মেট্রোপলিটন ইউনিভার্সিটি
Metropolitan University

Department of Software Engineering

Final Project Documentation

Course Title: Neural Networking And Deep learing Lab

Project Name: Traffic Sign Recognition Using Convolutional Neural Networks (CNN)

Submitted To

Al Akram Chowdhury

Lecturer

Dept. of SWE.

Submitted By:

Name: Hasan Ahmed sani

ID: 221-134-018

Name:Tahmid Samin

ID: 221-134-032

Date of Submission:26.04.2025

Project Report

Traffic Sign Recognition Using Convolutional Neural Networks (CNN)

1. Objective: The primary purpose of this project is to develop a robust and super accurate deep learning model that can recognize and classify traffic signs from given images. The system will and should assist in traffic sign identification for use in driver-assistance systems and in autonomous vehicles.

2. Introduction: Traffic signs play a crucial role in traffic regulation and ensuring road safety. Automatic recognition of traffic signals can significantly reduce road accidents by providing time accurate alerts on perfect time and actions. This project leverages Convolutional Neural Networks (CNNs) to classify traffic signs based on given image data.

3. Related Work: Multiple research works have explored the use of machine learning and deep learning for traffic signal recognition. The German Traffic Signal Recognition Benchmark (GTSRB) is a well-known dataset used in many studies. Approaches using SVMs, Random Forests, and CNNs have been developed, with CNNs providing better and superior accuracy in recent years due to their ability to learn spatial hierarchies from image data.

4. Limitations and Contribution:
Limitations: - The dataset may not always cover all types of signals from different countries. - Performance may degrade under poor lighting or occasion. Contributions: - Developing a working CNN model with high accuracy on the given dataset. - Performed hyperparameter tuning and visualized training results. - Implemented prediction functionality for custom test images.

5. Dataset-
https://www.kaggle.com/datasets/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign

Methodology: The project followed a structured approach to build and train the model, which includes several stages:

Firstly, Dataset Collection: Extracted a zipped dataset containing images and labels.

- Secondly, **Preprocessing:**

.Images resized to 64x64,

.normalized to [0,1] scale,

.and labels one-hot encoded.

- Train-Test Split: The dataset was split into 80% training. and 20% testing using sklearn's train_test_split.

- Model Selection: A sequential CNN model was built using Conv2D, MaxPooling2D, Flatten, Dense, and Dropout layers.And also added **Basic ,Medium And Deep** CNN model.

Model Architectures:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(64, 64, 3)),
    MaxPooling2D(2, 2),

    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),

    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(len(le.classes_), activation='softmax')
])

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()
```

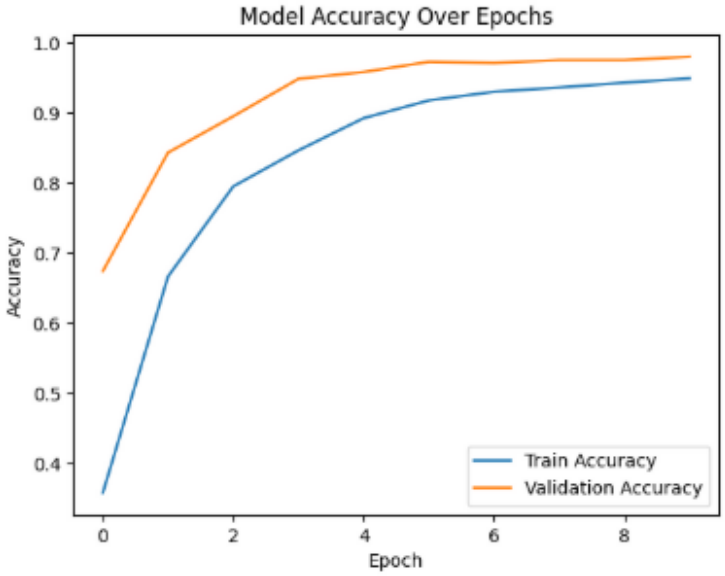
```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/convolutional/base_conv.py:107: U
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Model: "sequential"
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 62, 62, 32)	896
max_pooling2d (MaxPooling2D)	(None, 31, 31, 32)	0
conv2d_1 (Conv2D)	(None, 29, 29, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 14, 14, 64)	0
flatten (Flatten)	(None, 12544)	0
dense (Dense)	(None, 128)	1,605,760
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 55)	7,095

