Deep Learning based License Plate Recognition

Monica Dommaraju
Jumana Zoeb Nadir
Sruthi Chilukuri

Introduction

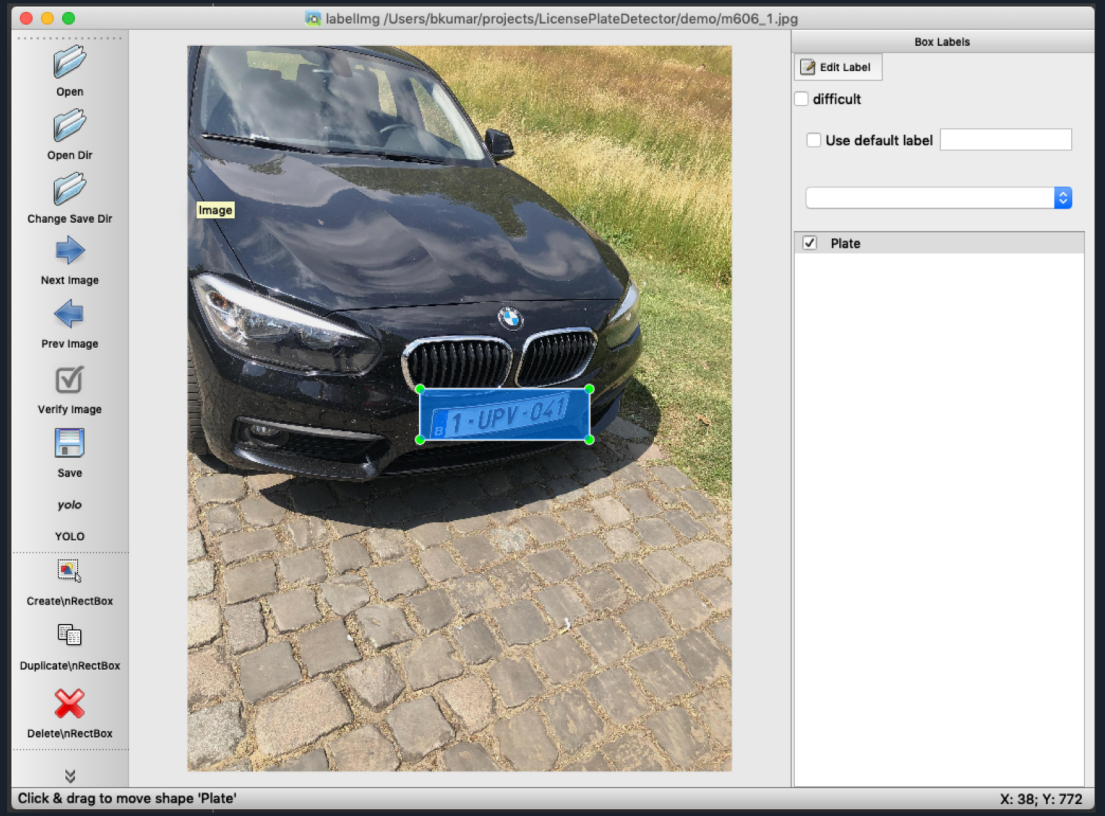
1. License Plate Detection using
    a. YOLOV3 (Darknet)
    b. CNN
2. Trained models on Belgium Cars dataset.
3. Three main modules:
    a. License Plate Detection
    b. Characters Segmentation
    c. Character Identification

Dataset

- Approximately 700 Car images with multiple images for each car from different angles.
- License plates for each of the cars with Image Augmentation.
- ImageNet compatible annotations
- Converted to YOLO supported labels
- Additional test dataset from manually captured images.
- Approx 1000 images for each character from 0 - 9 and A to Z
- 90% for train and 10% for test
- Link

Annotations

- Used **LabelImg** tool for generating annotations.



