IMAGE-BASED TRAFFIC SIGN CLASSIFICATION USING DEEP LEARNING

OVERVIEW:

This project focuses on building a deep learning model to classify traffic signs using image datasets. The developed model is intended to support applications like autonomous vehicles, traffic monitoring systems, and driver assistance systems by accurately recognizing and interpreting traffic signs.

FEATURES:

- **Dataset Loading and Preprocessing:** Handles traffic sign image data, including resizing, normalization, and augmentation.
- **Model Training:** Uses deep learning techniques to classify traffic signs.
- **Evaluation:** Evaluates the model's performance using metrics like accuracy, precision, F1 score and confusion matrix.
- **Prediction:** Provides functionality for predicting traffic sign classes from new image

PROJECT APPROACH:

1. **Data Collection:** Use the GTSRB dataset.
2. **Data Preprocessing:** Standardize images, normalize pixel values, and apply augmentation techniques.
3. **Model Development:**
    - Build a CNN from scratch or utilize pre-trained models like VGG16, ResNet, or MobileNet with Transfer Learning.
    - Fine-tune models for optimal performance.

4. **Model Training:** Optimize hyperparameters such as learning rate, dropout, and optimizer.
5. **Evaluation:**
    - Use accuracy, F1-score, and confusion matrices.
    - Validate performance on unseen test data.

ARCHITECTURE:

1. **Loading**: Utilizes a standard traffic sign dataset, such as the German Traffic Sign Recognition Benchmark (GTSRB).
2. **Data Preprocessing**:
    - Resizing all images to a fixed size (e.g., 32x32 pixels).
    - Normalizing pixel values for efficient training.
    - Data augmentation techniques such as rotation, flipping, and zoom to make the model robust
3. **Preprocessing Steps:**
    - **Image Resizing**: All images resized to 224x224 pixels to match model input dimensions.
    - **Normalization**: Pixel values scaled to the range [0, 1].
    - **Data Augmentation**: Includes random rotation, zoom, and horizontal flipping for improved generalization.
4. **Training and Evaluation**
    - **Training Steps:**
    - Initialize the models (ResNet, VGGNet, MobileNet) with pre-trained weights (e.g., ImageNet).
    - Fine-tune the models on the traffic sign dataset:
    - Optimizer: Adam
    - Loss Function: Categorical Cross-Entropy
    - Metrics: Accuracy
    - Train for 25-50 epochs, monitoring the validation loss to prevent overfitting.
5. **Evaluation Metrics:**
    - **Accuracy**: Measures the percentage of correctly classified images.
    - **Confusion Matrix**: Identifies misclassified categories.
    - **Training Time**: Compares runtime efficiency across models.

MODEL VIEW:

1. Convolutional Neural Network (CNN)

Architecture Overview:

A traditional CNN consists of several layers that progressively extract and learn features from input images. Its structure typically follows this pattern:

1. **Input Layer**: Accepts image data (e.g., 32x32 RGB image).
2. **Convolutional Layers**: Extract spatial features by applying filters (kernels).
3. **Activation Function**: ReLU is applied after each convolution to introduce non-linearity.
4. **Pooling Layers**: Reduce spatial dimensions using max-pooling or average-pooling, retaining key features while reducing computation.
5. **Fully Connected Layers (Dense Layers)**: Flatten the output of the final pooling layer and connect it to a fully connected network for classification.
6. **Output Layer**: Produces probabilities using softmax for classification tasks.

Key Features:

- Simplicity and interpretability.
- Suitable for small-scale datasets.
- May overfit on complex datasets without regularization (e.g., dropout).

2. ResNet (Residual Network)

Architecture Overview:

ResNet introduced the concept of **residual connections** to address the vanishing gradient problem in very deep networks. This innovation enables training of extremely deep models (e.g., ResNet-50, ResNet-101).

1. **Building Block (Residual Block)**:
    - Contains two or three convolutional layers.
    - Includes a **shortcut connection** that skips one or more layers.
    - Adds the shortcut output to the output of the convolutions (F(x) + x).
2. **Structure**:
    - Stacks multiple residual blocks.
    - Begins with an initial convolutional layer and pooling.
    - Ends with a global average pooling and fully connected layer.

Key Features:

- Allows networks to be very deep without performance degradation.
- Improves training efficiency and reduces overfitting.
- Commonly used in ResNet-18, ResNet-34, ResNet-50 (uses bottleneck blocks), etc.

