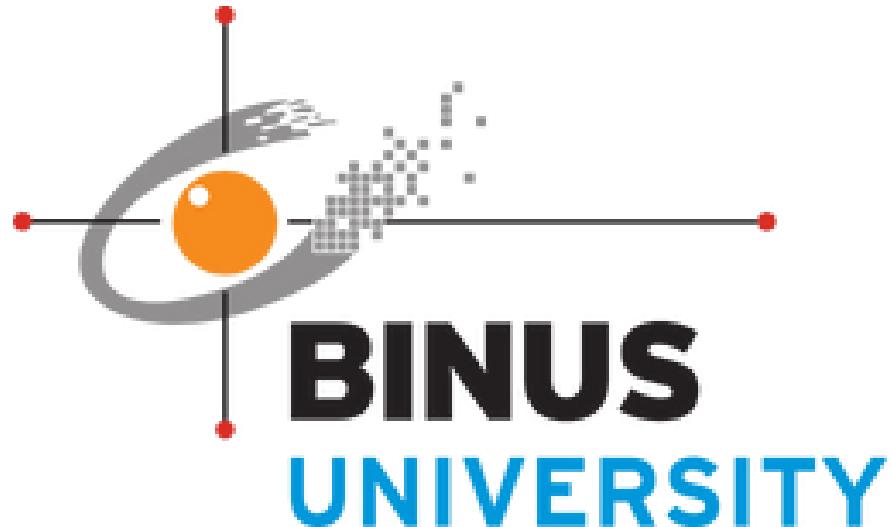


**Secure and Safe Traffic Watcher (SSTW): A Smart Application for  
Future Traffic Monitoring and Response**



Written by:

2702210232 - Stanley Jonathan Wahjudi

2702218891 - Ian Mulya Chiuandi

2702307972 - Jonathan

COMP7116001 - Computer Vision

Computer Vision Class LE01, Semester 5

BINUS University

2025

## TABLE OF CONTENT

TABLE OF CONTENT	1
CHAPTER I: INTRODUCTION	2
CHAPTER II: RELATED WORK	3
Accident Detection from Video	3
Advancement of Deep Learning in Accident Detection	3
CCTV / Surveillance-Focused Methods	3
CHAPTER III: METHODOLOGY	4
Dataset	4
Model Setups	4
Deployment	5
CHAPTER IV: IMPLEMENTATION & RESULTS	6
EDA and Preprocessing	6
Model Training	6
Deployment	7
Results	7
Classical Classifier	7
Classical Detector	8
YOLOv8-n	9
CHAPTER V: DISCUSSION & LIMITATIONS	12
CHAPTER VI: CONCLUSION & FUTURE WORK	14
REFERENCES	15
APPENDIX	16

## CHAPTER I: INTRODUCTION

Traffic accidents represent a significant global concern, accounting for 1.35 million fatalities and an additional 20-50 million non-fatal injuries annually. Given the constant interaction of humans with personal and public vehicles, such tragedies are statistically unavoidable. While the complete eradication of accidents may be an external inevitability, efforts should focus on enhancing accident response mechanisms. Minimizing the time between a traffic accident and the provision of medical treatment is paramount in reducing fatalities. Currently, conventional methods for monitoring and reporting traffic accidents in Indonesia, including manual observation, vehicular sensors, and bystander reports, are inadequate due to their limited coverage and high latency. The implementation of a dedicated monitoring system would address these shortcomings.

The current trend of smart cities in Indonesia are city-wide surveillance systems, CCTVs are strategically placed to cover the widest view possible to detect all vehicle traffic. Applying computer vision to these CCTVs offer a great possibility of detecting and reporting accidents in near real-time. Recently, researchers have shown advances in scene understanding using deep learning models, particularly Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN).

However, limited research has investigated CCTV-based accident detection using spatio-temporal neural architecture without relying on external object detectors such as YOLO. This methodology typically results in a binary classification problem, where the model discerns between an accident and a non-accident, rather than facilitating continuous scene comprehension and alert generation.

In this paper, we introduce a deep spatio-temporal learning framework for accident detection in surveillance footage, with the objective of generating timely alerts or providing event descriptions. This endeavor aims to contribute to a reproducible benchmark for live CCTV-feed accident detection and underscores the potential of purely video-based deep learning for intelligent traffic monitoring in smart city environments.

## CHAPTER II: RELATED WORK

### Accident Detection from Video

Early research treated accident detection as an anomaly detection problem, where motion trajectories and handcrafted features were analyzed to spot unusual events [Li et al., 2018; Tian et al., 2019]. These methods, however, struggled with complex traffic scenes and occlusions.

### Advancement of Deep Learning in Accident Detection

Researchers began using CNN to capture spatial features in each video frame, like what shapes or objects appear and where. LSTM models are able to capture how those objects in the video move or change over time, capturing temporal features. [Pawar & Attar, 2021; Robles-Serrano et al., 2021]. Several works treated accident detection as anomaly detection [Khan et al., 2022], applying spatio-temporal autoencoders on training datasets of normal driving, and the model flags vehicle accidents as anomalies based on its reconstruction value. These methods achieved strong performance on synthetic or dashcam datasets such as DADA-2000 and A3D, but often failed to generalize to fixed surveillance perspectives.

