YOLO V3 Detect Truck/Vehicle Types

Yingfan Xu

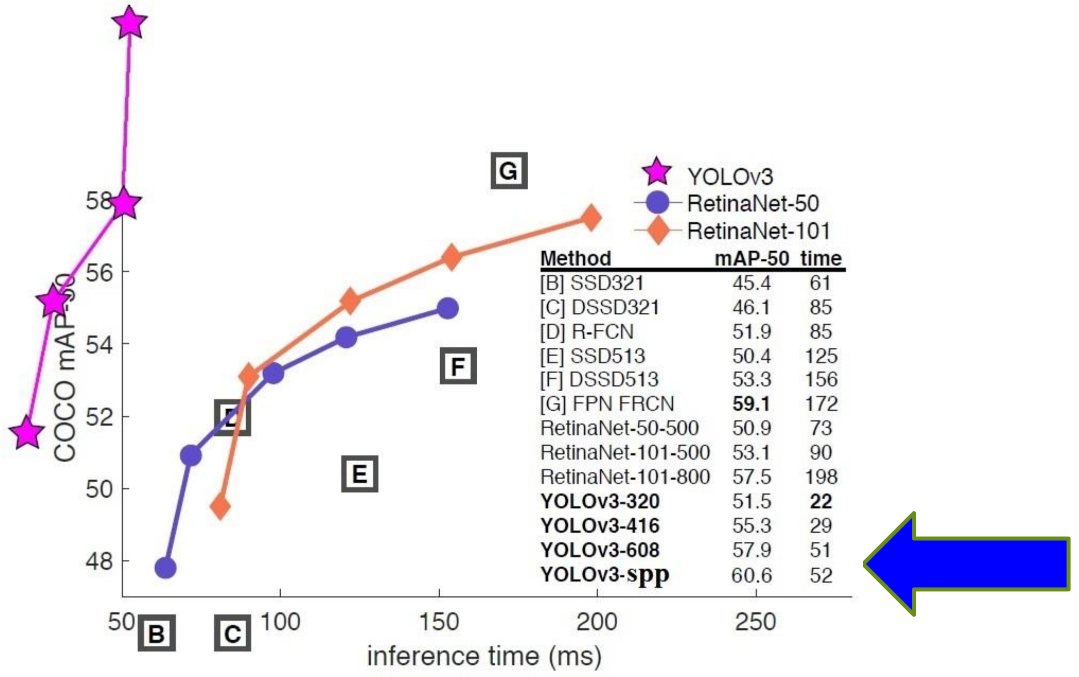
CSTU, June, 2020

Content

I. Why YOLO?

You only look once (YOLO) is a state-of-the-art, real-time object detection system.

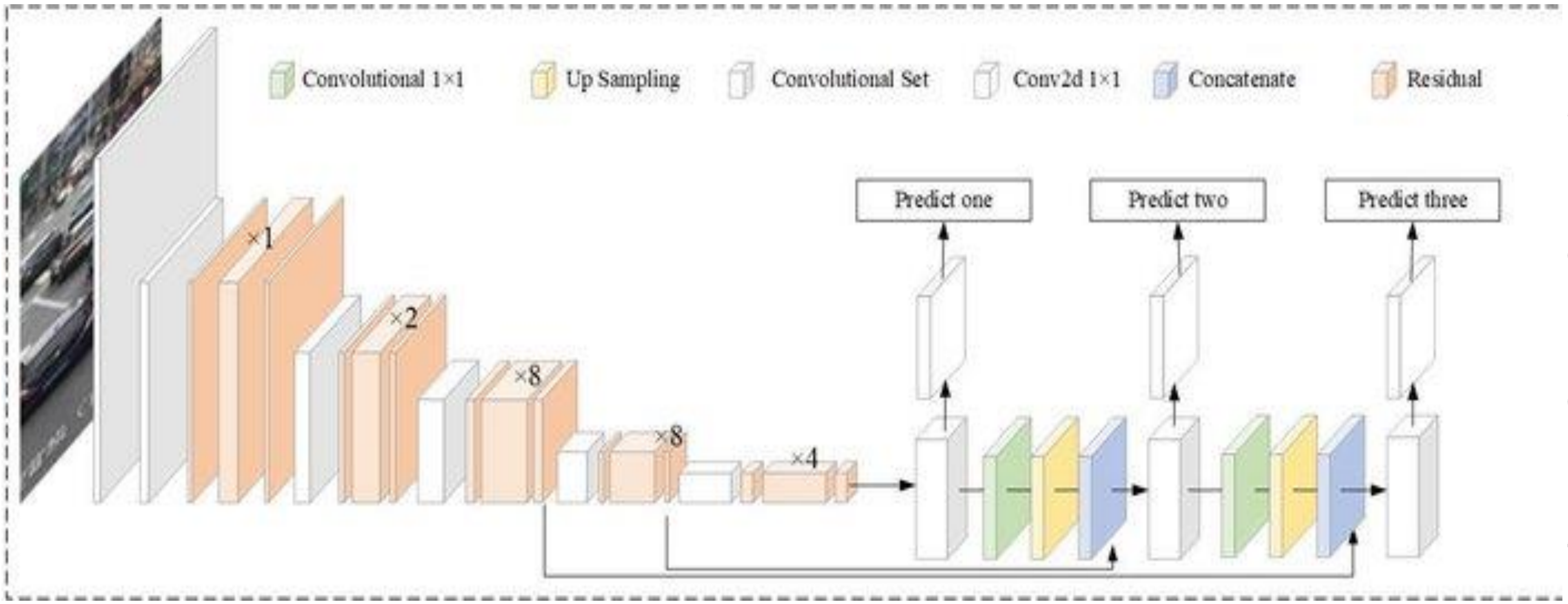
YOLOv3 is extremely fast and accurate. In mAP measured at 0.5 IOU YOLOv3 is on par with Focal Loss but about 4x faster. Moreover, you can easily tradeoff between speed and accuracy simply by changing the size of the model, no retraining required!

