Traffic sign detection
using CNN

KISHORE KUMAR A V - CH.EN.U4CSE20134

Problem statement:

Traffic Sign Detection is an essential component of Advanced Driver Assistance Systems and autonomous driving. The goal of this project is to develop a machine learning model using Convolutional Neural Networks (CNN) that can accurately detect and classify traffic signs from images captured by cameras mounted on vehicles. The model should be able to identify a wide range of traffic signs, including speed limit signs, stop signs, yield signs, and others, under various lighting and weather conditions. The objective is to improve road safety by providing real-time information to drivers about speed limits, no-passing zones, and other traffic regulations, and assist autonomous vehicles in making informed decisions about speed, direction, and traffic rules. The project will involve data preprocessing, model training and evaluation, and optimization of hyperparameters to achieve the best possible performance. The final model will be deployed and tested on real-world data to demonstrate its effectiveness in traffic sign detection.

Literature Survey

[1] "Traffic Sign Detection and Recognition Using Deep Learning Based Object Detection Technique" by G. Shukla and S. H. Kanhere. This paper proposes a novel approach for traffic sign detection and recognition using a deep learning-based object detection technique.

[2] "Traffic Sign Recognition Using Convolutional Neural Networks" by Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. This paper presents a deep learning approach for traffic sign recognition using Convolutional Neural Networks (CNNs).

[3] "Traffic Sign Detection and Recognition Using Convolutional Neural Networks" by A. M. Khan, S. K. Jindal, and R. K. Singh. This paper proposes a hybrid approach for traffic sign detection and recognition using a combination of Convolutional Neural Networks (CNNs) and Template Matching techniques.

About Dataset :

- More than 40 classes
- More than 50,000 images in total
- Large, lifelike database

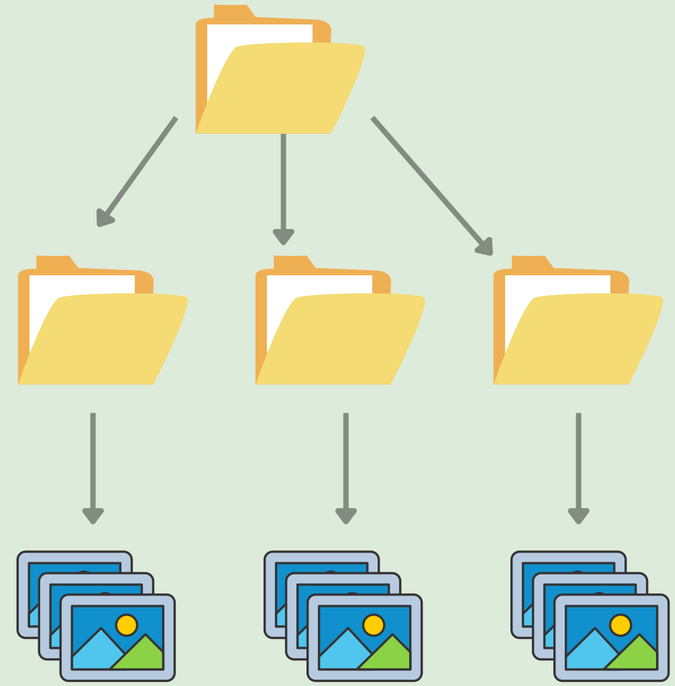
GTSRB - German Traffic Sign Recognition Benchmark

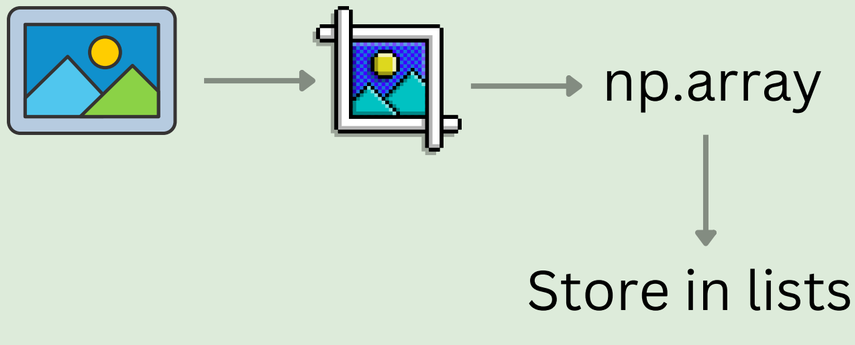
Multi-class, single-image classification challenge

