HUMAN ACTIVITY RECOGNITION ON EDGE DEVICES USING SALIENT REGION-GUIDED KNOWLEDGE-DISTILLED LIGHTWEIGHT MODEL

by

Md. Abrar Zahin
0422062540

MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONIC ENGINEERING

Department of Electrical and Electronic Engineering
Bangladesh University of Engineering and Technology
Dhaka, Bangladesh

November 2024

The thesis titled, "**HUMAN ACTIVITY RECOGNITION ON EDGE DEVICES USING SALIENT REGION-GUIDED KNOWLEDGE-DISTILLED LIGHTWEIGHT MODEL**", submitted by **Md. Abrar Zahin**, Roll No.: 0422062540, Session: April 2022, has been accepted as satisfactory in partial fulfillment of the requirement for the degree of Master of Science in Electrical and Electronic Engineering on 18 November 2024.

BOARD OF EXAMINERS

Candidate's Declaration

This thesis titled, "HUMAN ACTIVITY RECOGNITION ON EDGE DEVICES USING SALIENT REGION-GUIDED KNOWLEDGE-DISTILLED LIGHTWEIGHT MODEL", is the outcome of the research carried out by Md. Abrar Zahin under the supervision of Dr. Mohammad Ariful Haque, Professor, Department of Electrical and Electronic Engineering, Bangladesh University of Engineering and Technology (BUET), Dhaka-1205, Bangladesh.

It is also declared that neither this thesis nor any part thereof has been submitted anywhere else for the award of any degree, diploma, or other qualifications.

Signature of the Candidate

[illegible]

Md. Abrar Zahin
0422062540

Dedication

To my family

Contents

List of Figures

List of Tables

List of Important Abbreviations

CNN	Convolutional Neural Network
DNN	Deep Neural Network
HAR	Human Activity Recognition
KD	Knowledge Distillation
LMP	Layer-wise Magnitude-based pruning
mAP	Mean Average Precision
PWPR	Pre-trained Weight Pruned Retrained
SFGKD	Salient Feature Guided Knowledge Distillation
SOTA	State of the Art
YOLO	You Only Look Once

Acknowledgement

I would like to take this opportunity to express my heartfelt gratitude to everyone who has supported me during the course of my thesis. First and foremost, I am deeply thankful to my family for their unwavering love, encouragement, and understanding. Their constant support has been a source of strength, and I could not have completed this journey without their belief in me.

I am also incredibly grateful to my supervisor, Dr. Mohammad Ariful Haque, for his exceptional guidance throughout this research. His expertise, insightful feedback, and continuous encouragement have been instrumental in shaping the direction and quality of my work. I am truly fortunate to have had the opportunity to learn under his mentorship. Additionally, I would like to extend my sincere thanks to the course teachers for their valuable knowledge and guidance. Their teachings have greatly enhanced my understanding of the subject matter and have contributed significantly to the successful completion of this thesis.

I would also like to acknowledge the collective effort of everyone who has played a part in my academic journey. Whether through guidance, constructive feedback, or moral support, your contributions have significantly shaped this work and helped me navigate the challenges along the way. I am truly grateful for your presence in this journey, and I will always carry forward the lessons and encouragement you have shared with me.

Abstract

Understanding human behavior is crucial in disaster management, surveillance and elder care. In disaster management, drone surveillance can quickly identify activities like walking, running, lying down, standing, and sitting. This helps prioritize rescue operations and save lives. Accurate activity recognition also boosts security and effectively monitors public spaces. Additionally, in elder care, it assists in keeping an eye on the well-being of elderly individuals by promptly detecting falls or unusual behavior, which allows for timely intervention. Overall, these applications highlight the importance of real-time activity recognition for safety and support. Therefore, it's important to develop a lightweight and accurate model that can run on edge devices like Raspberry Pi used in drones and security cameras. Faster models often sacrifice accuracy compared to larger, complex Deep Neural Networks (DNN) which are highly accurate, require a lot of computational power, making it difficult to implement Human Activity Recognition (HAR) systems on devices with limited processing capabilities. In this thesis, we have made two major contributions. First, we introduced a new lightweight DNN model called YOLOv5n-light, which is 4 times smaller and 1.5 times less computationally intensive than YOLOv5n, the smallest model in the YOLOv5 family. Second, we proposed a novel training strategy called Pre-trained Weight Pruned Retrained by Salient Feature Guided Knowledge-Distillation (PWPR-SFGKD) to enhance the accuracy of lightweight YOLO models. In our approach, we first train the model, followed by pruning its weights, and then perform a retraining process using a Salient Feature Guided Knowledge Distillation (SFGKD) technique. For loss calculation, we compute the difference between the feature maps of the full image and the background region, derived from both the teacher and student models. This allows us to capture more meaningful, context-specific information during the distillation process. When we applied this method to the YOLOv5n model, we observed a significant improvement in performance, with the mAP-50 score increasing from 0.64 to 0.71. This result highlights the effectiveness of our proposed PWPR-SFGKD approach in enhancing model accuracy, outperforming traditional YOLOv5 training strategies.

Chapter 1

Introduction

1.1 Motivation

Identifying human behaviour plays a crucial role in disaster management, surveillance, and elder care. Effective disaster management requires quick decision-making and efficient resource allocation to save lives and minimize damage. Often in the forecast of disasters like cyclones, storms, and floods, it is crucial to evacuate large areas in a short amount of time. In this case, prioritization of resource allocation can be done efficiently if we have real-time data on human activities. Technologies like drones equipped with cameras and single-board computers can accurately detect human actions such as running, walking, standing, sitting, and lying down. This information is invaluable as it helps identify areas where people are in immediate danger and need urgent assistance. For example, detecting people running or walking can indicate ongoing evacuation efforts, while those sitting or lying down might need medical assistance or help with evacuation. Identifying people who are lying down is particularly important, as it could signify that they are injured, unconscious, or unable to move. This allows rescue teams to quickly locate and prioritize these individuals for medical aid and evacuation, potentially saving lives by addressing critical situations promptly. Moreover, it also gives a comprehensive picture of the ground situation. Which significantly helps to minimize the impact of disasters on human lives and property by ensuring timely and targeted interventions based on the actual conditions observed.

In the context of surveillance and monitoring, human activities such as running, standing, walking, lying, and sitting also play a crucial role in predicting behaviour. Recognizing normal behaviour patterns helps distinguish between security threats and everyday activities. For instance, observing someone running in a restricted area may signal an attempt to flee, prompting an immediate alert to security personnel for a swift response. Conversely, in elder care, detecting activities like lying on the ground is critical. This could indicate a fall, requiring prompt medical assistance for the well-being of elderly individuals. Rapid detection and response to such events through surveillance systems contribute significantly to ensuring the safety and health of seniors in care facilities. By integrating human action information with real-time population density data, organizers of large open-space events can significantly enhance crowd management strategies. Monitoring activities such as sitting, walking, and standing helps them efficiently allocate human resources. Prioritizing locations based on real-time drone data, which captures the number of people in a given region, offers authorities valuable insights. It enables organizers to identify high-priority zones where crowd management or emergency responses are urgently needed. This integrated approach allows for the strategic allocation of resources and the redirection of crowds to different areas as necessary, ensuring smoother event logistics and enhancing overall safety.

To enhance the effectiveness of disaster management, there’s a critical need for light architectured and precise object detection models tailored for edge devices, focusing on human activity datasets. However, achieving the desired accuracy while maintaining model speed poses a significant challenge. As accuracy increases, so does the number of model parameters, inevitably leading to longer inference times. Conversely, reducing the model size to improve inference speed often leads to a trade-off in accuracy. Balancing these trade-offs is essential to ensure that the model is both accurate and efficient enough to be deployed on edge devices, where computational resources are limited.

1.2 Challenges

Researchers have explored various techniques to create compact and lightweight models, including model pruning [1], low-rank decomposition [2], quantization [3], and knowledge distillation [4]. However, these methods still face several challenges, one of which is the difficulty in detecting small objects. In particular, detecting humans becomes challenging when they appear small relative to the overall image, as their size compared to the background makes it harder for object detection models to differentiate them. This task is further complicated when the background varies significantly between images, especially when a low-latency model is required for real-time processing. Additionally, certain activities, like distinguishing between walking and running, can be challenging to accurately identify from a single image, even for humans, which adds another layer of complexity to the detection task.

Another significant challenge in human activity recognition is the trade-off between speed and accuracy. Achieving higher accuracy typically requires the use of heavier models, which results in longer inference times. Conversely, to ensure faster inference, lighter models are necessary, but this often leads to a drop in accuracy due to reduced model parameters and size. This challenge becomes even more pronounced when using lightweight or trimmed versions of models with fewer parameters. When training such models, capacity limitations and poor generalization across diverse and heterogeneous data distributions prevent traditional training methods from improving their accuracy effectively. Furthermore, there are complexities in compressing larger models compared to existing lightweight models, as compressing larger models often results in significantly better performance than compressing models that are already designed to be lightweight. At the same time, models must be optimised for efficiency to achieve faster computations in resource-constrained environments. However, this optimization often amplifies the difficulty of ensuring generalization. Models that perform well on limited, homogeneous datasets usually struggle in more diverse and dynamic environments. This lack of generalization is particularly problematic for lightweight models, where balancing multiple factors becomes exponentially difficult. Achieving one aspect, such as speed or accuracy, often requires compromising the other, making it a delicate trade-off to maintain overall model performance.

1.3 Objectives of the Thesis

The objectives with specific aims are-

- To detect humans and recognize activity from video in edged devices.
- To evaluate the performance of trained models in raspberry pi.
- To enhance the trained model performance by Knowledge Distillation leveraging teacher student analogy.
- To compare the performance of conventional trained models and knowledge distilled models.

1.4 Contribution

Our research addresses the key challenges mentioned in Section 1.2 by introducing two primary contributions, along with two additional enhancements, in our proposed technique.

Major Contributions:

- We propose a novel technique called Salient Feature Guided Knowledge Distillation, which transfers knowledge from a pre-trained, larger teacher model to a lightweight, untrained student model by utilizing salient feature maps.
- We introduce a new lightweight variant of YOLOv5, named **YOLOv5n-light**, specifically designed for edge devices to detect small objects. This model achieves higher accuracy and faster inference, making it suitable for video processing in edge devices.

To further improve the performance of our technique, we incorporate two additional strategies:

- Incorporating SmoothL1 Loss as a loss function for knowledge distillation.
- Applying weight pruning after a set number of training epochs to optimize model efficiency.

In the end, integrating these contributions, We propose a novel knowledge distillation method called Pre-trained Weight Pruned Retrained by Salient Feature Guided Knowledge Distillation (PWPR-SFGKD) for lightweight models, designed to significantly enhance detection accuracy.

1.5 Thesis Outline

The thesis is organized in the following manner -

Chapter 2 provides a comprehensive review of existing research on human activity recognition on edge devices. It carefully analyzes various proposed techniques and their limitations, aiming to identify gaps and opportunities for further exploration within current methodologies.

Chapter 3 introduces the fundamental concepts behind state-of-the-art object detection models based on deep neural networks. It focuses on the well-known You Only Look Once (YOLO) model, offering a detailed explanation of its architecture and how the YOLOv5 variant operates to detect objects in images.

Chapter 4 explores the methods underlying our proposed technique, Pre-trained Weight Pruned Retrained by Salient Feature Guided Knowledge Distillation (PWPR-SFGKD). This chapter highlights the architectural overview and working principles of our innovative approach.

Chapter 5 discusses the experiments conducted and the results obtained. It begins with an overview of our data processing steps and the Okutama-Action dataset. Then, it systematically outlines the experiments we performed to validate our technique. We'll present figures, tables, and charts to illustrate potential reasons for performance improvements and degradations. Finally, we'll compare the performance of our proposed method with state-of-the-art models and existing knowledge distillation techniques to showcase the effectiveness of our approach.

Chapter 6 concludes with a summary of our findings and reflections on the challenges encountered throughout the research. This chapter provides an overview of key discoveries and insights while offering guidance for future research in this field.

Chapter 2

Literature Review

The field of Human Activity Recognition (HAR) has advanced significantly, driven by extensive research and the development of numerous techniques. However, it still faces substantial challenges in real-world applications. These challenges become even more pronounced when attempting to recognize human actions on edge devices, such as Raspberry Pi. To gain a deeper understanding of this field and its associated obstacles, we will adopt a structured approach to examine the research in HAR, as outlined below:

- **Challenges and Techniques in HAR**: We will begin by exploring the key challenges in HAR, focusing on two primary domains: (i) vision-based approaches and (ii) sensor and smartphone-based methods. This section will provide an overview of the available solutions and their effectiveness in tackling these challenges.
- **Optimizing Model for Edge Applications**: Next, we will investigate the methodologies researchers have developed to create lightweight, accurate HAR models. This includes examining techniques to improve model accuracy, address real-time constraints, and identify ongoing research gaps.

2.1 Challenges and Techniques in HAR

Human Activity Recognition (HAR) can be performed using various modalities, with vision-based HAR and sensor-based HAR being two of the most popular approaches. Each technique comes with its own set of limitations. In the following section, we will explore the proposed methods designed to address the specific challenges associated with each approach.

2.1.1 Vision based HAR

Human action recognition based on vision involves the use of images or videos, to automatically detect, and understand human actions by analyzing visual information. It aims to recognize various activities performed by individuals, ranging from simple actions to complex activities like sports or exercises. It is used across various domains such as surveillance and healthcare, where interpreting human actions from visual data is crucial for decision-making and monitoring purposes. In the following section, we will examine the evolution of vision-based Human Activity Recognition (HAR) by comparing traditional and modern approaches. First, we will explore the older techniques, which primarily rely on handcrafted features for activity recognition. Then, we will shift our focus to modern methods, where AI driven HAR techniques would be discussed.

2.1.1.1 Handcrafted feature-based HAR

Handcrafted features in human activity recognition are manually crafted features from images, or videos which captures different attributes of human motion or behaviors. In this approaches, human expertise plays a crucial role in identifying unique features. This process involves three main steps: first, detecting the main action area; second, carefully choosing and extracting important features based on expert knowledge; and finally, using these features to classify actions.

Various methods have been explored to extract image features, incorporating techniques such as spatial or temporal cues to accurately recognize actions. These cues are represented spatially by body models [5,6] , image models [7]. These spatial-temporal features are vital for identifying human actions. They come in two forms: volume-based, which treat videos as spatial-temporal volumes, and trajectory-based, which track joint positions to create a 3D activity representation. To extract volume based features Scale Invariant Feature Transformation (SIFT) is used which detects 2D interest points. On the other-hand, to capture motion characteristics descriptors like Histogram of Oriented Gradients (HOG), Histogram of Optical Flow (HOF), and Motion Boundary Histogram (MBH) are used [8]. These approaches excel in recognizing complex actions and are robust to noise and lighting variations. Furthermore, to collect information related to sudden motions or object appearance/disappearance, Space-Time Interest Point (STIP) methods are utilized [9]. Handcrafted methods encounter significant challenges when adapting to new environments, as they are designed to identify a custom feature only for a specific context. This process can be labor-intensive and often hampers generalization, since handcrafted features are designed for particular scenarios. Furthermore, these methods are vulnerable to noise, which can adversely affect their performance. Their ability to represent complex problems is also limited, often failing to capture the full intricacy of the task. To address these limitations, learning-based Human Activity Recognition (HAR) techniques have been developed, aiming to overcome the difficulties associated with handcrafted methods.

2.1.1.2 AI Driven HAR Techniques

So far we have discussed about the pre-deep learning era of vision-based human activity recognition. With the advancement of deep learning, the focus of current mainstream research shifted towards developing various deep learning frameworks due to their strong representation capabilities and superior performance compared to Handcrafted Feature based HAR. The research on human action detection can broadly be classified into three categories: (i) human action recognition using spatial and temporal streams (ii) gait-based and skeleton tracking methods, and (iii) deep neural network-based human action detection.

In spatial-temporal stream-based human action recognition (HAR), two-stream CNNs are predominantly used [10]. These CNNs are trained on multi-frame dense optical flow. The temporal stream handles motion through dense optical flow, while the spatial stream processes each video frames. The main goal in gait and skeleton-based human activity recognition is to extract significant features from the movement patterns exhibited by human body joints or skeletal structures. However, the difference in gait method and skeleton tracking is Gait focuses on feature from silhouette images (dark outlines of objects or people against a lighter background) and relies more on motion pattern, while skeleton tracking method focuses on capturing and processing movements of key body joint [11]. In gait-based action recognition, methods such as Graph Convolutional Networks (GCNs) with higher-order inputs [11] and the Spatial-Temporal Graph Convolutional Network (ST-GCN) architecture [12] are commonly used. Similarly, skeleton-based human action recognition often utilizes models like 3D-CNNs [13], CNN-LSTM networks [14], and Spatial Temporal Transformer networks [15].

There are three types of Deep Neural Network (DNN) models available in human activity recognition. They are : Single-stage detectors, Two-stage detectors, and Hybrid detectors. Each type varies in architectural approach and the number of stages involved in the detection process. Single-stage detectors like Single Shot Detectors (SSD) and You Only Look Once (YOLO) models perform object localization and classification simultaneously, offering real-time processing capabilities. In contrast, two-stage detectors such as Region-based Convolutional Neural Network (RCNN) and its variants like

Mask-RCNN, Fast-RCNN, and Faster-RCNN first propose Regions of Interest (ROIs) before identifying associated activities [16]. Hybrid detectors integrate features from both single-stage and two-stage detectors, often leveraging additional components like attention mechanisms, typically based on Transformer architectures. Well-known examples of hybrid detectors include Detection Transformer (DETR), Deformable DETR, and Segmentation Transformer (SETR) [17].

The adoption of deep learning has revolutionized Human Activity Recognition (HAR), significantly enhancing accuracy and simplifying feature extraction. However challenges in vision-based HAR, such as variations in subject positioning, changes in lighting, and privacy concerns, have led to a shift towards using different modalities for human activity recognition, with sensor-based solutions becoming a popular alternative. These solutions leverage data from smart wearables equipped with accelerometers and gyroscopes, eliminating the need for image or video sources [18, 19]. In the section below we will discuss about the proposed methodologies and challenges associated with this technique.

2.1.2 Smartphone and Sensor-Based HAR

Early wearables demonstrated impressive accuracy in recognizing daily activities, and smartphones have emerged as versatile sensing tools, utilizing sophisticated algorithms to handle complex activity recognition. Recent research has also explored the potential of wireless signals, particularly Wi-Fi, for HAR applications. Studies have investigated various sensing strategies, such as RSSI, CSI, FMCW, and Doppler shift-based approaches, aiming to overcome limitations and advance Wi-Fi-based HAR techniques for applications like fraud detection and daily activity monitoring [20].

Nevertheless, smartphone and sensor-based HAR are not without their own set of challenges. Key issues include limited computational resources on mobile devices, overfitting and poor generalization, a lack of comprehensive public datasets, difficulties in capturing subtle features, privacy concerns related to sensitive data, and the complexity of managing temporal dependencies in real-time applications [21].

2.2 Optimizing Model for Edge Applications

Optimizing models for edge applications involves preparing lightweight models tailored for edge devices through various model compression methods. This typically includes approaches like model quantization, pruning, and knowledge distillation, which are designed to reduce the model's size and computational requirements while maintaining high performance. These strategies make sure that models can deliver accurate results efficiently on compact and less powerful edge devices. To enable human recognition on edge devices, it's essential to ensure that the model is lightweight enough to operate efficiently within resource-constrained environments. Addressing the challenge of light-weighting a model, researchers have concentrated on various techniques for model compression like model pruning [22], low-rank factorization [23], quantization [24], and knowledge transferring technique like knowledge distillation [25]. In the section below we will discuss model compression techniques and their challenges in detail.

2.2.1 Model Pruning and Quantization

Model pruning and quantization are effective methods for reducing the number of model parameters and decreasing inference time. The main idea of model pruning is to remove redundant parameters by setting a set of standards through which network sparsing and compression are achieved. Mainly there are two types of pruning: (i) structured pruning and (ii) unstructured pruning. Unstructured pruning mainly involves removing unimportant weights from different layers to reduce the number of parameters with minimal impact on the overall structure. On the other hand structured pruning involves removing insignificant structures like kernels, layers, channels and filters. However different techniques have been proposed to perform unstructured pruning. Lin et al. [26] introduced a method called deep gradient compression (DGC), which selects only the most "important" gradient elements in each iteration for transmission. The technique of pruning weights by setting thresholds for each layer is known as layer-wise magnitude-based pruning (LMP), but one of the major challenges in using this technique is it requires professional expertise [27] to select pruning threshold properly to solve this problem. Moreover, Li et al. [28] suggested optimization-based LMP (OLMP) which can automatically adjust the pruning threshold and perform LMP operation. Though unstructured pruning is easy to implement, it poses several challenges and errors in the reconstruction of the new architecture. As weights from different parts of the network are removed so after pruning the network architecture needs to be updated which often results in unnecessary complexities. So as an alternative rather than removing insignificant weights, irrelevant structures are pruned. Based on the level of granularity, structural pruning techniques can be categorized into filter/feature map/kernel pruning, channel pruning, and layer pruning. Filter pruning is applied to the filter of the CNN where unimportant filters and their associated weights are removed.

Among the various techniques for filter pruning are structural sparse CNN regularization and acceleration methods, a channel selection strategy utilizing LASSO, and a least-squares reconstruction algorithm for pruning filters [29]. Filter pruning causes dimensional mismatch so to tackle this problem rather than removing different filters the whole irrelevent channels are removed from CNN. Liu et al. [30] introduced the use of L1 regularization for channels, which drives the value of the batch normalization (BN) scaling factor towards zero to identify and remove insignificant convolutional channels. Ding et al. proposed [31] lossless channel pruning technique reduce the number of output channels in convolutional layers to slim down a CNN. In addition to that, another pruning technique is also explored among the researchers called layer pruning where the entire unimportant layer is removed. Dong et al [32] presented a noble layer pruning where each layer's parameters are pruned independently using second-order derivatives of the layer-wise error function, and performance is restored through reconstruction error and retraining.

Another key technique for model compression is quantization, which reduces the precision of a model's parameters (weights) and/or activations from floating-point values to lower-bit formats. This reduction in precision decreases the model's storage size and accelerates inference. There are two main quantization techniques: Post-Training Quantization (PTQ) and Quantization-Aware Training (QAT). Among these, QAT generally delivers better accuracy compared to PTQ. Implementing model pruning and quantization techniques in Human Activity Recognition (HAR) introduces several significant challenges. One major issue is the potential degradation of model performance due to aggressive pruning or the use of lower precision quantization, which can lead to a loss of accuracy. Additionally, the fine-tuning process after these adjustments is often intricate and time-consuming, requiring specialized training methods to regain lost performance. Another challenge arises from hardware limitations, as not all devices are capable of efficiently supporting reduced-precision operations [33].

2.2.2 Knowledge Distillation

Knowledge distillation is an effective technique that was designed to enhance the performance of smaller, simpler models compared to methods like model pruning and quantization. It leverages the expertise of larger, more accurate models, known as the "teacher," to train smaller "student" models. The teacher model's complex architecture allows it to excel in learning and generalizing from the data. After training the teacher model, its knowledge is transferred to the student model by training it with soft labels. Soft labels/targets are the probability distribution of the teacher model. By following the soft targets and hard targets (ground truths), the student model efficiently learns the data's intricate patterns and relationships. This technique enables the student model to achieve higher accuracy and performance despite its reduced size and complexity.

Initially, knowledge distillation employed methods like logits mimicking and output distribution matching. However, for tasks like object detection, additional techniques such as feature-based knowledge distillation, relational knowledge distillation, and self-distillation were proposed to further improve the model's accuracy. As accuracy improvements became necessary, feature-based distillation gained popularity, using L2 Norm Loss (Mean Square Error) to align features extracted by both models. Feature-based knowledge distillation aims to minimize the disparity between the feature embeddings of the teacher and student models [34]. In contrast, relational distillation focuses on capturing structural relationships instead of directly imitating specific layers [35]. Self-distillation entails a model producing both hard and soft targets during training. It improves its learning process by utilizing its own past predictions as soft labels in addition to the ground truth labels [36].

When investigating the effects of knowledge-distilled SSD and lightweight YOLO models, knowledge distillation enhance model accuracy compared to conventional training approaches. Ma and Tian (2022) utilized ensemble knowledge distillation to enhance the accuracy of the Tiny YOLO model [37]. Similarly, Dong et al. (2022) introduced adversarial learning-based knowledge distillation for SSD-lite with the MobileNetV1 architecture, achieving superior accuracy compared to traditional model training [38].

When comparing knowledge distillation to other techniques, it's evident that knowledge distillation excels in improving accuracy of lightweight models by trasfering knowledge from larger teacher model. For instance, when humans are small in the context of the entire image, identifying the crucial region becomes vital. Lightweight models often struggle to differentiate between background and salient regions compared to traditional knowledge distillation techniques [39]. Furthermore, a significant challenge in knowledge distillation is the requirement for substantial computational resources and extended training time to effectively train a model. Additionally, an over-reliance on the teacher model can cause the student model to absorb incorrect knowledge, ultimately leading to a significant reduction in its accuracy [40].

2.3 Summary

The field of Human Activity Recognition (HAR) is well established, yet there exists a notable research gap in crafting lightweight, yet precise models suitable for edge devices. While larger models generally provide higher accuracy, they also have greater sizes and slower inference times, creating challenges for real-time video analysis based applications. Moreover, contemporary knowledge distillation techniques face challenges such as difficulty in grasping salient regions within images and an excessive reliance on the teacher model, hindering their effectiveness. Consequently, there's a pressing demand for innovative methodologies to develop and train robust Human Activity Recognition (HAR) models tailored for edge devices, especially in real-time applications.

In the next chapter, we will explore one of the leading object detection models, YOLO, known for its exceptional accuracy and fast inference time. We will dive into YOLOv5 and its earlier versions, discussing their challenges, advancements, and the powerful features they offer for modern object detection tasks.

Chapter 3

State of the Art Object Detection Model

There are various types of state-of-the-art object detection models, including transformer based detection models, two-stage detectors, and single-stage detectors. While transformer based and two-stage detector models provide higher accuracy, their high computational demands make them impractical for resource-constrained devices like the Raspberry Pi and Jetson Nano [41]. In contrast, single-stage detectors, such as YOLO, are preferred for their faster inference times compared to two-stage and hybrid detectors [42]. Between the single-stage detectors, YOLO generally achieves better accuracy than SSD models [43]. Additionally, different variants of the EfficientDet model offer higher accuracy for various classes compared to YOLOv3 and YOLOv5. However, EfficientDet models can struggle with real-time applications, where YOLOv5 often excels with lower inference times and higher frames per second (FPS) [44, 45].

Before getting into YOLOv5, it's helpful to look at how the YOLO models evolved from YOLOv1 to YOLOv4. By seeing internal architecture like the backbone networks, detection heads, and loss functions changed over time, we can understand the choices made to balance accuracy, speed, and computing power. This gives a clear picture of the challenges and improvements addressed in YOLOv5 [46, 47].

3.1 You Only Look Once (YOLO)

You Only Look Once (YOLO) is a pioneering end-to-end neural network for object detection that simultaneously predicts bounding boxes and class probabilities, departing from traditional algorithms that rely on a two-stage approach of first identifying regions of interest and then predicting bounding boxes within each region. This unified approach enables YOLO to achieve state-of-the-art real-time performance.

3.1.1 YOLOv1

YOLOv1 changed object detection by detecting multiple bounding boxes simultaneously. It divides the input image into an S × S grid and predicts B bounding boxes per grid cell for C classes. Each bounding box prediction includes p_c for confidence of the prediction bbbox, b_x and b_y for box center coordinates relative to the grid, and b_h and b_w for box height and width relative to the entire image. The model's output is an S × S × (B × 5 + C) tensor, after the predictions non-maximum suppression (NMS) algorithm was used to remove duplicates.

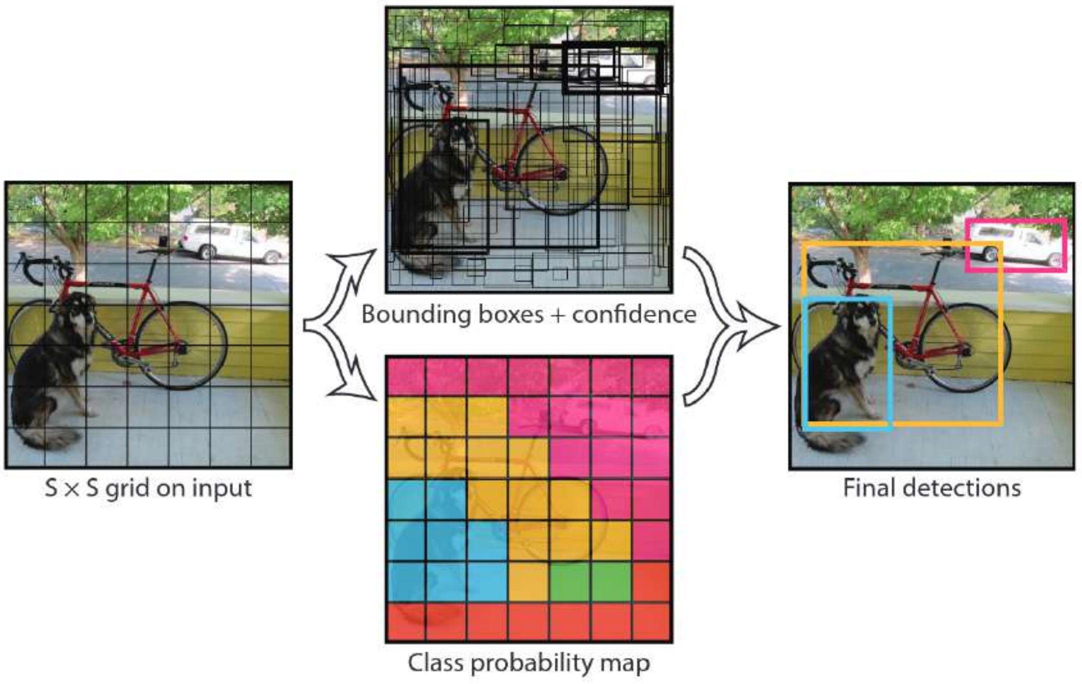


Figure 3.1: Detecting Objects with YOLOv1 [48]

In the original YOLO paper [48], the authors used the PASCAL VOC dataset with 20 classes (C = 20), a grid size of 7 × 7 (S = 7), and allowed up to 2 bounding boxes per grid cell (B = 2), resulting in a 7 × 7 × 30 prediction output. For a detailed overview of the YOLOv1 object detection process, refer to Figure 3.1.

From Table 3.1 its clear that YOLOv1 model consists of 24 convolutional layer and pooling layers followed by two fully connected layers that predict bounding box coordinates and class probabilities. Its important to note that all layers employ leaky ReLU activations, except for the final layer, which uses a linear activation function [49]. Draw-

Type	**Filters**	**Size**	**Stride**	**Output**
Conv	64	7×7	2	224×224
Max Pool	-	2×2	2	112×112
Conv	192	3×3	1	112×112
Max Pool	-	2×2	2	56×56
1× Conv	128	1×1	1	56×56
Conv	256	3×3	1	56×56
Conv	256	1×1	1	56×56
Conv	512	3×3	1	56×56
Max Pool	-	2×2	2	28×28
4× Conv	256	1×1	1	28×28
Conv	512	3×3	1	28×28
Conv	512	1×1	1	28×28
Conv	1024	3×3	1	28×28
Max Pool	-	2×2	2	14×14
2× Conv	512	1×1	1	14×14
Conv	1024	3×3	1	14×14
Conv	1024	3×3	1	14×14
Conv	1024	3×3	2	7×7
Conv	1024	3×3	1	7×7
Conv	1024	3×3	1	7×7
FC	4096	-	-	4096
Dropout	0.5	-	-	4096
FC	-	-	-	$7 \times 7 \times 30$

Table 3.1: YOLOv1 model architecture overview.

ing inspiration from GoogLeNet and Network in Network, YOLOv1 incorporates 1×1 convolutional layers to reduce the number of feature maps, effectively minimizing the number of parameters and maintaining computational efficiency.

3.1.2 YOLOv2

After the first version of YOLO, a new and improved version called YOLO9000 was released. The creators came up with a new way to combine two important tasks: identifying objects and classifying them. They did this by merging two big datasets: COCO and ImageNet [50]. As a result, YOLOv2 became much better at detecting objects than its predecessor. However, it still had trouble finding small objects and objects with unusual shapes. It also didn't perform as well as some other top-notch detectors, like Faster R-CNN. Additionally, YOLOv2 required more computer power because it had to handle more objects and categories, making it less suitable for use on lower-end devices in real time.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	GlobalPooling			
	FC		1000	
	Softmax			

Table 3.2: DarkNet-53 Architecture [51]

3.1.3 YOLOv3

YOLOv3 model made significant changes in architecture. First change it made is the use of new backbone network called Darknet-53 which consists of 53 layer convolutional network with residual connections (see Table: 3.2). It also included modified spatial pyramid pooling block for larger receptive field, where the AP50 was improved by 2.7%. It also included prediction boxes at three different scales along with bounding box priors which are determined by k-means clustering algorithm [51].

Despite introducing improvements such as the Darknet-53 architecture and spatial pyramid pooling operation, YOLOv3 had several drawbacks, including high computational intensity, longer training times, increased susceptibility to overfitting, and higher memory requirements. These challenges were later addressed in subsequent versions, including YOLOv4 and YOLOv5, which aimed to improve the performance and efficiency of the YOLO object detection algorithm.

3.1.4 YOLOv4

YOLOv4 is an advanced object detection model that builds upon the success of YOLOv3, introducing several key improvements to enhance its performance and efficiency. Based on the Darknet-53 architecture, it incorporates Cross-Stage Partial (CSP) connections, a new activation function called Mish, and is divided into three sections: backbone, neck, and head. The backbone is responsible for feature extraction, the neck for feature fusion with CSP connections, and the head for object detection and classification. YOLOv4 uses binary cross-entropy as its loss function for classification, and employs two techniques to improve training efficiency and robustness: Bag-of-Freebies and Bag-of-Specials. Additionally, it utilizes genetic algorithms on the first 10% of the training periods to find optimal hyperparameters, self-adversarial training for robustness, and CIoU loss for accuracy. However, despite these improvements, YOLOv4 still struggles with slower inference speeds and increased computational requirements, making it less suitable for real-time object detection applications on lower-end hardware.

3.1.5 YOLOv5

As the successor to YOLOv4, YOLOv5 introduced optimization features like the Auto-Anchor algorithm, which enhances anchor boxes for specific datasets and training configurations. The k-means function initializes anchor sizes for the Genetic Evolution (GE) algorithm, which refines them using CIoU loss and Best Possible Recall as fitness criteria. YOLOv5 is a powerful object detection algorithm that processes images in several steps. The Backbone layer first extracts essential features, represented as feature maps, which are then merged using a feature fusion network (Neck) into three distinct feature maps (P3, P4, P5) for detecting small, medium, and large objects. These maps predict object class confidence and bounding box coordinates, after which Non-

Maximum Suppression (NMS) filters out overlapping boxes to produce final, accurate object detections. Figure 3.2 displays the overview of how from an input we get the bounding boxes as the prediction output.

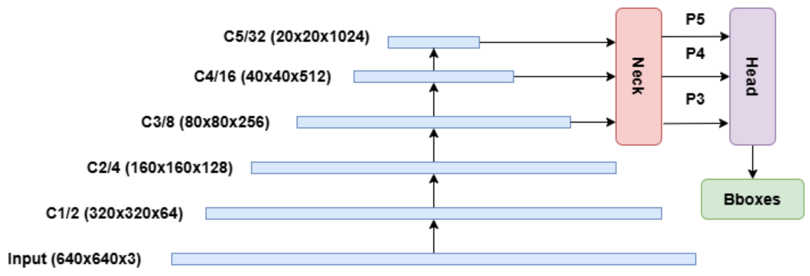


Figure 3.2: YOLOv5 Core Architecture Overview

3.1.5.1 Backbone

In YOLOv5 the backbone is based on CSPDarknet53, built by stacking Cross Stage Partial (CSP) layers. Each CSP layer combines a sequence of CBS (Conv + BatchNorm + SiLU) blocks with residual connections, enhancing feature extraction capabilities for comprehensive representation across different object scales. While residual connections help mitigate the vanishing gradient problem, they can also introduce redundant gradients, increasing inference time. However, CSP layers ensure efficient gradient propagation by splitting the base layer into two parts and recombining them through a cross-stage hierarchy. YOLOv5 introduces C3 with bottleneck, a more efficient variant of CSP that employs three convolutional layers and a bottleneck layer to reduce parameters and computations (ref Figure 3.4). This design allows YOLOv5 to achieve state-of-the-art object detection performance while maintaining a relatively low computational cost, making C3 with bottleneck a crucial component of its success [52].

