Sr No.	Name	Enrollment no.
1	Ruchin Patel	220143

German Traffic Sign Recognition Benchmark (GTSRB)

Branch : CSE(AI-ML)

Batch :C3

Deep Learning

Submitted to:

Dr. Tejas Modi

Table of Contents

German Traffic Sign Recognition Benchmark (GTSRB)

1. Introduction to Traffic Sign Classification Using Artificial and Convolutional Neural Networks on the German Traffic Sign Recognition Benchmark(GTSRB).

- **Traffic Sign Classification in Intelligent Systems**
  Traffic sign recognition plays a pivotal role in the development of intelligent transportation systems and autonomous driving. The ability to automatically detect and classify road signs ensures that vehicles operate safely, comply with traffic rules, and respond appropriately to road conditions. Manual recognition is prone to human error, especially in challenging environments, making automated solutions increasingly essential.

- **Application of Neural Networks in Traffic Sign Recognition**
  Recent advancements in machine learning, particularly in **Artificial Neural Networks (ANNs)** and **Convolutional Neural Networks (CNNs)**, have significantly enhanced the capabilities of image classification systems. ANNs are designed to simulate the behavior of the human brain in recognizing patterns, but they often fall short when dealing with high-dimensional data like images. In contrast, CNNs are specifically engineered to process and learn from image data, making them highly effective in identifying patterns such as shapes and colors found in traffic signs.

- **German Traffic Sign Recognition Benchmark (GTSRB) Dataset Overview**
  The **German Traffic Sign Recognition Benchmark (GTSRB)** is a widely used dataset for traffic sign classification tasks. It contains over 39,000 training images and 12,000 test images, spanning 43 unique traffic sign categories. The dataset presents a variety of real-world conditions such as different lighting, angles, and occlusions, providing a robust benchmark for evaluating machine learning models in real-life driving scenarios.

- **Model Development and Comparative Approach**
  In this study, both ANN and CNN models are trained and tested on the GTSRB dataset. The CNN architecture leverages convolutional and pooling layers to extract relevant features from the images, followed by fully connected layers for classification. On the other hand, the ANN model is built using dense layers only, which flattens the image data before learning. Performance metrics such as accuracy, loss, and generalization to unseen data are used to assess and compare both models.

2. Aim & Scope –

Aim:

- The aim of this project is to design, implement, and compare the performance of an Artificial Neural Network (ANN) and a Convolutional Neural Network (CNN) for the task of traffic sign classification using the German Traffic Sign Recognition Benchmark (GTSRB) dataset.
- The objective is to evaluate which model provides better accuracy, generalization, and robustness in real-world image recognition scenarios, particularly for applications such as autonomous driving and traffic monitoring systems.

Scope:

1. To preprocess and prepare the GTSRB dataset for model training and evaluation.
2. To build an ANN model using fully connected layers and evaluate its performance on the classification task.
3. To build a CNN model using convolutional and pooling layers tailored for image data and compare its performance to the ANN.
4. To train both models using the same dataset split and hyperparameters (as applicable) for a fair comparison.
5. To demonstrate the practical implications of choosing one architecture over the other for image classification tasks, particularly in traffic sign recognition.

- **Objective**:
  The primary goal of this project is to design, implement, and evaluate neural network-based models capable of accurately classifying traffic signs. By comparing the performance of ANN and CNN, the study aims to determine the most effective approach for image-based traffic sign recognition. Ultimately, the findings will contribute to the broader development of safer and more reliable autonomous driving technologies.

- **Key Components**:

1. **Reinforcement Learning (RL)**: Used for optimal feature selection from sensor data, improving prediction accuracy.
2. **Artificial Neural Networks (ANNs)**: Employed for deep learning to forecast machine conditions based on historical sensor data.
3. **Gaussian Naive Bayes (GNB)**: Applied for classification of machine states to determine potential failures.
4. **Practical Applications:**

   Real-world deployment for:
   - Autonomous driving systems
   - Traffic monitoring
   - Driver assistance alerts
   - Smart transport infrastructure

3. Dataset and Experimental setup

➢ Dataset

The dataset used in this project is the German Traffic Sign Recognition Benchmark (GTSRB), which contains real-world images of German traffic signs captured under various environmental and lighting conditions. It is a multi-class classification dataset suitable for training and evaluating computer vision models.