Training - Darknet

- Installed Darknet
- Modified the yolo config files (classes, filter, steps, etc..).
- Trained on GPU enabled EC2 box.
- First model to detect License plate took 3 hrs (1 bounded box per car image)
- Second model to Segment Characters and identify them took 8 hrs (Max 35 classes)

Training-Darknet

Results from final steps - License plate detection

```
Region 106 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.000000, .5R: -nan, .75R: -nan,  count: 0
Region 82 Avg IOU: 0.818924, Class: 0.999857, Obj: 0.997116, No Obj: 0.000264, .5R: 1.000000, .75R: 1.000000,
Region 94 Avg IOU: 0.843183, Class: 0.999914, Obj: 0.996877, No Obj: 0.000226, .5R: 1.000000, .75R: 1.000000,
Region 106 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.000000, .5R: -nan, .75R: -nan,  count: 0
Region 82 Avg IOU: 0.901864, Class: 0.999781, Obj: 0.997062, No Obj: 0.000782, .5R: 1.000000, .75R: 1.000000,
Region 94 Avg IOU: 0.788954, Class: 0.999435, Obj: 0.524586, No Obj: 0.000211, .5R: 1.000000, .75R: 0.500000,
Region 106 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.000000, .5R: -nan, .75R: -nan,  count: 0
Region 82 Avg IOU: 0.725491, Class: 0.998672, Obj: 0.755271, No Obj: 0.000456, .5R: 0.500000, .75R: 0.500000,
Region 94 Avg IOU: 0.838893, Class: 0.999872, Obj: 0.500832, No Obj: 0.000119, .5R: 1.000000, .75R: 1.000000,
Region 106 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.000000, .5R: -nan, .75R: -nan,  count: 0
Region 82 Avg IOU: 0.883862, Class: 0.999803, Obj: 0.996992, No Obj: 0.000561, .5R: 1.000000, .75R: 1.000000,
Region 94 Avg IOU: 0.806010, Class: 0.999845, Obj: 0.997554, No Obj: 0.000134, .5R: 1.000000, .75R: 1.000000,
Region 106 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.000000, .5R: -nan, .75R: -nan,  count: 0
3000: 0.071277, 0.062509 avg, 0.000010 rate, 3.262568 seconds, 192000 images
Saving weights to ../Yolo_LicensePlateDetection/models/yolov3-license-plates.backup
Saving weights to ../Yolo_LicensePlateDetection/models/yolov3-license-plates_final.weights
[1]+  Done                    ./darknet detector train ../Yolo_LicensePlateDetection/config/license_plate.data
```

Training-Darknet

Results from final steps - - Character Segmentation and Identification

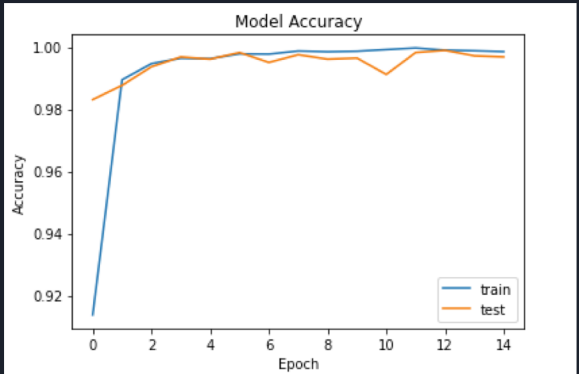
```
Region 94 Avg IOU: 0.899716, Class: 0.998312, Obj: 0.997887, No Obj: 0.003960, .5R: 1.000000, .75R: 1.000000,
Region 106 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.000000, .5R: -nan, .75R: -nan,  count: 0
Region 82 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.000001, .5R: -nan, .75R: -nan,  count: 0
Region 94 Avg IOU: 0.912103, Class: 0.999583, Obj: 0.999700, No Obj: 0.003408, .5R: 1.000000, .75R: 1.000000,
Region 106 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.000000, .5R: -nan, .75R: -nan,  count: 0
Region 82 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.000001, .5R: -nan, .75R: -nan,  count: 0
Region 94 Avg IOU: 0.924464, Class: 0.998221, Obj: 0.999106, No Obj: 0.004380, .5R: 1.000000, .75R: 1.000000,
Region 106 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.000000, .5R: -nan, .75R: -nan,  count: 0
10000: 0.112187, 0.119860 avg, 0.000010 rate, 3.656955 seconds, 640000 images
Saving weights to ../Yolo_CharacterSegmentation/models/yolov3_character.backup
Saving weights to ../Yolo_CharacterSegmentation/models/yolov3_character_10000.weights
Saving weights to ../Yolo_CharacterSegmentation/models/yolov3_character_final.weights
[1]+  Done                    ./darknet detector train ../Yolo_CharacterSegmentation/config/characters.data ./
```

