IIT BHUBANESWAR

DS LAB REPORT-2021

Traffic Sign Recognition with Deep Learning

ALAKARA KADIRESAN KARTHIK

21EC06007

School Of Electrical Sciences

Master Of Technology (MTech)

Abstract:

Recognition of traffic signs plays a crucial role in management of the traffic-sign inventory. It provides an accurate and timely way to manage traffic-sign inventory with a minimal human effort. In the computer vision community, the recognition and detection of traffic signs are a well-researched problem. A vast majority of existing approaches perform well on traffic signs needed for advanced driver-assistance and autonomous systems. However, this represents a relatively small number of all traffic signs and performance on the remaining set of traffic signs, which are required to eliminate the manual labor in traffic-sign inventory management, remains an open question. In this paper, we address the issue of detecting and recognizing a large number of traffic-sign categories suitable for automating traffic-sign inventory management.

In this paper, a deep learning-based road traffic signs recognition method is developed which is very promising in the development of Advanced Driver Assistance Systems (ADAS) and autonomous vehicles. The system architecture is designed to extract main features from images of traffic signs to classify them under different categories. The presented method uses a modified LeNet-5 network to extract a deep representation of traffic signs to perform the recognition. It is constituted of a Convolutional Neural Network (CNN) modified by connecting the output of all convolutional layers to the Multilayer Perceptron (MLP). The training is conducted using the German Traffic Sign Dataset and achieves good results on recognizing traffic signs.

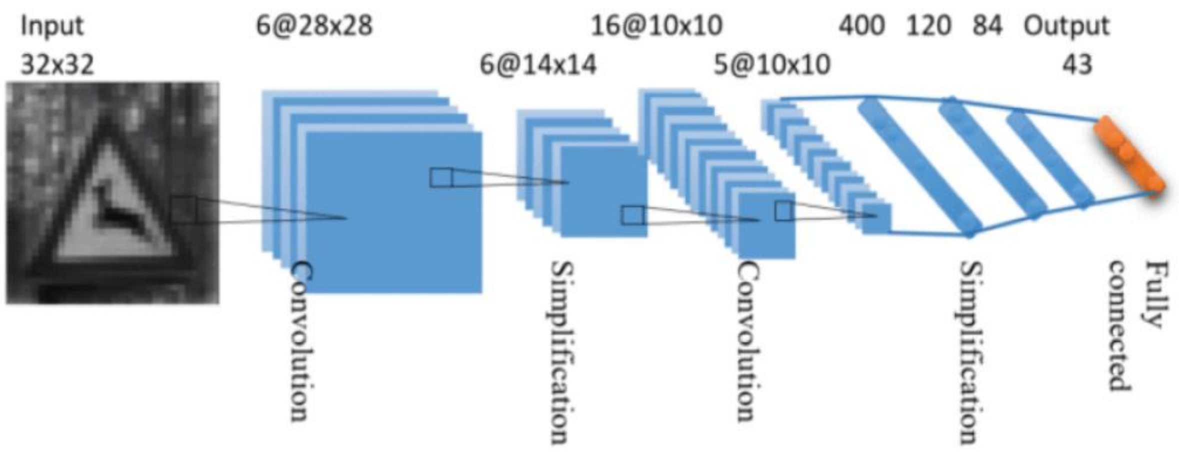
Theory:

LeNet-5 Model:

Lenet-5 is one of the earliest pre-trained models proposed by Yann LeCun and others in the year 1998. used this architecture for recognizing the handwritten and machine-printed characters.

The main reason behind the popularity of this model was its simple and straightforward architecture. It is a multi-layer convolution neural network for image classification.

The network has 5 layers with learnable parameters and hence named Lenet-5. It has three sets of convolution layers with a combination of average pooling. After the convolution and average pooling layers, we have two fully connected layers. At last, a SoftMax classifier which classifies the images into respective class.



Lenet-5 Architecture

CNN Model:

The convolutional neural network uses a special mathematical operation called convolution instead of matrix multiplication in at least one of its layers. It is officially formed by a stack of layers.