Each image corresponds to one of 43 traffic sign categories, such as speed limits, stop signs, and yield signs. The images vary in resolution, color distribution, and background complexity, making this dataset ideal for benchmarking deep learning models.

➢ Key Features

- **Image Data:** RGB images representing different traffic signs.
- **Label:** Integer values (0 to 42) indicating the traffic sign class.
- **Class Names:** Includes signs like "Speed limit (30km/h)", "Stop", "No entry", "Turn right ahead", etc.
- **Variations:** Differences in lighting, angles, weather, and partial occlusion to simulate real-world conditions.

➢ Experimental Setup

1. Data Preprocessing

- Resize: All images resized to 32x32 pixels.
- Normalization: Pixel values scaled between 0 and 1.
- One-hot Encoding: Labels converted to categorical format for training.
- Train-Test Split: Dataset divided into 80% training and 20% testing using stratified sampling.

2. Deep Learning with ANN

- Input Layer: Flattened input layer (32x32x3 = 3072 features).
- Hidden Layers: Dense layers with ReLU activation.
- Output Layer: Dense layer with 43 nodes using Softmax activation for multi-class classification.
- Loss Function: Categorical Crossentropy.
- Optimizer: Adam.
- Epochs: 30
- Batch Size: 128

3. Deep Learning with CNN

- Input: 32x32 RGB image tensor.
- Convolutional Layers: Multiple Conv2D layers with ReLU activation.
- Pooling: MaxPooling2D layers for dimensionality reduction.
- Dropout: Used to prevent overfitting.
- Fully Connected Layers: Followed by output layer with Softmax activation.
- Loss Function: Categorical Crossentropy.
- Optimizer: Adam.
- Epochs: 30
- Batch Size: 128

- **IDE Setup**
  - IDE Choice: Jupyter Notebook (preferred), VS Code, or PyCharm.
  - Python Version: Python 3.8+ recommended.
  - Environment Setup: Use venv or conda for virtual environment management.
  - Install Libraries: Required libraries include tensorflow, keras, numpy, matplotlib, sklearn, etc.
  - Project Structure:
    - /data – Contains image files and labels.
    - /models – Saved models (H5 format).
    - /notebooks – Jupyter notebooks for exploration and training.
    - /scripts – Python scripts for preprocessing, training, and evaluation.
  - Requirements File: requirements.txt for managing dependencies.
  - Version Control: Optional use of Git for tracking changes and collaboration.
  - Load Dataset: Initial step involves reading and visualizing sample data before model training.

4. Methodology

➢ Artificial Neural Network (ANN)

The Artificial Neural Network (ANN) model is built using fully connected layers for multi-class classification. Although ANN is not inherently image-focused, it is included in this study to analyze its effectiveness in handling high-dimensional image data.

1. Input Layer:

- Each RGB image is resized to 32×32×3 and then flattened into a 1D vector of 3072 features.
- Input data is normalized to a range between 0 and 1 for faster convergence.

2. Hidden Layers:

- The ANN includes one or more hidden layers, each with ReLU activation functions to introduce non-linearity.
- Dropout layers are added between hidden layers to prevent overfitting by randomly deactivating neurons during training.

ReLU (Rectified Linear Unit):

$$f(x) = \max(0, x)$$

3. Output Layer:

- The output layer consists of 43 neurons, each representing a traffic sign class.
- A Softmax activation function is applied to generate a probability distribution across all classes.

Softmax:

$$\hat{y}_i = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}}$$

Where:

- z_i is the input to the softmax function for class i
- C is the total number of classes

4. Training:

- The model is trained using the Adam optimizer and categorical cross-entropy loss function.
- Training and validation sets are split from the dataset (e.g., 80:20 ratio).
- The model is trained over multiple epochs with batch processing.

