WHEAT DETECTION USING COMPUTER VISION

Project report submitted in partial fulfilment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering

By

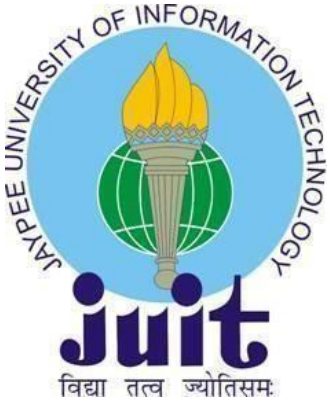
Chirayushya M Singh (181300)

Shubham Chauhan (181316)

UNDER THE SUPERVISION OF

Dr. Pradeep Kumar Gupta

to



Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology, Waknaghat, Solan- 173234, Himachal Pradesh

CERTIFICATE

Candidate's Declaration

I hereby declare that the work presented in this report entitled **"** wheat detection using computer vision**"** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from January 2022 to May 2022 under the supervision of **Dr. Pradeep Kumar Gupta,** Associate Professor Department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat.

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Chirayushya M Singh (181300)

Shubham Chauhan (181316)

This is to certify that the above statement made by the candidate is true to the best of my knowledge

Dr. Pradeep Kumar Gupta
Associate Professor
Computer Science & Engineering and Information Technology
Jaypee University of Information Technology, Waknaghat,

AKCNOWLEDGEMENT

Firstly, I express my heartiest thanks and gratefulness to almighty God for His divine blessing makes us possible to complete the project work successfully.

I really grateful and wish my profound my indebtedness to Supervisor **Dr. Pradeep Kumar Gupta,** Associate Professor, Department of CSE, Jaypee University of Information Technology, Wakhnaghat. Deep Knowledge & keen interest of my supervisor in the field of **Computational and Machine Intelligence** to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stage have made it possible to complete this project.

I would like to express my heartiest gratitude to **Dr Pradeep Kumar Gupta,** Department of CSE, for his kind help to finish my project.

I would also generously welcome each one of those individuals who have helped me straight forwardly or in a roundabout way in making this project a win. In this unique situation, I might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated my undertaking.

Finally, I must acknowledge with due respect the constant support and patients of my parents.

Chirayushya M Singh
(181300)

Shubham Chauhan
(181316)

TABLE OF CONTENTS

List of Abbreviations

NN: Neural Network

CNN: Convolutional Neural Networks

R-CNN: Region Based Convolutional Neural Networks

YOLO: You Only Look Once

ResNet: Residual Neural Network

PSPNet: Pyramid Scene Parsing Network

BB: Bounding Box

List of Figures

List of Equations

List of Tables

ABSTRACT

Object detection is a computer technique that deals with finding instances of semantic items of a specific class (such as individuals, buildings, or automobiles) in digital photos and videos. It is linked to computer vision and image processing. Face detection and pedestrian detection are two well-studied object detection areas. Object detection may be used in a variety of computer vision applications, such as picture retrieval and video surveillance.

Detecting wheat heads in plant photos is crucial for assessing critical wheat attributes including head population density and head characteristics like health, size, maturity stage, and awn presence. Several research have used machine learning techniques to build approaches for detecting wheat heads from high-resolution RGB photography. These approaches, on the other hand, have often been calibrated and verified on small datasets. Wheat head detection is difficult for computer vision because of the wide range of observing settings, genotypic variances, developmental phases, and head orientation. This task is made significantly more difficult by the possibility of blurring due to motion or wind, as well as overlap between heads in dense crowds.

The YOLO (You Only Look Once) technique was developed using a novel approach that reframed object identification as a regression issue that could be solved with a single neural network. As a result, the area of object detection has exploded, with far more amazing results than only a decade ago.

YOLO has been improved to five versions and rated as one of the best object identification algorithms by merging several of the most original concepts from the computer vision research field. The fifth generation of YOLO, dubbed YOLOv5, is the most recent version that was not created by the original creator of YOLO. However, the YOLOv5 performs better than the YOLOv4 in terms of both the accuracy and speed.

Chapter 1 INTRODUCTION

1.1 Introduction

Wheat is highly researched due to its prominence as a food and crop. Plant scientists utilise image identification of "wheat heads"—spikes atop the plant that carry grain—to obtain huge and reliable data about wheat fields throughout the world. The density and size of wheat heads in various types are estimated using these photographs. Farmers can use the information to gauge their fields' health and maturity while making management decisions.

Wheat, “chawal” and corn, is the most widely planted staple part on the crop. Then Borlaug produced semi-dwarf changes of wheat also a supplementary-agronomy methord (the Doubly-Green Revolution) in the fifties, it protected three hundred million humans from the deprivation. Later, rising at a high rate for many years, wheat fields has declined from the year 1990. Traditional breeding still relies on manual observation to a considerable extent. Increased genetic gain may be achieved by genomic choosing, novel high output phenotyping approaches, and a connection of the two. The following methods are necessary for selecting crucial wheat features related to produce probable, bacteria protection, and adaptability for abiotic stress. Creating working and effective structure to get the attributes from raw database is still a serious issue, despite the fact that increased output of phenotypic information gathering is real. Crop-head volume (this amount of the wheathead in per land region) is an important producing attribute that is even today analysed manually in the breding experiments, This is time consuming so results in measurement faults of up to Ten percent. To assist breeders in manipulating the balance in the produced components (number of plants, head-density, grains/head, weight of grain) in the breding choices, photo based technologies to boost their output and efficacy of numbering the wheathead in the field are required.

When people look at an image, they can immediately recognise things and where they are in the image. The capacity to identify items quickly mixed with a person's knowledge aids in making an appropriate judgement on the object's nature. Scientists are working on a model that can imitate the capacity of a person’s eye system to identify items. The two criteria for evaluating an object detection algorithm are speed and accuracy.

One of the most well-known challenges in computer vision is object detection. It not only classifies but also locates the object in the picture. The ways implemented to solve the

problem in prior decades is made of 2 parts: (1) extracting distinct regions of the photo with a sliding-window with various widths, and (2) applying the prediction statement to identify which group the item belong to. These methods has a drawback of requiring a lot: processing and having split into several phases. As a result, speed optimization of the system is challenging.

For the first time, in 2015, researcher [10] Joseph Redmon and colleagues unveiled the YOLO method, an item identification model that executes every of the necessary processes to recognise an item with a single NN (than known as YOLO). It transforms the obj. recognition to a one regresion issue, starting and then moving from picture pixel to BB co-ordinates and class predictions.

The singular model calculates several BB and class predictions for itmes surrounded by the multiple boxes at the same instance. The You Only Look Once model have achieved outstanding results, that outperform the leading methods in areas of both accuracy and speed for predicting and calculating target co-ordinates at the time of its introduction (Redmon, [10] 2016). In the coming five years, the You Only Look Once model was refined to 5 types (which includes the real model also), including lot of the highly innovative ideas arriving from the machine vision study community. The initial 3 iterations were investigated and evolved by Joseph Redmon, the developer of the YOLO model. He said that he will leave the machine vision industry after the development of YOLO version 3. The YOLO's updated version four, "YOLOv4", was released on the official "YOLO Github" account in the start of 2020 by Alexey Bochkovskiy, the Russian researcher who developed the previous three versions of You Only Look Once which were based on the Joseph Redmon's Darknet-architecture. Glenn Jocher with his Ultralytics LL-C research team, they created the YOLO-7 model using the Python's Pytorch-framework, published YOLO version 5 a month after YOLOv4 was launched, within a few adjustments and upgrades.

Even though, the fact that, this was not developed by the model's creators, YOLO-v5 surpassed the other three versions.

1.2 Problem Statement

Accurate wheat head recognition in outdoor field photos, on the other hand, might be visually difficult. The thick wheat plants often overlap, and the wind can cause the photos to blur. Single heads are difficult to distinguish in both cases. Maturity, colour, genetics, and

head orientation all influence looks. Finally, varied types, planting densities, patterns, and environmental conditions, planting materila must all be addressed since wheat is farmed all over the world. Wheat phenotyping models must be able to generalise across a variety of growth situations. Current detection approaches include one- and two-stage detectors (Yolo-V3 and Faster-RCNN), however even when trained with a huge dataset, there is still a bias toward the training location.

Wheat is a staple all over the world, thus this competition must take into consideration a variety of growth circumstances. Wheat phenotyping models must be able to generalise across different conditions. Researchers will be able to precisely evaluate the density and length or size of wheat heads in different types if they are successful. Farmers can better analyse their crops with enhanced detection, delivering cereal, toast, and other beloved foods to your table.

1.3 Objectives

Accurate wheat head recognition in outdoor field photos, on the other hand, might be visually difficult. The thick wheat plants often overlap, and the wind can cause the photos to blur. Single heads are difficult to distinguish in both cases. Maturity, colour, genetics, and head orientation all influence looks. Finally, varied types, planting densities, patterns, and environmental conditions, planting materila must all be addressed since wheat is farmed all over the world. Wheat phenotyping models must be able to generalise across a variety of growth situations. Current detection approaches include one- and two-stage detectors (Yolo-V3 and Faster-RCNN), however even when trained with a huge dataset, there is still a bias toward the training location.

With the problem at hand and the Methodology, Our Objective can be divided as followed:

- Our primary objective is to develop a model that is able accurately to detect wheat heads for field images.
- Improving the speed and accuracy of the model.
- Further improving the model so that it can determine features like density, the number of spike heads Or such that it can be used for different cropes.

1.4 Methodology

Object identification techniques are divided into two categories: NN-based and non-NN methords. Non-neural techniques require first defining features whith some of the algorithms listed below, followed by classification with a method like Support Vector Machine. And on the other side, NN approaches, which are often based on convolutional neural networks, are capable of doing complete detection without specifying characteristics (CNN).

- NN methods:
    - Region Pro-posals (RCNN , Fast RCNN, Faster RCNN, cascade RCNN)
    - Single-Shot MultiBox Detector (SSD)
    - You Only Look Once
    - DefineDet
    - Retina Net
    - Deformable Convolutional Network
- Non-NN methods:
    - Viola Jones detection framework on Haar features
    - Scale invariant feature transform (SIFT)
    - Histogram of oriented gradients

1.5 Organization

In this section, we'll go through the arrangement of the report's chapter-by-chapter layout. We've previously seen the Introduction and Problem Statement in Chapter 1. Following that, we looked at the report's aims and methods. We are now at the Organization portion of the report, which will clarify the chapters and subjects covered further down.

A brief description of the literature survey process will be presented in Chapter 2. This will include some of the resources we used in the past to gain necessary knowledge for this project, as well as other work we came across through the community that helped us get to the point where we were able to finish the project and lay the groundwork for future solutions to similar problems. Then we'll go on to Chapter 3, where we'll talk about how our model was conceived and evolved. First, we'll go through the dataset that we used to create this report and look at some of its fundamental aspects. We'll go through the fundamental architecture that leads to the random forest method, which includes decision trees and feature

selection. We'd compare the various models we used to the random forest and their comparative accuracies. We've gone through them briefly to help the reader comprehend them better. Following that, we've included a list of some of the mathematical formulae that were utilised in this study.

In Chapter 4, we demonstrated how our system worked and compared it to other systems such as linear regression, decision trees, and others. Then, at various phases of our project activity, we begin to display the outcomes, which comprise numerous graphs and figures. This would be followed by the model's final result.

Furthermore, in Chapter 5, we began the conclusion section by discussing what we were able to accomplish in the project and how this model was the ideal approach for us to study and explore this topic. Following that, we presented some future initiatives that may be realised through this project and added to it.

Chapter 2 LITERATURE SURVEY

2.1 General

The research are of crop detection is fairly important in the computer vision's application in agriculture. The uses of machine learning techniques are not new and one can find its application in agriculture, decades back. What really makes the technique so successful in agriculture and other fields is that it only needs a camera and a computer to run the model and find the crop in images. There are agro startups around the world that are using these techniques in not only wheat but, also in grapes, rice and other valuable crops. These methods are also not very expensive and are computationally cheap to execute. The only constraint with these techniques are trainable images, which are also not very difficult to find. This application will prove to be very important in future as global warming will damage much of the crop growth and it will be essential for farmers to use such techniques to assess health and maturity of their crop while managing the fields. But, training a model can be visually challenging as each crop can have different maturity, color, genotype and orientation. The model must also work world wide on different varieties of crop.

2.2 Image recognition using classical machine learning algorithms

If look at the earlier models used, to recognize and then assess the quality of grain, then we can look at the example of [1] Zhijun's (2007) paper where the used Neural network to identify external quality of wheat grain. Since, most of the early image classifiers were based on SVM, SVM divides data points into two or more distinct pools using a hyperplane margin, and it uses a technique called kernel method to map data into higher dimension feature space, allowing it to do classification efficiently. Then there's Random Forest, which was developed by [2] Tim Kam Ho (1995). Random Forest is an ensembling-based strategy for creating a decision tree that's less prone to overfitting on training data. The most famous of these algorithms is the neural network, which was designed by Marvin Minsky in 1969 and is based on human brain neurons. In a network, a neuron is a function that collects and categorises data using a specified design. A neural network is made up of layers with interconnected nodes, like seen above. Each node functions similarly to a perceptron, taking input from the previous layer and feeding information into the next. In most cases, an input

layer with several neurons accepts the initial input, performs some calculations, and then feeds the results to a hidden layer (s). The information is then transferred to the output layer from this hidden layer. We can also use modified neural network called convolutional neural network which is popular for winning four image competition. The name "convolutional neural network" suggests use of mathematical convolutions that are used in the model. CNN has atleast one convolution layer rather than simple matrix multiplication that most ANNa have. Every neuron has atleast one coupled neuron that creates a complex system of layers.

It works in the same way as a multi-layer perceptron neural network (MLP). To categorise the photos, It functions similarly as a multilayer perceptron neural network. The flattened matrix is sent through a FC layer to categorise the photographs. Advanced neural networks, such as AlexNet, RCNN, and YOLOs, are built on CNN and perform well in image recognition applications.

2.3 Crop detection using SVM and Random Forest.

Early machine learning techniques includes SVM and Random Forest, before the invention of neural network these models were often used and gave good result considering the complexity of the problem. One such example is [3] Z Tong's (2006) paper where they used SVM to detect grain pest and this proved to be better than ANNs under the condition of limited training samples. Another such example is Ibrahim et. al.'s [4, 2019] paper were they used multi class SVM to detect rice grains for faster sorting of grains, they were able to achieve 92.22% accuracy in their dataset. They used three attributes of color descriptor which are saturation, hue and value, four attributes such as area size, length of perimeter, minor axis length, and major axis length and . H Zhang[5, 2009] used the single fitness function was built to assess the feature subset using SVM and another optimizer for stored-grain insects by incorporating the v_fold cross validation training model accuracy and the number of chosen features. Nine species of stored-crop insects decayed badly in crop-depots, including Tenebroidees mauritanicus(L.) and Rhizoperthae dominica Fabricius. The approach founded on PSO and SVM was used to select feature subsets for the stored-grain insects. H Kaur's[6,2013] paper they used Multi class SVM for classification and grading SVM, after the rice kernels had been separated from background, the Maximum Variance approach was used to extract the chalk from the rice. Ten geometric parameters were used

to determine the amount of broken rice, head rice,and Brewers in rice samples. The Chalkiness, Shape and Percentage of Broken (Broken, Head Rice and Brewers) kernels were used to classifying the rice by Multi-Class SVM. More than 86 percent of SVM classifications are correct. According to the findings, the method was adequate for classification and grading the various varieties of rice grains based on their exterior and internal quality. Random forest has been used to predict soil surface texture in a semiarid region. Other staple crops can also be graded and classified by the SVMs and Random Forest. We can also use Random forest for detecting pest in grains. Overall, it can be said that SVMs and random forest are useful in precision agriculture application. Also, many other algorithms such as logistic regression and artificial neural network can be used for this task.

2.4 Crop detection using RCNNs

To face the difficulty of picking a great number of areas, Ross Girshick et al.[7,2014] proposed a technique called region recommendations, in which we try to use selective search to extract just 2000 regions from a picture. As a consequence, instead of attempting to identify a vast number of locations, you may now focus on just 2000. The CNN works as a feature extractor, with the extracted features being used as input data into an SVM to categorise the object's existence inside the probable region suggestion. In addition to predicting the presence of an object within the specified zone, the method predicts four offset values to boost the precision of the BB. For instance, the algorithm might have predicted the presence of an object in an area recommendation, but that object’s part inside that region proposal may have been halved. As a result, the offset values aid in altering the region proposal's bounding box. RCNN still had some problems, You'd have to categorise two thousand region proposals every image to train the network, which would take a long time. Because every test image takes around 50 seconds, it can't be done in actual time. The chosen search algorithm is a fixed algorithm. As a result, there is no learning process at that moment.As a result, an instance of poor probable region ideas might be generated. Then

there's the same author's fast rcnn, which is similar to the R-CNN method. We feed the CNN the input picture instead of the region recommendations to build a convolutional feature map.

We take the convolutional feature map's area of recommendations, distort them into quadrilaterals, and then rebuild them into a fixed size regions using a “RoI” pooling layer so they can be used as inputs into a fully connected layer. To anticipate the class of the proposed region as well as the BB offset values from the RoI feature vector, we use a softmax layer. "Fast R-CNN" is quicker than R-CNN since you don't have to input the convolutional neural network 2000 area suggestions every time. Instead, the convolution procedure, which is performed just once per picture, produces a feature map. When comparing the performance of Fast R-CNN during testing, using region proposals considerably slows down the algorithm compared to not utilising region proposals.As a result, region proposals become bottlenecks in the Fast R-CNN algorithm, slowing it down. To find region proposals, both of the above algorithms use selective search. Selective search is a slow and time-consuming operation that degrades network performance. As a result, Shaoqing Ren et al. created an object identification algorithm that does away with the selective search algorithm and allows the network to learn region proposals. The image is fed into a convolutional network, which outputs a convolutional feature map, similar to Fast R-CNN. Instead of using a selective search strategy on the feature map to identify the region suggestions, a separate network is used to anticipate the region proposals. After that, a RoI pooling layer is used to categorise the image inside the suggested region and predict the bounding box offset values, and the projected region proposals are reshaped.The research by Y Shen et al.[8,2018] employed a quicker rcnn to identify stored grain insects.

A method for detecting and recognising stored-grain insects was built using RCNN. Cryptoleste Pusillius(S.), Sitophiluse Oryzae(L.), Oryzaephiluse Surinamensis(L.), Tribolium Confusume(Jaquelin Du Val), Rhizoperthae Dominicas (F.).These live insects' Red, Green, and Blue (RGB) images were assembled into a database.To extract regions in these pictures that potentially contain insects and categorise the insects in those spots, a faster R-CNN was utilised. An upgraded inception network was constructed to extract feature maps.Excellent results were obtained in the detection and categorization of these insects. Mean Average Precision (mAP) of 88.0, the developed method was shown to be capable of detecting and identifying insects in stored grain.

2.5 Crop detection using YOLOs

In contrast to other region proposal classification networks, which perform detection task on large number of region proposals and performs predictions in multiple times for various regions in an image, Yolo architecture is more similar to FCNN and passes the image (nxn) once through the Fully connected NN and outputs (mxm) prediction.The input picture is split into mxm grids, each of which is given two bounding boxes and class probabilities. YOLO has gone through several incarnations, the most recent of which is YOLO9000: Better, Faster, Stronger (i.e., YOLOv2), which can identify over 9,000 object detectors. By undertaking combined training for both object detection and classification, Redmon and Farhadi are able to obtain such a vast number of object detections. The authors used combined training to train YOLO9000 on both the ImageNet classification dataset and the COCO detection dataset at the same time. The outcome is the YOLO9000 model, which can forecast detections for object classes with no labelled detection data. YOLOv2's performance was interesting and original, however it fell short of the title and abstract of the article. YOLO9000 achieved 16 percent mean Average Precision (mAP) on the 156-class version of COCO, and while YOLO can recognise 9,000 distinct classes, the precision isn't nearly what we'd like. B Gong et. al.[9,2019] paper used Yolo model on GWHD dataset for detecting wheatheads.

Chapter 3 SYSTEM DEVELOPMENT

3.1 Analysis/Design/Development/Algorithm

3.1.1 RCNN

R-CNNs (Region-basedConvolutional NN) are a example of machine-learning model used in computer vision and image processing. The basic purpose of any R-CNN, which is specifically built for item detection, is to detect objects in any input pictures and define boundaries around them.

The RCNN model uses a method known as “selective search” to extract information about the region of interest from an input picture.

- The rectangular boundaries can be used to illustrate the region of interest.
- There could be over 2000 regions of interest depending on the scenario.
- This region of interest is fed into CNN, which generates output features.
- The objects presented in a region of interest are then classified using these output attributes by an SVM (support vector machine) classifier.

Object localisation can be accomplished in a variety of ways in any object detection technique.
One strategy, which we call an exhaustive search approach, is to use sliding filters of various sizes on the image to extract the object from image.
In an exhaustive search technique, calculation work increases as the number of filters or windows increases.

The selective search algorithm employs exhaustive search, but in addition to that, it works with the segmentation of the image's colours. In more formal terms, selective search is a technique for separating items from an image by assigning them distinct colours.

This algorithm begins by creating a large number of small windows or filters, then grows the region using the greedy algorithm. Then it looks for colours that are similar in other places and combines them together. The similarity between the regions can be estimated using the following formula:

$S(a,b) = S_{texture}(a,b) + S_{(a,b)}$ Where $S_{texture}$(a,b) is visual similarity and S_{size}(a,b) is region-to-region similarity. The model continues to use this approach to merge all of the regions together in order to increase the size of the regions.

A selective search algorithm is depicted in the image. Following the region selection, the picture containing regions is passed through a CNN, which extracts the objects from the region.
Because the image size must be regulated according to CNN's capability, reshaping the image will take some time, if not all, of the time. We wrap the region in 227 × 227 x 3 images in basic R-CNN.

.1 Using a CNN, extract objects : The object of size 4096 dimensions will be extracted using a wrapped input for CNN.
.2 Classification: The basic R-CNN uses an SVM classifier to categories things into their respective classes.

R-entire CNN's process architecture can be expressed as: The boundary box repressor operates at the conclusion of the model to define items in the image by covering it with a rectangle.

3.1.2 Fast RCNN

Instead of conducting maximum pooling, we use ROI pooling in fast R-CNN to use a single feature map for all regions. This warps ROIs into a single layer, and the ROI pooling layer converts the features using max pooling. Because max pooling also works here, we can think of fast R-CNN as an update to the PSPNet. It simply creates one layer, rather than multiple layers in a pyramid shape.

Using linear regression and softmax, create a fully connected network for categorization. is shown in the image above. With linear regression, the bounding box is fine-tuned even more. R-CNN is quicker than PSPNet.

3.1.3 Faster RCNN

PSPNet and Fast R-CNN did not have any methods for choosing regions of interest until now, as we saw in the article for region proposals. The fundamental difference in the Fast RCNN and the Faster RCNN is the following. To build the sets of regions, faster R-CNN uses a region proposal method. The regional proposal network, which we name the faster R-CNN, has an extra CNN for acquiring the regional proposal. The proposal network in the training region accepts the feature map as input and produces region proposals. These recommendations are then forwarded to the Region of Interest pooling layer for further processing.

3.1.4 YOLO

With a research published in 2015 by Joseph Redmon [10], YOLO entered the computer vision landscape. YOLO: Unified, Real Time Object-Detection drew a high interest from other computer-vision experts right away. Prior to the invention of YOLO, Convolutional NeuralNetworks (C-NN) like as Region-Convolutional Network uses Regions-Proposal Network (RPN) to produce proposal bounding boxes on imputed data, after that run a prediction on the BB, and then apply post processing to delete duplicate predictions and improve the BB. Single levels of the RCNN network's could not be trained independently. It was tough and time-consuming to optimise the R-CNN network.

The author's goal is to use a neural network to create a unified representation of all stages. After running the input picture through a NN comprising many convolutional NN, the model provides prediction vector's to every object in these pictures, whether it contains (or does not contain) the items. Rather of iterating process of categorising individual regions on picture, YOLO system computes all of the image's characteristics at once and generates predictions for all items. "You Only Look Once" is based on this concept. [10] (Redmon and colleagues, 2016)

3.1.5 YOLO Basic Concepts

YOLOv1's core concept is, insert a gridcell with a shape of SxS (7x7 default) onto a picture. If an item's centre falls inside a gridcell, then the gridcell is in charge of predicting the object (Fig. 1). As a result, all the other cells ignore the apearance of an item that has been shown in numerous cells.

Every gridcell forecasts B B.B. with there parameters and the confidence ratings in order to execute object detection (Figure 1). (V Thatte,[11] 2020). The existence or absence of the object in the enclosing box is reflected by this confidence score. The confidence points is calculated as follows:

$confidence$ points = $p(Obj.) * IOU^{truth}_{pred}$

IOU pred truth is IOU of calcification box and base truth box, where P(obj.) this is possibility that there is an obj. within the cell, and P(obj.) is the possibility that there is an obj. within the cell. The confidence score is closer to 0 if no obj. is present within this cell since P(obj.) is in the range 0-1. Instead, the score will be equal to IOUpred truth.

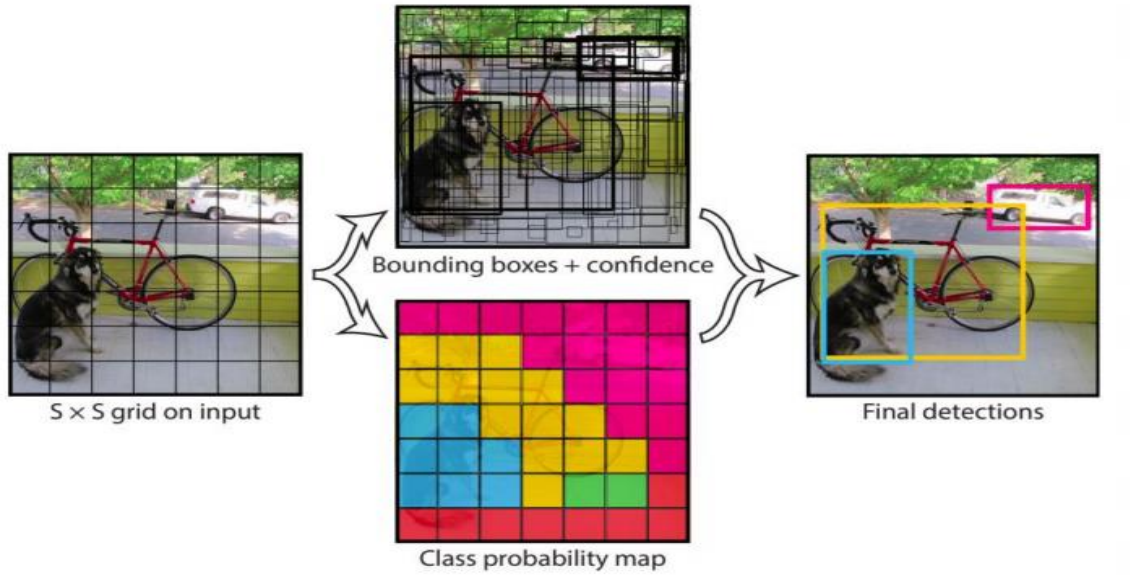


Figure 1: You Only Look Once algo. with 7x7 gridcell was applied on input photo's.[10]

Furthermore, every BB has 4 additional attributes (x, y, w, k) that correspond to the bounding box's (centre coordinate(x, y), width, and height) (Figure 2). Each bounding box has 5 parameters when combined with the confidence score.

The YOLO algorithm's goal is to predict an item by correctly calcification its BB and then localising it using the bounding box coordinates. As a result, anticipated BB vectors corespond to the resulting vector y and base truth B.B. vectors-to-vector label y. . Vector labeled y and calcification vector y may be shown in Fig.4, there the cell coloured purple has no item and the bounding box confidence score in cell coloured purple is equal to zero, therefore every remaining attributes are ignored.

3.1.6 YOLO v1 architecture

The YOLO model includes an architecture that analyses all picture characteristics (dubbed Darknet architecture by the developers) and two fully linked layers that execute bounding box estimation for objects (Figure 2). The authors utilised S = 7.0, B = 2.0, also C = 20.0 in the Pascal VOC database to test this model. This clarifies why the output size was (7 * 7 * (2 * 5 + 20)) and the final feature maps were 7*7.

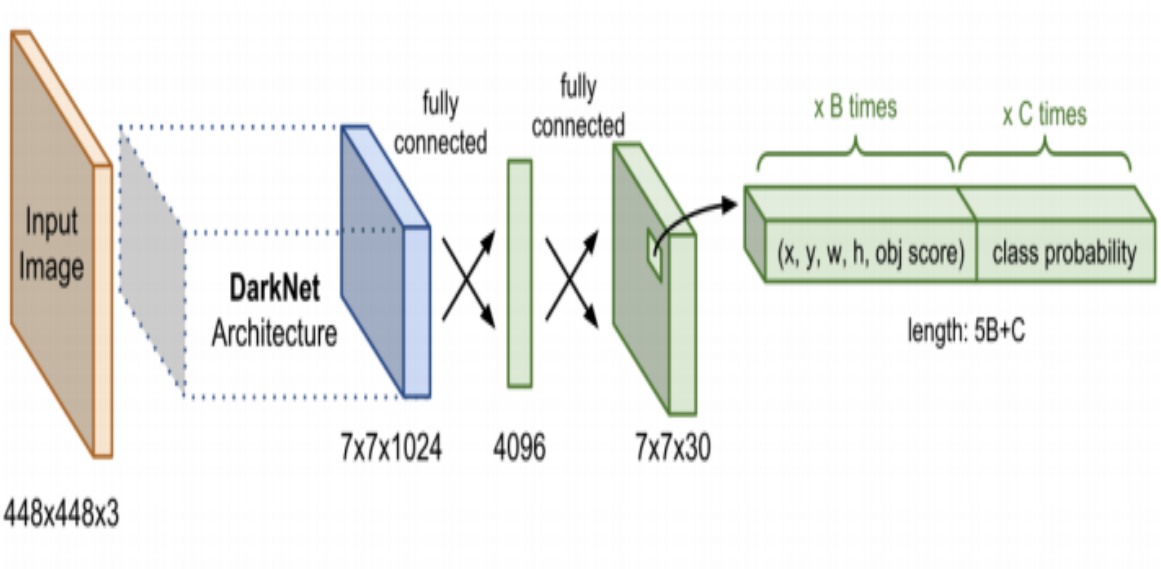


Figure 2: Preliminary YOLOv1 architecture

For uncomplicated datasets, the authors presented the fast YOLO structure with nine C-NN layers in the Darknet-architecture, while the normal YOLO structure with twenty four C-NN layers in the Darknet design can handle higher complex data and produce greater accuracy (Figure 3). The GoogLeNet (Inception) model, which can help to minimise the features space from preceding layers, inspired the sequences of 1x1 and 3x3 convolutional layers (Menegaz,[11] 2018). Rather than using Leaky Rectified Linear Unit (leaky ReLU) activation, the final layer utilises a Linear activation function:

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

Name	Filters	Output Dimension
Conv 1	7 x 7 x 64, stride=2	224 x 224 x 64
Max Pool 1	2 x 2, stride=2	112 x 112 x 64
Conv 2	3 x 3 x 192	112 x 112 x 192
Max Pool 2	2 x 2, stride=2	56 x 56 x 192
Conv 3	1 x 1 x 128	56 x 56 x 128
Conv 4	3 x 3 x 256	56 x 56 x 256
Conv 5	1 x 1 x 256	56 x 56 x 256
Conv 6	1 x 1 x 512	56 x 56 x 512
Max Pool 3	2 x 2, stride=2	28 x 28 x 512
Conv 7	1 x 1 x 256	28 x 28 x 256
Conv 8	3 x 3 x 512	28 x 28 x 512
Conv 9	1 x 1 x 256	28 x 28 x 256
Conv 10	3 x 3 x 512	28 x 28 x 512
Conv 11	1 x 1 x 256	28 x 28 x 256
Conv 12	3 x 3 x 512	28 x 28 x 512
Conv 13	1 x 1 x 256	28 x 28 x 256
Conv 14	3 x 3 x 512	28 x 28 x 512
Conv 15	1 x 1 x 512	28 x 28 x 512
Conv 16	3 x 3 x 1024	28 x 28 x 1024
Max Pool 4	2 x 2, stride=2	14 x 14 x 1024
Conv 17	1 x 1 x 512	14 x 14 x 512
Conv 18	3 x 3 x 1024	14 x 14 x 1024
Conv 19	1 x 1 x 512	14 x 14 x 512
Conv 20	3 x 3 x 1024	14 x 14 x 1024
Conv 21	3 x 3 x 1024	14 x 14 x 1024
Conv 22	3 x 3 x 1024, stride=2	7 x 7 x 1024
Conv 23	3 x 3 x 1024	7 x 7 x 1024
Conv 24	3 x 3 x 1024	7 x 7 x 1024
FC 1	-	4096
FC 2	-	7 x 7 x 30 (1470)

Figure 3: Normal-YOLOv1 NN model having 24 C-NN layers and two fullyconnected layers

3.1.7 YOLOv2

In the deep learning model, batch normalisation is one of the most used ways of normalising. It provides for quicker and more stable deep neural network training by stabilising the input

layer distribution during training. The purpose of this method is to normalise the features (each layer's outputs after activating) to a 0 mean condition with a SD of one.

$$Mini\ batch\ mean: \mu = \frac{1}{m}\sum_{i=1}^{m} z^{(i)}$$

Equation 1: Mini-batch mean

$$Mini - batch\ variance: \sigma^2 = \frac{1}{m}\sum_{i=1}^{m}\left(z^{(i)} - \mu^2\right)$$

Equation 2: Mini-batch variance

$$Normalize: z_{norm}^{(i)} = \frac{z^{(i)} - \mu}{\sqrt{\sigma^2 + \varepsilon}}$$

Equation 3: Normalize

$$Scale\ and\ shift: \tilde{z}^{(i)} = \gamma z_{norm}^{(i)} + \beta$$

Equation 4: Scale and shift

After batch normalisation, all of YOLO veraion2's layers are applied. This method not only cuts down on training time, but it also improves network's generalisation. Batch normalising improved m-AP (mean precision) by roughly 2.0% in YOLOv2 (Redmon, [10] 2016). To avoid overfitting, the network does not need to utilise any more Dropouts.

The very initial twenty convo-layers (in the YOLO version 1 model) were utilised to training the feature extractor (prediction networks) using 224x224 input picture in the original YOLO (YOLOv1). The remaining four convo-layers and two fully-linked layers were then included, and this input image's resolution was concurrently raised to 448 448 to be utilised as a detection algorithm. (Kamal, [13] 2019)

YOLOv2, on the other hand, after finishing the feature extractor's training phase with the 224x224 input picture, the model continued the training of the extracted features for another 15 epochs including the 448x448 input pictures already employing the framework for the object-detector training. Whenever the features extraction train phase transitions to the object-detector train phase, the model may "adapt" to a higher resolution of 448x448 in place of instantly improving the picture pixels. This high-resolution classification network results in a nearly 4% increase in mAP.

3.1.8 Convolutional with anchor box

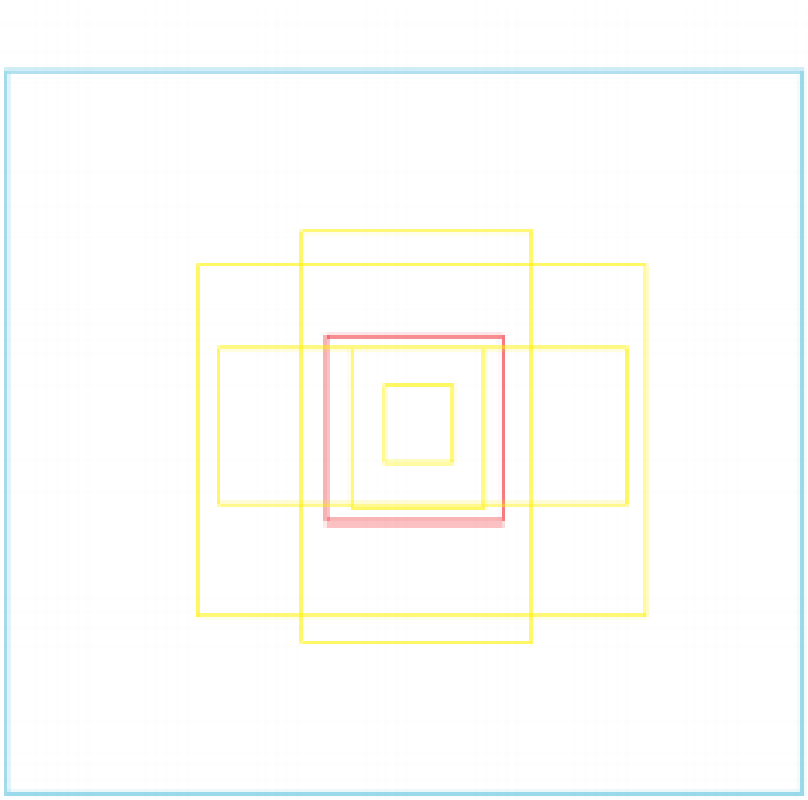


Figure 4: model prediction bounding boxes

The concept behind YOLOv1 is to utilise a grid cell to identify an item that has its centre within that grid cell. As a result, if two or more objects have their centres in the same grid cell, the forecast may be incorrect. The author attempted to tackle this challenge by allowing a grid cell to anticipate many objects. Instead of employing completely linked layers like in YOLOv1, the author used an anchor box architecture to anticipate bounding boxes in YOLOv2 ([10] Redmon, 2016). A collection of preconfigured boxes that's the good fit the

intended items is called an anchor box. Not only were ground truth boxes used to anticipate the bounding boxes, but also preset k anchor boxes.

Rather than manually choosing the bestfit anchor-boxes, the training dataset BB (including every base truth box) were clustered using the k-means clustering algorithm, and the mean IOU was plotted with the nearest centroid (Fig.4). Rather then using Euclidean distance, the developer used IOU within the BB and the centroid. Having different values for k, k = 5.0 is a great compromise between recall and model complexity. The developer evaluated in the VO-C and CO-CO datasets, and the right figure depicts the tradeoff in recall and model-complexity as a function of the number of clusters (k).The right photo in both datasets displays 5 centroids (which may be used as anchor boxes).(Redmon and colleagues, 2016; Redmon and colleagues, 2016)

When it comes to anticipating the boundary position of a box, YOLOv1 has no limitations. The bounding box may be predicted anywhere in the picture when attributes are randomly initialised. In the early stages of training, this renders the model unstable. The BB can be located far away from the gridcell that is important for anticipating that BB.

In YOLO, every gridscell is described on a scale of 0 to 1, with the top-left point (0, 0) and the lower right point (1, 1) As a result, YOLO-v2 employed the sigmoid function, f() to limit the bounding box centre value to the range 0-1, allowing it to establish the bounding box estimates all around grid cell.

3.1.9 YOLOv3

For Darknet architecture, YOLO v2 featured a proprietary deep 30-convolutional layer architecture, which was larger than YOLOv1's 11 layers. More layers in deep neural networks equals more accuracy. When forwarding to further layers, however, the input picture was downsampled, resulting in the loss of fine-grained characteristics. That's why YOLOv2 had a hard time detecting little objects. ResNet proposed the use of skip connections to aid activation propagation over deeper layers without vanishing the gradient (Figure 4). (He, [14] 2015).

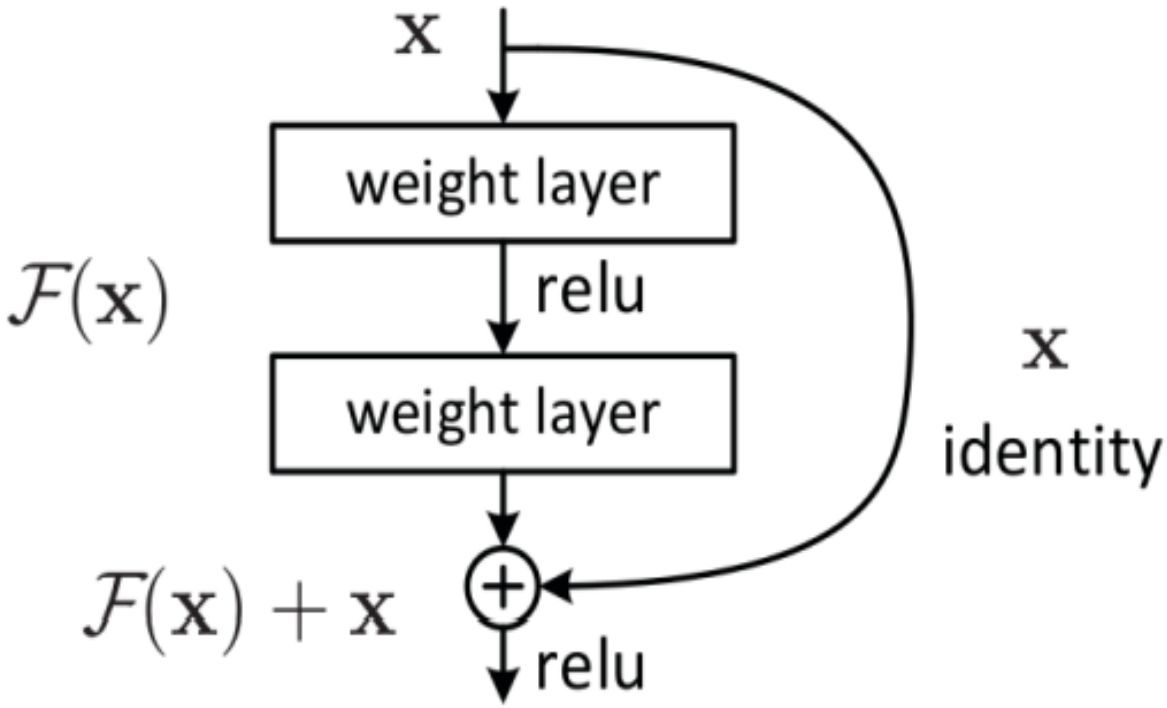


Figure 5: ResNet skip connection architecture

Redmon, et al. [10] proposed a superior architecture in which the feature extractor was a combination of YOLO-v2, Darknet53, and the Residual NN. Inside each residual block, the network is created using a bottleneck structure (1x1 followed by 3x3 convolution layers) and a skip connection (Figure 6).

Overlapping layers will not impact network performance thanks to ResNet's residual blocks. Furthermore, because the deeper layers receive more information directly from of the upper layers, the bulk of fine-grained characteristics is not lost.

	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
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 6: Darknet-53 (5 residual blocks)

The model made use of the Darknet-53 architecture, which was designed with a 53-layer network for feature extraction training. The detecting head for training object detector was then layered with 53 additional layers, giving YOLO version 3 a complete of 106 layers of fully-convolutional under lying model.

3.1.10 Multiscale detector

After training in the feature extractor using Darknet architecture in two earlier versions of YOLO, the input was sent to a few additional layers before being used to make calcification in the final levels of the item calcifies. More over, instead of stacking the calcification layers at the final layers as before, YOLOv3 added them to the side network (Figure 7). YOLOv3's most significant feature is that it detects at three distinct scales (Redmon, et al.,[10] 2018). Three distinct scale detectors were created using the characteristics from the last three residual blocks.

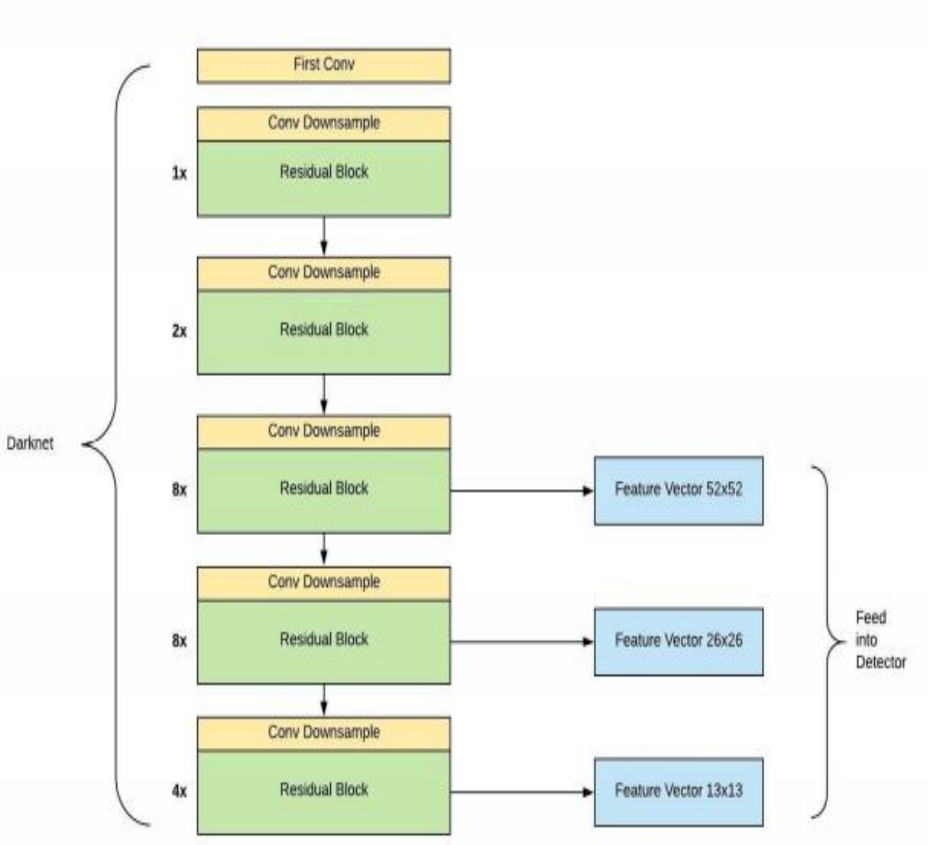


Figure 7: Multi-scale detector

Specifically, YOLOv3 predicts at three scales in layers 82, 94, and 106, that are precisely determined by the network's stride of 32nd, 16th, and 8th, respectively.

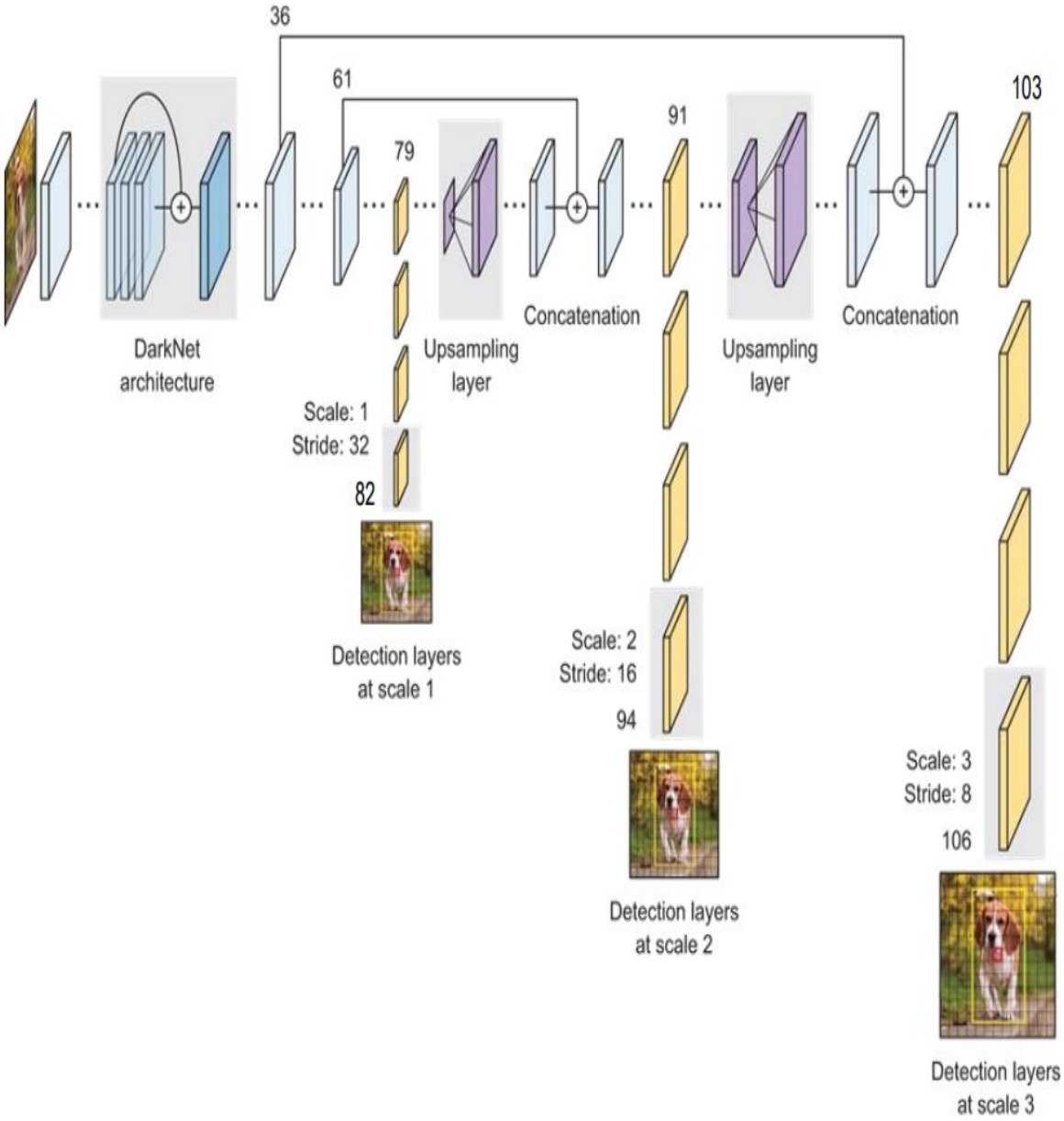


Figure 8: YOLOv3 network architecture)

Unlike YOLO verssion1, where bounding boxes forecast by the same gridcell portion a combination of C class calcification probabilities, every gridcell is responsible for calcifying one obj., YOLOv2 introduces the concept of a gridcell having the ability to forcast the many objects at once. B.B. will predict different type of objects even if they are calculated by the same gridcell. As a result, rather of sharing a set of C class prediction probabilities, predicted

bounding boxes have their own (Figure 14). For each separate detector, the total YOLOv3 output parameters will be S S (B (5 + C)).

The 82nd layer is the first to notice something. The author chose the 416 416 input picture as the default for simplicity of understanding. After passing from the starting 81 layers, these input picture is down sampled by 32, resulting in a feature map with a size of 13x13 matching to 13x13 gridcells (Fig.9). Detection is carried out at each detection layer by using 1x1 detection kernels on feature maps. The 1x1 kernel is in charge of estimating the B bounding box for each feature map grid cell. The COCO dataset was used to train YOLO version 3 with B = 3.0 (three BB for every cell) and C = 80.0. (classes),

As a result, the kernel size is 1 1 (3 (5 + 80) = 1 1 255. The final generated feature map at the first detection layer will be 13x13x255. (Fig.9).

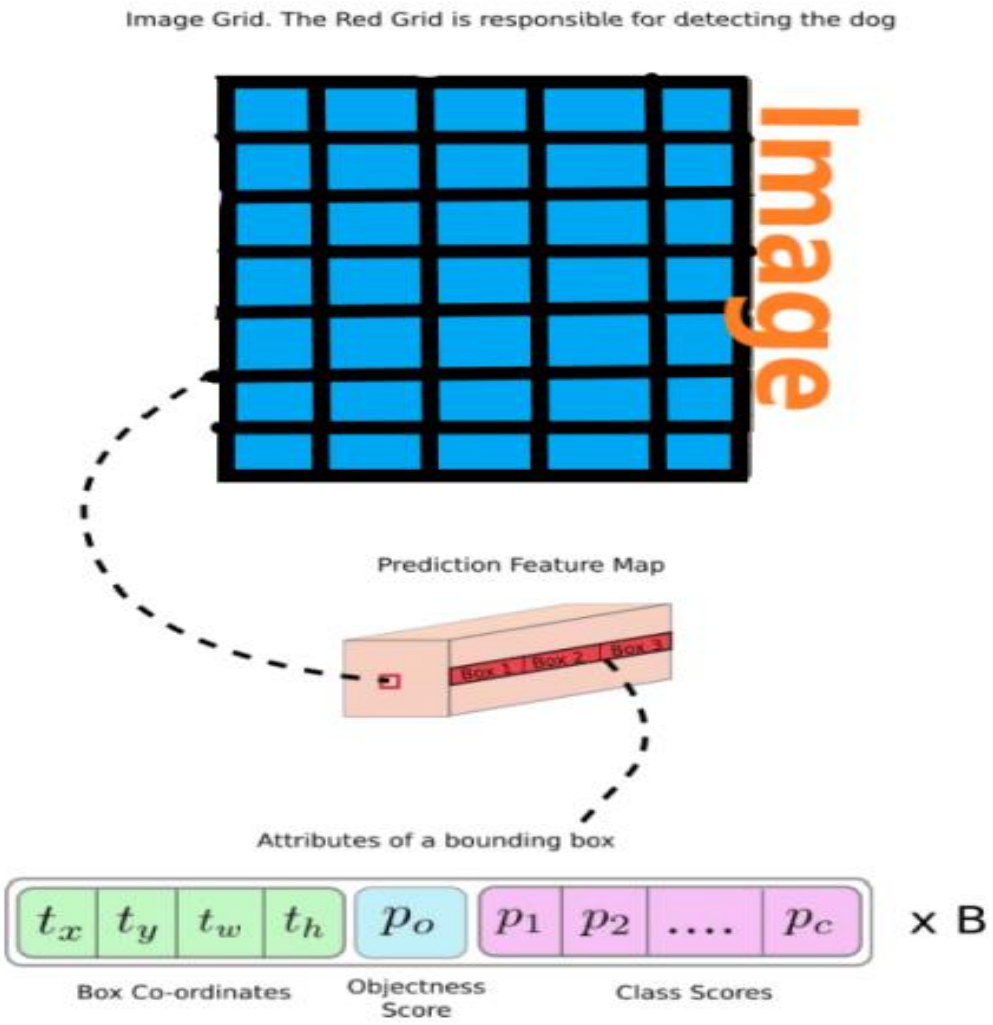


Figure 9: YOLO-v3 detecting an obj. by applying an 1x1 kernel

The same technique is followed again. Even though, the featuremaps at layer 79 and 91 are up-sampled before being forwarded to two more detection layers for prediction. Following down - sampling by the strides of 16 and 8, the feature maps have sizes of 26x26 and 52x52, respectively, equivalent to the 94th and 106th detection layers (Figure 9).

Furthermore, detections at various size layers aid in addressing the problem of recognising small objects, which is a common criticism with YOLOv2. The detailed feature map is the one with the largest size. Small items are detected by the large-scale detection layer (52x52), whereas bigger objects are detected by the small-scale detection layer (13x13).

The fine-grained characteristics from previous layers can be preserved by concatenating with the upper layers after up-sampling in the deeper layer (combines with layer 61 before reaching layer 91 and concatenate with layer 36 before reaching layer 103rd as shown in Fig.9). This helps the largescale detection layer detect small objects.

3.1.11 YOLOv4

Joseph Redmon, the creator of the YOLO algorithm, is also the creator of the Darknet custom model. After 5 years of development and research on the 3rd version of the YOLO algorithm (YOLO version 3), J. Redmon said his departure from the field of machine vision and stopped researching the YOLO algorithm due to concerns that his technology will be misused in military applications. He does not, however, object to any individual or group continuing to do research based on the YOLO algorithm's early concepts.

Alexey Bochkovsky,a soviet scientist and engineer who developed the Darknet architecture and three earlier You Only Look Once architectures in C based on Joseph Redmon's theoretical theories, collaborated with C. Yao and H. Yuan to release YOLO version 4 in April 2020.

Along with the development of YOLO, several object identification systems using various methodologies have also made significant progress. Since then, two design object detection ideas have emerged: onestage sensor and twostage detectors (Fig.10).

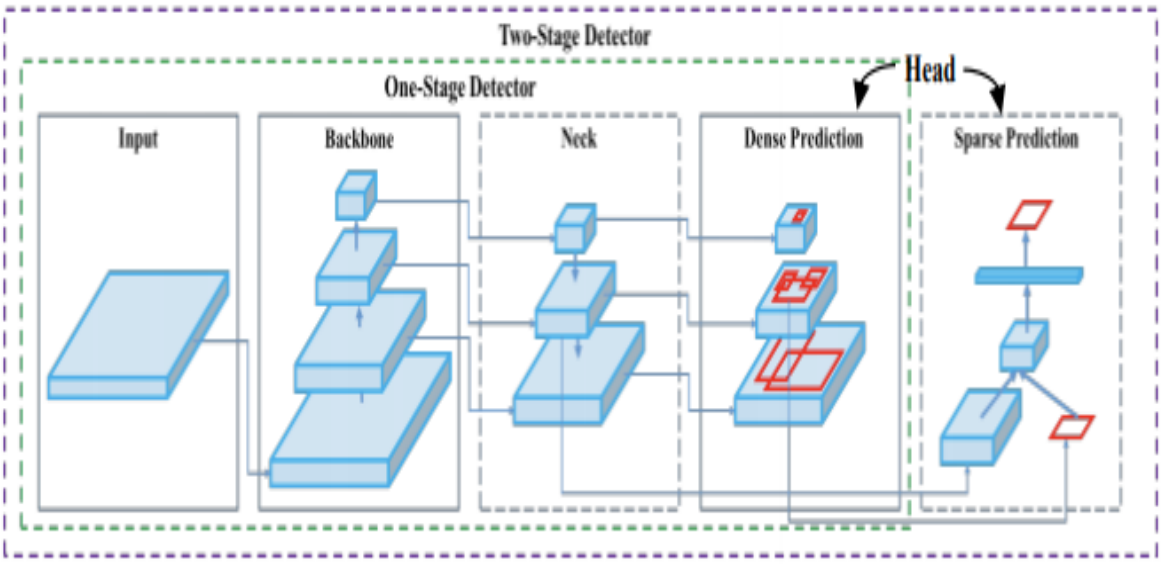


Figure 10: 2 concepts of architectural object detection

The input picture characteristics are compressed down by feature extractor (Backbone) and then forwarded to item detector (containing Detection Neck and Detection Head) in all object prediction architectures, as shown in Figure 10. The Detection Neck (or Neck) is a feature aggregator charged with mixing and combining the features created in the Backbone in order to prepare for the detection process in the Detection Head (or Head).

In all object detection designs, the input picture features are compressed down by the feature extractor (Backbone) and then transmitted to the object detector, as illustrated in Figure 10. The Detection Neck (or Neck) is a feature aggregator responsible for mixing and merging the features developed in the Backbone in order to prepare the Detection Head for the detection process (or Head).

For each aspect of the architecture, the YOLO author conducted a series of trials with several of the most sophisticated computer vision innovation concepts (Figure 11).

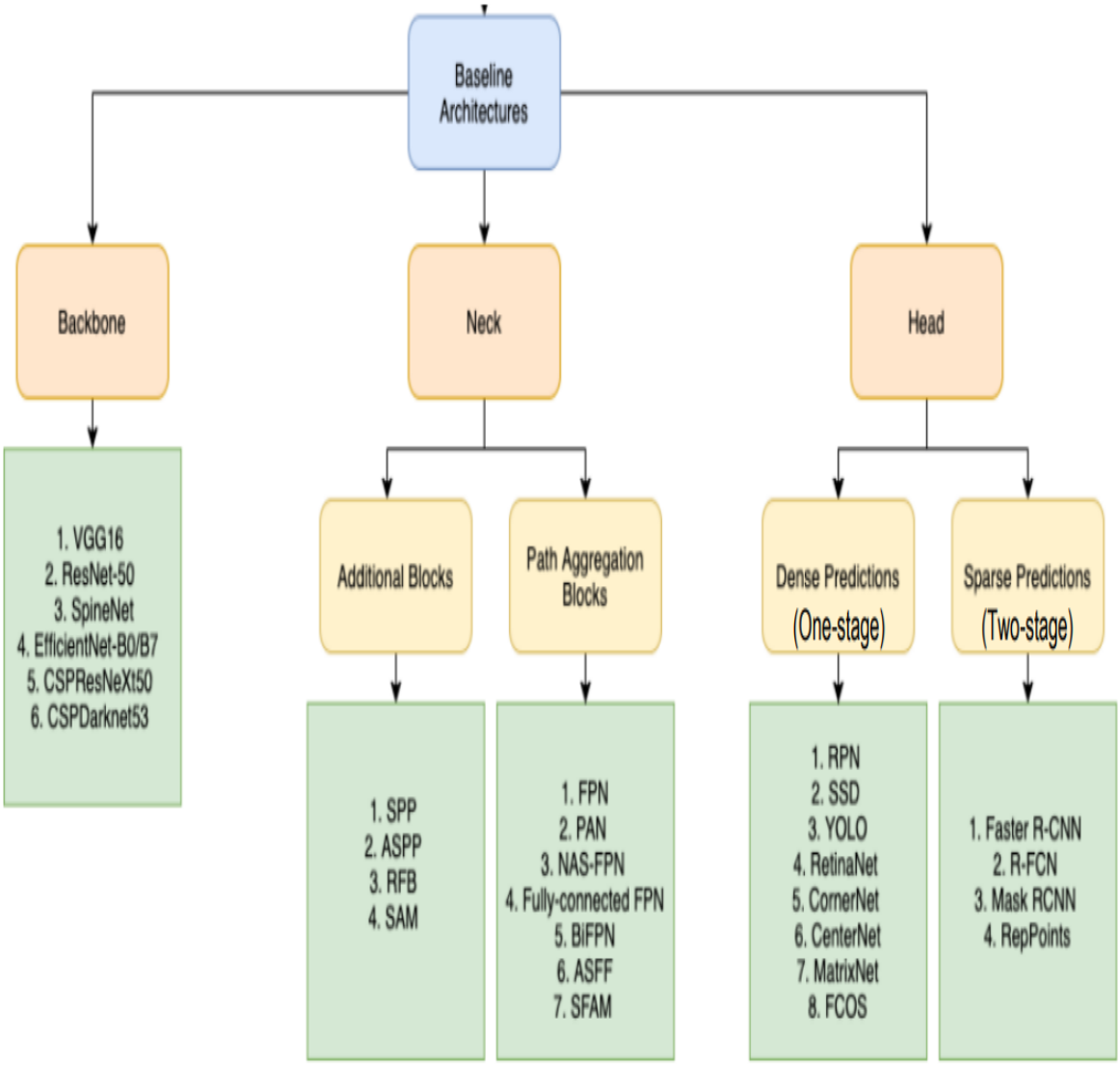


Figure 11: YOLOv4 different architecture

3.1.12 EfficientNet

EfficientNet was released in 2019 in a research paper by Google. This new CNN family provide better accuracy and also improve the efficiency of the model. It does this by reducing Floating Point Operations Per Second and parameters.

- Creating a small basic architecture: EfficientNet-B0

- Provides an efficient compound scaling strategy for expanding model size and maximising precision improvements.

EfficientNet-B0 Architecture

Table 3.1: Architecture Details for the baseline network

Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

Conventional CNN designs, such as Mobile Net and ResNet, can benefit from the compound scaling strategy. Nevertheless, because the compound scaling approach only improves the predictive capability of the systems by reproducing the base network's fundamental convolutional processes and network structure, selecting a suitable basic network is important for attaining the best outcomes.

Towards this purpose, the authors apply Neural Design Search to create EfficientNet-B0, an efficient network infrastructure. With only 5.3M parameters and 0.39B FLOPS, it achieves 77.3 percent accuracy on ImageNet. (With 26M parameters and 4.1B FLOPS, Resnet-50 delivers 76 percent accuracy.)

Squeeze-and-excitation optimization is added to the fundamental building component of this network, which would be MBConv. MBConv is identical to MobileNet v2's inverted residual blocks. These provide as a quick link between the start and finish of a convolutional block. To improve the depth of the feature maps, the input initiation maps are first extended using 1x1 convolutions. Following that, 3x3 Depth-wise with Point-wise convolutions are used to

minimise the number of channels in the final feature image. The thin layers are connected by shortcut connections, whereas the broader layers are present between the skip connections. This structure aids in reducing the overall number of operations as well as the length of the framework.

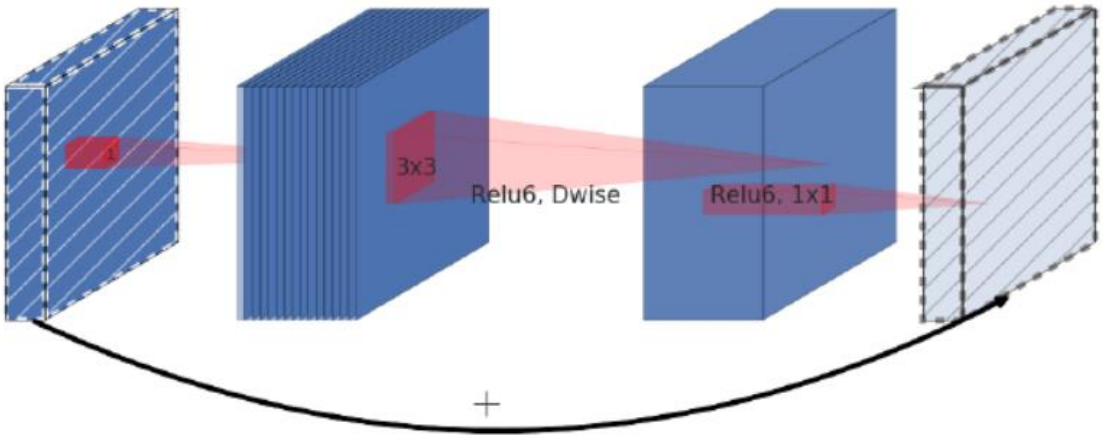


Figure 12: Inverted residual block

3.1.13 Compound Scaling

The depth, breadth, and resolution of a convolutional neural network may be scaled in three dimensions. The number of layers in a network relates to the network's level. The breadth of a convolutional layer is proportional to the number of neurons inside the layer or, more specifically, the quantity of filters in the layer. The length and breadth of the supplied picture are used to determine the resolution. Fig.13 above depicts scaling in these three dimensions more clearly.

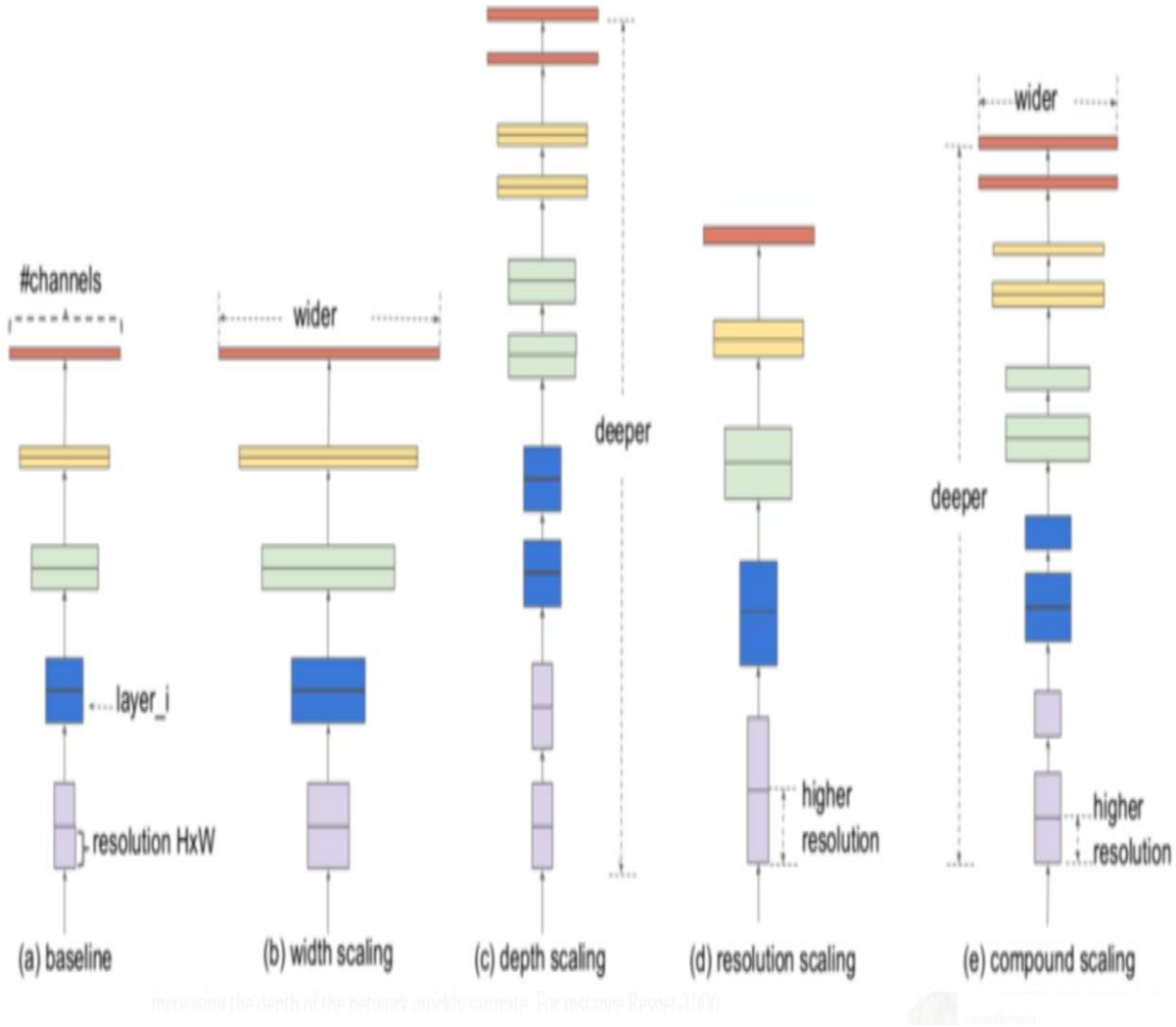


Figure 13: Model Scaling

(a) is a baseline-network example; (b)-(d) are conventional scaling that only increases one dimension of network width, depth, or resolution. (e) is our proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio.

The neural network learns more complicated characteristics by increasing the depth by stacking additional convolutional layers. Deeper networks, on the other hand, are prone to disappearing gradients, making training more challenging. Although new approaches like batch normalisation and skip connections can help solve this challenge, empirical studies show that just expanding the network's depth would fast saturate the reliability improvements. Despite the additional layers, Resnet-1000 achieves the same rate of precision as Resnet-100.

Layers can learn more fine-grained information by increasing the network width. This notion has been applied in a variety of works, including Wide ResNet and Mobile Net. Increasing breadth, like increasing depth, stops the network from learning complicated characteristics, resulting in a decrease in efficiency gains.

Increased input resolution gives the model more information about the picture, allowing it to explain about tiny objects and identify finer connections. However, like with the other scaling dimensions, this alone only delivers modest precision benefits.

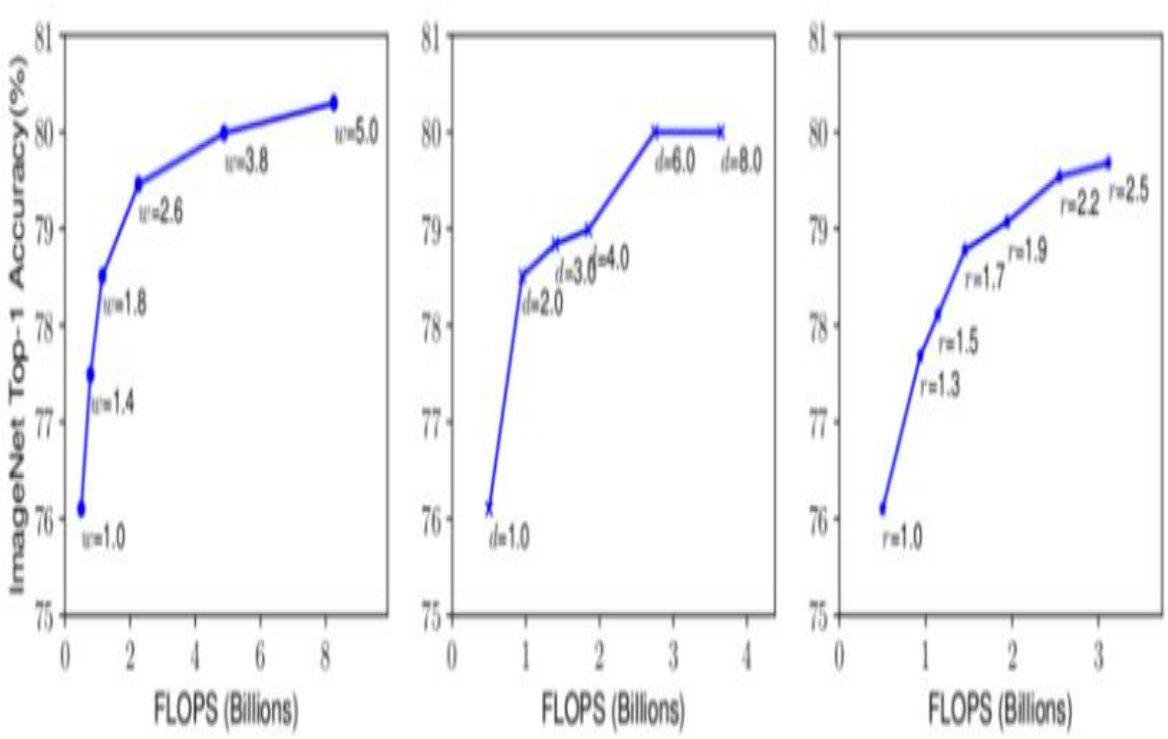


Figure 14: Scaling Up a Baseline Model

3.1.14 StreamLit

Streamlit is an open-source Python library that is used to make interactive machine learning and data science application that can run in a web browser without much front-end development required. Streamlit works with a simple python script that can customize the look of the application and add plots, charts, sliders, buttons and inputs. Every time the

application is runs, the whole python scripts is re-read. Streamlit supports magic commands that will basically read any variable on its own line and automatically parse it using st.write() method. st.write() can work with almost anything such as text, dataframe, plots, etc. and render it correctly. However, to customize the plots and other data types, you need to use data-specific functions like st.dataframe() and st.table(). Streamlit supports widgets and the user can add them using st.slider(), st.button() or st.selectbox().

Even when importing data from the web, handling enormous datasets, or conducting expensive computations, the Streamlit cache allows your app to run swiftly. When a function is marked with st.cache, the streamlit will automatically check input parameters called, value of any external variable, body of function, function used inside cache. Streamlit employs a particular hash function for both the key and the output hash that knows how to traverse code, hash special objects, and may have its behaviour adjusted by the user. One can also deploy a streamlit application on cloud and manage it using Streamlit Cloud

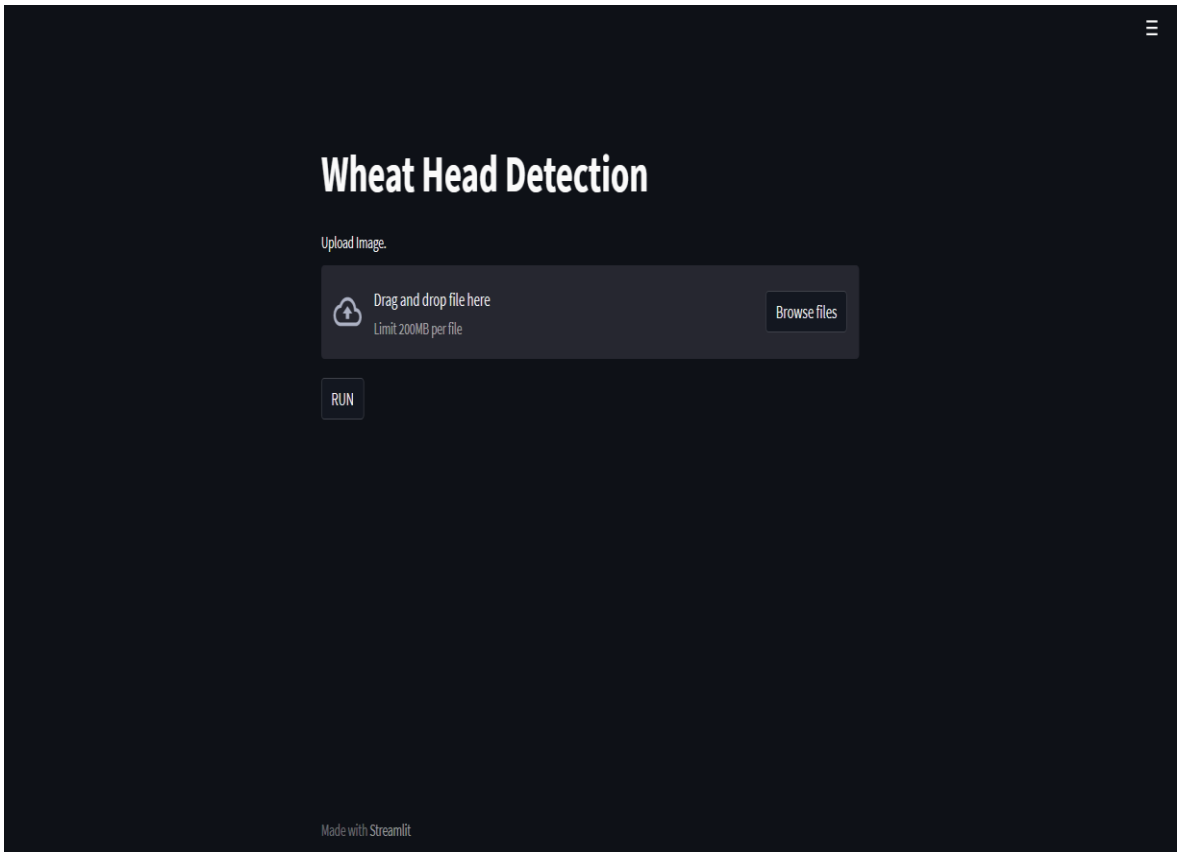


Figure 15: StreamLit 1

For our application we have first called YOLOX.tools.demo, this import tools necessary for the application to run smoothly. st.title() is used to give a name to our application. st.file_uploader() is then called to import the image that will be processed by our algorithm. When clicking the RUN button image is passed to the algorithm, where inference is performed and we get the result. st.progress() can be used to add a progress bar that can show the time left.

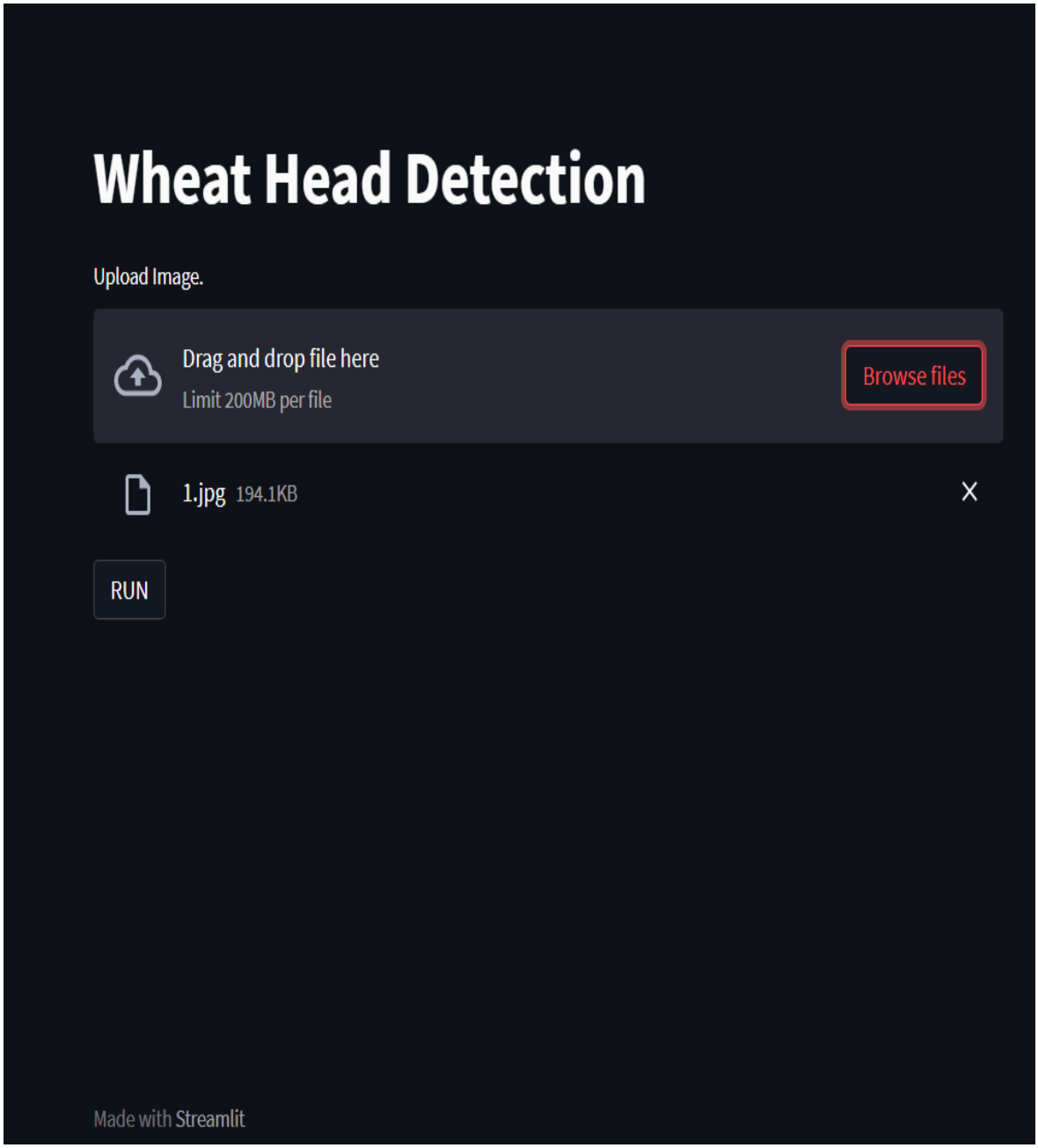


Figure 16: StreamLit 2

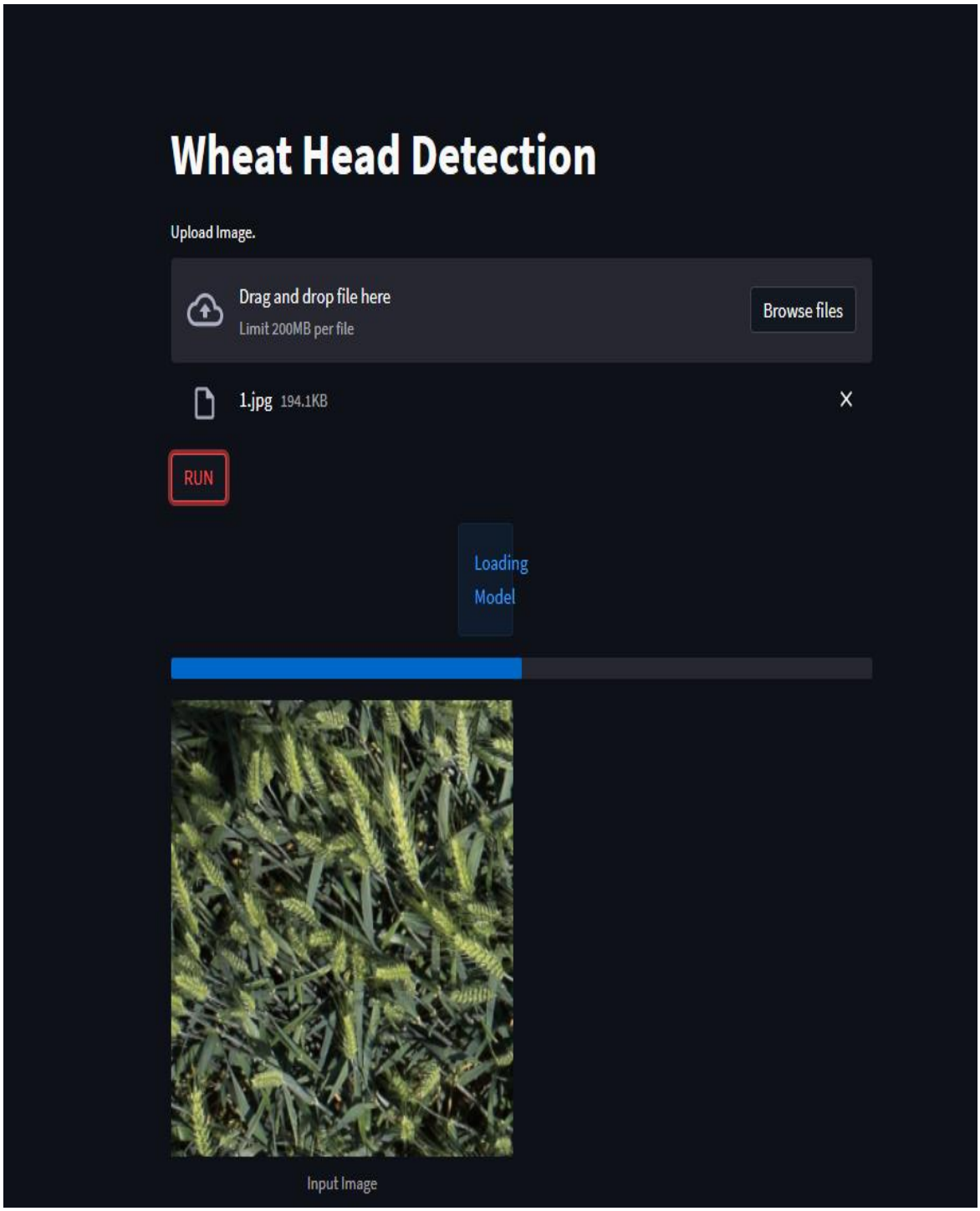


Figure 17: Processing on StreamLit

Figure 18: StreamLit Output

3.2 Model Development

A month following the introduction of YOLOv4, Glenn and his team produced YOLOv5, a new iteration of the YOLO family (Jocher, 2020). Glenn Jocher is the CEO of Ultralystics LLC and a scholar. YOLO models were created using Alexey Bochkovsky's own framework Darknet, which is primarily written in C. Ultralystic is a firm that transforms prior versions of YOLO to PyTorch, a well-known deep learning system developed in the Python programming language.

3.2.1 YOLOv5

Glenn Jocher is also the creator of the Mosaic data augmentation, which was recognised in the YOLOv4 publication by Alexey Bochkovsky. Due to its nomenclature and advances, his YOLOv5 model attracted a lot of controversy in the computer vision area. Despite the fact that it was introduced a few months after YOLO version 4, the start of YOLO4 and YOLO5 research was quite near (March to April, 2020). To avoid confusion, Glenn named his version of YOLO, YOLO5. Both researchers essentially employed the most cutting-edge breakthroughs in the area of object identification available at the time. Having a output, the architectures of YOLO4 and YOLO5 are quite similar, and many people are dissatisfied with the appellation YOLO5 (5th generation of You Only Look Once) because it does not offer many substantial improvements over YOLO4. Glenn also failed to publish any YOLOv5-related papers, which raised even more doubts.

However, YOLOv5 seemed to have the advantage of engineering. Unlike previous versions, YOLO version 5 is available in Python not in C. This is the reason for its easier to implementation and integrate to Internet of Things devices. Lastly, the PyTorch community isa very bigger than that of the Darknet community, meaning that PyTorch will get higher rate of contributions and will have higher posibility for future expansion. Because YOLO version 4 and YOLO version 5 are implemented in two different languages and operate on two different frameworks, it's difficult to compare their performance. Still, in addition to YOLO version 4, YOLO version 5 has gained some trust in the computer vision community as a result of its greater performance over YOLOv4 in some conditions.

3.2.2 Adaptive anchor boxes

As previously indicated, the YOLO5 architecture incorporates the most current advancements in a way similar to the YOLOv4 architecture, therefore there are few significant differences in concept. In place of publishing a full article, the author created a Github’s repository and updated it on the platform. By deconstructing the structural program in file.yaml, the YOLO5 framework may be described as follows :

- Head: YOLOv3 head using GIoU-loss

- Neck: SPP block, PANet
- Backbone: Focus structure, CSP network

The YOLOv5 author brings forth an interesting issue about an design difference. In YOLOv2, Redmon presented the anchor box design as well as a method for picking anchor boxes that are similar in area and form to the ground truth B.B. in the training set. The authors chose the five best-fit anchor boxes for theCOCO (which has 80 classes) and used them as the default using the k-mean clustering approach with various k values. This minimises the amount of time it takes to train the network and improves its accuracy.

Furthermore, when these five anchor boxes are applied to unique dataset (including a class that is not one of the eighty classes in the COCO dataset), they are unable to swiftly adapt to the unique dataset's ground truth bounding boxes. A giraffe database, for example, likes anchor boxes that are narrow and taller than square boxes. Computer vision engineers commonly start by running the kmean clustering method on the unique anatomical to find the best fit box for the given dataset. These settings will then be configured in the framework.

Glenn Jocher, suggested that the anchor box selection procedure be included into YOLOv5. As a consequence, the network does not need to take any of the databases as input; instead, this would "learn" and apply the optimal anchor-boxes for each database during train time.

3.3 Analytical

The GWHD collection is made up of tagged photos gathered between 2016 and 2019 by 9 institutions in 10 distinct places (Table1), and it includes genotypes from Europe, North America, and Australia as well as Asia. Individual datasets are referred to as "sub-datasets." They were obtained through a series of tests diverse growth methods, with row spacing ranging from 12.5 cm to 25 cm (ETHZ_1) to 30.5 cm (USask_1). Table 1 summarises the features of the experiments. Low sowing density (UQ 1, UTokyo 1, UTokyo 2), average seeding density (Arvalis 1, Arvalis 2, Arvalis 3, INRAE 1, NAU 1), and large seeding density (RRes 1, ETHZ 1, NAU 1) are among them. The GWHD dataset includes a variety of pedoclimatic conditions, including highly productive environments such as the loamy soil of Picardy, France (Arvalis 3), silt-clay soil in mountainous environments such as the Swiss Plateau (ETHZ 1), and the Alpes de Haute Provence (Arvalis 1, Arvalis 2). The trials in

Arvalis 1, Arvalis 2, UQ 1, NAU 1 were meant to contrast irrigated and water-stressed ecosystems.

Table 3.2: Attributes of the experiments used to acqire photos for dataset

Sub-dataset name	Institution	Country	Lat (°)	Long (°)	Year	Nb. of dates	Targeted stages	Row spacing (com)	Sowing density (seed m-2)	Nb. of genotypes
UTokyo_1	NARO & UTokoyo	Japan	36.0N	140.0 E	2018	3	Post-flowering	15	186	66
UTokyo_2	NARO & UTokoyo	Japan	42.8N	143.0	2016	6	Flowering*	17.5	300	20
Arvalis_1	Arvalis	France	43.7N	5.8E	2017	3	Post-flowering-Ripening	17.5	300	20
Arvalis_2	Arvalis	France	43.7N	5.8E	2019	1	Post-flowering	17.5	300	20
Arvalis_3	Arvalis	France	49.7N	3.0E	2019	3	Post-dlowering-Ripening	17.5	300	4
INRAE_1	INRAE	France	43.5N	1.5E	2019	1	Post-flowering	16	300	7
USask_1	University of Saskatechewan	Canada	52.1N	106. W	2019	1	n.a	30.5	250	16
RRes_1	Rothamsted Research	UK	51.8N	0.36 W	2016	1	n.a	n.a	350	6
ETHZ_1	ETHZ	Switzerland	47.4N	8.6E	2018	1	n.a	12.5	400	354
NAU_1	Nanjing Agric. University	China	31.6N	119.4 E	2018	1	Flowering*	20	300 or 450	5
UQ_1	UQueensland	Australia	27.5S	152.3 E	2016	1	Flowering-Ripening	22	150	8

* Images were checked carefully to ensure that heads have fully developed and flowered.

3.3.1 Image acquisition

RGB photos recorded using a variety of groundbased phenotyping systems and cameras are included in the GWHD collection (Table 2). The picture capture height varies between 1.8 and 3 metres high. With a variety of sensor sizes, the camera focal length ranges from 10 to 50 mm. Because of the variances in camera configuration, the Ground Sampling Distance (GSD) varies from 0.10 to 0.62 mm, and the half field of view along the picture diagonal

varies from 10 to 46 degrees. The obtained GSDs are high enough to distinguish heads and even awns visually, assuming wheat heads have a diameter of 1.5 cm. Despite the fact that all of the photos were taken in the nadir-viewing direction, due to the varied lens properties of the cameras employed, some geometric distortion may be seen for a few sub-datasets. This problem is especially severe in the datasets UTokyo 1 and ETHZ 1. Images were collected on a variety of platforms, including handheld, cart, mini-vehicle, and gantry systems, by each organization. The large range of picture attributes produced by the variety of camera sensors and acquisition settings will aid in training deep learning models to generalise better across varied image acquisition situations.

Table 3.3: Image Characteristics of the sub-datasets comprising the GWHD dataset

Sub-dataset name	Vector	Camera	Focal Length (mm)	Field of view (°)*	Shooting mode	Image size (pixels)	Distance to Ground (m)	GSD (mm/px)
UTokyo_1	Cart	Canon PowerShot G9 X Mark II	10	38.15	automatic	5472x3648	1.8	0.43
UTokyo_2	handheld	Olympus u850 & Sony DSC-HX90V	7/4	45/5	automatic	3264x2488 & 4608x3456	1.7	0.6
Arvalis_1	handheld	Sony Alpha ILCE-6000	50 & 60	7.1	automatic	6000x4000	2.9	0.10-0.16
Arvalis_2	handheld	Sony RX0	7.7	9.99	automatic	800x800**	1.8	0.56
Arvalis_3	handheld	Sony RX0	7.7	9.99	automatic	800x800**	1.8	0.56
INRAE_1	handheld	Sony RX0	7.7	9.99	automatic	800x800**	1.8	0.56
USask_1	Mini-vehicle	FLIR Chameleon3 USB3	16	19.8	Fixed	2448x2048	2	0.45
RRes_1	gantry	Prosilica GT 3300 Allied Vision	50	12.8	automatic	3296x2472	3-3.5***	0.33-0.385
ETHZ_1	gantry	Canon EOS 5D Mark II	35	32.2	Fixed	5616x3744	3	0.55
NAU_1	handheld	Sony RX0	24	16.9	automatic	4800x3200	2	0.21
UQ_1	handheld	Canon 550D	55	17.3	automatic	5184x3456	2	0.2

[Note: All cameras looked vertically downward]

* The field of the view is measured diagonally. The reported is the half-angle.

** Original images were cropped, and a sub-image of the size 800x800 war extracted from the central area

*** The camera was positioned perpendicular to the ground and automatically adjusted to ensure a 2.2m distance was maintained between the camera and canopy.

3.3.2 Data Harmonization

Harmonizing the numerous sub-datasets was a key element in assembling the GWHD dataset Images were initially manually inspected to verify that they could be properly comprehended. When heads were not clearly visible, images taken at an early stage of development were eliminated . Because heads tend to overlap as stems begin to bend at this time, the majority of the photographs were taken before the advent of head senescence.

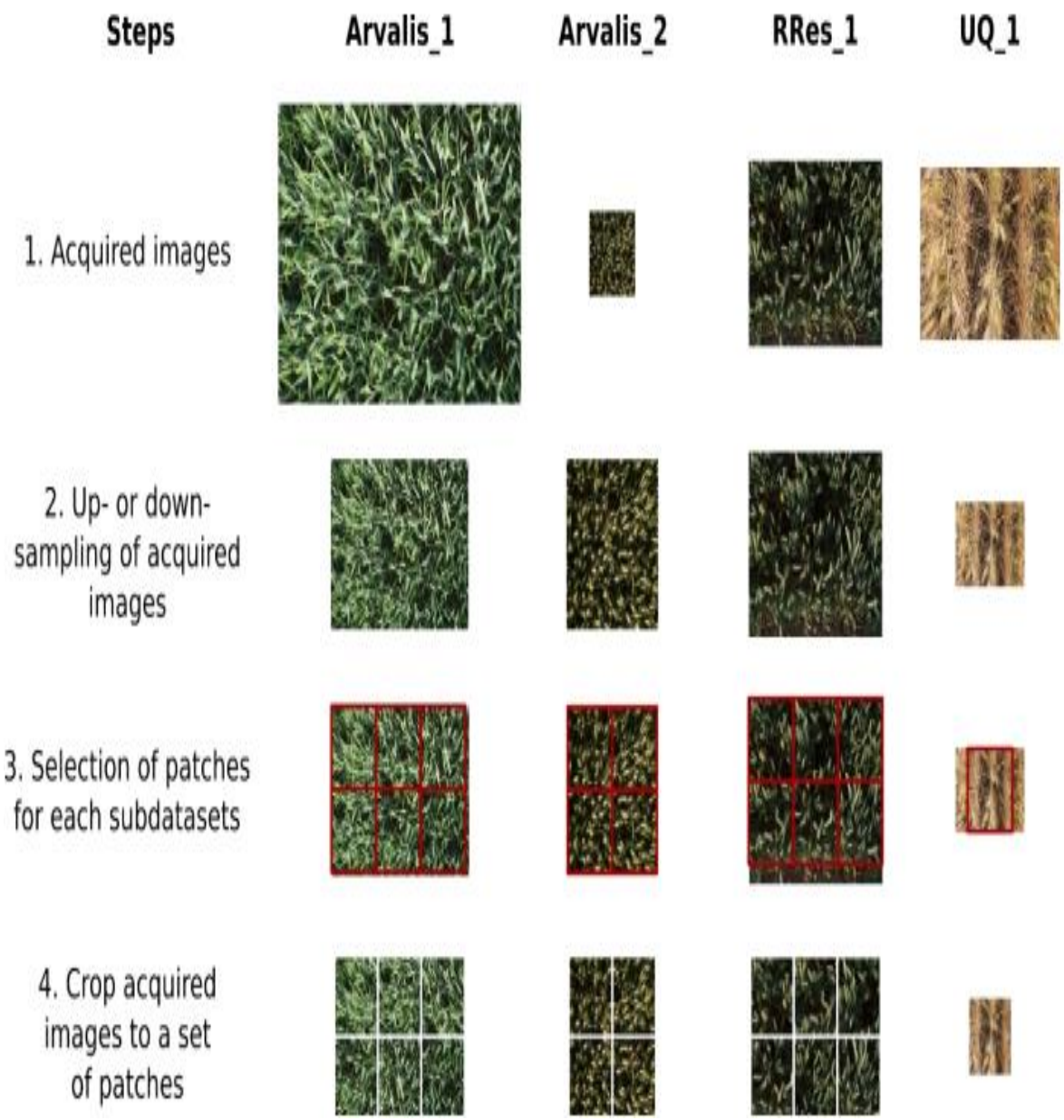


Figure 19: Review of harmonization process conducted.

Item scale, or the size of an object in pixels, is a critical consideration in the development of object detection algorithms. Object scale is determined by the object's size (in mm) and the

image's resolution. Wheat heads vary in size according on genotype and growing circumstances, but they are typically 1.5 cm in diameter and 10 cm long. The real picture resolution at the level of wheat heads changed dramatically between sub-datasets: the GSD changes by a factor of five, while the real resolution at the head level is further influenced by canopy height and the camera's panoramic effect. When photographs are taken too close to the canopy, the panoramic effect will be significantly greater. As a result, images were rescaled to maintain a more consistent resolution at the head level. The original photos were up- or down-sampled using bilinear interpolation. The factor that was used to scale each sub-dataset is displayed in Table 2.

