YANGON TECHNOLOGICAL UNIVERSITY
Department of Mechatronic Engineering

Efficient foreground analysis for real-time surveillance using Transfer Learning

Interim Report Seminar I

29-7-2020
Wednesday

Supervised by

Dr. Phyu Phyu Htun

Presented by

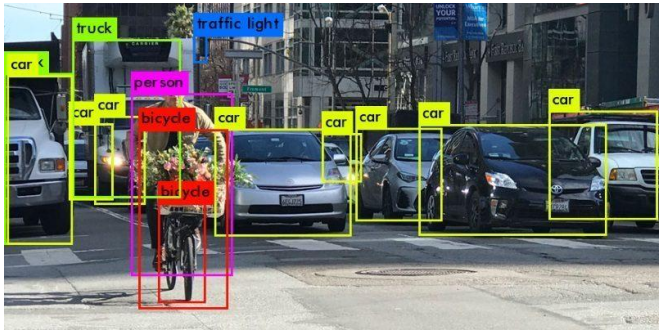
Aw Thura (VI-McE-11)

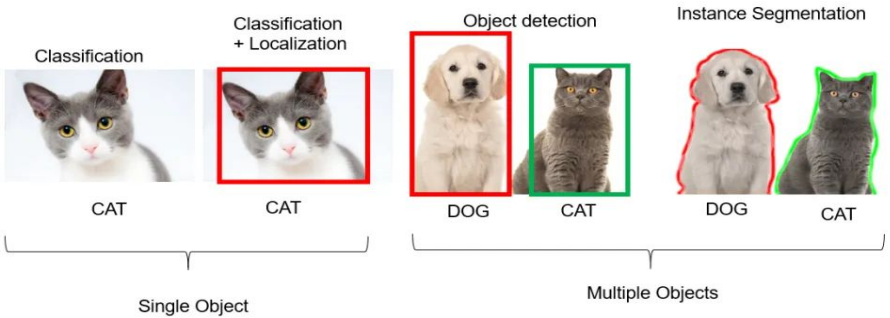
Outline

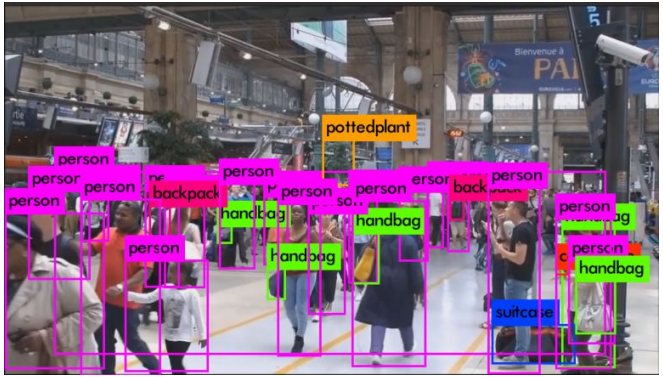
- Background & Problem Statement
- Methodology
- Contributions
- Results & Evaluation
- Conclusion
- Future Works

Background & Problem Statement

- The need of a **customized object detection model** dedicated **for surveillance & self driving cars**
- **Possibility of modification** in state of the art **YOLOv4 model** in such a way that weights can be concentrated **to detect specific classes**.



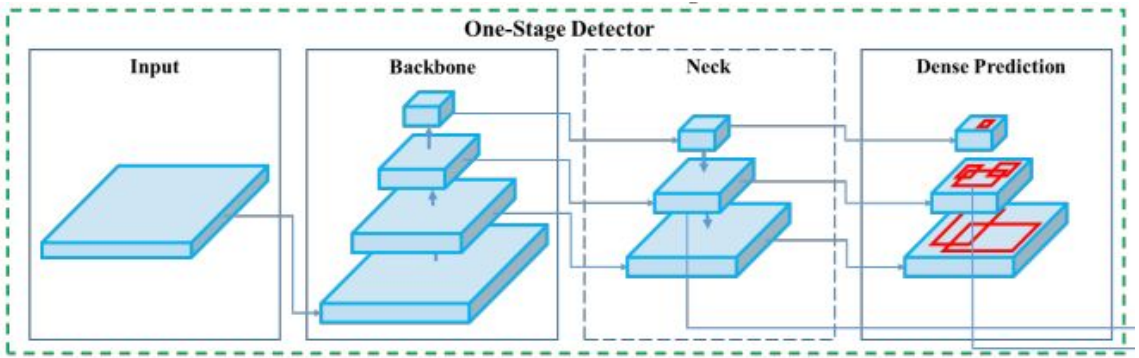




Methodology

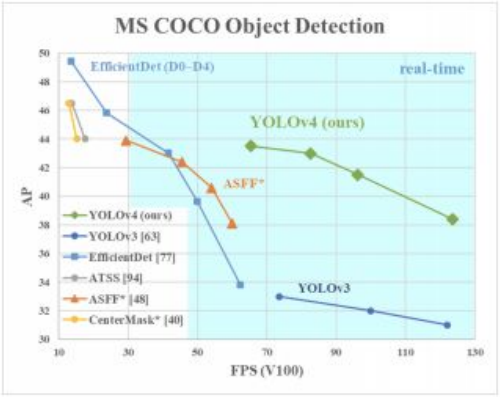
Model Architecture (YOLOv4 default)

- Backbone (CSPDarknet53)
  Feature Extraction
- Neck (PANet)
  Feature Aggregation
- Head (YoLo)
  Dense Prediction



Features

- Bag of Freebies
- Bag of Specials



Methodology(Continued)

Model Architecture (YOLOv4 default)

- **Backbone (CSPDarknet53) Feature Extraction**
- Neck (PANet)
  Feature Aggregation
- Head (YoLo)
  Dense Prediction

Features

- Bag of Freebies
- Bag of Specials

	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			

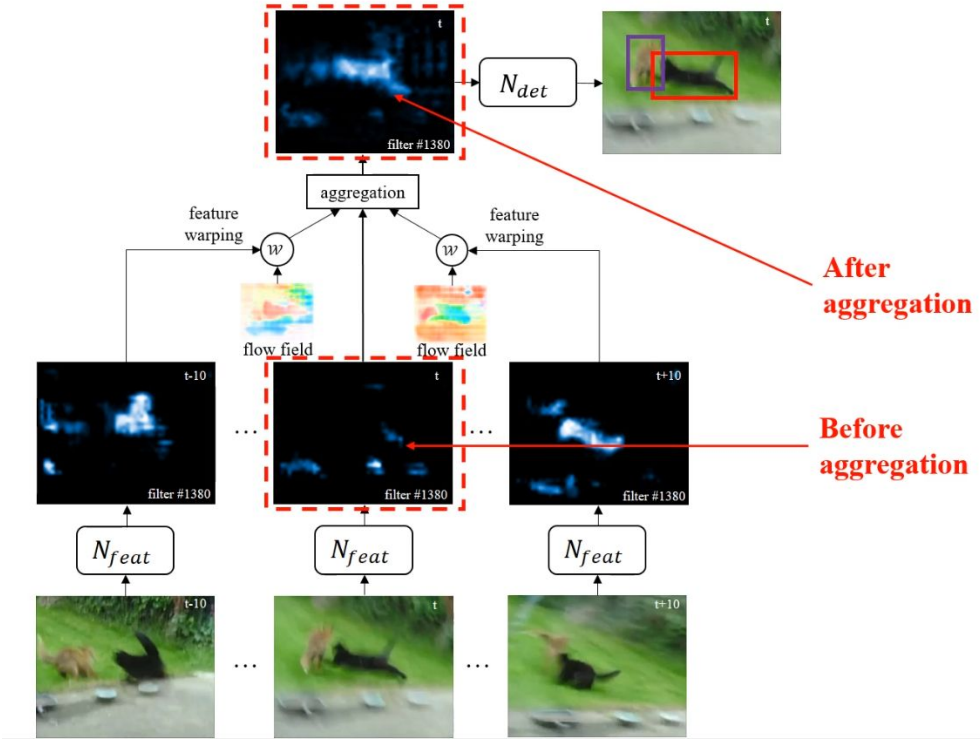
Methodology(Continued)

Model Architecture (YOLOv4 default)

- Backbone (CSPDarknet53) Feature Extraction
- **Neck (PANet) Feature Aggregation**
- Head (YoLo) Dense Prediction

Features

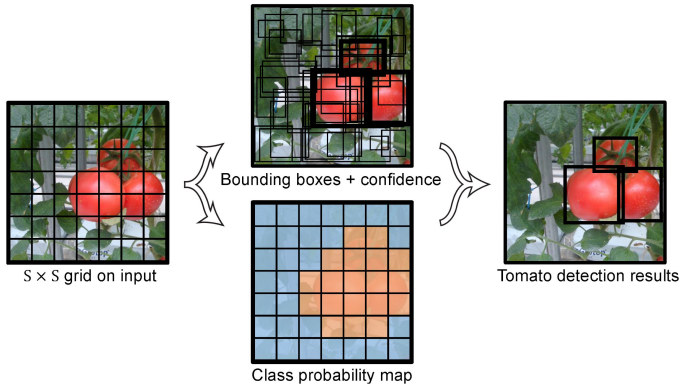
- Bag of Freebies
- Bag of Specials



Methodology(Continued)

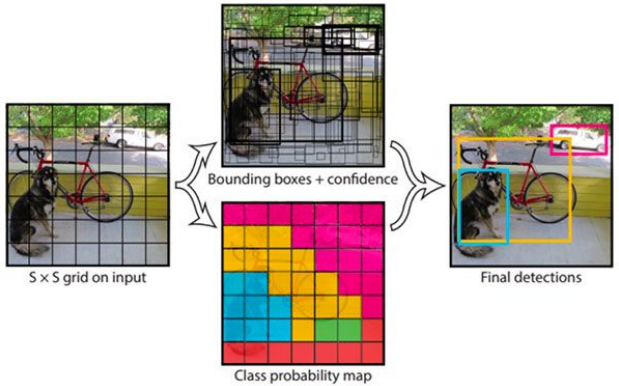
Model Architecture (YOLOv4 default)

- Backbone (CSPDarknet53)
  Feature Extraction
- Neck (PANet)
  Feature Aggregation
- **Head (YoLo)**
  Dense Prediction



Features

- Bag of Freebies
- Bag of Specials



Methodology(Continued)

Model Architecture (YOLOv4 default)

- Backbone (CSPDarknet53) Feature Extraction
- Neck (PANet) Feature Aggregation
- Head (YoLo) Dense Prediction

Features

- Bag of Freebies
- Bag of Specials

	Backbone	Detector
Bag of Freebies (BoF)	• CutMix • Mosaic data augmentation • DropBlock • Class label smoothing	• CIoU-loss • Cross mini-Batch Normalization • DropBlock • Mosaic data augmentation • Self-Adversarial Training • Multiple anchors for a single ground truth • Cosine annealing scheduler • Optimal hyperparameters • Random training shapes
Bag of Specials (BoS)	• Mish activation • Cross-stage partial connections (CSP) • Multi-input weighted residual connections (MiWRC)	• Mish activation • SPP-block • SAM-block • PAN path-aggregation block • DIoU-NMS

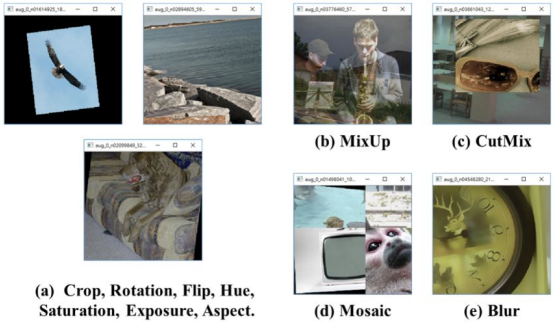
Methodology(Continued)

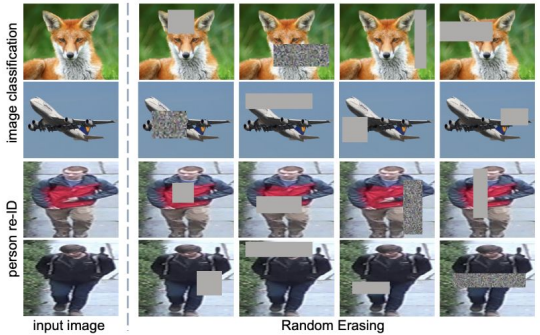
Model Architecture (YOLOv4 default)

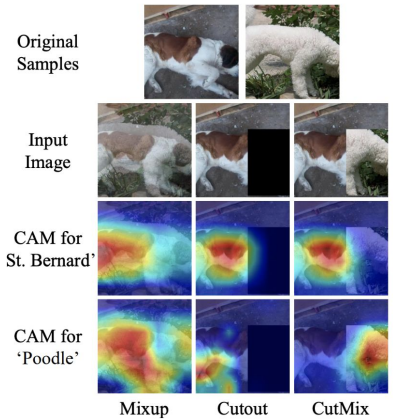
- Backbone (CSPDarknet53) Feature Extraction
- Neck (PANet) Feature Aggregation
- Head (YoLo) Dense Prediction