- The convolution layer (CONV) which processes the data of a receiver field.
- The pooling layer (POOL); which compresses the information by reducing the size of the intermediate image.
- The correction layer (ReLU), with reference to the activation function (Linear Rectification Unit).
- The Fully Connected Layer (FC), which is a perceptron-type layer.

Currently, Convolutional networks are gradually replaced traditional computer vision algorithms for different applications such as object classification and pattern recognition. It is used for the extraction and the learning of depth description of the traffic signs.

Methodology:

In this project the implementation is done in different phases in the following manner: collecting the dataset, pre-processing the dataset, training the Convolutional Neural Network (CNN) model.

Traffic Sign Dataset:

A rich dataset is needed in object recognition based on neural network in order to train the system and evaluate its results. For the purpose of traffic signs classification, we used the German Traffic Sign Benchmark (GTSB) which contains 43 classes.

Training Data:

The unbalanced distribution of images in the German Traffic Sign Benchmark privileges some classes over others during the training phase because they are better represented in terms of number of images. In order to make sure that the learning of the network is well performed, a data augmentation of some classes is done by applying some geometric transformations (rotation, translation, and shear mapping) on many of their images.

The Dropout Operation:

In neural network training, the dropout method is established to prevent from over-fitting by shutting down some neurons during the training phase to give the network a flexible margin to react to inputs out of the training examples data. In our case, we performed a shutting down of 30% and 50% of the total neurons of the second and third layers of the fully connected network respectively. A 90% dropout (shutting down 10% of neurons) on the layer preceding the fully connected network is used in addition to the previous operation.

Implementation Result:

To build and train the network, the TensorFlow deep learning library is used. Training and testing were implemented using the dataset and the developed method succeeds in classifying the 43 traffic signs classes. The implementation results of the network LeNet-5 and its improvement operations show the impact of each changed element.

Out of the total images, 80% were used for training the model and the rest 20% were used for validation. After experimentations, found that our trained model achieves an accuracy of **94.86%** after training for **25** epochs.

Dataset	Model	Epoch	Accuracy	Loss
German Traffic Sign Benchmark (GTSB)	LeNet-5 and Convolutional Neural Network	1-25	At 1st accuracy= 58.46 % **At 25th epoch accuracy = 94.86%**	1st epoch loss=1.6711 **25th epoch =0.2455**

Dataset Link: GTSRB - German Traffic Sign Recognition Benchmark \| Kaggle

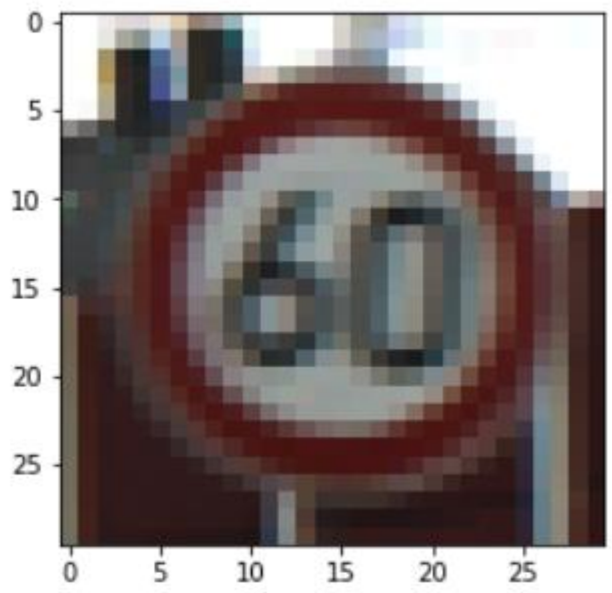


Fig1. Input Test Image

```
'Speed limit (60km/h)'
```

Fig2. Corresponding Output

Conclusion:

This paper presented a convolutional neural network implementation used for traffic signs recognition. The basic proposed network together with the different improvement operations allowed us to be aware of which parts and phases that have the control on the system reliability. It is likely that we would obtain better results by reinforcing the convolution stage of the network with more layers in order to extract more features. It also would be interesting to exclude confusions by comparing classes with the highest proportions in the confusion matrix and pull out their common factors to reverse them by image adjustment.

