# Road user detection with YOLO

Yan Xu and Hughes Perreault

June 26, 2017

## Abstract

## 1 Introduction

Detecting road users in real-time has many applications, e.g. monitoring, violation detection and self-driving car. These applications may create great value for people's life, but also pose more and more challenges on detection system's accuracy and speed. Using self-driving cars as an the example, the system need to take the correct action depending on it's surroundings, in less than 0.1s. In 2016, Redmon et. al. propose a new Object detection algorithm, YOLO[Joseph Redmon, 2016b] and its upgrade version YOLO v2[Joseph Redmon, 2016a]. It detects objects in real-time, from 45 FPS to 155 FPS, while maintaining high reliability. It is a good candidate for a road user detection application. In this project, we train the YOLO model and its variants with the MIO-TCD road user dataset[MIO, 2017], test the model's accuracy, apply them for vehicle tracking and get excellent results.

## 2 Related work

**Image Recognition**. Convolutional Neural Network(CNN) applied on image recognition can be firstly found in LeCun's article[Y. LeCun and Haffner, 1998]. In recent year, thanks to the development of GPU and big image recognition datasets[Ima, 2017], CNN shows great advantages comparing with conventional image recognition algorithm such as SVM. Starting from AlexNet in 2012[Krizhevsky et al., 2012], CNN becomes the absolute first choice for image recognition. As of today at 2017, the most advanced CNN algorithms, such as VGG[Karen Simonyan, 2014], GoogLeNet[Christian Szegedy, 2014], ResNet[Kaiming He, 2015], have reached a higher precision than human expert on image recognition.

**Object Detection.** Object detection algorithms need to predict both object classes and position in a given image. Object detection algorithms such as DPM(Deformable parts models)[Pedro F. Felzenszwalb and Ramanan, 2010] apply sliding windows with different scales on the original image, and run classifier on each window's image. More recent approaches like R-CNN[Ross Girshick, 2014] first proposes potential bounding boxes and classify these proposed boxes. Algorithm YOLO[Joseph Redmon, 2016b] has a simpler pipeline. It predict the objects' class and position in one forwarding network and have both good precision and speed. Tested on Pascal VOC(Visual Object Classes)Pascal, YOLO achieve 45 FPS to 155 FPS and 78.6 to 69.0 mAP.

**Road User Detection.** Sivaraman et.al. use sliding windows, prepared features such as HOG, and SVM to detect on-road vehicle[Sayanan Sivaraman, 2011]. This method processes one image in several seconds. Dollar et.al. evaluate 16 pedestrian detection algorithm, including SVM, SIFT, Shape based algorithm[Piotr Dollar and Perona, 2011]. Algorithm FPDW, recommended by the authors, has a 6.5 FPS and log-average miss rate 57%.

**MIO-TCO dataset.** MIO-TCD traffic camera dataset challenge is prepared and hosted at University of Sherbrooke and Miovision Inc. Canada. The dataset consists of 786676 images acquired at different times of the day during different seasons by thousands of traffic cameras in Canada and the United States. The images have been selected to cover a wide range of challenges and are representative of typical visual data captured today in urban and rural traffic scenarios. Each foreground object (vehicles of all kinds as well as pedestrians and bicycles) has been carefully identified to enable a quantitative comparison and rigorous ranking of algorithms[MIO, 2017].

## 3 Methods

### 3.1 Dataset

The MIO-TCD dataset contains two parts: classification and localization.

**Classification Dataset.** The classification part contains 519,164 training images and 129,795 test images. Images are classified to 11 classes: articulated truck, bicycle, bus, car, motorcycle, non-motorized vehicle, pedestrian, pickup truck, single unit truck, work van, and background. When we train our model, we don't use the class "background". The size of the images is not uniformed. Their widths and heights vary from 50 to 300px.

**Localization Dataset.** The localization part contains 110,000 training images and 27,743 test images. The dataset provide bounding boxes positions and their image classes. Images are classified to 11 classes. Comparing with classification data, this dataset don't contain the class background, but have the class motorized vehicle, which is not included in the classification dataset. The images are in two sizes, 342x228 and 720x480.

## 3.2 Model Architecture

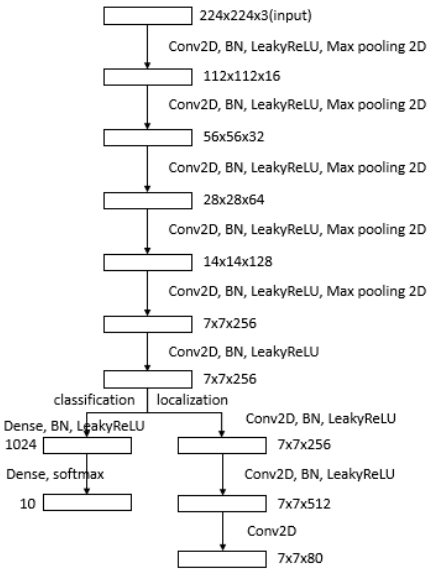


Figure 1: Network Model

We use a convolutional neural network to the classification and localization. Because of our limited computational power and time, we build a neural network similar but smaller to YOLO v2. As shown in Fig. 1, after 6 convolutional layers, we build the classifier and localizer on top of these 6 layers: 1. add 2 fully connected layers to classifier the image to 10 classes; 2. add 3 convolutional layers to predict bounding boxes and located classes. In the last layer of localization, we predict 7x7x5 bounding boxes. Each bounding box consists of 5 location predictions : $x, y, w, h$, confidence and 11 classification probabilities. The output is a 7x7x80 tensor. The output boxes' width and height are deformed by pre-defined anchor boxes before making final predictions.

Batch normalization is used in each layer to improve the convergence. Activation function is LeakyReLU:

$$f(x) = \begin{cases} 0.1x, & \text{if } x < 0 \\ x, & \text{if } x \geq 0 \end{cases} \tag{1}$$

LeakyReLU have small gradient when the input is negative, to avoid the death of neurons.

## 3.3 Anchor box

Anchor boxes serve as reference at multiple scales and aspect ratios, which avoids enumerating images or filters of multiple scales or aspect ratios. Anchor boxes are generated based on the training data. The algorithms used is the kmeans algorithm with k = 5. Similar to the YOLO model, for kmeans clustering, we calculate two boxes' distance with the following formula:

$$d = 1 - IoU \tag{2}$$

Finally, we generate 5 anchor boxes: (1.064, 0.970), (2.204, 2.05), (4.180, 3.992), (7.524, 7.129), (13.680, 13.003). Our anchor boxes are quite different than the anchor boxes of YOLO v2. The anchor boxes in YOLO v2 are thinner, ours are larger. One explanation is that the objects in the data understudy are mostly vehicles, which favor larger boxes.

## 3.4 Loss Function

**Classification.** We use softmax as the loss function for classification:

$$p(c_k|\bar{x}) = \frac{exp(a_k(\bar{x}))}{\sum_j exp(a_j(\bar{x}))} \tag{3}$$

**Localization.** The algorithm predict bounding boxes with coordinates and a confidence level. For each bounding box, it gives the probabilities for the 10 classes. Therefore, the Localization loss function consists of three parts: confidence loss, coordinates loss and classification loss.

$$L_{total} = L_{conf} + L_{coor} + L_{class} \tag{4}$$

$$L_{conf} = \lambda_{obj} \sum_{true} (1 - Conf)^2 + \lambda_{noobj} \sum_{false} Conf^2 \tag{5}$$

$$L_{coor} = \lambda_{cor} \sum \left\| \hat{B} - B_{pred} \right\| \quad (6)$$

$$L_{class} = \lambda_{cl} \sum \|\hat{p}_{class} - p_{class}\| \quad (7)$$

In equation 5, $true$ means the object is present in ground truth, $false$ means absent. In equation 7, $\hat{p}$ is the one hot encoded vector for classes in ground truth. In equation 6, $B$ is the bounding boxes vector $(x, y, w, h)$. we take $\lambda_{obj} = 5, \lambda_{noobj} = 1, \lambda_{class} = 1, \lambda_{coor} = 1$.

### 3.5 Training

We are using Keras framework[Ker, ] for the experiment.

**Classification.** Dataset is split in to 9:1 training and validation dataset. Images are resized to 224x224. We set the learning rate to 0.001 with decay 0.0005 after each batch. The updating Momentum is 0.9, batch size is 64. Training algorithm is backward propagation. We use early stopping to prevent the over-fitting. The training is stopped after 5 epoch when the model performance stop improving.

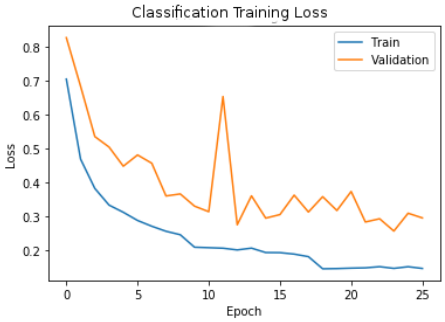


Figure 2: The training loss during classification part


Figure 3: The accuracy over time for the classification part

**Localization.** For the localization, we used the weights from the convolutional layers trained in the classification part. We used the same hyper parameters as the classification training.

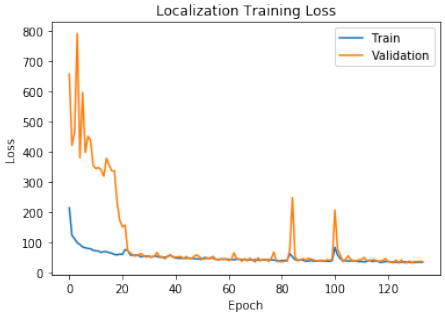


Figure 4: The training loss for the localization part

### 3.6 Vehicle Tracking

For tracking vehicle in a video, we apply the trained model. The model predict bounding boxes for vehicles in each frame. We process the image in the boxes and extract their color features, save these feature together with boxes' size and position in the application memory. For the next frames, based on these features and time, we calculate the difference score:

$$s = \frac{\lambda_{dist} \|p_m - p_t\| + \lambda_{size} \|A_m - A_t\|}{\Delta t} + \lambda_{col} \|C_m - C_t\| \quad (8)$$

p is the object position, A is its area, C is its color histogram features, as shown in Figure 5.

For the objects detected in new frame, we calculate the $s$ with each existing memory elements. If s is smaller than a preset threshold for one element, we determine they are belong to the same vehicle and update this element. Otherwise we determine this vehicle is new and insert it to the memory.

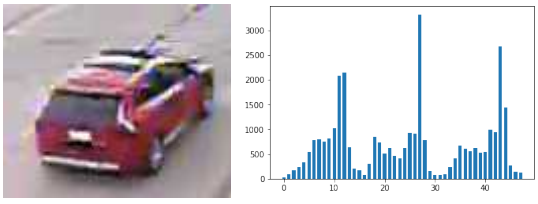

Figure 5: Histogram as color feature for a vehicle image.

Table 1: Evaluation of the Model

| Class | Average precision |
|---|---|
| Articulated truck | 0.58 |
| Bicycle | 0.19 |
| Bus | 0.76 |
| Car | 0.42 |
| Motorcycle | 0.32 |
| Motorized vehicle | 0.25 |
| Non-motorized vehicle | 0.14 |
| Pedestrian | 0.04 |
| Pickup truck | 0.62 |
| Single unit truck | 0.2 |
| Work van | 0.15 |
| Mean | 0.33 |

# 4 Results

## 4.1 Classification

When trained for classification, we could obtain a model with accuracy of 92%. By comparison, the best model in the benchmarks provided by MIO [MIO, 2017] could achieve a accuracy of 97%.

## 4.2 Localization

For the localization of vehicles, we apply the model on all test images in the MIO-TCD dataset. Some detected images are shown as example in Figure 6. To evaluate the model, we use the script provided by the MIO-TCD dataset. If the predicted box overlap with the ground truth box with a a minimum IOU of 50% and the correct class, it count as a true positive. If there is no truth in the predicted box, it count as a false positive. With the true positive, false positive, total objects in ground truth, the script calculate the recall, precision for each class of the model. The results we obtained as shown in Table 1 are worse than the benchmarks provided the dataset organizers, the reasons why this happened will be discussed in the discussion.

To compare, the state of the art on this dataset can achieve a mAP of 79% [MIO, 2017].

## 4.3 Vehicle Tracking

We could achieve tracking object in video. Figure 7 is a snapshot of the video we produced with out model. The full video is available at https://youtu.be/-y8zuYIldlg.

