University of Ottawa

Single Object Tracking

Computer Vision Final Project

Group 13:

Enas Ali 300389379

Muhammad Ali 300389890

Naser Mousa 300389909

Rim Touny 300389922

Shady Elsabow 300389887

Enhancing Single Object Tracking with YOLOv8: A Comprehensive Study

I. Abstract

You Only look once- YOLOv8, an advanced framework for object detection and tracking, serves as the linchpin of this research exploring Single Object Tracking (SOT) within a tailored dataset encompassing three distinct categories, each comprising 30 videos. Within video sequences spanning 24 frames, each frame showcases 16 items, with particular emphasis placed on standardizing tracking methodologies for Item 14, identified as 'Shoe,' within the dataset.

This study rigorously evaluates various iterations of YOLOv8 (n, s, m, L) and custom adaptations of YOLOv8s, in tandem with diverse learning rate (0.05, 0.01, 0.001). The assessment also integrates distinct tracking methodologies such as the Bot-SORT tracker and ByteTracker tracker to showcase YOLOv8's prowess in Single Object Tracking (SOT). The comprehensive analysis aims to elucidate YOLOv8's performance variations concerning model iterations, learning rate, and augmentation techniques, all aimed at refining item tracking precision within the dataset.

The utilization of YOLOv8 as the primary framework underscores its pivotal role in shaping advancements in Single Object Tracking methodologies. This investigation not only evaluates YOLOv8's adaptability to diverse model iterations and learning strategies but also accentuates its potential for enhancing object tracking accuracy within specific dataset domains.

II. INTRODUCTION

In our pursuit of refining methodologies for Single Object Tracking (SOT) within our project, the integration of state-of-the-art object detection models has emerged as a crucial approach to enhance tracking precision and efficiency. YOLOv8, the latest iteration in the esteemed YOLO series, stands as a fundamental component within our project framework, elevating the capabilities of object localization and tracking. [1]

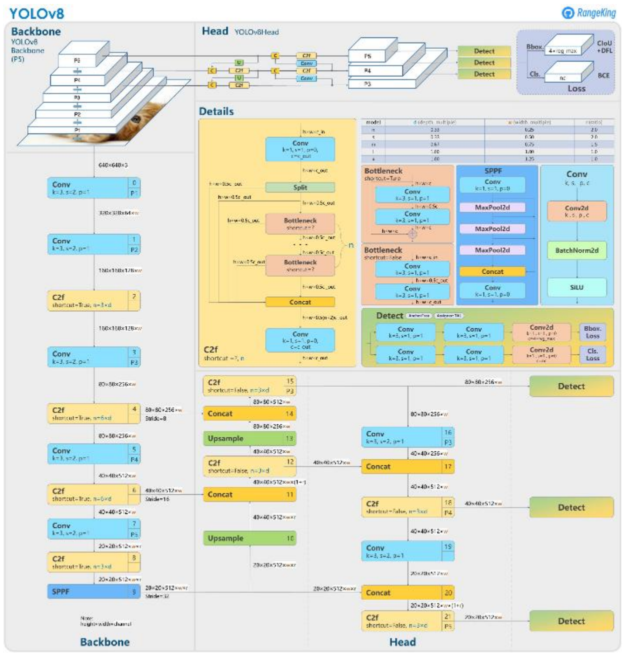


Figure 1:YOLOv8 Architecture, visualization made by GitHub user RangeKing.

Our project aims to harness the strengths of YOLOv8, which has been meticulously crafted and optimized for object detection tasks, to apply these advancements in the context of Single Object Tracking. With its advanced architecture and developer-friendly attributes, YOLOv8 holds the potential to significantly enhance our project’s SOT capabilities.

This paper serves as an extensive exploration and evaluation of the integration of YOLOv8 within our project’s Single Object Tracking framework. We delve deep into the intricacies of incorporating YOLOv8's object detection expertise to improve object tracking across video frames, highlighting the tailored features and modifications implemented to align with our specific tracking requirements.

Furthermore, our study delves into the practical aspects of utilizing YOLOv8 within our project, assessing its adaptability, accuracy, and computational efficiency in various tracking scenarios encountered within our application domain. Through empirical evaluations and comparisons with existing methodologies employed in our project, our goal is to shed light on the impact of YOLOv8 integration on the overall SOT performance metrics.

This paper encapsulates our project’s journey in utilizing YOLOv8 for Single Object Tracking, providing insights into the implementation, adaptability, and effectiveness of this cutting-edge object detection model within our specific application context. Our findings aim to contribute not only to the enhancement of our project’s SOT capabilities, but also to a broader understanding of utilizing YOLOv8 in real-world tracking scenarios.

III. LITERATURE SURVEY

J. Fan et al [2] focuses on introducing a novel algorithm for target tracking that merges the YOLO (You Only Look Once) object detection technique with the CSR-DCF (Discriminative Correlation Filter) tracking method. An essential augmentation to the algorithm involves integrating a self-attention mechanism to address the potential target loss during the tracking process. By harnessing the efficient object detection capabilities of YOLO and the precise tracking functionality of CSR-DCF, the algorithm achieves heightened tracking accuracy. Furthermore, it implements a closed-loop control strategy, integrating YOLO into the tracking loop to facilitate the re-tracking of lost targets, thus fortifying its resilience. The paper asserts the algorithm’s superiority over traditional CSR-DCF and DCF algorithms, highlighting its superior real-time performance, re-tracking capabilities, and overall tracking accuracy. By synergizing various neural network layers, feature representations, and the self-attention mechanism, the algorithm enhances feature representation, ensuring precise tracking through the CSR-DCF tracker. This fusion of YOLO’s robust detection capabilities with CSRDCF’s proficient tracking functionality improves robustness and accuracy, particularly in dynamic real-world scenarios.

J. Qu and S. Zhang [3] introduce an innovative multi-target tracking algorithm that merges YOLOv4, an advanced object detection architecture, with the Sequentially-Optimized Real-Time (SORT) tracker. The fusion aims to address real-time target detection challenges, emphasizing precise localization and efficient multi-target tracking at high frame rates. By integrating YOLOv4's upgraded architecture with SORT's Kalman filtering, the algorithm achieves superior accuracy and performance.
The algorithm leverages YOLOv4's architectural advancements, including CSPDarknet53, Spatial Pyramid Pooling (SPP), PANet, and the YoloHead module, to refine feature representation and enhance detection accuracy. SORT's essential elements, such as Kalman filtering and Hungarian matching, are highlighted for their role in displacement estimation and track identity updating between frames.

Experiments on the PETS09-S2L1 video dataset demonstrate the algorithm's efficacy in accurately tracking multiple targets, especially in scenarios with occlusions. Comparative analyses with YOLOv3 underscore the proposed algorithm's improvements, showcasing enhanced tracking precision and detection accuracy. The paper presents a significant contribution to video tracking by combining the strengths of YOLOv4's robust object detection and SORT's efficient multi-target tracking, offering a promising solution for improved accuracy and performance in dynamic real-world scenarios.

