National College of Ireland

Real-Time Pedestrian Detection using YOLO-NAS

MSc Research Project
MSc Data Analytics

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:	MSc Data Analytics
Year:	2023
Module:	MSc Research Project
Supervisor:	Christian Horn
Submission Due Date:	14/12/2023
Project Title:	Real-Time Pedestrian Detection using YOLO-NAS
Word Count:	5789
Page Count:	20

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:	31st January 2024

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):	

Real-Time Pedestrian Detection using YOLO-NAS

Lucky Shrivastava
x21198594

Abstract

This research delves into the dynamic field of object detection, particularly focusing on the crucial realm of pedestrian detection. Drawing upon advancements in machine learning and deep learning, this study explores the effectiveness of the YOLO-NAS technique—an efficient real-time detection method based on Single Stage Detection (SSD). The importance of pedestrian detection is underscored by the rising incidents of accidents and fatalities. The YOLO-NAS technique, a derivative of the YOLO framework, has proven successful across diverse applications, including object detection, security, monitoring, and safety. This research employs YOLO-NAS for pedestrian detection using a Kaggle-acquired dataset consisting of video sequences transformed into 640x640 pixel images with applied bounding boxes for accurate prediction. Four models, labeled Model 1 to Model 4, were trained for varying epochs. The third model achieved an outstanding mean Average Precision (mAP) of 0.67, outperforming the other models. Visualizations further enhance the understanding of the obtained result. The trained model can be further honed to improve its accuracy and can be employed in real-time environment for detecting pedestrians.

1 Introduction

Pedestrian detection has been a widely discussed and prominent subject in research within the field. As the science is progressing, there has been many machine learning and deep learning algorithms employed for development in this area. Pedestrian safety and timely detection of people has been a widely discussed topic from the last several decades. The statistics have shown increasing number of accidents and deaths in pedestrians while walking along the road, crossing the streets, and getting hit by a vehicle. One of the major reasons for these unfortunate incidents is not able to visually see and identify person or object within the time. This has been a major concern for the researchers to prevent the accidents. This has affected all over the world. The developed and developing nations have been facing these problems as there has been increase in number of vehicles which has been a prominent reason in pedestrian deaths. Every year around 1.35 million people die in road accidents as per statistics from World Health Organization(WHO). Most recently in 2023, the pedestrian fatalities observed an increasing curve from its previous years. In 2022, the developed nations like United Kingdom and United States reported a 13 to 15% rise in Pedestrian deaths. In 2021, 17% of the fatalities accounted for pedestrians which can be due to covid effect. Approximately 60,577 pedestrian suffered injuries in 2021, which is 11% more than from 54,771 reported in 2020. The WHO released a released a list of countries which has the most pedestrian fatalities. The list

analysed database of 183 countries and their territories, the Caribbean Island nation of Dominican Republic had the most death of 64.6% in 2019. Some other countries in the list include Zimbabwe, Venezuela, Saudi Arabia, Iran, Brazil etc. The main causes of pedestrian deaths include not knowing their right of way when walking along the road, improperly crossing a roadway or intersection, doing unnecessary activities like standing, lying, playing in the roadway, having poor visibility, being under the influence of drugs and alcohol. The implementation of different machine learning and deep learning algorithms can prevent these incidents.

(a) Undetected Pedestrians (b) Detected Pedestrians

Figure 1: Bounding Box visualization

Researchers have been using various methods for pedestrian detection. They have tried traditional and modern techniques for detecting and extracting pedestrians. The traditional approach was mainly based on hand-crafted methods for extracting features in an image. It requires human intervention, and the accuracy of the features depend on the human excellence. This method was more time consuming and less effective. The evolution of machine learning gave a new edge to this task of feature extraction as the humans now train the machine to detect and correct the pattern and machines will extract the useful features from the input image. This method became successful with the daytime pedestrian detection, but the nighttime pedestrian detection was still not resolved. Many researchers tried to solve the Pedestrian detection during the daytime and nighttime using different experimental setup. The nighttime pedestrian detection has more hindrance like low light, less visibility, noisy background, shadowing, blur etc in comparison to daytime pedestrian detection. These factors affect the model's performance and accuracy to detect the pedestrians.

Computer Vision-based Pedestrian Detection has exhibited significant advancements in the identification of objects and prevention of accidents. The primary aim of computer vision-based pedestrian detection, which endeavours to automatically discern pedestrians in images or video streams, is fundamental and pivotal. The principal objective is to devise algorithms and models that can discern individuals traversing through intricate and dynamic real-world scenarios with precision and efficacy. A multitude of industries can reap the benefits of this technology, encompassing autonomous vehicles, robots, surveillance, pedestrian safety, and urban planning.

The inherent diversity in the physical appearance, body postures, clothing choices, obstructions, and surroundings of individuals poses a formidable task in the field of pedestrian detection. The presence of other objects such as automobiles and bicycles make

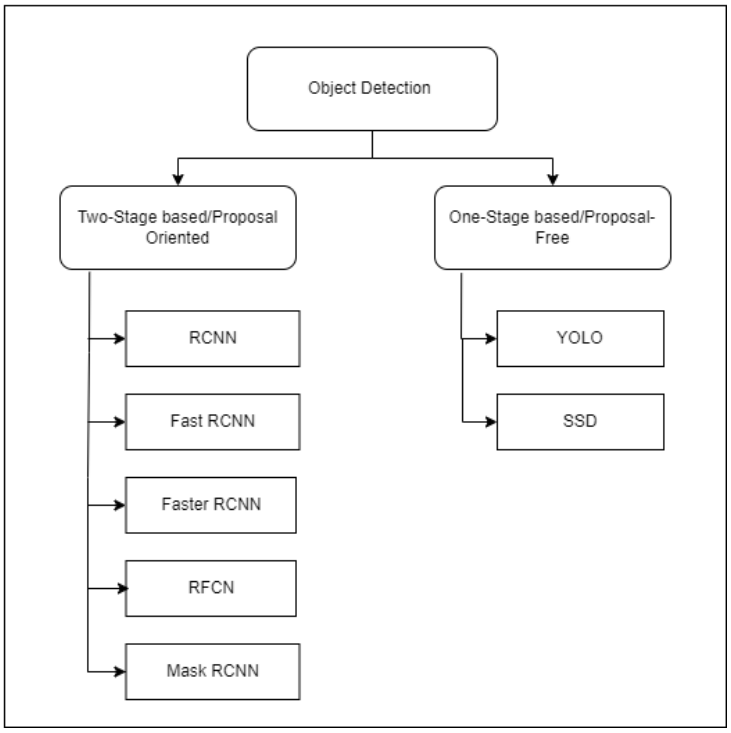


Figure 2: Object detection techniques

the task more difficult. Consequently, in order for algorithms designed for pedestrian recognition to be of practical value, they need to exhibit traits of dependability, expediency, and adaptability to various circumstances. This is where real-time pedestrian detection is very important and essential to prevent any accidents.

