About Traffic Sign Classification

This project is a Python-based machine learning application that implements a convolutional neural network (CNN) to recognise and classify traffic signs into one of 8 categories. The purpose of the programme is to demonstrate how deep learning techniques can be applied to real-world computer vision problems, such as those used in driver assistance systems and autonomous vehicles.

The programme is built using TensorFlow/Keras for the machine learning component, OpenCV for handling image input and preprocessing, and NumPy for array manipulation. In addition, scikit-learn is used to split the dataset into training and testing sets.

The main objective of this project was to train a CNN that could identify traffic sign images with high accuracy. To achieve this, all images were resized to a consistent 30×30 resolution, providing a balance between keeping enough information for learning and reducing computational cost. Each image is fed into the network, trained over 10 epochs, and evaluated with accuracy as the main performance measure.

Techniques and Features

The programme combines several important computer science and AI concepts:

- Convolutional Neural Networks (CNNs):
  The architecture includes multiple convolutional and pooling layers to extract patterns from images, followed by dense layers for classification. Filters and kernel sizes were chosen to capture both fine-grained edges and more abstract patterns.
- Regularisation with Dropout:
  A dropout layer was added to reduce overfitting, improving model generalisation to unseen data.
- Categorical Classification:
  The programme uses a softmax activation layer with 43 outputs, allowing classification into multiple categories. Labels are processed with one-hot encoding for compatibility with categorical cross-entropy loss.
- Optimisation Strategy:
  The model is compiled with the Adam optimiser, well-suited for faster and more stable convergence without heavy fine-tuning.
- Data Splitting for Evaluation:
  The dataset is divided into training and testing sets using train_test_split, ensuring reliable performance evaluation.

Challenges and Problem Solving

During development, some key challenges included:

- Collision Between Data and Model Needs:
  Handling the dataset was not straightforward. Many images varied in size and quality,

so resizing and preprocessing were essential. It took some trial and error to find the right input dimensions (30 × 30) that balanced speed and accuracy.

- Architecture Design Choices:
  Choosing how many convolutional layers to include and how many filters to use was challenging. Too few layers led to poor accuracy, while too many increased training time without noticeable improvements. Through experimentation, three convolutional layers followed by dense layers gave the best trade-off.
- Overfitting:
  At first, the model performed well on training data but poorly on testing data. Adding a dropout layer solved this by forcing the model to generalise.

These challenges deepened my understanding of CNNs, data preprocessing, and the balance between computational efficiency and model complexity.

Reflection

This project helped me strengthen my skills in working with machine learning frameworks and computer vision datasets. I learned how architectural design decisions (kernel size, filter count, dropout, etc.) directly affect model performance. More importantly, I developed problem-solving strategies for handling issues like overfitting and dataset preprocessing. This was the first neural network I built and I definitely learnt a lot through developing it.

Future Work

To further develop this project, I would like to:

- Add data augmentation (rotations, flips, brightness changes) to simulate real driving conditions.
- Explore transfer learning with pretrained models such as VGGNet or ResNet to boost performance.
- Implement confusion matrix analysis to identify which traffic signs are most difficult for the model.
- Deploy the trained model onto lightweight platforms (e.g., TensorFlow Lite for mobile or embedded devices in cars).