Traffic Veichle Detection & Classifiaction

Samuel Nyberg, sany21@student.bth.se
Tobias Mattsson, tomt21@student.bth.se

***Abstract*—This report will present methods to develop a traffic monitoring system integrated into a web application by utilising deep learning models to detect and classify vehicles on the road. The report will cover challenges encountered when building a robust deep learning model, showing how a model can be split into two models connected in sequence, pipelining dataflow to improve predictive performance. The project with ground-up deep learning architecture showcases promising results when compared to other well-known detection methods like the YOLO model. Lastly, the report covers limitations and provides a discussion about challenges to overcome for further development.**

***Index Terms*—Vehicle Detection, Computer Vision, Single Shot Detector, Traffic, Classification, Deep Learning**

I. Introduction

Over the years, there has been a steady increase in the number of registered vehicles on roads around the world specificity in Sweden [1]. As a result, roads have become more heavily trafficked, making traffic monitoring increasingly challenging. Modern vehicles also come with more advanced technologies and support systems for each year, some even features autonomous driving.

Evolving driving technologies and the increase of vehicles on the road shows demand for reliable and intelligent systems that can monitor, analyse and respond in real time has significantly grown. The use for accurate real time vehicle detection and classification play a crucial role in multiple advanced systems such as autonomous vehicles or traffic monitoring. This report will cover different strategies for developing such system, benchmarks against another well known detection model, challenges, results and limitations.

II. Problem Definition

The problem this project aims to solve is to develop a real time traffic flow monitoring tool by using deep learning that can detect the position of a vehicle but also classify category. This technology can for example be used by government institutes such as Swedish Traffikverket to automaticity monitor roads and detect those that encounter much heavy traffic such as lorries and buses, thus being able to focus resources to repair and rearm roads and bridges encountering more wear then others. The technology can also be applied by auto manufacturers in development of autonomous cars to detecting different vehicles around it and their respective position.

III. Proposed Method

A. Data Preparation

Annotated data was downloaded from an annotations service called Roboflow [2]. Downloaded data comes in form of images of traffic with pre annotated coordinates of the four corners of a boundary box and also class label of the vehicle inside of the box. From this dataset two different subsets was generated by transforming, resizing and augmenting the images. The first set was made to train and validate the regression model while the other dataset was developed for the classification model. Train, test and validation dataset for the regression model was generated by first restructuring the coordinate system from four corners to centre point of x and y of the vehicle together with width and height. The restructure of the coordinate system made a huge improvident on the models performance while maintaining a the same output vector of 1x4. Following, the images was augmented to generate more data points. Each image underwent three augmentations each performing flip, brightness and crop. Also each bounder box needed to be recalculated to take the new image into account.

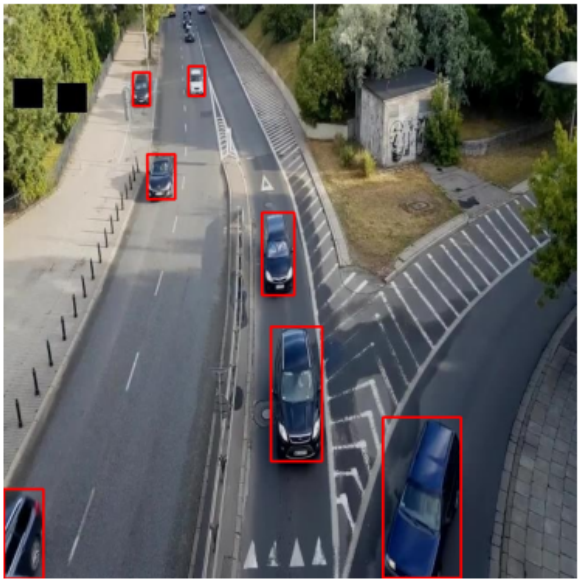


Fig. 1: Regression task data with boundary boxes

The subset for the classifier aims to train the model on learning the classes for vehicles, therefore a dataset was generated by using the set from [2]. This subset was generated by taking the annotated boundary boxes from the original dataset [2], cropping out only the vehicle. Then resizing the image to the classifiers desired input size of 128x128x3 and combining it with the class of the vehicle. This resulted in a

train dataset of 7049 vehicles with a unbalanced distribution as seen in figure below.

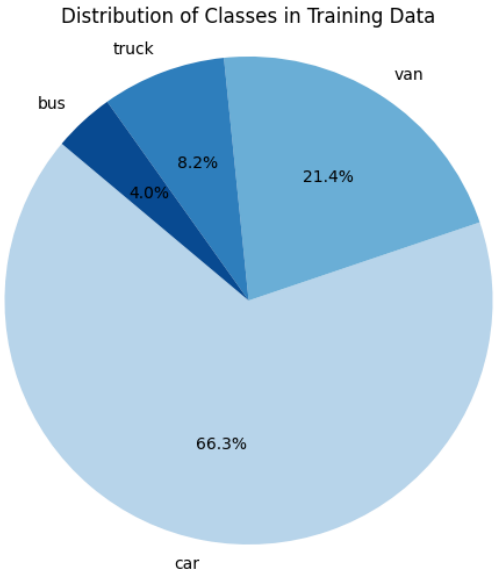


Fig. 2: Class distribution of classes in classifier dataset

B. Custom CNN

The first method was to build a custom deep-learning model for detection and classification. The project underwent many different iterations where multiple approaches where tested. The first iteration was to build a CNN (convoluted neural network) that consisted of four parts, a feature extractor, model adaptor consisting of a series of fully connected layers that splits into two different computing head layers. One for regression tasks made for detecting object position and a second for classifying type of vehicle. This approach was made to try to save on computing resources, allowing the data stream to only pass trough one set of convoluted layers and one set of fully connected layers. This method, while promising encounter a multitude of problems mainly concerning the training data for the classifier. It was quickly discovered that the data used for training the first iteration of the model was not ideal for either object detection nor classification, this was because data was consisting of images with multiple vehicles and multiple classes. Only separable by annotations where each corner is marked and a class label resulting in difficulties when calculating loss and lack of clean training data.

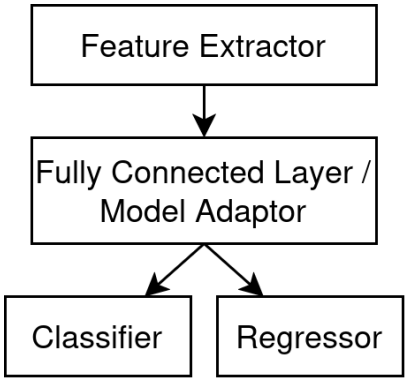


Fig. 3: Model Structure of first iteration

Final iteration of the custom model features a completely different approach then previously mentioned. Data is processed by two CNN models, first by a regression model. Objective of the regressor is to detect the center location in x and y of the object of interest, that being a car, van, bus or truck but also height and width of the object. The regressor consist of a feature extractor with an input size of 244x244x3 meaning processing a RGB image. The feature extractor has 14 layers, eight convoluted layers, four 2x2 max-pool layers and two dropout layers with a dropout chance of 30%. The architecture is built into four equal size groups consisting of two convoluted layers with equal amount of filters. Following the two convoluted layers a max-pooling layer is added to reduce dimensionality and reduce the amount of computational power needed since the goal of the model is to be as lightweight as possible. Each section of convoluted and max pool layers have double the amount of filters to the one previous.

Using the features that was extracted in previous step, everything is feed into the fully connected layers. This part consist of four dense layers with 128 neurons each all of them with a dropout layer in-between making four dense layers and three dropout layers. The decision to use this approach instead of having one dense layer of 1024 or two layers of 512 was made after noticing that the model tended to overfit and not being able to detect objects in new scenarios for example when deployed on a real life system. Following, data from the fully connected layer is feed into one final dense layer consisting of four neurons, one for each class with a sigmoid activation function. This is to predict centre of x and y coordinate, height and with of the object. The regressor model was trained with a optimizer of 0.0001 and evaluation mse (Mean Square Error) and all dropout layers have a a rate of 30%.

Once the image has been processed by the regressor model. All detections gets the four corners of a square calculated and cropped out from the image isolating the vehicle. The detected vehicle is then rescaled to 128x128x3 and feed into a classifier model. The classifier consist of the same feature extractor and fully connected layer as the regressor only differing in the last top layer where a class is predicted with a soft-max activation function. This was the solution to the problem described in

the first iteration of the custom model, since this allowed for having different training data and evaluation metrics for the two models, the regressor training on images with annotated boundary locations while the classier was trained on only isolated vehicles and corresponding class. To counter for the class imbalance presented in 2 weights to each class was calculated and added to the training of the classifier to ensure more stable predictions.

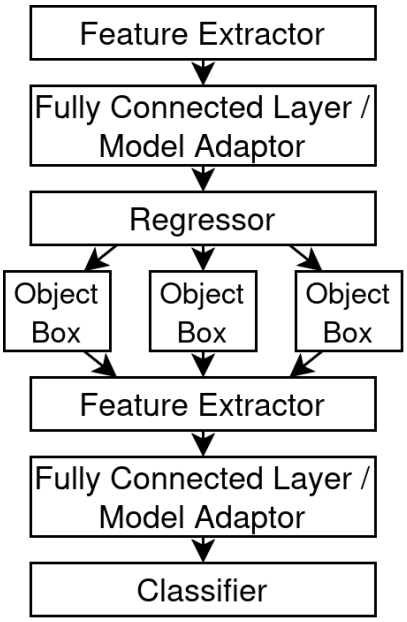


Fig. 4: Model Structure of custom model

C. YOLO Model

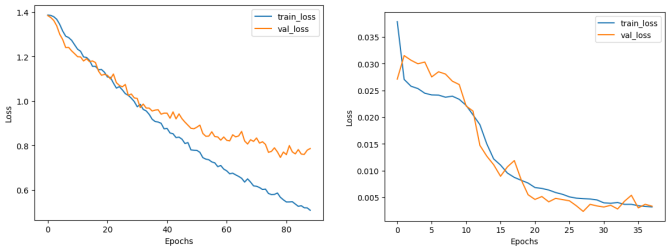
The second model that was implemented during this project was the YOLOv8 Nano model. It's the lightest version of the powerful YOLO framework from Ultralytics. It's very accurate and precise for real-time object detection. The pretrained YOLO model can classify approximately 80 classes, which is far too many for the scope of this project and would give to many false positives. Instead, the model was retrained on the custom dataset obtained for the project, which instead only contained four vehicle classes: bus, car, truck, and van. Training consisted of 50 epochs with a batch size of 16. During training, standard YOLO data augmentation techniques such as random scaling and flipping were applied. After training, the model was evaluated on an isolated validation dataset. This model was mainly implemented as a benchmark for the custom model.

D. Integration with a web-application

To apply the developed models on real-life scenarios, it was integrated into a Flask web server. This would make it possible to use the models on lightweight devices like phones. The client sends frames from the videostream to the flask server for processing, which is done on the host GPU. The server then returns the new image with the label and boundary boxes applied onto the original image using CV2. Unfortunately, the combination of large models and GPUs not powerful enough meant that images could not be processed at native framerate. Instead, the number of processed frames was lowered to one processed frame for every 20 incoming frames. For the frames that didn't get processed, they kept the labels and boundary boxes from the previously processed image.

IV. Results

A. Custom Model



(a) Training vs. Validation Loss for the classifier.

(b) Training vs. Validation Loss for the regressor.

Fig. 5: Comparison of training and validation losses for the classifier and regressor.

1) Training and Validation Loss Custom Model: The loss was calculated during training, which is an important metric for evaluating the performance of the model. Plotting both training and validation loss makes it easy to visualize the difference. In cases where the training loss continues to decrease while the validation loss stays the same or even increases, it is a good indicator of model overfitting on the training data.

2) Confusion Matrix Custom Model:

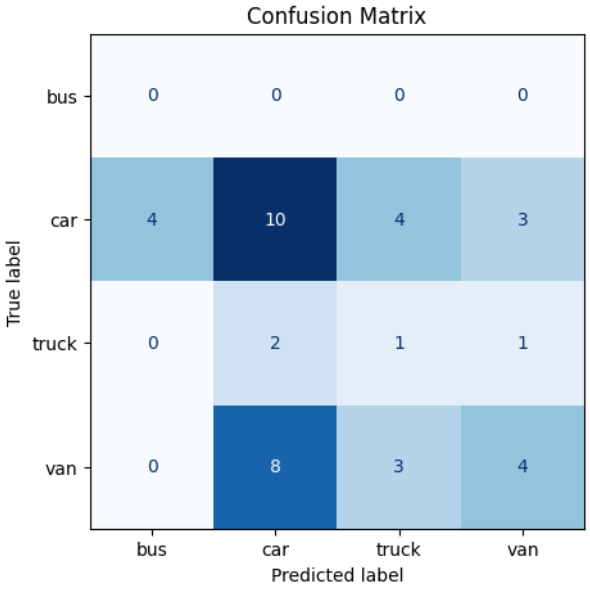


Fig. 6: Confusion Matrix for the custom model.

A confusion matrix is a good tool for model evaluation, which gives better insights into the results than by just using accuracy or precision. By using a confusion matrix, it becomes easier to evaluate improvements for specific classes.

As can be seen in figure: 6, there is a notable similarity between the ”car” and ”van” classes. Many samples of vans are misclassified as cars, indicating that the model struggles to distinguish between these two categories. This is valuable feedback for improving the model or the dataset.

In addition, the confusion matrix can tell us if a specific class is being over-represented or over-predicted by the model. If only one class is shown in the predicted column consistently, regardless of the true label. If this is the case, the model is not learning anything meaningful, and it clearly has a bias towards that class only. This kind of information can be very good for identifying if the model is overfitting to one class in particular or is biased.

3) IOU Scores Custom Model:

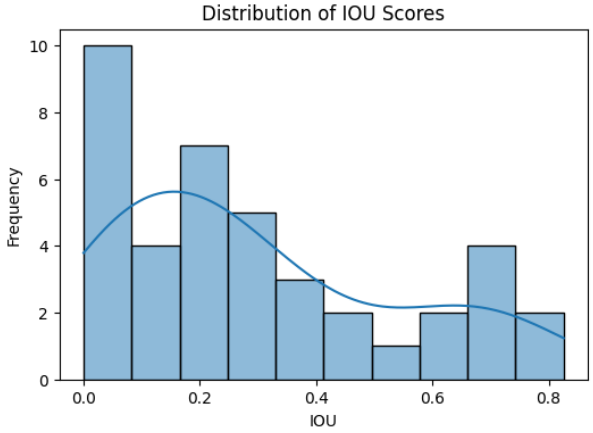


Fig. 7: Intersect over Union for custom model.

IOU scores display to what extent the predicted and true boundary boxes overlap each other, where zero means no overlapping and 1 means complete overlapping. By plotting the IOU scores in a histogram, it is easy to see how well the predictions align with the ground truth. As can be seen in this plot, the majority of predicted boundary boxes differ from the true boxes, which indicates that the model struggles to accurately place the boundary boxes. IOU scores can also be used to set a threshold for what is considered a true positive or not.

4) Sample plots YOLO Model:

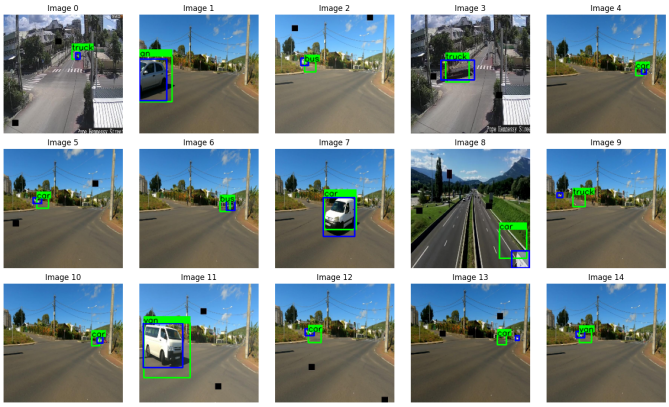


Fig. 8: Sample plots of the resulting custom model predictions.

The sample plots from the predictions on the test dataset show that in some instances the model can accurately place the boundary boxes, while in some images it is completely off.

B. YOLO Model

1) Confusion Matrix YOLO Model:

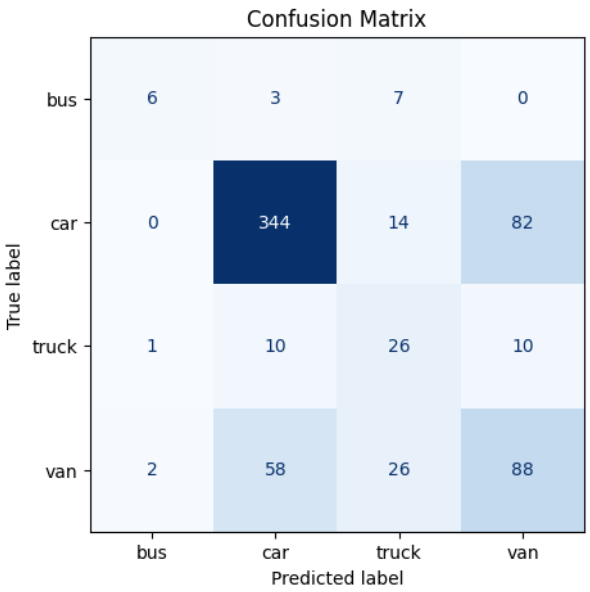


Fig. 9: Confusion Matrix for the YOLO model.

For the YOLO model it can be observed that the model struggled to differentiate between cars and vans, similar to the custom model. The model is the most accurate at classifying cars, and struggles to classify the other classes. This is probably due to cars being the majority of the training data, and therefore the model can learn from significantly more examples than the other classes.