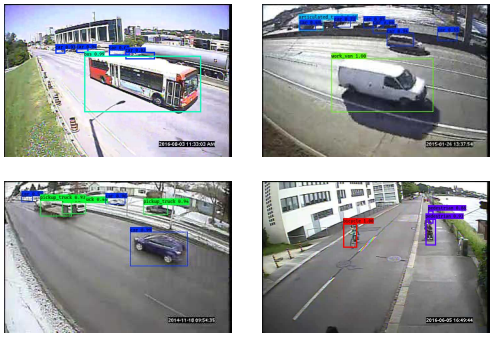

Figure 6: Detected images in MIO-TCD test set

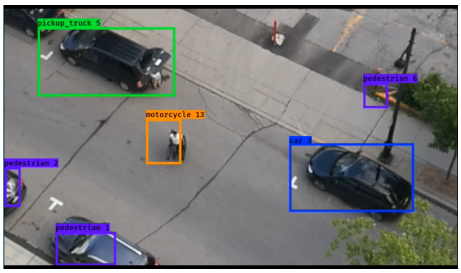


Figure 7: Vehicle tracking snapshot

# 5 Discussion

## 5.1 Model and Dataset

Our localization accuracy is not so high, and there some reasons that explain this. First of all, due to the lack of time and GPU resources, we wanted to train a smaller and faster model, and we knew that meant sacrificing accuracy. It is our belief that our model would scale quite efficiently, if we were to retrain it on a deeper architecture.

Second of all, the accuracy for pedestrians is extremely low. I noticed that this is also the case for all the benchmark data provided by MIO [MIO, 2017]. We can explain this by the fact that pedestrians are smaller than other vehicle, and thus harder to detect. Also, they are the only class with a vertical shape, so the network maybe has a harder time detecting them because of that.

## 5.2 Loss Function

It is also important to notice that the loss function used in our model is different than the evaluation method provided by MIO-TCD. According to Formula 4, the total loss is a addition of box loss and class loss, which means the model have a reduced loss when it predict a correct box with wrong class. However, the evaluation method provided by MIO-TCD judge this as a totally failed prediction. To improve the precision calculated under MIO-TCD criteria, we

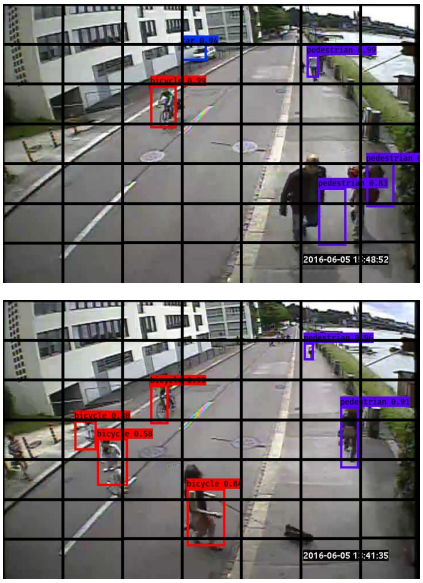

Figure 8: Detected images with Grid. Upper: in the right-bottom corner, object are divided into 4 grid cells, the prediction boxes are not accurate. Lower: no object are divided into more than 2 grid cells.

may adjust the loss function accordingly.

From the tracking video as well as its snapshot(Fig. 7), we find that our model is good at predict object boxes, however, sometimes has difficulties to predict the correct class. One possible improvement is the change the class loss calculation method(Eq. 7) from array distance to a Softmax function as softmax function is better for representing probability.

## 5.3 YOLO Grid

As shown in section 3.2 our YOLO model divide the images to 7x7 grid. We find that once a object is divided into several grid cells, the prediction results are bad. An example is shown in Figure 8. In the right-bottom corner of the upper image, two persons are not correctly detected because they are divided into 4 grid cells.

## 5.4 Tracking Memory

We use the color histogram feature for tracking objects. It is a fast method with low precision because this feature changed with the changing of background and lighting. In the generated video, we find that a moving car is tagged as a new object when the background changed. We can improve this by using a convolutional neural network to generate the image features, and to judge if two images belongs to a same object.

