Drift-Aware Online Learning System for Robust Object Detection

**TABLE OF CONTENTS**

|  |  |  |  |
| --- | --- | --- | --- |
| Chapter |  | Title | Page No. |
|  |  | Abstract | 10 |
| 1 |  | Introduction | 11 |
| 2 |  | Literature Survey | 12 – 13 |
| 3 |  | Problem Definition | 14 |
| 4 |  | Solution Strategy | 15 – 17 |
| 5 |  | Design | 18 – 19 |
| 6 |  | Implementation details | 20 – 24 |
| 7 |  | Results and discussion | 25 – 35 |
| 8 |  | Conclusion | 36 |
| 9 |  | Limitations | 37 |
| 10 |  | Future scope of the project | 38 |
| 11 |  | Gantt Chart | 39 |
|  |  | References | 40 |
|  |  | Plagiarism Report | 43 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Figure Name** | **Page No.** |
| 1. | Closed loop drift-aware machine learning system | 18 |
| 2. | PIACERE approach for optimized and self-healed IaC. | 19 |
| 3. | Code Results | 25 |
| 4. | Confusion matrix normalized | 26 |
| 5. | F1 confidence score | 27 |
| 6. | Precision-Recall Curve | 28 |
| 7. | Precision-Confidence Curve | 28 |
| 8. | Recall-Confidence Curve | 29 |
| 9. | Results – Baseline YOLO (Graphed) | 30 |
| 10 | Results – CCTV Baseline YOLO | 31 |
| 11 | Results – CCTV Incremental model | 32 |
| 12 | Results – Incremental model (Graphed) | 33 |
| 13 | Final recording | 34 |
| 14 | Website we trained | 35 |
| 15 | Gantt Chart | 39 |
| 16 | Plagiarism check | 43 |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **Table No.** | **Title Name** | **Page no.** |
| 1 | Literature survey | 12 – 13 |
| 2 | Dataset | 17 |

**CHAPTER 0: ABSTRACT**

Modern computer vision systems, even in practical applications, have shown the vulnerability of performance degradation due to data distribution changes, known as data drift. These data patterns are not addressed in the traditional fixed-based training process, leading to decreased performance, lower accuracy, and unreliable performance of the system with the passage of time. The project, therefore, aims to develop a system for drifting in the object detection system by monitoring the data, identifying the data patterns, and adapting the system using the online learning process, with the goal of preventing the catastrophic effect of forgetting in the system.

**CHAPTER 1: INTRODUCTION**

The state-of-the-art models based on deep learning for the object detection task, like YOLO, are accurate when trained on perfectly curated and static datasets. But when the models are deployed in real-world settings, the models are faced with real-world data, which may vary from the original distribution because of reasons such as changes in the environment, noise in the sensor, changes in light conditions, motion blur effects, and changes in the appearance of the objects.

Despite these challenges, most academic endeavors are focused on a one-time training process and offline evaluation rather than post-deployment dynamics. This is not the case with real-world machine learning applications, as these models also need the capacity to realize when their predictions start deviating and adjust as a result without impairing stability. This project therefore fills the existing gap by developing an end-to-end drift-aware object detection solution capable of monitoring input streams, recognizing drift occurrences, adjusting the model, and making model deployment decisions.

**CHAPTER 2: LITERATURE SURVEY**

**Table 1**: Literature Survey

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sl. No.** | **Author(s)** | **Name of Paper** | **Publication Details** | **Details of the Work** | **Relevance to the Present Work** |
| 1 | Zhang, Y. et al. | *A Survey on Data Drift Detection in Deep Learning Systems* | arXiv:2508.17975v1, 2025 | This paper provides a comprehensive survey of different types of data drift including covariate, concept, and label drift. It reviews statistical methods, embedding-based approaches, and monitoring techniques for detecting drift in deployed deep learning models. | This work provides theoretical grounding for the drift detection techniques used in the project, particularly statistical drift measures and feature embedding drift. |
| 2 | Iguazio Team | *Concept Drift Deep Dive: How to Build a Drift-Aware ML System* | Iguazio Technical Blog, 2024 | The article discusses real-world challenges of concept drift in production ML systems and proposes architectural patterns for drift monitoring, automated retraining, and governance. | This resource strongly influenced the system-level design of the project, including automated drift detection, retraining triggers, and deployment governance. |
| 3 | Li, X., Wang, Y. | *Online Learning Strategies for Deep Neural Networks Under Distribution Shift* | arXiv:2411.00186, 2024 | The authors explore online and incremental learning strategies to adapt deep neural networks under changing data distributions, emphasizing replay buffers and stability–plasticity trade-offs. | This paper directly supports the project’s incremental fine-tuning strategy using replay buffers to prevent catastrophic forgetting. |
| 4 | Solugenix Research | *Harnessing AI/ML for Self-Healing Systems* | Solugenix Blog, 2023 | This article introduces the concept of self-healing systems where AI models can detect failures, adapt autonomously, and restore optimal performance without human intervention. | The self-healing philosophy presented here aligns with the project’s objective of automated recovery from drift through adaptive retraining. |
| 5 | SAR Council | *The Resilient Enterprise: Building Self-Healing MLOps Pipelines* | SAR Council Industry Report, 2025 | The report discusses resilient MLOps pipelines with automated monitoring, retraining, evaluation, and rollback mechanisms for large-scale predictive systems. | This work inspired the rollback logic and shadow testing mechanisms implemented in the project to ensure safe deployment. |
| 6 | Jay, H. et al. | *Machine Learning and Self-Healing Capabilities* | International Journal of Advanced Computing, 2023 | The paper explores how machine learning systems can autonomously detect performance degradation and apply corrective actions through retraining or rule-based governance. | The project extends these ideas by integrating self-healing concepts specifically into computer vision object detection pipelines. |
|  |  |  |  |  |  |

**CHAPTER 3: PROBLEM DEFINITION**

* Object detection models deployed in real-world environments are necessarily exposed to continuously evolving data distributions, generally referred to as data drift, which degrades model performance dramatically with time.
* The topic at the heart of this project is how an object detection system can adapt to such drift in a streaming environment without forgetting previously learned knowledge or compromising system stability.
* This challenge is further exacerbated by a lack of labeled data during deployment, the risk of catastrophic forgetting during model updates, and a need to make sure that recently trained models truly outperform previous ones before replacing them.
* This, along with proper model governance, requires evaluation, shadow testing, and mechanisms for rollback to avoid unsafe or performance-degrading deployments.

**CHAPTER 4: SOLUTION STRATEGY**

The proposed approach follows a multi-stage adaptive pipeline to maintain the performance of object detection in real-world, non-stationary environments. In each stage, there is a specific challenge related to handling drifts while maintaining system stability and governance.

**1. Monitoring Incoming Streams of Images**

The system shall continuously monitor incoming image data when deployed. Instead of depending on labeled data, it gathers low-level image statistics, internal feature representations, and prediction confidences to automatically present a real-world view of the behaviors of the data.

**2. Drift Detection Using Multiple Independent Signals**

Drift is detected by a combination of:

* Image-level statistics-mean brightness, contrast, entropy
* Feature embedding drift computed from backbone representations using cosine distance
* Prediction confidence and entropy trends to capture rising uncertainty
* Using multiple signals improves false positives of detection by increasing reliability.

**3. Controlled Incremental Learning**

Upon detecting drift, the system will immediately send a signal for an incremental fine-tuning rather than a full retraining. The replay buffer contains both the stable historical samples and newly drifted samples to balance the adaptation against knowledge retention. Early layers of the model are frozen in order not to break the already learned representations, while higher layers are updated in the pursuit of adapting new patterns of data.

**4. Automated Evaluation Against Baseline**

Then the new fine-tuned model is evaluated against this baseline, using quantitative metrics like mAP, precision, recall, and inference latency. Shadow testing will be done where the new model runs silently alongside the existing model with no interference to the live predictions.

**5. Rule-Based Model Governance and Rollback**

It follows a rule-based decision framework, which chooses whether the new model gets promoted or rejected. Then, if the adapted model cannot perform better than the baseline or violates any constraints of latency, the system automatically rolls back to a stable version. This allows for safe and explainable deployment decisions. This layered approach ensures robustness, interpretability, and safety in production, while allowing them to continue adapting.

## **Algorithm: Drift-Aware Adaptive Learning Pipeline**

**Input:**  
Baseline model Mb​, validation dataset Dval​, streaming data Dt​

**Output:**  
Updated stable model Ms​

*1. Initialize baseline statistics and embeddings using D\_val*

*2. For each incoming data stream D\_t:*

*3. Compute image statistics and feature embeddings*

*4. Measure drift using statistical, embedding, and confidence signals*

*5. If drift score exceeds threshold:*

*6. Construct replay buffer with old and new samples*

*7. Fine-tune model with frozen early layers*

*8. Evaluate new model against baseline*

*9. If performance improves and latency constraints satisfied:*

*10. Promote new model as stable model*

*11. Else:*

*12. Roll back to baseline model*

*13. Continue monitoring*

**DATASETS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl No.** | **Name Of Dataset** | **Author(s)** | **Attributes Of Datasets** | **Models / Algorithms Used / Attributes** |
| 1. | Street Scene Object Detection Dataset | **NullClass Ltd. (Proprietary Industrial Dataset)** | **Train Size:** ~3,800 images  **Test Size:** ~550 images  **Validation Size:** ~550 images  **Length:** ~4,900 annotated images  **Number of Classes:** 6 (Car, Bus, Bicycle, Motorbike, Person, Background)  **Number of Dimensions:** 640 × 640 pixels  **Annotation Type:** Bounding Boxes (YOLO format)  **Image Type:** RGB Street-view Images | **Models Used:** YOLOv8 (Baseline & Incremental Models)  **Learning Type:** Supervised + Online Incremental Learning  **Drift Handling:** Statistical Drift Detection, Embedding Drift, Confidence Monitoring |

For this project, the dataset has been provided by NullClass Ltd. It has been used in the context of a computer vision-based use case related to the industry domain. The dataset contains the annotation of street view images that reflect actual traffic scenarios. It bears a close resemblance to the actual datasets available for urban driving scenarios. It has been considered a proprietary asset. The annotation is in the YOLO format.

**CHAPTER 5: DESIGN**

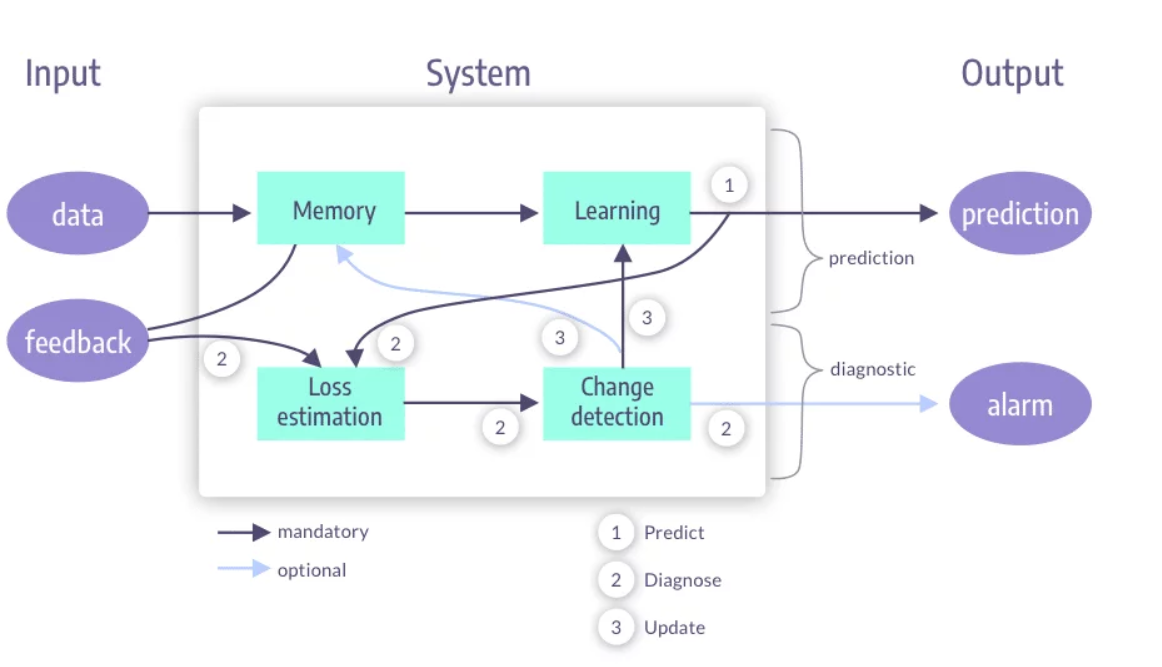


Figure 1: Closed loop drift-aware machine learning system [9]

This workflow illustrates a closed-loop drift-aware machine learning system designed for continuous operation in production. First, incoming data passes through the memory module, which holds the historical patterns. The learning module uses the current model to make predictions. Simultaneously, a loss estimation and change detection component monitors prediction behavior and data characteristics to identify deviations from the learned distribution. When significant drift is detected, the system will activate an update mechanism that adapts the model in a way that preserves prior knowledge. The outcome of this workflow consists of two paths: one with predictions and one with diagnostic alarms, thus enabling proactive intervention before severe degradation of the performance occurs.

Relevance to this project: It thus directly fits the philosophy of the proposed drift-aware object detection system, which tightly integrates a prediction with diagnosis of drift and controlled updates to ensure long-term robustness.

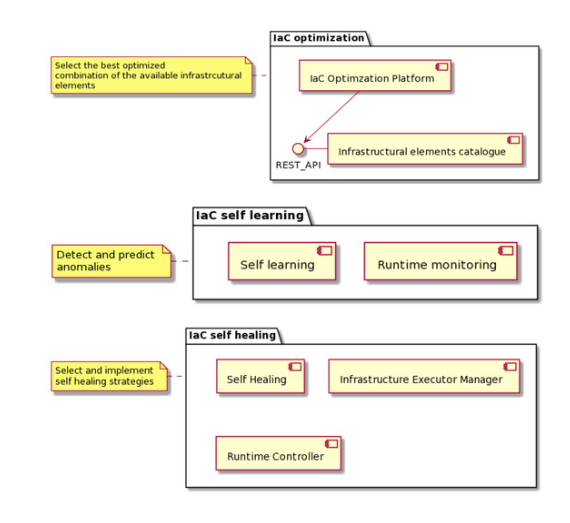


Figure 2: PIACERE approach for optimized and self-healed IaC. [8]

This workflow offers a hierarchical self-learning and self-healing approach to system reliability that is self-controlled. This is because the optimization level of the workflow is responsible for identifying the most optimal combination of system resource configuration. The self-learning level is self-controlled because it is constantly conducting system analysis at run time to identify inconsistencies that can be used to predict system failure when such inconsistencies occur. This informs the self-healing level to act accordingly to ensure that system stability is restored.

