UNIVERSITY PARTNER

UNIVERSITY OF WOLVERHAMPTON

HERALD COLLEGE KATHMANDU

Artificial Intelligence and Machine Learning (6CS012)

Portfolio-2 Deep Learning

Student Id : 2050453

Student Name : Kunga Nyima Gurung

Group : L6CG5

Submitted on : 1/05/2022

Table of Contents

Abstract

Utilizing the German Traffic Sign Recognition Benchmark dataset, we created a classifier model using Convolutional Neural Networks to classify traffic signs. In a nutshell, CNN is a computer vision supervised learning and method that is frequently used in picture categorization. We built a model with a total of 240651 trainable hyperparameters using 12 layers. With the testing dataset, we acquired above 95% accuracy from the model after evaluating it. It is recommended to obtain a region of interest and do hyperparameter adjustment for increased accuracy.

Introduction

CNN is employed in situations where images contain a large number of objects, features, etc. for classification, recognition, object detection, or facial recognition. Image classification can be simply defined as the process of taking an image as input and generating a class or set of classes that best describes it. The German Traffic Sign Recognition Benchmark is used in this paper to create a classifier model to classify traffic signs. The dataset, which included 43 different sign classes, was published in 2011 at the International Joint Conference on Neural Networks.

Aims and Objective

The goal of the research is to create a classifier model that can accurately classify traffic signs using a deep learning computer vision method called Convolutional Neural Network.

The objective includes exploring and visualizing the data, preprocess dataset pictures accurately, such as turning a color image to grayscale, in the classifier model, correct the hyperparameter and add the required number of layers, etc.

Convolution Neural Network (CNN):

CNN can be defined as the state-of-the-art in image categorization. It's also known as Convnet, which is a multi-layer neural network modeled after the vision systems of real animals (Sultana, et al., 2018). It's used to recognize people, classify photos, and recognize objects in images. It is a form of the neural network model that is specifically developed to work with image data. The process of identifying which class or combination of classes best describes an input image is known as image classification. Artificial neural networks (ANNs) are often used to address pattern recognition problems. An ANN is a mathematical model made up of neural units connected by artificial neurons, similar to biological neural networks. Neurons are often organized in layers, with connections occurring exclusively between neurons at neighboring levels. The input low-level feature vector is placed in the first layer, and it is changed to the high-level feature vector as it advances from layer to layer. The number of classification classes is equal to the number of output layer neurons. As a result, the output vector is a probability vector indicating the likelihood that the input vector belongs to a given class (Shustanov & Yakimov, 2017).

Why Use CNN for image classification?

Because of its great accuracy, it is utilized for image categorization and recognition. After being inspired by human visual perception of object recognition, computer scientist Yann LeCun introduced it in the late 1990s (MALADKAR, 2018). (Venkatesh, 2018) proposed a paper employing a convolutional neural network to segment MRI images, and the findings show that tumor detection in MR images using CNN is predictable, and the prediction accuracy is good. It can be utilized in a variety of fields and perform important tasks like as facial recognition, document analysis, climate comprehension, image recognition, and item identification, among others. CNN is the most popular deep learning algorithm since it achieves the benefits of offering maximum performance and efficiency. Deep learning has aided greatly in the growth of science sectors.

Methodology

At first, we decided on data collection. We used various data exploration tools and then used EDA techniques to visualize the data. Finally, in this study, we create a CNN-based classification model. After that, the model is trained and validated, and we attempt a test using the validated model. After that, we put our classifier to work. As previously stated, we employed the CNN model for classification, which is based on the convolution of filters and images or raw inputs. The CNN-based technique will be described in this section. Figure 1 depicts a flowchart of the key phases.

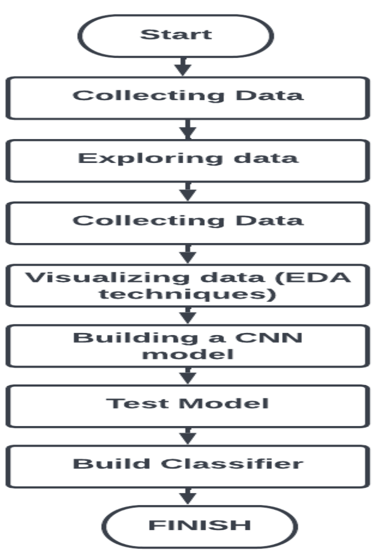
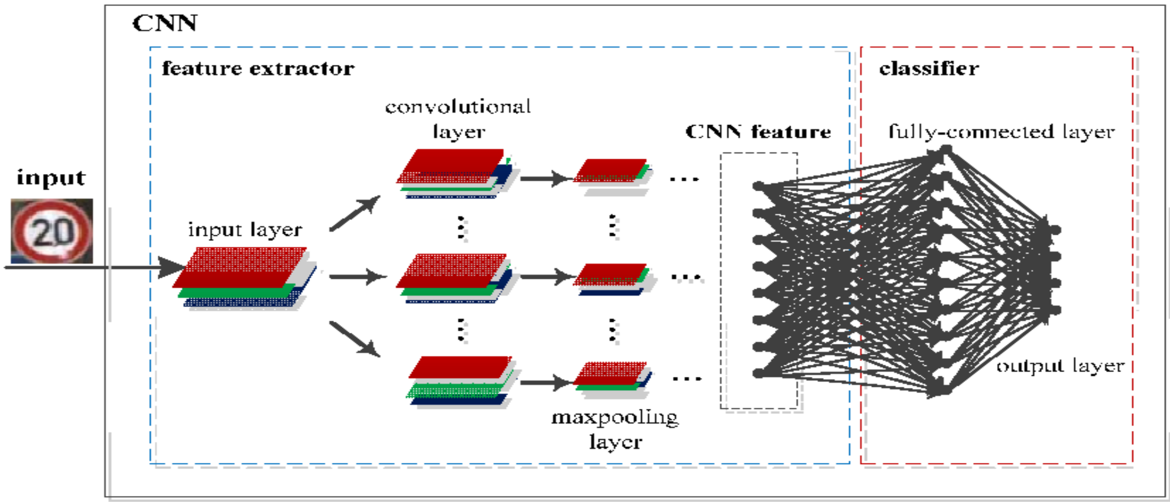


Figure 1: Workflow

The classification workflow began with the collection of the dataset. After that, using exploratory data analysis, the data was explored and visualized. Then, in order to acquire a classification result, a CNN model and a test model were established.



The layers of the CNN include:

- **Input Layer**: It collects and resizes images before passing them on to further layers for feature extraction.
- **Convolution Layer:** During testing, it acts as an image filter, extracting features from images and calculating match feature points.
- **Pooling Layer:** After that, the extracted feature sets are sent to the 'pooling layer.' This layer reduces the size of huge photographs while keeping the most crucial details. It keeps each feature's perfect fit within the window, boosting the window's value.
- **Rectified Linear Unit Layer:** It replaces every negative integer in the pooling layer with 0. This prevents learnt values from becoming stuck near 0 or bursting up to infinity, keeping the CNN mathematically stable.
- **Fully Connected Layer:** This layer categorizes and names the high-level filtered pictures.

