# Configuration Manual

MSc Research Project
Programme Name

Lucky Shrivastava
Student ID: x21198594

School of Computing
National College of Ireland

Supervisor: Christian Horn

**National College of Ireland**
**Project Submission Sheet**
**School of Computing**

| | |
|---|---|
| **Student Name:** | Lucky Shrivastava |
| **Student ID:** | x21198594 |
| **Programme:** | Programme Name |
| **Year:** | 2023 |
| **Module:** | MSc Research Project |
| **Supervisor:** | Christian Horn |
| **Submission Due Date:** | 14/12/2023 |
| **Project Title:** | Configuration Manual |
| **Word Count:** | 594 |
| **Page Count:** | 7 |

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

**ALL** internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

| | |
|---|---|
| **Signature:** | |
| **Date:** | 14th December 2023 |

**PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:**

| | |
|---|---|
| Attach a completed copy of this sheet to each project (including multiple copies). | ☐ |
| **Attach a Moodle submission receipt of the online project submission**, to each project (including multiple copies). | ☐ |
| **You must ensure that you retain a HARD COPY of the project**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. | ☐ |

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

| **Office Use Only** | |
|---|---|
| Signature: | |
| Date: | |
| Penalty Applied (if applicable): | |

# Configuration Manual

Lucky Shrivastava
x21198594

## 1 Introduction

The Configuration manual explains step by step process involved in the implementation of this research Real-time Pedestrian Detection using YOLO-NAS algorithm. The pre-requisite configuration of Hardware and Software for this research has been mentioned in this Configuration Manual. The aim of this research is to identify and detect Pedestrian using the YOLO-NAS technique. It is a Single Stage Detection (SSD) technique which uses the Neural Architecture Search for finding the best architecture for detection. It is versatile, robust and fast enough to be implemented for real-time implementation. The Caltech dtatset [1] has been explored in this research for detecting Pedestrians.

## 2 System Specification

The research project was conducted using the open-source software Jupyter Notebook and Google Colab which is an effective tool with pre-loaded work environment for easy installation and use of libraries. The Hardware and Software configuration required for this research are listed below.

### 2.1 Hardware

The hardware parts consist of the CPU (central processing unit), memory, storage, network cards, motherboards, and other important components. The specifics employed in this project are outlined in Figure 1. During the project's initial phase, a local environment was utilized for data sorting, visualization, and processing. Additionally, the system incorporates a storage space of 512 GB. System Specification:

Google Colab Specifications:

- RAM - 12 GB
- GPU - NVIDIA T4
- Disk Space : 100 GB

[1] https://www.kaggle.com/datasets/kalvinquackenbush/caltechpedestriandataset/data

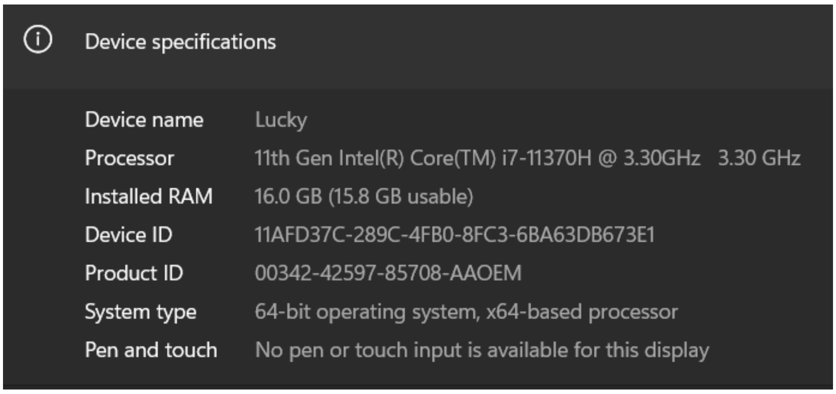


Figure 1: Local System Hardware Specification

### 2.2 Software

This research was executed using Python Programming Language. All the processes from Importing libraries, data extraction, data Pre-processing, Exploratory Data Analysis, Model Building and Evaluation metrics are done through the python on Jupyter Notebook and Google colab. The Local environment is used in the initial phase for Data Pre-processing and then Google Colab is used for setting the training parameters and Model Building.

- Programming Language - Python 3.10.12
- Cloud IDE - Google Colab
- Annotation File - Provided with the dataset and used the recommended github notebook for converting it to YOLO format [2]

Run this command to check the GPU specification in your system as shown in figure 2

## 3 Importing Libraries

The first step in this research is to import all the necessary libraries required like numpy, pandas, torch vision, super-gradients, pycocotools, matplotlib, scikit learn, etc as shown in Figure 3. If any other libraries are required they can be imported later.

### 3.1 Generating Annotation File

Now, the annotation files are generated for the video sequences of the data as shown in Figure 4. to index it with Bounding Boxes later.

### 3.2 Generating Image Files from Video Sequences

The Video sequences are converted into images for detecting the pedestrians using the Bounding Box as shown in Figure 5 and Figure 6

[2]https://github.com/simonzachau/caltech-pedestrian-dataset-to-yolo-format-converter

```
!nvidia-smi
executed in 125ms, finished 00:40:25 2023-12-12
Tue Dec 12 00:40:25 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 512.78       Driver Version: 512.78       CUDA Version: 11.6     |
|-------------------------------+----------------------+----------------------+
| GPU  Name            TCC/WDDM | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA GeForce ... WDDM  | 00000000:01:00.0 Off |                  N/A |
| N/A   38C    P8     2W /  N/A |      0MiB /  4096MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    0   N/A  N/A      2404    C+G   ...cw5n1h2txyewy\LockApp.exe    N/A      |
|    0   N/A  N/A      9260    C+G   ...artMenuExperienceHost.exe    N/A      |
|    0   N/A  N/A     10084    C+G   ...ystemEventUtilityHost.exe    N/A      |
|    0   N/A  N/A     12612    C+G   ...ser\Application\brave.exe    N/A      |
|    0   N/A  N/A     13608    C+G   ...4vj0pshhgkwm\Telegram.exe    N/A      |
|    0   N/A  N/A     13652    C+G   ...2txyewy\TextInputHost.exe    N/A      |
|    0   N/A  N/A     14048    C+G   ...v10z8vjag6ke6\HP.myHP.exe    N/A      |
|    0   N/A  N/A     14788    C+G   ...8bbwe\WindowsTerminal.exe    N/A      |
|    0   N/A  N/A     23312    C+G   ...n1h2txyewy\SearchHost.exe    N/A      |
+-----------------------------------------------------------------------------+
```

Figure 2: Checking GPU Specification

# 4 Data Exploration

In this section, the bounding boxes are converted from YOLO to COCO format as shown in Figure 7 The ‘convert_yolo_to_coco‘ function is designed to switch bounding box coordinates from YOLO style to COCO format. In YOLO, a bounding box is characterized by its midpoint (x, y) and size (w, h). This function transforms these YOLO coordinates into COCO format by considering the image width and height (set to 640x480). The result is a bounding box specified by its minimum and maximum x and y values. This conversion is handy for adapting bounding box information between different formats when working on object detection tasks.

# 5 Mounting Google Drive with Google Colab

After data exploration, a small section of dataset is taken and uplaaded in Google Drive so that to use that in Google colab and the directories are set as shown in Figure 7

# 6 Preparing data for training

The Trainer modules are set for setting up the training environment. The training parameters for this model are configured to ensure optimal performance and accuracy. Key settings include the use of the Adam optimizer with specified weight decay, a cosine learning rate schedule, and mixed precision training for improved efficiency. Exponential Moving Average (EMA) is employed to stabilize training, and specific metrics, such as mAP at different IoU thresholds, are monitored during validation. The loss function utilized is PPYoloELoss, tailored for object detection tasks, with considerations for class

```python
# Import Libraries

# Warning
import warnings
warnings.filterwarnings("ignore")

# System
import os
import gc
import shutil
import time
import glob

# Main
import random
import numpy as np
import pandas as pd
import json
import cv2
from tqdm import tqdm
tqdm.pandas()
from scipy.io import loadmat

# Data Visualization
import matplotlib
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import seaborn as sns
import plotly
import plotly.graph_objects as go
import plotly.express as px
import plotly.figure_factory as ff
from plotly.subplots import make_subplots
from IPython.display import Image, display, HTML
```

executed in 5.04s, finished 00:40:40 2023-12-12

Figure 3: Importing Libraries

count and regularization. The model is trained for a defined number of epochs, and the training process includes warm-up epochs to gradually adjust the learning rate as shown in figure 8. Overall, these configurations aim to facilitate effective training and evaluation of the model for object detection tasks.

# 7 Model Training

This research required multiple iterations of training our model using different and epochs as shown in figure 8, The model is trained for 4 set training epochs which 25,50,75 and 100. The training time for 25 epochs was half an hour and it took much time for running 100 epochs.

```python
# Generate Annotations
def convertBoxFormat(box):
    (box_x_left, box_y_top, box_w, box_h) = box
    (image_w, image_h) = (640, 480)
    dw = 1./image_w
    dh = 1./image_h
    x = (box_x_left + box_w / 2.0) * dw
    y = (box_y_top + box_h / 2.0) * dh
    w = box_w * dw
    h = box_h * dh
    return (x, y, w, h)

annotation_dir = 'C:/Users/lucks/RIC/caltech/annotations/annotations/*'
classes = ['person']
number_of_truth_boxes = 0

img_id_list = []
label_list = []
split_list = []
num_annot_list = []

# Sets
for sets in tqdm(sorted(glob.glob(annotation_dir))):
    set_id = os.path.basename(sets)
    set_number = int(set_id.replace('set', ''))
    split_dataset = "train" if set_number <=5 else "val"

    # Videos
    for vid_annotations in sorted(glob.glob(sets + "/*.vbb")):
        video_id = os.path.splitext(os.path.basename(vid_annotations))[0] # Video ID
        vbb = loadmat(vid_annotations) # Read VBB File
        obj_lists = vbb['A'][0][0][1][0] # Annotation List
        obj_lbl = [str(v[0]) for v in vbb['A'][0][0][4][0]] # Label List
```

Figure 4: Generating Annotation File

```python
# Generate Images from Video Files
def save_img(dir_path, fn, i, frame):
    cv2.imwrite('{}/{}_{}_{}.png'.format(
        dir_path, os.path.basename(dir_path), os.path.basename(fn).split('.')[0], f"{i:04d}"),
        frame)

def convert_caltech(split, df):
    # Directory Path
    print(split)
    input_dir = r'C:\Users\lucks\RIC\caltech'
    output_dir = 'datasets\images'
    if(split=="Train"):
        output_dir = os.path.join(output_dir, "train")
    else:
        output_dir = os.path.join(output_dir, "val")
    output_dir = os.path.join(output_dir, "caltechpedestriandataset")
    if(os.path.exists(output_dir)==False):
        os.mkdir(output_dir)


    # Sets
    sets_list = sorted(glob.glob(os.path.join(input_dir, split+"/*")))
    print("Total Sets:", len(sets_list))
    for dname in sets_list:
        dname2 = dname.split("\\")[-1]
        output_dir2 = os.path.join(output_dir, dname2)
        if(os.path.exists(output_dir2)==False):
            os.mkdir(output_dir2)
        df_filtered = df[df["set_id"]==dname2].reset_index(drop=True)

        # Videos
        videos_list = list(df_filtered["video_id"].unique())
        print("Total Videos:", len(videos_list))
        for i, vd in enumerate(videos_list):
            fn = os.path.join(dname, dname2, vd+".seq")
            print(fn)
```

Figure 5: Generating Image Files

```
convert_caltech("Train", df_train_filtered)
executed in 8m 42s, finished 00:50:09 2023-12-12
Train
Total Sets: 6
Total Videos: 12
C:\Users\lucks\RIC\caltech\Train\set00\set00\V001.seq
Total Frames: 299
C:\Users\lucks\RIC\caltech\Train\set00\set00\V002.seq
Total Frames: 226
C:\Users\lucks\RIC\caltech\Train\set00\set00\V004.seq
Total Frames: 76
C:\Users\lucks\RIC\caltech\Train\set00\set00\V006.seq
Total Frames: 634
C:\Users\lucks\RIC\caltech\Train\set00\set00\V007.seq
Total Frames: 494
C:\Users\lucks\RIC\caltech\Train\set00\set00\V008.seq
Total Frames: 425
C:\Users\lucks\RIC\caltech\Train\set00\set00\V009.seq
Total Frames: 526
C:\Users\lucks\RIC\caltech\Train\set00\set00\V010.seq
Total Frames: 517
```

Figure 6: Cutting Video into frames

```
from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive
```

Figure 7: Mounting Google Drive

```
train_params = {
    'silent_mode': False,
    "average_best_models":True,
    "warmup_mode": "linear_epoch_step",
    "warmup_initial_lr": 1e-6,
    "lr_warmup_epochs": 3,
    "initial_lr": 5e-4,
    "lr_mode": "cosine",
    "cosine_final_lr_ratio": 0.1,
    "optimizer": "Adam",
    "optimizer_params": {"weight_decay": 0.0001},
    "zero_weight_decay_on_bias_and_bn": True,
    "ema": True,
    "ema_params": {"decay": 0.9, "decay_type": "threshold"},
    "max_epochs": EPOCHS,
    "mixed_precision": True,
    "loss": PPYoloELoss(
        use_static_assigner=False,
        num_classes=len(dataset_params['classes']),
        reg_max=16
    ),
    "valid_metrics_list": [
        DetectionMetrics_050(
            score_thres=0.1,
            top_k_predictions=300,
            num_cls=len(dataset_params['classes']),
            normalize_targets=True,
            post_prediction_callback=PPYoloEPostPredictionCallback(
                score_threshold=0.01,
                nms_top_k=1000,
                max_predictions=300,
```

Figure 8: Preparing data for training

```
%cd /content/drive/MyDrive/train/ped_dataset

for model_to_train in models_to_train:
    trainer = Trainer(
        experiment_name=model_to_train,
        ckpt_root_dir=CHECKPOINT_DIR
    )

    model = models.get(
        model_to_train,
        num_classes=len(dataset_params['classes']),
        pretrained_weights="coco"
    )

    trainer.train(
        model=model,
        training_params=train_params,
        train_loader=train_data,
        valid_loader=val_data
    )
```

Figure 9: Model Training