3.1.5.2 Neck

The Neck module in YOLOv5 efficiently combines multi-scale features from different layers into the network's feature map. It incorporates multi-scale feature representation and fusion strategies, similar to Feature Pyramid Networks (FPN). The Path Aggregation Network (PANet) enhances this by using top-down and bottom-up approaches for bidirectional feature flow at various pyramid levels. PANet fuses different spatial resolutions and semantic levels through lateral connections, employing varying pooling operations (1x1, 5x5, 9x9, 13x13) to capture contextual information.

Additionally, the Spatial Pyramid Pooling Fusion (SPPF) module, an enhancement of the traditional Spatial Pyramid Pooling (SPP) module from SPPNet, optimizes feature representation in the neck. SPPF addresses SPPNet's limitations by aggregating features from various scales using max pooling, combining them through weighted sums or concatenation, and reducing dimensionality with a 1x1 convolution followed by a 3x3 convolution. This leads to improved computational efficiency, preserved spatial information, and enhanced feature representation.

3.1.5.3 Head

The main task of Head module is to uses the neck generated feature map to generate bounding boxes and labels for detected objects. YOLOv5 uses the same head architecture as YOLOv3 and YOLOv4, consisting of three feature layers that are used to predict the location of bounding boxes, confidence score, and class label. In YOLOv5, bounding box predictions are made using three feature maps from the network's neck, which depends on image size (i_hxi_w). These feature maps are of ($i_h/8 \times i_w/8 \times 256$), ($i_h/16 \times i_w/16 \times 512$), and ($i_h/32 \times i_w/32 \times 1024$) size. Each feature map goes through a prediction head, which consists of two convolutional layers. The first convolutional layer has a 3×3 kernel and uses 256, 512, or 1024 filters, depending on the feature map. The second convolutional layer has a 1×1 kernel and $3(\text{num\_classes} + 5)$ filters. The resulting output tensors have sizes of ($i_h/8 \times i_w/8 \times 3(\text{num\_classes} + 5)$), ($i_h/16 \times i_w/16 \times 3(\text{num\_classes} + 5)$) and ($i_h/32 \times i_w/32 \times 3(\text{num\_classes} + 5)$) respectively.

The predicted outputs are then decoded by applying a sigmoid activation, reshaping to $(H \times W \times 3, \text{num\_classes} + 5)$. Here YOLOv5 gives 3×5 values for each grid cell. The 5 values are (t_x, t_y, t_w, t_h, and t_{conf}). t_x and t_y are the predicted bounding box's center coordinate values, and t_w and t_h are the predicted bounding box's width and height. t_{conf} is the confidence score of the predicted bounding box.

Center coordinates (c_x, c_y)

The center coordinates (c_x, c_y) of the bounding box are computed as follows:

$$c_x = \sigma(t_x) + x_{\text{grid}}$$
$$c_y = \sigma(t_y) + y_{\text{grid}}$$

Here, t_x and t_y are passed through a sigmoid function σ to ensure they are between 0 and 1. The resulting values are then added to the grid coordinates (x_{grid} and y_{grid}) to get the final center coordinates (c_x and c_y) of the bounding box.

Width and height (b_w, b_h)

The width and height (b_w, b_h) are computed as follows:

$$
\begin{aligned}
b_w &= w_{\text{anchor}} \times e^{t_w} \\
b_h &= h_{\text{anchor}} \times e^{t_h}
\end{aligned}
$$

Here, t_w and t_h are exponentiated using e^x to get a positive value. The resulting values are then multiplied by the anchor box dimensions (w_{anchor} and h_{anchor}) to get the final width and height (b_w and b_h) of the bounding box.

Final bounding box coordinates

Using the center coordinates (c_x, c_y) and the width and height (b_w, b_h), we can compute the final bounding box coordinates as follows:

$$
\begin{aligned}
x_1 &= c_x - \frac{b_w}{2} \\
y_1 &= c_y - \frac{b_h}{2} \\
x_2 &= c_x + \frac{b_w}{2} \\
y_2 &= c_y + \frac{b_h}{2}
\end{aligned}
$$

Objectness score, Class probabilities, Confidence score

The objectness score (confidence) is computed by passing the predicted value through a sigmoid function σ. The class probabilities are computed by passing the predicted values through a softmax function. By multiplying the objections score with class probabilities the confidence score for each prediction is calculated.

Final prediction

The final prediction is a tensor with shape $(H \times W \times 3, 6)$, where each row represents a bounding box. The 6 values are: x_1, y_1, x_2, y_2 (bounding box coordinates), confidence, class_id (class with the highest probability).

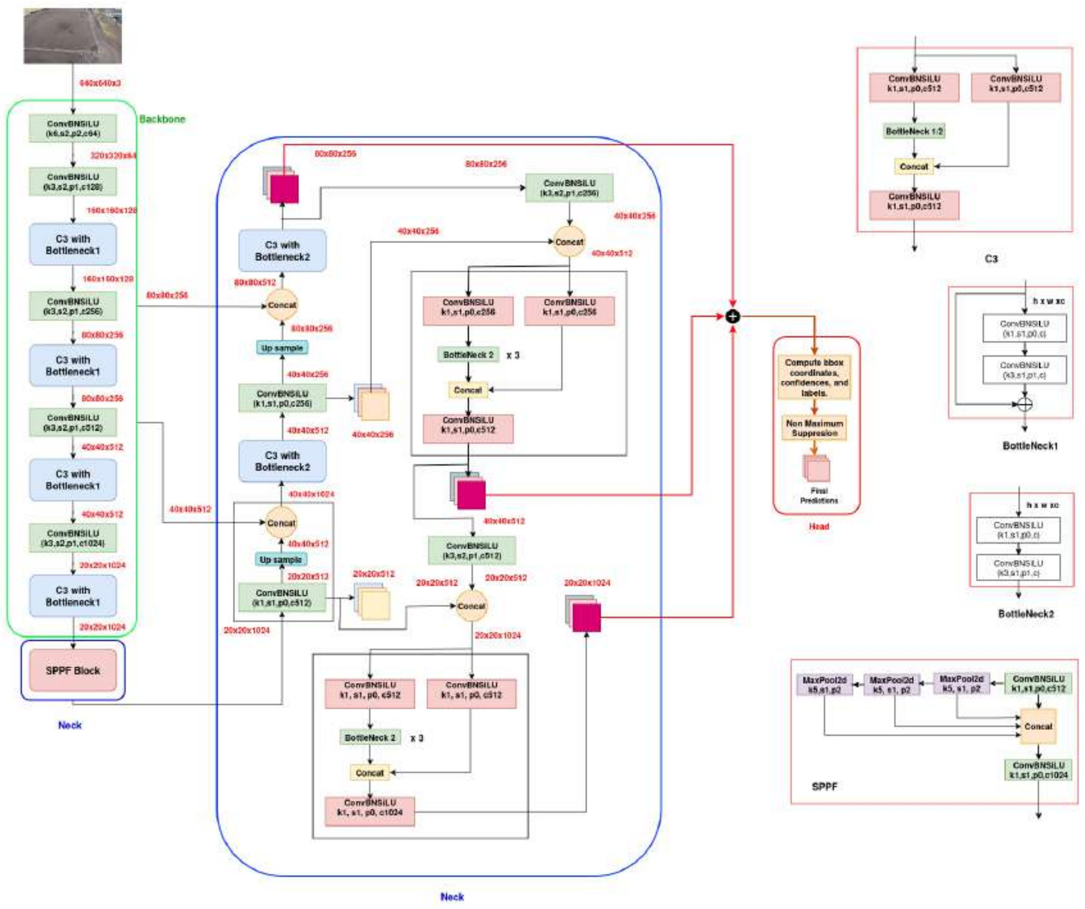


Figure 3.3: YOLOv5 full architecture

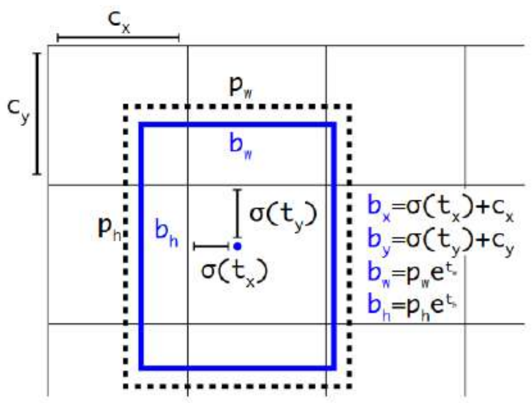


Figure 3.4: Bounding boxes with dimension priors and location prediction [50]

Final Bounding Box (x_1, y_1, x_2, y_2)

Using the center coordinates (c_x, c_y) and the width and height (b_w, b_h), the final bounding box's top-left (x_1, y_1) and bottom-right (x_2, y_2) corners can be calculated using the following equations:

$$x_1 = c_x - \frac{b_w}{2}$$

$$y_1 = c_y - \frac{b_h}{2}$$

$$x_2 = c_x + \frac{b_w}{2}$$

$$y_2 = c_y + \frac{b_h}{2}$$

In YOLOv5, five model variants—Extra-Large, Large, Medium, Small, and Nano—differ mainly in their depth and width multipliers (ref: Table 3.3). The depth multiplier adjusts the number of layers, while the width multiplier controls the number of channels per layer, allowing for a balance between computational efficiency and accuracy.

Model	Depth Multiplier	Width Multiplier
YOLOv5x	1.33	1.25
YOLOv5l	1.0	1.0
YOLOv5m	0.67	0.75
YOLOv5s	0.33	0.50
YOLOv5n	0.33	0.25

Table 3.3: YOLOv5 Model Variants with Depth and Width Multipliers

For this research, we selected YOLOv5 due to its superior balance between precision and speed. YOLOv5 outperforms its predecessors with higher mean Average Precision (mAP) and frames per second (FPS), making it ideal for various computer vision tasks. Its lower computational requirements and reduced risk of over-fitting make it a more practical choice compared to the more complex successors like YOLOv6 to YOLOv8. Furthermore, its reliability, strong community support, and extensive research on YOLOv5 provide additional motivation for selecting this model. Additionally, the instability and frequent updates of newer versions, such as YOLOv9 and YOLOv10, make them less suitable for our study. By choosing YOLOv5, we aim to leverage its optimal performance and reliability. In the next chapter, we will detail our methodology, including the integration of new modules and a novel training strategy to develop an accurate, lightweight version of the model.

Chapter 4

Proposed Methodology

In this chapter, we introduce our proposed methodology, Pre-trained Weight Pruned Retrained by Salient Feature Guided Knowledge Distillation (PWPR-SFGKD), which integrates three key components to enhance model accuracy and performance: Weight Pruning, Knowledge Distillation, and a Lightweight Model. First, we apply Unstructured Weight Pruning, where a large portion of lower-magnitude weights are set to zero. This reduces model complexity and forces the model to fine-tune its parameters, rediscover important features, and improve generalization. After pruning, the model is retrained using Salient Feature Guided Knowledge Distillation (SFGKD), our second component. SFGKD transfers essential features from a larger, pre-trained teacher model to a smaller student model, focusing on the most informative features rather than output probabilities. To ensure stable knowledge transfer, we use the SmoothL1 Loss function. Finally, the third component of our methodology is the design of a Lightweight Model for deployment in resource-constrained environments. We propose the YOLOv5 Nano-Lite, a compact version of the YOLOv5 architecture optimized for edge devices like the Raspberry Pi. This lightweight model reduces computational and memory requirements, enabling real-time object detection while maintaining high performance in resource-limited settings.

4.1 Standard YOLOv5 Training Approach

In YOLOv5, the training process (ref Figure 4.1) involves three primary loss components: objective loss, classification loss, and bounding box loss. The objective loss measures the confidence of the model in predicting whether an object exists within a bounding box, using binary cross-entropy. The classification loss ensures the model

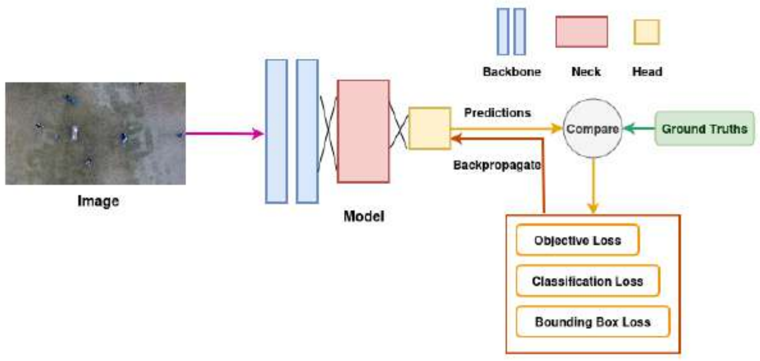


Figure 4.1: Conventional YOLO Training Strategy

correctly classifies objects by comparing predicted class probabilities to the true labels using categorical cross-entropy. The bounding box loss calculates the error between the predicted and ground truth bounding box coordinates, typically using CIoU or GIoU to penalize mismatches in position, size, and aspect ratio. These losses are combined into a total loss function, which is minimized during training through backpropagation, allowing the model to adjust its parameters and improve accuracy in detecting and classifying objects, while also localizing them precisely.

4.1.1 YOLOv5 Loss Function

The YOLO loss function consists of three losses bounding box loss L_{bbox}, classification loss L_{cls} and objective loss L_{obj} (ref: Eqn. 4.1).

$$L_{\text{yolov5}} = L_{\text{bbox}} + L_{\text{cls}} + L_{\text{obj}} \tag{4.1}$$

In YOLOv5, the box loss L_{bbox} is used to quantify the difference between the predicted and actual bounding box coordinates. This loss is defined as follows:

$$L_{bbox} = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} I_{ij}^{obj} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + (w_i - \hat{w}_i)^2 + (h_i - \hat{h}_i)^2 \right] \tag{4.2}$$

where λ_{coord} is a hyperparameter that controls the weight of the box loss, S is the number of grid cells in the output feature map, B is the number of bounding boxes predicted by each grid cell, I_{ij}^{obj} is an indicator function that is 1 if the j-th bounding box in the i-th grid cell is responsible for detecting an object, and 0 otherwise, x_i, y_i,

w_i, and h_i are the true bounding box coordinates, and $\hat{x}_i$, $\hat{y}_i$, $\hat{w}_i$, and $\hat{h}_i$ are the predicted bounding box coordinates. The objective loss L_{obj} checks whether there's an object at each anchor box. It helps the model learn to accurately predict by increasing the confidence for bounding boxes in the correct locations while reducing the confidence score for incorrect ones, ultimately leading to improved detection outcomes.

$$L_{obj} = \lambda_{obj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} I_{ij}^{obj} (1 - \hat{C}ij)^2 + \lambda noobj \sum_{i=0}^{S^2} \sum_{j=0}^{B} I_{ij}^{noobj} \hat{C}_{ij}^2 \tag{4.3}$$

The hyperparameters λ_{obj} and λ_{noobj} control the weights for the object and no-object loss terms, respectively. Here, C_{ij} represents the actual object confidence score, while $\hat{C}_{ij}$ is the predicted confidence score. The indicator function I_{ij}^{noobj} is set to 1 when the j-th bounding box in the i-th grid cell does not correspond to any object, and I_{ij}^{obj} is set to 1 when it does. Conversely, I_{ij}^{noobj} equals 0 when the box is responsible for detecting an object, and I_{ij}^{obj} equals 0 when it is not. The class loss (L_{cls}) measures the difference between the predicted class probabilities and the actual class labels using a binary cross-entropy loss for multi-class classification and defined as:

$$L_{cls} = -\lambda_{cls} \sum_{i=0}^{S^2} \sum_{j=0}^{B} I_{ij}^{obj} \sum_{c=0}^{C-1} (y_{ijc} \log(p_{ijc}) + (1 - y_{ijc}) \log(1 - p_{ijc})) \tag{4.4}$$

In this equation, the term λ_{cls} is a scaling factor for the class loss. The indicator variable I_{ij}^{obj} is equal to 1 if an object is present in the j-th bounding box of the i-th grid cell, and 0 otherwise. The term y_{ijc} is the ground truth for class c in the j-th bounding box of the i-th grid cell, which is 1 if the class matches and 0 otherwise. Lastly, p_{ijc} represents the predicted probability of class c for the j-th bounding box in the i-th grid cell.

4.2 Pre-trained Weight Pruned Retrained Strategy

In our research, we employed unstructured weight pruning not for model compression but as an initialization strategy to enhance the model's training process. Typically, unstructured pruning involves training a model for several epochs, then sorting its weights by magnitude and setting a pruning threshold to zero out all weights below this threshold. A mask is created to track the pruned weights to prevent them from affecting inference, and these weights are eventually removed. After pruning, the model is retrained for a few epochs to recover any lost accuracy. In contrast, our approach adapted this process for better initialization. Initially, the model was trained for a few epochs to stabilize the weights. Then, 80% of the weights were pruned, retaining only the highest-magnitude weights, allowing the model to focus on the most significant parameters while starting fresh. The model was pre-trained for 10 epochs, followed by retraining for the remaining 40 epochs out of a total of 50. This approach ensured that the majority of the training time was spent on recovering the accuracy, resulting in improved model performance. The full process of pruning with improved initialization, as applied to the YOLOv5 model, that is illustrated in Figure 4.2.

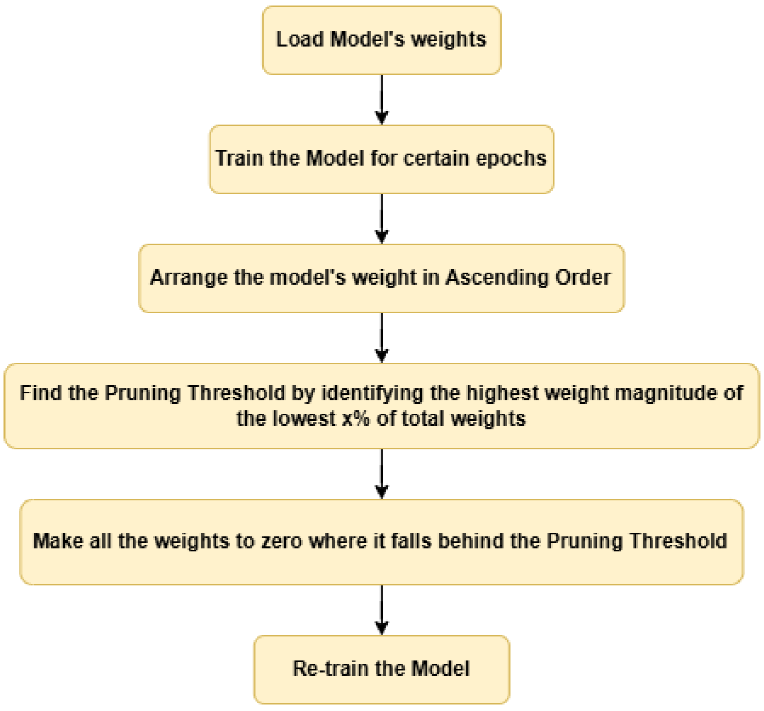

Load Model's weights
Train the Model for certain epochs
Arrange the model's weight in Ascending Order
Find the Pruning Threshold by identifying the highest weight magnitude of the lowest x% of total weights
Make all the weights to zero where it falls behind the Pruning Threshold
Re-train the Model

Figure 4.2: Block Diagram of the PWPR technique on a YOLO model

4.3 Knowledge Distillation (KD)

In Knowledge Distillation, the training of a smaller model (called the student model) is enhanced by leveraging insights from a larger, pre-trained model (the teacher model). Both models are fed the same input image, which allows them to generate their respective feature maps: the teacher's feature map and the student's feature map.

To guide the student in learning effectively, the difference between these feature maps are calculated using a loss function called Mean Squared Error (MSE). This difference is referred to as the distillation loss. The distillation loss is then combined with other loss components, such as the losses for bounding boxes, classification, and the overall objective. By incorporating this additional loss, the student model learns from the teacher model's knowledge, leading to a well-trained final model (see Figure 4.3). This approach ensures that the student model not only learns from its own predictions but also benefits from the teacher's deeper understanding, ultimately improving its performance.

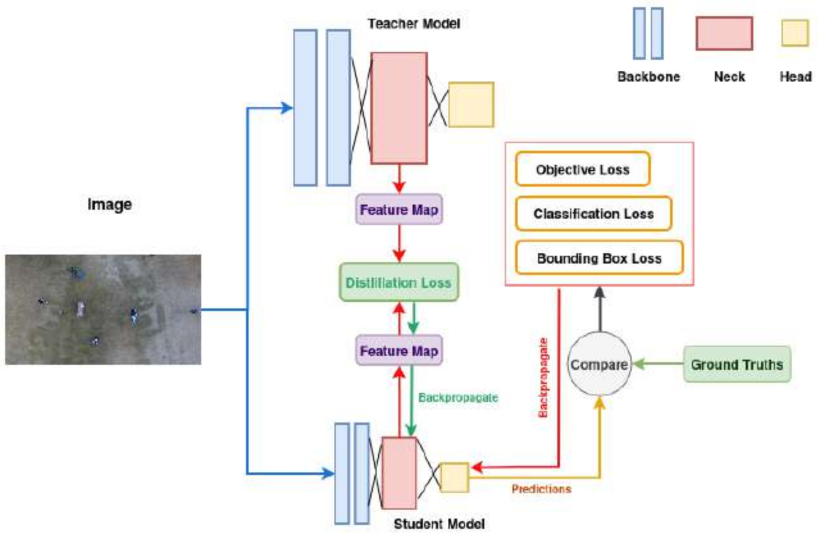


Figure 4.3: Conventional Knowledge Distillation Training Strategy

4.4 Salient Feature Guided Knowledge Distillation

A significant challenge in knowledge distillation is effectively transferring essential information from a larger teacher model to a smaller student model, especially when key objects or regions in an image are small. The student model, with its limited capacity, struggles to detect such features, unlike the over-parameterized teacher model. Simply replicating the teacher's feature map often fails to transfer deep knowledge to the student due to its reduced receptive capacity. Optimizing accuracy in the student model thus requires a more strategic focus on specific layers, but conventional feature-based distillation methods often overlook this crucial aspect. In our proposed approach, we introduce a novel knowledge distillation technique called Salient Feature Guided Knowledge Distillation (SFGKD), where we do not rely on directly transferring the teacher's feature map. Instead, we leverage the salient feature map produced by the teacher model, which is derived by subtracting the background image feature map from the full-image feature map (ref Figure 4.4). Our intuition is that this subtraction in the feature domain highlights key discriminative features, enhancing detection capabilities in the student model. In contrast, subtraction in the image domain removes important background information, leading to a feature map that does not provide significant improvements in accuracy.

Let (i, j) denote the coordinates or indices of a pixel in the image.

$$\text{foreground\_mask}(i, j) = \begin{cases} 1 & \text{if bounding box overlaps with mask region} \\ 0 & \text{otherwise} \end{cases} \tag{4.5}$$

$$\text{background\_image}(i, j) = (1 - \text{foreground\_mask}(i, j)) \times \text{image}(i, j) \tag{4.6}$$

To generate the feature maps, we input both the complete image and the background region into the model. To derive the background region from the full image, first we compute the foreground mask using Equation 4.5. Subsequently, by applying Equation 4.6, we obtain the background image.

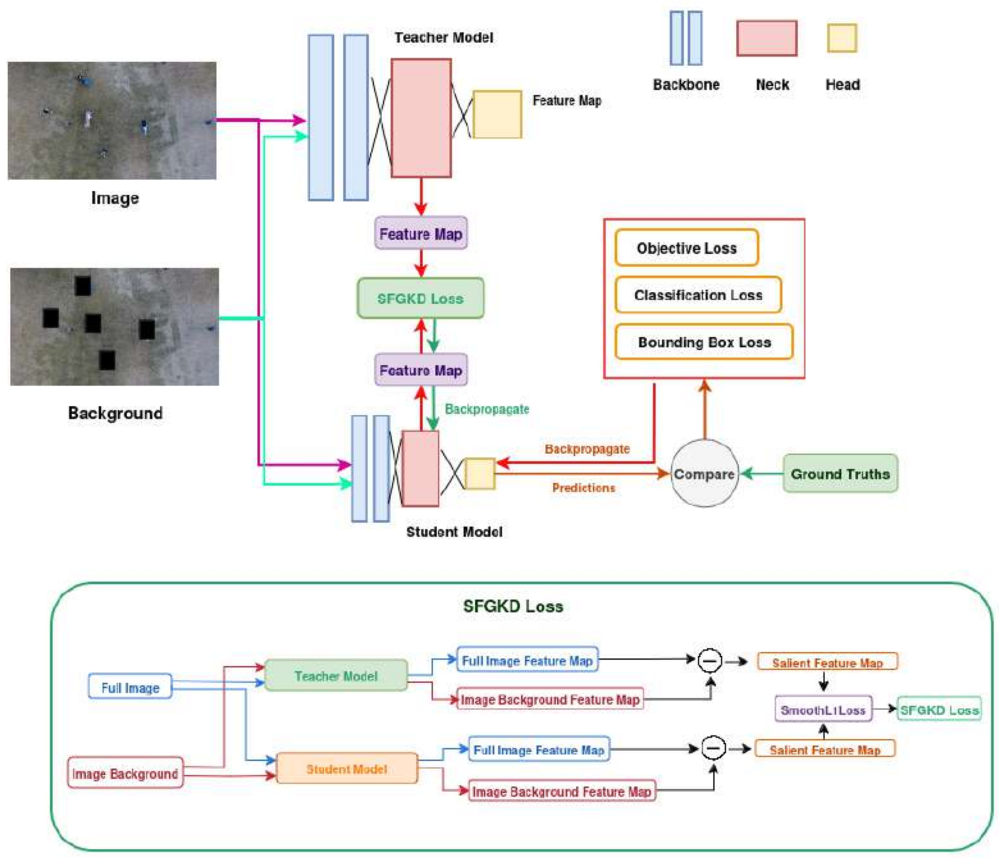


Figure 4.4: Block Diagram of Salient Feature Guided Knowledge Distillation for Object Detection

For the teacher model, we compute the differential feature map (teacher salient feature) F_{T} (Eqn 4.7) by subtracting the background feature map $F_{\text{T\_background}}$ from the full image feature map $F_{\text{T\_full}}$:

$$F_{\text{T}} = F_{\text{T\_full}} - F_{\text{T\_background}} \tag{4.7}$$

Similarly, for the student model, we obtain the differential feature map (student salient feature) F_{S} (Eqn 4.8) by subtracting the background feature map $F_{\text{S\_background}}$ from the full image feature map $F_{\text{S\_full}}$:

$$F_{\text{S}} = F_{\text{S\_full}} - F_{\text{S\_background}} \tag{4.8}$$

4.4.1 SFGKD Loss Function

The overall loss in the SFGKD is calculated by combining four losses (Eqn 4.9). Among them three are the default YOLOv5 losses: Bounding box loss (L_{bbox}), Class loss (L_{cls}) and Objective loss (L_{obj}) and Smooth L1 Loss as the knowledge distillation loss, (L_{SmoothL1}) (Eqn 4.10) to measure the discrepancy between the teacher and student salient feature maps.

$$L_{\text{total}} = L_{\text{bbox}} + L_{\text{cls}} + L_{\text{obj}} + L_{\text{SmoothL1}} \tag{4.9}$$

$$L_{\text{SmoothL1}}(F_{\text{T}}, F_{\text{S}}) = \begin{cases} 0.5 \cdot \frac{(F_{\text{T}} - F_{\text{S}})^2}{\beta} & \text{for } |F_{\text{T}} - F_{\text{S}}| < \beta \\ |F_{\text{T}} - F_{\text{S}}| - 0.5 \cdot \beta & \text{otherwise} \end{cases} \tag{4.10}$$

In conventional knowledge distillation methods, the difference between feature maps is typically minimized using L2 Loss (Mean Squared Error), which penalizes larger discrepancies more heavily. In this study, however, we experimented with SmoothL1 Loss, a hybrid loss function that combines characteristics of both L2 and L1 Loss. SmoothL1 Loss introduces a hyperparameter, β, which balances the influence of L1 and L2 behaviors, creating a smooth transition between them. By adjusting β (e.g., setting $\beta = 1$), we achieve a flexible loss function that leverages L2's stability for minor errors and L1's resilience to outliers. By incorporating this approach we prevent the potential chance of exploding gradient problems.

4.5 PWPR-SFGKD Training Strategy

In the PWPR-SFGKD approach, both the student and teacher models are loaded first. For the initial 10 epochs, the student model is trained using just the class loss, object loss, and bounding box loss—essentially sticking to the standard YOLOv5 training. After that, an unstructured pruning technique is applied, removing 80% of the student model's weights. This is achieved by determining a pruning threshold based on the lowest 80% of weights. Once the Pre-trained and Weight Pruning technique is successfully implemented, the student model is retrained using the SFGKD technique, where the distillation loss is combined with the three main losses, and the model is trained for the remaining epochs. For a visual understanding of the training process, please refer to Figure 4.5. A detailed workflow is presented in the flowchart of Figure 4.6.

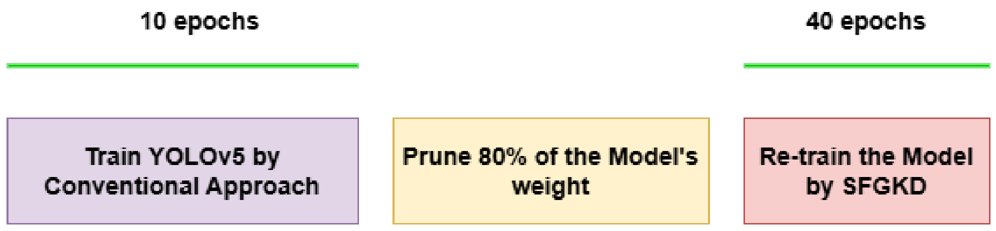

10 epochs
40 epochs
Train YOLOv5 by Conventional Approach
Prune 80% of the Model's weight
Re-train the Model by SFGKD

Figure 4.5: PWPR-SFGKD Training Strategy

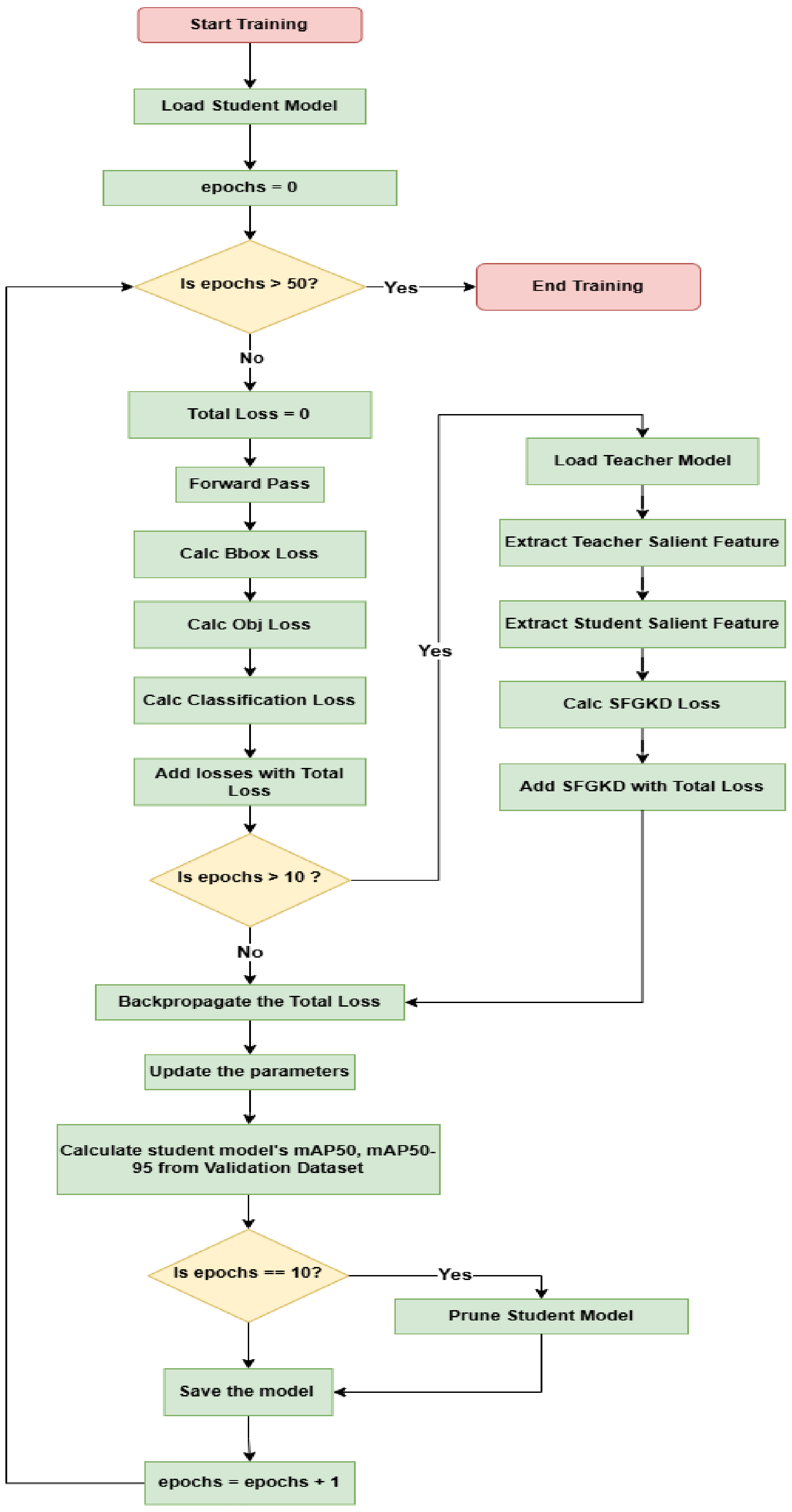


Figure 4.6: Flow chart of PWPR-SFGKD training strategy

4.6 YOLOv5 Nano Light

To create a lighter model based on the YOLOv5 architecture, we specifically focused on reducing some layers in the backbone and removing some feature maps which would not impact much in our case. Since the detected humans are much smaller in size relative to the entire image, We focused on optimizing for small objects by removing the feature maps, layers associated with medium and large object predictions, enhancing the model's ability to detect smaller objects effectively. We removed the last C3 bottleneck and ConvSiLU block along with the 20x20 size feature layer from the backbone to reduce the model's architecture. Similarly, we originally had three feature maps in the neck section—20x20x1024, 40x40x512, and 80x80x256—for detecting large, medium, and small objects. So we excluded the 20x20x1024 and 40x40x512 feature maps and their corresponding layers from the neck (see Figure 4.7, Figure 4.8). These adjustments led to a lighter and more efficient network, named YOLOv5-lite. By applying the as usual width and depth scaling factors used in YOLOv5n model, we created our new lightweight model, YOLOv5n-lite (see Chapter 3). This model features lower GFLOPS, higher frames per second (FPS), and a smaller size than YOLOv5n.

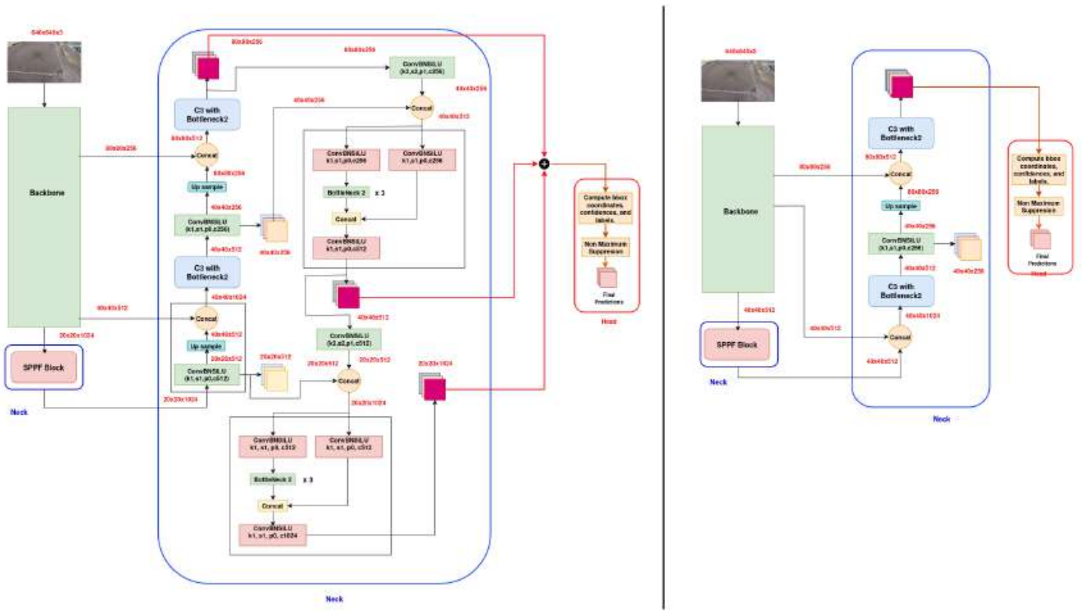


Figure 4.7: Comparison of YOLOv5 and YOLOv5-lite architecture

In the next chapter, we will explore the impact of applying various training strategies to the YOLOv5 nano and nano-lite models through a series of experiments. Additionally, we will conduct an in-depth analysis to uncover the potential causes of performance degradation, providing insights that can guide future research aimed at addressing these issues.