### CCTV / Surveillance-Focused Methods

Recent studies have started leveraging city surveillance datasets with static camera views [Adewopo et al., 2023]. Unlike mobile dashcams, CCTV footage presents stable viewpoints but complex occlusions, varied weather, and multiple motion scales. To handle these challenges, methods have employed ConvLSTM or 3D CNN-based encoders that model scene-level motion rather than relying on object detection.

## CHAPTER III: METHODOLOGY

### Dataset

The A9 Dataset, requested via TUM through providentia, used in this research is not made publicly available by the creators. One subset, subset 0, is the primary focus for exploration and serves as the dataset for this research. This subset of the dataset contains recordings and frames from four camera angles, which is then divided into three parts; where each part contains frames of different weather conditions: **sunny**, **foggy**, and **snowy**. There are 6 categories in total, which are (label ID shown in parentheses): Truck (0), Car (1), Van (2), Trailer (3), Other Vehicles (4), and Pedestrian (5).

### Model Setups

Since the main goal of our application, the SSTW, is to detect vehicles thus enabling and providing good tracking performance, we proposed the usage of two models for this task: a model with classical feature extraction in addition to Linear SVC (mirroring the original RCNN architecture), and a YOLOv8n model. We use these two models because we want to compare the performance between traditional and modern SOTA models. For the traditional model, the preprocessing pipeline is custom-made as it doesn't have a trainable features extraction layer. The feature extraction itself was done offline and only once before training, because the resulting output will not change at any point. These features then became the actual inputs instead of the raw pixels. On the other hand, the modern model YOLOv8n uses the raw pixel as the input. Resizing the raw image to 640x640 px is done for both models.

The models will then be evaluated using different approaches. The classical model focuses a lot more on how the classifier would predict a patch of image to be of substance. This was the reason why the classical model was evaluated in how it would classify a patch of image (in the form of a feature vector) and the classifier as through the detector performance. YOLOv8n is only evaluated as a detector. The metrics are as follows:

- mAP\*
- FPS\*
- Accuracy\*\*
- Recall\*\*

- Precision\*\*
- F1-Score\*\*
- ROC-AUC\*\*

\*detector metric, \*\*classifier metric

The tracker model uses DeepSORT on top of the detector model as a tracker which is the aim of this paper. DeepSORT uses a combination of mathematical formulas (Kalman Filter, Hungarian Algorithm, and Cosine Distance) and deep neural network to track objects with IDs in collaboration with a detector. DeepSORT acts only to track the vehicles and is not trained using the data rather just as an implementation of the tracker.

## **Deployment**

YOLOv8n is used as the deployed model in the website as it actually performs at a reasonable, possibly real time pace, instead of the two-stage detector like the classical model. YOLOv8n was exported to ONNX, where it was loaded using the Ultralytics' YOLO class. The backend employs a multithreading strategy to ensure that the model's calculations are non-blocking to each other and the server, which may cause huge performance damage if this strategy wasn't employed.

## CHAPTER IV: IMPLEMENTATION & RESULTS

### EDA and Preprocessing

To ensure the same size for all inputs, all images were resized to 640x640 while the patches were resized to 128x64 before feature extraction would be applied to every single image. The preprocessing step involved different methods for different weather conditions. Since vehicles of the same type might have different colors of paint, the colors were omitted by the classical model. This was done by converting all frames to GRAY for the feature extraction itself and all feature extraction methods only received one grayscale channel. All frames were a little grainy, hence a weak median filter was applied before dehazing was done only to foggy frames. The foggy frames had little variations across the axis (which was the primary method to identify them). This characteristic also unfortunately hazed the vehicles, resulting in bad predictions from the model. Dark Channel Prior (DCP) was applied to address this problem. In inference, the feature extraction layer looks for foggy images which are images with variance of laplacian below 50 (the value has been calibrated) before converting to grayscale (or else the layer will not know the variance of laplacian). CLAHE was applied to finalize the preprocessing because some images showed signs of unbalanced saturation.

All preprocessed images' features were then extracted using two methods: HOG and LTP. HOG was used because most variants were in the gradients, such as in how the values differ compared to their surroundings. However, HOG produced dynamic-sized vectors which were not ideal for classical models. This is why PCA was used to scale down the HOG length to a fixed-size. The PCA was trained on all features, then was run on 20 images to get the median of those images, where the coefficient computed up to around 95%; roughly about 839. Uniformity was applied to LTP to fix the vector length to 119, making it static. These features were then concatenated to form a vector of length 958 for the input.