Adam Optimizer – Combines Momentum and RMSprop:

Update equations:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla L(\theta_t)$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2)[\nabla L(\theta_t)]^2$$
$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$
$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$
$$\theta_{t+1} = \theta_t - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

Where:

- θ: Parameters (weights)
- α: Learning rate
- m, v: Moving averages of gradients and squared gradients
- β_1, β_2: Decay rates (usually 0.9 and 0.999)

Loss Function:
Binary Cross-Entropy (used for binary classification):

$$L = -\left[y \log(\hat{y}) + (1-y)\log(1-\hat{y})\right]$$

5. Evaluation:

- The ANN's performance is evaluated using accuracy, precision, recall, F1 score, and a confusion matrix.
- These metrics help identify strengths and weaknesses in recognizing specific traffic sign categories.

➢ Pseudo Code

Start

#Step 1. Load dataset
#Step 2. Preprocess data:
  - Normalize/Standardize features
  - Encode labels if needed
  - Split data into training and testing sets

#Step 3. Feature selection using Reinforcement Learning (FSRLearning)

#Step 4. Build ANN model:
  - Input Layer (nodes = number of selected features)
  - Hidden Layer(s) with ReLU activation
  - Output Layer with:
    - Sigmoid activation for binary classification

- Softmax activation for multi-class classification

#Step 5. Compile the model:
- Optimizer: Adam
- Loss:
- Binary Cross-Entropy for binary classification
- Categorical Cross-Entropy for multi-class classification
- Metrics: Accuracy, Precision, Recall, F1 Score

#Step 6. Train the model using training data

#Step 7. Evaluate the model on testing data

#Step 8. Output results:
- Accuracy
- Precision
- Recall
- F1 Score
- Confusion Matrix

End

- **Convolutional Neural Network (CNN)**

The Convolutional Neural Network (CNN) is better suited for image classification due to its ability to capture spatial and hierarchical patterns. It is expected to outperform ANN in this context.

1. Input Layer:

- The CNN takes in 32×32×3 RGB images as input.
- Images are preprocessed with normalization, similar to the ANN pipeline.

2. Convolutional & Pooling Layers:

- One or more convolutional layers with ReLU activation extract spatial features like edges, shapes, and textures.
- These layers are followed by MaxPooling layers to downsample feature maps and reduce dimensionality.

ReLU (Rectified Linear Unit):

$$f(x) = \max(0, x)$$

MaxPooling Layer:

Given a 2D window over feature map X:

$$Y_{i,j} = \max\{X_{m,n} \in \text{window}\}$$

3. Flatten & Dense Layers:

- After feature extraction, the resulting feature maps are flattened and passed through one or more dense (fully connected) layers.
- Dropout is used to reduce overfitting.

Flatten Layer:

Transforms a multi-dimensional tensor into a 1D vector:

$$\text{Flatten}(X) \rightarrow \text{vector } x \in \mathbb{R}^n, \text{ where } n = \prod_{i=1}^{d} x_i$$

(for a tensor of shape $x_1 \times x_2 \times \ldots \times x_d$)

4. Output Layer:

- The final dense layer consists of 43 neurons with a Softmax activation function for multi-class output.

5. Training:

- Similar to ANN, CNN is trained using the Adam optimizer and categorical cross-entropy loss.
- Training is performed over several epochs, and the model is validated on unseen test data.

Adam Optimizer – Combines Momentum and RMSprop:

Update equations:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\nabla L(\theta_t)$$
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)[\nabla L(\theta_t)]^2$$
$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$
$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$
$$\theta_{t+1} = \theta_t - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

Where:

- θ: Parameters (weights)
- α: Learning rate
- m, v: Moving averages of gradients and squared gradients
- β_1, β_2: Decay rates (usually 0.9 and 0.999)

Loss Function:
Categorical Cross-Entropy (used for multi-class classification):

$$L = -\sum_{i=1}^{C} y_i \log(\hat{y}_i)$$

Where:

- y_i: True label (one-hot encoded)
- $\hat{y}_i$: Predicted probability for class i
- C: Number of classes