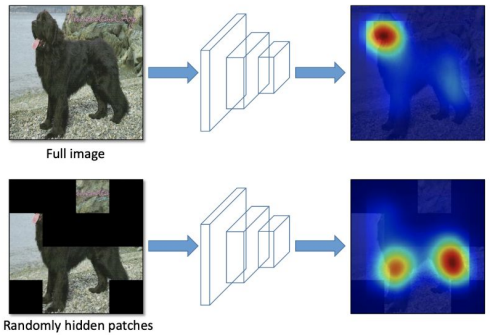
Features

- **Bag of Freebies**
- Bag of Specials









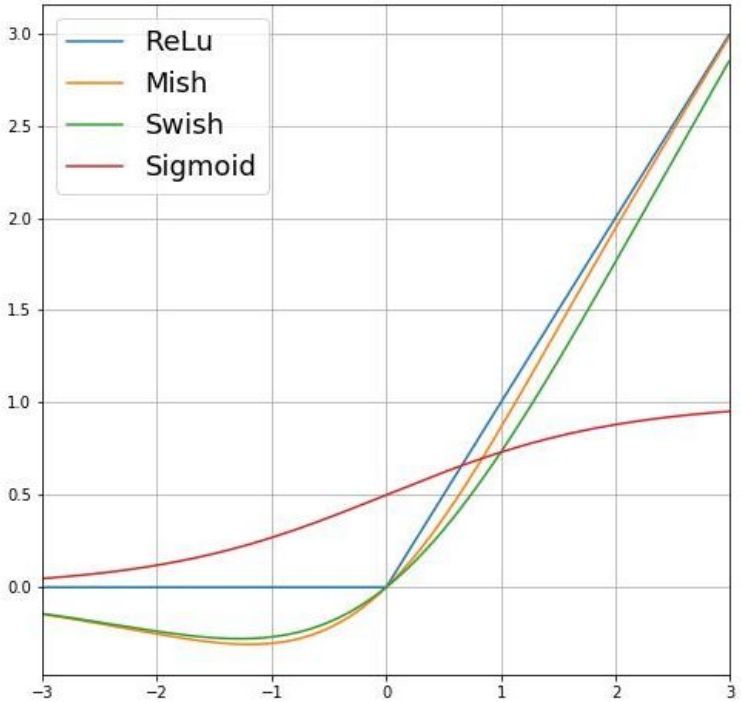
Methodology(Continued)

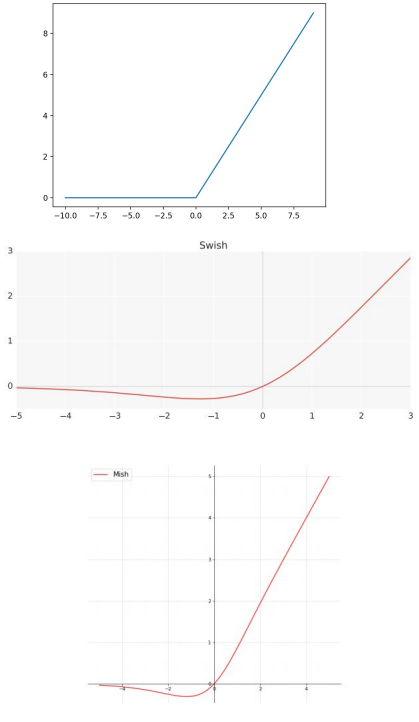
Model Architecture (YOLOv4 default)

- Backbone (CSPDarknet53)
  Feature Extraction
- Neck (PANet)
  Feature Aggregation
- Head (YoLo)
  Dense Prediction

Features

- Bag of Freebies
- **Bag of Specials**





Contributions

Contributions

- Model **customization**
- **Optimization** of training parameters
- **Dataset Preparation & adjustment** for specific classes
- Training the model with **different class combinations** and settings on **multiple datasets** to optimize the results.

Contributions (Continued)

Model customization

- Number of classes reduced (from 80 to 3)
- Layers 1 to 137 of Yolov4 are left unchanged. Weights from layer 137 are downloaded for training.
- Layer 138 to Layer 161 of Yolov4 is modified to train reduced number of classes
- All the weights are now concentrated on specific classes.

```
 138 conv     24       1 x 1/ 1     56 x  56 x 256 ->   56 x  56 x  24 0.039 BF
 139 yolo
[yolo] params: iou loss: ciou (4), iou_norm: 0.07, cls_norm: 1.00, scale_x_y: 1.20
nms_kind: greedynms (1), beta = 0.600000
 140 route  136                                           ->   56 x  56 x 128
 141 conv    256       3 x 3/ 2     56 x  56 x 128 ->   28 x  28 x 256 0.462 BF
 142 route  141 126                                       ->   28 x  28 x 512
 143 conv    256       1 x 1/ 1     28 x  28 x 512 ->   28 x  28 x 256 0.206 BF
 144 conv    512       3 x 3/ 1     28 x  28 x 256 ->   28 x  28 x 512 1.850 BF
 145 conv    256       1 x 1/ 1     28 x  28 x 512 ->   28 x  28 x 256 0.206 BF
 146 conv    512       3 x 3/ 1     28 x  28 x 256 ->   28 x  28 x 512 1.850 BF
 147 conv    256       1 x 1/ 1     28 x  28 x 512 ->   28 x  28 x 256 0.206 BF
 148 conv    512       3 x 3/ 1     28 x  28 x 256 ->   28 x  28 x 512 1.850 BF
 149 conv     24       1 x 1/ 1     28 x  28 x 512 ->   28 x  28 x  24 0.019 BF
 150 yolo
[yolo] params: iou loss: ciou (4), iou_norm: 0.07, cls_norm: 1.00, scale_x_y: 1.10
nms_kind: greedynms (1), beta = 0.600000
 151 route  147                                           ->   28 x  28 x 256
 152 conv    512       3 x 3/ 2     28 x  28 x 256 ->   14 x  14 x 512 0.462 BF
 153 route  152 116                                       ->   14 x  14 x1024
 154 conv    512       1 x 1/ 1     14 x  14 x1024 ->   14 x  14 x 512 0.206 BF
 155 conv   1024       3 x 3/ 1     14 x  14 x 512 ->   14 x  14 x1024 1.850 BF
 156 conv    512       1 x 1/ 1     14 x  14 x1024 ->   14 x  14 x 512 0.206 BF
 157 conv   1024       3 x 3/ 1     14 x  14 x 512 ->   14 x  14 x1024 1.850 BF
 158 conv    512       1 x 1/ 1     14 x  14 x1024 ->   14 x  14 x 512 0.206 BF
 159 conv   1024       3 x 3/ 1     14 x  14 x 512 ->   14 x  14 x1024 1.850 BF
 160 conv     24       1 x 1/ 1     14 x  14 x1024 ->   14 x  14 x  24 0.010 BF
 161 yolo
```

Contributions (Continued)

Optimization of training parameters

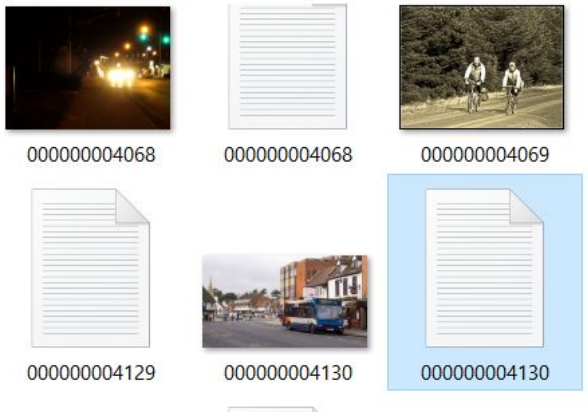
- Several training parameters can be tweaked.
- For optimization, number of subdivisions, learning rate are altered & cutmix method is also tested for data augmentation
- Image resizing can also be altered and thus, experimented.

```
[net]
# Testing
#batch=1
#subdivisions=1
# Training
batch=64
subdivisions=32
width=448
height=448
channels=3
momentum=0.949
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1

learning_rate=0.001
burn_in=1000
max_batches = 6000 #2*numclasses #4000_for_one_class
policy=steps
steps=4800,5400 #0.8*maxbatches,0.9*maxbatches
scales=.1,.1

#cutmix=1
mosaic=1
```

Contributions (Continued)

Dataset Preparation & adjustment

- Coco dataset & Google's OpenImage dataset are used for training with train-test ratio of 90:10 & 80:20 respectively.
- Coco dataset does not allow separation of classes upon download and thus, is separated with extra coding
- All the annotation files are converted to Yolo format.
- Corrupted images and annotations are inspected and labelled manually using labelImg




```
1 0.29396875 0.654224376731302 0.029343750000000002 0.03614958448753463
1 0.3635312500000003 0.6543490304709141 0.03821875 0.05490304709141274
2 0.6704062500000001 0.6645983379501385 0.38050000000000006 0.4136288088642659
0 0.2158437500000003 0.66494459833795 0.013531250000000002 0.0948753462603878l
0 0.18267187500000004 0.6737396121883656 0.015562500000000002 0.08188365650969529
1 0.2704140625 0.6542797783933518 0.019546875000000002 0.02878116343490305
1 0.25314843750000005 0.6557617728531856 0.028609375 0.03795013850415512
1 0.22245312500000003 0.6676731301939058 0.046125 0.0746814404432133
0 0.055375 0.6728531855955678 0.029343750000000002 0.08742382271468144
0 0.40247656249999997 0.6554986149584489 0.007671875000000001 0.04373961218836565
0 0.467390625 0.6545290858725762 0.01209375 0.05448753462603879
0 0.026773437499999997 0.6722437673130194 0.018640625 0.09373961218836566
0 0.4137734375 0.6563434903047092 0.007859375 0.0450415512465374
0 0.4205390625 0.6548337950138504 0.009609375000000002 0.050055401662049866
1 0.142921875 0.6329639889196675 0.029281249999999998 0.03695290858725762
```

Contributions (Continued)

Training the model

- The model is trained in three different styles to compare & analyze the results
  - Version 1 (Single class separate detectors)
  - Version 2 (Two class detector)
  - Version 3 (Three class detector)

- Version 1 & 2 are trained with OpenImage dataset and Version 3 is trained on Coco dataset

- The trained models are then evaluated with both training & testing datasets.

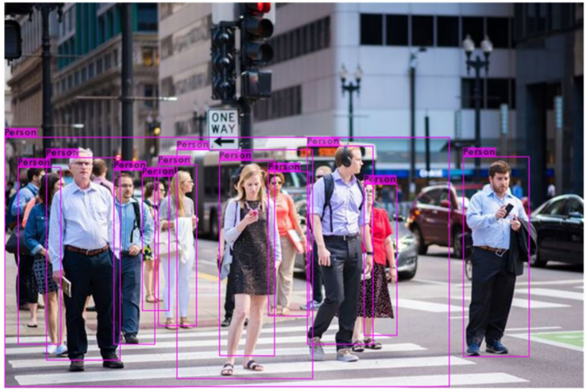
Results & Evaluation

Version 1 (Single Class Detectors)

- Single class detectors are first trained for “car” & “person” classes respectively with 4000 images each from Google’s OpenImage dataset.

- The models are trained for 2000 batches and achieved final loss values of 2.6 for “car” and 2.8 for “person”

- The training time is approximately 9 hours for each class.



Results & Evaluation(Continued)

Version 1 (Single Class Detectors)

- Prediction time is ~98.5 ms for each frame with 12 FPS. FPS can be doubled to 24 by using 1000 iteration weights, thus, sacrificing accuracy.

- MAP values are 75% for car class and 68% for person class respectively tested on a subset of OpenImage's training dataset.

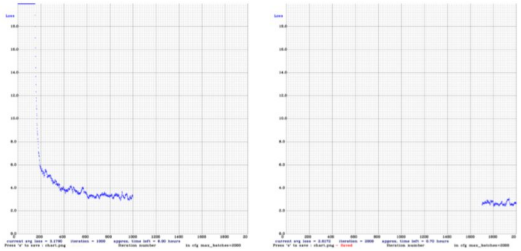


Fig 1: Loss Graphs of car detector after training 1000 & 2000 batches

```
 calculation mAP (mean average precision)...
 Detection layer: 139 - type = 27
 Detection layer: 150 - type = 27
 Detection layer: 161 - type = 27
180
 detections_count = 2735, unique_truth_count = 547
class_id = 0, name = Car, ap = 75.01%           (TP = 394, FP = 143)
```

```
[yolo] params: iou loss: ciou (4), iou_norm: 0.07, cls_norm: 1.00, scale_x_y: 1.05
nms_kind: greedynms (1), beta = 0.600000
Total BFLOPS 69.079
avg_outputs = 568027
 Allocate additional workspace_size = 52.43 MB
Loading weights from /mydrive/yolov4_car/backup/yolov4_custom_1000.weights...
 seen 64, trained: 64 K-images (1 Kilo-batches_64)
Done! Loaded 162 layers from weights-file
/mydrive/images/highway.jpg: Predicted in 21.231000 milli-seconds.
Car: 49%
Car: 78%
Car: 70%
Car: 43%
Car: 50%
Car: 50%
Unable to init server: Could not connect: Connection refused
```

Fig 4(a): Detection time and confidence levels (1000 batches)

```
[yolo] params: iou loss: ciou (4), iou_norm: 0.07, cls_norm: 1.00, scale_x_y: 1.05
nms_kind: greedynms (1), beta = 0.600000
Total BFLOPS 69.079
avg_outputs = 568027
 Allocate additional workspace_size = 3.61 MB
Loading weights from /mydrive/yolov4_car/backup/yolov4_custom_final.weights...
 seen 64, trained: 128 K-images (2 Kilo-batches_64)
Done! Loaded 162 layers from weights-file
/mydrive/images/highway.jpg: Predicted in 98.282000 milli-seconds.
Car: 65%
Car: 81%
Car: 97%
Car: 48%
Car: 41%
Car: 48%
Car: 51%
Car: 95%
Car: 90%
Car: 61%
Car: 89%
Car: 52%
```

Fig 4(b): Detection time and confidence levels (2000 batches)

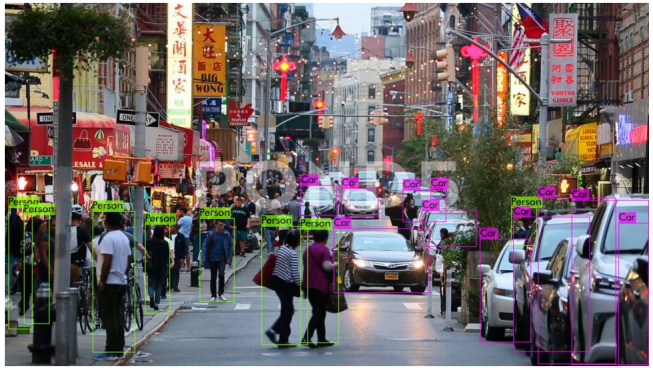
Results & Evaluation(Continued)

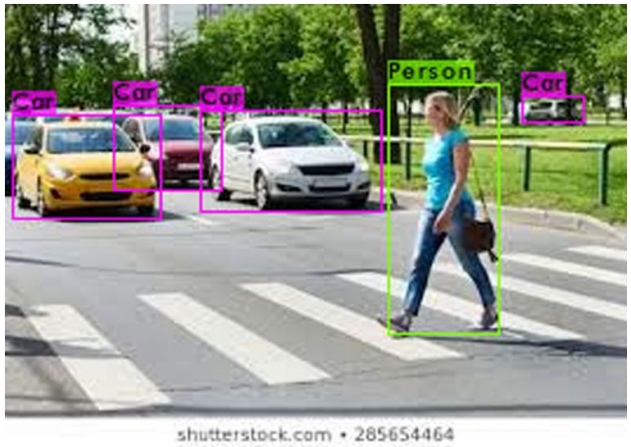
Version 1 (Single Class Detectors)

Results & Evaluation(Continued)

Version 2 (Two Class Detector)

- "Car" and "Person" classes are trained together as a multiclass detector, with 5000 images from Google's OpenImage dataset, 2500 images for each class.
- The model is trained for 4000 batches final loss value is stuck at 3.5
- The model is then trained again with same settings until 5000 batches and loss further decreased to 3.2
- The total training time is approximately 16 hours.

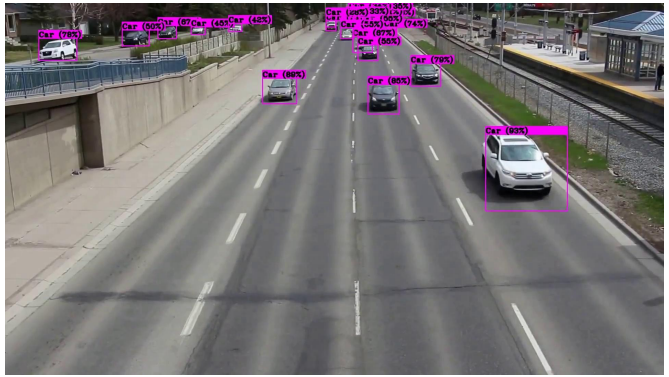




Results & Evaluation(Continued)

Version 2 (Two Class Detector)

- Prediction time remains at ~98.5 but FPS drops to 11.5
- MAP values are not impressive yet. After 5000 batches trained, Car class saw an increase with 83.28% MAP on OpenImage testing dataset but person class MAP is at a mediocre 48.3% , thus averaging the MAP of this detector to 65.8%
- 4000 batches version previously achieved average MAP of 64.62% and thus it is a 1.2% increase.

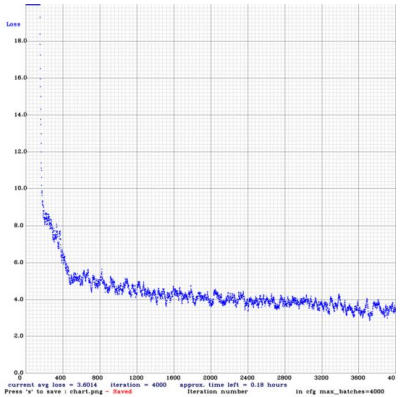


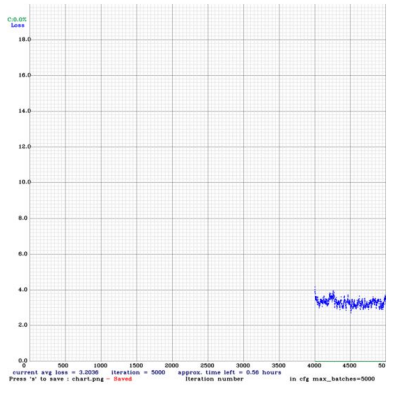
Results & Evaluation(Continued)

Version 2 (Two Class Detector)

```
[yolo] params: iou loss: ciou (4), iou_norm: 0.07, cls_norm: 1.00, scale_x_y: 1.05
nms_kind: greedynms (1), beta = 0.600000
Total BFLOPS 69.087
avg_outputs = 568179
 Allocate additional workspace_size = 52.43 MB
Loading weights from /mydrive/yolov4_twoclass/backup/yolov4_custom_final.weights...
 seen 64, trained: 320 K-images (5 Kilo-batches_64)
Done! Loaded 162 layers from weights-file

 calculation mAP (mean average precision)...
 Detection layer: 139 - type = 27
 Detection layer: 150 - type = 27
 Detection layer: 161 - type = 27
388
 detections_count = 4704, unique_truth_count = 828
class_id = 0, name = Car, ap = 83.28%           (TP = 324, FP = 103)
class_id = 1, name = Person, ap = 48.30%        (TP = 254, FP = 322)

 for conf_thresh = 0.25, precision = 0.58, recall = 0.70, F1-score = 0.63
 for conf_thresh = 0.25, TP = 578, FP = 425, FN = 250, average IoU = 47.64 %

 IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
 mean average precision (mAP@0.50) = 0.657878, or 65.79 %
Total Detection Time: 12 Seconds

Set -points flag:
 `-points 101` for MS COCO
 `-points 11` for PascalVOC 2007 (uncomment `difficult` in voc.data)
 `-points 0` (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset
```

```
 calculation mAP (mean average precision)...
 Detection layer: 139 - type = 27
 Detection layer: 150 - type = 27
 Detection layer: 161 - type = 27
4004
 detections_count = 57876, unique_truth_count = 14424
class_id = 0, name = Car, ap = 86.20%           (TP = 4594, FP = 943)
class_id = 1, name = Person, ap = 83.62%        (TP = 6952, FP = 1622)

 for conf_thresh = 0.25, precision = 0.82, recall = 0.80, F1-score = 0.81
 for conf_thresh = 0.25, TP = 11546, FP = 2565, FN = 2878, average IoU = 65.00 %

 IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
 mean average precision (mAP@0.50) = 0.849090, or 84.91 %
Total Detection Time: 321 Seconds
```



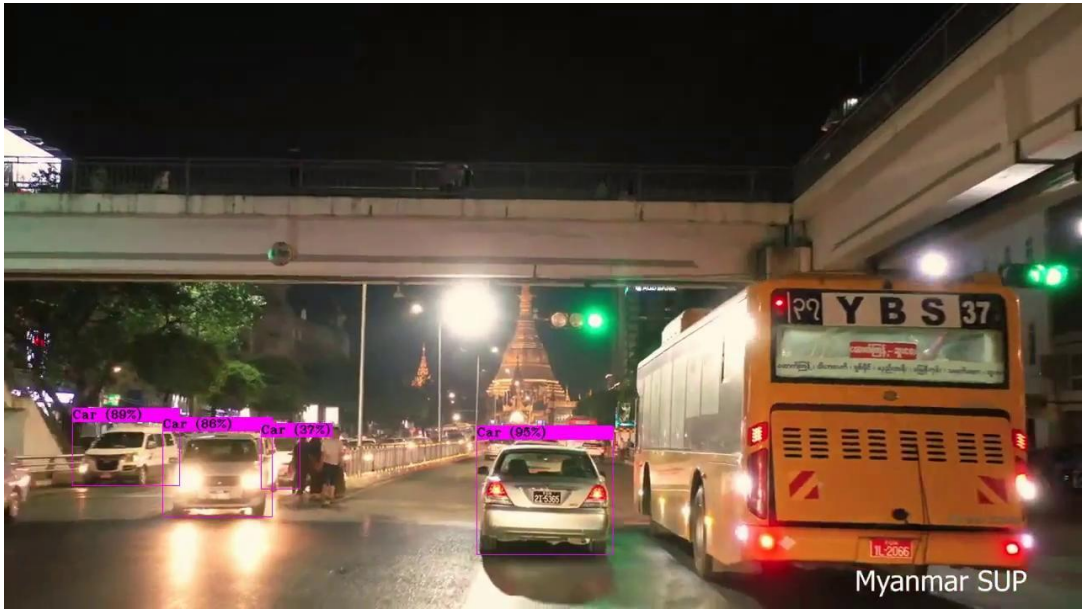


Results & Evaluation(Continued)

Version 2 (Two Class Detector)

- As we can see here, this version can detect cars very well but still struggles in detecting pedestrians

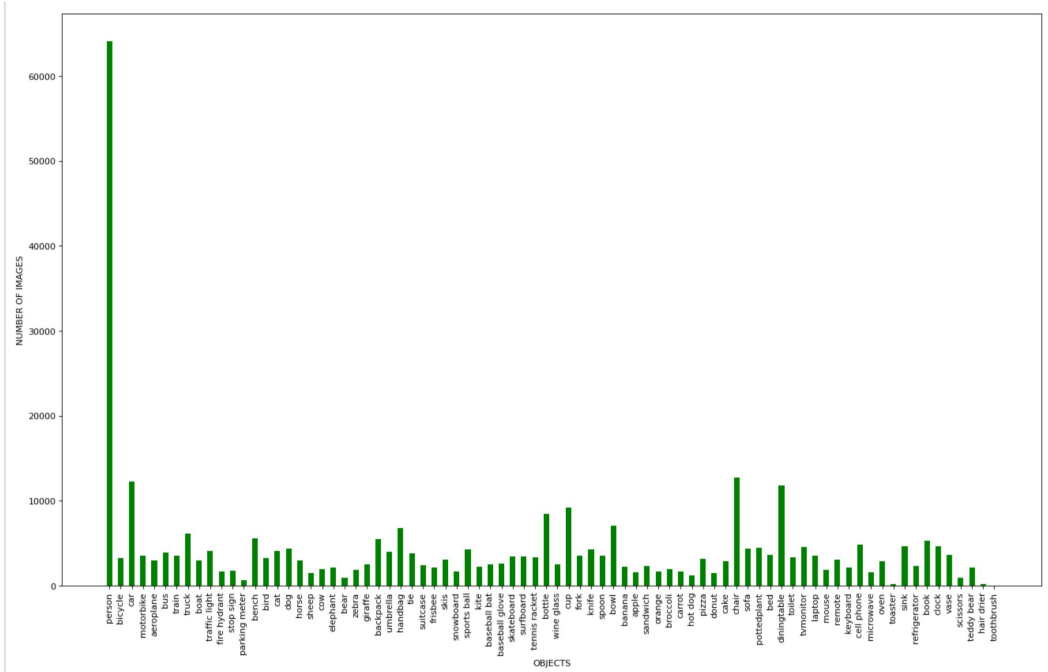
- This is concluded to be the cause of small size of the data used and the problem is tackled in the next version



Results & Evaluation(Continued)

Version 3 (Three Class Detector)

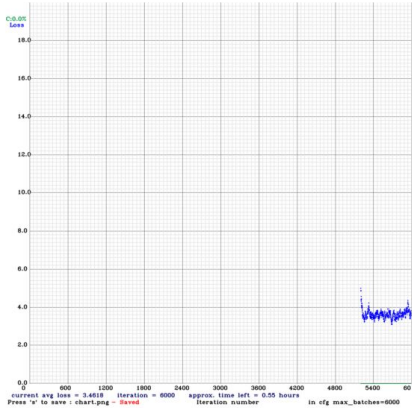
- “Bus” class is added and the three classes are trained with 65,000 images from COCO dataset, 13 times more images than the previous version
- COCO dataset distribution means unbalanced data, but the person class seem to require more data due to previous findings
- After training for target 6000 batches loss values remained at 3.46
- Total training time is ~15 hours



Results & Evaluation(Continued)

Version 3 (Three Class Detector)

- Prediction time is at ~98.7ms at 12.5 FPS
- Average MAP values on coco dataset is 79.3% on training set and 78.4% on validation set.
- MAP for person class is significantly increased but MAP for car class slightly decreased.




```
FPS:11.9        AVG_FPS:12.5
```

```
/mydrive/images/highway.jpg: Predicted in 98.733000 milli-seconds.
car: 12%
car: 66%
car: 87%
car: 13%
car: 97%
car: 57%
car: 14%
car: 56%
car: 94%
car: 97%
car: 79%
car: 95%
car: 71%
Unable to init server: Could not connect: Connection refused
```

Results & Evaluation(Continued)

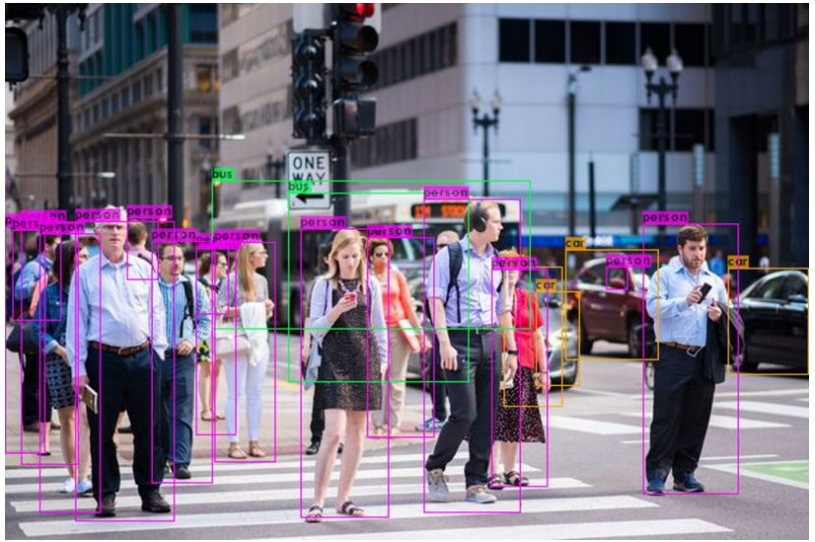
Version 3 (Three Class Detector)

```
[yolo] params: iou loss: ciou (4), iou_norm: 0.07, cls_norm: 1.00, scale_x_y: 1.05
nms_kind: greedynms (1), beta = 0.600000
Total BFLOPS 69.096
avg_outputs = 568332
 Allocate additional workspace_size = 3.61 MB
Loading weights from /mydrive/yolov4_twoclass/backup/yolov4_custom_last.weights...
 seen 64, trained: 384 K-images (6 Kilo-batches_64)
Done! Loaded 162 layers from weights-file

 calculation mAP (mean average precision)...
 Detection layer: 139 - type = 27
 Detection layer: 150 - type = 27
 Detection layer: 161 - type = 27
12800Premature end of JPEG file
12804Can't open label file. (This can be normal only if you use MSCOCO): data/obj/000000390969.txt
45964
 detections_count = 1131470, unique_truth_count = 206344
class_id = 0, name = person, ap = 79.90%        (TP = 132534, FP = 44925)
class_id = 1, name = car, ap = 70.21%           (TP = 19600, FP = 8322)
class_id = 2, name = bus, ap = 87.71%           (TP = 3351, FP = 608)

 for conf_thresh = 0.25, precision = 0.74, recall = 0.75, F1-score = 0.75
 for conf_thresh = 0.25, TP = 155485, FP = 53855, FN = 50859, average IoU = 60.36 %

 IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
 mean average precision (mAP@0.50) = 0.792721, or 79.27 %
Total Detection Time: 3600 Seconds

Set -points flag:
 `-points 101` for MS COCO
 `-points 11` for PascalVOC 2007 (uncomment `difficult` in voc.data)
 `-points 0` (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset
```

```
 calculation mAP (mean average precision)...
 Detection layer: 139 - type = 27
 Detection layer: 150 - type = 27
 Detection layer: 161 - type = 27
88
 detections_count = 2375, unique_truth_count = 401
class_id = 0, name = person, ap = 69.31%        (TP = 117, FP = 40)
class_id = 1, name = car, ap = 72.17%           (TP = 89, FP = 34)
class_id = 2, name = bus, ap = 93.78%           (TP = 74, FP = 8)

 for conf_thresh = 0.25, precision = 0.77, recall = 0.70, F1-score = 0.73
 for conf_thresh = 0.25, TP = 280, FP = 82, FN = 121, average IoU = 63.27 %

 IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
 mean average precision (mAP@0.50) = 0.784172, or 78.42 %
Total Detection Time: 7 Seconds

Set -points flag:
 `-points 101` for MS COCO
 `-points 11` for PascalVOC 2007 (uncomment `difficult` in voc.data)
 `-points 0` (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset
```

Results & Evaluation(Continued)

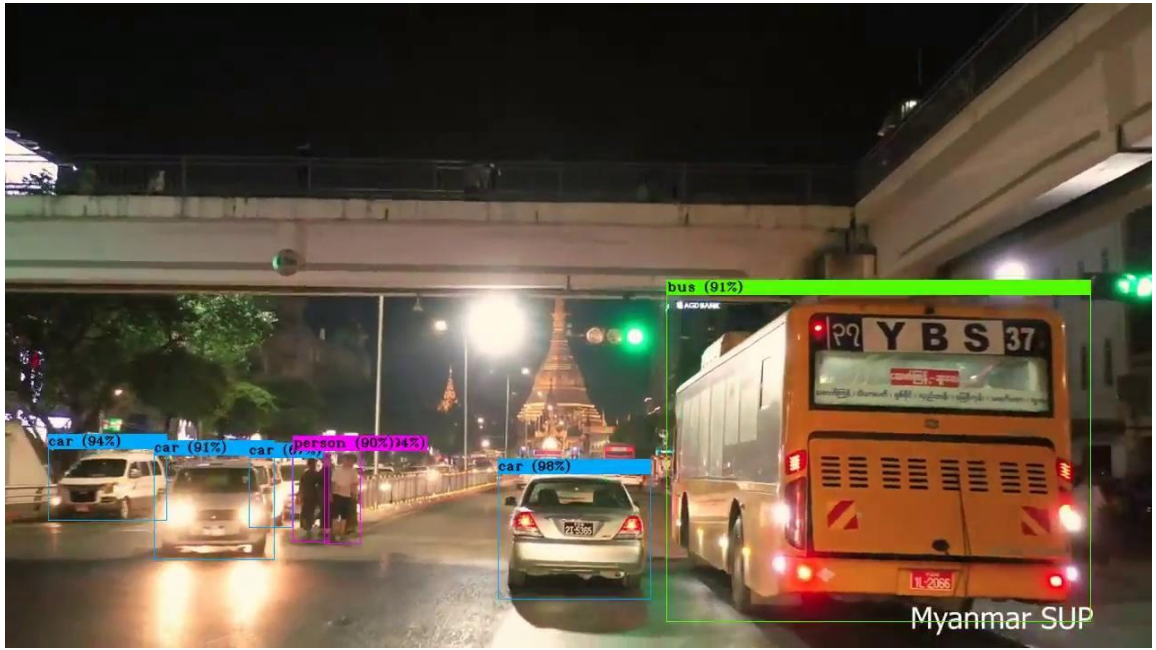
Version 3 (Three Class Detector)



Results & Evaluation(Continued)

Version 3 (Three Class Detector)

- As we can see in the video, this version runs very well and is efficient for detecting all three classes

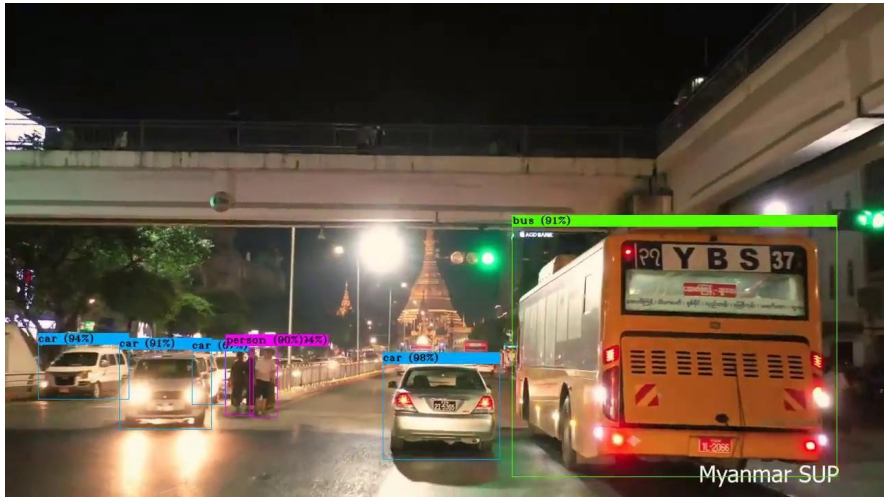


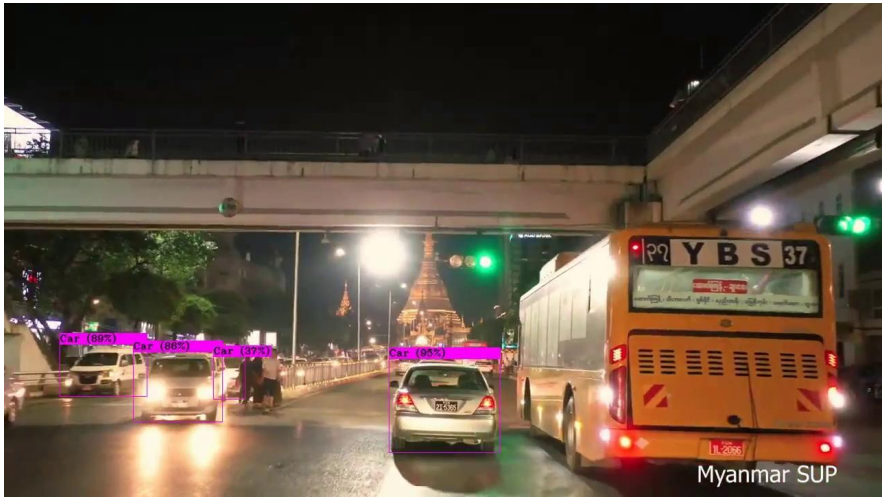
Results & Evaluation(Continued)

Version 3 (Three Class Detector) Vs Version 2 (Previous two class detector)

Results & Evaluation(Continued)

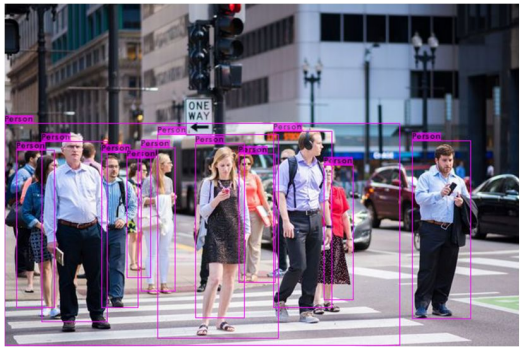
Version 3 (Three Class Detector) Vs Version 2 (Previous two class detector)

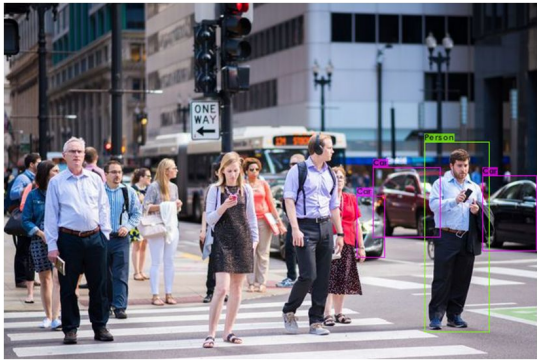


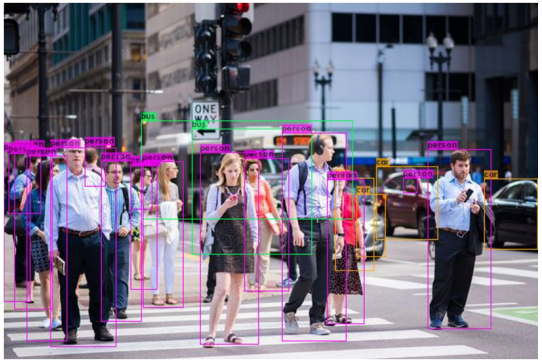


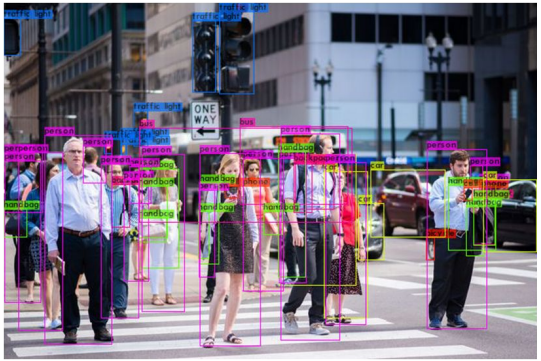
Results & Evaluation(Continued)

Comparison of three versions Vs YOLO v4 Original



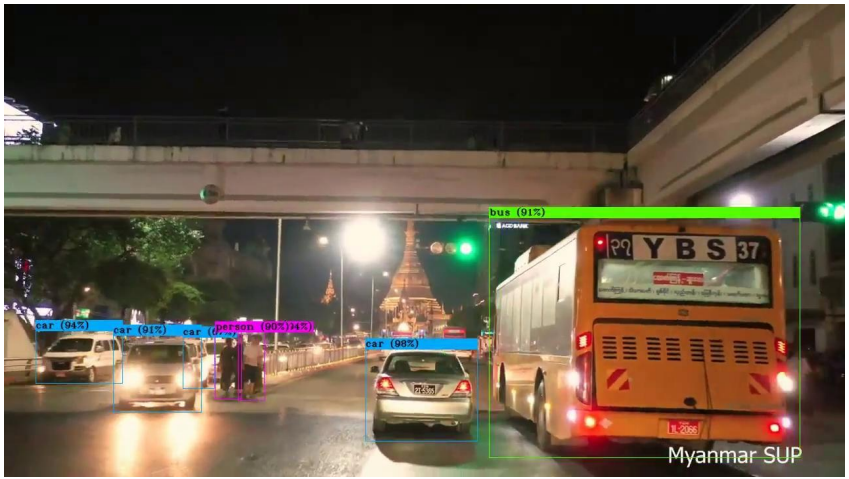


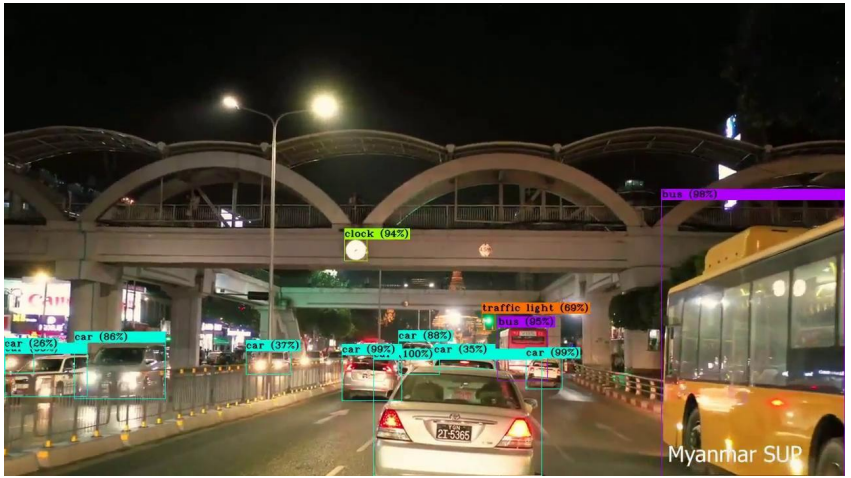




Results & Evaluation(Continued)

Version 3 (Three Class Detector) Vs YOLO v4 Original






```
FPS:11.9        AVG_FPS:12.5
```

```
FPS:7.4         AVG_FPS:7.3
Stream closed.
```

Conclusion

- Yolov4 is customized and trained in three different versions with different settings
- It is discovered that person class requires more data to perform well
- Data imbalance in version 3 causes decrease in MAP of other classes, the ratio of the dataset distribution should be tweaked to improve precision.
- This MAP can be increased by training more batches but it can eventually result in overfitting.
- In terms of performance, current version 3 almost matches the Original Yolov4 in trained classes with a better FPS rate. It can also predict images twice faster.

Future Works

- Next versions will be trained with a different data distribution ratios to further increase precision for detecting cars.
- Yolov4-tiny will also be tested as a new model to increase FPS.
- The final detection model will be combined with deep-SORT tracking algorithm to perform object tracking

End

Thank You For Your Attention.