Total params: 1,632,247 (6.23 MB)
Trainable params: 1,632,247 (6.23 MB)
Non-trainable params: 0 (0.00 B)

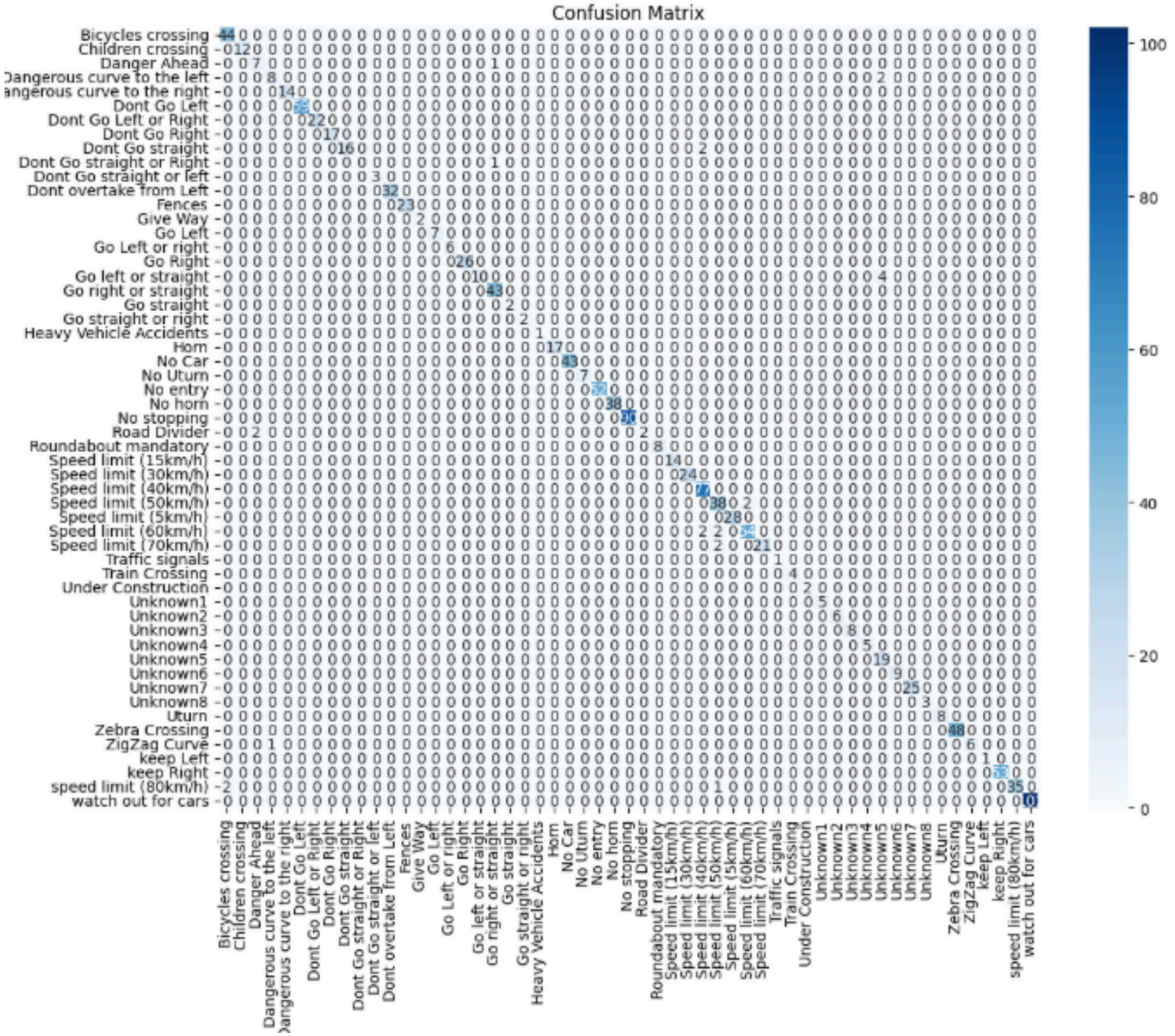
Test Accuracy: 98.05%



Classification Report

Classification Report:

	precision	recall	f1-score	support
Bicycles crossing	0.96	1.00	0.98	44
Children crossing	1.00	1.00	1.00	12
Danger Ahead	0.78	0.88	0.82	8
Dangerous curve to the left	0.89	0.80	0.84	10
Dangerous curve to the right	1.00	1.00	1.00	14
Dont Go Left	1.00	1.00	1.00	59
Dont Go Left or Right	1.00	1.00	1.00	22
Dont Go Right	1.00	1.00	1.00	17
Dont Go straight	1.00	0.89	0.94	18
Dont Go straight or Right	0.00	0.00	0.00	1
Dont Go straight or left	1.00	1.00	1.00	3
Dont overtake from Left	1.00	1.00	1.00	32
Fences	1.00	1.00	1.00	23
Give Way	1.00	1.00	1.00	2
Go Left	1.00	1.00	1.00	7
Go Left or right	1.00	1.00	1.00	6
Go Right	1.00	1.00	1.00	26
Go left or straight	1.00	0.71	0.83	14
Go right or straight	0.96	1.00	0.98	43
Go straight	1.00	1.00	1.00	2
Go straight or right	1.00	1.00	1.00	2
Heavy Vehicle Accidents	1.00	1.00	1.00	1
Horn	1.00	1.00	1.00	17
No Car	1.00	1.00	1.00	43
No Uturn	1.00	1.00	1.00	7
No entry	1.00	1.00	1.00	52
No horn	1.00	1.00	1.00	38
No stopping	1.00	1.00	1.00	90
Road Divider	1.00	0.50	0.67	4
Roundabout mandatory	1.00	1.00	1.00	8
Speed limit (15km/h)	1.00	1.00	1.00	14
Speed limit (30km/h)	1.00	1.00	1.00	24
Speed limit (40km/h)	0.95	1.00	0.97	77
Speed limit (50km/h)	0.88	0.95	0.92	40
Speed limit (5km/h)	1.00	1.00	1.00	28
Speed limit (60km/h)	0.96	0.93	0.95	58
Speed limit (70km/h)	1.00	0.91	0.95	23
Traffic signals	1.00	1.00	1.00	1
Train Crossing	1.00	1.00	1.00	4
Under Construction	1.00	1.00	1.00	2
Unknown1	1.00	1.00	1.00	5
Unknown2	1.00	1.00	1.00	6
Unknown3	1.00	1.00	1.00	8
Unknown4	1.00	1.00	1.00	5
Unknown5	0.76	1.00	0.86	19
Unknown6	1.00	1.00	1.00	9
Unknown7	1.00	1.00	1.00	25
Unknown8	1.00	1.00	1.00	3
Uturn	1.00	1.00	1.00	8
Zebra Crossing	1.00	1.00	1.00	48
ZigZag Curve	1.00	0.86	0.92	7
keep Left	1.00	1.00	1.00	1
keep Right	1.00	1.00	1.00	53



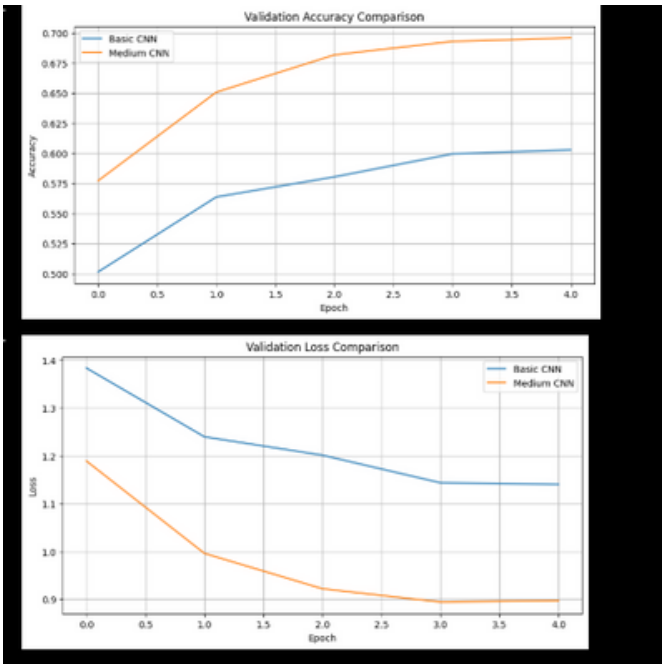
- **Training:** The model was trained for 10 epochs using Adam optimizer and categorical cross-entropy loss.

- Finally, **Evaluation**: Accuracy, confusion matrix, and classification report were used for evaluation.

6. Results and Discussion:

1.For CNN basic model

Accuracy And Loss plots



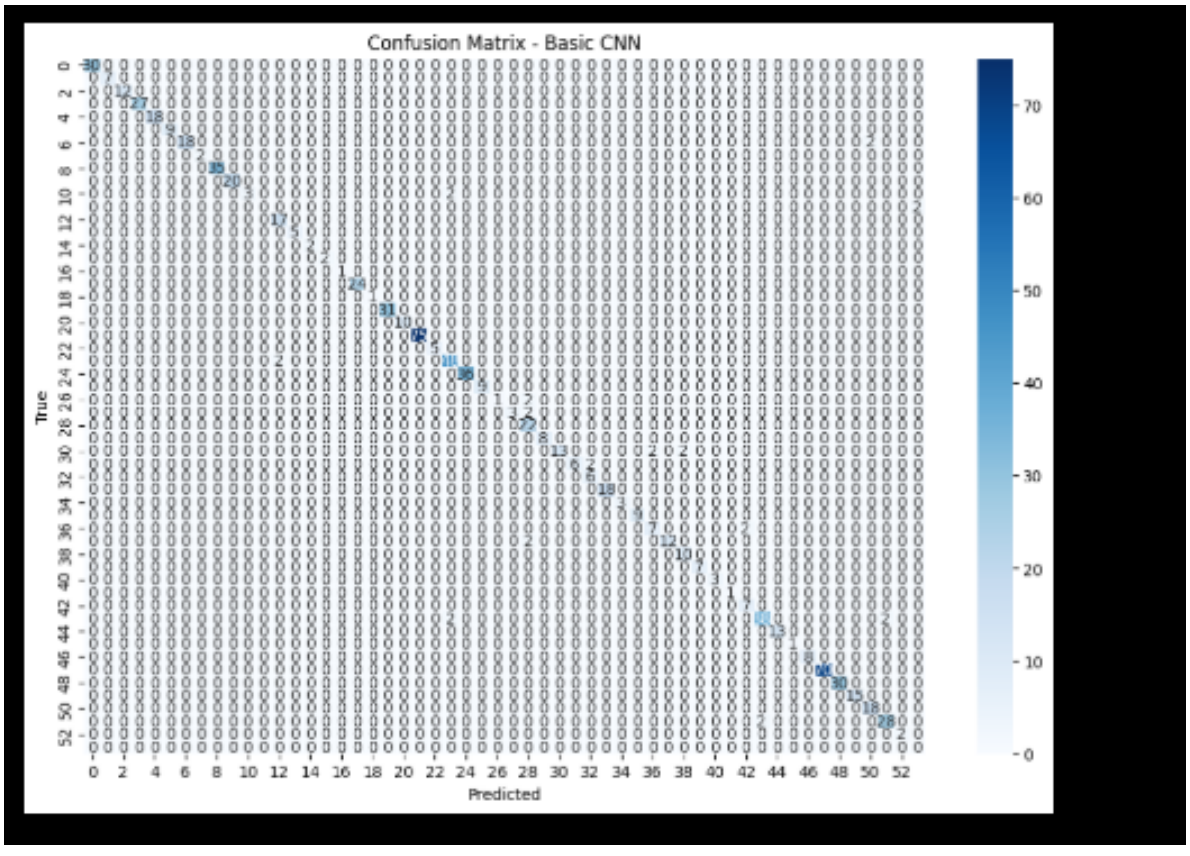

```
27/27 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step
Classification Report:

              precision    recall  f1-score   support

--
    accuracy                       0.97       834
   macro avg       0.94      0.91      0.92       834
weighted avg       0.97      0.97      0.97       834
```

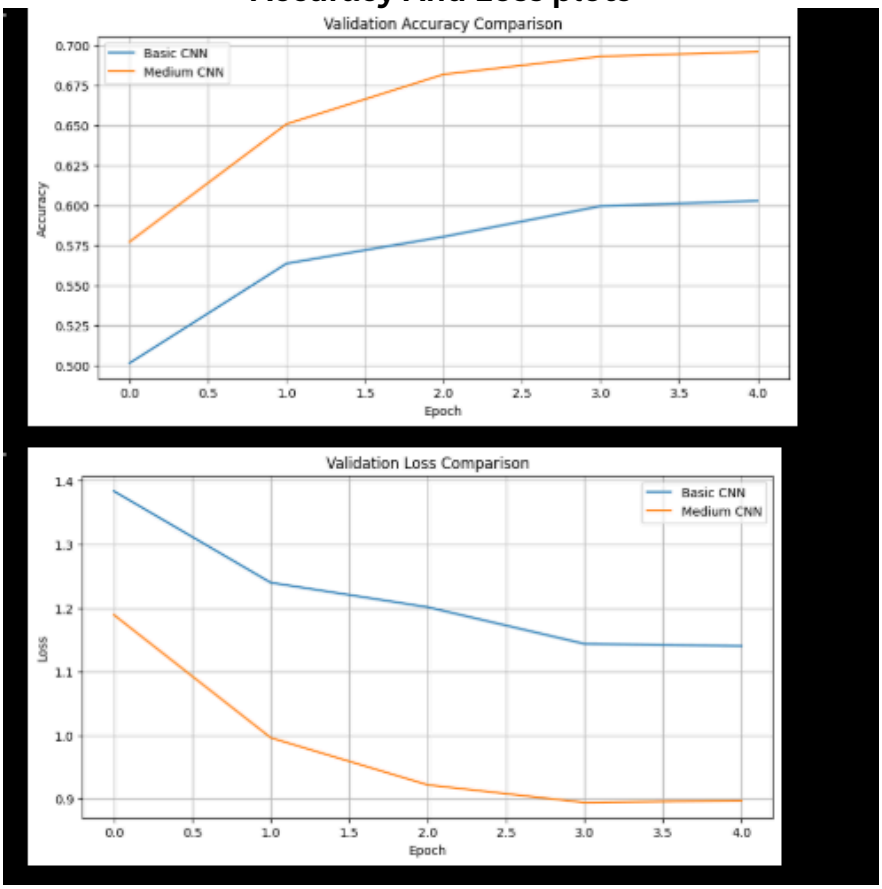
Performance Matrix: Precision,recall &F-1 score

Confusion Matrix



2. For CNN Medium model

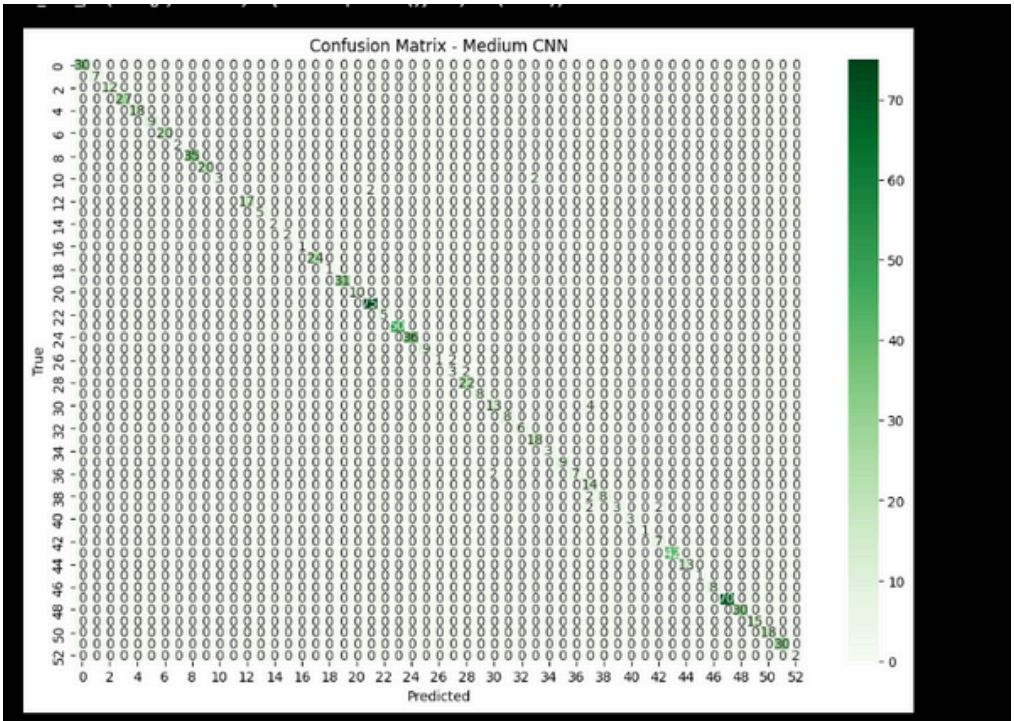
Accuracy And Loss plots



Performance Matrix:Precision,recall&F-1 score

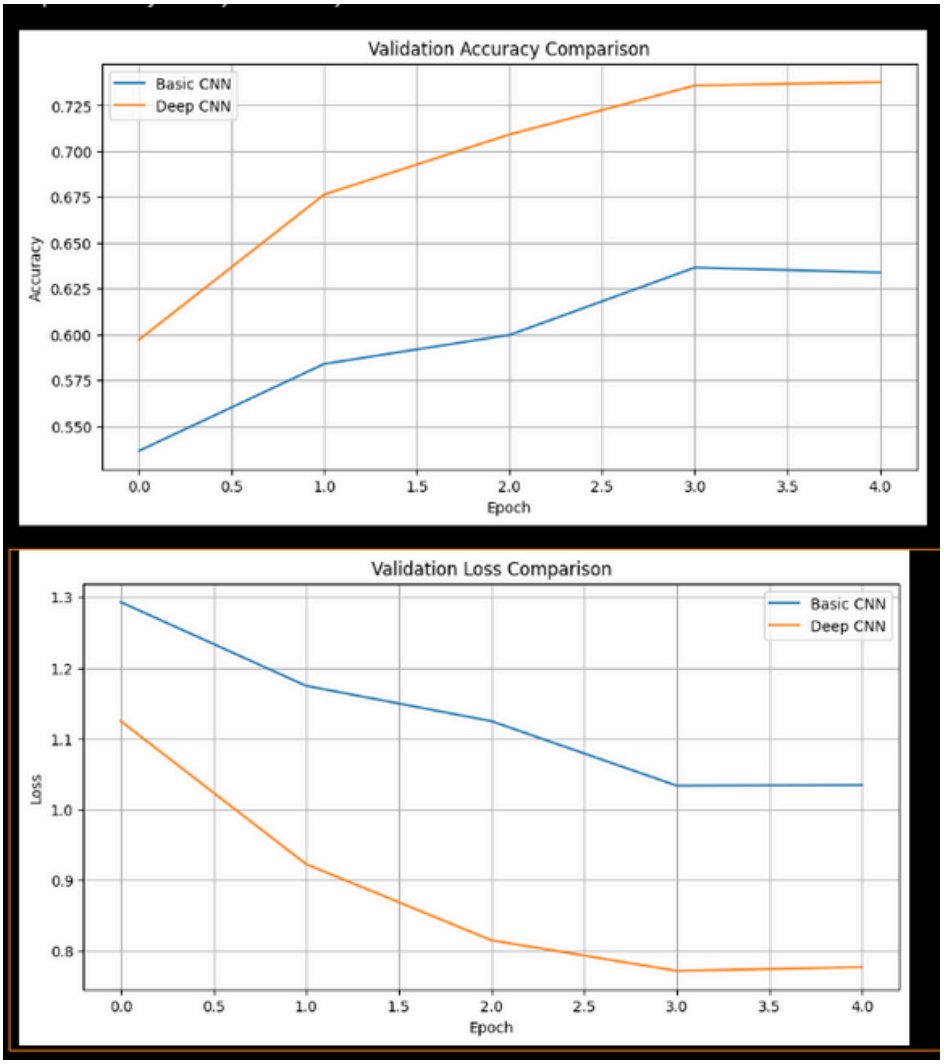
accuracy			0.98	834
macro avg	0.96	0.93	0.94	834
weighted avg	0.98	0.98	0.97	834