Transfer Learning

- We have used the YoloV3 weights trained on ImageNet (darknet53.conv.74).
- We used them as our network initial weights and trained the network.

Training - CNN

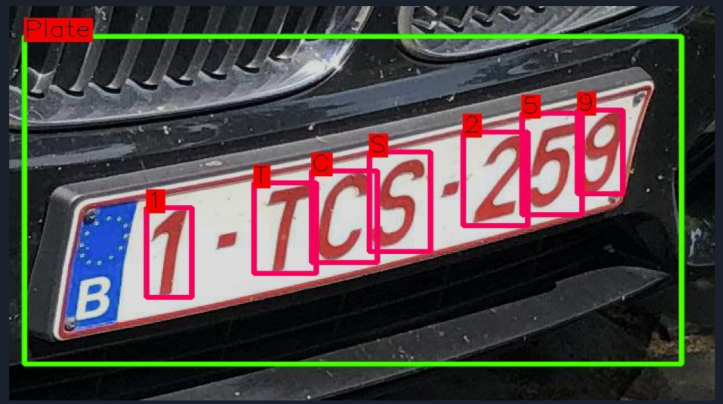
- 99.7% Accuracy, 15 Epochs




```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 100, 75, 32)       320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 50, 37, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 48, 35, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 24, 17, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 22, 15, 128)       73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 11, 7, 128)        0
_________________________________________________________________
dense (Dense)                (None, 11, 7, 128)        16512
_________________________________________________________________
flatten (Flatten)            (None, 9856)              0
_________________________________________________________________
dense_1 (Dense)              (None, 35)                344995
=================================================================
Total params: 454,179
Trainable params: 454,179
Non-trainable params: 0
```

```
Epoch 13/15
8520/8520 [==============================] - 24s 3ms/step - loss: 0.0060 - accuracy: 0.9990 - val_
loss: 0.0097 - val_accuracy: 0.9989
Epoch 14/15
8520/8520 [==============================] - 25s 3ms/step - loss: 0.0061 - accuracy: 0.9988 - val_
loss: 0.0227 - val_accuracy: 0.9972
Epoch 15/15
8520/8520 [==============================] - 24s 3ms/step - loss: 0.0098 - accuracy: 0.9986 - val_
loss: 0.0227 - val_accuracy: 0.9968
222/222 [==============================] - 1s 4ms/step - loss: 0.0287 - accuracy: 0.9970
0.9970422387123108
```

Prediction



Configurations

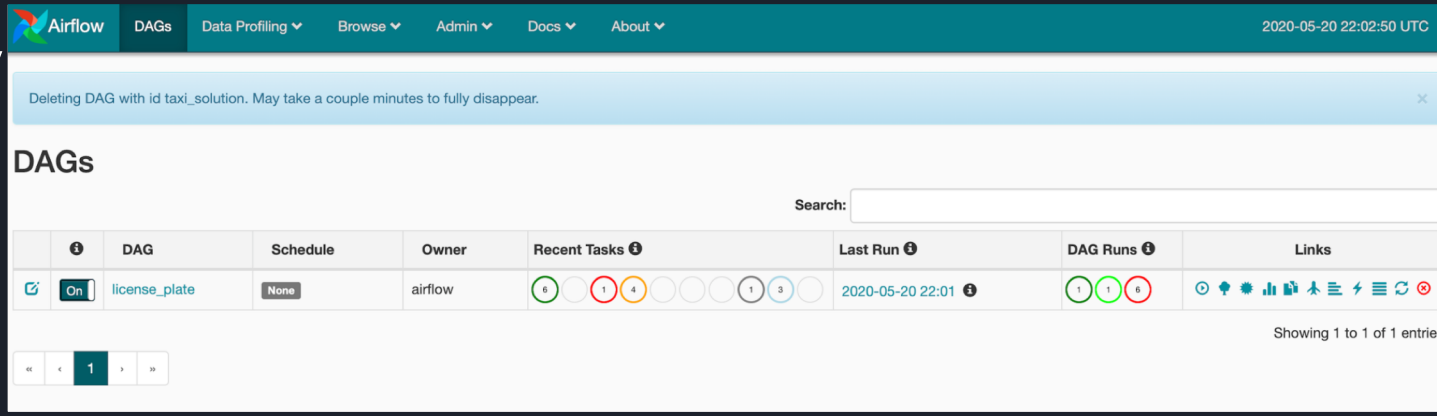
1. YOLOV3 + YOLOV3
    a. License plate detection with one bounded box will be predicted using YOLO.
    b. Characters Segmentation and identification also uses YOLO

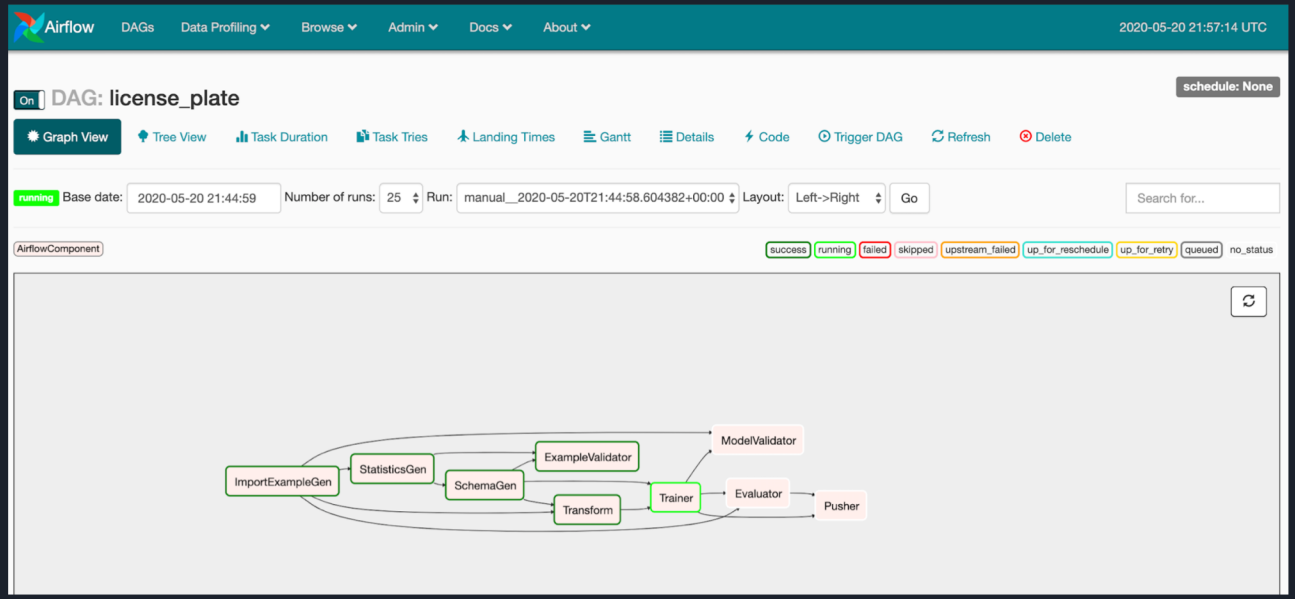
1. YOLOV3 + CNN
    a. License plate detection with one bounded box will be predicted using YOLO.
    b. Segmentation of Characters in the bounded box is predicted using OpenCV contour image processing
    c. Characters inside the segmented boxes are identified by trained CNN.

TFX- Airflow

- Used TFX and Airflow to create a DAG pipeline.
- CNN model training for character recognition was used in TFX
- Darknet training cannot be performed on TFX, as the training requires compile C libraries and shell commands.

TFX - Airflow

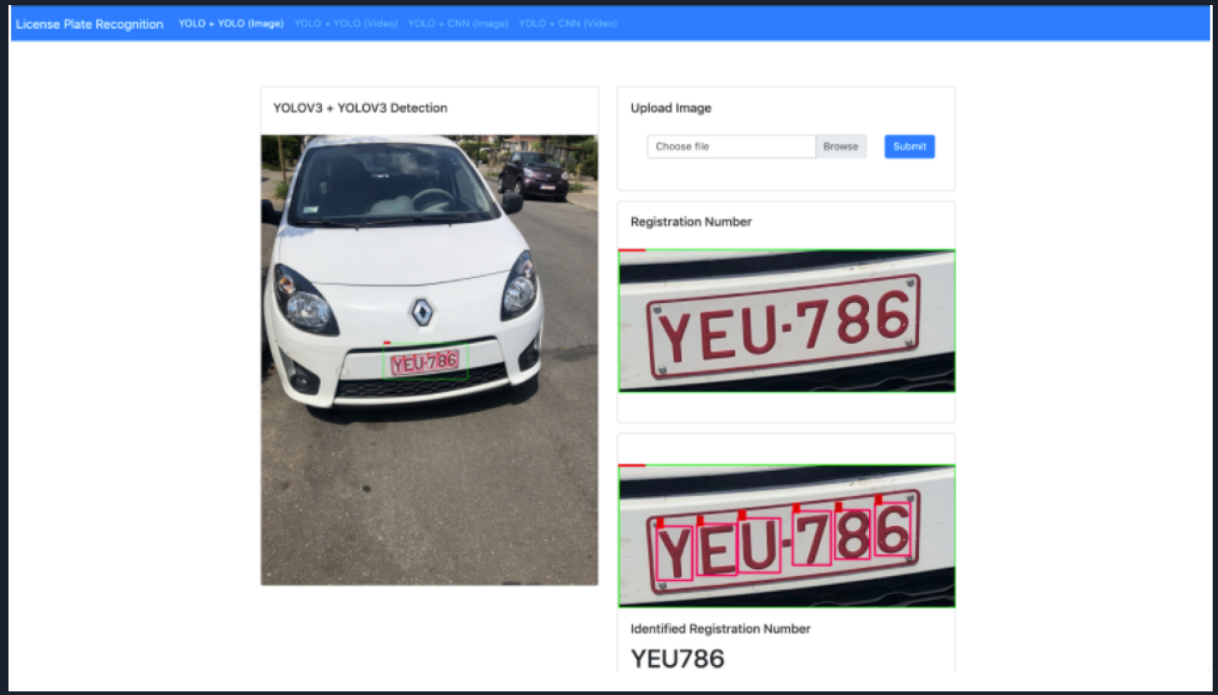

Airflow
DAGs
Data Profiling
Browse
Admin
Docs
About
2020-05-20 22:02:50 UTC
Deleting DAG with id taxi_solution. May take a couple minutes to fully disappear.
DAGs
Search:
DAG
Schedule
Owner
Recent Tasks
Last Run
DAG Runs
Links
On
license_plate
None
airflow
2020-05-20 22:01
Showing 1 to 1 of 1 entrie


Airflow
DAGs
Data Profiling
Browse
Admin
Docs
About
2020-05-20 21:57:14 UTC
On
DAG: license_plate
schedule: None
Graph View
Tree View
Task Duration
Task Tries
Landing Times
Gantt
Details
Code
Trigger DAG
Refresh
Delete
running
Base date:
2020-05-20 21:44:59
Number of runs:
25
Run:
manual__2020-05-20T21:44:58.604382+00:00
Layout:
Left->Right
Go
Search for...
AirflowComponent
success
running
failed
skipped
upstream_failed
up_for_reschedule
up_for_retry
queued
no_status
ImportExampleGen
StatisticsGen
SchemaGen
ExampleValidator
Transform
ModelValidator
Trainer
Evaluator
Pusher

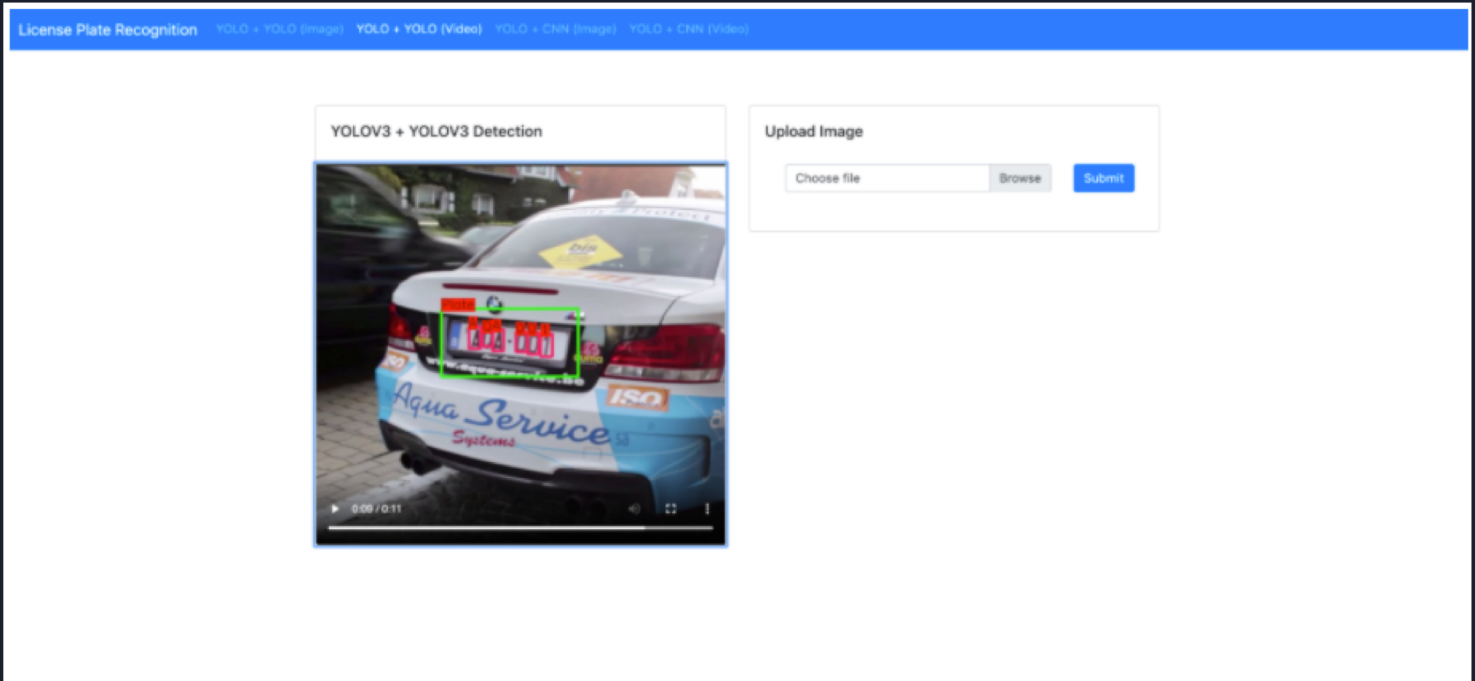
Web Application

- Used Python Flask to develop APIs and FrontEnd
- Loaded Models and Weights that were generated in the training process.
- Can predict License Registration Number on both Image and Video
- Deployed the app to Amazon EC2

Web Application



Web Application



DEMO