THE UNIVERSITY OF ADELAIDE

MASTER'S THESIS

Object Counting and Tracking

Author:
Harsh Alpesh BHATT
ID: a1872953

Supervisor:
Dr. Jinan Zou

A thesis submitted in fulfillment of the requirements for the course Research Project Part A and B for the degree of Master's of Computer Science.

June 7, 2024

Declaration of Authorship

I, Harsh Alpesh BHATT, declare that this thesis titled, "Object Counting and Tracking " and the work presented in it are my own. I confirm that:

- This work was done fully and mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: Harsh Alpesh Bhatt

Date: June 7, 2024

"Let the future tell the truth and evaluate each one according to his work and accomplishments. The present is theirs; the future, for which I have really worked, is mine."

"Life is and will ever remain an equation incapable of solution, but it contains certain known factors."

Nikola Tesla

THE UNIVERSITY OF ADELAIDE

Abstract

Computer Science
School of Computer and Mathematical Sciences

Master's of Computer Science

Object Counting and Tracking

by Harsh Alpesh BHATT

Object counting and tracking is s computer algorithm where we try to detect and count the total number of objects in a given image or a video file accurately and automatically. With advancements in technology, there has been a rise in the number of object counting algorithms and technique. With recent increase in interest in the field of Machine Learning and Artificial Intelligence, object counting and tracking research has also shown remarkable growth with countless authors publishing their works.

With the introduction of Segment Anything Model(SAM) by META last year, a new branch in object counting space has grown. In this task, we use this newly introduced Segment Anything Model and perform the counting task. Segment Anything Model by META is an image segmentation model that can perform high quality masks from any given set of prompts (inputs) using promptable segmentation. These prompts can range from a simple point to boxes and can be used to mask specific objects in an image file. In this novel approach, we combine Segment Anything Model with a detection layer of YOLO(You Only Look Once) to achieve greater results with more accuracy and prediction. YOLO is a real time is a single stage object detection algorithm introduced in 2015 to perform object detection tasks with more accuracy, more precision and also be time efficient. The aim of this research is to find the least number of images required using YOLO trained model to get better results than previous works.

In this report we will also see the comparison of different approaches taken by various authors and through this analysis, find the effectiveness of the novel approach. Overall, this study will shed lights on the previous works by authors in similar fields and trying to solve similar tasks. It will also show how this novel approach to the problem can get more accurate results than the previous approaches.

Acknowledgements

I would like to give special thanks to Dr. Jinan Zou and Yuhao Lin for their firm support, guidance and patience during the course of this research which helped towards achieve the desired results.

Contents

List of Figures

List of Abbreviations

SAM	Segment Anything Model
YOLO	You Only Look Once
GPU	Graphical Processing Unit
HOG	Histogram of Oriented Gradients
CNN	Convolutional Neural Network
SA-1B	Segment Anything 1 Billion
ViT	Vision Transformer
MAE	Masked Autoencoders
MLP	Multilayer Perceptron
BMNet	Bilinear Matching Network
FPN	Feature Pyramid Network
NMS	Non-Maximum Suppression
IoU	Intersection Over Union
RMSE	Root Mean Squared Error
MAE	Mean Absolute Error
PR	Precision-Recall
FSOD	Few Shot Object Dectection
AI	Artificial Intelligence

Dedicated to my lovely parents, grandparents and sister…

Chapter 1

Introduction

1.1 Background

Humans can identify objects and distinguish between objects present in an image accurately and precisely. We can also perform complex tasks like differentiating these objects into different categories or classes and can perform these tasks with high accuracy and high speed. With the recent advancement in technology, with faster Graphics Processing Units (GPU), more training Data and faster and better algorithms, we perform objects detection and other tasks related to it using trained computers with accuracies similar to that of a human. Object detection is a common computer vision algorithm that is focused on locating and identifying specific types of objects in an image. These objects can be anything from humans, animals, or specific objects such as grain of rice, figuring out the different materials. These models can often be trained to detect specific objects and can figure out the objects in a video file, image file or real-time. Detecting these specific objects and other common objects can be done by creating an application that performs two major tasks, image segmentation and object recognition. Large amounts of image and video file data can be fed to deep learning computer vision based model to make it learn a specific type of objects. But object detection had a scope even before the existence of deep learning models and algorithms.

There existed algorithms like Histogram of Oriented Gradients(HOG) which were specifically created to perform object detection tasks. With the introduction of Convolutional Neural Networks(CNN) (Redmon et al., 2016)and other modern Machine Learning and Deep Learning algorithms, object detection has become a much more prominent task for current generation of computer algorithms. Face Detection and Recognition, Number Plate recognition, Object tracking, self-driving cars etc are just some of the few object detection applications that we use on our day to day basis. The use cases of Object detection can often be categorized into the following categories like locating the object, tracking the object, counting the object, and detecting anomalies and outliers in the given scenario.

The task of estimating the number of objects in a given image is called as object counting. There are two major ways object counting can be achieved. First is using Image segmentation. In this computer vision technique, the image is divided into groups of pixels known as image segments. These image segments are then processed using various techniques and individual pixels are annotated to a specific class. Classical image segmentation models used the colours and intensity of these annotated image segments to determine the class of the objects, while the newer more advanced deep learning models does pattern recognition using complex neural networks. So, in this method we separate the background and the objects in the

image to count objects of specific class. The second major approach is the object detection approach. In this we use a machine learning model to detect objects and group them in classes and then perform the counting tasks on these grouped classes.

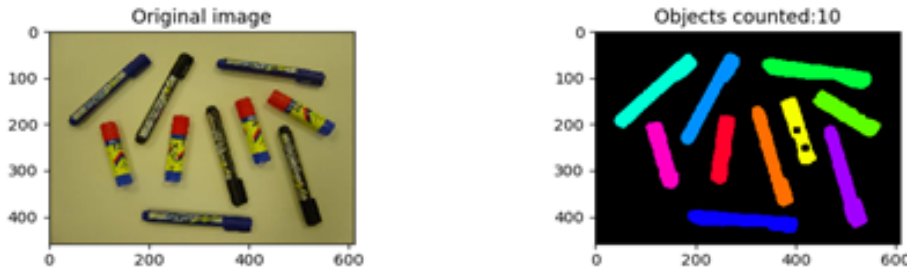


FIGURE 1.1: Image Segmentation Approach

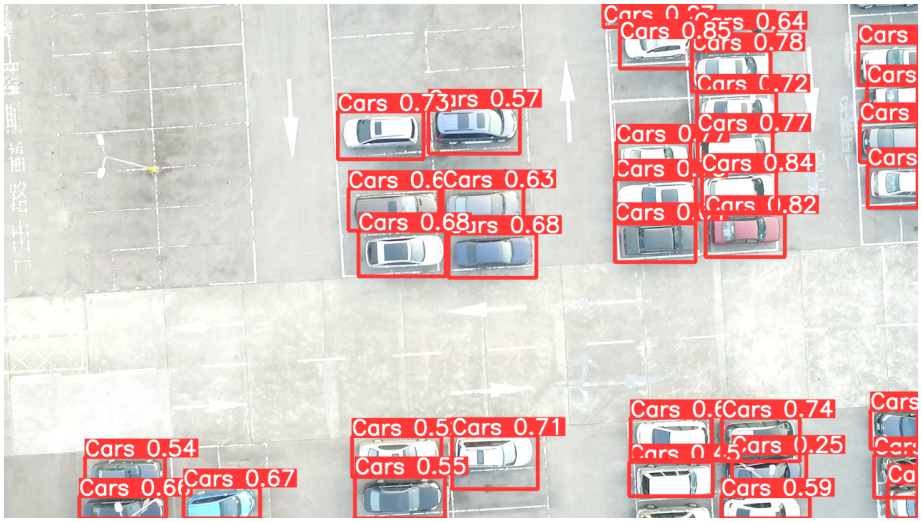


FIGURE 1.2: Object Detection Approach

Using these Object Counting algorithms along with various other machines like cameras and sensors there are many various applications of these algorithms. These applications can range in various industries from manufacturing, retail to logistics. It can also have applications in healthcare and smart city applications. Some of the task were object counting techniques are heavily used are inventory management, checking the quality of the products, monitoring traffic and it also has applications in automated production. The reason behind this popularity of these algorithms are the various features and benefits that these algorithms provide. They automate manual work and are more efficient and manages the time better. They improve the quality control and provide better decision-making. The other reason is that they are highly scalable and customizable. You can use these algorithms to track and count just about anything you wish for. They are not only more accurate but also can handle large volumes of data and can performs the operation quicker and with more efficiency.

Last year Meta AI released a new segmentation model which was trained on over 1 billion masks and 11 million licensed images. They named it the Segment Anything Model (SAM) (Kirillov et al., 2023). SAM has exceptional performance in various

types of segmentation tasks on image and scenes. This deep learning based segmentation model can perform lots of real world segmentation tasks. Using this model, users can generate masks of any objects in a given image by selecting individual objects or points. They can also include or omit certain objects. If the model faces any uncertainty, it generates multiple valid masks for the users to choose from. We can also perform the mask generation on any given objects based on prompts which can either be a bounding bx, a text prompt or any other kind of prompt. In this report we compare different counting techniques using SAM and ours counting task using SAM. Whether SAM is successful in performing counting tasks would generally depend on two aspects. First, would be whether SAM can segment each object and the second would be whether SAM can differentiate between the targeted object in the image and the reference images provided. We compare the works of various authors and their various mask generation technique using SAM to solve the counting task and our proposed idea to add a YOLO layer(Redmon et al., 2016) to SAM.

The current object detection algorithms can be divided into two primary categories: single stage detectors and two-stage detectors. Single stage detectors like YOLO and its versions use a single Convolutional Neural Network layer to directly locate and classify objects. Two stage detectors such as Faster R-CNN, use a fully complete Convolutional Neural Network layer to first extract features and then use the extracted features to locate and identify objects. Two stage detectors are usually more accurate than single stage detectors but are relatively slower. The study done by Demetriou et al., 2023 shows that apart from YOLO series, Faster R-CNN has a better accuracy than all the relative single stage detectors. YOLO series on the other hand, achieves even higher accuracy at a very high inference speed. YOLOv8 is the latest stable version of the YOLO series. Similar to its predecessors is a real time object detection system that performs bounding box prediction and class probabilities simultaneously with the help of a single neural network.

1.2 Motivation

Since the introduction of SAM, there have been constant study on figuring out the best way to perform the counting task using SAM. Various techniques from simple counting the object by crop box generation, or using class-agnostic counting, the main focus has been counting using SAM. Most of the previous studies have used segmentation approach for object detection since SAM is an image segmentation model. Its only natural to use SAM to its full extent. But SAM and this model has its limitations.

The limitation of using the segmentation approach is that it tends to segment congested objects into the same mask. This essentially means that same is unable to distinguish between objects that are very close to each other in an image. If it was a normal segmentation problem then it would be fine as it still would mean that SAM is correctly segmenting all the relevant objects but for a counting task where we want to count all the individual objects, this is a big problem. The below image from the FSC-147 dataset we can clearly see that for an image with lots of objects and all being congested into a small area, with objects overlapping, SAM masks these congested objects as one.

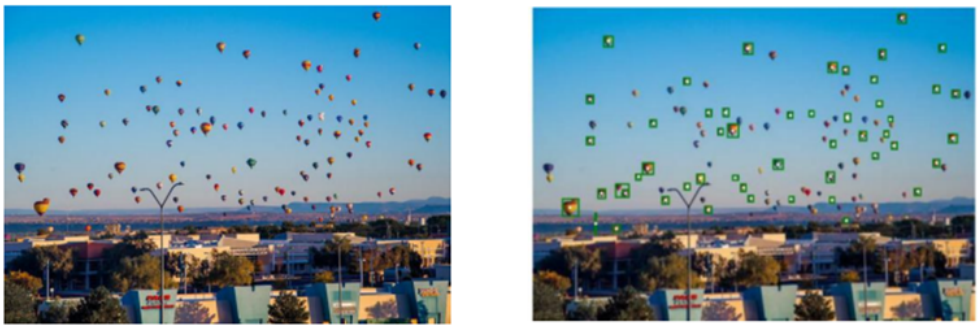


FIGURE 1.3: SAM failing to identify congested objects

The other major problem with using the segmentation approach is that it fails to distinguish between individual objects. The Segmentation approach works by removing the background from the image and then generating a segmentation map for all the similar objects. For general clearly differentiable objects, this approach can be relatively useful, but for counting problems, where there can be lots of objects of different classes in an image this can be problem. This approach would lead to lots of false markings and intern the results won't be as accurate. The below image also from FSC-147 dataset shows that how SAM model is unable to distinguish between strawberries and flowers that are in the same image. Shi et al., 2022 tried to solve this problem using class-agnostic counting, more on it later.

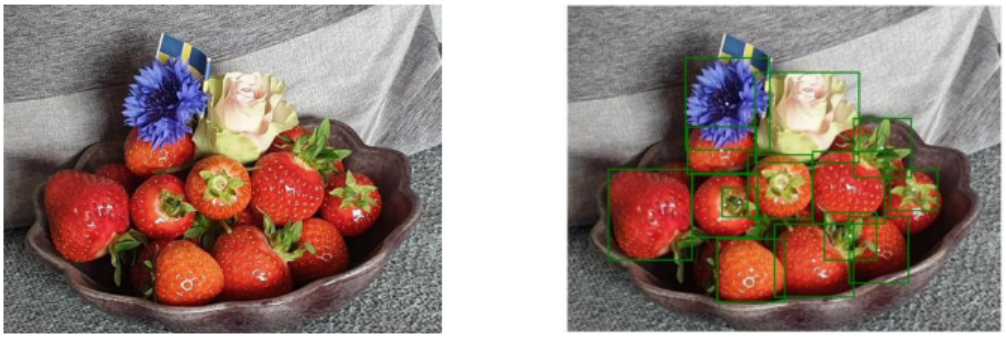


FIGURE 1.4: SAM failing to distinguish between objects

The last problem is the 'Noise' or false positives that can occur in similarity map generation step. This is the general false positives in every scenario that can occur, and it can occur due to a lot of reasons. It can occur due to bad lightning conditions in the image, low resolution of the targeted images and blurs etc. All these problems can affect the similarity map generation step and can lead to false positives or 'Noise'.

1.3 Objective

The main objective of this study is to try a different approach to the problem of using SAM for counting. The approach is using a combination of SAM and a single

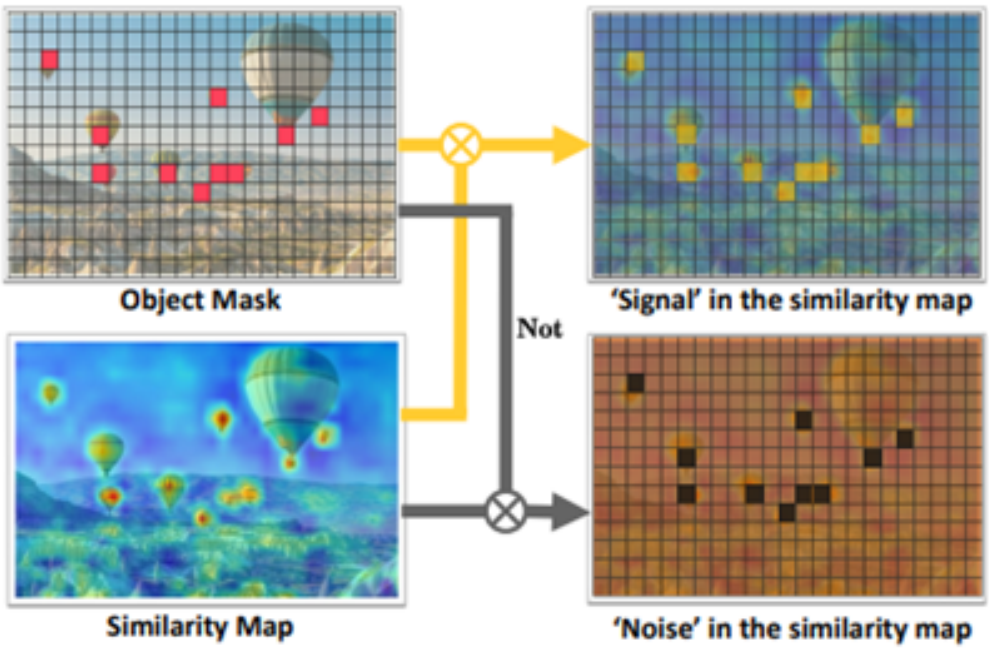


FIGURE 1.5: Noise in Similarity Map Generation step

stage detector YOLOv8 to perform the counting tasks. The use of YOLO and SAM is studied by a lot of scientific community to perform various computer vision tasks [Osco et al., 2023; Xu, Yan, and Ji, 2023; Qureshi et al., 2023]. The main reason behind this approach is to solve the two major problems. We aim that with a trained YOLO layer we would be able to identify tiny objects better and will also be able to clearly distinguish between objects. This YOLO layer is to be added before the SAM segmentation layer so that SAM can clearly distinguish between identified objects and improve the counting stage. The idea is to use the bounding boxes coordinates generated by YOLO to improve the SAM similarity map generation process and intern improve accuracy.

