Analysis of Deep Learning models On Low-Light Pest Detection

by

Md. Samin Irtiza
18101429
Fattah Ahmed
18101442
Md. Tahmidul Haque
18101570
Arifur Rahman Tamim
18101510
Samia Sultana
18101446

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
September 2022

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted or submitted for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Md. Samin Irtiza
18101429

Fattah Ahmed
18101442

Md. Tahmidul Haque
18101570

Arifur Rahman Tamim
18101510

Samia Sultana
18101446

Approval

The thesis/project titled Analysis of Deep Learning models On Low-Light Pest Detection" submitted by

1. Md. Samin Irtiza (18101429)
2. Fattah Ahmed (18101442)
3. Md. Tahmidul Haque (18101570)
4. Arifur Rahman Tamim (18101510)
5. Samia Sultana (18101446)

Of Summer, 2022 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on September 25, 2022.

Examining Committee:

Supervisor:
(Member)

Dr. Amitabha Chakrabarty
Associate Professor
Department of Computer Science and Engineering
BRAC University

Thesis Coordinator:
(Member)

Dr. Md. Golam Rabiul Alam
Professor
Department of Computer Science and Engineering
BRAC University

Head of Department:
(Chair)

Dr. Sadia Hamid Kazi
Associate Professor and Chairperson
Department of Computer Science and Engineering
BRAC University

Abstract

It is undeniable that in recent years, exceptional progress has been made toward building the most accurate and efficient object detectors. However, existing low-light object detectors still require a substantial amount of resources to perform at their best. Our main goal in this research is to train and evaluate recently developed deep learning object detection models on low-light images and see if they can show decent performance without any additional enhancement networks. Furthermore, we aim to achieve those results with minimum computational cost. In this research, we have created our own custom dataset from a publicly available insect image dataset called 'IP102'. The new dataset now named 'IP013' consists of 13 classes of insects and approximately 8k annotated images. Moreover, we chose recently developed YOLOv7 and DETR object detectors and compared their performance against now older state-of-the-art RetinaNet and EfficientDet deep learning models. YOLOv7, EfficientDet, and RetinaNet are purely CNN-based models whereas DETR uses a Transformer as both encoder and decoder and a CNN as the backbone. Our research shows that YOLOv7 outperforms all of the other models with a mAP0.5:.95 of 45.9 while using the lowest training time and the model that used the least computational resources was EfficientDet which admittedly showed lackluster mAP0.5:.95 of 33.2 with only 3.9M parameters and using 2.5 GFLOPs.

Keywords: Insect; Object Detection; Deep Learning; Enhancement network; Low-Light Images

Acknowledgement

All praise to the Great Allah for whom our thesis have been completed without any major interruption.
Our work would not be even close to completion without our honorable supervisor Dr. Amitabha Chakrabarty Sir and Shahriar Hossain Sir. For their kind support and advice in our work, we have come this far.
And finally to our parents without their throughout support it may not be possible. With their kind support and prayer we are now on the verge of our graduation.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

CNN Convolutional Neural Network

$ELAN$ Efficient Layer Aggregation Network

FPN Feature Pyramid Network

$R-CNN$ Region-Based Convolutional Neural Network

$ReLU$ Rectified Linear Unit

$ResNet$ Residual Network

SSD Single-Shot Detector

$YOLO$ You Only Look Once

Chapter 1

Introduction

Without agriculture the world can not survive. Agricultural products are the most important component in our daily life. But, like all other issues we have to face problems during harvesting. And it can be solvable through technology .Yes ,it is true that technology is advancing, development is occurring in the agriculture sector .Many inventions are also there. But it is also sad that during harvesting farmers are facing difficulties specially keeping insects away from agricultural fields. During the daytime it is easier to keep the crops safe from insects. However, in dark or in bad illumination it is one kind of impossible because of absence of light. Besides, in real life it is not always possible to get a perfect lighting condition for images, thus an insect detection model should be able to work with all kinds of realistic lighting conditions. One of the most common challenges of insect detection in a realistic environment is low illumination. Poor lighting conditions introduce noise and a lot of other factors such as bad contrast, reflectivity, shadow, etc. which makes it very difficult to insect pest objects with decent accuracy in such an environment. In recent years, the CNN(Convolutional Neural Network) based feature networks have been very promising and efficient for object detection. Their performance has led to a reduction of image enhancement models before training the datasets. The models we have used for training our datasets also come with a backbone which has a CNN feature network. Those have done the work of enhancing the images. So we have decided to not use any image enhancement algorithms before training the images through our models. Moreover, for object detection there are multiple approaches that mainly emphasize two performance metrics: 1) Accuracy 2) Runtime. The most accurate models are two-stage models that require a high level of computational resources and have a longer runtime. Among the two-stage models, transformer models are giving promising results and accuracy closer to the state of the art models. On the other hand, there are one-stage models that are usually faster than two-stage models but are less accurate and demand significantly fewer resources. The majority of the research on object detection employs CNN-based models for example R-CNN, FASTER R-CNN, YOLO, SSD, and EfficientDet. These models are becoming faster and more precise as more research is being done each year. We will try to implement CNN and transformer based models for our dataset.

1.1 Problem Statement

Object detection, as more and more projects depend on its accuracy and speed, the necessity of creating a faster and more accurate model is also increasing. Moreover, object detection has to meet certain requirements such as real-time processing on a large enough scale. The major problems of object detection are bad illumination, which includes low light, lens flare, high noise, contrast issues, incorrect color grading, etc. For fixing these errors, multiple algorithms are run which increases more complexity. More complexity makes the process resource-heavy and time-consuming.

To correctly detect an object from an image, the computer vision needs to compare the given image with the reference images it has. And the reference images are taken in good lighting so that the distinguishable attributes of the object can be easily seen. The images in low illuminated places face a big problem when it is compared to the reference images, Guo et al [25]

Moreover, noise increases in an image when the lighting conditions of an image are inadequate. When an image has noises in it, firstly, the noise and the object must be differentiated. After that, it increases complexity since extra work has to be put in to remove the noise from the picture and replace the pixels with the original color grading, Loh et al [11].

Another important aspect for the correction of low light images is correcting the color grading and saturation. Different objects might look the same if the correct color and saturation is achieved. For example, to find the last known whereabouts of a missing person, surveillance footage is generally checked. For this scenario, correct color and saturation are required to find a certain individual, Yu et al [23].

In the Agriculture field, the most difficult situation is to detect the insects in night time.Hence , we are attempting to find a solution to this ,like our proposed method will detect the pests in the dark .we will make a system which will work range basis and it will be efficient toward the whole field range .Besides It will give alarm when any insect or pest will try to cross their range. So, we are attempting to find a solution to this,

Are the newer detection models better or worse, in terms of efficiency, accuracy, than the older robust models in detecting pests in low light conditions?

1.2 Research Objective

Since 2012, almost all of the state-of-the-art object detection models have a CNN-based backbone. These models have performed well with many types of datasets and kept on giving promising results with almost all of them. But in the last two years, a totally new way of object detection has emerged, which is known as the Transformer based model. And these models have been performing on par with the robust and established CNN-based models. That is why we have decided to implement Transformer based models in our dataset. Not only will we train our dataset by the transformer model but also we will try to put up a comparative analysis between the new models and the older but still state-of-the-art models. Moreover, our goal is also to find a better-performing model for our low-light images of agricultural pests.

1.3 Thesis Outline

The introduction which is chapter 1 provides information about the motivation of our research, the problem statement, and the objective of our research.
Chapter 2 highlights similar research papers which are related to our works of object detection in low light conditions.
Chapter 3 explains the architecture of the models used in our research.
Chapter 4 describes how we have implemented, trained, and in the end tested all the models.
Chapter 5 showcases and comparisons of all test results.
Chapter 6 gives a conclusion of our project followed by the working process of COCO evaluation metrics.

Chapter 2

Literature Review

2.1 Related Works

According to the paper [17] , Carion et al. has implemented the DEtection TRansformer model. It is a new design based on the transformer architecture used for object detection and panoptic segmentation. They have trained their model on COCO dataset and compared results with an optimized Faster R-CNN baseline. They have created not only one type of model but also a few different variants named DETR-DC5, DETR-R101 which uses Resnet-50 and Resnet-101 as their backbone. Their self attention model has performed better in detecting comparatively bigger objects in the images. The architecture and the model as a whole has a lot of challenges to overcome since the model is very new. After training different variants of the model against the COCO dataset, in 50 epoch, DETR model has achieved an average precision of 62.4% with 28 fps(frame per second), DETR -DC5 model has achieved an average precision of 63.1% with 12 fps and DETR-R101 model has achieved an average precision of 63.8% with 20 fps.

On paper [9], a one-stage model was proposed which is called RetinaNet which is enhanced with the help of a function called FocalLoss. Two-step models are generally better object models compared to one-step models. But one-step models have the potential to have the same accuracy while also retaining the same speed and simplicity they had before. The extreme foreground-background class imbalance created by dense models during training is the root cause. Reshaping the standard cross entropy loss so that it can down-weight the loss assigned to well-classified examples. FocalLoss makes the model focus more on the hard examples and ignore the huge number of easy examples. And to implement this and evaluate the loss, a new model is designed which is called RetinaNet. With the help of FocalLoss, RetinaNet performed with the same speed as a one-step model with the same accuracy as a two-step model.

In another research paper [27] , Wang et al have presented the latest version of the YOLO family, the YOLOv7 which excels in both precision and speed and has the highest precision above all the known state of the art real time object detection models. They have followed the architecture of the previously released YOLOv4, scaled YOLOv4 and YOLOR and enhanced it. The authors have introduced E-ELAN (Extended Efficient Layer Aggregation Network) and also implemented compound scaling for concatenation based models. E-ELAN is inspired from the ELAN [26]

architecture, the paper of which has not yet been released. They also got inspiration from EfficientNet [15] and a paper from Dollar et al [23], to implement the compound scaling in their concatenation based architecture which helped them to preserve the characteristics of the model's original design and its ideal structure. The author also brought from its predecessor, the bag of freebies which is to enhance the model precision by changing the training strategy. The strategy they have used to improve the architectures are , planned reparameterization and multiple head structure that is deep supervision. In the deep supervision method, they have generated soft labels where the fine soft label is used for the main head and coarse labels for the assistant heads. All these things will be elaborated in the architecture section of the YOLOv7. Combining all these techniques they have created the YOLOv7 which outperforms all the transformer based models and convolutional based models in the world of real time object detection. YOLOv7 has the accuracy of 56.8% with 36 fps(frame per second) inference speed whereas the best transformer model SWIN-L at 9.2 fps speed has the precision of 53.9% and previously best convolutional model ConvNeXt-XL has 55.2% AP at 8.6 fps.

An approach proposed in this paper [18] aims to improve conventional CNN based on one-stage detector architecture. There exist several popular CNNs for object detection such as FAST R-CNN, R-CNN and FASTER R-CNN which are two-stage detectors and have high resource demand and long runtime. For this very reason, the researchers opted for proposing a one-stage CNN like SSD and YOLO. However, these one-stage detectors have a poor accuracy rate compared to the two-stage detectors. The researcher proposed a new CNN scaling method called 'Compound Scaling' that efficiently scales the network and can perform very close to scaled two-stage detectors. Moreover, they proposed a new Bidirectional Feature Pyramid network. EfficientNet and BiFPN(Bidirectional Feature Pyramid Network). This detection model uses EfficientNet as the foundation network and BiFPN as the feature fusion network. Tan et al [18] tested the network on COCO 2017 detection datasets [3] with 118K training images and 5k validation images. The test results were compared against many popular models such as RetinaNet and YOLOv3 which are both one-stage detectors in addition the results were tested against two-stage detectors such as MASK R-CNN and AmoebaNet.The model's baseline network EfficientDet-D0 performs better than YOLOv3 with an mAP of 34.6 having only 3.9M parameters and the highest scaled model EfficientDet-D7x performed better than both AmoebaNet and Mask R-CNN with an amazing mAP of 55.1 but only having 77M parameters.

In [4] They proposed a methodology named residual learning to make it less complex as well as to prepare systems that are much more profound than those already utilized.The Researcher assessed their strategy using the 1000-class ImageNet 2012 identification sample [5] . Besides that, using that dataset they analyze residual nets(ResNet) with a depth of up to 152 layers —8× more profound .But nevertheless it has less difficulty than VGG nets [6].They have also examined a variety of plain/residual nets and found repeating patterns. The Plain Net they have used was primarily propelled by the logic of VGG nets [6] and for ResNet they added easy route associations to the plainNet which turn the network into its partner remaining form. Nonetheless An outfit of these ResNets accomplishes 3.57% error on the ImageNet training set. Moreover,That output helped the Authors to get first

position on the ILSVRC 2015 classification errand.Furthermore ,They did additional research on the CIFAR-10 dataset [1] , which comprises 50k training examples including 10k pictures for testing in 10 categories. The astonishing thing is that on the challenging COCO dataset They achieved a 6.0% increment in COCO's standard metric (mAP@[.5, .95]), which could be a 28% relative enhancement.

The researchers of this paper [21] used LIME, Retinex-Robust, and Retinex-Net to enhance the low-illuminated images and trained a model on the enhanced data using a lightweight one-stage RFB-Net. To evaluate the performance they used ExDark [11] and on an edited version of COCO datasets. After the evaluation, they were disappointed after discovering that the accuracy in unit time of the model which trained on enhanced data was inferior to that of the model trained on unenhanced data. Following this discovery, they proposed a new model based on tried and true RFB-Net model which they aptly named Night Vision Detector or NVD for short. This model is built with the combination of an FPN(Feature Pyramid Network) and a Context Fusion Network. The researchers again used the aforementioned COCO* and ExDARK datasets to train and test their model and the detection performances were compared using standard COCO evaluation APIs. After the experiments, they compared their results with a basic RFB-Net and it showed that NVD was able to outperform basic RFB-Net on low-light detection performance by 0.5% $\sim$ 2.8% on all standard COCO evaluation metrics where the FPN alone improves the performances, particularly on small objects detection by a factor of 2.2%.

This [19] paper used a pre trained EfficientDet model (uses Efficient Net as backbone and BiFPN as the feature fusion network) and compared it against the YOLOv3 model. The results of EfficientDet are significantly better than that of YOLOv3. The 2 models were trained on COCO dataset consisting of 1.65m training images, 80k validation images, and 80k test images. YOLO-v3 combined with their pyramid image enhancement network was able to achieve 24.3(pedestrian) and 36.7(car) precision on the COCO dataset whereas EfficientDet and the same image-enhancement network was able to achieve a great precision of 53.8(pedestrian) and 71.4(car). Further experiments showed that their image enhancement network model is better than U-Net and other similar networks for low light enhancement and EfficientDet's object detection accuracy is better than YOLO-v3.

Chapter 3

Background Analysis

For our research, we have chosen four state-of-the-art models to train on our dataset. The EffcientDet and RetinaNet are both older CNN-based models whereas YOLOv7 and DETR are newer models with both CNN and CNN-Transformer architecture respectively. In this research, we studied the architectures of each model and learned about their unique characteristics. In this section, we provide an in-depth explanation of each model's architecture.

3.1 EfficientDet Architecture

First introduced in this [18] paper by the Google Brain Research team, EfficientDet became a very robust and accurate object detector SOTA model. The researcher built the model using their previous work EfficientNet[32]CNN as the backbone. Moreover, they used a Bidirectional Feature Pyramid Network that acts as the feature network and helps to integrate high and low-level features together. Then they used two feed-forward convolution networks that serve as the shared class/box prediction network. the figure 3.1 below shows the whole network architecture.

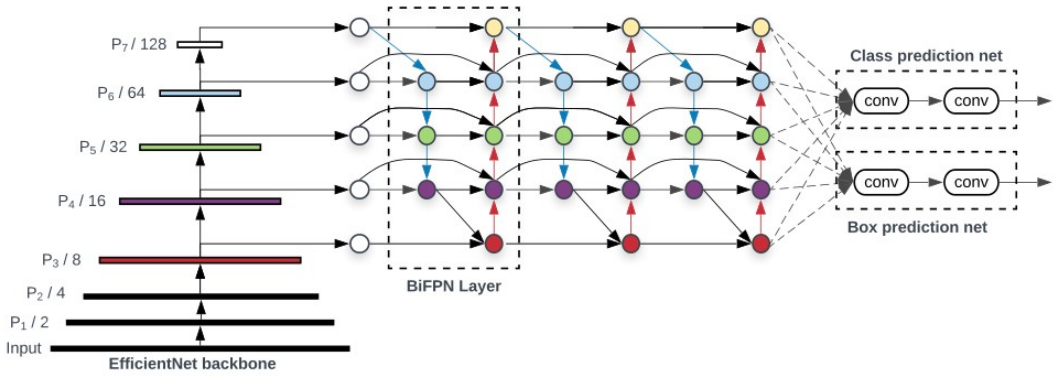

Figure 3.1: EfficientDet Network Architecture. [18]

3.1.1 EfficientNet: The Backbone

Conventionally, a CNN's performance was increased by depth scaling, meaning only increasing the number of layers of convolution. Increasing the number of layers used to be a problem because of the vanishing gradient problem. Since, the deeper the layers, the more a model was prone to face the 'vanishing gradient problem' before reaching convergence. However, after the introduction of ResNets[4] (residual networks) increasing the depth to achieve better results was no longer the problem. When residual networks came along, creating bigger and better networks by increasing depth became the new convention. At the time, networks such as ResNext [10] and AmoebaNet[14] used depth scaling and used bigger image sizes to achieve significantly better results than earlier models. However, further research on scaling later revealed that width and resolution scaling where width is the number of channels and resolution is the resolution of the input image respectively could further improve the network's performance. Even though scaling was needed to improve performance, there were no good methods that can be used to scale a network efficiently. To solve that issue, Tan et al. in their previous work, created a new network scaling method called 'Compound Scaling. This is the core concept behind EfficientNet. Compound scaling helps to calculate how much a network's depth, width and resolution need to be scaled with respect to one another to achieve the best performance efficiently. The following equation 3.1 is used to determine the compound scaling factor:
Here,
α = depth,β = width, γ = resolution

$$f = \alpha \cdot \beta^{\phi} \cdot \gamma^{\phi} \tag{3.1}$$

The value of the alpha, beta, gamma, and phi each represents depth, width, resolution and compound coefficient respectively. In the paper, the researchers calculated the values of the variables using a grid search algorithm on a baseline model they named 'EfficientNet-B0'. The values for the baseline model they came up with are the following: $\alpha = 1.2, \beta = 1.1, \gamma = 1.15, \phi = 0$ The baseline model itself was designed using a multi-objective NAS(Neural Architecture Algorithm). The following diagram 3.2 shows the design of the baseline model:

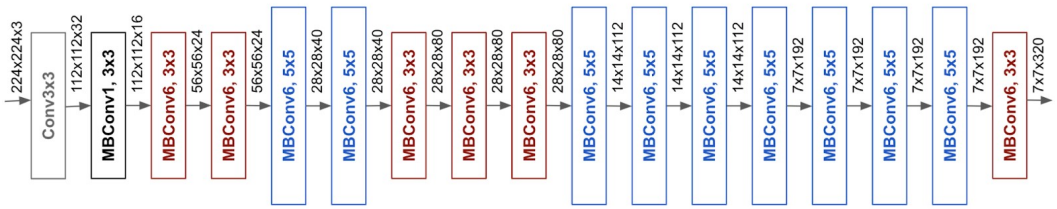

Figure 3.2: EfficientNet-B0 architecture. [18]

By increasing the compound coefficient ϕ by 1 each time, the baseline model was scaled and the researchers produced 7 different EfficientNet models named 'EfficientNet-B1' to 'EfficientNet-B7'. The 'EfficientNet-B7' was the largest and had by far the best performance with significantly fewer parameters than most CNNs at the time. That is why It is still considered a state-of-the-art backbone even after 3 years.

3.1.2 Bidirectional Feature Pyramid Network: The Neck

A feature pyramid network or FPN in short is widely used to combine low and high-level features together. In a CNN, each convolution operation gets rid of smaller information and only retains larger ones. That means, the deeper layers contain only the largest or most abundant pieces of information and lose all the finer details and become abstract. Moreover, the produced feature maps scale down significantly the deeper the layers go. In order to represent these multi-scale features, FPN was introduced in this paper [8] which proposed a top-down process that can combine multi-scale features. Further improvements were added to the architecture by other researchers in later years. However, these improvements were not very efficient and sometimes required thousands of GPU hours to get good results. By analyzing these architectures and making further optimizations, Tan et al. in their paper [18] proposed a new type of Bidirectional FPN with weighted Feature Fusion. This new FPN has both top-down and bottom-up information flow so that low-scale and high-scale features can combine either way. Moreover, The weighted feature fusion algorithm gives a weight to each input that signifies how important the input feature is. The researchers came to a realization during their research that all of the feature fusion methods when fusing features together usually resize the images to a single resolution and treat each of the inputs equally. However, this slows down learning because all of the features are not equally important. Therefore, giving weight and prioritizing some features over the rest can greatly lower the computation cost and achieve better performance. The researchers conducted an ablation study to measure the impact of EffcientNet and BiFPN on the accuracy, parameters, and FLOPs(FLoating point OPerations) compared against ResNet50 a popular lightweight backbone paired with a traditional FPN. The table 3.1 below shows the results:

Models	AP	Parameters	FLOPs
ResNet50 + FPN	37.0	34M	97B
EfficientNet-B3 + FPN	40.3	21M	75B
EfficientNet-B3 + BiFPN	44.4	12M	24B

Table 3.1: EfficientNet and BiFPN performance comparison [18]

As shown in the table 3.1 BiFPN is proven to be really effective in increasing accuracy while reducing the parameters thereby lowering computation cost.

3.1.3 Feed-Forward Class/Box Prediction Network: The Head

The head of the network consists of 2 feed-forward convolution networks that share layers with the aforementioned BiFPN. The width of the class/box prediction network, therefore, is fixed and the same as the BiFPN width. Moreover, scaling the network only requires depth to be increased and it is increased using the following equation [18]:

$$D_{box} = D_{\text{class}} = 3 + [\phi/3] \tag{3.2}$$

Overall, EfficientDet is one of the pioneering models and it is still considered a robust state-of-the-art object detector. During our research, we have found that

although it has been surpassed by others specially transformer-based models in accuracy in recent years, only a handful of other models show promise in delivering good accuracy with minimal computation cost.

3.2 YOLOv7 Architecture

In the world of object detection YOLO(You Only Look Once) models have been one of the greatest due its outstanding performance. YOLO models are based on Fully Connected Neural Network. YOLOv7, the latest addition to the YOLO family, follows the architectures of the previous YOLO models. Bounding boxes and class probabilities are predicted by YOLO using a sole convolution neural network. YOLO predicts multiple bounding boxes and class probabilities for each box as well as for all bounding boxes across the classes in a single evaluation in one step and for one unit, making it a one stage detection model. YOLOv7 is derived and has brought reformation to the architecture of YOLOv4 [16] , scaled YOLOv4 [20] and YOLO-R [24]. Also, since the authors of YOLOv4 and YOLOv7 are the same, YOLOv7 is trained on only the COCO dataset rather than using any other datasets' pretrained backbones.

YOLO predicts the bounding boxes, the classes of the bounding box, and the object of the bounding boxes after featuring input photos through a backbone, combining and mixing them in the neck, and passing them on. The input image is at first preprocessed then aligning it into a 640*640 RGB image, input it into the backbone network, and pass the backbone network at the head layer according to the three-layer output in the backbone network. YOLOv7 is using ELAN [26] and E-ELAN as the backbone of the network. As of today, the ELAN[26] paper has not yet been released.

3.2.1 Extended Efficient Layer Aggregation Networks

E-ELAN is used as the computational block in the backbone as shown in the figure 3.3. In the paper the authors have introduced Extended-ELAN which is based on the ELAN computational block. In ELAN they have shown that a deeper network may successfully learn and converge by managing the shortest longest gradient path. It is used as the computational block in the backbone. The Extended-ELAN is using expand,shuffle and merge cardinality to continually improve the network's capability for learning while maintaining the initial gradient path. So rather than changing the transition layer it only changes the computational block. The backbone layer of YOLOv7 consists of multiple E-ELAN layers, BConv layers(composed of convolution layer, batch normalization layer and an activation function) and MPConv layers at times doubling the channels, halving the length and width and extracting features.

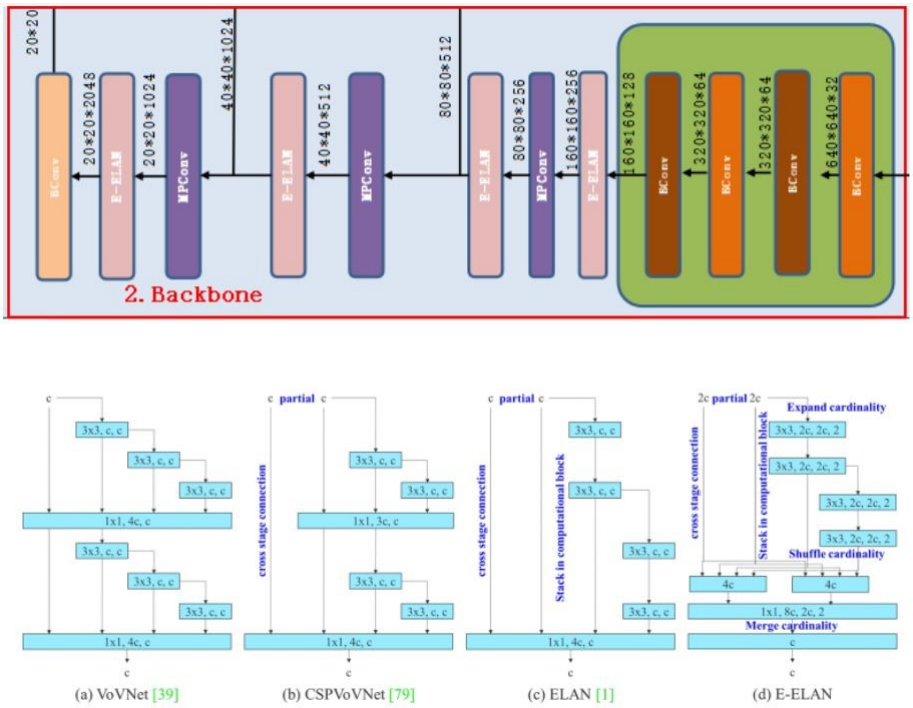

Figure 3.3: Extended Efficient Layer Aggregation Networks (Extended-ELAN) [27]

While redesigning this backbone, the authors considered the memory cost of all the layers and model parameters, ratio of the I/O channels, the amount of element wise operation, efficient gradient path calculation and back propagation. As a result the convergence is faster since the weights are updated efficiently and also improving the accuracy.

3.2.2 Model Scaling for concatenation based models

Model scaling is mostly used to modify specific model properties and produce models at various scales to accommodate various inference rates. Compound model scaling was first introduced by the authors of EfficientNet [15], which takes into account resolution, depth and width as a factor for scaling. However YOLOv7 is concatenation based architecture which means width and depth are interrelated. Unlike model scaling in usual CNN models, in compound scaling of concatenation based architecture it does not allow to assess various scaling variables individually, they must be taken into account together. The model's original design qualities could well be preserved by using the compound scaling approach, keeping the ideal structure intact i.e. scaling any of the factors of the computational block will change that channel's output channel. So scaling the depth (number of layers) will also coherently scale the width (number of channels) as shown in the figure 3.4. So, when the compound model scaling that are used in traditional CNN is used in the concatenation based model, it is seen that the width is changed when scaling is performed on depth.

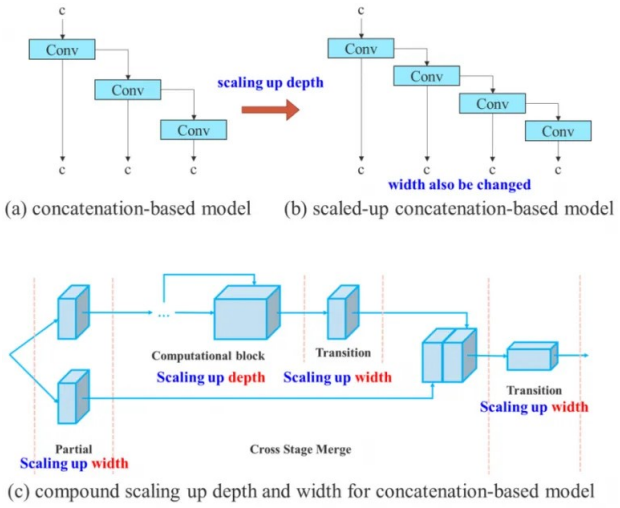

Figure 3.4: Concatenation based model with compound scaling [27]

3.2.3 Trainable Bag of Freebies

In YOLOv4 a method was introduced that could enhance the performance and inference speed by changing the training strategy. YOLOv7 introduced two new strategies in the form of trainable bag of freebies which includes planned model reparameterization and making the head coarse to fine.

Planned Reparameterized Convolution

Model parameterization is closely similar to model ensembling. It is used after training to improve the inference results, which also leads to increase in training time. The idea is to develop a model that is more resilient to the general patterns by adding up a set of model weights and averaging them. YOLOv7 is using module level reparameterization.In this method shown in the figure 3.5, the model is split into a number of modules during training and then during inference the modules are ensemble into one single model. RepConv [22] performs excellently on the VGG[6] , but in ResNet [4] or DenseNet [7] the accuracy becomes poorer which happens because of the identity connection. For this reason, they used the RepConv without the identity connection calling it the RepConvN because identity connection makes RepConv perform poorly when reparameterized convolution is used.

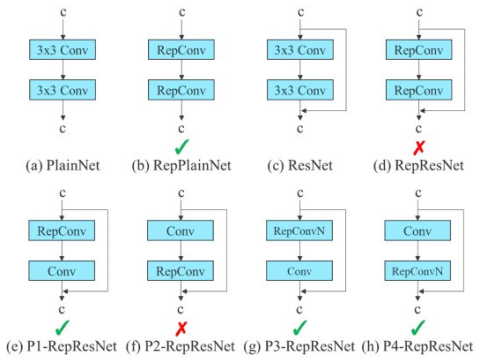

Figure 3.5: Model Reparameterization for YOLOv7 [27]

Coarse for auxiliary and fine for lead loss

YOLOv7 uses the technique of multiple auxiliary heads in the network. The final output is contained by the lead head. The auxiliary heads help to train in the middle layers. This mechanism is called Deep Supervision [2] . On the top of this, they also introduced the label assigner method which assigns soft labels after taking into account ground truth and network prediction outcomes. So the auxiliary head and lead head are guided by the lead head prediction in the label assignment procedure. To develop a coarse-to-fine hierarchical labels for the head learning for both auxiliary and lead, YOLOv7 employs the lead head prediction as an assistance. For this, YOLOv7 is following two strategies e.g, lead guided assigner and coarse to fine lead guided assigner. The explanation is show in the figure 3.6

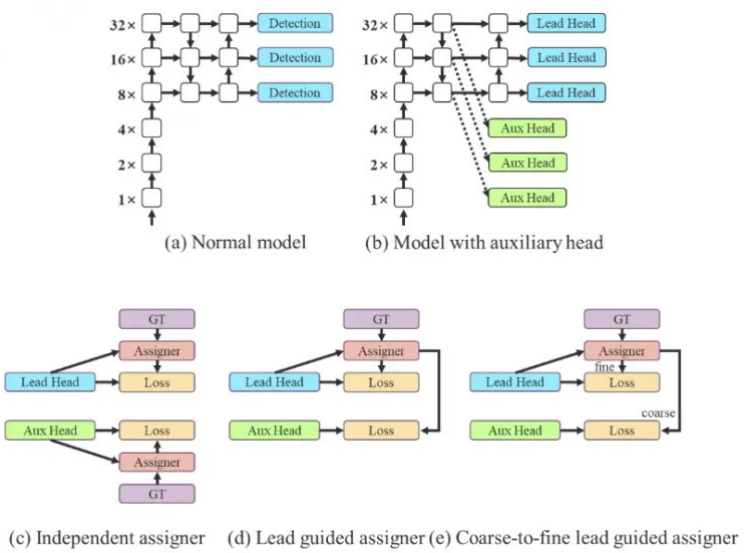

Figure 3.6: Coarse label for auxiliary and fine label for lead head [27]

Lead head guided label assigner: It is primarily determined using the lead head's prediction outcome and the ground truth, and soft labels are generated via the optimization process. Both the heads will train using this set of soft labels. This is necessary since lead head has a somewhat robust learning potential, which means the soft label that results from it should be more accurate in capturing the dispersion and connection between the source and target data. Lead head will be better able to concentrate on learning residual information which has not been learned, by enabling the shallower auxiliary head to grasp the data that the lead head has learned.

Coarse to fine head guided label assigner: Here everything is the same but instead of producing one soft label it is generating two soft labels e.g, one coarse and one fine. The fine soft label is the same as the soft label generated in the previous assigner. However, by loosening the restrictions on how many grids may be considered as positive targets during the positive sample assignment procedure, the coarse label is produced. It is because an auxiliary head's capacity for learning is less than that of a lead head, thus in the object detection task, we will concentrate on maximizing the recall of the auxiliary head so as to avoid losing the information that has to be learnt. Due to this mechanism, the upper limit of the optimizable

fine label is always greater than the upper bound of the optimizable coarse label throughout the learning process by allowing to dynamically adjusting the labels. Because if the added weight of the coarse label is almost equal to that of the fine label, it might lead to inaccurate final predictions.

3.3 DETR Architecture

Developed by the Facebook AI team, DEtection TRansformer detects objects in a very unique way. This model uses a feedforward convolution network which is connected to the transformer encoder decoder architecture. To explain the model in simple terms, it uses the transformer architecture to predict the objects and their location in the image. DETR has come up with accuracies and runtime performances almost as equal as the Faster R-CNN when they are trained on the COCO dataset. The figure 3.7 gives a brief idea about the architecture.

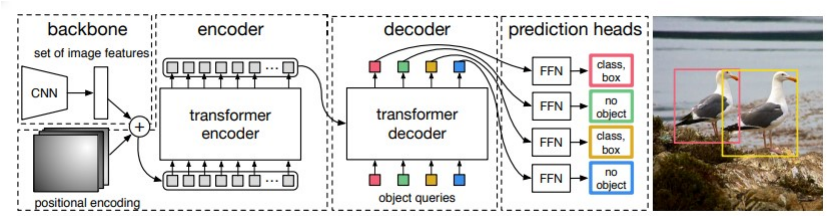

Figure 3.7: DETR architecture [17]

3.3.1 Architecture Breakdown

DETR comprises 3 parts, such as: a CNN backbone for getting a set of image features, a transformer and a feed forward network which uses a bipartite matching system to detect the loss and accuracy of the model.

The CNN backbone takes an input image with 3 color channels (RGB). CNN batches all the images together using adequate 0-padding to make sure that all the images have the same dimensions. The resolution of the image is also toned down and is given a number. A typical activation map is generated,

$$f \in R^{(C \times H \times W)} \tag{3.3}$$

Here, c is 2048 and H,W is the height/32 and width/32 of the largest image in the batch.

The encoder creates a height times width squared matrix which is also called the attention matrix to predict the bounding box of the object. A transformer model has a unique feature that lets it communicate and exchange data between any part of the sequence. Suppose, a part of the sequence is detecting the beak of a bird and another part is detecting the tail of the bird. And this model can connect these two pieces of information and predict that the beak and the tail are both the parts of the same bird. The encoder also uses a height times width squared matrix as an attention matrix in every feature channel. Any point in this attention matrix represents two diagonally located vertices of a bounding box. For the positional

encodings, they have used fixed positional encodings according to Parmar et al. [12] and Bello et al. [13]. When the model detects an object in the image, it refers to a point in this matrix, and aggregating all the respective points found in the matrix it becomes more accurate with the prediction of the bounding box of the image

Afterward, this is passed on to the transformer decoder and the decoder does a specific number of object queries in the image.. A certain query always tries to detect an object in a specific location in the image regardless of the position of the object in the image. For example, a query named K will always look for objects on the top left for all the images, and another quert D will always try to detect objects in the middle part of the image. While training the COCO dataset, they did 100 queries for every image. A visualization of the object queries is given below

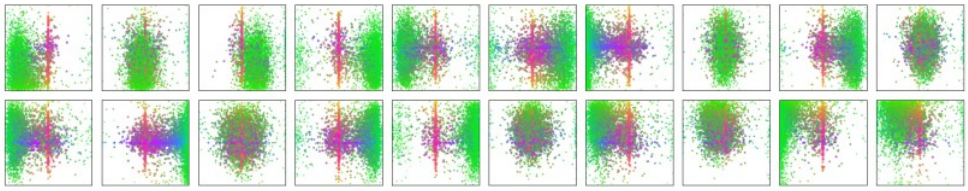

Figure 3.8: Object query [17]

In the figure 3.8, the green parts are places where the object query is trying to detect objects in the image. Each box denotes different queries done in the same image.

A 3-layer perceptron is used with ReLU activation function to do the final prediction which also has a hidden dimension, d. A feed-forward network deduces the normalized center coordinate and height, width of the bounding box. It also uses a softmax function to predict the class label of the input image. Since, this model predicts a fixed number, N, of objects in every image, N is much larger than the number of objects in the image. So the extra objects must be labeled as no object (∅). This is similar to the "background" class that is used for standard object detection. A visual representation of the encoder self attention can be seen in figure 3.9.

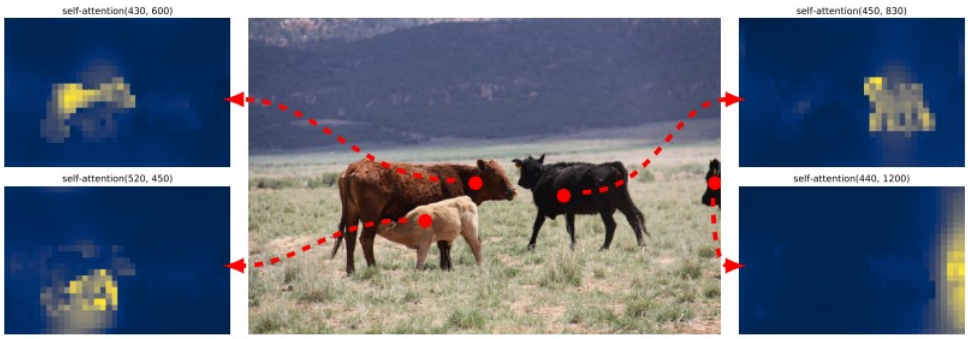

Figure 3.9: Encoder self attention [17]

3.4 RetinaNet Architecture

RetinaNet is a one-stage object detection network that states class imbalance during training by adding a modulating term to the Cross-Entropy(CE) loss. This puts more of an emphasis on learning hard negative examples. RetinaNet is a single, integrated network made up of two task-specific subnetworks plus a backbone network. A convolutional feature map over the input image is found using the backbone, which is a convolutional network in and of itself. On the output of the backbone, the first subnet applies object classification, while the second subnet applies bounding box regression. For one-stage, dense detection, using these two basic networks are recommended. The Below given figure gives a demonstration of RetinaNet Model Architecture.

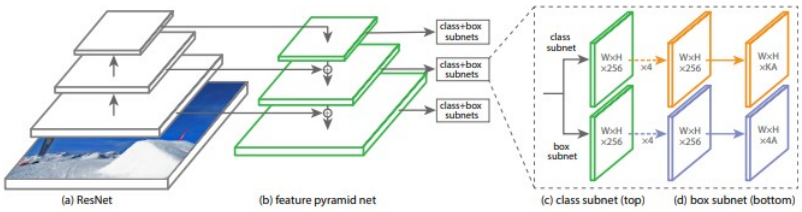

Figure 3.10: RetinaNet Model Architecture [9]

3.4.1 FocalLoss

Before discussing FocalLoss, we should know the concept of Calcification Entropy first.

$$\mathrm{CE}(p, y) = \begin{cases} -\log(p) & \text{if } y = 1 \\ -\log(1-p) & \text{otherwise} \end{cases} \tag{3.4}$$

Calcification Entropy(CE) loss for binary calcification example is given in Above equation y $\in \{\pm 1\}$ indicates the ground-truth class and $p \in [0, 1]$ is the model's estimated probability for the class with the label y = 1.

$$p_{\mathrm{t}} = \begin{cases} p & \text{if } y = 1 \\ 1-p & \text{otherwise,} \end{cases} \quad \mathrm{CE}(p, y) = \mathrm{CE}\left(p_{\mathrm{t}}\right) = -\log\left(p_{\mathrm{t}}\right). \tag{3.5}$$

For convenience, CE is defined as p_t
The CE loss can be seen as the blue curve in the figure. In this figure, we can see that the hard examples with even the smallest increase can drastically increase the amount of loss. Combining all the easy examples can easily overwhelm the rare classes with its sheer amount alone.

$$\mathrm{CE}\left(p_{\mathrm{t}}\right) = -\alpha_{\mathrm{t}} \log\left(p_{\mathrm{t}}\right) \tag{3.6}$$

α : **offset class balance** Introducing a weighting factor $\alpha \in [0, 1]$ in the equation above for class 1 and $1 - \alpha$ for class 1 can solve this issue. α can be used as a hyper-parameter to be set by cross-validation or set by inverse class frequency.

$$\mathrm{FL}\left(p_{\mathrm{t}}\right) = -\left(1 - p_{\mathrm{t}}\right)^{\gamma} \log\left(p_{\mathrm{t}}\right) \tag{3.7}$$

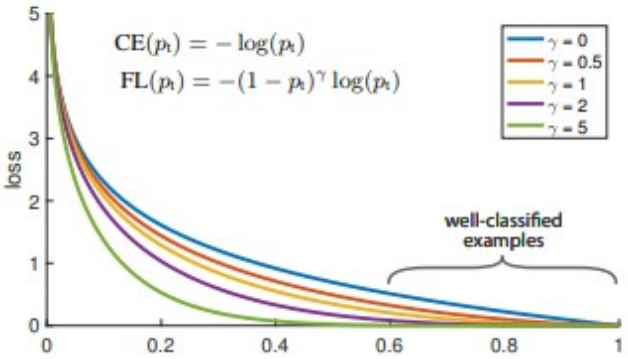

Figure 3.11: Probability of ground truth class [9]

γ : **Focus more on hard examples:** alpha balances the importance of positive/negative examples. And the purpose of using FL is to ignore easy examples and focus more on hard examples. By importing a modulating factor $(1 - \mathrm{p_n t})$^Y to the CE loss, with tunable focusing parameter$\gamma \geq 0$, we get FL mentioned in the equation above. There are two factors of FL:

1. When an example is misclassified and pt is small, the modulating factor is near 1 and the loss is unaffected. As pt $\rightarrow$1, the factor goes to 0 and the loss for well-classified examples is down-weighted.

2. The focusing parameter γ smoothly adjusts the rate at which easy examples are down-weighted. When $\gamma = 0$, FL is equivalent to CE. When γ is increased, the effect of the modulating factor is likewise increased. ($\gamma = 2$ works best in the experiment.)

$$\mathrm{FL}\left(p_{\mathrm{t}}\right) = -\alpha_{\mathrm{t}}\left(1 - p_{\mathrm{t}}\right)^{\gamma} \log\left(p_{\mathrm{t}}\right) \tag{3.8}$$

Adding alpha into the equation will further increase the accuracy.

In the model initialization stage, the initial value π is added to p which in turn will reduce the value of model estimated p of the rare class by a significant amount. Not using a predetermined value of ϕcan lead to training failure while using RetinaNet.

3.4.2 Backbone

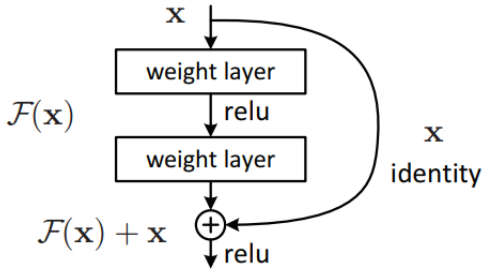

Figure 3.12: ResNet skip connection [4]

ResNet [4] is used for deep feature extraction and a rich multi-scale feature pyramid is built from a single resolution input image using the Feature Pyramid Network (FPN) [8], which is applied on top of it. In ResNet[8], the Residual Blocks concept was developed by this design to address the issue of the vanishing/exploding gradient. We employ a method called skip connections in this network. Figure[3.12] showcases the working procedure of skip connection. FPN [8] applied here is different compared to normal FPN [8]. A P3-P7 pyramid is generated here. Here P2 is excluded, strided convolution is used to calculate P6 rather than downsampling, and P7 is included to improve accuracy. Different anchors of different sizes and aspect ratios are added at different levels of the pyramid. Anchors are added to ground-truth object boxes using IOU threshold of 0.5 and to the background if IOU is in [0, 0.4). A maximum of one object box is allocated to each anchor, and the corresponding class entry in the K one-hot vector is set to one while all other elements are set to 0. The anchor is unassigned and ignored during training if IOU is between [0.4,0.5). When there is no assignment, the difference between the anchor and assigned object box is used to compute the box regression. The possibility that each of the K object classes and A anchors will be present at a specific spatial position is determined by the Classification Subnet. This subnet is a small FCN attached to every FPN[8] level. It starts with an input feature map with C channels from a specific pyramid level. The subnet applies four 3×3 Conv layers, each with C filters and each followed by ReLU activations, followed by a 3×3 Conv layer with KA filters. Inference includes forwarding images through the network. Decoding box predictions from max 1k top-scoring predictions per FPN [8] level after thresholding detector confidence at 0.05 increases the speed significantly. Merging top predictions from every level and applying non-maximum suppression with a threshold of 0.5 produces the final result.

Chapter 4

Methodology

In this section of our research, we have tried to show how we have processed our dataset and augmented them, so that it becomes easy for our models to train on them. We have also shown the hardware we are using and how we are training the model in our dataset. Furthermore, the hyperparameters we have changed and the evaluation of each model is also shown in this section.

4.1 Data Preprocessing

Before training the data need to be prepared and have to be suitable in order to experiment. We tried to process the dataset and augment the whole dataset so as to eradicate discrimination from the classes of the dataset. Before training the data need to be prepared and have to be suitable for the model to experiment.

4.1.1 Data Preparation

We started our research with the open-source dataset IP102 [31]. The dataset has approximately 19k annotated images with 102 classes of insects. However, after analyzing the data, we soon found out that the dataset was severely imbalanced and unsuitable for going forward with our experiments. In order to create a fairly balanced dataset, we discarded most classes and images and only kept the classes that had similar and fairly large sample sizes. The new dataset now had 13 classes of insects and around 8k images. We named the new dataset 'IP013'. The below table shows the sample size of IP013 for each class:

Class	Sample Size
Blister beetle	935
Cicadellidae	930
Aphids	876
Miridae	865
Mole cricket	863
Locustoidea	533
Legume blister beetle	435
Grub	434
Wireworm	425
Corn borer	424
Prodenia litura	413
Beet army worm	413
Flax budworm	410

Table 4.1: Sample Distribution of IP013 dataset

4.1.2 Data Augmentation

The IP013 dataset has images of insects taken in regular lighting conditions. To serve our research purpose and train the neural networks on low-light images, we augmented the images in a number of ways. We applied a random brightness augmentation ranging from 0% to 67% of the original image. Moreover, we performed random 90 degrees rotations on both the images and the annotations to add more variety to the dataset. After the augmentations, the dataset now has approximately 13.4k images.

4.2 Hardware and Software

To conduct our experiment we used the Google Colab research environment with an Nvidia Tesla T4 GPU that is capable of 65 FP16 TeraFLOPS. Even though it's not the latest machine learning hardware, it performs similarly to the latest consumer hardware available nowadays. Moreover, Google Colab provided us with a stable and easy-to-use environment to train and validate our chosen models. For the machine learning framework, we used Pytorch and more specifically Torchvision package. However, for the RetinaNet implementation, the Keras Tensorflow framework was required. All of the frameworks worked flawlessly with Python 3.9. Additionally, we used Nvidia CUDA API to perform the majority of the computation on GPUs. Furthermore, the 'pycocotools' package was utilized which is the official python API for the MS-COCO dataset to perform COCO evaluation.

4.3 Training and validation

The dataset was roughly split into 80% for training and 20% for validation. Since the IP013 dataset is smaller in size compared to IP102 we decided to put most images

on the training set to provide adequate data to train all of the models. The models we used in this research have many implementations due to their state-of-the-art status. However, we mostly tried to use the ones that are considered 'official' because they were implemented by the researchers who invented the respective models. We trained each of the models on our custom dataset IP013 for 50 epochs and manually fine-tuned some of the many hyperparameters to get better results and reduce the chance of overfitting and used default parameter values for the rest. Below the details of training for each model are further explained.

EfficientDet: The training was completed using this [32] implementation based on the Pytorch framework. This particular implementation uses COCO annotation format. So the original PASCAL VOC annotations needed to be converted into COCO format in order to be used in this implementation. The 'EfficientDet-D0' pretrained weight was loaded into the model for transfer learning. The weight was pretrained with COCO 2017 dataset and had EfficientNet backbone. Since the weight is trained from scratch, we can fine-tune the model on our custom dataset much faster as most of the time of the training won't have to be spent on training the backbone to extract features better. As for the hyperparameters that were changed during the training process were learning rate, batch size, and optimizer function. The learning rate was set to 1e-3, batch size to 16 and "AdamW" optimizer was used for all epochs except for the last one where 'SGD' optimizer was used.

RetinaNet: This [29] is used to complete the training. The implementation is based on the Tensorflow framework and it supports COCO annotation format. Again a pretrained weight 'resnet50_coco_best_v2.1.0.h5' was used for transfer learning. As the name suggests it was trained on COCO 2017 dataset with a resnet50 backbone. The hyperparameters that were changed are the following: learning rate was set to 1e-5, batch size set to 2, steps to 5557 and 'Adam' optimizer was used.

DETR: The implementation [28] is based on PyTorch framework and supports COCO annotation format. The pretrained weight 'DETR-R50' was used which was trained with COCO 2017 dataset and had a resnet50 backbone. The changed hyperparameters are the following: learning rate was set to 1e-4, batch size set to 4, object query to 10, weights decay to 1e-4, and 'AdamW' optimizer was used.

YOLOv7: The training was done on this [30] which is based on the PyTorch framework. The implementation used its own YOLOv7 annotation format. The pretrained weight 'YOLOv7' was used and it trained on the COCO 2017 dataset. As for the hyperparameters: initial learning rate was set to 0.01, final learning rate was set to 0.1, batch size was set to 8 and 'AdamW' optimizer was used.

Chapter 5

Results

In order to evaluate the results of the different object detection models that we have used for this research, we have chosen the coco metrics. It is very popular among the object detection models to compare themselves with this evaluation metrics. We have trained our models on our custom dataset for 50 epochs to evaluate and compare the results. The models we have trained are mentioned above. From the table below we can see the performance of the trained models.'

Model	Dataset	Backbone	$AP_{50:95}$	AP_{50}	AP_{75}	Params (M)	FLOPs (G)	Training Time Per Epoch (min)
YOLOv7	COCO	ELAN	51.4	69.7	55.9	36.9	104.7	12
	Ours		45.9	79.8	-			
EfficientDet	COCO	EfficientNet	33.1	51.2	34.8	3.9	2.5	22
	Ours		33.2	61.8	31.3			
RetinaNet	COCO	ResNet-50	35	53.7	37.4	34	97	90
	Ours		35.9	66	34.4			
DETR	COCO	ResNet-50	42	62.4	44.2	41	86	30
	Ours		35.2	60.9	36.5			

Table 5.1: Comparison between models trained on our dataset and COCO

In the table 5.1,we can clearly see that YOLOv7 is crushing other models in terms of Average Precision and Training Time. Particularly, model performance is influenced by both training environments and architecture of the model. We have tried to achieve the best result from each model by hypertuning and configuring. In the main, even though EfficientDet is meant to be efficient with its smaller number of parameters and less FLOPs, it takes more time to train the dataset than YOLOv7. YOLOv7 is using a new technique which they named as planned reparameterization which helps them to enhance precision without actually incrementing training time. In the lowest time, the most recent model YOLOv7 is outperforming the other models by a lot. On the other hand the oldest model in the comparison is doing better in terms of precision than other models. RetinaNet which came two year earlier than EfficientDet is getting a slightly better result but at the cost of training time which is very inefficient. During the time Retinanet could complete one epoch, EfficientDet could have completed more than 4 epochs in that time. Considering the training time, EfficientDet could have performed way better than RetinaNet. The same can also be said for DETR which is a transformer based model. The full potential of the transformer based model is locked with epoch training, which is why the result is not that impressive yet very close to RetinaNet. Due to lack of

our computational capability, it was not possible to show the true potential of the DETR and EfficientNet. We only trained the base model of EfficientNet which is D0 which has lower AP, but instead if we had the computational power to implement the EfficientDet-D7x the result would have been a lot better. Yet the results that we got from only 50 epochs is pretty impressive given the state of the dataset. One of the reasons for this to happen is because we have used the pretrained weights which means we have taken the help of transfer learning. Since we did not have the computational power to train the model from scratch and also the custom dataset we had was pretty small for learning from scratch, we had no other way but to choose this path.

After comparing the models on our own dataset, we will now compare the results we got from the test on IP013 with the results the authors got from training on the COCO dataset from scratch. It is necessary to mention that we have used the approach of transfer learning for training the models on our own dataset since it was quite impossible for us to train from scratch with the processing capacity we have, which is why we used pretrained weights. Now if we notice the table 5.1 we can see that the models are performing quite well on our dataset in contrast to the COCO dataset. The results on COCO are taken from [16],[18],[9] and [17]. EfficientDet got a mAP@0.5:0.95 of 33.2% on our dataset contrary to 33.1% on the COCO dataset. RetinaNet is also performing very close on COCO in contrast to our dataset where the average precision on COCO is 35%, it is 35.9% in case of our dataset. However the recent models aren't as close as these older models were. The YOLOv7 is 45.9% on our dataset whereas it is getting 51.4% mAP on COCO. The same can be said for DETR. This may be because the models may not be properly hypertuned and need more training to fit the dataset in the model. Even though YOLOv7 wasn't performing as it should be, the results are much better than the other models. From our understanding, if we had the computational capacity we could show the full potential of YOLOv7 in training a low illuminated dataset without using any image enhancement network.

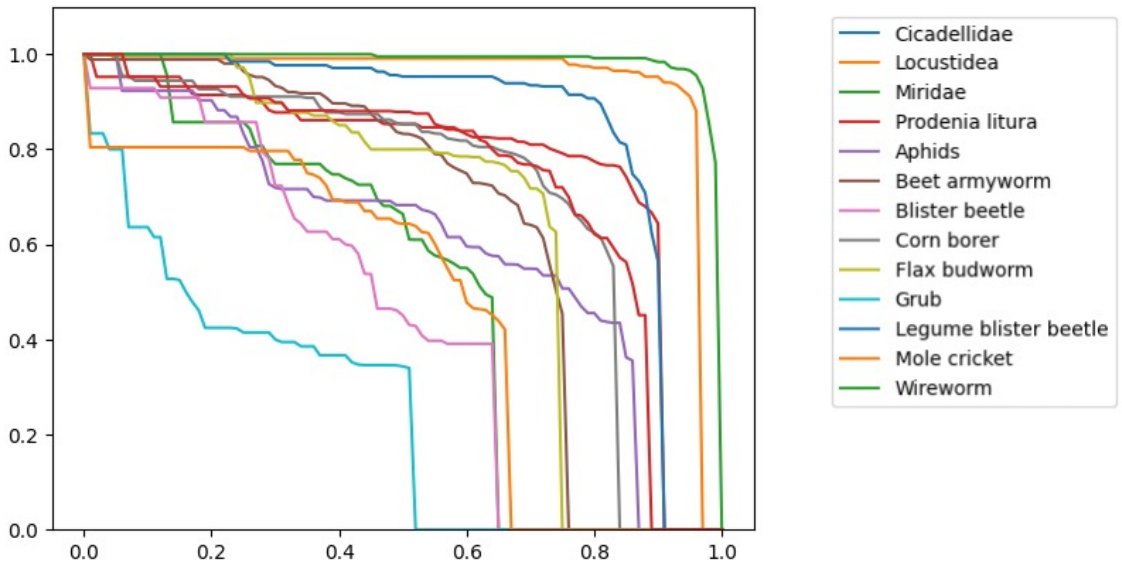

Figure 5.1: PR curve of RetinaNet on our dataset

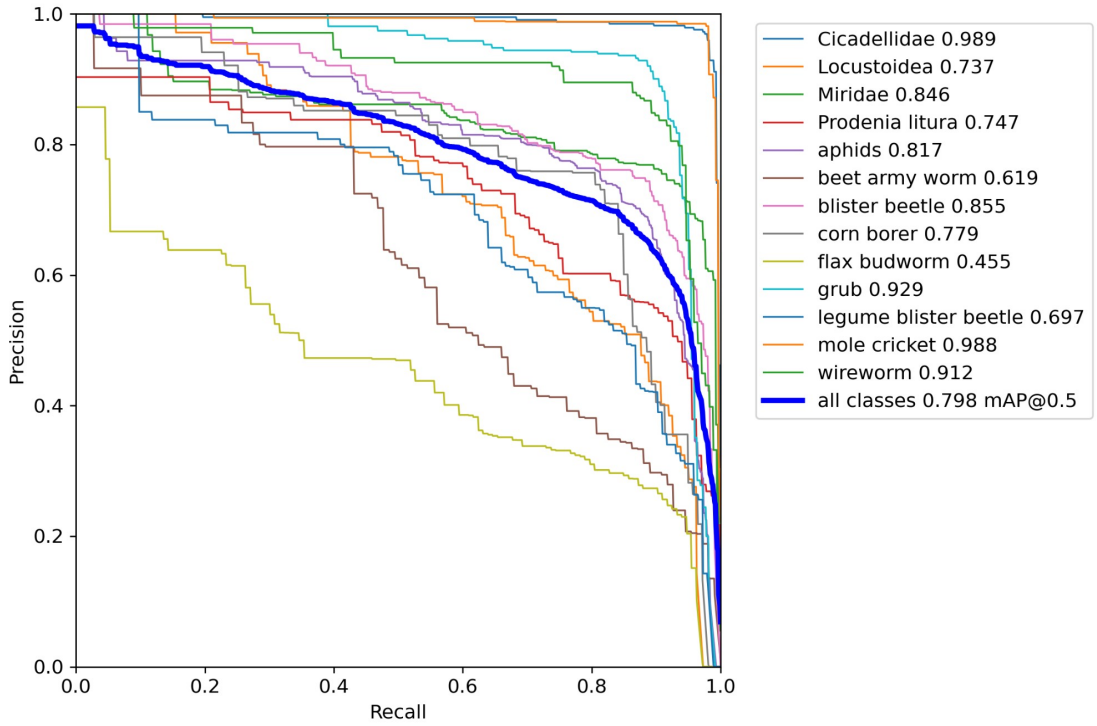

Figure 5.2: PR curve of YOLOv7 on our dataset

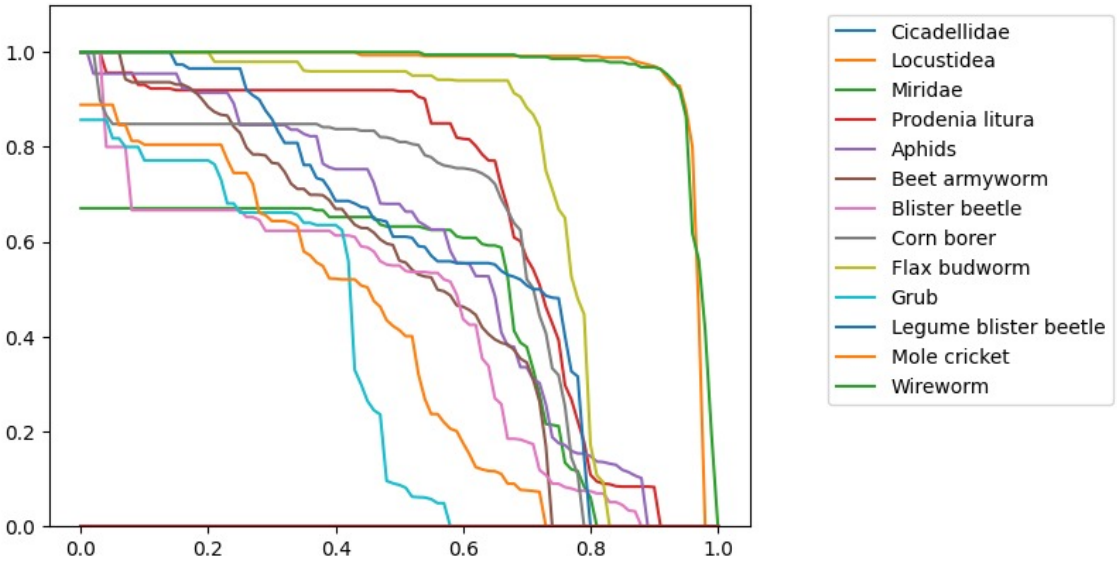

Figure 5.3: PR curve of DETR on our dataset

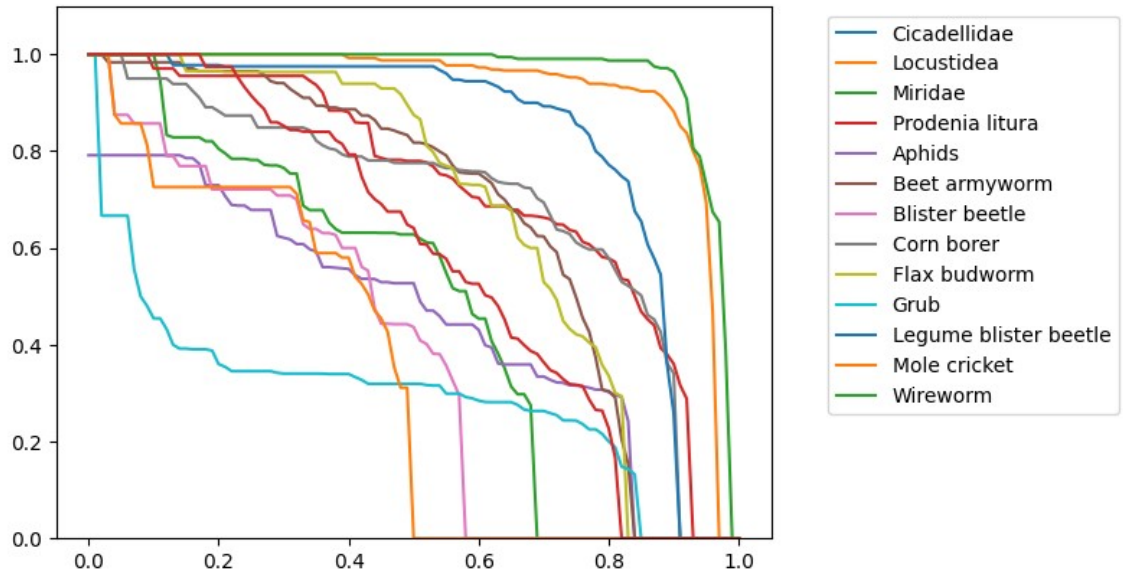

Figure 5.4: PR curve of EfficientDet on our dataset

In the above figure, we can see the Precision-Recall curve and result of YOLOv7, EfficientDet, DETR and RetinaNet in a pictorial form. We are emphasizing on the YOLOv7 as a representative of the recent models and the EfficientDet as a representative of the old models since the results from the YOLOv7 are very impressive and the EfficientDet was supposed to be efficient and used the least computational power of all. From the above figures we can see that YOLOv7 is performing decently and performance of EfficientDet is not as good. The YOLOv7 and EfficientDet have very accurately detected Cicadellidae, aphids, grubs, mole cricket and wireworm.

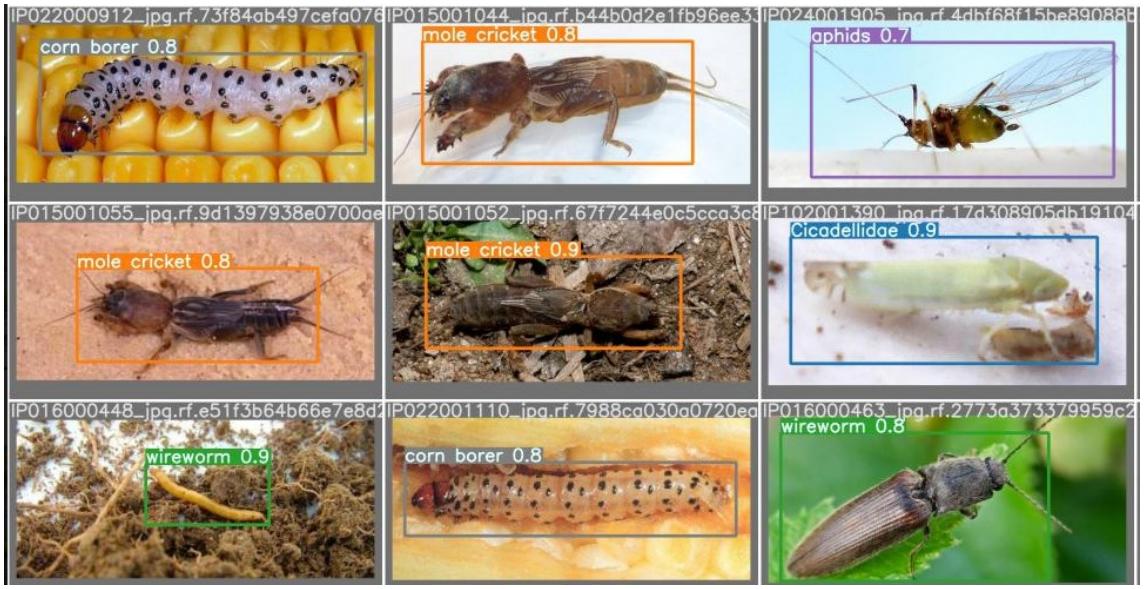

Figure 5.5: Test batch prediction

This happened because those classes have higher numbers in terms of object count. So, the model has precisely detected the objects from those classes. The true negative for those classes are lower in number. There are some classes which were

underprivileged because the number of objects was less in those classes. So, those classes did not perform as well as the classes like mole cricket and cicada.

From the results image, we can see that mAP(mean Average Precision) for IoU (Intersection over Union) of 0.5 is almost 0.8. This means that for each class the model has on average predicted the object 80% at IoU 0.5. These metrics are described in the appendix portion. The mAP for each and every class at 0.5:0.95 is on average as high as 45.9%. From the Precision-Recall curve, we can see that mAP for Cicadellidae is 98.9% which is the highest from all the classes and the lowest is 45.5% for class flax budworm. For EfficientDet the mAP for all the classes at 0.IoU 0.5 is 61.8% as shown from the above table 4.2. For DETR it is 60.9% and RetinaNet is 66% at IoU of 0.5.

Chapter 6

Conclusion

Even though research on object detection has come so far efficient low-light object detection more specifically remains a challenging task. Throughout our research, we studied the works of other researchers and came to an understanding that soon there will be not much of a need for an additional image enhancement network to detect objects in low light images and it will be possible to implement on low-end systems. To recap, we collected and created a low-light dataset of insect images called IP013. We chose old and new state-of-the-art models to train and validate on our dataset. Moreover, we collected the original research data that showed each model's performance on a regular image dataset called COCO. We compared each of the models with one another by measuring their performance using MS COCO evaluation metrics while taking into consideration their performance on regular images. Through our work in this project, we have come to an understanding that the current scene of deep learning models is improving rapidly in a very short period of time. The models we used (both old and new) are capable of giving inferences in real-time with correct lighting conditions. On the other hand, we have focused on detecting objects in low lighting conditions. Still, our models had given us results that are similar to those results achieved through training well-lit images. Even though we had faced challenges like an imbalanced dataset, hardware constraints, and limited time frame, we have achieved good results which led us to hope that we can implement our work in embedded systems and test it in real-life situations.

Chapter 7

Appendix

7.1 COCO Evaluation Metrics

Coco detection challenge uses 12 different metrics to evaluate the accuracy of object detection of different algorithms. The main goal is to compute the Mean Average Precision(mAP) to evaluate the algorithms. Google Open Images Dataset V4 Competition also uses mAP on 500 different classes to discover the most efficient algorithm. We have used these evaluation metrics on our custom dataset to evaluate the models that were used on our custom dataset. Before discussing mAP, The concept of IOU and several concepts have to be discussed first.

$$\text{IOU} = \frac{\text{area}\,(B_p \cap B_{gt})}{\text{area}\,(B_p \cup B_{gt})} \tag{7.1}$$

While creating bounding boxes around an object in an image and predicting the class of the object in that bounding box, a metric is used called Intersection Over Union (IOU) shown in equation(7.1). IOU is used to evaluate the overlap between two bounding boxes. By using a Ground truth bounding box and a Predicted bounding box, the validity of the detection is confirmed by calculating the overlapping area between the Predicted bounding box and the Ground truth bounding box by the area between them.

True Positive(TP) : A correct detection. Detection with IOU $\geq$ threshold

False Positive (FP): A wrong detection. Detection with IOU $<$ threshold

False Negative(FN): A ground truth not detected

True Negative(TN): corrected misdetection. During object detection, there are a lot of bounding boxes that are supposed to be not detected. TN indicates the correctly not detected bounding boxes. An example of all detection types is given below in figure(6.1).

$$\text{Precision } = \frac{TP}{TP + FP} = \frac{TP}{\text{all detections}} \tag{7.2}$$

The ratio of correctly identified Positive samples to all Positive samples overall is known as precision. To put it simply, Precision answers the question of how accurate

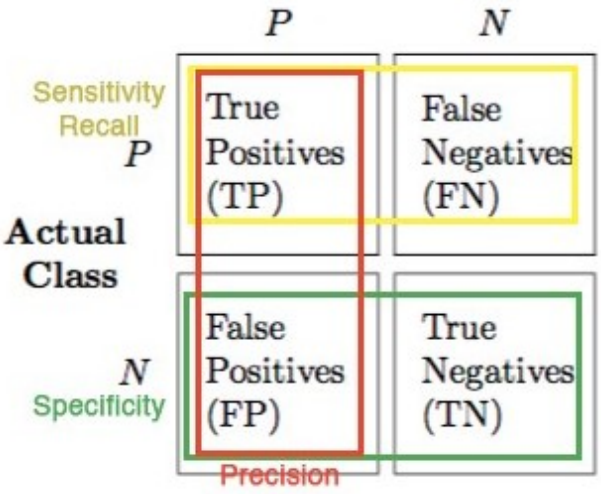

Figure 7.1: Confusion Matrix

the model's assumptions were. Precision would be high if the model correctly classified the majority of the Positive samples as Positive or classified fewer (incorrect) Negative samples as Positive, while precision would be low if the model classified many (incorrect) Negative samples as Positive or correctly classified fewer Positive samples. The equation of precision is given in equation (7.2)

$$\text{Recall } = \frac{TP}{TP + FN} = \frac{TP}{\text{all ground truths}} \tag{7.3}$$

The proportion of accurately categorized Positive samples to all actual Positive samples is known as Recall. Recall determines whether your model made a guess each time one was expected of it. More positive samples are detected with higher recall. The equation of recall is given in equation(7.3).

The Precision x Recall curve is a good way to evaluate the performance of an object detector as the confidence changes. As each object class has a curve, an object detector with high prediction and increasing recall of each class is considered good which will stay the same even if the confidence threshold changes. In the above equations of Precision and Recall, TP+FN = all ground truth = constant. Here if TP increases, FN decreases and the precision will remain high. Usually, the Precision x Recall curve starts with high precision values, decreasing as recall increases. Average Precision(AP) is calculated using the area under the curve (AUC) of the Precision x Recall curve. Normally AP is the precision averaged across all recall values between 0 and 1.

$$\text{mAP} = \frac{1}{|\ \text{classes}\ |} \sum_{c \in \text{ classes}} \frac{|TP_c|}{|FP_c| + |FP_c|} \tag{7.4}$$

Mean Average Precision(mAP) is calculated by taking the mean AP over all classes and/or overall IOU thresholds. For the evaluation of the models used on our dataset, the mAP was calculated by averaging the AP over all 13 object categories and all 10 IOU thresholds from 0.5 to 0.95 with a step size of 0.05. The IOU average helps the models with improved localization. In short, at first, AP is calculated for an IOU threshold of 0.5 for every single class. In this case, Compute the precision at every single recall value(0-1 with a step size of 0.01). Repeat this step for 0.55, 0.60, 0.65,

0.70,....., 0.95. After that, take the average over all 13 classes and 10 thresholds and compute the primary metric. The equation of Mean Average Precision(mAP) is given in figure(7.4).

Bibliography

[1] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[2] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, *Deeply-supervised nets*, Sep. 2014. [Online]. Available: https://arxiv.org/abs/1409.5185.

[3] T. Lin, M. Maire, S. J. Belongie, *et al.*, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014. arXiv: 1405.0312. [Online]. Available: http://arxiv.org/abs/1405.0312.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. arXiv: 1512.03385. [Online]. Available: http://arxiv.org/abs/1512.03385.

[5] O. Russakovsky, J. Deng, H. Su, *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[6] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, Apr. 2015. [Online]. Available: https://arxiv.org/abs/1409.1556.

[7] G. Huang, Z. Liu, and K. Q. Weinberger, "Densely connected convolutional networks," *CoRR*, vol. abs/1608.06993, 2016. arXiv: 1608.06993. [Online]. Available: http://arxiv.org/abs/1608.06993.

[8] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. DOI: 10.1109/cvpr.2017.106.

[9] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *CoRR*, vol. abs/1708.02002, 2017. arXiv: 1708.02002. [Online]. Available: http://arxiv.org/abs/1708.02002.

[10] S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. DOI: 10.1109/cvpr.2017.634.

[11] Y. P. Loh and C. S. Chan, "Getting to know low-light images with the exclusively dark dataset," *CoRR*, vol. abs/1805.11227, 2018.

[12] N. Parmar, A. Vaswani, J. Uszkoreit, *et al.*, "Image transformer," in *International conference on machine learning*, PMLR, 2018, pp. 4055–4064.

[13] I. Bello, B. Zoph, A. Vaswani, J. Shlens, and Q. V. Le, "Attention augmented convolutional networks," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 3286–3295.

[14] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 4780–4789, 2019. DOI: 10.1609/aaai.v33i01.33014780.

[15] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," *CoRR*, vol. abs/1905.11946, 2019. arXiv: 1905.11946. [Online]. Available: http://arxiv.org/abs/1905.11946.

[16] A. Bochkovskiy, C. Wang, and H. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *CoRR*, vol. abs/2004.10934, 2020. arXiv: 2004.10934. [Online]. Available: https://arxiv.org/abs/2004.10934.

[17] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European conference on computer vision*, Springer, 2020, pp. 213–229.

[18] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. DOI: 10.1109/cvpr42600.2020.01079.

[19] Q. Tao, K. Ren, B. Feng, and X. GAO, *An accurate low-light object detection method based on pyramid networks*, Oct. 2020. [Online]. Available: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/11550/1155015/An-accurate-low-light-object-detection-method-based-on-pyramid/10.1117/12.2573925.full?SSO=1.

[20] C. Wang, A. Bochkovskiy, and H. M. Liao, "Scaled-yolov4: Scaling cross stage partial network," *CoRR*, vol. abs/2011.08036, 2020. arXiv: 2011.08036. [Online]. Available: https://arxiv.org/abs/2011.08036.

[21] Y. Xiao, A. Jiang, J. Ye, and M.-W. Wang, "Making of night vision: Object detection under low-illumination," *IEEE Access*, vol. 8, pp. 123 075–123 086, 2020. DOI: 10.1109/ACCESS.2020.3007610.

[22] X. Ding, X. Zhang, N. Ma, J. Han, G. Ding, and J. Sun, "Repvgg: Making vgg-style convnets great again," *CoRR*, vol. abs/2101.03697, 2021. arXiv: 2101.03697. [Online]. Available: https://arxiv.org/abs/2101.03697.

[23] P. Dollár, M. Singh, and R. B. Girshick, "Fast and accurate model scaling," *CoRR*, vol. abs/2103.06877, 2021. arXiv: 2103.06877. [Online]. Available: https://arxiv.org/abs/2103.06877.

[24] C. Wang, I. Yeh, and H. M. Liao, "You only learn one representation: Unified network for multiple tasks," *CoRR*, vol. abs/2105.04206, 2021. arXiv: 2105.04206. [Online]. Available: https://arxiv.org/abs/2105.04206.

[25] J. Yu, X. Hao, and P. He, "Single-stage face detection under extremely low-light conditions," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 3523–3532.

[26] anonymous, *Designing network design strategies*, 2022.

[27] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, *Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors*, Jul. 2022. [Online]. Available: https://arxiv.org/abs/2207.02696.

[28] Facebookresearch, *Facebookresearch/detr: End-to-end object detection with transformers*. [Online]. Available: https://github.com/facebookresearch/detr.

[29] Fizyr, *Fizyr/keras-retinanet: Keras implementation of retinanet object detection.* [Online]. Available: https://github.com/fizyr/keras-retinanet.

[30] WongKinYiu, *Wongkinyiu/yolov7: Implementation of paper - yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors*. [Online]. Available: https://github.com/WongKinYiu/yolov7.

[31] xpwu95, *Xpwu95/ip102: Ip102: A large-scale benchmark dataset for insect pest recognition*. [Online]. Available: https://github.com/xpwu95/IP102.

[32] zylo117, *Zylo117/yet-another-efficientdet-pytorch: The pytorch re-implement of the official efficientdet with sota performance in real time and pretrained weights.* [Online]. Available: https://github.com/zylo117/Yet-Another-EfficientDet-Pytorch.