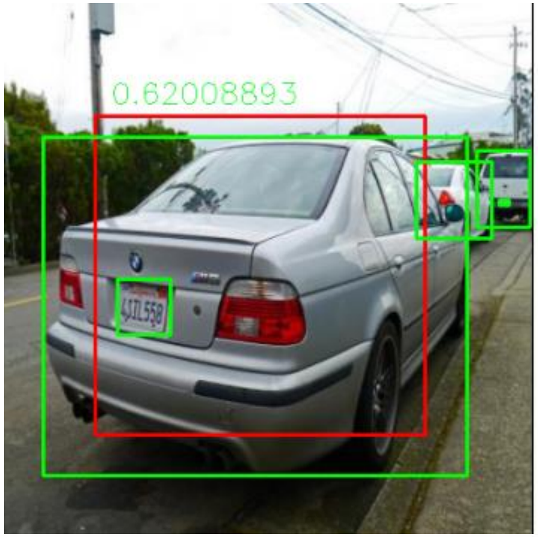
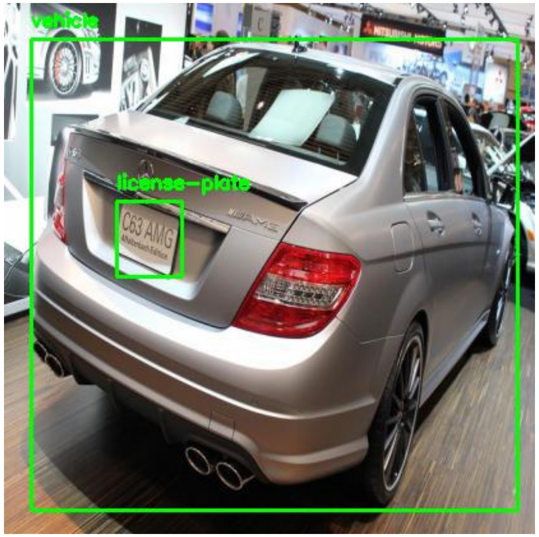
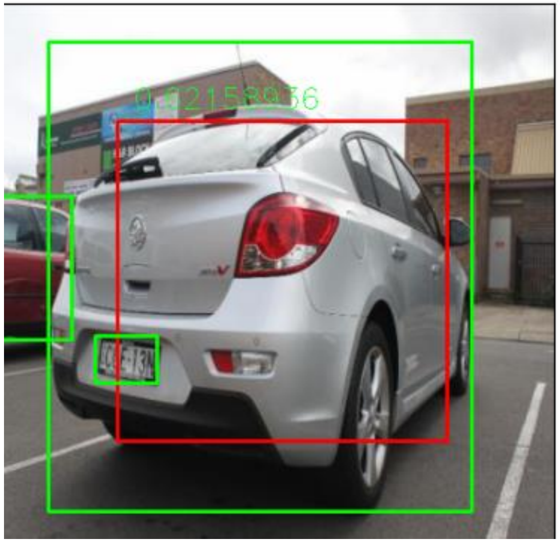
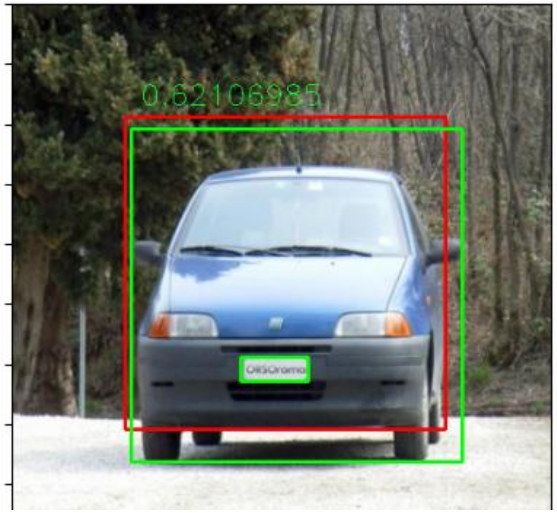
VEHICLE & LICENSE PLATE DETECTION APP









Course Project :
E9 246 Advanced Image Processing 2024

Submitted By:
Benjamin Debbarma(22554)
DS Rana(23699)

Objective

Develop an Object Detection Android Application to detect vehicle & license plates in images using Tranfer Learning & Android Studio

Dataset

Selection of Dataset:

1. Motivated by ability to demonstrate.
2. Vehicle & License Plate – Possible to test in real time & Wide range of Applications.
3. Dataset from Open Source Domain. (https://public.roboflow.com/object-detection/license-plates-us-eu).
4. Splitting of Dataset into Train, Validation & Test subsets:-
   1. Train – 245
   2. Valid – 70
   3. Test – 35
5. Labeling in Pascal VOC format

Model Architecture

Pre Trained Backbone for feature extraction – MobileNetV2

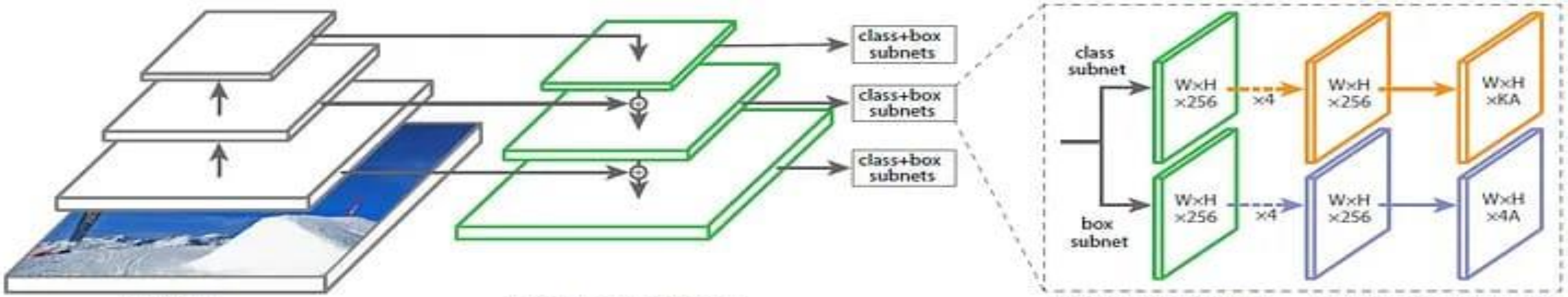
- Outputs feature maps at different resolutions.

Feature Pyramid Network

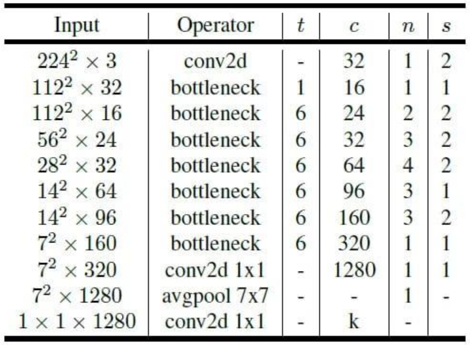
- Integrates multi-level features from the backbone

Object detection predictions - RetinaNet Head

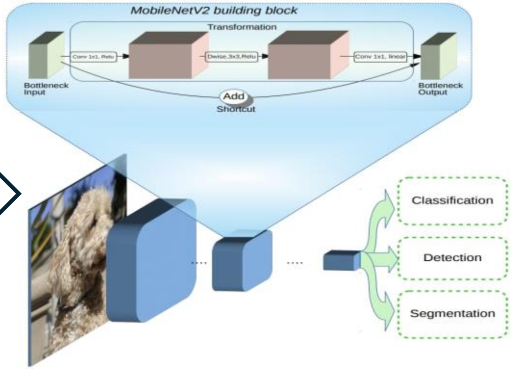
- Generates object detection predictions - Box & Cl

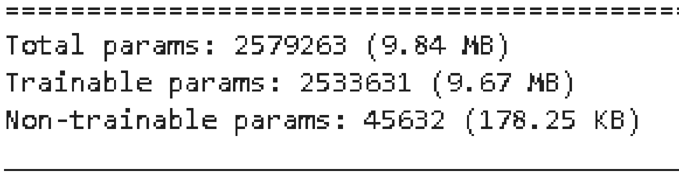


(a) MobileNetV2 (b) feature pyramid net (c) class subnet (top) (d) box subnet (bottom)



Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	





```
==========================================
Total params: 2579263 (9.84 MB)
Trainable params: 2533631 (9.67 MB)
Non-trainable params: 45632 (178.25 KB)
```

Training & Evaluation

Training

1. **Hyperparameters:**
   1. **Batch Size: 32**
   2. **Learning Rate: 0.3**
   3. **Epochs: 200.**
2. **Dataset :-Train – 245 & Valid – 70**
3. **Total Loss comprising of:-**
   1. **Box Loss**
   2. **Cls Loss**
4. **Trained using Mediapipe Library**

Evaluation

1. **Hyperparameters:**
   1. **Batch Size: 4**
   2. **Images: 35**
   3. **Evaluation Metric: Coco.**
2. **Moderate performance on test data, with room for improvement in detecting small objects**

Training Results:-

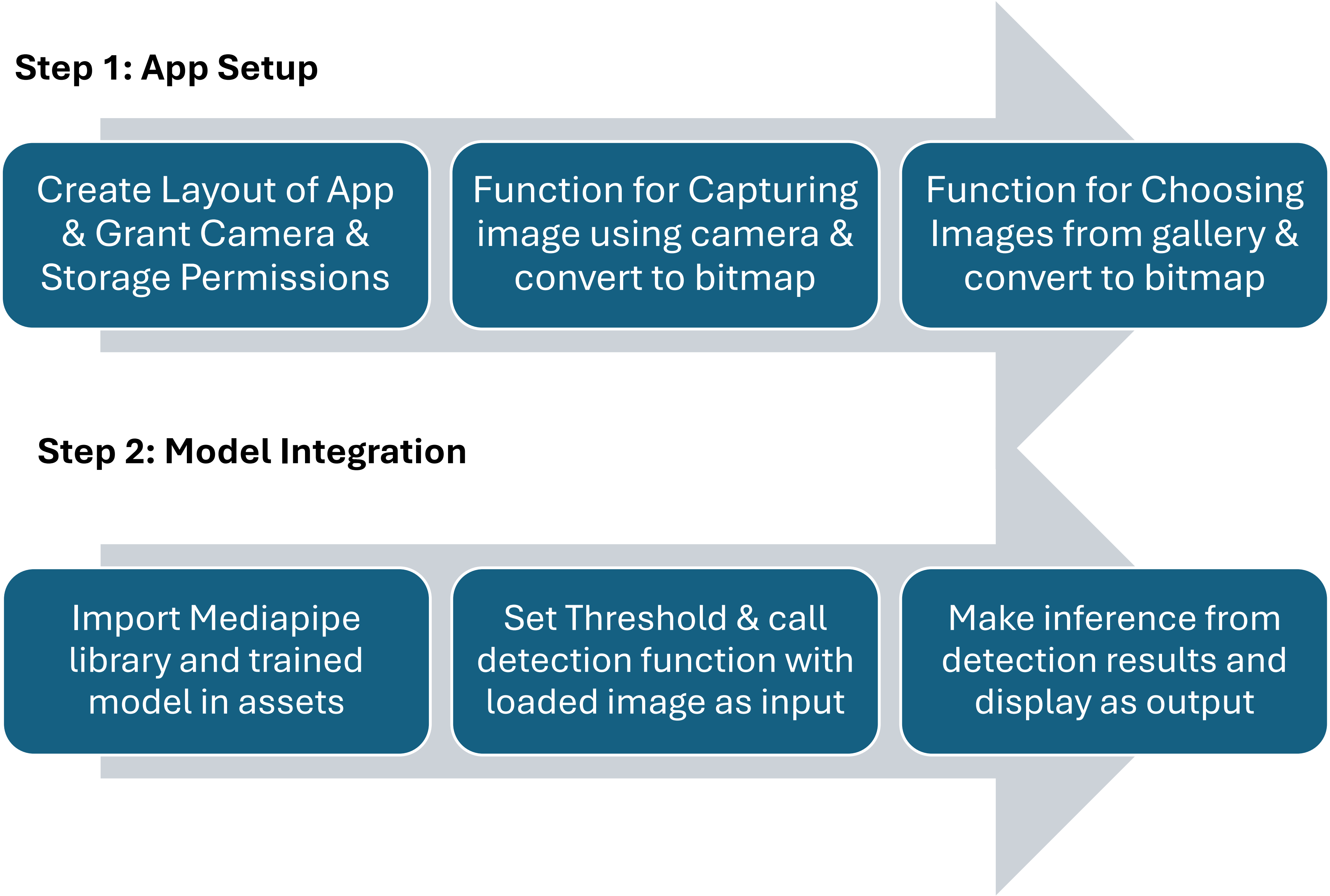
```
Epoch 200/200
Training Data: total_loss: 0.0698
cls_loss: 0.0101 box_loss: 1.1625e-04
Validation Data: total_loss: 0.8837
cls_loss: 0.6999 box_loss: 0.0026
```

Evaluation Results:-

```
total_loss: 0.9762  cls_loss: 0.8203 -
box_loss: 0.0020
AP @[ IoU=0.50:0.95 | area=   all] = 0.444
AP @[ IoU=0.50      | area=   all] = 0.613
AP @[ IoU=0.75      | area=   all] = 0.501
AP @[ IoU=0.50:0.95 | area= small] = 0.007
AP @[ IoU=0.50:0.95 | area=medium] = 0.374
AP @[ IoU=0.50:0.95 | area= large] = 0.689
AR @[ IoU=0.50:0.95 | area=   all] = 0.463
AR @[ IoU=0.50:0.95 | area=   all] = 0.581
AR @[ IoU=0.50:0.95 | area=   all] = 0.584
AR @[ IoU=0.50:0.95 | area= small] = 0.025
AR @[ IoU=0.50:0.95 | area=medium] = 0.555
AR @[ IoU=0.50:0.95 | area= large] = 0.773
```

Export Model in .tflite format
for Android Integration

Android App Development



Step 1: App Setup

Create Layout of App & Grant Camera & Storage Permissions

Function for Capturing image using camera & convert to bitmap

Function for Choosing Images from gallery & convert to bitmap

Step 2: Model Integration

Import Mediapipe library and trained model in assets

Set Threshold & call detection function with loaded image as input

Make inference from detection results and display as output

Real Time Test Results

- **There are two modes of op**:-
  - **Camera** Long Press the IISc logo, After doing so, capture an image using camera, and the app will display the results. This operation method with real time demonstration under:-
    - https://drive.google.com/file/d/1n-DqLOzsUCxMzIg2lUaGed5Z8rXWTdqL/view
    - https://drive.google.com/file/d/1n0LktRwfWLvx_HAh_KiO2lQNXuD0QUxX/view
  - **Gallery** tap the IISc logo to import images from phone's gallery. A real time demonstration of this method is provided below:-
    - https://drive.google.com/file/d/1mzQGT5RyCcS_tk9XUhcjtQkBNhe-6qF9/view

Observations & Challenges

- Good Results in Single Instance
- Performance degraded with Multiple Instances
- Few vehicle like objects detected as false positives
- Few Small size objects not detected
- Struggled in low light environments
- Performed well against rotation in instances

Points for Improvement

- Larger dataset can improve performance
  - Include more images with multiple instances
  - Include more images with smaller instances
  - Augment the existing data with fwg:-
    - Scaling
    - Rotation
    - Blurring
  - Include more images with night/ low light settings
- Add Image enhancement for better detection
- Add options of Live footage & Video Stream in App

References

1. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H, arXiv:1704.04861, 2017.

2. MobileNetV2: Inverted Residuals and Linear Bottlenecks, Sandler M, Howard A, Zhu M, Zhmoginov A, Chen LC. arXiv preprint. arXiv:1801.04381, 2018.

3. http://research.google/blog/mobilenetv2-the-next-generation-of-on-device-computer-vision-networks/

4. Focal Loss for Dense Object Detection Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, Piotr Dollar Facebook AI Research (FAIR) arXiv:1708.02002v2, 2018

5. https://developers.google.com/mediapipe/solutions/model_maker

6. https://developers.google.com/mediapipe/solutions/vision/object_detector

7. Vehicle Object Detection Based on Improved RetinaNet Luyang Zhang,, Haitao Wang, Xinyao Wang, Shuai Chen, Huaibin Wang, Kai Zheng and Hailong wang, 2020.

8. https://hamzaasif-mobileml.medium.com/android-capturing-images-from-camera-or-gallery-as-bitmaps-d3eb1d68aeb2