Heet Thakkar, Noopur Tambe, Sanjana Thamke, and Prof. Vaishali K. Gaidhane[4] explores the realm of visual object tracking, emphasizing the pivotal role of computer vision in comprehending object movements within video frames. Addressing the complexities inherent in object tracking, such as rapid motion, orientation changes, and occlusions, the research introduces a tracking-by-detection approach that utilizes YOLO for object detection and SORT for tracking. The authors trained their model on a custom dataset comprising six distinct classes. The paper extensively details the methodology adopted, encompassing the training process involving YOLOv3 and the utilization of SORT for tracking objects within video sequences. Additionally, the paper presents experimental results that include precision and recall metrics, accompanied by qualitative analyses via output screens depicting the performance of the tested videos.

IV. METHODOLOGY

Dataset

The dataset utilized in this study is specialized for object detection and tracking tasks, comprising three distinct classes of images: **'linear_movement_rotate'**, **'rotation_rotate'**, and **'fixed_random_rotate'**. Each class encapsulates 30 videos, showcasing diverse variations in object movement or rotation characteristics.

Comprising video sequences with 24 frames per video, each frame delineates 16 individual objects exhibiting a spectrum of movement types—rotational, linear, or fixed random rotations. Notably, each image within the dataset can feature multiple objects, and the same object may appear across different frames.

The dataset schema encompasses the last 2160 rows (frames) and 6 columns, providing detailed information:

- '**Image**' (RGB): Represents image pixels in a 3-D format.
- '**Target**' (Bounding boxes and IDs for each object within the frame): Includes annotations comprising bounding box coordinates (formatted as y_min, x_min, y_max, x_max) defining object spatial extents and unique identifiers (Object IDs) for each detected object.
- **'Image Paths'**: Specifies paths denoting the location of each image within the dataset.
- **'VideoID'** and **'ImageID'**: Unique identifiers for videos and individual images, facilitating dataset organization and referencing.
- '**Class**' (Class Name) Information: Indicates the class to which each frame or object belongs, detailing the specific movement or rotation characteristic.

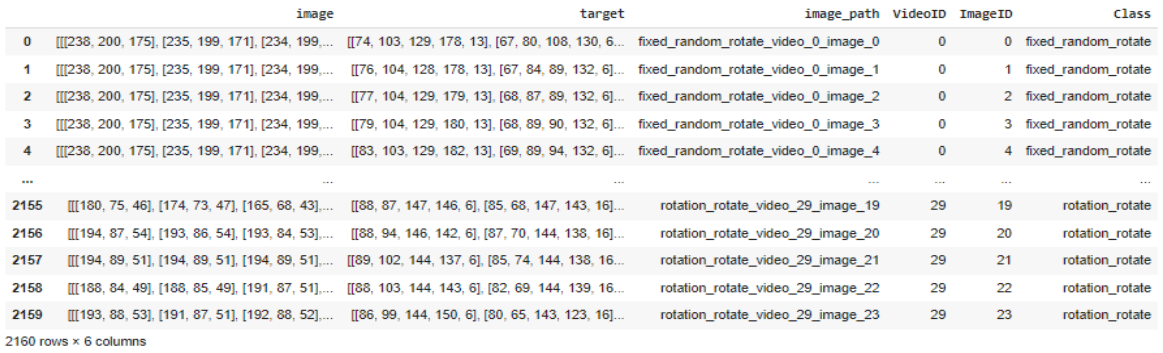


	image	target	image_path	VideoID	ImageID	Class
0	[[[238, 200, 175], [235, 199, 171], [234, 199,...	[[74, 103, 129, 178, 13], [67, 80, 108, 130, 6...	fixed_random_rotate_video_0_image_0	0	0	fixed_random_rotate
1	[[[238, 200, 175], [235, 199, 171], [234, 199,...	[[76, 104, 128, 178, 13], [67, 84, 89, 132, 6]...	fixed_random_rotate_video_0_image_1	0	1	fixed_random_rotate
2	[[[238, 200, 175], [235, 199, 171], [234, 199,...	[[77, 104, 129, 179, 13], [68, 87, 89, 132, 6]...	fixed_random_rotate_video_0_image_2	0	2	fixed_random_rotate
3	[[[238, 200, 175], [235, 199, 171], [234, 199,...	[[79, 104, 129, 180, 13], [68, 89, 90, 132, 6]...	fixed_random_rotate_video_0_image_3	0	3	fixed_random_rotate
4	[[[238, 200, 175], [235, 199, 171], [234, 199,...	[[83, 103, 129, 182, 13], [69, 89, 94, 132, 6]...	fixed_random_rotate_video_0_image_4	0	4	fixed_random_rotate
...	...	...	...	...	...	...
2155	[[[180, 75, 46], [174, 73, 47], [165, 68, 43],...	[[88, 87, 147, 146, 6], [85, 68, 147, 143, 16]...	rotation_rotate_video_29_image_19	29	19	rotation_rotate
2156	[[[194, 87, 54], [193, 86, 54], [193, 84, 53],...	[[88, 94, 146, 142, 6], [87, 70, 144, 138, 16]...	rotation_rotate_video_29_image_20	29	20	rotation_rotate
2157	[[[194, 89, 51], [194, 89, 51], [194, 89, 51],...	[[89, 102, 144, 137, 6], [85, 74, 144, 138, 16...	rotation_rotate_video_29_image_21	29	21	rotation_rotate
2158	[[[188, 84, 49], [188, 85, 49], [191, 87, 51],...	[[88, 103, 144, 143, 6], [82, 69, 144, 139, 16...	rotation_rotate_video_29_image_22	29	22	rotation_rotate
2159	[[[193, 88, 53], [191, 87, 51], [192, 88, 52],...	[[86, 99, 144, 150, 6], [80, 65, 143, 123, 16]...	rotation_rotate_video_29_image_23	29	23	rotation_rotate

2160 rows × 6 columns

Figure 2: Sample From the data

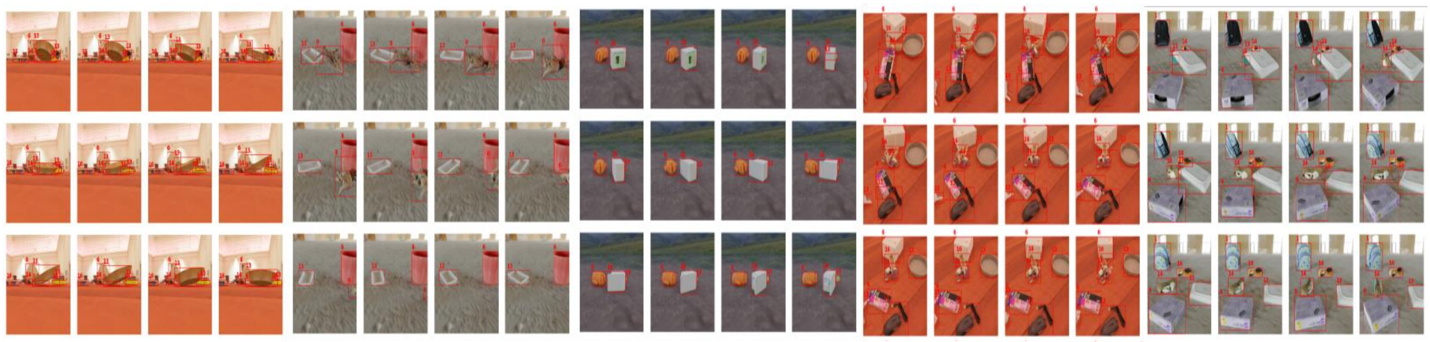
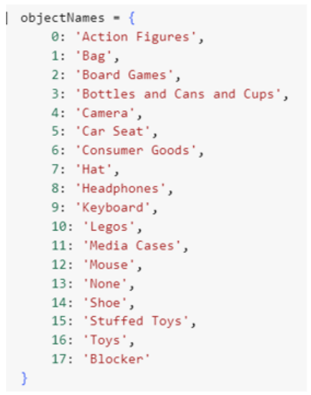


Figure 3: Videos Data Frame


```
objectNames = {
    0: 'Action Figures',
    1: 'Bag',
    2: 'Board Games',
    3: 'Bottles and Cans and Cups',
    4: 'Camera',
    5: 'Car Seat',
    6: 'Consumer Goods',
    7: 'Hat',
    8: 'Headphones',
    9: 'Keyboard',
    10: 'Legos',
    11: 'Media Cases',
    12: 'Mouse',
    13: 'None',
    14: 'Shoe',
    15: 'Stuffed Toys',
    16: 'Toys',
    17: 'Blocker'
}
```

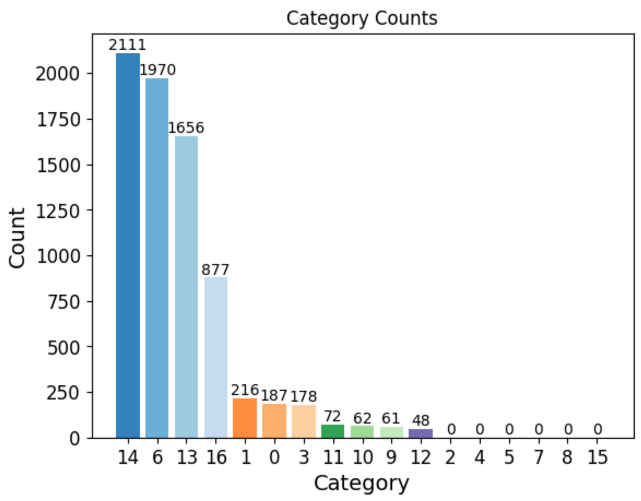


Figure 4: List detailing the names of objects along with their respective counts within the frames.

Data Preparation

I. Object Selection and Normalization:

The dataset preprocessing primarily focuses on isolating objects identified by the specific ID 14, specifically categorized as **'Shoe**' Subsequently, a newly introduced feature, named **'normalize_target**,' details the normalized bounding box coordinates specific to this designated object.

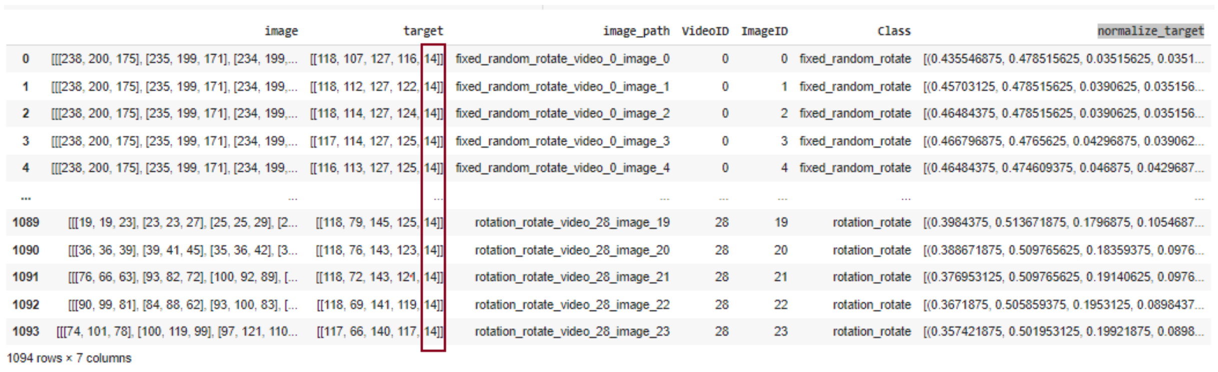


	image	target	image_path	VideoID	ImageID	Class	normalize_target
0	[[[238, 200, 175], [235, 199, 171], [234, 199,...	[[118, 107, 127, 116, 14]]	fixed_random_rotate_video_0_image_0	0	0	fixed_random_rotate	[(0.435546875, 0.478515625, 0.03515625, 0.0351...
1	[[[238, 200, 175], [235, 199, 171], [234, 199,...	[[118, 112, 127, 122, 14]]	fixed_random_rotate_video_0_image_1	0	1	fixed_random_rotate	[(0.45703125, 0.478515625, 0.0390625, 0.035156...
2	[[[238, 200, 175], [235, 199, 171], [234, 199,...	[[118, 114, 127, 124, 14]]	fixed_random_rotate_video_0_image_2	0	2	fixed_random_rotate	[(0.46484375, 0.478515625, 0.0390625, 0.035156...
3	[[[238, 200, 175], [235, 199, 171], [234, 199,...	[[117, 114, 127, 125, 14]]	fixed_random_rotate_video_0_image_3	0	3	fixed_random_rotate	[(0.466796875, 0.4765625, 0.04296875, 0.039062...
4	[[[238, 200, 175], [235, 199, 171], [234, 199,...	[[116, 113, 127, 125, 14]]	fixed_random_rotate_video_0_image_4	0	4	fixed_random_rotate	[(0.46484375, 0.474609375, 0.046875, 0.0429687...
...	...	...	...	...	...	...	...
1089	[[[19, 19, 23], [23, 23, 27], [25, 25, 29], [2...	[[118, 79, 145, 125, 14]]	rotation_rotate_video_28_image_19	28	19	rotation_rotate	[(0.3984375, 0.513671875, 0.1796875, 0.1054687...
1090	[[[36, 36, 39], [39, 41, 45], [35, 36, 42], [3...	[[118, 76, 143, 123, 14]]	rotation_rotate_video_28_image_20	28	20	rotation_rotate	[(0.388671875, 0.509765625, 0.18359375, 0.0976...
1091	[[[76, 66, 63], [93, 82, 72], [100, 92, 89], [...	[[118, 72, 143, 121, 14]]	rotation_rotate_video_28_image_21	28	21	rotation_rotate	[(0.376953125, 0.509765625, 0.19140625, 0.0976...
1092	[[[90, 99, 81], [84, 88, 62], [93, 100, 83], [...	[[118, 69, 141, 119, 14]]	rotation_rotate_video_28_image_22	28	22	rotation_rotate	[(0.3671875, 0.505859375, 0.1953125, 0.0898437...
1093	[[[74, 101, 78], [100, 119, 99], [97, 121, 110...	[[117, 66, 140, 117, 14]]	rotation_rotate_video_28_image_23	28	23	rotation_rotate	[(0.357421875, 0.501953125, 0.19921875, 0.0898...

1094 rows × 7 columns

Figure 5: Object Selection and Normalization

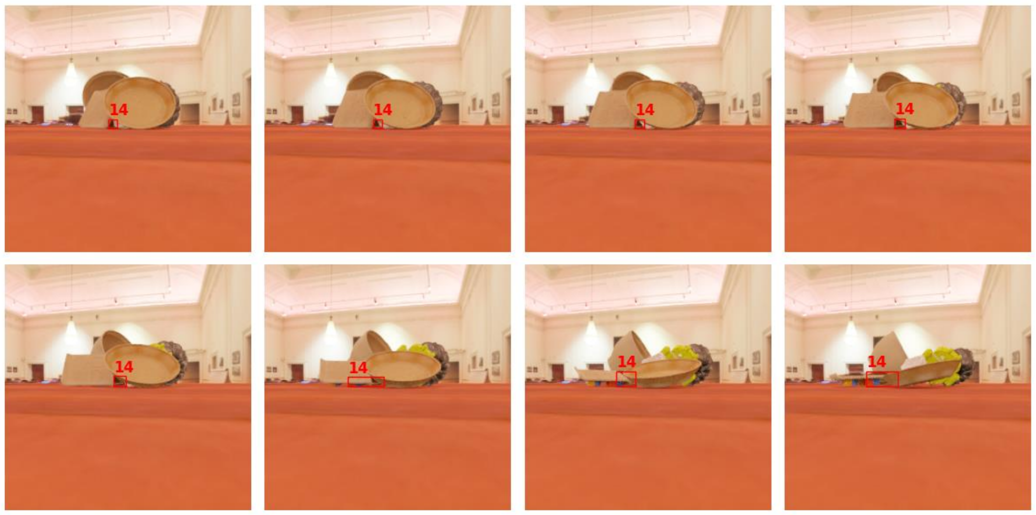
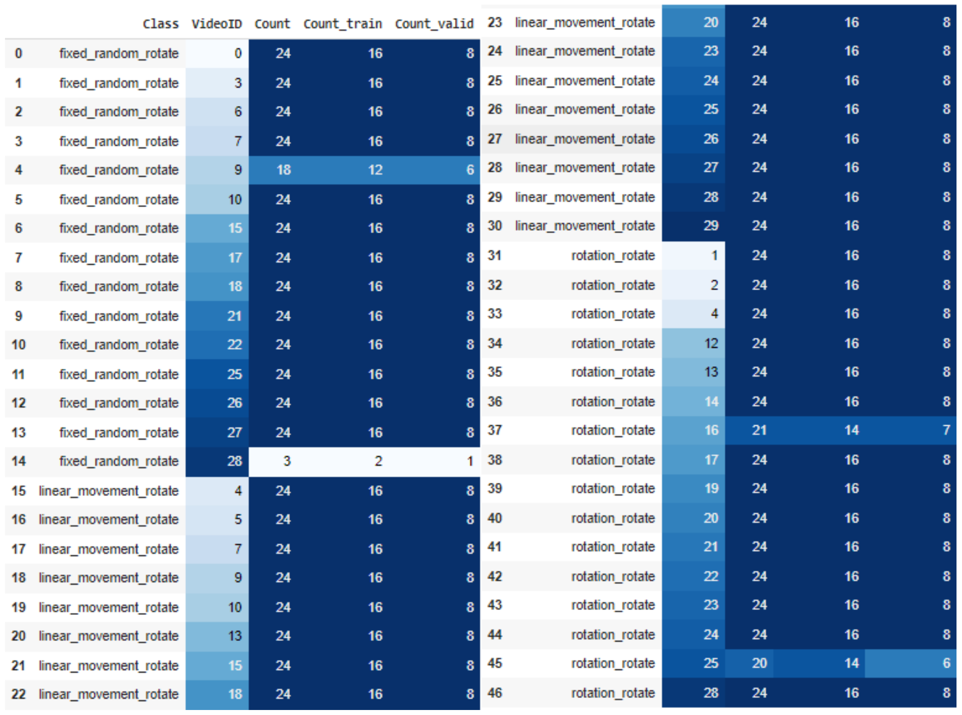


Figure 6: Sample from Data with Object ID 14: Shoe

II. Data Splitting for Balance:

A stratified method is employed to ensure fairness and consistency in the dataset, adeptly handling frame count variations associated with the object ID 14 criterion. From each video, 70% of the frame containing Object 14 are designated for the training set, while the remaining 30% are allocated for validation. This class-driven frame division ensures proportional representation across classes, effectively maintaining an unbiased distribution of frames among diverse classes and videos. This systematic strategy upholds the dataset's integrity and neutrality, ensuring equitable representation and fairness in model training and evaluation.



	Class	VideoID	Count	Count_train	Count_valid
0	fixed_random_rotate	0	24	16	8
1	fixed_random_rotate	3	24	16	8
2	fixed_random_rotate	6	24	16	8
3	fixed_random_rotate	7	24	16	8
4	fixed_random_rotate	9	18	12	6
5	fixed_random_rotate	10	24	16	8
6	fixed_random_rotate	15	24	16	8
7	fixed_random_rotate	17	24	16	8
8	fixed_random_rotate	18	24	16	8
9	fixed_random_rotate	21	24	16	8
10	fixed_random_rotate	22	24	16	8
11	fixed_random_rotate	25	24	16	8
12	fixed_random_rotate	26	24	16	8
13	fixed_random_rotate	27	24	16	8
14	fixed_random_rotate	28	3	2	1
15	linear_movement_rotate	4	24	16	8
16	linear_movement_rotate	5	24	16	8
17	linear_movement_rotate	7	24	16	8
18	linear_movement_rotate	9	24	16	8
19	linear_movement_rotate	10	24	16	8
20	linear_movement_rotate	13	24	16	8
21	linear_movement_rotate	15	24	16	8
22	linear_movement_rotate	18	24	16	8
23	linear_movement_rotate	20	24	16	8
24	linear_movement_rotate	23	24	16	8
25	linear_movement_rotate	24	24	16	8
26	linear_movement_rotate	25	24	16	8
27	linear_movement_rotate	26	24	16	8
28	linear_movement_rotate	27	24	16	8
29	linear_movement_rotate	28	24	16	8
30	linear_movement_rotate	29	24	16	8
31	rotation_rotate	1	24	16	8
32	rotation_rotate	2	24	16	8
33	rotation_rotate	4	24	16	8
34	rotation_rotate	12	24	16	8
35	rotation_rotate	13	24	16	8
36	rotation_rotate	14	24	16	8
37	rotation_rotate	16	21	14	7
38	rotation_rotate	17	24	16	8
39	rotation_rotate	19	24	16	8
40	rotation_rotate	20	24	16	8
41	rotation_rotate	21	24	16	8
42	rotation_rotate	22	24	16	8
43	rotation_rotate	23	24	16	8
44	rotation_rotate	24	24	16	8
45	rotation_rotate	25	20	14	6
46	rotation_rotate	28	24	16	8

Figure 7: Data Splitting

III. Data Organization:

The organized dataset architecture comprises two core directories: '**image**' and '**label**' Within '**image**,' separate subfolders labeled 'Train' and 'Valid' house correspondent training and validation images. Simultaneously, the 'label' directory encompasses 'Train' and 'Valid' subfolders, containing annotation files saved in '.txt' format.

Annotation Format: Formatted in '.txt' and situated within the '**label**' directory, adhere to a defined structure [0, x_center, y_center, width, height]. Each '.txt' file encompasses rows corresponding to the count of Object 14 instances detected within frame, ensuring comprehensive and structured object annotations crucial for proficient model training and evaluation.

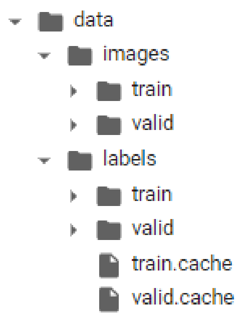
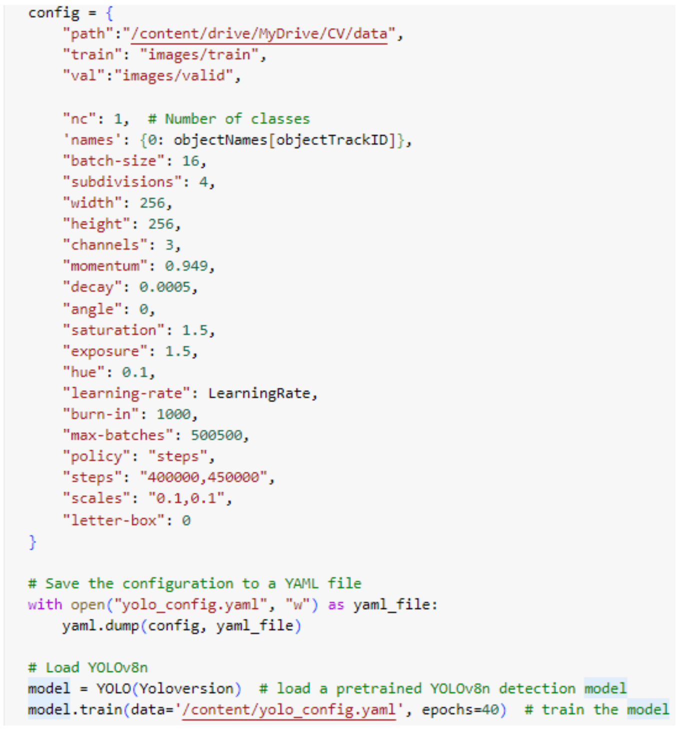


Figure 8: Data Preparation for Modeling

Modeling

The modeling phase revolves around utilizing YOLOv8 for object detection and tracking. This involves configuring the model parameters for training, training the YOLOv8n model, and subsequently applying it to detect and track objects, particularly focusing on Object ID 14, labeled as 'Shoe.'In the training phase, the YOLOv8n model is fine-tuned using the specified dataset configurations. Subsequently, in the detection phase, the trained model is applied to annotate video frames, detecting objects based on confidence levels and visualizing them with annotated bounding boxes.

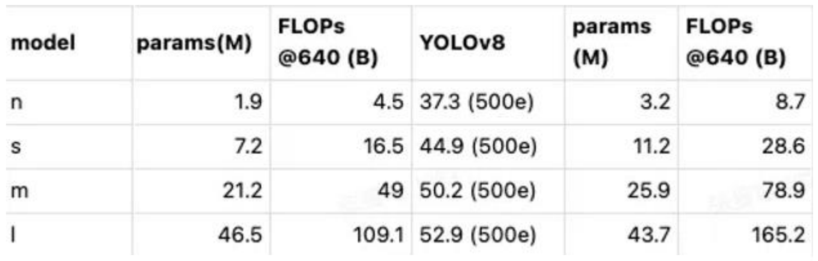


```
config = {
    "path":"/content/drive/MyDrive/CV/data",
    "train": "images/train",
    "val":"images/valid",

    "nc": 1,  # Number of classes
    'names': {0: objectNames[objectTrackID]},
    "batch-size": 16,
    "subdivisions": 4,
    "width": 256,
    "height": 256,
    "channels": 3,
    "momentum": 0.949,
    "decay": 0.0005,
    "angle": 0,
    "saturation": 1.5,
    "exposure": 1.5,
    "hue": 0.1,
    "learning-rate": LearningRate,
    "burn-in": 1000,
    "max-batches": 500500,
    "policy": "steps",
    "steps": "400000,450000",
    "scales": "0.1,0.1",
    "letter-box": 0
}

# Save the configuration to a YAML file
with open("yolo_config.yaml", "w") as yaml_file:
    yaml.dump(config, yaml_file)

# Load YOLOv8n
model = YOLO(Yoloversion)  # load a pretrained YOLOv8n detection model
model.train(data='/content/yolo_config.yaml', epochs=40)  # train the model
```

Figure9: Training Yolov8 Model

We assess various YOLOv8 versions—YOLOv8n, YOLOv8s, YOLOv8m, and a customized YOLOv8s model with additional layers—trained under different learning rates (0.01, 0.05, 0.1). The training entails 40 epochs with a batch size of 16, maintaining consistency with the specific training setup.

- The YOLOv8 series consists of different iterations—YOLOv8n, YOLOv8s, YOLOv8m, and YOLOv8l—each based on the YOLOv8 model but with distinct variations in learning rates and trackers. These versions exhibit discrepancies in layer count, parameters, gradients, and GFLOPs, showcasing diverse performance attributes. YOLOv8n is tailored for resource-limited devices, prioritizing faster inference over absolute accuracy. YOLOv8s achieves a balance between speed and accuracy, suitable for general-purpose applications. YOLOv8m enhances accuracy compared to YOLOv8s while maintaining relatively fast inference. YOLOv8l emphasizes either accuracy or speed, offering heightened precision at the expense of slower inference times.



model	params(M)	FLOPs @640 (B)	YOLOv8	params (M)	FLOPs @640 (B)
n	1.9	4.5	37.3 (500e)	3.2	8.7
s	7.2	16.5	44.9 (500e)	11.2	28.6
m	21.2	49	50.2 (500e)	25.9	78.9
l	46.5	109.1	52.9 (500e)	43.7	165.2

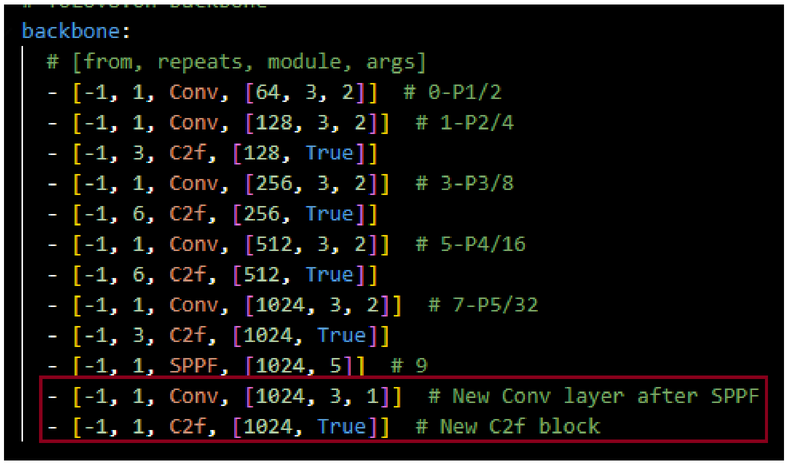
Figure 10: Performance Comparison for different models of Yolov8

Model	Summary
N	225 **LAYERS**, 3157200 **PARAMETERS**, 3157184 **GRADIENTS**, 8.9 **GFLOPS**
S	225 **LAYERS**, 11166560 **PARAMETERS**, 11166544 **GRADIENTS**, 28.8 **GFLOPS**
M	295 **LAYERS**, 25902640 **PARAMETERS**, 25902624 **GRADIENTS**, 79.3 **GFLOPS**
L	365 **LAYERS**, 43691520 **PARAMETERS**, 43691504 **GRADIENTS**, 165.7 **GFLOPS**

The YOLOv8 backbone describes the layer sequences used for feature extraction. This consists of convolutional layers, C2f (a custom layer), and SPPF (Spatial Pyramid Pooling). In the customized YOLOv8s model, two additional layers have been introduced after the SPPF layer, marked in Figure [11]. These extra layers can facilitate improved feature extraction, aiding the model in detecting more complex patterns.

The YOLOv8 had is responsible for the final output generation. It involves a series of operations such as up sampling, concatenation with backbone layers, convolutional layers, and detection. In the YOLOv8s model, there are additional C2f layers after concatenation with P3, P4, and P5 backbone, enhancing feature extraction at multiple scales.

Overall, These additional layers and modifications within the YOLOv8s model aim to enrich the feature representation, potentially improving the model's ability to detect and classify objects effectively.During the detection process, objects are identified with a confidence threshold set at >70%.



```
backbone:
  # [from, repeats, module, args]
  - [-1, 1, Conv, [64, 3, 2]]  # 0-P1/2
  - [-1, 1, Conv, [128, 3, 2]]  # 1-P2/4
  - [-1, 3, C2f, [128, True]]
  - [-1, 1, Conv, [256, 3, 2]]  # 3-P3/8
  - [-1, 6, C2f, [256, True]]
  - [-1, 1, Conv, [512, 3, 2]]  # 5-P4/16
  - [-1, 6, C2f, [512, True]]
  - [-1, 1, Conv, [1024, 3, 2]]  # 7-P5/32
  - [-1, 3, C2f, [1024, True]]
  - [-1, 1, SPPF, [1024, 5]]  # 9
  - [-1, 1, Conv, [1024, 3, 1]]  # New Conv layer after SPPF
  - [-1, 1, C2f, [1024, True]]  # New C2f block
```

Figure 10: Fixed model-adding extra layers

YOLO's fast detection allows real-time tracking, which can be improved by integrating tracking algorithms like **BotSort** and **ByteTracker**. We integrates BotSort and ByteTrack tracking mechanisms to evaluate their effectiveness alongside YOLOv8 models in object tracking across video frames.

In the absence of specific evaluation metrics for trackers BotSort and ByteTracker, an evaluation methodology reliant on Intersection over Union (IoU) was devised. The methodology commenced with standardizing frame sizes' to 256x256 pixels to ensure consistency across evaluations. Object tracking was performed using the track method for each tracker across diverse model configurations and learning rates. Subsequently, IoU calculations were employed to measure the alignment between predicted bounding boxes obtained from the trackers and the ground truth bounding boxes within the original data frame. By systematically handling various evaluation cases, a comprehensive approach ensured accurate computation of the Average IoU across all frames.

The handling of evaluation cases involved:

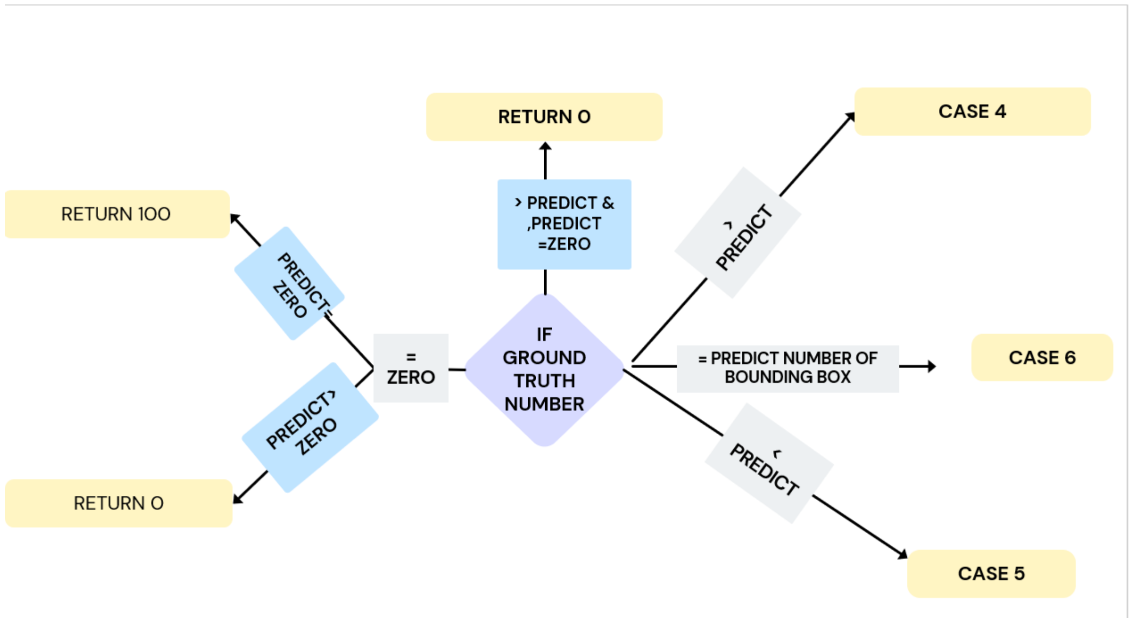


Figure 11::Tracking Evaluation Handeled Cases

- **CASE 1 BOTH GROUND TRUTH AND PREDICTED BOXES ARE EMPTY:** returns a score of 100, signifying a frame without detected objects and devoid of ground truth annotations.
- **CASE 2 NO GROUND TRUTH, BUT PREDICTED BOXES EXIST**: yields a score of 0, indicating object detection by the model without available ground truth annotations for comparison.
- **CASE 3 GROUND TRUTH EXISTS, BUT NO PREDICTED BOXES:** returns a score of 0, indicating the model's failure to detect objects despite ground truth annotations.
- **CASE 4 GROUND TRUTH(GT) ARE MORE THAN THE PREDICTED BOXES:** calculates the best iou for each gt box against all predicted boxes and gather them in list and get the summation then divide it by the number of gt boxes.

NOTE:: calculates the best iou for each gt box because the input might not be sorted, which means that the predicted objects are not sorted as same as the ground truth objects. also divide it by the number of gt boxes here is essential because the this will decrease the iou because of missing some objects.

- **CASE 5 GROUND TRUTH(GT) ARE LESS THAN THE PREDICTED BOXES:** calculates the best iou for each predicted box against all gt boxes and gather them in list and get the summation then divide it by the number of predicted boxes.
- **CASE 6 ALIGNED GROUND TRUTH AND PREDICTED BOXES WITH THE SAME COUNTS:** computes iou for each predicted box against all ground truth boxes then get the average.

This comprehensive analysis aims to understand the performance, accuracy, and robustness of the different YOLOv8 models, their customized variants, learning rates, and tracking mechanisms under varying scenarios, laying the groundwork for detailed discussions and insights into model performance in subsequent study sections.

V. EXPERIMENTATION AND RESULTS

- **Detection Evaluation:**

The evaluation of object detection performance is pivotal in assessing the effectiveness of models. Mean Average Precision (mAP) is a widely used metric for object detection evaluation due to its ability to gauge model precision and recall simultaneously.

Mean Average Precision (mAP): mAP measures the precision-recall balance across multiple detection confidence thresholds. It is computed by averaging the precision values at various recall levels. mAP provides a comprehensive understanding of how well a model identifies objects within an image dataset.

$$\text{mAP}=\frac{1}{n}\sum_{i=1}^{n} APi$$

where n is the number of object classes, and AP_i is the Average Precision for each class i.

Suitability of mAP for Object Detection:
mAP considers both precision and recall, making it suitable for evaluating object detection models. It quantifies the model's ability to precisely locate objects while ensuring a high recall rate, especially crucial in scenarios with multiple objects of various classes

The evaluation involved measuring the mAP50 (mAP at IoU threshold 0.5) and mAP50-95 (mAP from IoU

0.5 to 0.95) for YOLOv8 models (N, S, M, L, Fixed) across different learning rates. The mAP scores were computed to analyze the detection performance concerning the specific targets object ('Shoe' with Object ID 14) within the dataset.

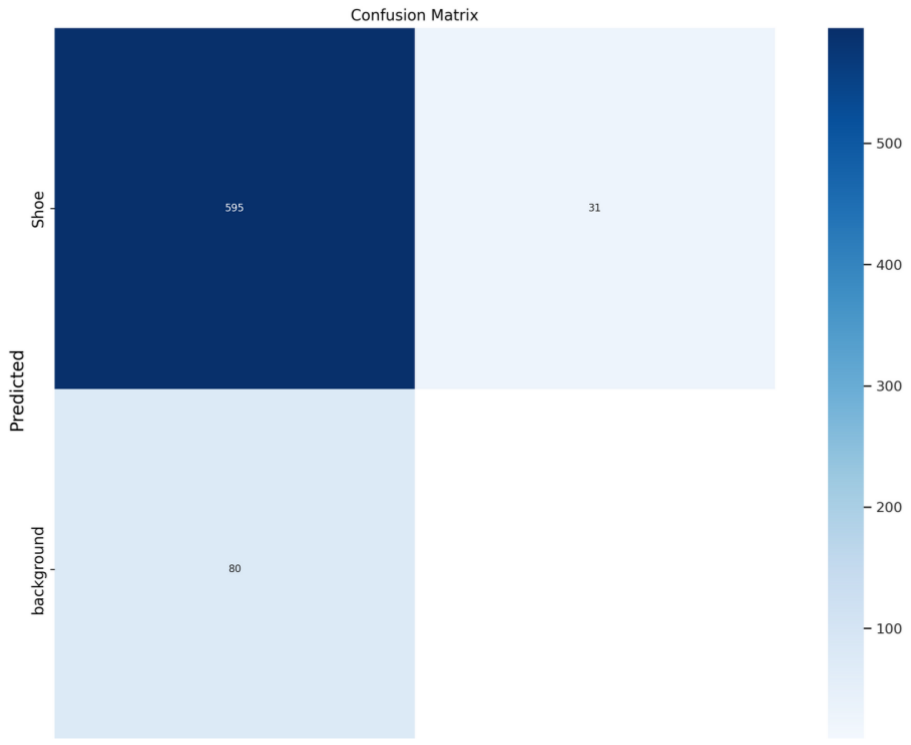


Figure 12: Confusion matrix of the champion model YOLOv8n 0.001 LR

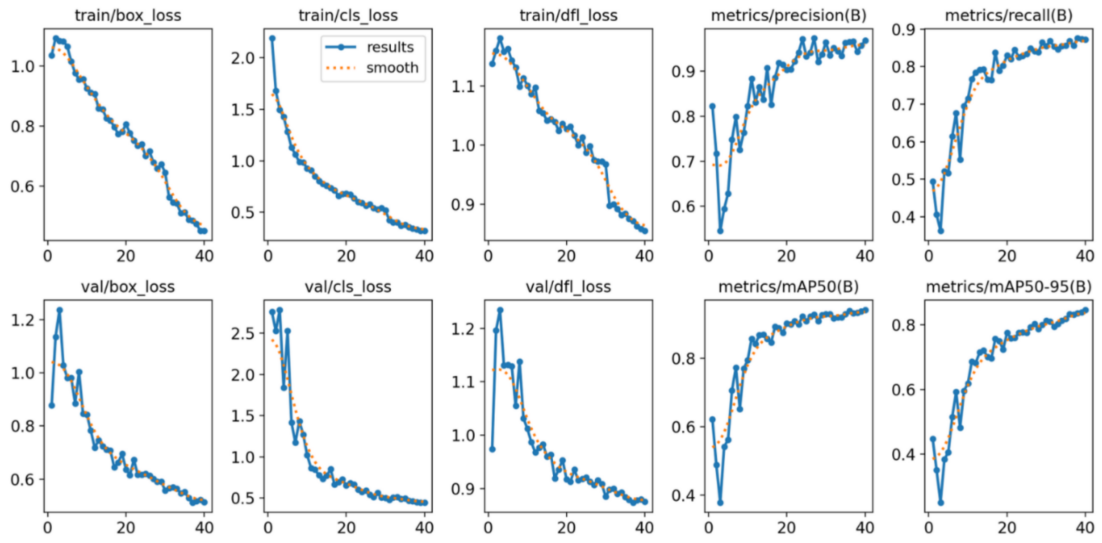


Figure 13:Loss function of the champion model YOLOv8N 0.001 LR

Model	Box Precision	Box Recall	mAP50	mAP50-95
N	0.05 LR = **96.8%** 0.01 LR = **96.8%** 0.001 LR = **96.8%**	0.5 LR = **87.3%** 0.1 LR = **87.3%** 0.001 LR = **87.3%**	0.05 LR = **94.2%** 0.01 LR = **94.2%** 0.001 LR = **94.2%**	0.05 LR = **84.7%** 0.01 LR = **84.7%** 0.001 LR = **84.7%**
S	0.05 LR = **97.1%** 0.01 LR = **97.1%** 0.001 LR = **97.1%**	0.05 LR = **85%** 0.01 LR = **85%** 0.001 LR = **85%**	0.05 LR = **94.1%** 0.01 LR = **94.1%** 0.001 LR = **94.1%**	0.05 LR = **84.5%** 0.01 LR = **84.5%** 0.001 LR = **84.5%**
M	0.05 LR = **94.9%** 0.01 LR = **94.9%** 0.001 LR = **94.9%**	0.05 LR = **86.5%** 0.01 LR = **86.5%** 0.001 LR = **86.5%**	0.05 LR = **93%** 0.01 LR = **93%** 0.001 LR = **93%**	0.05 LR = **83.3%** 0.01 LR = **83.3%** 0.001 LR = **83.3%**
L	0.05 LR = **92.4%** 0.01 LR = **92.4%** 0.001 LR = **92.4%**	0.05 LR = **84.3%** 0.01 LR = **84.3%** 0.001 LR = **84.3%**	0.05 LR = **91.4%** 0.01 LR =**91.4%** 0.001 LR =**91.4%**	0.05 LR =**81.6%** 0.01 LR =**81.6%** 0.001 LR =**81.6%**
Fixed	0.05 LR = **83.6%** 0.01 LR = **83.6%** 0.001 LR = **83.6%**	0.05 LR = **79.4%** 0.01 LR = **79.4%** 0.001 LR = **79.4%**	0.05 LR = **86%** 0.01 LR = **86%** 0.001 LR = **86%**	0.05 LR = **69.3%** 0.01 LR = **69.3%** 0.001 LR = **69.3%**

The consistent performance observed across different YOLOv8 configurations (N, S, M, L) despite variations in learning rates (LR) suggests stability due to various reasons:

- **Convergence to Optimal Solution**: Models likely reached an optimal solution, where LR adjustments showed minimal impact on learning improvements.
- **Stable Learning Dynamics**: YOLOv8 models displayed consistent learning patterns, maintaining performance stability despite changes in LR.
- **Robustness to LR Changes:** YOLOv8 architectures might inherently manage LR variations without significantly affecting their performance.
- **Optimal LR Range:** The chosen LR values possibly align well with these models, leading to stable and effective learning.
- **Data Complexity and Model Capacity:** The dataset complexity and model capacity may harmonize with the LR settings, contributing to consistent performance.

- **Tracking Evaluation**

  The evaluation of trackers BotSort and ByteTracker is rooted in the Intersection over Union (IoU) performance metric, which gauges the precision of object localization. The IoU is mathematically defined as:

$$IoU = \frac{\text{Area of Overlap (Intersection)}}{\textit{Area of Union}} = \frac{|A \sqcap B|}{|A| \sqcup |B|}$$

The area of overlap (intersection) between two bounding boxes, A and B, is calculated as:

$$\text{Area of Intersection} = \max(0, x\text{min} - x\text{max}) \times \max(0, y\text{min} - y\text{max})$$

Where:

$$x_{\min} = \max(x_A - \frac{w_A}{2}, x_B - \frac{w_B}{2})$$
$$y_{\min} = \max(y_A - \frac{h_A}{2}, y_B - \frac{h_B}{2})$$

$$x_{\max} = \max(x_A + \frac{w_A}{2}, x_B + \frac{w_B}{2})$$
$$y_{\min} = \max(y_A + \frac{h_A}{2}, y_B + \frac{h_B}{2})$$

The area of union between boxes A and B is determined by:

$$\text{Area of Union} = \text{Area}(A) + \text{Area}(B) - \text{Area of Intersection}$$

Utilizing IoU as the primary evaluation metric offers insight into the pre2cision of object detection and localization for the BotSort and ByteTracker, supporting robust assessment and comparison between predicted and actual object bounding boxes in tracking scenarios.

Model	**BoT-SORT Tracker**	**ByteTrack Tracker**
N	**0.05 LR = 77.5%** **0.01 LR = 77.5%** **0.001 LR = 77.5%**	**0.05 LR = 67.2%** **0.01 LR = 67.2%** **0.001 LR = 67.2%**
S	**0.05 LR = 66.36%** **0.01 LR = 66.36%** **0.001 LR = 66.36%**	**0.05 LR = 76.7%** **0.01 LR = 76.7%** **0.001 LR = 76.7%**
M	**0.05 LR = 81.82%** **0.01 LR = 81.82%** **0.001 LR =81.82%**	**0.05 LR = 67.88%** **0.01 LR = 67.88%** **0.001 LR =67.88%**
L	**0.05 LR = 74,29%** **0.01 LR = 74,29%** **0.001 LR = 74,29%**	**0.05 LR = 62.847%** **0.01 LR = 62.847%** **0.001 LR = 62.847%**
Fixed	**0.05 LR = 65.2%** **0.01 LR = 65.2%** **0.001 LR = 65.2%**	**0.05 LR = 56.6%** **0.01 LR = 56.6%** **0.001 LR = 56.6%**

1. YOLO isn't inherently a tracker; it serves as a detector with incorporated algorithms such as BoT-SORT Tracker and ByteTrack Tracker for tracking. This integration stems from YOLO's capability to swiftly and effectively detect objects while maintaining high performance levels.

2. The evaluation of trackers indicates a limited sensitivity to learning rate adjustments, suggesting that other parameters might overshadow the impact of changes in the learning rate.

VI. CONCLUSION & FUTURE WORK

- **Choosing Other Parameters than Learning Rate:** In machine learning models, the learning rate is a crucial hyperparameter that affects how quickly a model learns from data. However, in some cases, tweaking other parameters (such as batch size, optimizer choice, regularization techniques) apart from the learning rate might have a more significant impact on the model's performance. This approach involves experimenting with and adjusting various parameters other than the learning rate to improve the model's training process or performance.

- **Reducing Model Complexity:** Complex models might lead to overfitting or computational inefficiency. Lowering the complexity of a model involves simplifying its architecture by reducing the number of layers, neurons, or parameters. This process aims to create a more streamlined, easier-to-understand model that can generalize better to unseen data and potentially improve its performance.

- **Increasing Data Importance:** In machine learning, the quantity and quality of data play a crucial role in training robust models. Increasing the dataset size can help in understanding how a model reacts or generalizes to diverse scenarios. By providing more varied and representative data, one can assess how the model performs across different conditions, potentially improving its accuracy and generalization capabilities.

Each of these strategies aims to optimize different aspects of the machine learning process to enhance the model's performance, generalization, or training efficiency. Adjusting parameters beyond the learning rate, simplifying model complexity, and enriching the dataset are all methods to iteratively improve the model's capabilities and effectiveness.

VII. REFERENCES

[1] G. Jocher, "YOLOv8 Documentation," docs.ultralytics.com, May 18, 2020. https://docs.ultralytics.com/

[2] J. Fan et al., "A Research on CSR-DCF Tracking Algorithm based on YOLO Detection," 2023 4th International Conference on Computer Vision, Image and Deep Learning (CVIDL), Zhuhai, China, 2023, pp. 641-645, doi: 10.1109/CVIDL58838.2023.10166988.

[3] J. Qu and S. Zhang, "Research on Video Tracking Algorithm Based on Yolo Target Detection," *2023 6th International Conference on Computer Network, Electronic and Automation (ICCNEA)*, Xi'an, China, 2023, pp. 437-439, doi: 10.1109/ICCNEA60107.2023.00098.

[4] Heet Thakkar, Noopur Tambe, Sanjana Thamke, Prof. Vaishali K. Gaidhane, "Object Tracking by detection using YOLO and SORT", International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), Maharashtra, India, 2020, pp. 224-229, doi: 10.32628/CSEIT206256.

[5] Jocher, Glenn, et al. "YOLO by Ultralytics." GitHub, 1 Jan. 2023, github.com/ultralytics/ultralytics.

[6] J. Solawetz and Francesco, "What is YOLOv8? The Ultimate Guide.," Roboflow Blog, Jan. 11, 2023.https://blog.roboflow.com/whats-new-in-yolov8/