**The title is A Lightweight Convolutional Neural Network Model for Traffic Sign Classification Based on Enhanced LeNet-5 Network.**

**This is our table of content, and I want to explain our project in six parts.**

**Reference**

This project is based on an academic paper, and the convolutional neural network model constructed with Keras we use is based on that.

**Background**

The reason why we choose Traffic Sign Classification as our project is that the road safety is attracting the attention of many researchers around the world since it is indispensable in protecting human life. Traffic Sign Classification can help build up the driver assistance systems which could play a very important role in protecting human life on the road.

Moreover, driverless vehicles are very popular in the field of science and technology research nowadays. How to make driverless cars accurately recognize and classify traffic signs to help them drive safely is a very important issue.

So, traffic sign classifiers can be used for traffic sign recognition systems which can do great help to driverless vehicles. Moreover, a road and traffic sign classifier system could in principle be developed as part of an Intelligent Transport Systems that continuously monitors the driver, the vehicle, and the road in order, for example, to inform the driver in time about upcoming decision points regarding navigation and potentially risky traffic situations.

For several years now, systems for the detection, classification, and recognition of road signs have become a very important research topic for researchers.

**Dataset**

We are using the German Traffic Sign Recognition Benchmark (GTSRB) database as the dataset. It is an image classification dataset. And the images are photos of traffic signs which are classified into 43 classes.

We downloaded the training set data and test set data provided by the data set from its official website. The training set data has 43 folders, each folder represents a class, and each folder has many corresponding traffic sign pictures. After observation, it can be concluded that these files are a group of pictures taken from far and near. Although the same traffic sign is taken, the size and shooting angle of each picture is different. Moreover, the datatype of the images is ppm, aka Portable Pixmap Image File. And there are also CSV files that declare the class name of each image.

For the test set, there is only one folder, which contains pictures and CSV files.

And the main reason we chose this data set is that many high-quality papers related to traffic sign recognition are written based on this data set.

**Preprocessing**

After loading the data, we need to resize the data to fit the framework of our neural network input layer which means the data should be resized to 32, 32 scale.

The third step is resizing. Because the color of the traffic sign is not very important, and for the embedded implementation using the webcam, lightness is more significant than color, so the RGB space of the images is converted to grayscale.

The next one is normalization. It means mapping the pixel data of images to the interval [0, 1] in the form of float numbers, which ensure equal representation of all features, in other words, it is beneficial to the training of the model when doing the math.

The last step is histogram equalization. Histogram equalization is used afterward to improve the image contrast, especially for grayscale images. Adjusting the histogram of the training images effectively helps to make the training process more dynamic.

Like this figure, the bottom one is histogram equalized.

**Structure of the Algorithm**

The ideal machine-learning algorithm we chose is CNN, aka Convolutional Neural Network, which is powerful for image processing, and for more specific, the machine-learning algorithm is Lightweight Enhanced Lenet-5 network. And this network is the improved LeNet-5. We can see the contrast between them through the concept map.

The training section is mainly using Keras in python3.

The basic framework is that there are 4 convolution layers, 2 max-pooling layers, 2 dropout layers, 2 batch normalization layers, and 2 full connection layers.

There are two successive convolution layers (C1, C2); each convolution layer is followed by a Batch Normalization layer with a LeakyReLU activation function and finally a max-pooling layer(S3).

After the first max-pooling layer, there are also two successive convolution layers (C4, C5); each convolution layer is still followed by a Batch Normalization layer with a LeakyReLU activation function and finally a max-pooling layer(S6) with DropOut regularization.

After the data from the last max-pooling layer flatten(H7) whose nodes are equal to 480, there is a hidden layer(H8) whose nodes are equal to 500, followed by the Batch Normalization layer with LeakyReLU function and DropOut regularization.

The output layer contains 43 nodes, which are the number of classes for the traffic sign data.

**Train**

For training, the training dataset is split into 20% validation data and 80% training data. The test data is prepared officially. It should be noted that since the data set is relatively small, with only about 40,000 images when splitting the data set, it may appear that the verification set does not contain certain categories at all, so it should be divided according to the distribution ratio of the categories before the splitting.

This is our whole progress of machine learning. After feeding the algorithm dataset and the convolutional neural network will soon be well trained.

The figure in the right is the accuracy and loss value of the train and validation set when doing the training. It is clear that when the epoch is closing to 20, the accuracy of both the train and validation set is approaching 1.

After that, we can use a test dataset to evaluate the model. For our model trained 20 epochs, the accuracy is about 0.9755.

Moreover, we can input some other raw graphs into the model to get the predicted names and their probabilities.