Model Summary

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 32)        832
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 22, 22, 32)        25632
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 11, 11, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 11, 11, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 9, 9, 64)          18496
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 7, 7, 64)          36928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 3, 3, 64)          0
_________________________________________________________________
dropout_1 (Dropout)          (None, 3, 3, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 576)               0
_________________________________________________________________
dense (Dense)                (None, 256)               147712
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 43)                11051
=================================================================
Total params: 240,651
Trainable params: 240,651
Non-trainable params: 0
_________________________________________________________________
```

Figure 2: model summary

There are 12 layers in this classifier model: four convolution layers, two pooling layers, three dropout layers, one flattens layer, and two dense layers. There are three parts to the model.

At the start of the first set, there are two convolution layers. We present a 32-bit filter with a 5x5 size, a ReLU activation function, and an image input size as the hyperparameter. Then, in the following pooling layer, we added a dropout layer with a rate of 0.25 and Max Pooling of size 2x2. The hyperparameters max-pooling and rate of dropout are used here.

The second set is similar to the first, except with a different hyperparameter but the same layer. The convolution layer's filter has a size of 3x3 and a ReLU activation function, as well as the identical pooling and dropout layers as the first set.

The completely connected layer is made up of the ultimate set of layers. A flatten layer and dense layer with 256 nodes and ReLU activation function are first applied, followed by a dropout layer of 0.5 and a dense layer with 43 nodes and SoftMax activation function. Hyperparameters include activation functions, nodes, and dropout values.

There are 240651 trainable hyperparameters across all layers. By adding further convolution and pooling layers, you can further minimize the number of parameters. The more convolution layers we add, the more specific and nuanced the characteristics retrieved become.