Research Question : Can YOLO-NAS be used for real-time pedestrian detection? YOLO Model has a faster processing time and it can process more than 1 frame per second. YOLO models are applied to enhance security systems, ensure quality control in manufacturing, manage traffic, aid wildlife conservation, assist in robotics, and improve drone technology. For security systems, YOLO models enable real-time detection of suspicious activities. In quality control, they identify defects in products during manufacturing. In traffic management, YOLO models recognize license plates and traffic signs, contributing to intelligent transportation systems. In wildlife conservation, these models track endangered species. Additionally, YOLO models guide robots and drones, enhancing their accuracy and efficiency. The YOLO models play a versatile role in addressing various real-world challenges. The goal of this research is to develop an effective real-time method for detecting pedestrians quickly in a single shot. YOLO, introduced in 2015, has consistently demonstrated excellent performance in object detection.. The latest it-

eration, YOLO-NAS (You Only Look Once – Neural Architecture Search), developed by Deci.ai, is deemed more versatile than other object detection techniques.

The rest of the paper has been divided into four sections, which are - Related Work, Methodology, Result and Evaluation, Conclusion and Future Work.

2 Related Work

Pedestrian detection has been a longstanding and debated topic for several decades. Researchers have sought to tackle this issue by introducing different models, employing both conventional and computational methods. the adoption of hand-crafted techniques is most popular, such as the sliding window approach, which is when applied across all potential positions and sizes, proved to be less effective due to its time-consuming nature. A study by Zhou and Yu (2021) introduced the Histogram of Oriented Gradients (HoG) method to enhance the detection of pedestrians, using downstream classifiers such as Support Vector Machine (SVM) and Adaboost. Although the aforementioned models primarily focused on daytime detection, Pedestrian Detection can be further categorized into daytime and nighttime detection. The daytime Pedestrian Detection models have exhibited high accuracy owing to their training with a substantial volume of images. They have used the state-of-art Caltech Dataset, which is also used in this research. This is a diverse dataset which contains over 250,000 images of pedestrians taken in different backgrounds. Most of the images are in daytime for clear images differentiating person and shadows. This method was not able to handle occlusion problem and also it was taking too much time for predicting the result. To solve this occlusion problem, an Integrated Channel Features were introduced for pedestrian detection. This technique was performing satisfactorily but it was computationally expensive to implement the hardware setup. Also, sometimes it was not able to adjust the scale and pose of the pedestrians which is not suitable for correct detection of pedestrians.

2.1 YOLO models

Lan et al. (2018) utilizes advanced learning methods to improve the network structure of YOLOv2, leading to the development of the YOLO-R model, which has demonstrated effectiveness in pedestrian detection. The approach involves incorporating three Passthrough layers into the YOLOv2 network to capture shallow layer pedestrian features. Furthermore, enhancements are made to the original algorithm's Route layer, shifting the improvement from the 16th layer to the 12th layer, and combining these refined shallow layer features with deep layer features to extract more detailed information. The experimental outcomes demonstrate that this approach significantly improves the accuracy of detecting pedestrians, achieving a frame rate of 25 frames per second and meeting real-time performance requirements. Looking ahead, the plan is to integrate additional pedestrian context features to further enhance the precision of pedestrian detection in future iterations of the model.

Farooq et al. (2023) proposed a model based on the YOLOv3 D-NNN architecture. They achieved a 6.9% improvement in pedestrian detection accuracy compared to the base model. Moreover, they managed to reduce processing time for optimization by 22.76%, only sacrificing 2% of detection accuracy Nevertheless, it's important to consider that this trade-off could restrict the model's applicability in specific situations.

Boyuan and Muqing (2020) tried to detect the pedestrian in real-time by combining YOLOv4 and k-means clustering by modifying the architecture of YOLOv4. A large number of images are fed as an input to this improved model structure. Anchor parameters were optimised by a new SPP network was applied to neck of the model. In activation function, LeakyReLu was replaced by Mish activation function. However, the real-time speed of this algorithm was not fast and needed more improvement.

Sukkar et al. (2021) offered an overview of frameworks based on deep learning, focusing on addressing detection and tracking challenges. They specifically delved into pedestrian detection and tracking, exploring their relevance to real-time systems and potential additional solutions. Their approach included refining pedestrian detection within various subcategories. They assessed various YOLOvv5 models to find the most fitting one for our study. YOLOvv5 was examined as a detection algorithm, and we used data augmentation to improve detection performance, resulting in enhancements in both mean Average Precision (mAP) and Recall metrics Despite these positive outcomes, limitations exist, particularly in the annotated data, which we plan to address by increasing the diversity of real-life data. The structure of the proposed algorithm was complex and need further improvement for large datasets.

Jönsson Hyberg and Sjöberg (2023) The study YOLOv8 for object detection in self-driving cars. This studies humans in traffic images using YOLOv8 and checks the precision using the same model. The YOLOv8 model, post-training, achieved a mean Average Precision (mAP50) of 0.874, running at a speed of 66.7 frames per second for pedestrian detection in traffic. The TBM exhibited a 9.3% improvement in accuracy compared to YOLOv8's BM, with only a slight 0.1 ms speed increase over the original model BM. While these results don't definitively determine YOLOv8's suitability for autonomous cars, they underscore its faster performance compared to certain object detection models and even humans. The speed of the YOLOv8 detector could be crucial in traffic scenarios, suggesting its potential importance in the advancement of autonomous cars.

Terven et al. (2023) gave a through study of different versions of YOLO, starting from YOLOv1 to YOLO with transformers, which is currently the latest version of YOLO. This study meticulously explored 16 iterations of YOLO, spanning from its foundational model to the cutting-edge YOLO-NAS. The comprehensive overview provided revealed distinct trends and developments: Anchors: The original YOLO lacked anchors, but as newer versions, particularly two-stage detectors, embraced them, YOLOv2 introduced anchors, significantly improving bounding-box prediction accuracy. An anchorless approach, pioneered by YOLOX, became a defining shift later adopted by successive versions. Framework: YOLO initially relied on the Darknet framework; a pattern maintained by subsequent versions. However, the transition of YOLOv3 to PyTorch by Ultralytics led to a widespread adoption of PyTorch in subsequent iterations. Additionally, the open-source framework PaddlePaddle, developed by Baidu, found utility in certain versions. Backbone: The evolution of YOLO's backbone architectures was marked by significant transformations. Starting with the straightforward Darknet architecture, later models incorporated innovations such as cross-stage partial connections (CSP) in YOLOv4, re-parameterization in YOLOv6 and YOLOv7, and neural architecture search in DAMO-YOLO and YOLO-NAS. Performance: Notably, YOLO models have achieved improved performance over time. However, a key observation is their emphasis on striking a balance between speed and accuracy, rather than singularly prioritizing accuracy. This intentional tradeoff is integral to the YOLO framework, enabling real-time object detection across a diverse array of applications.

Considering the reviewed research methods, the proposed model will adopt a single-stage approach, specifically YOLO-NAS, known for its robustness and adaptability to learning patterns, resulting in reduced training time. To make sure it works well in both day and night, the model will be trained using Caltech dataset which has diverse images.

Version	Year	Description
YOLOv1	2015	The first version used a single neural network for object detection, providing fast but less accurate results.
YOLOv2	2016	Addressed YOLOv1 limitations with a deeper neural network, batch normalization, and anchor boxes for improved accuracy.
YOLOv3	2018	Improved accuracy further with a more complex architecture, feature pyramid networks, and multi-scale predictions.
YOLOv4	2020	Introduced a more complex architecture, weighted feature fusion, spatial attention, and cross-stage partial connections for improved accuracy and speed.
YOLOv5	2020	Unofficial version with differences in architecture and implementation compared to YOLOv4.
YOLOv6	2021	Another unofficial version based on YOLOv5, introducing new techniques like cross-stage partial connections and hard negative example mining.
YOLOv7	2022	Claimed to be the fastest and most accurate real-time object detector.
YOLOv8	2023	Considered the best YOLO model to date, incorporating new features and improvements for enhanced performance and flexibility.

Table 1: Uma et al. (2023) presented Evolution of YOLO Object Detection Versions

2.2 Non YOLO models

Deep learning, especially Convolutional neural networks (CNNs), is extensively employed for recognizing images and objects because of its quick processing and versatility. Zhao et al. (2019) CNNs have proven to be successful in spotting objects, recognizing faces, and detecting pedestrians. A particular CNN design called Faster R-CNN was created specifically for object detection. However, it faced challenges related to imbalances in class distribution, impacting its overall precision. Wang et al. (2020) To address this issue, RPN+BF (Region Proposal Network + Bounding Box Regression and Classification) was developed, which improved pedestrian detection accuracy by rectifying the class imbalance problem. RPN+BF tackles the computational inefficiency of the sliding window approach by automatically generating proposals for potential object locations. It employs Bounding Box regression and local contrast segmentation for object detection. Bounding Box regression detects general objects, while local contrast segmentation identifies salient objects. Pedestrian detection is achieved using a multi-feature boosting forest. The study compared the proposed RPN+BF method with other models, including multibox, Attenuation Net, G-CNN, and YOLO. It concluded that CNN-based methods generally provide more accurate results than traditional hand-crafted methods. However, designing the framework and classifiers, as well as extracting part-based semantic information, can be expensive for CNN-based approaches.

Another proposed method, developed by Kilicarslan et al. (2016) aimed to address the problem of pedestrian detection in a more cost-effective manner by analyzing pedestrian walking patterns. This method involves placing a high-resolution camera at the rear of

a vehicle to capture images of pedestrians parallel to the driving view. By dividing the focus of expansion into four parts, the system creates a chain of patterns that enables it to study the average height of pedestrians, leg motion, and body motion. The goal is to identify the distinctive X-pattern that the human body forms while walking. This method demonstrates robustness in detecting pedestrians amidst crowds and dynamic scenes. It specifically focuses on identifying short leg traces rather than long, smooth object trajectories by employing a cascading filter. However, its effectiveness is limited in low-lighting conditions, leading to inaccurate results.

The challenge of pedestrian detection during nighttime remained unresolved as previous methods were ineffective in low-lighting conditions. A recent model introduced by Kulhandjian et al. (2023) showcases efficient pedestrian detection capabilities in both daytime and nighttime conditions In their experiments, they set up multiple sensors on the car's dashboard. These include a regular camera, an infrared camera, and a radar sensor. The visual camera handles pedestrian detection in daylight, and the infrared camera takes charge during nighttime scenarios. Alongside these cameras, the radar sensor contributes crucial data regarding pedestrian presence, range, and motion direction. The suggested multi-sensor data analysis model attains an average accuracy of 98%. The model's training involved collecting 1200 images, evenly split between those with pedestrians and those without, and testing was conducted using 800 images from the FLIR dataset. Micro-doppler radar technology was utilized for data collection. Potential enhancements include incorporating a higher-range detection camera for extended coverage and applying the system to nighttime surveillance scenarios, addressing challenges related to distinguishing between humans and shadows from a downward perspective.

Neumann et al. (2019) conducted a study to assess the effectiveness of using daytime datasets, like Caltech, to train models for nighttime pedestrian detection. They compared the performance of pedestrian detection systems trained on popular datasets, including Caltech and KITTI, under both daytime and nighttime conditions. Recognizing the limitations of existing datasets, they introduced the Nightowls dataset, specifically designed to capture nighttime pedestrian data in various challenging scenarios. These scenarios include occlusions, diverse object poses, and multiple frames. The comparison of several models revealed that current pedestrian datasets struggle with nighttime detection challenges, indicating the necessity for dedicated datasets addressing low illumination, contrast variations, and limited color information. Another study delved deeper into this issue, comparing Caltech, KITTI, and Nightowls datasets. Emphasizing the difference between surveillance and autonomous vehicle cameras, the study considered factors like viewpoint, illumination imbalances, object scales, lightness, occlusions, rain, and blur. Their findings reinforced the conclusion that there is a shortage of comprehensive nighttime surveillance datasets.

Dai et al. (2021) developed a new approach for detecting pedestrians and estimating their distance at night. Their system combines Faster R-CNN and ResNet-50 algorithms to identify and locate pedestrians, while also incorporating distance estimation capabilities. The system utilizes a Near Infrared (NIR) sensor that encompasses SD-NIR and LD-NIR elements. This sensor is positioned on the roof of a vehicle.. To train and test their model, the team employed data from the Caltech, Nightowls, and NIR Image datasets. They gathered real-world data using an NIR camera, two fill light devices, and a LiDAR system. The dataset consisted of 84,504 high-quality images, including 19,343 pedestrian images with distances ranging from 3 to 102 meters. The R-CNN and ResNet-50 architecture were used for detection without requiring extra training or testing time.

The model achieved a pedestrian detection accuracy of 79.45%, a miss rate of 19.23%, and a total absolute error rate of 4.66%. The speed of detection was low and was more time consuming which makes it less suitable for real-time implementation.

Building upon previous work in surveillance detection, Chen and Shin (2020) explored the use of thermal imaging and convolutional neural networks (CNNs) to develop a nighttime dataset and evaluate their proposed technique against existing methods. By extending Faster R-CNN, they demonstrated its applicability in static surveillance scenarios. Utilizing a thermal camera, they compiled a nighttime dataset and contrasted it with the FLIR and KITTI datasets. Their proposed nighttime thermal-based R-CNN model employed masking to address occlusion and background segmentation issues. Furthermore, they introduced a novel loss function to facilitate joint model training. The model comprises three key branches: (1) a part model branch that captures partial pedestrian block features, (2) a segmentation branch that enhances pedestrian foreground positioning, and (3) a fusion loss function that integrates these branches for joint training and optimization. However, its applicability remains limited to outdoor scenarios.

Another study by Chen et al. (2020) focused on pedestrian detection using infrared images at night, employing an attention-guided encoder-decoder CNN architecture. They leveraged CNNs to detect pedestrians during nighttime, addressing the limitations of visual cameras in capturing information under low-light conditions. Instead, they utilized an infrared camera, incorporating an encoder-decoder system coupled with an attention module to eliminate background noise generated by the infrared camera while capturing multi-scale features. Additionally, they employed skip connections within the encoder-decoder system. This mechanism reweighs the information to ensure that no crucial details are lost during processing. Their model outperformed existing methods, achieving 5.1% and 23.09% higher accuracy on the KMU and CVC-09 datasets, respectively. Saliency detection played a crucial role in reducing background complexity. Further improvements could be achieved by refining the loss function to enhance accuracy and address false detections. The proposed model demonstrated impressive accuracy, reaching 97.50% on the KMU dataset and 87.68% on the CVC-09 dataset. Future enhancements could involve integrating low-resolution images and addressing persisting occlusion challenges.

3 Methodology

In this research, CRISP-DM is used for Exploring the data, processing the data, making the model and evaluating the model.

3.1 Dataset Description

The dataset used in this research is the state-of-art Caltech pedestrian Dataset. This dataset is openly accessible in Kaggle[1] and comprises over 250,000 images capturing pedestrians in daytime conditions. The dataset ensures favorable lighting conditions, making pedestrians clearly visible with distinguishable shadows. The size of this dataset is 12GB and it contains video sequences of people taken from car's dash camera. There are 11 set of video sequences, the duration of which is 13 minutes each approximately. The sets are divided into training, validation and testing. The set 0 to set 5 are used for

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

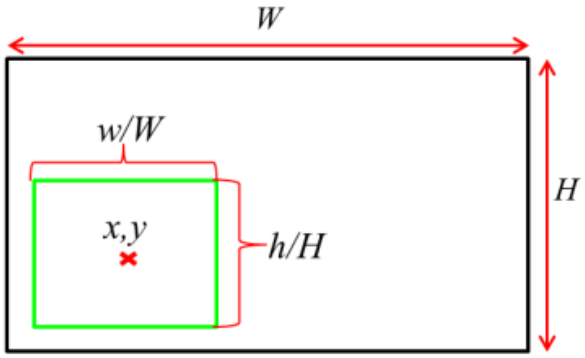


Figure 3: Salim (2017) Bounding Box labelling

training and the set 6 to set 10 are used for validation and testing. The video sequences are converted into frames of images to enable image-based processing and analysis. Each frame represents a snapshot of the video at a specific moment in time. Most of images are taken in highway. These images contains still person, moving person, occlusion, mixed images of persons crossing over the street in traffic signals etc which makes it a more versatile dataset for training the model. This contains small as well as large objects in the images and the model can be made more robust if there are more than one classes for training.

3.2 Data Pre-processing

In the data pre-processing step, the recommended github file of Zachau (2018) is followed and modified according to the requirement for this research. The size of dataset is checked and ensure that it has good amount of images for training and testing along with the directories are setup for setting up the image path. The raw Caltech dataset contains video sequences that are (.seq) format that are splitted and converted into frames of images of size 640*480, it can also be modified to 640*640 using a squared function which works better for detection. The format of this converted images is (.png). The total number of images in this dataset 12,500 approximately, in which 10,000 are training images and the rest 2,500 are validation and testing images. After splitting the video sequences into images, the annotations are generated for the images which are stored in the labels folder. The annotation file is provided with the dataset. The file format of raw annotation file is (.vbb) format which are converted into (.txt) which is compatible for indexing annotations with the images. The labels folder is matched with their proper image sequences. The annotations contain the information about the bounding box and ground truth which is required. After the conversion to (.txt) file, the bounding box annotation should contain Class id, x, y, Width and Height for every image. The Class denotes the id of the class to be detected. The x and y represent the centre of the bounding box. The Width and Height represent the width and height of the bounding box with respect to the original image. The annotated images can now be adjusted to YOLO format for further exploration with the height=640 and width=640, and the number of classes are 2 with the filters in final layer is 35, calculated as (classes + 5) * 5, to suit the YOLO architecture for subsequent exploration.

The dataset is further modified to make the images compatible from YOLO to COCO (Common Objects in Context) format. The Microsoft COCO dataset *COCO - Common Objects in Context* (n.d.) is a diverse dataset for using it as a base model. It has

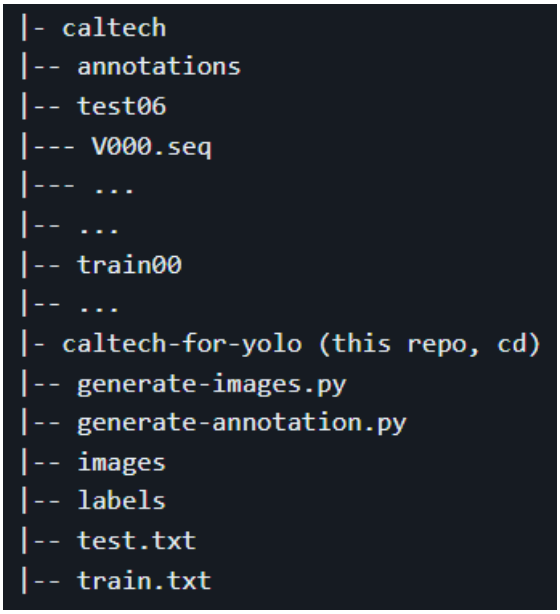


Figure 4: folder structure of converted file format

328,000 images and more than 80 category of classes with annotations, covering wide range of real-world images in indoor, outdoor and natural environment. The images are converted to COCO data format. The convert_yolo_to_coco function serves as a utility for translating bounding boxes from the YOLO format to the COCO format. These formats are distinct representations of rectangular bounding boxes, commonly used to identify and locate objects within images or videos. The people distribution per frame is plotted and anlaysed to visually see the ratio of males and females in the dataset.

3.3 YOLO-NAS Architecture

YOLO is a renowned object detection method, is recognized for its groundbreaking approach utilizing a single CNN for image processing. Its notable features include high speed and precision, rendering it valuable for real-time applications like autonomous driving, surveillance systems, and robotics. The YOLO-NAS (You Only Look One-level Neural Architecture Search) model architecture is composed of three key components: the backbone, the neck, and the head. The functioning of these components makes the model working.

Backbone: The backbone is the first step of the YOLO-NAS, which extracts the data from the input image and then that information is passed on to the next layers. It consists of layers like Convolution and Pooling layers, specifically engineered to capture a wide spectrum of features from the image. These features range from basic edges to more complex objects present in the image. This hierarchical representation facilitates the system's understanding of the visual structure of the image, thereby enhancing object recognition. YOLO-NAS employs innovative approaches to design the backbone, often integrating techniques like neural architecture search (NAS) to automatically identify efficient architectures. In YOLO-NAS, the backbone has CSPNet architecture, which introduces Depthwise Separable Convolutions (DSC) for increasing the model's efficiency and reducing parameters. Additionally, Channel Shuffle is employed to optimize information transmission, while Residual Connections are utilized to safeguard spatial information

within the input. This combination of techniques ensures that the backbone can effectively extract comprehensive and informative features from the image, thereby enabling accurate object detection by the model.

Neck: The neck is the middle part of the neural network that connects the backbone to the head. It helps refine the features extracted by the backbone, making them more precise and unified. This is an important step which helps the model to better distinguish between objects and analyze their locations more accurately. YOLO-NAS can use innovative designs or specific operations for the neck to optimize the refinement process. Sometimes, NAS is also employed to find the best configurations for the neck. The neck is like a polisher that refines and enhances the features extracted by the backbone, making them more valuable for object detection.

Head: The head is like the last step in the neural network journey, and its job is to make predictions. It has a bunch of fully connected layers and output layers that spit out the final results of the detection. Taking the polished features from the neck, the head predicts things like the boxes around objects, what type of objects they are, and how confident the model is about these predictions. In YOLO-NAS, there might be some tinkering with different designs or setups for the head to make sure the detection results are top-notch. This could mean trying out different combinations of layers and operations to find what works best. The YOLO-NAS model architecture can seen in figure 5[2]

YOLO-NAS introduces several innovative features, including quantization-aware modules (QSP and QCI) that employ re-parameterization for 8-bit quantization, minimizing accuracy loss during post-training quantization. The model's architecture is automatically designed using AutoNAC, Deci's proprietary NAS technology.

A hybrid quantization method is employed, allowing selective quantization of specific model parts to balance latency and accuracy, deviating from standard quantization where all layers are affected. YOLO-NAS follows a pre-training regimen involving automatically labeled data, self-distillation, and large datasets.

The versatile AutoNAC system, crucial in YOLO-NAS creation, accommodates various tasks, data specifics, inference environments, and performance goals. It assists users in identifying an optimal structure, balancing precision and inference speed for their specific needs. This technology considers data, hardware, and other inference process elements, such as compilers and quantization.

Additionally, RepVGG blocks are integrated into the model architecture during the NAS process for compatibility with post-training quantization (PTQ). Three architectures —YOLO-NASS, YOLO-NASM, and YOLO-NASL (representing small, medium, and large variations)—are generated by varying the depth and positions of QSP and QCI blocks.

3.4 Model Building

The models are build in Google colab beacuase of the hardware dependencies of supergradients, which is an essential library for YOLO-NAS. The dataset was split in ratio of 80 percent for training, 10 percent for validation, 10 percent for testing. A small portion of the dataset has been taken because of hardware constraints. The training is performed using the trainer module with a Batch size of 16 bits and the workers are set to 8. Dataloaders function is used for loading and training and validation data. The number of Epochs are set for 25, 50, 75 and 100. To make the result more interactive a

[2]https://jyothish-tech.medium.com/yolo-nas-sota-in-object-detection-5279eb4863b2

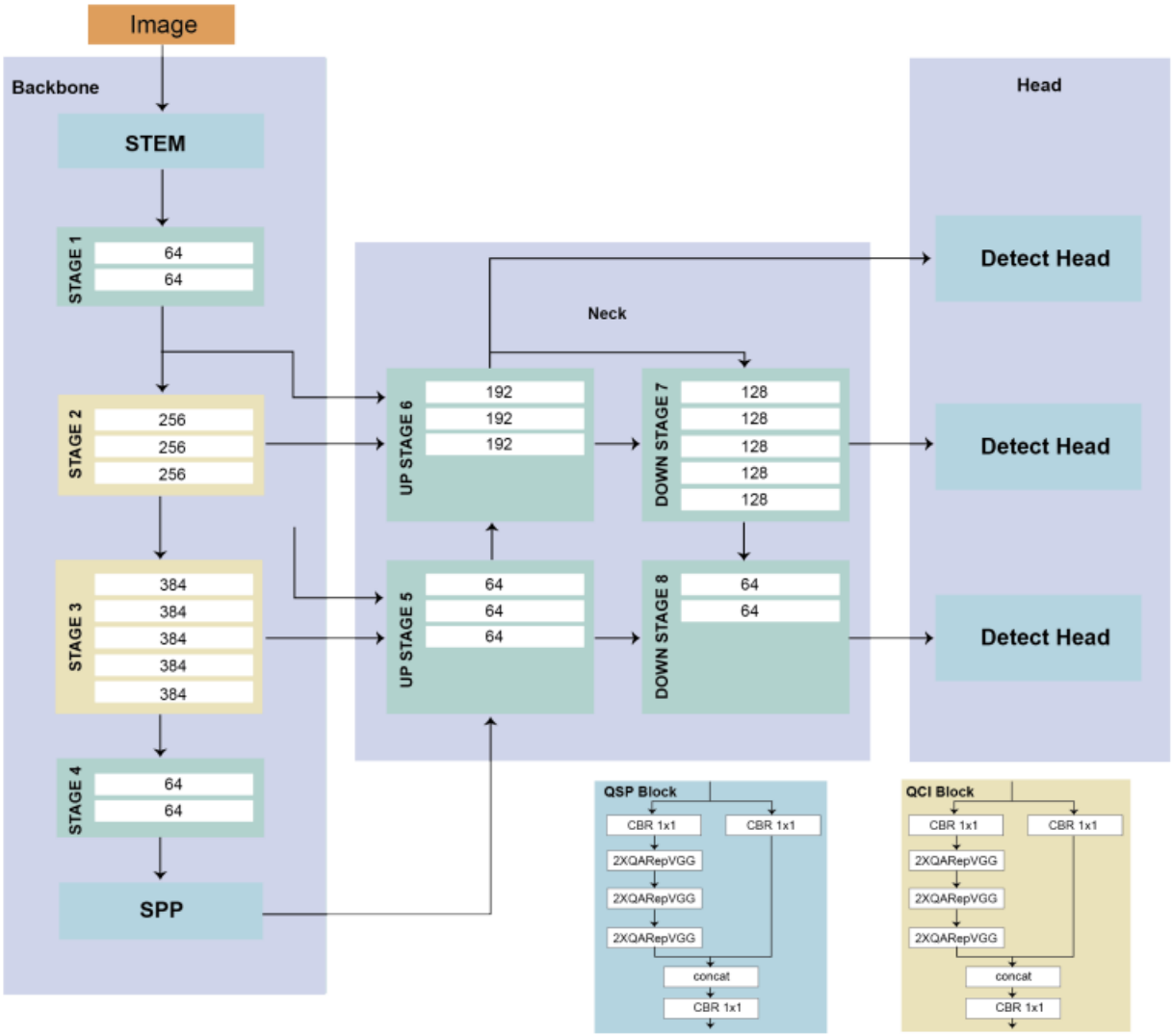


Figure 5: YOLO-NAS Architecture

different colour for bounding boxes are set for every iterations. The Bounding boxes are made according to image height and width. The squared images are easier for calculating bounding boxes so the images are taken in a squared format. The training and validation data are set to the COCO detection format and the number of classes are set to 2, for detecting pedestrians and don't care. The Data Augmentation is not needed as it is a frames of video sequences so the data augmentation will over train the model. The Learning rate is set to 0.1 for better training. The Adam Optimizer is used for predicting the result and the loss function is closely calculated for giving more accurate result. The results are stored in the google drive. Due to hardware incompetency, YOLO-NAS S model has been used in this research. The model has three blocks, Backbone, Neck and Head. The images are input to the backbone of the model which has multiple layers which extracts information and features of the images then this information is passed to the Neck where it is further filtered and more accurate features are extracted and at last the information is passed through a optimizer which detects what should be displayed in the output of the image.

4 Result and Evaluation

During the training phase for Pedestrian detection, the proposed models underwent training on a computer equipped with an 11th Gen Intel(R) Core(TM) i7 – 11370H CPU @ 3.30 GHz x 8, 16 GB RAM, and ran on a 64-bit Windows 11 Home operating system. The training process was supported by an NVIDIA GeForce GTX 1650 GPU. The training outcomes for Models 1, 2, 3, and 4 are depicted in Table 2. The figure displays metrics such as Precision, Recall, mAP, F1 score, and different average loss values throughout the training process. Notably, the average loss curves for all four models generally trended downward, despite some fluctuations during training. Model 3, among the trained models, exhibited the lowest overall average loss, indicating that the network model incorporating the CSPNet architecture and AutoNAC optimization yields more accurate results for human detection compared to models without these features.

Models	Epochs	Precision	Recall	mAP	F1-Score	IOU loss	DFL loss	Total loss
Model 1	25	0.03	0.83	0.64	0.05	0.52	0.46	2.12
Model 2	50	0.048	0.91	0.66	0.09	0.5	0.46	2.01
Model 3	75	0.03	0.95	0.67	0.06	0.38	0.39	1.77
Model 4	100	0.03	1	0.66	0.07	0.47	0.43	1.9

Table 2: Model Evaluation Metrics

To assess the detection performance of the trained models, the experiment utilized the commonly employed Average Precision (AP) metric. AP summarizes the precision/recall curve's shape and represents the mean precision at recall levels ranging from 0 to 1. The models are numbered from 1 to 4 which represents 25 epochs, 50, epochs, 75 epochs and 100 epochs respectively as shown in Table 2. The testing results of the four trained models on the Caltech dataset validation set are presented. The results are calculated at a confidence threshold of 0.50 and based on these factors, Precision@0.50, Recall@0.50, mAP@0.50 (mean average precision), F1 score@0.50 and different loss functions on which the accuracy of the model depends.

4.1 IOU (Intersection over Union)

The Intersection over Union (IoU) is the parameter that measures the extent to which a predicted bounding box aligns with the ground truth bounding box. It is an important parameter which measures the accuracy of object localization. It's like comparing where we guessed something is with where it actually is. We calculate it by dividing the shared area by the combined area of both boxes. In simple terms, IoU tells us how accurately our guess matches the true location of an object. The result ranges between number 0 (no overlap) and 1 (perfect match). IoU is important for judging how well computer vision systems find objects, showing how accurate and reliable they are. IoU is maximum for the first model as shown in Table 2.

$$\text{IoU} = \frac{Areaofoverlap}{AreaofUnion} \tag{1}$$

4.2 Speed/Inference Time

The YOLO-NAS models are popular because it gives result in less amount of time. The YOLO-NAS S model gives good accuracy with limited computational resources and its speed is fast, Where as YOLO-NAS M model gives balanced model which gives higher accuracy, The YOLO-NAS L model is suitable for large scale detection and where there less constraint for computational resources. Its training time is also slow because of the large model. The inference time of the third model is highest and it can detect more 25 frames per second. It takes less a mintue for detecting 100 test images. It is therefore suitable for implementing in real-time applications.

4.3 mAP (Mean Average Precision)

The acronym mAP, which refers to Mean Average Precision, holds utmost significance in the evaluation of model performance, particularly in the domain of object detection where YOLO-based models find frequent usage It acts as a comprehensive metric encompassing multiple classes, thereby providing valuable insights into the accuracy of the system as well as the instances where specific categories were missed. Average Precision (AP), a constituent of mAP, assesses the system's precision and its capability to avoid overlooking pertinent objects within a given category. The evaluation procedure involves constructing a curve for each category to visually represent the system's performance, followed by the determination of the area beneath said curve.

The mAP values for the four models are, 0.64, 0.66, 0.67, and 0.66 respectively as shown in Table 2. Model 3, which exhibits the highest mAP value of 0.67, emerges as the most suitable model for real-time pedestrian detection among the assessed alternatives. It effectively establishes the bounding boxes for a majority of the pedestrians. The mAP metric furnishes a comprehensive gauge of the model's overall performance across diverse categories, presenting a singular numerical representation that encapsulates the average precision attained throughout the entirety of the object classes.

4.4 F1 score

The F1-score is a measure which is mostly used in binary classification. It is used to evaluate the performance of neural networks which have 2 output like in this research in object detection, perform. It's determined as the harmonic mean of precision and recall. In simpler terms, the F1-score is like a well-rounded grade for a model. It considers how accurate the model is when it says something positive (precision) and how good it is at finding all the positive things (recall). This metric proves helpful in evaluating how well a model performs in tasks requiring decision-making between two alternatives. F1-score is calculated as follows:

$$\mathrm{F1} = \frac{(2 * Precision * Recall)}{(Precision + Recall)} \tag{2}$$

In this research, the F1 score of the model 1 to 4 are 0.05, 0.09, 0.06 and 0.07 respectively as shown in Table 2. The Model 2 with 50 epochs achieves the highest F1 score. A high F1-score indicates that the model is doing well in both precision and recall. A low F1-score indicates that the model is not performing well and needs more improvement in precision and recall.

F1-score is almost similar to mAP in that it considers both precision and recall. However, F1-score is a bit different and more complex metric than mAP because it calculates precision and recall differently. This can be helpful in some cases when the high precision and recall are required.

4.5 Different Losses

In this research, the YOLO-NAS model tries to minimize the differenet losses. It keeps a check on the loss function and adjust its layers in order to minimalise the loss. The losses are due to intersection over union loss, Distribution Focal loss and the Aggregated total loss.

Intersection over Union loss: It is a measure that calculates how much the predicted bounding box overlaps with the actual ground truth bounding box. A higher IoU generally indicates a better localization of the object. The IoU losses for the four models are 0.52, 0.50, 0.38 and 0.47 as shown in Table 2. Model 2 has the highest IoU loss which makes it more stable as compare to other models.

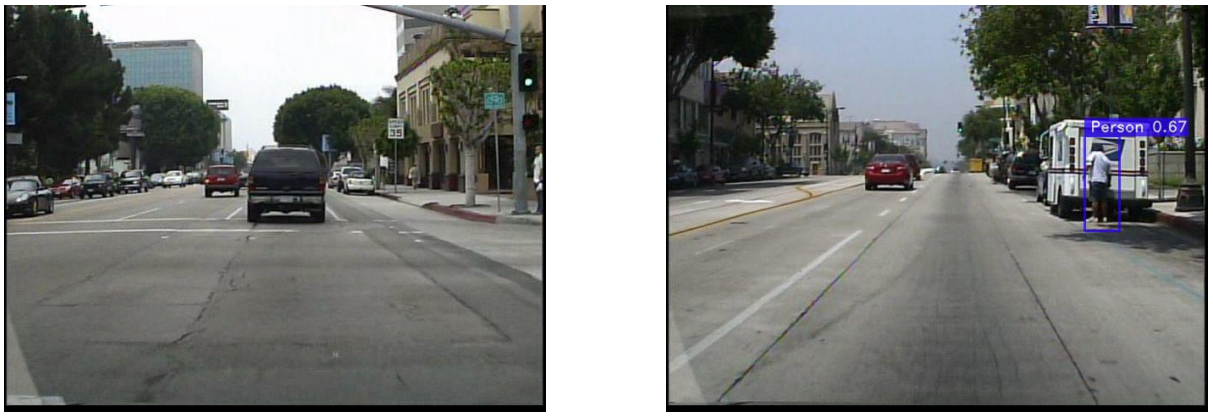
Distribution Focal loss: The DFL is a loss function that is frequently employed in the context of object detection tasks. It is mostly used to remove the class imbalance issue that are common within datasets. Class imbalance occurs when there is a significant difference in the number of instances belonging to various classes This discrepancy can result in the model prioritizing the class with a higher abundance. The DFL losses for the more models are 0.46, 0.46, 0.39 and 0.43 as shown in Table 2. The lower the the value of DFL loss, the better it is for the performance of the model. DFL aims to give more emphasis to hard-to-detect objects, typically the minority class, by modifying the standard focal loss. This adjustment helps the model focus on learning from challenging examples, improving its ability to detect and classify less common objects.

Aggregated loss: The Aggregated loss or the total loss is a comprehensive metric that encompasses various individual loss components in training a object detection model. The total loss comprises of IoU loss and DFL losses. The aim is keep the total loss as low as possible. If the total loss value is high that indicates that the model is not performing well on the training data. The total loss acquired by the four models are 2.12, 2.01, 1.77 and 1.90 as shown in Table 2. This makes the third model with 75 epochs most suitable for real-time Pedestrian detection.

4.6 Comparison with YOLOv8 model

YOLO-NAS has also shown impressive performance on the Caltech dataset. It obtained a mean average precision (mAP) of 0.67 with an intersection-over-union (IoU) threshold of 0.5. This performance is similar to YOLOv8 and notably better than the performance of earlier YOLO models on the same dataset. The YOLOv8 model secured a mean average precision (mAP) of 0.72 on the same Caltech dataset, which is slightly higher than YOLO-NAS. This improvement can be attributed to YOLOv8's more efficient backbone network and enhanced feature extraction capabilities. While YOLOv8 has a fixed model architecture, YOLO-NAS utilizes Neural Architecture Search (NAS) to optimize its architecture dynamically, enhancing its suitability and robustness for object detection and prediction tasks. Regarding speed, YOLO-NAS achieves a faster processing rate of 2 frames per second compared to YOLOv8, thanks to its optimized layers. The YOLO-NAS design requires greater training time for the model due to its complexity compared to the

YOLOv8 architecture.. The NAS algorithm explores various architectures before selecting the most appropriate one. Past performance indicates that YOLO-NAS consistently achieves high mAPs across different Intersection over Union (IoU) thresholds, showcasing its robustness in handling variations in object size and shape. This adaptability positions YOLO-NAS as an excellent choice for applications such as object detection, autonomous vehicles, and security systems.



(a) No Pedestrians (b) Detected Pedestrians

Figure 6: Inference results

4.7 Visualization

Analysis of People Distribution: The plot gives a clear and informative visual representation of how many annotations exist per frame in both the training and validation datasets. The figure employs vertical bar charts, using a distinct maroon color for the training subplot and a blue color for the validation subplot. To enhance readability, the plot customizes various elements, such as the axes' appearance, line colors, ticks, labels, and the font style for the title. Ensuring transparency and ease of sharing, the finalized figure is saved as an image file. It effectively shows how annotations are distributed across the training and validation datasets, offering valuable insights into the data distribution and potentially revealing any biases present in the dataset.

Area Plot : In the Area plot, there are two subplots for training and for validation. Each subplot displays a density distribution plot to illustrate the distribution of the area of annotations in the training and validation data. The plot is made of different colors for representing the training and validation subplots differently and saves the figure as an image file.

Epochs vs mAP : The line graph effectively depicts the association between mAP (Mean Average Precision) and epochs. The third model stands out with its exceptional mAP of 0.67, showcasing its superior performance. The graph shows plotted values for epochs and mAP. To provide a comprehensive overview, only the top 10 mAP and epoch values are considered for the visualization.

The Matplotlib library is used for making the visualization. The epoch list holds the number of training iterations, while the mAP list contains the corresponding mAP values for the top 10 epochs. This combination of different elements allows the graph to develop the relationship between mAP and epochs.

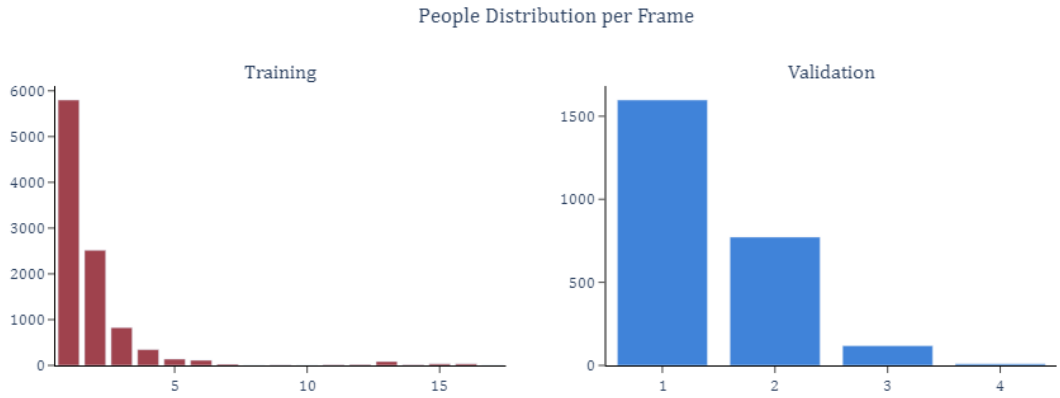


Figure 7: People Distribution per frame

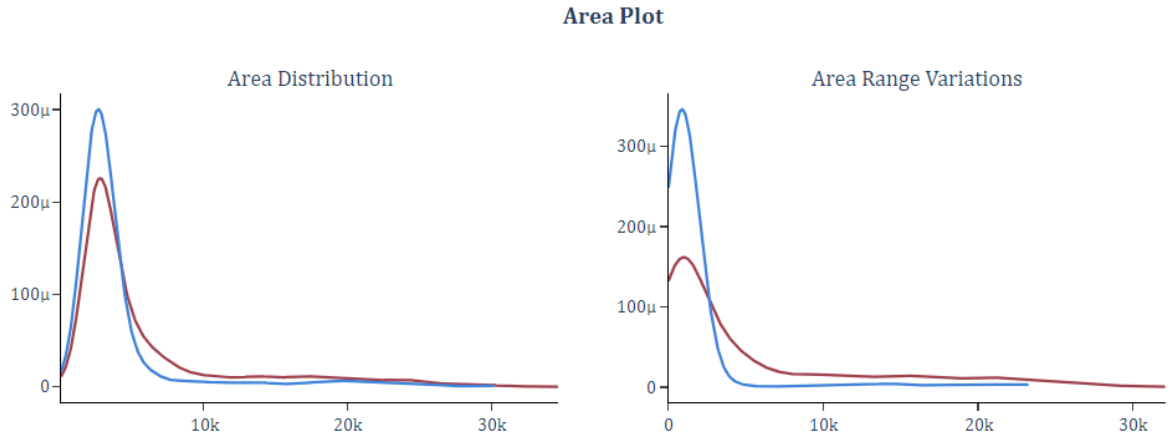


Figure 8: Area Plot

4.8 Tradeoff between Speed and Accuracy

When comparing YOLO-NAS and YOLOv8, finding the right balance between speed and accuracy is very important. YOLO-NAS places a high priority on real-time performance without compromising accuracy. It achieves this by using quantization-friendly building blocks, advanced training techniques, and post-training quantization. However, YOLOv8, created for real-time use, might have to make compromises between how quickly it operates and how accurate its detections are. It aims to strike a balance, providing swift object detection while maintaining acceptable accuracy. Both the models work best in object detection but choosing a model depends on the type task need to done. YOLO-NAS stands out in situations where cutting-edge performance is paramount, while YOLOv8 is better suited for applications that prioritize a middle ground between speed and accuracy. The decision in choosing the model is contingent upon the distinct demands of the undertaking task,

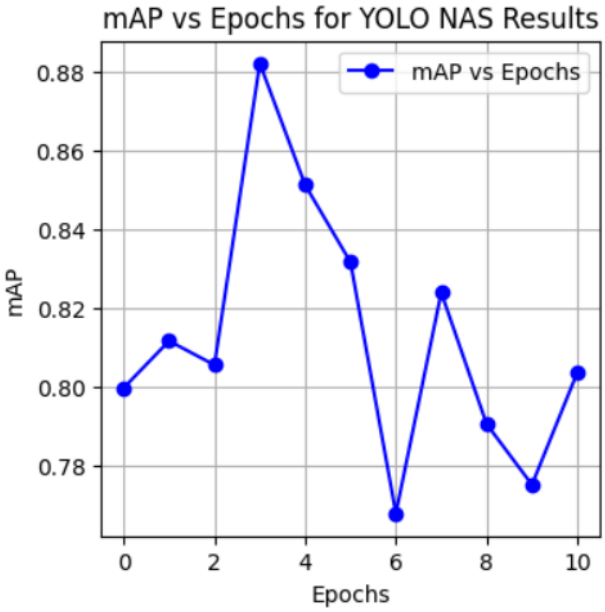


Figure 9: mAP vs Epochs for top 10 values

5 Conclusion and Future Work

In this research, the Single Stage Detection model like YOLO is studied and employed for pedestrian detection using the advanced YOLO-NAS object detection technique on a dataset obtained from Kaggle. The dataset, comprising video sequences, was preprocessed to yield images arranged in a square grid of 640x640 pixels. Bounding boxes were applied for pedestrian prediction, and the models were labeled as Model 1 to Model 4, each corresponding to 25, 50, 75, and 100 epochs of training, respectively. Remarkably, the third model displayed superior performance, achieving an exceptional mean Average Precision (mAP) score of 0.67, surpassing the other models.. Visualizations such as size plots and area plots were made to extract more detailed result for a comprehensive visual exploration of the data.

The implementation of YOLO-NAS showcased promising outcomes in pedestrian detection, with Model 3 demonstrating outstanding accuracy. The visualizations added depth to our comprehension of the dataset. This research contributes valuable insights into the effectiveness of YOLO-NAS for pedestrian detection applications.

Future work in this research could explore the extension of the pedestrian detection model to challenging conditions such as nighttime scenarios. Future research avenues for pedestrian detection include adapting the model for diverse weather conditions, refining training for small object detection and overcoming occlusion challenges. Exploring scenarios involving pedestrians crossing paths adds to the model's resilience. Further work is needed to enhance real-time detection by improving processing speed and efficiency, making the model more practical for dynamic environments. These refinements aim to elevate the pedestrian detection model, increasing its effectiveness and applicability in real-world scenarios, ultimately contributing to its successful deployment.

References

Boyuan, W. and Muqing, W. (2020). Study on pedestrian detection based on an improved yolov4 algorithm, *2020 IEEE 6th International Conference on Computer and Communications (ICCC)*, pp. 1198–1202.

Chen, Y. and Shin, H. (2020). Pedestrian detection at night in infrared images using an attention-guided encoder-decoder convolutional neural network, *Applied Sciences* **10**(3): 809.

Chen, Y.-Y., Li, G.-Y., Jhong, S.-Y., Chen, P.-H., Tsai, C.-C., Chen, P.-H. et al. (2020). Nighttime pedestrian detection based on thermal imaging and convolutional neural networks, *Sensors and Materials* **32**(10): 3157–3167.

COCO - Common Objects in Context (n.d.). `https://cocodataset.org/#home`. Accessed: [2023].

Dai, X., Hu, J., Zhang, H., Shitu, A., Luo, C., Osman, A., Sfarra, S. and Duan, Y. (2021). Multi-task faster r-cnn for nighttime pedestrian detection and distance estimation, *Infrared Physics & Technology* **115**: 103694.

Farooq, M. S., Khalid, H., Arooj, A., Umer, T., Asghar, A. B., Rasheed, J., Shubair, R. M. and Yahyaoui, A. (2023). A conceptual multi-layer framework for the detection of nighttime pedestrian in autonomous vehicles using deep reinforcement learning, *Entropy* **25**(1): 135.

Jönsson Hyberg, J. and Sjöberg, A. (2023). Investigation regarding the performance of yolov8 in pedestrian detection.

Kilicarslan, M., Zheng, J. Y. and Raptis, K. (2016). Pedestrain detection from motion, *2016 23rd International Conference on Pattern Recognition (ICPR)*, IEEE, pp. 1857–1863.

Kulhandjian, H., Barron, J., Tamiyasu, M., Thompson, M. and Kulhandjian, M. (2023). Pedestrian detection and avoidance at night using multiple sensors and machine learning, *2023 International Conference on Computing, Networking and Communications (ICNC)*, IEEE, pp. 165–169.

Lan, W., Dang, J., Wang, Y. and Wang, S. (2018). Pedestrian detection based on yolo network model, *2018 IEEE international conference on mechatronics and automation (ICMA)*, IEEE, pp. 1547–1551.

Neumann, L., Karg, M., Zhang, S., Scharfenberger, C., Piegert, E., Mistr, S., Prokofyeva, O., Thiel, R., Vedaldi, A., Zisserman, A. et al. (2019). Nightowls: A pedestrians at night dataset, *Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part I 14*, Springer, pp. 691–705.

Salim, S. (2017). Convolutional neural network for person detection using yolo framework, *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)* **9**(2-13): 1–5.

Sukkar, M., Kumar, D. and Sindha, J. (2021). Real-time pedestrians detection by yolov5, *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, pp. 01–06.

Terven, J., Córdova-Esparza, D.-M. and Romero-González, J.-A. (2023). A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas, *Machine Learning and Knowledge Extraction* **5**(4): 1680–1716.

Uma, M., Abirami, S., Ambika, M., Kavitha, M., Sureshkumar, S. and R, K. (2023). A review on augmented reality and yolo, *2023 4th International Conference on Smart Electronics and Communication (ICOSEC)*, pp. 1025–1030.

Wang, X., Chen, J., Wang, Z., Liu, W., Satoh, S., Liang, C. and Lin, C.-W. (2020). When pedestrian detection meets nighttime surveillance: A new benchmark, *Image* **20000**(30000): 40000.

Zachau, S. (2018). Caltech Pedestrian Dataset to YOLO Format Converter, `https://github.com/simonzachau/caltech-pedestrian-dataset-to-yolo-format-converter`.

Zhao, Z.-Q., Zheng, P., Xu, S.-t. and Wu, X. (2019). Object detection with deep learning: A review, *IEEE transactions on neural networks and learning systems* **30**(11): 3212–3232.

Zhou, H. and Yu, G. (2021). Research on pedestrian detection technology based on the svm classifier trained by hog and ltp features, *Future Generation Computer Systems* **125**: 604–615.