### Model Training

The Linear SVC training was done on a jupyter notebook with the default parameter, which is **C = 1.0**. The penalty parameter C controls the trade-offs between model complexity and performance. In this case, C=1.0 is a good balance as each sample has a lot of variations, ensuring that the model is not hallucinating or underfitted. A linear kernel was used to mimic

the original RCNN architecture and to also handle the typical pattern of classical feature extraction method output. The Linear SVC itself was trained using the method One-vs-One and One-vs-Rest to see which training and inference method would lead to better performance. The classifier was then exported to ONNX format to then be concatenated with the classical model class, that has the feature extraction layer, for inference.

YOLOv8n was trained using its default parameters without modifications on the unprocessed resized to 640x640 images. The training was done across 100 epochs with the batch size of 32 and a patience of 10. The resulting parameters were then exported to ONNX format in two different forms: an fp32 normal ONNX model and a 16-bit quantized format, both of which had been simplified through Ultralytics' method, reducing redundant calculations.

## Deployment

The deployed backend uses Flask to serve the routes while utilizing threads and ONNX runtime to run the actual ONNX models. This mimics how the model would run on the edge side before going through fog then cloud (dashboard). The frontend utilizes Next.js for its versatility when developing either static or dynamic pages. Since SSTW contains both static pages (landing page, articles, etc) and dynamic pages (dashboard, alerts, etc), Next.js rendering methods; such as SSG, SSR, and Client-Side Rendering, helps a lot in the frontend and potential future deployment.

## Results

### Classical Classifier

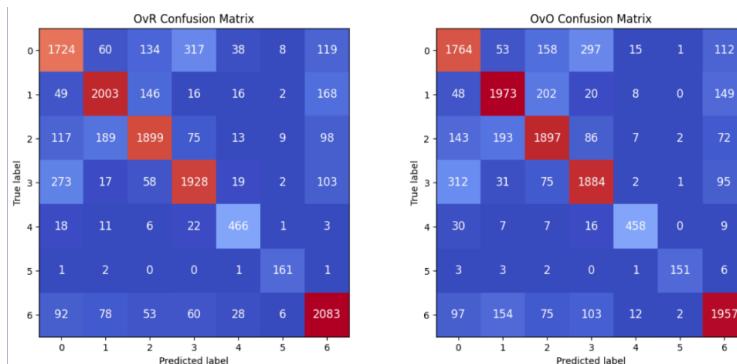


Fig. 1: Classical classifier confusion matrix.

One vs Rest (OvR)					One vs One (OvO)				
	precision	recall	f1-score	support		precision	recall	f1-score	support
0	0.76	0.72	0.74	2400	0	0.74	0.73	0.74	2400
1	0.85	0.83	0.84	2400	1	0.82	0.82	0.82	2400
2	0.83	0.79	0.81	2400	2	0.79	0.79	0.79	2400
3	0.80	0.80	0.80	2400	3	0.78	0.79	0.78	2400
4	0.80	0.88	0.84	527	4	0.91	0.87	0.89	527
5	0.85	0.97	0.91	166	5	0.96	0.91	0.93	166
6	0.81	0.87	0.84	2400	6	0.82	0.82	0.82	2400
accuracy			0.81	12693	accuracy			0.79	12693
macro avg	0.81	0.84	0.82	12693	macro avg	0.83	0.82	0.82	12693
weighted avg	0.81	0.81	0.81	12693	weighted avg	0.79	0.79	0.79	12693

Fig. 2: Classical classifier classification report.

The classical classifier performed well during training. There were two training setups for the classifier which involved the training method: One vs One and One vs Rest. The One vs One training method created 21 classifiers, where each classifier is tuned to compare one class with the other, and the highest confidence will be picked as the label. One vs Rest trained only 7 classifiers which are focused on determining one label against another label on one model, instead of one by one.

The one vs one training method had better generalization, especially on rare classes, which are class 4 and 5 (other vehicles and pedestrians). In comparison, the one vs rest training method performed better in every other class and had a better overall macro performance which potentially would do better to the overall detector performance.

### Classical Detector

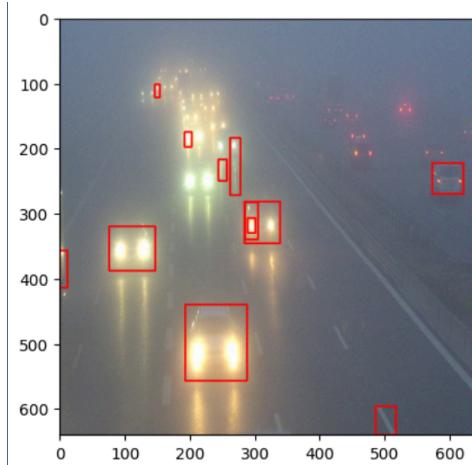


Fig. 3: Classical detector sample prediction.

Metrics	Value
mAP	0.77%
mAP50	1.6%
mAP75	0.6%

Fig. 4: Classical detector mAP over 30 frames.

The classical detector had really low recall. This most likely stems from the classifier inability to identify small and occluded vehicles. Not only that, the classical detector mAP is very low with only 0.77% overall. This indicated that there was a discrepancy between the classifier performance and the detector pipeline, suggesting the extraction layer was not robust to variance even with HOG and LTP combined.