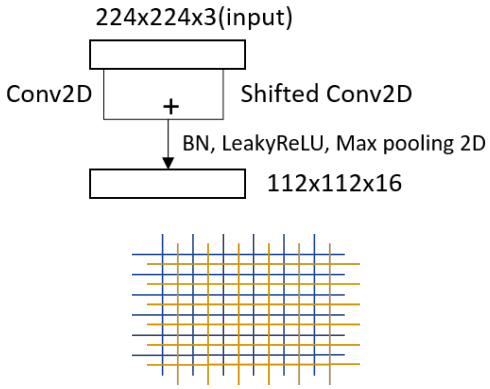


Figure 9: Concept of shift filter. Upper: layer design, Lower: a shift filter will minimize the impact for dividing objects into different grid. If one object is separated by blue grid, it will be in center of a yellow grid cell.

## 5.5 Improvement ideas

The loss function acts the same way whether the network mistakes a bicycle for a bus, or a bicycle for a motorcycle. This doesn't seem right. There exists inherent resemblances between some classes, and we should acknowledge that in some way. We could add a new variable in the loss function, that would scale down the error by half for a certain number of predefined pairs of classes that look alike. To start, we could propose (Bicycle, motorcycle) and (bus, single unit truck).

There is an evident hierarchical structure in our vehicle classes. We could use that to our advantage, by using the concept of the WordTree, introduced in [Joseph Redmon, 2016a]. We could define a tree structure of road users, and re-label every detection with all their relevant classes. For instances, a bike would also get the label "two wheel vehicle". During inference, we would go down the tree until a certain threshold for the level of confidence, and the we would make our prediction.

Avoid divide objects into different grid maybe important for the performance. We may improve YOLO by using the idea of GoogLeNet's inception module[Christian Szegedy, 2014]. In the first layer, we may use two filter, one is shifted to minimize the impact of separating objects into different grid cells, an illustration is shown in Figure 9.

## 5.6 Difficulties in Experiment

We had a hard time training the model, and add the adjust several hyper parameters in order to obtain a decent accuracy. We had to try multiple different learning rates and size batches, and this

took a lot of time.

Keras, one of the framework we use, has very limited support on customized loss function. We have to program this loss function into a Lambda layer added at the end of project. Having a good understand of lower layer implementation is essential to build customized deep neural network model.

# 6 Conclusion

We have built a real time road user detection system with decent accuracy, but still a lot of place for improvement. Systems like ours could be used in the future for self driving cars and city analytics, among other things. Our future works will be to train deeper models for a longer period of time in order to improve accuracy, to improve the YOLO architecture and to take advantage of the resemblances and relations between road users.

# Acknowledgment

The authors would like to thank Prof. Christopher Pal and Mr. Christopher Beckham for the course.

# References

[Ker, ] Keras.

[Ima, 2017] (2010-2017). Large scale visual recognition challenge.

[MIO, 2017] (2017). Miovision traffic camera dataset.

[Christian Szegedy, 2014] Christian Szegedy, Wei Liu, Y. J. P. S. S. R. D. A. D. E. V. V. A. R. (2014). Going deeper with convolutions. *arXiv:1409.4842*.

[Joseph Redmon, 2016a] Joseph Redmon, A. F. (2016a). Yolo9000: Better, faster, stronger. *arXiv:1612.08242v1*.

[Joseph Redmon, 2016b] Joseph Redmon, Santosh Divvala, R. G. A. F. (2016b). You only look once: Unified, real-time object detection. *arXiv:1506.02640v5*.

[Kaiming He, 2015] Kaiming He, Xiangyu Zhang, S. R. J. S. (2015). Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*.

[Karen Simonyan, 2014] Karen Simonyan, A. Z. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*.

[Pedro F. Felzenszwalb and Ramanan, 2010] Pedro F. Felzenszwalb, Ross B. Girshick, D. M. and Ramanan, D. (2010). Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[Piotr Dollar and Perona, 2011] Piotr Dollar, Christian Wojek, B. S. and Perona, P. (2011). Pedestrian detection: An evaluation of the state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[Ross Girshick, 2014] Ross Girshick, Jeff Donahue, T. D. J. M. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. *arXiv:1311.2524v5*.

[Sayanan Sivaraman, 2011] Sayanan Sivaraman, M. T. (2011). Active learning for on-road vehicle detection: a comparative study. *Machine Vision and Applications*.

[Y. LeCun and Haffner, 1998] Y. LeCun, L. Bottou, Y. B. and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*.