Relevance to this project: The concept of a workflow is equivalent to model governance and rollbacks in this project, whereby the automated evaluation and decisions by the rules ensure the safe promotion and rejection of the models.

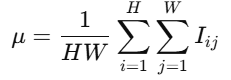
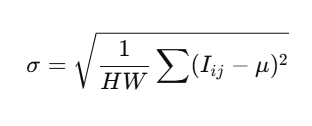
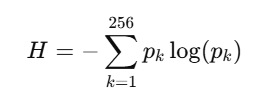
**CHAPTER 6: IMPLEMENTATION DETAILS**

## **The drift-aware object detection system would be implemented as a modular, end-to-end process that observes model performance even with non-stationary data and adjusts accordingly while maintaining system stability. The system would be developed with all the best practices of a drift-aware ML system.**

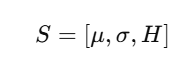
## **1. Drift Detection Module**

### **Image-Level Statistical Drift**

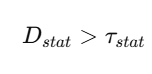
For each grayscale image ( I ) in the dataset, the following statistics are computed:

* **Mean brightness**  
  
* **Standard deviation (contrast)**  
  
* **Shannon entropy**  
  

where ( pk ) is the normalized grayscale histogram.

For a dataset of images, a feature matrix is formed:  


Drift score is computed as:  

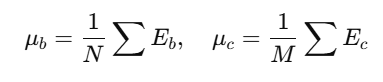

If:  


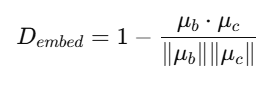
* **Statistical Drift Detected**

**Embedding-Based Drift Detection**

Feature embeddings are extracted from the YOLO backbone for each image. Let:

*  baseline embeddings
* current stream embeddings

Mean embedding vectors are computed:  


Drift is measured using **cosine distance**:  


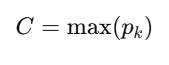
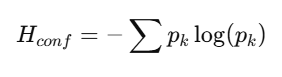
If:  


* **Representation Drift Detected**

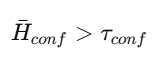
This captures semantic shifts even when pixel statistics remain stable.

### **Prediction Confidence Entropy**

For predicted class probabilities (p1, p2,…, pK ):

* **Confidence**  
  
* **Entropy**  
  

Average entropy across predictions is monitored:  


If:  


* **Model Uncertainty Increased**

### **Drift Decision Rule**

Final drift decision:

*IF (D\_stat > τ\_stat)*

*OR (D\_embed > τ\_embed)*

*OR (H\_conf > τ\_conf)*

*→ Drift Detected*

## **2. Online / Incremental Learning Module**

Once drift is detected, controlled model adaptation is triggered.

### **Replay Buffer Construction**

A replay buffer ( *R* ) is formed:  


where:

* ( Rstable ): randomly sampled pre-drift images
* ( Rdrift ): recent drifted samples

This prevents catastrophic forgetting by preserving prior data distribution.

### **Incremental Fine-Tuning Algorithm**

* Early backbone layers are frozen:  
  
* Only detection head parameters are updated:  
  

Training parameters:

* Learning rate 
* Limited epochs ( E )
* Small batch size for stability

This balances **plasticity (learning new patterns)** and **stability (retaining old knowledge)**.

## **3. Evaluation Module**

Both baseline and incremental models are evaluated using identical validation data.

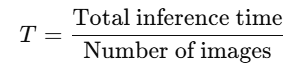
### **Performance Metrics**

Computed metrics:

* **mAP@0.5**
* **Precision**
* **Recall**

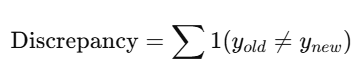
Delta improvement:  


### **Inference Latency Measurement**

Latency per image:  


Latency increase:  


### **Shadow Testing**

Predictions from both models are compared silently:  
****

High discrepancy indicates behavioral divergence.

## **4. Governance and Rollback Module**

Final deployment decision follows strict rule-based logic.

### **Promotion Rule**

*IF (ΔmAP > 0)*

*AND (ΔLatency ≤ Threshold)*

*→ Promote New Model*

*ELSE*

*→ Rollback*

### **Model Versioning**

Models are stored as immutable artifacts:

*model\_v1.pt → model\_v2.pt → stable\_model.pt*

Ensuring traceability and auditability.

## **Algorithm Summary**

*1. Extract statistics, embeddings, confidence*

*2. Compute drift scores*

*3. Detect drift using thresholds*

*4. Build replay buffer*

*5. Incrementally fine-tune model*

*6. Evaluate against baseline*

*7. Apply governance rules*

*8. Promote or rollback*

**CHAPTER 7: RESULTS AND DISCUSSIONS**

The proposed drift-aware object detection system was evaluated by comparing the baseline model with the incrementally fine-tuned model after drift adaptation. Performance was assessed using standard object detection metrics along with inference latency to ensure real-time feasibility.

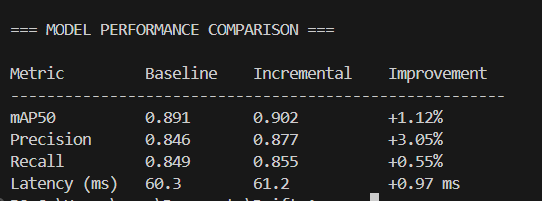


Figure 3: Code Results

On the other hand, the results of the progressive approach clearly show progressive improvement on all related metrics of accuracy. In addition to this, the largest improvement, which occurred on the precision metric, illustrates the reduction of false positives with adaptive learning.

The boost in the value of mAP @0.5 index reaffirms the effectiveness of the online learning approach in improving the overall detection accuracy without the need for retraining. Despite incurring an additional delay in processing, the computationally overhead is still well within the limits of being suitable for real-time execution applications.

From the shadow test result shown above, the prediction differences between the baseline model and the incremental model are very minimal. This confirms the efficiency of the replay buffer/partial fine-tune approach in reducing the problem of catastrophic forgetting.

To sum up, it can be seen from the results that controlled adaptation through an incremental approach is more effective than static training pipelines in combating data drift.

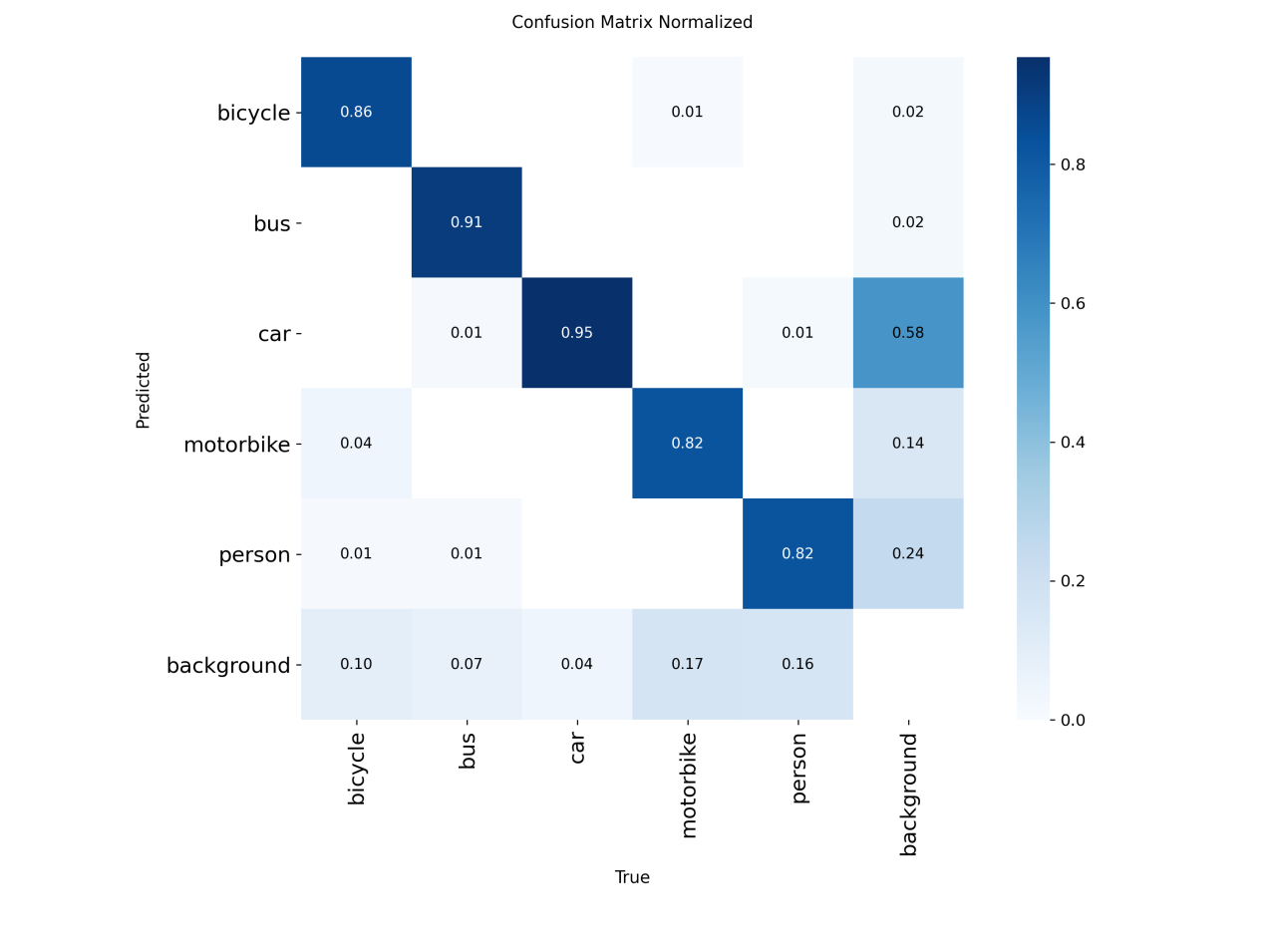


Figure 4: Confusion matrix normalized

Confusion Matrix (Normalized)

The normalised confusion matrix represents the class-level prediction characteristics of the object detection system. Strong diagonal elements prove correct classification on a large scale for classes like car, bus, and bicycle. This represents proper object discrimination. Smaller elements in the off-diagonal part of the matrix show that the object detection system has less confusion between classes with considerable likeness (motorbike and person). Misclassification with the background class is also small, which proves appropriate separation between foreground and background.

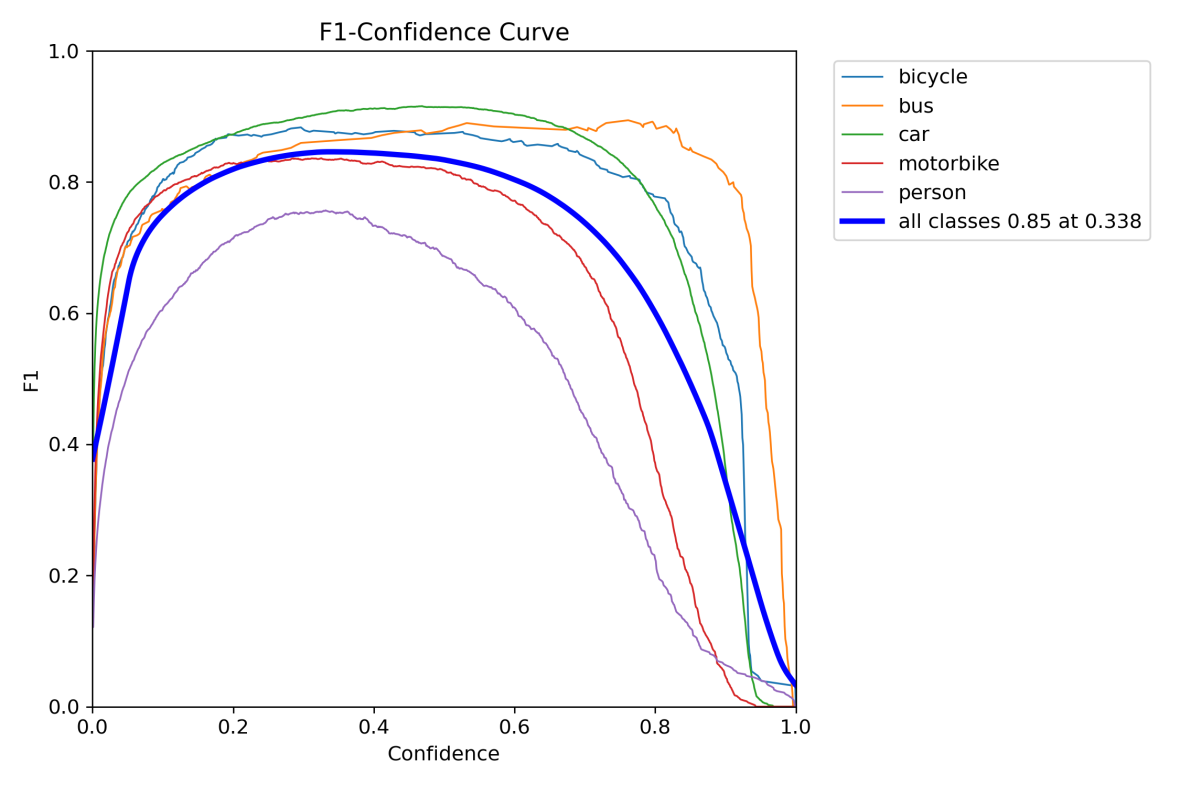


Figure 5: F1 confidence score

F1 – Confidence

The F1-confidence curve explores the trade-off between precision and recall at various confidence levels. The curve maximizes at the confidence of about 0.33 with the highest F1 value of ~0.85 for all the classes together. This implies the best trade-off between the false positives and false negatives. The class-wise curves illustrate that the vehicles are more stable at confidence thresholds, but the person class has higher sensitivity associated with it due to high-intra-class variability.

****

Figure 6: Precision-Recall Curve

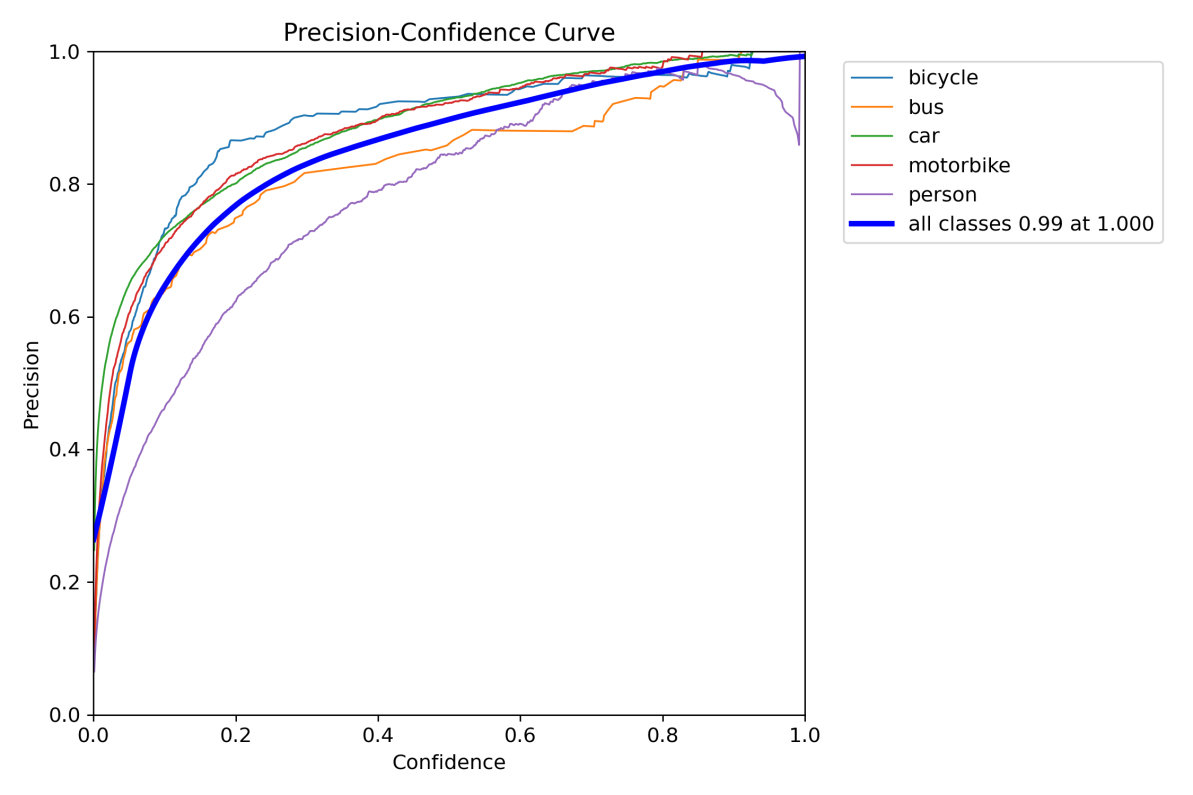
****

Figure 7: Precision-Confidence Curve

Precision–Confidence Curve

It can be seen from the precision-confidence curve that there is a monotonic increase in precision with an increase in the threshold of confidence. The model achieves almost perfect precision for thresholds higher than mid, which assures that high-confidence predictions are very reliable. This behavior becomes very important when the model has to be taken to deployment for which false positives need to be kept as low as possible. All-class aggregated curve reaches precision of ~0.99, indicating strong confidence calibration of the detector.

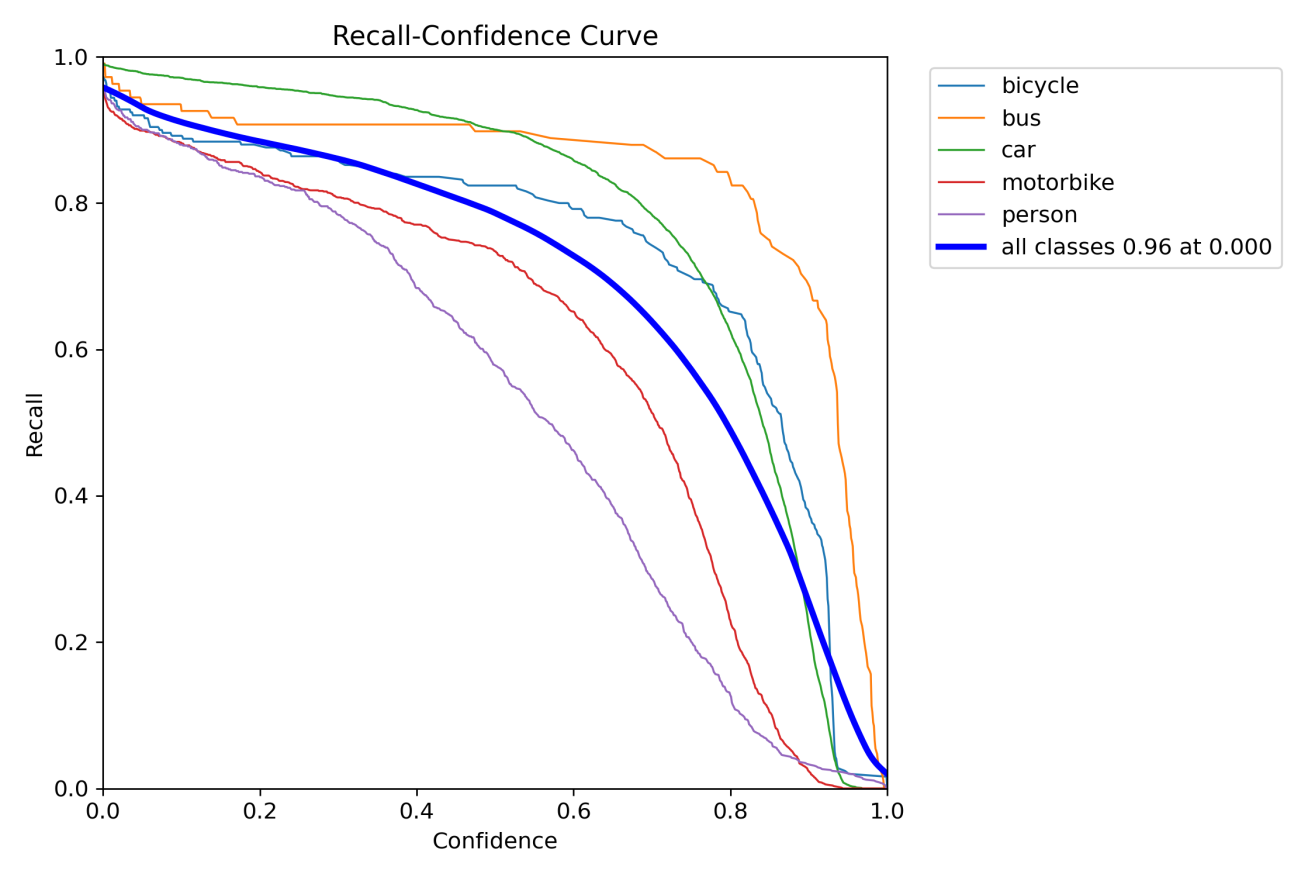
****

Figure 8: Recall-Confidence Curve

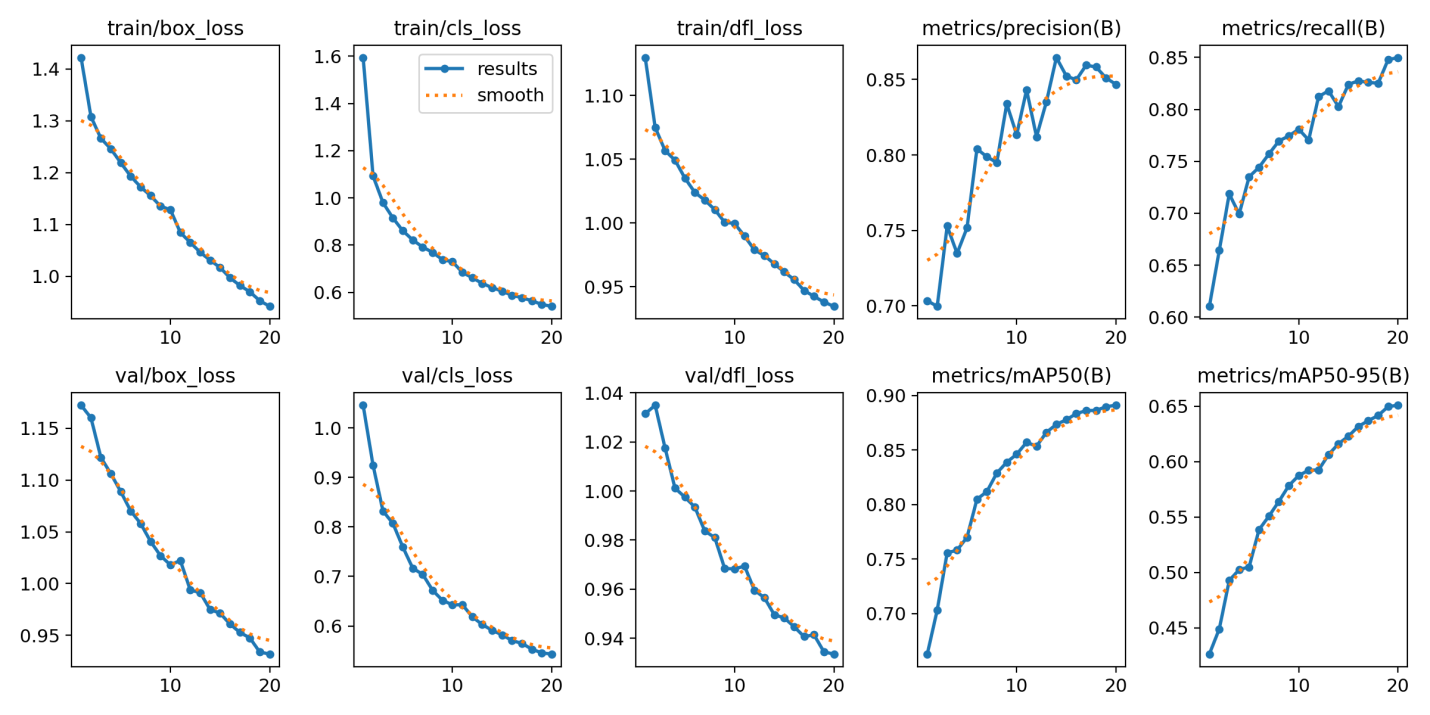


Figure 9: Results – Baseline YOLO (Graphed)

Training and Validation Metrics (Baseline YOLO)

The baseline YOLO model converges consistently during training. The losses of the training and validation sets include box loss, classification loss, and distribution focal loss that keep decreasing with the epoch, while precision and recall and mAP climb upwards correspondingly, thereby confirming that stable learning without overfitting has happened. However, performance plateaus after convergence, which indicates that the model may have limited adaptability in the future data shift.



Figure 10: Results – CCTV Baseline YOLO

This was trained in baseline yolo, this was the initial training phase where we were training and testing the data on, working cctv cameras, since it was initial training we labeled the data, as in the vehicles with separate numbers like for cars – 2, bicycles – 0, people – 4, then we trained more with incremental model, were we were able to label it with names.

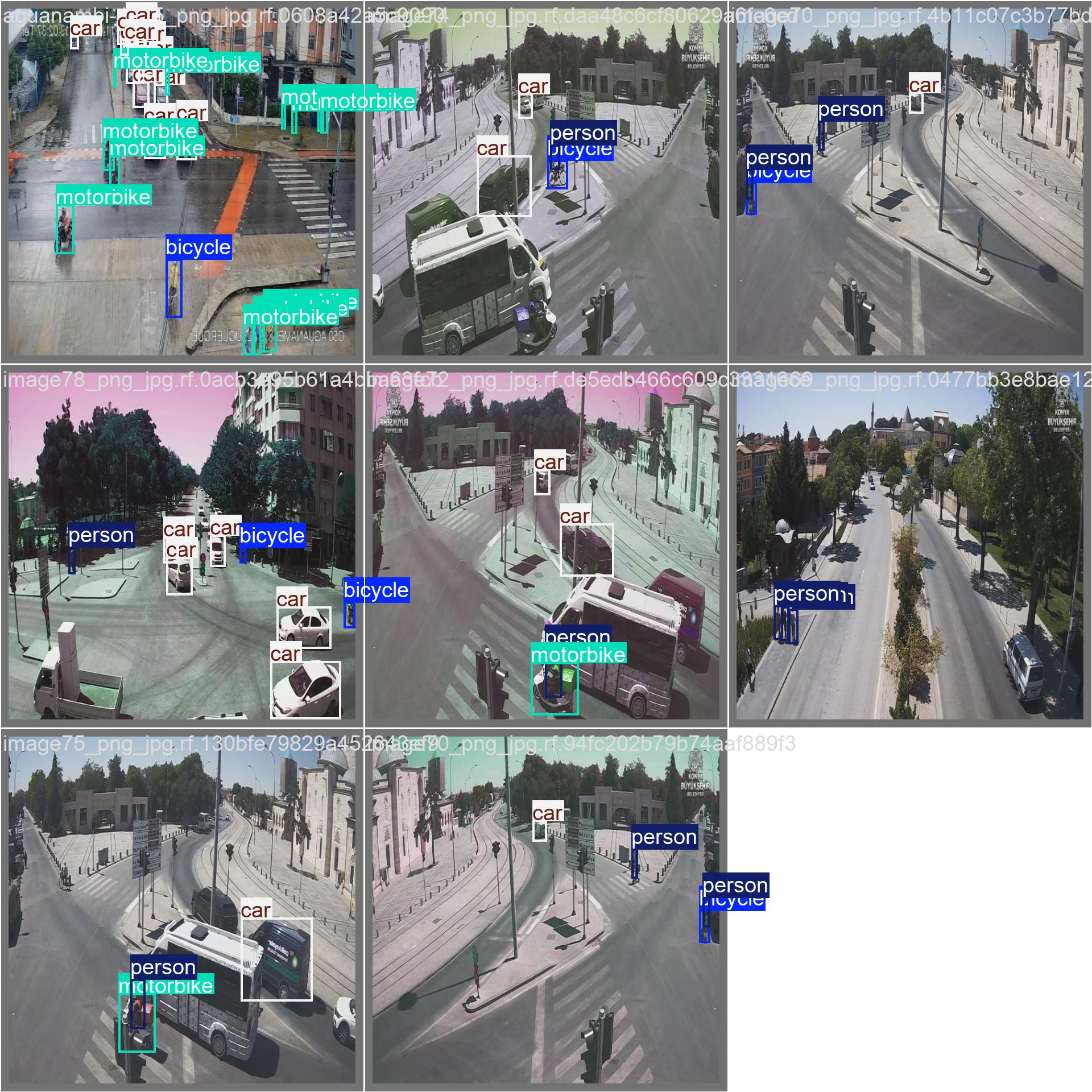


Figure 11: Results – CCTV Incremental model

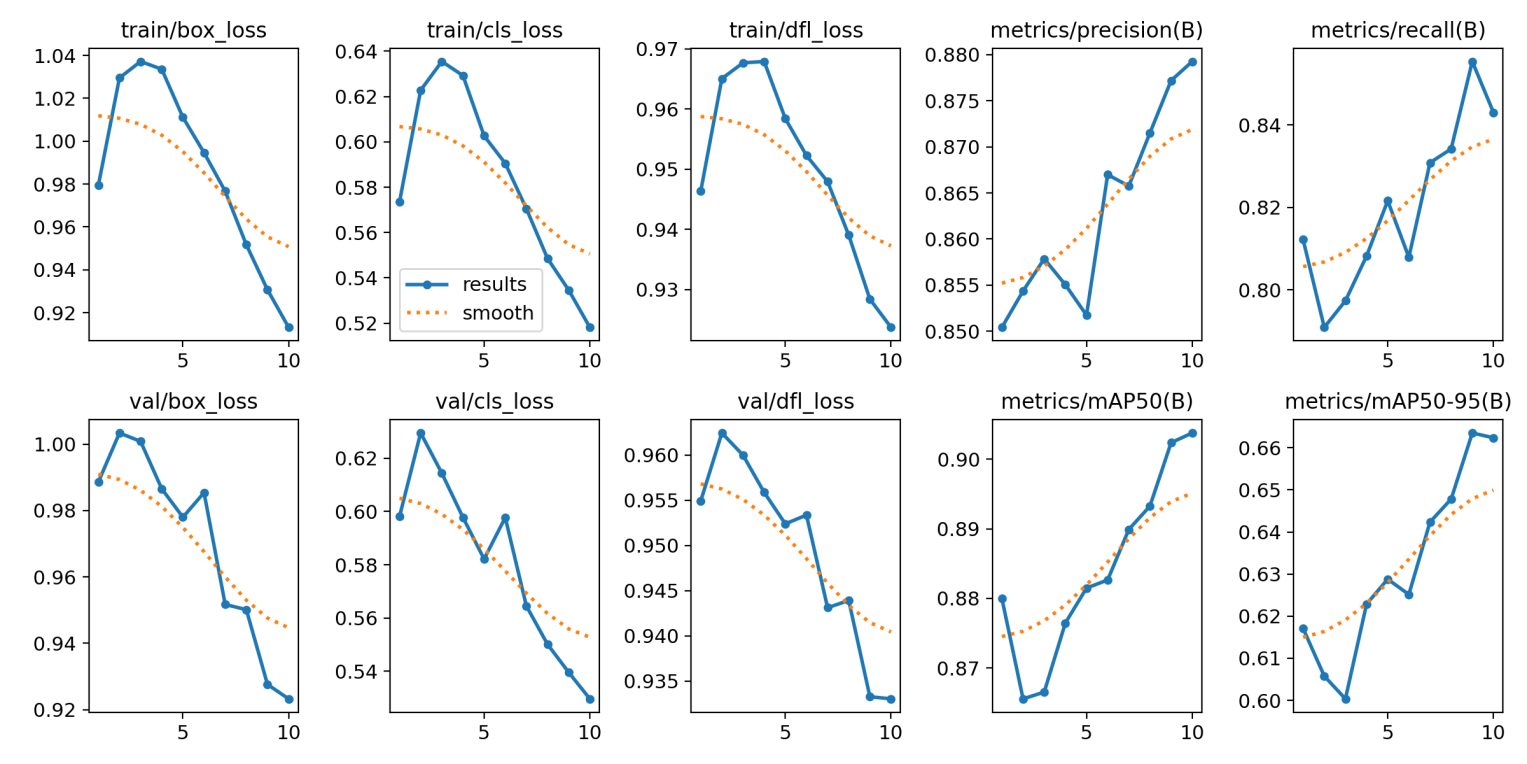


Figure 12: Results – Incremental model (Graphed)

**Incremental Model Performance (Incremental Model 2)**

The incremental learning approach shows better convergence properties with a lesser number of epochs because of the benefits of transfer learning and the freeze of the initial layers. The incremental learning approach outperforms the baseline in terms of mAP50, precision, and recall values. The learning latency remains comparable. The validation curves show better adaptation with the selected approach that uses the benefits of the replay buffers.

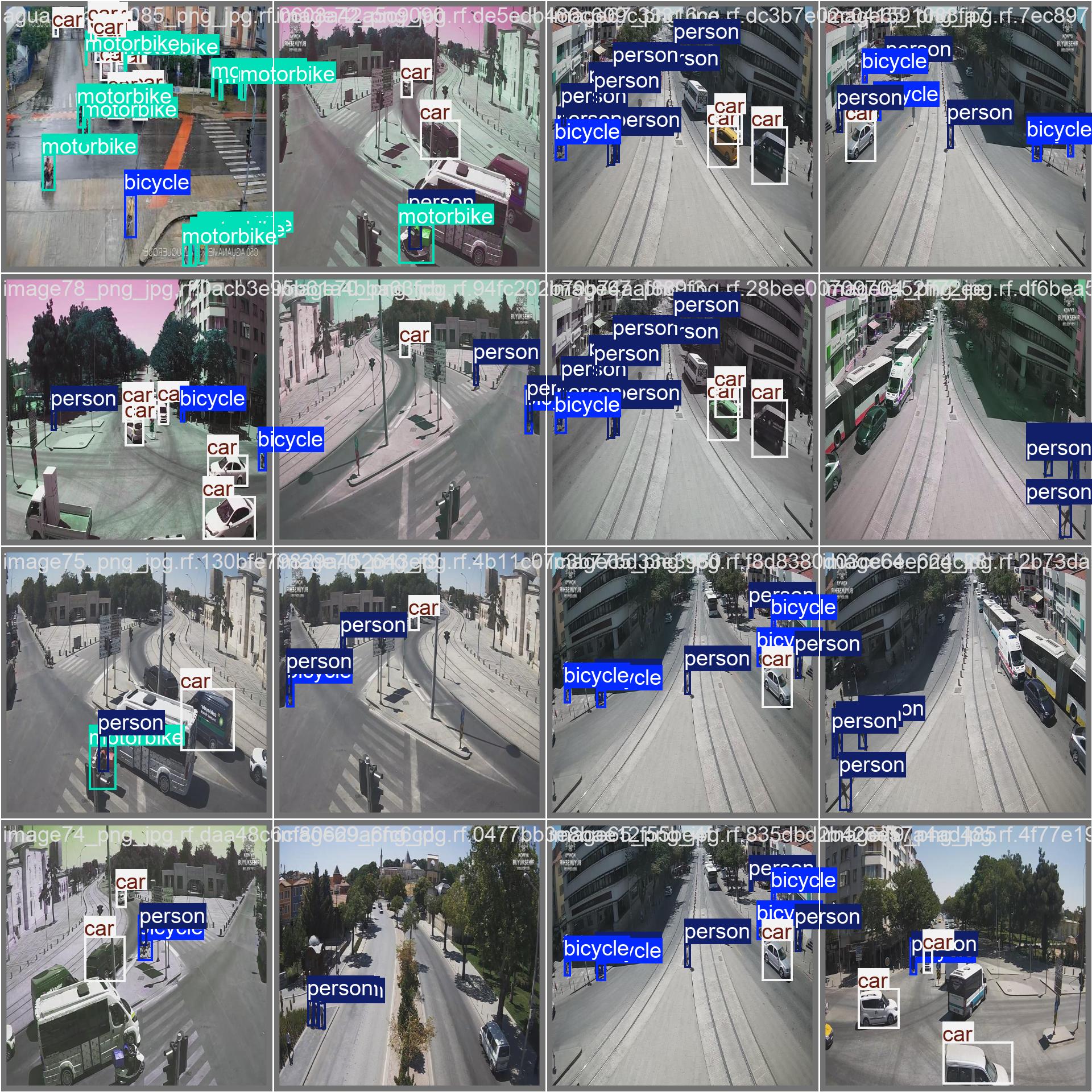


Figure 13: Final recording

The above is the final trained outputs, which was retrieved while the cctv was taking the footage simultaneous

Below are the images from our company website, where we train and applied (we were told to show older images)

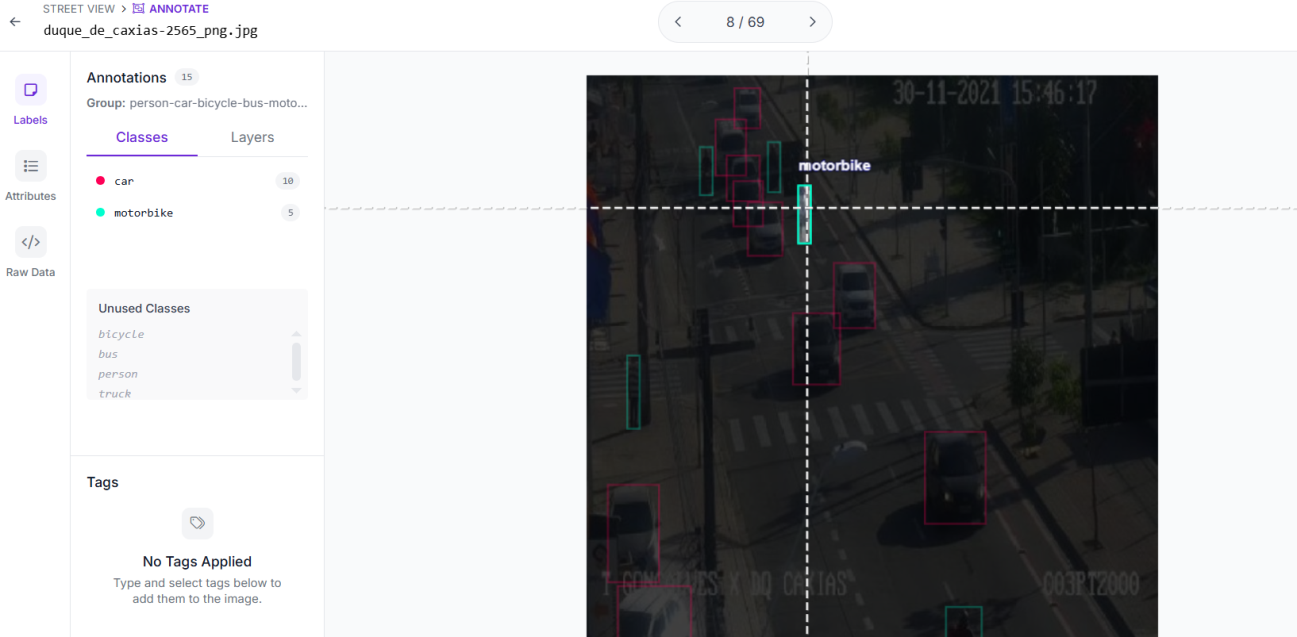
** **

Figure 14: Website we trained

**CHAPTER 8: CONCLUSION**

This project has been successful in proving the design and implementation of an online learning system with awareness of drift that can sustain and enhance object detection performance in an environment which keeps on changing all the time. By implementing statistical drift detection, embedding space monitoring, confidence stability measurement, online learning using a replay buffer, automated test, and rollback control according to rule-based systems, it has overcome regular machine learning systems.

Contrary to conventional object detection tasks, where data distributions are modeled within a static environment, this study directly tackles issues of model degradation post-deployment. The research findings ensure the effectiveness of incremental learning in enhancing model accuracy post-drift while maintaining consistency in model inference and usability. The addition of shadow testing and automated promotion evaluation enhances the deployability of the system.

In general, the proposed framework is long-term stable, self-aware, and self-governed, hence suitable to be used in real-world industrial scenarios.

**CHAPTER 9: LIMITATIONS**

In spite of its efficiency, the proposed system has a few limitations. The thresholds of the drift detector are set manually, which may require adjustments depending upon environments as well as the nature of the data. This is because the incremental training phase is computationally expensive when performed by CPU-based systems.

In addition, the existing shadow test process emphasizes comparing at an output level, rather than at a semantic level of accuracy, which might make it difficult to understand, for example, in more complex settings. Lastly, it must be mentioned that despite simulated reality constraints, fine-tuning even now involves access to labeled examples, which is not necessarily always readily available.

**CHAPTER 10: FUTURE SCOPE**

There are a number of improvement areas which may be pursued in order to further enhance this theme. Future versions of this work may include adaptive drift threshold levels based on historical data from the system. Semi-supervised or self-supervised machine learning methods may be employed in this respect.

The system can be expanded further to incorporate MLOps pipelines that support CI/CD. The addition of multiple modes of drift detection, such as the combination of visual data with metadata such as time, location, and sensors, can increase the sensitivity of drift detection. As a final suggestion, running the system on a cloud-native setup with real-time monitoring dashboards can increase the scalability and enterprise readiness of the system.

**REFERENCES**

[1] <https://arxiv.org/html/2508.17975v1>

[2] <https://www.iguazio.com/blog/concept-drift-deep-dive-how-to-build-a-drift-aware-ml-system/>

[3] <https://arxiv.org/abs/2411.00186>

[4] <https://blog.solugenix.com/harnessing-ai-ml-for-self-healing-systems>

[5] <https://sarcouncil.com/download-article/SJECS-246-2025-923-931.pdf>

[6]<https://d1wqtxts1xzle7.cloudfront.net/121802411/2023_paper_8_Harshal_Jay-libre.pdf?1741873773=&response-content-disposition=inline%3B+filename%3DMachine_Learning_and_Self_Healing_Capabi.pdf&Expires=1767004698&Signature=JA1uh4qi9AmZu6YbUqekGXl1FXIPF-ExS73v7ukIPt8U0sBBzRw9BXRDYV~wVyRSs7X9jsLIFFYREAEDJbhlYDWGoMsgi44U7b50TEszdcrLwFIMHCod5csaxMHLlU5iqh4MVb4D0FV-TzmVqBSFFag8-bb7HOBgltv7bksjepiKBsbMvQuNO30MtM5EfTr94oVmL0lRsGoqeQI2WEpWy2dgaFSjj9MiPVHcRbaHCpp5wB0VxawFm6wE7Vv~KLlSqxZ~Vi1UMq36s5FlqtZr8BK-vjoEEKOAASximvekZO-wViKu3-dyvTXdlGnoznKrUFcuIHwfGIKSkhHEvgUqPQ__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA>

[7] <https://www.researchgate.net/profile/Oladeji-Olaniran/publication/396166782_RESILIENT_MLOPS_INTEGRATING_REAL-_TIME_ANOMALY_DETECTION_WITH_SELF-_HEALING_DATA_PIPELINES_FOR_ROBUST_MODEL_DEPLOYMENT/links/68e00055ffdca73694b5387d/RESILIENT-MLOPS-INTEGRATING-REAL-TIME-ANOMALY-DETECTION-WITH-SELF-HEALING-DATA-PIPELINES-FOR-ROBUST-MODEL-DEPLOYMENT.pdf>

[8] <https://www.mdpi.com/2078-2489/12/8/308>

[9] <https://www.iguazio.com/blog/concept-drift-deep-dive-how-to-build-a-drift-aware-ml-system/>