6. Evaluation:

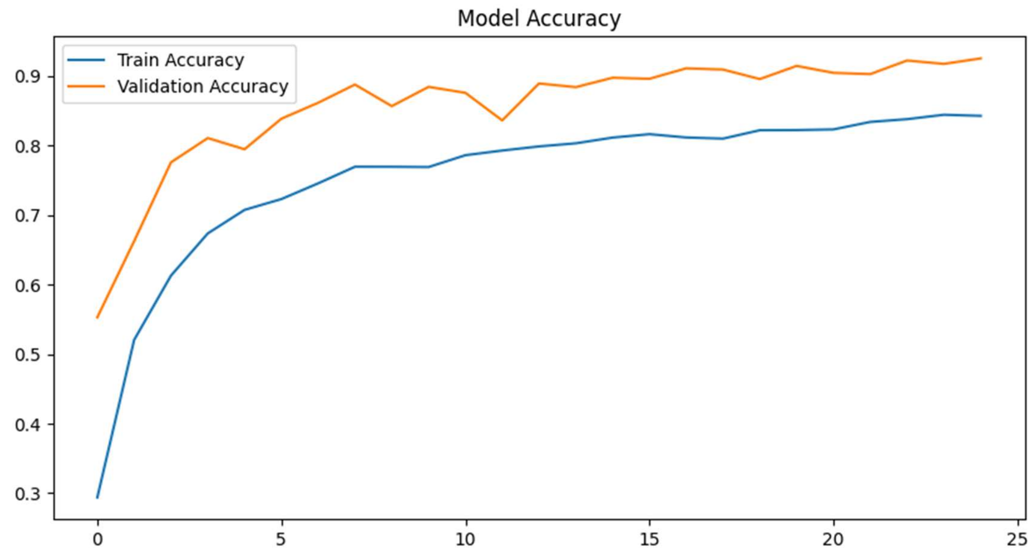
- CNN is evaluated using the same set of metrics: accuracy, precision, recall, F1 score, and confusion matrix.
- The model's capability to generalize across all 43 traffic sign classes is thoroughly assessed.

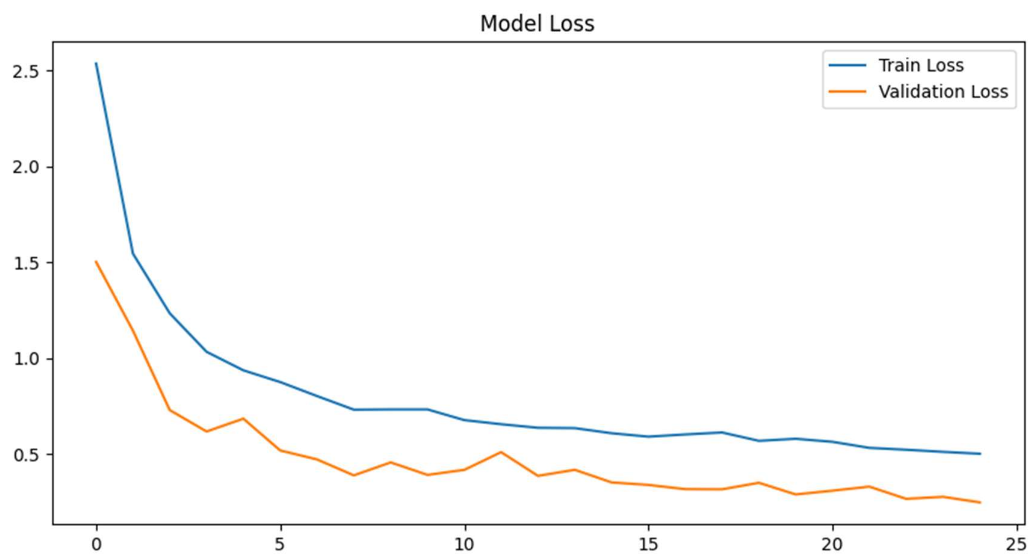
➢ Pseudo Code

```
Start

#Step 1. Load image dataset
#Step 2. Preprocess data:
  - Resize and normalize image pixels
  - Encode labels
  - Split data into training and testing sets

#Step 3. Build CNN model:
  - Input Layer: Image input shape (e.g., 32x32x3)
  - Convolution Layer(s) with ReLU activation
  - MaxPooling Layer(s) to downsample
  - Flatten Layer to convert 2D to 1D
  - Dense Hidden Layer(s) with ReLU
  - Output Layer:
      - Sigmoid for binary classification
      - Softmax for multi-class classification

#Step 4. Compile the model:
  - Optimizer: Adam
  - Loss:
      - Binary Cross-Entropy (binary)
      - Categorical Cross-Entropy (multi-class)
  - Metrics: Accuracy, Precision, Recall, F1 Score

#Step 5. Train the model on training images

#Step 6. Evaluate the model on test images

#Step 7. Output results:
  - Accuracy
  - Precision
  - Recall
  - F1 Score
  - Confusion Matrix

End
```

5. Results

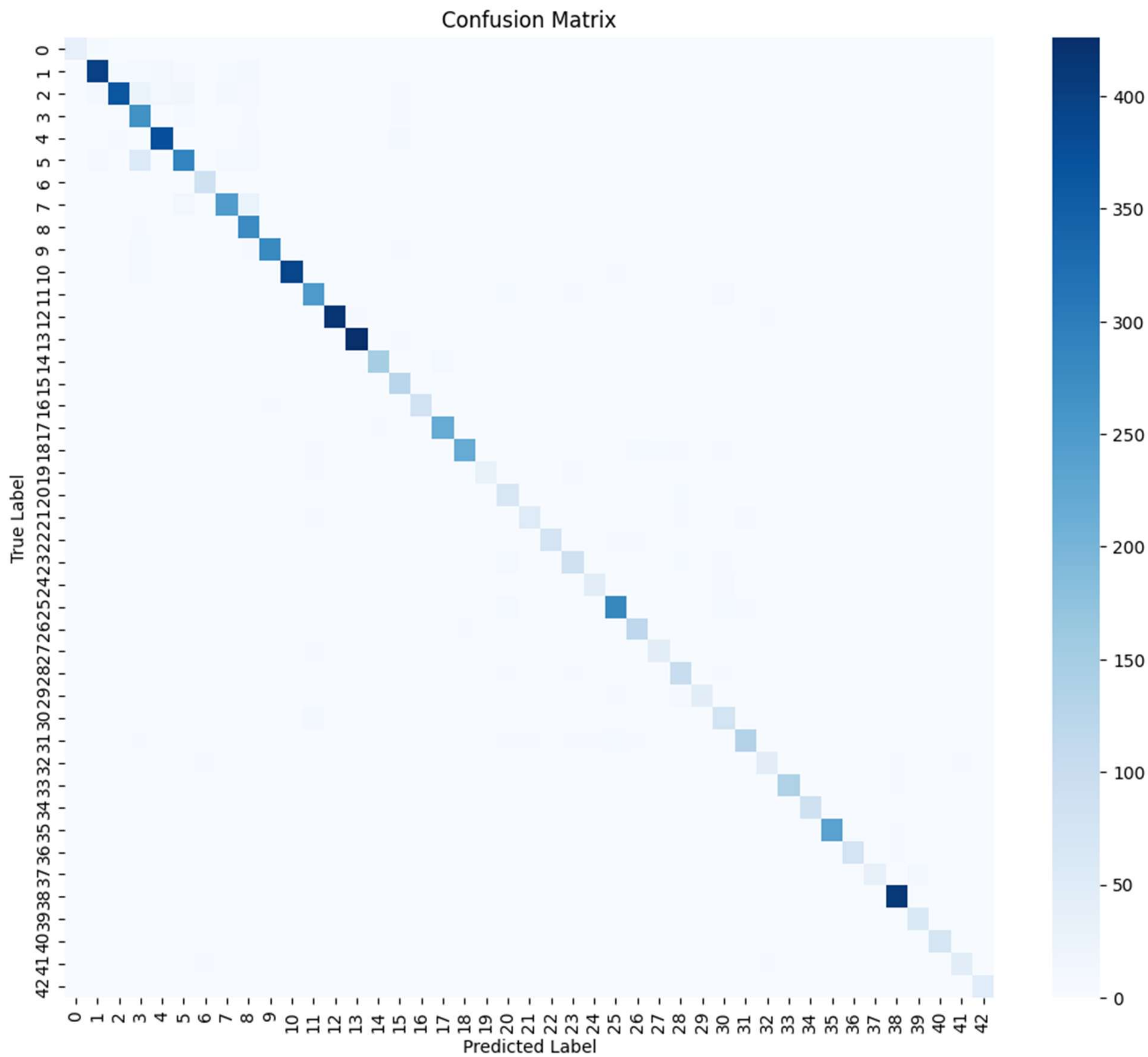
- **Artificial Neural Network (ANN)**





Percentage of model Accuracy: 92.49 %

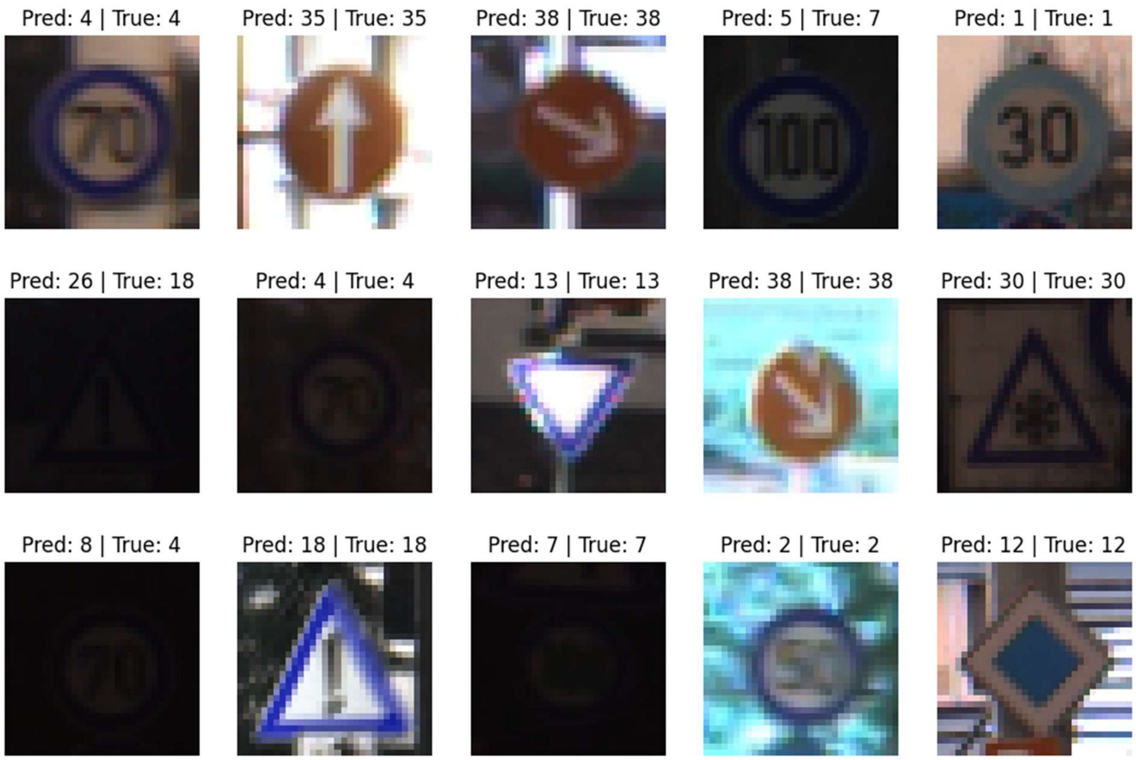
German Traffic Sign Recognition Benchmark (GTSRB)



Property	Value
Accuracy	0.9249
Precision	0.9309
Recall	0.9249
F1 Score	0.9255
Loss (Log)	0.2493

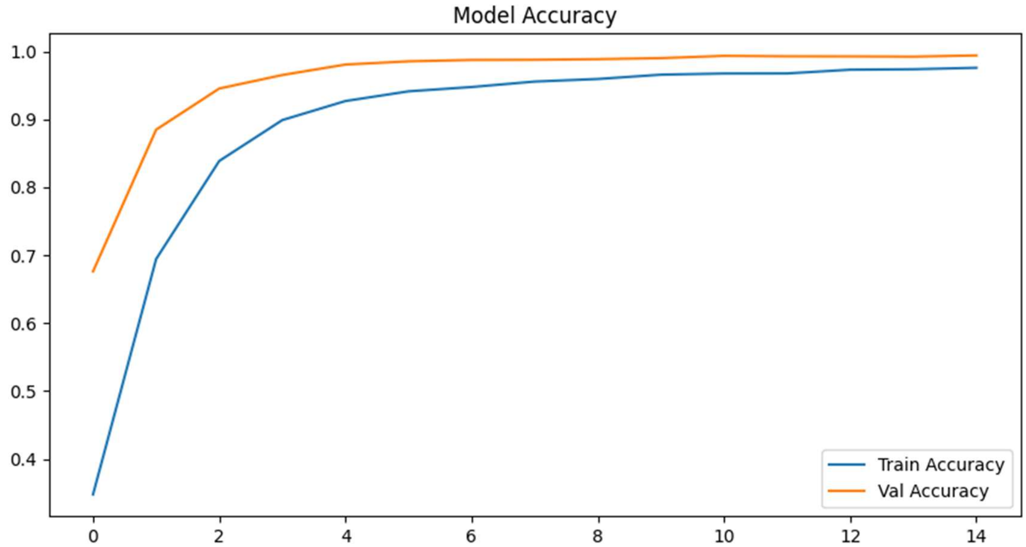
German Traffic Sign Recognition Benchmark (GTSRB)

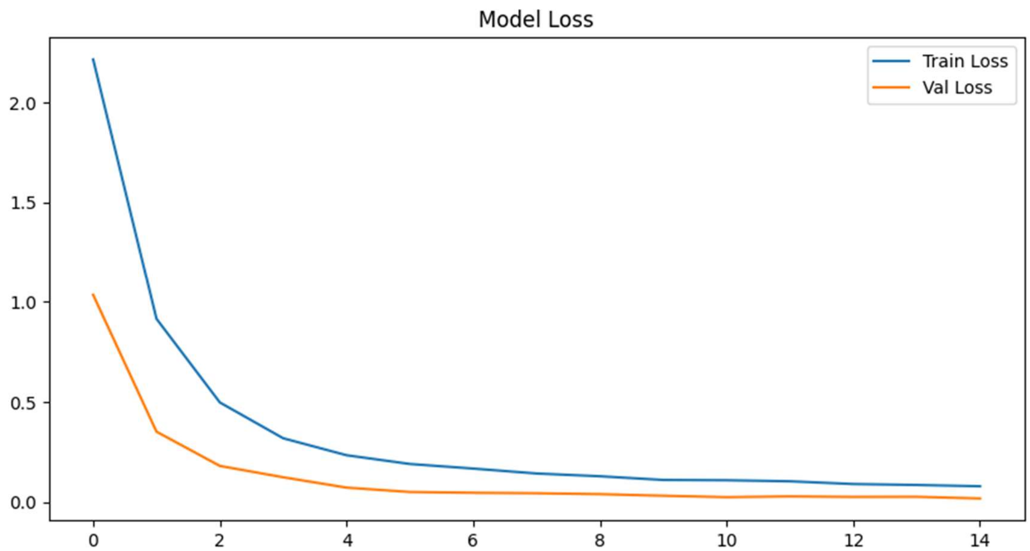
Actual and predicted value:



German Traffic Sign Recognition Benchmark (GTSRB)

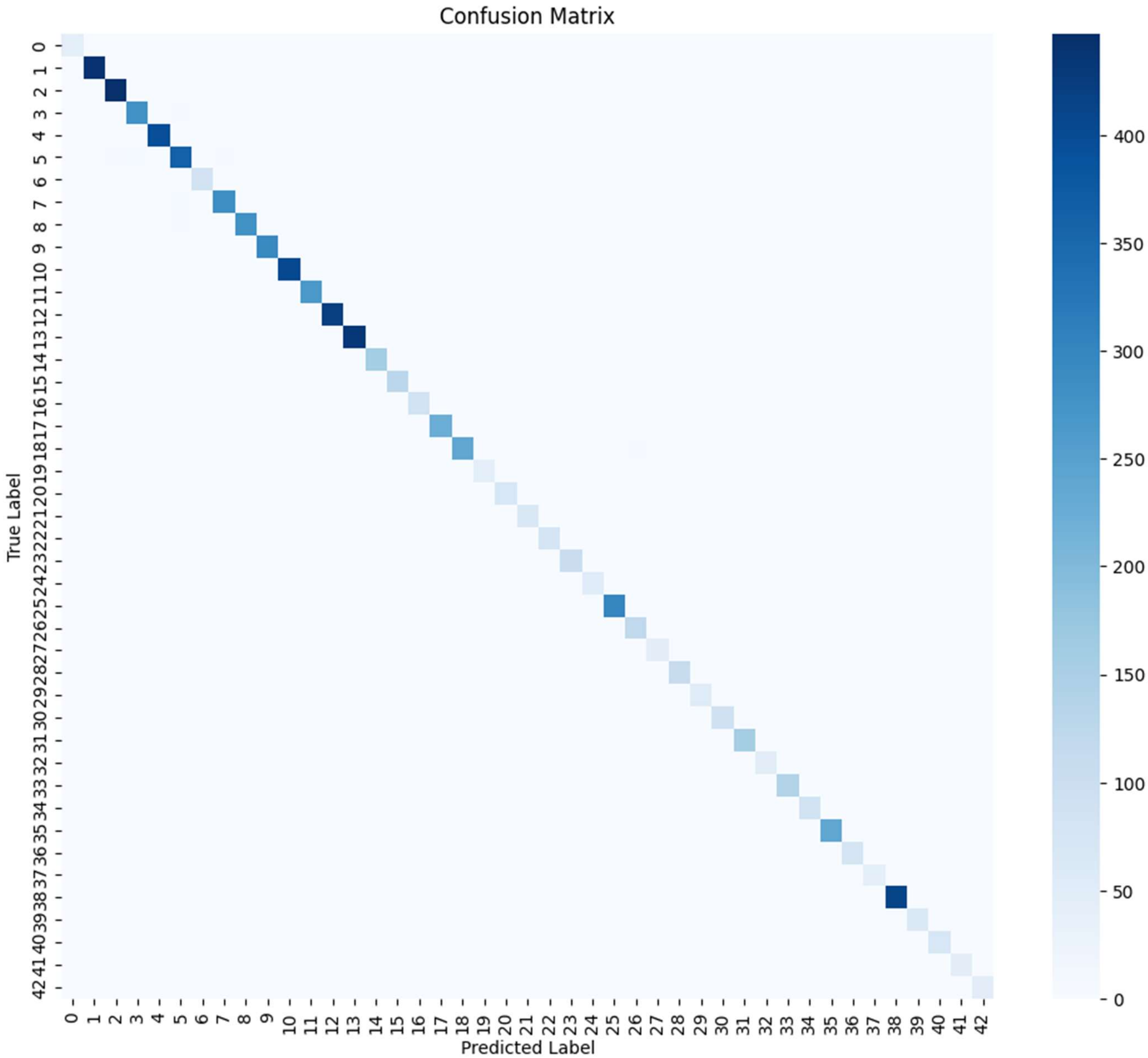
➢ **Convolutional Neural Network (CNN)**





Percentage of model Accuracy: 99.41 %

German Traffic Sign Recognition Benchmark (GTSRB)



Property	Value
Accuracy	0. 9941
Precision	0.9942
Recall	0.9941
F1 Score	0.9941
Loss (Log)	0.0180

German Traffic Sign Recognition Benchmark (GTSRB)

Actual and predicted value: