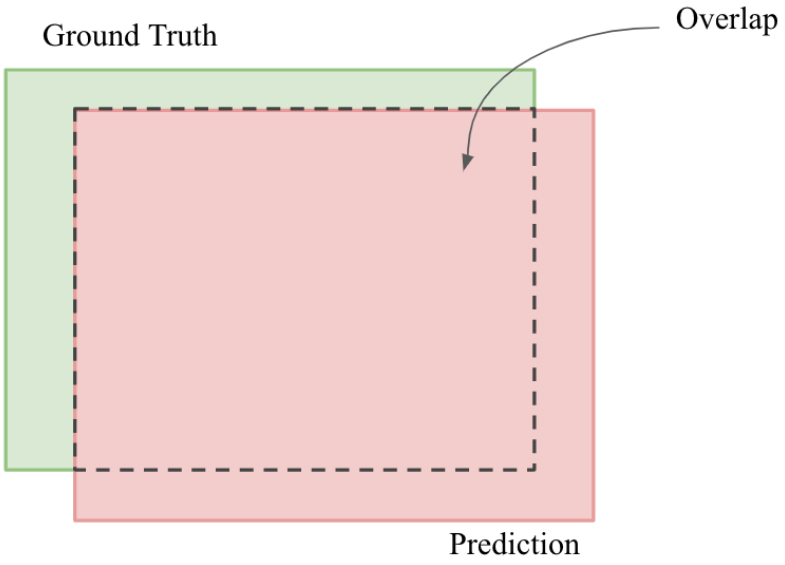



Figure 4: Ground truth bounding box vs. predicted bounding box

## 3.6 Evaluating Performance of Localizer + Classifier versus Classifier

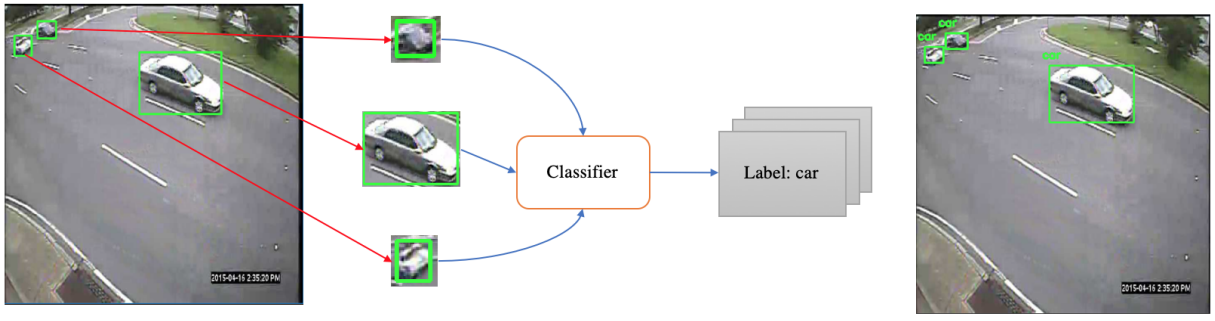



Figure 5: Workflow Depicting Sample Image Being Run through our Localization-Classification Architecture Using YOLO and SVM

When evaluating performance metrics such as accuracy, precision and recall between our classifier on the classification dataset, versus classification on our localized images using the previously trained SVM classifier, we note the following results:

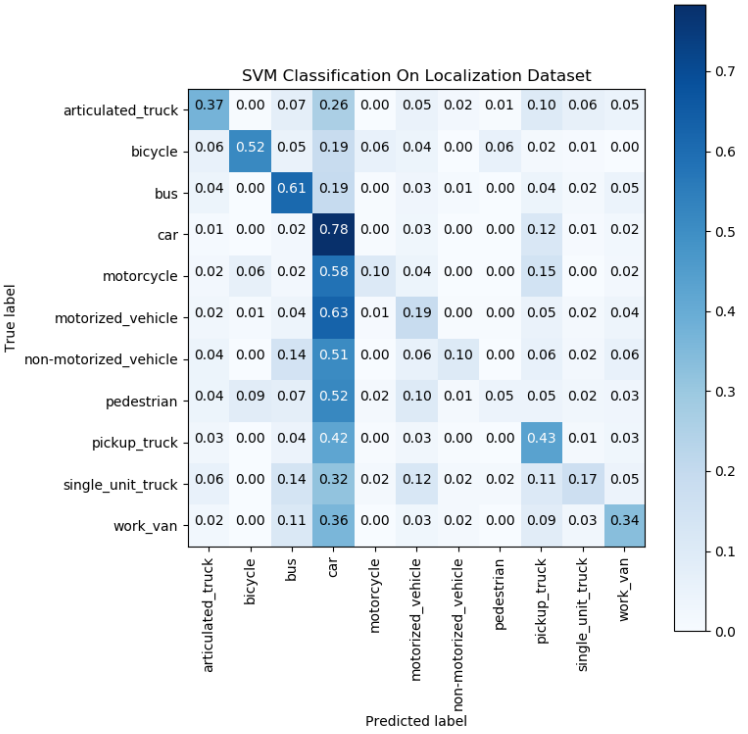



Figure 6: Confusion Matrix on the Localized Images - note the normalization of the matrix makes it possible to read accuracy values off the diagonal.

| | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| articulated_truck | 0.37 | 0.27 | 0.31 | 1 121 |
| bicycle | 0.28 | 0.23 | 0.25 | 182 |
| bus | 0.64 | 0.35 | 0.45 | 1 987 |
| car | 0.80 | 0.88 | 0.84 | 42 023 |
| motorcycle | 0.01 | 0.01 | 0.01 | 161 |
| motorized_vehicle | 0.19 | 0.06 | 0.09 | 1 938 |
| non-motorized_vehicle | 0.04 | 0.02 | 0.02 | 191 |
| pedestrian | 0.07 | 0.35 | 0.12 | 100 |
| pickup_truck | 0.47 | 0.46 | 0.47 | 9 117 |
| single_unit_truck | 0.22 | 0.02 | 0.04 | 592 |
| work_van | 0.26 | 0.08 | 0.13 | 1 572 |
| avg/total | 0.69 | 0.72 | 0.70 | 58 984 |

Table 8: SVM Classification Results on Localized Images

Upon inspection of the values along the diagonal of the normalized confusion matrix, we see that accuracy has significantly decreased from the 0.7790 average accuracy obtained on the classification task. Precision and recall performance are also worst on the localized dataset.

The significant performance drop across all the metrics can be explained in part by the addition of the very loosely defined motorized vehicle and non-motorized vehicle classes. For example, many vehicles in our data are actually a subclass of motorized vehicle. Hence, classes are no longer mutually exclusive and very difficult to distinguish from one another. This is demonstrated in the figure below, where an object detected on the left is classified as a car, whereas the ground truth classifies it as a motorized vehicle.

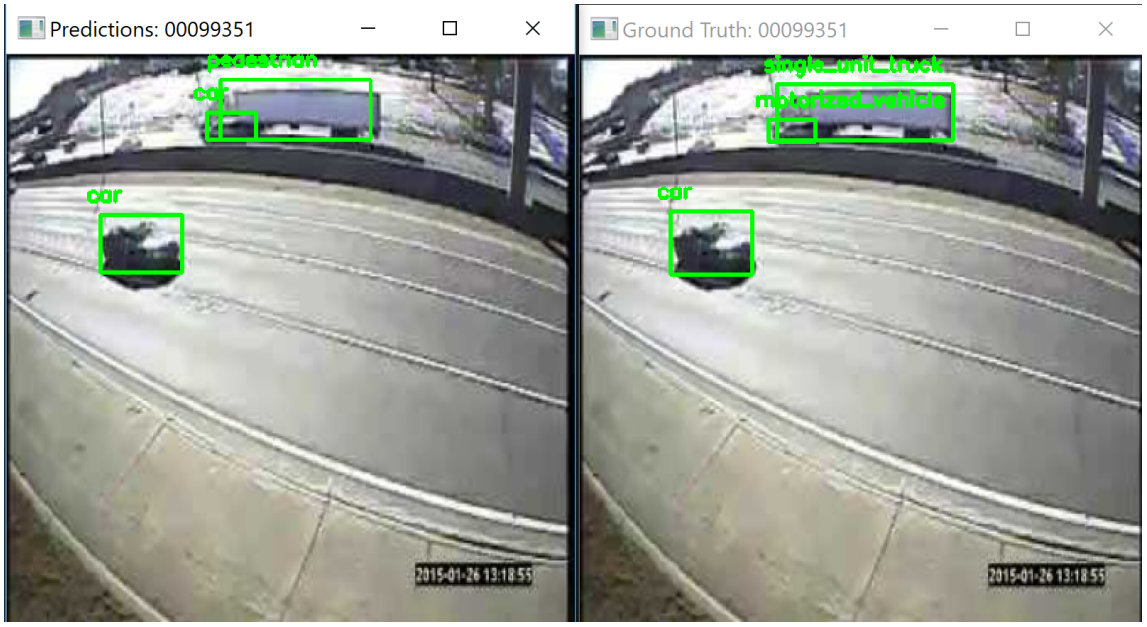



Figure 7: SVM Misclassification on Localized Dataset Due to the Introduction of Ambiguous Classes - on the left are our predictions using the SVM, and on the right are the ground truth values

Furthermore, we can see very poor classification results on the non-motorized vehicle class, single unit truck, motorcycle and pedestrian in the confusion matrix.

In terms of the background label, we decided not to include it in our classification model, since introducing a new class would add more noise, showing poorer results in our classification performance. However, if we were training another localizer, we could run candidate object locations through a classifier with the background class in order to see how many false positive objects are detected.

# 4 Deep Learning Approach

## 4.1 Architecture

We used YOLOv3 [6] as our classification method. This state-of-the-art neural network architecture was developed by Joseph Redmond specifically to perform both localization and classification tasks simultaneously using a single neural network, whereas other architectures would typically train a feature extraction network, e.g. VGG, ResNet, followed by a shallow task specific network for segmentation, classification, etc. In the following, we describe our deep learning classification approach, and report our performance results on the localization dataset.

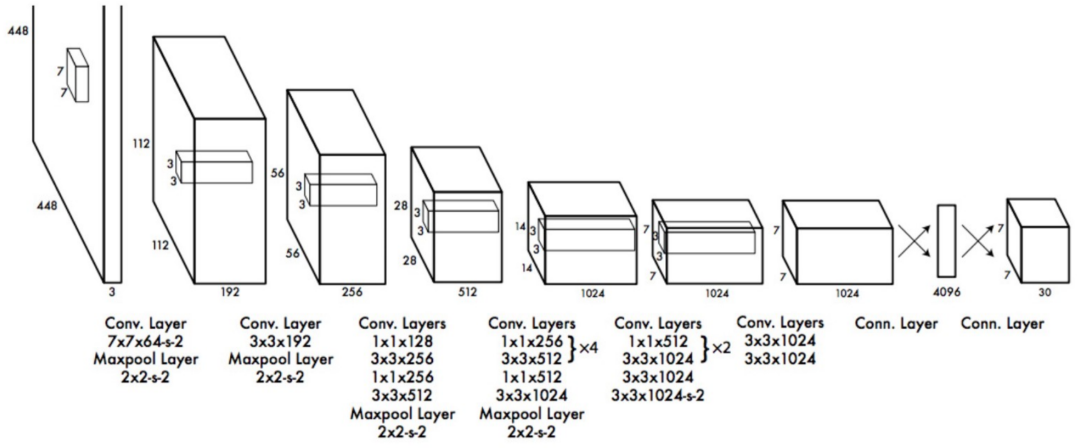



Figure 8: YOLO Macro-Architecture

## 4.2 Training and Validation

Although YOLO came with pre-trained weights for traffic objects, the model did not come with the correct class names nor was localization very accurate on our dataset. Therefore, we trained our own custom YOLO model from scratch using the labelled training data from MIO_TCD. This required modifying the labelled data into the correct format, re-configuring the model to work with the number of classes that we had, and training the model for 25,000 iterations with batch sizes of 64.

For validation of the performance, the data was split into train, validation, and test sets of 70%, 20%, and 10% respectively. The iterative improvements of the DICE scores was used to evaluate the performance of the localization, as explained and

reported in the localization section. Precision, Recall, and F1-Scores were used to evaluate the result of the classifications using the same methodology explained in the classification section.

## 4.3 Results

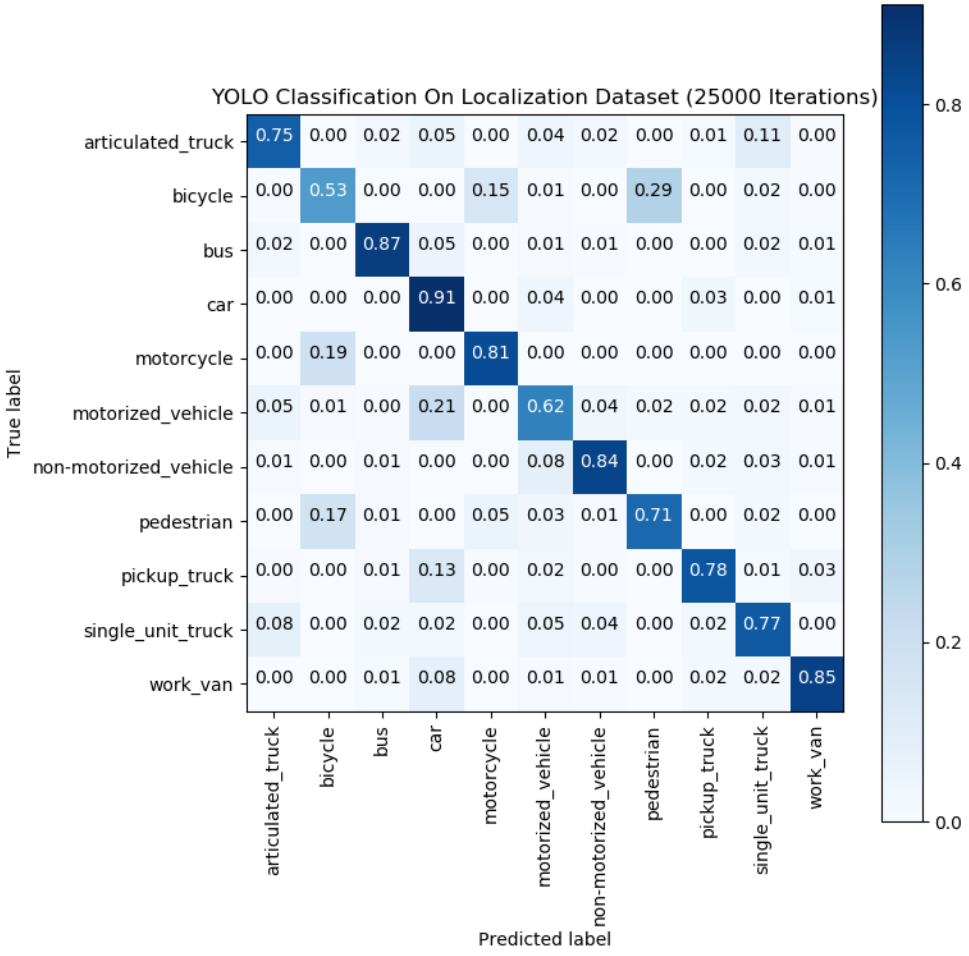
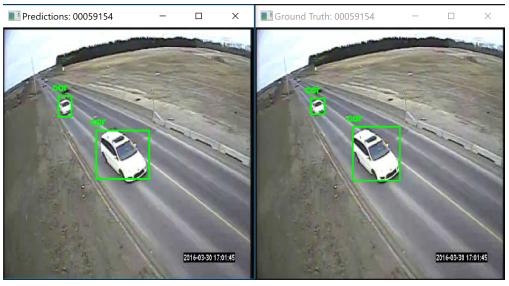


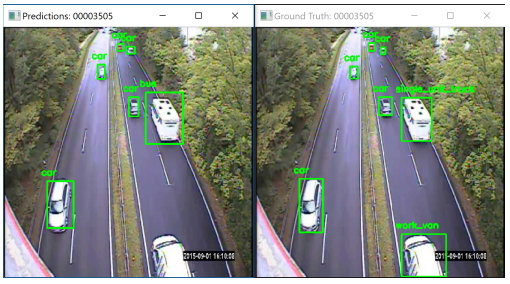

Figure 9: YOLO Classification Confusion Matrix

| | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| articulated_truck | 0.75 | 0.87 | 0.81 | 1 309 |
| bicycle | 0.53 | 0.85 | 0.65 | 272 |
| bus | 0.87 | 0.92 | 0.89 | 2 028 |
| car | 0.91 | 0.96 | 0.93 | 39 622 |
| motorcycle | 0.81 | 0.07 | 0.12 | 196 |
| motorized_vehicle | 0.62 | 0.08 | 0.14 | 2 076 |
| non-motorized_vehicle | 0.84 | 0.37 | 0.51 | 241 |
| pedestrian | 0.71 | 0.37 | 0.49 | 246 |
| pickup_truck | 0.78 | 0.84 | 0.81 | 9 101 |
| single_unit_truck | 0.77 | 0.38 | 0.50 | 662 |
| work_van | 0.85 | 0.46 | 0.60 | 1 556 |
| avg/total | 0.87 | 0.88 | 0.86 | 57 309 |

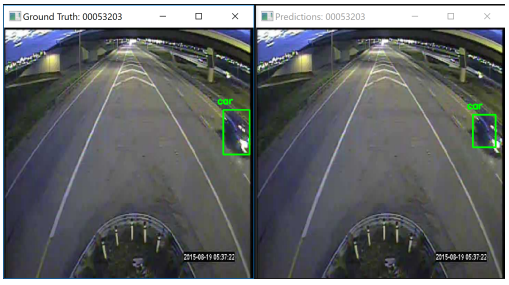
Table 9: YOLO Classifier Summary



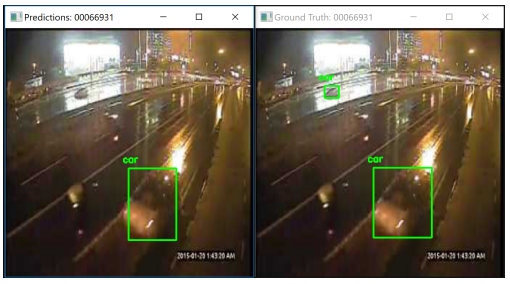

(a) Regular Scene




(b) Complex Scene




(c) Dark and distorted Scene




(d) Dark Scene

Figure 10: YOLO classification results under different scenarios

The YOLO implementation is capable of properly localizing and classifying pictures under extreme scenarios. The feature extraction component is invariant to lighting, rotation, aspect-ratio and perspective conditions.

### 4.4 Comparison with non deep-learning classifiers

In terms of performance, the deep-learning model performs significantly better than our SVM classifier. For example, the precision obtained is on average 18% higher. These results are not surprising, since the Linear SVM only works very well on linearly separable spaces. As mentioned earlier, the usage of overlapping classes (motorized vehicle vs car, truck, etc.) and lack of distinguishing features between certain classes imply that the classes are not linearly separable.

Alternatively, the convolutional and maxpooling layers in YOLO extract and localize the most important features within an image. A deeper neural network architecture helps YOLO learn more complex non-linear relationships between the HoG feature space and the outputs. This is due to the linear combination of non-linear activation functions within each hidden layer of the network. In brief, neural networks are known to be able to to efficiently approximate complex non-linear functions between an input space and an output space.

## 5 Constraints and Limitations

Due to the immense size of the datasets used in this project, most of the actions carried out on them were extremely computationally heavy. To help solve this issue, the data was split into batches for computation and Microsoft Azure cloud computing resources were used to train the models used. Specifically, two K80 GPUs were used for parallel distributed training (https://www.nvidia.com/en-gb/data-center/tesla-k80/).

## 6 Conclusion

In brief, we evaluated the performance of various classification models on the We extracted HoG features on the images in the MIO-TCD classfication and localization datasets, as well as the impact of introducing new classes. We then investigated localization and classification using a deep neural network architecture, YOLOv3. We found that YOLOv3 performs significantly better overall.