3. VGGNet (Visual Geometry Group Network)

Architecture Overview:

VGGNet is known for its simplicity and uniformity in layer design. It uses a stack of convolutional layers with fixed-size kernels (3x3) and max-pooling layers.

1. **Design Philosophy**:
    - Uses small receptive fields (3x3 filters) throughout the network.
    - Increases depth by stacking more convolutional layers.
    - Reduces spatial dimensions with max-pooling.
2. **Structure**:
    - Input layer (e.g., 224x224 RGB image).
    - Sequential convolutional layers (3x3 kernels, stride=1, padding=1).
    - Max-pooling layers (2x2 kernels, stride=2).
    - Fully connected layers at the end (often three dense layers).
    - Softmax output for classification.

Key Features:

- High performance on benchmarks like ImageNet.
- Heavy computational cost and memory usage.
- Basis for modern architectures.

4. MobileNet

Architecture Overview:

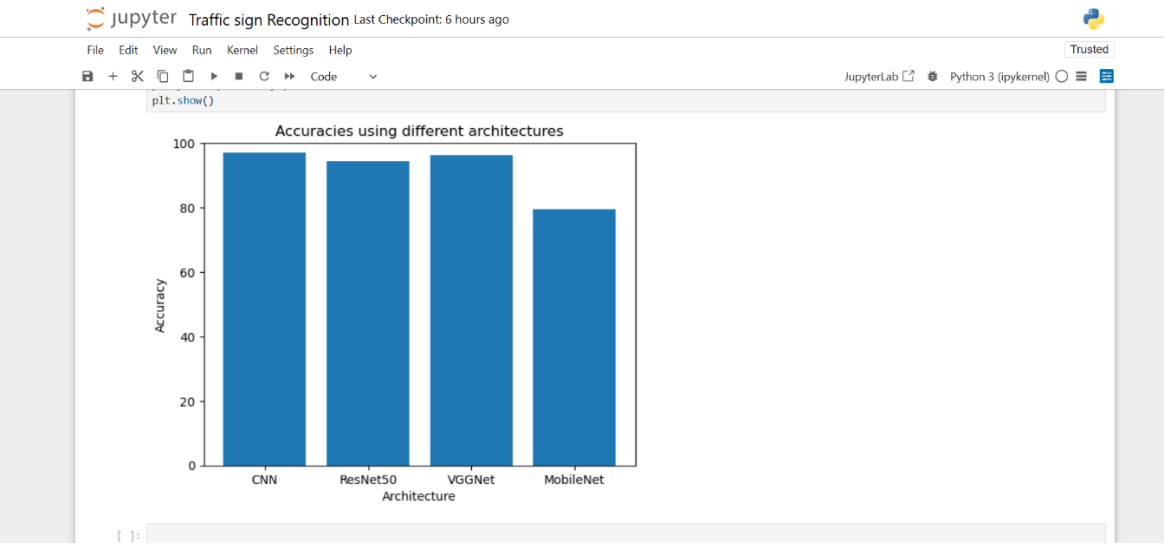
MobileNet is optimized for efficiency, making it suitable for mobile and embedded devices. It uses **depthwise separable convolutions** to reduce the number of parameters and computational cost.

1. **Depthwise Separable Convolutions**:
    - **Depthwise Convolution**: Applies a single filter per input channel.
    - **Pointwise Convolution**: Applies a 1x1 filter to combine the outputs of depthwise convolutions.
    - Reduces computation compared to standard convolution.
2. **Structure**:
    - Consists of depthwise separable convolutional layers followed by batch normalization and ReLU.
    - Includes a global average pooling layer before the output layer.
3. **Variations**:
    - MobileNetV2 introduces **inverted residual blocks** and **linear bottlenecks** for better performance.
    - MobileNetV3 adds lightweight attention mechanisms.

Key Features:

- Efficient for low-power devices.
- Achieves high accuracy with fewer parameters.
- Scalable using width and resolution multipliers.

OUTPUT:



CONCLUSION:

The Traffic Sign Recognition model demonstrates excellent performance in classifying traffic signs, making it suitable for integration into autonomous driving systems. By leveraging a CNN architecture, the system achieves robust and accurate classification, even in challenging conditions such as occlusion or poor lighting.