3.4 Computational

We utilised the machine with the following specifications for this project work at the time of training. CPU: The specifications of the PC we utilised were as follows:

Table 3.4: CPU Specifications

Parameter	Specifications
CPU Model name	Intel(R)Core(TM)
CPU frequency	2.3 Ghz
No of CPU cores	8
Available Ram	15.56 GB
Disk Space	800 GB

These are the specifications of the computer we used to perform the calculations on. The GPU handles the majority of the computing effort. However, the CPU does the majority of the preprocessing. The huge quantity of RAM did not put a lot of strain on the system, making it simpler to load the entire dataset in a timely manner. We also didn't have to worry about system problems. The 2.3 Ghz clock speed of the CPU is the base clock speed, which may be increased to 5 Ghz if necessary. However, no overclocking was required because the machine could complete the task with its standard four cores.

Table 3.5: GPU Specifications

Parameter	**Specifications**
GPU	**NVIDIA-GeForce-GTX 1060**
GPU-Memory	**16 GB**
GPU, Memory Clock	**1.40 Ghz**
GPU Release Year	**2016**
Cores	**4**
Available RAM	**16 GB**
Disk Space	**800 GB**

Tools Used: We used the following tools in making our model.

- Python
- Matplotlib
- Seaborn
- Jupyter Notebook
- Pandas
- Plotly
- Numpy
- Scikit Learn

These packages mentioned above were used in their latest upto date editions. The code works properly and would not cause any issue until any further updates in them.

3.5 Experimental

Now we tried a few algorithms just to check them in comparison to what we used.

i. RCNN
ii. Fast RCNN
iii. Faster RCNN
iv. YOLO-v5

We've also included some basic descriptions of the aforementioned algorithms to make it easier to understand and investigate why they didn't perform as expected.

i. R-CNN

   To overcome the difficulty of selecting a high range of areas, Ross Girshick et al. proposed a method called region recommendations, in which we use selective search to identify just 2000 areas from an image. As a consequence, instead of attempting to identify a vast number of places, you can now focus on only 2000. These 2000 region concepts were compiled using a selective search strategy.

   Selective-Search:

   1. Produce starting sub-sagmentation, then create many candidate areas

2. Using greedy approach to iterative combine same areas to bigger areas
3. Using the created areas to calculate the last candidate area proposals

ii. Fast RCNN

The Fast R-CNN detector, like the RCNN predictor, gives region recommendations implementing an model similar to Edge-Box. Fast RCNN detector analyses the full image, unlike this RCNN predictor, which cuts and resizes region recommendations. Fast RCNN pools C-NN characteristics in related to every area pro-posal, whereas an RCNN predictor must categorise every area. Fast RCNN is greater efficient than R-CNN since the computation for intersecting areas are distributed in this Fast RCNN predictor.

iii. Faster RCNN

In place of employing an outside method same as Edge-Box, In Faster RCNN detecetor includes a area providing network (known as RPN) to create area suggestions straight to the model. For Object Detection, the RPN employs Anchor Boxes. The network-based generation of region recommendations is faster and more tailored to your data.

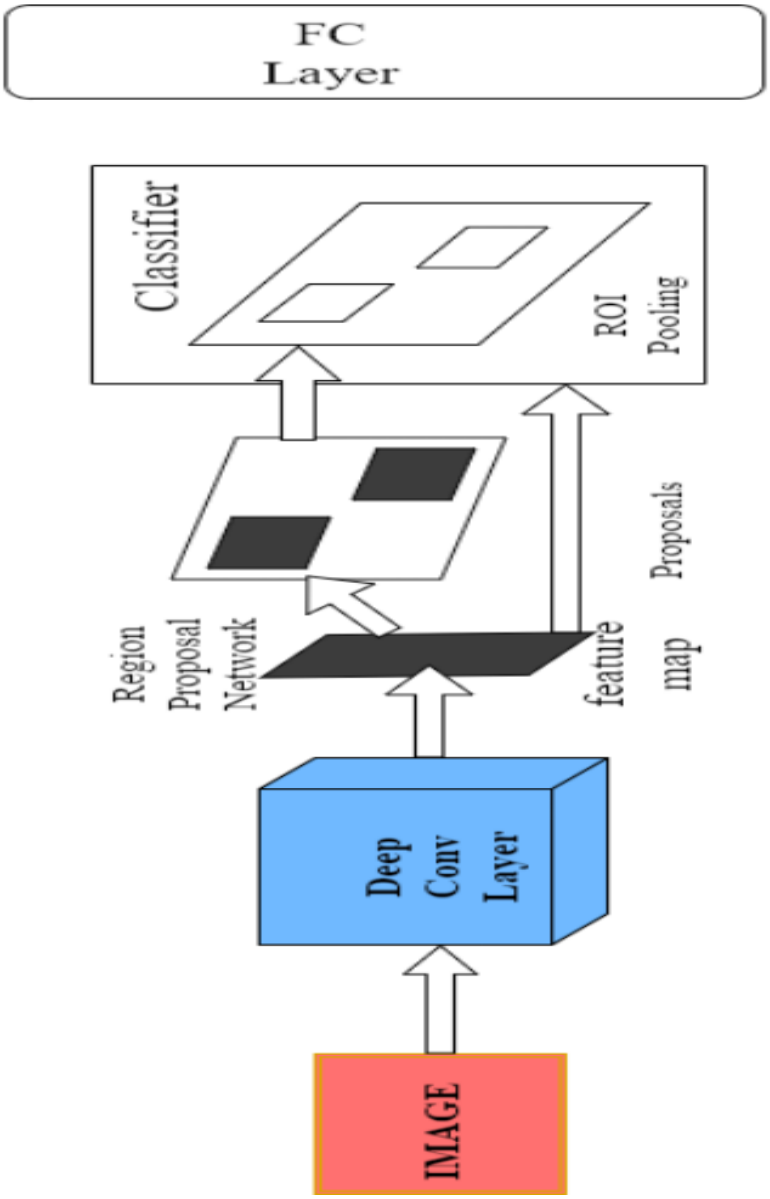


Figure 20: Faster R-CNN

iv. YOLOv5

These innovations were initially known as YOLOv4, however owing to the latest update of YOLO version 4 in Darknet structure, it was changed to YOLOv5 to prevent releases clashes. We produced an article differentiating YOLO version 4 with YOLO version 5, In which you are able to run both YOLO's together on your

dataset, because there was a lot of disagreement about the YOLO version 5 nomenclature at first. In this essay, we'll avoid bespoke dataset comparisons and instead focus on the resent tech. and matrix's that the You Only Look Once scientists are sharing in GitHub communications.

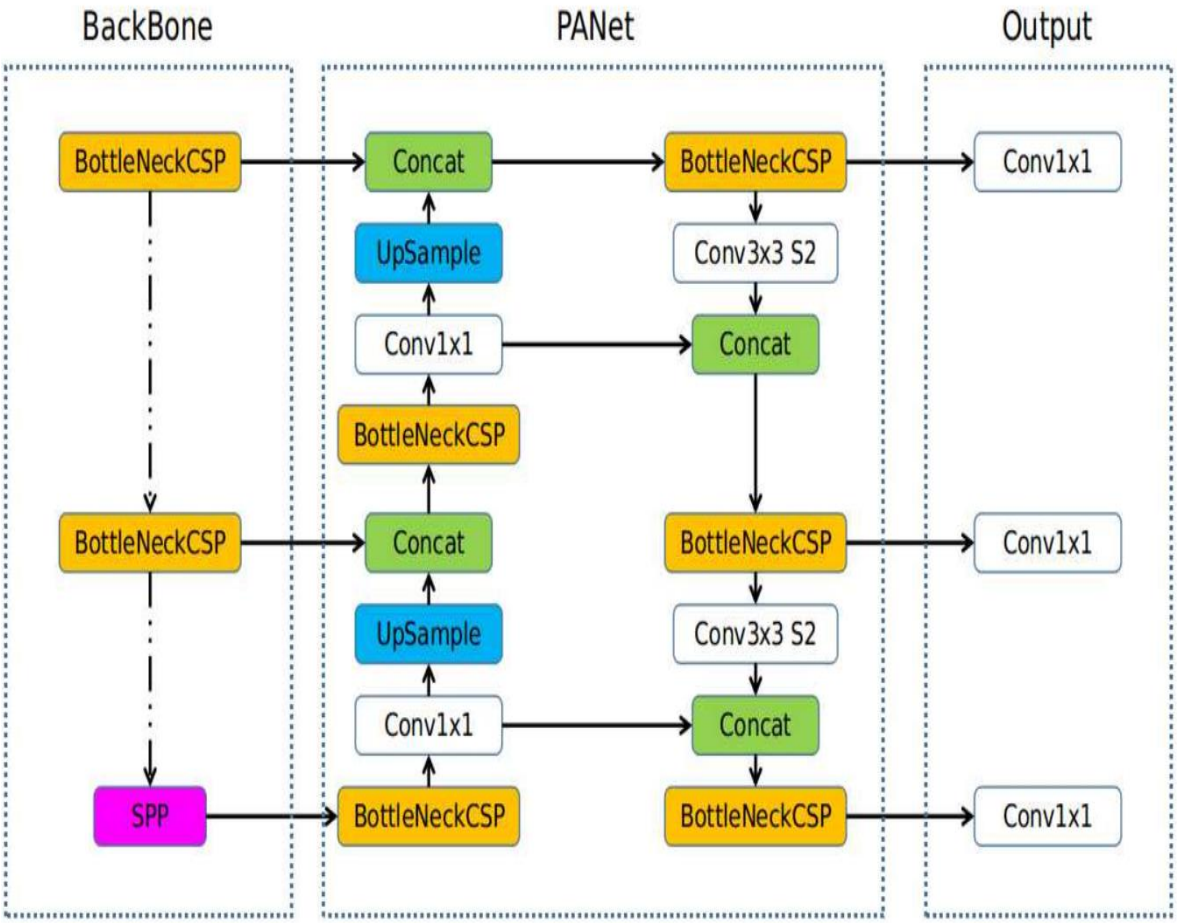


Figure 21: YOLOv5

3.6 Mathematical

The core of You Only Look One's loss function is the sum-squared error. There are some grid cells that have no items with a confidence score of 0. They completely overpower the gradients of cells that house the items. To reduce learning diveergence and system non-stability, You Only Look Once uses the maximum penalty (λ_{cord} = 5.0) their calculations

from BB including items and its minimum penalty (λ_{noob} = 0.5) for calcification when no object is present (V Thatte, [11] 2020). The loss function of YOLO is determined by adding the loss functions of all BB variables (x, y, w, k, confidence point, and class probability).

$$\mathcal{L} = \lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \Pi_{ij}^{obj} [(x_i - \hat{x}_i)_+^2 (y_i - \hat{y}_i)^2]$$
$$+ \lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \Pi_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)_+^2 \left(\sqrt{k_i} - \sqrt{\hat{k}_i}\right)^2 \right]$$
$$+ \sum_{i=0}^{s^2} \sum_{j=0}^{B} \Pi_{ij}^{obj} (c_i - \hat{c}_\iota)^2 + \lambda_{noobj} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \Pi_{ij}^{noobj} (c_i - \hat{c}_\iota)^2$$
$$+ \sum_{i=0}^{s^2} \Pi_{ij}^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(\hat{c}))^2$$

Equation 5: YOLO's Loss Function

The first portion of the eq$^{n.}$ calculates the loss associated with the estimated BB location and ground-truth BB location according to the co-ordinates (x_{center}, y_{center}).$\prod_{ij}^{obj}$, where 1 indicates if an item is detected in the jth forecasted BB in ith box and 0 indicates that it is not. Based on this assumption have the greatest resent IOU according to the base-truth, the predicted BB will be "responsible" for predicting an item.

$$\lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \Pi_{ij}^{obj} [(x_i - \hat{x}_i)_+^2 (y_i - \hat{y}_i)^2]$$

Equation 6: YOLO's Loss Function Part 1

In the same way as the first half of the equation, the 2nd portion of the "YOLO", loss function determines the error in BB length and height prediction. The quantity of mistake in this large boxes, on the other hand, has less of an impact on the equation than it does in the tiny box. Because both side and length are standardised between 0.0 and 1.0, the roots of their square

roots magnify the discrepancies between smaller and bigger numbers. Therefore the output, instead of using the side and length of the enclosing box explicitly, the square root of those values that is utilised.

$$\lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \Pi_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)_+^2 \left(\sqrt{k} - \sqrt{\hat{k}_i} \right)^2 \right]$$

Equation 7: YOLO's Loss Function Part 2

Either the item is visible in the BB or not, the loss of confedence points is calculated in both circumstances. The item certainty error is only penalised by the loss function if the forcast is accountable for that ground truthbox. When there is an item in the cell $\prod_{ij}^{obj}$, equals 1; otherwise, it equals 0. The reverse is $\prod_{ij}^{noobj}$.

$$\sum_{i=0}^{s^2} \sum_{j=0}^{B} \Pi_{ij}^{obj} (c_i - \hat{c_i})^2 \; + \; \lambda_{noobj} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \Pi_{ij}^{noobj} (c_i - \hat{c_i})^2$$

Equation 8: YOLO's Loss Function Part 3

Except for the $\prod_{ij}^{obj}$ term, the last half of the function used to calculate loss is comparable to the conventional prediction loss, that calculates the class probability's loss. Since You Only Look Once doesn't punish prediction errors since the obj. in not present the cell, this term is utilised.

$$\sum_{i=0}^{s^2} \Pi_{ij}^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(\hat{c}))^2$$

Equation 9: YOLO's Loss Function Part 4

3.7 Statistical

3.7.1 General

This GWH Detection database is made up of 4,698 squared patches derived from 2219 high-resolution RGB pictures taken over eleven sub-database (Table no. 3). It has 188,445.0 identified crop heads, with an average of 40 heads/image, which is in line having 15 to 65 heads/image that were desired. Nether less, there is a wide range of distribution in-between and inside sub-databases. To replicate on-field captured situations and increase challenge to bench-marking, we included roughly 100 photos with no heads. Only a few photos have more than 100 heads, with the most having 120. Differences in head volume, which varies depending on the genotypes and external circumstances, result in many peaks corresponding to various sub-datasets.

Table 3.6: Statistics for every attributes of the GWHD

Sub-dataset name	Nb. of acquired images	Nb. of patch per image	Original GSD (mm)	Sampling factor	Used GSD (mm)	Nb. of labelled images	Nb. of labelled heads	Average Nb. of heads/images
UTokyo_1	994	1	0.43	1	0.43	994	29174	29
UTokyo_2	30	4	0.6	2	0.3	120	3263	27
Arvalis_1	239	6	0.23	0.5	0.46	1055*	45716	43
Arvalis_2	51	4	0.56	2	0.28	204	4179	20
Arvalis_3	152	4	0.56	2	0.28	608	16665	27
INRAE_1	44	4	0.56	2	0.28	176	3701	21
USask_1	100	2	0.45	1	0.45	200	5737	29
RRes_1	72	6	0.33	1	0.33	432	20236	47
ETHZ_1	375	2	0.55	1	0.55	747*	51489	69
NAU_1	20	1	0.21	1	0.21	20	1250	63
UQ_1	142	1	0.2	0.5	0.4	142	7035	50
Total	2219	-	-	-	-	4698	188445	-

* Some labelled images have been removed during the labelling process

A somewhat skeewed gausian distribution with a mean average shape of 77.0 pixel can be seen in the shape of the BB surrounding the head. The root of the surface that is included in the find the usual dimension. Eventhough the individual longitudinal region applied is really not correspond to this siting geometrical of the coloured camera, this goes nicely with the required scale, i.e. near about 1.50 cm x 10.00 cm head dimensions with an mean resolution near about 0.40 mm/pixel, that symbolises a normal side of 97.0 pixel for every top part. The visual confirmation of item scale harmonisation over sub-database may be found in Fig 15.

