Real Time Detection Of Traffic Offenders In Two Wheeler Vehicles Using Computer Vision.

Geetanjali Bhola,Bruno Omondi Adul, Ibrahim Abdihakim Dahir

IJSER

Abstract

In this paper, we present a method for real time automatic detection of two wheeler traffic offenders who are not wearing helmets and tripplers(riders carrying more than one passenger) with the use of real time surveillance video. The suggested method recognizes bike riders in surveillance video and then assesses whether or not they are wearing a helmet.Also whether the bike rider is tripling or not that is he /she is carrying more than one passenger. All the violators of both helmet and tripling are all classified as offenders.

The implementation of this approach has been done using tensorflow object detection api with a pre-trained model and fine tuned to suit our problem and datasets. Since all traffic violations are classified as offenders while innocents are classified as non offenders this greatly helps improve the reliability of the system contrary to current existing systems with single modules for only helmet detection operations,The two models performed pretty well giving an average detection precision of 83%. Most law enforcement agencies do not have enough manpower to identify all traffic offenses.By deploying the system on strategic junctions traffic police can have a better grasp of two wheeler traffic offenders..

Keywords: object detection,Computer Vision,Tensorflow, Two Wheeler Vehicles, Tripling Detection, Helmet Detection,Transfer learn

Introduction

We have taken a binary approach to classify all motorists detected as either offenders or non offenders . There are several scenarios to consider.

Offensive Scenarios

1. When a motorist is alone and not wearing a helmet, he/she is an offender
2. When neither the motorist nor the pillion is wearing a helmet,it is classified as an offense
3. When the motorist has tripled and all persons on board are wearing helmets, it is still an offense,
4. When the motorist has tripled and one of the persons on board is not wearing a helmet, it is an offense

IJSER

Non offensive scenarios

1.The motorist is alone and wearing a helmet, he/she is a non offender

2. When there is a pillion , and both the motorist and the pillion are wearing helmets,they are none offenders

II. Existing Work

Chiverton suggested using circular arcs to recognize helmets in a video stream;[1] this method has a low accuracy. On the other hand it does not identify other two wheeler offenders who are tripling hence this method does not detect all traffic offenses committed by two wheeler riders.

Existing work that uses image processing techniques to tackle the challenges use technologies such as HOG[2], LBT [3]and WT. The suggested technique crops the most likely area where a helmet may be present and passes it to the feature extraction system which isolates the motorcycles from the photos by approximation, Abstraction and Mechanism for matching.

According to Sorin draghici et al, an artificial vision system based on an artificial neural network can access a camera generated image of a motorbike,find the registration plate and detect the motorcycle's registration number.[4]

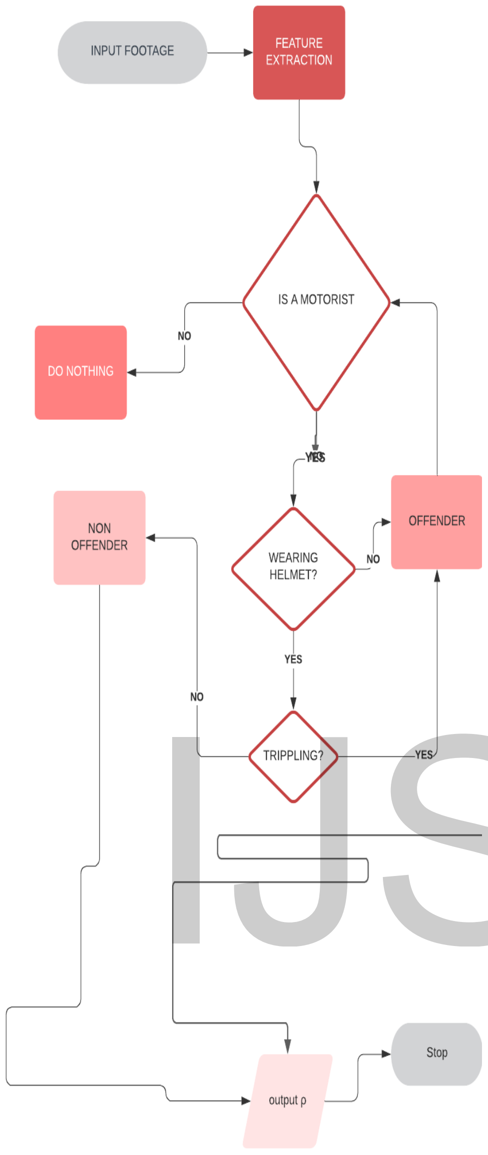
Sorin draghichi et al., developed a modular system that enables for easy upgrades and/or substitutes of various sub modules, possibly making it ideal for a wide range of visual applications. The suggested algorithm is divided into three sections ;i) Extraction of plate region ,ii) Character Segmentation, iii) Plate character recognition. Fringe detection algorithms and smearing algorithms are utilized in the segmentation phase , smearing algorithm filtering and morphological algorithms are also employed, and statistical based template matching is used for plate character identification. Realworld photos were used to test the suggested system's algorithm performance. Despite the impressive work that has been done in computer vision of motorcycles, recognition of tripling riders has not been developed yet.

III. Proposed Work

This section outlines the suggested method for detecting bike riders in real time who commit offenses by either not wearing helmets or tripling that is carrying more than one passenger. This approach has been implemented in two modules that are executed serially.. In the starting module, we expose a motorcyclist in the video frame who is not wearing a helmet. In the second module, we detect a bike rider who is tripling. An input footage has to be processed by both the modules to find out if the motorcyclist is an offender or a non-offender

Figure 1.0 Below shows a flow diagram of the implementation of the solution

IJSER



Since we are using tensorflow object detection api to reduce false predictions, we collected as much data as possible given in different environments ,different angles, augmented it and then fine tuned our pipeline to suit our dataset.

Tensorflow has pre-trained models which are saved networks that were previously trained on big datasets such as a large-scale image classification task, you may either use the pre-trained model as is or twerk it using transfer learning to personalize the current model to a given task. When a model is developed on a big and general enough dataset,it may successfully serve as a generic model of the visual world. According to transfer learning for image classification,you may then use these learned feature maps to train a huge model on a large dataset without having to build from zero.

IJSER

In this approach, we tried two ways to customize a pretrained model:

1. Feature Extraction: Use the representations learned by a previous network to extract meaningful features from new samples. We simply added a new classifier, which was trained from zero, on top of the pretrained model so that it could repurpose the feature maps learned previously for the dataset.

   There is no reason to completely

rebuild the whole model. The base convolutional network already contains features that are generically useful for classifying images. However, the final classification part of the pretrained model is specific to the original classification task, following that, it's particular to the collection of classes on which the model was trained.[5]

2. Fine-Tuning: We Unfreezed a few of both the newly-added classifier layers and the base model's final levels were concurrently trained using the top layers of a frozen model base. This allowed us to fine tune the original model's higher-order feature representations in order to make them more relevant for the specific task.[5]

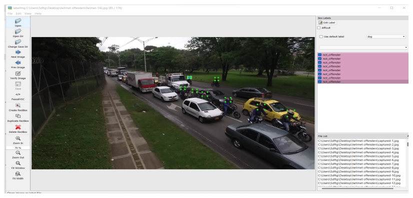
Datasets

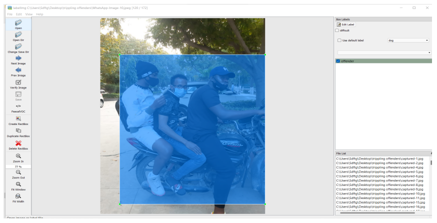
-We got a video datasets from http://velastin.dynu.com/videodatasets/UrbanMotorbike/
The on road video data is a dataset for public use, which contains footage taken using a Phantom 4® drone, with an HD camera[8]. Images were resized to 640 x 364 pixels, with a minimal height size set to 25 pixels. 60% of the annotated data corresponds to occluded motorcycles. This dataset is specifically used for helmet detection. Since there were no available datasets on the internet for tripling detection we captured more images and videos using a mobile phone ; we asked people to sit on two wheeler vehicles in threes and captured their images in different environments and angles.
Below are samples of images we captured using a mobile phone
-The datasets are divided into two parts, 90% are used for training and 10% used for validation

IJSER

Preparing The Data

Before the data is used for training we have to prepare them by extracting the features we want to use in our training dataset and validating datasets and augment the data to increase diversity. In this project we have used labelimg, an open source tool for labeling images, below ,the images below shows sample data and labeling too in use[9]





The Left images shows samples of tripling riders while on the right we have snapshots of the lebelimg tool.

IJSER

Data Augmentation

We applied the following translation techniques to enhance the accuracy and results of our models by creating fresh and diverse instances to train datasets, because a model works better and more correctly when the data is rich and sufficient.[6]

-Flip Horizontal & Vertical

-Rotation Between -15° and +15°

-Grayscale Applied to 25% of images

-Brightness Between -25% and +25%

-Noise Up to 5% of pixels

-Mosaic

-Bounding Box: Flip Horizontal

-Bounding Box: Rotation Between -15° and +15°

-Bounding Box: Brightness Between -25% and +25%

-Bounding Box: Exposure Between -25% and +25%

-Bounding Box: Noise

Training The Model

To start , we downloaded the most up-to-date pre-trained network for the model we'd be using. This was accomplished by simply tapping on the name of our selected model in the Tensor-flow 2 Detection Model Zoo table. The name of our model was clicked and a *.tar.gz file was downloaded. We used a decompression software to open the *.tar.gz file when it was downloaded. Next, open the *.tar folder and extracted its contents inside our workspace folder. We downloaded the SSD ResNet50 V1 FPN 640x640 model[7]

Configuring the Training Pipeline

After downloading and extracting the pre-trained model, We established a directory my_ssd_resnet50_v1_fpn inside the working space/models and copied the file pipeline.config from the workingspace/pretrained-models/ssdresnet 50v1fpn640*640coco17tpu-8 into it.

After copying the config file , we made the following changes;

1.We set num_classses to 2 for each module in our training job

2.We set our batch size to 2 because limited computing resources on the laptop we are using

3. Fine tune checkpoint was set to the last checkpoint of our pretrained model[10]

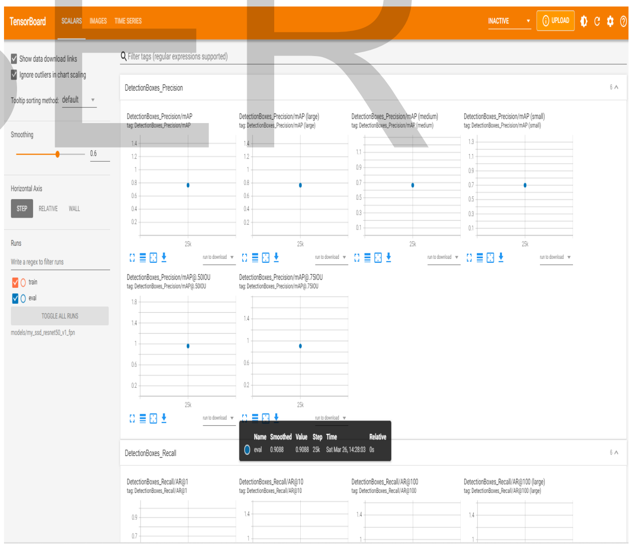
Testing The Models.

Once our training job was completed we tested each model with test images check the next page for excerpts of the helmet detection module and tripling detection in module in action. Note that all traffic violations are classified as offenders.[11]

Evaluating the model.

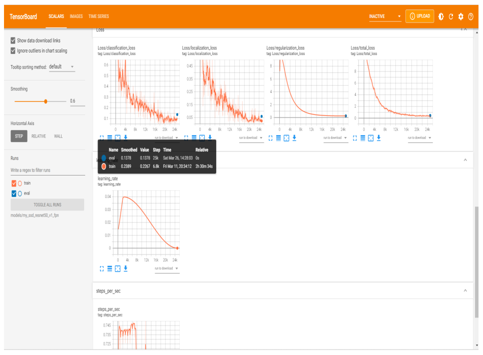
After testing the model we evaluated it using loading logs generated during testing using tensorboard, below is a excerpt of the logs as displayed in tensorboard[7]

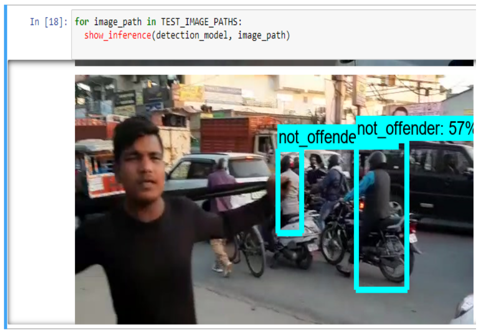
IJSER


TensorBoard
SCALARS
IMAGES
TIME SERIES
INACTIVE
UPLOAD
Show data download links
Ignore outliers in chart scaling
Tooltip sorting method: default
Smoothing
0.6
Horizontal Axis
STEP
RELATIVE
WALL
Runs
Write a regex to filter runs
train
eval
TOGGLE ALL RUNS
models/my_ssd_resnet50_v1_fpn
Filter tags (regular expressions supported)
DetectionBoxes_Precision
DetectionBoxes_Precision/mAP
DetectionBoxes_Precision/mAP (large)
DetectionBoxes_Precision/mAP (medium)
DetectionBoxes_Precision/mAP (small)
DetectionBoxes_Precision/mAP@.50IOU
DetectionBoxes_Precision/mAP@.75IOU
DetectionBoxes_Recall
DetectionBoxes_Recall/AR@1
DetectionBoxes_Recall/AR@10
DetectionBoxes_Recall/AR@100
DetectionBoxes_Recall/AR@100 (large)
25k
run to download

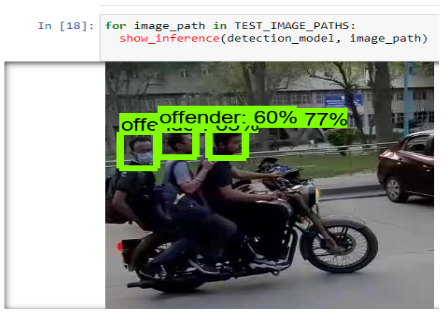
Name	Smoothed	Value	Step	Time	Relative
eval	0.9088	0.9088	25k	Sat Mar 26, 14:28:03	0s

The figure above shows the models metrics as seen from tensorboard

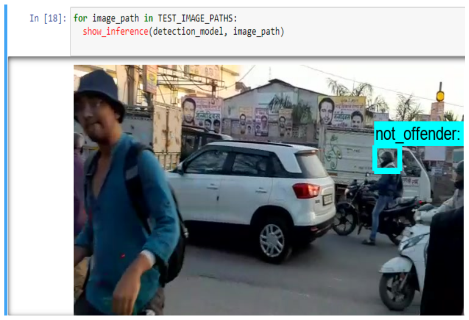




```
In [18]: for image_path in TEST_IMAGE_PATHS:
             show_inference(detection_model, image_path)
```



```
In [18]: for image_path in TEST_IMAGE_PATHS:
             show_inference(detection_model, image_path)
```



```
In [18]: for image_path in TEST_IMAGE_PATHS:
             show_inference(detection_model, image_path)
```

The top left image on the left show model training as viewed from tensorboard, the, the top right,bottom left and bottom right shows detection of non offenders, detection of helmet offenders and detection on non offender consecutively

IJSER

Metrics of Tripling detection module

Model	Detection Boxes Precision	Recall Precision	Average Detection Boxes Precision	Average Recall Precision
Tripling model	**0.76**	**0.54**		
Helmet Model	**0.90**	**0.70**		
Average performance			**0.83**	**0.62**

Conclusion

The two models performed pretty well giving an average precision of 83%, with more datasets the models can be improved further and deployed in a traffic environment where two wheeler traffic offenders can be detected and charged. More modules can also be added for accurate pinpointing of the riders like number plate detection and extraction. We faced various challenges during the development of the models,for instance there we insufficient data available for training in the web so we had to capture new data, We also had inadequate computing resources which made it difficult to trained the model as we used a small batch size of 2 and still the models took a very long time time to train. There are significant improvements that can be made in the future to enhance the reliability and efficiency of the models; More data can be added for better accuracy and performance, One can use a more powerful computing unit with dedicated GPUs and an alert module can be added for instance notifications to the law enforcement agents.

IJSER

References

[1] .Chiverton, J. (2012). Helmet presence classification with motorcycle detection and tracking. IET Intelligent Transport Systems, 6(3), 259. https://doi.org/10.1049/iet-its.2011.0138

[2] Socarras, Yainuvis & Vázquez, David & López, Antonio & Geronimo, David & Gevers, T.. (2012). Improving HOG with Image Segmentation: Application to Human Detection. 178-189. 10.1007/978-3-642-33140-4_16.

[3] Liela Khobanizad, Mahmood Khobanizad, Behrouz Vaseghi, Hamid Chegini, Implement of Face Recognition in Android Platform by Using Opencv and LBT Algorithm, International Journal of Wireless Communications and Mobile Computing. Volume 4, Issue 2, March 2016 , pp. 25-31. doi: 10.11648/j.wcmc.20160402.13

[4] Sorin Draghici ()
International Journal of Neural Systems 1997 08:01, 113-126

[5] https://www.tensorflow.org/tutorials/images/transfer_learning#:~:text=A%20pre%2Dtrained%20model%20is,model%20to%20a%20given%20task

[6] Data Augmentation - Advantages, Challenges, and Instances - SnapStack

[7] https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/training.html

[8] http://velastin.dynu.com/videodatasets/UrbanMotorbike/

[9] GitHub - tzutalin/labelImg: ✏️ LabelImg is a graphical image annotation tool and label object bounding boxes in images

[10] Training checkpoints | TensorFlow Core

[11] TensorBoard Scalars: Logging training metrics in Keras | TensorFlow