References:

- P. Dewan, R. Vig, N. Shukla and B. K. Das, "An Overview of Traffic Signs Recognition Methods", *International Journal of Computer Applications*, vol. 168, no. 11, June 2017.

- D. Jianmin and V. Malichenko, "Real time road edges detection and road signs recognition", *IEEE International Conference on Control Automation and Information Sciences (ICCAIS)*, 29-31 Oct. 2015.

Code-flow:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import cv2
import tensorflow as tf
from PIL import Image
from sklearn.model_selection import train_test_split
#from keras.utils import to_categorical
from keras.utils.np_utils import to_categorical
from keras.models import Sequential, load_model
from keras.layers import Conv2D, MaxPool2D, Dense, Flatten, Dropout
import os
cur_path = os.getcwd()
data =[]
labels = []
classes =43
cur_path = os.getcwd()
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))
            # Resizing all images into 30*30
            image =np.array(image)
            data.append(image)
            labels.append(i)
        except Exception as e:
            print(e)

data = np.array(data)
labels = np.array(labels)

X_train, X_test, y_train, y_test =train_test_split(data, labels, test_size=0.2, random_state=0)

y_train = to_categorical(y_train,43)
y_test = to_categorical(y_test,43)

model =Sequential()
model.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu', input_shape=X_train.shap
e[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'))

#Compilation of the model
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
epochs = 25
history = model.fit(X_train, y_train, batch_size=32, epochs=epochs, validation_data=(X_test,
 y_test))

model.save('traffic_sign_training_model.h5')

from keras.models import load_model
model=load_model('traffic_sign_training_model.h5')

# Classes of trafic signs
classes = { 0:'Speed limit (20km/h)',
            1:'Speed limit (30km/h)',
            2:'Speed limit (50km/h)',
            3:'Speed limit (60km/h)',
            4:'Speed limit (70km/h)',
            5:'Speed limit (80km/h)',
            6:'End of speed limit (80km/h)',
            7:'Speed limit (100km/h)',
            8:'Speed limit (120km/h)',
            9:'No passing',
            10:'No passing veh over 3.5 tons',
            11:'Right-of-way at intersection',
            12:'Priority road',
            13:'Yield',
            14:'Stop',
            15:'No vehicles',
            16:'Vehicle > 3.5 tons prohibited',
            17:'No entry',
            18:'General caution',
            19:'Dangerous curve left',
            20:'Dangerous curve right',
            21:'Double curve',
            22:'Bumpy road',
            23:'Slippery road',
            24:'Road narrows on the right',
            25:'Road work',
            26:'Traffic signals',
            27:'Pedestrians',
            28:'Children crossing',
            29:'Bicycles crossing',
            30:'Beware of ice/snow',
            31:'Wild animals crossing',
            32:'End speed + passing limits',
            33:'Turn right ahead',
            34:'Turn left ahead',
            35:'Ahead only',
            36:'Go straight or right',
            37:'Go straight or left',
            38:'Keep right',
            39:'Keep left',
            40:'Roundabout mandatory',
            41:'End of no passing',
            42:'End no passing vehicle > 3.5 tons' }

from PIL import Image
import numpy as np
import matplotlib.pyplot as plt

def test_on_image(img):
    model = load_model('traffic_sign_training_model.h5')
    data=[]
    image = Image.open(img)
    image = image.resize((30,30))
    data.append(np.array(image))
    X_test=np.array(data)
    #Y_pred = model.predict_classes(X_test)
    Y_pred = (model.predict(X_test))>.5
    return image,Y_pred


plot,predictions=test_on_image(r'C:\Users\karth\Desktop\Traffic_sign_python_01112021\T
est\00115.png')

print('predcitions=',predictions)
plt.imshow(plot)
plt.show()
count=-1
for i in predictions[0]:
    count+=1
    if i==True:
        print(i)
        break
    else:
        pass
print(classes[count])
```