Now object counting using YOLO is not a new concept and there are countless papers and research are doing this or improving this. Combining a YOLO and SAM to perform counting is a novel approach. We know that YOLO does object detection better than any other object detection model. We also know from the studies done by Wu et al., 2021, where they try and perform the counting of vehicle and its class prediction, Ren et al., 2020 work on people counting in a real time environment and Mercaldo et al., 2022 work on counting the blood cells and its localisation, that YOLO is a very efficient and effective tool in counting a vast number of counting tasks and can also learn new classes with very limited training. With such a vast number of categories in FSC-147 dataset and the challenging drone view CarPK dataset, we took up the challenge to figure out the least amount of training images required to get significant results. The aim was to use as less images as possible for YOLO training and then perform the counting tasks using the combination of SAM and this newly trained YOLO layer to get better or comparable results to previous works on counting with SAM through segmentation approaches.

Chapter 2

Literature Review

The ability to segment images into tiny fragments and assigning them labels is known as image segmentation. SAM performs this task on pixel level determining the precise outline of an object in an image. These outlines are then grouped all in one colour or separate colours depending on the type of segmentation. The core idea behind developing the Segment Anything Model was to reduce the need for task specific modeling expertise and to train, compute custom data annotation. The goal was to build a foundation model based on image segmentation. They created a promptable model which is trained on diverse data and can adapt to specific tasks. However, the dataset required to train such a big model was not readily available and thus they simultaneously also developed a segmentation dataset of very large scale.

This dataset they developed was also developed using SAM. They did the annotation of images using 3 gears. In the first gear, the SAM model assists the annotators to interactively annotate images, in the second gear was a mixture of fully automatic annotation with assisted annotation leading to more diverse annotation. And finally, the third gear is fully automatic annotation of images. Each time the annotation process is completed, the annotated masks are added to the dataset for training SAM again. The final dataset includes more than 1.1 billion segmentation masks which were collected over 11 million licensed images. The SA-1B (Segment Anything 1 Billion) (Kirillov et al., 2023) dataset has around 400 times more segmentation masks than any other segmentation dataset.

There are two classes of approaches to any segmentation task. The first is the interactive segmentation. With this segmentation we can perform segmentation of any class, but it requires a person to guide the segmentation process by refining the mask generation manually. The second is the automatic segmentation, in this case we can segment specific categories predefined as the model is trained using manually annotated objects ahead of time. For example, a model can be trained on thousands or more annotated cat images and then it can automatically segment a new image. SAM is a model that can perform both these segmentation approaches easily and hence is the generalization of both these approaches. The promptable segmentation approach of SAM allows it to be more flexible and dynamic in its approach. The prompts can be clicks, boxes, texts and since SAM is trained on a very large, high quality, diverse dataset, it is capable to handle such segmentation tasks easily. SAM can automatically find and generate masks for all objects in a given image. It can also generate segmentation mask from any given prompt in real time after the image embeddings are precomputed making it model that can perform real time interactions. SAM can also generate multiple valid masks when it is faced with any uncertainty about the segmentation masks of any given objects.

FIGURE 2.1: Segmentation done by SAM

The main approach behind Segment Anything Model is performing Promptable Segmentation. The goal is to generate a valid segmentation mask based on any given prompt. The main goal was to generate valid segmentation masks from any given prompt which can be in the form of points, boxes, masks, text, or any other information that indicates the object to segment in an image, the model should generate masks for images even when the prompts are ambiguous and could refer to multiple objects in an image. The model uses a general pre-training algorithm and a more general method for zero-shot transfer to reduce the time required in segmentation tasks through prompting. Few shot learning algorithms are those algorithms which can make predictions to any given image based on very few examples of labelled data. Zero shot learning algorithms take this task even further as they are provided with no labelled data in new class and are required to make predictions based on the prior knowledge it has about the relationships between classes it already knows.

SAM has three basic components as seen from the above image. The components are an image encoder, a flexible prompt generator and a fast mask decoder. They build SAM on transformer vision model with specific trade-offs to meet the real time performance requirements. Vision Transformer (ViT) introduced in 2021 is a revolutionary neural network architecture that redefines how we process images. As shown by Dosovitskiy et al., 2020, ViT introduces a new way to process images. It does so by dividing the images into smaller section or patches and with the help of self-attention mechanisms they leverage it. With this approach the model is able to capture both local and global relationships of an image leading to more efficient performance in various computer vision tasks like the Segment Anything Model.

SAM uses the MAE(Masked Autoencoders) pre-trained Vision Transformer(ViT) (Dosovitskiy et al., 2020) which was slightly adapted to meet the requirements. They used this transformer for its scalability and very powerful pre-training methods. According to He et al., 2022, Masked Autoencoders (MAE) has a very simple approach were they mask random patches of the images and then reconstruct the missing pixels. In this two stage approach, the first step is encoder-decoder architecture. In this the encoder first operates on the visible subset patches with no mask tokens and then the decoder reconstructs the original image based on the latent representation and the mask tokens. The second step is masking of an image up to 75% instead of fully masking the image as they found that masking the image up to 75% would lead to

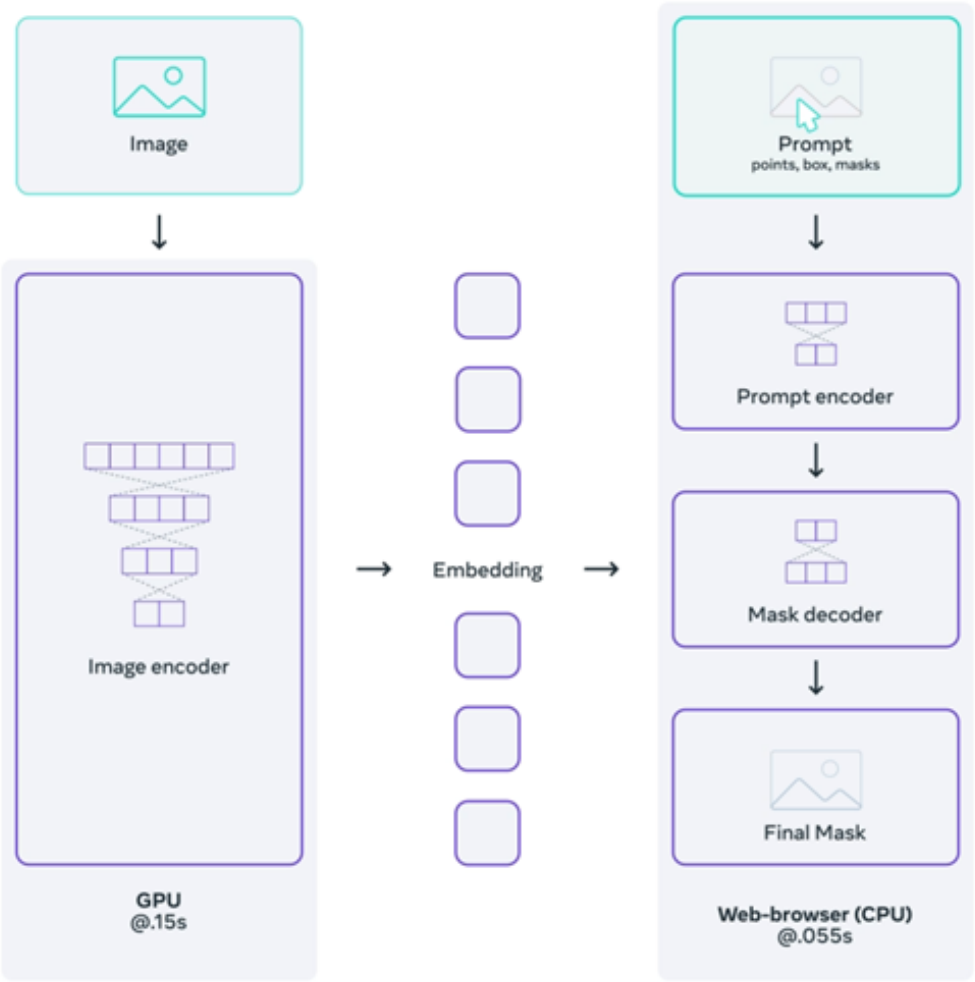


FIGURE 2.2: Simple Structure of Segment Anything Model

equal yields. Once these two steps are combined together we can improve the accuracy of the model and also reduce the training time by three times or more.

SAM mainly consider two sets of prompts: sparse and dense. The Sparse prompts are the points, texts and boxes which can be represented using positional encoding which are further combined with embeddings for each prompt type. For text prompts SAM uses a text encoder. For Dense prompts such as Masks, are combined using convolutions and then added with element wise image embeddings.

The mask decoder links the image, prompt embeddings and the output tokens to a mask. SAM uses a combination of a modified Transformer decoder block along with the dynamic mask prediction. The decoder block can perform both prompt self-attention and cross attention in both directions meaning it can generate a mask from the prompt for an image and it can also give prompt to an image masks. After the two blocks are run, the model then up-samples the image embeddings, meaning it adds zero-valued samples in-between the actual samples to increase the sampling rate. Once that is done a Multi-layer Perceptron (MLP) maps the output tokens to a dynamic linear classifier which basically computes the probability of masks at each location of the image. The model is designed in such a way that it predicts multiple

output masks for a single uncertain prompt. They found that for these situations generating three masks would be sufficient in addressing most uncertain cases. The loss acquired during these is used during training and also for each mask a confidence score associated with that mask is generated.

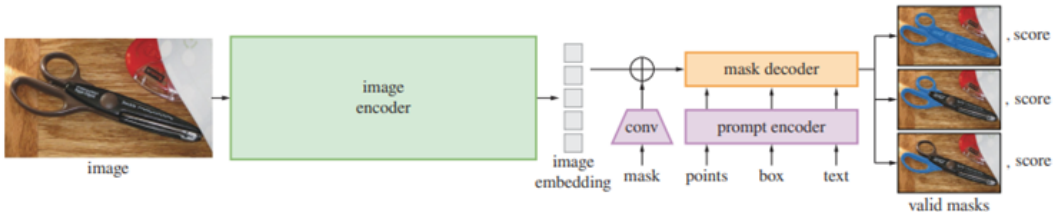


FIGURE 2.3: Overview of Segment Anything Model

Since its release in 2023, SAM has been very popular because of its performance in class-agnostic segmentation. Seeing this impressive results, Ma, Hong, and Shangguan, 2023 took the task of doing few shot counting with the help of SAM. In their research they wanted to perform object counting of unseen category to SAM by providing a few bounding boxes as examples for SAM. They also compared SAM's results with other few shot counting methods. Instead of introducing any new additional zero-shot object detectors for example Grounding DINO(Liu et al., 2023) or zero-shot classifier like CLIP(Radford et al., 2021), they used the SAM image features to distinguish between objects. They compared SAM to other few shot object counting methods based on two aspects. First would be to figure out whether SAM can segment each object and to find out whether SAM can distinguish between the targeted objects from other objects in an image using the reference examples provided. There model works by firstly generating the dense image feature of a given image with the help of SAM's ViT-H image encoder. Basically, get the features of the objects in the image. They then generate the segmentation masks for the examples images by also providing the bounding boxes for it. These are then multiplied with the above generated dense image feature and then average is used to generate the feature vectors for the reference objects. They then use point grids as prompts to generate segments which are then again multiplied with the dense image featured and averaged to give the image features. They compute the cosine similarity between the feature vectors of the predicted masks and the feature vectors of the reference examples and if the cosine similarity exceeds the predefined threshold, they considered that to be a targeted object. The count of all these targeted objects would get us the total targeted objects in an image. They evaluated their approach on FSC-147 dataset and MS-COCO dataset and compared their results with other state of the art few shot counting algorithms.

From the below two images we can see that SAM works well on the top two images but does perform as accurately for the bottom two images. They concluded in their study that SAM might get impressive results in different scenarios, but it clearly lags behind the current state of the art few shot counting algorithms available. The two reasons why SAM performed such badly are, SAM tends to segment congested objects together in a single mask and as SAM was not trained with any semantic class annotations, it hinders its ability to distinguish between objects of different classes.

FIGURE 2.4: Performance of SAM on FSC-147 Dataset - Ma, Hong, and Shangguan, 2023

To solve these problems faced with counting with SAM, Shi, Sun, and Zhang, 2024 took a different approach. In their study they propose a training free object counter which treats the object counting step as a segmentation problem. The vanilla mask generation method that SAM uses lacks any class specific information which intern results in not so great counting accuracy. To overcome this issue, the authors introduced a prior guided mask generation method that incorporates three different types of priors in the segmentation step resulting in improved efficiency and accuracy. These priors act as a guide to refine the mask generation process. Along with that they also introduced a two stage approach to solve the problem the object counting through text. Their two stage approach is a combination of referenced object selection and the above mentioned prior guided mask generation. With the help of Segment Anything Model, they separated and identified individual objects for the input prompts which intern resulted in the generation of binary segmentation maps which correspond to different target objects. The total number of objects in an image can then be obtained by counting the number from these maps. Essentially, in this approach the authors are generating a similarity maps for each object, and then refining the maps with the help of priors using the prior guided mask generation method they introduced and then processing these smaller batches of maps to get the number of objects in an image. Prompts specific to all targeted objects is not possible. To overcome this, we use a regular grid of t x t points that covers the entire image as prompts to generate masks. This approach often leads to inaccurate results as they often segment non-targeted objects. To face this difficulty, they incorporated three types of priors into the segmentation step. This priors will help the in figuring out the positive and negative points within the grid and help us get only the desired targeted objects are segmented. The three priors they introduced are similarity priors, semantic prior and segment prior.

To use similarity prior, they computed the cosine similarity of the reference object feature to the image feature and generated a similarity map. With the help of cosine similarity, we can measure the similarity between two vectors of an inner product space. Once the similarity map is generated, it is converted to a binary similarity map using Otsu's binarization approach. This approach is a global thresholding

technique that is used in image processing to convert a gray scale image into binary image. The idea behind this is to automatically determine the optimal threshold value that separates the pixel intensity values into two classes: foreground (typically represented using white) and background(typically represented using black). The label map generated using this approach, will point to 1 for positive points and 0 to negative points. By using this technique SAM can focus on only segmenting the areas surrounding the positive points while disregarding the negative points. With the use of similarity prior we can segment the object from image with more precision and separate the objects better from our original image. From the image we can see similarity prior works.

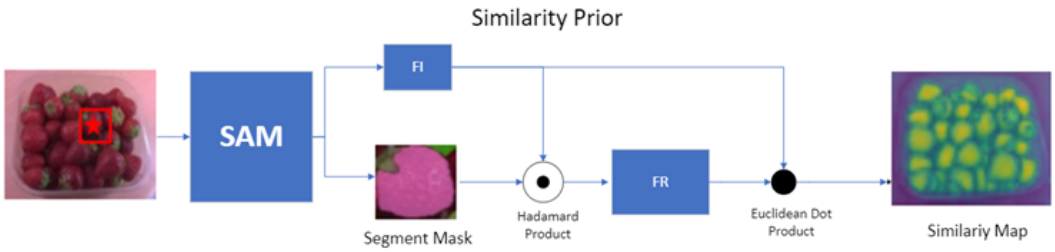


FIGURE 2.5: Similarity Prior - SAM

SAM does the segmentation part in batches. It divides the images into batches for sequential processing. In this way, SAM can use the segments generated from the first processed batches as priors to guide the segmentation process for the following batches. Since SAM can utilize lots of these batches as priors it leads to SAM redundantly processing of batches. To tackle this issue, the authors maintained an overall similarity map that would have all the processed segmented areas till the ongoing batch. If any point in the ongoing batch is already being processed, we simply remove them. With this segment prior SAM can reduce the redundancy it is facing leading to improvement in accuracy and efficiency.

Additionally, to the similarity prior and the segment prior, they also introduced a semantic prior. In this step, they suggest on addition of a reference object feature as a semantic prior so that using this reference object feature SAM's mask decoder can better identify the targeted object and resulting in better segmentation.

By adding the priors, the authors Shi, Sun, and Zhang, 2024 where able to improve the counting process using SAM considerably. On comparison to other state of the art learning based approaches like GMN(Cao et al., 2022), FamNet+(Chu and Ling, 2019), CFOCNet+(Yang et al., 2021)and BMNet+(Shi et al., 2022), their results were better when compared on FSC-147 dataset and CARPK dataset then most of these above mentioned technique.

The results achieved by Shi, Sun, and Zhang, 2024 are significantly better than the results achieved by Ma, Hong, and Shangguan, 2023 when compared on FSC-147 results. Hence, we can say that improving the priors does improves the SAM's counting approach. The authors result only failed to beat Shi et al., 2022 's using the BMNet+ model on both FSC-147 and CARPK datasets.