## YOLOv8-n

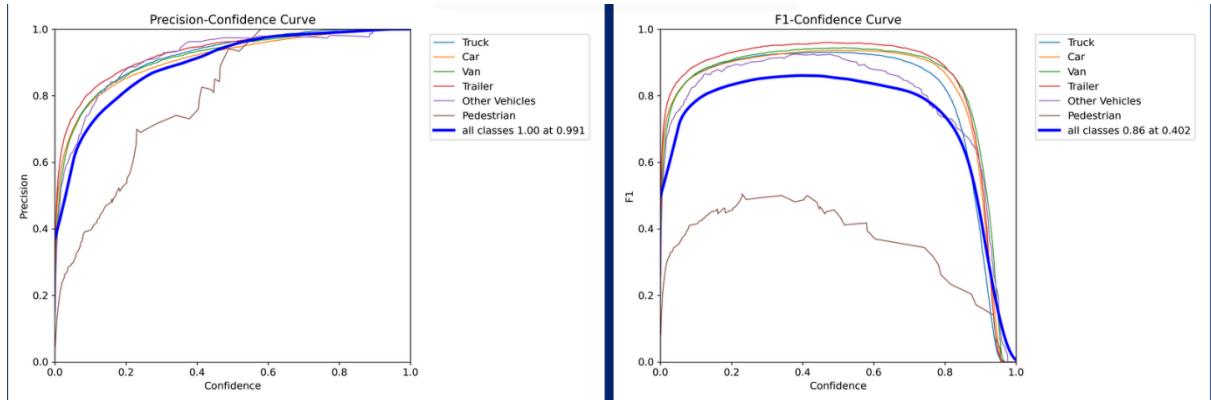


Fig. 5: Precision-confidence and F1-confidence curve

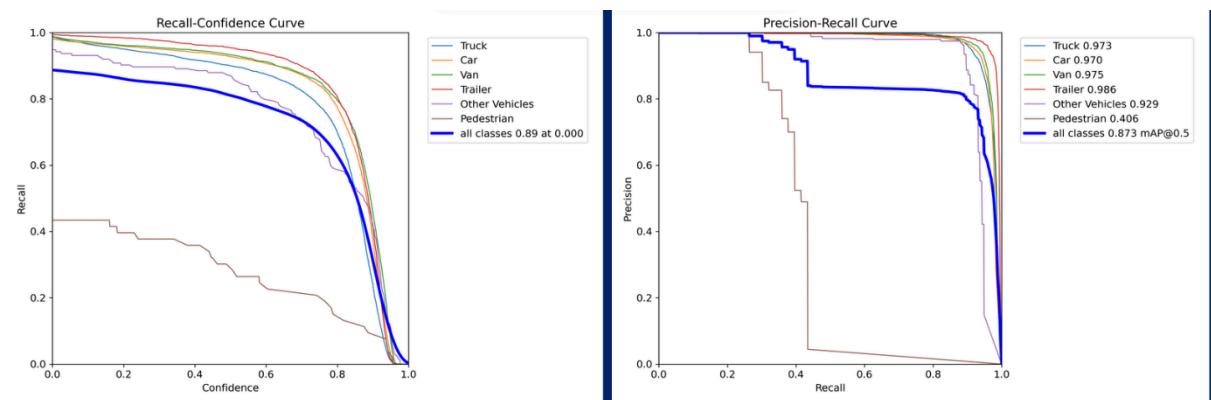


Fig. 6: Recall-confidence and precision-recall curve

All three metrics: precision, recall, and F1; are dropped on high-confidence boxes. This is because YOLO is more selective with its detections. This caused low recall by the model when predicting high confidence boxes. This also caused a drop on the precision when predicting these high confidence boxes.



Fig. 7: Sample predictions

YOLO created tight bounding boxes. This is a typical behaviour as YOLO employed a box refinement process, not only creating raw bounding boxes. From all predictions, YOLO could even predict occluded and small objects.

Metrics	Value (FP16)	Value (FP32)
mAP	76.15%	76.09%
mAP50	83.68%	83.68%
mAP75	81.09%	81.16%

Fig. 8: YOLO mAP over 30 frames.

Comparing the compression, YOLO performed similarly before and after quantization. The original FP32 model size was around 12 MB before quantization and only 6 MB remained after quantization. Quantization proved to be good since the performance did not drop.

Model	Frame Per Second	Latency (s)
Classical Model	0.005	200
YOLOv8n FP16	19.94	0.05

YOLO v8n FP32	27.77	0.036
---------------	-------	-------

Fig. 9: FPS and latency of each model.

The classical model is a two-stage detector, hence the high latency and < 1 FPS. This immediately eliminated the viability of it being in a real-time traffic watcher system. YOLOv8n models have lower latency as they are one-stage detectors. YOLOv8n FP16 once again delivered better performance compared to the original FP32 version. It inferred one model with only 71.8% of time the original FP32 version took.

## CHAPTER V: DISCUSSION & LIMITATIONS

From our study, we compared a classical two-stage detection pipeline with a modern deep learning-based one-stage detector for traffic object detection under challenging environmental conditions. The classical approach displayed a comprehensive feature extraction pipeline, combining handcrafted features (HOG and LTP) with a Linear SVC classifier for inference, while YOLOv8n represented the deep learning baseline.

The classical classifier demonstrated strong performance during isolated classification, particularly when trained using the One-vs-One (OvO) strategy, which showed better generalization on rare classes such as pedestrians and other vehicles. Meanwhile, the One-vs-Rest (OvR) strategy achieved better overall macro performance, suggesting that OvR is more effective when class balance and overall detector consistency are prioritized. This indicates that the choice of multi-class strategy significantly affects performance depending on class distribution and target use cases.

The test results for the classical models conclude that the combination of classical feature extraction and classifier is not ideal for a real-time traffic monitoring system, as it is computationally slow and not robust when implemented in the final detection pipeline. From the test results, it shows extremely low recall and mAP, thus shown being unable to handle large variances in scale, occlusion and environmental noise. Small and partially occluded vehicles were particularly difficult for the classical detector to identify, which significantly impacted recall.

On the other hand, YOLOv8n achieved a high performance and tight bounding boxes in the final detection pipeline, which shows its capability of detecting small and occluded objects. YOLOv8n also appeared to be stable in the training process with no sharp fluctuations, overshooting, or any other symptoms of bad training behaviours. We can credit this advantage to the internal feature hierarchy and learning process that allows the model to learn contextual and spatial relationships that classical feature extraction is unable to capture.

Moreover, the difference in latency in both models are drastically different. The classical detector's two-stage architecture was inefficient and computationally expensive, only achieving a maximum of 1 FPS. While YOLOv8n was a one-stage detector that was quantized to its FP16 version (50% of its original size), achieved much faster inference, making it ideal for real-time edge-deployment purposes for monitoring systems in CCTVs.

The experiment concludes that modern deep learning based detectors are far more effective than classical methods for real-time, end-to-end traffic object detection with low visibility, intense environmental noise and changing weather conditions.

We acknowledge the several limitations and gaps in our research. Firstly, the classical feature extraction relied on grayscaling images and handcrafted features, which limits representational capacity. Classes that may have similar shapes but different visual contexts, like van and SUV, may be misclassified due to the removal of potentially useful discriminative information.

Second, the Laplacian threshold for fog detection and preprocessing may not have generalised well to different lighting conditions, noise and camera qualities than the ones in the training dataset. This can increase errors in fog classification, which impacts preprocessing and downstream detection performance. Third, PCA-based dimensionality reduction may also discard discriminative features that are key for rare classes.

## CHAPTER VI: CONCLUSION & FUTURE WORK

This research examined whether a traditional two-stage object detection approach using handcrafted HOG and LTP features could realistically compete with a modern deep learning model, YOLOv8n. The comparison focused on detection performance and inference speed, with a particular focus on how well each approach could be deployed on edge devices.

From our experiments, YOLOv8n quantized to FP16 clearly performed better than the classical pipeline in terms of accuracy, speed, and overall practicality. With a model size of around 6 MB and an average inference speed of 30 FPS, the one-stage design of YOLOv8n allows it to run in real time even on limited hardware. In practice, this made it well suited for use in modern CCTV systems, especially those used in smart city settings. The model was also integrated into our deployed web application, where it showed stable and reliable performance during testing.

As part of this work, we also developed the first version of SSTW, a web-based dashboard for monitoring traffic from multiple CCTV feeds. The backend is built using non-blocking, multi-threaded processing, which allows video streams to be handled and analyzed at the same time without noticeable performance drops. This implementation shows that real-time object detection can be integrated into a scalable traffic monitoring system.

For future work, we plan to focus on two main areas: further optimization of the YOLO model and automation of alerting and reporting across multiple CCTV locations.

On the model side, we plan to explore task-specific improvements for vehicle detection, such as adjusting anchor configurations, focusing training on vehicle classes, and applying domain-specific data augmentation. These steps are expected to improve performance under different weather, lighting, and traffic conditions.

On the system side, we aim to extend the backend with an automated alerting and reporting mechanism that can detect unusual traffic situations or incidents and notify patrol officers in near real time. By combining intelligent alerts with data from multiple CCTV locations, the system is expected to support faster responses and more effective traffic management, helping address traffic congestion and incident handling challenges in Indonesia.

## REFERENCES

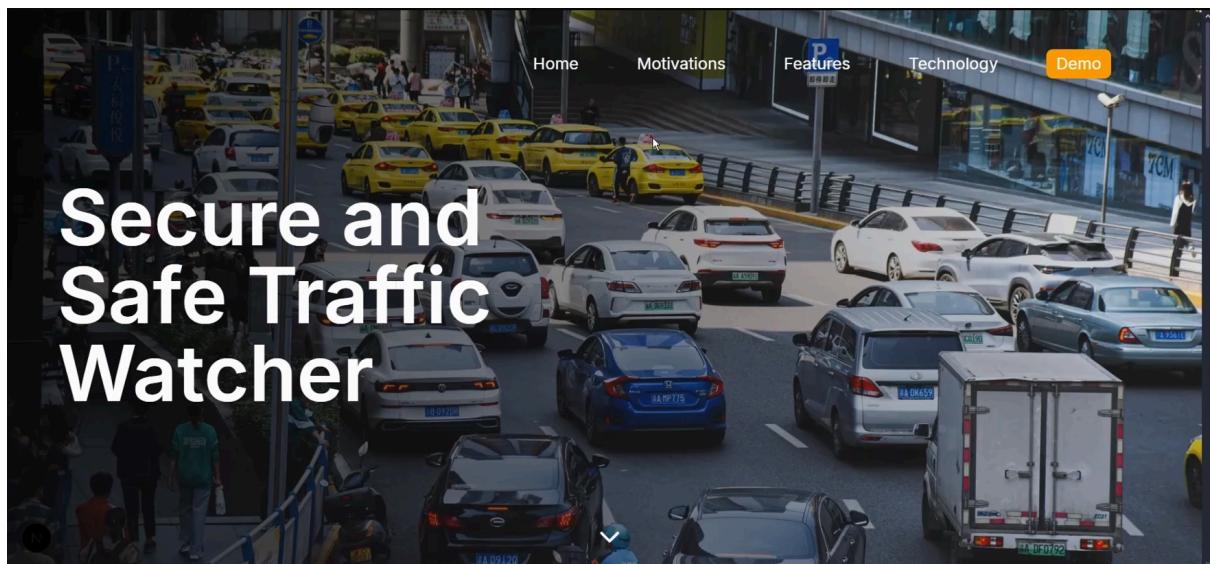
- BL, S. K., Suchetha, N. V., & Kumari, S. (2022). Enhanced local ternary pattern method for face recognition. *Journal of Scientific Research*, 66(2), 139-143.
- Creß, C., Zimmer, W., Strand, L., Fortkord, M., Dai, S., Lakshminarasimhan, V., & Knoll, A. (2022, June). A9-dataset: Multi-sensor infrastructure-based dataset for mobility research. In 2022 IEEE Intelligent Vehicles Symposium (IV) (pp. 965-970). IEEE.
- Girshick, R. (2015). Fast r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 1440-1448).
- Guo, H., Zhang, Y., Chen, L., & Khan, A. A. (2024). Research on vehicle detection based on improved YOLOv8 network. arXiv preprint arXiv:2501.00300.
- Kaiming He, Jian Sun and Xiaoou Tang, "Single image haze removal using dark channel prior," 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 2009, pp. 1956-1963, doi: 10.1109/CVPR.2009.5206515.
- Muhammad, M. B., & Yeasin, M. (2020, July). Eigen-cam: Class activation map using principal components. In 2020 international joint conference on neural networks (IJCNN) (pp. 1-7). IEEE.
- Prathamesh Amrutkar. (2024, May 9). The Complete Guide to Object Detection Evaluation Metrics: From IoU to mAP and More. Medium. <https://medium.com/@prathameshamrutkar3/the-complete-guide-to-object-detection-evaluation-metrics-from-iou-to-map-and-more-1a23c0ea3c9d>
- Ren, S., He, K., Girshick, R., & Sun, J. (2016). Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6), 1137-1149.
- Solawetz, J. (2020, May 6). What is Mean Average Precision (mAP) in Object Detection? Roboflow Blog. <https://blog.roboflow.com/mean-average-precision/>
- Suman, Y. R. (2025, June 13). Learn DeepSORT: Real-Time Object Tracking Guide. Labelerr. <https://www.labellerr.com/blog/deepsort-real-time-object-tracking-guide/>
- Ultralytics. (n.d.). Object Detection Datasets Overview. Docs.ultralytics.com. <https://docs.ultralytics.com/datasets/detect/>
- Wojke, N., Bewley, A., & Paulus, D. (2017, September). Simple online and realtime tracking with a deep association metric. In 2017 IEEE international conference on image processing (ICIP) (pp. 3645-3649). IEEE.

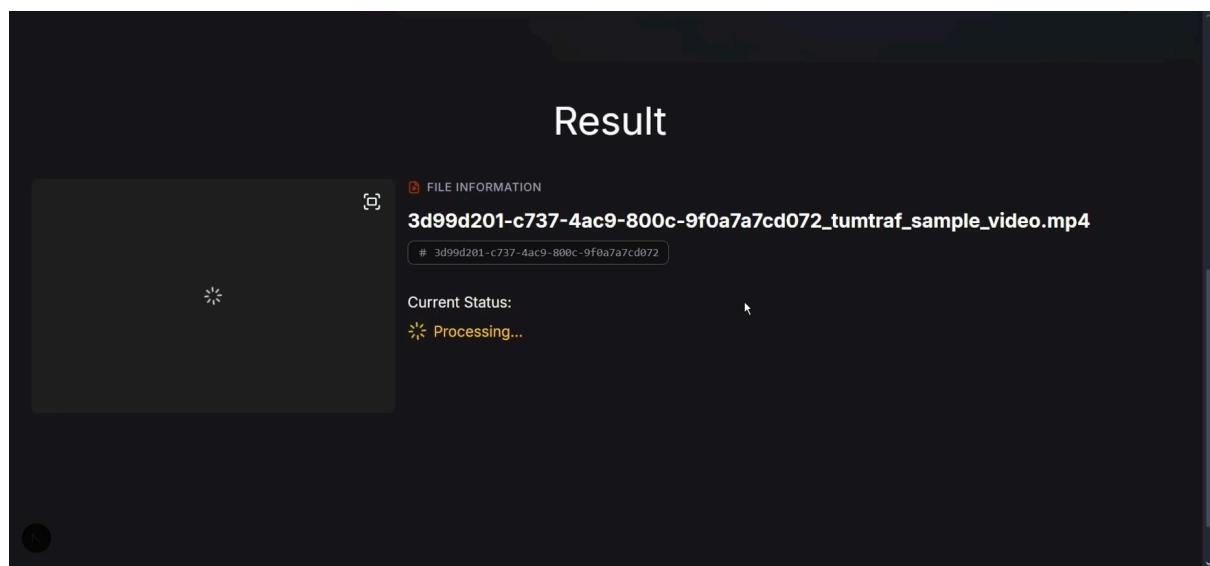
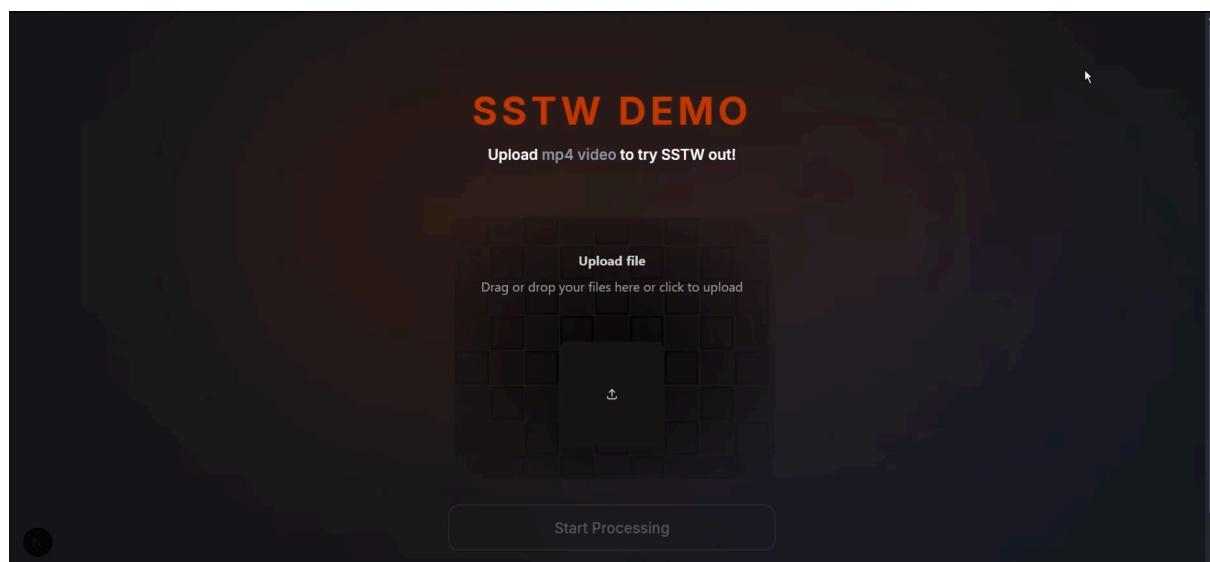
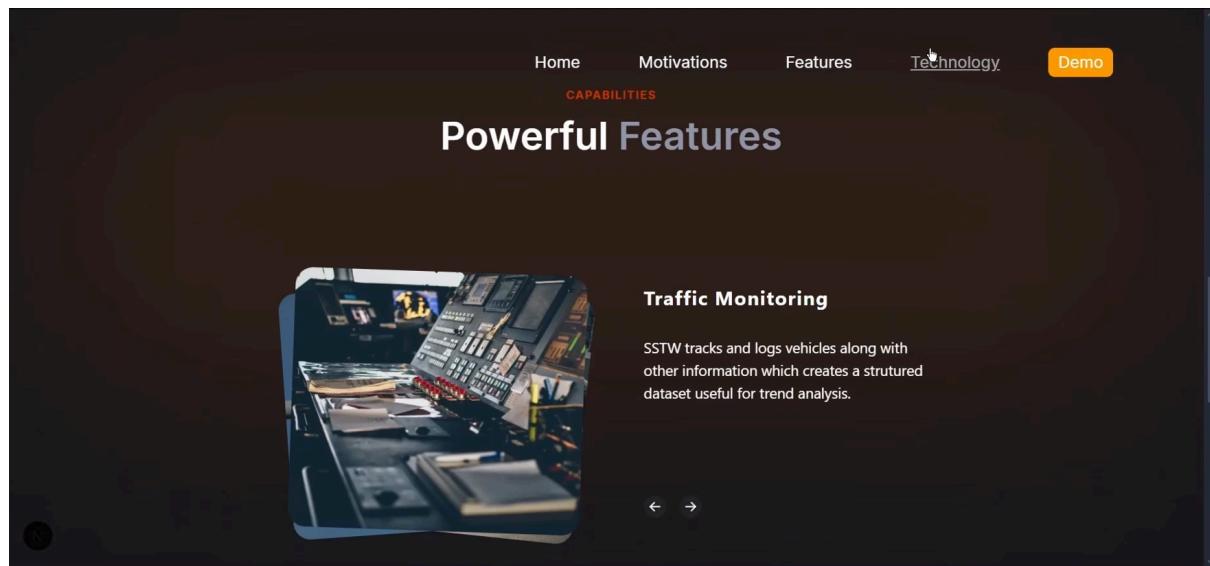
## APPENDIX

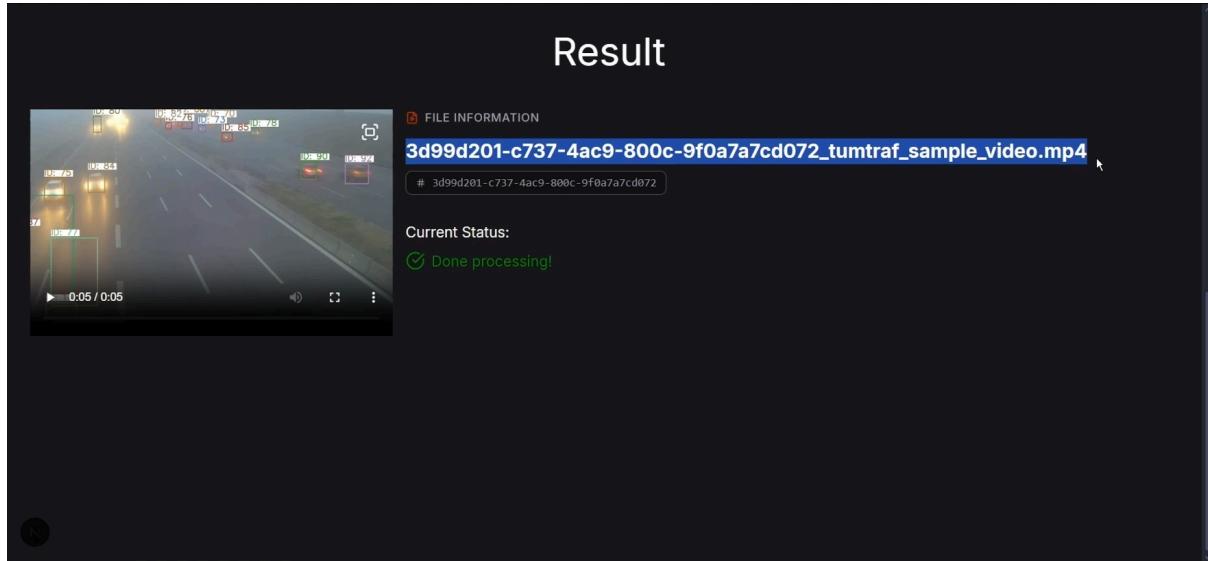
### Team Contribution Statements:

Name	Contributions
Ian Mulya Chiuandi - 2702218891	PPT Presentation (Original + Refactoring), Final Report (Chapter 1, 2, 6), Web App Frontend (Jinja Templating via index.html), Web App Backend (Setup Flask app.py, Input Image/Video, Send Media to Server), Model (FasterRCNN Attempt), Demo Video Recording.
Stanley Jonathan Wahjudi - 2702210232	Data handling, preprocessing, model training, frontend and backend refactoring (Enhance pages and adapt websocket), demo video recording + editing, and documentation, ppt creation, final report (Chapter 3-4).
Jonathan - 2702307972	Ideation and research into preprocessing methods and model, ppt creation and refactoring, initial demo video recording, backend frontend prototyping, final report (Chapter 5-6), final report refinement.

### Application Screenshots:







**Demo video Link:**

<https://drive.google.com/file/d/1VfVia68L5qTILrR8O4hvC0Yaw2rjerTd/view?usp=sharing>

**GitHub Link:**

<https://github.com/StanleyJo-37/Secure-and-Safe-Traffic-Watcher-SSTW>

**PPT Link:**

[https://www.canva.com/design/DAG7q468Clw/M7cxmP5Pc2rzqXFamutJMQ/edit?utm\\_content=DAG7q468Clw&utm\\_campaign=designshare&utm\\_medium=link2&utm\\_source=sharebutton](https://www.canva.com/design/DAG7q468Clw/M7cxmP5Pc2rzqXFamutJMQ/edit?utm_content=DAG7q468Clw&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton)