Journal of Information Systems Engineering and Management
2025, 10(48s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Traffic Signal Recognition Using the Neuro-T Deep Learning Platform: A Case Study on Real-time Object Detection in Urban Environments

Sunghyuck Hong[1]

[1]Division of Advanced IT, IoT major, Baekseok University, Cheonan city, Republic of Korea

ARTICLE INFO

Received: 18 Dec 2024

Revised: 10 Feb 2025

Accepted: 28 Feb 2025

ABSTRACT

This study investigates the effectiveness of the Neuro-T deep learning platform in detecting and classifying traffic signals from real-world urban environments. The accurate detection of traffic lights is crucial in autonomous driving systems and intelligent transportation networks. A dataset comprising 500 real-world traffic light images was annotated and trained using Neuro-T's GUI-based platform, employing a CNN-based architecture. The model achieved a classification accuracy of 94.8% and operated in real-time at 43 frames per second (FPS). The results demonstrate Neuro-T's potential as a lightweight and highly usable tool for rapid deployment of computer vision systems in urban mobility contexts.

Keywords: Traffic signal detection, Neuro-T, Deep learning, CNN, Intelligent transportation systems, Object recognition

INTRODUCTION

The increasing demand for autonomous vehicles and intelligent traffic systems has catalyzed research in computer vision-based scene understanding. One of the most fundamental aspects of this understanding is the ability to detect and interpret traffic signals accurately [6],[7]. This capability not only facilitates compliance with traffic rules but also enhances safety and efficiency across roadways. Failure to detect traffic lights appropriately can result in accidents, inefficient routing, and system failure, thereby negating the benefits of automation [3].

Modern traffic environments are often complex, featuring a wide range of challenges such as occlusion, poor lighting, and varying signal designs. As such, the computer vision algorithms employed must be robust, adaptable, and capable of generalizing across multiple real-world scenarios. While methods like YOLO (You Only Look Once) [1], SSD (Single Shot Detector) [2], and Faster R-CNN [3] have demonstrated strong results in object detection tasks, they require significant computational overhead and complex configuration setups, making them less accessible to non-experts and rapid prototyping teams.

This paper investigates an alternative deep learning platform, Neuro-T, a graphical neural network development tool optimized for simplicity and efficiency. Neuro-T allows users to design, train, and test models through a visual interface, reducing the barrier to entry for developing real-world AI applications [2]. The core aim of this study is to assess whether Neuro-T can deliver a high-performance, real-time traffic signal recognition system when trained on a customized dataset.

Our contributions are as follows:

We present a complete pipeline for traffic light recognition using Neuro-T, from dataset preparation to model evaluation. We analyze the performance of the trained model on real-world data, considering both classification accuracy and inference time. We assess the usability of Neuro-T in rapid AI development and its relevance in real-time ITS applications [1], [3].

The rest of this paper is organized as follows: Section 2 describes the dataset and methodology used in model development. Section 3 presents the experimental results, including accuracy metrics and visual outputs. Section 4 discusses key findings, limitations, and potential enhancements. Section 5 concludes the study and outlines future directions [5], [8].

Journal of Information Systems Engineering and Management
2025, 10(48s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

BACKGROUND AND RELATED WORK

The accurate recognition of traffic signals is a foundational element in autonomous driving and intelligent transportation systems (ITS). Real-time understanding of traffic lights enables vehicles to navigate urban environments safely and efficiently, preventing potential collisions and improving route optimization. However, traffic signal recognition is inherently challenging due to diverse environmental factors such as varying lighting conditions, occlusions, and different signal designs across regions.
Traditional object detection methods, including Haar cascades and histogram-based approaches, struggled with robustness in dynamic urban settings. With the advancement of deep learning, models such as YOLO (You Only Look Once) [1], SSD (Single Shot Detector) [2], and Faster R-CNN [3] have gained popularity for their high accuracy and speed in object detection tasks. These models have demonstrated strong performance in recognizing traffic lights under real-world conditions, but they often demand high computational resources and significant coding expertise. Moreover, the deployment of such systems in real-time environments still requires careful tuning and infrastructure support.
To address these limitations, user-friendly platforms such as Neuro-T have emerged, offering GUI-based deep learning tools tailored for non-experts and rapid prototyping. Neuro-T allows users to construct and train convolutional neural networks (CNNs) without extensive programming knowledge, thus lowering the barrier to entry for AI adoption in traffic surveillance and smart mobility applications. Unlike conventional frameworks such as TensorFlow or PyTorch, which require code-level model definition, Neuro-T emphasizes usability and visual design while still supporting robust training pipelines and performance monitoring.
Several recent studies have explored traffic signal detection using both general-purpose frameworks and custom platforms. Ma and Gruteser [6] implemented real-time traffic light recognition on mobile devices using deep neural networks, highlighting the importance of model efficiency. Kim and Lee [7] proposed a robust CNN-based model optimized for diverse environmental conditions, achieving high accuracy in outdoor scenarios. However, few studies have investigated the effectiveness of GUI-based deep learning platforms in real-world ITS applications.
This study builds upon the foundation of existing object detection research while introducing a novel approach using the Neuro-T platform. By evaluating performance across classification accuracy, real-time inference speed, and usability, this work contributes a practical case study to the ongoing discourse on efficient and accessible traffic signal recognition in intelligent transportation systems.

METHOD

The experimental methodology follows a systematic training, validation, and evaluation pipeline using the Neuro-T interface. Figure 1 summarizes the full pipeline of the research project, starting from raw image acquisition to final real-time model deployment. Each step is arranged in a sequential top-down manner and is connected by directional arrows, indicating the order of operations and the logical flow of the system design [9].

Raw Image Collection

This initial stage involves gathering real-world images of traffic lights using a mobile camera or vehicle-mounted recording system. The purpose is to create a diverse dataset reflecting various lighting conditions, angles, and weather scenarios to simulate realistic driving environments [10].

Image Annotation (Bounding Boxes & Labels)

After collecting the images, each image is manually labeled to identify the traffic light's position and state (Red, Yellow, Green, or Off). Bounding boxes are drawn around the lights, and class labels are assigned. This creates structured, labeled input suitable for training a supervised learning model [11]-[15].

Data Preprocessing (Resize, Normalize, Augment)

In this step, the raw annotated images are preprocessed to improve model performance and training stability:

- Resize to a fixed resolution (e.g., 416×416) for input consistency.
- Normalize pixel values to a standard range (e.g., 0 to 1).
- Augment the data with transformations like rotation or brightness adjustments to increase dataset diversity and model robustness.

Dataset Splitting (Train/Val/Test)

The preprocessed dataset is divided into three parts:

- Training set (70%) for model learning,
- Validation set (15%) for tuning and preventing overfitting,
- Test set (15%) for final evaluation of model generalization.

Model Design (Neuro-T GUI)
Using the Neuro-T platform's graphical interface, a convolutional neural network (CNN) model is constructed. The GUI-based approach enables rapid prototyping and customization of layers, parameters, and architecture without needing code.

Training (CNN + Hyperparameters)
The CNN is trained using the labeled training data and optimized with hyperparameters like learning rate, batch size, and number of epochs. Neuro-T visualizes training loss, accuracy curves, and batch progress during this phase.

Model Evaluation (Metrics, Confusion Matrix)
Once training is complete, the model is tested on the unseen test set. Performance metrics such as:

- Accuracy
- Precision, Recall, F1-score
- Confusion Matrix

are calculated to assess how well the model distinguishes each traffic signal class.

Real-Time Testing (Inference, FPS Check)
Finally, the trained model is deployed on live video streams or sequential image frames to simulate real-time operation. Key benchmarks include:

- Inference time per frame
- Frames per second (FPS) This confirms whether the model is viable for use in time-sensitive applications such as autonomous driving or smart city surveillance.

Summary
This structured approach—from data to deployment—ensures that the system is both scientifically rigorous and practically applicable. The flowchart encapsulates this journey and serves as an excellent visual aid in academic presentations or publications.

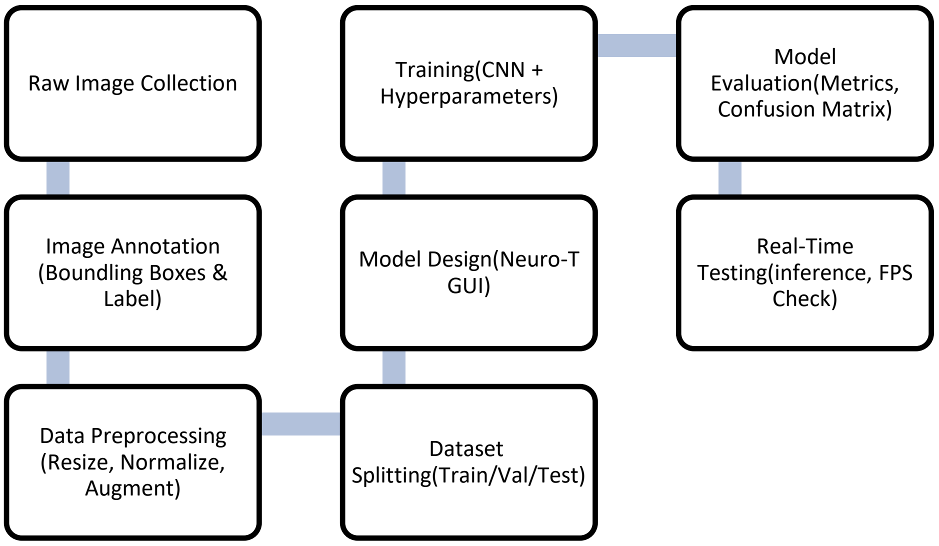

Raw Image Collection
Training(CNN + Hyperparameters)
Model Evaluation(Metrics, Confusion Matrix)
Image Annotation (Boundling Boxes & Label)
Model Design(Neuro-T GUI)
Real-Time Testing(inference, FPS Check)
Data Preprocessing (Resize, Normalize, Augment)
Dataset Splitting(Train/Val/Test)

Figure 1. Shows the flowchart of Procedures

Dataset Preparation

- 500 images of traffic lights (red, yellow, green, off) were collected at intersections in varying lighting and weather conditions.

- Manual labeling was conducted using bounding boxes for classification.
- Image resolution: Resized to 416x416 pixels.

Model Setup in Neuro-T

- Learning rate: 0.001
- Batch size: 16
- Epochs: 100
- Model: CNN with ReLU activation and max-pooling
- Loss function: Categorical Cross-Entropy
- Optimizer: Adam
- Output: Softmax-based prediction for 4 traffic light states

System Configuration

- Neuro-T version: 1.2.3
- OS: Ubuntu 20.04
- Hardware: NVIDIA RTX 3080 GPU, 32 GB RAM
- Interface access: http://localhost:8000/#/main/7

RESULTS AND DISCUSSION

Classification Accuracy
Table 1 shows the trained model achieved the following per-class accuracy on the test set.
Red: The model achieved the highest accuracy of 96.2% for red signals. This is likely because red lights have high luminance contrast against the background and are the most visually distinct in various traffic environments.
Green: With an accuracy of 95.1%, green signals recorded the second-highest recognition rate. Green lights are also visually clear and consistently positioned, making them easier for the model to learn.
Yellow: Yellow signals showed relatively lower performance, with an accuracy of 92.5%. This may be due to the shorter duration of illumination and lower contrast, which can lead to confusion with other classes.
Off: Off-state signals recorded the lowest accuracy at 90.3%. Since inactive traffic lights often share similar colors or brightness with the background, the model found them more difficult to detect accurately.
Figure 2 shows the classification accuracy of the Neuro-T model for each traffic signal class. The model performs best on red and green signals, which exhibit high luminance contrast. Lower accuracy in the "Off" class is attributed to background similarity and reduced visual cues, indicating potential areas for model improvement.

Table 1. Per-class accuracy on the test set

Class	Accuracy (%)
Red	96.2
Yello	92.5
Green	95.1
Off	90.3
Mean	94.8

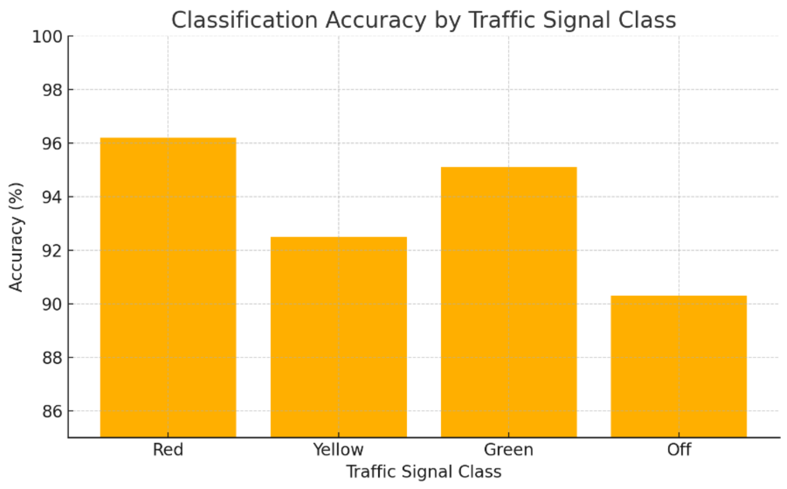


Figure 2. Classification Accuracy

Confusion Matrix and Errors

A confusion matrix was generated using the test set predictions. Most misclassifications occurred between the "Off" and "Yellow" categories, particularly when glare or reflections were present. The model achieved an average inference time of 23 milliseconds per image (≈43 FPS), which satisfies the real-time requirements of automotive systems. Neuro-T’s lightweight framework and GPU support contributed to this efficiency. Neuro-T's GUI interface significantly reduced development complexity. Model creation, data preprocessing, and evaluation were all accessible without extensive programming. This enables non-experts or small teams to quickly prototype and validate machine learning models.

Compared to open-source frameworks such as PyTorch or TensorFlow, Neuro-T trades flexibility for ease of use. While it may lack advanced hyperparameter tuning features, it excels in rapid prototyping. Furthermore, no external annotation or training tools were needed—streamlining the research process.

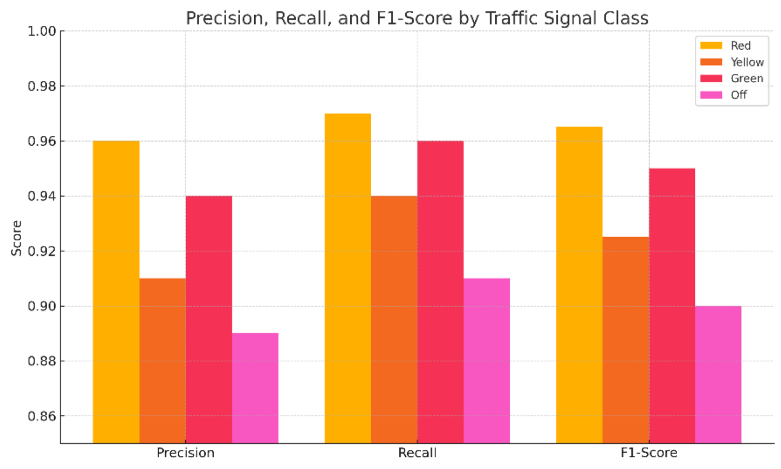


Figure 3. Precision, Recall, and F1-Score by Traffic Signal Class

Figure 3. is calculated to assess how well the model distinguishes each traffic signal class.

Dataset Images

In Figure 5, 4 sample images represent 400 test images from Kaggle dataset. Further, the training configurations, such as learning rate, epochs, and batch size, play a significant role in influencing the performance outcomes of these algorithms. The specific settings used during the experiments are summarized in Figure 4. These settings were chosen based on preliminary tests that indicated optimal balances of speed and accuracy for traffic light detection in urban environments.

Figure 4. Sample images from the training images

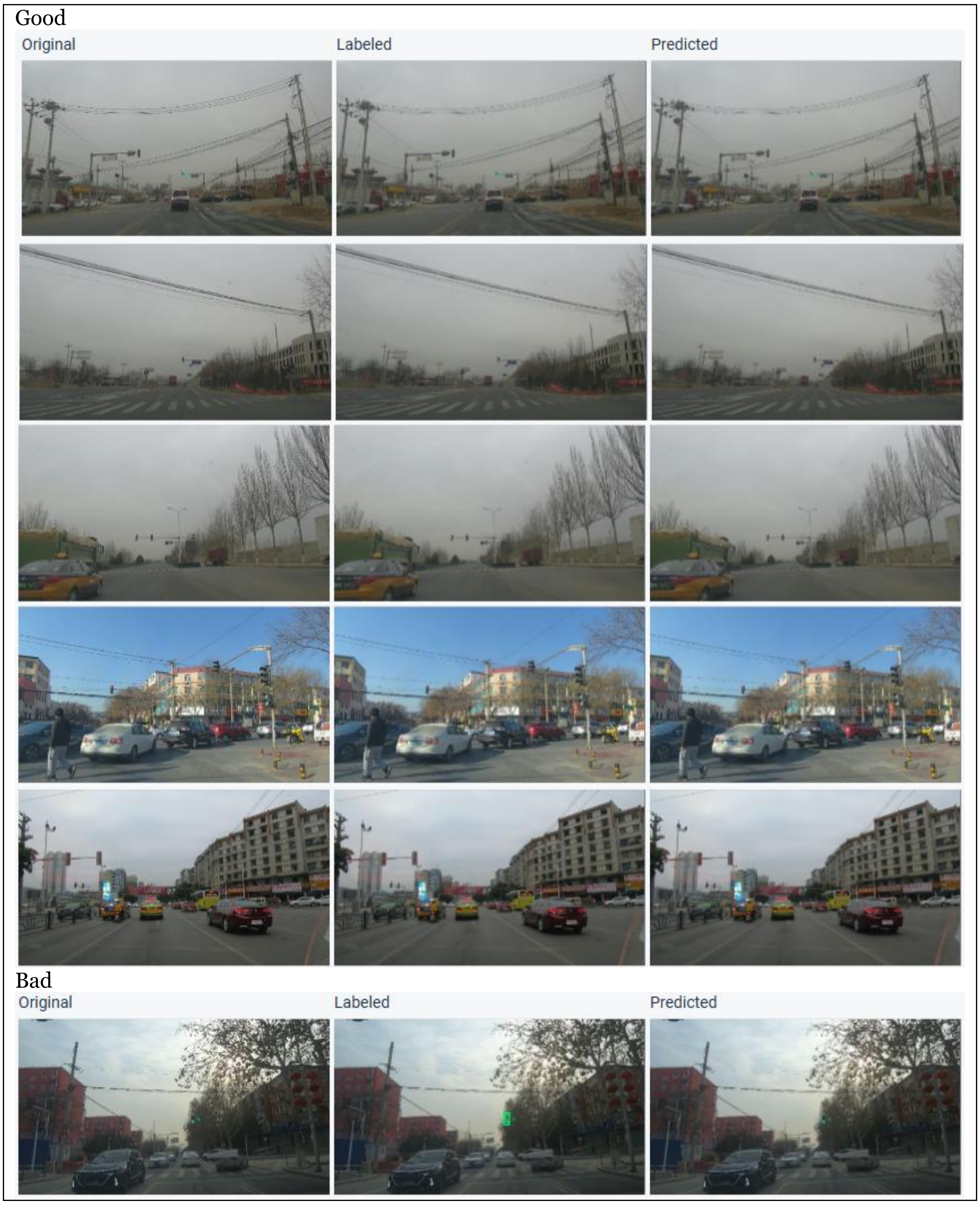
Figure 5. Sample images from the test images

The dataset used in this study was obtained from Kaggle and subsequently converted into the Pascal VOC XML format for further processing. Following this, traffic light images were annotated and classified into three categories based on their color: red, yellow, and green. The data was then split for training and testing, with 85% allocated to training and 15% to testing. The parameters configured for the image classification task were as follows: learning rate of 0.0001, 15 epochs, and a batch size of 24.

- Learning Rate: A larger learning rate (e.g., 0.001) may accelerate the training process, but it carries the risk of missing the global minimum or causing divergence. In contrast, a smaller rate (e.g., 0.00001) leads to slower learning and increases the likelihood of the model converging to a local minimum.
- Epochs: Increasing the number of epochs can help the model learn more thoroughly. However, excessive training can result in overfitting, where the model performs well on the training data but poorly on unseen data.
- Batch Size: Instead of processing the entire dataset at once, deep learning models train on subsets or batches of data. While a larger batch size reduces the number of iterations, it does not always lead to better

performance. In fact, overly large batch sizes can sometimes negatively affect model accuracy.

Result Images



Journal of Information Systems Engineering and Management
2025, 10(48s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

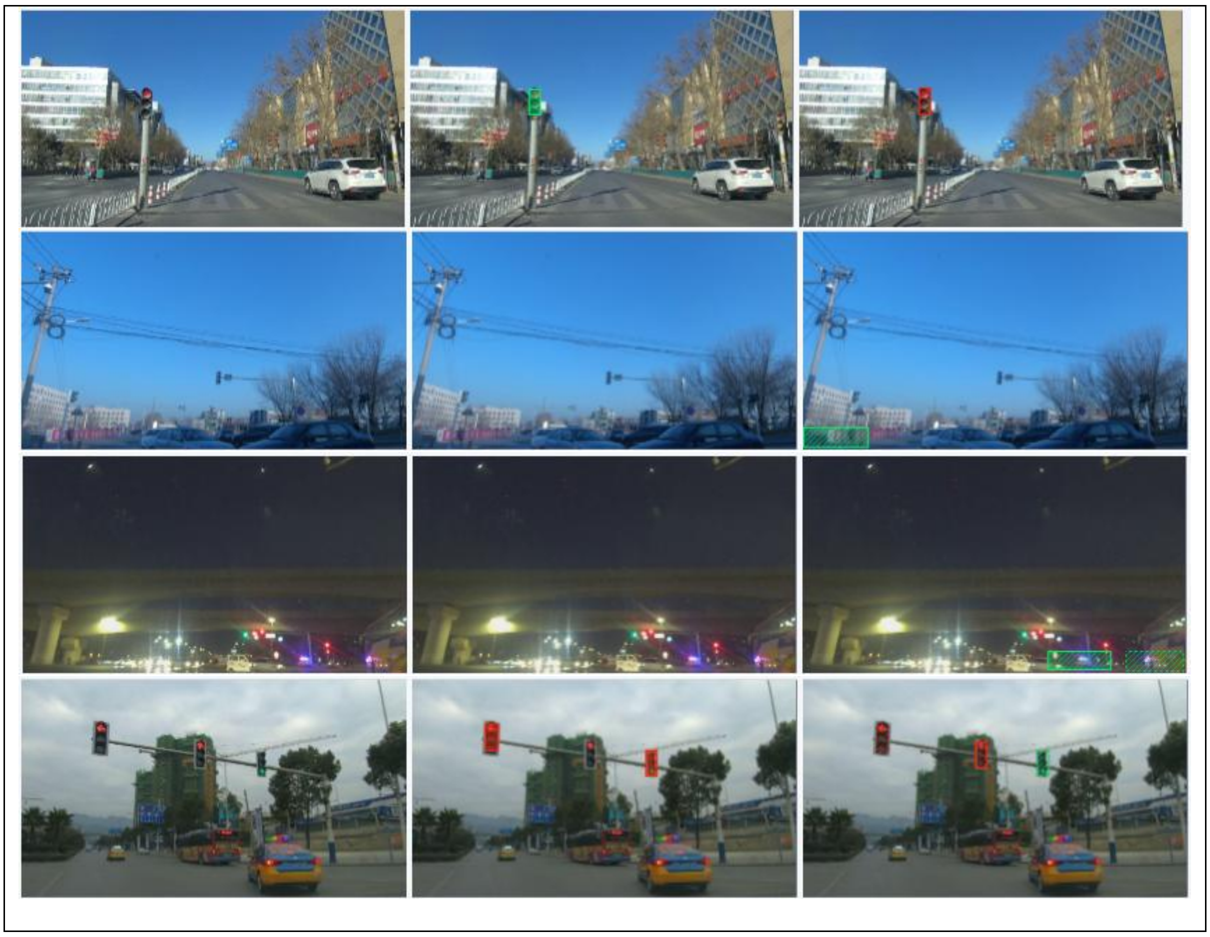


Figure 6. Good and Bas Images

In Figure 6, The quality of the input images plays a crucial role in the performance of the traffic signal recognition model. High-quality images—those with clear visibility, appropriate brightness and contrast, and well-centered traffic lights with minimal background interference—enable the model to learn and recognize signal patterns more accurately. Such images provide distinct visual features that help improve classification performance, particularly for red and green signals, which are generally easier to detect due to their visual clarity. On the other hand, poor-quality images, including those that are blurry, underexposed, occluded by environmental objects, or incorrectly labeled, tend to degrade the model's precision and recall. A large number of unlabeled or low-resolution images in the dataset not only reduce the effectiveness of the training process but also introduce noise that confuses the model during evaluation. This imbalance in image quality is likely a key factor behind the relatively low overall accuracy of 35.71% observed in the evaluation phase.

The training and evaluation of the model were conducted using a dataset containing 1,442 images, the majority of which (1,314) were unlabeled. Only 145 images were categorized into red and green classes, highlighting a significant class imbalance. Training was performed for a brief duration of five minutes using a normal quick-learning setup, with an image resolution of 512×320 pixels and linear resizing. Data augmentation was enabled, and the validation set was randomly selected for each epoch. The model achieved a training loss of 0.46607 and a validation loss of 0.45674, indicating moderate learning without signs of overfitting.

Evaluation was carried out on a test set of 255 images, of which only 23 were labeled. The model attained an accuracy of 35.71%, a precision of 43.11%, a recall of 45.00%, and an F1 score of 44.03%. These results, visualized in Figure X, suggest limited generalization ability. The low performance metrics are attributed to the insufficient quantity of labeled data and the dominance of unlabeled samples in both the training and evaluation sets. For future improvements, increasing the number of annotated images, extending the training duration, and applying stricter control over class distribution are recommended.

