**Allianz Logo Detection**

**GitHub Repo :**

<https://github.com/RithuRmN/logo_detector/>

**Folder Structure**

**Data** :

Contains all the data related to project

[ Annotation CSV , Train & Test Images , Augmented Output , TF\_record ]

**Experiment** :

Contains seperate folders for each experiment with seperate params file to ensure reproducibility and reuse of code stored in src

[ Params.yaml , Readme.md]

**Src** :

Contains all the source code

**Data\_Processing** – Code for pre-process the data

**Metric** – code for evaluating the model and finding mAP charts

**Object\_Detection** – Contains YOLO training and inference code

**Utils** – General util functions to use across the codes

**Web\_App** – FASTAPI webapplication codes , Trained Model

**Procedure to Run the FASTAPI Application**

* Download and extract github repo <https://github.com/RithuRmN/logo_detector/>
* Go to directory **src->Web\_App**
* Open **Readme.md** and install required packages in an environment
* Type “ **uvicorn main:app –reload ”** in command terminal
* Open browser and go to **http://127.0.0.1:8000/**
* Select input image ( New pop-up will be opened in taskbar)
* Select output path ( New pop-up will be opened in taskbar)
* Click **Detect**
* After completion , Click **Open Output Folder** to see the saved output images



Fig : Screenshot of logo\_detector application

**Problem statement**

Build a custom Allianz Logo Detector which detect only Allianz logos , crop the detected logo and save.

**Solution**

Developed a YOLOV3 detection model which detects only Allianz logo and deployed the model as webservice using FASTAPI .

**Modeling**

**Dataset collection**

Collected 75 images which contains Allianz logo from internet for training.

**Labelling**

Labelled the dataset (75 images )collected using **VGG Annotator** ( VIA is an opensource annotation tool ) saved output as csv.

**Augmentation**

Performed Rotation [ 0 ,90 , 270 ] on training dataset(with BBOX) to increase the size of dataset ( 225 images ).

**Preprocessing**

* Converted VIA CSV output to required format [ filename ,class, xmin,ymin,xmax,ymax]
* Performed Train and Validation Split ( 80 :20)
* Created TF\_Records

**Training**

Trained Yolov3 model ( TF2 implementation) in 2 stages for better accuracy and saved the model.

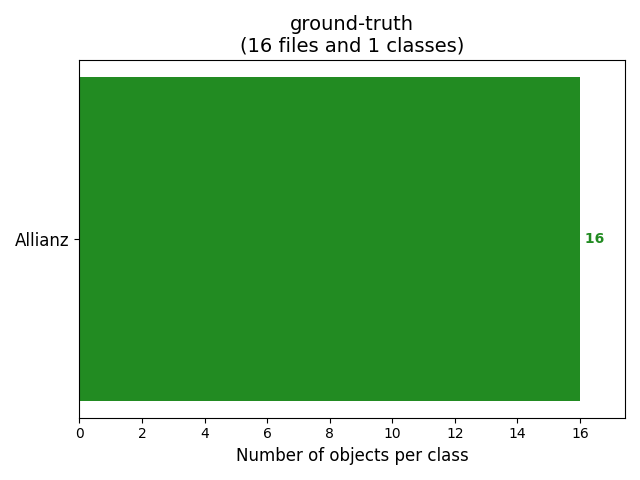
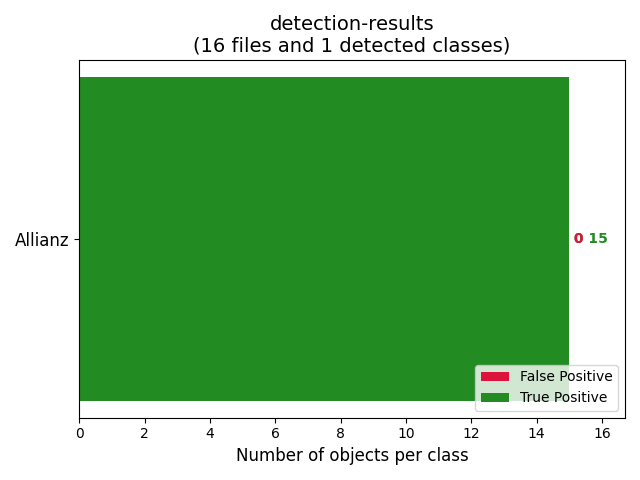
(Params file available inside *experiment->OD\_LOGO\_DETECTOR* folder )

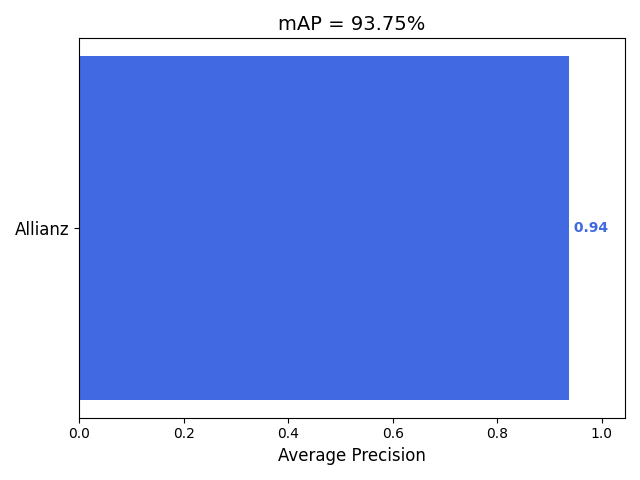
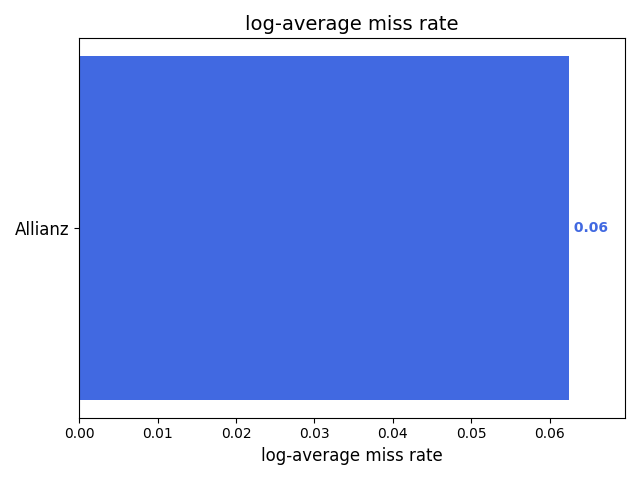
Trained model uploaded as TF\_Serving model inside *Src->WebApp->Model*

\*(Model checkpoint , Pre-trained weight , TF\_Record not uploaded in Github due to large file size )

**Model Evaluation**

**mAP** : Checked model performance using Mean Average Precision charts. ( Validation Data )



 **Test**  : Downloaded and created new test set , Performed inference using trained model and evaluated the result .

|  |  |  |  |
| --- | --- | --- | --- |
| **Total test images** | **Logo Detected** | **Logo Not Detected** | **Accuracy** |
| **11** | **10** | **1** | **91%** |

**Deployment**

Converted trained model into TF\_serving and deployed it locally using FASTAPI.