Automated Traffic Detection in Dhaka-ai 2020 Challenge

Dr. Shaikat Galib | H2O.ai | Cedar Park, TX, USA | 13 Dec. 2020

Introduction

- Dhaka city has one of the worst traffic systems around the world, primarily due to limited road and lack of regulations.
- Automated vehicle detection can help improve traffic monitoring, regulation and public safety.
- Machine learning / deep learning methods are promising techniques for automatic detection of vehicles.
- In this competition, a traffic image dataset was collected and annotated with bounding boxes. The goal is to accurately detect different vehicles from images.
- In this work, different state-of-the-art object detection models were trained under controlled experimental conditions to find a suitable method for Dhaka traffic detection.

Dataset

Dataset stats:

- Train Images: 3,003 (at different scale and resolution)
- Total classes: 21
- Total boxes: 24,368

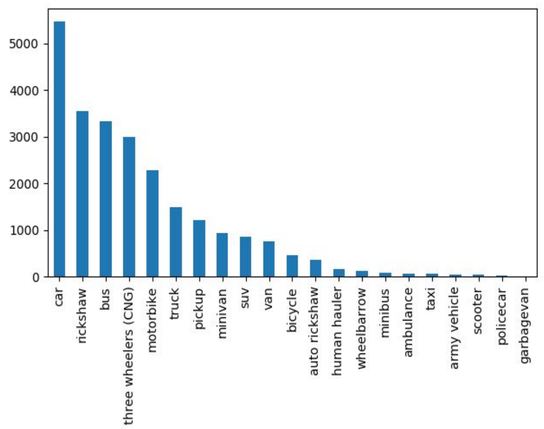


Fig: Training class distribution

Dataset challenges:

- Small dataset
- Images taken from different cameras and poses
- Class imbalance observed
- Label noise, inconsistent annotation

Crowded traffic | From motorbike | From bus

From top | Traffic at night | From side

Fig: Sample training images

Methods

- The traffic detection pipeline is built on 2 stages: **Detection and Classification**.
- At detection stage, 7 **YOLOv5x** [1] models were trained for generating the detection bounding boxes. Weighted boxes fusion (WBF) [2] approach was used to ensemble the boxes. These models were trained on **original train data and pseudo-label test data**.
- At classification stage, 4 image classification models were trained on the bounding-box crops. **EfficientNet-B4** and **CSPResNet50** backbones were used with imagenet pretrained weights. These models were then applied to refine the predictions of detection models. The models were trained on **original train data, pseudo-label data from both stage1 and stage2 test sets and external data collected by web scraping**.
- Post-processing was a part of the algorithm to further refine the results. Detection area too small or too large was discarded. Moreover, multiple class detection for a same vehicle was kept, if it satisfies a certain threshold. Rare classes were dealt with by allowing more predictions from classification models.

Experiments

Expt. 1: Baseline solution

- Baseline model was trained with a YOLOV5x architecture.
- COCO pretrained weight was used for transfer learning.
- ~2500 images were used for training and rest for validation.
- Folds were created by annotator name.

Model	YOLOV5x
Resolution	1024
Epoch	50
Learning rate scheduler	One-cycle
Augment.	Basic
LB	0.175

Fig. Training augmentations

Expt. 2: Fine-tune with strong augment (@1024)

- Finetune with strong augmentation helped improve the results. Best augmentation parameters were found using **Genetic algorithm** on COCO dataset.

Augmentations	**Pre-training (50 epoch)**	**Fine-tuning (50 epoch)**
image rotation (+/- deg)	0.0	0.373
image translation (+/- fraction)	0.1	0.245
image scale (+/- gain)	0.5	0.898
image shear (+/- deg)	0.0	0.602
image flip left-right (probability)	0.5	0.5
image mosaic (probability)	1.0	1.0
image mixup (probability)	0.0	0.243
LB	**0.175**	**0.185**

Expt. 3 & 4: Image resolution and Dataset size

- Higher resolution yielded better accuracy.
- Increasing the size of dataset by pseudo-labeling improved generalization.

Image Resolution	**LB (full train)**
800x800	0.2034
1024x1024	0.2003
1280x1280	0.2051
1600x1600	0.2088

Dataset size	**LB (@1024)**
Fold 1	0.1826
Fold 2	0.1759
Full train	0.2003
Full + Pseudo-label stg 1	0.2059

Expt. 5: Effect of Image pre-processing (@1024)

- CLAHE image contrast enhancement algo. was used to improve image quality.
- This pre-process did not improve single model results but contributed in improving the ensemble.

Processing	**LB (full tr.)**
None	0.2003
CLAHE	0.1953

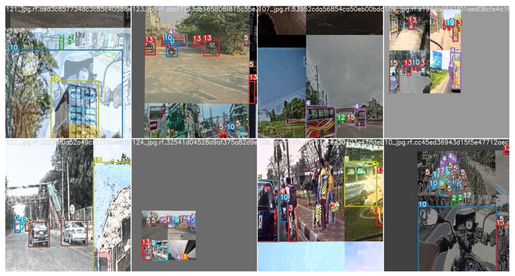
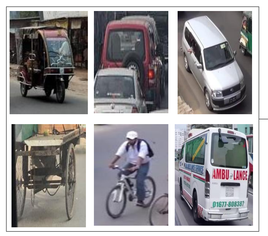


Fig. Training images using CLAHE preprocessing

Expt. 6: Effect of Classification models (@1024)



- 40 epoch
- Heavy Augmentation
- Resolution: 180, 224
- Total models: 4
- TTA applied as train augmentations (x4)

EfficientNet-B4 (x2)
CSPResNet50 (x2)
→ 21 Class

Fig. Crops from training dataset

Method	**LB**
Detection	0.2003
Detection + Classification	0.2215

Expt. 7: Add external images

- About 1000 rare class images were collected by web scraping for training classification models.

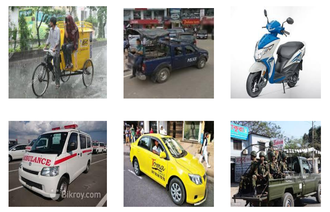


Fig: Images from the internet

Train data (crops)	**LB**
Stage 1 train	0.2215
Stage 1 Train + External	0.2350
Stage 1 Train + External +Stage1 Pseudo	0.2748

Expt. 8: Post-processing

- **Filter by confidence:** Discard detections that has confidence score < 0.1.
- **Filter by area:** Discard any detection that has area >200,000 or <200 pixels.
- **Alter predictions by class-groups:** Categorised the 21 classes into 3 categories: Rare, Medium and High. Allow more predictions from classification models for rare category than the other two.

Results

- Leaderboard scores are on stage 2 test set.
- Runtime: 3.6 sec / image for running the full pipeline.

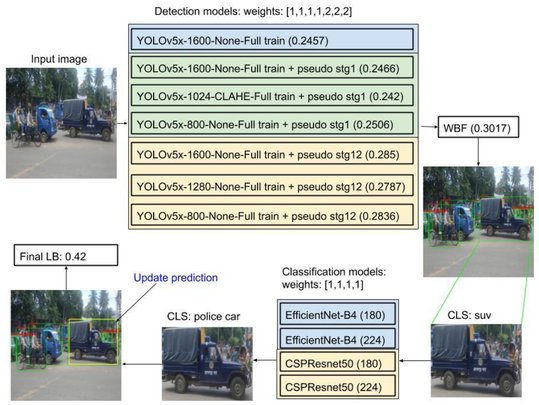


Fig. Schematic diagram of full pipeline and LB scores

Computation resources:

- 1 2080Ti and 1 TitanX GPU for most of the development.
- 4 P100 GPUs for training large models occasionally.

Conclusions

- YOLOv5 object detection architecture was studied for dhaka traffic detection problem. Adding classification models with external data helped increasing the rare class vehicle detection and overall accuracy.
- EfficientDet architectures were also experimented with. However, they yielded relatively poor results.

Reference

[1] https://github.com/ultralytics/yolov5
[2] https://github.com/ZFTurbo/Weighted-Boxes-Fusion