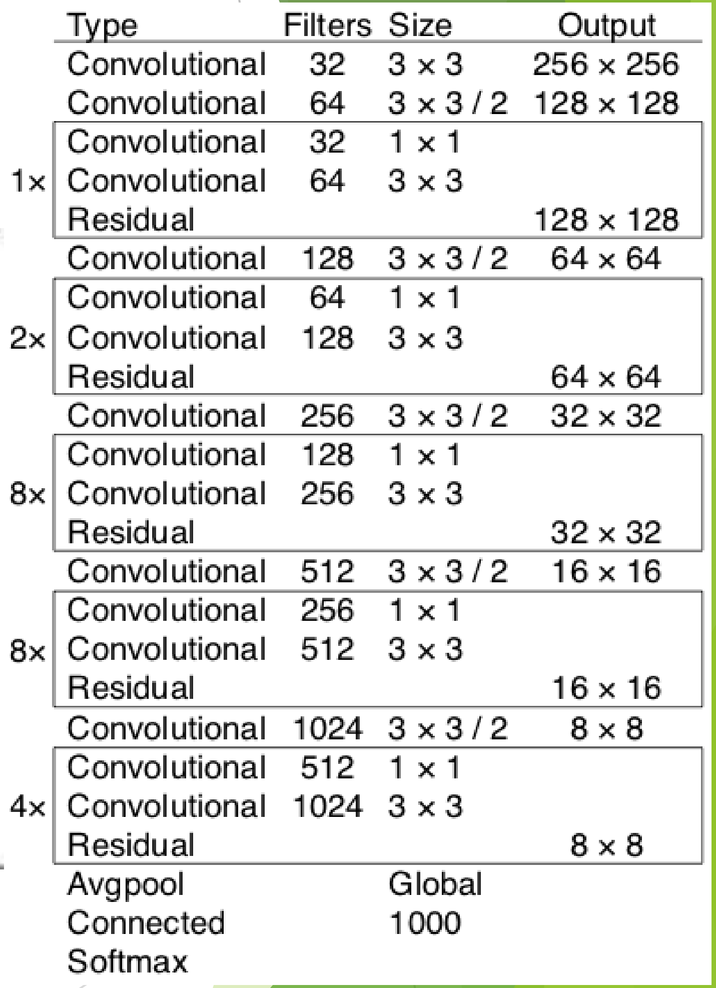


Method	mAP-50	time
[B] SSD321	45.4	61
[C] DSSD321	46.1	85
[D] R-FCN	51.9	85
[E] SSD513	50.4	125
[F] DSSD513	53.3	156
[G] FPN FRCN	**59.1**	172
RetinaNet-50-500	50.9	73
RetinaNet-101-500	53.1	90
RetinaNet-101-800	57.5	198
YOLOv3-320	51.5	**22**
YOLOv3-416	55.3	29
YOLOv3-608	57.9	51
YOLOv3-spp	60.6	52

Figure 3. Again adapted from the [7], this time displaying speed/accuracy tradeoff on the mAP at .5 IOU metric. You can tell YOLOv3 is good because it's very high and far to the left. Can you cite your own paper? Guess who's going to try, this guy → [14].

YOLOv3 model structure detail. It uses Darknet-53 as the backbone network and uses three scale predictions.





	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

II. What We do ?

Train a Yolo v3 model using Darknet by transfer learning with GPU on Colab.
(Darknet is an open source neural network framework written in C and CUDA.
YoloV3 and Darknet were developed by Joseph Redmon)

- ❑ Collect 8 classes samples: USPS Truck, UPS Truck, Ambulance, Fire Truck, FedEx Truck, Bus, Police Car, Other Vehicle
- ❑ Collect negative samples with objects that we do not want to detect
- ❑ Sample labeling; Train, Validation and Test sample preparation
- ❑ Speeding up training by using Fine Tune technique
- ❑ Colab NoteBook code edit, install Darknet and cuDNN, compile Darknet, configuration training file
- ❑ Inference 8 classes Truck/Vehicle images
- ❑ Inference videos

1. Samples Preparation and Labeling (+ Negative samples)

Collected 480 jpg images with 8 classes (USPS Truck, UPS Truck, Ambulance, Fire Truck, FedEx Truck, Bus, Police Car, Other Vehicle) from internet, each class having 60 images, 50 for training and10 for Validation;

Collected 25 testing images (not labeled) and several videos for model inferences.

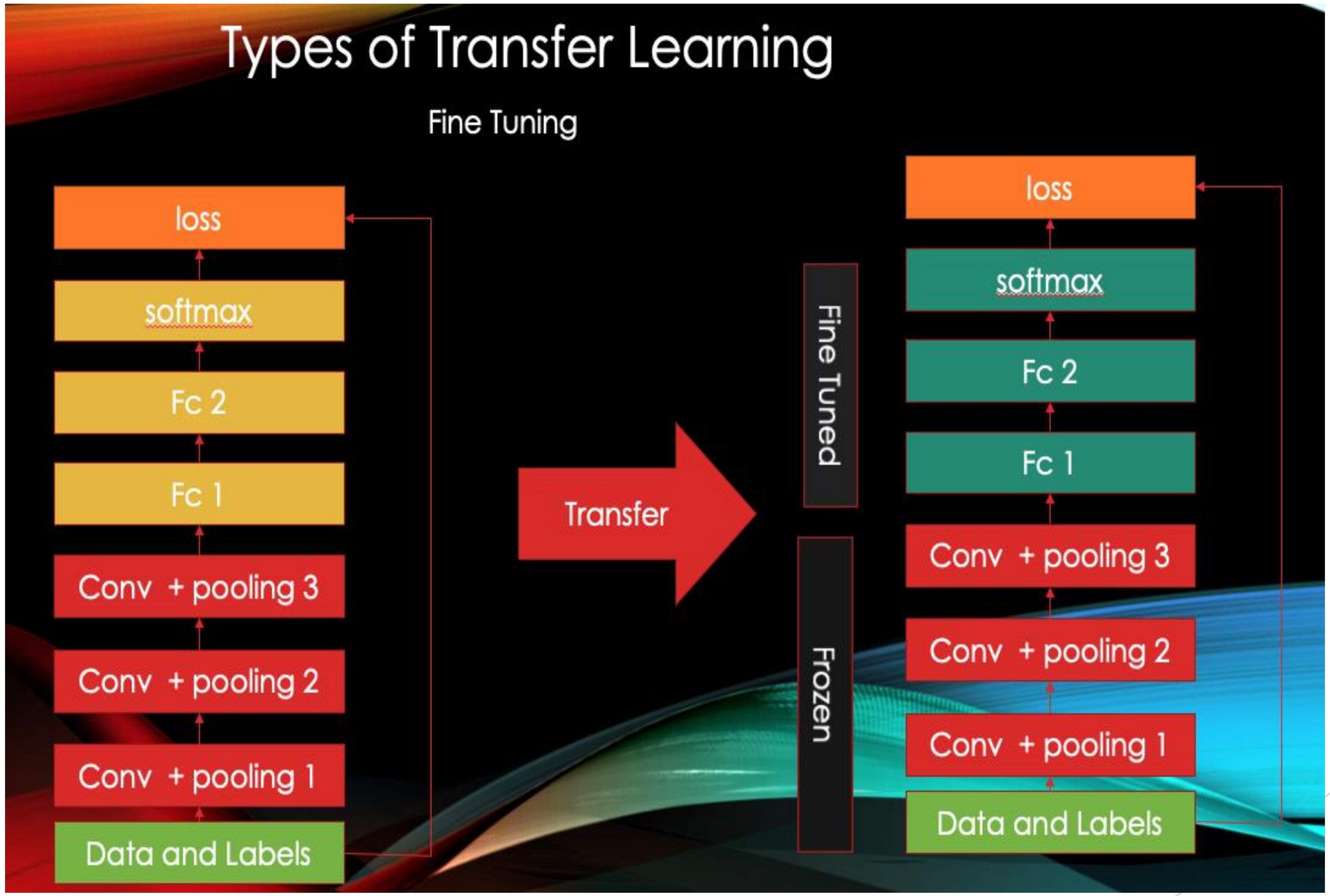
Collected 100 negative samples (80 for training 20 for validation) with objects we do not want to detect, such as streets, buildings, pedestrian, traffic lights, etc. Negative sample will tell machine do not detect those objects, so that it improves the model performance.

Training and validation samples are labeled with software manually, each labeled sample has a txt file with Xc, Yc, w, h which are the center coordinates, width and height of bounding box respectively; a snippet python code is used to label empty txt file for negative samples.

Data augmentation is done in Darknet by default. Random crop, flip, distort are used with bounding box. At first iteration image size is 64, after 4000 iterations, image size can reach 256,000.

2. Speeding up Training by Fine Tune Technique

We used new weights generated with modified configuration file, improved training time more than 80%.



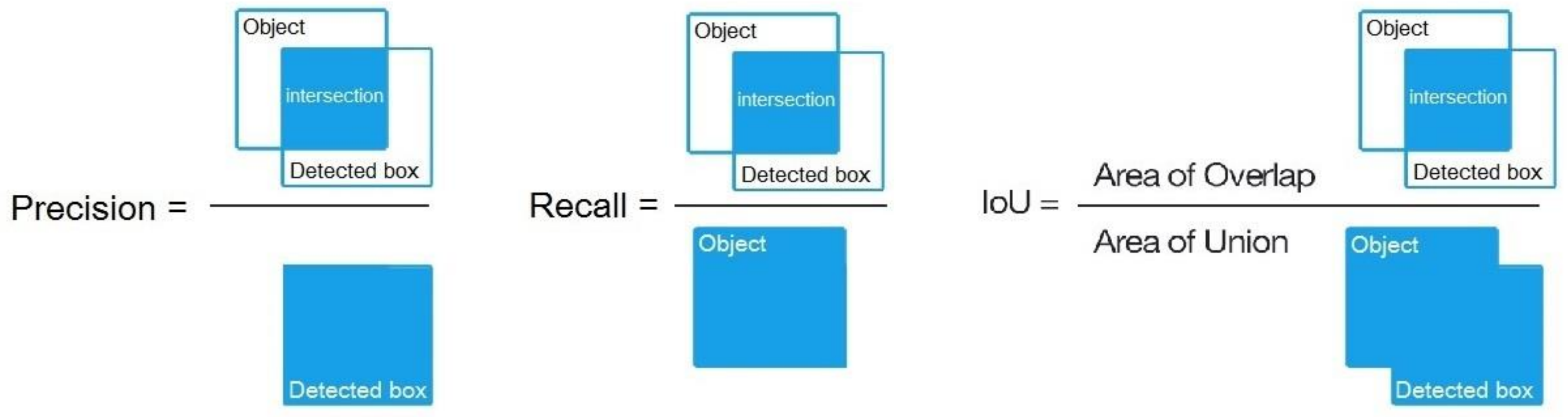
3. Validation

Run train and validation at same time, to do so, just train with -map flag.

In this case, validation is measured by mAP: mean average precision, it is mean value of average precisions for each class. See mAP and Loss chart in results section.

IoU (intersection over union) - average intersection over union of objects and detections for a certain threshold = 0.24

Precision = TP/(TP+FP)



4. YoloV3 Model Final Train and Validation Results

After Colab Notebook coding edit and model compile, we ran several trial and error runs for 2000 iterations, then the final train divided into 2 runs: 4000 + 4000 iterations. (The Notebook is attached).