# References

[1] Z. Luo, F.B.Charron, C.Lemaire, J.Konrad, S.Li, A.Mishra, A. Achkar, J. Eichel, P-M Jodoin "MIO-TCD: A new benchmark dataset for vehicle classification and localization" in press at *IEEE Transactions on Image Processing*, 2018

[2] Kohavi, Ron. "A study of cross-validation and bootstrap for accuracy estimation and model selection." Ijcai. Vol. 14. No. 2. 1995.

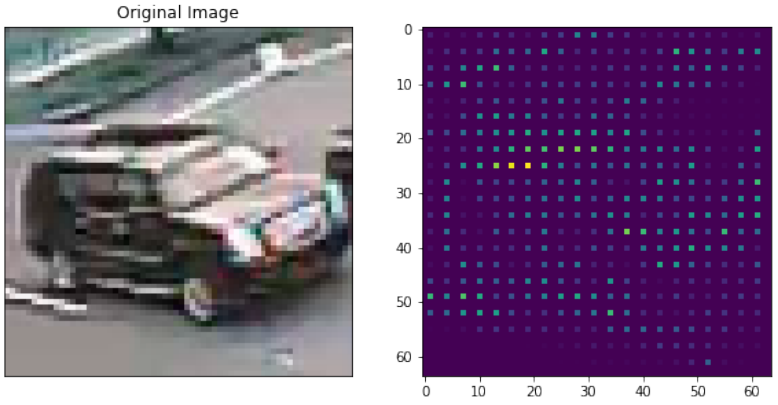
[3] Wong, Tzu-Tsung. "Performance evaluation of classification algorithms by k-fold and leave-one-out cross validation." Pattern Recognition 48.9 (2015): 2839-2846.

[4] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

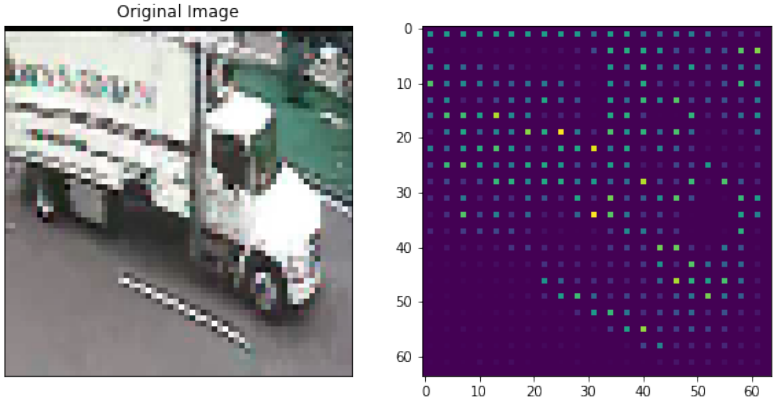
[5] Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection. In *CVPR, 2016*

[6] Redmon, J.: Darknet: Open Source Neural Networks in C, in *http://pjreddie.com/darknet/*, 2013-2016

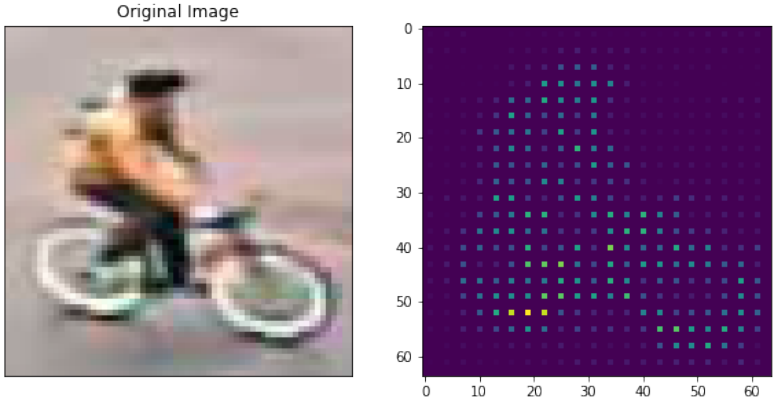
# Appendix A HoG




(a) Car




(b) Articulated truck




(c) Articulated truck

Figure 11: HoG Feature Extraction Results