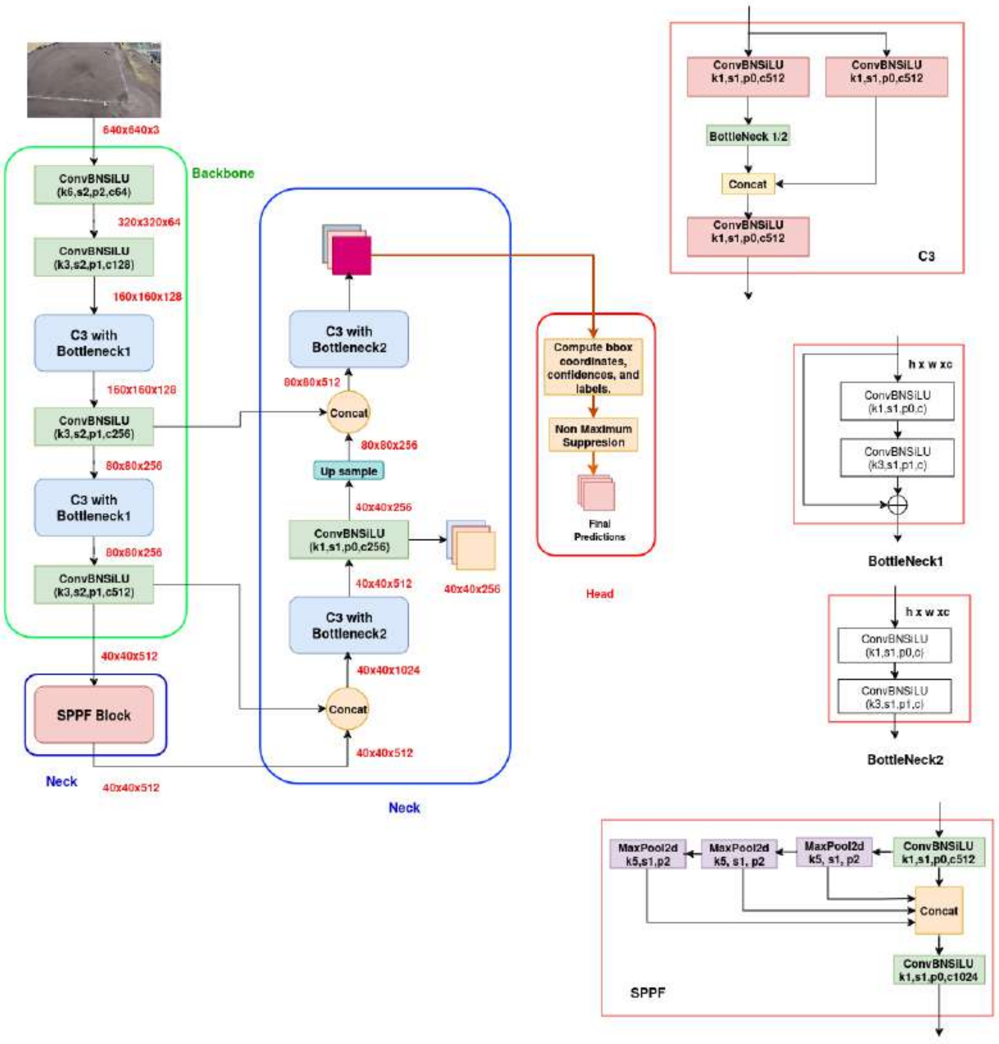


Figure 4.8: Block Diagram of YOLOv5-lite architecture

Chapter 5

Experiments & Results

In this chapter, we focus on the experiments conducted and the key insights derived from them. We provide a comprehensive overview of the dataset and the experimental procedures, accompanied by relevant graphs and tables. By conducting an in-depth analysis, we intend to interpret these findings and pave the way for more extensive discussions and future research directions. Step by step, we will explore our data processing techniques and experimental approaches while addressing potential reasons for performance degradation and the limitations of the proposed methods across YOLOv5n nano and nano-lite models.

5.1 Data Preparation

Data processing is a critical step in preparing a dataset for DNN. The first duty is to choose the dataset. Once the dataset is chosen, then data needs to be extracted, appropriately formatted and filtered. In the upcoming sections, we will discuss how we chose the dataset and formatted and filtered to ensure the dataset's quality and reliability.

5.1.1 Selection of the Dataset

In our research, we focused on recognizing basic human actions, especially in scenarios where the person occupies a small portion of the image, such as when viewed from an elevated or aerial perspective. Capturing detailed information about a person's actions can be challenging from higher vantage points, where individuals may appear quite small relative to the entire scene. Our primary objective was to accurately identify simple but critical actions—such as walking, running, sitting, standing, and lying down—particularly in environments like disaster zones or security-sensitive areas. For instance, detecting actions like running, walking, or standing in a restricted area can signal potential security risks. In the context of disaster response, recognizing actions such as large groups of people running or walking can indicate an ongoing evacuation or rescue operation. Conversely, if a person is detected sitting or lying on the ground, it could suggest that they are in distress, injured, or unable to move. For example, in an emergency or disaster scenario, a person lying on the ground may require immediate attention, signaling a need for rescue or medical assistance. In elder care, the detection of a person lying on the floor can be particularly important, as it may indicate a fall, which requires swift intervention. Early detection of such actions can significantly improve response times and outcomes in both security and healthcare settings, helping to ensure the safety and well-being of individuals in critical situations.

We explored several existing datasets related to human action recognition. For example, the Penn Action Dataset contains 2,326 video clips of 15 different actions (such as baseball swing, jumping jacks, push-ups, golf swings, etc.), along with detailed annotations of human body joint positions. The UTD-MHAD (University of Texas at Dallas Multimodal Human Action Dataset) includes 27 action classes and utilizes multiple types of data, including RGB video, depth data, and skeleton data. The Something-Something V2 Dataset contains 174 action categories with over 318,000 annotations, illustrating how people interact with objects in various ways. Unfortunately, these datasets did not capture the specific actions of interest, such as walking, running, sitting, standing, and lying down. Moreover, even when similar actions were present, the datasets lacked bounding box annotations, as they were primarily annotated with key points and skeleton data. After further investigation, we identified three other potentially useful datasets for our purpose: Drone Action, UAV-Human, and Okutama Action.

The Drone Action Dataset includes 13 action categories, such as clapping, punching, walking, jogging, running, etc. While this dataset provides valuable insights into human actions, the majority of the images are only labeled with the action category, and only a few include bounding box annotations to specify the person's location within the image. This limitation makes it challenging to precisely detect and track individuals, especially in situations where spatial context is important. Similarly, the UAV-Human Dataset covers a broader range of 155 action categories, including walking, reading a book, mowing the lawn, talking on the phone, etc. However, like the Drone Action Dataset, most images are annotated solely with action categories, without any bounding box annotations. This lack of spatial data complicates tasks like person localization and tracking, which are critical for surveillance, disaster response, or elder care applications. In both datasets, the absence of bounding box annotations creates a significant barrier for detailed action detection and tracking, particularly in complex real-world scenarios where both the recognition of actions and the accurate localization of individuals are essential for effective intervention or analysis.

In the end, we decided to use the Okutama Action Dataset [53] for our research. The Okutama Action dataset is a comprehensive and reliable resource developed in collaboration with the National Institute of Informatics in Tokyo, Japan. It encompasses 12 action categories, including handshaking, hugging, reading, drinking, pushing/pulling, carrying, calling, running, walking, lying, sitting, and standing. The dataset contains nearly 54,000 images, each annotated with bounding boxes for various human actions. With its diverse collection of images captured from different angles and altitudes, and featuring a range of actors, the dataset is well-suited for our requirements. Additionally, its widespread use in numerous research studies affirms its quality and reliability.

5.1.2 Data Pre-processing

Our data processing workflow began by organizing the dataset, which consisted of two separate folders: one containing extracted frames from a video, and the other holding bounding box annotations with labels corresponding to specific frame numbers. To meet the input requirements of the YOLO model, we generated separate annotation files for each image, allowing us to provide the necessary bounding box coordinates and class labels in the format expected by the model. We converted the raw annotation format $(l, x_{\text{min}}, y_{\text{min}}, x_{\text{max}}, y_{\text{max}})$ into the YOLO format $(l, x_{nc}, y_{nc}, w_{n_{\text{bbox}}}, h_{n_{\text{bbox}}})$. Here, $(x_{\text{min}}, y_{\text{min}})$ represents the top-left corner of the bounding box, while $(x_{\text{max}}, y_{\text{max}})$ corresponds to the bottom-right corner. The label (l) indicates the detected human action. To convert to the YOLO format, we recalculated the bounding box center coordinates and dimensions, normalizing them relative to the image size. The normalized values include the bounding box center's x-coordinate (x_{nc}), y-coordinate (y_{nc}), and the normalized width $(w_{n_{\text{bbox}}})$ and height $(h_{n_{\text{bbox}}})$ of the bounding box. These values were normalized based on the dimensions of each image (1280x720 pixels).

As part of the preprocessing step all images were resized to a consistent dimension of 640x640 pixels, just before sending it to model ensuring uniform input size for the model. In our dataset there were 47,430 training, 11,186 validation and 16,402 test images.

5.2 PC Configuration

The experiments were primarily conducted on a high-performance desktop featuring an Intel Core i7-12700K processor (with a base clock speed of 3.6 GHz and turbo boost up to 5.0 GHz), dual NVIDIA GeForce RTX 3060 GPUs, and 64 GB of DDR4 RAM, running Ubuntu 23.04. This setup provided robust computing power, enhanced graphical performance, and seamless compatibility with required software tools, making it ideal for resource-intensive tasks such as model training and execution. Both the Raspberry Pi 4B+ and the laptop were used to calculate the model's FPS during real-time video processing. The Raspberry Pi, equipped with a 1.4 GHz quad-core CPU, 16 GB of RAM, and versatile connectivity options like Wi-Fi, Bluetooth, and Gigabit Ethernet, served as a compact edge device for FPS calculation. Similarly, the laptop, with an AMD Ryzen 5 5600H processor (3.30 GHz) and 16 GB of RAM, was also utilized for FPS calculations on real-time video, allowing for cross-device performance analysis.

5.3 Experiments & Results

We explored three distinct training strategies. The first strategy is the conventional YOLO training approach. The second strategy is the PWPR Method, which involves initially training the model for 10 epochs, followed by weight pruning, and then retraining for the remaining epochs. The third strategy is the PWPR-SFGKD Method. This approach follows the same steps as PWPR, but incorporates SFGKD during the retraining phase. In this technique, key feature maps are extracted from both the teacher and student models. These feature maps are then used to calculate a distillation loss, which is added to the student's bounding box loss, objectness loss, and classification loss. Afterwards, the combined loss is back-propagated through the network, and the optimizer updates the model's weights and biases. This cycle of forward pass, loss calculation, back-propagation, and weight updates continues throughout the entire training process, helping the student model learn effectively from the teacher model.

Our first experiment was the comparision of YOLOv5n-lite with YOLOv5n model. The results in Table 5.1 show that our proposed light architecture YOLOv5n-light model outperforms the standard YOLOv5n model, achieving a 1.4% improvement in mAP comparing to conventional YOLOv5n training strategy. Additionally, the YOLOv5n-lite model is more compact and has lower computational requirements (FLOPs) compared to the YOLOv5n model.

In our second experiment, we tested the effect of the PWPR technique by applying

Experiment Details	**mAP-50**	**mAP 50-95**	**Epochs**	**Img Size**	**Model Size (MB)**	**FLOPs (Billions)**	**mAP Improvement (%)**
Base Model							
YOLOv5n Conventional Training	0.643	0.324	50	640px	3.8	4.2	-
Compared Model							
YOLOv5n-lite Conventional Training	**0.652**	0.319	50	640px	0.9	2.8	1.4

Table 5.1: Comparison of YOLOv5n-lite model with YOLOv5n model

pruning of different percentages of weights on the YOLOv5n model. We began with normal YOLO training for 10 epochs, then pruned 70%, 80%, and 90% of the model's weights, followed by retraining for the rest of the epochs. Weight pruning works by setting certain weights to zero based on a threshold determined by the pruning percentage.

Experiment Details	**mAP 50**	**mAP 50-95**	**Epochs**	**Img Size**	**Model Size (MB)**	**FLOPs (Billions)**	**mAP Improvement**
Base Model							
YOLOv5n Conventional Training	0.643	0.324	50	640	3.8	4.2	
Compared Models							
PWPR YOLOv5n (10 Epochs Pre-trained, 70% Pruned, 40 Epochs Re-trained)	0.693	0.375	50	640	3.8	4.2	7.77%
PWPR YOLOv5n (10 Epochs Pre-trained, 80% Pruned, 40 Epochs Re-trained)	**0.697**	**0.378**	50	640	3.8	4.2	8.4%
PWPR YOLOv5n (10 Epochs Pre-trained, 90% Pruned, 40 Epochs Re-trained)	0.696	0.376	50	640	3.8	4.2	8.24%

Table 5.2: Comparative results of applying different weight pruning on YOLOv5n model

The experiments presented in Table 5.2 demonstrate that applying weight pruning to the YOLOv5n model leads to a significant increase in mAP value compared to conventional training, with an increase in the range of 7.77% to 8.4%. Notably, we observed the highest mAP-50 performance improvement with 80% weight pruning.

Experiment Details	**mAP 50**	**mAP 50-95**	**Epochs**	**Img Size**	**Model Size (MB)**	**FLOPs (Billions)**	**mAP Difference**
Base Model							
YOLOv5n-lite Conventionally Trained Model	0.652	0.319	50	640	0.9	2.8	
Compared Models							
YOLOv5n-lite C3-Attention Conventionally Trained Model	**0.645**	0.313	50	640	0.9	2.9	**-1.07%**
PWPR YOLOv5n-lite (10 Epochs Pre-trained, 80% Pruned, 40 Epochs Re-trained)	**0.661**	0.328	50	640	0.9	2.8	**1.38%**

Table 5.3: Comparative results of applying weight pruning on YOLOv5n-lite model

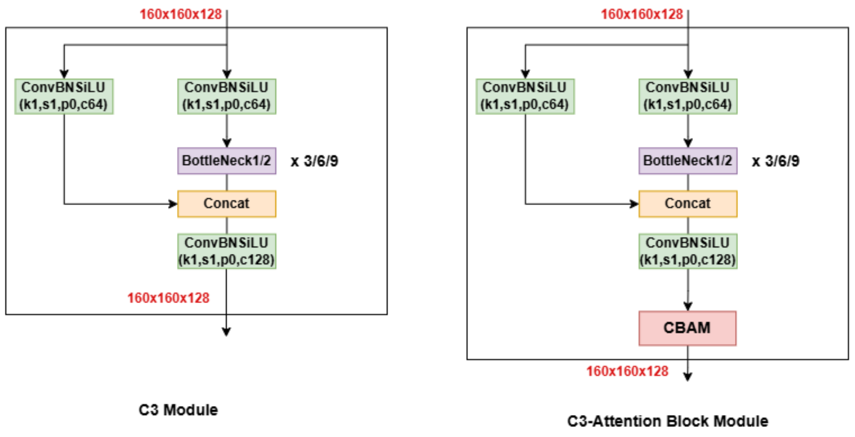

160x160x128
ConvBNSiLU (k1,s1,p0,c64)
ConvBNSiLU (k1,s1,p0,c64)
BottleNeck1/2
x 3/6/9
Concat
ConvBNSiLU (k1,s1,p0,c128)
160x160x128
C3 Module
160x160x128
ConvBNSiLU (k1,s1,p0,c64)
ConvBNSiLU (k1,s1,p0,c64)
BottleNeck1/2
x 3/6/9
Concat
ConvBNSiLU (k1,s1,p0,c128)
CBAM
160x160x128
C3-Attention Block Module

Figure 5.1: C3 and C3 Attention Block Module

In the third stage of our experiments, we focused on the YOLOv5n-lite and YOLOv5n-lite-attention models. The YOLOv5n-lite-attention model is a variation of YOLOv5n-lite, where the C3 block is replaced with the C3+CBAM block, which we refer to as the C3-Attention block (see Figure 5.1). The primary reason for choosing the YOLOv5n-lite-attention model is to incorporate attention mechanisms into the outputs of the convolutional blocks, aiming to boost the lightweight model's prediction accuracy. The model applies attention both spatially and channel-wise using the CBAM module. Channel-wise attention helps identify the most important feature channels, while spatial attention highlights the most relevant regions within those channels. By combining these two attention types, CBAM enhances the model's ability to focus on both what features are important and where they are located, ultimately improving its performance. The CBAM module enhances feature representation through a two-step process. First, the Channel Attention mechanism identifies important feature channels by using global average and max pooling across the spatial dimensions. The pooled outputs are then processed through a shared multi-layer perceptron (MLP) to create attention weights, which are applied to the feature map to highlight the most significant channels. Next, the Spatial Attention mechanism targets key areas within the feature map. It performs average and max pooling across the channel dimension, producing two 2D maps that are combined and sent through a convolutional layer to generate a spatial attention map. This map is used to emphasize crucial regions in the feature map (ref: Figure 5.2). We initially expected that incorporating CBAM with the C3 layer would improve the model's prediction performance, but this approach did not deliver the desired results. Even with combining CBAM to YOLOv5n-lite model the overall mAP decreased by 1.07% (ref: Table 5.3). To further assess the capabilities of the YOLOv5n-lite model, we applied the PWPR strategy. We began by training the model for 10 epochs, followed by pruning 80% of its weights, and then retraining it for an additional 40 epochs. The implementation of PWPR on the YOLOv5n-lite model resulted in a significant improvement in the mAP-50 score, achieving a 1.38% increase compared to the conventional YOLOv5n-lite model (see Table 5.3). This suggests that even though the attention mechanism harmed performance in the nano-light model, the PWPR strategy brought about a substantial enhancement in the mAP score.

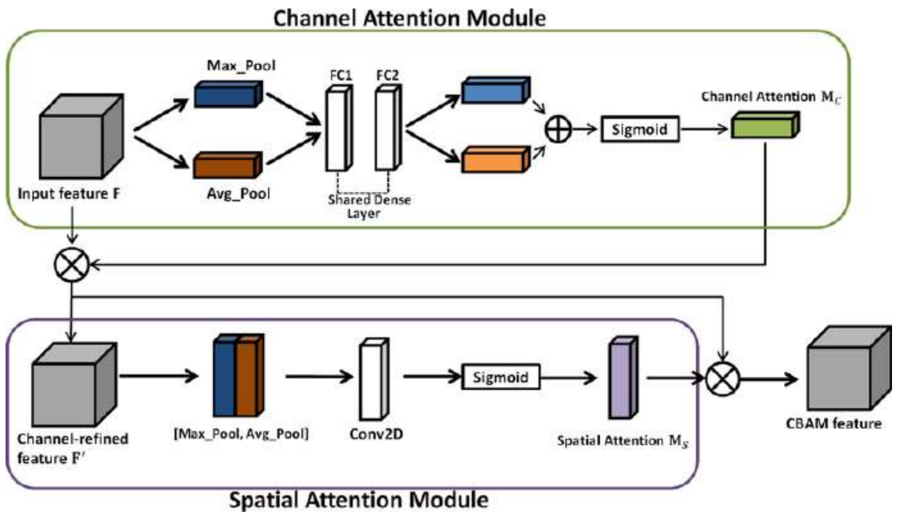


Figure 5.2: Convolutional Block Attention Module (CBAM)

In the fourth experiment, we applied our SFGKD method directly to the YOLOv5n-lite model, utilizing two different loss functions: MSE loss and SmoothL1 loss. The

Experiment Details	**mAP50**	**mAP50-95**	**Epochs**	**mAP Improvement (%)**
YOLOv5n-lite Conventional Training	0.532	0.220	10	-
YOLOv5n-lite SFGKD (MSE Loss)	0.557	0.227	10	**4.70%**
YOLOv5n-lite Conventional Training	0.617	0.292	25	-
YOLOv5n-lite SFGKD (MSE Loss)	0.583	0.259	25	-5.51%
YOLOv5n-lite SFGKD (SmoothL1 Loss)	0.606	0.281	25	**-1.78%**
YOLOv5n-lite Conventional Training	0.652	0.319	50	-
YOLOv5n-lite SFGKD (SmoothL1 Loss)	0.638	0.313	50	**-2.15%**

Table 5.4: Comparative result of applying SFGKD on YOLOv5n-lite with different losses and epochs

findings, illustrated in Table 5.4 , revealed several interesting insights. Firstly, we observed that SFGKD was effective in the early stages of training. However, as the model began to saturate in performance, SFGKD struggled to match the results of the traditional training approach. Although SFGKD initially demonstrated a 4.70% increase in mAP (compared to yolov5n-lite conventional training), after 10 epochs, it started to lag behind, unable to sustain the same mAP improvement in the middle and end stage of training. During the middle stage (25 epochs) the SFGKD with MSE loss had 5.51% lower mAP score compared to 25 epochs conventionally trained YOLOv5n-lite model. While exploring the effect of MSE and SmoothL1 loss from Table 5.4 fourth and fifth row revealed that SmoothL1 loss gives higher mAP score compared to MSE loss function during SFGKD technique. This proves for feature map distillation loss calculation SmoothL1 loss yields higher mAP score compared to MSE loss.

So far, we have observed that while SFGKD alone does not significantly enhance the accuracy of the nano-model, it positively influences loss reduction during the early stages of training. Therefore, in our PWPR training strategy, to re-train the model after pruning we chose the SFGKD technique rather than conventional training strategy. In our fifth experiment, we combined SFGKD with the PWPR on YOLOv5n model to assess their synergistic effects. This approach integrated traditional bounding box, classification, and objectiveness losses with a noble feature distillation loss, calculated using Smooth L1 loss. As shown in Table 5.5, this combination resulted in a 1.72% increase in the mAP-50 score compared to the PWPR conventionally re-trained YOLOv5n model

Experiment Details	**mAP 50**	**mAP 50-95**	**Epochs**	**Img Size**	**Model Size (MB)**	**FLOPs (Billions)**	**mAP Improvement**
Base Model							
PWPR YOLOv5n (10 Epochs Pre-trained, 80% Pruned, 40 Epochs Re-trained)	**0.697**	0.378	50	640	3.8	4.2	
Compared Model							
PWPR-SFGKD YOLOv5n (10 Epochs Pre-trained, 80% Pruned, 40 Epochs Knowledge Distillation)	**0.709**	0.397	50	640	3.8	4.2	**1.72%**

Table 5.5: Comparative Results of applying Salient Feature Guide Distillation on weight pruned YOLOv5n model

Experiment Details	**mAP 50**	**mAP 50-95**	**Epochs**	**Img Size**	**Model Size (MB)**	**FLOPs (Billions)**	**mAP Improvement**
Base Model							
Fine-Grained Feature Immitation based Knowledge Distillation on YOLOv5n	**0.647**	0.331	50	640	3.8	4.2	
Compared Model							
PWPR-SFGKD YOLOv5n (10 Epochs Pre-trained, 80% Pruned, 40 Epochs Knowledge Distillation)	**0.709**	0.397	50	640	3.8	4.2	**9.58%**

Table 5.6: Comparative Results of PWPR-SFGKD YOLOv5n model with Fine Grained Feature Immitation based Knowledge Distillation

alone, demonstrating that PWPR-SFGKD surpasses the standard PWPR training approach. To further evaluate its effectiveness, we compared it to the Fine-Grained Feature Imitation-based Knowledge Distillation (FGFI-KD) method [54], which enables a student model to replicate the feature responses of a teacher model at key object locations. When comparing PWPR-SFGKD with FGFI-KD on YOLOv5n, we observed a 9.58% improvement in mAP-50 (see Table 5.6). However, applying PWPR-SFGKD to the YOLOv5n-lite model resulted in a 4.54% decrease in mAP-50 (Table 5.7).

Experiment Details	mAP 50	mAP 50-95	Epochs	Img Size	Model Size (MB)	FLOPs (Billions)	mAP Difference
Base Model							
PWPR YOLOv5n-lite (10 Epochs Pre-trained, 80% Pruned, 40 Epochs Re-trained)	**0.661**	**0.328**	50	640	0.9	2.8	
Compared Model							
PWPR-SFGKD YOLOv5n-lite (10 Epochs Pre-trained, 80% Pruned, 40 Epochs Knowledge Distillation)	**0.631**	**0.296**	50	640	0.9	2.8	-4.54%

Table 5.7: Comparative Results of applying Salient Feature Guide Distillation on weight pruned YOLOv5n-lite model

5.4 Loss vs. Epochs Analysis

We have conducted a series of experiments on two models: YOLOv5n and YOLOv5n-lite, with the primary difference being the training strategies employed. In this section, we will compare the loss vs. epochs curves for both models using PWPR and PWPR—SFGKD techniques. The goal is to identify the underlying factors contributing to performance improvements or degradations in each case.

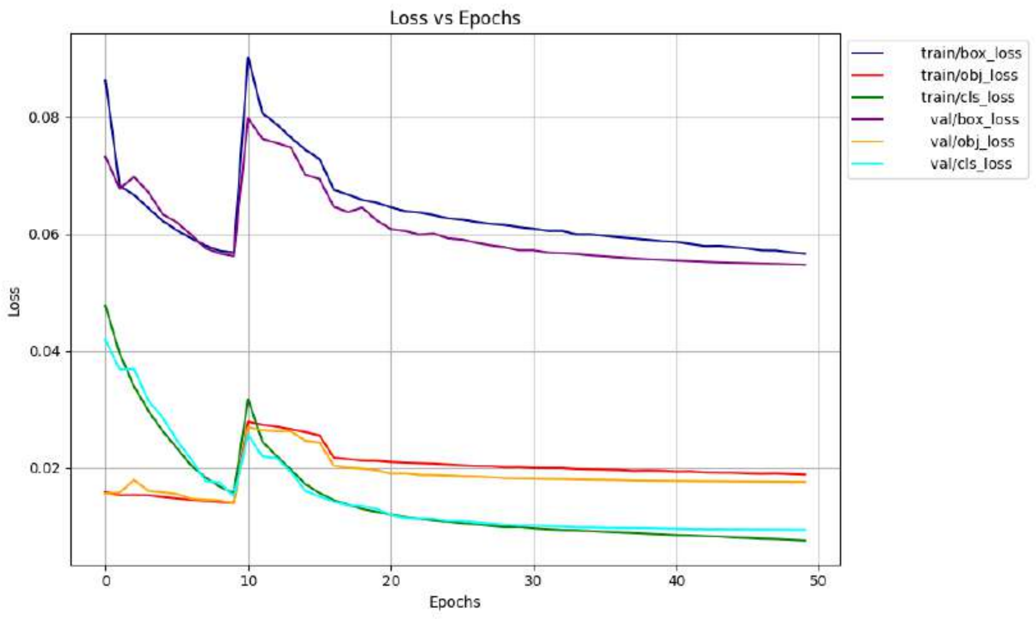


Figure 5.3: Training vs Validation loss of PWPR YOLOv5n model

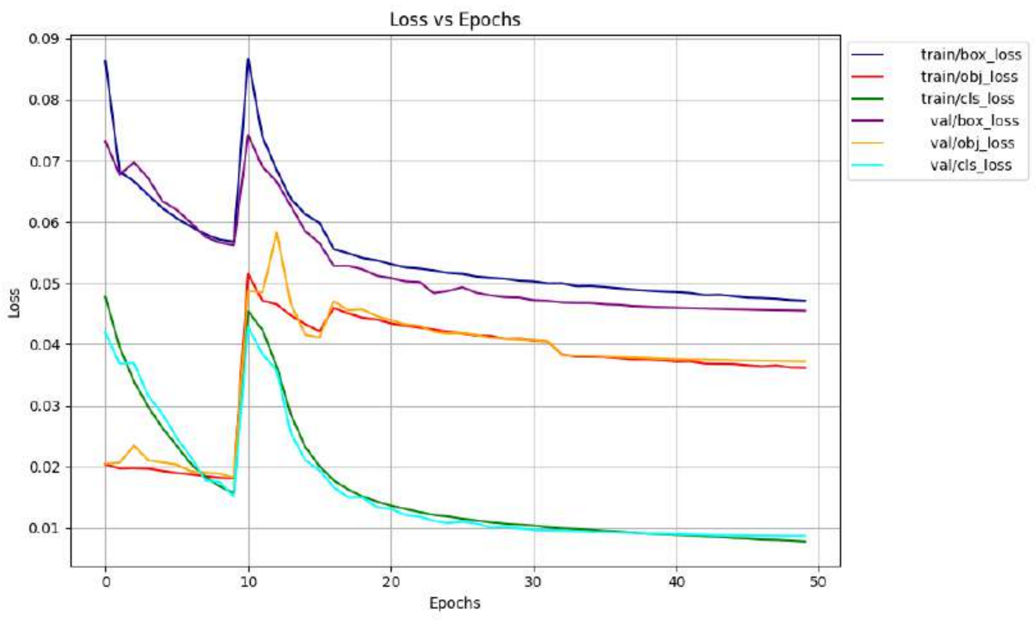


Figure 5.4: Training vs Validation loss of PWPR YOLOv5n-lite model

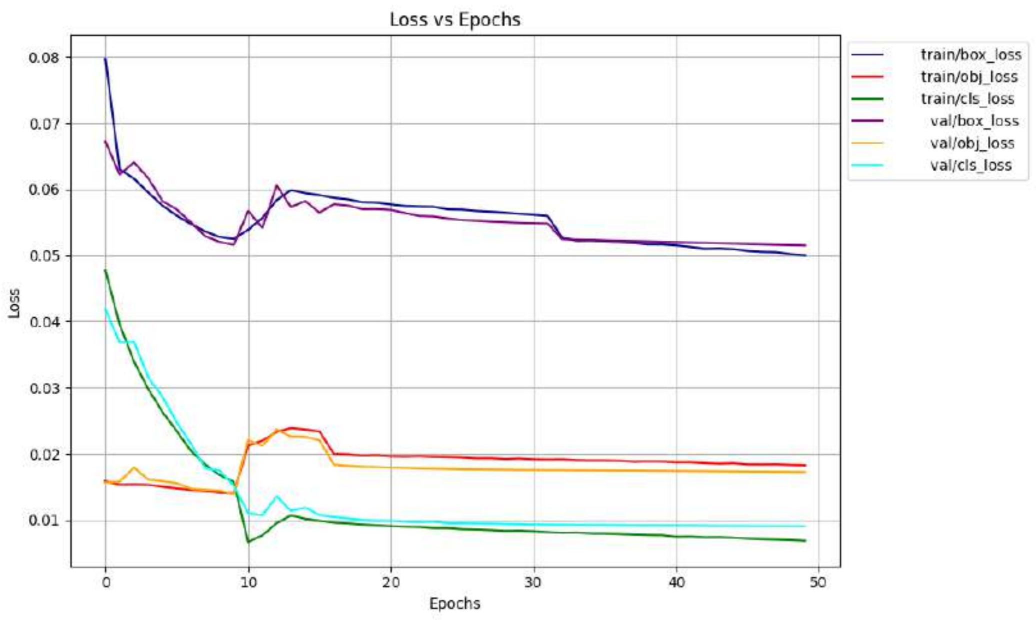


Figure 5.5: Training vs Validation loss of PWPR-SFGKD YOLOv5n model

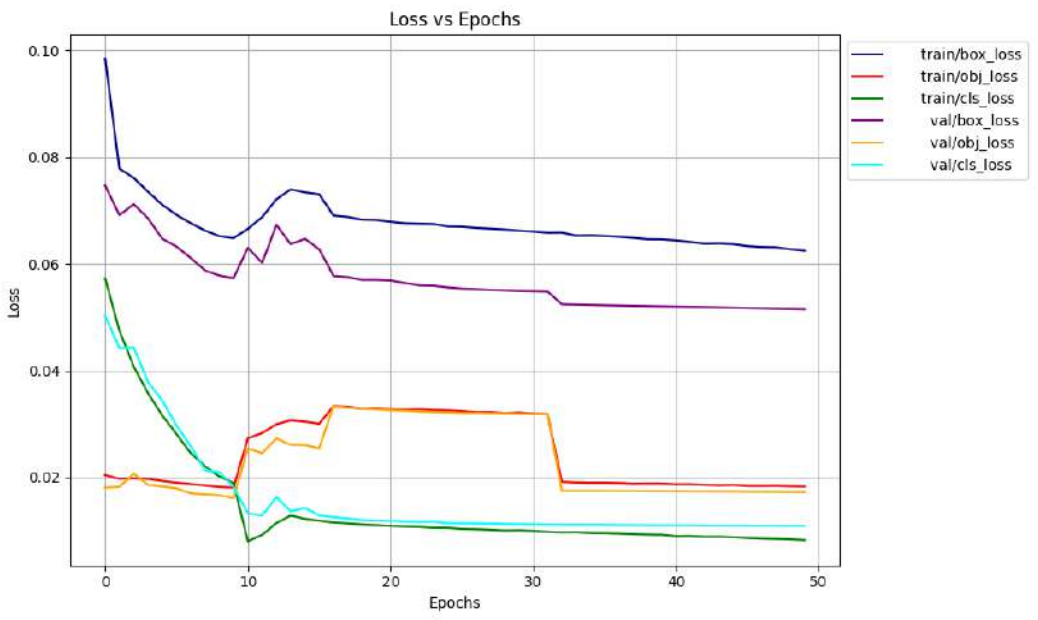


Figure 5.6: Training vs Validation loss of PWPR-SFGKD YOLOv5n-lite model

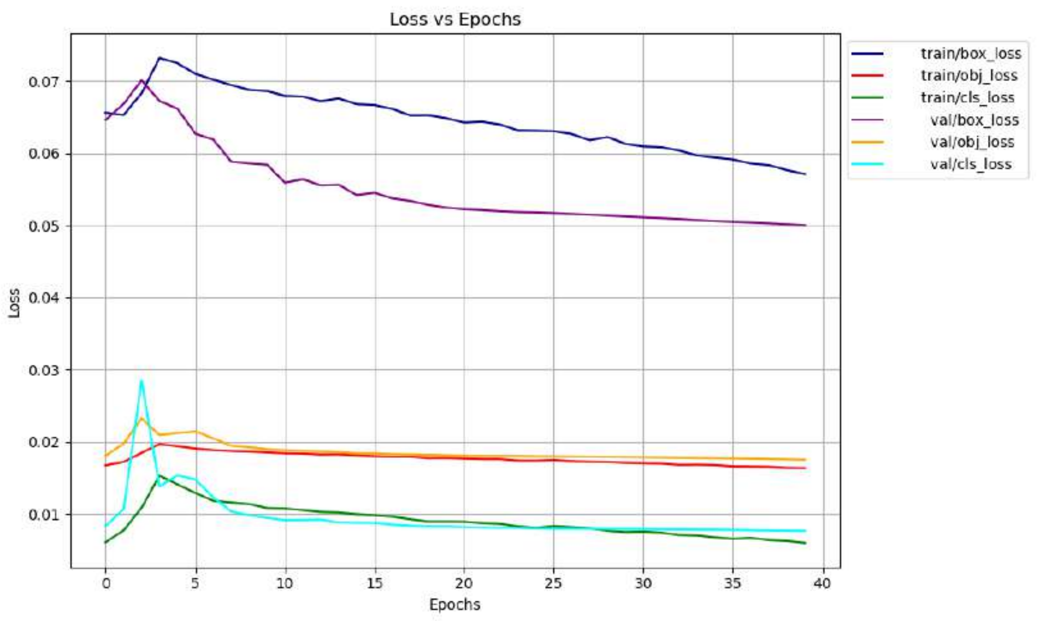


Figure 5.7: Training vs validation loss of Fine-Grained Feature Imitations based Knowledge Distillation technique

Our experiments revealed that the YOLOv5n-lite model exhibited higher objective loss compared to the YOLOv5n model, as shown in Figures 5.4 and 5.6. Moreover, when we combined the PWPR technique with knowledge distillation (PWPR-SFGKD), the nano-lite model's training and validation losses increased, as depicted in Figures 5.3 and 5.4. This suggests that the higher objective loss of the nano-lite model hindered its accuracy compared to the nano model. Furthermore, the PWPR-SFGKD technique exacerbated the difference between training and validation losses, leading to increased bounding box and objective losses, which negatively impacted the lite model's performance.

In summary, our analysis of the loss curves revealed that each technique exhibited distinct loss patterns. However, we observed a consistent trend: a reduction in bounding box and objective losses, accompanied by a smaller gap between training and validation losses, correlated with an increase in the mAP-50 score. Conversely, a decrease in the mAP score was associated with a widening gap between training and validation losses, as well as increased bounding box and objective losses, compared to other techniques.

5.5 Classification Error Analysis

During the testing pipeline, a normalized multi-class confusion matrix is generated to evaluate the classification performance of the models. The matrix's diagonal elements represent correct predictions, while the last column and bottom row indicate false positives and false negatives, respectively. The off-diagonal values show incorrect predictions where the labels are wrong but the bounding boxes are accurate. By examining the confusion matrices for the PWPR-SFGKD YOLOv5n and PWPR YOLOv5n-lite models (Figure 5.9), we can see the classification performance of each model. To gain a deeper understanding of the differences between the two models, we calculated a differential confusion matrix by subtracting the PWPR YOLOv5n-lite matrix from the PWPR-SFGKD YOLOv5n matrix (Figure 5.10). The differential confusion matrix reveals that the PWPR-SFGKD YOLOv5n model generally outperforms the PWPR YOLOv5n-lite model, with more correct predictions and fewer misclassifications. This is evident from the higher values along the diagonal and lower values in the off-diagonal elements of the differential matrix.

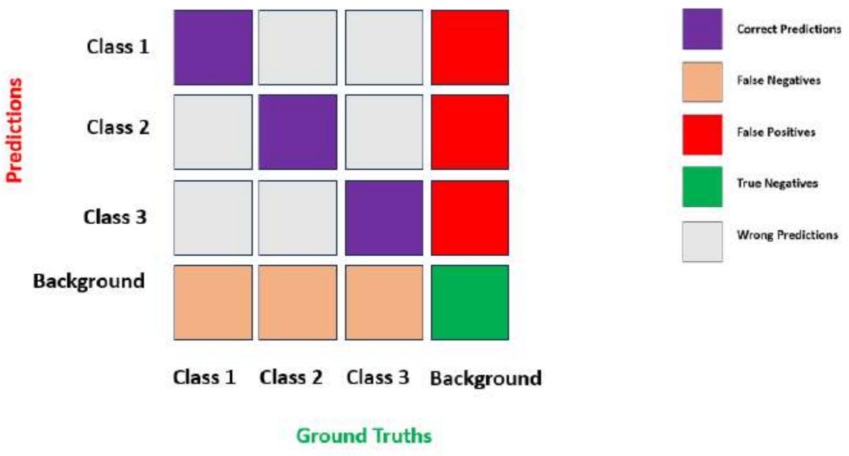


Figure 5.8: Components of Confusion Matrix

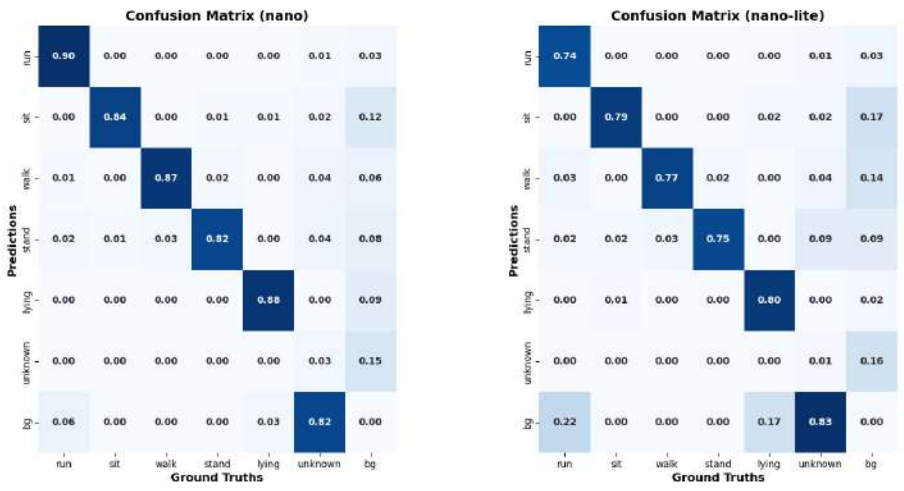


Figure 5.9: Confusion Matrix of PWPR-SFGKD YOLOv5n model and PWPR YOLOv5n-lite model.

Our comparative analysis of the YOLOv5n and YOLOv5n-lite models revealed distinct performance characteristics. The YOLOv5n model exhibited a higher tendency to generate false positives, whereas the YOLOv5n-lite model struggled with missed predictions. In terms of training efficiency, the SFGKD technique proved to be more computationally intensive, requiring greater GPU resources and longer training times. In contrast, the PWPR YOLOv5n-lite model demonstrated superior real-time performance, achieving higher frames per second (FPS) on both Raspberry Pi and laptop devices (see Tables 5.8 and 5.9). A thorough visual examination of the models' performance on diverse images revealed four distinct patterns: (1) similar predictions, (2) similar but not identical predictions, (3) inconsistent predictions on varying backgrounds, and

(4) failure to make predictions. Notably, the models occasionally produced inconsistent results when confronted with images featuring different contexts and backgrounds, underscoring the need for further refinement and improvement.

Experiment Details	**Batch Size**	**Single Epoch Runtime**	**GPU Memory Used**
YOLOv5 SFGKD	36	21 min	9.63 GB
YOLOv5n-lite SFGKD	36	19 min	8.66 GB
YOLOv5 PWPR	36	3.5 min	4.06 GB
YOLOv5n-lite PWPR	36	2.5 min	3.68 GB

Table 5.8: Runtime and GPU Memory Consumption for Various YOLOv5 Training Methods During a Single Epoch

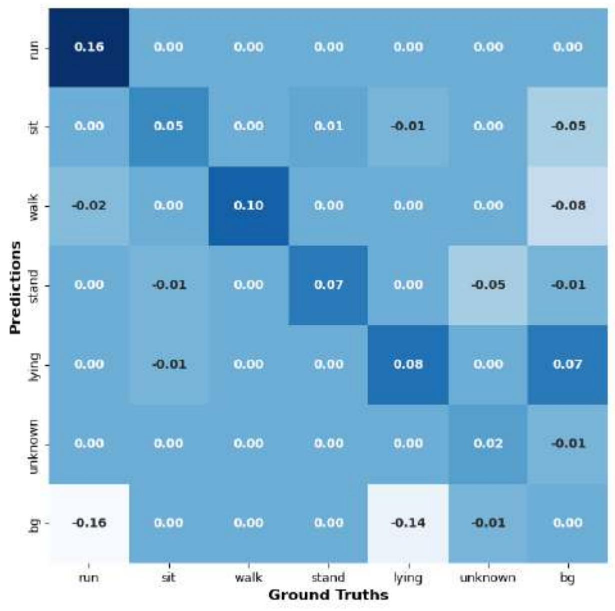
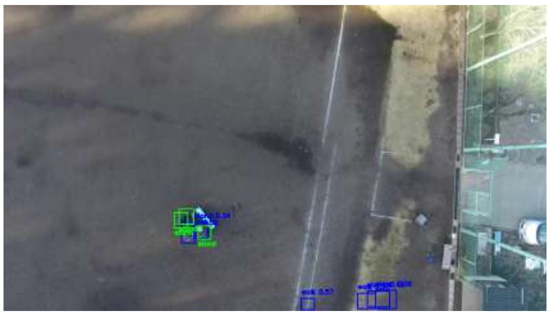
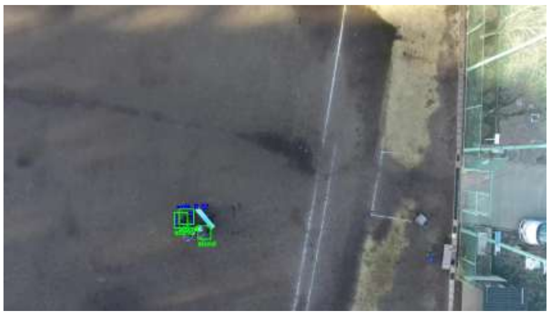


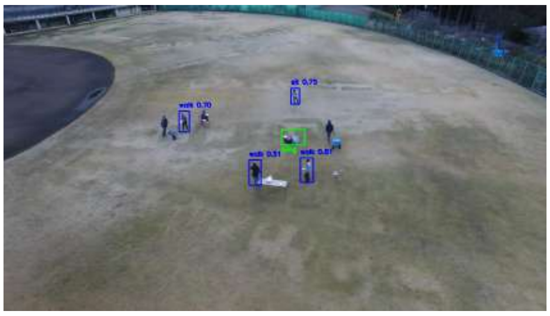
Figure 5.10: Difference in Confusion Matrix of PWPR-SFGKD YOLOv5n and PWPR YOLOv5n-lite.

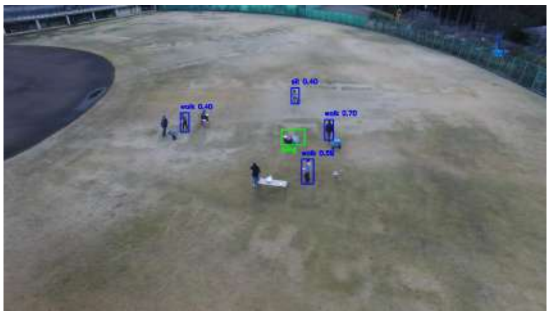
Experiment Details	**Device**	**Image Size (pixels)**	**FPS (Frames Per Second)**
PWPR YOLOv5n-lite	Raspberry Pi	640	2.07
PWPR-SFGKD YOLOv5n	Raspberry Pi	640	1.63
PWPR YOLOv5n-lite	Laptop	640	21.10
PWPR-SFGKD YOLOv5n	Laptop	640	16.47

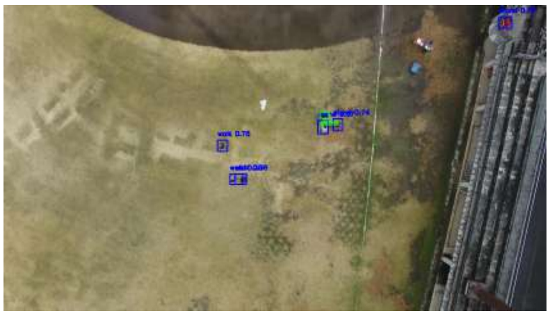
Table 5.9: Comparison of FPS between YOLOv5n-lite and YOLOv5n models on raspberry pi and laptop using real-time played video.


(a) Visualizing Ground Truths along with Predicted Bounding Boxes from PWPR-SFGKD YOLOv5n (similar but non identical)


(b) Visualizing Ground Truths along with Predicted Bounding Boxes from PWPR YOLOv5n-lite (similar but non identical)


(c) Visualizing Ground Truths along with Predicted Bounding Boxes from PWPR-SFGKD YOLOv5n (similar but non identical)


(d) Visualizing Ground Truths along with Predicted Bounding Boxes from PWPR YOLOv5n-lite (similar but non identical)


(e) Visualizing Ground Truths along with Predicted Bounding Boxes from PWPR-SFGKD YOLOv5n (similar but non identical)

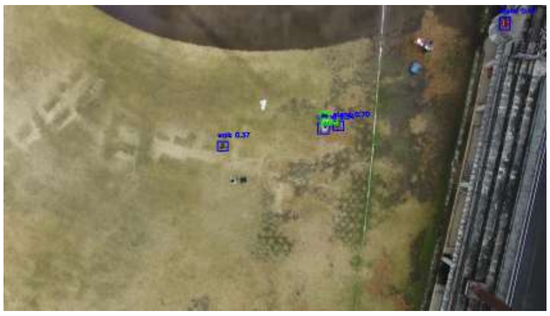
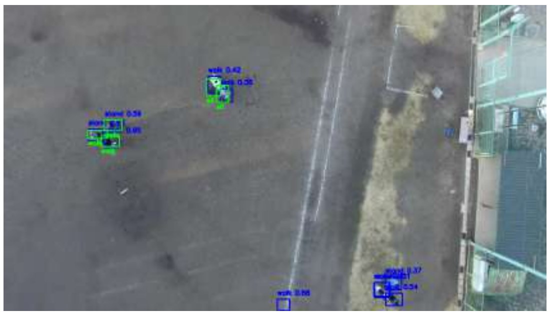
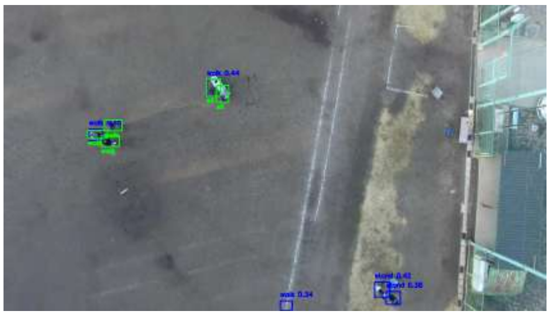
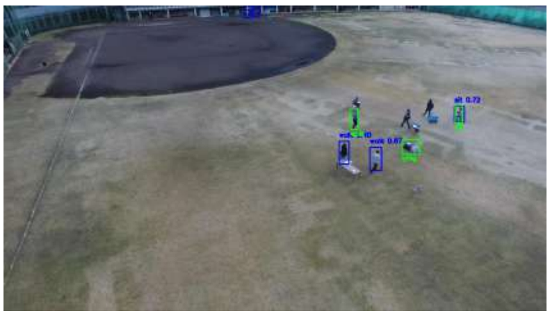
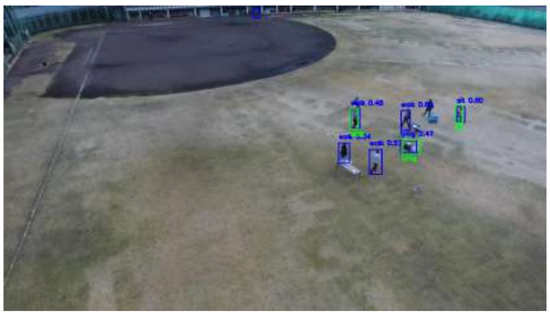

(f) Visualizing Ground Truths along with Predicted Bounding Boxes from PWPR YOLOv5n-lite (similar but non identical)

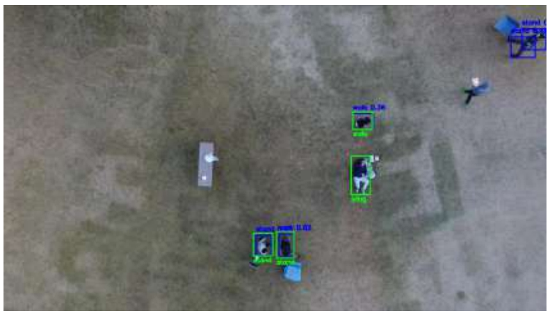
Figure 5.11: Ground Truths along with Predictions from PWPR-SFGKD YOLOv5n and PWPR YOLOv5n-lite: similar but not identical result


(a) Visualizing Ground Truths along with Predicted Bounding Boxes from PWPR-SFGKD YOLOv5n (similar result)


(b) Visualizing Ground Truths along with Predicted Bounding Boxes from PWPR YOLOv5n-lite (similar result)


(c) Visualizing Ground Truths along with Predicted Bounding Boxes from PWPR-SFGKD YOLOv5n (similar result)


(d) Visualizing Ground Truths along with Predicted Bounding Boxes from PWPR YOLOv5n-lite (similar result)


(e) Visualizing Ground Truths along with Predicted Bounding Boxes from PWPR-SFGKD YOLOv5n (similar result)

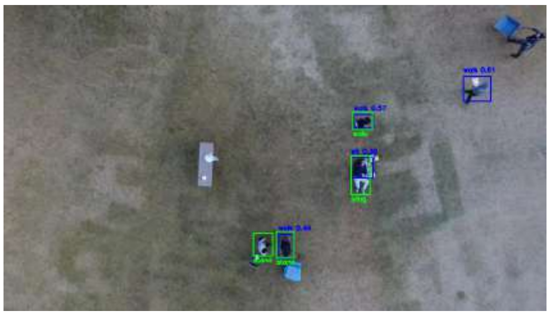
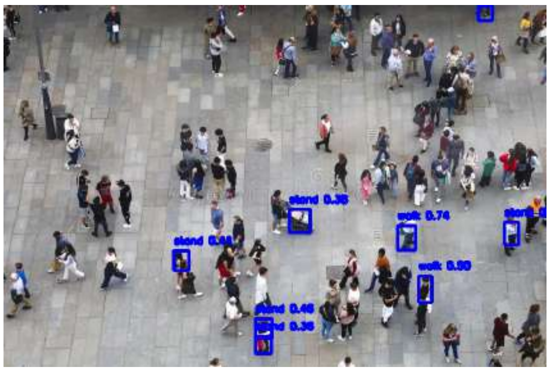
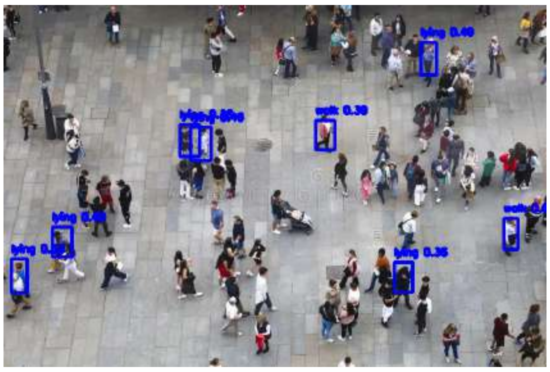

(f) Visualizing Ground Truths along with Predicted Bounding Boxes from PWPR YOLOv5n-lite (similar result)

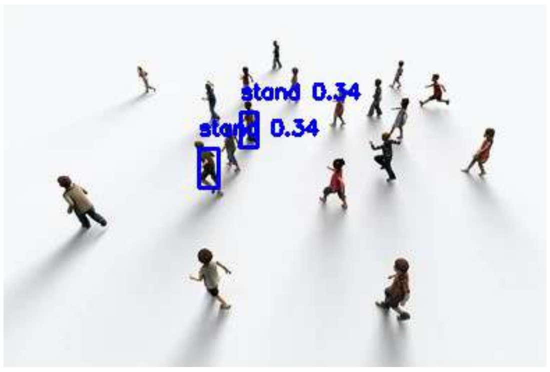
Figure 5.12: Ground Truths along with Predictions from PWPR-SFGKD YOLOv5n and PWPR YOLOv5n-lite: Similar Result



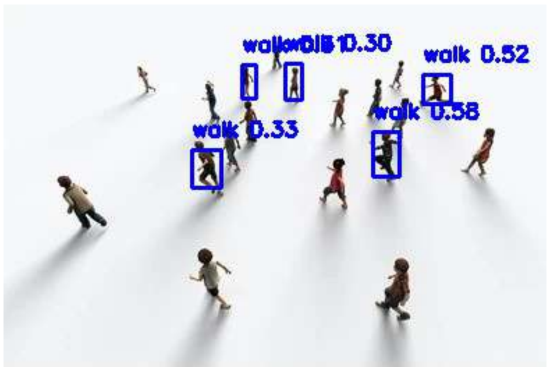
(a) Predictions with bounding boxes from PWPR-SFGKD YOLOv5n (different backgrounds, non-similar results)



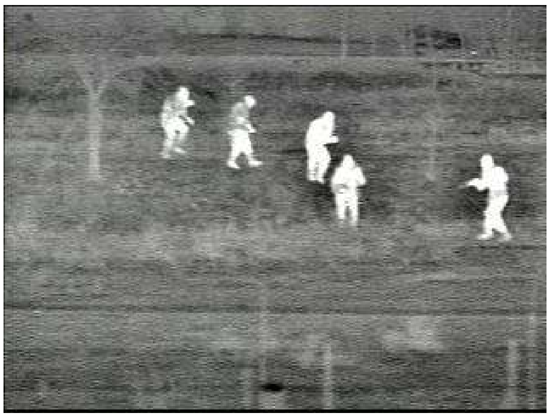
(b) Predictions with bounding boxes from PWPR YOLOv5n-lite (different background non-similar result)



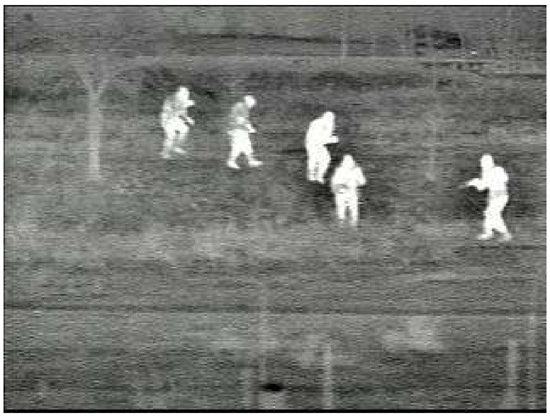
(c) Predictions with bounding boxes from PWPR-SFGKD YOLOv5n (different background different result)



(d) Predictions with bounding boxes from PWPR YOLOv5n-lite (different background different result)



(e) Predictions of PWPR-SFGKD YOLOv5n (Different background no result)



(f) Predictions of PWPR YOLOv5n-lite (Different background no result)

Figure 5.13: Predictions from PWPR-SFGKD YOLOv5n and PWPR YOLOv5n-lite: different backgrounds

Chapter 6

Conclusion

6.1 Summary

During this research, we applied various training methods to evaluate their merits and challenges. Despite the obstacles encountered throughout the research, we successfully developed a novel lightweight architecture called YOLOv5n-lite and proposed a new training strategy, PWPR-SFGKD, to enhance model efficiency. Our main objective was to design an improved knowledge distillation strategy using the teacher-student analogy. However, a major challenge with knowledge distillation is that it often provides high accuracy in the early stages but fails to maintain this performance over time. To address this, we introduced weight pruning, a technique that zeros out certain portions of the model's weights. By retraining the pruned model, we were able to restore its accuracy, achieving better prediction performance than traditional methods. Moreover, the incorporation of SFGKD during the retraining process further boosted the model's prediction accuracy. We compared the performance of our lightweight model trained with the proposed approach to other existing techniques, including the Fine-grained Feature-based Knowledge Distillation (SFGKD) method. Our results demonstrated that the PWPR-SFGKD approach yielded a 9.58% improvement in mAP-50 compared to fine-grained distillation, and a 10.26% improvement when compared to conventional YOLO training. To evaluate the practical applicability of our model, we tested its real-time performance on a Raspberry Pi. Initially, the YOLOv5-nano model achieved only 1.63 FPS in real-time video processing on the Raspberry Pi, while achieving 16.47 FPS on a laptop. Given the low FPS on the Raspberry Pi, we sought to make the model even more lightweight to reduce inference time. We introduced a trimmed version of the YOLOv5-nano model, named YOLOv5-nano-lite, which performed slightly better, achieving 2.07 FPS on the Raspberry Pi and 21.10 FPS on the laptop.

The YOLOv5n-lite model demonstrated a 1.47% improvement in mAP over the conventional YOLOv5n model. When we applied the PWPR training process (weight pruning and retraining), the mAP-50 score improved by 1.38%. However, when we applied the full PWPR-SFGKD method to the YOLOv5n-lite model, we observed a 4.54% decrease in mAP, indicating that while the PWPR-SFGKD strategy can enhance the performance of lightweight models in some cases, it may also have a negative impact on predictions for certain architectures. This highlights the importance of carefully considering the applicability of PWPR-SFGKD based on the specific characteristics of the model.

In short, while the PWPR technique has shown promising results for both the YOLOv5-nano and YOLOv5-nano-lite models, the PWPR-SFGKD strategy should be applied with caution, as it may not always lead to improvements in performance. Nevertheless, the PWPR method continues to show strong potential in improving the efficiency of lightweight models, particularly in real-time applications.

6.2 Future Works

One of the significant challenges faced in this research was the substantial training time required by our approach. While we achieved improved accuracy, this came at the cost of high GPU usage, which presents scalability limitations. To address this, future research could focus on developing more efficient training techniques that reduce time and computational overhead, such as optimizing the model architecture, employing more advanced pruning strategies, or leveraging distributed computing to accelerate the process without compromising performance. Additionally, while the YOLOv5-nano and YOLOv5-nano-lite models show promising results, they are still not ideal for real-time deployment on edge devices like the Raspberry Pi. A promising direction for future work would be to explore even more lightweight variants of the model by rethinking the YOLOv5 building blocks. For example, optimizing convolutional layers, reducing model depth, or experimenting with novel activation functions could help achieve faster inference times, making the models more practical for real-time applications in resource-constrained environments.

Quantization in Deep Neural Networks (DNNs) could also be explored to further optimize model performance. While quantization often leads to challenges such as potential accuracy loss and added model complexity, it offers significant benefits when done correctly. Reducing precision through quantization can greatly decrease memory usage, accelerate inference times, and reduce power consumption. This makes it an attractive approach for deploying models in resource-constrained environments like mobile devices and edge computing. Future work could investigate the integration of quantization techniques into the lightweight variants we developed, evaluating the trade-offs between accuracy and efficiency to make models more suitable for real-time deployment on devices like the Raspberry Pi.

Our study was primarily focused on addressing the specific challenges presented by the selected dataset. However, for broader applicability, future research should involve collecting diverse, domain-specific data to ensure the model can generalize well to different environments and real-world applications. Data diversification—covering various lighting conditions, angles, and backgrounds—would significantly improve model robustness and adaptability. Furthermore, to increase the model's versatility and applicability to a wider range of tasks, future studies should incorporate a more comprehensive set of human action categories. Expanding beyond the five action categories covered in this research, and including additional variations of human movements, would enhance the model's ability to recognize a broader spectrum of actions and contexts, paving the way for more comprehensive human action recognition systems.

References

[1] Yang, K., Jiao, Z., Liang, J., Lei, H., Li, C. and Zhong, Z. "An application case of object detection model based on yolov3-spp model pruning." In "2022 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)," pp. 578–582. IEEE, 2022

[2] Liu, Y., Liu, X., Hao, X., Tang, W., Zhang, S. and Lei, T. "Single-frame infrared small target detection by high local variance, low-rank and sparse decomposition." *IEEE Transactions on Geoscience and Remote Sensing*, 2023

[3] Rokh, B., Azarpeyvand, A. and Khanteymoori, A. "A comprehensive survey on model quantization for deep neural networks." *arXiv preprint arXiv:2205.07877*, 2022

[4] Li, Z., Xu, P., Chang, X., Yang, L., Zhang, Y., Yao, L. and Chen, X. "When object detection meets knowledge distillation: A survey." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023

[5] Zhao, Y., Dong, L., Li, G., Tang, Y., Zhang, Y., Mao, M., Li, G. and Tan, L. "Svim: A skeleton-based view-invariant method for online gesture recognition." In "International Conference on Advanced Data Mining and Applications," pp. 241–255. Springer, 2023

[6] Nikpour, B. and Armanfard, N. "Spatio-temporal hard attention learning for skeleton-based activity recognition." *Pattern Recognition*, volume 139:p. 109428, 2023

[7] Pavan, M. and Jyothi, K. "Human action recognition based on body shape and orbicular grid." In "International Conference on Advances in Data-driven Computing and Intelligent Systems," pp. 97–108. Springer, 2023

[8] Bui, N.N., Kim, J.Y. and Yongbong-dong, B.g. "A study on n-spherical histogram for unbounded tensor features in human action recognition." *International Journal*

of Computational and Applied Mathematics & Computer Science, volume 2:pp. 5–10, 2022

[9] Jing, J., Gao, T., Zhang, W., Gao, Y. and Sun, C. "Image feature information extraction for interest point detection: A comprehensive review." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 45, no. 4:pp. 4694–4712, 2022

[10] Simonyan, K. and Zisserman, A. "Two-stream convolutional networks for action recognition in videos." *Advances in neural information processing systems*, volume 27, 2014

[11] Teepe, T., Gilg, J., Herzog, F., Hörmann, S. and Rigoll, G. "Towards a deeper understanding of skeleton-based gait recognition." In "Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition," pp. 1569–1577, 2022

[12] Yan, S., Xiong, Y. and Lin, D. "Spatial temporal graph convolutional networks for skeleton-based action recognition." In "Proceedings of the AAAI conference on artificial intelligence," volume 32, 2018

[13] Hoang, V.N., Le, T.L., Tran, T.H., Nguyen, V.T. et al. "3d skeleton-based action recognition with convolutional neural networks." In "2019 international conference on multimedia analysis and pattern recognition (MAPR)," pp. 1–6. IEEE, 2019

[14] Zhu, A., Wu, Q., Cui, R., Wang, T., Hang, W., Hua, G. and Snoussi, H. "Exploring a rich spatial–temporal dependent relational model for skeleton-based action recognition by bidirectional lstm-cnn." *Neurocomputing*, volume 414:pp. 90–100, 2020

[15] Plizzari, C., Cannici, M. and Matteucci, M. "Spatial temporal transformer network for skeleton-based action recognition." In "Pattern recognition. ICPR international workshops and challenges: virtual event, January 10–15, 2021, Proceedings, Part III," pp. 694–701. Springer, 2021

[16] Kaur, R. and Singh, S. "A comprehensive review of object detection with deep learning." *Digital Signal Processing*, volume 132:p. 103812, 2023

[17] Hussain, A., Hussain, T., Ullah, W. and Baik, S.W. "Vision transformer and deep sequence learning for human activity recognition in surveillance videos." *Computational Intelligence and Neuroscience*, volume 2022, 2022

[18] Wan, S., Qi, L., Xu, X., Tong, C. and Gu, Z. “Deep learning models for real-time human activity recognition with smartphones.” *Mobile Networks and Applications*, volume 25, no. 2:pp. 743–755, 2020

[19] Serpush, F., Menhaj, M.B., Masoumi, B. and Karasfi, B. “Wearable sensor-based human activity recognition in the smart healthcare system.” *Computational intelligence and neuroscience*, volume 2022, 2022

[20] Saleem, G., Bajwa, U.I. and Raza, R.H. “Toward human activity recognition: a survey.” *Neural Computing and Applications*, volume 35, no. 5:pp. 4145–4182, 2023

[21] Dentamaro, V., Gattulli, V., Impedovo, D. and Manca, F. “Human activity recognition with smartphone-integrated sensors: A survey.” *Expert Systems with Applications*, p. 123143, 2024

[22] Ghosh, S., Srinivasa, S.K., Amon, P., Hutter, A. and Kaup, A. “Deep network pruning for object detection.” In “2019 IEEE International Conference on Image Processing (ICIP),” pp. 3915–3919. IEEE, 2019

[23] Swaminathan, S., Garg, D., Kannan, R. and Andres, F. “Sparse low rank factorization for deep neural network compression.” *Neurocomputing*, volume 398:pp. 185–196, 2020

[24] Li, Z., Sun, Y., Tian, G., Xie, L., Liu, Y., Su, H. and He, Y. “A compression pipeline for one-stage object detection model.” *Journal of Real-Time Image Processing*, pp. 1–14, 2021

[25] De Rijk, P., Schneider, L., Cordts, M. and Gavrila, D. “Structural knowledge distillation for object detection.” *Advances in Neural Information Processing Systems*, volume 35:pp. 3858–3870, 2022

[26] Lin, Y., Han, S., Mao, H., Wang, Y. and Dally, W.J. “Deep gradient compression: Reducing the communication bandwidth for distributed training.” *arXiv preprint arXiv:1712.01887*, 2017

[27] Han, S., Mao, H. and Dally, W.J. “Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding.” *arXiv preprint arXiv:1510.00149*, 2015

[28] Li, G., Qian, C., Jiang, C., Lu, X. and Tang, K. “Optimization based layer-wise magnitude-based pruning for dnn compression.” In “IJCAI,” volume 330, pp. 2383–2389, 2018

[29] Li, H., Kadav, A., Durdanovic, I., Samet, H. and Graf, H.P. "Pruning filters for efficient convnets." *arXiv preprint arXiv:1608.08710*, 2016

[30] Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S. and Zhang, C. "Learning efficient convolutional networks through network slimming." In "Proceedings of the IEEE international conference on computer vision," pp. 2736–2744, 2017

[31] Ding, X., Hao, T., Tan, J., Liu, J., Han, J., Guo, Y. and Ding, G. "Resrep: Lossless cnn pruning via decoupling remembering and forgetting." In "Proceedings of the IEEE/CVF international conference on computer vision," pp. 4510–4520, 2021

[32] Dong, X., Chen, S. and Pan, S. "Learning to prune deep neural networks via layer-wise optimal brain surgeon." *Advances in neural information processing systems*, volume 30, 2017

[33] Liu, Z., Wang, Y., Han, K., Ma, S. and Gao, W. "Instance-aware dynamic neural network quantization." In "Proceedings of the IEEE/CVF conference on computer vision and pattern recognition," pp. 12434–12443, 2022

[34] Zhang, L. and Ma, K. "Improve object detection with feature-based knowledge distillation: Towards accurate and efficient detectors." In "International Conference on Learning Representations," , 2020

[35] Park, W., Kim, D., Lu, Y. and Cho, M. "Relational knowledge distillation." In "Proceedings of the IEEE/CVF conference on computer vision and pattern recognition," pp. 3967–3976, 2019

[36] Zhang, P., Kang, Z., Yang, T., Zhang, X., Zheng, N. and Sun, J. "Lgd: label-guided self-distillation for object detection." In "Proceedings of the AAAI conference on artificial intelligence," volume 36, pp. 3309–3317, 2022

[37] Ma, T., Tian, W. and Xie, Y. "Multi-level knowledge distillation for low-resolution object detection and facial expression recognition." *Knowledge-Based Systems*, volume 240:p. 108136, 2022

[38] Dong, N., Zhang, Y., Ding, M., Xu, S. and Bai, Y. "One-stage object detection knowledge distillation via adversarial learning." *Applied Intelligence*, pp. 1–17, 2022

[39] Yang, Z., Li, Z., Jiang, X., Gong, Y., Yuan, Z., Zhao, D. and Yuan, C. "Focal and global knowledge distillation for detectors." In "Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition," pp. 4643–4652, 2022

[40] Nagarajan, V., Menon, A.K., Bhojanapalli, S., Mobahi, H. and Kumar, S. “On student-teacher deviations in distillation: does it pay to disobey?” *Advances in Neural Information Processing Systems*, volume 36:pp. 5961–6000, 2023

[41] Soviany, P. and Ionescu, R.T. “Optimizing the trade-off between single-stage and two-stage deep object detectors using image difficulty prediction.” In “2018 20th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC),” pp. 209–214. IEEE, 2018

[42] Li, C., Li, L., Jiang, H., Weng, K., Geng, Y., Li, L., Ke, Z., Li, Q., Cheng, M., Nie, W. et al. “Yolov6: A single-stage object detection framework for industrial applications.” *arXiv preprint arXiv:2209.02976*, 2022

[43] Atik, M.E., Duran, Z. and ÖZGÜNLÜK, R. “Comparison of yolo versions for object detection from aerial images.” *International Journal of Environment and Geoinformatics*, volume 9, no. 2:pp. 87–93, 2022

[44] Rahman, A., Lu, Y. and Wang, H. “Performance evaluation of deep learning object detectors for weed detection for cotton.” *Smart Agricultural Technology*, volume 3:p. 100126, 2023

[45] Aldahoul, N., Karim, H.A., Sabri, A.Q.M., Tan, M.J.T., Momo, M.A. and Fermin, J.L. “A comparison between various human detectors and cnn-based feature extractors for human activity recognition via aerial captured video sequences.” *IEEE Access*, volume 10:pp. 45678–45689, 2022

[46] Tang, S., Zhang, S. and Fang, Y. “Hic-yolov5: Improved yolov5 for small object detection.” In “2024 IEEE International Conference on Robotics and Automation (ICRA),” pp. 6614–6619. IEEE, 2024

[47] Khanam, R. and Hussain, M. “What is yolov5: A deep look into the internal features of the popular object detector.” *arXiv preprint arXiv:2407.20892*, 2024

[48] Redmon, J., Divvala, S., Girshick, R. and Farhadi, A. “You only look once: Unified, real-time object detection.” In “Proceedings of the IEEE conference on computer vision and pattern recognition,” pp. 779–788, 2016

[49] Maas, A.L., Hannun, A.Y., Ng, A.Y. et al. “Rectifier nonlinearities improve neural network acoustic models.” In “Proc. icml,” volume 30, p. 3. Atlanta, GA, 2013

[50] Redmon, J. and Farhadi, A. “Yolo9000: better, faster, stronger.” In “Proceedings of the IEEE conference on computer vision and pattern recognition,” pp. 7263–7271, 2017

[51] Redmon, J. and Farhadi, A. "Yolov3: An incremental improvement." *arXiv preprint arXiv:1804.02767*, 2018

[52] Tsang, S.H. "Brief review: Yolov5 for object detection." *Medium, https://sh-tsang.medium.com/brief-review-yolov5-for-object-detection-84cc6c6a0e3a*, June 9 2023. Accessed: 2024-06-25

[53] Barekatain, M., Martí, M., Shih, H.F., Murray, S., Nakayama, K., Matsuo, Y. and Prendinger, H. "Okutama-action: An aerial view video dataset for concurrent human action detection." In "Proceedings of the IEEE conference on computer vision and pattern recognition workshops," pp. 28–35, 2017

[54] Wang, T., Yuan, L., Zhang, X. and Feng, J. "Distilling object detectors with fine-grained feature imitation." In "Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition," pp. 4933–4942, 2019