# Object Recognition Pipeline Documentation

This documentation provides an overview of the implementation and steps taken to complete the "Object Recognition" code challenge. The goal of this challenge is to build an object recognition pipeline using a pre-trained model and evaluate its performance on a dataset.

I have followed <https://towardsdatascience.com/convert-pascal-voc-xml-to-yolo-for-object-detection-f969811ccba5> for dataset preparation and <https://blog.paperspace.com/train-yolov5-custom-data/> for model training and evaluation.

### Dataset

The chosen dataset for this challenge is the "The Oxford-IIIT Pet Dataset". This dataset contains images of 37 different breeds of cats and dogs. The dataset provides images and their corresponding labels for training and testing. Downloaded from the link <https://www.robots.ox.ac.uk/~vgg/data/pets/>[1]

### Object Detection Model

For this challenge, the YOLOv5 model architecture from the open-source Ultralytics YOLOv5 project was selected as the object detection model. YOLOv5 is a state-of-the-art object detection algorithm known for its accuracy and efficiency.

To download the architecture: “git clone <https://github.com/ultralytics/yolov5>”

### Implementation Steps

The implementation of the object recognition pipeline follows the provided instructions. The following steps were performed:

1. Dependencies installation
   * The dependencies required for this system is given in “requirements.txt”. “pip install -r yolov5/requirements.txt” will install all unistalled dependencies
2. Data Augmentation: YOLOv5 applies image space and colour space augmentations in the train loader. [https://github.com/ultralytics/yolov5/discussions/10469] So I haven’t done any augmentation for this model. “data\_augmentation.py” is a sample function I have written for data augmentation.
3. Dataset Loading and Splitting:

* The downloaded dataset was in XML format and Yolo need the dataset in “.txt”. I have converted XML annotations to YOLO annotations using the function “data\_ annotation.py”. The function parses the content of the XML and creates a Txt file with the class number and bounding box. All Txt files with stored in the label folder.
* The function “data\_split.py” is used to split the dataset into train, validation and test.

1. Model Training:

* Yolov5 have inbuilt code to train the model.
* Before training, we need to build a data config YAML file. The file is stored in “yolov5/data/petdataset.yaml”. This file has locations of train, test, and validation images, the number of classes and the name of classes.
* Yolo has many hyperparameter config files which help us define the hyperparameters for our neural network. I have used “hyp.scratch-med.yaml” which gives medium augmentation to the dataset.
* The architecture I selected is YOLOv5s, the second-smallest and fastest model available. The architecture is given in in “yolov5s.yaml” file
* Trained the model for 3 epoch with 32 batch size. The command
  + “python3 train.py --img 640 --cfg yolov5s.yaml --hyp hyp.scratch-med.yaml --batch 32 --epochs 3 --data petdataset.yaml --weights yolov5s.pt --workers 24 --name pet\_det\_26jun”.
* The trained model stored as pet\_det\_26jun and it has .925 map@50.

5. Model inference and evaluation: Yolo provides the function for inference and evaluating the model.

* The model is tested on a test dataset. The command used to test the model
  + python3 detect.py --source ../test/images/ --weights runs/train/ pet\_det\_26jun/weights/best.pt --conf 0.25 --name pet\_det\_26jun
  + the test result stored in “runs/detect/pet\_det\_26jun/”
* To evaluate the model
  + python3 val.py --weights runs/train/pet\_det\_26jun/weights/best.pt --data petdataset.yaml --task test --name yolo\_det
  + This script calculates for us the Average Precision for each class, as well as mean Average Precision. The evaluation result of this model is given below.
  + The evaluation result is stored in “yolov5/runs/val/yolo\_det2/” This includes confusion matrix, f1 curve, p\_curve, pr\_curve,a nd R\_curve.

A picture containing text, screenshot, font

Description automatically generated

### Handling Bias:

* The dataset had some missing data. It is handled during the data-splitting step. Some images in the dataset didn’t have labels, Removed such images during the data splitting steps.
* The Default hyperparameters are in hyp.scratch-low.yaml. I have used hpy.scratch-med.yaml file which has higher augmentation hyperparameters. This will reduce and delay overfitting, allowing for longer training and higher final mAP. hyperparameters hyp['obj’] given 0.7 which will help reduce overfitting.

##Steps:

1. Download dataset from <https://www.robots.ox.ac.uk/~vgg/data/pets/>[
2. git clone <https://github.com/ultralytics/yolov5> #install yolov5
3. pip install -r yolov5/requirements.txt #install dependencies
4. copy image and XML folder from dataset to the current directory
5. python data\_ annotation.py #this convert xml file to yolo format
6. python data\_split.py #split dataset to train test and validation set
7. create data config file as shown in “yolov5/data/petdataset.yaml”
8. cd yolov5
9. python train.py --img 640 --cfg yolov5s.yaml --hyp hyp.scratch-med.yaml --batch 32 --epochs 3 --data petdataset.yaml --weights yolov5s.pt --workers 24 --name pet\_det #train the model
10. python detect.py --source ../test/images/ --weights runs/train/ pet\_det/weights/best.pt --conf 0.25 --name pet\_det #do inference in the test dataset
11. python val.py --weights runs/train/pet\_det/weights/best.pt --data petdataset.yaml --task test --name yolo\_det #evaluate the model