

[einstein]

.

TRIANGULATING PRECISION: A COMPARATIVE STUDY OF MANUAL AND AUTOMATED ANNOTATIONS WITH YOLO, AZURE CUSTOM VISION AND GROUNDED SAM ON A CUSTOMIZED DATA SET FOR CREATION OF A PRODUCT FOR SAFETY OF RECYCLING INDUSTRIES

A Comparative Study of Manual and Automated Annotations with YOLO Model Training ,Azure Custom Vision,GroundingDINO and GroundedSAM,on a Customized battery Data set for creation of an end-end product for safety of recycling industries

Rida Aaliya

SUPERVISORS

Morten Goodwin,Per-Arne Andersen

University of Agder, 2024
Faculty of Engineering and Science
Department of Engineering and Sciences

Obligatorisk gruppeerklæring

Den enkelte student er selv ansvarlig for å sette seg inn i hva som er lovlige hjelpemidler, retningslinjer for bruk av disse og regler om kildebruk. Erklæringen skal bevisstgjøre studentene på deres ansvar og hvilke konsekvenser fusk kan medføre. Manglende erklæring fritar ikke studentene fra sitt ansvar.

1.	Vi erklærer herved at vår besvarelse er vårt eget arbeid, og at vi ikke har brukt andre kilder eller har mottatt annen hjelp enn det som er nevnt i besvarelsen.	Ja
2.	Vi erklærer videre at denne besvarelsen: <ul style="list-style-type: none">• Ikke har vært brukt til annen eksamen ved annen avdeling/universitet/høgskole innenlands eller utenlands.• Ikke refererer til andres arbeid uten at det er oppgitt.• Ikke refererer til eget tidligere arbeid uten at det er oppgitt.• Har alle referansene oppgitt i litteraturlisten.• Ikke er en kopi, duplikat eller avskrift av andres arbeid eller besvarelse.	Ja
3.	Vi er kjent med at brudd på ovennevnte er å betrakte som fusk og kan medføre annullering av eksamen og utestengelse fra universiteter og høgskoler i Norge, jf. Universitets- og høgskoleloven §§4-7 og 4-8 og Forskrift om eksamen §§ 31.	Ja
4.	Vi er kjent med at alle innleverte oppgaver kan bli plagiattkontrollert.	Ja
5.	Vi er kjent med at Universitetet i Agder vil behandle alle saker hvor det forligger mistanke om fusk etter høgskolens retningslinjer for behandling av saker om fusk.	Ja
6.	Vi har satt oss inn i regler og retningslinjer i bruk av kilder og referanser på biblioteket sine nettsider.	Ja
7.	Vi har i flertall blitt enige om at innsatsen innad i gruppen er merkbart forskjellig og ønsker dermed å vurderes individuelt. Ordinært vurderes alle deltakere i prosjektet samlet.	Nei

Publiseringsavtale

Fullmakt til elektronisk publisering av oppgaven Forfatter(ne) har opphavsrett til oppgaven. Det betyr blant annet enerett til å gjøre verket tilgjengelig for allmennheten (Åndsverkloven. §2).

Opgaver som er unntatt offentlighet eller taushetsbelagt/konfidensiell vil ikke bli publisert.

Vi gir herved Universitetet i Agder en vederlagsfri rett til å gjøre oppgaven tilgjengelig for elektronisk publisering:	Ja
Er oppgaven båndlagt (konfidensiell)?	Nei
Er oppgaven unntatt offentlighet?	Nei
Er utviklet programvare konfidensiell?	Ja

Acknowledgements

This Master's thesis was conducted at the University of Agder (UiA) in Grimstad, Norway. I, Rida Aaliya, from the Department of Information and Communications Technology (ICT), extend my gratitude to my professors Morten Goodwin and Per-Arne Andersen for their unwavering guidance and patience throughout this project. Special thanks to Mattias Sohl, the CEO of StellAI, for providing the dataset and necessary licenses. I express my appreciation to my colleagues at StellAI, Trygve Solberg and Kristoffer Sand. Additionally, heartfelt thanks to my husband, Vaseem Javad and my two kids for their support during late-night and weekend work sessions. I extend my heartfelt gratitude to my father and my late mother, whose unwavering motivation every day fueled my journey to complete my Master's degree. Their support and encouragement have been invaluable throughout this academic pursuit.

I express my gratitude to the various technologies and their knowledge documentation that played a crucial role in the completion of this project. A special thanks to OpenAI's ChatGPT, which I utilized for paraphrasing the entire thesis. The availability of this tool significantly contributed to the quality of my writing. I want to acknowledge that I did not copy any content directly from ChatGPT; instead, I utilized it solely for paraphrasing purposes.

Abstract

This thesis centers on the imperative task of detecting and segmenting batteries within the recycling industry, addressing the need for an efficient and accurate solution. The primary goal is to conduct a comprehensive comparison of state-of-the-art methods to discern the most suitable approach for the specified task. The comparative analysis extends to both manual and auto annotations, where manual annotations employ Roboflow, feeding data into Ultralytics' YOLOv5 and YOLOv8 models, as well as Azure Custom AI. Auto annotations leverage Grounded SAM and GroundingDINO, with Grounded SAM data integrated into YOLOv8 and YOLOv5. Remarkably, YOLOv8, combined with manual annotations for the custom battery dataset, demonstrates significant success. The thesis concludes by selecting the most effective method and enhancing a health dashboard based on the chosen model, providing a comprehensive and practical solution for the recycling industry's battery detection and segmentation challenges.

Contents

Acknowledgements	ii
Abstract	iii
List of Figures	viii
List of Tables	x
1 Introduction	1
1.1 Introduction	1
1.2 Motivation	2
1.3 Thesis Goal	2
1.4 Contributions	2
1.5 Hypothesis	3
1.6 Thesis Outline	3
2 Background	5
2.1 Deep Learning	5
2.1.1 Activation Function	6
2.1.2 Bias	8
2.1.3 Weights	9
2.1.4 Loss Functions	9
2.1.5 Propagation in Neural networks	10
2.2 Confidence Score:	10
2.3 Algorithms	11
2.3.1 Supervised Learning	11
2.3.2 unsupervised Learning	12
2.4 YOLO Architecture	12
2.4.1 Precision	12
2.4.2 Recall	12
2.4.3 Average Precision	12
2.4.4 mAP-Mean Average Precision	13
2.4.5 epochs and Batches	14
2.4.6 IoU	14
2.4.7 Bounding boxes:	17
2.4.8 Sigmoid weighted linear Unit (SiLu)	21
2.4.9 YOLOV5 Architecture	21
2.4.10 YOLOV8 Architecture	23
2.5 Roboflow	27
2.6 Azure Custom AI	29
2.7 Grounded SAM	29
2.8 GroundingDino	30
2.9 Streamlit:	30

2.10	LangChain and OpenAI API	30
3	Methods	31
3.1	Problem statement - Dataset <i>and proposed plan of action</i>	31
3.2	Problem statement - Detection <i>and proposed plan of action</i>	31
3.2.1	Experiment1	32
3.2.2	Experiment2	32
3.3	Problem statement -Segmentation and proposed plan of action	32
3.3.1	Experiment 3	32
3.3.2	Experiment 4	33
3.4	Problem statement - Health Dashboard <i>and proposed plan of action</i>	34
4	Related Literature	35
4.1	Azure Custom AI	35
4.2	YOLO-You Only Look Once	38
4.3	GroundingDIno	42
4.4	GroundedSAM	45
5	Performance Experiments and Results	47
5.1	Creation of Custom data set:	47
5.1.1	Objective	47
5.1.2	Steps Taken	47
5.1.3	Inferences	48
5.2	Comparison of image Detection Model using YOLOV5,YOLOV8 and Azure Custom AI	49
5.2.1	Objective:	49
5.2.2	Steps Taken	49
5.2.3	Inference:	56
5.3	GroundingDINO	65
5.3.1	Objective	65
5.3.2	Steps Taken	65
5.3.3	Inference	66
5.4	Image Segmentation using Roboflow,YOLO Models and Grounded SAM	69
5.4.1	Objective	69
5.4.2	Steps Taken	69
5.4.3	Inference	69
5.5	Health Dashboard and Chat bot	74
5.5.1	Objective	74
5.5.2	Steps Taken	74
5.5.3	Inference	74
6	Discussions	77
6.1	Methodology and Approaches:	77
6.1.1	Effectiveness of manual annotations using Roboflow	77
6.1.2	Impact of Auto annotations	77
6.1.3	Comparison of Results	79
6.2	challenges	79
6.3	Dataset Considerations	80
6.4	Future Directions and Improvements:	80
6.5	Integrations with Health Dashboard	81
6.6	Discussion on Contributions	82
6.7	Discussion on Hypothesis	82

7	Conclusions	83
A	Abbreviation	84
	Bibliography	85

List of Figures

2.1	Simple single layer Neural network	6
2.2	Binary Step function	7
2.3	Linear Activity Function	7
2.4	Sigmoid function	8
2.5	ReLu Function	8
2.6	Leaky ReLu	9
2.7	Loss Function	9
2.8	Propagation-Forward and Backward	10
2.9	Confusion Matrix	11
2.10	Average Precision	13
2.11	Numpy Calculation of Average precision from Scikit	13
2.12	Ground Truth vs Predicted image bounding box	15
2.13	Intersection Area	15
2.14	IoU formula	16
2.15	YOLO Bounding box calculation	17
2.16	Pascalvoc	18
2.17	Grid Box augmented from [35]	19
2.18	Convolution with 3x3 matrix Kernel	20
2.19	Striding	20
2.20	2x2 matrix Max pooling	21
2.21	YOLO Architecture from Ultralytics Docs - https://docs.ultralytics.com/yolov5/tutorials/architecture-yolov51-model-structure	22
2.22	BiFPN	22
2.23	Binary Cross Entropy	23
2.24	c2f and Bottleneck	24
2.25	SPPF Yolov8 emulated by [36]	25
2.26	Loss function in Yolov8	26
2.27	Roboflow	27
2.28	Roboflow Upload	27
2.29	Roboflow Annotation	28
2.30	GroundingDino emulated from [U]	30
3.1	Object Detection Module	32
3.2	Segmentation Module	33
3.3	Health dashboard	34
4.1	Timeline	36
4.2	CNN	37
4.3	AlexNet	38
4.4	Yolo Architecture- Timeline emulated from [19]	39
4.5	DETR	42
4.6	Timeline -Grounded SAM and GroundingDino	42
4.7	Grounding DiNO-Part I	44
4.8	GroundingDino-PartII	44

4.9	GroundingDino-III	45
4.10	Grounded SAM -General atchitecture	46
5.1	Trained generic dataset with generic image Predictions-1	47
5.2	Trained generic dataset with generic image Predictions-2	48
5.3	Recycling plant images with trained model	48
5.4	Recycling plant images with trained model of custom dataset	48
5.5	Experiment Approach	50
5.6	YOLOV5 vsYOLOV8 Box Loss	51
5.7	YOLOV5 vsYOLOV8 object Loss	52
5.8	YOLOV5 vsYOLOV8 class Loss	53
5.9	Confidence Score Yolov5 vs Yolov8	54
5.10	Overall Metric Performance in YoloV5 vs YoloV8l	55
5.11	Azure Custom AI Project creation	56
5.12	Azure Custom AI Predictionresult1	56
5.13	Azure Custom AI Predictionresult2	57
5.14	Azure Custom AI API	57
5.15	Azure Custom AI Dashboard	58
5.16	Validation of predicted Object Detection in YoloV5 and YOLOv8	60
5.17	Validation of predicted Object Detection in YoloV5 and YOLOv8	61
5.18	Bird's eye view validation of images in YoloV5 and YOLOv8	62
5.19	Bird's eye view validation of images in YoloV5 and YOLOv8	63
5.20	Precision and Recall Differences in Yolov5 and Yolov8 across 100 Epochs	64
5.21	GroundingDINO-Prediction1	67
5.22	GroundingDINO-Prediction 2	67
5.23	Grounding DINO-Prediction 3	68
5.24	Annotated Images from Roboflow	69
5.25	Instance Segmentation-Yolov8 Manually annotated images	70
5.26	Instance Segmentation-Yolov8 using GroundedSAM	70
5.27	Instance Segmentation-Yolov5 Using Manually annotated images	70
5.28	Instance Segmentation -Yolov5 using Grounded SAM	70
5.29	Instance Segmentation validation data of Yolov5,Yolov8 and GroundedSAM	73
5.30	Health Dashboard	75
5.31	chatbot with langchain	76
6.1	Roboflow workflow	78
6.2	RepViT-SAM	81

List of Tables

2.1	Roboflow Export model	29
4.1	Common Network Architecture and its Applications	36
5.1	Comparison of metrics of YoloV5detection model vs YOLOV8 detection models Azure Custom AI	58
5.2	Segmentation-Comparison	72

Chapter 1

Introduction

In this section, the backdrop of recycling industries is explored, emphasizing the necessity of implementing deep learning within this sector. The discussion then progresses to delve into the motivations, goals, contributions, and hypotheses that underpin the subsequent exploration.

1.1 Introduction

In Norway's recycling sector, a significant number of fire incidents occur annually, particularly within the period of January 2016 to May 2019, with 141 reported cases according to BRIS [1]. It is noteworthy that many minor fire incidents go unreported. The primary cause of these accidents is the incorrect sorting of batteries in recycling plants. An interview with a plant employee reveals that challenges primarily arise during the sorting of constructional waste materials, where smaller batteries such as AA, AAA, Quad A, and coin batteries become concealed within the waste, leading to potential hazards ranging from minor to severe.

StellAI, a startup, is employing deep learning and computer vision to segregate batteries from waste, aiming to mitigate fire risks. The primary goal of this project is to deliver a Proof of Concept (PoC) that supports recycling industries which explains as how this product made in PoC aims to enhance the sorting of batteries by implementing features that utilize the most suitable model tailored to the specific requirements of this startup. This involves the development and optimization of a customized dataset to ensure the effective functioning of the model within the context of the recycling industry's needs.

The project's scientific objective is to apply diverse models and various deep learning techniques to a dataset of nearly 300 custom images exclusively obtained from a recycling plant. The aim is to assess precision, develop a proof of concept with a model yielding a higher mean average precision value, and enhance the capability to detect smaller objects like batteries, especially from a 'bird's eye view.' Additionally, the project seeks to establish a dashboard for presenting statistical analyses, which will be displayed on monitors for employees. This dashboard aims to help employees understand patterns or trends in battery waste. Furthermore, the project involves utilizing LLM (presumably a type of machine learning model) to query custom data for gaining additional insights.

In this initiative, a diverse array of models and techniques is employed to identify a model that achieves a comprehensive mean average precision (mAP). Initially, the focus is on training models for object detection, utilizing various YOLO (You Only Look Once) models (both YOLOV5 and as well as YOLOV8). The training data is sourced from Roboflow, and the images undergo manual annotation to enhance the accuracy of the models. In the subsequent phase, the YOLO model undergoes training using automatically annotated images through the utilization of Grounding DINO. This approach involves leveraging Grounding DINO for the annotation process to enhance the efficiency of the model training along with instance

segmentation to train a model. Grounded SAM is utilized for this purpose, enhancing the training process. Subsequently, Azure Custom Vision is employed for detection tasks, providing a comprehensive framework for model development and evaluation.

Following the completion of all the experiments, a comprehensive comparison of the models is conducted. This assessment is based on several factors, including Mean Average Precision (mAP), Confusion Matrix values, and the time required for training each model. These metrics provide a holistic view of the models' performance, aiding in the identification of the most effective and efficient solution for sorting batteries in the recycling industry.

1.2 Motivation

Engaging in StellAI and gaining insights into the potential hazards posed by batteries in the recycling industry has served as a significant source of motivation. The prospect of mitigating accidents through the development of this product serves as a compelling drive for my efforts in this endeavor.

1.3 Thesis Goal

Custom Dataset

To establish a specialized dataset for batteries in a recycling industry, the following steps will be undertaken:

1. Utilize DRONE camera images for the purpose of object detection.
2. Employ high-resolution camera images to facilitate instance segmentation tasks.

Image Detection

- Manually annotate all custom images using the state-of-the-art tool Roboflow. Subsequently, the dataset is sent through the API to the Ultralytics portal for model training, conducting 100 epochs, and comparing the performance between YOLOv5 and YOLOv8.
- Train the model with the same dataset using Azure Custom AI and then compare its mean Average Precision (mAP) efficiency with that of YOLO.
- Test the dataset using GroundingDINO , a zero shot auto detector and observe the patterns in the results.

Instance Segmentation

- The dataset was sent to YOLOV5 and YOLOV8 for instance segmentation, and both models were trained for 50 epochs.
- Auto-annotated Grounded SAM results were used to train the YOLO model for 50 epochs, and the results were compared with a manually annotated model.

1.4 Contributions

The thesis makes several significant contributions to the field of object detection and segmentation, particularly in the context of the recycling industry. The key contributions can be summarized as follows:

1. Creation of a Custom Dataset:

- A curated dataset comprising 300 images has been meticulously crafted to address the unique challenges posed by the recycling industry.
- The dataset, including diverse scenarios and environments, has been made available on GitHub, serving as a valuable resource for further research and development in the domain.

2. Development of a Proof-of-Concept (PoC) Product:

- The thesis introduces a practical and innovative PoC product tailored for the recycling industry.
- This product leverages advanced deep learning techniques for the detection and segmentation of batteries in recycling processes, contributing to enhanced safety and efficiency.

3. Comparative Analysis of Annotation Techniques:

- A comprehensive comparative analysis has been conducted, evaluating the efficacy of state-of-the-art manual annotation methods against cutting-edge automated annotation technologies.
- The study includes a detailed examination of results obtained from GroundedSAM and GroundingDINO, shedding light on the strengths and limitations of these annotation approaches.

These contributions collectively advance our understanding of object detection and segmentation in recycling environments. The dataset and PoC product provide tangible resources for researchers and practitioners, while the comparative analysis offers insights into the evolving landscape of annotation methodologies.

1.5 Hypothesis

Null Hypothesis H0

"The accuracy and efficiency of battery detection and segmentation of custom dataset using advanced automated annotation techniques, specifically Grounded SAM and Grounding DINO, are significantly superior compared to traditional manual annotation methods."

Alternative Hypothesis H1

"The accuracy and efficiency of battery detection and segmentation of custom dataset using traditional manual annotation methods like Roboflow, are significantly superior compared to specifically Grounded SAM and Grounding DINO."

1.6 Thesis Outline

The thesis consists of the following topics in details

I **Background** - This topic explains about the following topics in detail which is applied as part of scientific experiment: A. Deep Learning - Overview, Activation Functions and Bias in Deep Learning are discussed. B. Algorithms in Supervised and Unsupervised Learning 1. Classification 2. Regression 3. Recommendations 4. Clustering 5. Generative Adversarial Networks (GANs) 6. Auto encoders C. YOLO Architectures where Evolution of YOLO Models and comparison of YOLOv5 vs. YOLOv8 architecture are covered D. Roboflow - where a role of Roboflow in Data Annotation and steps of Data Preprocessing are discussed. E.

Azure Custom AI - An Overview and Training Models with Custom Data F. GroundedSAM - An Introduction and Application Image Segmentation G. GroundingDINO -An Overview and about Object Detection with Text Prompts H. Streamlit and Langchain - A Frontend Development tool with Chatbot Integration is discussed

II. Methods

An architecture overview and problem statements with proposed action of plans like Dataset Creation, Annotation Strategies, Model Training Approaches and Evaluation Metrics are discussed in this chapter

III. Related Literature

Related scientific methods and about applications are discussed

IV. Experiment and Results

Experimentation of all plans proposed and results are discussed in this chapter

V. Conclusions and Discussions This chapter talks about Hypothesis review, Methodology and approaches, Challenges, Data set Considerations, Future directions and improvements

Chapter 2

Background

In this section, we will delve into the elements of deep learning, convolution networks, the YOLO architecture, GroundedSAM, GroundingDINO, and Roboflow models, as these are the fundamental components utilized in this project.

2.1 Deep Learning

Deep learning, a subset of machine learning, possesses the capability to process instructions akin to the human brain. For instance, when a human views an image of a car, regardless of the specific model, the ability to comprehend remains intact. In this process, neurons receive visual information from the eyes and the brain engages in processing. Similarly, deep learning emulates the human brain through artificial neurons. Deep learning employs representational learning, a process where feeding raw input into a system enables automatic generation of outputs for image detection and image segmentation[22]. Deep learning employs neurons to simulate a system for generating output.

Deep learning comprises of multiple neuron network where several calculations happens on layers including few hidden layers, and after the processing, the output is displayed, predicting a value. To comprehend the processes occurring between the layers, deep learning performs intricate calculations based on principles of linear algebra, Jacobian vectors, and Jacobian chain rules. In addition to extensive calculations, deep learning involves forward and backward propagation, essential mechanisms for adjusting and optimizing the network during the learning process.

Let's examine a simple example of how a convolutional neural network works.

In figure 2.1, x_1, x_2, x_3 are inputs, w_1, w_2 and w_3 are weights and N is a neuron network and σ is an activation function and b is called as bias and Let us see each component and its functionalities

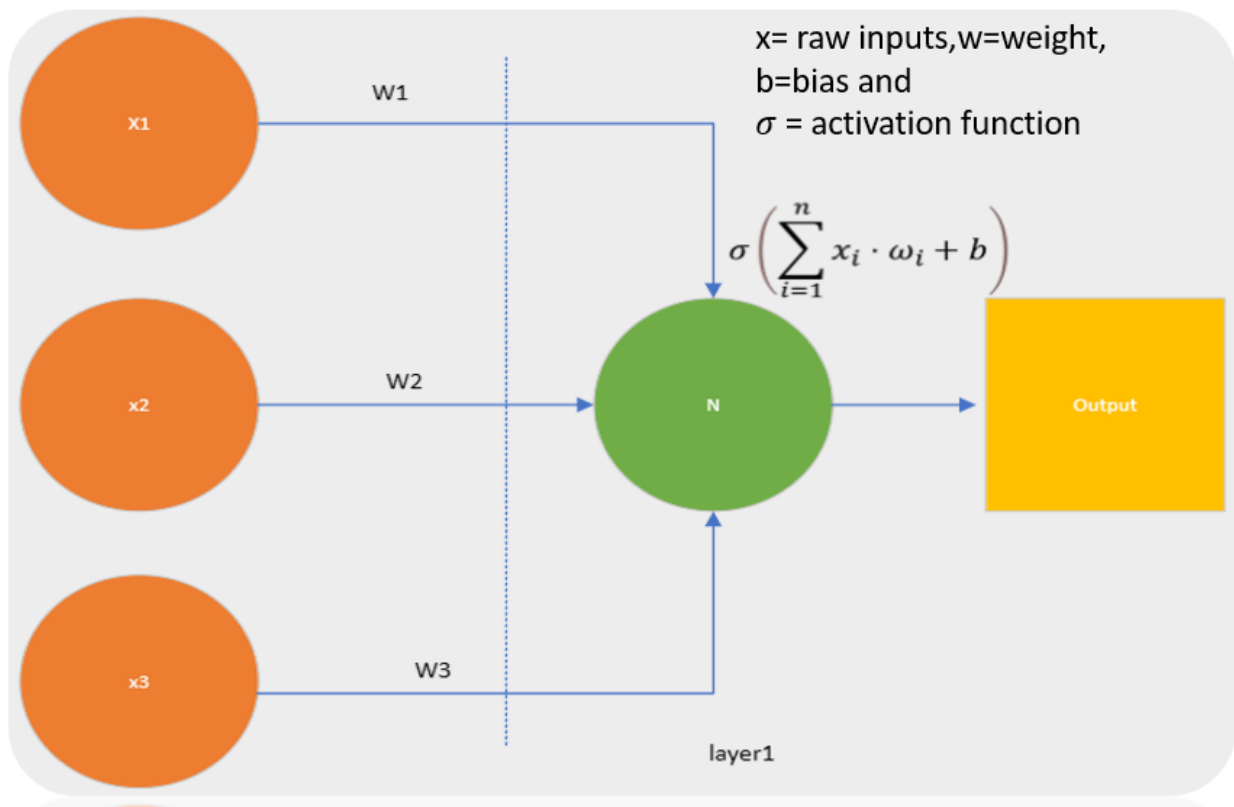


Figure 2.1: Simple single layer Neural network

2.1.1 Activation Function

In neural networks, when weights and biases are initialized with randomized values, there is a risk of the output ranging from negative infinity to positive infinity. If values become extreme, accurate predictions become challenging. Therefore, to ensure the system operates within a reasonable range of values, an activation function is employed. There are various activation functions available[32], and they can be categorized into different types. Some of these include:

Binary Step function:

Binary step function is the simplest activation methods used for classification problems where 0s and 1s are the output.[32] as shown in fig 2.2

Linear function:

This is yet another simple linear activation function which uses $f(x) = x + 3$ as shown in fig 2.3

Sigmoid function:

This is one of the non linear functions where output values fluctuates between 0 and 1 and mostly used in neural network as shown in fig 2.4

Relu Function:

Relu stands for rectified linear unit and also this model is popular in neural network because not all neurons are activated at a same time.there are many improvized versions of Relu such

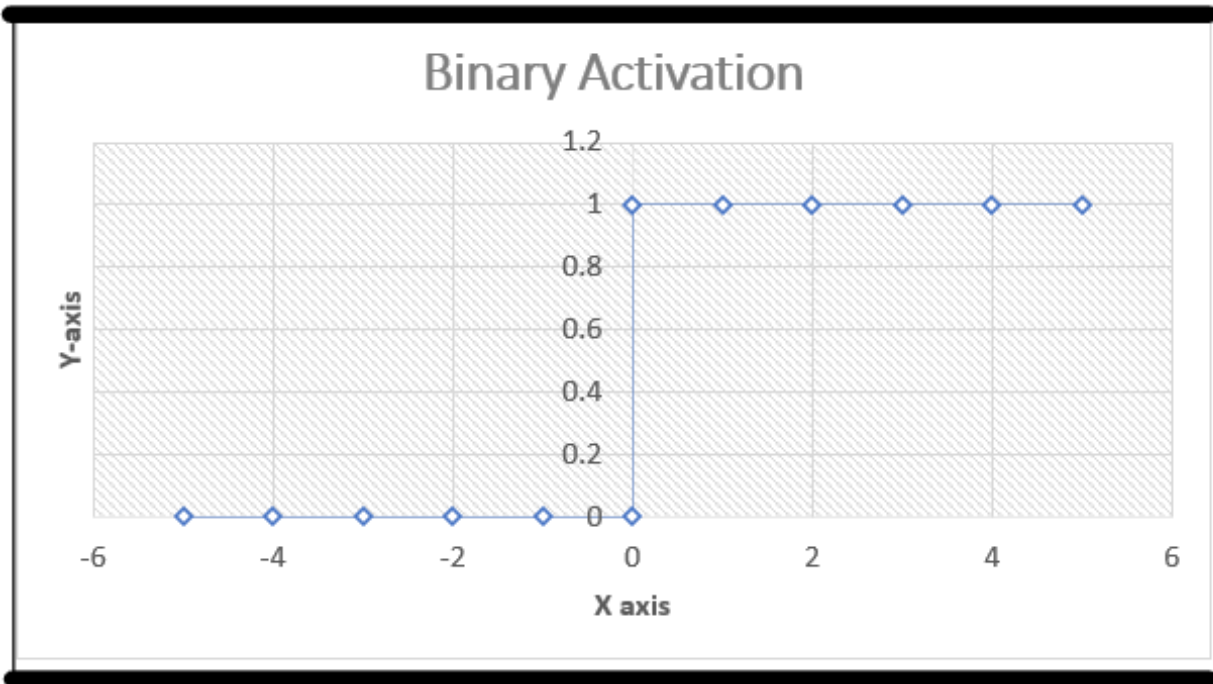


Figure 2.2: Binary Step function

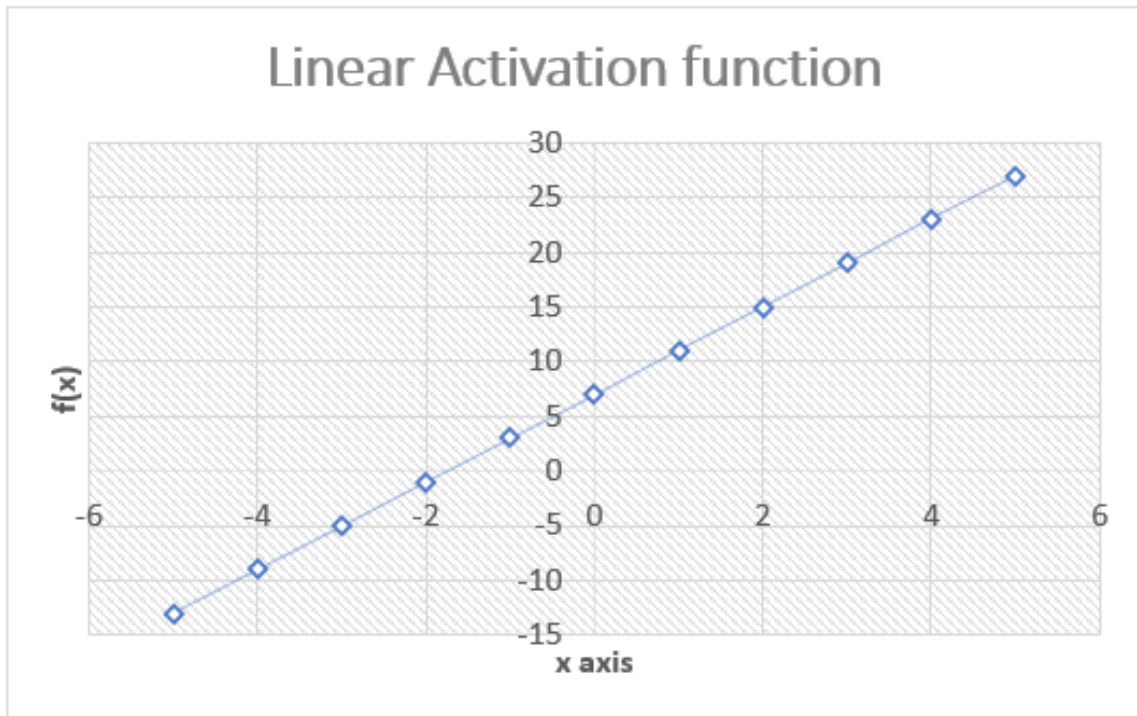


Figure 2.3: Linear Activity Function

as leaky Relu where unlike ReLu all negative values equating to zero like will be slightly lineated. as shown in fig 2.5 and fig 2.6 for Leaky ReLu

Softmax Function:

This is a combination of multiple Sigmoid functions which makes model easy to work for multi class classifications problems also.

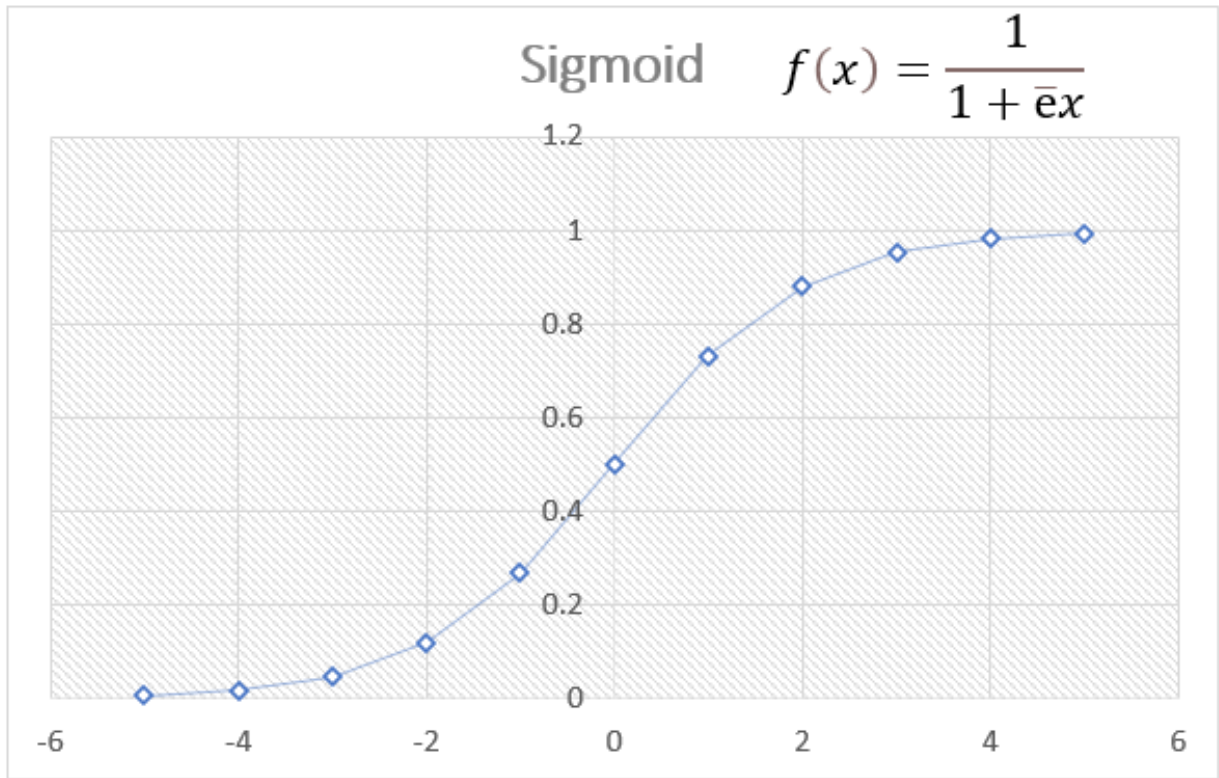


Figure 2.4: Sigmoid function

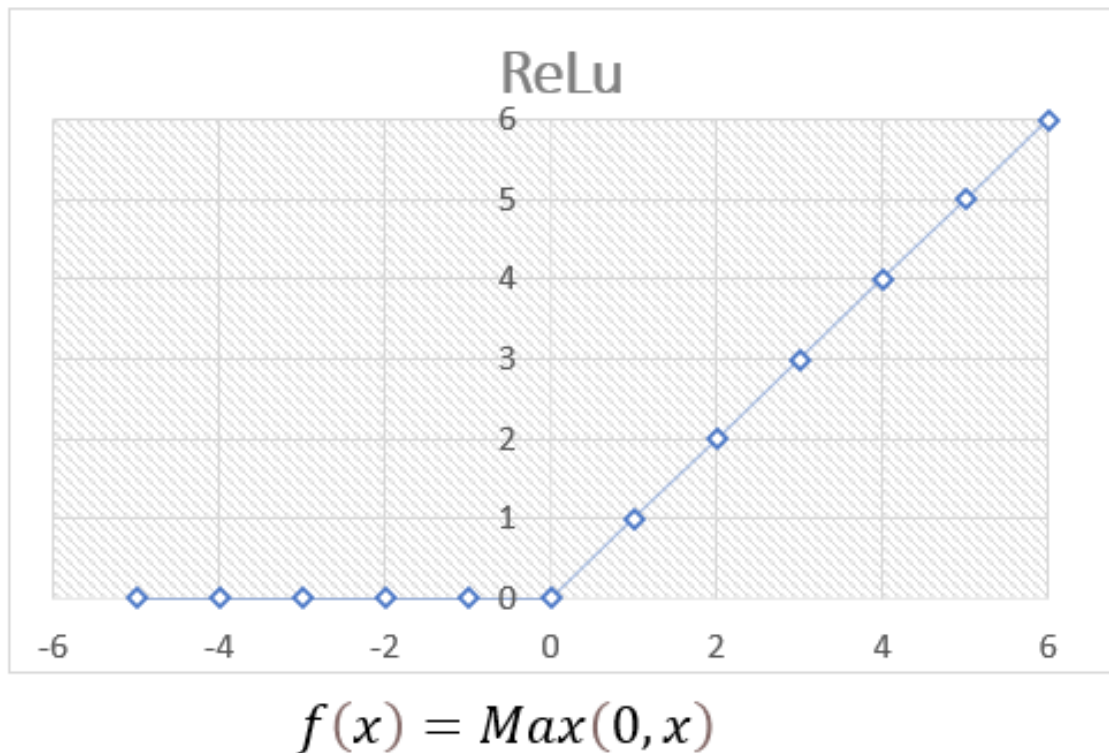
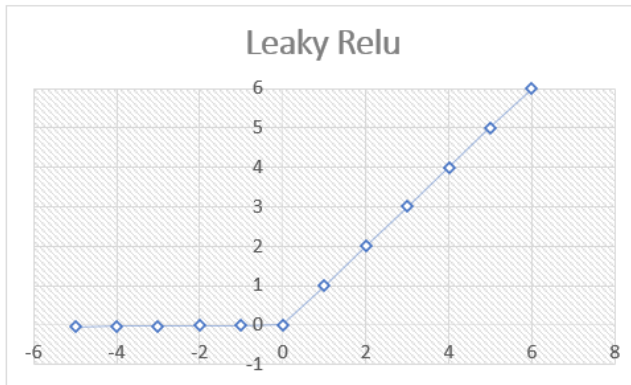


Figure 2.5: ReLU Function

2.1.2 Bias

Bias is a constant value that is introduced before providing information to a neuron, specifically as the product of the input and weights. In analogy, bias can be likened to a constant



$$f(x) = ax \quad x < 0$$

$$f(x) = \text{Max}(0, x), \quad x > 0$$

Figure 2.6: Leaky ReLU

$$\text{Mean Squared error} = \frac{1}{n} \sum_{l=1}^n (y - y_i)^2$$

n = Number of iterations
 y = original output
 y_i = predicted output

$$\text{Root Mean Squared Error} = \sqrt{\frac{1}{n} \sum_{l=1}^n (y - y_i)^2}$$

Figure 2.7: Loss Function

term added in a linear function. Bias is a scalar value and generally denoted by b or b^l . The primary rationale behind adding bias to the system is to ensure that even if the input value is zero, a neuron can still be activated through the addition of a constant value. This helps introduce flexibility and allows neurons to respond to different input conditions.

2.1.3 Weights

Weights play a crucial role in determining the output of a neuron. The product of weights and raw inputs, combined with the activation function, significantly influences the prediction of outputs. Various methods exist for determining how weights are assigned and adjusted in a neural network. One type of weights in a neural network is Neural Network with Random Weights (NNRW), [17] where the weights are randomly adjusted at each layer during training in an unsupervised manner. On the contrary, there is the Backward Propagation (BP) technique [17], where weights are selected based on the outputs and loss functions. Backward Propagation (BP) is known for being memory-intensive and requiring intensive calculations compared to the Neural Network with Random Weights (NNRW) approach.

2.1.4 Loss Functions

Loss functions play a vital role in neural networks by aiding in the retraining of a model to approach the desired output. These functions calculate the difference between the predicted output and the original output, providing a measure of the disparity. In the Backpropagation model, the weights are subsequently adjusted through Jacobian Differential calculations based on this difference. Various types of calculations are employed in neural networks, with Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) being commonly used methods. The formula is given in fig 2.7

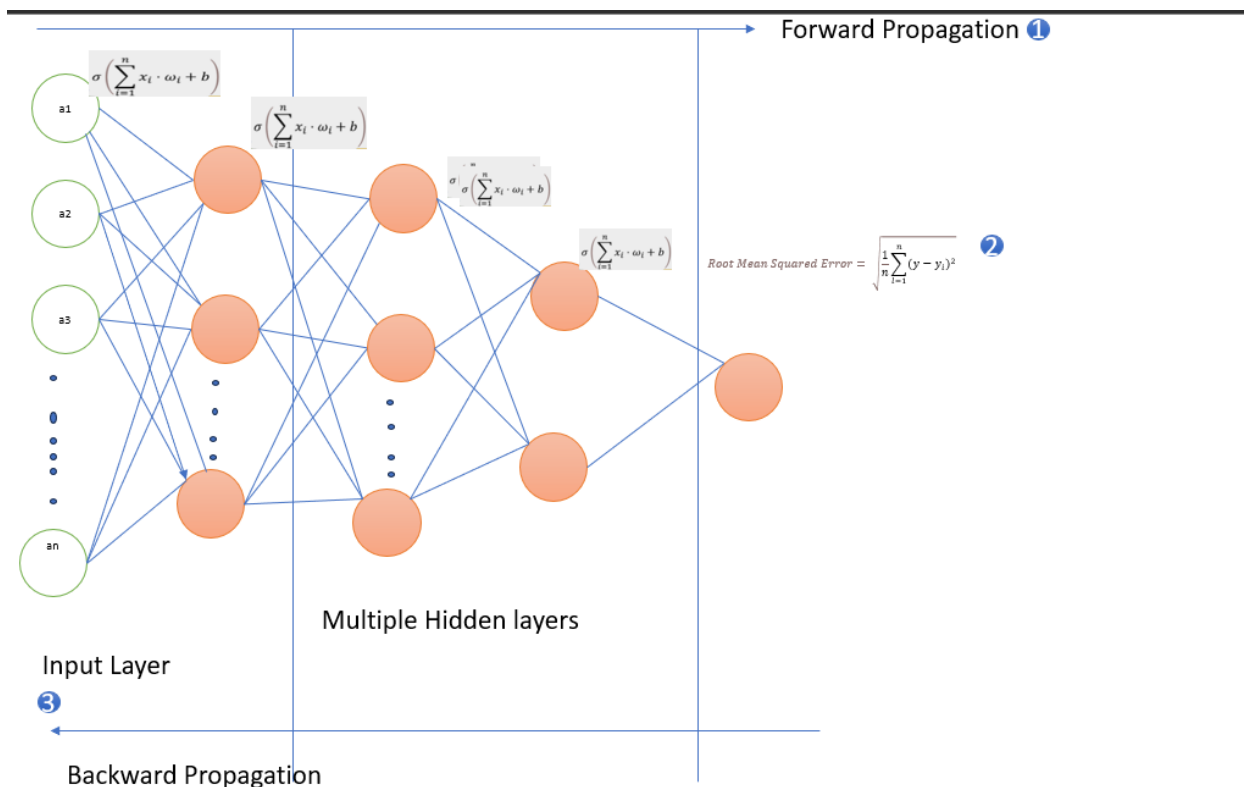


Figure 2.8: Propagation-Forward and Backward

2.1.5 Propagation in Neural networks

Backward propagation is a method in neural networks where a model is trained based on the loss function. It assigns weights progressively to an input layer through multiple hidden layers to minimize the error. Subsequently, using the updated weight values assigned by the backward propagation model, Forward propagation is performed again to retrain with new values. The process is recursive, involving the calculation of loss values and taking measures accordingly. This iterative process continues until the loss value reaches an acceptable threshold.[23] as shown in fig 2.8

2.2 Confidence Score:

Similar to humans, machines incorporate a level of confidence in predicting outputs. For instance, when detecting batteries in a recycling industry, if the images of batteries are small, humans might misinterpret objects as batteries or vice versa. There are instances when humans assign a probability to objects, considering them as potential batteries. Similarly, machines predict objects along with a confidence percentage. For example, if Image detection indicates a 90% confidence level, the machine is highly certain that the object is a battery. Conversely, if the confidence percentage falls below 30%, the machine may consider the object less likely to be a battery. Indeed, training a model through feedback is invaluable in refining machine learning systems. Humans can consistently provide data and monitor the confidence percentage exhibited by the machine. This iterative process allows for ongoing improvement over time, enabling the machine to enhance its accuracy and reliability in predictions. Certainly, when dealing with a large number of images, manually checking all outputs becomes impractical. To address this, a systematic approach is developed where the machine is trained with a subset of the total images, say 70%, while 20% is reserved for testing the model's performance, and the remaining 10% is allocated for validation purposes. This division allows for effective training, testing, and validation, ensuring

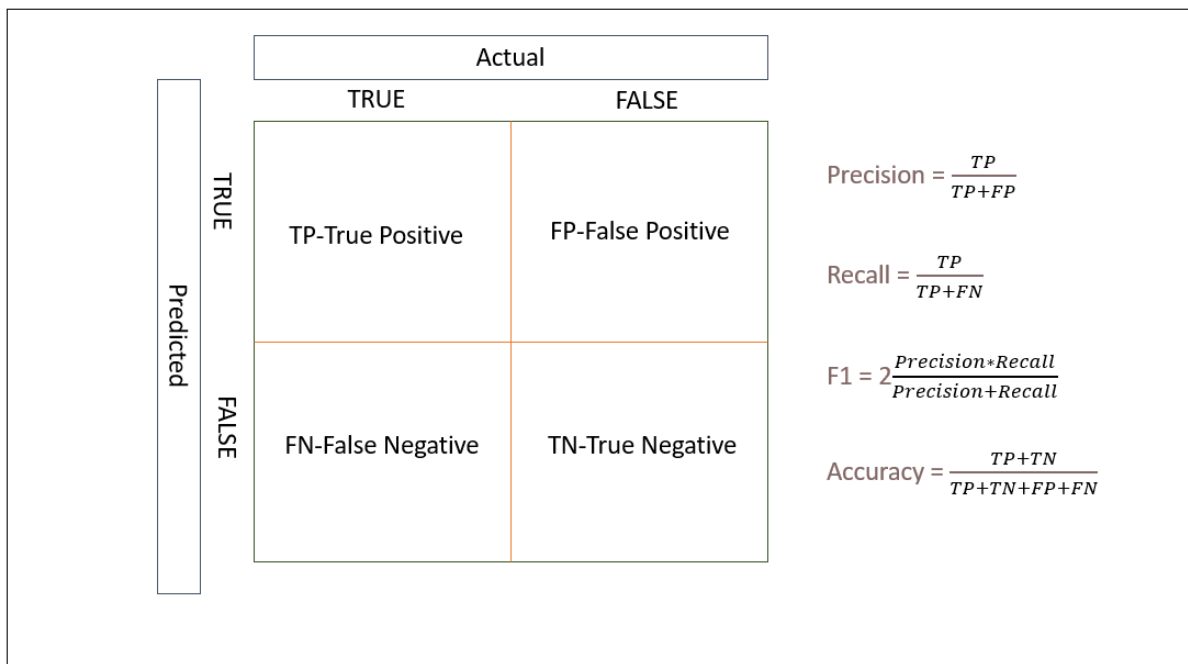


Figure 2.9: Confusion Matrix

the model’s robustness and generalization to unseen data. In this process, it is crucial for machines to communicate with humans during testing to report how many predictions align with the actual objects, referred to as True Positives. This information is vital for assessing the model’s accuracy and effectiveness in recognizing the intended objects. As illustrated in the figure 2.9, the confusion matrix comprises four values: True Positive, True Negative, False Positive, and False Negative. These metrics provide a comprehensive assessment of the model’s performance in binary classification tasks.

2.3 Algorithms

In Machine learning, there are various types of mechanisms as how machine learns and predicts outputs. Most important types being Supervised Learning, unsupervised learning, semi-supervised learning, Reinforcement learning

2.3.1 Supervised Learning

Supervised learning can be likened to the process of teaching a child. Similar to showing a child diagrams of cars, trucks, and planes and subsequently testing their knowledge by asking questions about cars on roads, supervised learning involves training a model with predefined labels using training data. The model is then tested with various images, and its predictions are compared with the expected outputs for evaluation and refinement. Supervised learning is broadly divided into two primary categories: Classification and Regression. Classification involves scenarios where the expected output is discrete, such as determining whether an email is spam or not. On the other hand, regression is applied in time series analysis when the time series is continuous. For instance, it is commonly used in predicting stock prices, which exhibit continuous changes over a period of time. One of the best examples of regression is YOLO model which will be explained further in section..

2.3.2 unsupervised Learning

In general, supervised learning is effective when the expected output is well-defined, and the input features are known. This makes the process of labeling data more straightforward and facilitates the training of models to make accurate predictions based on the provided labeled examples. However, in situations like anomaly detection, where the input is not well-defined, and the characteristics cannot be precisely defined, unsupervised learning proves to be more effective. Unsupervised learning allows the model to identify patterns or anomalies in the data without the need for labeled examples, making it well-suited for scenarios with unclear input characteristics. Unsupervised learning is applied in clustering scenarios, such as the K-Means clustering algorithm. In clustering, similar categories are grouped together into clusters, and the algorithm calculates the mean distance within these clusters to predict output values. This approach allows the model to identify inherent patterns or groupings within the data without predefined labels. Association is another type of unsupervised learning where a set of common characteristics is associated to predict the output. For instance, the recommendation systems used by platforms like Prime Video utilize association rules, suggesting movies based on patterns such as "People who saw this movie also liked other movies like these." This approach helps predict user preferences without explicit labeling of the data. In this thesis, a combination of both supervised and unsupervised algorithms will be employed and comparison will be made on the outcomes.

2.4 YOLO Architecture

Before delving into its architecture, it's crucial to comprehend the following terms associated with YOLO (You Only Look Once).

2.4.1 Precision

Precision is calculated as the ratio of True Positives to the sum of True Positives and False Positives. For instance, if there are 10 objects, and 8 are True Positives (correctly detected) while 2 are False Positives (incorrectly detected), then the precision is calculated as $8 / (8 + 2) = 0.8$. This implies that 8 out of the 10 detections were accurate, and 2 were incorrect. For a model to perform effectively, it is desirable for the precision to be higher.

2.4.2 Recall

Recall is calculated as the ratio of True Positives to the sum of True Positives and False Negatives. For example, if there are 10 objects, and there are 8 True Positives (correctly detected) and 2 False Negatives (missed detections), then the Recall is calculated as $8 / (8 + 2) = 0.8$. This implies that the model missed detecting 2 out of the 10 actual objects, indicating a proportion of the objects that were not identified. For a model to perform well, it is preferable for the recall percentage to be higher.

2.4.3 Average Precision

Average Precision is a metric employed when mean of precisions obtained at each threshold and it is computed as shown in the figure 2.10 [2][18]

$$AP = \sum_{k=0}^{n-1} [(Recall(k) - Recall(k + 1)] * Precision(k)$$

n =number of threshold

Recall(k) = Recall value of kth image

Recall(k+1) = Recall value of (k+1)th image

Rule:

Recall value to be appended by 0.0 in an array and Precision value must be appended with 1.0 in an array

Recall = [0.7,0.5] → [0.7,0.5,0.0]

Precision=[0.7,0.8] → [0.7,0.8,1.0]

Calculation:

Threshold = 0.5

Values > 0.5 = 1

Values <= 0.5 =0

AP = {[0.7-0.5]*0.7 + [0.5-0.0]*0.8} =[0.2*0.7]+[0.5*0.8] =0.54

Figure 2.10: Average Precision

2.4.4 mAP-Mean Average Precision

The average precision is the mean value calculated when multiple objects in an image are detected, and there are three classes. In this scenario, individual average precisions (APs) are computed separately for each class, and then the mean of these APs is calculated. To provide a straightforward example, if the average precisions (APs) for three classes are 0.91, 0.92, and 0.96, then the mean average precision (mAP) is calculated as the sum of these values divided by the total number of classes: $mAP = (0.91 + 0.92 + 0.96) / 3$. [fig 2.11]

```
In [11]: import numpy as np
         from sklearn.metrics import average_precision_score
         y_true = np.array([1, 0 ])
         y_scores = np.array([1, 1])
         average_precision_score(y_true, y_scores)

Out[11]: 0.5
```

Figure 2.11: Numpy Calculation of Average precision from Scikit

2.4.5 epochs and Batches

When dealing with a dataset of 1000 image samples, attempting to feed all images into a neural network can strain computing capacity. This is because the calculations involved in gradient descent, particularly for backward propagation, are computationally intensive. The resources required for processing such a large dataset in one go can exceed the available computational capacity, leading to potential performance issues. Instead of handling 1000 images simultaneously, they can be segmented into batches. For instance, if a batch size of 20 images is selected in the given example, it would result in 50 batches of data. Conversely, a single epoch is regarded as complete when a neural network processes through all 50 batches. If a neural network is set with 100 epochs, it indicates that the program has finished training on the dataset of 1000 images, organized into 50 batches, for a total of 100 cycles. The purpose behind this approach is to guide the program in minimizing errors during the training process. Following each epoch, the YOLO model provides values for recall, precision, and mAP. [16]

2.4.6 IoU

In YOLO image detection, images are recognized using bounding boxes. After the model is trained, images are sent through the network for validation. If the bounding box of the prediction perfectly aligns with the ground truth box, the Intersection over Union (IoU) is considered to be 100 percent, indicating that the prediction precisely matches the trained value. The IoU is computed using a formula that involves the ratio of the intersection area over the union of the bounding boxes. For example, if a threshold value is set at 50 percent, denoted as 0.5, and the IoU is 0.7, then the result is positive, indicating successful object detection. Conversely, if the IoU is 0.3, the object detection is considered unsuccessful. Hence, threshold is a critical factor in model detection; when it's set to higher values and the mAP remains elevated, it indicates that the model is effective, given the higher IoU values. [18]. It's explained in a figure 2.12, 2.13 and 2.14

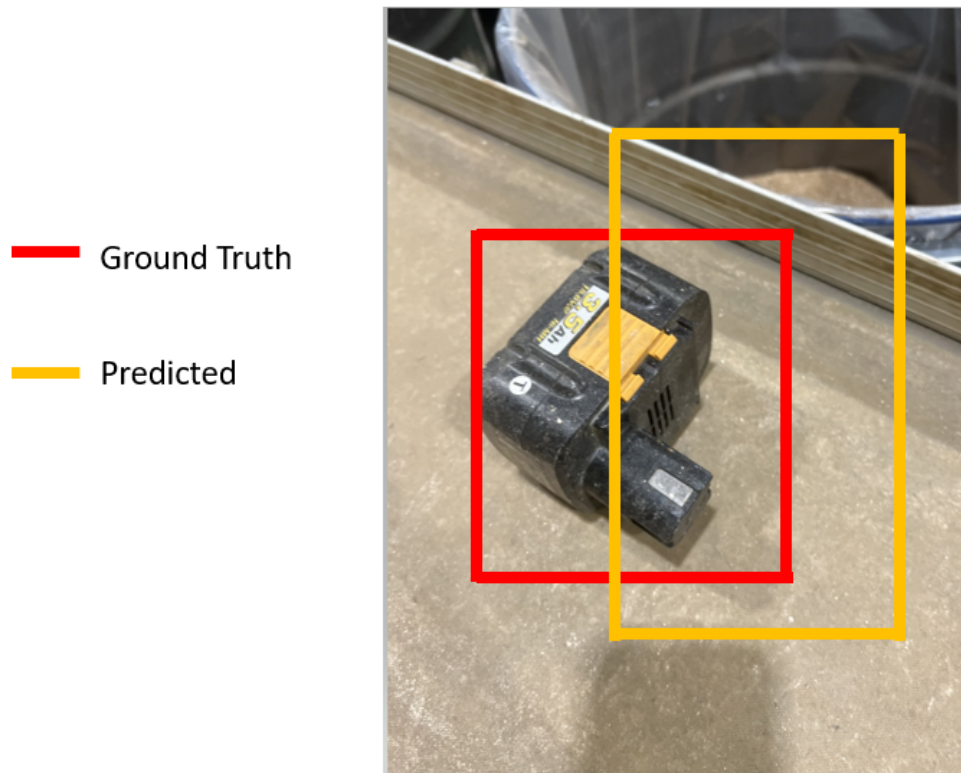


Figure 2.12: Ground Truth vs Predicted image bounding box

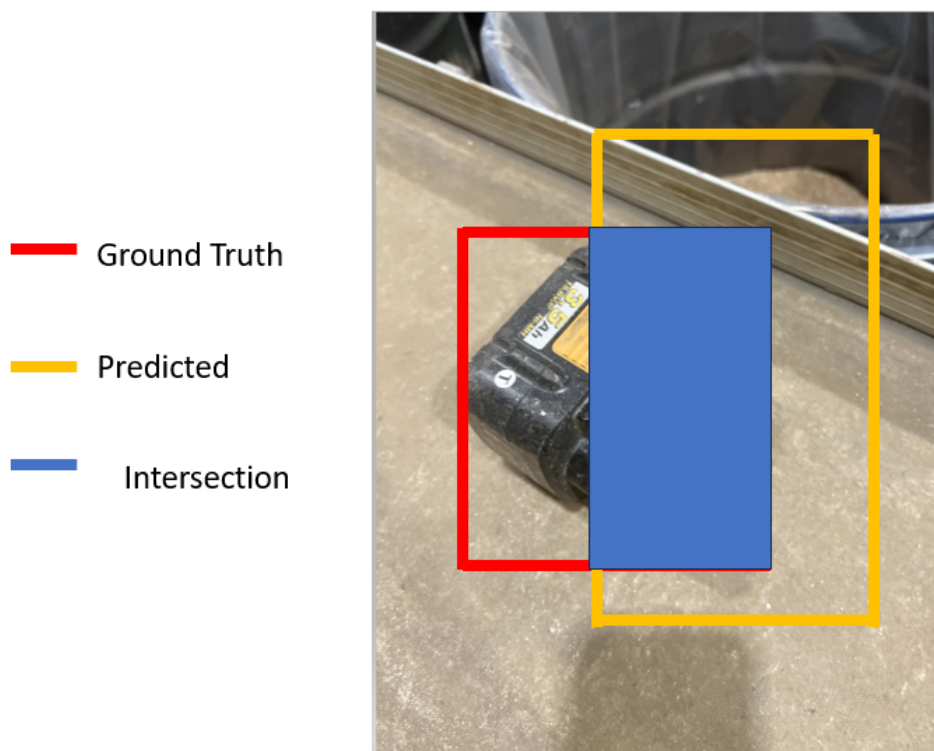


Figure 2.13: Intersection Area

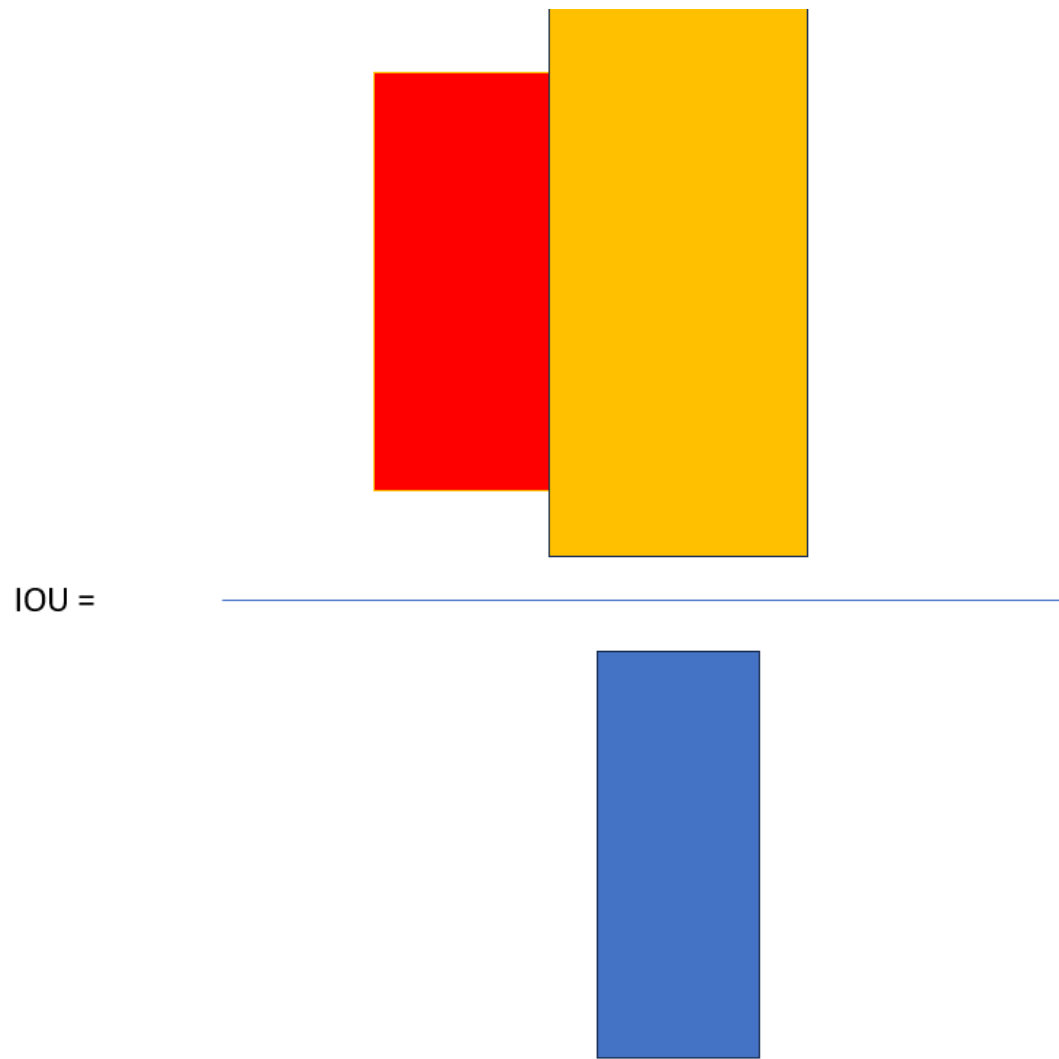


Figure 2.14: IoU formula

Bounding box –YOLO =Normalized(x_center,y_center,width,height)

X_center= [(486+86)/2/640] =0.4468
Y_Center=[(257+112)/2/480]=0.384
Width =400/640=0.625
Height=145/480 =0.302
=[0.4468,0.384,0.625,0.302]

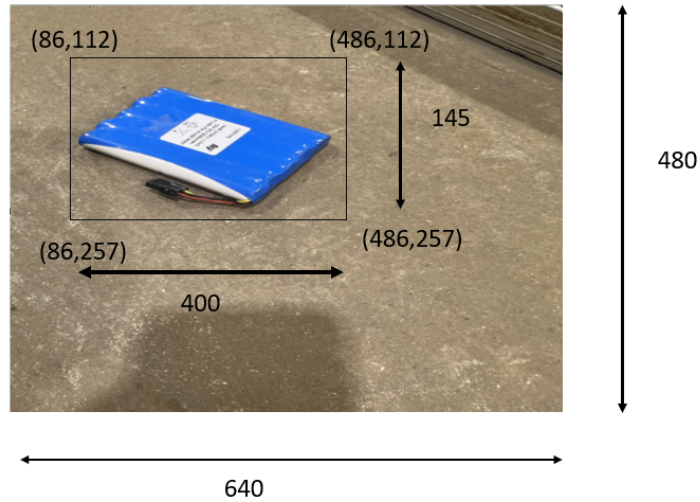


Figure 2.15: YOLO Bounding box calculation

2.4.7 Bounding boxes:

YOLO Bounding box

The bounding box is a fundamental component of image detection in the YOLO model. In YOLO, the coordinates are computed in the following format: (x-center, y-center, width, height), as illustrated in the figure 2.15

In the illustration, the x, y coordinates for the top-left, top-right, bottom-left, and bottom-right corners of the rectangle are (86,112), (486,112), (86,257), and (486,257) respectively. The x_center can be determined by averaging the x-axis values [(486+86)/2] and then dividing by 640, given the image resolution of 640x480. Similarly, the y-center is calculated by averaging the y-coordinates [(257+112)/2] and then dividing by 480. The width is determined as the length of the rectangle, which is 400 divided by 640, and the breadth is considered as the height of the rectangle, which is 145 divided by 480. [14]

Pascal VOC

There is a second type of bounding box known as the PASCAL_VOC XML format, employed in Grounded SAM. In this format, a PASCAL_VOC XML file is generated for each image, recording dimensions in the form [x_min, y_min, x_max, y_max]. For the provided diagram, the PASCAL_VOC coordinates would be [86, 112, 486, 257]. The coordinates for the PASCAL_VOC format for the diagram below are illustrated in the diagram.image1 .jpg is a file which is annotated and database it is stored in roboflow.ai and width ,height and depth is also mentioned and bounding box for one of the cylindrical batteries are mentioned as [441,838,511,920][14] as shown in the figure 2.16

One hot encoding

The one-hot encoder, or one-hot encoding method, transforms variables into a binary format of 1s and 0s, indicating the presence or absence of a particular entry. For example, in this project there are 5 classes of data is detected and they are represented as follows:

['cylindrical battery', 'Rectangular battery', 'Industrial Battery', 'wires-cables', 'coin battery'] are classes and they are encoded as follows:

```

<?xml version="1.0" ?>
<annotation>
  <folder>VOC</folder>
  <filename>image1.jpg</filename>
  <source>
    <database>roboflow.ai</database>
  </source>
  <size>
    <width>1080</width>
    <height>1350</height>
    <depth>3</depth>
  </size>
  <segmented>0</segmented>
  <object>
    <name>cylindricalbattery</name>
    <bndbox>
      <xmin>441</xmin>
      <ymin>838</ymin>
      <xmax>511</xmax>
      <ymax>920</ymax>
    </bndbox>
  </object>
</annotation>

```

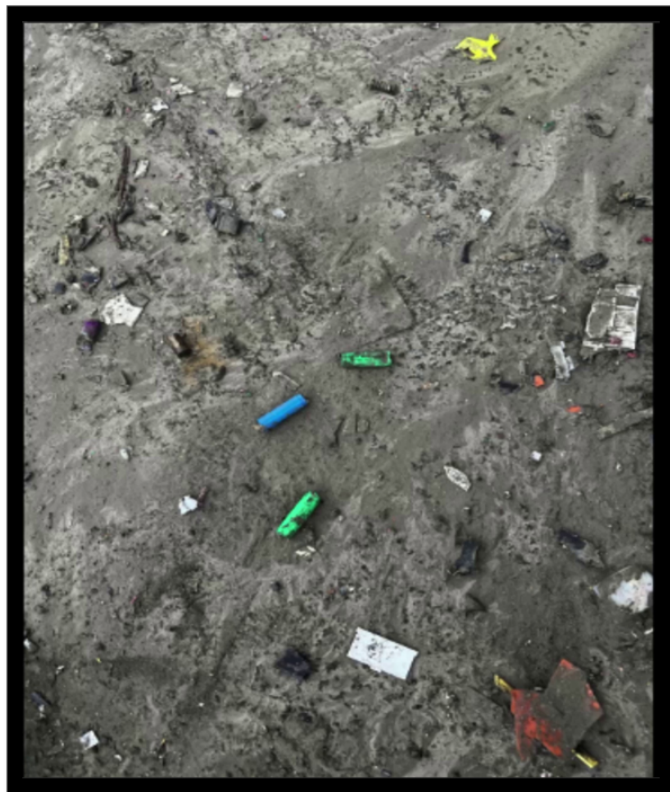


Figure 2.16: Pascalvoc

- 1.'Cylindrical battery' = [1,0,0,0,0]
- 2.'Rectangular battery' = [0,1,0,0,0]
- 3.'Industrial battery' = [0,0,1,0,0]
- 4.'Wires-Cables' = [0,0,0,1,0]
- 5.'Coin Battery' = [0,0,0,0,1]

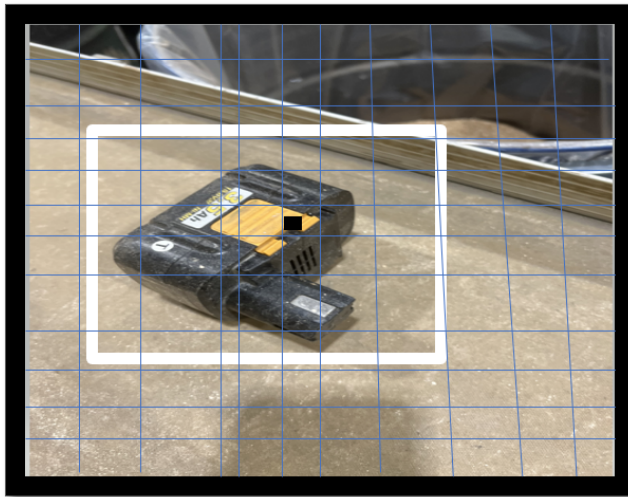
Grid Boxes

Images must be segmented into smaller sections to be input into the neural network. For instance, as depicted in the figure below, if the image has a resolution of 640x480, it is transformed, let's say, into a 640x640 image resolution (the specific size depends on the YOLO model). This image is then divided into a grid, and assuming an 8x8 grid, each box will contain 80 pixels. as shown in fig 2.17

The center point of the truth box is identified, and the distance from the center point of the truth image to the boxes is established, with the difference being computed. The difference from the midpoint is computed using the formulae: $(x - x1)/80$, $(y - y1)/80$, $w/640$, $h/640$, where 80 is the grid size, and width and height are calculated based on the image resolution, which is 640x640 and along with 4 parameters, the 5th parameter i.e class objectness score, a presence of object is calculated will be added and along with that (x,y,w,h) [35]

$(x1, y1, w1, h1, c1) = [(x\text{ordinatedistancebetweenthegridboxandthecenterpointoftruthbox})$
 $, (y\text{ordinatedistancebetweenthegridboxandthecenterpointoftruthbox})$
 $, (widthoftruthbox/resolutionofanimage)$
 $, (heightoftruthbox/resolutionofanimage)$
 $, classobjectiveness]$

In addition to the initial 5 parameters, a 6th parameter, Probability, is introduced, incorporating the probabilities of the 5 classes. For instance, if the probability values are [0.035, 0.00, 0.86, 0.065, 0.04], and a threshold is applied, the grid box will be associated with an im-



Input Image = 640* 480

Image Size_resized by YOLO = 640*640

$S*S = 8*8$ (to determine Grid size each blue box in a figure)

Each grid = $\frac{640*640}{8*8} = 80*80$ grid boxes in a picture

White box is a ground truth box

Figure 2.17: Grid Box augmented from [35]

age related to the industrial battery, as 0.86 represents the probability of class 3, "industrial battery," as assumed in the "one hot encoding" section.

Following this example, if it is assumed that there are 2 bounding boxes for each grid, then: $(2 * 5) + 5 = 15$ parameters per grid

- 2: 2 bounding boxes
- 5: 5 parameters (x, y, w, h, c)
- 5: Probability of 5 classes

Based on this calculation, the prediction vector for each grid should be $80 * 80 * 15$, where $80 * 80$ represents the grid boxes, and 15 represents the parameters for each grid.

Hence, the total prediction vector will be 6400 grid boxes * 15 parameters. If there are multiple bounding boxes created for a single image, all the bounding boxes are evaluated based on the probability and class objectness.

Kernel,Stride,Padding ,Pooling and Flattening

Modern images often have higher resolutions, and sending such images directly to a fully connected neural network can result in an excessively large single array input size. This can lead to increased computational power consumption. To mitigate or modify the resolution, a kernel is employed. This kernel acts as a filter with randomized values and is adjusted based on the back propagation loss values. To illustrate, let's consider a simplified example of a 5x5 matrix image. Using a 3x3 kernel, convolution begins by applying the 3x3 kernel to the first 3x3 matrix in the image, resulting in a single value. Next, the convolution process continues by applying the 3x3 kernel to the highlighted portion from the first row and second column to the first row and fourth column, extending to the third row and fourth column. This results in another value. Similarly, the convolution process is repeated nine times, each time applying the 3x3 kernel to a different portion of the 5x5 matrix, resulting in nine values as illustrated in the figure 2.18 and fig 2.19 [27]

In the scenario described above, if an mxm matrix image is convoluted with an nxn matrix kernel, then the output dimension would be $[(m-n)+1]$. In this case, it is $[5-3]+1 = 3$.

If further reduction of the image output is desired, striding can be applied. Striding is a process where, instead of performing sequential convolution, steps are skipped, and this is referred to as 2-step striding.



Figure 2.18: Convolution with 3x3 matrix Kernel

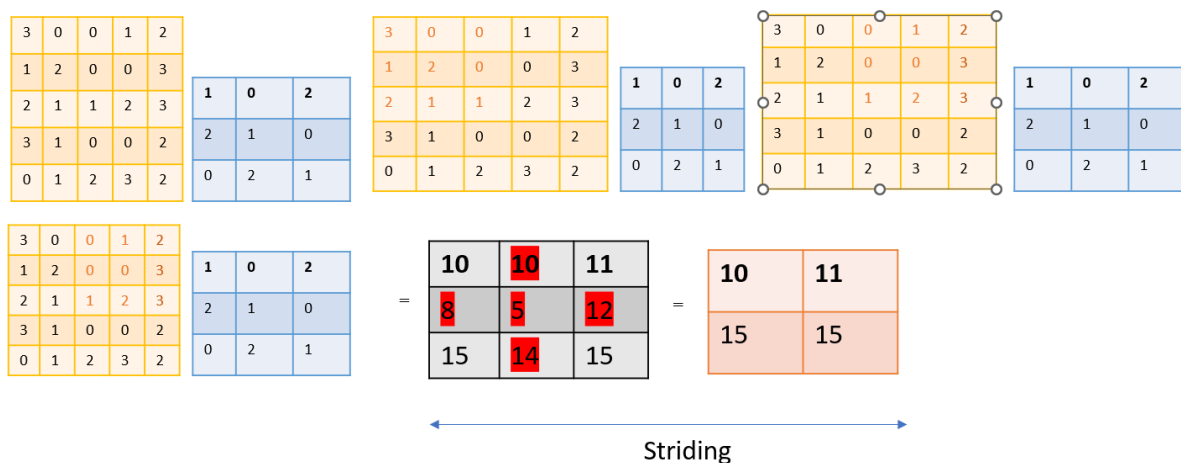


Figure 2.19: Striding

In striding, values are utilized only once. For example, if the first 3x3 matrix of a 5x5 image is convoluted, it then moves to the last 3x3 matrix. As a result, there are only 4 convolutions happening. In other words, in the output from convolution in the last example, the values in the 2nd row and 2nd column are not necessary anymore because those multiplications are skipped in the 2-step striding, resulting in a 2x2 matrix. If there is an $m \times m$ image resolution and $n \times n$ kernel, and 2-step striding is to be applied, the output can be calculated using the formula $\lfloor (m-n)/s \rfloor + 1 = \lfloor (5-3)/2 \rfloor + 1 = (2/2) + 1 = 2$. Therefore, the result will be a 2x2 matrix, as illustrated in the figure below.

To maintain border information during convolution, **Padding** can be applied, which involves adding additional 0s around the matrix. For instance, in an example above 5x5 matrix will become 7x7 matrix because of additional 0s across the border and if a 7x7 matrix is obtained through padding and then convoluted with a 3x3 kernel, the output will be $(7-3)+1 = 5$, preserving the 5x5 matrix information.

Max Pooling Max pooling is employed to retain essential information from an image while simultaneously reducing its size. This method, such as max pooling, is commonly utilized in YOLO. refer fig 2.20 [3]

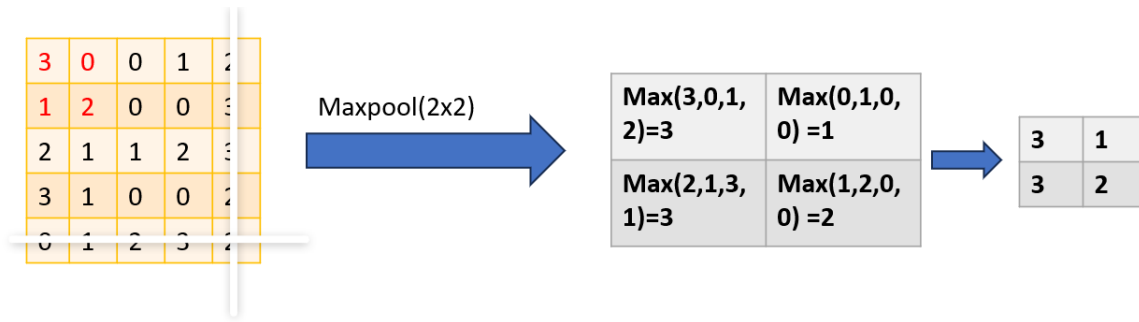


Figure 2.20: 2x2 matrix Max pooling

2.4.8 Sigmoid weighted linear Unit (SiLu)

The SWISH activation function is represented as $f(x) \Rightarrow x * \sigma(\beta * x)$.

When the value of Beta is set to 1.702, the activation function is recognized as GELU. If Beta is set to 1 in the SWISH function, it transforms into the SiLU function.

$$f(x) \Rightarrow x * \sigma(x)$$

2.4.9 YOLOV5 Architecture

The Architecture diagram of YOLOV5 is shown in fig 2.21 copied from YOLO architectural website[4] To enhance the accuracy and efficiency of an EfficientDet Model, the utilization of Feature Pyramid Network (FPN) is implemented. [34]

FPN employs a top-down approach and utilizes convolution as a function to enhance efficiency. However, the information flow is unidirectional. To enhance efficiency, PANet was introduced in the preceding model, featuring two flows—top-down and down-up. In YOLOv5, repeatable modules were introduced, referred to as BiFPN (Bidirectional Feature Pyramid Network), where all features are looped back, resulting in an efficient and improved model as shown in fig 2.22[34]

Augmentation Techniques to reduce overfitting model

To prevent overfitting, YOLOv5 employs various augmentation techniques, including: [4]

- Mosaic Augmentation: Mosaic Augmentation involves merging four images into a single picture. This technique helps the model better respond to random images during validation that may not resemble the Mosaic Augmented picture.
- COPY-PASTE Augmentation: Random images are pasted into other pictures to introduce noise and prevent overfitting of the model.
- Random Affine Transformations: This augmentation technique includes flipping and shearing images to effectively train the model.
- Albumentations: YOLOv5 utilizes the Albumentations library, a powerful tool with pre-defined augmentation techniques.
- HSV (Hue, Saturation, Value) Augmentation: HSV Augmentation involves changing the hue, saturation, and value of images during training to enhance the model's robustness. [4]

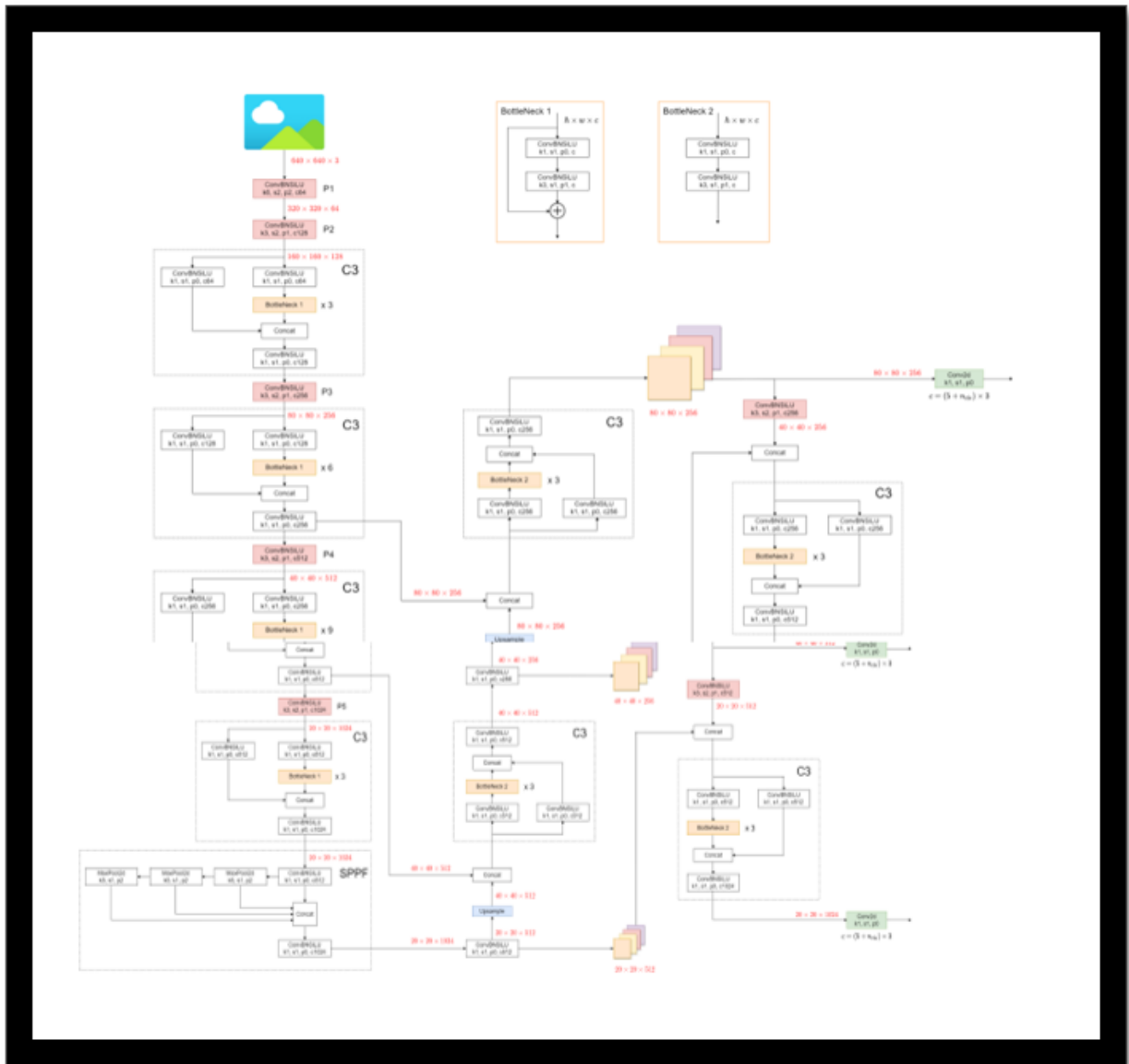
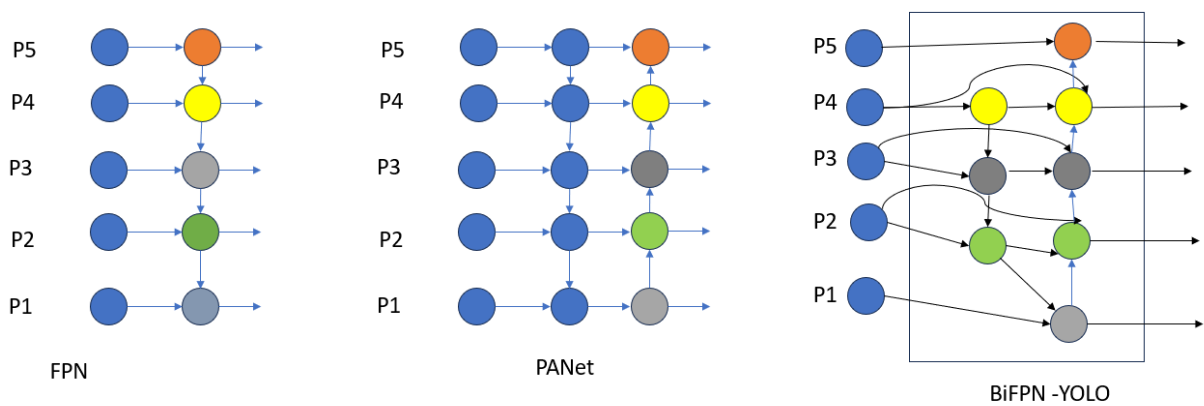


Figure 2.21: YOLO Architecture from Ultralytics Docs - [https://docs.ultralytics.com/yolov5/tutorials/architecture_description/?h = yolov51 - model - structure](https://docs.ultralytics.com/yolov5/tutorials/architecture_description/?h=yolov51-model-structure)



$$P_{out} = \text{Conv}(P_{in})$$

$$P_{out-1} = \text{conv}(P_{in-1} + \text{Resize}(P_{out}))$$

Figure 2.22: BiFPN

$$-\frac{1}{N} \sum_{i=1}^N Y_i * \log P(Y_i) + (1 - Y_i) * \log(1 - P(Y_i))$$

Figure 2.23: Binary Cross Entropy

Losses in YOLOv5

There are three losses calculated in YOLO model[4]

- Class Loss: It belongs to binary cross entropy , which is also known as logarithmic loss, which tracks the incorrect labelling by penalizing and rewarding a model based on its prediction and the formula for calculation is shown in figure 2.23
- Objectness Loss:Objectness means if the object is present in a particular grid or not.Also, this is calculated using Binary cross entropy method
- Location Loss:IoU loss which explains if the bounding box is aligned as per the ground truth image
- Total Loss: Total loss is a calculation of sum of losses $\lambda_1 \cdot L_{cls} + \lambda_2 \cdot L_{obl} + \lambda_3 \cdot L_{ls}$

2.4.10 YOLOV8 Architecture

YOLOv8 comprises three primary components: Backbone, Neck, and Head. The Backbone is the region where all deep learning calculations occur, serving as the stage for feature extraction. The output from the Backbone is then directed to the Neck, followed by the Head, where the determination of bounding boxes and labels takes place.

There are multiple components in YOLOV8 additionally as compared to YOLOV5 such as :

C2F and Bottleneck

In contrast to YOLOv5, which employs the C3 model where the output is derived from the final bottleneck operation, YOLOv8 adopts a different approach. YOLOv8 concatenates outputs from multiple iterations of bottleneck operations, convolutes the results, and then sends the output. Additionally, the bottleneck in YOLOv8 adds outputs from sequential convolutions, as illustrated in the figure 2.24.[36]

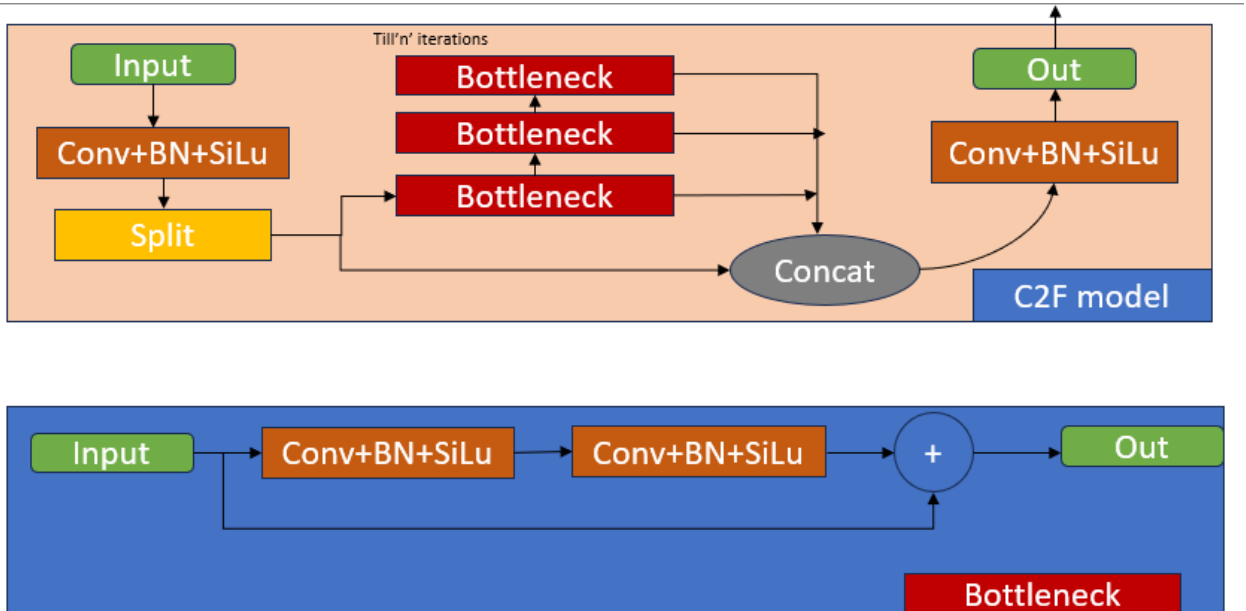


Figure 2.24: c2f and Bottleneck

SPPF - Special Pyramid pooling Fast

SPPF, or Special Pyramid Pooling Fast, is an enhanced version of SPP (Spatial Pyramid Pooling). One significant modification in SPPF is that all MaxPools are concatenated instead of being performed sequentially as shown in the fig 2.25

Loss-Yolov8

In contrast to YOLOv5, the losses in YOLOv8 are decoupled from each other and it is an anchor free model. Decoupled in Yolov8 context means that the results Boundingbox loss and class loss are independent of each other as shown in the figure 2.26 [5][6]

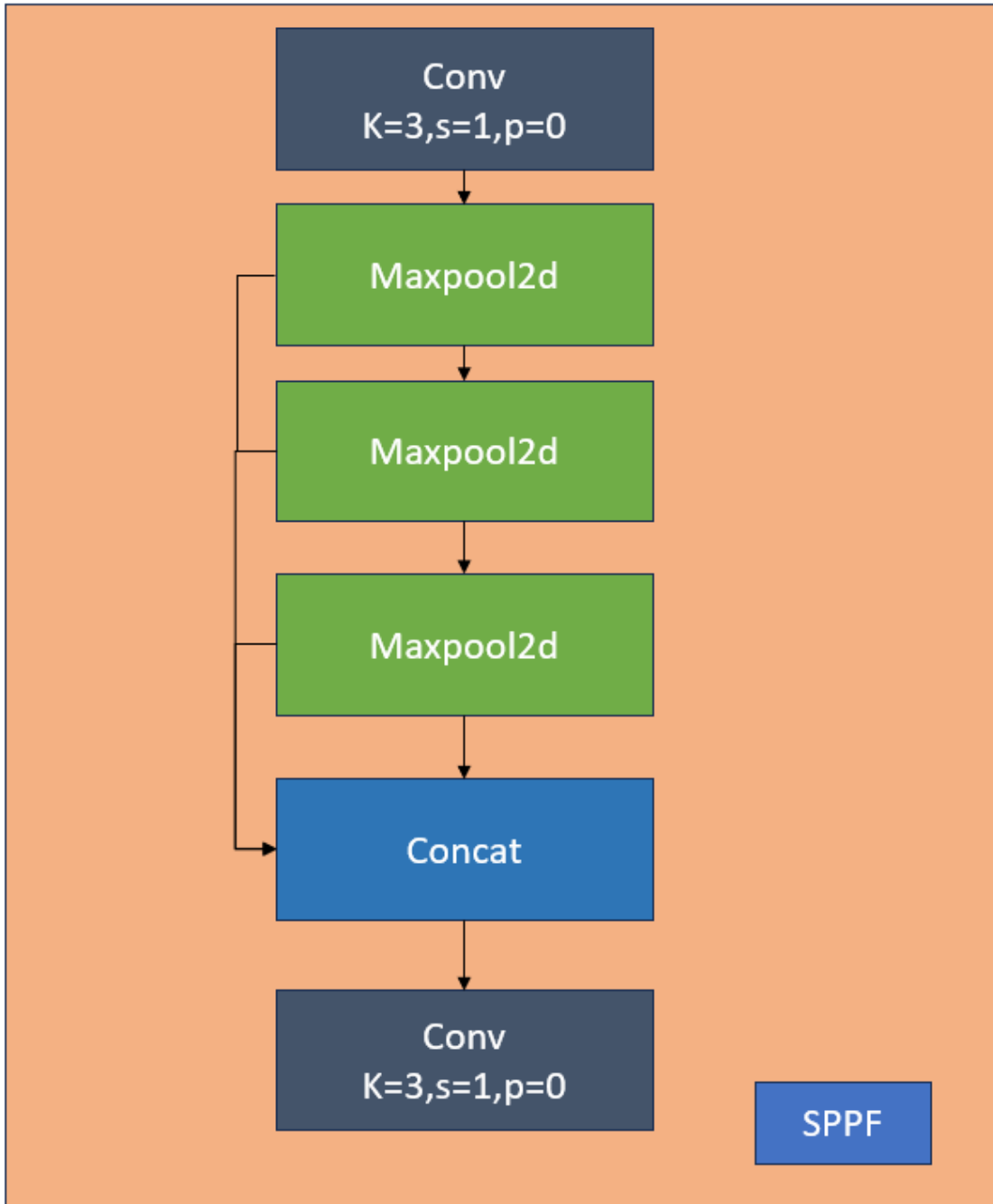


Figure 2.25: SPPF YOLOv8 emulated by [36]

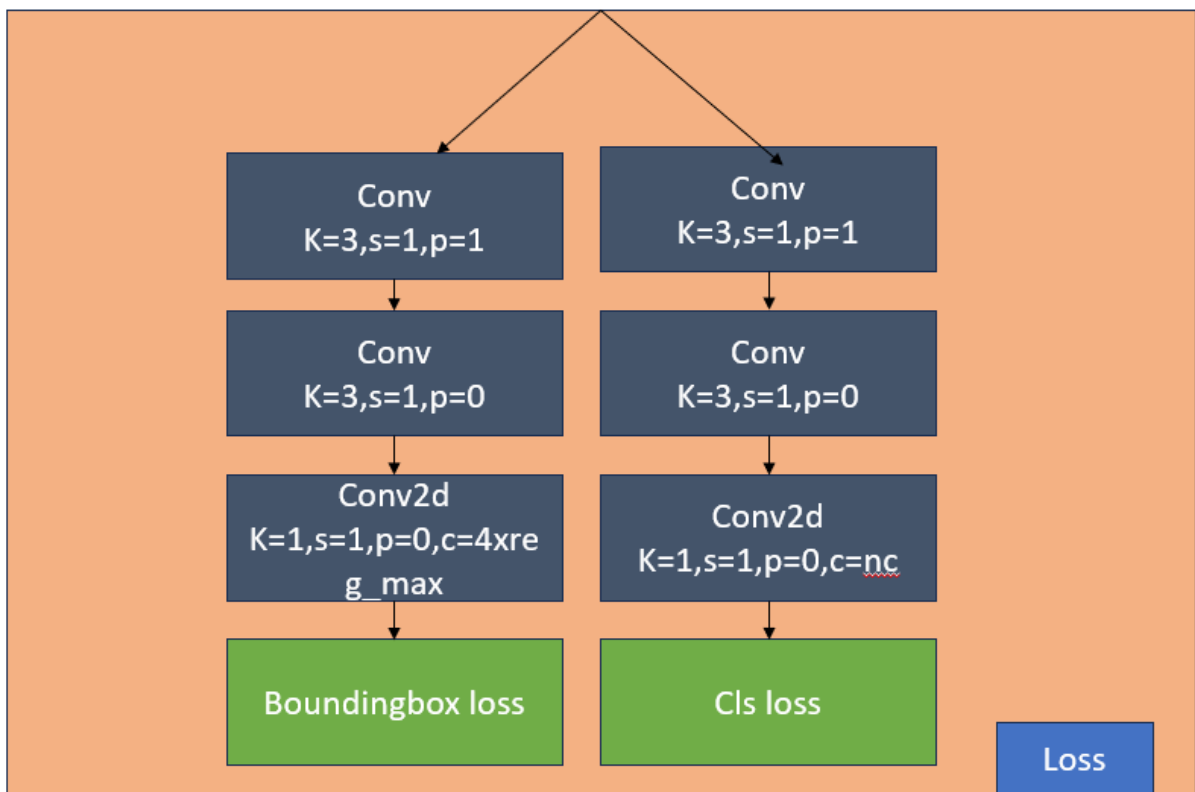


Figure 2.26: Loss function in YOLOv8

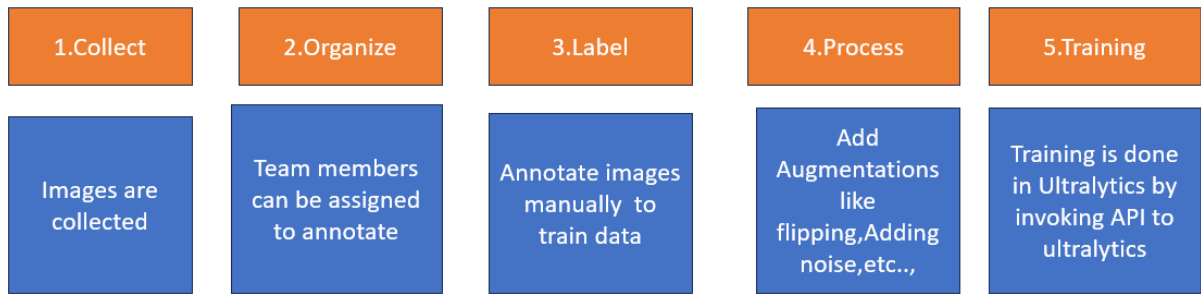


Figure 2.27: Roboflow

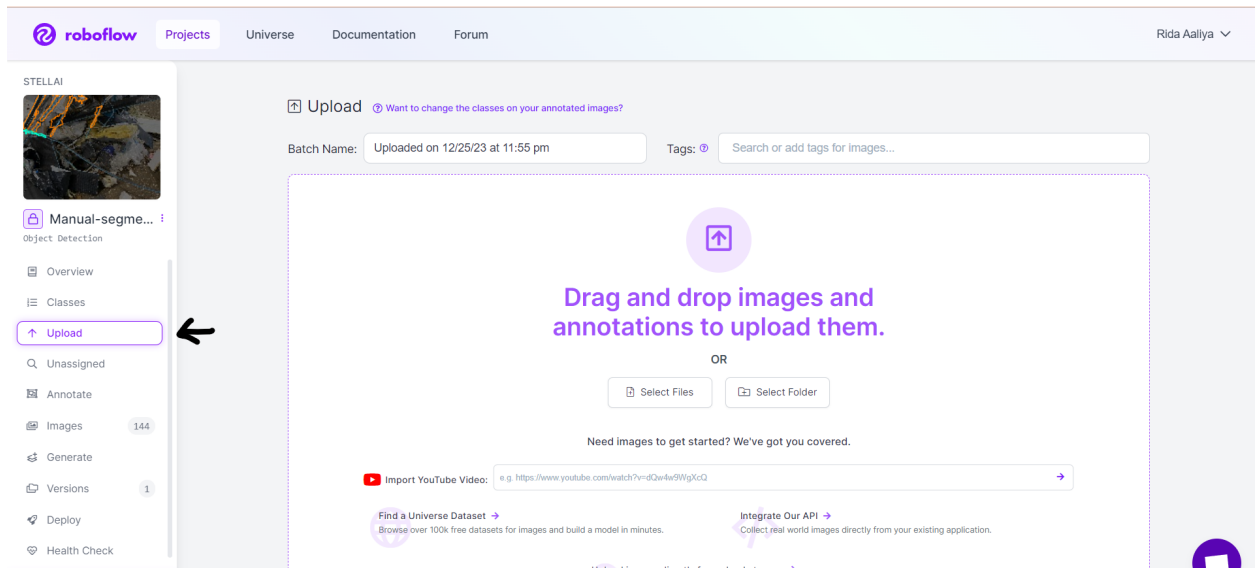


Figure 2.28: Roboflow Upload

2.5 Roboflow

Roboflow is a comprehensive tool that efficiently gathers, organizes, labels, processes, trains, deploys, and displays datasets. It supports easy integration with various machine learning algorithms, such as YOLO, Azure Custom Vision, and others. Additionally, it seamlessly connects with the Ultralytics webpage, simplifying the overall workflow as shown in fig 2.27[7]

Collection of dataset

For this thesis, 300 custom images from the recycling industry were uploaded. If more generalized images are needed, the Roboflow universe can also be utilized as shown in the figure 2.28[7]

Annotation

After uploading the images, they can be assigned to different team members for manual annotation. The datasets can then be divided into training, testing, and validation sets.

Augmentation

During the dataset creation, augmentation techniques can be applied, including flipping, upside-down transformations, adding noise, and converting images to grayscale. These aug-

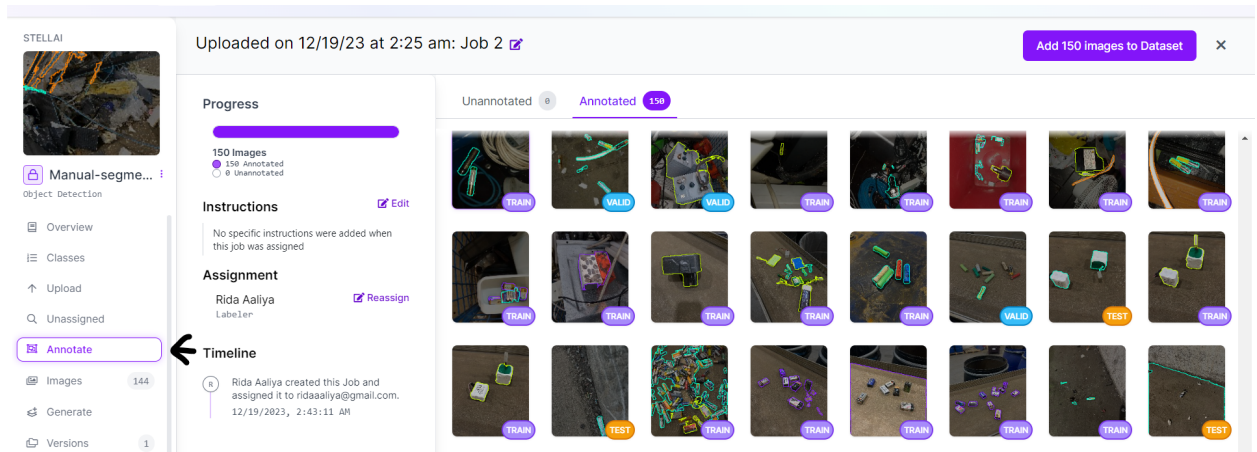


Figure 2.29: Roboflow Annotation

mentations help diversify the dataset and improve the model’s ability to generalize to different scenarios as shown in fig 2.29[7]

Invoking APIs to Various ML Platforms

Roboflow can invoke an API for Various ML platforms in various formats as shown in a table 2.1: [7]

Format	ML
CSV	Tensor flow object detection
CSV	RetinaNetKeras
CSV	Multi-Label Classification
JSON	COCO
JSON	COCODETECTION
JSON	CREATEML
XML	PASCALVOC
Txt	YOLO DARKNET
Txt	YOLOV3 KERAS
Txt	YOLOV4 PyTorch
Txt	Scaled-YOLOV4
Txt	YOLOV5 ORIENTED BOUNDING BOXES
Txt	MEITUAN/yolo6
Txt	YOLOV5 PyTorch
Txt	YOLOV7 PyTorch
Txt	YOLOV8
Code-Free Training Integrations	Ultralytics Hub
Code-Free Training Integrations	AWS Rekognition Custom labels
Code-Free Training Integrations	Google Cloud AutoML
Code-Free Training Integrations	Microsoft Azure Custom Vision
Code-Free Training Integrations	OpenAI Clip Classification
Code-Free Training Integrations	Tensorflow TF Record
Code-Free Training Integrations	Server BenchMark

Table 2.1: Roboflow Export model

2.6 Azure Custom AI

Microsoft provides Cognitive Services, including the Azure Custom AI service, which allows for the training and validation of models. Azure Custom AI offers versatility for both classification and object detection tasks, providing precise identification of objects along with their coordinates. The cloud service boasts auto-scaling capabilities during model training, ensuring efficient and rapid processing even with a large volume of images. In Custom AI, manual annotation of images is essential, and this can be done either through the portal interface or by downloading APIs directly from the Roboflow model. To train a model in Azure Custom AI, a minimum of 15 images of a particular class is required for training. During the validation phase, if there are any false positives or false negatives, feedback can be provided to improve the robustness of the model.

2.7 Grounded SAM

Grounded SAM is an innovative technology that enables automatic image segmentation using text prompts. It boasts an extensive pre-trained model with 11 million diverse, high-resolution images, averaging 3300x4950 pixels. The model includes 1.1 billion masks from SA -1B dataset, with approximately 99.1% of these masks being auto-generated. Grounded SAM is primarily comprised of three components[21]:

1. Image Encoder: This component processes each image once, and it can be applied before prompting the model.[21]
2. Prompt Encoder: The Prompt Encoder encodes text using the off-the-shelf text encoding from CLIP. It comes in two types: Sparse and Dense. These components work

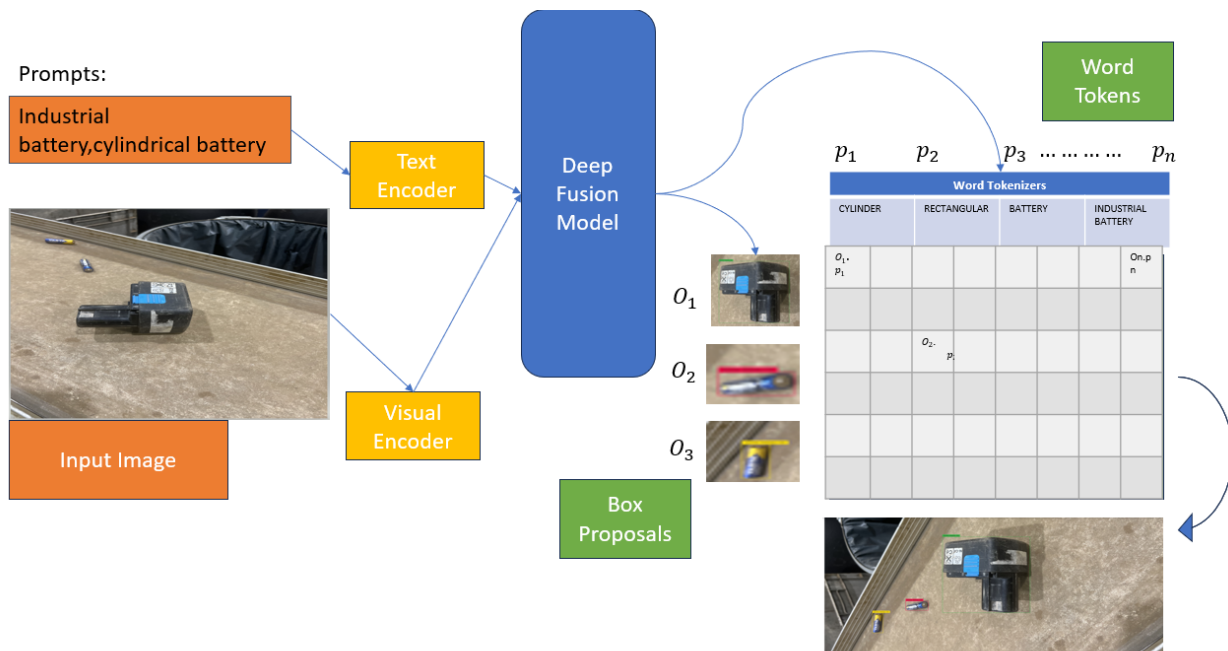


Figure 2.30: GroundingDino emulated from [U]

together to enable the segmentation of images based on text prompts.

3. Mask Decoder: This component involves the mapping of image embeddings, prompt embeddings, and output tokens to generate a mask. It is responsible for the up-sampling of images when the outputs are displayed. [21]

2.8 GroundingDino

GroundingDINO, an object detection model, utilizes GLIP (Grounded Language Image Pre-training), which is considered a superior model compared to CLIP (Contrastive Language Image Pre-training). [8] Grounding Dino is employed for object detection. When text prompts are provided, they are sent to the text encoder, while the image is processed by the visual encoder. Both sets of information are fed into the deep fusion model, incorporating Bert layers for processing. In the image, box proposals related to visual regions are generated. Simultaneously, word tokenizers are applied to tokenize text prompts. The matrices of words and pictures are then matched to precisely identify objects as shown in fig 2.30. [9]

2.9 Streamlit:

Streamlit is a front-end web framework seamlessly integrated with the Python library, making it extremely straightforward to install. Its primary objective is to offer a web service platform where dashboards and chatbot interfaces can be effortlessly set up. Streamlit is constructed using ReactJS components, earning it the designation of a JavaScript web application with Python integration. [20]

2.10 LangChain and OpenAI API

Langchain and OpenAI API can be harnessed to construct personalized chatbots for users. This chatbot is seamlessly integrated into a dashboard deployed on Streamlit, and the interactions, including queries and responses, are stored and managed through the PineCone vector database pod.

Chapter 3

Methods

Within recycling industries, batteries are routinely sorted, and even smaller AA batteries, which contain hazardous materials, pose a potential fire hazard. Among AA batteries, two distinct types are present: Lithium batteries and Alkaline batteries. Lithium batteries, in particular, are acknowledged for being more inherently hazardous[37]. There are majorly four issues have been identified in recycling industries and this project aims to provide scientific solutions in developing a product as a Proof of concept , and these include:

3.1 Problem statement - Dataset *and proposed plan of action*

To initiate the comprehension of the recycling plant and the data set model, attempts were made to explore different data sets such as the COCO model set, Roboflow Universe. While generic images of batteries were found, a challenge arose due to issues related to backgrounds, diverse textures of batteries, and other factors. Consequently, the generic pre-trained computer vision model didn't prove to be very effective for this project. Subsequently, efforts were directed towards creating a custom dataset by gathering a diverse set of images directly from the recycling plant. Approximately 300+ images and a few videos were collected for this purpose. The training of the project involved utilizing custom images that encompassed various backgrounds, including open ground, piles of waste materials, conveyor belts, and the floor. The objectives or implementation provided for this problem statement were as follows:

- Gather images from the recycling plant encompassing diverse textures and backgrounds.
- Annotate images using Roboflow or GROUNDING DINO

The primary rationale for validating precision in this project is rooted in the design objective of prioritizing safety and minimizing fire hazards induced by batteries. Precision and Recall assessment becomes crucial to ensuring the effectiveness and reliability of the product in addressing these safety concerns.

3.2 Problem statement - Detection *and proposed plan of action*

Recycling plants, characterized by their larger units, exhibit a disorganized arrangement of numerous batteries strewn across the ground, and anywhere on floors and bins, as illustrated in the figure below. This scenario poses a potential fire hazard, particularly when in contact with combustible substances or under the pressure of inadvertently placed objects. During an interview with an employee of StellAI, instances were also highlighted wherein an entire sealed drum/bin exploded due to the presence of lithium batteries few months ago in Oslo recycling unit. To tackle this issue, the paramount focus is on the detection of objects, specifically understanding the precise location of batteries in this context.

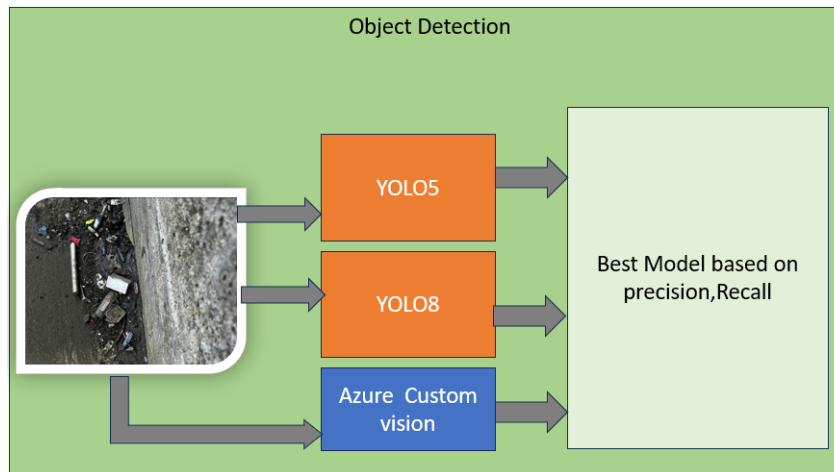


Figure 3.1: Object Detection Module

For this project, aerial drone images of a recycling plant have been collected, with a primary focus on pinpointing the exact locations of various items, including Cylindrical AA batteries, wires/cables, Rectangular Batteries, Industrial batteries, and small coin batteries. The experiment is designed to accomplish this specific identification task. To address this concern, following two experiments will be performed

3.2.1 Experiment 1

The initial phase of the experiment involves evaluating the accuracy of annotations by comparing manual annotations performed in Roboflow with automated annotations generated by GroundingDino.

3.2.2 Experiment 2

- Image detection utilizing YOLO models such as YOLOv5, YOLOv8 was undertaken. The focus on image detection serves as the initial component of the overall solution.
- The primary objective of this project is to detect 6 different types of objects into six distinct classes: Cylindrical battery, Coin battery, Wires/cables, Rectangular battery, Industrial batteries, and No batteries.
- To broaden the scope of this experiment, an additional attempt was made to upload trained models in Azure Custom Vision. The aim is to ensure that both developers and non-developers can effortlessly manage this module with ease.

The model that exhibits superior performance will be chosen for the Proof of Concept (PoC). as shown in fig 3.1

3.3 Problem statement -Segmentation and proposed plan of action

When materials are transported on a conveyor belt, various objects such as bulbs, wires, and batteries are often clustered together, creating a challenge in effectively segmenting and distinguishing individual objects.

To tackle this challenge the following experiments are performed:

3.3.1 Experiment 3

- Polygon annotations are performed manually using Roboflow, and automated annotations are generated using GroundedSAM. The accuracy of the model is then compared.

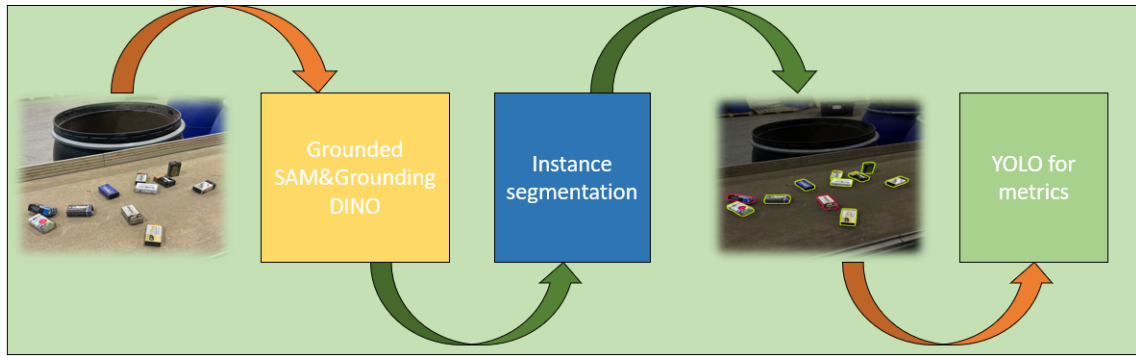


Figure 3.2: Segmentation Module

3.3.2 Experiment 4

- To provide output of segmentation which yields better accuracy to YOLOV8 and YOLOV5 model to calculate metrics and also to use the trained dataset to predict live stream data

Ref fig3.2

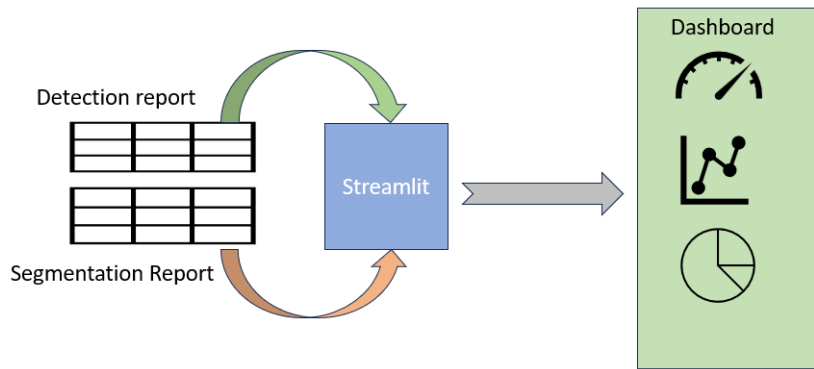


Figure 3.3: Health Dashboard

3.4 Problem statement - Health Dashboard *and proposed plan of action*

Upon completion of the model training and the initiation of predictions, a pivotal step involves creating a dashboard. This dashboard serves as a user-friendly interface, providing users with comprehensive statistics and facilitating a better understanding of patterns. To achieve this, inputs from the previous experiments are integrated into a Streamlit Application to generate the desired dashboard.

This project places a strong emphasis on employing diverse methods and consolidating them into a comprehensive package based on the results that demonstrate superior accuracy.ref fig 3.3

Chapter 4

Related Literature

4.1 Azure Custom AI

This section covers the article written by Salvaris, M et al[29] and Alom, M. Z. et al.[15] In the article by Salvaris, M et al[29], it is discussed that AI has undergone significant advancements over the years, with notable improvements made in this decade. It emphasizes the importance of understanding the timeline of milestones to observe the remarkable progress in machine learning and deep learning.

This book delves into the architecture of Convolutional Neural Networks, Recurrent Neural Networks, and Generative Adversarial Networks, along with their advancements over the years. It explores how these technologies have seamlessly integrated into everyday human life, contributing to innovations like speech-to-speech translators, automated chatbots, and autonomous cars. Microsoft collaborates with a Power and Utility company in Norway, contributing to an energy management system. The system utilizes drones for capturing images of power lines, employing deep learning image detection techniques known as eSMarts. This scenario mirrors the challenge of drone images requiring image detection, showcasing the application of advanced technologies in the Recycling or sustainability sector. Microsoft offers Cognitive Services, a platform that includes pre-trained models derived from extensive datasets such as COCO and ImageNet. Users can seamlessly leverage these pre-trained models by invoking APIs, facilitating the integration of powerful and ready-to-use AI capabilities into their applications or projects. In addition to pre-trained models, Microsoft provides Custom AI solutions. This allows users to train models on their own custom datasets by labeling objects, conducting model training, and validating the model's efficiency using specific validation data. Custom AI solutions offer a more tailored approach for users with specific needs and datasets. Similar to other deep learning technologies, Microsoft supports supervised learning through Custom AI and unsupervised learning through their cognitive services. This provides users with flexibility in choosing the learning approach that best fits their specific use cases and data availability. Machine learning encompasses various tasks, including classification, regression, recommendations, ranking, and clustering. One of the most prominent learning approaches in machine learning is supervised learning, where humans provide labels to represent the ground truth. Nevertheless, significant progress has been made in unsupervised learning through the utilization of techniques such as Generative Adversarial Networks (GANs) and autoencoders. Deep learning, a subset of Machine Learning (ML) and Artificial Intelligence (AI), faced challenges in its early years due to limited data availability and computational power. However, advancements such as the introduction of GPUs by NVIDIA and the creation of datasets like ImageNet played a crucial role in making deep learning feasible. Deep learning encompasses four main applications:

1. Classification: Determines whether an object is present in an image or not.
2. Object Classification and Localization: Identifies the location of the object in an image.

Timeline

• MICROSOFT AI/ML & DEEP LEARNING PROGRESS

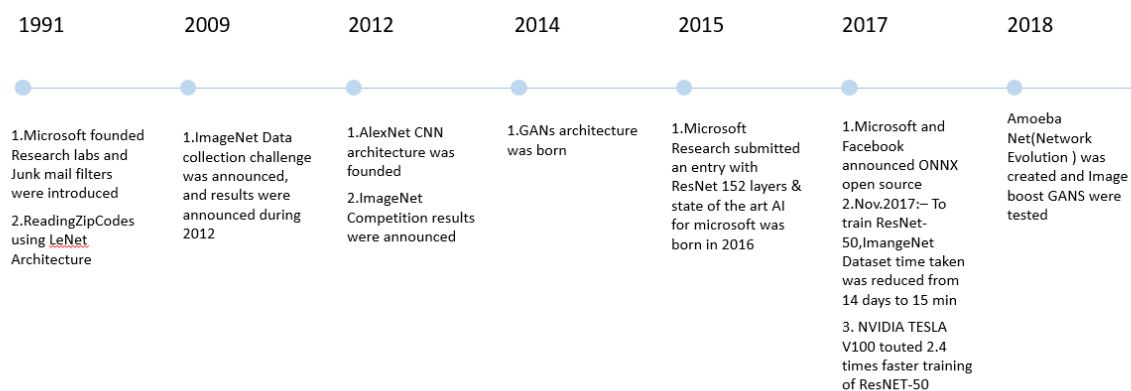


Figure 4.1: Timeline

Common Network Architecture	Applications
CNN	Classification & Detection
RNN	NLP, Time Series Analysis
GANs	Image to image ,text to image creation
Auto encoders	Anomaly detection

Table 4.1: Common Network Architecture and its Applications

3. Detection: Specifies the coordinates where an object is situated in an image.
4. Segmentation: Identifies the pixels that contain objects in an image.

The deep learning workflow involves several key steps, including identifying a relevant dataset, pre-processing the data, training the model, assessing its performance, fine-tuning as necessary, deploying the model, and iterative refining the process with continuous data collection and monitoring for ongoing improvement.

Convolutional Neural Networks (CNNs) exhibit two distinctive features:

1. The Automatic Feature Extractor: This component of the neural network is responsible for extracting features from images using convolution layers and pooling layers.
2. The Classifier: This is a fully connected network that facilitates the classification of images into various classes.

In CNN's, weights are initially chosen randomly. During the training process, if the error is calculated in the classifier, back propagation begins, and weights are adjusted accordingly. This iterative process continues, with the network comparing its loss value after each adjustment, aiming to minimize the error. Loss functions can be calculated using Mean Squared Error, Binary Cross-entropy, or Categorical Cross-entropy. The choice of the loss function depends on the activation function applied in the last layer of the neural network. Microsoft's pre-trained CNN models (ResNet-50, AlexNet, etc..) are trained on large datasets such as ImageNet, CIFAR-10, COCO.

Dr. Fei Fei Li, a professor from Stanford, pioneered ImageNet, a project where images are collected across the internet and also labelled. Her team, with the assistance of 'mechanical

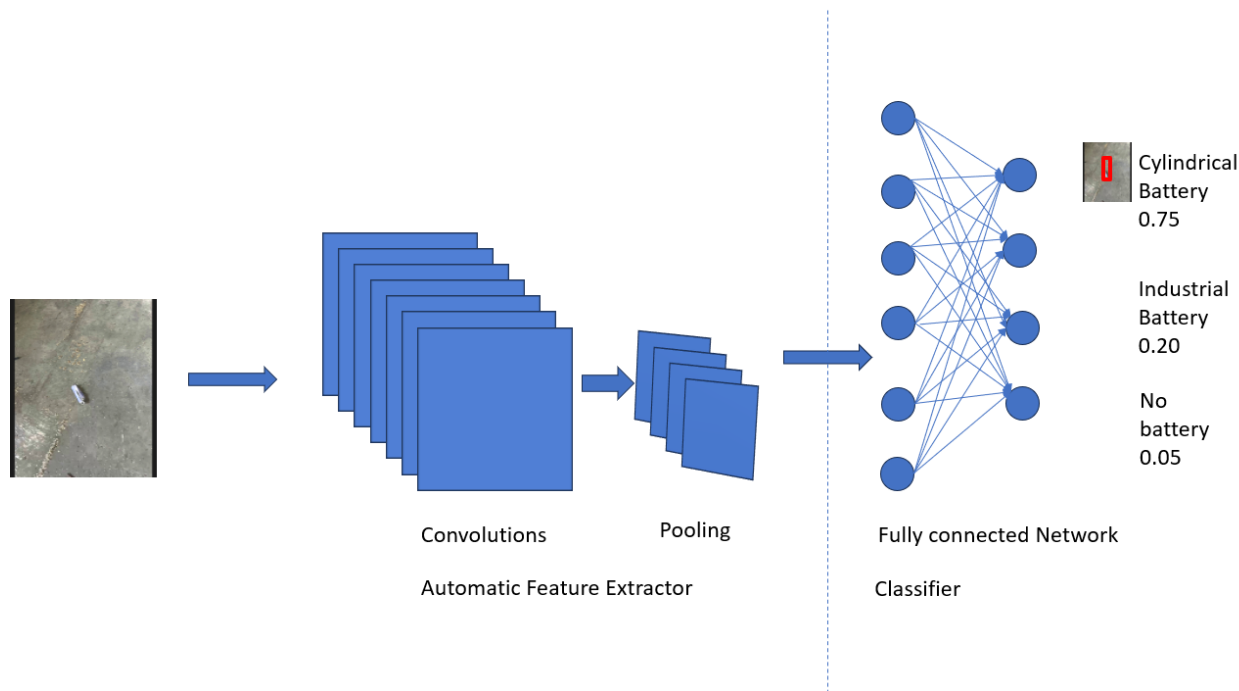


Figure 4.2: CNN

Turk’—a platform where people are hired on a contract basis—worked on labeling data for millions of images to create a comprehensive dataset.[29] As datasets were becoming more extensive, the ImageNet Competition was announced, challenging participants to propose a CNN model with minimal error function. This competition led to the development of the AlexNet algorithm in 2012. The AlexNet algorithm introduced two additional elements to the traditional CNN: Local Response Normalization (LRN) and Drop Out. LRN enhances peak values and dampens lesser values in the CNN network, a concept known as lateral inhibition. It is added after the ReLU activation function. Drop Out is a concept incorporated in the Fully Connected Network, where neurons with a probability less than 0.5 are dropped out, preventing their participation in both backward propagation and forward feed. To understand further about Algorithms, as per Alom, M. Z. et al. [15] referred AlexNet and comprises five convolutional layers and two fully connected networks. The first layer consists of 96 filters of an 11x11 matrix, with a max pool of 3x3 filters and a stride of 2. The second layer has 256 filters, followed by layers with 384, 256, and 256 filters, respectively. Drop Out elements are added in the Fully Connected Network (FCN). The input samples are of size 224x224x3, and the output is of size 55x55x96.[15]

Following AlexNet, several architectures were developed, and ZFNet received an award in 2013. Addressing the computational heaviness of AlexNet, ZFNet modified the 11x11 kernels in AlexNet to 7x7, reducing the number of weights and making computations more manageable.

The 2014 award winner was GoogleNet, a model that introduced dimensionality reduction, leading to increased accuracy. GoogleNet also managed to reduce the total number of parameters to 7 million, a significant reduction compared to AlexNet’s 60 million parameters. This architecture comprised 22 layers. [15] In 2015, ResNet was introduced as a Residual network. True to its name, this model aimed to tackle the vanishing gradient problem. ResNet provided various versions with multiple layers, including 34, 50, 101, 152, and an impressive 1202 layers.[15] The diminishing gradient problem occurs when the network’s complexity increases, causing the time required for calculations to rise significantly, and sometimes even leading to the network becoming unresponsive. ResNet successfully addressed this challenge. [15] Microsoft has trained using these algorithms. Microsoft employs transfer learning in both Cognitive Services and Custom Vision. In Custom Vision, users can upload and manually

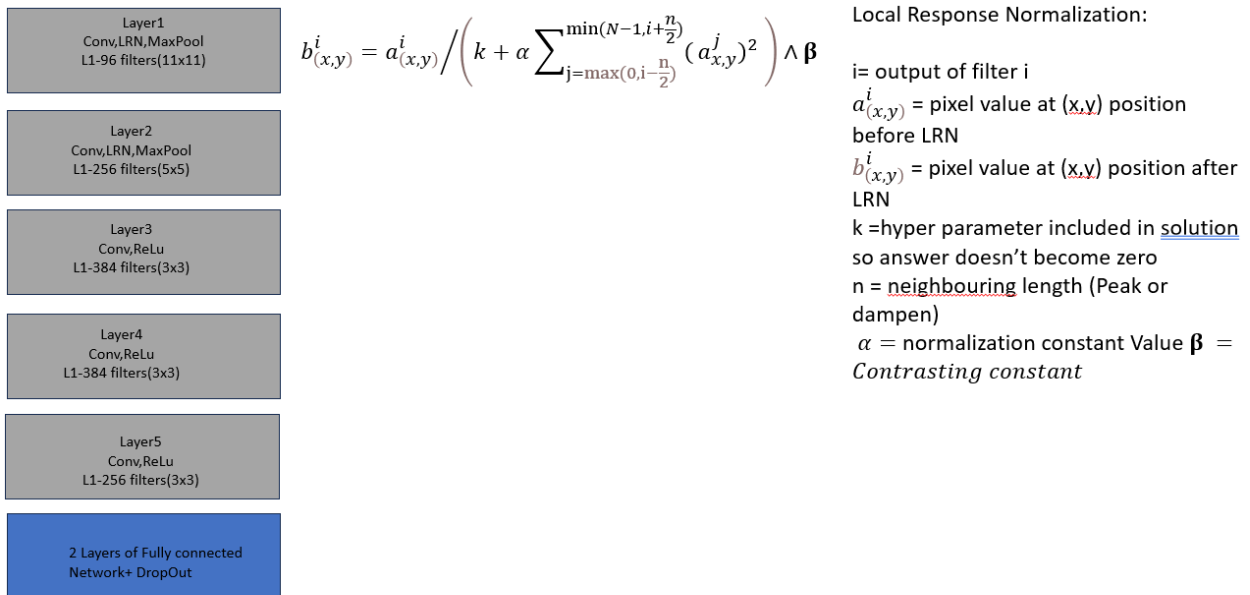


Figure 4.3: AlexNet

label custom datasets on a web page. Alternatively, they can retrieve manually labeled images from Roboflow. After training a model, quick tests can be conducted, and if the results and accuracy meet the requirements, the model can be used as a baseline by invoking an API. [29] The advantages of using Microsoft include the capability in Custom AI to provide immediate feedback when objects are not detected. This facilitates quick relearning for the AI system. Microsoft is already involved in a project in Norway that focuses on energy and sustainability. The project involves custom data and aerial images, aligning with the requirements of this project. This existing work in a similar domain could provide valuable assistance and resources for this project. There are options available such as BatchAI and Batch Shipyard that facilitate the automatic scaling of GPUs for training. This feature helps in accelerating the learning process and simplifying the training of machine learning models. Additionally, Microsoft ensures the privacy of custom data, providing a secure environment for handling sensitive information. Currently, Microsoft has invested in OpenAI and focuses on developing LLM applications, with significant integration of Co-Pilot in Microsoft's offerings.

4.2 YOLO-You Only Look Once

According to the thesis, two papers were thoroughly examined:

1. YOLO architecture, spanning from YOLO1 to YOLO8.Hussain, M. (2023) "YOLO-v1 to YOLO-v8"[19]
2. A comparative analysis of YOLOv5 versus YOLOv8 for image detection using drone camera images.[30]

As detailed in an earlier section, significant architectural changes have occurred since 2012. YOLO draws inspiration from the GoogleNet Architecture and was initially crafted in 2015. The primary focus of developing this architecture has consistently been on achieving high speed and accuracy. YOLO, functioning as a single-shot detector, will be further explored in subsequent sections of this chapter. The lightweight computation characteristic contributes to the model's robustness and swiftness in processing data. This efficiency has played a

Timeline

• YOLO ARCHITECTURE



Figure 4.4: Yolo Architecture- Timeline emulated from [19]

pivotal role in making YOLO a preferred model for real-time streaming data, establishing its popularity.

YOLOV1 introduced a revolutionary concept by dividing an image into grids, with each grid of size $s \times s$ predicting bounding boxes. The final selection involved choosing the box with the highest confidence score. YOLOV1 also implemented the Non-Max Suppression technique, which required defining threshold values. Boxes with confidence scores below the specified threshold were eliminated from consideration. This is also one of the factors influencing how changes in threshold values impact mean Average Precision (mAP). In spite of YOLOv1 bringing few new concepts like single shot detector, Non-Max suppression value and grids with 2 bounding boxes each, the mAP it achieved was 63.4 percent at 45 frames per second for YOLO as compared to faster R-CNN which had mAP of 71 percent. As a result, various variants of YOLO have been introduced.

In YOLOv2, the goodness of speed which YOLOv1 model gave had to be retained and had to address limitations and hence few new concepts were introduced

- **Batch Normalization:** This technique was implemented to reduce covariate shift. In batch normalization, when input is divided into batches and fed into layers, the mean and standard deviation of the first layer are calculated. The data before entering any layer is then normalized based on these mean and variance values, and it is further shifted and scaled using hyperparameters like beta and gamma. This process helps reduce the need for dropout, which was introduced in ResNet to address issues like overfitting and diminishing gradient problems.
- **Anchor Boxes:** Anchor boxes are predefined template boxes whose dimensions are typically calculated by pre-training models. This concept is beneficial in determining the correct bounding box with respect to confidence scores and the region of interest for an object.

After these enhancements, YOLOv2, when trained on the ImageNet dataset, exhibited an improvement in mean Average Precision (mAP) compared to state-of-the-art architectures such as Faster R-CNN.

YOLOv2 still faced challenges in identifying smaller objects, leading to the development of

the YOLOv3 architecture. In YOLOv3, the goal was to preserve the positive aspects of YOLOv2 while introducing additional improvements. This involved increasing the number of layers to 106 convolution layers and implementing multi-scale object detection. The latter implies that detection occurs in various regions across multiple layers, each with different granularities. These detections are then converged to facilitate the identification of smaller objects.

After YOLOv3, the author of the proposed models did not continue further, and the development of YOLOv4 was taken up by the computer vision community. Initially, multiple models were experimented with, such as CSPResNext-50, CSP-DarkNet-53, Efficient Net-B3. The primary goal was to enhance performance, and after experimenting with various backbones, CSPDarknet-53 was selected. This part of the architecture is referred to as the Backbone. For the Neck region, a slight improvement was made by choosing PANet (Path Aggregation Network), where networks are concatenated instead of using the ADD function. This can be used for both top-bottom and bottom-up architecture. Along with these features, data augmentation, i.e., intentional addition of noise, was introduced to improve and train models better. One of the highlights of YOLOv4 is the creation of the Mish Activation function, where $f(x) = x \cdot \tanh(\text{softplus}(x))$. YOLOv5 brought several advancements compared to its predecessors. First, it was open-sourced, and secondly, it was written in PyTorch to better accommodate production environments. In YOLOv2, the Anchor Box concept was introduced, and in YOLOv5, this algorithm was integrated into a pipeline, enabling automatic detection of anchor boxes and reducing the number of parameters in the network. Additionally, YOLOv5 introduced various variations, such as YOLOv5-s, YOLOv5-m, YOLOv5-l, and YOLOv5-x models, each corresponding to different computational parameters. Yolov6 was developed by the Meituan technical team in China and introduced several enhancements[19]:

1. Industrial Focus: Yolov6 was specifically designed for industrial applications.
2. Anchor-Free Model: Unlike some other models, Yolov6 is an anchor-free model, meaning it does not rely on predefined anchor boxes for object detection.
3. Decoupled Classification and Regression: Yolov6 separates the tasks of classification and regression, utilizing different heads for each. This decoupled architecture results in two distinct losses: VFL (Varifocal Loss) for classification and Distributed Focal Loss for regression (bounding box loss).
4. Teacher-Student Model: Yolov6 employs a Teacher-Student model approach. The Teacher model contains labels and ground truth information, contributing to the training process.

Yolo v7 was launched concurrently with YOLOv6, focusing on GPU enhancements and architectural redesigns. It was designed for compound scaling, where models can be scaled in depth or width depending on the specific requirements, whether it be accuracy or speed. Additionally, YOLOv7 introduced some additional trainable "bag of freebies" to enhance its capabilities.

Yolov8, developed by Ultralytics in January 2023, is specifically designed for tasks such as detection, segmentation, pose estimation, tracking, and classification. As a state-of-the-art model within the YOLO family, it is known for its exceptional speed and accuracy, particularly in real-time video applications. The integration with Python further contributes to its popularity. Yolov8 is capable of tracking up to 50 frames per second, showcasing its efficiency in dynamic environments.

Comparing YOLOv5 and YOLOv8 is significant, especially considering they both come from

Ultralytics. Both frameworks are well-suited for real-time video applications, aligning with the specific needs of the recycling industry. This makes the comparison essential to determine which model better addresses the requirements of the industry in terms of efficiency, accuracy, and real-time processing capabilities.

In the article written by Indri Purwita Sary et al [30] it presents a comparative analysis between YOLOv5 and YOLOv8 in the context of detecting people using unmanned aerial vehicles (UAVs). The project involves experimenting with drone images across different backgrounds and lighting conditions to assess the models' ability to detect people. The study utilizes the Auth-Persons pre-trained dataset and discusses the advantages of the YOLO single-shot detector model. The author compares the results of YOLOv5 and YOLOv8 on this dataset, providing insights into their performance. The evaluation involves Confusion Matrix parameters, and the conclusion suggests that YOLOv5 outperforms YOLOv8 by 0.54%.

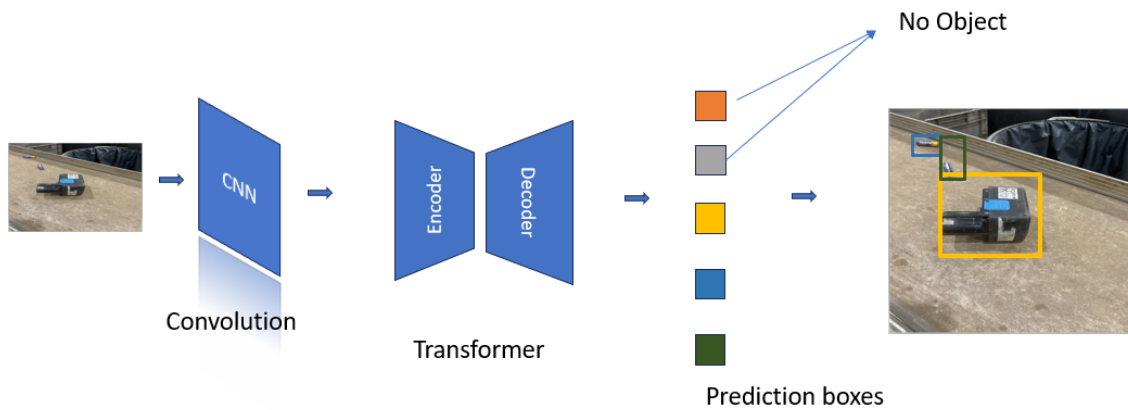


Figure 4.5: DETR

Timeline

• GROUNDED SAM & GROUNDINGDINO

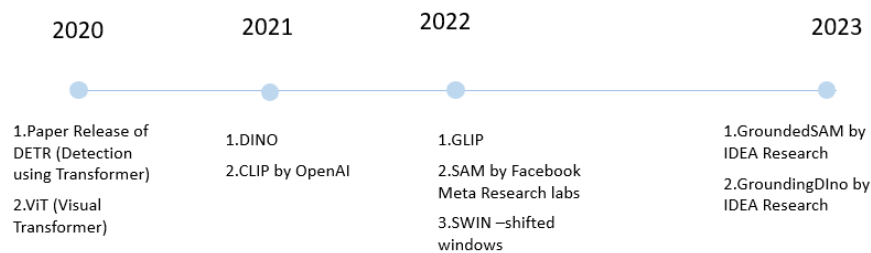


Figure 4.6: Timeline -Grounded SAM and GroundingDino

4.3 GroundingDino

DeTR, short for Detection Transformer, leverages the transformer architecture for object detection tasks. While transformers are well-established in natural language processing (NLP), the application of the same model to image detection has gained significant attention in recent years. This approach allows for effective object classification and labeling without the need for explicit supervision, showcasing the versatility of the transformer architecture across different domains.

In the realm of unsupervised learning for image detection, notable contributions have been made by organizations such as OpenAI, Google, and Microsoft in recent years. Ongoing efforts focus on continuous advancements and improvements in this domain. CLIP, which stands for Contrastive Language-Image Pre-training, is a model influenced by the context of ViT visual representations from textual annotations. In this approach, an image is provided as input to the model, and it generates text as a response or prompt based on the visual content. In simpler terms,

CLIP combines a transformer model for text embeddings with either ResNet or Visual Trans-

former (ViT) for vision embeddings. During training, 32,768 random snippets were paired with 32,768 images, and the model learned to associate text and visual content. By Radford, A. et al [28] ResNet performs effectively for image encoding in the ImageNet dataset. However, when applied to CLIP with ImageNet sketches, it doesn't yield satisfactory results. Therefore, Visual Transformer was introduced to address this issue, and it proves to be more effective in handling noisy data.

GLIP, or Grounded Language Image Pretraining, is a model introduced by Microsoft that draws inspiration from CLIP (Contrastive Language-Image Pretraining). Liunian, H. et al. [24]. GLIP has been pretrained using a vast dataset of 27 million grounding instances. This dataset includes 3 million human-annotated images and 24 million image-text pairs obtained from web crawling. According to the author, GLIP has achieved an Average Precision (AP) of 60.8 on the COCO dataset.

GLIP utilizes a phrase grounding mechanism, making it proficient at identifying crucial phrases within a sentence. In terms of architecture, GLIP differs from CLIP by integrating vision and text at a deep fusion stage, unlike CLIP which fuses them in the final step. The model boasts 58.4 million unique noun phrases, facilitating the identification of phrases. GLIP is user-friendly and deployable as it leverages a transfer learning model. It comes pre-trained with 80 common object categories from MS COCO and 1000 common object categories from LVIS, and phrase grounding is derived from the FLICKR30 dataset.

DINO, an acronym for DETR with Improved DeNoising anchor boxes, is an image detection model featuring multi-layer transformer encoders, multi-layer transformer decoders, and multiple prediction heads for object identification. Grounding DINO extends this model by incorporating GLIP, creating an open-set object detection framework. Open-set object detection enhances the model's ability to discern intricate details about objects through text phrases or text comprehension. ViT, initially a popular transformer for computer vision, faces limitations when considering images with nuanced details. The standard 16x16 pixel conversion may not be suitable for images with tiny objects, leading to computational challenges when expanding to larger dimensions. To address this, the Swin Transformer was introduced, where SWIN stands for Shifted Windows. In Swin Transformer, the concept of shifted windows is employed, allowing self-attention to work only with neighboring pixels. This approach reduces the computational load, and by iteratively shifting windows, the model covers all parts of the image efficiently. The Swin Transformer thus mitigates the challenges associated with large-scale computations in ViT. as per Liu, Z. et al. [26]

The author, Liu, S. *et al.* [25] has given an explanation of working architecture. The architectural diagram was extracted from the GroundingDINO paper and has been divided into three sections.

When given a prompt such as "Batteries, Wires, Circular battery" and an input image for object extraction, two parallel operations commence. The images are sent to the SWIN Transformer, which operates on the shifted window concept. This choice is justified by the understanding that there are numerous extractions from images compared to tokens from text. Simultaneously, the text prompt is processed in the next stage of transformers, where vanilla features are extracted, as per the author's explanation. The extracted features are subsequently directed to the Feature Enhancer. Within the Feature Enhancer, the images undergo Deformable Self-Attention, which involves processing the input images with queries to generate offsets and multiple heads for extracting valuable information. Simultaneously, text features are subjected to self-attention as shown in fig 2. In the third stage, the outcomes from the previous steps are input into a cross-modality query, where the results from both images and text are combined. The values in the matrix are analyzed, and this layer aids in injecting text information into queries.

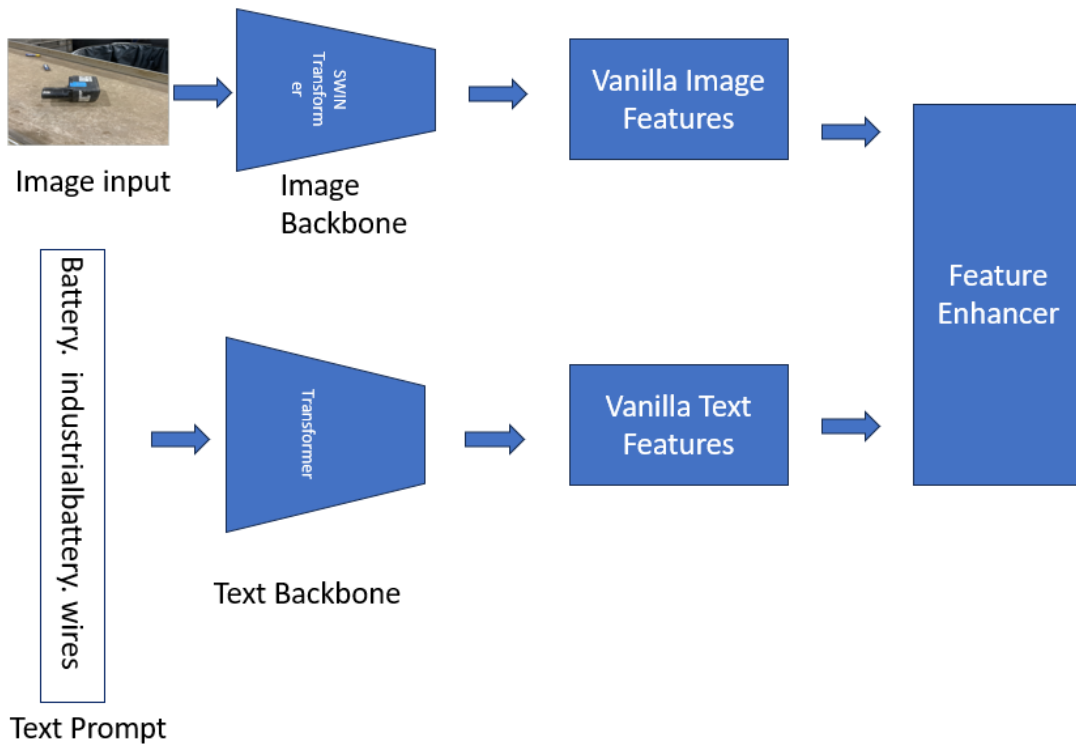


Figure 4.7: Grounding DiNO-Part I

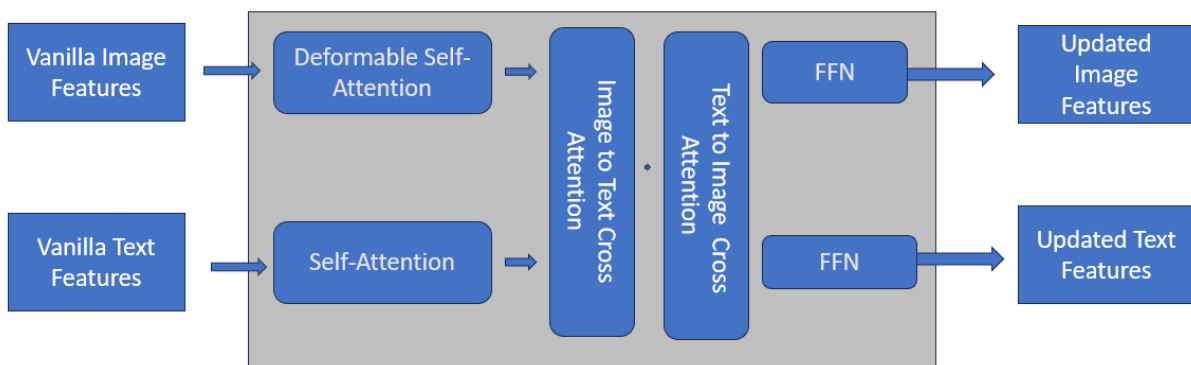


Figure 4.8: GroundingDino-PartII

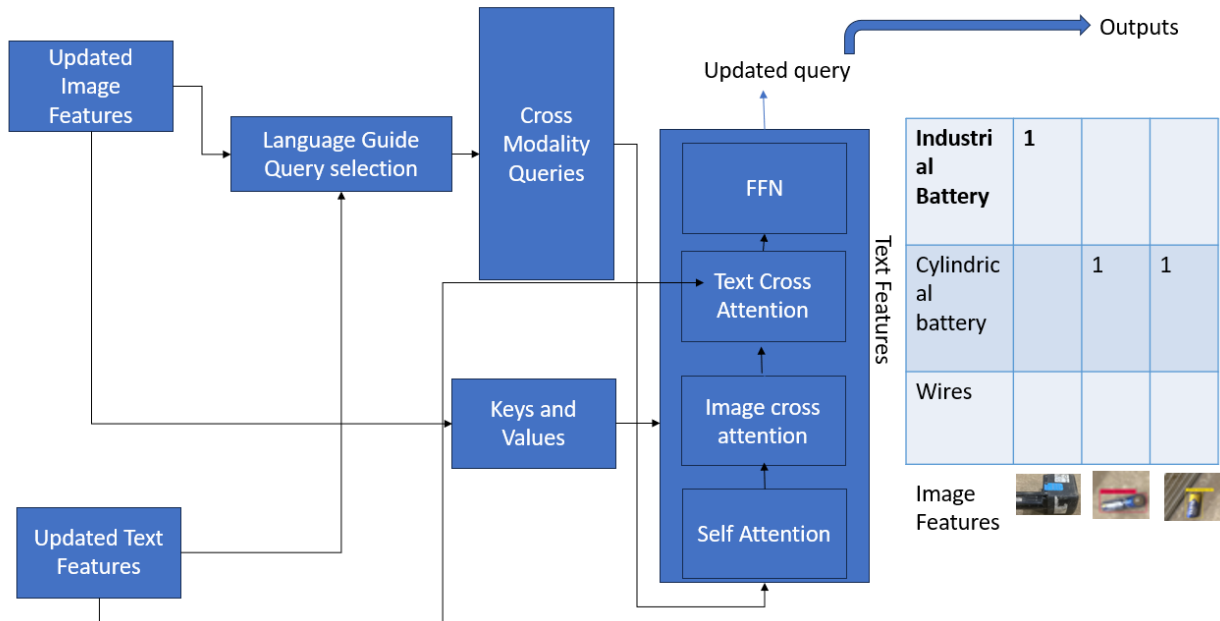


Figure 4.9: GroundingDino-III

4.4 GroundedSAM

Grounded SAM is a fusion of GroundingDINO and the Segment Anything model. As detailed by Kirillov, A. et al. [21] regarding the Segment Anything model, it drew inspiration from the functioning of NLP. Initially, the authors constructed an extensive segmentation dataset consisting of 1 billion masks on 11 million data points, referred to as SA-1B. The primary objective of Segment Anything is to generate masks based on a given text prompt. This model is influenced by CLIP for prompt encoding and ViT (Visual Transformer) for image encoding. The Segment Anything model comprises three data engines:

1. Assisted-Manual: SAM assists human annotators in the annotation process.
2. Semi-Automatic: SAM automatically annotates some parts of images, which are then refined by human annotators.
3. Fully Automatic: SAM performs complete automatic annotation without human intervention.

Segment Anything consists of three main components:

1. Image Encoder: It utilizes a pre-trained Vision Transformer (ViT) based on the MAE architecture for processing images.
2. Prompt Encoder: This component uses Contrastive Language-Image Pre-training (CLIP) for encoding prompts or text.
3. Mask Decoder: Responsible for mapping the image to text embeddings and generating a token-to-mask output.

Grounded SAM paper was released on July 2023 and authors Zhang, C. et al. [38] explain GroundedSAM a fusion of GroundingDINO and SAM.

In a simplified explanation, the model works by first detecting edges and identifying boxes using GroundingDINO for image encoding. Then, the Segment Anything model is employed to generate masks. Finally, the text prompt is incorporated, resulting in a masked Grounded

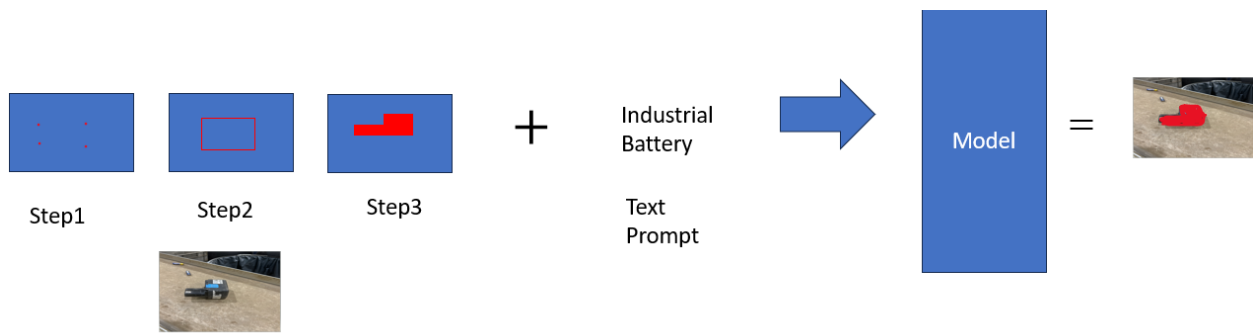


Figure 4.10: Grounded SAM -General architecture

SAM output. The integration of these components allows for the identification of objects based on both visual and textual information.

While Grounded SAM has made significant strides in computer vision research, there is an ongoing debate about its applicability in medical technology. The paper mentioned [38] highlights that medical image segmentation still requires certain supervision requirements. This observation may also be relevant to the case of battery segmentation, as further elaborated in subsequent sections.

Chapter 5

Performance Experiments and Results

As mentioned earlier in the problem statement in Methods Chapter, this section will elaborate on the intricacies of each problem statement, providing a detailed account of the Objectives, Steps Taken and Inferences

5.1 Creation of Custom data set:

5.1.1 Objective

Upon initiation of this project, an initial search for images was conducted across renowned datasets such as COCO Dataset, yielding a limited number of images related to batteries. Subsequently, efforts were directed towards the Roboflow universe, where the discovered images proved to be overly simplistic. Consequently, a decision was made to train a model using a set of 27 generic images. the link is <https://universe.roboflow.com/school-gchcr/batteries-1aib9/browse?queryText=&pageSize=50&startIndex=50&browseQuery=true> The observed phenomenon is that when presented with randomly selected generic battery images, the model exhibited excellent performance as shown in fig 5.1 and fig 5.2. However, when the same model was applied to predict images derived from real-world recycling industry settings, its efficacy was notably diminished as explained in figure 5.3

5.1.2 Steps Taken

As illustrated in Figure 5.3, there were instances where very few batteries were either recognized inadequately or not identified at all. The primary factor contributing to this discrepancy was the diverse backgrounds present in recycling industry images, such as floor surfaces, dust, and conveyor belt textures, making predictions challenging. In response, a strategy was devised to incorporate real-time images into the training process, resulting in improvements in prediction accuracy. All custom images are present in Github link <https://github.com/learnerrida/Battery-custom-dataset/tree/main>



Figure 5.1: Trained generic dataset with generic image Predictions-1

5.2 Comparison of image Detection Model using YOLOV5, YOLOV8 and Azure Custom AI

5.2.1 Objective:

A significant challenge in recycling industries lies in the disorganized distribution and identification of battery locations across the plant. This haphazard arrangement poses a risk of unexpected blasts from various locations. To address this issue, a solution was proposed to precisely identify the location of batteries. This involves capturing images from a drone camera to enhance the organization and safety of the recycling plant. In the subsequent stages of the plan, the selection of an image detection model became crucial. After conducting some research [10], it was decided to explore both YOLOv5 for its user-friendly operation and stability, and YOLOv8 for its reputed accuracy. The objective was to assess and determine which model proves to be more effective for the custom dataset at hand. Simultaneously, a request was made to evaluate the stability, mean average precision (mAP), and precision using Azure Custom AI. This consideration stems from the ease of training that this platform offers, catering even to non-developers. The aim is to explore the viability and performance of Azure Custom AI in comparison to the YOLOv5 and YOLOv8 models.

YOLO Model

One of the most notable advantages of training YOLOv5 and YOLOv8 includes:

- **Open Source Software:** Both YOLOV5 and YOLOV8 are open-source software, available for use at no cost. This accessibility contributes to their widespread adoption and facilitates experimentation and development without financial barriers. Even in the scenario where the software needs to be commercialized, the process is straightforward and user-friendly. [11]
- **Ease of Comparison:** Comparing YOLOV5 and YOLOV8 is streamlined as the metrics to be measured remain consistent. This simplifies the evaluation process, and results of this comparison will be presented in the subsequent section.

Azure Custom AI

Azure AI Custom visions has the following observations:

- The implementation of Azure Custom AI stands out for its simplicity, enabling users with no coding experience to easily train models. Nevertheless, it's crucial to acknowledge that this convenience comes at a cost. Azure Custom AI is considered one of the pricier solutions due to charges associated with services such as Vision Studio, Custom Vision resource, and image storage
- Another notable feature is that, in the event of incorrect predictions, immediate adjustments can be made to the model by modifying the labels. This allows for enhancing the precision of the model without the need for retraining from the beginning.

The objective is to compare the precision of these models under identical conditions, including the baseline data set and the same proportions for training, validation, and testing data distributions.

5.2.2 Steps Taken

A trial was performed utilizing three distinct approaches as shown in the figure 5.5

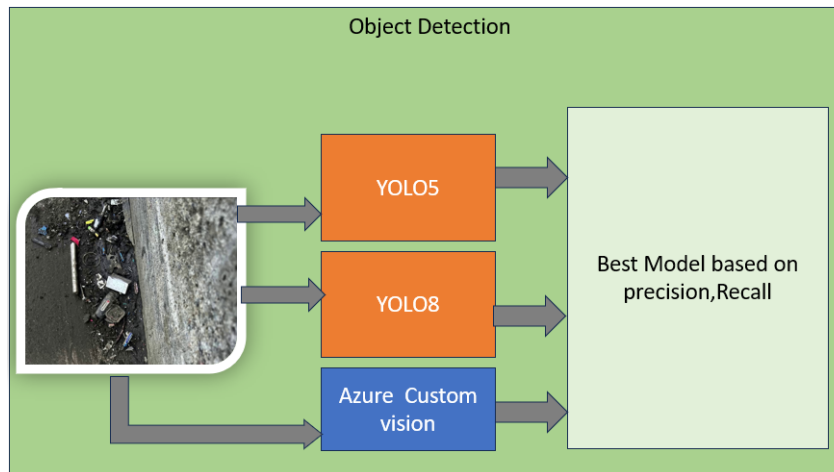


Figure 5.5: Experiment Approach

YOLOv5 vs YOLOV8vs Azure Custom AI

- Manually annotating the complete dataset through Roboflow and subsequently training it with the Yolo model.
- Employing automatic annotations on the customized dataset through Grounded SAM and Grounding Dino, followed by training the model with YOLO.
- Manually annotating the customized dataset using Azure Custom Vision specifically tailored for battery data.

A- Manual Annotation of Data set using Roboflow

For this experiment, first of all, Data is collected and uploaded in Roboflow and then manual annotations were done and then the data set is fed into yolov5xu.pt model for 186 MB of data for 100 epochs. The results of Box loss, Class loss and Object loss was calculated. The trained models are present in the following URLs. The results are displayed in fig 5.6 -5.10

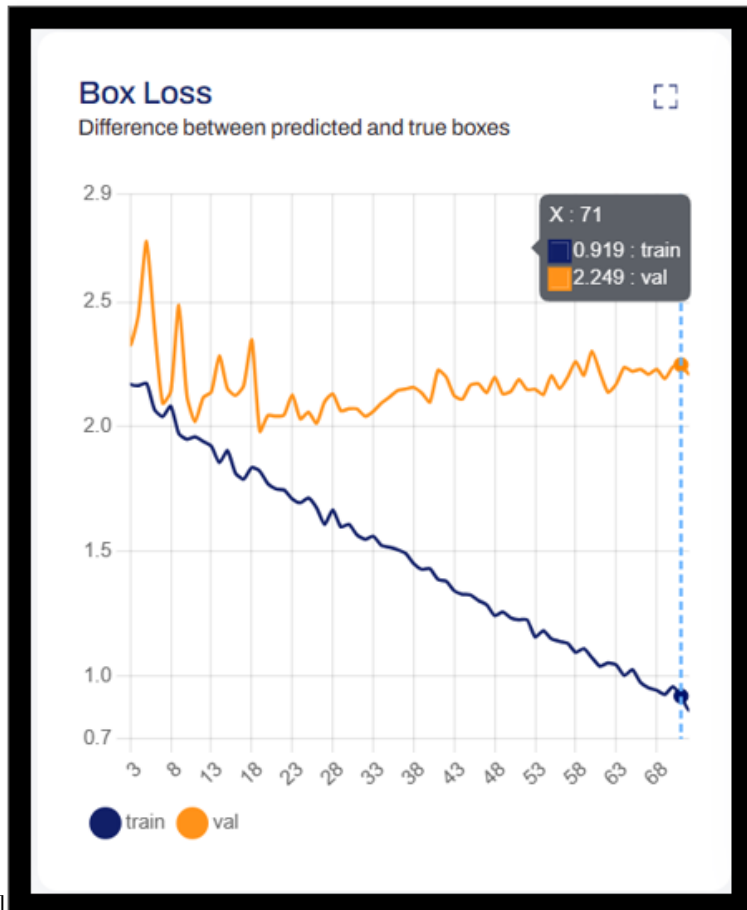
YoloV5xu Trained Model for detection -

<https://hub.ultralytics.com/models/LX1wdhvNAwnvMndtKGw4>

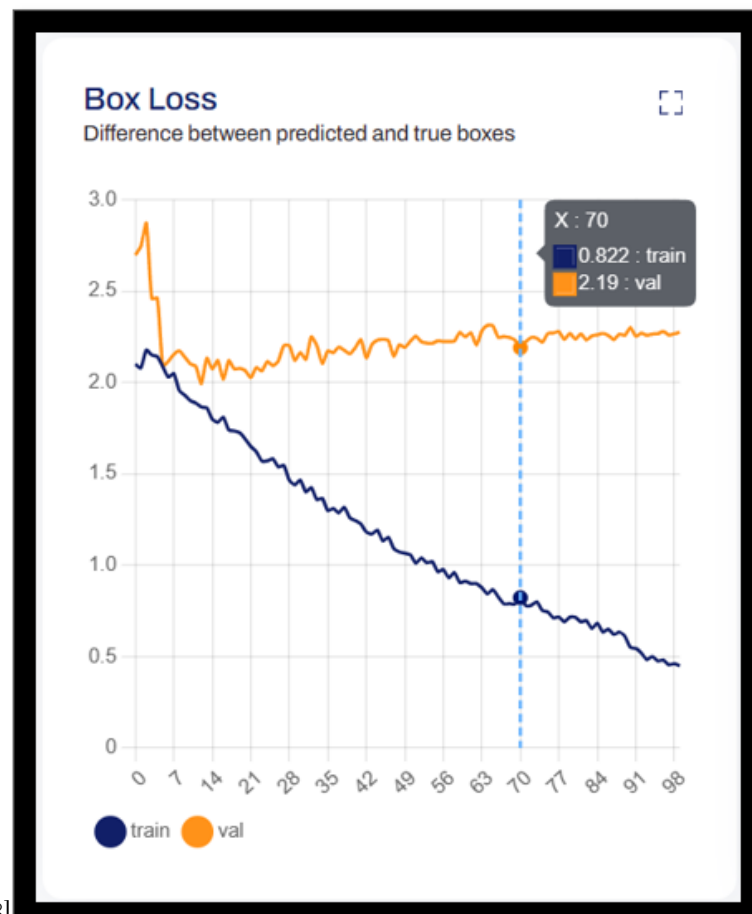
YOLOv8x Trained Model for detection -

<https://hub.ultralytics.com/models/YEA2yKPgWckbpzcpj9Qr>

Azure API - Is domain restricted API and hence could not share the link

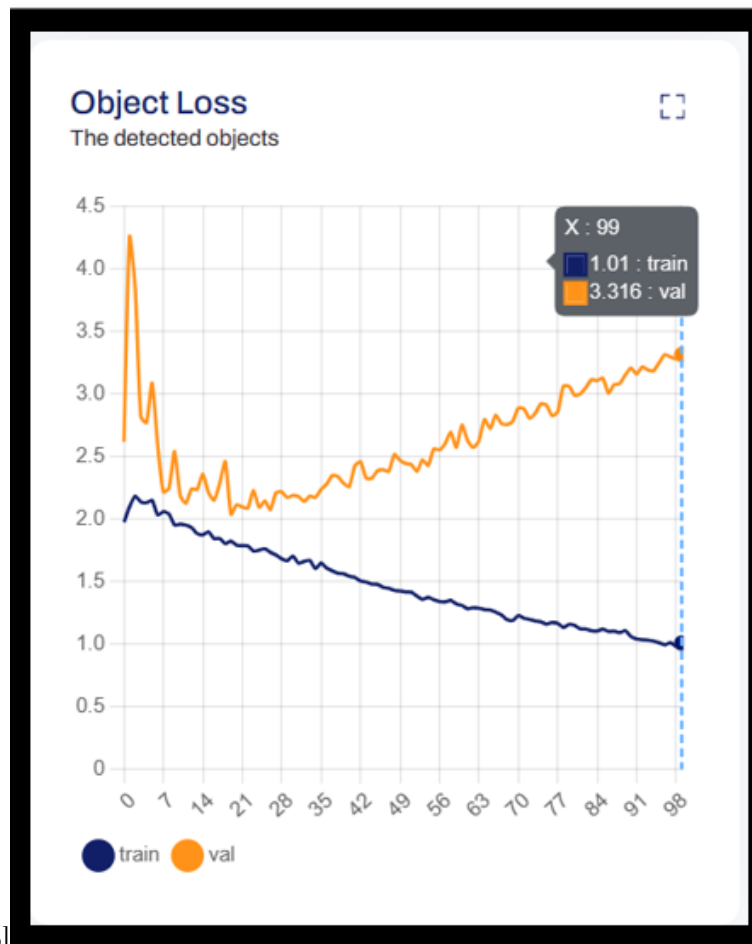


[Yolov5]



[YOLOV8]

Figure 5.6: YOLOV5 vs YOLOV8 Box Loss



[Yolov5]



[YOLOV8]

Figure 5.7: YOLOV5 vs YOLOV8 object Loss

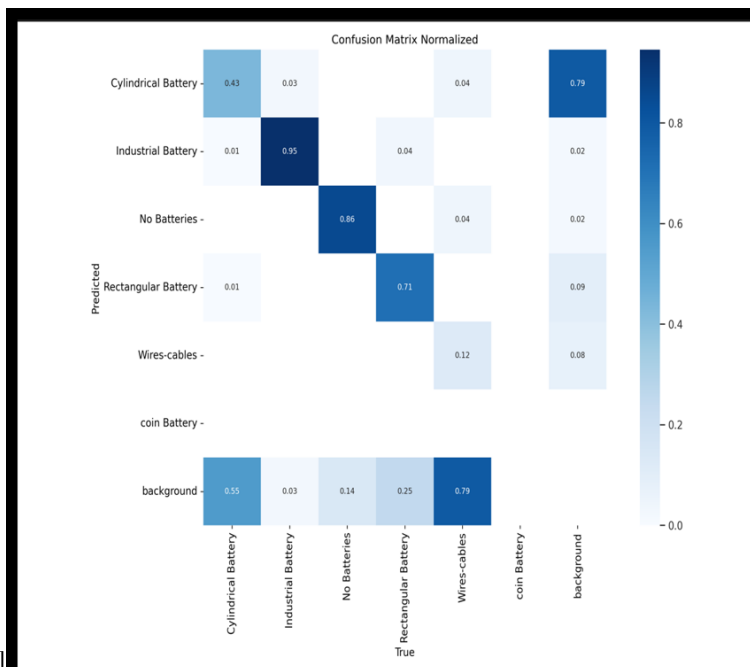


[Yolov5]

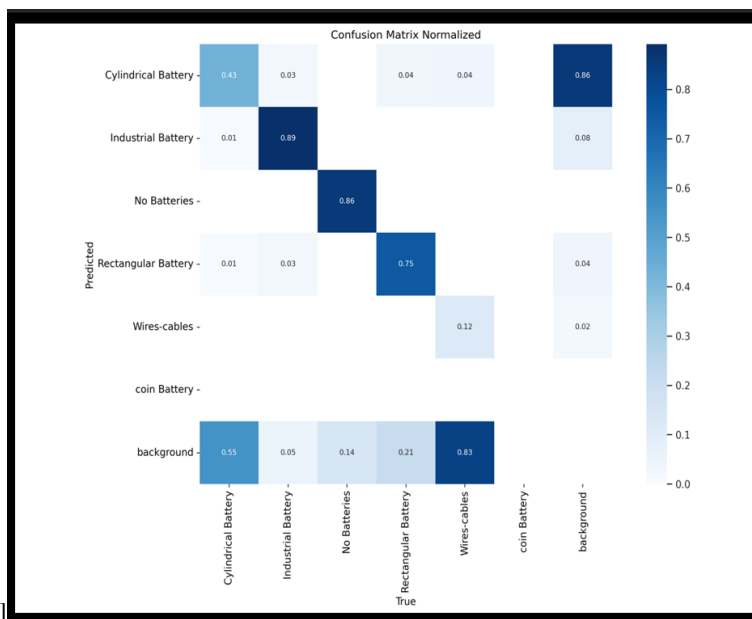


[YOLOv8]

Figure 5.8: YOLOv5 vs YOLOv8 class Loss



[Yolov5]



[YOLOV8]

Figure 5.9: Confidence Score Yolov5 vs Yolov8

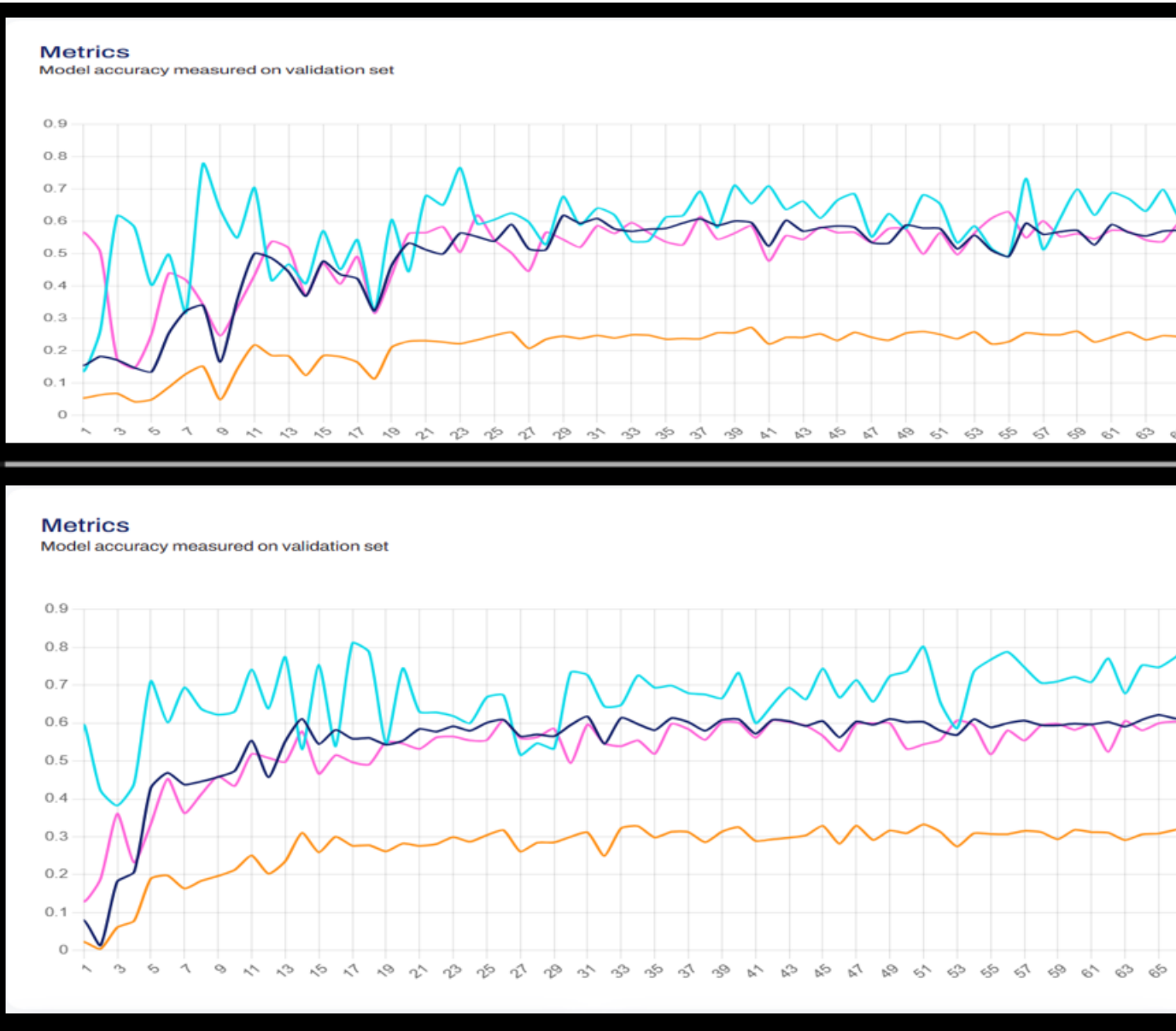


Figure 5.10: Overall Metric Performance in YoloV5vs YOLOV8

B - Manual Detection using Azure Custom vision AI

Implementation steps

- Step1: Login to Azure portal <https://www.customvision.ai/>
- Step2: Create a project as shown in the figure 5.11
- Step3: Upload Images
- Step4: Unlike Roboflow, Azure requires minimum of 15 tagging of images based on the classes. In our example, it was 6 classes like batteries, industrial batteries and so on
- Step5: Once trained, prediction of images can be done as shown in figure 5.12 and 5.13
- Step6: Overall mAP, precision and recall can be seen in the figure for same custom dataset which is fed into Azure Custom AI training as shown in the figure 5.15
- Step7: APIs can be invoked to be used in SDKs and hence this is one of the easiest ways to train model as shown in the figure 5.14

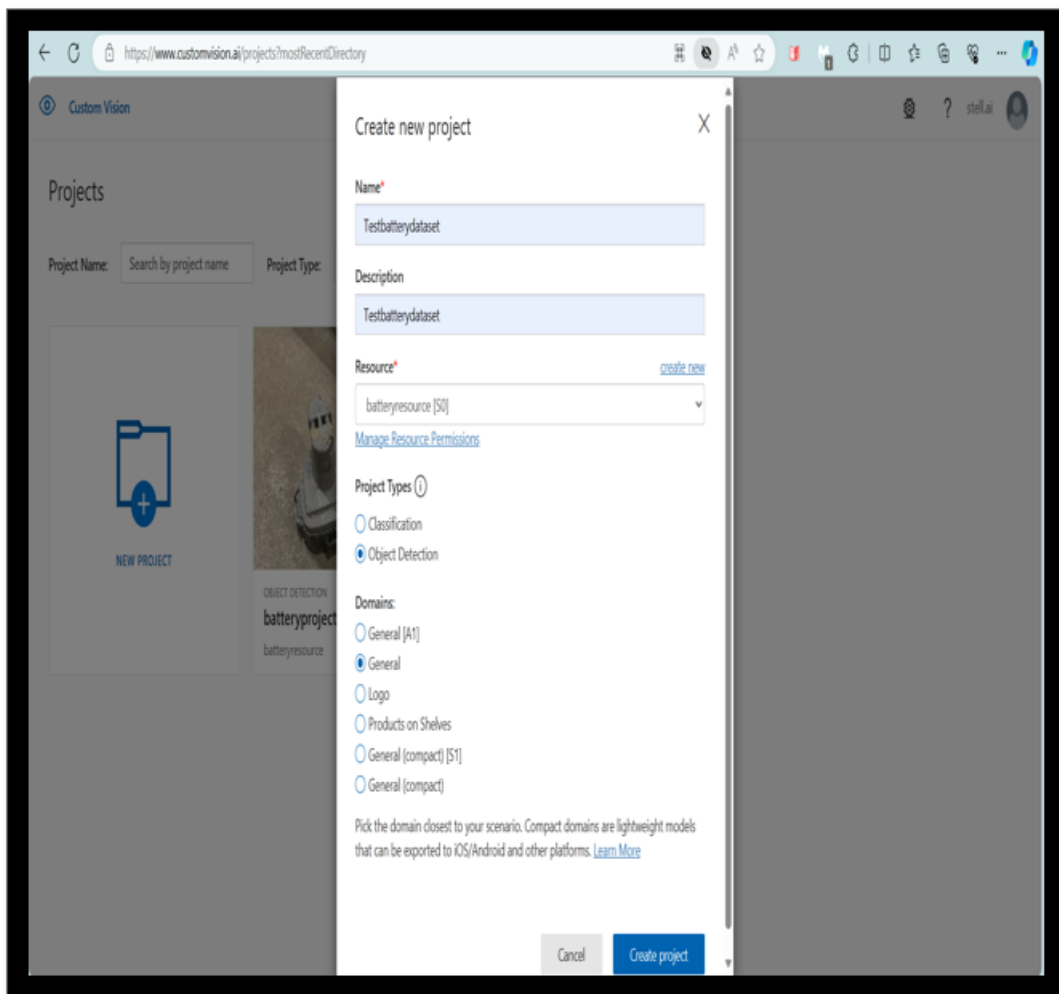


Figure 5.11: Azure Custom AI Project creation

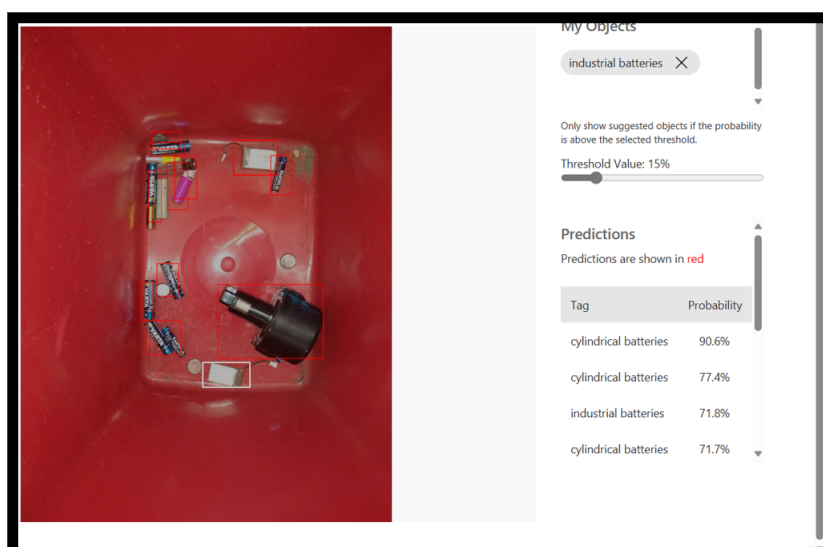


Figure 5.12: Azure Custom AI Predictionresult1

5.2.3 Inference:

Comparison of Performance Metrics

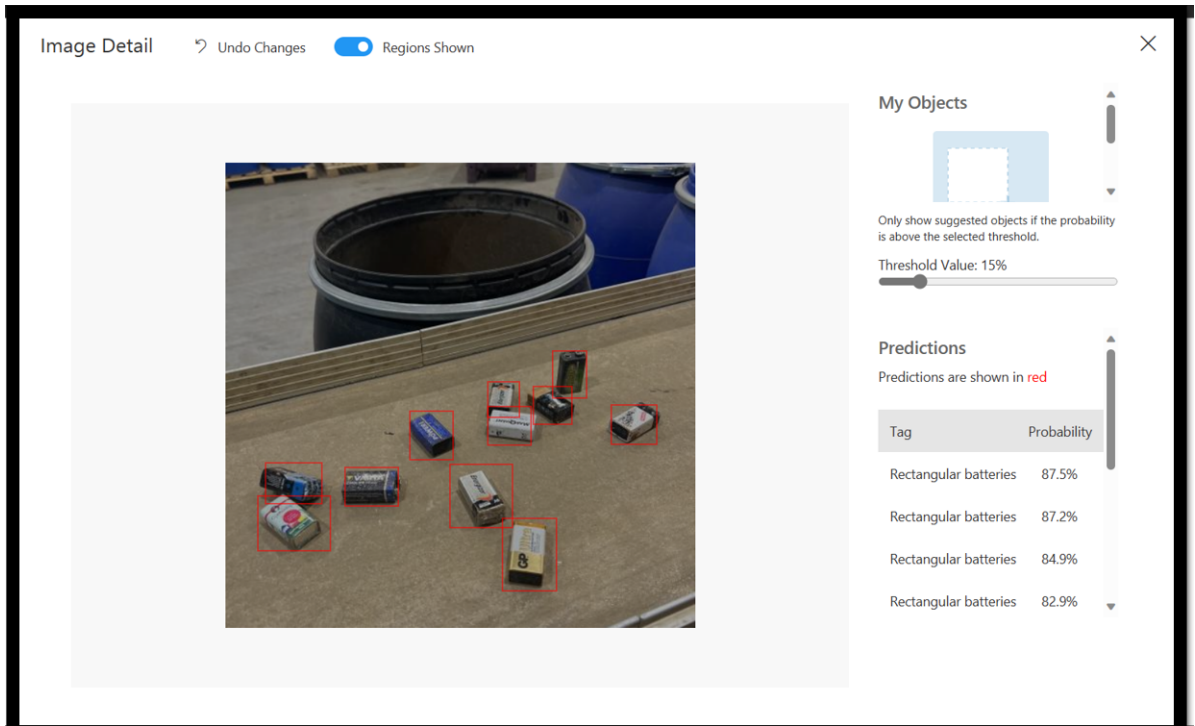


Figure 5.13: Azure Custom AI Predictionresult2

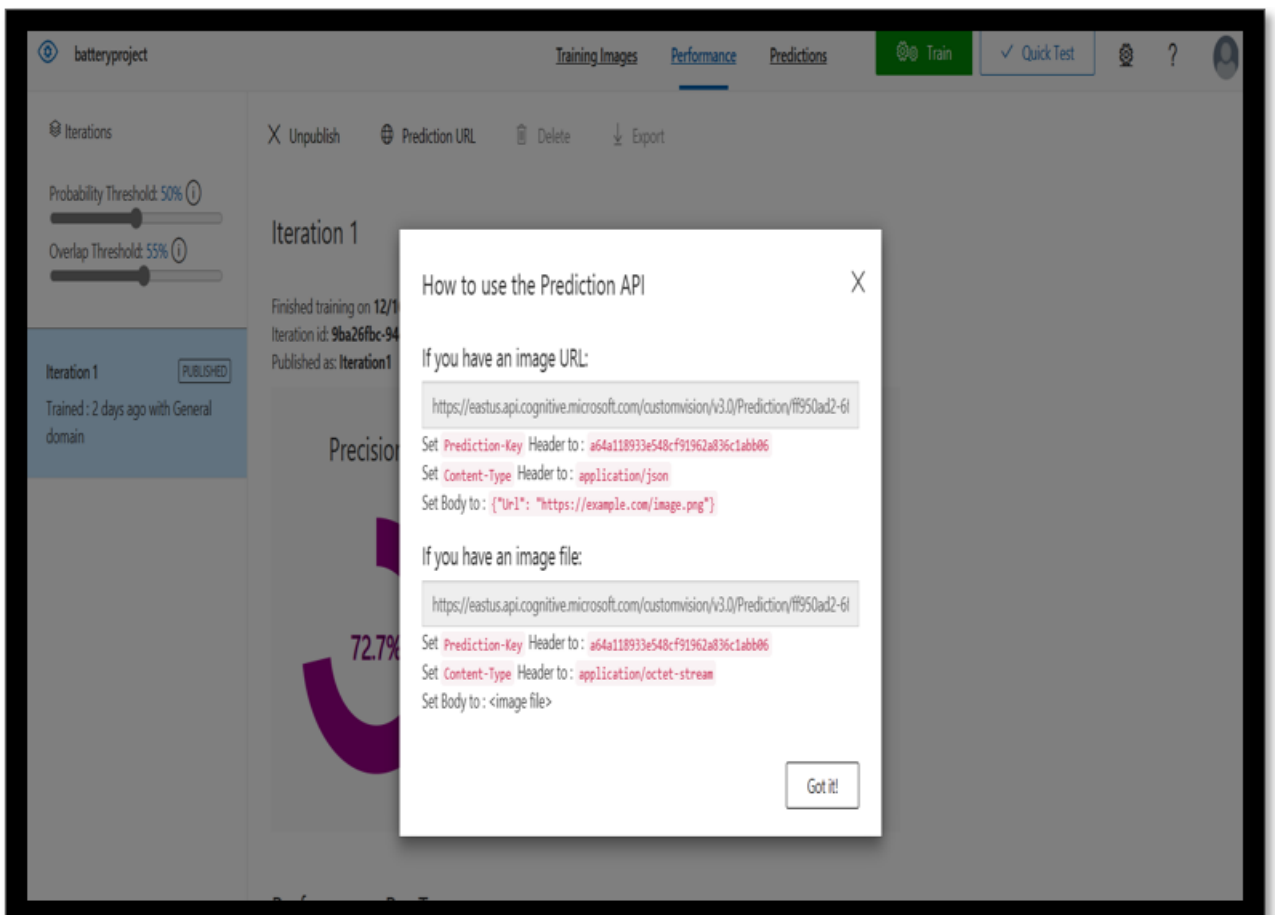


Figure 5.14: Azure Custom AI API

Yolov5 vs Yolov8 vs Azure Custom AI

In the comparison between YOLOv5 and YOLOv8, the loss graphs depict the training progress. The x-axis represents epochs, while the y-axis represents percentages. An ideal

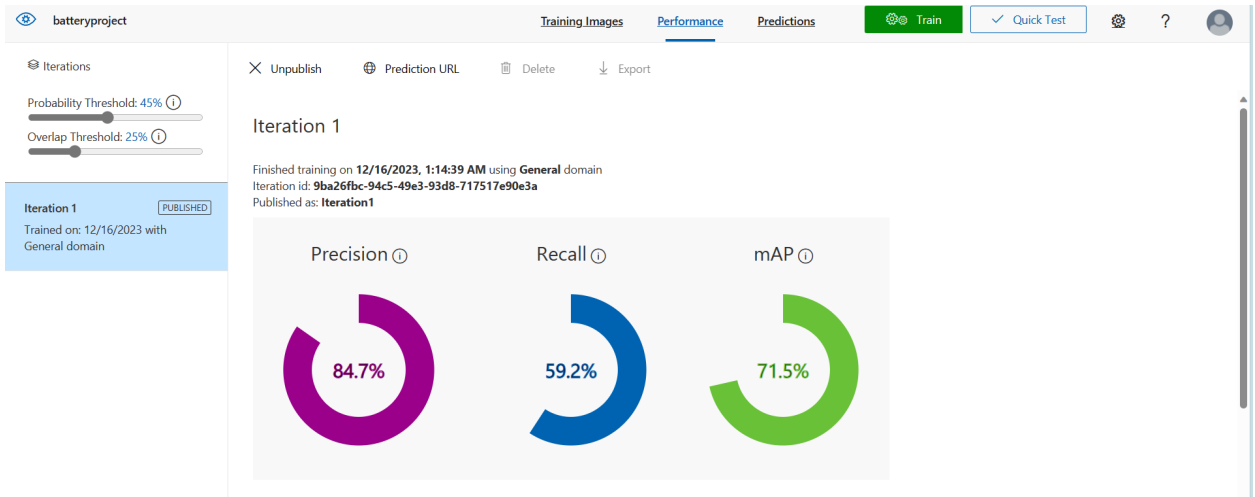


Figure 5.15: Azure Custom AI Dashboard

Comparison Metrics	YOLO5xu	YOLOv8x	Custom AI
Dataset training	296 Custom images	296 Custom images	225 Custom images
Annotations	Manually using Roboflow	Manually using Roboflow	Manual Training
mAP50(B)	0.59	0.613	0.715
mAP50-95(B)	0.278	0.354	Common mAP
Precision	0.733	0.796	0.847
Recall	0.596	0.597	0.592
Image Size	640	640	Min:256,Scale ratio:25:1

Table 5.1: Comparison of metrics of YoloV5detection model vs YOLOV8 detection modelvs Azure Custom AI

loss graph would show a decrease in the y-axis (loss) as the number of epochs increases. Overfitting occurs when the loss approaches zero.

For all losses- box loss,class loss and object loss, the training model exhibits a decreasing trend with an increasing number of epochs on the x-axis. However, the validation set does not follow the same pattern. This discrepancy suggests that during validation, the model recognizes that the data is not being detected as expected, leading to a drop in precision, recall, and overall loss.

- Differences in Prediction results In the output directory, when identical images were observed for validation, as illustrated in the figure below,fig 5.16 and fig 5.17, it is evident that in some of the images, the YOLOV5 model failed to correctly identify the objects and vice versa.
- The total time taken for training a dataset in YOLOV8 was comparatively lower than YOLOV5 and based on the metrics
- Both the models exhibit Bird’s eye view of prediction proficiently well :Both YOLOV5 and YOLOV8 exhibit proficient detection capabilities for smaller batteries when observed from a bird’s eye view as shown in the fig 5.18 and 5.19
- Comparison of Precision and Recall While plotting graphs of Precision and Recall for both the models, YOLOV8 outsmarts YOLOV5 by marginal numbers as shown in the figure 5.20
- In Azure Custom AI, the mAP (mean Average Precision) score is reported as 71.5%, with a precision score of 84.7% and a recall of 59.2%. These scores indicate that the model exhibits a lower rate of false positives, resulting in a higher precision score.

However, the higher number of false negatives contributes to a lower recall score. In situations where there is an imbalance between recall and precision, the F1 score may be affected, reflecting the trade-off between these two metrics.

- The reported confidence percentages for all three models are consistent at 0.25%. YOLOv8 achieving 61.3% and Azure Custom AI achieving 71.5% for image detection suggest effective performance of these models in identifying objects with confidence.



[Yolov5]



[YOLOV8]

Figure 5.16: Validation of predicted Object Detection in YoloV5 and YOLOv8



[Yolov5]



[YOLOv8]

Figure 5.17: Validation of predicted Object Detection in YoloV5 and YOLOv8



[Yolov5]



[YOLOV8]

Figure 5.18: Bird's eye view validation of images in YoloV5 and YOLOv8

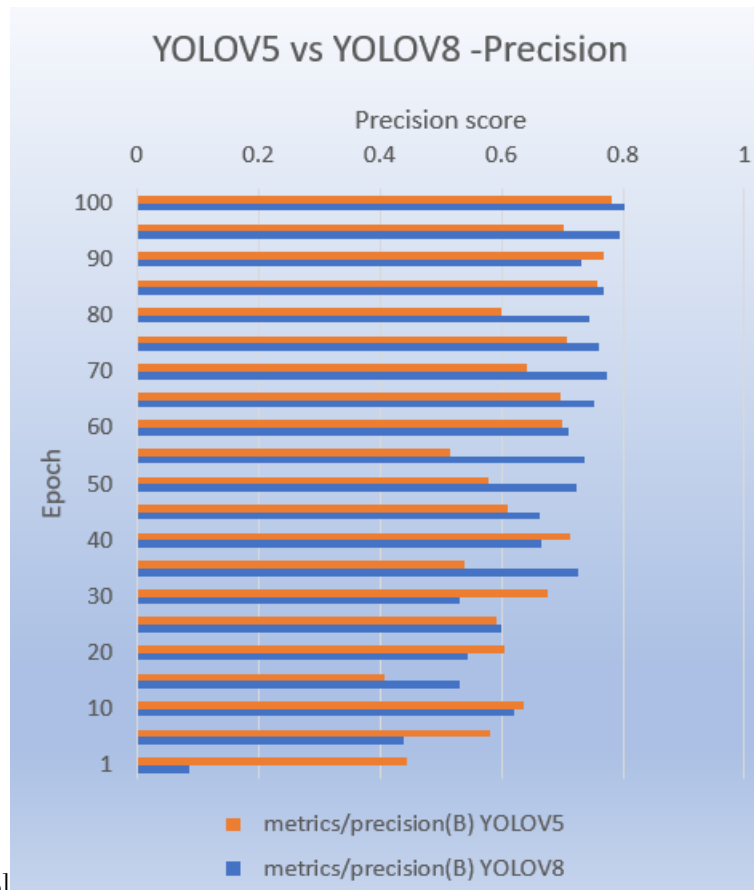


[Yolov5]

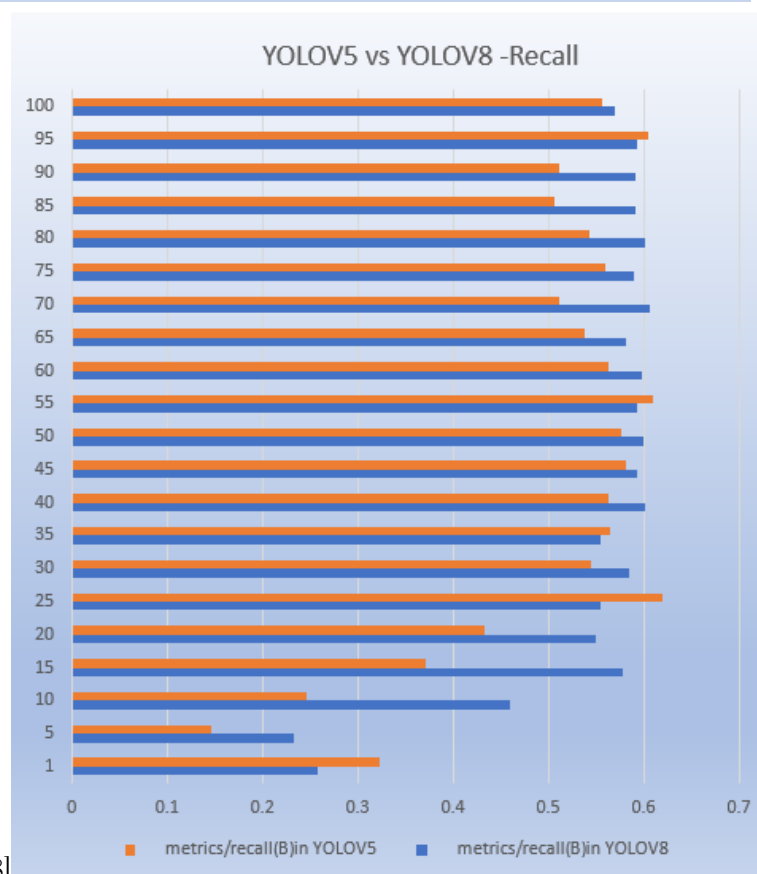


[YOLOv8]

Figure 5.19: Bird's eye view validation of images in YoloV5 and YOLOv8



[Yolov5]



[YOLOV8]

Figure 5.20: Precision and Recall Differences in Yolov5 and Yolov8 across 100 Epochs

5.3 GroundingDINO

5.3.1 Objective

While GroundingDINO is designed as a zero-shot object identifier, its performance metrics are assessed within the context of Grounded SAM. This experiment aims to evaluate the effectiveness and capabilities of GroundingDINO in practical applications.

For image detection, data was input into GroundingDINO. GroundingDINO is a pre-trained model that doesn't require additional training as it is already trained on a diverse set of images. This indicates that GroundingDINO comes pre-trained on a vast database. However, the purpose of this experiment is to assess how well this model performs in the specific use case of recycling industry batteries. One intriguing aspect of GroundingDINO is its ability to classify objects without getting confused by the background. On the contrary, the different distribution of environments and the presence of diverse objects like woods and logs in the dataset, can indeed impact the performance of GroundingDINO. It's crucial to recognize that models like GroundedSAM heavily rely on the characteristics and patterns present in their training data from GroundingDINO. In cases where the dataset includes a variety of environments and objects not well-represented during training, the model may struggle to accurately generalize to such scenarios.

The higher incidence of false positives, particularly in the presence of wood and logs, can lead to a decrease in precision. Addressing this issue may involve strategies as suggested by [12] such as:

1. Data Augmentation: Expanding the diversity of the training dataset through techniques like data augmentation, ensuring the model encounters a broader range of scenarios during training.
2. Fine-Tuning: If feasible, fine-tuning the model on a dataset that more closely resembles the testing environment can enhance its ability to handle specific challenges.
3. Adjusting Thresholds: Experimenting with confidence thresholds for predictions may help in filtering out false positives, although it requires careful consideration to avoid affecting overall recall negatively.
4. Curated Dataset: Ensuring that the dataset used for training is well-curated, balanced, and representative of the target environment can contribute to better generalization.

By iteratively refining the model and addressing specific challenges in the dataset, the performance of GroundingDINO in diverse environments can be improved.

5.3.2 Steps Taken

GroundingDINO can be installed and can be used by 4 lines of script[38].

```
%cd {HOME}
!git clone https://github.com/IDEA-Research/GroundingDINO.git
%cd {HOME}/GroundingDINO
!pip install -q -e .
```

by this code, GroundingDINO model can be downloaded directly from github .To configure a model there are two important things which are required Configuration path and weights which has already been downloaded in a previous step

```
IMAGE_NAME = "dataset/Image1.jpg"
IMAGE_PATH = os.path.join(HOME, "data", IMAGE_NAME)
```

```

TEXT_PROMPT = "battery. wires. cylindricalbattery. rectangularbattery."
BOX_TRESHOLD = 0.30
TEXT_TRESHOLD = 0.25

image_source, image = load_image(IMAGE_PATH)

boxes, logits, phrases = predict(
    model=model,
    image=image,
    caption=TEXT_PROMPT,
    box_threshold=BOX_TRESHOLD,
    text_threshold=TEXT_TRESHOLD
)

annotated_frame = annotate(image_source=image_source, boxes=boxes, logits=logits, phrases=phrases)

%matplotlib inline
sv.plot_image(annotated_frame, (10, 10))

```

In this section of the code, an input image is provided. In the second part, a text prompt is given to instruct the algorithm on what objects to detect. Additionally, Box Threshold and Text Threshold are specified, indicating the desired confidence percentage for the model to detect an object. In the example mentioned, the model is configured to detect objects with a confidence level above 30%.

5.3.3 Inference

GroundingDINO performs well in identifying batteries of various shapes, such as cylindrical, circular, rectangular, cubic, and cuboidal. It is capable of identifying even small batteries, as demonstrated in the example where a tiny battery was successfully detected as shown in fig 5.22. However, it's important to note that GroundingDINO associates the concept of a "battery" with its shape as shown in the figure 5.21. In cases where images contain objects that resemble the shape of a battery, such as wood logs as shown in fig 5.22 with a rectangular shape, the model may falsely identify them as batteries, albeit with lower confidence. In the experiment involving approximately 150 images, GroundingDINO demonstrated strong performance in environments with real batteries but produced false positives in environments with different objects like wood logs. In fig 5.23, even Microwave oven is identified as a battery falsely

The visual examples provided highlight the strengths and limitations of GroundingDINO in battery detection:

1. Grounding DINO-Prediction 1: The model successfully predicts and identifies batteries correctly, showcasing its robust performance in detecting batteries irrespective of the background.(fig 5.21)
2. Grounding DINO-Prediction 2: Some wood logs with a rectangular shape are mistakenly identified as batteries. This indicates a limitation where the model might associate the rectangular shape with the concept of a battery, leading to false positives.(fig 5.22)
3. Grounding DINO-Prediction3: In this case, a microwave is incorrectly detected as a battery. This demonstrates a scenario where the model may misinterpret objects that do not conform to the expected characteristics of a battery.(fig 5.23)

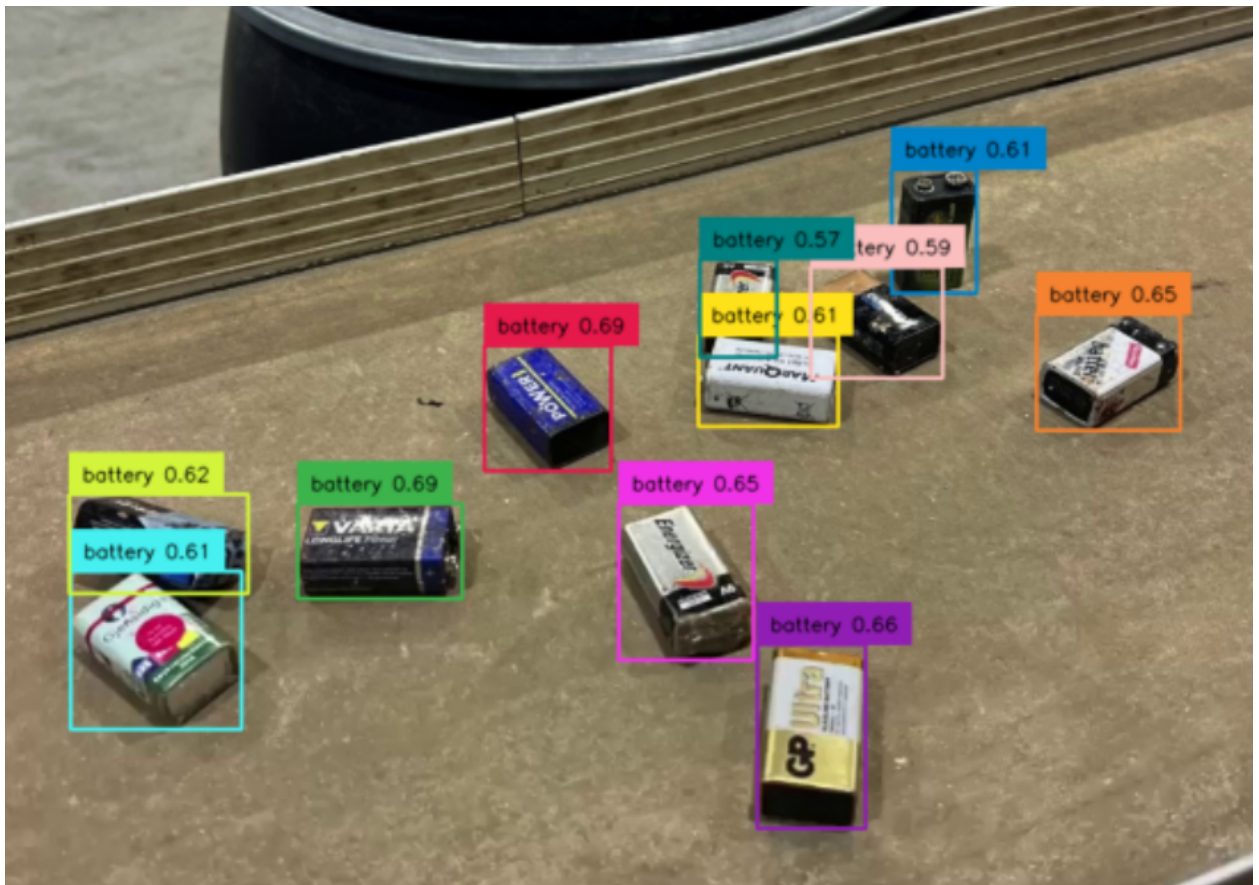


Figure 5.21: GroundingDINO-Prediction1



Figure 5.22: GroundingDINO-Prediction 2



Figure 5.23: Grounding DINO-Prediction 3

These examples emphasize the importance of understanding the context and potential challenges associated with zero shot object detection models like GroundingDINO, especially in scenarios with diverse objects and backgrounds. Further refinement and fine-tuning may be necessary to address specific false positive cases and improve overall accuracy.



Figure 5.24: Annotated Images from Roboflow

5.4 Image Segmentation using Roboflow, YOLO Models and Grounded SAM

To address the concerns outlined in problem statement 3, a solution involves capturing images when objects, including keyboards, bulbs, lamps, and batteries, are in motion on a conveyor belt. The crucial aspect is to perform segmentation specifically for batteries, ensuring their distinct identification in the midst of various materials. Therefore, the segmentation process was executed using YOLO models. In this section, segmentation using both YOLOv5 and YOLOv8 was undertaken, alongside instance segmentation utilizing Grounded SAM. The outcomes of these segmentation methods were compared to determine the most effective model for application in the product.

5.4.1 Objective

GroundedSAM employs a zero-shot image detection and segmentation technique, enabling autonomous detection of images. Nevertheless, when applied to custom data, numerous false positives were observed. Additionally, training data with GroundedSAM and GroundingDINO becomes time-consuming, especially when dealing with large image resolutions. Upon uploading the GroundedSAM dataset into the YOLO model for metric calculation, it was observed that the precision obtained was lower compared to a manually annotated and trained YOLOv8 model. In the comparison between the YOLOv5 segmentation model and the YOLOv8 segmentation model, it was noted that YOLOv5 exhibits fast model training. However, in terms of instance segmentation, YOLOv8 outperforms both models, demonstrating superior performance.

5.4.2 Steps Taken

After manually annotating 60 images using Roboflow, the model was trained using yolov5s-seg.pt. and yolov8s-seg.pt respectively and the data set was called from the Roboflow API, and the obtained results are detailed as follows:

5.4.3 Inference

The following table consists of many parameters and each parameter explains the differences in performance such as :

Epochs

-All models were trained for 50 epochs

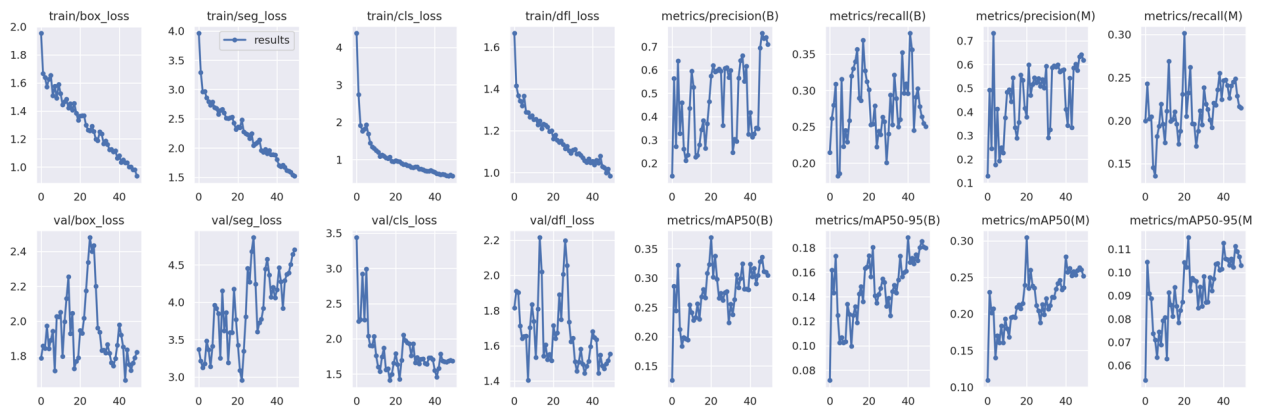


Figure 5.25: Instance Segmentation-Yolov8 Manually annotated images

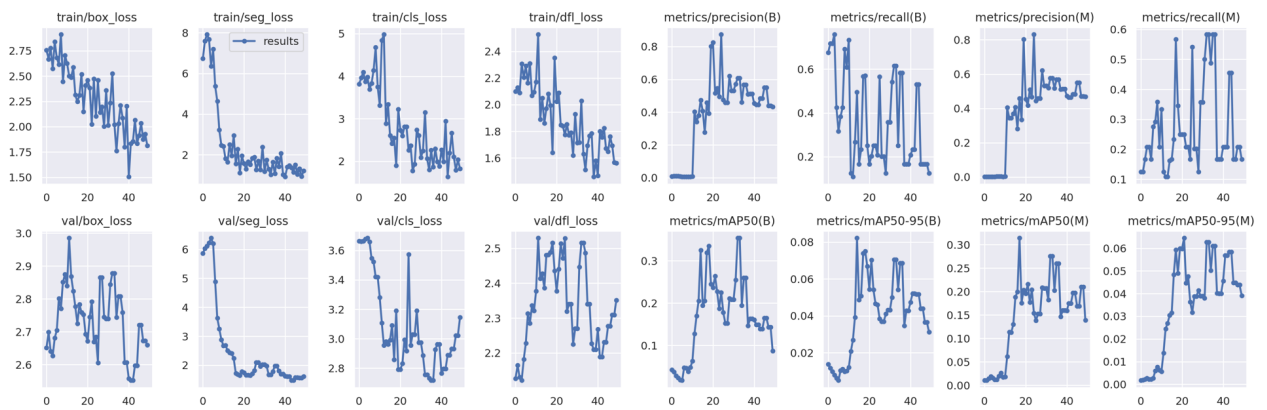


Figure 5.26: Instance Segmentation-Yolov8 using GroundedSAM

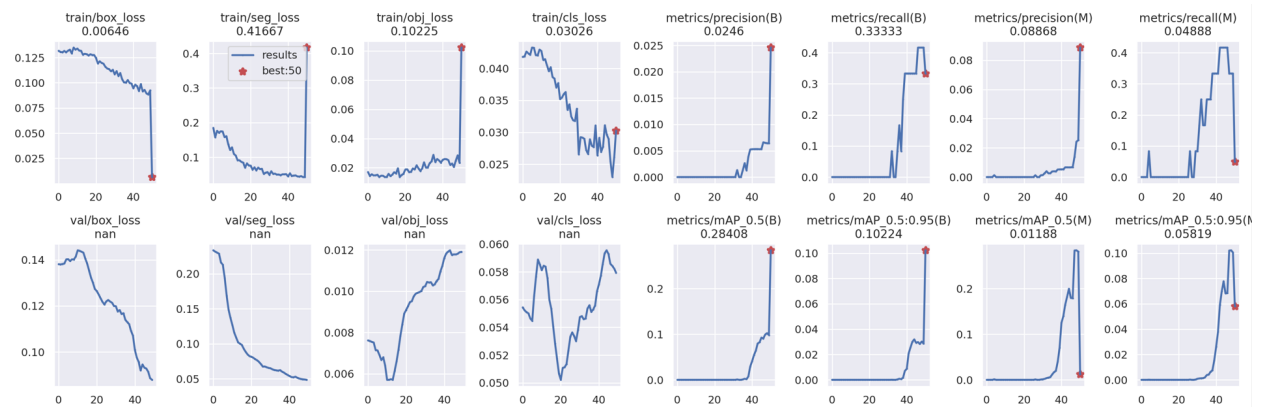


Figure 5.27: Instance Segmentation-Yolov5 Using Manually annotated images

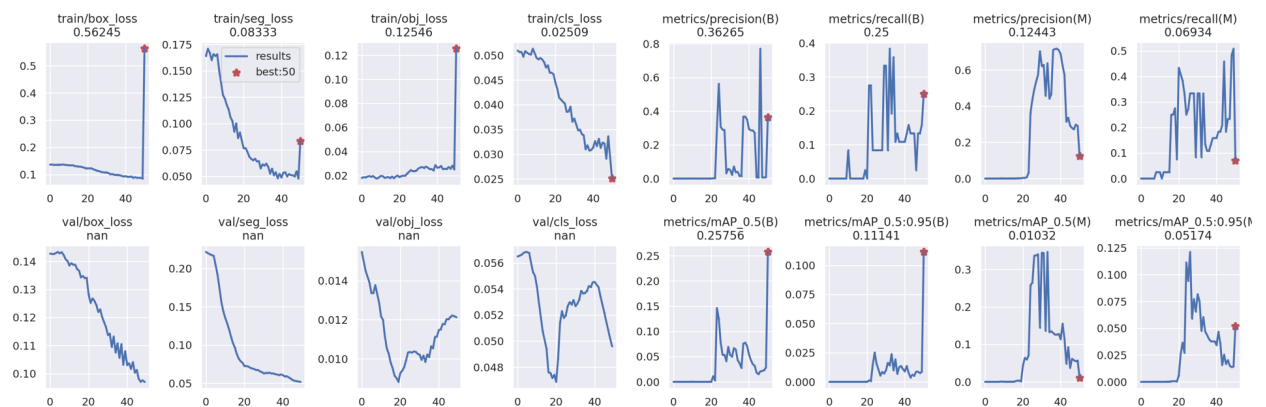


Figure 5.28: Instance Segmentation -Yolov5 using Grounded SAM

Dataset

- All Models were trained on 60 images which includes augmentations

train/box_loss

- This parameter means that more the box_loss value is more deviation exists between Ground truth and predicted value

train/seg_loss

- This parameter means that more the loss is deviation of segmentation annotated using smart polygons with predicted values differ a lot

train/cls_loss

- This parameter means that more the value is, more the loss of classification

train/dfll_loss

-DFL stands for distributed focal loss which is meant to optimize boundary box boundaries, the more compact the bbox is better the segmentation and hence if the value in this factor is more, wider is the boundary box

metrics/Precision

- These metrics will be repeated twice with B and M where B signifies Best of all values and M represents Macro average. In all these rows, more the value is better the performance and this holds true for various other metrics as follows:

- metrics/mAP50(B) Mean Average precision value with IoU confidence range less than 50
- metrics/mAP50-95(B) Mean Average precision value with IoU confidence range less than 95 and more than 50
- metrics/precision(M)
- metrics/recall(M)
- metrics/mAP50(M)
- metrics/mAP50-95(M)

val/box_loss

- This parameter means that more the box_loss value is more deviation exists between Ground truth and predicted value, This metrics is for Validation data

val/seg_loss

- This parameter means that more the loss is deviation of segmentation annotated using smart polygons with predicted values differ a lot. This metrics is for Validation data

Val/cls_loss

- - This parameter means that more the value is, more the loss of classification. This metrics is for validation

Segmentation	YOLOV8	Grounded SAM-YOLOV8	YOLOV5	GSAMYOLOV5
epoch	50	50	50	50
Dataset	60	60	60	60
train/box _{loss}	1.084752	2.26512	0.11302139	0.123149373
train/seg _{loss}	2.001162	2.4467446	0.089526784	0.086745941
train/cls _{loss}	1.5459318	2.792062	0.021174098	0.024479784
train/df _{loss}	1.0873678	1.88914	0.034321843	0.040543569
metrics/precision(B)	0.7138844	0.4	0.001921739	0.107298652
metrics/recall(B)	0.3027166	0.3666884	0.098039	0.096240667
metrics/mAP50(B)	0.2778142	0.167597	0.022145275	0.033259044
metrics/mAP50-95(B)	0.1613134	0.041063	0.00737086	0.008485959
metrics/precision(M)	0.6897268	0.3847366	0.004606759	0.261771252
metrics/recall(M)	0.2741694	0.2734862	0.123507451	0.175978863
metrics/mAP50(M)	0.235001	0.1480624	0.043327351	0.091815693
metrics/mAP50-95(M)	0.1137376	0.0362958	0.015972111	0.030961466
val/box _{loss}	1.764864	2.733034	0.12153008	0.12244308
val/seg _{loss}	3.045642	2.51341	0.0969484	0.09714994
Val/cls _{loss}	2.975968	3.079798	0.009203312	0.011306654
val/df _{loss}	1.768856	2.335256	0.05549036	0.05274348
lr/pg0	0.02767023	0.051749898	0.0763492	0.0763492
lr/pg1	0.003170231	0.00174983	0.000849169	0.000864499
lr/pg2	0.003170231	0.00174983	0.000849169	0.000864499

Table 5.2: Segmentation-Comparison

val/df_{loss}

- -DFL stands for distributed focal loss which is meant to optimize boundary box boundaries, the more compact the bbox is better the segmentation and hence if the value in this factor is more, wider is the boundary box. This metrics is evaluated for Validation

lr/pg0

- lr stands for Learning Ratio as all models for simplification creates batches and in this case its a learning rate for Pg0. Better the value is better the learning rate

lr/pg1

- lr stands for Learning Ratio as all models for simplification creates batches and in this case its a learning rate for Pg1. Better the value is better the learning rate

lr/pg2

- lr stands for Learning Ratio as all models for simplification creates batches and in this case its a learning rate for Pg2. Better the value is better the learning rate

All the Values obtained in a table is an average sum of values from 50 epochs

Again, it is clearly seen that YOLOV8 manually annotated data outperforms rest of the model. The results of validation batches can be seen in fig 5.29

5.5 Health Dashboard and Chat bot

5.5.1 Objective

A health dashboard is designed to provide users in a plant with comprehensive statistics on batteries, including different types of batteries at different timings. This enables users to identify patterns and trends in the battery data.

To address problem statement 4, two components were developed. Firstly, a visual dashboard was created to provide statistics on the uploaded data. Additionally, a specialized chat bot function was implemented, enabling users to query information from the generated data. The dashboard was created using Streamlit App, a web application developed with a Python library. As far as chatbot is concerned, its created by using the Lang chain with Pine cone vector database

5.5.2 Steps Taken

In Streamlit, code can be deployed either on Streamlit Cloud or through Docker or Kubernetes. For this thesis, the application is deployed on Streamlit Cloud. As part of the prerequisites, a GitHub link needs to be created, and the code is stored in a repository, providing access for the Streamlit app to access the GitHub code. In GitHub, after uploading the code, a requirements.txt file needs to be added, including all dependencies such as matplotlib, pandas, etc. Below is a sample output graph illustrating how the dashboard would appear. The sample code can be found at the GitHub location mentioned in the above section.??

The dashboard is also integrated with a customized chatbot using Langchain and Pinecone vector database. It provides specific outputs based on the input documents, enhancing the user interaction experience. For the purpose of demonstration, A table with login details and logout details were given as an input to show the results. In the future, a live stream video with a count of batteries will be incorporated into the dashboard

5.5.3 Inference


Visual dashboards have a positive impact as they make it easy for users to understand patterns. These patterns can be related to the type of data batch, timing, or even geographical locations. This would also serve as an effective proof of concept package.

StellAI Dashboard!!!

Welcome to our page! In StellAI, We are revolutionizing the circular economy by utilizing cutting-edge technology in artificial intelligence, neural networks, and vision systems in conjunction with robotics.

Our state-of-the-art identification and sorting system is not only cost-efficient, but it also has the power to make a significant impact on the environment.

upload file

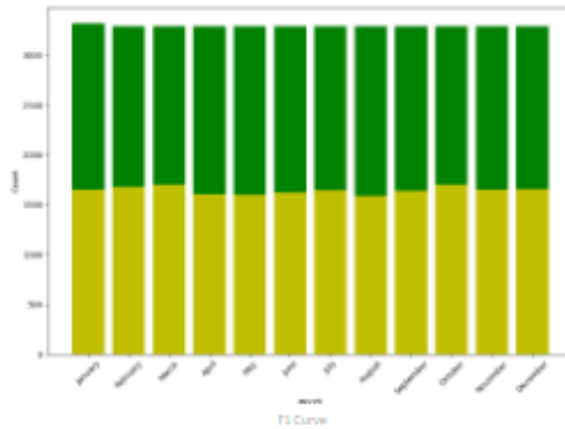


Drag and drop file here
Limit 200MB per file

Browse files

 xyz.xlsx 2.2MB ✕

Execute



Unnamed: 0.L	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9
26	26	26	04	PP	Woolite	857	614	1,012	192	0
27	27	27	04	PP	Woolite	857	614	1,012	192	0
28	28	28	04	PP	Woolite	857	614	1,012	192	0
29	29	29	04	PP	Woolite	857	614	1,012	192	0
30	30	30	04	PP	Woolite	857	614	1,012	192	0
31	31	31	04	PP	Woolite	857	614	1,012	192	0
32	32	32	04	PP	Woolite	857	614	1,012	192	0
33	33	33	04	PP	Woolite	857	614	1,012	192	0
34	34	34	04	PP	Woolite	857	614	1,012	192	0
35	35	35	04	PP	Woolite	857	614	1,012	192	0

Graphs

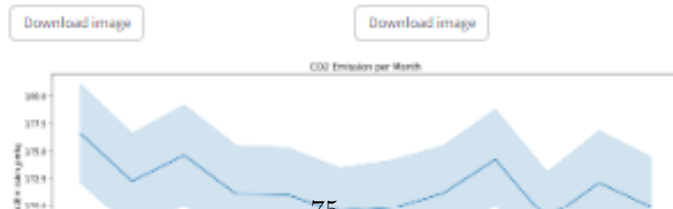
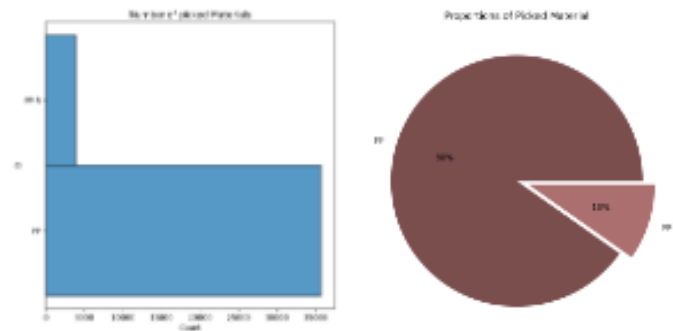


Figure 5.30: Health Dashboard



Figure 5.31: chatbot with langchain

Chapter 6

Discussions

6.1 Methodology and Approaches:

6.1.1 Effectiveness of manual annotations using Roboflow

In recent times, there has been significant progress and emphasis on the field of deep learning. Reflecting on the past six months since the initiation of this thesis, notable improvements are evident in the Roboflow interface. Specifically, for annotating image detections, the creation of ground truth boxes is crucial, demanding high precision, especially in instances where batteries are closely positioned in recycling datasets. Regarding instance segmentation, the requirement for polygon annotations has been streamlined with the introduction of a new feature in Roboflow known as "smart polygons." This recent addition simplifies the segmentation annotation process, a task that previously consumed considerable time when utilizing the polygon tool. The introduction of such features is not only intriguing but also holds significant importance for industrial applications, where data accuracy at the millimeter level is imperative for defect detection. Roboflow offers an additional benefit by allowing users to convert the dataset into their preferred format. The integration of Roboflow with Ultralytics, the developer of Yolov5 and Yolov8, proved to be an added advantage for our project. This integration facilitated the invocation of an API, enabling seamless integration with Ultralytics for training purposes. Ref fig 6.1

6.1.2 Impact of Auto annotations

Following this Proof of Concept (PoC) project, the intention is to develop a working prototype for recycling industries. The challenge lies in collecting thousands of images and manually annotating them, which can be a tedious task. While Roboflow offers an "auto label" option from a previous dataset, it still involves semi-supervised learning. Looking ahead, with the continuous advancements in computer vision, the expectation is that auto-annotations will become a future approach to training models. This method is anticipated to provide robust and feasible options, particularly for startup companies where allocating resources for manual annotations may not be practical.

It is truly remarkable how technologies like Grounding DINO and Grounded SAM, embodying state-of-the-art architectures, shape the future of deep learning. In the course of this thesis, a significant challenge emerged during the training of Grounded SAM, mainly due to its intensive computational requirements. Attempts to train the model with a dataset comprising 294 images consistently led to system hang-ups, even with the contribution of a single GPU in the hardware setup. In response to this constraint, the segmentation dataset was downsized to just 29 images, which were augmented to over 60 images using various preprocessing techniques such as flipping and contrast adjustments. This reduced dataset was then utilized for training both YOLOv5 and YOLOv8.

Information sourced from the Ultralytics page [12] indicates that achieving optimal instance

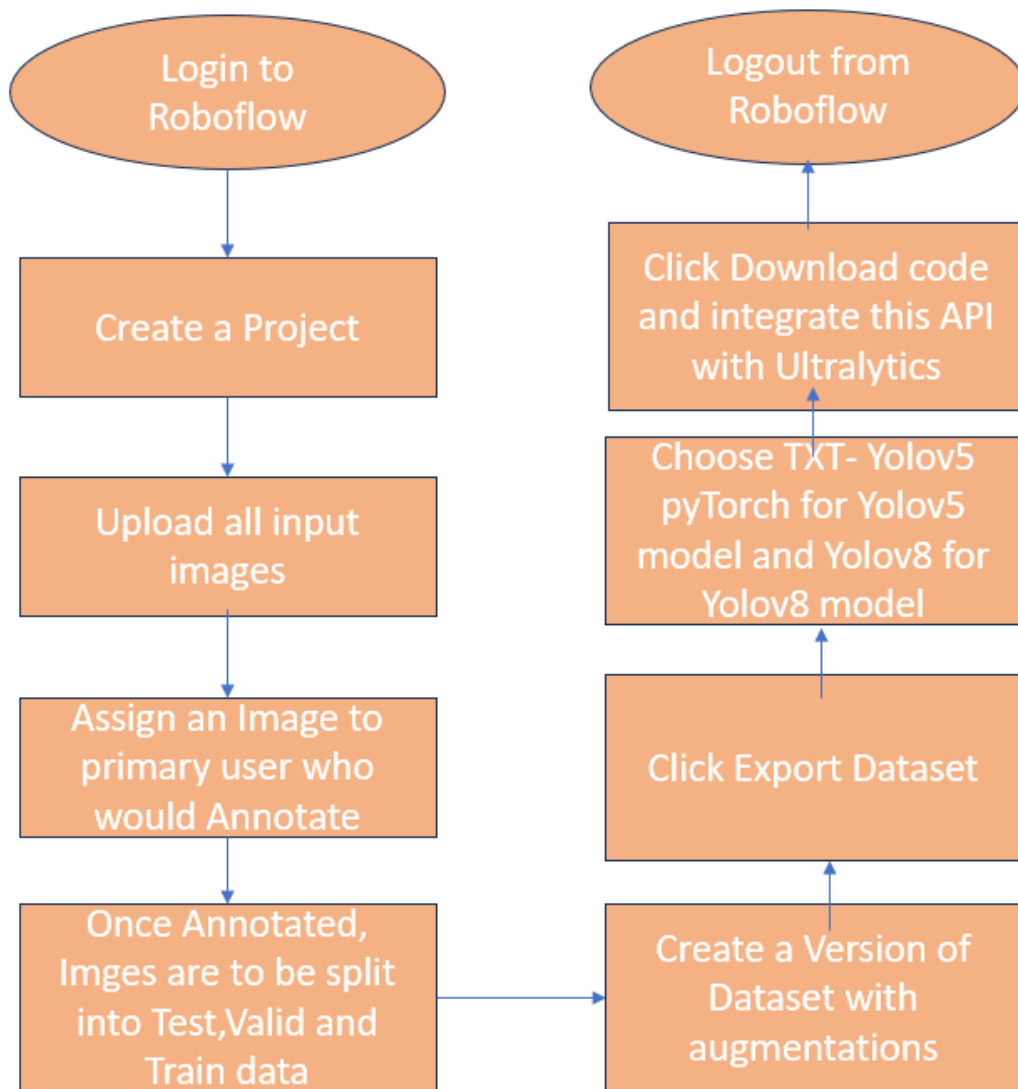


Figure 6.1: Roboflow workflow

segmentation results with Grounded SAM necessitates pretraining on a dataset comprising over 1000 images. However, given the recent development of Grounded SAM, future enhancements may lead to a reduction in computational demands, making it more viable for training on larger datasets.

6.1.3 Comparison of Results

Image Detection

In the domain of image detection, the results displayed in Table 5.1 reveal that the mAP50(B) is 59%, 61.3%, and 71.5% for YOLOv5, YOLOv8, and Azure Custom AI, respectively, with manually annotated datasets. The mAP50-95(B) scores stand at 27.8% and 35.4% for YOLOv5 and YOLOv8 models. Furthermore, when examining precision and recall, YOLOv8 with manually annotated images emerges as the top-performing model as explained in table 5.1.

Comparisons were conducted among YOLOv5, YOLOv8, and Azure Custom Vision for image detection. Additionally, a comparative analysis was performed between manual segmentation using YOLOv8 and YOLOv5 models and GroundedSAM data in both YOLOv8 and YOLOv5 models to assess and contrast the results. The selection of these models for comparison was deliberate and based on specific considerations. Firstly, YOLO models, namely YOLOv5 and YOLOv8, were chosen for two main reasons: 1. Both models are developed by Ultralytics, which provides robust API support and seamless integration with tools like Roboflow. 2. These models are considered state-of-the-art, and research papers [31][30] have conducted comparisons between YOLOv5 and YOLOv8, with conflicting findings on which model performs better in terms of detection capabilities.

Azure Custom AI was specifically selected for this project due to its relevance and alignment with a similar project undertaken by Azure in Norway. The Norwegian project, known as "esmartz," focuses on sustainable industry practices and involves the use of custom data. This made Azure Custom AI an ideal choice for the current project, ensuring consistency and compatibility with the goals and objectives of both initiatives. The deliberate omission of auto annotation for image detection using Grounded SAM stems from its reliance on GroundingDINO images as input for segmenting images. The performance of GroundingDINO has been rigorously tested through manual screening of all outputs, with the results and observations elaborated in Section 5.4.

Image Segmentation

In terms of segmentation, YOLOv8 exhibited a performance of 27.78% on manually annotated images, whereas Grounded SAM's segmented images, trained on YOLOv8, achieved a score of 16.67%. on average of metrics/mAP0.5(B) and 16.13% and 4.1% on metrics/mAP50-95(B) . Once again, in the realm of segmentation, YOLOv8 with manual annotations showcased superior performance compared to other models, as outlined in Table 5.2.

6.2 challenges

Challenge1

One of the initial challenges encountered was related to the dataset. Despite the vast number of images available in extensive datasets like ImageNet and COCO, acquiring a sufficient number of battery images posed a significant challenge. The initial dataset was scraped from Roboflow Universe, consisting of generic images. However, when the model was trained on YOLOv8 and subsequently implemented as a mobile app using Ultralytics Mobile App with a custom dataset, issues arose. The model struggled to detect batteries in diverse

backgrounds, and problems with image resolutions were evident. Consequently, collaboration with the CEO of StellAI was necessary, involving the capture of real-time images. This led to a restart of the annotation, training, and comparative analysis processes. Despite the challenges, sourcing a battery dataset tailored to the recycling industry proved to be a complex task.

Challenge 2

The selection of a suitable model and technology for deep learning in image detection and segmentation posed a considerable challenge due to the abundance of state-of-the-art models. The decision-making process involved extensive brainstorming, with a primary focus on creating a sustainable solution for industrial applications. While YOLOv6 was designed specifically for industrial datasets, the versatility and recent advancements observed in YOLOv5 and YOLOv8 from Ultralytics made them potential candidates for sustainability. The choice was further influenced by the seamless integrations these models offer with various technologies. Additionally, Microsoft Azure was considered as a plug-and-play technology, ensuring accessibility for users without a development background. This alternative was carefully evaluated to cater to a broader range of customers.

Challenge 3

As detailed in an earlier section, the challenge of training GroundedSAM on multiple images with higher resolution led to the decision to train the model in multiple batches. However, this approach resulted in challenges, such as the creation of duplicate projects in Roboflow when collecting mini-batches of images. This added complexity and significantly extended the time required, surpassing even the annotation process for the 300 images. Due to these constraints, the image dataset was restricted, limiting the potential for more extensive sampling. It is acknowledged that expanding the dataset could have contributed to improved efficiency. In the Idea Research GitHub repository, issues related to the performance of GroundedSAM are documented. There are speculations that GroundingDINO might be contributing to the slowing down of GroundedSAM. It is suggested that waiting for this model to mature further before retraining might be a prudent approach [38]

6.3 Dataset Considerations

Fei Fei Li, a distinguished scientist, adopted a unique perspective in 2006 by focusing on image datasets while many others were engrossed in improving and creating new models[13] Her groundbreaking work on the ImageNet idea, coupled with the creation of a comprehensive dataset, has played a pivotal role in the flourishing field of computer vision. While the advancements in models are awe-inspiring, it's crucial to acknowledge the significant role played by datasets and training datasets. In the context of this thesis, the emphasis is on collecting a specific dataset for the recycling industry, particularly a battery dataset. Training this dataset with state-of-the-art models is expected to enhance efficiency significantly. The plan is to continue collecting more images and continually train them on state-of-the-art models to identify batteries effectively, serving the intended purpose.

6.4 Future Directions and Improvements:

- In prior discussions, it was emphasized that creating an exhaustive dataset is imperative. The objective is to augment the database, and with each subsequent iteration involving a new dataset and the application of transfer learning, there is an anticipation of a progressive enhancement in the model.

RepViT-SAM

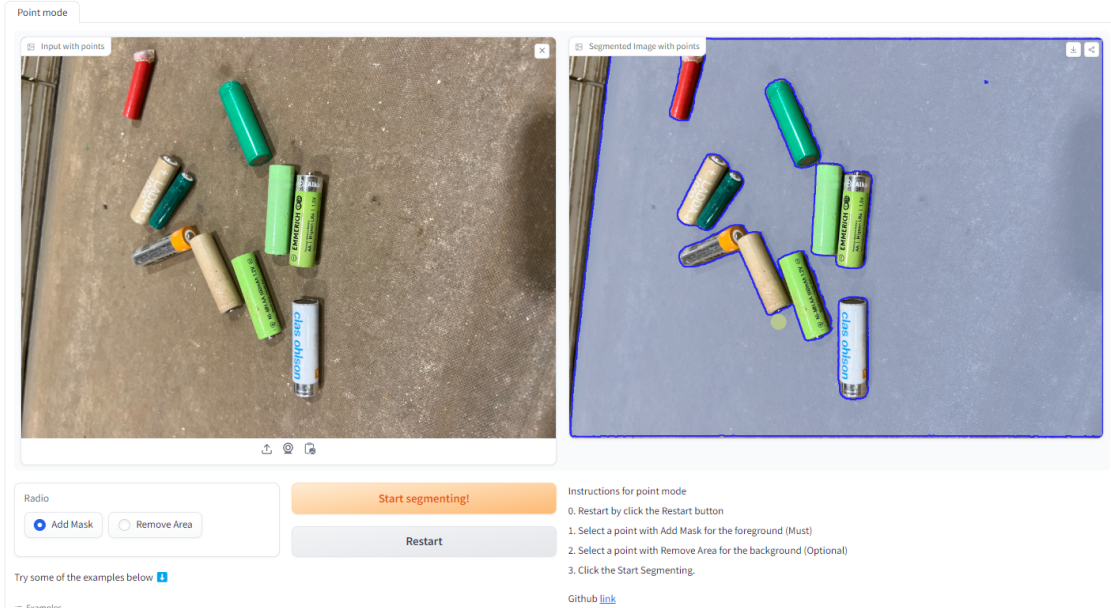


Figure 6.2: RepViT-SAM

- The application of this product extends beyond the recycling industry; it can also find utility in settings where battery examinations are essential, such as quality checks in a battery manufacturing unit.
- This system's capabilities can be further extended by incorporating robotic arms, thereby achieving a fully autonomous ecosystem. Integration with the Robot Operating System (ROS) would enhance the overall functionality and efficiency of the system.
- The future outlook for Grounded SAM appears promising, with several noteworthy features on the horizon, as indicated in the [38] Github updates from the Idea Research center. Notable features that could prove beneficial for the recycling battery industry include:
 1. Grounded-Mobile SAM: Enabling employees to scan conveyor belts directly from their mobile devices, providing a lightweight and user-friendly solution.
 2. RepViT-SAM: This feature allows for the instant segmentation of images, as demonstrated in the figure. A demo of this capability can be accessed via the Hugging Face like [33]

6.5 Integrations with Health Dashboard

- Within the scope of this thesis, the focus for the dashboard has been on utilizing Streamlit and Langchain as the front-end application, with the YOLOv8 trained dataset model serving as the input. Currently, a dashboard has been developed using the results.csv file generated from the model. As a potential improvement, there is a plan to incorporate a URL input feature on the webpage. If a video link is provided, the dashboard will dynamically update the count of batteries. For instance, a sample video has been produced using Grounded SAM, along with counters utilizing the OpenCV library, and is accessible at the following link: [Sample Video](#). The plan is to replace the existing algorithm with YOLOv8 in the backend and integrate it with video streaming capabilities in Streamlit. The advantage of employing Streamlit and the YOLO model is the cost-effectiveness, as all components are free of charge, except for the OpenAI API. However, it's crucial to note that continuous training of models, maintenance

of libraries and software, and adaptation to evolving technology may require constant updates and version changes in the program.

- If customers prefer to utilize the commercial product, the integration of Azure Custom AI as the backend, Power BI as the frontend, and Microsoft Chatbot can be facilitated, aligning with the project’s roadmap. The advantage of utilizing an Azure Custom Vision dashboard lies in its plug-and-play functionality, easy installation, and streamlined feedback provision to the model. However, it’s important to note that every component of Azure comes with associated costs.

6.6 Discussion on Contributions

- To summarize the contributions and conclusions discussed in Section 1.4:
 1. Custom Dataset Contribution: A significant contribution of this research is the creation and utilization of a custom dataset focused on batteries. This dataset serves as a valuable resource for future research in the field.
 2. Proof of Concept (PoC) Development: The development of a Proof of Concept (PoC) is a notable achievement. The PoC leverages technologies such as Streamlit and Langchain, as elaborated in Section 5.5. This demonstrates the practical implementation of the research findings.
 3. Comparative Analysis and Recommendations: A comprehensive comparative analysis has been conducted, highlighting that YOLOv8 with manual annotations excels in both image detection and segmentation. This suggests that for commercial applications, an alternative approach involving Azure Custom AI, PowerBI, and a Chatbot could be considered for enhanced functionality and user interaction.

These contributions collectively form a foundation for advancements in battery-related research, showcase practical application through the PoC, and offer insights for selecting appropriate technologies for commercialization.

6.7 Discussion on Hypothesis

In Section 1.5, it has been discussed that while the results of GroundedSAM and GroundingDINO are promising for generic datasets encompassing common categories such as humans, cars, and animals, there is a debate regarding their suitability for a custom battery dataset at the current stage. Consequently, the alternate hypothesis suggests that YOLOv8 and Azure Custom AI, particularly when trained with manually annotated images, offer better performance for the specific requirements of the custom battery dataset. This comparison emphasizes the importance of choosing models and technologies that align with the characteristics and nuances of the targeted dataset.

Chapter 7

Conclusions

Image detection and segmentation have indeed seen significant advancements in recent years, driven by the continuous research and development in the field of computer vision and deep learning. The abundance of image data and the growth of custom datasets tailored for specific industries contribute to the progress of these technologies. The need for fine-tuned datasets specific to certain domains, such as the recycling industry in this case, is crucial for developing accurate and reliable models that can address industry-specific challenges.

As technology evolves, the demand for sophisticated image processing techniques, including detection and segmentation, is likely to increase across various sectors. Custom datasets allow machine learning models to be trained on domain-specific patterns and nuances, improving their performance in real-world scenarios. It's an exciting time for the intersection of computer vision and industry applications, and this work in the recycling industry will be a tip of an iceberg to this ongoing progress.

GroundingDINO and Grounded SAM represent recent innovations in advanced deep learning, demonstrating effective auto-annotations across various categories. Their remarkable capability to detect even the smallest objects is noteworthy. However, our experiments revealed several false positives in GroundingDINO, where it tends to generalize batteries as rectangular or cylindrical objects. This leads to instances where even rectangular wood logs are misidentified as batteries, causing Grounded SAM to produce false positives since it relies on Grounding DINO's input. Therefore, based on our hypothesis, we conclude that both GroundingDINO and Grounded SAM still require industry-specific training datasets. Until this refinement occurs, the alternative hypothesis remains valid, Annotating data manually on Battery dataset on both Azure Custom AI and YOLOV8 showed promising results with better performance and better MAp score.

The comparison among YOLOv5, YOLOv8, and Azure Custom AI reveals that Azure Custom AI and YOLOv8 exhibit superior performance. Moreover, providing more data with complex backgrounds is expected to enhance performance efficiency even further. Consequently, StellAI aims to present two models to its customers. In the first setup, Azure Custom AI will operate in the backend, and the dashboard will be developed using POWER BI. As a second model, YOLOv8 will function as a training model in the backend, and Streamlit will serve as the front-end web UI.

Appendix A

Abbreviation

– AI	Artificial Intelligence
– BBOX	Boundingbox
– BiFPN	Bidirectional Feature Pyramid Network
– CLIP	The Contrastive Language Image Pre-training
– CNN	Convolution Neural Network
– DeTR	Detection Transformer
– DL	Deep Learning
– FPN	Feature Pyramid Network
– GAN	Generative Adversarial Network
– GLIP	Grounded Language Image Pre-Training
– GPU	Graphical Processing unit
– ICT	Information of Communications Technology
– LLM	Large Language model
– lr	Learning Rate
– PoC	Proof of concept
– ReLu	Rectified Linear Unit
– RNN	Recurrent Neural Network
– SA-1B	Segment Anything- 1 Billion
– SiLu	Sigmoid Liner Unit
– SWIN	Shifted Window Transformer
– UiA	University i Agder
– ViT	Visual Transformer
– YOLO	You Only Look Once

Bibliography

- [1] [1] Risefr.no. [Online]. Available: <https://risefr.no/media/publikasjoner/upload/2019/rise-rapport-2019-61-branner-i-avfallsanlegg.pdf>. [Accessed: 04-Jan-2024] .. [Accessed 05-01-2024].
- [2] Precision-Recall, scikit-learn. [Online]. Available: https://scikit-learn/stable/auto_examples/model_selection/plot_precision_recall.html. [Accessed: 05-Jan-2024] .. [Accessed 05-01-2024].
- [3] H. Borse, CNN Quicklearn - Analytics Vidhya - Medium, Analytics Vidhya, 09-May-2020. [Online]. Available: <https://medium.com/analytics-vidhya/cnn-quick-learn-12dced578b01>. [Accessed: 05-Jan-2024] .. [Accessed 05-01-2024].
- [4] Ultralytics, Architecture Summary, Ultralytics.com. [Online]. Available: https://docs.ultralytics.com/yolov5/tutorials/architecture_description. [Accessed: 05-Jan-2024] .. [Accessed 05-01-2024].
- [5] Ultralytics, Loss, Ultralytics.com. [Online]. Available: <https://docs.ultralytics.com/reference/utils/loss>. [Accessed: 05-Jan-2024] .. [Accessed 05-01-2024].
- [6] Ultralytics, Loss, Ultralytics.com. [Online]. Available: <https://docs.ultralytics.com/reference/utils/loss>. [Accessed: 05-Jan-2024] .. [Accessed 05-01-2024].
- [7] Introduction (no date) Roboflow.com. Available at: <https://docs.roboflow.com/> (Accessed: December 25, 2023) .. [Accessed 05-01-2024].
- [8] S. Kirch, GLIP: Introducing language-image pre-training to object detection, Towards Data Science, 01-Sep-2023. [Online]. Available: <https://towardsdatascience.com/glip-introducing-language-image-pre-training-to-object-detection-5ddb601873aa>. [Accessed: 05-Jan-2024] .. [Accessed 05-01-2024].
- [9] GLIP: Grounded Language-image Pre-training. <https://github.com/microsoft/GLIP> .. [Accessed 05-01-2024].
- [10] YOLOv8 vs. YOLOv5: Choosing the Best Object Detection Model, Augmented Startups.com. [Online]. Available: <https://www.augmentedstartups.com/blog/yolov8-vs-yolov5-choosing-the-best-object-detection-model>. [Accessed: 05-Jan-2024] .. [Accessed 05-01-2024].
- [11] LICENSE at master · ultralytics/yolov5. Available at <https://github.com/ultralytics/yolov5/blob/master/LICENSE>. [Accessed 05-01-2024].
- [12] Ultralytics, Tips for Best Training Results, Ultralytics.com. [Online]. Available: https://docs.ultralytics.com/yolov5/tutorials/tips_for_best_training_results. [Accessed: 05-Jan-2024] .. [Accessed 05-01-2024].
- [13] R. Chow, ImageNet: A pioneering vision for computers, History of Data Science, 27-Aug-2021. [Online]. Available: <https://www.historyofdatascience.com/imagenet-a-pioneering-vision-for-computers/>. [Accessed: 05-Jan-2024] .. [Accessed 05-01-2024].

- [14] *Albumentations Documentation - Bounding boxes augmentation for object detection — albumentations.ai*. https://albumentations.ai/docs/getting_started/bounding_boxes_augmentation/. [Accessed 05-01-2024].
- [15] Md Zahangir Alom et al. “The history began from alexnet: A comprehensive survey on deep learning approaches.” In: *arXiv preprint arXiv:1803.01164* (2018).
- [16] Jason Brownlee. “What is the Difference Between a Batch and an Epoch in a Neural Network.” In: *Machine Learning Mastery* 20 (2018).
- [17] Weipeng Cao et al. “A review on neural networks with random weights.” In: *Neurocomputing* 275 (2018), pp. 278–287.
- [18] Ahmed Fawzy Gad. “Evaluating object detection models using mean average precision (mAP).” In: *PaperspaceBlog* (2020).
- [19] Muhammad Hussain. “YOLO-v1 to YOLO-v8, the Rise of YOLO and Its Complementary Nature toward Digital Manufacturing and Industrial Defect Detection.” In: *Machines* 11.7 (2023), p. 677.
- [20] Mohammad Khorasani, Mohamed Abdou, and Javier Hernández Fernández. “Streamlit Use Cases.” In: *Web Application Development with Streamlit: Develop and Deploy Secure and Scalable Web Applications to the Cloud Using a Pure Python Framework*. Springer, 2022, pp. 309–361.
- [21] Alexander Kirillov et al. “Segment anything.” In: *arXiv preprint arXiv:2304.02643* (2023).
- [22] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning.” In: *nature* 521.7553 (2015), pp. 436–444.
- [23] Jing Li et al. “Brief introduction of back propagation (BP) neural network algorithm and its improvement.” In: *Advances in Computer Science and Information Engineering: Volume 2*. Springer. 2012, pp. 553–558.
- [24] Liunian Harold Li et al. “Grounded language-image pre-training.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 10965–10975.
- [25] Shilong Liu et al. “Grounding dino: Marrying dino with grounded pre-training for open-set object detection.” In: *arXiv preprint arXiv:2303.05499* (2023).
- [26] Ze Liu et al. “Swin transformer: Hierarchical vision transformer using shifted windows.” In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 10012–10022.
- [27] HamidReza Naseri and Vahid Mehrdad. “Novel CNN with investigation on accuracy by modifying stride, padding, kernel size and filter numbers.” In: *Multimedia Tools and Applications* (2023), pp. 1–19.
- [28] Alec Radford et al. “Learning transferable visual models from natural language supervision.” In: *International conference on machine learning*. PMLR. 2021, pp. 8748–8763.
- [29] Mathew Salvaris, Danielle Dean, and Wee Hyong Tok. “Deep learning with azure.” In: *Building and Deploying Artificial Intelligence Solutions on Microsoft AI Platform*, Apress (2018).
- [30] Indri Purwita Sary, Safrian Andromeda, and Edmund Ucok Armin. “Performance Comparison of YOLOv5 and YOLOv8 Architectures in Human Detection using Aerial Images.” In: *Ultima Computing: Jurnal Sistem Komputer* 15.1 (2023), pp. 8–13.
- [31] Burcu Selcuk and Tacha Serif. “A Comparison of YOLOv5 and YOLOv8 in the Context of Mobile UI Detection.” In: *International Conference on Mobile Web and Intelligent Information Systems*. Springer. 2023, pp. 161–174.
- [32] Sagar Sharma, Simone Sharma, and Anidhya Athaiya. “Activation functions in neural networks.” In: *Towards Data Sci* 6.12 (2017), pp. 310–316.

- [33] *Spaces* — *huggingface.co*. <https://huggingface.co/docs/hub/spaces>. [Accessed 05-01-2024].
- [34] Mingxing Tan, Ruoming Pang, and Quoc V Le. “Efficientdet: Scalable and efficient object detection.” In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 10781–10790.
- [35] Daksha Uchagaonkar. *YOLO v1 working explained..* — *daksha.uchagaonkar*. <https://medium.com/@daksha.uchagaonkar/yolo-v1-working-explained-ec20750be682>. [Accessed 05-01-2024].
- [36] Xueqiu Wang et al. “BL-YOLOv8: An Improved Road Defect Detection Model Based on YOLOv8.” In: *Sensors* 23.20 (2023), p. 8361.
- [37] Robert Zalosh, Pravinray Gandhi, and Adam Barowy. “Lithium-ion energy storage battery explosion incidents.” In: *Journal of Loss Prevention in the Process Industries* 72 (2021), p. 104560.
- [38] Chaoning Zhang et al. “A survey on segment anything model (sam): Vision foundation model meets prompt engineering.” In: *arXiv preprint arXiv:2306.06211* (2023).