Training the Model

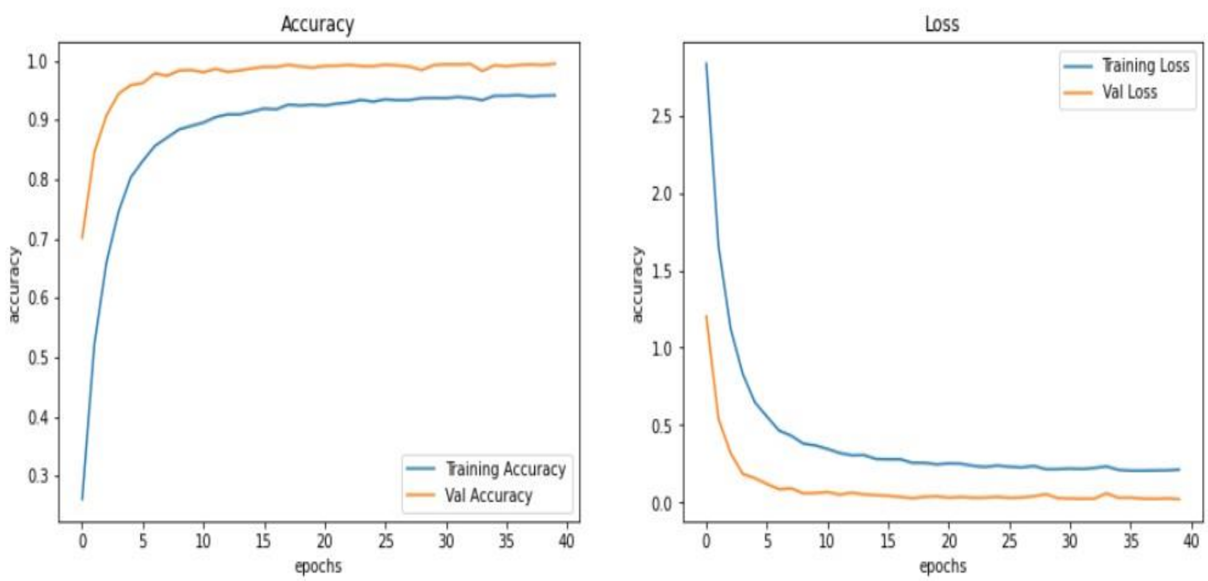


Figure 3: Training loss vs. validation loss (right) and training accuracy vs. validation accuracy (left)

Finally, for our traffic sign classifier, we create a graphical user interface. The GUI (Graphical User Interface) is designed for uploading images, and we must offer the same dimension we used while creating the model in order to anticipate the traffic sign. A graphical user interface will save us a lot of time when it comes to testing and viewing the outcomes of our model prediction.

Categorical cross-entropy is the loss function utilized in the model. In all Kera's Optimizers, the constant learning rate is the default schedule. The learning rate in the SGD optimizer, for example, is set to 0.01. We simply instantiated an SGD optimizer and passed the input learning rate=0.001 to use a custom learning rate. We utilized the Adam optimization technique to train the model. For 40 epochs, we trained our model.

Findings and discussion

Evaluation metrices:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	60
1	0.99	0.99	0.99	720
2	0.96	1.00	0.98	750
3	0.99	0.96	0.97	450
4	0.99	0.98	0.98	660
5	0.98	0.99	0.98	630
6	0.98	0.91	0.94	150
7	0.99	1.00	0.99	450
8	0.99	0.99	0.99	450
9	0.97	1.00	0.99	480
10	0.99	1.00	0.99	660
11	0.97	0.92	0.95	420
12	0.99	0.97	0.98	690
13	0.99	1.00	0.99	720
14	0.92	1.00	0.96	270
15	0.96	0.88	0.92	210
16	0.99	1.00	1.00	150
17	1.00	0.93	0.96	360
18	0.98	0.93	0.96	390
19	1.00	0.97	0.98	60
20	0.94	0.93	0.94	90
21	0.98	0.68	0.80	90
22	0.99	0.96	0.97	120
23	0.84	1.00	0.91	150
24	0.90	0.96	0.92	90
25	0.90	0.99	0.94	480
26	0.96	0.94	0.95	180
27	0.91	0.87	0.89	60
28	0.99	1.00	1.00	150
29	0.80	1.00	0.89	90
30	0.88	0.75	0.81	150
31	0.93	0.99	0.96	270
32	0.86	1.00	0.92	60
33	1.00	0.99	0.99	210
34	0.93	1.00	0.96	120
35	0.98	0.99	0.99	390
36	0.99	1.00	1.00	120
37	1.00	0.98	0.99	60
38	1.00	0.96	0.98	690
39	1.00	0.98	0.99	90
40	0.86	0.92	0.89	90
41	1.00	0.78	0.88	60
42	0.98	0.94	0.96	90
accuracy			0.97	12630
macro avg	0.96	0.95	0.95	12630
weighted avg	0.97	0.97	0.97	12630

Figure 4: Classification report

The categorization report is used to assess the model. A total of 12930 distinct test photos were used to compile the classification report. The majority of the classes in the report have precision and recall greater than 0.90, as well as f1 scores greater than 0.90. Classes 21, 27, 30, 40, and 41 received f1 scores of 0.80, 0.89, 0.81, 0.89, and 0.88, respectively, which are lower than the overall average f1 score. For example, accuracy informs us about how many of the predicted image classes are actually correct. Recall tells you how many of all the image classes were properly predicted. The F1 value is the average of recall and precision.