Total 480 positive samples + 100 negative samples

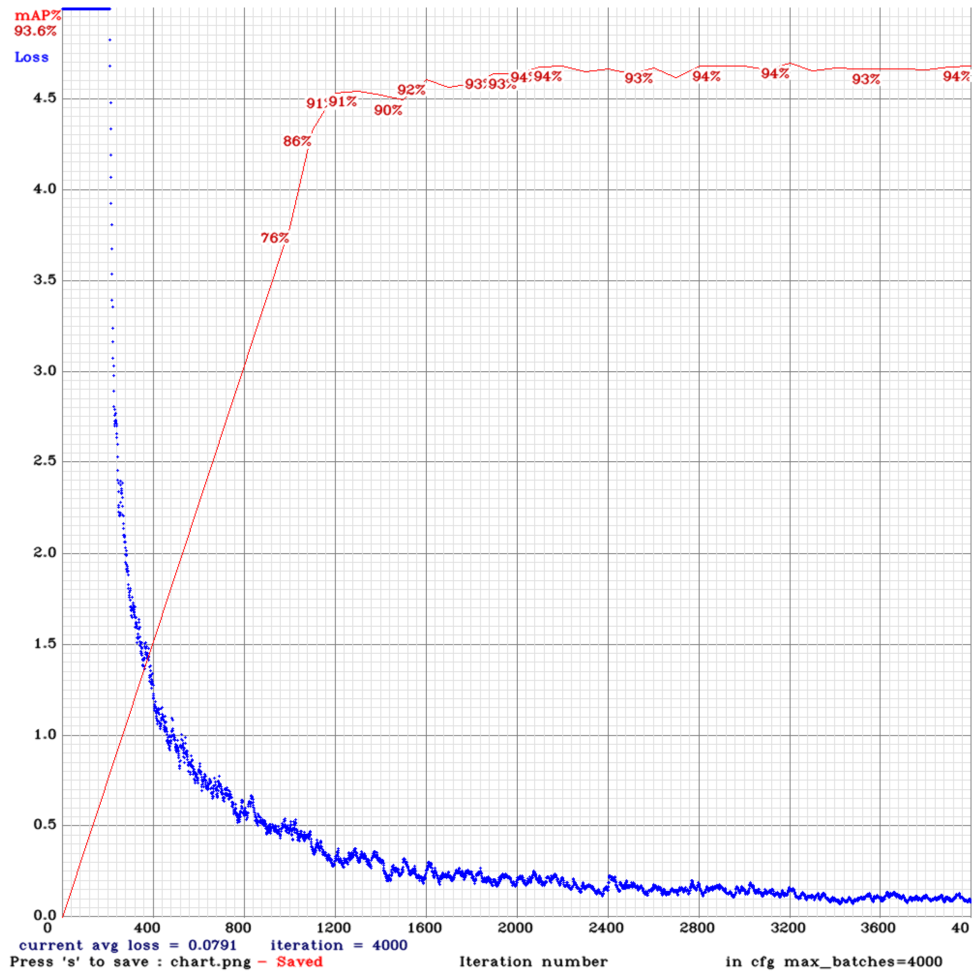
(1) mAP for each class after 8000 iterations:

class_id = 0, name = usps, mAP = 72.71%
class_id = 1, name = bus, mAP = 96.45%
class_id = 2, name = police car, mAP = 98.18%
class_id = 3, name = fedex mAP = 100.00%
class_id = 4, name = fire truck, mAP = 89.09%
class_id = 5, name = other vehicle, mAP = 100.00%
class_id = 6, name = ambulance, mAP = 98.33%
class_id = 7, name = ups, mAP = 100.00%

Average IoU = 76.55 %
At IoU threshold = 50 %, (mAP@0.50) = 94.45 %

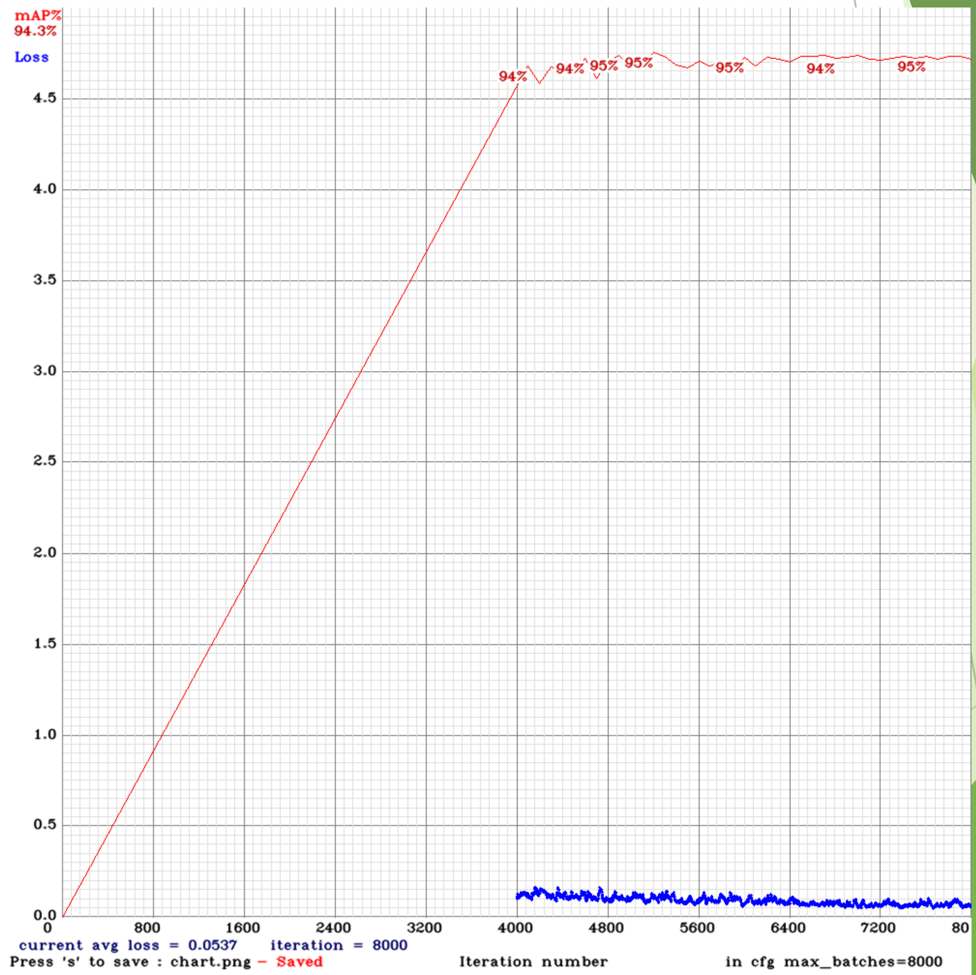
(2) Overall loss and mAP chart:

First 4000 iterations



```
current avg loss = 0.0791    iteration = 4000
Press 's' to save : chart.png - Saved                    in cfg max_batches=4000
```

Second 4000 iterations



```
current avg loss = 0.0537    iteration = 8000
Press 's' to save : chart.png - Saved                    in cfg max_batches=8000
```

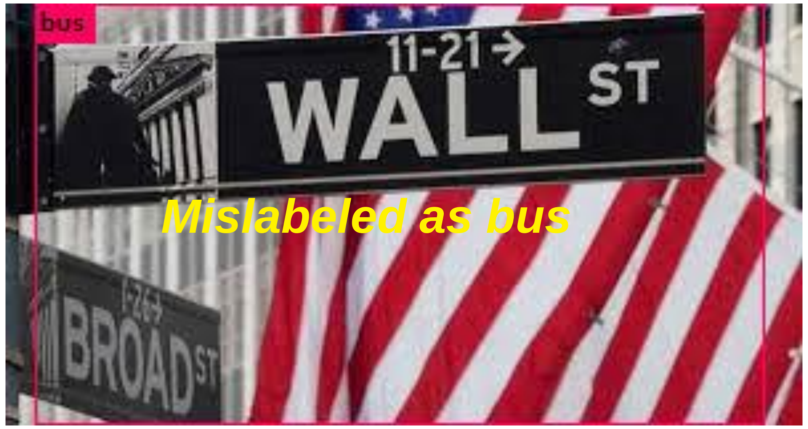
mAP of 8 classes vs. iterations (Train with negative samples)

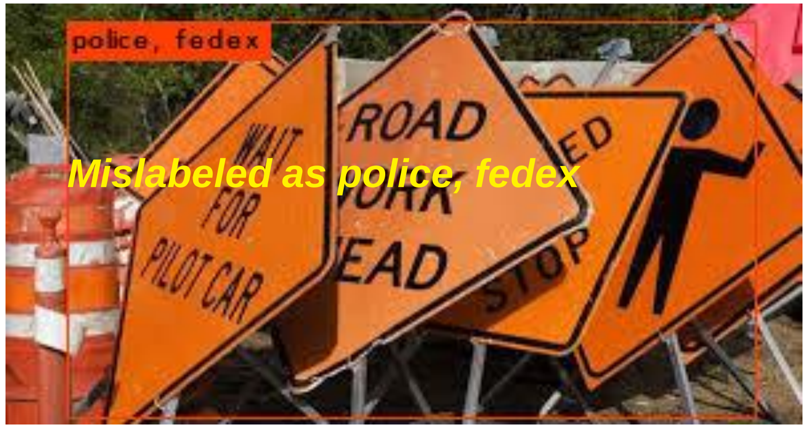
Iterations	USPS Truck	Bus	Police Car	Fedex Truck	Fire Truck	Others	Ambulance	UPS Truck	Average IoU	Average mAP
500	32.51%	47.28%	58.34%	37.41%	40.64%	27.57%	14.67%	29.44%	52.73 %	35.98 %
1000	63.21%	71.97%	82.75%	83.76%	92.33%	58.07%	68.37%	85.76%	48.35 %	75.78 %
1500	67.62%	92.36%	99.09%	89.34%	96.33%	91.26%	84.55%	98.48%	67.22 %	89.88 %
2000	68.37%	95.83%	97.69%	97.69%	90.00%	99.05%	93.64%	100.00%	69.39 %	92.78 %
2500	73.92%	87.97%	98.18%	91.65%	98.33%	100.00%	91.82%	100.00%	72.34 %	92.73 %
3000	70.44%	96.01%	88.18%	96.67%	100.00%	100.00%	98.18%	99.24%	69.75 %	93.59 %
3500	71.37%	93.75%	88.18%	95.38%	100.00%	100.00%	98.18%	99.24%	72.99 %	93.26 %
4000	72.09%	94.89%	98.18%	95.87%	100.00%	100.00%	88.18%	99.24%	74.17 %	93.56 %
4500	69.92%	93.28%	87.33%	94.85%	100.00%	99.52%	95.56%	100.00%	69.64 %	92.56 %
5000	72.96%	90.13%	100.00%	98.33%	100.00%	100.00%	90.00%	96.97%	73.50 %	93.55 %
5500	72.31%	96.47%	98.18%	97.69%	95.56%	92.86%	95.14%	99.24%	71.50 %	93.43 %
6000	72.80%	95.31%	99.09%	97.14%	100.00%	92.86%	100.00%	99.24%	73.21 %	94.56 %
6500	73.46%	95.83%	100.00%	99.09%	100.00%	100.00%	89.09%	99.24%	77.45 %	94.59 %
7000	74.05%	96.45%	98.18%	100.00%	100.00%	100.00%	89.09%	100.00%	76.81 %	94.72 %
7500	73.10%	96.45%	98.18%	100.00%	89.09%	100.00%	98.33%	100.00%	75.70 %	94.39 %
8000	72.71%	96.45%	98.18%	100.00%	89.09%	100.00%	98.33%	100.00%	75.73 %	94.35 %

(3) Negative samples improve the model performance

4000-iteration training without negative samples (2 out of 36 mislabels)