2) IOU Scores YOLO Model:

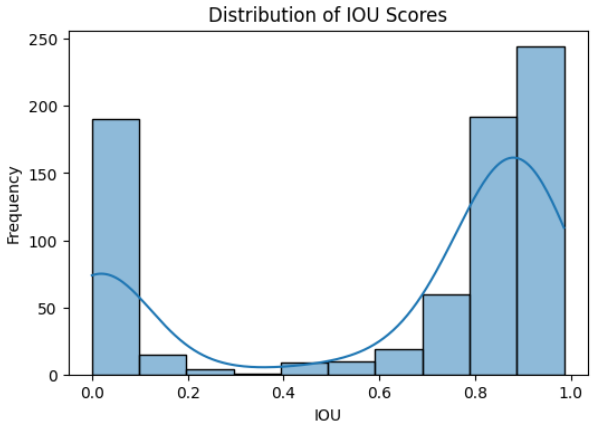


Fig. 10: Intersect over Union for YOLO model.

The IOU histogram for the YOLO model display that the majority of the predicted boundary boxes are very accurate. However it is important to point out the spike in frequency between 0.0-0.1, which is an interesting result.

3) Sample plots YOLO Model:

Fig. 11: Sample plots of the resulting YOLO predictions.

The sample plots from the predictions on the test dataset display high precision for the predicted labels and boundary boxes.

C. webbapp integration

Fig. 12: Screenshot from web server.

To visualize how the model performs in real life, a Flask web server was set up. Unfortunately, mobile browsers only allow camera access by secure (https) sites with a valid SSL certificate, which was not obtainable for the project. This meant that phones could not be used for demonstration. In addition, the laptops that were used during the project were not powerful enough for processing when running on battery power due to throttling. As a result, a webcam connected to a desktop computer was used to capture images of a toy car for demonstration of the model.

V. Discussion and Limitations

A. Unexpected spike in IOU Score

The IOU score histogram for the YOLO model figure: 10, contains a noticeable spike between 0.0-0.1. This suggests that the predicted boundary box frequently does not overlap with the true boundary box at all. One reason for this could be that the model assigns the predicted box to the wrong class, or simply doesn't overlap with any ground truth box.

B. Simultaneous vs two-step prediction solution.

When creating an object detection algorithm, the two most common methods are one-step and two-step. In one-step methods like YOLO, the network predicts the object class and the object's boundary box location simultaneously. This was the first method we tried for our project. The downside of a one-step approach is that the model is learning both tasks at the same time, so if one part is not learning well, it could make the other part worse. In our situation, the classifier could not find features, and its predictions were close to random, which therefore hurt the performance of the boundary box prediction

as well. Essentially, a weakness in one part of the model affects the entire system. In cases like this, it makes more sense to train the steps separately, especially if one task is more difficult than the other.

This is why we chose the two-step solution, where the boundary boxes get predicted first, and the region of interest(ROI) then gets classified. This means that if the classifier performs badly, it does not affect the other part. The disadvantage is that this approach takes a lot longer to train and to use, meaning it is inconvenient for use in real-life situations, which was the goal for the project. Another drawback with this solution is that it needs different kinds of data formats while training. The regressor needs entire images without labels, while the classifier needs cropped images of individual objects and their labels.

C. Difficulties detecting multiple objects

Initially, the intention was to create a model that was capable of detecting multiple objects simultaneously. While training and validating the custom model on multiple objects, it became clear that it was a task more complex than the model could handle, the majority of predictions, both for boundary boxes and labels, were completely random. To ensure good results and allow the model to show its capabilities, the model was limited to predicting on images containing only one object at a time.

D. Challenges with real-time processing

After switching to the two-step solution, we found that the model became too slow for real-time processing. While the two-step method gave much better precision and results, the delay made it impractical for real-world applications where processing speed is critical.

VI. Conclusion

The goal of the project was to develop an object detection model capable of detecting different types of vehicles. Two models were trained during the project. The first was a custom model that consisted of two CNNs, one regressor for predicting boundary boxes, and one classifier for classifying objects. The models were then combined to create a two-step detector. The second trained model was YOLOv8, which was retrained on the custom data. This model served as a benchmark to compare against the custom model and to help determine what could be considered a good result. The conclusion is that developing models that can accurately predict multiple boundary boxes and classes is extremely difficult, and it gave us a greater appreciation of the work and complexity involved in creating models like YOLO.

References

[1] Statistiska centralbyrån (SCB), "Registrerade personbilar i trafik 1923–2024," *SCB*, 2024. [Online]. Available: https://www.scb.se/hitta-statistik/statistik-efter-amne/transporter-och-kommunikationer/vagtrafik/fordon/pong/tabell-och-diagram/registrerade-personbilar-i-trafik-19232024/

[2] Roboflow, "VCIATR Dataset – Train Split," *Roboflow Universe*, 2024. [Online]. Available: https://universe.roboflow.com/new-ife0x/vciatr/dataset/1/images?split=train