Allianz Logo Detection (ALD)

Contents

1. Introduction

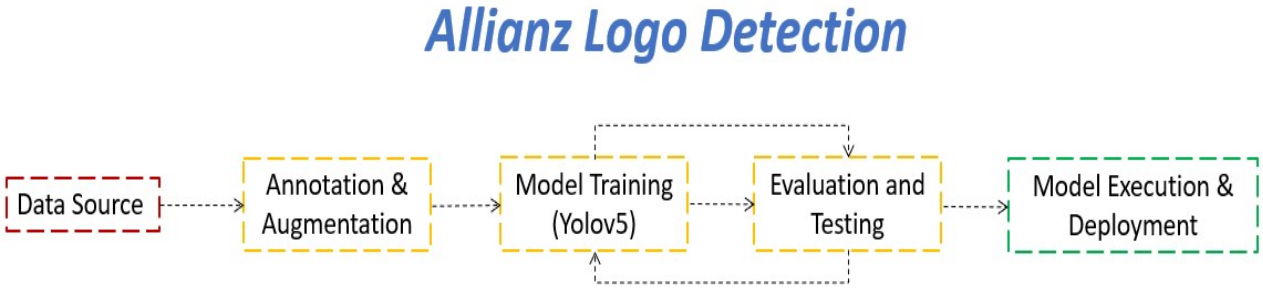
This is a development and design document is currently capturing the development process, implementation details, evaluation and deployment procedure Allianz Logo Detection.

Goal

The goal is to detect only Allianz logos in given images.

Example:

2. Approach



2.1 Data Source & Availability

For the above goal, we need multiple images with Allianz logos to annotate and train the model.

Two Source of Availability:

1. Google Images:
    a. Download the images from google images.
2. Allianz Video:
    a. Download Allianz advertisement video
    b. Extract the frames/images from the video using CV library in python.
    c. Refer the below screen shoot(Image-1) to convert video to images in python.

Image-1: convert video to images in python.

```
# Importing all necessary libraries
import cv2
import os

# Read the video from specified path
vid = cv2.VideoCapture("Allianz_advertisement_1.mp4")

try:

    # creating a folder named data
    if not os.path.exists('frames'):
        os.makedirs('frames')

# if not created then raise error
except OSError:
    print('Error: Creating directory of data')

# frame
currentframe = 0

while (True):

    # reading from frame
    success, frame = vid.read()

    if success:
        # continue creating images until video remains
        name = 'frames/frame' + str(currentframe) + '.jpg'
        print('Creating...' + name)

        # writing the extracted images
        if currentframe%10 == 0:
            cv2.imwrite(name, frame)

        # increasing counter so that it will
        # show how many frames are created
        currentframe += 1
    else:
        break

# Release all space and windows once done
vid.release()
cv2.destroyAllWindows()
```

Data for this task was obtained by downloading images from google and captured few frames from a video clip as well. A total of 377 images were captured. I used 347 images for training, 14 for validation, and 16 for test part.

2.2 Annotation & Augmentation

The next step is to annotate the images obtained. To do that, I used ROBOFLOW

Roboflow is a great platform to annotation and augmentation.

Preprocessing:

I resized all the images to 416x416.

Augmentation:

I flipped the images to horizontal and did rotation -20 to +20. Finally, I applied grayscale for 30% of images.

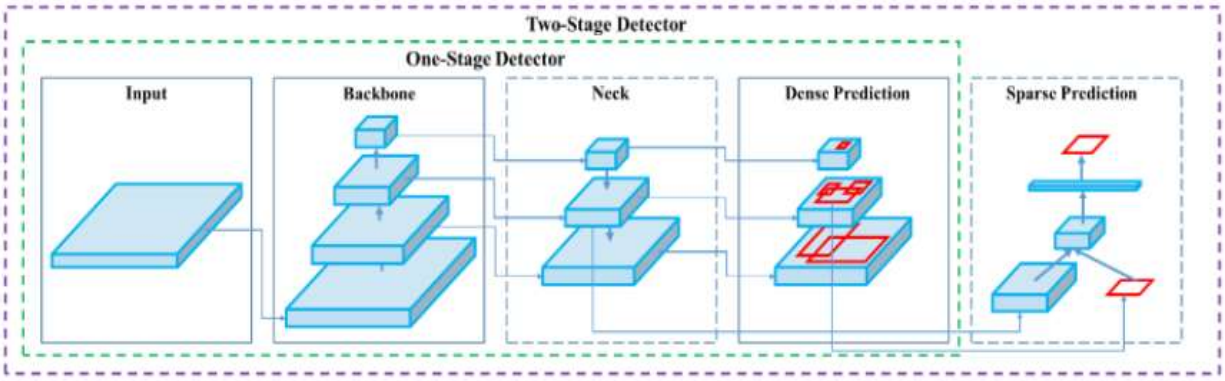
Watch this tutorial for better practice.

2.3 Model Training, Evaluation and Testing

To detect the Allianz logos we are using YOLOv5, to know more about YOLOv5 click here

To understand how Yolov5 improved the performance and its architecture, first we have to take look into YOLOv4, because YOLOv5 almost resembles YOLOv4.

let us go through the following high-level Object detection architecture:



Input: { Image, Patches, Image Pyramid, ... }

Backbone: { VGG16 [68], ResNet-50 [26], ResNeXt-101 [86], Darknet53 [63], ... }

Neck: { FPN [44], PANet [49], Bi-FPN [77], ... }

Head:

Dense Prediction: { RPN [64], YOLO [61, 62, 63], SSD [50], RetinaNet [45], FCOS [78], ... }

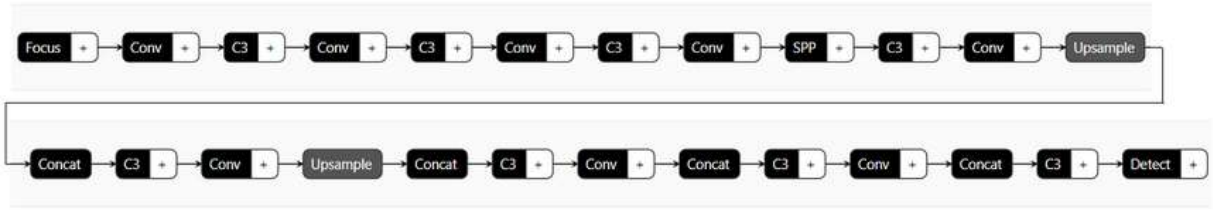
Sparse Prediction: { Faster R-CNN [64], R-FCN [9], ... }

General Object Detector will have a backbone for pre-training it and a head to predict classes and bounding boxes. The *Backbones* can be running on GPU or CPU platforms. The *Head* can be either one-stage (e.g., YOLO, SSD, RetinaNet) for Dense prediction or two-stage (e.g., Faster R-CNN) for the Sparse prediction object detector. Recent Object detectors have some layers (*Neck*) to collect feature maps, and it is between the backbone and the Head.

YOLOv5 almost resembles YOLOv4 with some of the following differences:

- YOLOv4 is released in the Darknet framework, which is written in C. YOLOv5 is based on the PyTorch framework.
- YOLOv4 uses .cfg for configuration whereas YOLOv5 uses .yaml file for configuration.

YOLOv5s model displayed in Netron



Yolov5s model

Please go through the YOLOv5 Github repo for additional information.

Model Training and Evaluation Procedures:

- Install all dependencies
- Clone the YOLO5 git repo using https://github.com/ultralytics/yolov5
- Import all required libraries
- Import dataset using roboflow api, refer below.

```
!pip install roboflow
from roboflow import Roboflow
rf = Roboflow(api_key="WfUGPJklzBhlucxS69aA")
project = rf.workspace("logodetection-yq2aw").project("logo-owfjy")
dataset = project.version(1).download("yolov5")
```

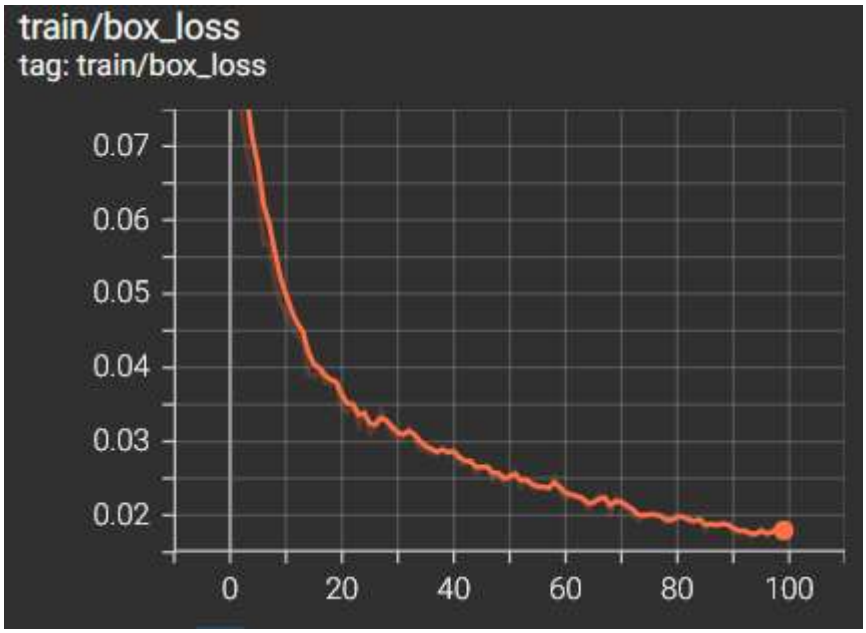
- Train the model using YOLO5 train.py script with configs and params like below given.

```
!python train.py --img 416 --batch 16 --epochs 100 --data {dataset.location}/data.yaml --weights yolov5s.pt --cache --single-cls
```

- Use tensorboard to evaluate the model with all relevant metrics.
- Retrain a model, with updated params, then again evaluate with updated model.
- Save the model.

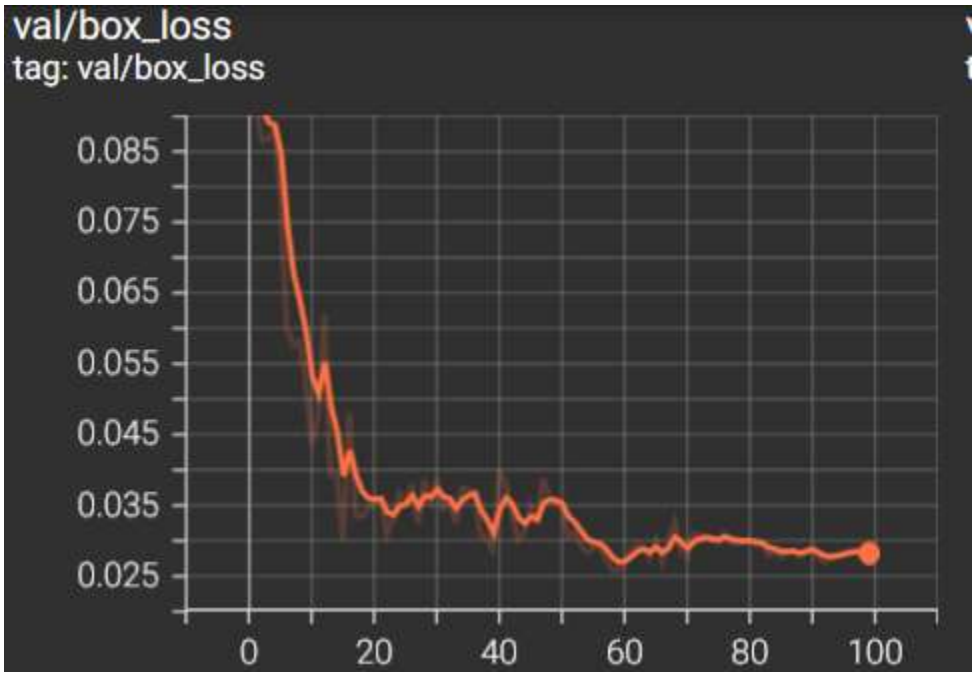
Model Performance:

Trained Model Loss Box:



Recall: 0.926

Evaluation Set Loss Box:



Test Result:

2.4 Model Execution/Deployment:

1) Local Host

To deploy a model in LocalHost using FastAPI follow the below steps:

- Install python in your system
- Create new virtual environment
- Install all dependencies which are all available in requirement.txt
- Clone/download https://github.com/SEKARSARAVANAKUMAR/Logo_Detection.git
-
- Run WebApplication.py file
  python WebApplication.py

No, you can detect the Allianz logos in your local host with below URL

http://0.0.0.0:3000/docs#/default/main__post

2) AWS - EC2 Deployment

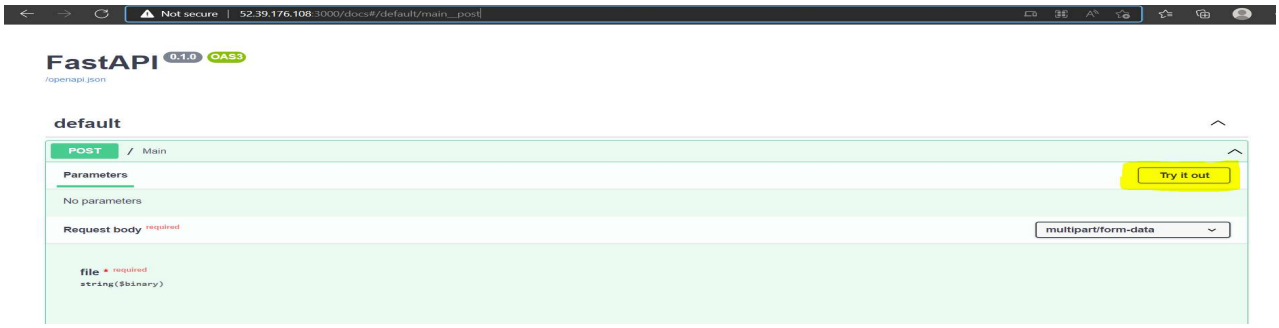
a. Deployed similarly in EC2-centos instance.

You can detect the Allianz logos in your browser using below URL

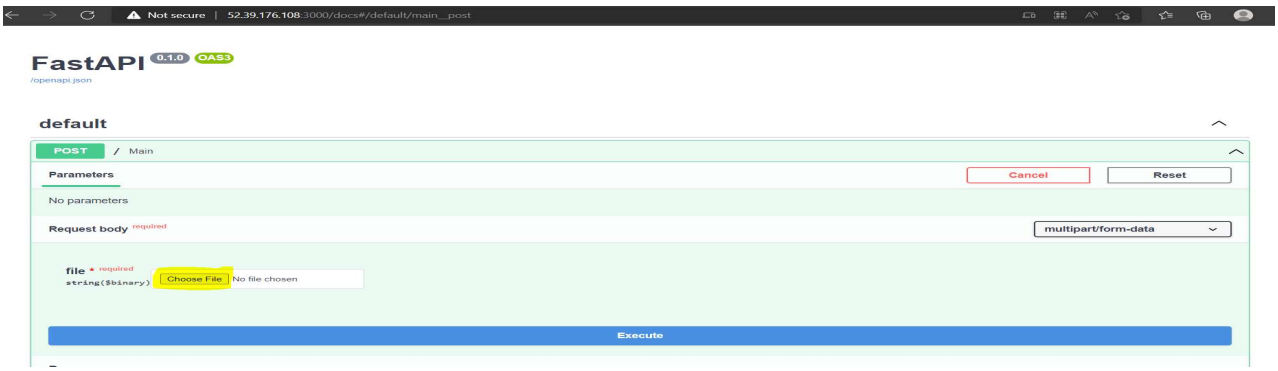
http://52.39.176.108:3000/docs#/default/main__post

Steps to detect the images using above URL

- Click “Try it out” – highlighted in yellow color.



- Click “choose file” box – highlighted in yellow color.
- Upload any image



- You can see the result in response body – see below – highlighted in yellow color