k kaggle.com

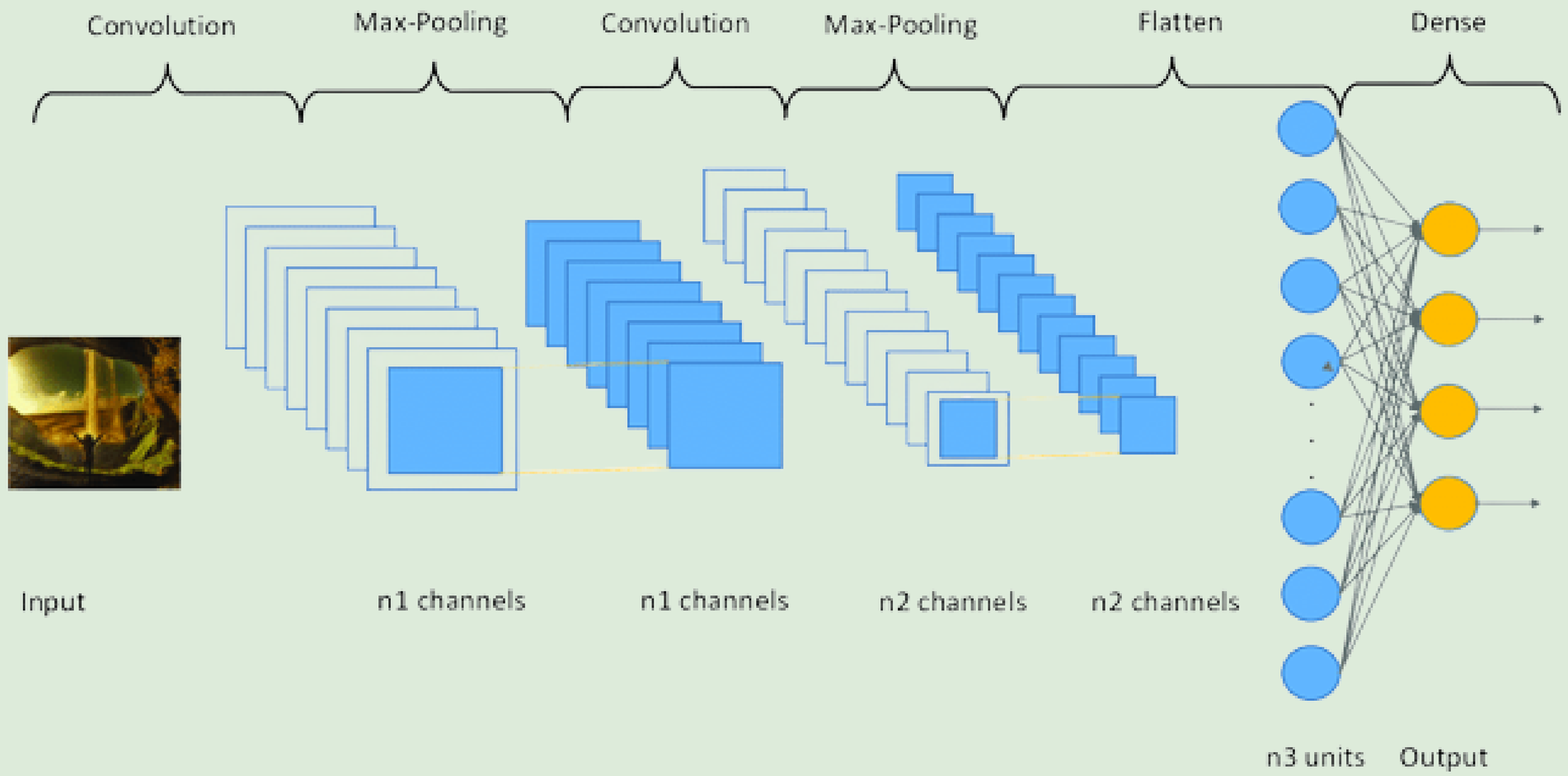
Data preprocessing

```
for i in range(classes):
    path = os.path.join(cur_path,'train',str(i))
    images = os.listdir(path)
    for a in images:
        try:
            image = Image.open(path + '\\'+ a)
            image = image.resize((30,30))
            image = np.array(image)
            data.append(image)
            labels.append(i)
        except Exception as e:
            print(e)
```