Confusion Matrix



3. For CNN Deep model

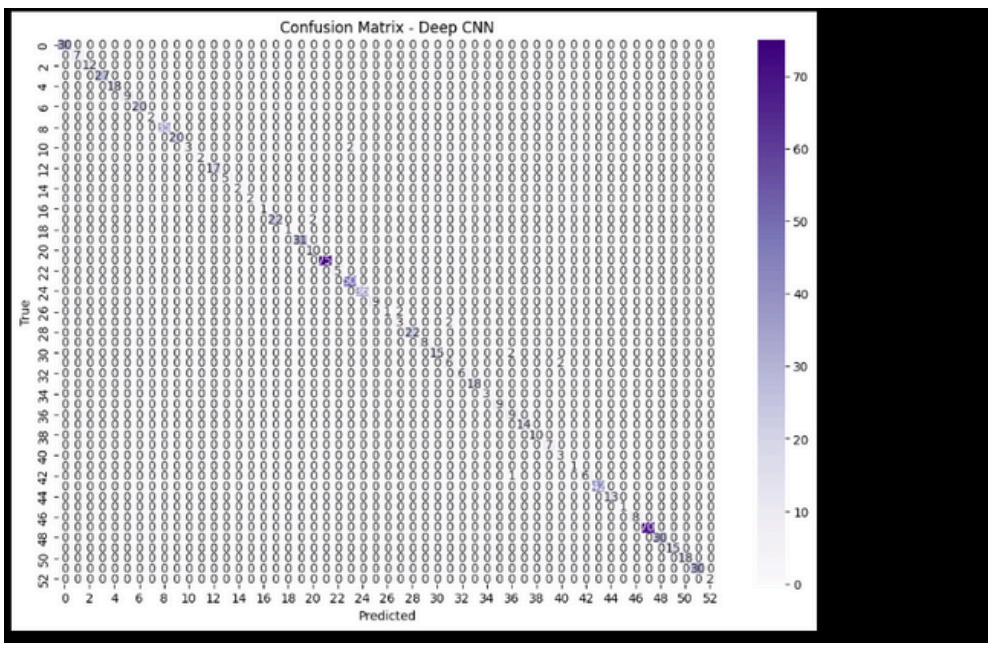
Accuracy And Loss plots



Performance Matrix:Precision,recall&F-1 score

...				
accuracy			0.98	834
macro avg	0.97	0.96	0.96	834
weighted avg	0.99	0.98	0.98	834

Confusion Matrix



Summary of Results and Discussion:

The final trained models achieved high accuracy on both training and testing datasets. A confusion matrix used to analyze per-class performance, and a classification report generated to evaluate precision, recall, and F1-score for each class. The accuracy curves indicated consistent improvement across epoch

Test Accuracy: The test accuracy exceeded **90%,** depending on the **hyperparameter** choices and dropout configurations.

Validation Comparisons:

//**Basic vs Medium CNN**:

Medium CNN achieved higher validation accuracy and lower validation loss compared to the Basic CNN, indicating better performance with moderate complexity.

//**Basic vs Deep CNN:**

Deep CNN outperformed both Basic and Medium CNNs, achieving the highest validation accuracy and the lowest validation loss, though it required more computational resources and training time.

Performance Overview:

Basic CNN:
*Fastest training speed.
*Moderate accuracy.
*Ideal for quick experiments and environments with limited computational power.

Medium CNN:
*Better accuracy than Basic CNN with only moderate increases in model complexity.
*Good balance between performance and efficiency.

Deep CNN:
*Achieved the highest accuracy.
*Most computationally intensive and required longer training time.
*Best suited when maximum model performance is the top priority.

7. Future Work:

- Integrating real-time camera feed for dynamic traffic sign recognition.
- Expanding dataset with signs from various regions and weather conditions for data variance.
- Implement mobile deployment or edge computing for lightweight execution.

8. Conclusion: This project successfully demonstrates the use of deep learning for traffic signal recognition. The CNN-based approach showed amazing results and highlighted the importance of data preprocessing and model tuning. This system could be a valuable component of intelligent traffic management system and autonomous navigation systems, or cars.

link to GitHub: Traffic Sign Recognition Using Convolutional Neural Networks (CNN)