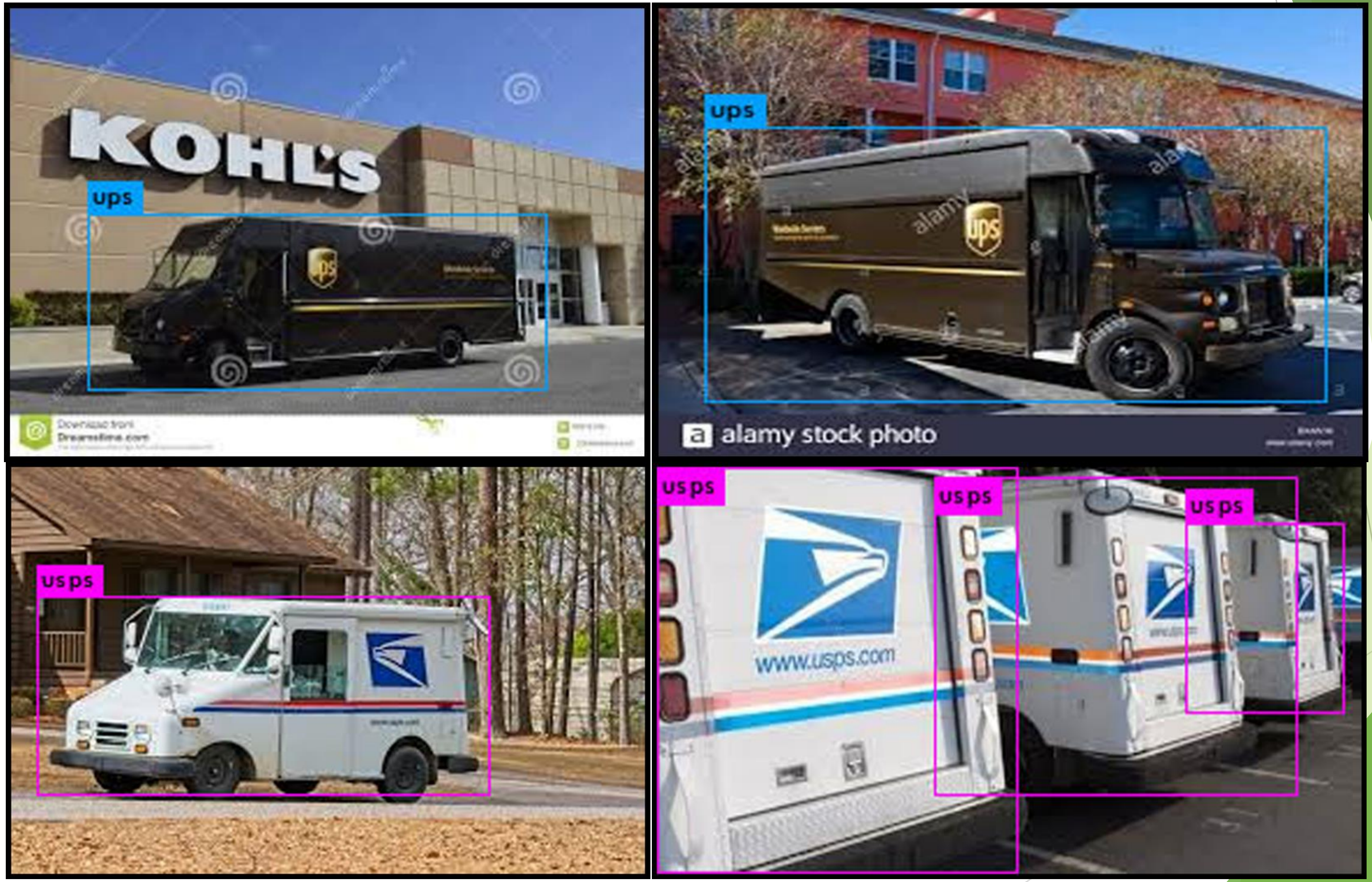
4000-iteration training using negative samples (0 out of 36 mislabels)





- Using negative samples in training helps to discriminate non-truck objects

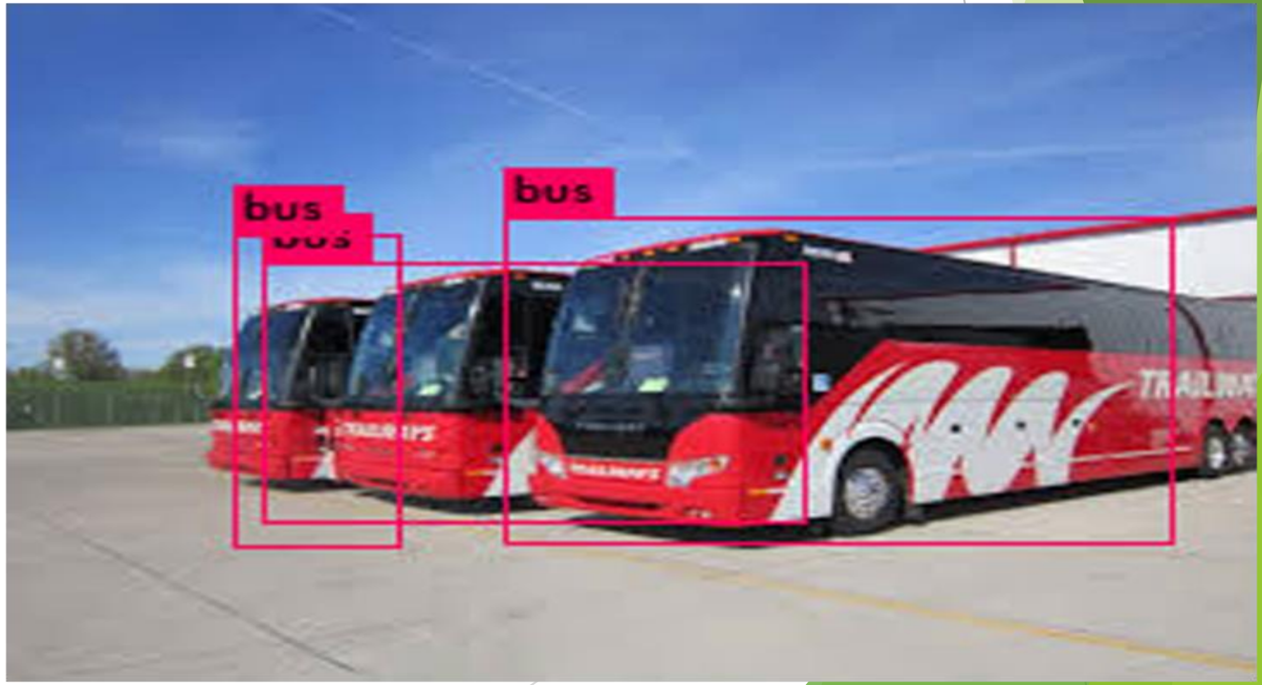
(4) Model for positive sample inference (Train with negative samples)

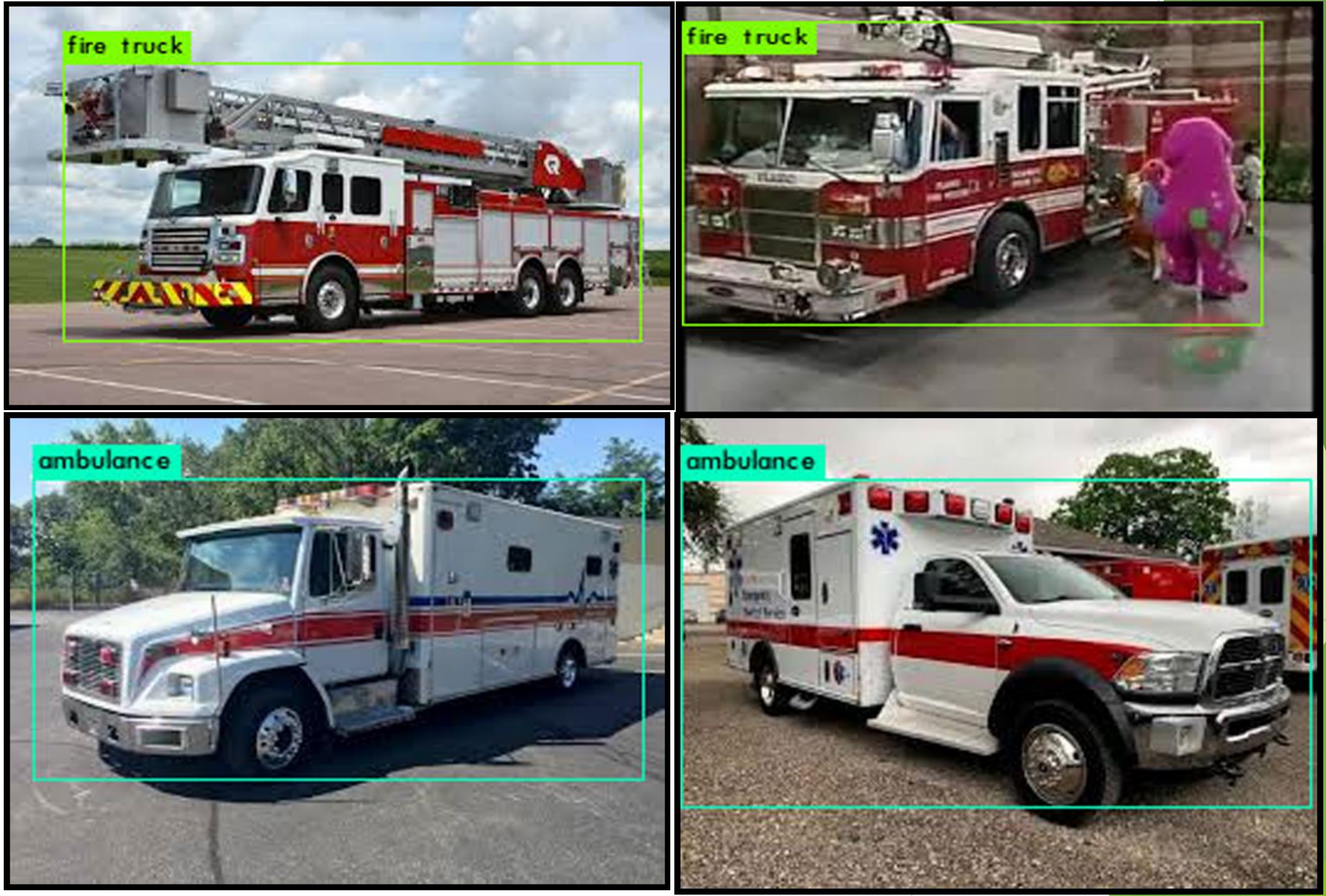


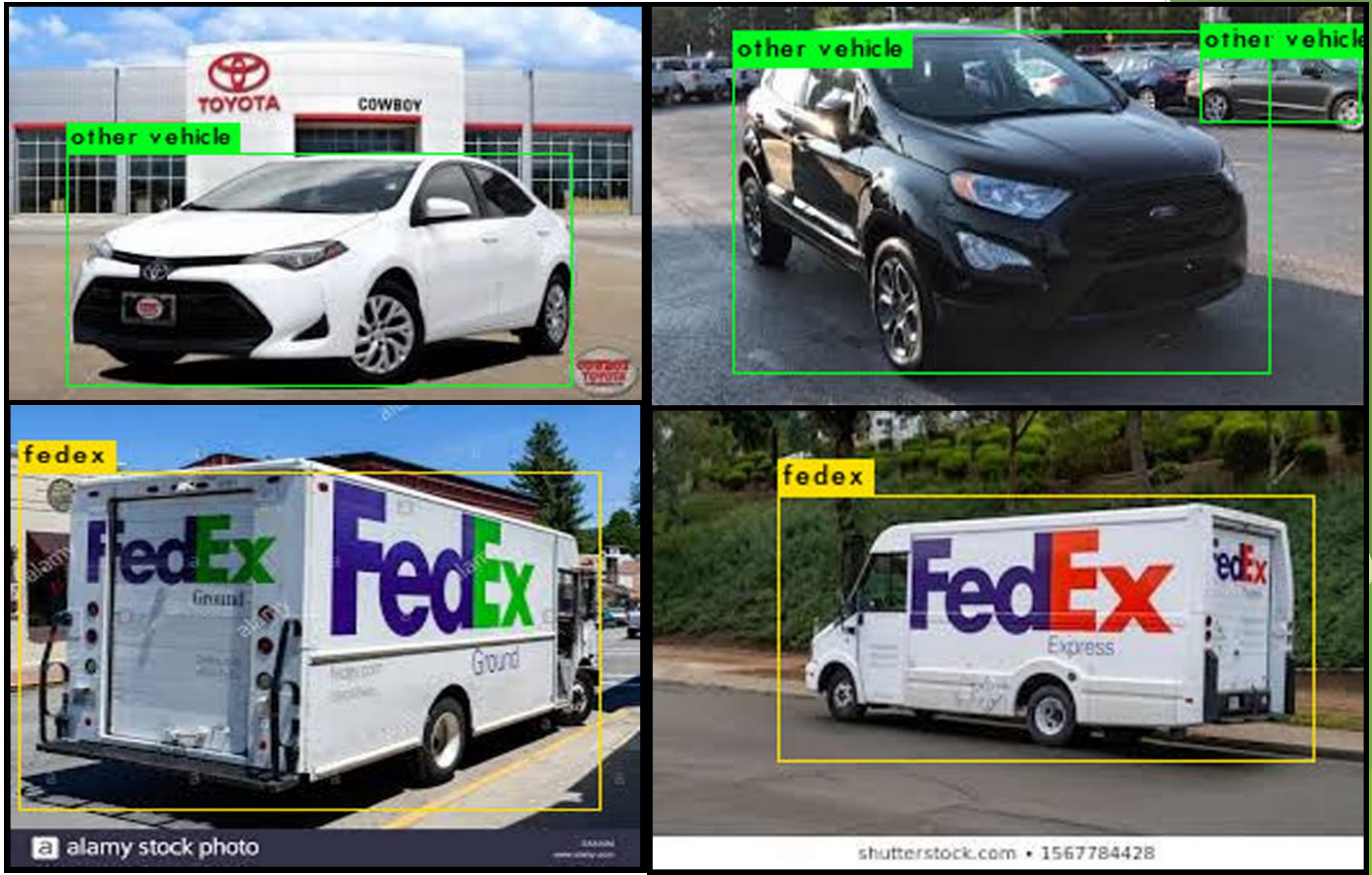
police car
police car
POLICE
POLICE

police car
POLICE
BARNWELL

bus
HELE-ON


bus
bus
bus


fire truck
fire truck
ambulance
ambulance


TOYOTA
COWBOY
other vehicle
other vehicle
other vehicle
fedex
FedEx
Ground
FedEx
Ground
a alamy stock photo
fedex
FedEx
Express
shutterstock.com • 1567784428

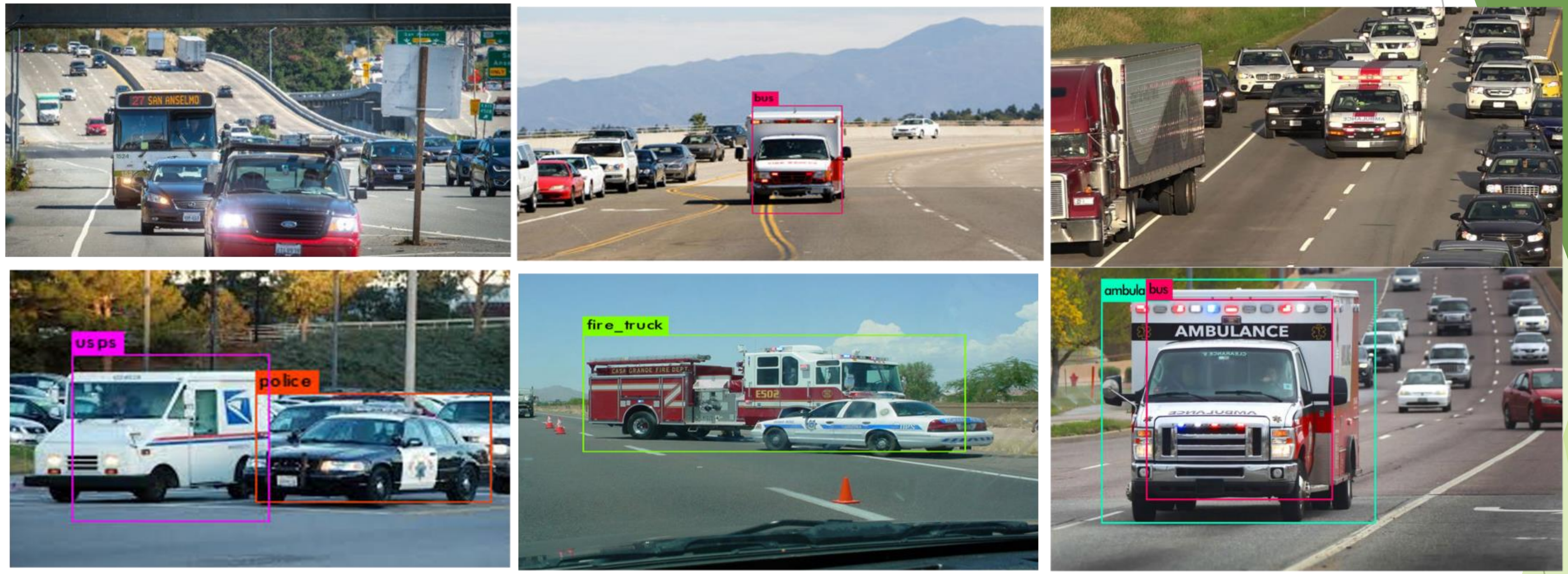
(5) Model for Video Inference

Using Microsoft movies editor we join several small pieces of trucks movies together, so we can detect all types of trucks/vehicles in one video. The length of time: 4:30 min.

We apply the 8000 iterations weights for the video inference (the notebook is attached).

Video samples: AllVehicles.mp4
Animation show (below)

(6) What we learned



- ❑ Sample size with 60 images for each class is too small; And the images without small size of objects made the model can’t detect small vehicles.
- ❑ Negative samples help machine to detect the object correctly, the more the better.
- ❑ Fine tune learning to speed up the training.
- ❑ Although darknet has built-in image augmentations, in future we still need to make physical augmented images with different view angles so that the model can improve the performance much better.