Visualization:

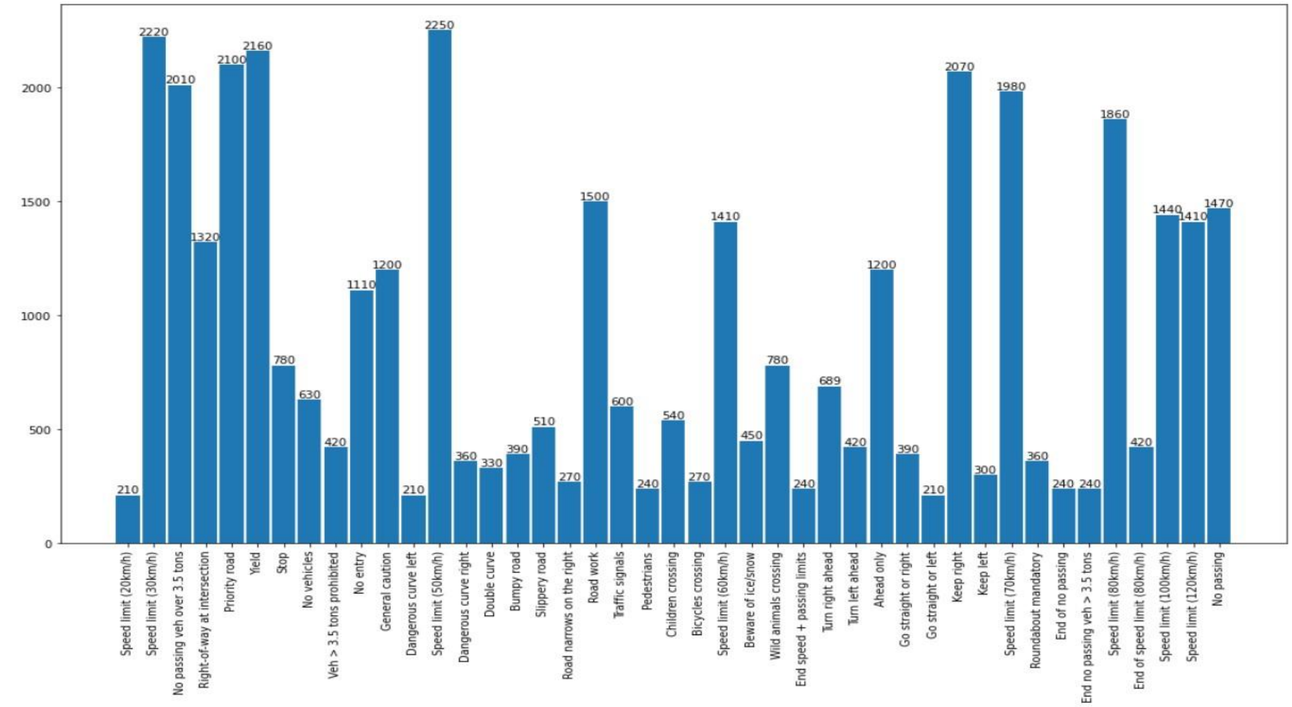


Figure 5: Number of data

As can be seen from the bar graph, the training data for "Speed limit 50 km/h" has the most values (2250), followed by "Speed limit 30 km/h," and so on. The Convolution Neural Network was utilized to classify traffic sign detection using these data.

Conclusion:

We demonstrated and designed an effective alert traffic sign detection and identification system in this study. The identified traffic signs are classified using both color information and the geometric property of the road signs. The experiment demonstrates that the system has a high detection rate of 99.4%. Under various lighting situations, weather conditions, day light conditions, and vehicle speed levels, the system provides precise results. With 95% accuracy, we detected the traffic signs classifier and saw how our accuracy and loss change over time, which is rather good for a simple CNN model. The methods utilized in this study can be applied to the development of general-purpose, sophisticated intelligent traffic surveillance systems.

Recommendation:

The hyperparameter of the model can be tweaked in the future, and more in-depth image preprocessing can be done. We may also use image detection to define the areas of interest (ROI) for data training and testing.

ACKNOWLEDGEMENT

We appreciate the University of Wolverhampton, as well as our lecturer, for allowing us to perform this research.

References

MALADKAR, K., 2018. *https://analyticsindiamag.com/*. [Online]
Available at: https://analyticsindiamag.com/convolutional-neural-network-image-classification-overview/#:~:text=CNNs%20are%20used%20for%20image,visual%20perception%20of%20recognizing%20things.
[Accessed 23 April 2022].

Shustanov, A. & Yakimov, P., 2017. *CNN Design for Real-Time Traffic Sign Recognition.* Samara, Russia, ELSEVIER.

Sultana, F., Sufian, A. & Dutta, P., 2018. *Advancements in Image Classification using Convolutional Neural Network.* Kolkata, India, s.n.

Venkatesh, R. V. a. C., 2018. *Segmentation and Detection of Tumor in MRI images Using CNN and SVM Classification.* Tiruchengode, India, 2018 Conference on Emerging Devices and Smart Systems (ICEDSS), 2018.