Learning Model

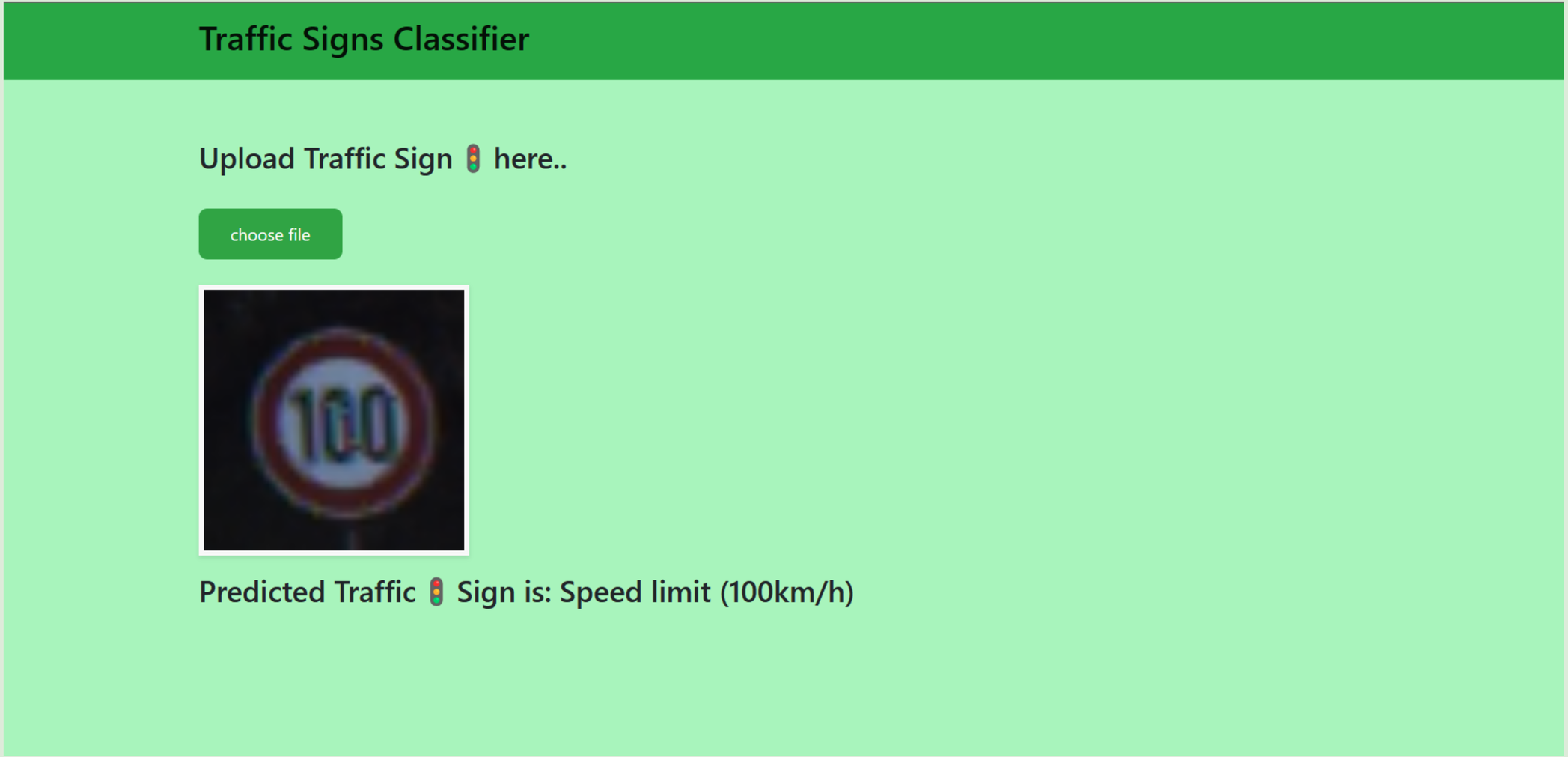


Learning Model

First, we create a Keras Sequential Model and create a Convolution layer with 32 feature maps at size (3,3). Relu is the activation is used and later we downsample the data by using the MaxPooling technique. We further scale down the image by passing it through the second Convolution layer with 64 feature maps. This process is called Feature Extraction. Once feature extraction is done, we can flatten the data into a single vector and feed them to hidden dense layers. The softmax activation is used at the output layer to make sure these outputs are of categorical data type which is helpful for Image Classification.

```
model = Sequential()
model.add(Conv2D(filters=32, kernel_size=
(5,5), activation='relu',
input_shape=X_train.shape[1:]))
model.add(Conv2D(filters=32, kernel_size=
(5,5), activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.25))
model.add(Conv2D(filters=64, kernel_size=(3,
3), activation='relu'))
model.add(Conv2D(filters=64, kernel_size=(3,
3), activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.25))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(rate=0.5))
model.add(Dense(43, activation='softmax'))
```

Hands On:


Traffic Signs Classifier
Upload Traffic Sign 🚦 here..
choose file
Predicted Traffic 🚦 Sign is: Speed limit (100km/h)

Accuracy:

```
from sklearn.metrics import accuracy_score
print(accuracy_score(label, Y_pred))
```

```
0.9463182897862232
```

Thank You