# CROWD COUNTING USING DEEP LEARNING BASED HEAD DETECTION

**Maryam Hassan**

**Reg. No: 000000277802**

This Thesis Submitted as a partial fulfilment of the requirements for the degree of Masters of Science in Computer Engineering

**Supervised by: Dr Farhan Hussain**

**Co-Supervisor: Dr Sultan Daud**

**Department of Computer Engineering**

College of Electrical and Mechanical National

University of Sciences and Technology(NUST)

# CROWD COUNTING USING DEEP LEARNING BASED HEAD DETECTION

Author

Maryam Hassan

**Reg. No: 000000277802**

This Thesis Submitted as a partial fulfilment of the requirements for the degree of Masters of Science in Computer Engineering

Supervised by: Dr Farhan Hussain

Co-Supervisor: Dr Sultan Daud

Thesis supervisor signature -------------------------------------

**Department of Computer Engineering**

College of Electrical and Mechanical National

University of Sciences and Technology(NUST)

ISLAMABAD, PAKISTAN

**2021**

## Disclaimer

I hereby state that this thesis is my own unique work and has not been submitted to any institution for assessment purpose. To the best of my knowledge, this thesis contains no material previously published by other person expect for which the due acknowledgement has been made.

Maryam Hassan Dated:

## Dedication

This thesis is proudly dedicated to my beloved Mother and Friend.

# Acknowledgements

As a matter of first importance all acclaim to Almighty ALLAH, Lord of the Creations, the Creator, for giving me the quality, ability, and chance to attempt this exploration, persistent and to continue and finish it agreeably. May the harmony and favors upon the noblest of Prophets and Messengers, our Prophet Muhammad (PBUH) and on his Family and majority of his Companions.

An obligation of appreciation is additionally owed to Head of department for furnishing me with all the facilitation. The fulfillment of this theory could not have been conceivable without the expertise of my research advisor's Dr Farhan Hussain and Dr Sultan Daud. Their prompt inspiration, timely suggestions with kindness, enthusiasm and dynamism have enabled me to complete my thesis.

**Table of Contents**

# List of Table

# List of Figures

# List of Symbols and Abbreviations

| | |
|---|---|
| **mAP** | Mean Average Precision |
| **IOU** | Intersection Over Union |
| **R-CNN** | Region Convolutional Network |
| **RPN** | Region Proposal Network |
| **CNN** | Convolutional Neural Network |
| **YOLO** | You Look Only Once |
| **SSD** | Single shot detector |

# Abstract

Accurate and fastest object detection models are in high demand due to its wide variety of applications in the fields of computer vision, such as pedestrian detection, video surveillance and especially crowd counting applications. Automated crowd has been and continues to be a difficult problem for autonomous visual surveillance for many years. In the relevant literature, a substantial amount of research has been undertaken on the subject of crowd-counting and different architectures have been proposed for accurate and timely detection of heads in a crowd. Most of the approaches are based on regression, segmentation, image processing, machine learning techniques, counters and sensor-based models. Although the advancements in infrastructure has significantly improved the prediction accuracy but small heads are often missed by most of the proposed architectures in the literature. Scale invariance and high miss detection rates for small objects leads to the inaccurate results. The purpose of this research is to provide an accurate and fastest detection model for crowd counting by focusing on human head detection in real time scenarios acquired from publicly available datasets of Casablanca, Hollywood-Heads and Scut-head. In this study, we have tuned a yolov5 which is a deep convolutional neural networks (CNN) based object detection architecture by improving the mAP, precision and recall. The loss factors are reduced and accurate results are achieved by accurate tuning of hyper-parameters. Transfer learning approach is used for fine-tuning the architecture. From the experimental results, it can be seen that this yolov5 architecture showed significant improvements in small head detections in crowded scenes as compared to the other baseline approaches such as that Faster R-CNN and VGG-16 based SSD MultiBox Detector. In Faster R-CNN, features are extracted in the last layer therefore image resolution is decreased and small objects are not detected while yolov5 perform slicing of feature maps in the backbone region Therefore, small heads are detected accurately. Another main contribution of our research is use of merge dataset which include every kind of heads that is medium, large and small.

**Keywords: Crowd Counting, Head detection, Yolov5, Precision, Mean average Precision**.

# Chapter 1: Introduction

## 1.1 Introduction

Today, the overcrowding in many places worldwide led to a multitude of congested situations. Parades, station departures and entrances, political protests, and strikes all contribute to the overcrowding. These circumstances suggest a multiplicity of security concerns. The key task in carrying out crowd surveillance is accurate crowd counts. For applications such as video surveillance and traffic management, crowd counting is essential. Due to strong occlusions, scene perspective distortions, and a wide range of crowd distributions, crowd counting is a difficult process.

Vision based security systems are becoming increasingly common in modern society. Mostly every public area has its own security system, as it is vital to use such systems to protect public security. Due to the availability of low-cost security video cameras and high-speed computer networks, it is now technologically viable and financially reasonable to install such a system for crime reduction and detection [1]. Detection and recognition is a long-standing computer vision topic that has seen a great deal of research and impressive advancements. In the visual surveillance community, for crowd monitoring and management, crowd analysis has recently emerged as an increasingly important and targeted subject. The estimation of crowd density is also attracting a lot of attention.

The global population has been rapidly growing in recent decades. As a result of global urban population growth, the crowd problem has become more prevalent. Large crowds can be seen in both enclosed and open spaces, such as building halls, airports, and stadiums, as well as on walkways, parks, sporting events, and public demonstrations. The reason for the gathering has a significant impact on the crowd's large-scale features and behaviors. As a result, many scientific studies in psychology, sociology, public services, safety, and computer vision are interested in analyzing crowd dynamics and behaviors.

In the field of crowd analysis, automated crowd counting is a popular issue. Many notable articles have been published in this topic over the last few decades, and it has been and continues to be a difficult problem for autonomous visual surveillance for many years[2]. Artificial intelligence allows computers to think in human-like ways. By including training and learning components, machine learning makes the path more equitable. The availability of large datasets and high-performance computers gave rise to the deep learning idea, which automatically extracts features or elements of variation that distinguish objects.

Due to the advent of autonomous vehicles, smart video monitoring, facial detection, and a variety of people counting applications, robust and precise object detection models are in high demand. Many practical applications of computer vision rely significantly on object detection, such as human face identification, pedestrian detection, vehicle detection, and video surveillance. In classification, a classifier is taught to categories and identify the image's content worldwide. During object detection, the developed models not only label and classify the content, but also find the bounding box of the item.

One of the most important tasks for crowd counting is Human Head detection in any scenario. Human detection is essential in a variety of real-world applications, including enhanced human-machine interactions, video surveillance, and crowd analysis. The most common approaches used are motion detection and background elimination. To carry out these approaches most common approaches are regression, segmentation, image processing, machine learning techniques, counters and sensor-based models. Initially, hand-crafted features were employed in machine learning algorithms for computer vision problems. In comparison to learnt features, these features are less reliable and discriminative. Deep learning approaches have recently been used to successfully apply features learning to major computer vision problems such as segmentation [3], classification [4], detection and identification [5]. For many years, deep learning approaches have consistently won classification and detection contests such as ImageNet [6].

## 1.2 Motivation

Crowd surveillance is a popular field of research because of its safety and security applications. A substantial amount of research has been undertaken on the subject of crowd-counting. With advances in infrastructure and better technologies developing on a daily basis, prediction accuracy has significantly improved. Crowd counting is the most challenging task in crowd scene analysis.

The purpose of this Research is to provide an accurate approach of crowd counting by focusing on Human Head detection in daily life events and real time scenarios for formulating the accurate and fastest crowd counting model. In terms of face recognition and identification algorithms the technology has attained its maturity level. Human detection still faces several hurdles because of the human body's articulate nature.

As head detection does not have the constraints of multiple features as in facial identification, it is more suited to people counting and locating, especially in an indoor setting. However, there are numerous obstacles to detecting heads in an indoor scenario, including a wide range of head scales and looks, as well as small head detection.

## 1.3 Problem Statement

Crowd counting is aimed at counting the number of instants in a crowded area and various potential solutions for single image crowd counting have been developed. The head detection of individuals has several video monitoring applications. Although important, few literature study concerning head detection has been described. The main focuses are on facial detection and pedestrian detection in the current technologies. These solutions model the head detection as a problem with multi-object detection. Head detection can offer localization information that can be used in real-time applications for crowd counts. Scale invariance, high miss detection rates for small objects and time constraint are still a challenge, which leads to the problem of inaccurate crowd count. In real time scenarios and in crowd surveillance activities, time constraint along with accuracy is an important factor for better results.

To estimate the number of persons from a sequence of photos, most people utilize a regression based method and high performance object detectors. Because the head is the most apparent part of the body in a crowded situation so the focus of this research is on crowd counting by head detection. Therefore, we proposed an improved deep learning based approach for providing accurate head count by improving the mean Average precision (mAP), precision, recall and reduced loss factors in less time constraint.

Another common issue faced in the crowd counting application is the limitation of the publically available datasets. The available datasets mostly contain same scenario based images. The focus of this research is on making a single dataset containing multiple scenario based images so that the proposed model would be able to detect all heads for any scenario.

## 1.4 Contribution

For crowd counting mostly used approaches are computer vision, machine learning algorithms, image processing and high performance object detectors. Crowd counting by head detection is a new approach which provide numerous applications in person counting and real time tracking of human's. In real time tracking the most concerned problem is time lag. The proposed approach in this research is to use the deep learning model i.e. yolov5 to compute the crowd counting using head detection. Following are the main contributions:

- Crowd counting by head detection using deep learning approach.
- Use of Yolov5 i.e. the newest and the fast object detection architecture for head detection in different scenarios. The model suggested is one of the fastest models released up till now for object detection.
- Fine-tuning to improve the mAP, Precision, Recall and intersection over union (IoU) of the predicted results.
- Provision of a Pre-retrained model which can detect head in any type of scenario by compiling a single dataset having the images of multiple scenarios, to provide the most stable model.
- The proposed model can be efficiently used as preprocessor for human tracking, behavioral understanding for obtaining trajectories and for obtaining anomalies.

## 1.5Outline

The rest of the thesis is structured as follows:

Chapter 2 reviews the literature of crowd surveillance methods. The methodology of the proposed method is described in detail in Chapter 3. In Chapter 4, experimental work and their results are discussed. Chapter 5 comprises of the conclusion and future work of the research.

# Chapter 2: Literature Review

Crowd analysis generally, and specially counting, are highly developed areas of computer vision due to their wide range of applications [7]. Crowd counting is an ongoing research subject that has experienced several advancements since the emergence of deep learning algorithms[8]. In order to build technologies that alert security officers to persons or events of interest in crowded scenarios, much research is underway. Methods for extracting information from video footage in order to understand crowd behavior, monitor people in crowds, and spot unusual behavior are critical [9].

Several approaches have been presented for estimating the persons in a picture or video. Scientists and researchers initially developed fundamental ML and computer vision algorithms such as density-based techniques, regression and object detection to estimate crowd density and density maps [10].

These solutions are associated with a number of difficulties, including size, occlusions, and uniform density [11]. Later on, when Convolutional Neural Networks(CNN) demonstrated their ability to overcome these flaws in a variety of computer-vision applications, researchers concentrated on them in order to leverage their foot in algorithmic derivations. Extensive research has been conducted in this area [12].

## 2.1 Object detection Based Crowd Counting

Initially, most crowd counting research was done using detection-based techniques. The detection algorithms employ standard detectors helps identifying and counting the target items in photos or videos. This topic, in particular, has drawn the attention of the scientific community for automated identification of aberrant crowd behavior during public events [13]. represented three-dimensional features to represent the human body, and a random method to compute the number of persons [14].To retrieve head features, several studies employed the Haar wavelet transform [15].

Leibe, B., Seemann, E., & Schiele, B. in 2005, introduced a unique method for detecting pedestrians in cluttered environments. Their method works in a series of iterative evidence collection steps rather than in a single run. The mixing of local and global inputs via probabilistic top-down segmentation was a key component of their strategy. Qualitative and quantitative results from a large data set demonstrate that their technique can detect pedestrians in crowded environments, even when they merge and substantially dislocate each other [16].

Topkaya, Erdogan & Porikli, in 2013 presented a crowd counting model using Dirichlet Process Mixture Model(DPMM) in parallel with clustering, local clusters of people were acquired and the proposed measure allowed them to estimate the number of persons in the clusters. DPMMs allow for the totally automatic detection and tracking of an unknown and variable number of objects without initial tagging. This enabled them to use person detectors without segmenting distinct persons one by one [17]. Li, M., Zhang, Z., Huang, K., & Tan, T. in 2008 proposed a unique method for calculating the number of individuals in CCTV scenarios where people are gathering and waiting. The proposed technique combines a MID (Mosaic Image Difference) based foreground segmentation algorithm with a HOG (Histograms of Oriented Gradients) based head-shoulder recognition algorithm to provide a precise estimation of persons counts in the given region [18]. Wu & Nevatia, in 2007, described a method for automatically detecting and tracking multiple, perhaps partially occluded persons in a walking or standing stance from a single camera that can be fixed or moving via Bayesian Edgelet-based part detectors combination [19].

These detection-based approaches frequently fail to count people accurately in excessively congested settings. The classifier model chosen for classification has a major impact on the accuracy of human detection. This technique provides acceptable detection in sparse environments, however it fails to perform accurate in crowded scenarios with interference and congestion even in surveillance applications where image resolution influences accuracy.

## 2.2 Regression Based Crowd Counting

In the dense crowds and where the backdrop clutter is considerable, counting by detection is not particularly precise. Counting by regression is implemented to overcome these challenges, in which the features retrieved from the neighboring pixels' patches are mapped to count. It nether segment nor track the individuals in this scenario [20].

Without employing explicit object segmentation or tracking, Chan, Liang, & Vasconcelos in 2008 proposed a privacy-preserving approach for assessing the size of inhomogeneous crowds full of individuals travelling in separate directions. From each segmented region, a collection of simple integrative features is retrieved, and the correlation among features and the number of individuals per segment is computed using Gaussian Process regression [21].

Mei & Yanyun in 2013, proposed the improved method of crowd counting using regression. To highlight the feature set's describable potential, we suggest a new low-level feature, the number of corner points. Then, to improve the performance of the suggested algorithm, they introduced a fusion scheme combining relevance vector regression (RVR) and Gaussian process regression (GPR) [22].

Yang, Zhou, & Kung in 2018 suggested that network of generative adversaries can produce high quality crowd density maps of various crowd density scenarios. They used the discriminator's adverse loss to increase the efficiency of the estimated density map that is essential to predict crowd numbers properly. Multiple aspects of the hierarchy from the crowd image can be extracted [23].

Yao, H., Han, K. & Wan, in 2017 proposed the model that is built on Convolutional Neural Networks (CNN) and long short-term memory (LSTM). They first ran the dataset through a pre-trained CNN to extract a collection of high-level features. Then, features from neighboring regions are used to regress the local counts using an LSTM structure that gets spatial data into consideration. A summing of the local patches yields the final global count [24].

Previously, regression-based techniques based on global images features were produced, but these approaches were unable to extract the information's of region wise distribution. One of the most important aspects of these approaches is the extraction of relevant features. When the crowd size is small, this method may overestimate the prediction, it is susceptible to background interference since it rely on foreground extraction.[25].

## 2.3 Crowd counting via density based approach

The density-based crowd counting approaches provide density values that are estimated using low-level features including pixels or areas, which resolves the disadvantage of regression-based methods while simultaneously preserving location information. Density estimation methods can integrate spatial information and build a mapping link between object characteristics and density maps [26].

Pham, Kozakaya, Yamaguchi, & Okada in 2015, proposed a method for learning the mapping between patch characteristics and the relative similar locations including all objects within each patch, which leads to the generation of the patch density map using Gaussian kernel density estimation. They also suggested a congestion problem prior and an effective forest reduction approach to increase prediction accuracy rate [27]

Luo et al. presented FF-CNN (Feature Fusion of Convolutional Neural Network), a deep convolutional neural network technique based on feature fusion. Before including the head count, the proposed FF-CNN localized the crowd image to its crowd density map. High-quality density maps that served as ground truths for network training were created using geometry adaptive kernels. The deconvolution approach was utilized to combine high and low-level functions for joint optimization, and two loss functions were employed: the loss of density maps and the absolute count loss. [28].

Xu & Qiu in 2016, proved that using more picture features could potentially improve crowd detection ability. they employed random forest as the regression model to improve computational performance and robustness. To overcome the dimensionality of data, they incorporated random projection with in tree nodes. Finally, the random forest algorithm combines all of the tree findings to estimate crowd density. Following the training of a CNN with a fixed dataset, a data-driven approach is used to fine-tune (adapt) the trained CNN to an unknown target scene, in which training samples comparable to the target scene are gathered from the training scenes [29].

## 2.4 Crowd counting using Deep learning methods

Leaving old approaches aside, Convolutional Neural Network (CNN)-based computer vision techniques are currently being employed to obtain higher accuracy than existing method. There is a plenty of CNNs developed to achieve crowd density [30]. Fu, Min et all, developed an end-to-end CNN regression predictive model for high density crowd counting, represents the first time, CNN based approach has been used in crowd counting [31].

D. B. Sam et all, designed the LSC-CNN model, which can consistently distinguish people's heads in scattered to crowded gatherings  To improve human resolution and provide improved predictions at several resolutions, LSC-CNN uses a multi-column architecture with top-down feature modulation. Surprisingly, the suggested training regime only needs point head annotation yet can estimate approximate head size information [32].

In one of the researches, the deep architecture has been introduced, which clearly extracts the features of several receptive field sizes and acknowledges the value of every single image spot, thereby making possible rapid changes in scale. In other words, our method adaptively encodes the scale of contextual information required to forecast crowd density [33].

Elad Walach et all, presented a layered technique in which training is performed in stages. They added CNNs continuously, so that each new CNN is trained to estimate the residual error of the previous prediction. Following the training of the first CNN, the next CNN is trained on the gap between the estimation and the ground reality. The procedure is then repeated for the third CNN etc. [34].

A. Vora and V. Chilaka designed fully convolutional, end-to-end trainable head detection model that uses a systematic anchor selection methodology based on the network's optimal receptive field to produce good results in cluttered situations. In contrast to Faster-two-stage RCNN's pipeline (Region Proposal Network + ROI Pooling and Classification), this design employs a single fully-convolutional network capable of classification and bounding box prediction [35].

An approach for detecting activity variations in crowd photographs using cluster-based analysis is described. For object recognition, the Gaussian YOLOv3 model is employed. The suggested clustering method is used to track changes in the cluster's number, coordinates, and orientation. As a consequence of the data extractions, change in activity time, locality, and classifications are accomplished [36].

L Boominathan et all, presents a new deep learning framework for predicting crowd density from static pictures of densely populated areas. To identify the density map for a particular crowd picture, they applied a mix of deep and shallow fully convolutional networks. This combination is used to capture both high-level contextual features (face/body detection) and low-level characteristics (blob detection) that are required for crowd counts at large scale fluctuations [37].

For near-real-time crowd counting, a deep convolution neural network (DCNN)-based system can be employed. The system makes use of an NVIDIA GPU processor to take advantage of the parallel computing architecture and process video feeds from a camera quickly and efficiently. This study aids in the development of a model for detecting heads filmed by CCTV cameras. The model is fully trained by presenting it with a variety of situations, including overlapping heads, limited visibility of heads, and so on. In densely populated areas, this method gives substantial accuracy in predicting the head count [38].

Zeng, Xin, et al presented a new deep scale purification network (DSPNet) for dense crowd counting that can encode multiscale characteristics while reducing contextual information loss. There are two parts to the DSPNet model: a frontend and a backend. The unified deep neural network backend uses a “maximal ratio combining” method to learn complementing scale information at multiple layers, while the frontend is a traditional deep convolutional neural network. To facilitate model learning and reduce contextual information loss, DSPNet conducts full RGB image-based inference [39].

Sultan Daud Khan et all, presented a temporal consistency model (TCM) to increase the accuracy of a generic object detector by combining spatial-temporal information from successive frames of a video. Their approach, in general, takes detection from a generic detector and increases mean average precision (mAP) by recovering missing detection and reducing false positives [40].

R-CNN has made significant progress in the area of object detection. It does, however, have drawbacks, such as low efficiency, a long processing time, and a slew of issues. As a result, R-CNN applications do not have a broad variety of applications. Furthermore, R-CNN must extract visual characteristics for many candidate locations ahead of time. This process can consume a significant amount of disc space. Traditional CNN requires a fixed-size input map; however, distortion of pictures during the *normalization* process causes the image's size to vary, which is deadly to CNN feature extraction.

R-CNN requires a high number of features as training samples for classifiers and regression model. Fast R-CNN does not need more storage [41]. Even though Fast R-CNN has made significant improvements over R-CNN, it still has flaws. Fast R-CNN is a time-consuming technique that takes around 2-3 seconds to extract candidate regions and 0.32 seconds to extract feature categorization, making it difficult to fulfil real-time application requirements [42].

The faster R-CNN is to recognize and identify visual objects in the proposal based on RPN extraction. This is a faster version of the R-CNN network that detects and classifies objects at a rate of seven frames per second. The R-CNN uses a deep neural network to determine where visual items are and what class they belong to. R-CNN speed has been increased considerably [43].

# Chapter 3

# Proposed Methodology

The recommended methodology for my research is detailed in this chapter. In this research work the main focus is to use deep learning based approach. A YOLOv5 network is used to recognize the head area of the human body. Several datasets of people in various scenarios are used. The neural network was then trained to detect each human head in order to count the number of people in the crowd.

For the first time, in 2015, researcher Joseph Redmon and colleagues presented the YOLO method, an object identification system that executes all of the necessary phases to recognize an item using a single neural network (the term as You Only Look Once) [44]. It transforms object recognition into a single regression task, moving from picture pixels onto bounding boxes and class probabilities. YOLO differs from R-CNN, Fast R-CNN, and Faster R-CNN in terms of network architecture. To begin with, YOLO training and testing occur on a different network. The method of obtaining a regional proposal was not depicted in YOLO. R-CNN and Fast R-CNN, on the other hand, employ different modules to generate the candidate box.

As a result, the training is divided into the following parts. RPN convolution network replaces the selective search module of R-CNN and Fast R-CNN utilizing Faster R-CNN. It combines RPN with the Fast R-CNN network to provide an integrated detection system. Despite the fact that RPN and Fast R-CNN have the same convolutional layer, the RPN and Fast R-CNN networks must be trained separately during model training.

## 3.1 YOLO previous versions

Until now, five versions of YOLO have been released (including the original version). Each version has been improved and updated to incorporate the latest cutting-edge concepts from the computer vision research field. In addition, due to the algorithm's failure to achieve the needed performance and accuracy, several concepts have been eliminated. As a result, YOLO is now one of the most powerful object recognition algorithms available today. Understanding the YOLOv5 algorithm requires first learning about the four prior versions.

### 3.1.1 YOLO v1

Joseph Redmon released YOLO v1 in May 2016 with the article "You Only Look Once: Unified, Real-Time Object Detection." This was a significant advancement in real-time object detection [44]. YOLO v1 is a cutting-edge object identification method that analyses images at 45 frames per second in real time. YOLO stands for "you only live once." Multiple bounding boxes and class probabilities for those boxes are predicted simultaneously by a single neural network. Yolo v1 is also known as simple YOLO.

The real-time object identification algorithm YOLO (You Only Look Once) is one of the most efficient and incorporates many of the most original concepts from the computer vision research field. The methodology of YOLO is entirely different. YOLO is a real-time object identification convolutional neural network (CNN). The method splits the image data into regions and predicts bounding boxes and probabilities for every region using a single CNN. YOLOv1 has a relatively simple structure, consisting of convolution, pooling, and eventually adding two layers of full connectivity. Pascal VOC detection dataset was used to test the YOLO model. The network's first convolutional layers extract visual features, while the fully connected layers predict output probabilities and coordinates. The most significant distinction is that the final output layer employs linear functions as the activation function since the position of a bounding box must be anticipated, as well as the likelihood of confidence and the size of the bounding box. The YOLOv1 network has 24 convolutional layers and two complete connected layers in its construction. 448 X 448 is the network entry.

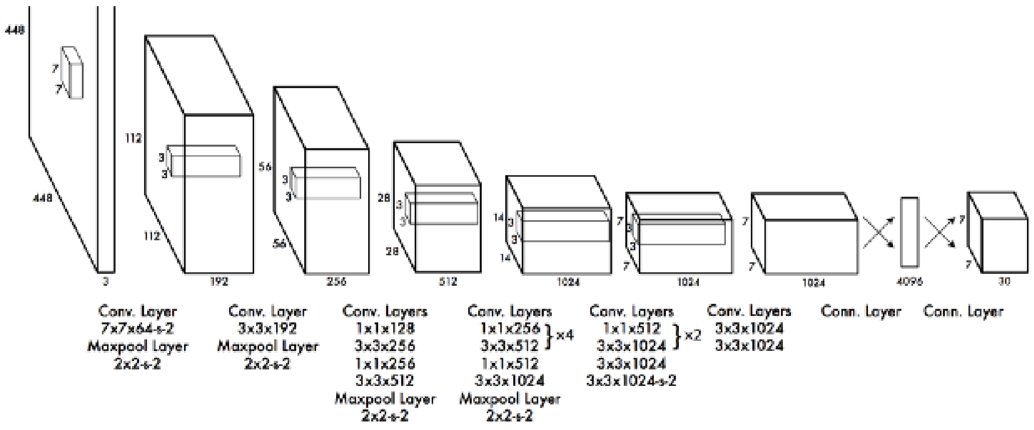

448
7
7
448
3
112
3
3
112
192
56
3
3
56
256
28
3
3
28
512
14
3
3
14
1024
7
3
3
7
1024
7
7
1024
4096
7
7
30
Conv. Layer 7x7x64-s-2 Maxpool Layer 2x2-s-2
Conv. Layer 3x3x192 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x128 3x3x256 1x1x256 3x3x512 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x256 3x3x512 }x4 1x1x512 3x3x1024 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x512 3x3x1024 }x2 3x3x1024 3x3x1024-s-2
Conv. Layers 3x3x1024 3x3x1024
Conn. Layer
Conn. Layer

Figure 1: Architectural Diagram of Yolov1

The fundamental idea behind YOLOv1 is to insert a grid cell into a picture with a size of SxS (7x7 by default). As indicated in the diagram below, if the center of an item falls within a grid cell, that grid cell is responsible for detecting that object.

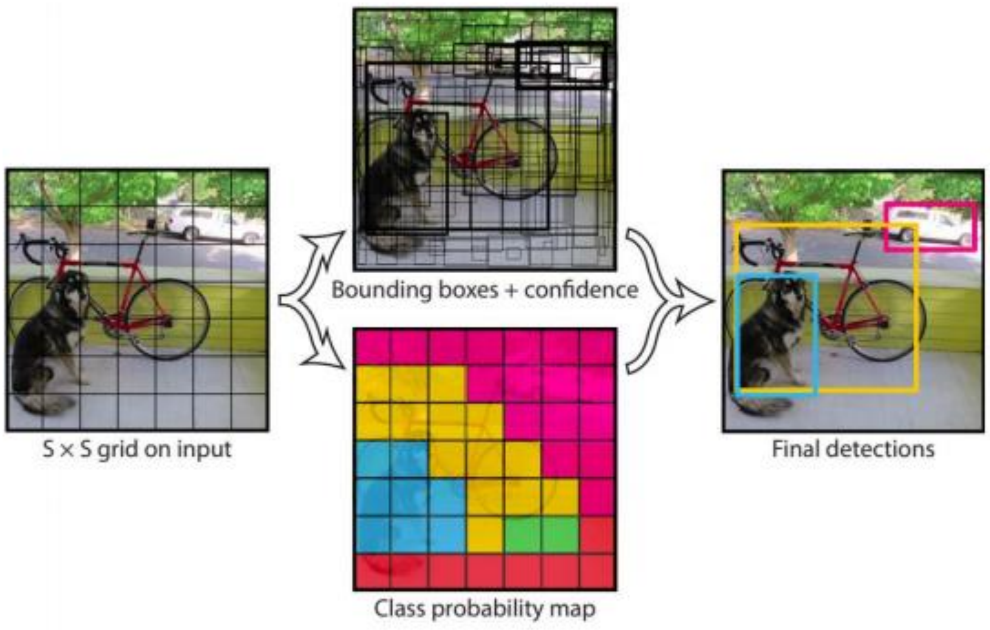


Figure 2: Working of Yolov2

YOLO imposes strong spatial constraints on bounding box predictions since each grid cell only predicts two boxes and can only have one class. This spatial constraint limits the number of nearby objects that the model can predict.

The GoogLeNet (Inception) network, which helps to lower the features space from subsequent layers, inspired the sequences of 1x1 and 3x3 convolutional layers [45]. Instead of Leaky Rectified Linear Unit (leaky ReLU) activation, the final layer employs a Linear activation function:

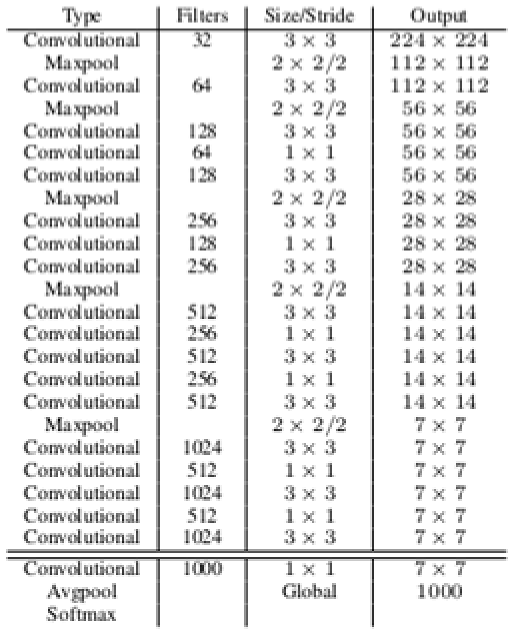
$$\phi(x) = \begin{cases} \mathrm{x}, \text{if } \mathrm{x} > 0 \\ 0.1\mathrm{x},\ otherwise \end{cases}$$

### 3.1.2 YOLO v2

Further research was done, culminating in the December 2016 publication "YOLO9000: Better, Faster, Stronger," by Redmon and Farhadi, both of the University of Washington, which improved the YOLO detection approach by simultaneously improving detection and classification to identify over 9,000 item types [44].

#### 3.1.2.1 Architecture of YOLO v2

DarkNet-19, a novel classification model proposed by YOLOv2, comprises 19 convolutional plus 5 max pooling. YOLOv1 is replaced with 11 convolutional layers in this model. After pooling, the 1x1 convolution layer uses 33% convolution and doubles the number of channels. For prediction classification, average pooling is employed instead of complete connection, and 1x1 convolution compression is used between 3x3 convolution. Batch normalization is a technique for improving model stability, speeding up convergence, and regularizing models. After batch normalization, all of YOLOv2's convolution layers are applied [45].



| Type | Filters | Size/Stride | Output |
|---|---|---|---|
| Convolutional | 32 | $3 \times 3$ | $224 \times 224$ |
| Maxpool | | $2 \times 2/2$ | $112 \times 112$ |
| Convolutional | 64 | $3 \times 3$ | $112 \times 112$ |
| Maxpool | | $2 \times 2/2$ | $56 \times 56$ |
| Convolutional | 128 | $3 \times 3$ | $56 \times 56$ |
| Convolutional | 64 | $1 \times 1$ | $56 \times 56$ |
| Convolutional | 128 | $3 \times 3$ | $56 \times 56$ |
| Maxpool | | $2 \times 2/2$ | $28 \times 28$ |
| Convolutional | 256 | $3 \times 3$ | $28 \times 28$ |
| Convolutional | 128 | $1 \times 1$ | $28 \times 28$ |
| Convolutional | 256 | $3 \times 3$ | $28 \times 28$ |
| Maxpool | | $2 \times 2/2$ | $14 \times 14$ |
| Convolutional | 512 | $3 \times 3$ | $14 \times 14$ |
| Convolutional | 256 | $1 \times 1$ | $14 \times 14$ |
| Convolutional | 512 | $3 \times 3$ | $14 \times 14$ |
| Convolutional | 256 | $1 \times 1$ | $14 \times 14$ |
| Convolutional | 512 | $3 \times 3$ | $14 \times 14$ |
| Maxpool | | $2 \times 2/2$ | $7 \times 7$ |
| Convolutional | 1024 | $3 \times 3$ | $7 \times 7$ |
| Convolutional | 512 | $1 \times 1$ | $7 \times 7$ |
| Convolutional | 1024 | $3 \times 3$ | $7 \times 7$ |
| Convolutional | 512 | $1 \times 1$ | $7 \times 7$ |
| Convolutional | 1024 | $3 \times 3$ | $7 \times 7$ |
| Convolutional | 1000 | $1 \times 1$ | $7 \times 7$ |
| Avgpool | | Global | 1000 |
| Softmax | | | |

Figure 2: Architecture of Darknet 19 [45]

Instead of predicting coordinates directly from the convolution network like Fast R-CNN and Faster R-CNN do, YOLO employs fully connected layers to predict bounding boxes. The fully connected layer in this release is removed and replaced with anchor boxes to forecast the bounding boxes. The following changes were made to the architecture.

#### 3.1.2.2 Batch normalization

Batch normalization improves convergence significantly while reducing the requirement for other types of regularization. We gain a greater than 2% increase in mAP by using batch normalization on all of the convolutional layers in YOLO. Batch normalization also aids in model regularization. We may remove dropout from the model using batch normalization without overfitting. Batch normalization layers were added to all of the convolutional layers in YOLO v2. This resulted in considerable convergence gains while removing the necessity for other types of regularization.

This method not only cuts down on training time, but it also improves the network's applicability. Batch normalization enhanced mAP (mean average precision) by around 2% in YOLOv2.

### 3.1.2.3 Fine grained approach for feature extractions.

One of the primary challenges that must be addressed in the YOLO v1 is the recognition of microscopic entities on the picture. This has been addressed in the YOLO v2 version, which divides the image into 13*13 grid cells, resulting in a smaller image size than the previous version. This enables the yolo v2 to detect and locate both small and large items in the image.

### 3.1.3 YOLO v3

YOLO v3 has all we need for real-time object detection and accurate categorization. The authors referred to this as an incremental improvement.

### 3.1.3.1 Architecture of YOLO v3

For Darknet architecture, YOLO v2 uses a unique deep 30-convolutional layer architecture, compared to YOLOv1's 11 layers. More layers in deep neural networks equals more accuracy. When forwarding to further layers, however, the input picture was down sampled, resulting in the loss of fine-grained characteristics. That's why YOLOv2 had a hard time detecting tiny objects. Skip connections were introduced using ResNet in yolov2 to aid activation propagation across deeper layers without gradient diminishing. Figure 4 shows the ResNet architecture deployed in yolov2.

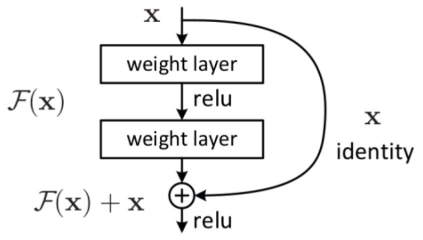
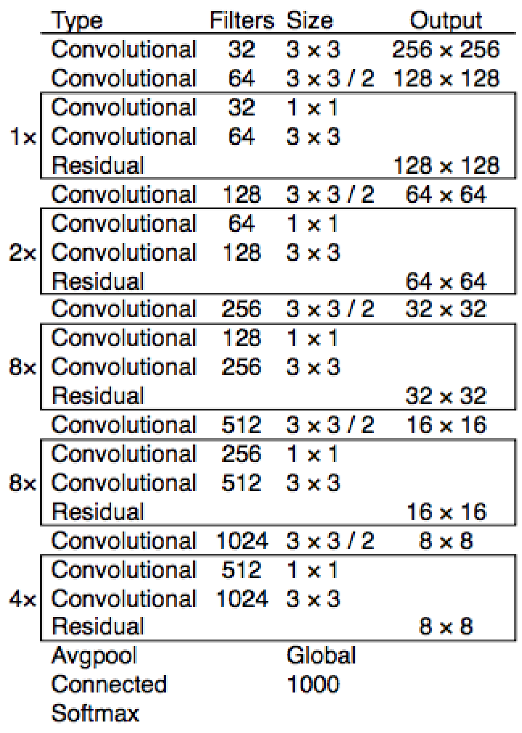

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 3: Architecture of ResNet skip connections

Redmon, et al proposed a superior architecture in which the feature extractor was a mix of YOLOv2, Darknet-53 (53 convolutional layers), and Residual networks (ResNet). Inside each residual block, the network is formed using a bottleneck structure (1x1 followed by 3x3 convolution layers) and a skip connection [45].

The model employed the Darknet-53 architecture, which was designed with a 53-layer network



| | Type | Filters | Size | Output |
|---|---|---|---|---|
| | Convolutional | 32 | 3 × 3 | 256 × 256 |
| | Convolutional | 64 | 3 × 3 / 2 | 128 × 128 |
| | Convolutional | 32 | 1 × 1 | |
| 1× | Convolutional | 64 | 3 × 3 | |
| | Residual | | | 128 × 128 |
| | Convolutional | 128 | 3 × 3 / 2 | 64 × 64 |
| | Convolutional | 64 | 1 × 1 | |
| 2× | Convolutional | 128 | 3 × 3 | |
| | Residual | | | 64 × 64 |
| | Convolutional | 256 | 3 × 3 / 2 | 32 × 32 |
| | Convolutional | 128 | 1 × 1 | |
| 8× | Convolutional | 256 | 3 × 3 | |
| | Residual | | | 32 × 32 |
| | Convolutional | 512 | 3 × 3 / 2 | 16 × 16 |
| | Convolutional | 256 | 1 × 1 | |
| 8× | Convolutional | 512 | 3 × 3 | |
| | Residual | | | 16 × 16 |
| | Convolutional | 1024 | 3 × 3 / 2 | 8 × 8 |
| | Convolutional | 512 | 1 × 1 | |
| 4× | Convolutional | 1024 | 3 × 3 | |
| | Residual | | | 8 × 8 |
| | Avgpool | | Global | |
| | Connected | | 1000 | |
| | Softmax | | | |

Figure 4: Darknet-53 [46]

for feature extraction training. The detecting head for training object detector was then layered with 53 additional layers, giving YOLOv3 a set of 106 layers of fully convolutional network architecture.

The last layer of the Yolov3 framework, which was influenced by Feature Pyramid Networks (FPNs) [47], is the most distinct improvement. The last layer of Yolo v3 [46] is made up of three detection tensors, each having its own preceding boxes and double the resolution of the previous. The first tensor size might be 7750, the second 141450, and the third 282850 if each detection tensor has two previous boxes and the dataset is PascalVOC, which is partially why the authors named this "Prediction Across Scales."

Yolo v3 combines the feature extractor network's early layers with later levels (the extra CNN layers), similar to how FPNs work. Small objects are easier to recognize in high resolution early layers than in substantially subsampled low resolution later layers. However, because the early layers of a CNN have semantically weak features, FPNs blend these with up sampled later levels that contain semantically sharp features rather than utilizing them directly.

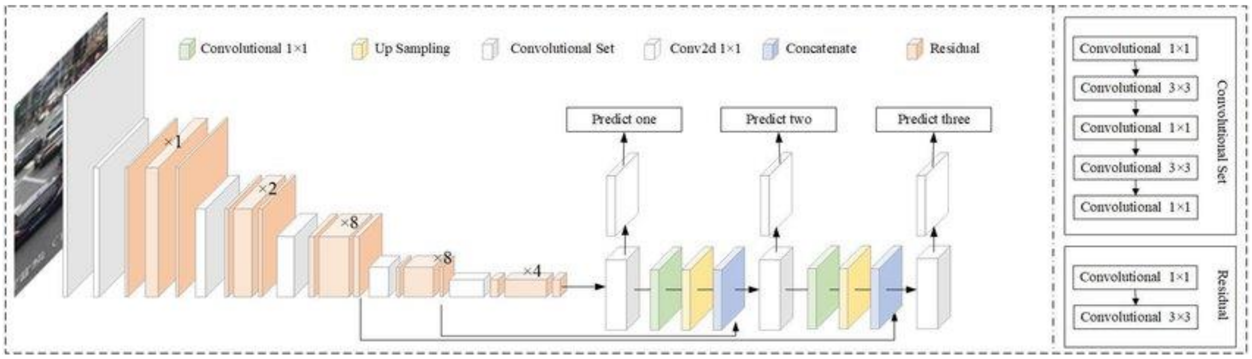


Figure 5: Architectural Diagram of YOLO v3

Class Predictions: Instead of utilizing softmax, like in YOLO v2, logistic classifiers are used for each class in YOLO v3. By doing so, we may achieve multi-label classification in YOLO v3. If the network has been trained for both a person and a man, the softmax layer will suggest a probability between the two, say 0.4 and 0.47. The probability for each item type is calculated by the independent classifier. For example, if the network is trained to differentiate between a person and a man, it will assign the person a probability of 0.85 and the man a probability of 0.8, classifying the object in the image as both a man and a human.

### 3.1.4 YOLO v4

Alexey Bochkovsky, a Russian researcher and engineer who developed the Darknet framework and three earlier YOLO architectures in C based on Joseph Redmon's theoretical theories, collaborated with Chien Yao and Hon-Yuan to release YOLOv4 in April 2020.

The feature extractor (Backbone) compresses the input picture features before sending them to the object detector (which includes the Detection Neck and Detection Head) as shown in Figure. The Detection Neck (or Neck) is a feature aggregator charged with mixing and combining the features produced in the Backbone in order to prepare for the detection phase in the Detection Head (or Head).

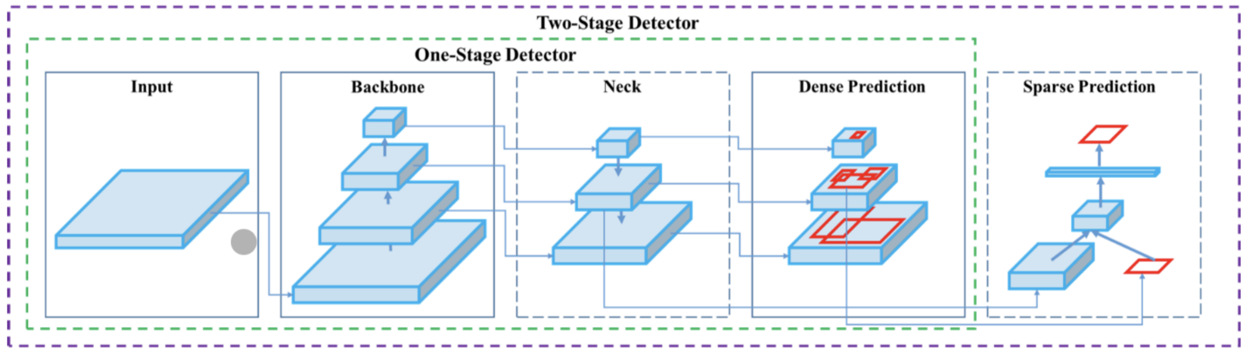


Figure 6: Architecture of object detection [48]

The distinction here is that Head is in charge of detecting each bounding box, as well as localization and categorization. The two-stage detector does these two jobs independently and then aggregates the findings (Sparse Detection), whereas the one-stage detector does so all at once (Dense Detection).

### 3.1.4.1 Architecture of YOLO v4

**Backbone:**

The authors used CSPResNext53, CSPDarknet53, and EfficientNet-B3, the most sophisticated convolutional network at the time, as the backbone (feature extractor) of the YOLOv4 model. The

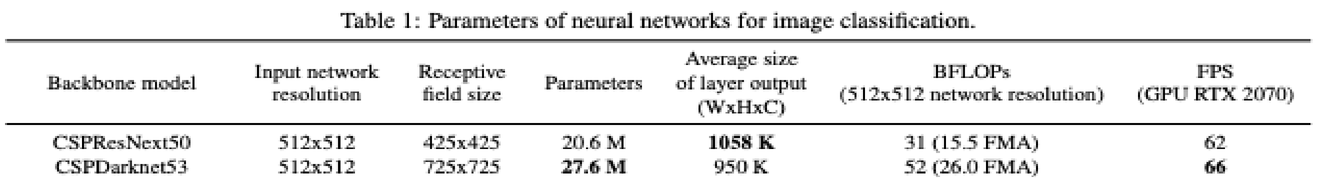


Table 1: Parameters of neural networks for image classification.

| Backbone model | Input network resolution | Receptive field size | Parameters | Average size of layer output (WxHxC) | BFLOPs (512x512 network resolution) | FPS (GPU RTX 2070) |
|---|---|---|---|---|---|---|
| CSPResNext50 | 512x512 | 425x425 | 20.6 M | **1058 K** | 31 (15.5 FMA) | 62 |
| CSPDarknet53 | 512x512 | 725x725 | **27.6 M** | 950 K | 52 (26.0 FMA) | **66** |

Figure 7: Comparison of convolutional networks

CSP Darknet53 neural network was judged to be the best optimum model based on theoretical reasoning and a lot of trials.

The CSPResNext50 and CSPDarknet53 (CSP stands for Cross Stage Partial) architectures are both developed from the DenseNet model, which takes the prior input and combines it with the current input before going into the dense layer. DenseNet was created to connect layers in an extremely deep neural network in order to reduce vanishing gradient difficulties (like ResNet does).

The CSP (Cross Stage Partial) is built on the same idea as the DenseNet described above, except that instead of using the full-size input feature map at the base layer, the input will be divided into two halves. A chunk will be sent via the thick block as normal, while another will be routed directly to the next step without being processed. As a result, various thick layers will learn uplicated gradient information again [48].

When these concepts were combined with the Darknet-53 design in YOLOv3, the leftover blocks were replaced with dense blocks. CSP preserves features through propagation, promotes the network to reuse features, lowers the number of network parameters, and aids in the preservation of fine-grained features for more efficient forwarding to deeper layers. Given that increasing the number of densely linked convolutional layers may result in a loss in detection performance, only the last convolutional block in the Darknet-53 backbone network that can extract richer semantic information is upgraded to be a dense block.

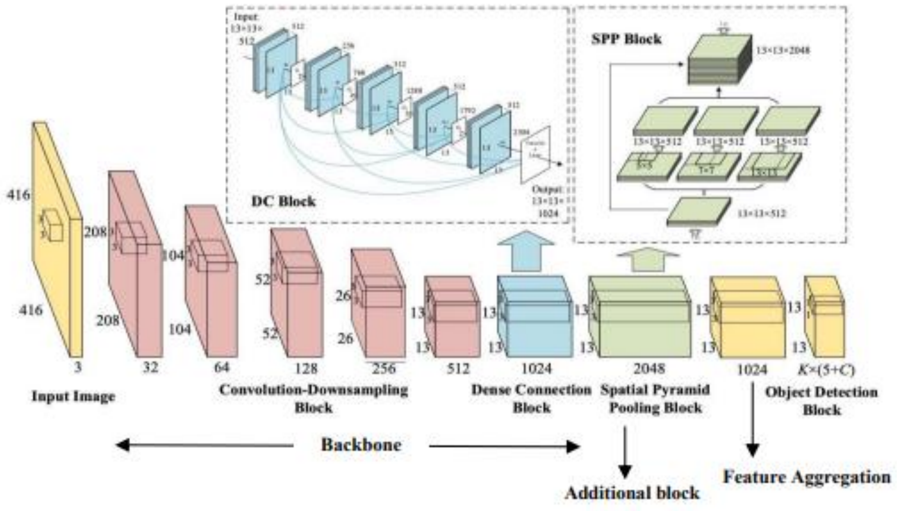


Figure 8: Dense block connection and Spatial Pyramid Pooling Based YOLOv2 architecture.

**Neck:**

To prepare for the detection stage, the next step in object detection is to mix and integrate the features produced in the ConvNet backbone. The components of the neck generally flow up and down between levels, connecting just a few layers at the convolutional network's conclusion.

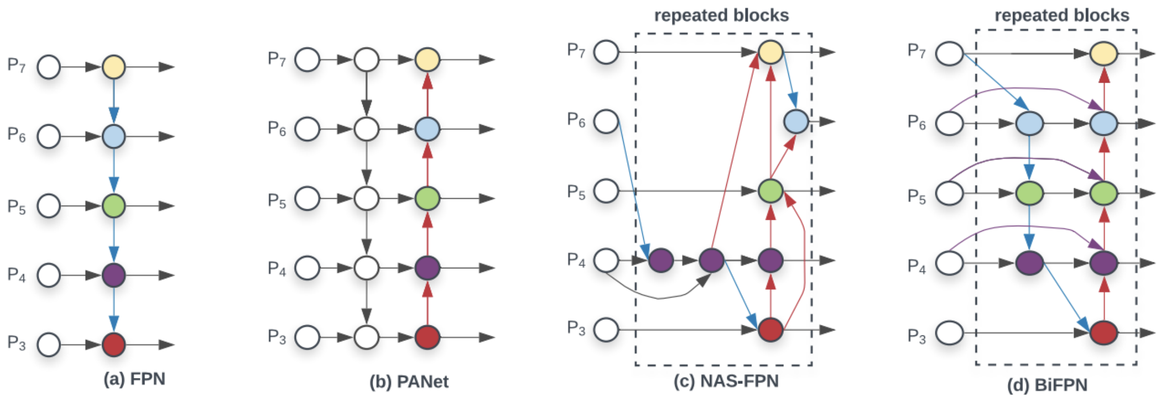


Figure 9: Representation of feature layers in CSP Darknet 53 [49]

FPN, PAN,NAS-FPN ,BiFPN, ASFF and SFAM represents a feature tier in the CSPDarknet53 backbone. The picture above is from EfficientDet, YOLOv4's predecessor. EfficientDet, created by Google Brain, employs neural architecture search to discover the optimal shape of blocks in the network's neck, resulting in NAS-FPN. The creators of EfficientDet then alter it somewhat to make the design more intuitive [49].

YOLOv4 selects PANet for network feature aggregation. YOLOv4 also includes an SPP block after CSPDarknet53 to expand the receptive area and isolate the most essential features from the backbone.

For predicting tiny objects in the large-scale detector, the FPN architecture provided a top-down approach to transmit semantical features (from the high-level layer) and then combine them to fine-grained features (from the low-level layer in the backbone).

The PAN architecture's developers offered a bottom-up augmentation method in addition to the top-down path employed in FPN. The bottom-up augmentation path is similar to the FPN top-down path in that each step has layers that create feature maps with the same spatial sizes. The element-wise addition operation connects these feature maps to the lateral architecture (Figure 11a), but the authors replaced it with the concatenation operation in the updated PAN architecture for YOLOv4 (Figure 11b).

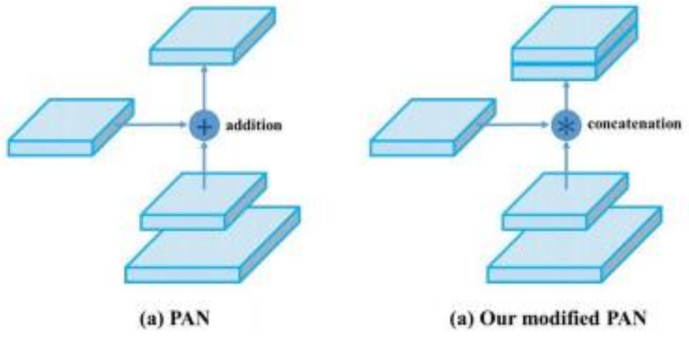


Figure 10: Original and modified PAN [49]

In FPN predictions were produced at different levels individually and independently. This can give double forecasts and does not use information from other maps. By applying ROI (region of interest) align and fully connected layers with max element-specified operation, PAN has merged all bottom-up increase pyramid output feature maps (Figure 12). So all feature map variabilities are pooled and considered for prediction.

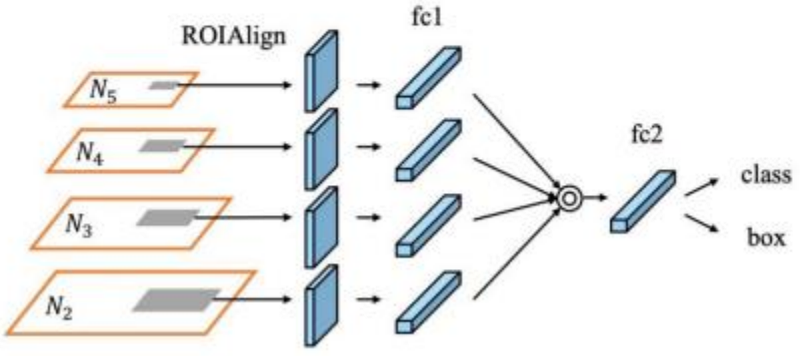


Figure 11: PAN employed ROI Aligns adaptive pooling and fully connected layers for all-stage fusing features.

**Head**

For the detection using anchor-based detection stages and three detection grades, YOLOv4 is the same head YOLO as YOLOv3. In order to further improve algorithm efficiency and accuracy, the researchers have tested and implemented a series of marginally enhanced approaches to YOLOv4 architecture. The writers refer to these enhancements as "Bag of Freebies" and "Bag of Specials."

### 3.1.5 Bag of Freebies

The bag of freebies aids in training while not raising inference time. The bag of freebies contains two techniques: the bag of freebies for the backbone, which includes the cut mix and mosiac for data augmentation and the drop block for regularization, and the bag of freebies for detection, which adds more to the backbone, such as self-adversarial training, random training shapes, and so on [49].

### 3.1.6 Bag of Specials

The bag of specials modifies the architecture and somewhat increases the inference time. The bag of specials also includes two techniques: the bag of specials for the backbone, which employs mish activation and cross-stage partial connections, and the bag of specials for detection, which employs the SPP-block, the SAM block, and others.

## 3.2 YOLO v5 (You look only once Version 5)

We have used yolov5 for head detection and crowd counting approach which outperforms all the previous versions of yolo in terms of training time and speed. YOLO (You Only Look Once) is one of the most prominent algorithms which performs real time detection of objects with highest accuracy.

### 3.2.1 Variants of YOLOv5

The current framework uses YOLOv5s, which is the smallest model, YOLOv5m, which is the next largest model, YOLOv5l it is the lager one, and YOLOv5xl, that is the largest model. As the network grows in size, its performance may improve as well, although at the cost of longer processing times.

### 3.2.2 Architectural Diagram

We have used the YOLOv5 version for training our model due to three reasons. First, in Yolov5 architecture, the CSPNet [48] network was integrated into the Darknet which created CSPDarknet [48]. CSPNet solves gradient information issues and integrates gradient changes into the feature map, lowering model parameters and FLOPS (floating point operations per second). It reduces model size and, when trained on the ImageNet database, cuts calculations by 20% when compared to other state-of-the-art detectors [48]. Second, the Yolov5 uses a route aggregation network (PANet) [50] as its neck to increase information flow. PANet employs a unique feature pyramid network (FPN) architecture with an enhanced bottom-up approach to boost low-level feature propagation. Simultaneously, adaptive feature pooling, which connects the feature grid and all feature levels, is used to extract useful information in each feature level and then it propagates directly to the next subnetwork. PANet improves the utilization of accurate localization signals in lower layers, which obviously improves the object's location accuracy. Third, the Yolo layer i-e the head of Yolov5, creates three distinct sizes (18x18, 36x 36, 72 x72) of feature maps to provide multi-scale prediction, allowing the model to handle smaller, medium, and large objects.

In Yolov5 official code, there are 4 object detection models namely Yolov5s, Yolov5m, Yolov5l and Yolov5x. We have used the yolov5vs as it is the smallest network. Like other single-stage detectors, YOLO v5 has three important parts which are:

1. Backbone
2. Neck
3. Prediction

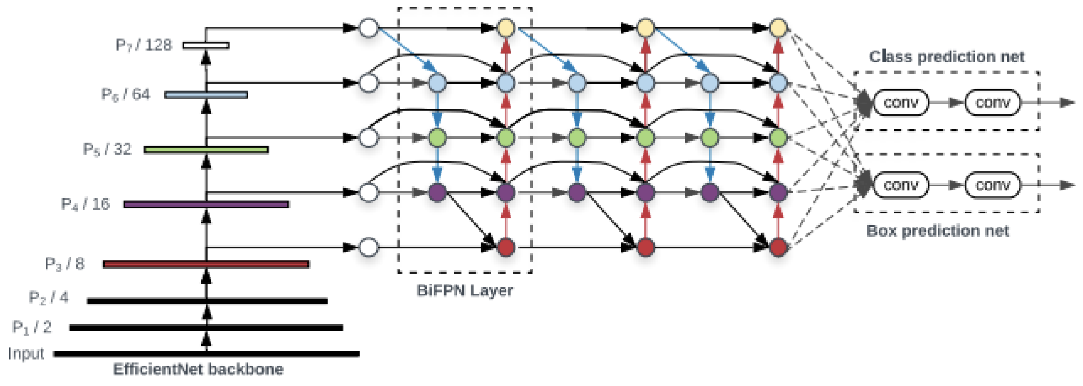


Figure 13: Yolo v5 Architecture

**Backbone:**

Model Backbone is mostly used to extract key features from an input image. Backbone is divided into these segments i.e., Focus and CSP structure.

The focus structure basically performs the slicing operation, and there is no such structure in previous yolo versions. The focus structure takes an image as an input and slice it into multiple feature maps. We have used the Yolov5s architecture. Suppose the image of 608*608*3 shape is provided as input into the Focus structure of Yolo5s. It will first create a 304*304*12 feature map by concatenating all the feature maps. Then it is fed to convolution layer which performs convolution operation of 32 kernels to produce 304*304*32 feature map.

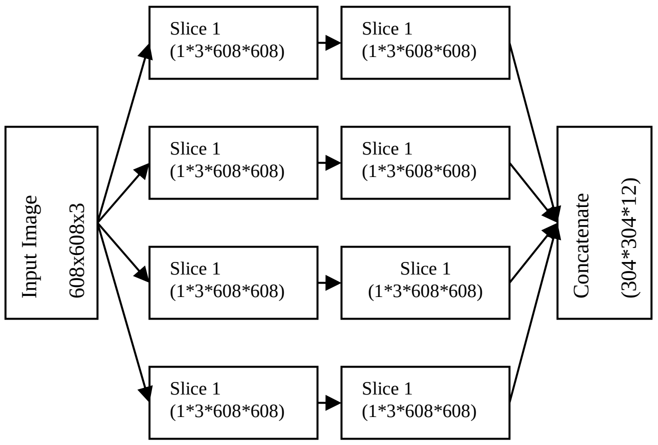


Figure 14: Focus Operation

Backbone of YOLO v5 uses CSPNet to extract rich and meaningful features from an input image. With deeper networks, CSPNet has shown a considerable reduction in processing time. Deeper neural networks have been shown to be more powerful in feature extraction which brings up more computations and increased training time. Thus, making object detection tasks unaffordable. On the other hand, the accuracy of neural networks decreases on reducing layers. The CSPNet basically strengthens the feature learning capability of convolutional neural networks. Therefore, high accuracy can be obtained with reduced computations. Yolov5 toYolov5s network has two CSP structures i.e., CSP1X and CSP2X. CSP1 X structure is used in the Backbone network, whereas the CSP2 X structure is used in the Neck. Because Backbone has five CSP modules and the input image is 608*608, the feature map change rule is as follows: 608->304->152->76->38->19. A 19*19 feature map is obtained after 5 CSP modules. In Backbone, the author only employs the Mish activation function, while the Leaky relu activation function is employed behind the network. Backbone also includes an important module i.e., SPP. In the SPP module, the author uses the maximum pooling method with kernels of sizes k={1*1,5*5,9*9,13*13}, and then performs concatenation operation on feature maps of different scales.

**Model Neck*:***

The Model Neck creates feature pyramids which detects the small and occluded objects accurately which is quite challenging. When it comes to object scaling, feature pyramids help models generalize successfully. These pyramids are used to identify the same thing in different sizes and scales. They are quite useful in aiding models to perform well on previously unknown data. Other models, such as FPN, BiFPN, and PANet, make use of other types of feature pyramid techniques.

The FPN and PAN module is used as a neck in YOLO v5 to generate feature pyramids. FPN is made up of two pathways: bottom-up and top-down. For feature extraction, it uses the standard convolutional network.

**Model Head:**

The model Head is mostly responsible for the final detection step. It uses anchor boxes to construct the final vectors consisting of output class along with class probabilities objectness scores and parameters defining the boundary box coordinates. Head is the last part of object detection model. It takes feature maps generated from neck as an input and performs the prediction step and gives the co-ordinates of the bounding box.

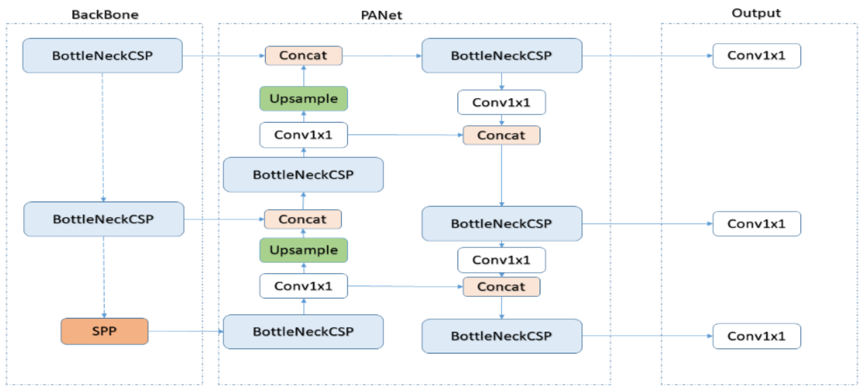


Figure 12: Architecture of yolov5

# Chapter 4

# Experiments & Results

In this section the experiment results on several benchmark datasets are discussed. Several detection results were computed by training and testing the most authentic and extensively used datasets on Yolov5. Different datasets comprise of different sizes of heads and then by testing another dataset the yolov5 performance is analyzed. After that another dataset is formulated which include all the most accurate pictures for head detection having higher precision rate. That dataset is then trained and accuracy of head detection and crowd counting is improved.

## 4.1 Datasets

The qualitative and quantitative evaluation of the experimental outcomes is discussed in this section. Three publicly available datasets, Casablanca, Scut Head, and Hollywood Heads dataset, are used to assess our technique. Indoor, outdoor, and crowded scenarios are all included in these datasets. Aside from that, there are many other head sizes, such as small, medium, very small, and very huge. All these datasets are annotated and the dimensions of bounding boxes are provided in the form of ground truth values.

*Casablanca dataset:* The Casablanca dataset includes pictures from the film Casablanca. There are 1466 frames in all, with annotated head bounding boxes. The Hollywood dataset is annotated similarly to the Casablanca dataset, with the exception that the frontal head annotation has been reduced to faces. Casablanca is an old video with an unusual scenario for head detection, with visuals that are greyscale, have low illumination, and are crowded. Monochromic photos with a resolution of 464x640 pixels constitute the dataset. Due of the dense background, large differences in scales, positions, and appearances of human heads, the dataset is extremely complicated to handle.

*Hollywood Heads dataset:* The dataset is the largest in the world, with 224,740 photos drawn from 21 distinct Hollywood films. The collection contains 369,846 annotations (bounding boxes) that cover human heads of various sizes, shapes, and looks. The data is separated into three groups. The set includes 216,719 photos for training the models, which span 15 different films. The second batch contains 6,719 photos from three films for validation, while the third set has 1,302 images from the remaining three films for testing.

*Scut Head dataset:* Part A and Part B are the two portions of the dataset. Part A contains 2000 image samples taken from a zoom-out camera mounted in the corner of a university classroom. Part A has 67,321 human head annotations. Human heads usually have similar positions and orientations inside the classroom, thus the photographs are carefully picked to minimize the resemblance and increase the variance across the images. Part A's population density ranges from 0 to 90 people per picture, with an average count of 51.8.

Table 1: Dataset Description

| Datasets | Images | Pixels | Head size |
|---|---|---|---|
| Casablanca | 1466 | 464x640 | All mostly Medium |
| Hollywood heads | 224,740 | 384x864, 640x864 & 480x864 | Various sizes |
| Scut Head (PartA) | 67,321 | 384x640 | Small |

## 4.2 Experiments

Several experiments are performed to analyze the performance of our proposed methodology.

### 4.2.1 Training Casablanca

Casablanca dataset is considered to be the most commonly available and used dataset for human detection or crowd counting. As Casablanca includes the black and white images and include mostly large and medium heads. The following table shows all the parameters used for training the dataset.

Table 2: Parameters for training Casablanca dataset

| Parameters | Values |
|---|---|
| Input Size | 650x650 |
| Output Size | 650x650 |
| Batch | 10 |
| Epoch | 100 |
| Learning Rate | 0.01 |
| Confidence threshold | 0.4 |

|  |  |
|---|---|
| Weight decay | 0.0005 |
| Data Split | 70% training, 20 % Val, 10% testing |

Batch size is varied from 10 to 16 and for the better results in less time batch size 10 is selected. Epochs size was varied from 50 to 200 and the best result was obtained at 100.

### 4.2.2 Training Hollywood heads Dataset

Hollywood heads dataset is the most extensive dataset of human heads containing 224,740 photos. It contains all the colored images. I have used Google Colab pro for training the dataset and it cannot handle large dataset therefore I have just considered the most useful and non-repeated images for my experiments. As the dataset is large batch size was varied from 10 to 15 and at 16 best results were achieved. Following parameters are used to train the dataset:

Table 3: Parameters for training Hollywood Heads dataset

| **Parameters** | **Values** |
|---|---|
| Input Size | 650x650 |
| Output Size | Variable |
| Batch | 15 |
| Epoch | 100 |
| Learning Rate | 0.01 |
| Confidence threshold | 0.4 |
| Weight decay | 0.0005 |
| Data Split | 70% training, 20 % Val, 10% testing |

### 4.2.3 Training on Scut Head dataset:

Scut Head dataset comprises of small heads only. It is publically available. In this research only Dataset Part A is used to analyze the results as both the dataset contain like images. The accuracy rate for this dataset is not satisfactory as it misses the medium and large heads. Following parameters are used for training the dataset:

Table 4: Parameters used for training Scut head dataset

| **Parameters** | **Values** |
|---|---|
| Input Size | 650x650 |
| Output Size | 384x640 |
| Batch | 10 |
| Epoch | 100 |
| Learning Rate | 0.01 |
| Confidence threshold | 0.4 |
| Weight decay | 0.0005 |
| Data Split | 70% training, 20 % Val, 10% testing |

### 4.2.4 Merge Dataset

All other datasets contain limited size of heads i-e Casablanca contain medium size of heads, Hollywood heads contain variable but colored heads and Scut Head contain only small heads. In this research in order to improve the accuracy of the head detection all these three datasets are combined to form one dataset containing every type of head. This dataset is named as merge dataset and it is further trained on yolov5. It contains 15466 images. This dataset is divided into training, testing and validation in the ratio of 70% training, 20 % Val and 10% testing. In other words, 10826 are used for training, 1546 for testing and 3093 for validation.

Following parameters are used for carrying out this experiment:

Table 4: Parameters used for the Merge Dataset

| **Parameters** | **Values** |
|---|---|
| Input Size | 850x850 |
| Output Size | Variable |
| Batch | 15 |
| Epoch | 100 |
| Learning Rate | 0.01 |
| Confidence threshold | 0.4 |
| Weight decay | 0.0005 |
| Data Split | 70% training, 20 % Val, 10% testing |

.

## 4.3 Results and Discussion

### 4.3.1 Mean Average Precision (mAP)

The mean average precision (mAP) is used to evaluate object detection algorithms such as R-CNN and YOLO. The mAP computes a score by comparing the ground-truth bounding box to the detected box. The higher the score, the more precise the model's detections.

The mean Average Precision, or mAP score, is computed by averaging the AP over all classes and/or the total IoU thresholds, depending on the detecting constraints. The mAP score indicates how well your model predicted the bounding box. The higher the score, the greater the accuracy. The mAP metric, which reflects an algorithm's accuracy, is used to evaluate the outcomes of this algorithm.

$$\textit{Intersection over Union (IoU)} = \frac{\textit{Area of Overlap}}{\textit{Area of Union}}$$

### 4.3.2 Precision

Precision= TP/ (TP + FP)

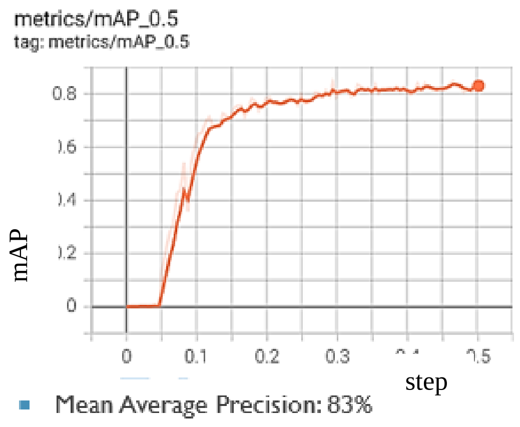
Where TP and FP indicates true positives and False Positives. Recall measures how well you find all the positives. Precision measures how accurate your predictions. In Precision IOU threshold value is fixed and predicted IOU are compared with the fixed value to evaluate the results.

### 4.3.3 Qualitative and Quantitative Analysis

Casablanca Dataset: The mAP of the Casablanca is high i-e 83%. Hollywood heads mAP is 70.9% and the Scut head dataset contain all small heads so its mAP is very less i-e 6%. Results are quite obvious from the table 5. Therefore, the draw back in using this dataset was its accuracy rate detecting small heads is very less and we cannot use for a stable model.

Table 5: Results obtained when yolov5 is trained on the Casablanca dataset and tested on the Casablanca, Hollywood Heads and Scuthead dataset

| **Trained Casablanca dataset** | | | |
|---|---|---|---|
| **Testing dataset** | Training mAP | Testing mAP | Detection time |
| **Casablanca** | 0.83% | mAP= 83.87% | 0.538s |
| **Hollywood Heads** | | mAP = 70.92% | 0.77s |
| **Scut Head** | | mAP = 6% | 0.512s |



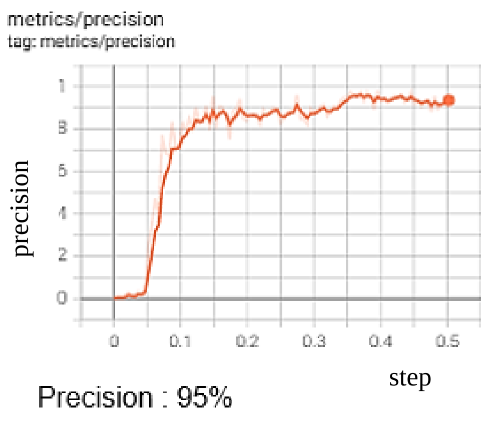


Figure 13: Mean average precision and Precision when trained the Yolov5 on Casablanca Dataset

Most of the heads are detected in the Casablanca dataset while small heads are not detected in Hollywood heads and Scut head dataset. The Detection time is very less in this method.

**Results on Casablanca Dataset:**

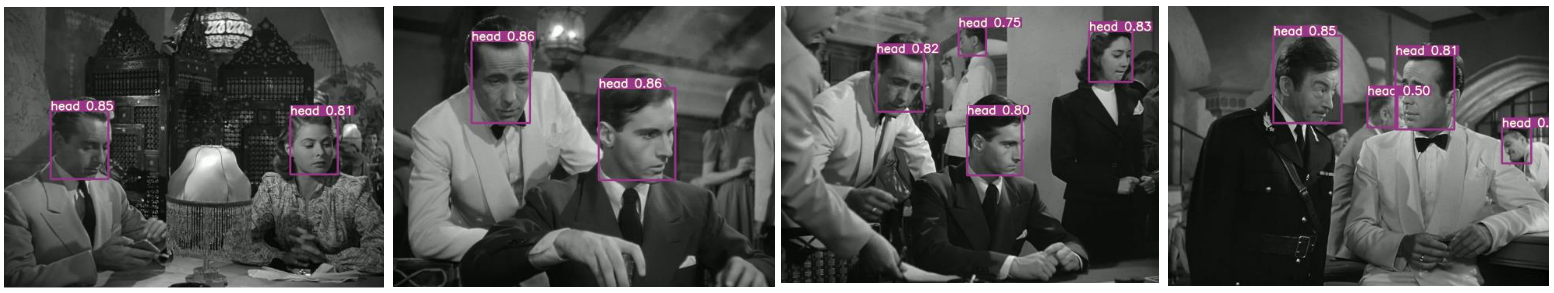
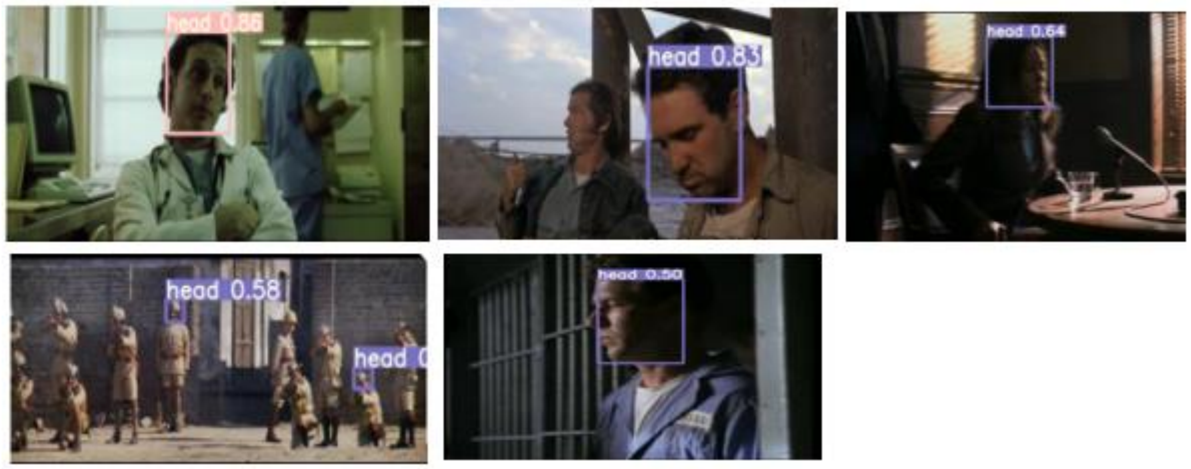


Figure 14: Head detection results using Casablanca dataset for training yolov5 and testing the Casablanca dataset

**Results on Hollywood Heads Dataset:**



\

Figure 15: Head detection results using dataset for training Casablanca dataset on yolov5 and testing Hollywood Heads dataset

**Results on Scuthead Dataset:**

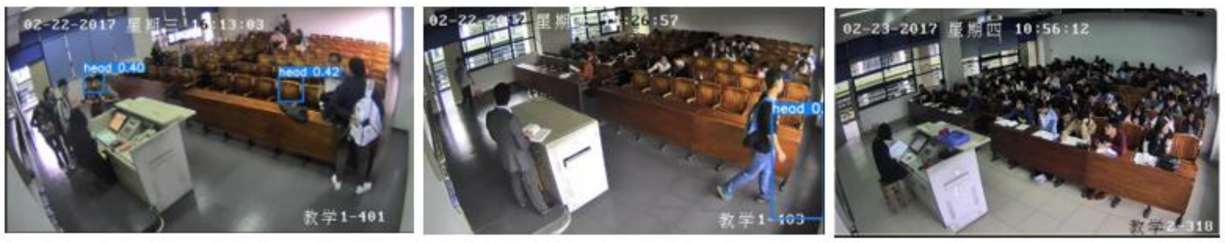


Figure 16: Head detection results using dataset for training Casablanca dataset on yolov5 and testing ScutHead dataset

*Hollywood Heads Dataset:* Training mAP of this dataset is very high. It contains mostly medium size heads and it lags the accuracy rate in detecting small heads. As mentioned in the below table 3, the mAP of the Scut head is 0% as it contains all the small heads, Casablanca dataset is 31.8% as these are all black and white images while the mAP of the Hollywood Heads is 56%. We cannot consider it the most stable dataset as it is not detecting small heads.

Table 6: Mean Average obtained when yolov5 is trained on the Hollywood Heads dataset and tested on the Casablanca and Scuthead dataset

| **Trained Hollywood Heads dataset** | | | |
|---|---|---|---|
| **Testing dataset** | Training mAP | Testing mAP | Detection time |
| **Casablanca** | 0.98 % | 31.8% | 2.135s |
| **Hollywood Heads** | | 56% | 0.821s |
| **Scut Head** | | 0% | 0.61s |

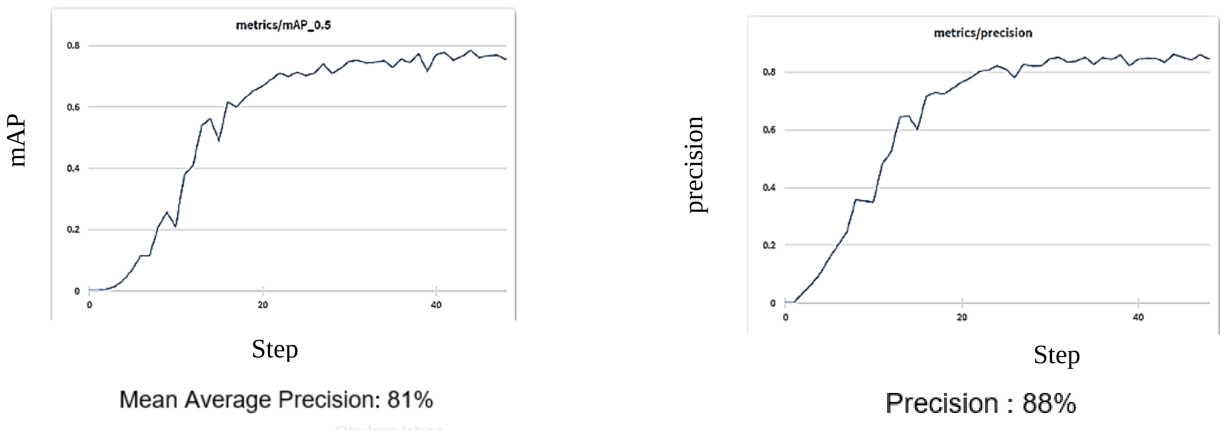


Figure 17 Mean average precision and Precision when trained the Yolov5 on Hollywood Heads Dataset

**Results on Casablanca dataset:**

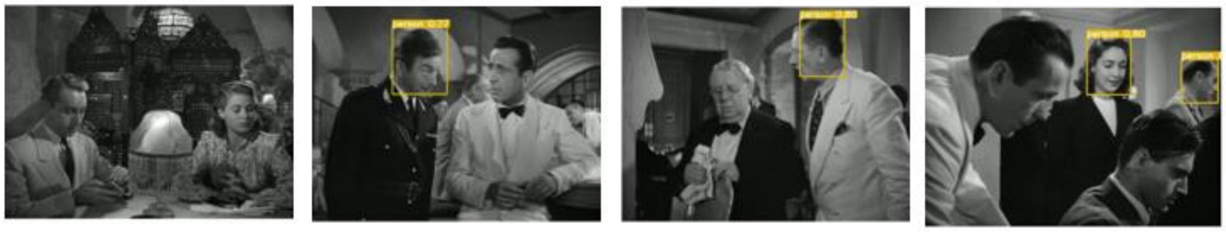


Figure 18: Head detection results using Hollywood heads dataset for training and Casablanca dataset for testing on yolov5

**Results on Hollywood Heads Dataset:**

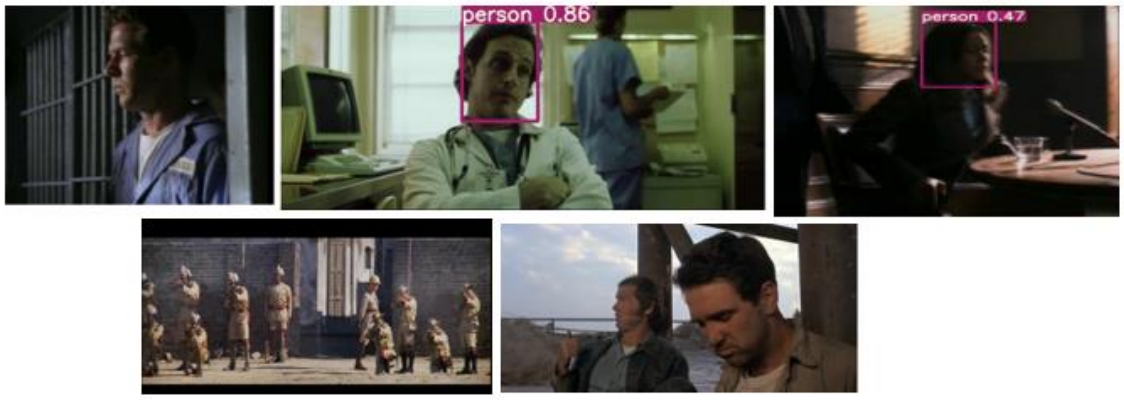


Figure 19: Head detection results using Hollywood heads dataset for training and Hollywood Heads dataset for testing on yolov5

**Results on Scuthead Dataset:**

Figure 20: Head detection results using Hollywood heads dataset for training and ScutHeads dataset for testing on yolov5

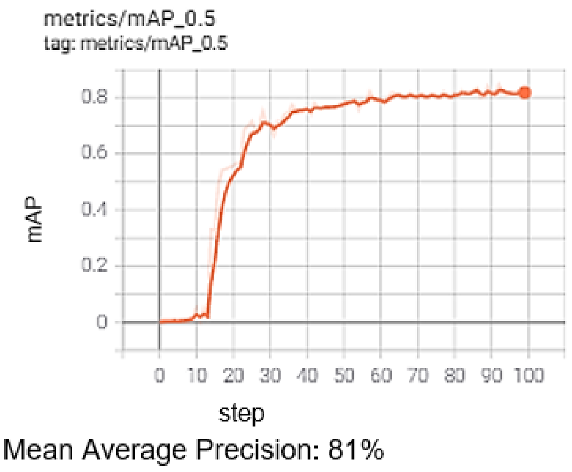
The detecting accuracy of this method is very less. It detects medium heads with greater precision but strongly ignore the small heads.  The detection time is greater as compared to the Casablanca dataset training.

**Scut Head Dataset:**

As this dataset comprises of only small heads it completely ignores the medium and large size heads, The mAP of Casablanca and Hollywood Heads is very less but it detects the scut head dataset with 66% mAP. The detection rate is very less in this as compared to the other methods.

Table 7: Mean average precision and Detection rate obtained when yolov5 is trained on the Scut Heads dataset and tested on the Casablanca, Hollywood Heads and Scuthead dataset

| **Trained Scut Head Dataset** | | | |
|---|---|---|---|
| **Testing dataset** | **Training mAP** | **Testing mAP** | **Detection time** |
| Casablanca | | mAP = 30.70% | 3.52s |
| Hollywood Heads | 0.82% | mAP = 11.25% | 0.797s |
| Scut Head | | mAP = 66.85% | 0.537s |



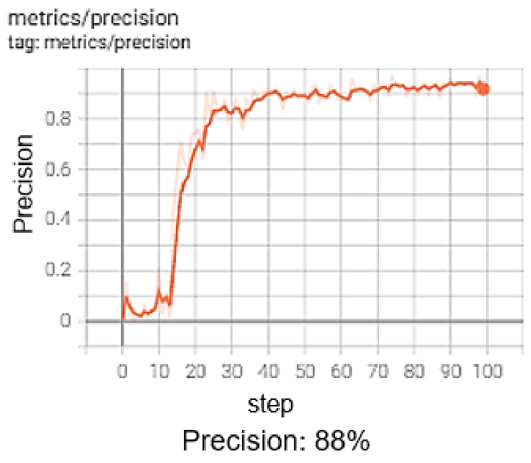


Figure 22: Mean average precision and Precision when trained the Yolov5 on Scut Heads Dataset

**Detections results on Casablanca:**

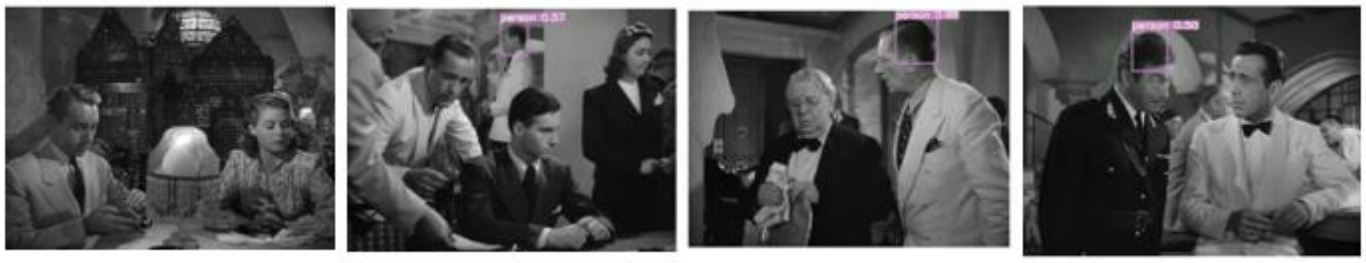


Figure 21 Head detection results using Scutheads dataset for training and Casablanca dataset for testing on yolov5

**Detections results on Hollywood Heads:**

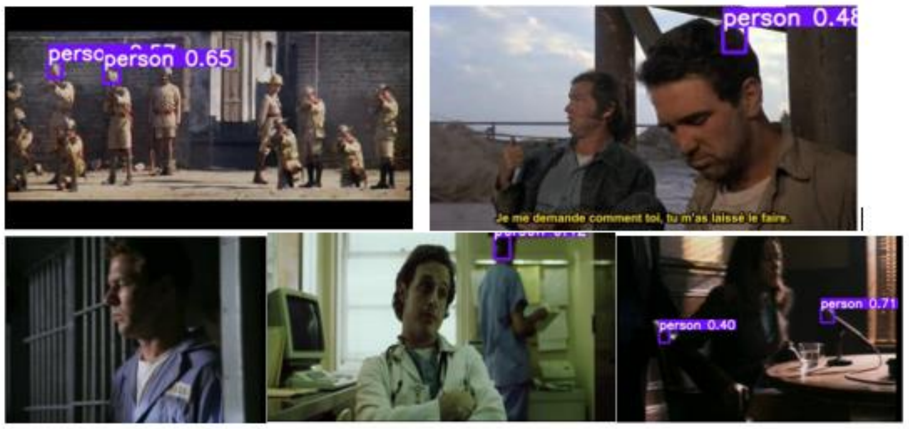


Figure 22: Head detection results using Scutheads dataset for training and Hollywood Heads dataset for testing on yolov5

**Detections results on Scut Heads:**

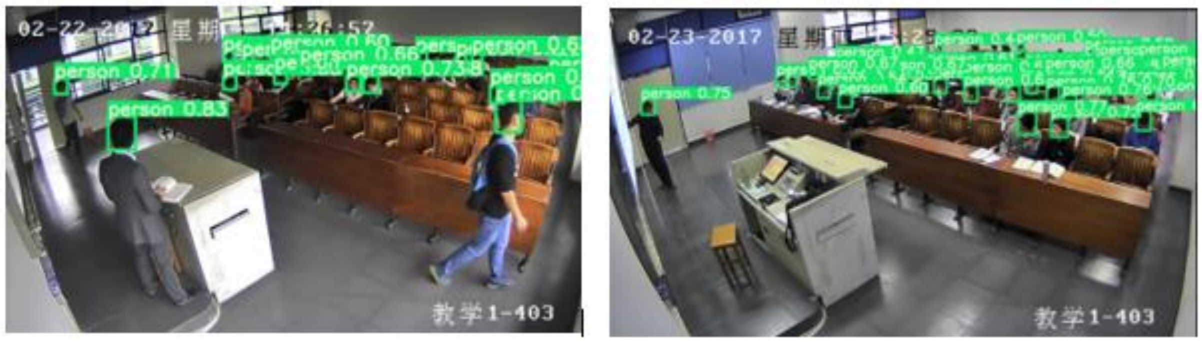


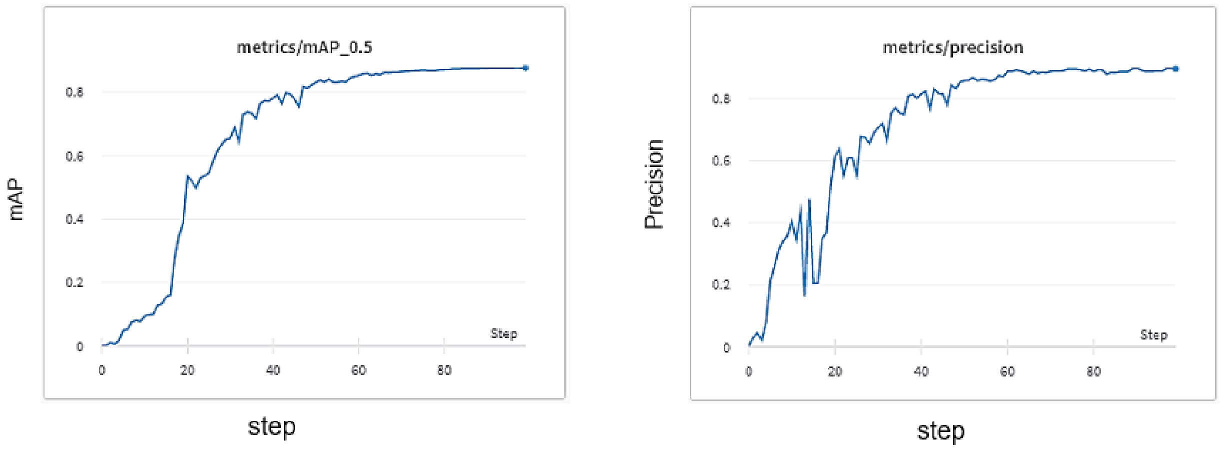
Figure 23 Head detection results using Scutheads dataset for training and ScutHead dataset for testing on yolov5

**Merge dataset:**

This dataset comprises of the images of all previous dataset used. Therefore, we can say it contains every kind of Head size in different scenarios from multiple angles. The mAP of this dataset is improved and more stable as compared to the other datasets. Its detection rate is also high as compared to the previous experiments. Following table summarizes the results of testing other dataset by using this dataset for training:

Table 8: Mean average precision and Detection rate obtained when yolov5 is trained on the Merge dataset and tested on the Casablanca, Hollywood Heads and Scuthead dataset.

| **Trained Merge dataset** | | | |
|---|---|---|---|
| **Testing dataset** | Training mAP | Testing mAP | Detection time |
| **Casablanca** | 0.97% | mAP = 72.70% | 7.072s |
| **Hollywood Heads** | | mAP = 75.25% | 0.78s |
| **Scut Head** | | mAP = 78.85% | 8.94s |



Mean Average Precision: 87%

Precision: 89%

Figure 24: Mean average precision and Precision when trained the Yolov5 on Merge Dataset

**Results on Casablanca dataset:**

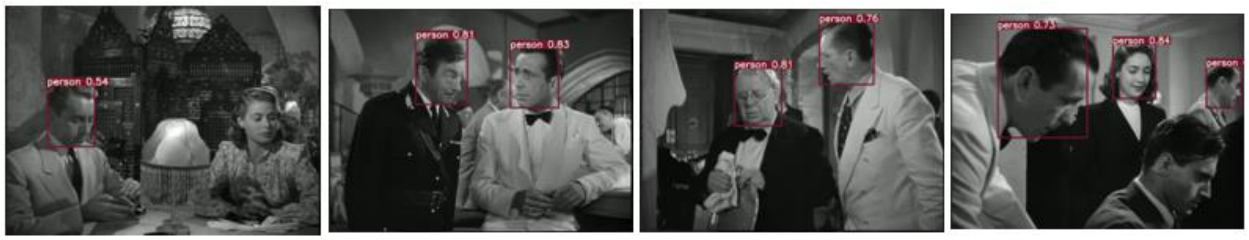


Figure 25: Head detection results using Merge Dataset for training and Casablanca dataset for testing on yolov5

**Results on Hollywood Heads Dataset:**

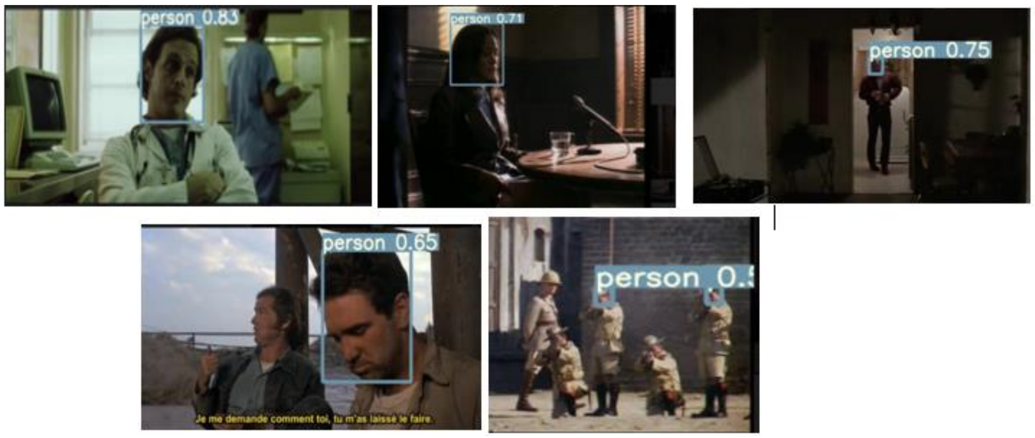


Figure 26: Head detection results using Merge Dataset for training and Hollywood Heads for testing on yolov5

**Detections results on Scut Head Dataset:**

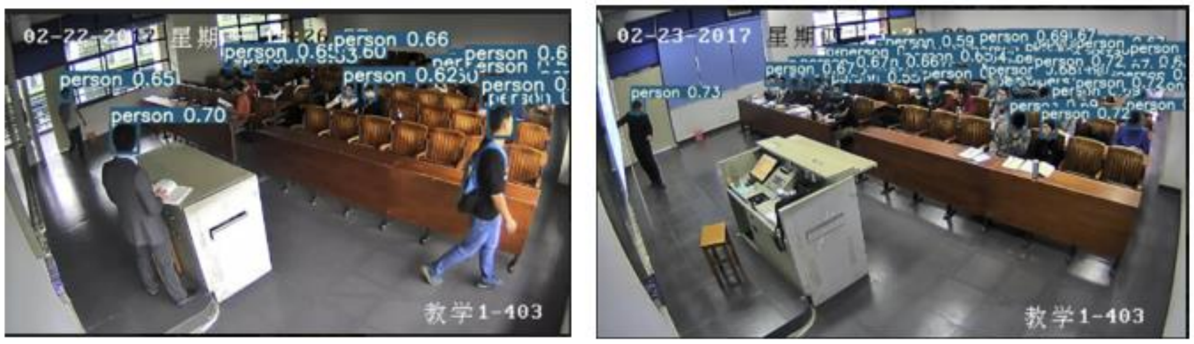


Figure 27: Head detection results using Merge Dataset for training and ScutHead dataset for testing on yolov5

It is obvious from the above results that using this approach most of the heads are detected and the miss rate is very less.

## 4.4 Comparison

The merge dataset yields the best and the most accurate results so far. The mAP of the Merge Dataset is higher than the Scut head and Hollywood heads while less than the Casablanca itself i-e when we train our model using the Merge dataset give more accurate results than Hollywood heads and scut head dataset. It means that our precision rate of improved and this pre-trained model head detection rate is more than our previous experiments.

Table 9 summarizes the comparison of the mAP merge datasets with other datasets.

Table 9 Comparison of the results obtained from merge dataset with other datasets

| **Datasets** | **Training mAP** | **Casablanca mAP** | **Hollywood heads mAP** | **Scut Head mAP** |
|---|---|---|---|---|
| **Casablanca** | 0.83% | 83.87% | 70.92% | 6% |
| **Hollywood heads** | 0.98% | 31.85 | 56% | 0% |
| **Scut Head** | 0.82% | 30.7% | 11.25% | 66.85% |
| **Merge dataset** | 0.81% | 72.7% | 75.5% | 78% |

# Chapter 5

# Conclusion & Future Work

## 5.1 Conclusion

Crowd Counting via head detection using deep learning is a new approach in this area. In this research Multiple datasets re used to evaluate the architecture performance. After obtaining the results another dataset is formulated which contain all the images of other dataset and hence result in the stabilized pertained model. Other main goal achieved by this research is the improvement in the accuracy rate as compared to the other architectures. The rate of detecting the Small heads in raised from 66% to 78%.

## 5.2 Future Work

In future the idea is to extend the same idea to High density crowds for example in Haram there are billions of people. In such scenarios it is approximately impossible to detect every head. Therefore, we cannot achieve the accurate results. For Crowd Counting in such scenarios dot annotations are used. The idea is to develop another architecture to carry out this research and to overcome the challenges in detection high density crowd.

## References

[1] G. Sreenu and M. A. S. Durai, “Intelligent video surveillance: a review through deep learning techniques for crowd analysis,” *J. Big Data*, vol. 6, no. 1, pp. 1–27, 2019.

[2] S. A. M. Saleh, S. A. Suandi, and H. Ibrahim, “Recent survey on crowd density estimation and counting for visual surveillance,” *Eng. Appl. Artif. Intell.*, vol. 41, pp. 103–114, 2015.

[3] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[4] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[5] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” *Adv. Neural Inf. Process. Syst.*, vol. 28, pp. 91–99, 2015.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE conference on computer vision and pattern recognition*, 2009, pp. 248–255.

[7] H. Rahmalan, M. S. Nixon, and J. N. Carter, “On crowd density estimation for surveillance,” 2006.

[8] M. Bendali-Braham, J. Weber, G. Forestier, L. Idoumghar, and P.-A. Muller, “Recent trends in crowd analysis: A review,” *Mach. Learn. with Appl.*, p. 100023, 2021.

[9] B. E. Moore, S. Ali, R. Mehran, and M. Shah, “Visual crowd surveillance through a hydrodynamics lens,” *Commun. ACM*, vol. 54, no. 12, pp. 64–73, 2011.

[10] D. Ryan, S. Denman, S. Sridharan, and C. Fookes, “An evaluation of crowd counting methods, features and regression models,” *Comput. Vis. Image Underst.*, vol. 130, pp. 1–17, 2015.

[11] K. Khan, W. Albattah, R. U. Khan, A. M. Qamar, and D. Nayab, “Advances and trends in real time visual crowd analysis,” *Sensors*, vol. 20, no. 18, p. 5073, 2020.

[12] V. A. Sindagi and V. M. Patel, “A survey of recent advances in cnn-based single image crowd counting and density estimation,” *Pattern Recognit. Lett.*, vol. 107, pp. 3–16, 2018.

[13] B. Zhan, D. N. Monekosso, P. Remagnino, S. A. Velastin, and L.-Q. Xu, “Crowd analysis: a survey,” *Mach. Vis. Appl.*, vol. 19, no. 5, pp. 345–357, 2008.

[14] T. Zhao, R. Nevatia, and B. Wu, “Segmentation and tracking of multiple humans in crowded environments,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 7, pp. 1198–1211, 2008.

[15] S.-F. Lin, J.-Y. Chen, and H.-X. Chao, “Estimation of number of people in crowded scenes using perspective transformation,” *IEEE Trans. Syst. Man, Cybern. A Syst. Humans*, vol. 31, no. 6, pp. 645–654, 2001.

[16] B. Leibe, E. Seemann, and B. Schiele, “Pedestrian detection in crowded scenes,” in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, 2005, vol. 1, pp. 878–885.

[17] I. S. Topkaya, H. Erdogan, and F. Porikli, “Detecting and tracking unknown number of objects with Dirichlet process mixture models and Markov random fields,” in *International Symposium on Visual Computing*, 2013, pp. 178–188.

[18] M. Li, Z. Zhang, K. Huang, and T. Tan, “Estimating the number of people in crowded scenes by mid based foreground segmentation and head-shoulder detection,” in *2008 19th international conference on pattern recognition*, 2008, pp. 1–4.

[19] B. Wu and R. Nevatia, “Tracking of multiple, partially occluded humans based on static body part detection,” in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06)*, 2006, vol. 1, pp. 951–958.

[20] C. C. Loy, K. Chen, S. Gong, and T. Xiang, “Crowd counting and profiling: Methodology and evaluation,” in *Modeling, simulation and visual analysis of crowds*, Springer, 2013,

pp. 347–382.

[21] A. B. Chan, Z.-S. J. Liang, and N. Vasconcelos, “Privacy preserving crowd monitoring: Counting people without people models or tracking,” in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–7.

[22] J. Mei and Z. Yanyun, “An Improved Method of Crowd Counting Based on Regression,” 2013.

[23] J. Yang, Y. Zhou, and S.-Y. Kung, “Multi-scale generative adversarial networks for crowd counting,” in *2018 24th international conference on pattern recognition (ICPR)*, 2018, pp. 3244–3249.

[24] H. Yao, K. Han, W. Wan, and L. Hou, “Deep spatial regression model for image crowd counting,” *arXiv Prepr. arXiv1710.09757*, 2017.

[25] X. Chen, M. Liu, J. Ren, and C. Zhao, “An Overview of Crowd Counting on Traditional and CNN-based Approaches,” in *2020 6th International Conference on Robotics and Artificial Intelligence*, 2020, pp. 126–133.

[26] V. Lempitsky and A. Zisserman, “Learning to count objects in images,” *Adv. Neural Inf. Process. Syst.*, vol. 23, pp. 1324–1332, 2010.

[27] V.-Q. Pham, T. Kozakaya, O. Yamaguchi, and R. Okada, “Count forest: Co-voting uncertain number of targets using random forest for crowd density estimation,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 3253–3261.

[28] H. Luo *et al.*, “A high-density crowd counting method based on convolutional feature fusion,” *Appl. Sci.*, vol. 8, no. 12, p. 2367, 2018.

[29] B. Xu and G. Qiu, “Crowd density estimation based on rich features and random projection forest,” in *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2016, pp. 1–8.

[30] S. Jia and Q. Zhao, “CNN-based method for Crowd Counting with additional learning of

Image semantic understanding," in *Proceedings of the 12th International Convention on Rehabilitation Engineering and Assistive Technology*, 2018, pp. 303–306.

[31] M. Fu, P. Xu, X. Li, Q. Liu, M. Ye, and C. Zhu, "Fast crowd density estimation with convolutional neural networks," *Eng. Appl. Artif. Intell.*, vol. 43, pp. 81–88, 2015.

[32] D. B. Sam, S. V. Peri, M. N. Sundararaman, A. Kamath, and V. B. Radhakrishnan, "Locate, size and count: Accurately resolving people in dense crowds via detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2020.

[33] W. Liu, M. Salzmann, and P. Fua, "Context-aware crowd counting," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5099–5108.

[34] E. Walach and L. Wolf, "Learning to count with cnn boosting," in *European conference on computer vision*, 2016, pp. 660–676.

[35] A. Vora and V. Chilaka, "FCHD: Fast and accurate head detection in crowded scenes," *arXiv Prepr. arXiv1809.08766*, 2018.

[36] M. A. Kızrak and B. Bolat, "Cluster-Based Monitoring and Location Estimation for Crowd Counting," in *International Online Conference on Intelligent Decision Science*, 2020, pp. 240–253.

[37] L. Boominathan, S. S. S. Kruthiventi, and R. V. Babu, "Crowdnet: A deep convolutional network for dense crowd counting," in *Proceedings of the 24th ACM international conference on Multimedia*, 2016, pp. 640–644.

[38] U. Bhangale, S. Patil, V. Vishwanath, P. Thakker, A. Bansode, and D. Navandhar, "Near Real-time Crowd Counting using Deep Learning Approach," *Procedia Comput. Sci.*, vol. 171, pp. 770–779, 2020.

[39] X. Zeng, Y. Wu, S. Hu, R. Wang, and Y. Ye, "DSPNet: Deep scale purifier network for dense crowd counting," *Expert Syst. Appl.*, vol. 141, p. 112977, 2020.

[40] S. D. Khan, A. B. Altamimi, M. Ullah, H. Ullah, and F. A. Cheikh, "TCM: Temporal

Consistency Model for Head Detection in Complex Videos," *J. Sensors*, vol. 2020, 2020.

[41] S.-C. Hsu, C.-L. Huang, and C.-H. Chuang, "Vehicle detection using simplified fast R-CNN," in *2018 International Workshop on Advanced Image Technology (IWAIT)*, 2018, pp. 1–3.

[42] R. Qian, Q. Liu, Y. Yue, F. Coenen, and B. Zhang, "Road surface traffic sign detection with hybrid region proposal and fast R-CNN," in *2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, 2016, pp. 555–559.

[43] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

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

[45] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[46] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv Prepr. arXiv1804.02767*, 2018.

[47] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[48] C.-Y. Wang, H.-Y. M. Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh, "CSPNet: A new backbone that can enhance learning capability of CNN," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 2020, pp. 390–391.

[49] R. Kanjee, "YOLOv4—Superior, Faster & More Accurate Object Detection," *Mediu. Seach*, 2020.

[50] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, “Path aggregation network for instance segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8759–8768.