Class agnostic counting is a technique where we aim to count all the objects in a

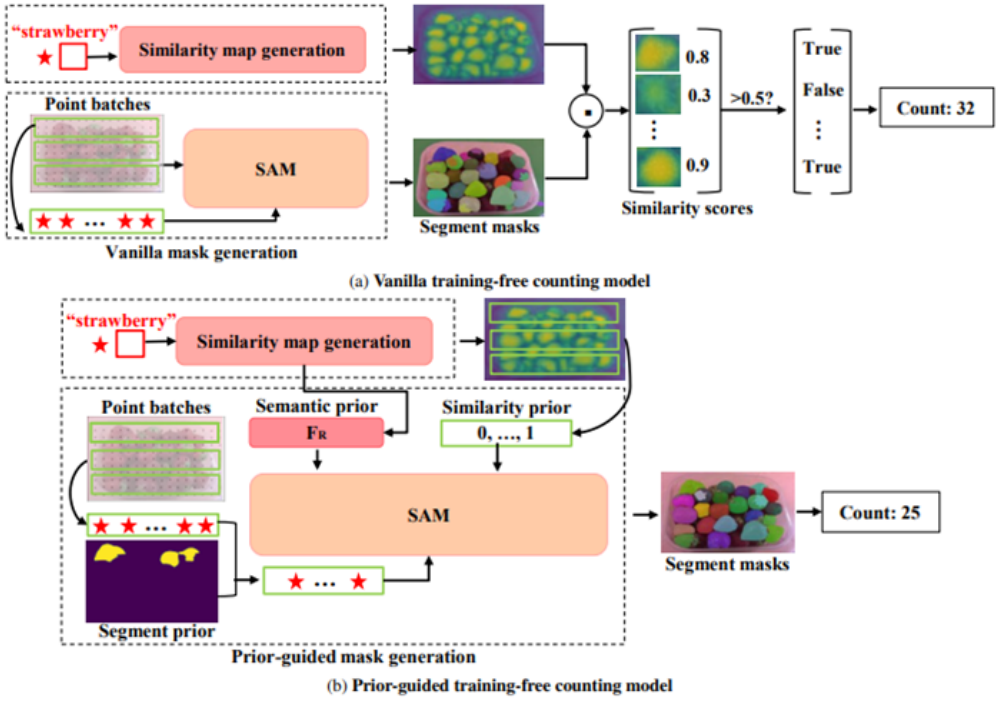


FIGURE 2.6: Difference between Vanilla Mask of SAM and Prior Guided Mask Generation

given image with a few exemplars. In standard class agnostic counting tasks, we extract visual features from the exemplars and match them to the input images to get the count of objects. The two key ideas in this technique are similarity metric and feature representation. The authors Shi et al., 2022found out that this class agnostic counting approach can be even further improved as this approach leads to noisy similarity matching and hence harms the counting accuracy. To overcome this difficulty, they proposed a new similarity aware class agnostic counting framework that learns representation and similarity metric simultaneously. They called this approach Bilinear Matching Network or BMNet. They then improved on their work by further advancing BMNet to BMNet+ which performed similarity from 3 aspects. The first aspect was by representing the instances using their self-similarity to improve the feature robustness in intraclass variations. The second aspect was directly comparing the similarities to focus on the key features of each exemplars. The final aspect was to learn from the supervision signals and imposing constraints on the matching results.

We can clearly see from the below image that the BMNet+ framework is not fixed and is learnable. It learns from its similarity loss making it more robust and achieve greater results and efficiency. BMNet model allows for representation and the similarity metric to occur simultaneously. The core of this BMNet model is its bilinear similarity metric that captures the flexible interactions in feature channels to the similarity model. Although BMNet doesn't use SAM for its segmentation tasks, it currently is the best segmentation model in all the state of the art segmentation models available. The following figure shows the pipeline of BMNet models.

In our proposed idea, we wish to add a YOLO (You only look once) to tackle the

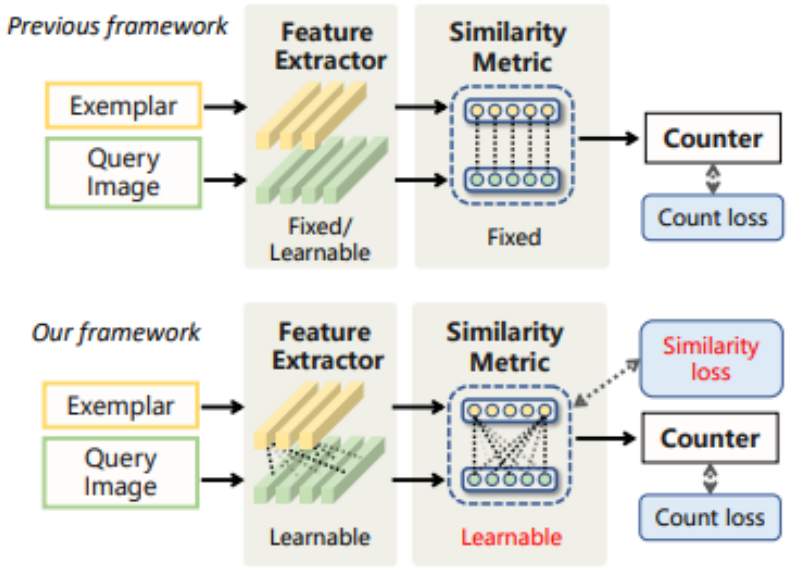


FIGURE 2.7: Difference between Class Agnostic Counting Framework and BMNet Framework

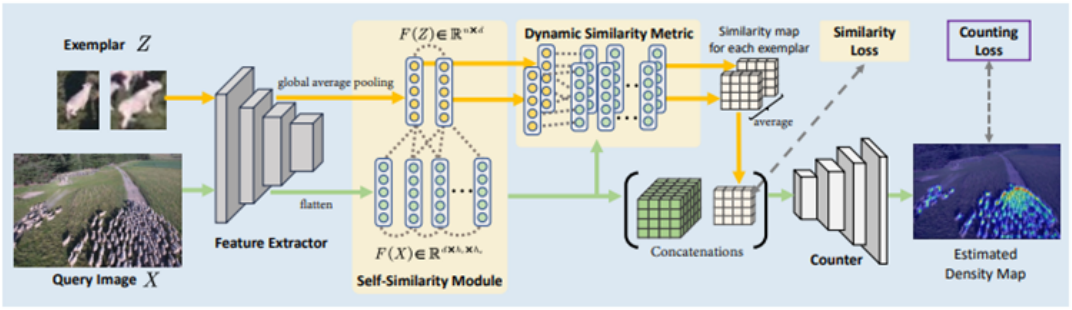


FIGURE 2.8: Pipeline of the BMNet Models

problems faced by the segmentation approach to perform the counting task using SAM. In the year 2015,Redmon et al., 2016 first developed and introduced a real time object detection algorithm known as YOLO (You Only Look Once). It is a single stage detector which used convolutional neural networks (CNN) to predict the class of the objects and mark the objects with the help of bounding boxes. The idea behind this algorithm was to divide the input image into a grid of cells and for each of these cells, it will predict the probability of the object being in that grid and along with that it would also predict the class of the object in the given input image. Unlike its two stage detection predecessors such as R-CNN(Girshick et al., 2014) and its variants, YOLO processed the entire image in one pass, making it faster and more efficient. On comparison to different CNN models like R-CNN, MaskR-CNN(He et al., 2017) etc, YOLO performed much faster and performed detection at a much higher accuracy(Redmon et al., 2016).

On comparing YOLO to its predecessors, author Redmon et al., 2016 listed some benefits of YOLO over its predecessors. First benefit is that YOLO is extremely fast, and it processes images at a much faster rate than any other model. The reason behind this is that YOLO performs frame detection task as a regression problem and therefore it doesn't require complex pipeline unlike its predecessor. The base

model can perform 45 frames per second and the faster version was able to perform at more than 250 frames per second. Not only just speed, YOLO also managed to achieve more than double the mean average precision when compared to other real-time systems. Therefore, YOLO is much quicker and also performs at a relatively higher accuracy. The second benefit of using YOLO is that it considers the entire image into consideration when making predictions. YOLO doesn't use any region based technique or sliding window technique to make predictions but sees the entire image during its training and testing time. A top detection model like Fast R-CNN(Girshick, 2015), would often mistake the background of the image as one of the objects as it doesn't take the larger context of the image into consideration. YOLO on the other hand would make almost less than half of those errors when compared with Fast R-CNN. The third benefit is that YOLO is highly generalizable. Meaning that YOLO could easily detect objects in natural images compared to other detection models. It also means that YOLO is trained on a broader range of images than most detection models and that when applied on a new domain of images, it is highly likely that YOLO will not break down like other detection models. The following figure shows the mechanism behind any object detection model.

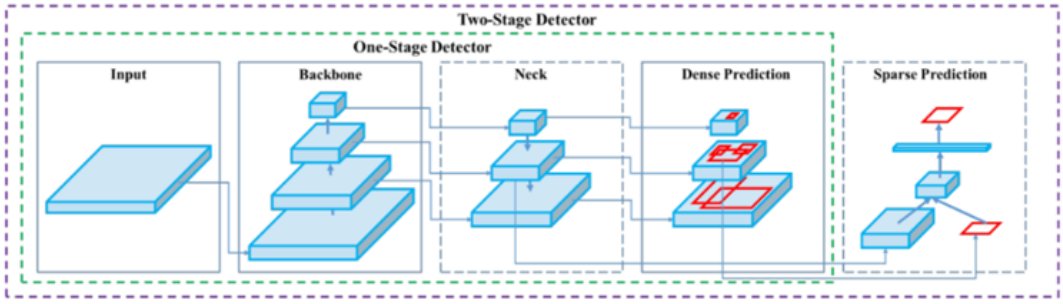


FIGURE 2.9: Mechanics of Object Detection Models

The main architecture consists of a neck, main head, and a backbone. The backbone consists of a pre-trained Convolutional Neural Network that is used to extract low, medium, and high level of feature maps from the input image. The neck is used to merges these feature maps that are extracted from the backbone, and they are merged together using path aggregation blocks like the Feature Pyramid Network (FPN). The feature maps are then passed onto the head where the classification of the objects takes place and also the prediction of bounding boxes. The head can either be a one stage detection model which does dense predictions like YOLO, or it can also have a second-stage detection that perform sparse predictions like in R-CNN series.

YOLOv8 is the latest stable version of the YOLO version. This version also outperforms all its predecessors as it introduced various modifications to its architecture such as spatial attention, feature fusion and context aggregation modules.

From the below figure we can see the architecture of the YOLOv8 model. We can clearly see the backbone, neck, and head structure of the YOLOv8 model. Modified version of the CSPDarknet(Wang et al., 2020) 53 architecture forms the backbone of the YOLOv8 model. The architecture has 53 different convolutional layers in it and is arranged in a cross-stage partial configuration for better information flow between the different layers. The head of the YOLOv8 model has multiple convolutional layer which are directly connected to a series of fully connected layers. These layers are the ones that detect the presence of an object, its bounding boxes and also

predicts the class probabilities for them. The key feature of the YOLOv8 model is the self-attention mechanism which mostly occurs in the head of the model. This important mechanism helps the model to focus on different parts of the image and according to their relevance to the task in hand, adjust the importance of different features. YOLOv8 is also capable of performing multiscale object detection as it utilizes feature pyramid network. This feature helps the model to detect objects within the image which are of different scales and sizes.

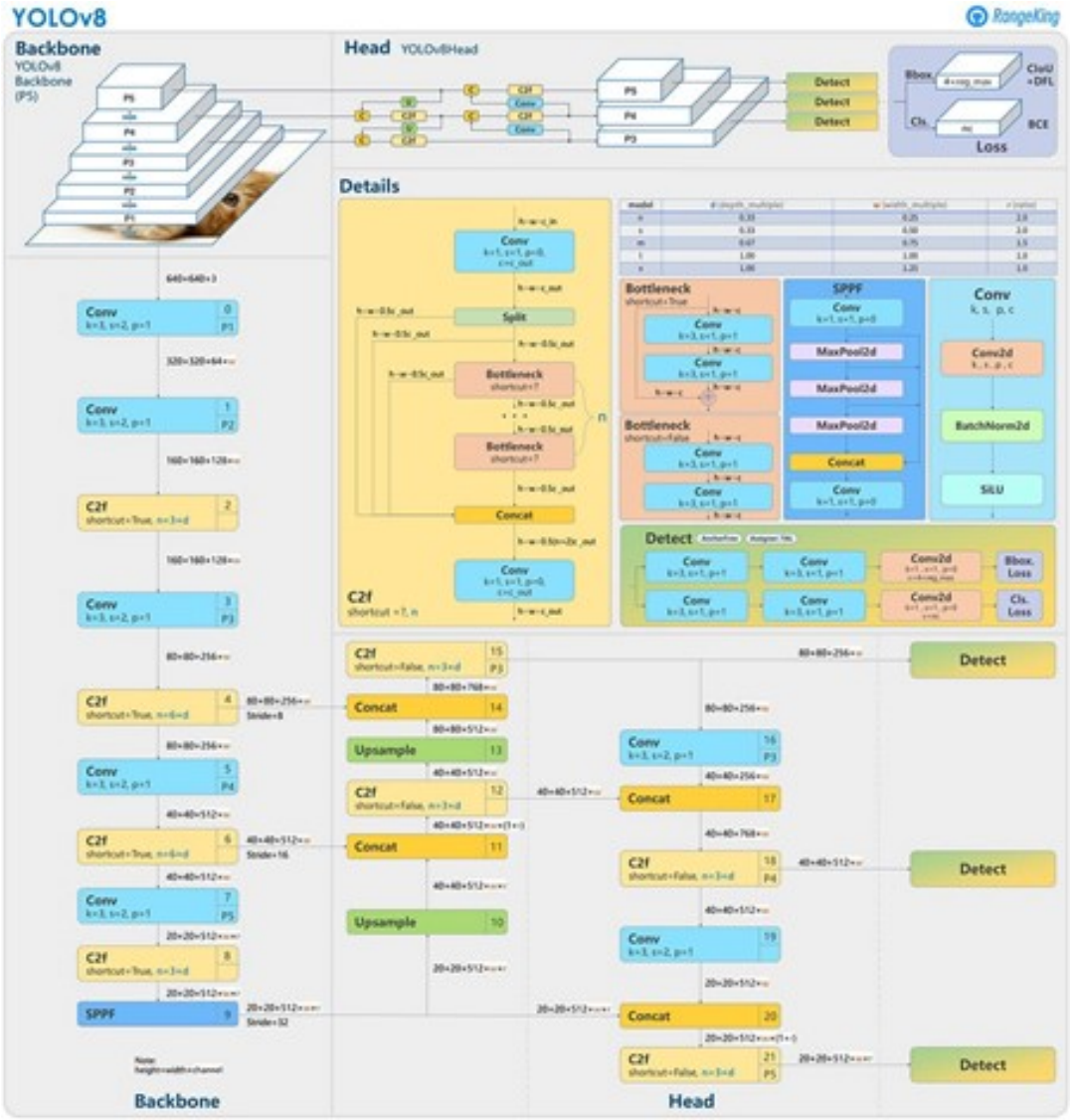


FIGURE 2.10: Architecture of Yolov8 Model

YOLOv8 outperforms its predecessors and other object detection models in both model accuracy and speed of processing the task at hand. The multiple features of YOLOv8 that helps it achieve this feat are first improved accuracy. By using new techniques and optimization the new model is much more accurate in object detection tasks. Second is Faster Speed. Model performs the tasks at much higher speed while also maintaining the accuracy. It is supported by multiple backbones like EfficientNet(Tan and Le, 2019), ResNet(He et al., 2016) and CSPDarknet, providing the users flexibility of choosing the backbone according to the task in hand. To improve

the model performance, it uses adaptive training to get better learning rate and reduce the loss function during training. To improve the robustness and generalization of the model it uses advanced data augmentation techniques like MixUp(Zhang et al., 2017) and CutMix(Yun et al., 2019). Users can choose the models structure and parameters making YOLOv8 highly customizable. It also provides pretrained models which are trained on dense and diverse datasets. One of the key aspects in YOLO's high accuracy and speed is its training routine. As YOLOv8 changes the images during training process, at each epoch the model will see a slightly different variation of the original image. On one of those augmentation YOLOv8 performs the mosaic augmentation(Wei et al., 2020) where it combines 4 images together leading to model to learn objects in different locations.

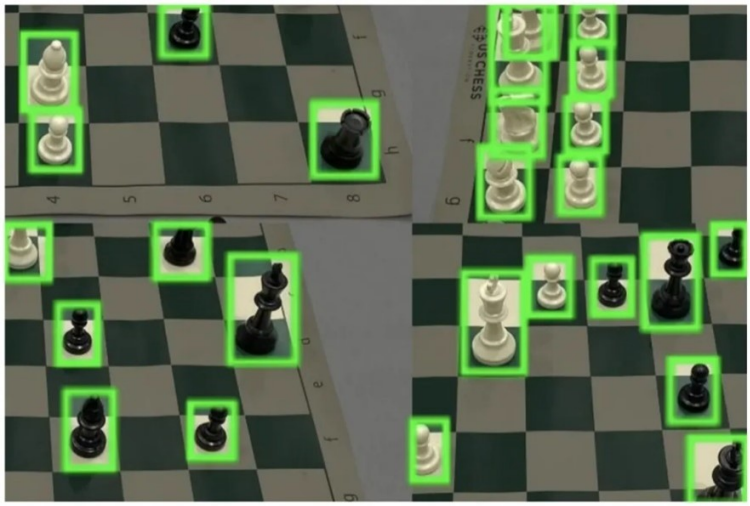


FIGURE 2.11: Mosaic Augmentation of a Chess Board

Although YOLO was designed to perform object detection tasks, it can perform object counting tasks just fine. Now object counting using YOLO is not a new task and there have been countless studies done in the past using YOLO. There have been countless research papers and studies in the object counting using YOLO task that are being done in the past and are still going on to this day. The research done by Wu et al., 2021 could count vehicles and also detect its class. The study done by Ren et al., 2020 counts the number of people in a real time environment and the study done by Mercaldo et al., 2022 could perform the count of blood cells and also perform its localisation using YOLO. The challenge that we took up was to find out the least amount of images required to get significant results. The goal was to use as less images as possible for training and get comparable results to the image segmentation results done through SAM as mentioned earlier. CARPK dataset is a dataset with over 1000 images and contains nearly 90,000 cars from 4 different parking lots which are collected by a drone. During the collection of the dataset, the drone was at the approximate height of 40 meters. Since YOLO was not trained on this view of cars it initially found it difficult to get the accurate results. Studies done by different researchers used YOLO to solve this drone view problem. A study done by Ma et al., 2023 shows us what is the one way we can solve this problem. They used the scalability feature of the YOLO series and decided to customize the YOLO structure to suit their requirements. They started by reconstructing the backbone and the feature fusion networks and simplifying the structure of the overall network and

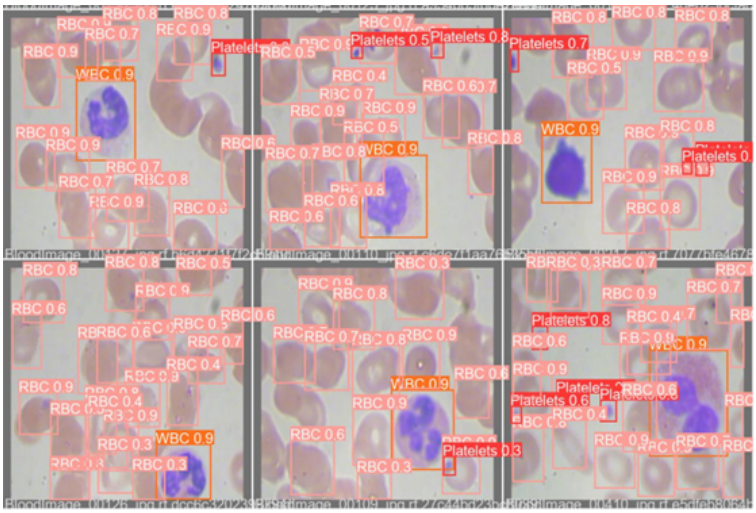


FIGURE 2.12: Work of Author Mercaldo et al., 2022

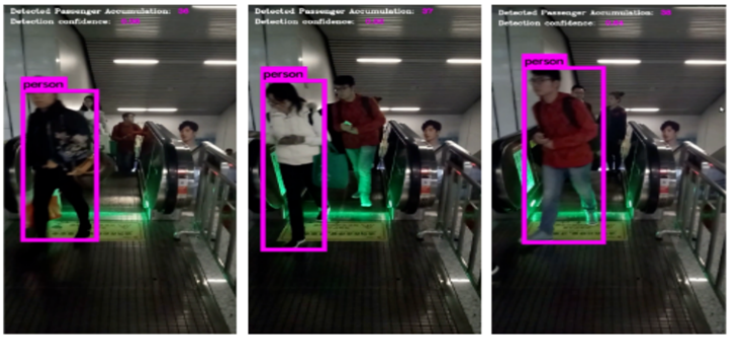


FIGURE 2.13: Work of Author Ren et al., 2020

resulting in the reduce in computational burden. To extract the information of the images better, they decided to incorporate the Dense_CSPDarknet53 in the backbone and also used dense connections for better information flow. They also incorporated an efficient feature fusion block to extract and learn from both local and global features from the feature map. They introduced GS-Decoupled Hero to remove the prediction biases that can be caused because of models localisation and regression capabilities. They named this custom YOLOv5 model, YOLO-UAV. The following figure shows the network structure of YOLO-UAV.

Their change in the YOLO structure and creating a new YOLO model worked as they were able to achieve a much higher results and accuracy then normal YOLO model. The following figure shows the differences between the normal YOLOv5 model and the custom YOLO-UAV model. We can clearly see from these images that the YOLO-UAV model was able to correctly predict most of the given objects in a given image and performed very highly on VisDrone2019 dataset(Zhu et al., 2019). Also, in comparison to other models achieved a higher mAP score.

Another study done by Wang et al., 2023 also used similar approach to Ma et al.,

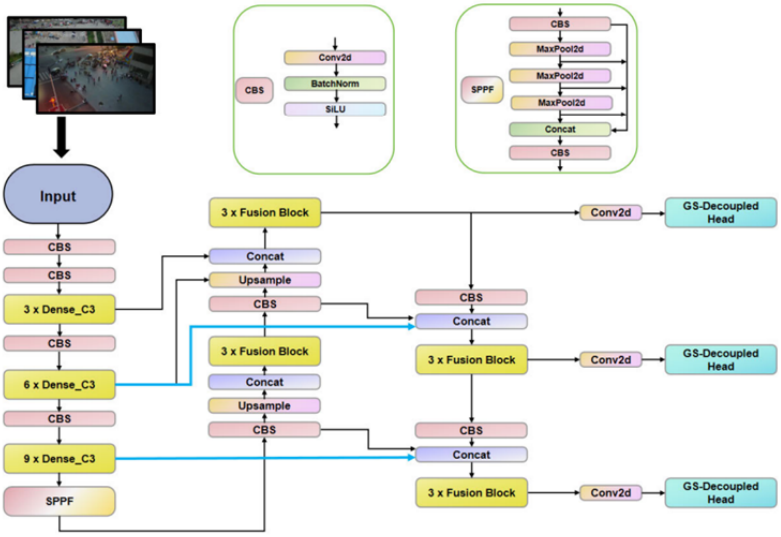
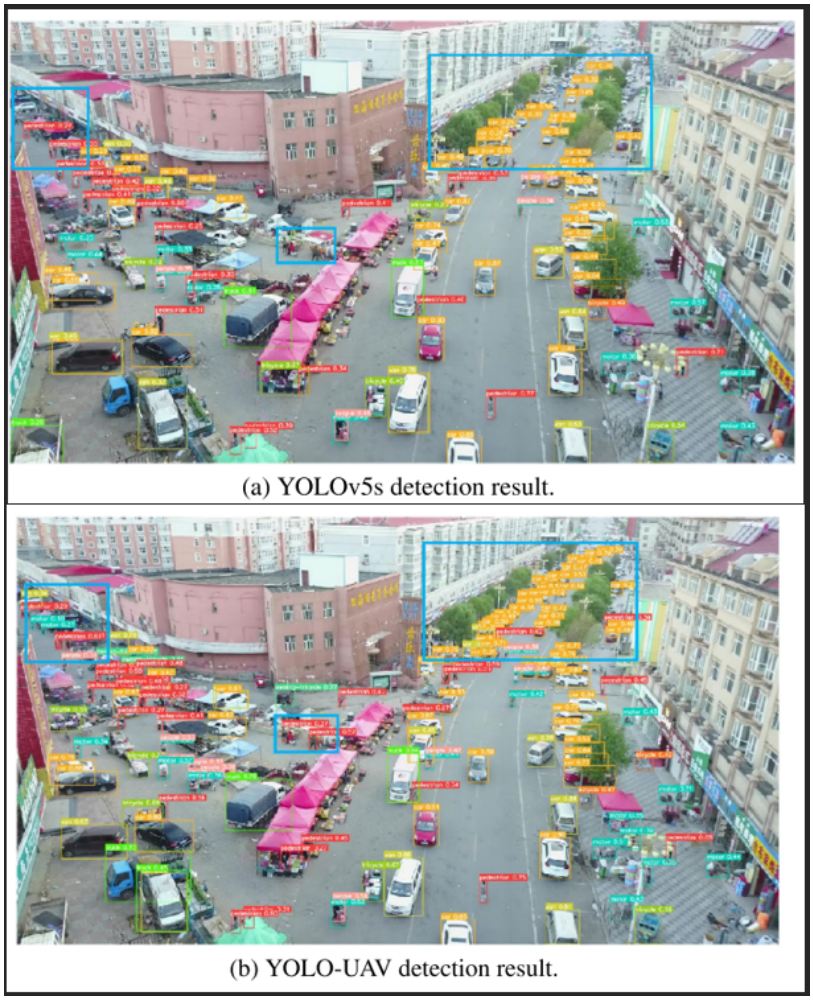


FIGURE 2.14: Network Structure of YOLO-UAV



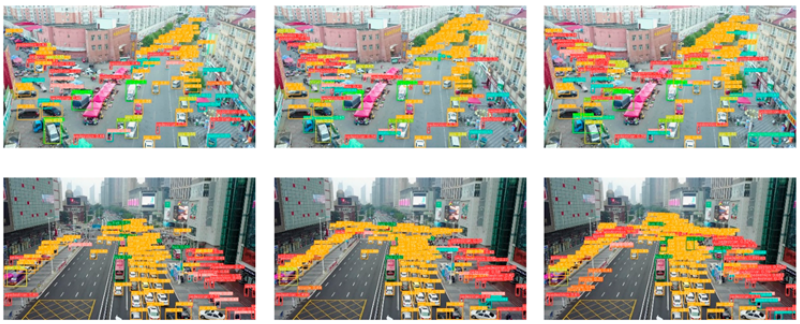
(a) YOLOv5s detection result.

(b) YOLO-UAV detection result.

FIGURE 2.15: Comparison of results between YOLOv5 and YOLO-UAV

2023 work and proposed a lightweight YOLOv8 model that could read the UAV images from the Visdrone2019 dataset. They named their model UAV-YOLOv8s. To

improve the localization ability of the model they used Wise-IoU v3 for bounding box regression loss. To improve the models attention to critical information, they introduced an attention mechanism called BiFormer in their backbone. The proposed model increased the detection performance and reduced the missed rate for smaller objects. When compared to the baseline model the mean detection accuracy of their model is 7.7% higher. Also, when compared to other mainstream models, the performance was also much better. Their models main goal was to improve the models performance to detect small objects. The following figure shows the comparison of YOLOv5, YOLOv8 and the custom UAV-YOLOv8s on the VisDrone 2019 dataset. The results of model successfully identifying tiny objects from drone view



(a) YOLOv5s (b) YOLOv8s (c) UAV-YOLOv8s

FIGURE 2.16: Comparison of results between different YOLO models

are pretty evident in the following image which shows the heat map comparison of YOLOv8 model and UAV-YOLOv8s. These work from the previous authors proved

(a) original images (b) YOLOv8s (c) UAV-YOLOv8s

FIGURE 2.17: Heatmap Comparison between Models

that YOLO is pretty capable in identifying any given object at any given condition. For our model we took a different approach than Ma et al., 2023 and Wang et al., 2023. The reason behind this is that the model that they introduced was solely focused on drone view images and also they used high amount of these images for training. In our problem statement we want to identify images that are in drone view for CARPK dataset but also want to identify 147 different categorises of the FSC-147 dataset and also combining such a heavy restructured model with SAM would be another problem. The restructuring of the YOLO architecture approach would work fine if it was for targeted to one dataset but to identify 147 different classes would not be feasible. And hence we took a different and much more simple approach to solve our problem.

Chapter 3

Methodology

Instead of restructuring the YOLO model's architecture our model required a much simpler approach as we had to then combine this model with SAM and use the bounding box information that YOLO generates to improve the SAM's similarity map generation process. To tackle this problem instead of using YOLO's scalability feature, I used YOLO's generalisation and robustness. I wanted to use the models capability to learn anything.

To do that we trained YOLO on that specific images using Roboflow. Roboflow is a free computer vision developer framework for better data collection and preprocessing and model training technique. With the help of simple dragging across the objects in an image we can annotate the objects in an image. We could set the image set, its resolution the number of images in the train, validation, and test sets. Using this we could also annotate the images into multiple classes as the Roboflow environment has no limits to the total number of classes. The following figures shows how objects can be annotated using Roboflow.

FIGURE 3.1: Annotation process using Roboflow

We would simply drag bounding box on the desired object and once we have done that we could select the class of the selected objects in the bounding box. Roboflow keeps track of the classes present in the image and we could also figure out how many images contain what classes. There is also a paid option to perform Roboflow labelling where Roboflow's AI perform the annotation of the objects in an image. We could also divide the annotation work between our teammates. Once we have performed all the annotation steps and are ready with our annotated dataset we can perform the different preprocessing steps like auto orient, resize the images to desired image size, gray scale the images crop certain regions etc. we can also apply different augmentation steps such as flipping the images, rotating the images

adjusting the brightness, saturation, and hue, along with mosaic and exposure all these factors can affect the models performance. The following figure shows the Roboflow environment.

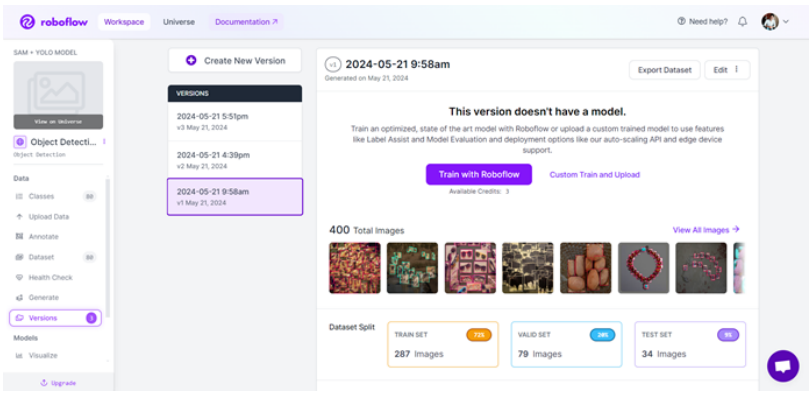


FIGURE 3.2: Roboflow Environment

From the above figure we can see the Roboflow environment where we annotated the model for FSC-147 datasets and CARPK datasets. The figure shows the dataset split for the training annotated dataset and also shows the number of classes and total number of annotated images. Instead of the standard 147 classes of the FSC-147 dataset our dataset had 80 classes. We combined some of the different similar categories into one category. We then trained our annotated dataset on YOLOv8 model.

For CarPK dataset we trained the YOLOv8 model on 30 annotated images, 20 annotated images and 10 annotated images and trained the model for 100 epochs and other model on 400 epochs respectively. So, in total we had 6 custom trained YOLOv8 model for CarPK dataset. For FSC-147 dataset we trained the model on 5 annotated images per class, i.e. for 80 classes, that makes it 400 images, 3 annotated images per class, i.e. for 80 classes, that makes it 240 images and 1 annotated image per class, i.e. for 80 classes that makes it 80 total images in the dataset. We also trained this model on 100 epochs and 400 epochs. So, also for FSC-147 dataset we had 6 custom trained YOLOv8 models.

Our goal is to figure out the least amount of training images required to get results similar to segmentation models from previous author. The goal was to use as less training images as possible and achieve higher results after combining it with SAM. We also wanted to find out how the model would perform when it is trained on different approach and whether training it for longer epochs will results in more accurate results.

Once we had all our models we next step was combining the YOLO layer into SAM. Combining the YOLO layer into SAM was a much more difficult task as it had lots of computational requirements that needed to be met. We wanted to use the YOLO bounding boxes and class information to help SAM understand and differentiate between the objects better. The main problem was to figure out where to add the YOLO layer for SAM to use the YOLO feature information. To get this information we had to better understand the SAM structure and how and what stage it performs what task and using what processing step. Since SAM does not store or evaluate any

class information we had to store the YOLOv8 class information into SAM and help SAM differentiate between these classes using the bounding box information.

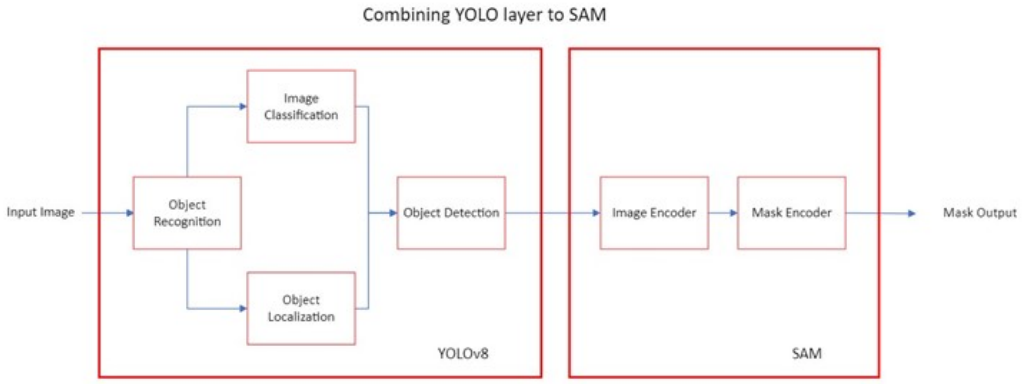


FIGURE 3.3: Basic Structure of Proposed Model

This image depicts a conceptual diagram for combining YOLOv8 and the Segment Anything Model (SAM) to enhance object counting capabilities. The integration aims to leverage YOLO's robust object detection capabilities to overcome limitations inherent in SAM. Let's understand the above figure:

Left Block YOLOv8:

1. Input image: The process will begin with the input image that needs to be analysed.
2. Object Recognition: This first module identifies the objects present in the image.
3. Image Classification and Object Localization: Image classification identifies the objects in an image and object localization determines where the objects can be within an image. In this step, the recognized objects are classified into categories and their locations within the image are determined.
4. Object Detection: Combining the Object recognition, image classification and object localization steps, we can perform object detection. In this step, we perform identification and locating multiple objects, outputting the bounding boxes around each object.

Right Block SAM(Segment Anything Model):

1. Image Encoder: The detected objects from YOLO with bounding boxes are passed into SAM's image encoder. It then processes the image and coverts the image to match the Mask Encoder input.
2. Mask Encoder: The final output will be a set of segmentation masks for all the detected objects allowing for improved object counting.

By integrating YOLOv8 and SAM together we can overcome the problems faced by the previous segmentation approaches. YOLOv8 is very precise in detecting objects quickly and accurately, which helps in improving the segmentation step of SAM.

This combination will not only improve the object detection but will also improve the segmentation step as we will mask only the desired objects. Using YOLOv8 for detection streamlines the process by quickly narrowing down the areas of our interest, allowing SAM to focus on these selected regions for segmentation. This combined approach leverages the strengths of both SAM and YOLOv8 enhancing the overall accuracy and efficiency of the object counting tasks.

Similar to previous works, we also applied a post processing technique to further improve our accuracy and efficiency. The authors Ma, Hong, and Shangguan, 2023, Shi, Sun, and Zhang, 2024 and Shi et al., 2022 all in their works have used Non-Maximum Suppression post processing technique. And therefore, we also decided to use the same post processing technique and checks the improvements it makes to our model.

Non-Maximum Suppression is a post processing technique that is commonly used in object detection algorithms to eliminate duplicate detections and to select the most relevant bounding boxes that corresponds to that detected object. The main idea behind this technique is to compare the confidence scores of the proposed bounding boxes and eliminating the ones that overlap significantly with a higher scoring bounding box. This overlapping between the bounding boxes is usually measured using Jaccard Index also known as Intersection over Union metric (IoU). IoU measures the ratio of the area of overlap between two bounding boxes to the area of the unions as shown in the image below. By balancing the confidence thresholds and IoU thresholds we can improve the models performance.

FIGURE 3.4: NMS eliminating overlapping Bounding Boxes

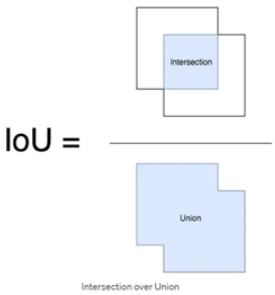


FIGURE 3.5: Intersection over Union

Chapter 4

Experiments & Results

4.1 Datasets

We will perform out analysis on two very different datasets. One is the CARPK dataset. It is a drone view dataset of 4 different parking locations. The other dataset is FSC-147 dataset. This common dataset has images from 147 different categories. The categories can include different things such as simple fruits like oranges, grapes, apples etc to man-made things like, bricks, windows, cars, glasses. It also had animals like elephants, bulls, camels, horses etc and birds like flamingos and pigeons.

4.1.1 CARPK Dataset Results:

Before CARPK dataset was introduced in 2017, all the existing work on object detection and counting focused there work on counting objects in static environments with fixed cameras. Inspired by the invention of unmanned flying vehicles (i.e., drones), the authors Hsieh, Lin, and Hsu, 2017 where interested in detecting and counting objects in a dynamic environments. They also introduced there own Convolutional Neural Network based counting method. To evaluate their model, they collected large scale car parking lot dataset(CARPK dataset) which contained nearly 90,000 cars and was captured from different parking locations using a drone. To the best of my knowledge, this drone-view based car parking lot dataset is the largest drone view based dataset that is focused on object counting tasks. The images where collected from a (PHANTOM 3 PROFESSIONAL) drone from an approximate height of 40 meters. 4 different car parking locations where captured.

We will train our model on 30, 20 and 10 annotated CARPK images separately and train the model on 100 epochs and 400 epochs for each. This will give us 6 models that will be focused on CARPK dataset. Our goal is to achieve high accuracy and to find the least amount to training images required to get the highest possible results.

1.1 30 Annotated images and 400 epochs (Yolo-30-400e):

The first model that we will evaluate is the 30 Annotated CARPK images which are trained on 400 epochs. To Annotate our images, we used Roboflow. Once we had our annotated dataset, we trained the YOLOv8 model on this dataset for 400 epochs. Our model has 144 True Positive Values, 15 False Negative and 15 False Positive values. The high number of True Positives indicate that the model is identifying the cars accurately. With the Maximum F1 score being around 0.9, the indicated F1 score suggests that the model has a high balance of precision and recall at this threshold. The high precision and recall values also indicate that the model is very accurate in

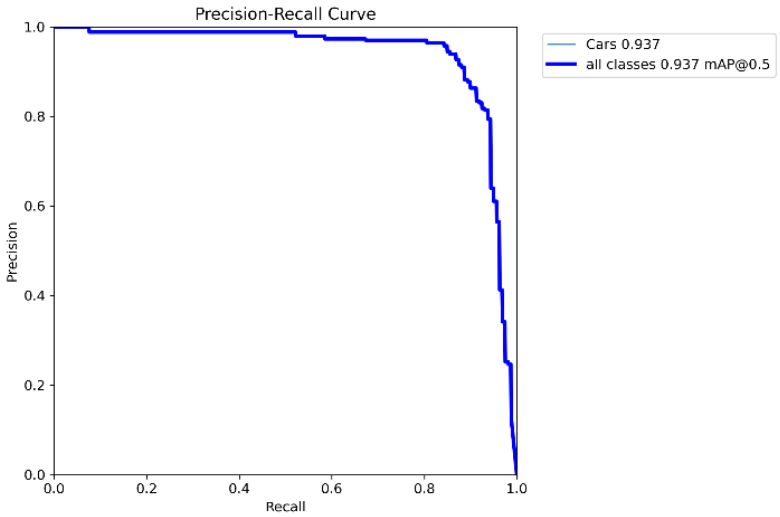


FIGURE 4.1: Precision-Recall Curve-Yolo-30-400e

predicting cars and that it captures almost all the actual car instances. The Mean Absolute Error(MAE) and the Root Mean Square Error(RMSE) results also prove this with the values being **7.40124** and **15.62411** respectively. The following figure shows the cars detected by the model: These results can be further improved by the use

FIGURE 4.2: Car Detection Before NMS

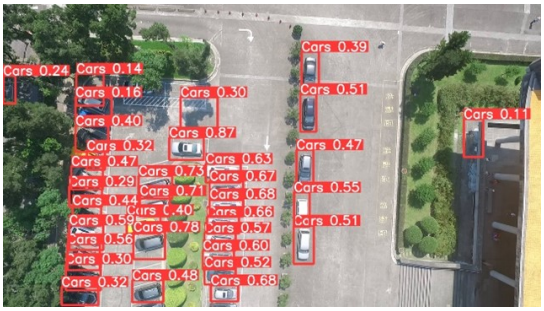


FIGURE 4.3: Car Detection After NMS

of post processing technique known as NMS. By using NMS, it will remove the unwanted overlapping of bounding boxes, which is clearly evident in the first image and help improve the results. After applying the post processing technique NMS, the results have also improved massively with the new MAE and RMSE values being **6.5027** and **13.6489**. This increase in accuracy is also evident in the images as the object as much more clearly denoted with less overlaps.

1.2 30 Annotated images and 100 epochs (Yolo-30-100e):

The second YOLO model was trained on 30 Annotated images and was trained for 100 epochs. The results observed are also pretty similar to the previous results. As we can see from the PR curve from the image below the precision and recall of the model is similar to that of the previous model.

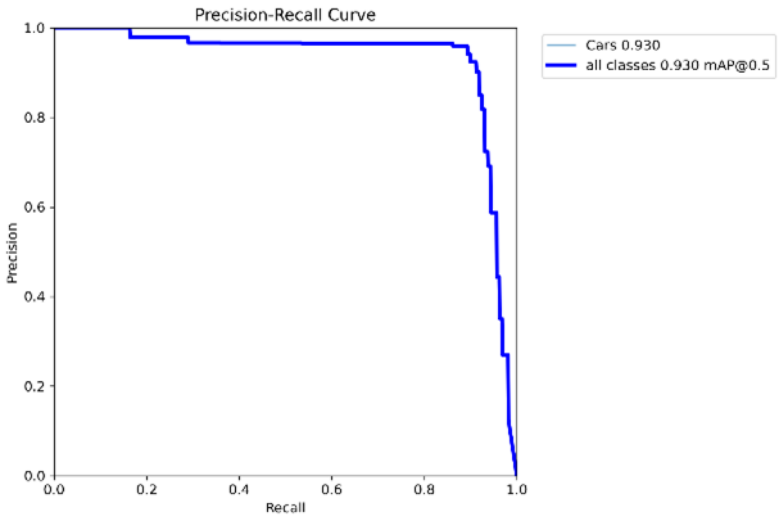


FIGURE 4.4: Precision-Recall Curve-Yolo-30-100e

This is also evident in the observed results with the MAE and RMSE values being **6.8866** and **17.8666** respectively. The image below shows the detection of objects generated by the model.

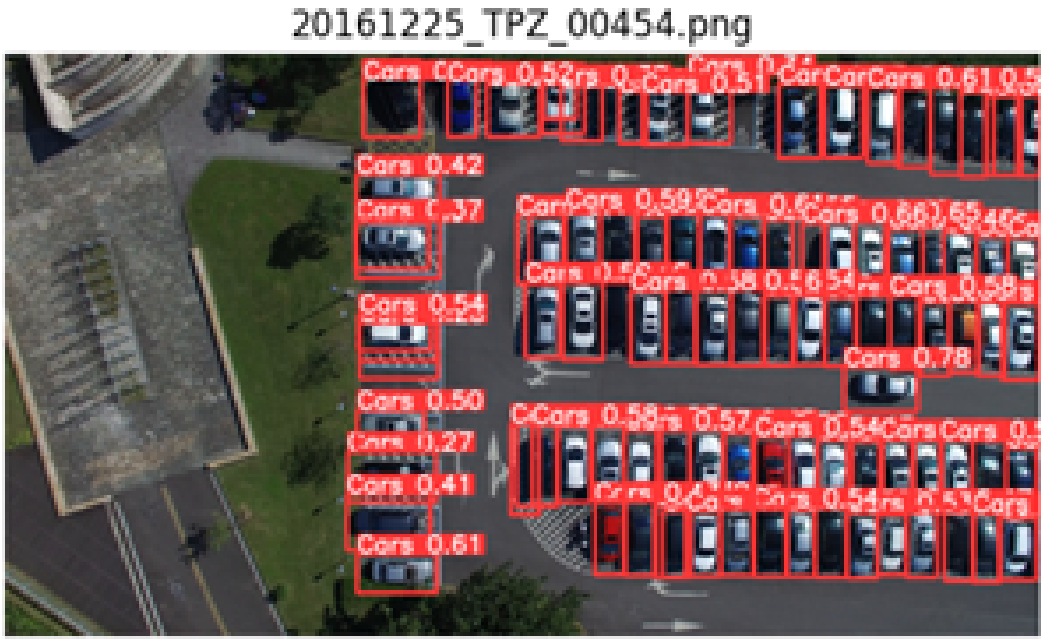


FIGURE 4.5: Car Detection by model Yolo-30-100e

1.3 20 Annotated images and 400 epochs (Yolo-20-400e):

For the third model which was trained using 20 Annotated images and trained for 400 epochs, the results where slightly lower than that achieved by the 30 annotated model. The PR curve also tells the same story.

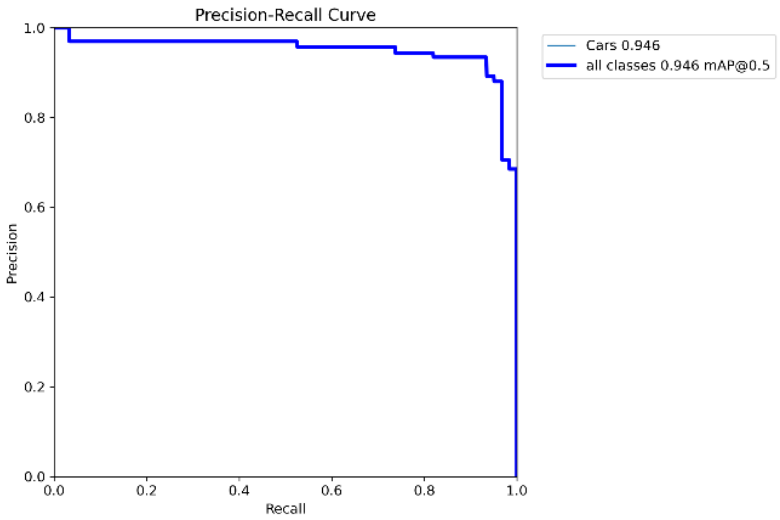


FIGURE 4.6: Precision-Recall Curve-Yolo-20-400e

Similar to the two previous models the maximum F1 score of this mode is also 0.93 with a high precision and recall score. And similar to the previous models, this model is also equally accurate as observed by the results. The MAE and RMSE values after applying the post processing technique NMS are **6.55244** and **14.9403** respectively. The images show the accuracy of the model.

FIGURE 4.7: Car Detection by model Yolo-20-400e

1.4 20 Annotated images and 100 epochs (Yolo-20-100e):

Now this is where we start to see slight drop in accuracy from the model. This model was trained on 20 Annotated images and was trained for 100 epochs. The precision and recall curve shows the accuracy of the model being just slightly below that of the previous models. The models also start showing lots of overlapping.

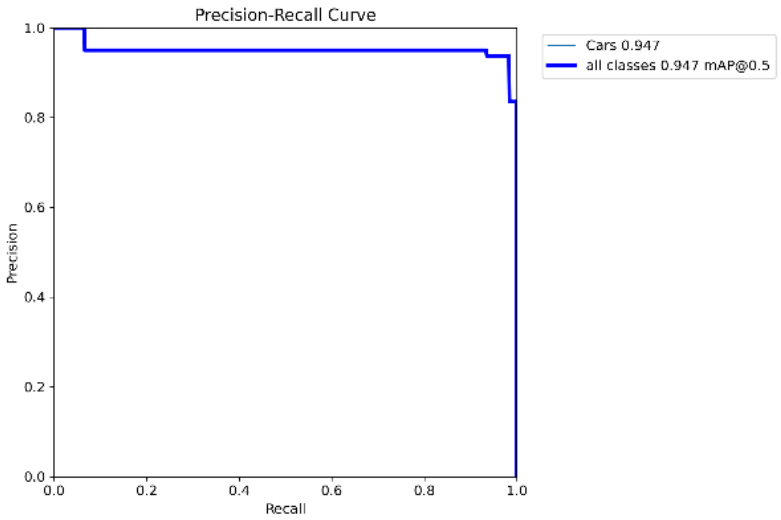


FIGURE 4.8: Precision-Recall Curve-Yolo-20-100e

The MAE and RMSE results show the drop in accuracy and precision as suggested by the PR Curve and F1 score. The results are **8.66919** MAE and **20.63325** RMSE. The detected object in the images also shows the drop as the detected objects have lots of overlaps.

The following two images shows the drop in performance and the overlaps in the object detection by the model.

FIGURE 4.9: Drop in accuracy and overlaps in detection by model Yolo-20-100e

1.5 10 Annotated images and 400 epochs (Yolo-10-400e):

For 10 Annotated images for both 100 and 400 epochs we see a drop in accuracy. We can see this drop evident from the drop in precision score from 0.93$\tilde{0}$.94 to now around 0.91$\tilde{0}$.92. This is also evident from the confusion matrix as now we only have 58 true positive value compared to around 144 in the beginning. The following images shows the PR-Curve of the model.

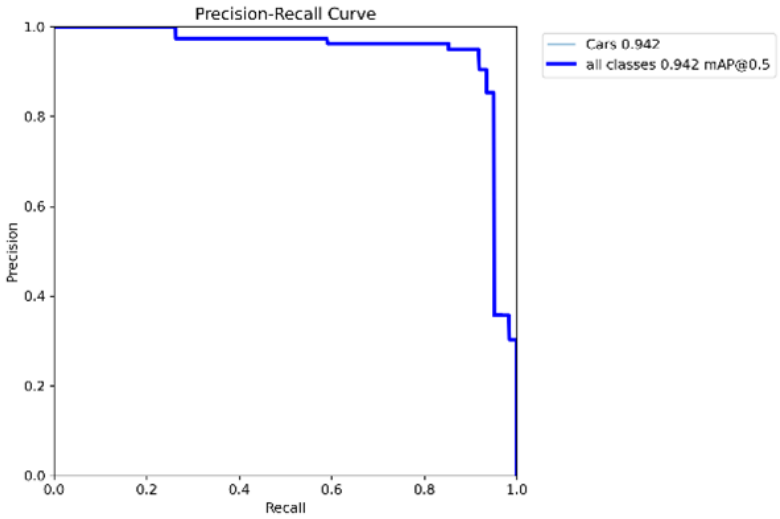


FIGURE 4.10: Precision-Recall Curve-Yolo-10-400e

The models performance evaluation on the CARPK dataset also showed the drop with MAE values being around **11.4132** and RMSE values being around **17.73834**. These values were achieved prior to the post processing step of NMS. After NMS we see an improve in performance where the MAE and RMSE values being around **6.7899** and **13.4617** respectively. The following image shows the object detection achieved by the model.

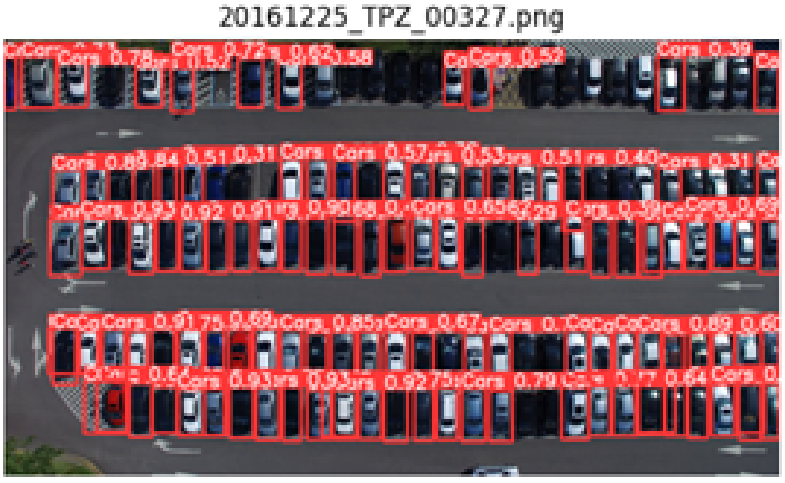


FIGURE 4.11: Car Detection by model Yolo-10-400e

1.6 10 Annotated images and 100 epochs (Yolo-10-100e):

Similar to its 400 epoch counterpart, the 10 Annotated images trained for 100 epochs model shows a significant drop in accuracy when compared to other models. As evident from its precision score of around 0.85, the model shows drop in precision in detecting objects and classifying them. This drop when compared to other models which performed at around 0.91 to 0.94 is significant. With only 56 True Positive values from the confusion matrix and around 5 false positive and false negative values each shows the performance drop. The following figure shows the PR-Curve of the model.

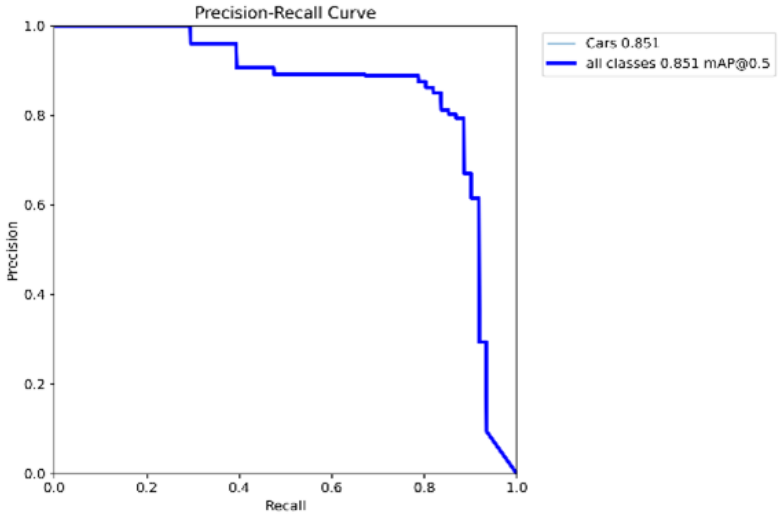


FIGURE 4.12: Precision-Recall Curve-Yolo-10-100e

The model start shows lot more false results and the overlapping's of objects has also increased significantly. Along with that the model also starts missing objects. The MAE and RMSE values drop to **21.11203** and **34.29612**. The following images shows the drop in performance by the model with the help of object detection as the model fails to detect all the objects in the image, model detecting false objects and also overlapping between detected objects.

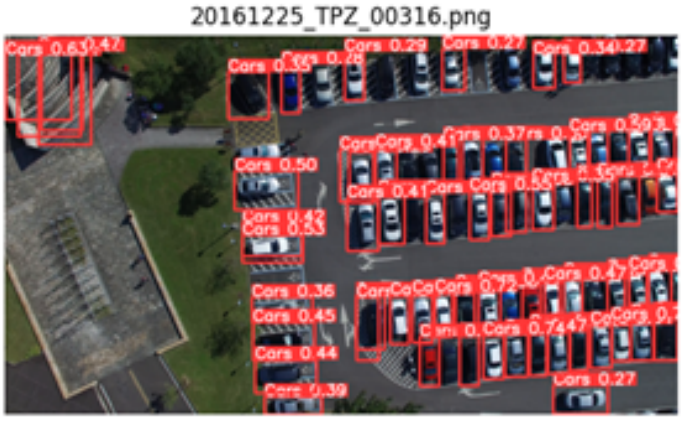


FIGURE 4.13: Car Detection by model Yolo-10-100e

1.7 Comparing 100 Epoch Results

Lets compare our 100 epochs results together and see what we can understand from this We can infer from the above results that as we decrease the number of training images there is an increase in drop of precision and accuracy. The following image shows the comparison of the 100 epochs results for 30, 20 and 10 Annotated image models.

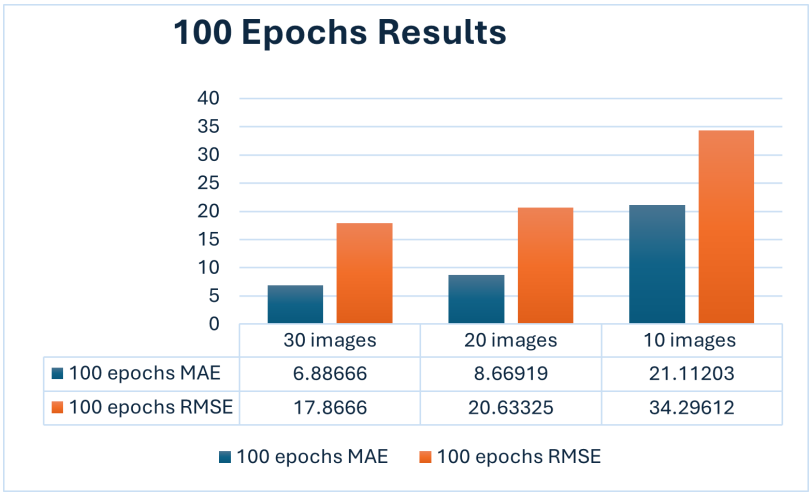


FIGURE 4.14: Comparison of 100 Epoch Results

1.8 Comparing 400 Epoch Results

Now let's compare the results of 400 epochs from together and check the results. We will see two different 400 epochs results, one will show the results of the model before the NMS was applied and the second image will show us the improvement in results. The following image shows both the 400 epoch results. We can infer from these results; NMS helps the models to achieve consistent results. Without NMS, similar to 100 epochs results, we see a drop in performance as the number of training images decreases.

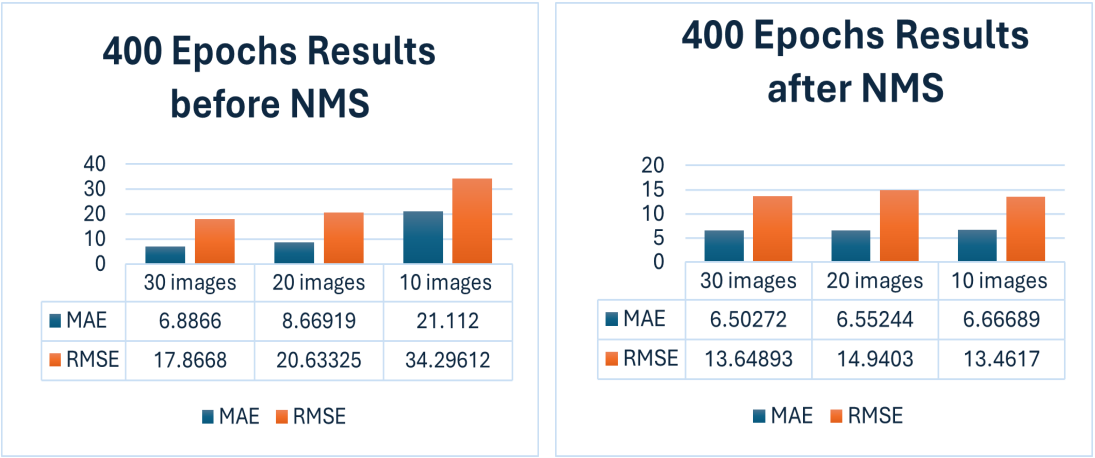


FIGURE 4.15: Comparison of 400 Epoch Results

1.9 Overall Comparison

Now lets compare results of 100 epochs and 400 epochs and see the difference in performance. WE can infer from these results that as the number of training images decreases the higher the number of epochs in training the higher the accuracy of model. The results for 30 images remain consistent regardless the number of epochs in training. Both 30 images and 20 images results are good regardless of the epochs and get us comparable results to previous works using segmentation models.

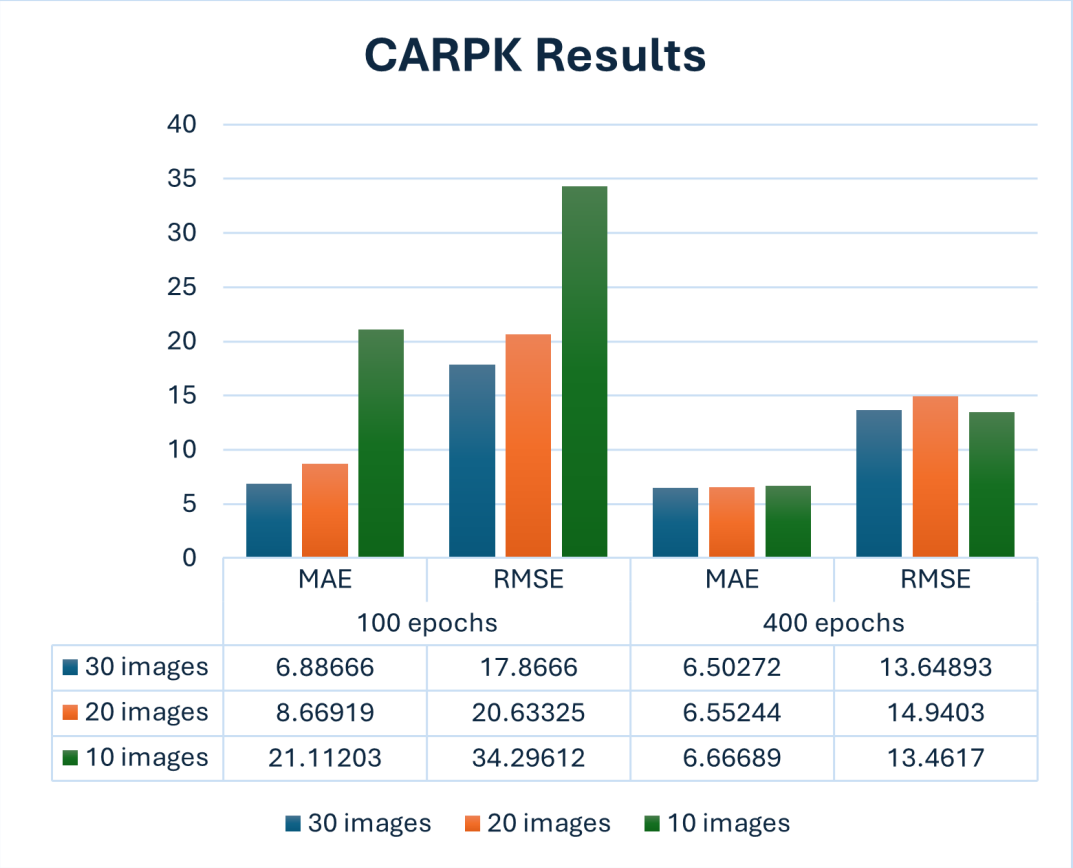


FIGURE 4.16: Overall Comparison Results

4.1.2 FSC-147 Dataset Results:

Authors Ranjan et al., 2021 introduced this dataset in there work on Learning to Count Everything. This dataset contained 6000 images and is divided into 147 different categories. The categories can range from hot air balloons, elephants, bees, cows, apples, banana's etc. Instead of dividing this dataset into the usual 147 categories, we decided to divide the classes into 80 classes and combined some of the similar looking classes together. Lets now look into the performance of our models on this dataset.

2.1 5 Annotated images per class for 400 epochs (Yolofsc-5-400e)

We train this Yolo model on 5 Annotated images per class and trained it for 400 epochs. As we had 80 classes the total number of Annotated images would be 400 images. Since we only trained the model for 5 images per class, the PR curve and F1 score are slightly different with the precision being around 0.4. This along with the increased number of classes from the CARPK dataset would result in decrease of results compared to CARPK results. The PR-curve can be seen from the following images. We can see the changes in this curves with the increase in number of classes.

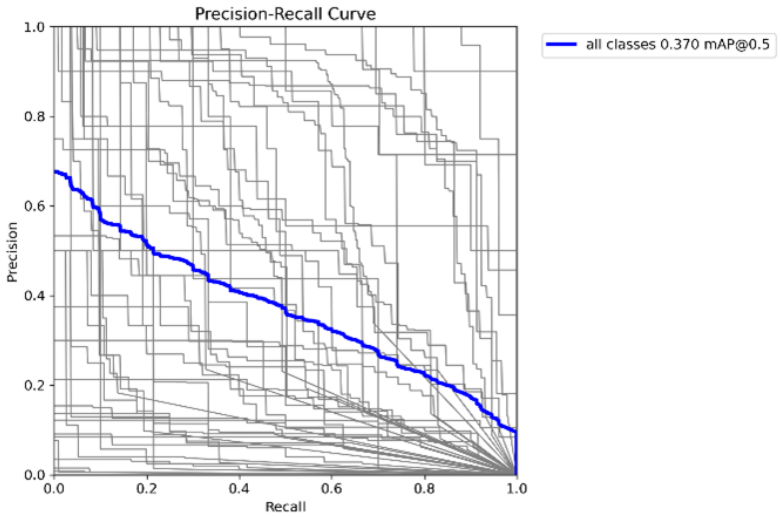


FIGURE 4.17: Precision-Recall Curve-Yolofsc-5-400e

The results of this low accuracy of the model are shown in the MAE and RMSE values being around **27.02008** and **95.15603** respectively after NMS. The following images shows the objects detected from the model.

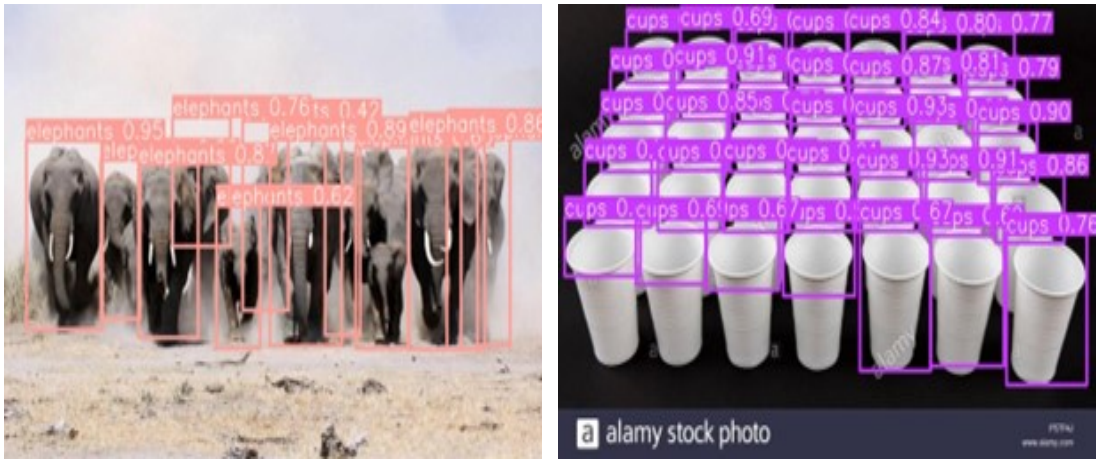


FIGURE 4.18: Object detection from the model Yolofsc-5-400e

2.2 5 Annotated images per class for 100 epochs (Yolofsc-5-100e)

The results of 5 Annotated images per class for 100 epochs are less accurate than its 400 epoch counterpart. The precision score of the model also drops to around 0.38. The PR-curve and F1-score show the models decrease in accuracy.

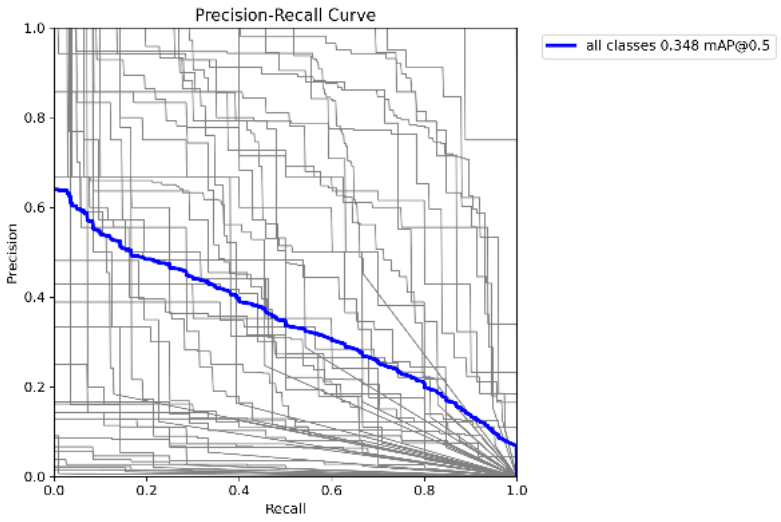


FIGURE 4.19: Precision-Recall Curve-Yolofsc-5-100e

The MAE and RMSE values are **28.593183** and **96.8849** respectively. The following images shows the detected objects by the model. We can see from the below images that the model is very similar in accuracy to the previous model Yolofsc-5-400e. This shows that for 5 images, the number of epochs doesn't matter as we get very similar results. We can also see from these two models that model is not confusing between objects which will be the case for the upcoming models.

FIGURE 4.20: Object detection from the model Yolofsc-5-100e

2.3 3 Annotated images per class for 400 epochs (Yolofsc-3-400e)

With this model we see a decrease in accuracy and precision. The models accuracy will continue to decrease as we decrease the number of images. With the precision value dropping to around 0.34 as indicated by the PR graph. The following image shows the PR-Curve.

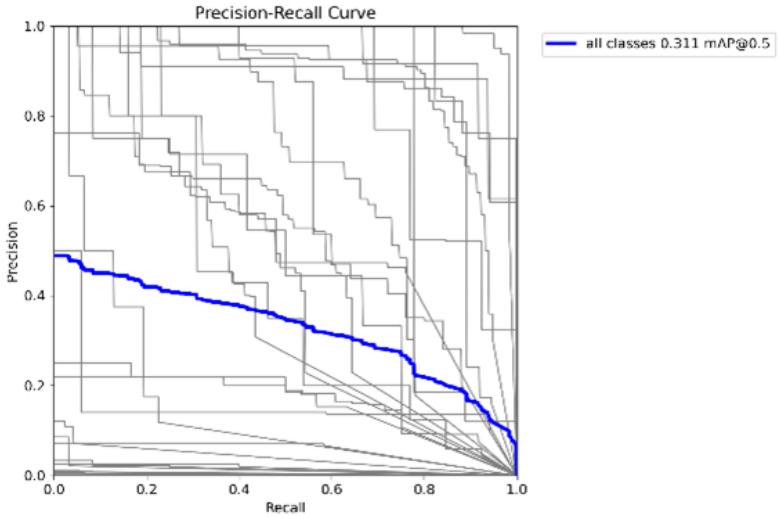


FIGURE 4.21: Precision-Recall Curve-Yolofsc-3-400e

The models dip in accuracy is also depicted by the drop in its MAE and RMSE values. The MAE and RMSE values of this model are **31.46624** and **92.342030** respectively. We can also see the drop in performance by watching at the detection of the objects. We can see that the models starts to miss a few objects. This is clearly indicated in the below image where we show the object detected by the model. We can see from the images that the model is detecting less objects in the image but it is detecting correctly and not giving the object wrong classes.

FIGURE 4.22: Object detection from the model Yolofsc-3-400e

2.4 3 Annotated images per class for 100 epochs (Yolofsc-3-100e)

With the precision score of around 3.2 the models accuracy drops even further for 3 Annotated images per class. This is also evident with the models mixing up between classes and adding fake classes to the objects. The MAE and RMSE values of this model is **33.37508** and **92.9604** respectively. The PR-Curve for the model is as follow.

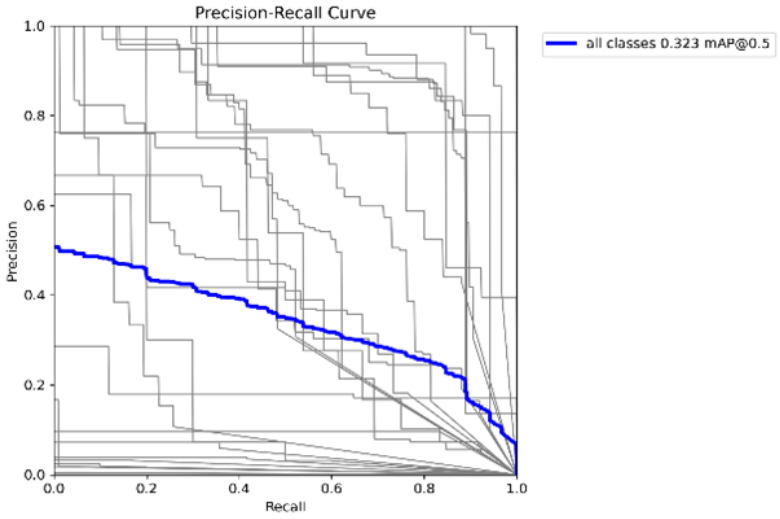


FIGURE 4.23: Precision-Recall Curve-Yolofsc-3-100e

With the following images we can clearly see the mixing up of objects being done by the model Yolofsc-3-100e. Similar to the previous model we can see that the model has drop in accuracy. We can also see that for similar looking classes the model starts to mix up between objects. We can see from the image, unlike previous models, Yolofsc-3-100e starts mixing up objects classes like elephants and penguins. Along with that the other classes it mixes up are coins and shells, apples and oranges with balls etc. We can see that the model is falsely detecting the shape of the object but also falsely detecting the colour of the object.

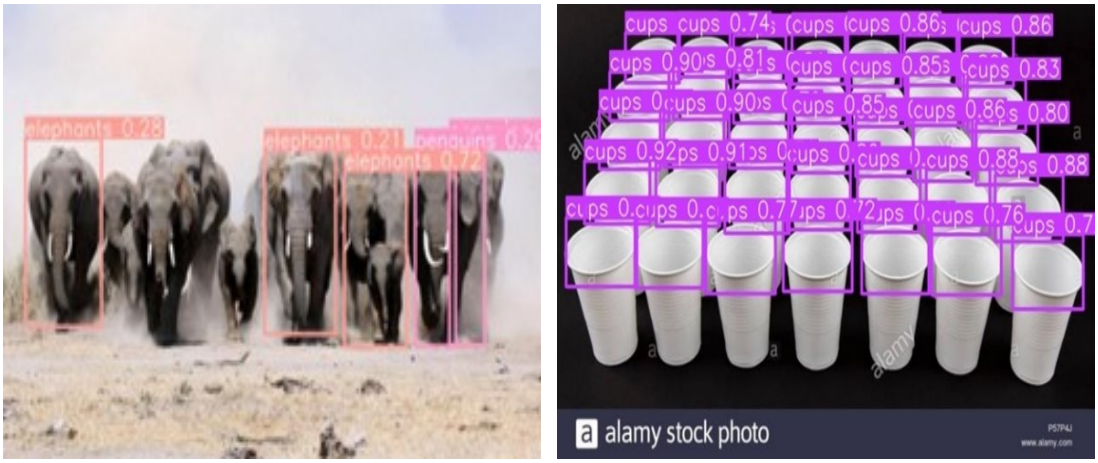


FIGURE 4.24: Object detection from the model Yolofsc-3-100e

2.5 1 Annotated images per class for 400 epochs (Yolofsc-1-400e)

With the number of images per class decreasing even further the models accuracy also decreases. The precision accuracy of the model is 0.27. The PR Curve shows that the model even fails to recognise some objects.

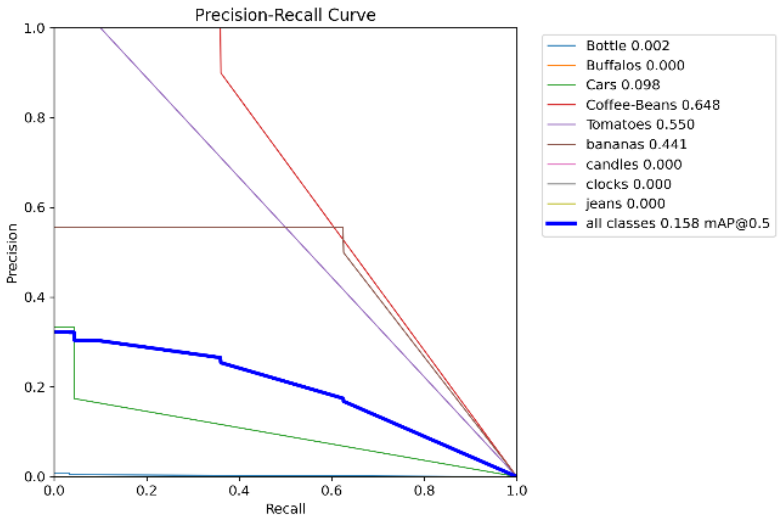


FIGURE 4.25: Precision-Recall Curve-Yolofsc-1-400e

The MAE and RMSE values being around **39.37054** and **102.4155** respectively. We can see from the below images that the model starts mixing up between objects.

We can see from the images that the model starts mixing up simple objects like shells and coins. The previous models where able to clearly identify and distinguish cups from other objects, but from this model we can see that the model starts identifying them anything but cups.

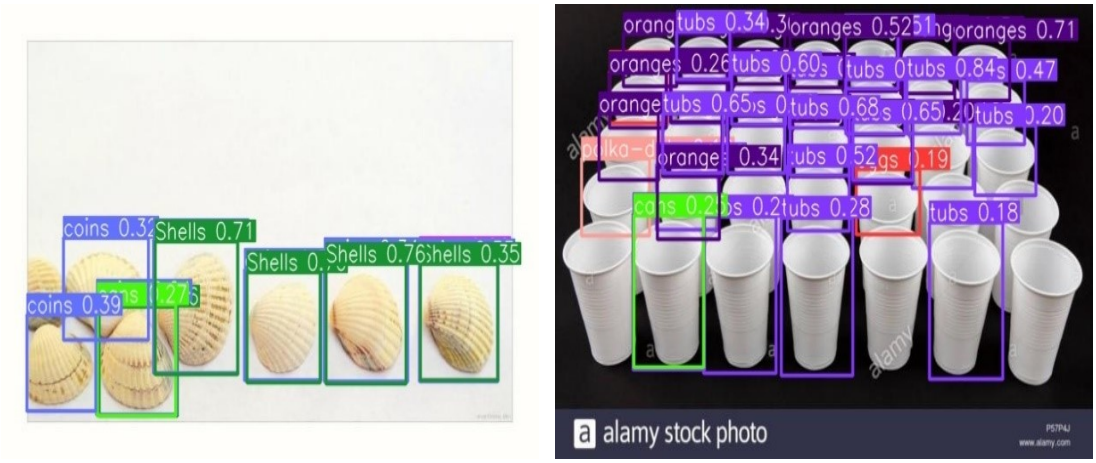


FIGURE 4.26: Object detection from the model Yolofsc-1-400e

2.6 1 Annotated images per class for 600 epochs (Yolofsc-1-600e)

With 1 Annotated images per class for 100 epochs, this model has the least accuracy. The precision values are around 0.25. This is depicted from the PR-Curve of the model.

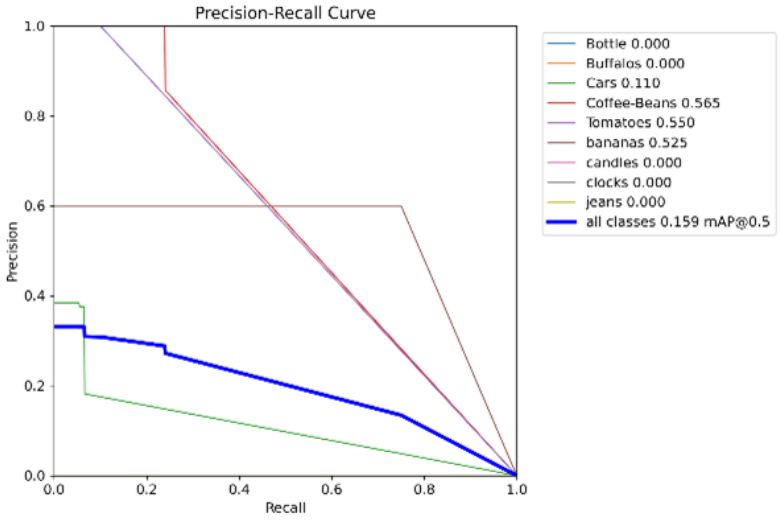


FIGURE 4.27: Precision-Recall Curve-Yolofsc-1-100e

The following images shows how the model mixing objects and the low accuracy of the model.

FIGURE 4.28: Object detection from the model Yolofsc-1-100e

From these images we can clearly see that this model is the worst performing out of all the models. We can see that model unable to differentiate between objects.

2.7 Comparing Results

We trained the YOLO models on 5, 3 and 1 Annotated images per class and trained the model for 100 and 400 epochs respectively. We categorized the model into 80 classes instead of the standard 147 for the FSC-147 classes.

We can clearly see from the results that as we increase the number of classes from 1 in CARPK dataset to 80 in FSC-147 dataset the difference in the 100 epoch and 400 epoch results is not as large as observed in CARPK results.

We can also observe that as we decrease the number of training images, the decrease in performance is not as large. We can also see that with the increase in number of classes, YOLO now confuses between objects in an image. We can say that with 5 images per class results gives us comparable results to previous work done through segmentation model. The following image shows the comparison results for the FSC-147 dataset.

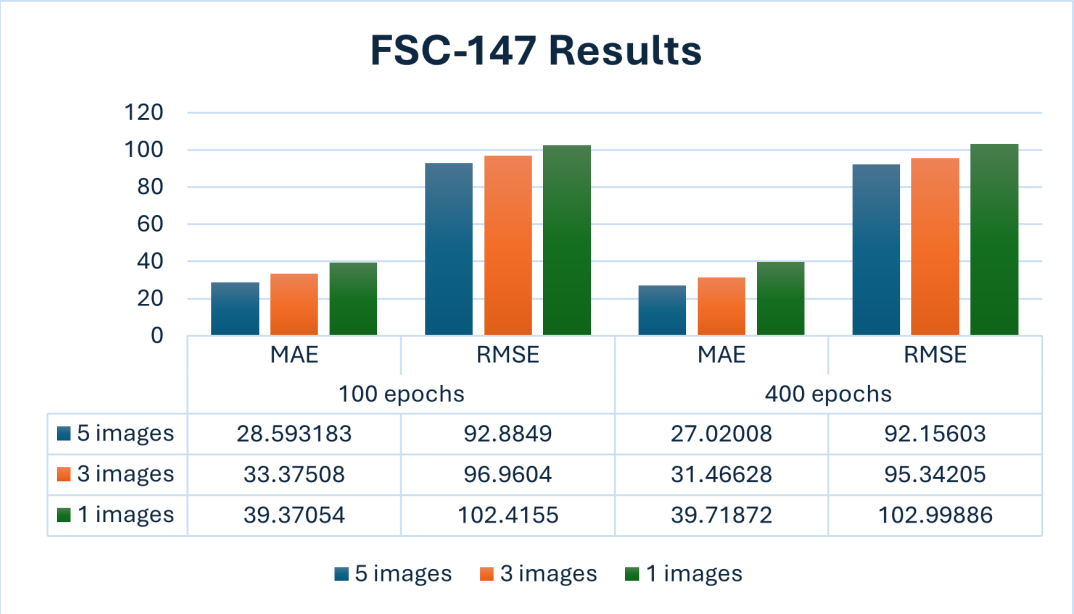


FIGURE 4.29: Overall FSC Comparison Results

4.1.3 Comparing Results with Previous Work:

We will compare our work with Ma, Hong, and Shangguan, 2023 work in counting using SAM, Shi, Sun, and Zhang, 2024 work in Training free object counting using Prompts and will also compare our work with the best segmentation model for object counting BMNet and BMNet+ work done by author Shi et al., 2022 in their similarity aware framework.

Let's first compare CARPK results and see how they fair against previous authors work

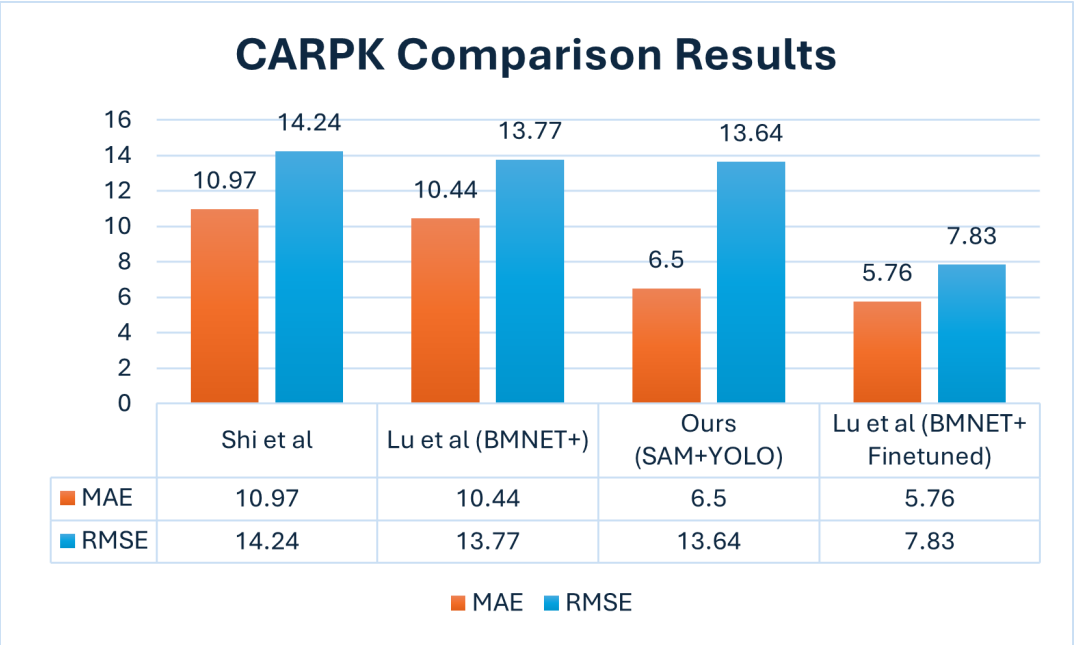


FIGURE 4.30: Comparing CARPK results with previous work

From the above results we can see that our CARPK results are better than most image segmentation previous work results. We can see that our SAM+YOLO model (Yolo_5_400e), outperforms the other SAM object counting model work done by author Shi, Sun, and Zhang, 2024. Our model also outperforms Shi et al., 2022 using BMNet +, although this model does not use SAM in any sense, we still use it for comparison as this model is the best segmentation model that performs the tasks of object detection and counting. Their BMNet+ model when fine-tuned for the CARPK dataset slightly outperforms our model. Therefore, we can conclude from these results that training YOLO model on 30 Annotated CARPK images for 400 epochs and combining the YOLO layer into SAM, we can get better results then previous work done on Object Counting using SAM.

Now lets look at the FSC-147 dataset results and see how our model fares against the previous work results.

For FSC-147 dataset results, our SAM+YOLO model(Yolofsc_5_400e) gives better performance than Ma, Hong, and Shangguan, 2023 work using SAM but fails to outperform Shi, Sun, and Zhang, 2024 also using SAM. Our model also doesn't have the accuracy to match the best segmentation model BMNet+ 's results. The reason behind this is the increase in classes. With only 5 Annotated training images per class the model fails to distinguish between similar looking objects.

We can conclude from these results that 5 images per class for 80 classes for FSC-147 dataset trained on 400 epochs, our model gives us comparable results to previous models but if we want to outperform them using our method, we need more training images then 5 per class as that would help YOLO identifying objects better and help it better distinguish between objects.

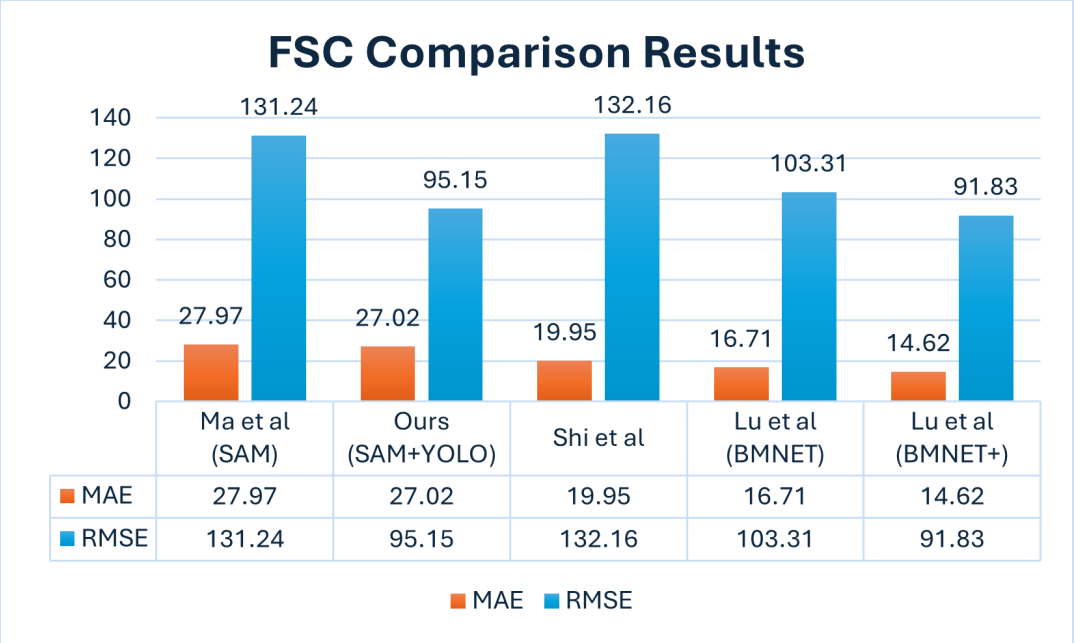


FIGURE 4.31: Comparing FSC-147 results with previous work

Lets look at how YOLO helps the model and solve the previous problems encountered using Segmentation methods. Let's take image number 5 from the FSC-147 dataset. We can see from the below image that the previous models fail identify all the objects in the image. In the figure below that we see that the YOLO model detects all the 119 hot air balloons in the image, then the Segmentation step occurs using that detection's in mind and finally our model detects all the hot air balloons in the image.

FIGURE 4.32: First YOLO detection, then Segmentation using those detections

The following two images will help us better understand this process. Part a of the image shows the object detection step using YOLO, Part b shows the segmentation

FIGURE 4.33: Final Result with Segmentation masks and Yolo Detections

masks, and finally part c shows the combination of these two models.

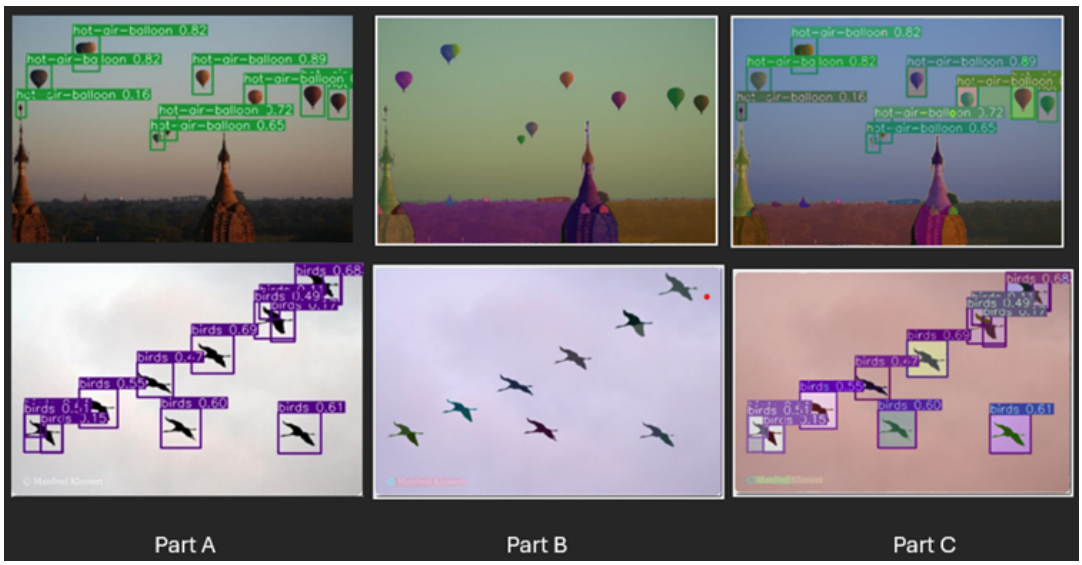


FIGURE 4.34: Different parts of the model in action

Chapter 5

Discussion

5.1 Limitations

While SAM and YOLOv8 offers great results and accuracy in the object counting and detection department, there are also limitations associated with this approach. The limitations are mostly related to resources requirements. The following are the limitations in this approach:

1. Complexity of Integrity: integrating these two sophisticated models was technically challenging and required extensive tuning and optimization.
2. Inference speed and latency: Although YOLOv8 model is extremely fast, after adding it to SAM, it increased the computational load and slowed down the inference speed.
3. Resource Consumption: Combining and running two models in tandem required lots of computational resources including memory and processing powers.
4. Dependency on Detection Accuracy: The accuracy of the segmentation of SAM masks would be heavily depended on the YOLO object detection. YOLOv8 may also provide bounding boxes which may not precisely fit the contours of the objects.
5. Training the models: Since the YOLO model was custom trained for the datasets, it required manual annotation of images which was time consuming and also had lots of computational challenges.

While these limitations are important, they can be easily avoided and tackled. If you possess a good state of the art computational resource then running these models in tandem to achieve even higher accuracy would be an easy feat.

5.2 Conclusion

To conclude, our research has presented with a novel approach and also taken a unique and new approach to solving the object counting with SAM problem. We can conclude that the overall results of the YOLO layer when combined with SAM are highly accurate and we can easily get comparable results to the segmentation results achieved by previous authors performing similar tasks.

We can also conclude that with 30 trained annotated images, with model trained on

400 epochs for the CARPK dataset the results achieved are better than the segmentation approach model results. The results achieved are comparable to the highly accurate and efficient Shi et al., 2022 work in Bilinear Matching Network. Similarly with 5 images per class for 80 classes when model is trained for 400 epochs, for the FSC-147 dataset we can get comparable results to the segmentation approach results. Hence we conclude that the minimum number of images required to get results comparable or better than the segmentation approach models are 20‘30 images for CARPK dataset and 5 images per class for 80 classes, i.e. 400 images for the FSC-147 dataset.

5.3 Future Scope

There are several different directions for the future for this research can go. One of the scopes can be to optimize the YOLO and SAM integration to solve the limitations. Use tiny-YOLO to train to solve the computational challenges. Tiny-YOLO is a smaller and lighter version of YOLO family that can perform similar tasks to its family members. This model was introduced to help edge computers and projects with computational difficulties achieve object detection tasks.

Conversely we can also go for the integration of different backbones, necks, and heads to customize the YOLO model to perform specific tasks. We can check whether the combination of this custom YOLO model with SAM and how well it performs on generic and specific datasets.

Try to train on a more generic dataset and achieve similar results. In this research we trained our YOLO model specifically for the FSC-147 and CARPK dataset. We can try to train the model on Omnicount-191(Mondal et al., 2024)or the FSOD dataset(Fan et al., 2020).

We can also try the segmentation model YOLO provides and try combining with SAM to perform counting. Try to annotate 10 images per class for the same 80 classes for the FSC-147 dataset and check the improvements in results and compare to the previous works. And we can also do the annotation of the FSC-147 dataset again and instead of the 80 classes approach taken by me, we can do the standard 147 classes or do even fewer classes and check how different the YOLO layer performs to this more generic classification of objects.

Bibliography

Cao, Haoyu et al. (2022). "GMN: generative multi-modal network for practical document information extraction". In: *arXiv preprint arXiv:2207.04713*.

Chu, Peng and Haibin Ling (2019). "Famnet: Joint learning of feature, affinity and multi-dimensional assignment for online multiple object tracking". In: pp. 6172–6181.

Demetriou, Demetris et al. (2023). "Real-time construction demolition waste detection using state-of-the-art deep learning methods; single–stage vs two-stage detectors". In: *Waste Management* 167, pp. 194–203.

Dosovitskiy, Alexey et al. (2020). "An image is worth 16x16 words: Transformers for image recognition at scale". In: *arXiv preprint arXiv:2010.11929*.

Fan, Qi et al. (2020). "Few-Shot Object Detection with Attention-RPN and Multi-Relation Detector". In: arXiv: 1908.01998 [cs.CV].

Girshick, Ross (2015). "Fast r-cnn". In: pp. 1440–1448.

Girshick, Ross et al. (2014). "Rich feature hierarchies for accurate object detection and semantic segmentation". In: pp. 580–587.

He, Kaiming et al. (2016). "Deep residual learning for image recognition". In: pp. 770–778.

He, Kaiming et al. (2017). "Mask r-cnn". In: pp. 2961–2969.

He, Kaiming et al. (2022). "Masked autoencoders are scalable vision learners". In: pp. 16000–16009.

Hsieh, Meng-Ru, Yen-Liang Lin, and Winston H Hsu (2017). "Drone-based object counting by spatially regularized regional proposal network". In: pp. 4145–4153.

Kirillov, Alexander et al. (2023). "Segment anything". In: pp. 4015–4026.

Liu, Shilong et al. (2023). "Grounding dino: Marrying dino with grounded pre-training for open-set object detection". In: *arXiv preprint arXiv:2303.05499*.

Ma, Chengji et al. (2023). "YOLO-UAV: Object Detection Method of Unmanned Aerial Vehicle Imagery Based on Efficient Multi-Scale Feature Fusion". In: *IEEE Access*.

Ma, Zhiheng, Xiaopeng Hong, and Qinnan Shangguan (2023). "Can sam count anything? an empirical study on sam counting". In: *arXiv preprint arXiv:2304.10817*.

Mercaldo, Francesco et al. (2022). "Blood cells counting and localisation through deep learning object detection". In: pp. 4400–4409.

Mondal, Anindya et al. (2024). "OmniCount: Multi-label Object Counting with Semantic-Geometric Priors". In: arXiv: 2403.05435 [cs.CV].

Osco, Lucas Prado et al. (2023). "The segment anything model (sam) for remote sensing applications: From zero to one shot". In: *International Journal of Applied Earth Observation and Geoinformation* 124, p. 103540.

Qureshi, Rizwan et al. (2023). "A Comprehensive Systematic Review of YOLO for Medical Object Detection (2018 to 2023)". In: *Authorea Preprints*.

Radford, Alec et al. (2021). "Learning transferable visual models from natural language supervision". In: pp. 8748–8763.

Ranjan, Viresh et al. (2021). "Learning to count everything". In: pp. 3394–3403.

Redmon, Joseph et al. (2016). *You only look once: Unified, real-time object detection*.

Ren, Peiming et al. (2020). "A novel squeeze YOLO-based real-time people counting approach". In: *International Journal of Bio-Inspired Computation* 16.2, pp. 94–101.

Shi, Min et al. (2022). "Represent, compare, and learn: A similarity-aware framework for class-agnostic counting". In: pp. 9529–9538.

Shi, Zenglin, Ying Sun, and Mengmi Zhang (2024). "Training-free object counting with prompts". In: pp. 323–331.

Tan, Mingxing and Quoc Le (2019). "Efficientnet: Rethinking model scaling for convolutional neural networks". In: pp. 6105–6114.

Wang, Chien-Yao et al. (2020). "CSPNet: A new backbone that can enhance learning capability of CNN". In: pp. 390–391.

Wang, Gang et al. (2023). "UAV-YOLOv8: a small-object-detection model based on improved YOLOv8 for UAV aerial photography scenarios". In: *Sensors* 23.16, p. 7190.

Wei, Zhiwei et al. (2020). "Amrnet: Chips augmentation in aerial images object detection". In: *arXiv preprint arXiv:2009.07168*.

Wu, Jian-Da et al. (2021). "Vehicle Classification and Counting System Using YOLO Object Detection Technology." In: *Traitement du Signal* 38.4.

Xu, Lingzhi, Wei Yan, and Jiashu Ji (2023). "The research of a novel WOG-YOLO algorithm for autonomous driving object detection". In: *Scientific reports* 13.1, p. 3699.

Yang, Shuo-Diao et al. (2021). "Class-agnostic few-shot object counting". In: pp. 870–878.

Yun, Sangdoo et al. (2019). "Cutmix: Regularization strategy to train strong classifiers with localizable features". In: pp. 6023–6032.

Zhang, Hongyi et al. (2017). "mixup: Beyond empirical risk minimization". In: *arXiv preprint arXiv:1710.09412*.

Zhu, Pengfei et al. (2019). "VisDrone-VID2019: The Vision Meets Drone Object Detection in Video Challenge Results". In: pp. 227–235. DOI: 10.1109/ICCVW.2019.00031.