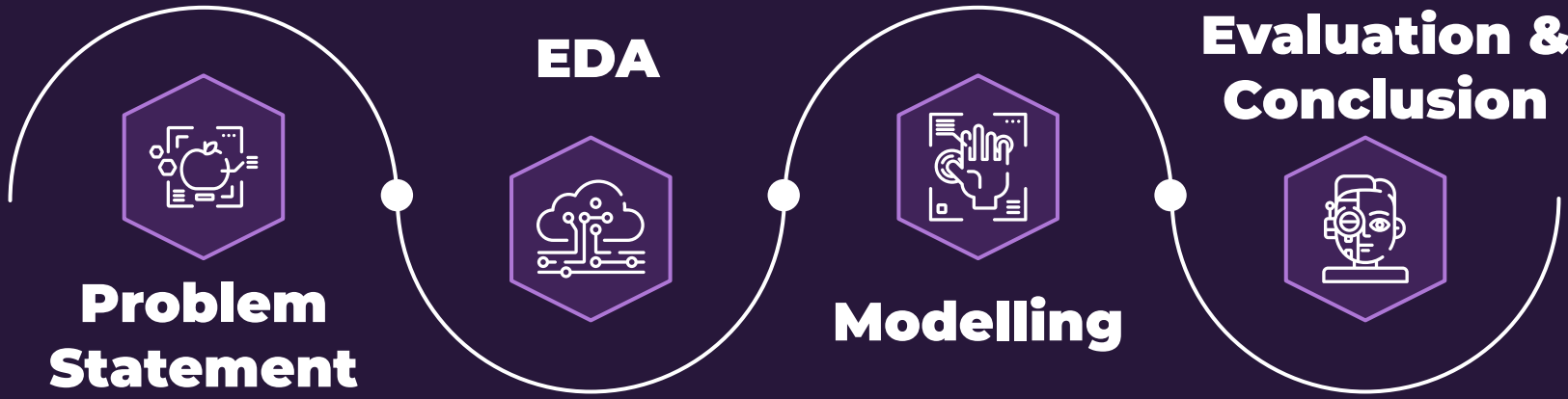
Shoe Image Classifier & Logo Detector

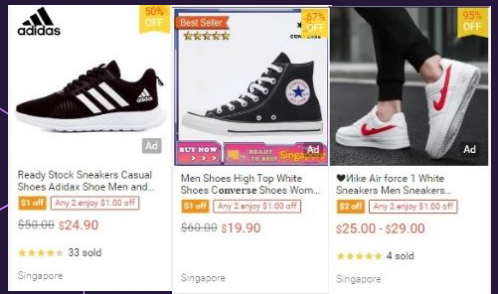
Project Outline



Problem Statement

Now

Product listings of counterfeit shoes are commonly found on major online marketplaces either as listings of counterfeit shoes or advertisements. They generally use images of the actual product, and are sold at a very low price.



Future

To protect the brand, consumer and merchant's interest, the aim is to build a model to classify the images of shoes from popular brands and detect their logos accurately. When combined with a selling price below a preset threshold, these listings can be proactively blocked from being posted/displayed in the search results and also allow for flagging of suspicious listings for review by the online merchant

EDA

Dataset

The image dataset which was obtained from Kaggle, consist of web images from popular shoe brands.

It contain 3 classes: Adidas, Converse & Nike

The images are organised in train & test folders, consisting of 711/114 images with train-test-split ratio 0.15

Resolution 240x240 pixels in RGB colour

EDA

Observations

- From the images of Adidas, Converse and Nike shoes, designs of the 3 brands are quite unique, hence we can expect to have a good separation when classifying the shoe images.

- Among the 3 brand logos, we expect Converse to have the highest accuracy in detection as it has the most distinct shape, a star in a circular shape. Adidas and Nike logos are expected to follow closely behind Converse. Nike may outperform Adidas in terms of detection accuracy as the 'tick' is more unique compared to '3 stripes'. The '3 stripes' could be misled by similar rectangular shapes in the shoe design

Modelling

Image Classification

Using pre-trained model and train a custom model to classify shoe images for the 3 brands.
Model will be evaluated using accuracy

Custom Object Detection

Train a custom model to recognize the objects(logos) in our dataset.
Model will be evaluated using mean average precision (mAP)

Image Classification Model

Selection of pre-trained model

- Our image classifier model consist of a pre-trained model and last layer which will be retrained to recognise images in our dataset

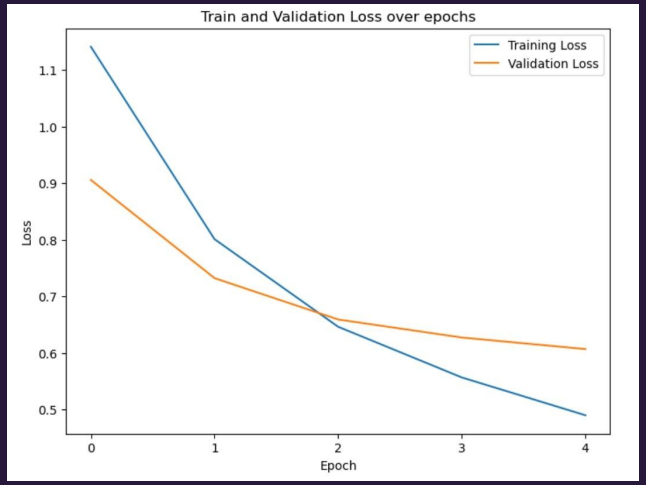
- The pre-trained transfer learning model is selected from the available ones in tensorflow keras based on the following criteria (i) inference time, (ii) accuracy, (iii) model size

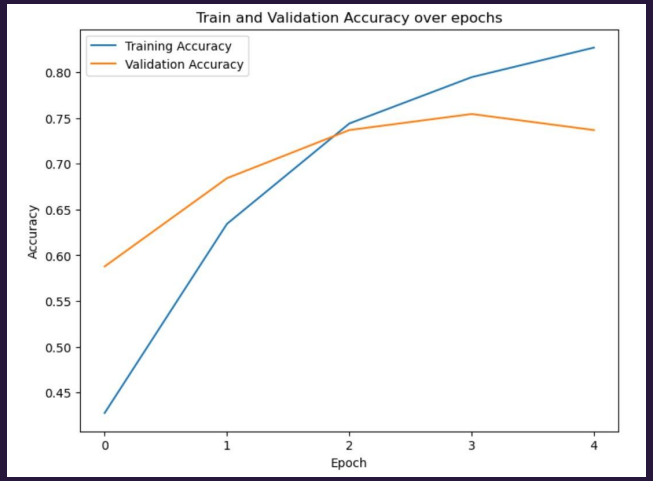
- MobileNetV2 is the most suitable, with the best combination, short inference time, high accuracy and small model size

Model	Size (MB)	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth	Time (ms) per inference step (CPU)	Time (ms) per inference step (GPU)
MobileNet	16	70.40%	89.50%	4.3M	55	22.6	3.4
MobileNetV2	14	71.30%	90.10%	3.5M	105	25.9	3.8
NASNetMobile	23	74.40%	91.90%	5.3M	389	27	6.7
InceptionV3	92	77.90%	93.70%	23.9M	189	42.2	6.9
ResNet50V2	98	76%	93%	25.6M	103	45.6	4.4
EfficientNetB0	29	77.10%	93.30%	5.3M	132	46	4.9

Image Classification Model

Training results





Validation accuracy

Validation loss saturate at 0.6 after 5 epochs

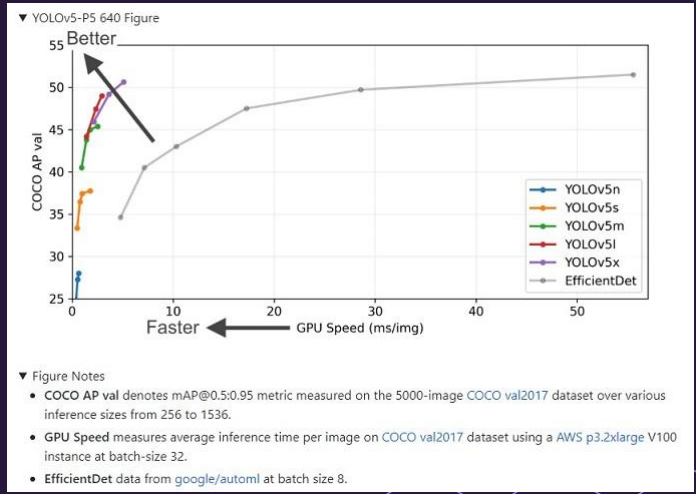
Object Detection Model Selection

2 main family of object detection models,

- two stage methods: higher detection accuracy/slower
- one stage methods: faster inference speed

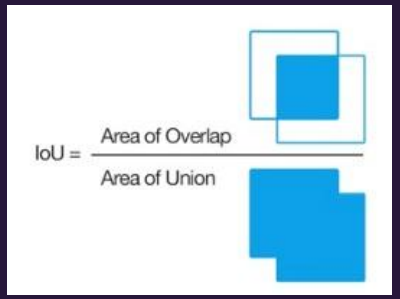
The models are evaluated using a mean Average Precision (mAP) metric on Microsoft COCO dataset (pre-trained to detect 80 common objects)

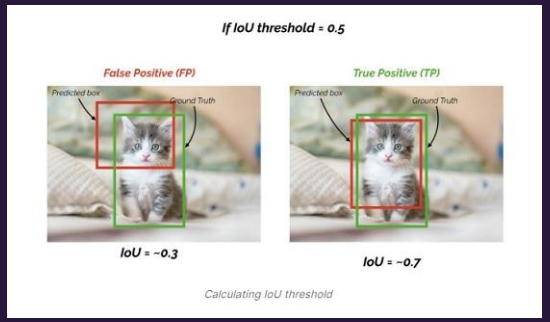
For our use case, we will select a suitable one stage pretrained model based on a good mAP and inference time. Among the popular models, YOLOv5(You Only Look Once) which was released in 2020 is chosen.



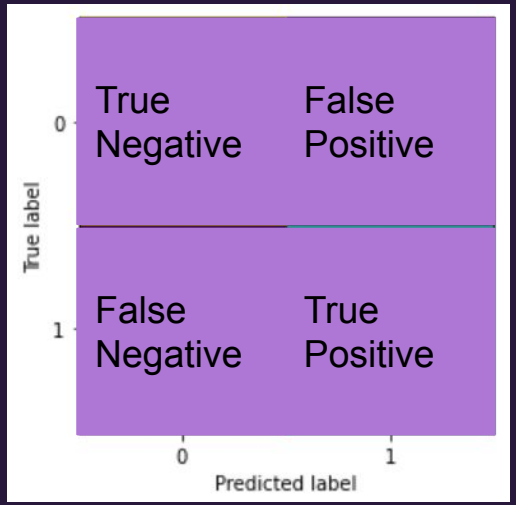
mean Average Precision (mAP)

1. Intersection over Union (IoU)





2. Calculate the Confusion Matrix from whether the model's prediction match with the ground truth



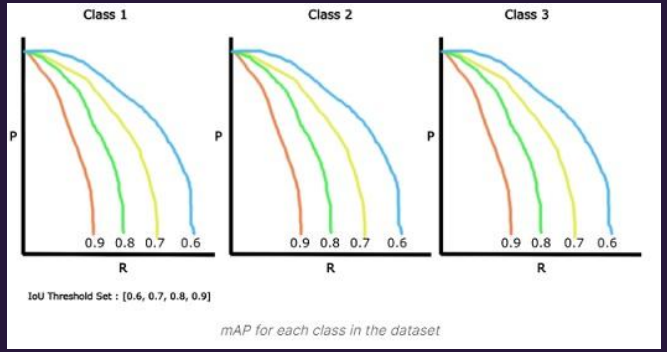
3. Calculate Precision & Recall

Precision = TP/(TP + FP)

Recall = TP/(TP + FN)

mean Average Precision (mAP)

4. Calculate AP from area under Precision-Recall(P-R) plot



mAP for each class in the dataset

5. Finally, for each class k, we calculate the mAP across different IoU thresholds, and the final metric mAP across test data is calculated by taking an average of all mAP values per class

$$mAP = \frac{1}{n}\sum_{k=1}^{k=n} AP_k$$

$AP_k =$ *the AP of class k*

$n =$ *the number of classes*

mAP multi-class formula

Custom Object Detection Model

Data Preparation

- Using Roboflow Annotate, a total of 56 shoe images are annotated by manually drawing bounding boxes over the logos

- Apply data augmentation, random rotation of +/- 15 degrees

- The final image dataset have 133 images after the preprocessing and data augmentation and exported in YOLOv5 PyTorch format

Custom Object Detection Model

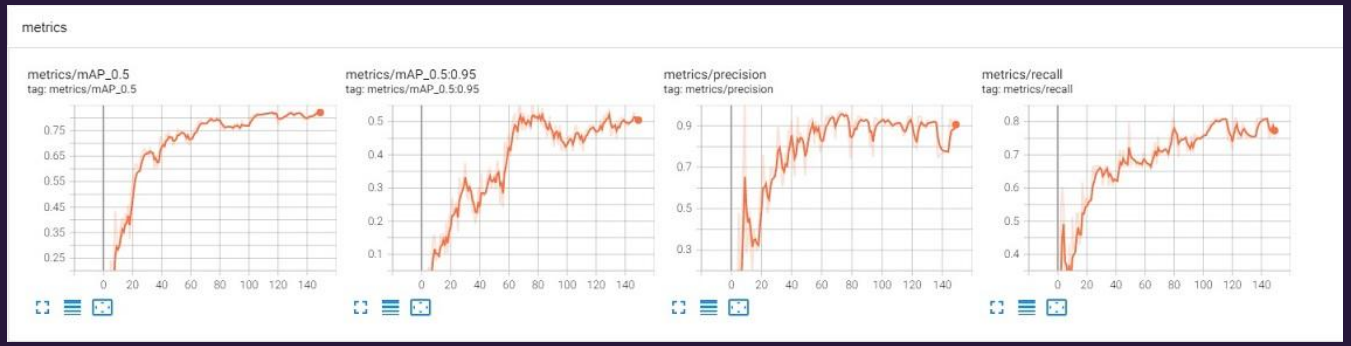
Model summary of training results

```
Model summary: 157 layers, 7018216 parameters, 0 gradients, 15.8 GFLOPs
```

Class	Images	Instances	P	R	mAP50	mAP50-95: 100% 1/1 [00:00<00:00, 10.29it/s]
all	11	17	0.96	0.691	0.788	0.559
adidas	11	7	1	0.358	0.528	0.359
converse	11	3	0.903	1	0.995	0.742
nike	11	7	0.977	0.714	0.84	0.576

mAP_0.5(all)

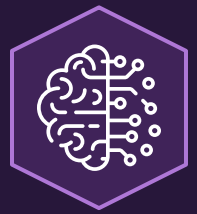
Tensorboard metrics



Evaluation & Deployment

User Input

Users upload shoe images, and click the predict button



Process

The model will process the uploaded images and predict the brand of the shoes based on the image classification model and detect the brand logos using the custom trained object detection model

Output

The predicted brand name and associated probability, brand logos with bounding boxes and confidence

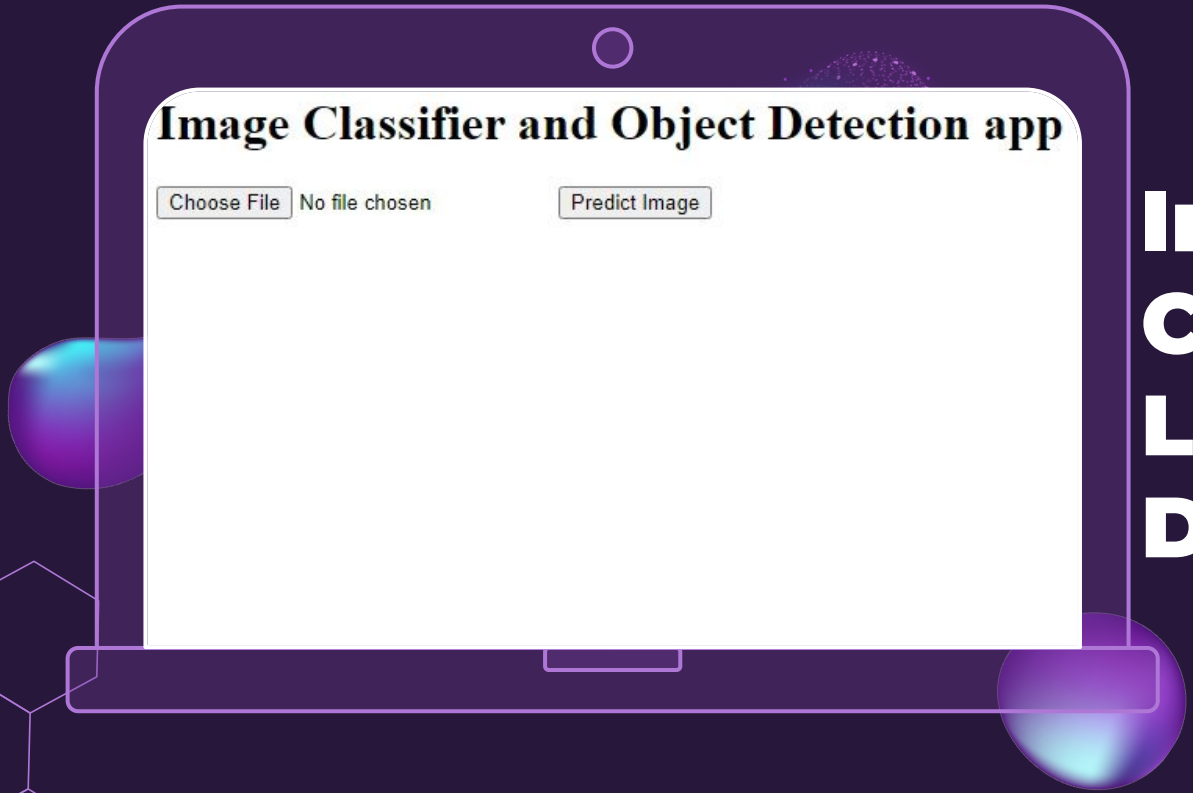


Image Classifier & Logo Detector app

Conclusion

Our project have successfully deployed an image classification model to classify shoe images with accuracy of 0.73, and a custom object detection model to detect logos with mAP_0.5 of 0.78.

When combined with a selling price below a preset threshold, the listings of counterfeit shoes/advertisements can be proactively blocked from being posted/displayed in the search results and also allow for flagging of suspicious listings for review by the online merchant.

It is recommended to use predictions from both models for our use case, with more weightage given to the logo detector as it has a better metric

Future work

- Improve on preprocessing by removing the image background automatically
- Improve on the model's performance with more image data