Final Project - Traffic Sign Classification using CNNs

Name: Sushruth Danivasa Sridhar

1. **Introduction**

   Problem Statement:
   The classification of traffic signs is a critical task for autonomous driving systems and advanced driver assistance systems (ADAS). Recognizing traffic signs accurately helps ensure safety and compliance on the road**.**

   Approach:
   This project addresses the problem using Convolutional Neural Networks (CNNs), a deep learning approach highly effective for image classification tasks. Three models were implemented:
   Model 1: Pre-trained MobileNetV2 as a feature extractor.
   Model 2: A custom CNN architecture with data augmentation and dropout layers.
   Model 3: An advanced CNN architecture with additional convolutional layers and optimizations.
   The models were trained and evaluated on the German Traffic Sign Recognition Benchmark (GTSRB) dataset.

   Significance:
   Accurate traffic sign classification improves road safety, supports automated navigation systems, and demonstrates deep learning's capability to solve real-world problems. This project also highlights how various CNN architectures influence accuracy and efficiency.

2. **Methodology**

   Dataset: The GTSRB dataset, consisting of images of 43 traffic sign classes.
   https://www.kaggle.com/datasets/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign

   Data Distribution:
   Training set: 70% of the images.
   Validation set: 30% of the images.
   Test set: Separate test images with labels provided in Test.csv.

   Image Preprocessing:
   Resized all images to 30x30x3 dimensions.
   Normalized pixel values to the range [0, 1] for consistent input into models.

   Algorithms and Models:
   Model 1: MobileNetV2-based Architecture

- A pre-trained **MobileNetV2** model was used as a feature extractor, followed by custom dense layers:
  - GlobalAveragePooling2D
  - Dense(512, ReLU), Batch Normalization, Dropout(0.5).

Model 2: Custom CNN

- Architecture:
  - Multiple convolutional layers: Filters of size 16 → 32 → 64 → 128.
  - MaxPooling layers after every 2 convolutional layers.
  - Batch Normalization and Dropout (0.3 and 0.5 rates) to reduce overfitting.
  - Fully connected layer (Dense 512) with Batch Normalization and Dropout.

Model 3: Advanced Custom CNN

- Deeper CNN with:
  - Convolutional layers: Filters 32 → 64 → 128 → 256.
  - Increased Dropout regularization (0.5 rate).
  - Optimized learning rate (0.0001).

Optimization

- **Optimizer**: Adam
- **Loss Function**: Categorical Crossentropy
- **Epochs**: 20
- **Data Augmentation**: Applied random rotation, zoom, shift, and shear transformations using ImageDataGenerator.

Workflow

1. **Data Preprocessing**: Image resizing, normalization, and augmentation.
2. **Model Training**: Trained three models on the preprocessed dataset.
3. **Evaluation**: Used accuracy, precision, recall, F1-score, and ROC-AUC to evaluate performance.
4. **Visualization**:
    - Training/validation accuracy and loss plots.
    - Confusion matrices.
    - ROC-AUC curves for performance comparison.

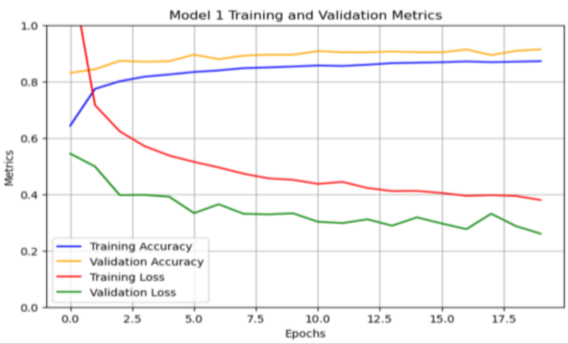
3. Results and Evaluation

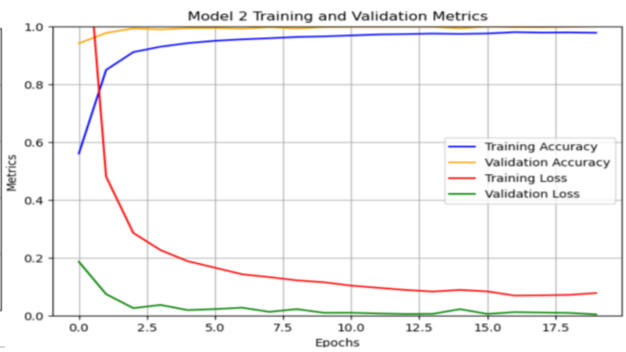
Key Observations:

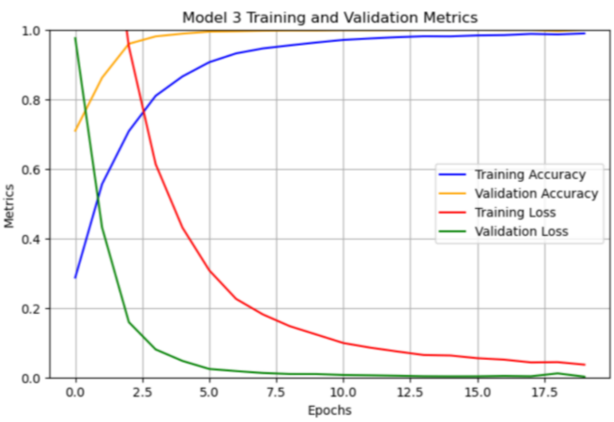
- Model 1 showed the lowest accuracy due to limited generalization capacity.
- Model 2 and Model 3 achieved high accuracy, with Model 3 slightly outperforming Model 2 in terms of loss and validation metrics.
- Both Model 2 and Model 3 demonstrated near-perfect AUC scores, indicating excellent performance.
- The best model is Model 3

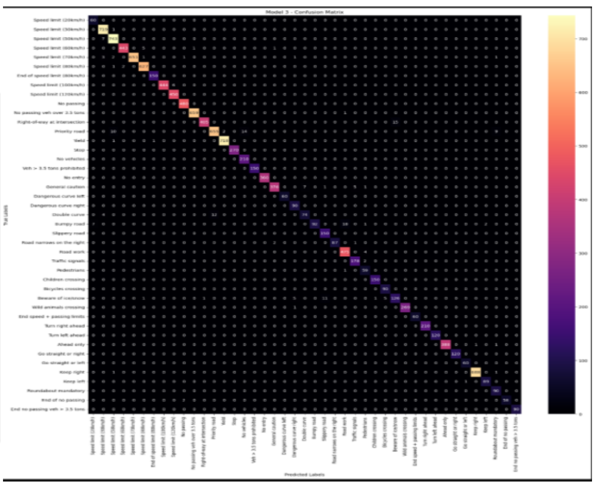
Model	Loss	Accuracy	Validation Loss	Validation Accuracy	AUC
Model 1	0.3795	0.8719	0.2598	0.9140	0.9905
Model 2	0.0773	0.9775	0.0035	0.9991	0.9998
Model 3	0.0367	0.9891	0.0022	0.9993	0.9999

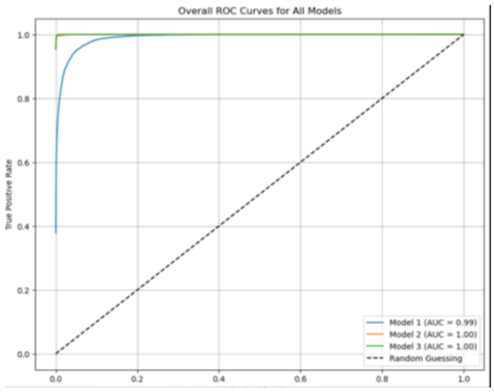
Visualizations:











- **Training and Validation Metrics:**

  Plots show rapid convergence for Model 2 and Model 3 compared to Model 1.

- **Confusion Matrices:**

  Model 3 shows minimal misclassifications across all 43 classes.

- **ROC Curves:**

  Model 2 and Model 3 display near-perfect ROC curves, confirming their robustness.

- **Test Accuracy:**

  - Model 1: 79.22%
  - Model 2: 98.87%
  - Model 3: 98.52%

Best Model: Model 3, based on its overall accuracy, AUC, and generalization.

4. Discussion

Challenges:

1. Class Imbalance: Some traffic sign classes had significantly fewer images, affecting performance for those classes.
2. Overfitting: Early experiments showed overfitting, mitigated by applying dropout and data augmentation.
3. Computation: Training deeper CNN models required substantial time and computational resources.

What I Learned:

- Pre-trained models like MobileNetV2 are effective for feature extraction but may not outperform well-optimized custom CNNs.
- Hyperparameter tuning, regularization, and data augmentation are critical for model generalization.
- A deeper architecture with appropriate dropout and learning rate scheduling improves accuracy.

Future Improvements:

1. Class Imbalance Handling: Use techniques like SMOTE or weighted loss to improve minority class performance.
2. Real-time Deployment: Optimize the model for real-time inference using TensorRT or quantization.
3. Transfer Learning: Experiment with other pre-trained models like ResNet or EfficientNet.
4. Edge Deployment: Optimize the best model for deployment on edge devices like Raspberry Pi.