Transfer Learning for Object Detection with Deep Convolutional Networks

Presented By,

Sugunadevi Palaniappan

Contents


Motivation


Theoretical Background


Experimental Results


Conclusion

Motivation

Use case:

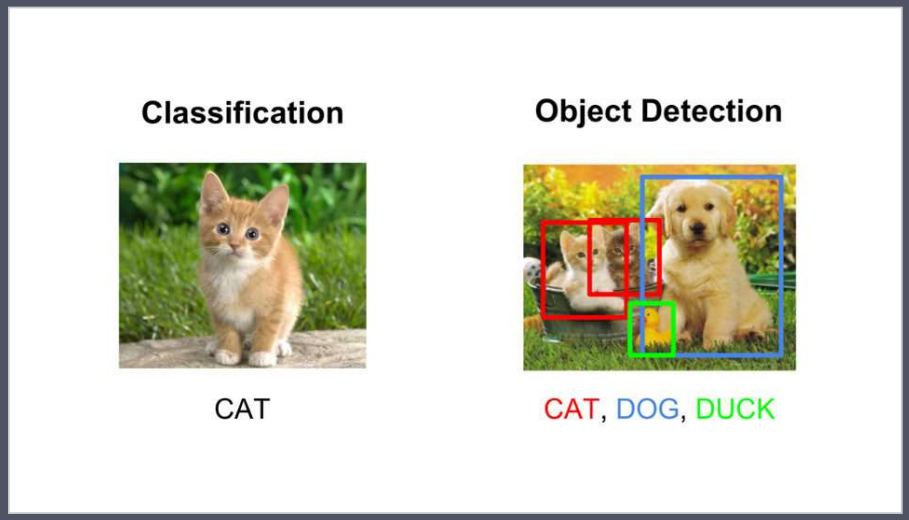
Customers send back ordered items.

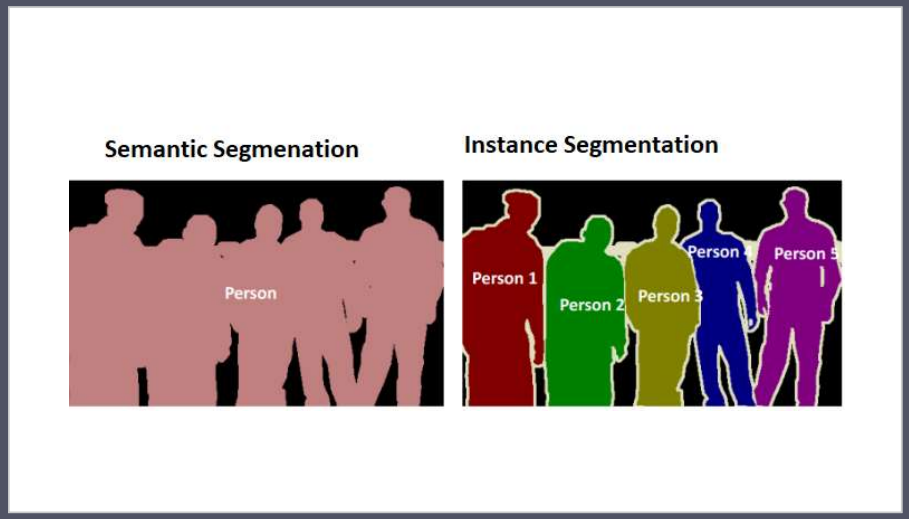
So, the retailer wants to automatically

- unpack the returned box
- sort the returned items.

https://www.google.com/imgres?imgurl=https%3A%2F%2Fi.ytimg.com%2Fvi%
https://pathguide.com/news/using-a-wms-to-combat-worker-fatigue/

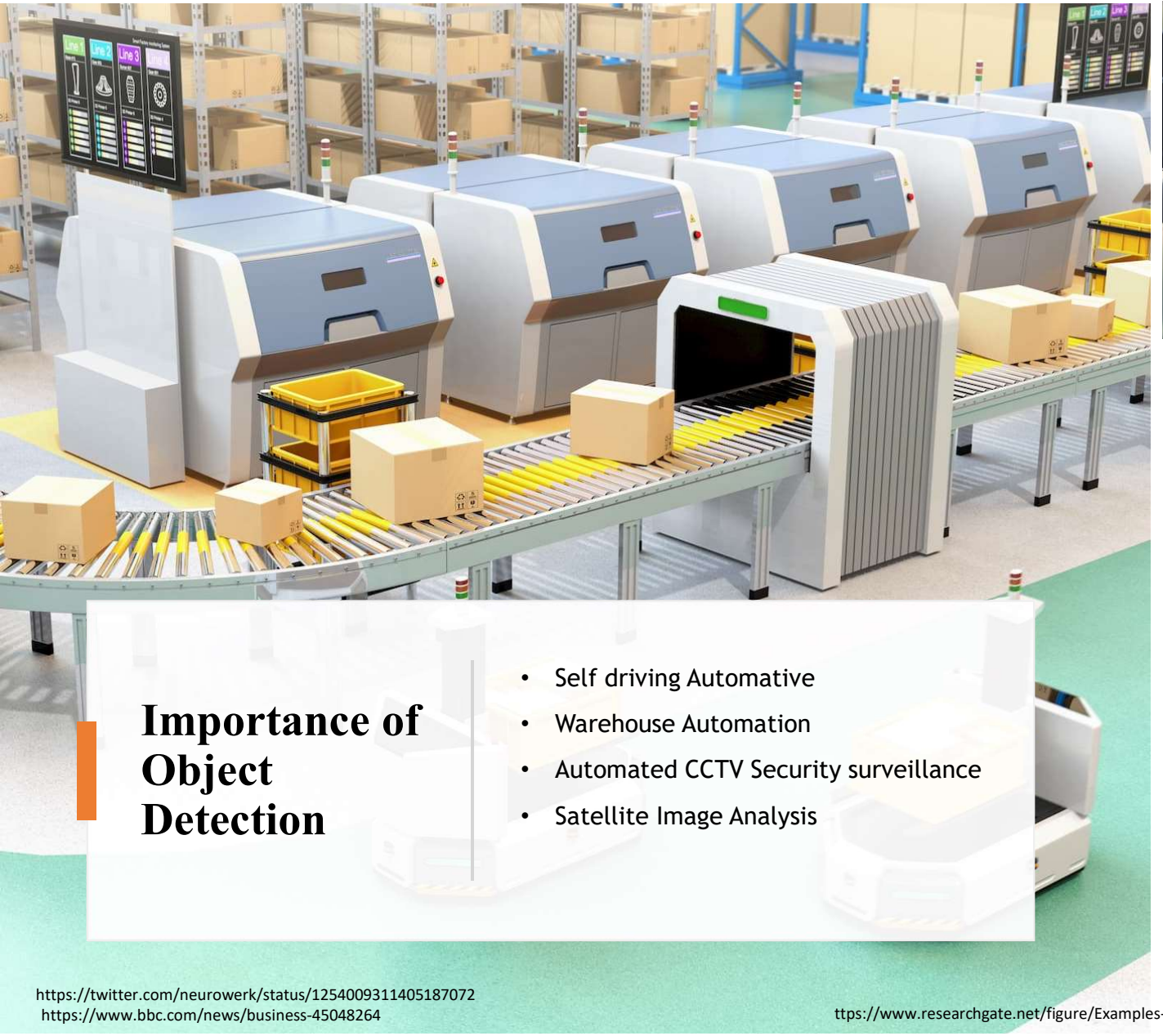
Theoretical Background: Computer Vision Tasks





https://www.analyticsvidhya.com/blog/2019/04/introduction-image-segmentation-techniques-python/

- https://www.datacamp.com/community/tutorials/object-detection-guide



Importance of Object Detection

- Self driving Automative
- Warehouse Automation
- Automated CCTV Security surveillance
- Satellite Image Analysis

https://twitter.com/neurowerk/status/1254009311405187072
https://www.bbc.com/news/business-45048264

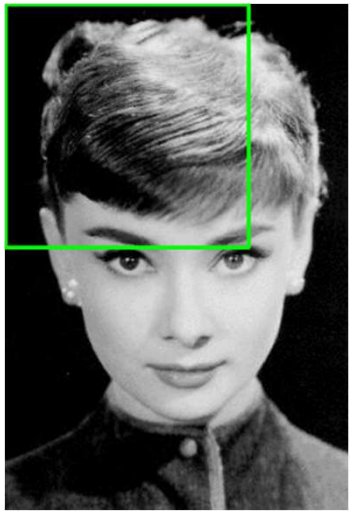
ttps://www.researchgate.net/figure/Examples-of-tracking-output-on-typical-surveillance-data-from-a-Torino-metro-station_fig2_221430071

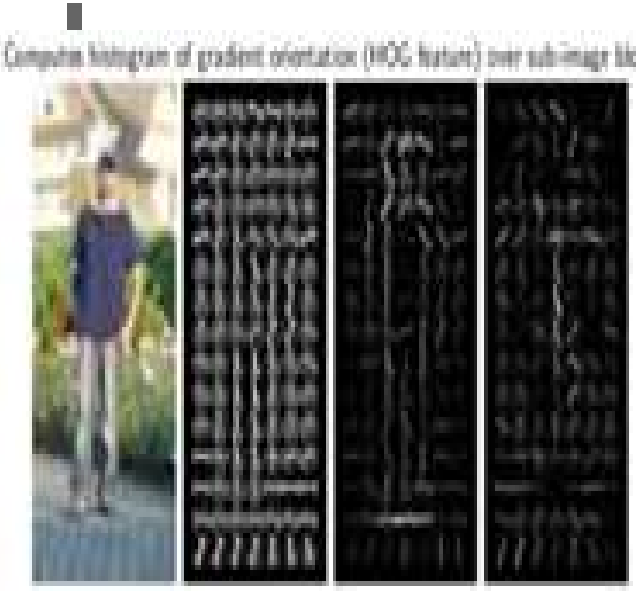
https://www.box-logic.co.uk/services/logistics-technology/warehouse-automation-and-roboti

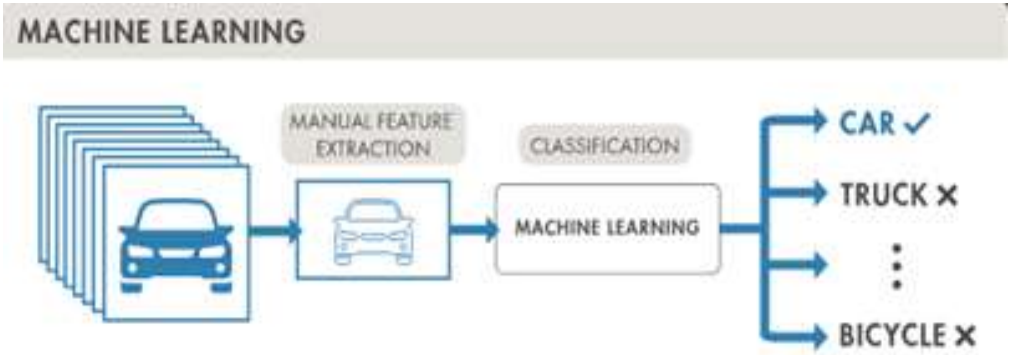
Traditional Object detection Models

Three stages:

- Informative Region selection
- Feature Extraction
- Classification

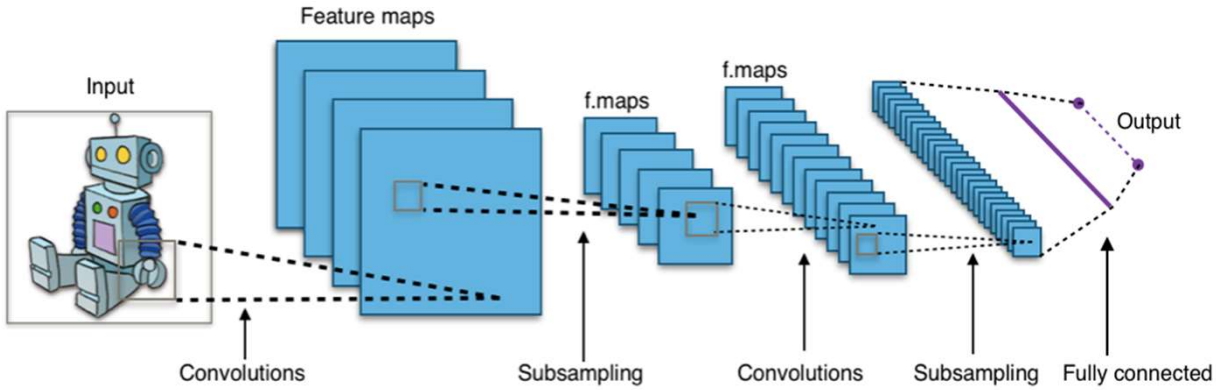




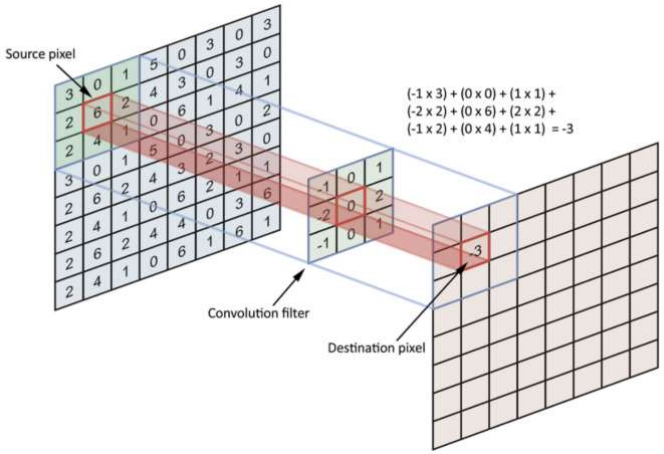


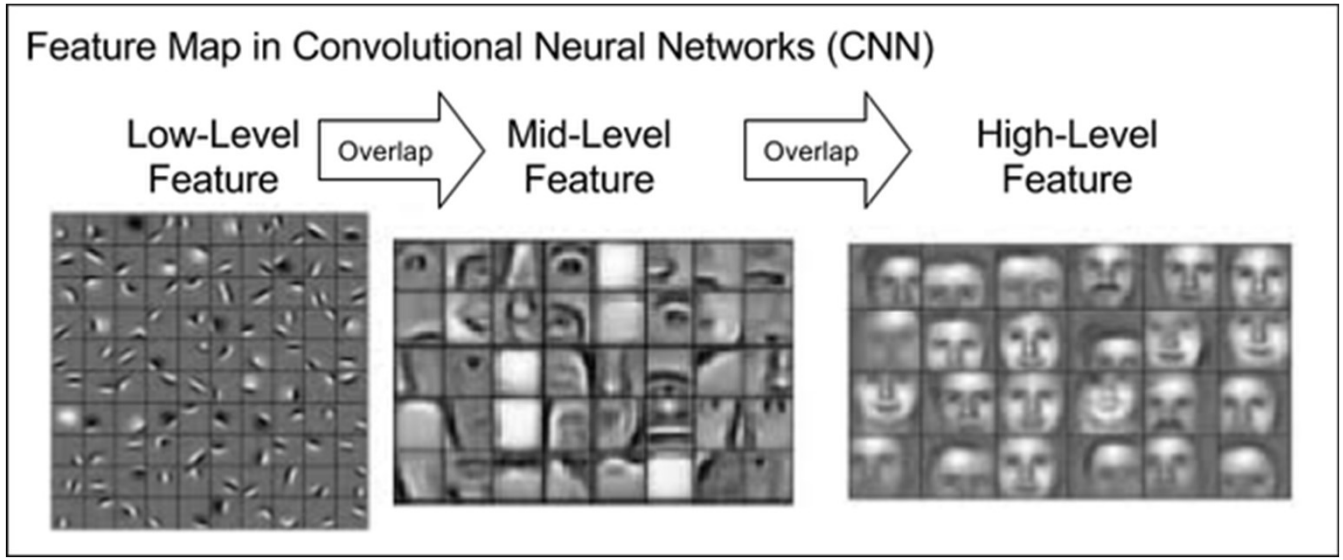
https://livebook.manning.com/book/grokking-deep-learning-for-computer-vision/ch
https://www.youtube.com/watch?v=ATw1Dy4p1GU&ab_channel=RanjanSharma
https://www.edge-ai-vision.com/2020/04/object-recognition-3-things-you-need-to-know/

CNN



Convolution

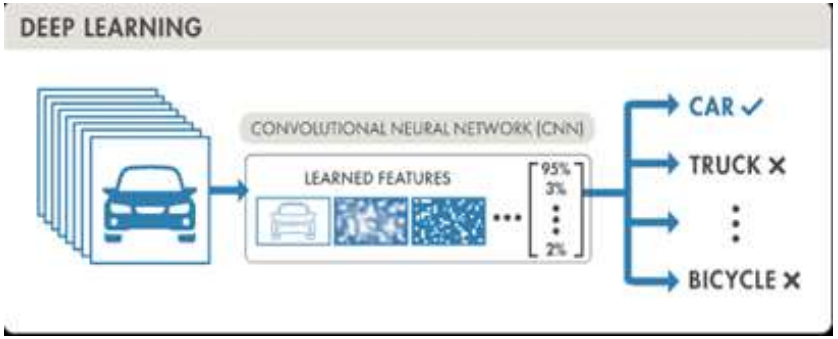


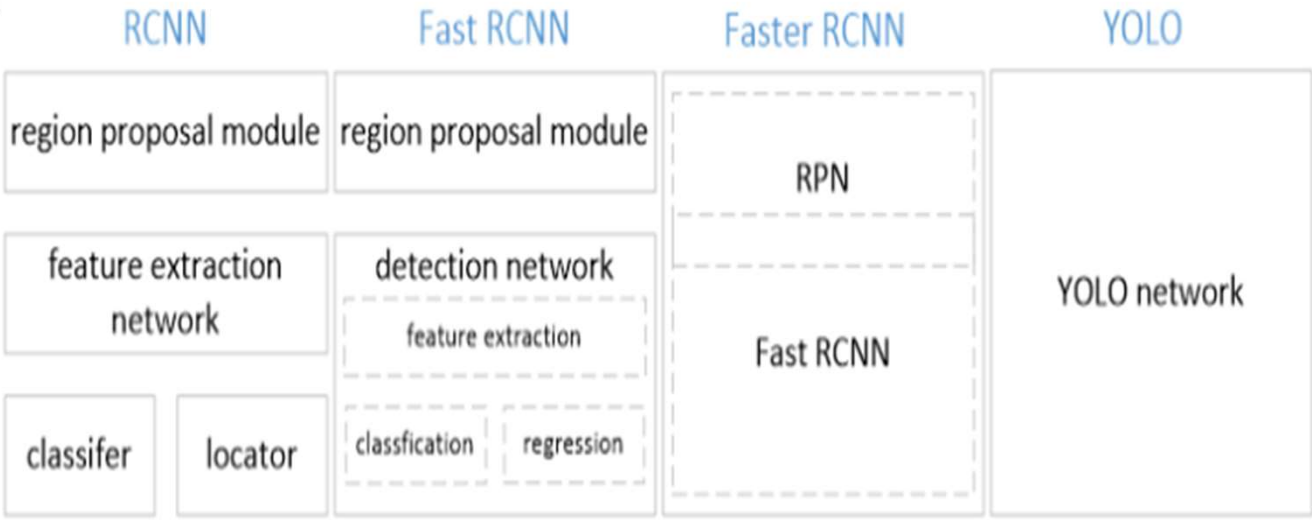


https://datascience.stackexchange.com/questions/77830/how-do-stacked-cnn-layers-work
https://en.wikipedia.org/wiki/Convolutional_neural_network#/media/File:Typical_cnn.png
https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac

Pre-trained Object Detection Algorithms with CNNs

- Two stage object detectors
  - R-CNN
  - Fast R-CNN
  - Faster R-CNN
- One stage object detectors
  - YOLO (You Only Look Once)
  - SSD- (Single-shot detector)

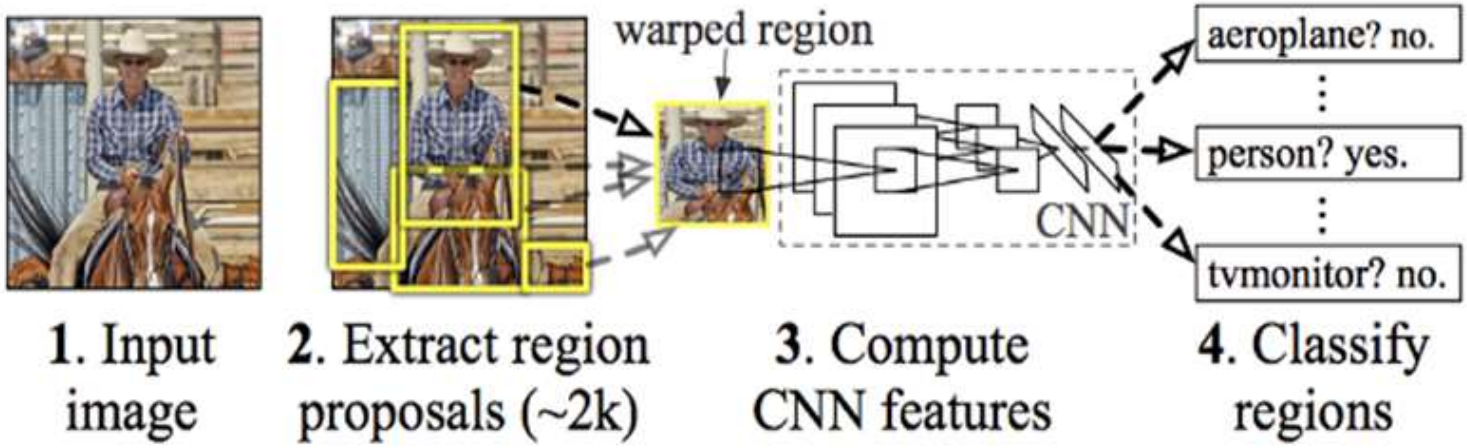




https://github.com/yehengchen/Object-Detection-and-Tracking/blob/master/img/yolo_vs_rcnn.png

https://www.edge-ai-vision.com/2020/04/object-recognition-3-things-you-need-to-know/

Two stage object detector: R-CNN



- https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e

Two approaches to object recognition with deep learning

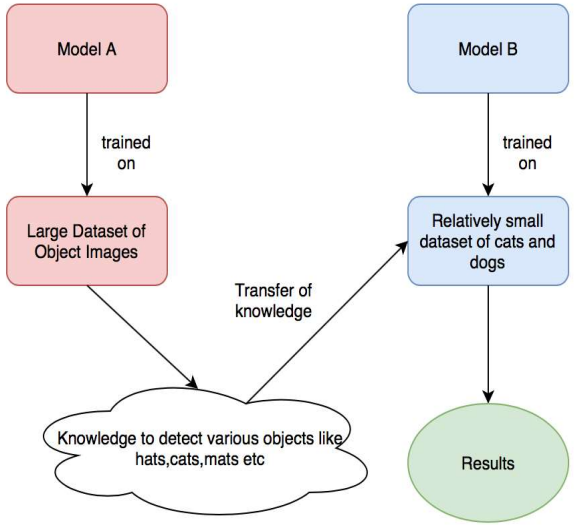
1

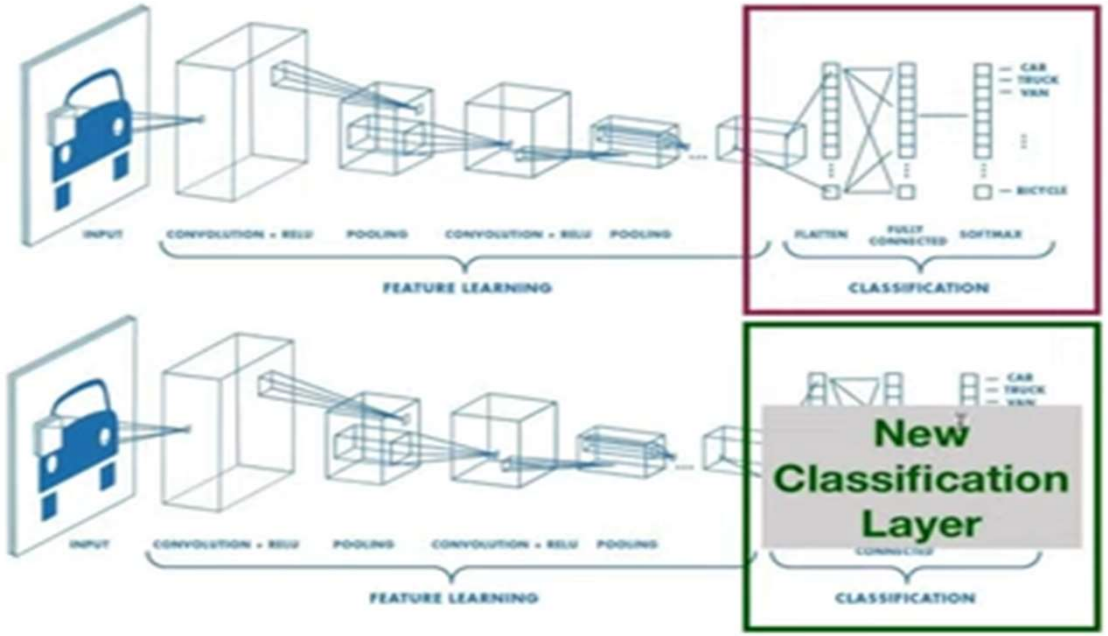
Training a model from scratch

2

Using a pre-trained deep learning model- Transfer Learning

Transfer learning





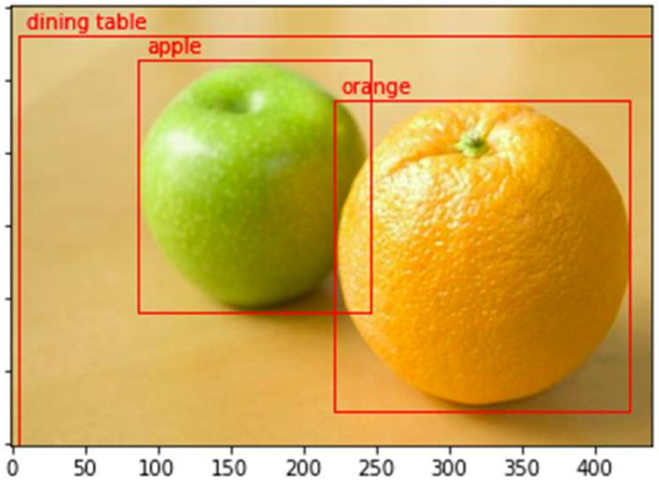
- https://www.youtube.com/watch?v=K0lWSB2QoIQ&ab_channel=PythonEngineer
  https://towardsdatascience.com/transfer-learning-using-differential-learning-rates-638455797f00

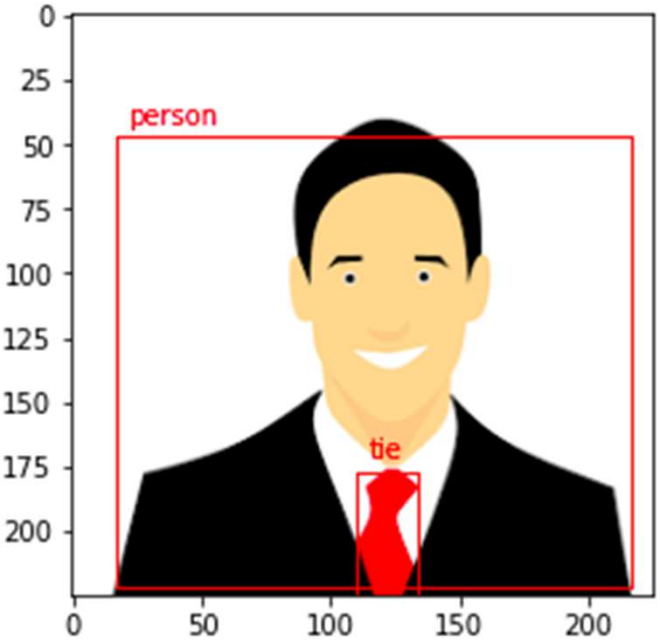
Experimental Results

Existing Work Implementation

- Existing Dataset: MS COCO17 with 80 Object Classes
- Training: ~118k images, val: ~5k images, Test:~41k images
- Deep Learning Framework: PyTorch
- Model: Faster R-CNN model with a ResNet-50(detecto)
- Real time: Detect-live method usage

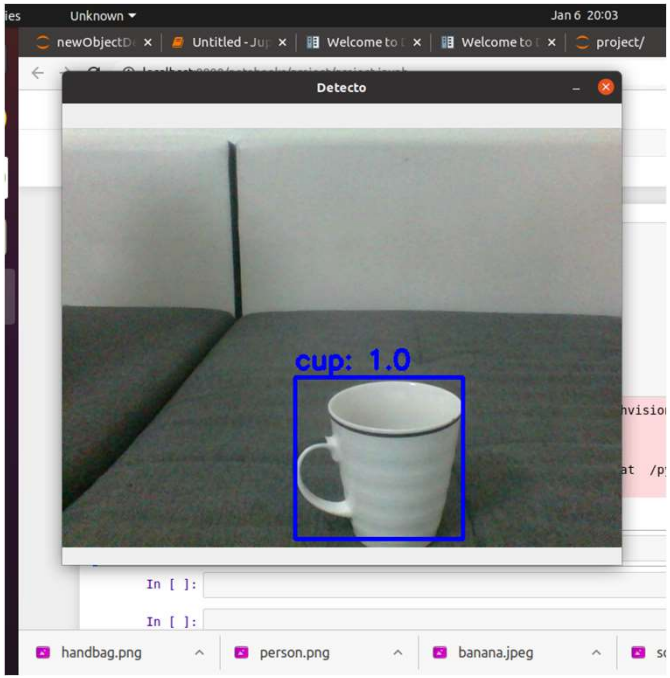
Results from existing models

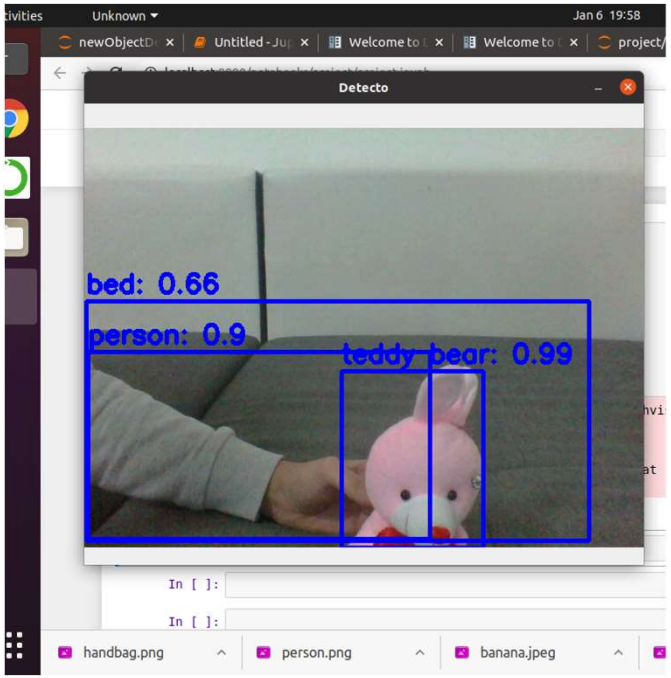


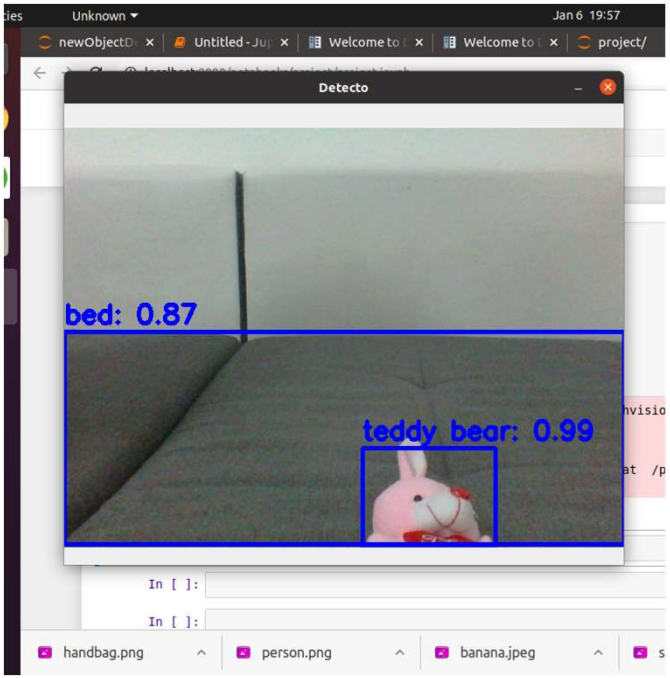


Results from Existing Model on Real time

- The model is able to detect the given objects and then prediction boxes are drawn exactly with expected confidence values.







Own Customized Implementation

Own customized Implementation

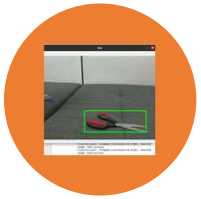
COLLECTED OWN DATASETS

LABELLED OBJECTS WITH THEIR CLASS NAMES AND THEIR POSITIONS IN IMAGE.

INTEGRATED EXTERNAL CAMERA FOR LIVE DETECTION USING OPENCV.

INITIALLY, CLASSIFICATION TASK IS DONE FOR OWN DATASET.

THEN, BOUNDING BOX IS DRAWN MANUALLY.



LATER, OBJECT DETECTION IS DONE USING PYTORCH LIBRARY DETECTO.

FINALLY, OBJECT DETECTION INSIDE THE BOX IS PERFORMED.

https://www.google.com/imgres?imgurl=https%3A%2F%2Fi.ytimg.com%2Fvi%

New Dataset Collection

293 Training Samples

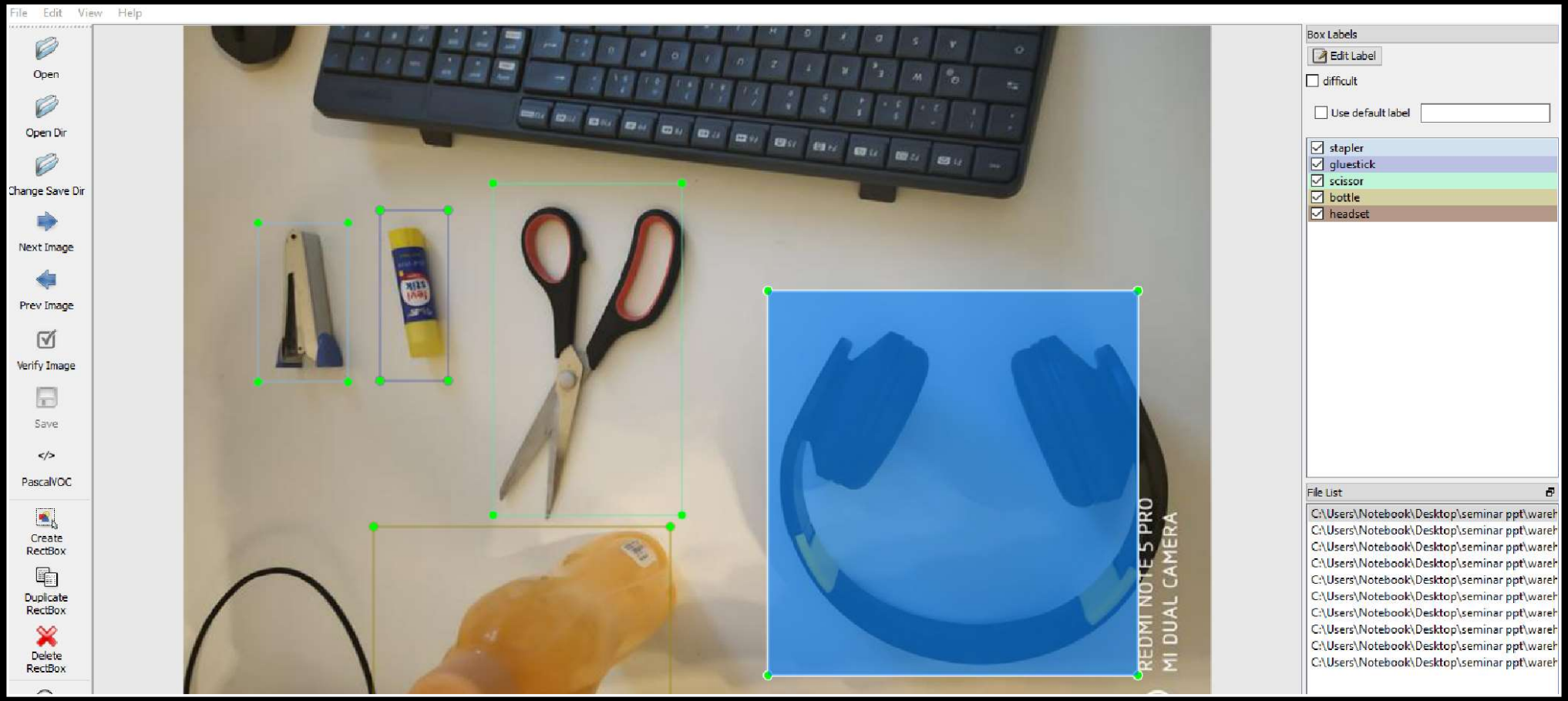
19 Test Samples

Classes:

- Gluestick,
- Scissor,
- Headset,
- Bottle
- Stapler

Data Augmentation (due to small dataset)

- RandomFlip('horizontal')
- RandomRotation(0.2)



1. Annotation tool- LabelImg for labelling

2.xml→csv

Modeling 1

Modeling 1- Classification + drawing Manual Bounding box

- Dataset: own customized dataset
- Deep learning Framework: keras
- Data Augmentation- Fliping(horizontal), Rotation
- **Pre-trained model Mobilnet v2**
- Convolution part is freezed from training,
- only the classification part is trained.
- Optimizer 'Adam' used for model compiling
- Loss- categorical cross entropy since more than two classes
- Learning rate: 0.0001, epochs=20

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_3 (InputLayer)         [(None, 160, 160, 3)]     0
_________________________________________________________________
sequential (Sequential)      (None, 160, 160, 3)       0
_________________________________________________________________
tf.math.truediv_1 (TFOpLambd (None, 160, 160, 3)       0
_________________________________________________________________
tf.math.subtract_1 (TFOpLamb (None, 160, 160, 3)       0
_________________________________________________________________
mobilenetv2_1.00_160 (Functi (None, 5, 5, 1280)        2257984
_________________________________________________________________
global_average_pooling2d_2 ( (None, 1280)              0
_________________________________________________________________
dropout_1 (Dropout)          (None, 1280)              0
_________________________________________________________________
dense_1 (Dense)              (None, 5)                 6405
=================================================================
Total params: 2,264,389
Trainable params: 6,405
Non-trainable params: 2,257,984
_________________________________________________________________
```

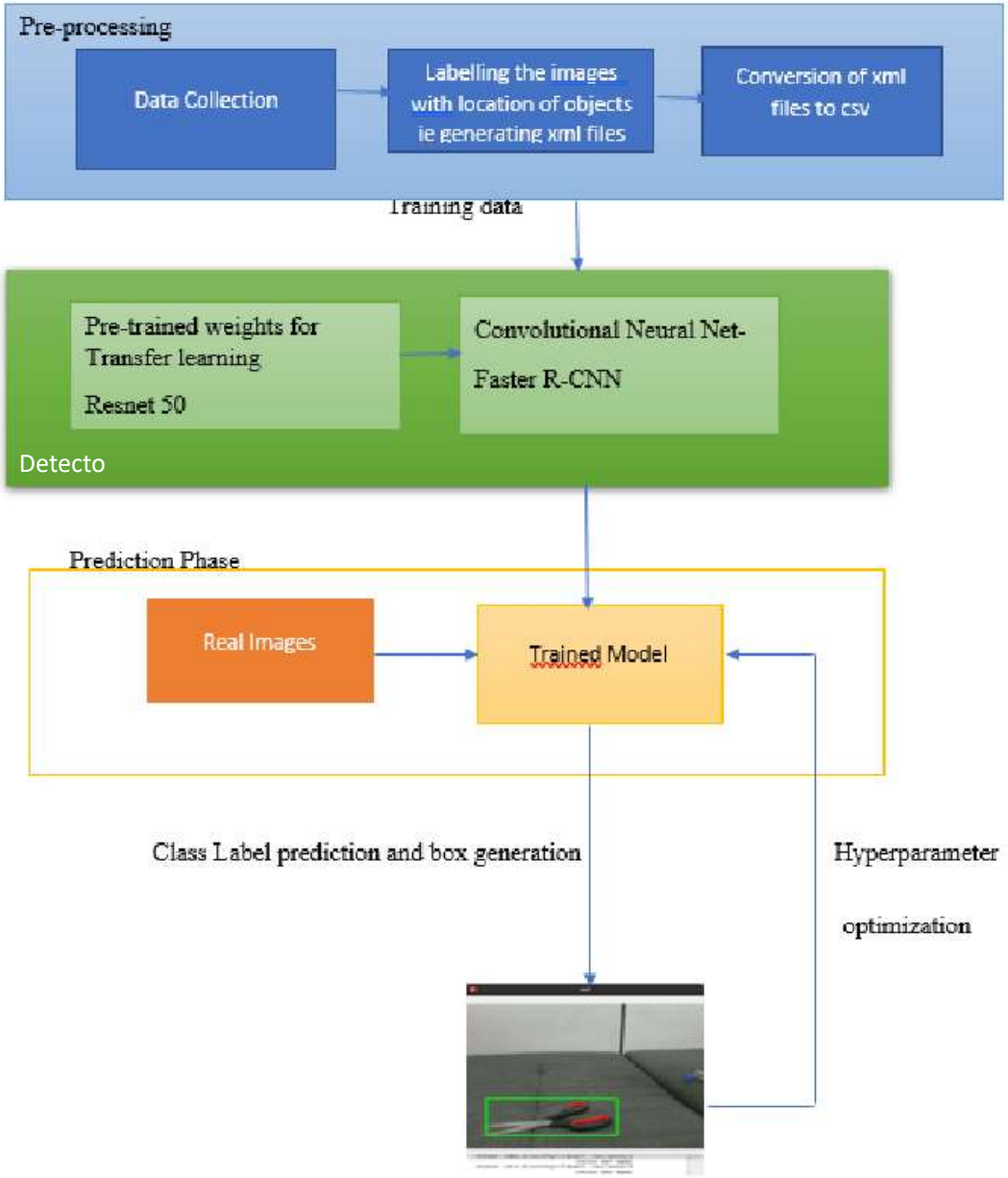
Drawing Bounding box

Manually image prediction bounding box is drawn.

Manually non-max suppression is done.

Results are not as expected, i.e, bounding box is not appropriate to the object.

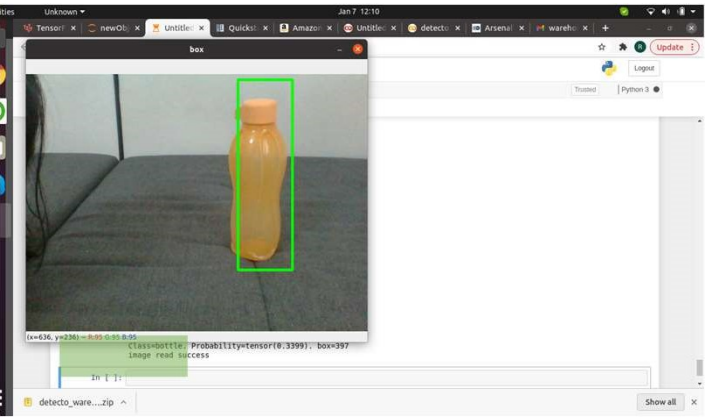
Modeling 2

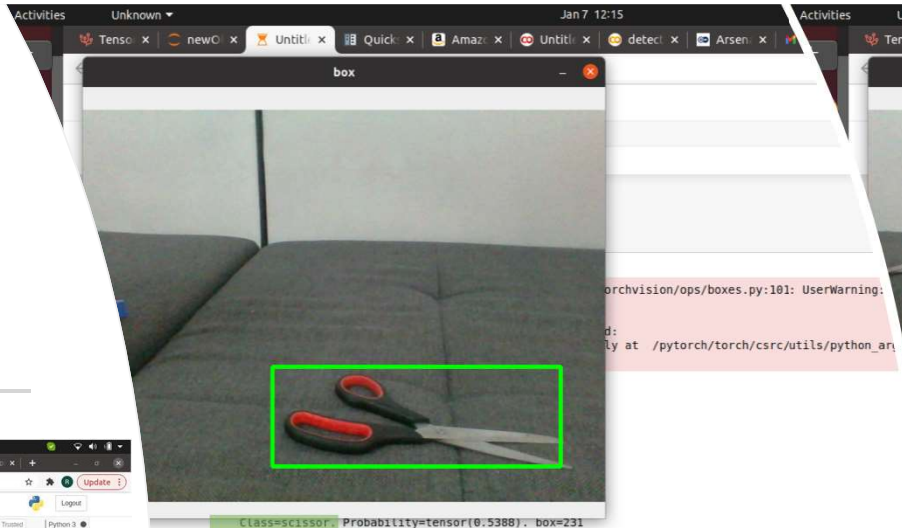


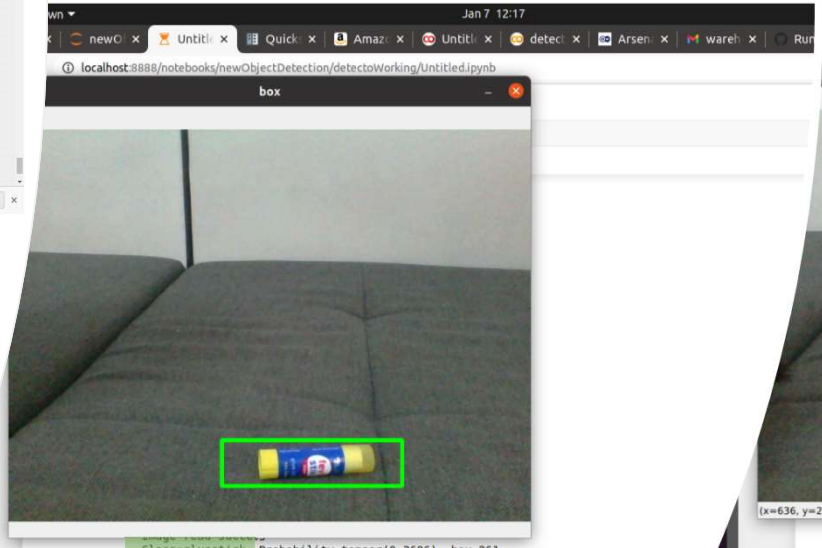
Modelling 2

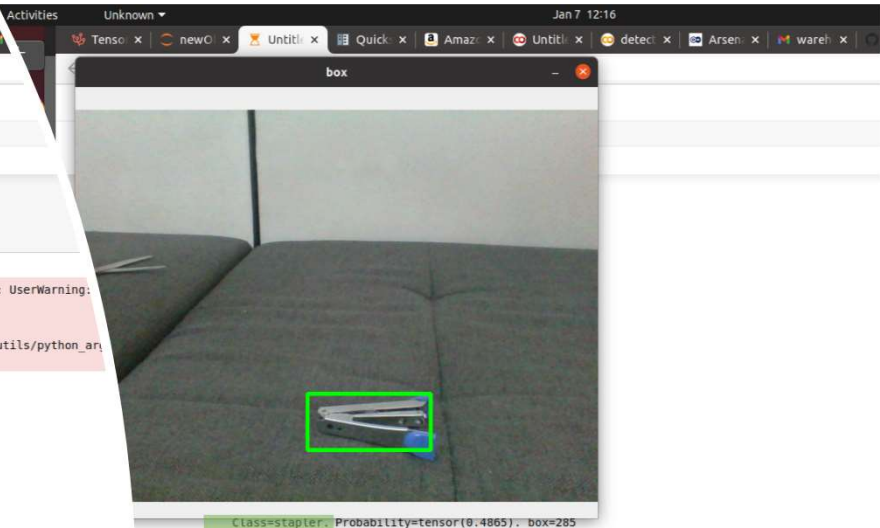
Dataset: own customized dataset
Preprocessing: torchvision.transforms
Model: faster r-cnn model (detecto)
Deep learning Framework: PyTorch
Camera: OpenCV

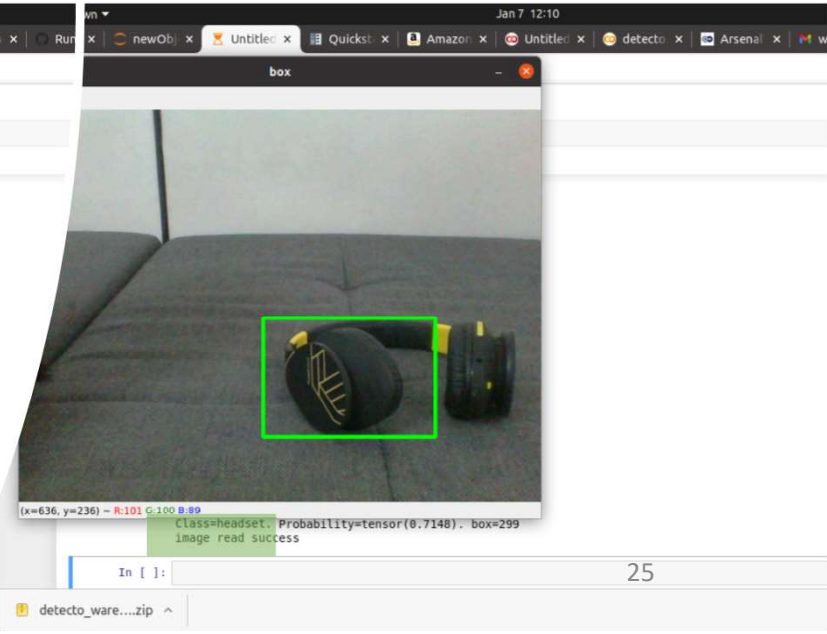
New Customized Implementation out-of Box



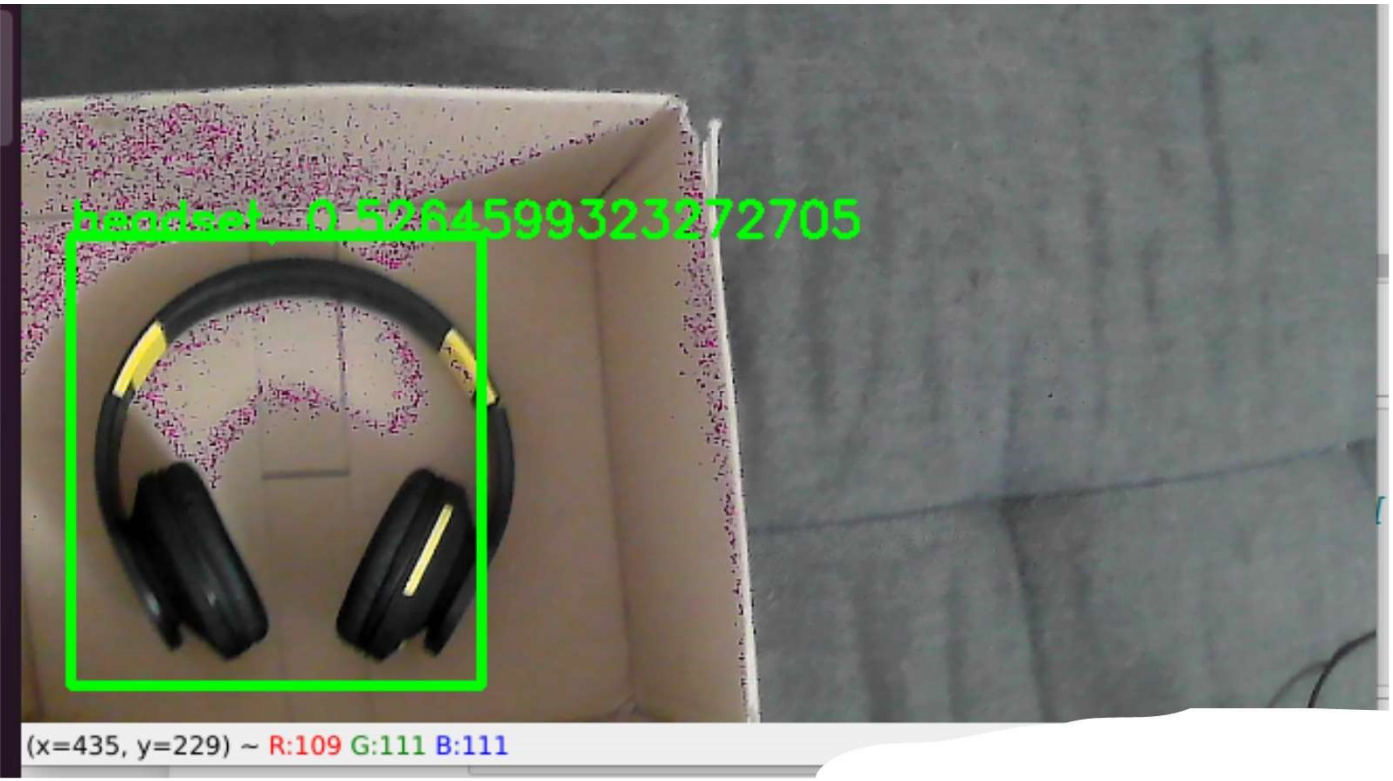






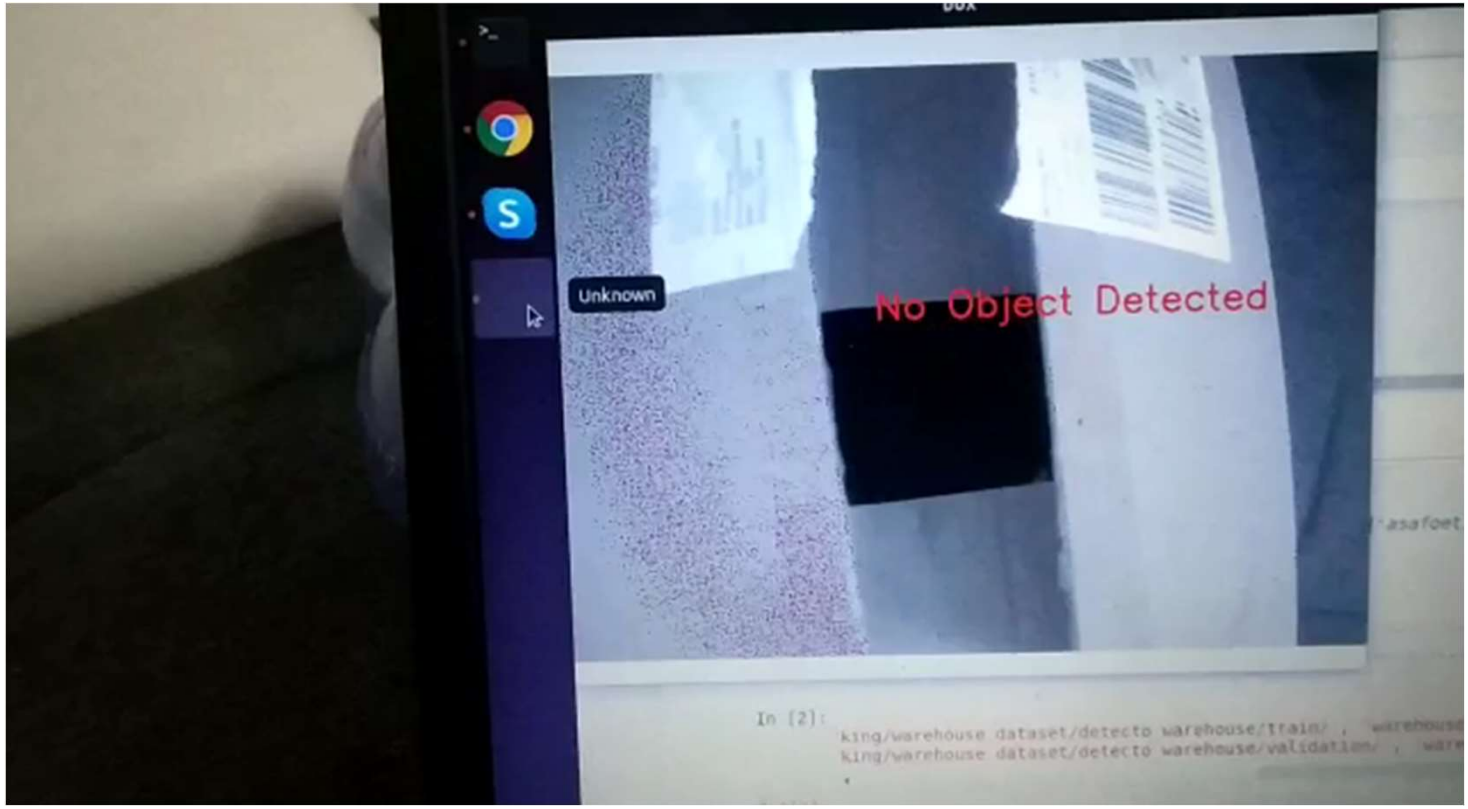


Note: labels and probabilities are displayed as text below

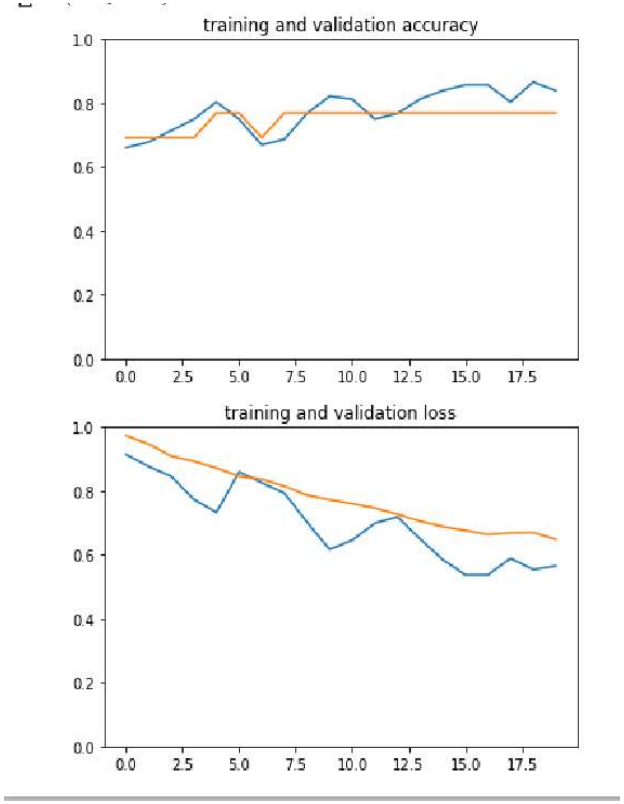


Object Detection inside the box

Demo in Real Time



Performance Analysis




```
from sklearn.metrics import classification_report


predictions=model.predict(valx)
y_pred = np.argmax(predictions, axis = -1)

print(classification_report(label_batch_v, y_pred))
```

```
              precision    recall  f1-score   supp

           0       0.71      1.00      0.83
           1       1.00      0.60      0.75
           2       0.80      1.00      0.89
           3       1.00      0.75      0.86
           4       1.00      1.00      1.00

    accuracy                           0.86
   macro avg       0.90      0.87      0.87
weighted avg       0.90      0.86      0.86
```

```
[ ]
```

Classification Report

Loss Analysis

Training loss calculated during each epoch is plotted above.

Conclusion

- Existing model is implemented and the results are promising.
- Own customized model is implemented and the results are good.
- Object detection inside the box is also implemented and the results are considerable.

Thanks for your Attention