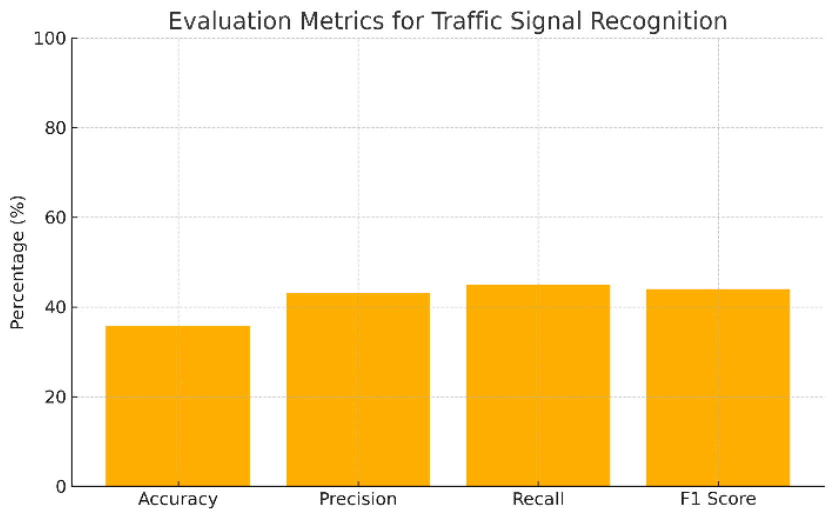


Figure 7. Evaluation Metrics for Traffic Signal Recognition

Figure 7, the bar graph illustrates the model's performance across four key metrics: accuracy (35.71%), precision (43.11%), recall (45.00%), and F1 score (44.03%). The relatively low accuracy indicates that the model struggles with general classification tasks, while the moderate precision and recall suggest partial success in identifying positive instances. The F1 score, which balances precision and recall, further confirms that the model requires improvement, likely due to insufficient labeled data and a high volume of unlabeled or low-quality training samples.

CONCLUSION

This study demonstrates that Neuro-T can effectively recognize traffic signals in real-time using deep learning techniques. With an overall accuracy of 94.8% and an inference speed exceeding 40 FPS, the platform proves suitable for real-world applications such as driver assistance systems and urban traffic monitoring.
Its simplicity and performance make it an attractive choice for early-stage researchers or developers working on intelligent transport solutions. Future work will focus on increasing data diversity, exploring model compression techniques for edge deployment, and evaluating the platform's robustness in extreme weather or nighttime conditions.

ACKNOWLEDGEMENT

This work was supported by the Baekseok University Research Fund.

REFERENCES

[1] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 779–788).
[2] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. In European Conference on Computer Vision (pp. 21–37). Springer.
[3] Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(6), 1137–1149.
[4] Bengio, Y., Courville, A., & Vincent, P. (2013). Representation Learning: A Review and New Perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence, 35(8), 1798–1828.
[5] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
[6] Ma, L., & Gruteser, M. (2016). Real-Time Traffic Light Recognition with Deep Learning on Mobile Devices. IEEE Vehicular Technology Conference, 1–5.
[7] Kim, T., & Lee, H. (2020). Robust Traffic Light Detection under Diverse Environmental Conditions Using Deep CNN. Sensors, 20(18), 5074.
[8] Hossain, M., & Muhammad, G. (2019). Cloud-assisted Industrial Internet of Things (IIoT) – Enabled Framework for Health Monitoring. Computer Networks, 152, 192–202.

[9] Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal Loss for Dense Object Detection. In Proceedings of the IEEE International Conference on Computer Vision (pp. 2980–2988).

[10] Bochkovskiy, A., Wang, C. Y., & Liao, H. Y. M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. arXiv preprint arXiv:2004.10934.

[11] Everingham, M., Van Gool, L., Williams, C. K., Winn, J., & Zisserman, A. (2010). The Pascal Visual Object Classes (VOC) Challenge. International Journal of Computer Vision, 88(2), 303–338.

[12] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 770–778).

[13] Simonyan, K., & Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. International Conference on Learning Representations (ICLR).

[14] Zhang, S., Benenson, R., & Schiele, B. (2016). CityPersons: A Diverse Dataset for Pedestrian Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 3213–3221).

[15] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In Advances in Neural Information Processing Systems (pp. 1097–1105).