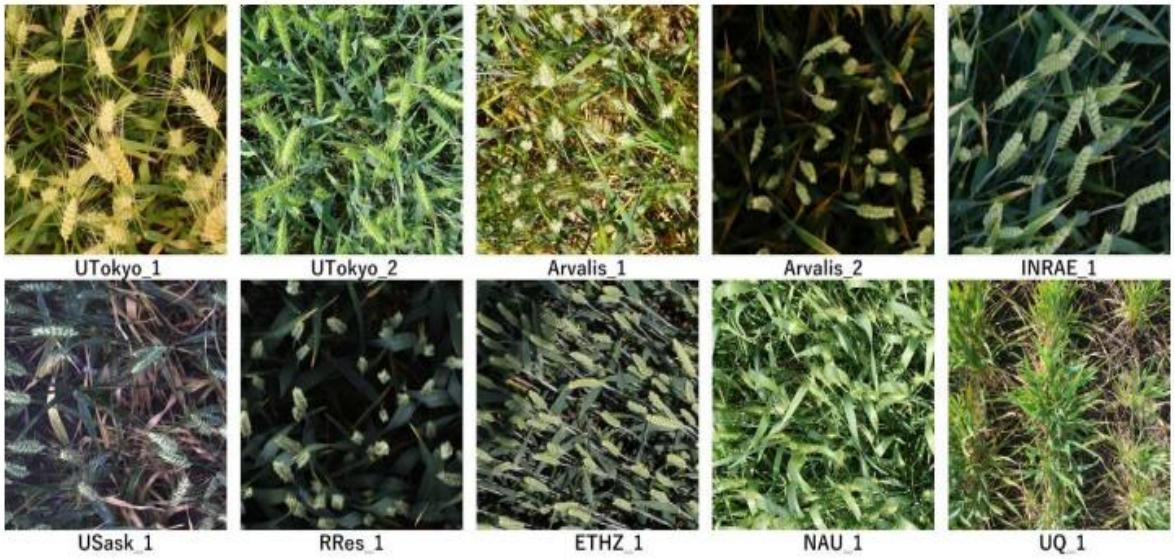


Figure 22: Samples of images, provided by different sites.

3.7.2 Comparison to other datasets

In the plant phenotyping industry, several open-source databases have previously been presented. For the counts of the leafs (rosette) and occasion segmentation, this CVPPP datasets have been frequently utilised. Segmented rosette leaves are also included in the KOMATSUNA collection, but only in time-lapse movies. Wheat head photos collected from a managed setting with single spike-lets tagged make up the Nottingham ACID Wheat database. However, photos from outside field situations, these are crucial in the use in real life of composition in the wheat breeding, are found in relatively few open-source databases. A few weeds categorization datasets have been released. Pictures of grazing grass and the semantic-segmentation classifications for field, weed plant, and clover plant categories are

included in the GrassClover database. Dot annotations have been added to data sets for measuring sorghum and wheat head.

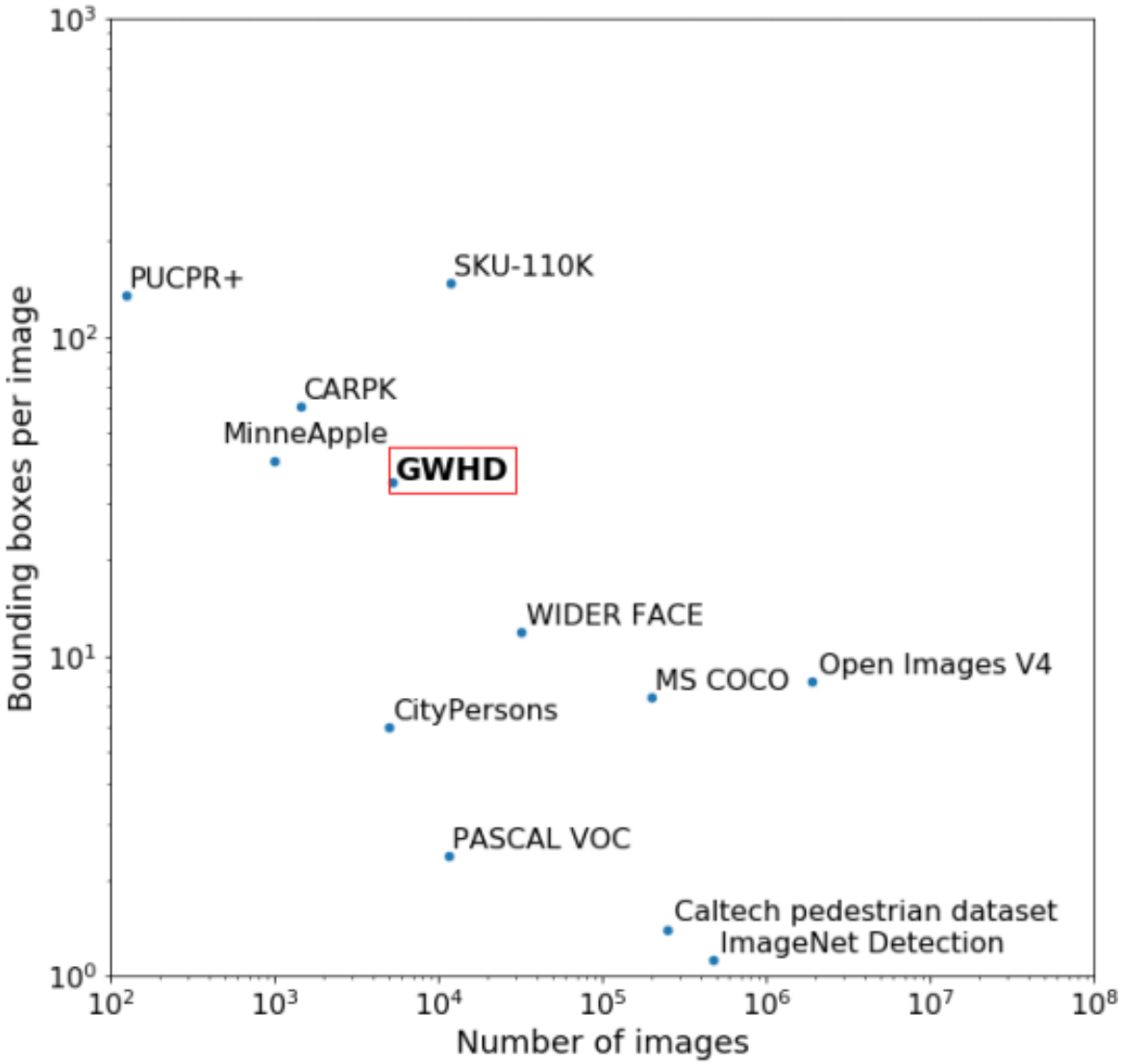


Figure 23: Comparing the GWHD dataset with other object datasets.

The dataset we are using, the GWHD database is now the biggest available to all annotated dataset publicly accessible for object recognition for the research of plant pheenotyping in terms of phenotyping datasets for object detection. MinneApple, the only similar collection in terms of phenotyping diversity, although it has fewer photos and less geographic variety. Additional datasets, such as MSCOCO or freely available Pictures of V4, has substantially bigger and sample a much wider range of item types, making them useful for a variety of other applications. Typically, the comparable photos include fewer items, usually below 10 per image (Figure 16). However, other datasets, such as PUCPR, CARPK, and SKU-110K,

are specifically designed to solve the challenge of detecting objects (e.g., automobiles, items) in crowded environments. All of them have a greater item density than the GWHD dataset, but less photos for CARPK and PUCPR, whereas SKU-110 has greater number of pictures than the GWHD database (Figure 16). This GWHD dataset is notable for its high prevalence of overlapping and occluded items. This complicates labelling and identification, especially when contrasted to SKU-110K, which does not appear to have any occlusion. Furthermore, wheat heads are sophisticated targets with a broad range of appearances, as earlier proven, and are enclosed by a diverse backdrop, making detection of wheat heads more challenging than recognising automobiles or tightly packed merchandise on shop tables.

Chapter 4 PERFORMANCE ANALYSIS

Any image may be implied to detect the wheat head using trained weights. If a wheat head is identifyed, then a bounding box is created around the item to enclose it and represent the likelihood that it is a wheat head.

The GWHD dataset contains 10 photos of outdoor wheat that are non-duplicated with the 3422 images used in the training. The ground truth bounding boxes are likewise not labelled on these photos. They may be seen as photographs of a agronomist went in his/her field and tested using weights learned in the model wheat-head identification network before.

Table 4.1 Result

Model	***IOU***
Faster RCNN	0.6889
EfficientNet	0.7152
YOLOv5	0.7644

```
/content/yolov5
Namespace(agnostic_nms=False, augment=False, classes=None, conf_thres=0.4, device='', exist_ok=
YOLOv5 v4.0-12-g509dd51 torch 1.7.0+cu101 CUDA:0 (Tesla V100-SXM2-16GB, 16130.5MB)

Fusing layers...
Model Summary: 232 layers, 7246518 parameters, 0 gradients, 16.8 GFLOPS
image 1/10 /content/yolov5/GWHD/test/2fd875eaa.jpg: 416x416 26 wheats, Done. (0.013s)
image 2/10 /content/yolov5/GWHD/test/348a992bb.jpg: 416x416 37 wheats, Done. (0.014s)
image 3/10 /content/yolov5/GWHD/test/51b3e36ab.jpg: 416x416 25 wheats, Done. (0.015s)
image 4/10 /content/yolov5/GWHD/test/51f1be19e.jpg: 416x416 18 wheats, Done. (0.011s)
image 5/10 /content/yolov5/GWHD/test/53f253011.jpg: 416x416 29 wheats, Done. (0.011s)
image 6/10 /content/yolov5/GWHD/test/796707dd7.jpg: 416x416 14 wheats, Done. (0.011s)
image 7/10 /content/yolov5/GWHD/test/aac893a91.jpg: 416x416 20 wheats, Done. (0.011s)
image 8/10 /content/yolov5/GWHD/test/cb8d261a3.jpg: 416x416 22 wheats, Done. (0.012s)
image 9/10 /content/yolov5/GWHD/test/cc3532ff6.jpg: 416x416 26 wheats, Done. (0.012s)
image 10/10 /content/yolov5/GWHD/test/f5a1f0358.jpg: 416x416 25 wheats, Done. (0.013s)
Results saved to runs/detect/exp3
Done. (0.588s)
```

Figure 24: Model Testing

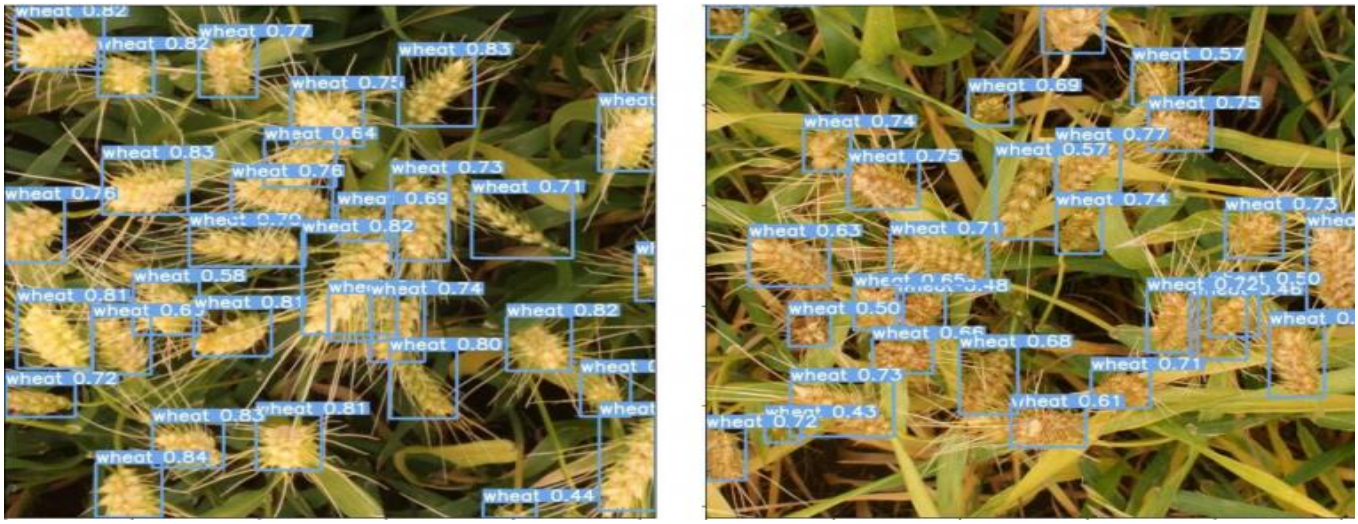


Figure 25: Result Example 1

Figure 26: Result Example 2

Chapter 5 CONCLUSIONS

5.1 Conclusions

In the computer vision world, there is still a lot of debate concerning the naming and enhancements of YOLOv5 for advancements that haven't truly created a breakthrough. Regardless of the nomenclature, YOLOv5's performance in terms of speed and accuracy is comparable to that of YOLOv4. There's little question that YOLOv5 will get greater contributions and recive greater growth potential in the upcoming years, thanks to the built-in Pytorch framework, which is more easy to use and has a greater community than the Daŕknet framework.

In reality, the subject of machine vision, particularly object identification, is only recently blossomed. As a result, even though the YOLO algorithm has evolved through five generations and is among the best object identification models, it is defective even right now. As a result, an AI system cannot be constructed just on the basis of an algorithm; it must incorporate other optimization methods as well as the most cutting-edge concepts in the study of machine vision in order for the given AI system to obtain these greatest results.

5.2 Future Scope

Artificial intelligence's contribution to several fields is growing all the time.
Deep learning models are becoming increasingly intelligent, and they can now handle complex tasks with ease. Agriculture is a sector that can benefit greatly from technological advancements. With numerous countries failing to satisfy demand and supply, it is critical to implement technology that can help improve production and overall efficiency. Computer vision is making good progress in agriculture. Although there are hurdles, as with any technology on the market, AI-powered computer vision services will need to address the associated issues before the technology is fully adopted. Before integrating such modern technologies, there are a few things to consider. We are, nevertheless, entering a world of digitization as a result of technical breakthroughs and upheavals. It is only natural to focus on the positives in order to maximize agricultural productivity with future technologies. In today's world, phenotyping is widely used to detect agricultural features for precision agriculture. Phenotyping has become a more efficient method because to advanced computer

vision algorithms. Image processing characteristics are integrated with computer vision algorithms to remove unnecessary crop data or information from photos, leaving only the relevant information on precise measurements. Techniques like depth estimation, colour enhancement, and identifying and segmenting the region of interest are all options for producing accurate results for future research. The crop breeds have greatly improved as a result of this research. We are quite sure that precision agriculture will prove much more useful in future, as global warming will make farming challenging, people will turn to computer vision techniques.

As for the new algorithms, the new RCNN and other region based Neural Network will come to light and researchers will be able to use them on new datasets for phenotyping and classifying grains from unwanted material. Also, as there is GWHD for grains, more datasets will come to surface. Our next step would be to use other datasets with other algorithms to create models that can be used with drones and other image capturing devices. Faster RCNN's new and evolved models can be used in future work. Since, hardware is becoming cheaper and faster, we expect more computational power will help to increase accuracy and FPS rates. Since, GWHD has recently updated their dataset, we can use more images to train the model to achieve better performance. Another things that can be done is that we can extend the model to detect wheat heads from different angles. Datasets that contain images of other staple crops such as rice, grapes and grains of importance, can be used to extend model's reach.

REFERENCES

[1] C. P. Z. J. Z. Wang Zhijun, "Method for identification of external quality of wheat grain based on image processing and artificial neural network," 2007.

[2] T. K. Ho, "Random Forest," in *IEEE*, 1995.

[3] F. Y. ZHEN Tong, "Research of Grain Pests Detection and Classification Based on SVM," 2006.

[4] N. A. Z. N. S. A. A. S. M. R. M. N. Shafaf Ibrahim, "Rice Grain Classification using Multi-class Support Vector Machine (SVM)," in *IAES International Journal of Artificial Intelligence*, 2019.

[5] H. M. Hongtao Zhang, "Feature Selection for the Stored-grain Insects Based on PSO and SVM," in *2009 Second International Workshop on Knowledge Discovery and Data Mining*, 2009.

[6] B. S. Harpreet Kaur, "Classification and Grading Rice Using Multi-Class SVM," in *International Journal of Scientific and Research Publications*, 2013.

[7] J. D. T. D. J. M. Ross Girshick, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[8] H. Z. J. L. Yufeng Shen, "Detection of stored-grain insects using deep learning," in *Computers and Electronics in Agriculture Vol.145*, 2018.

[9] D. E. Y. C. B. M. Bo Gong, "Real-Time Detection for Wheat Head Applying Deep Neural Network," in *Sensors (Basel, Switzerland)*, 2020.

[10] J. Redmon, "You Only Look Once: Unified, Real-Time Object Detection," in *IEEE conference on computer vision and pattern recognition*, 2016.

[11] V. Thatte, ". Evolution of YOLO — YOLO version 1," 2020.

[12] M. Menegaz, "Understanding YOLO," 2018.

[13] A. Kamal, "YOLO, YOLOv2 and YOLOv3: All You want to know," 2019.

[14] X. Z. S. R. J. S. K. He, "Deep Residual Learning for Image Recognition.," 2015.