Traffic Signal Recognition using Deep Leaning

Varun Goud Maragoni
Department of Computer Science
Blekinge Institute of Technology
Email: vama21@bth.student.se

I. Introduction

This project's objective is to develop a convolutional neural network (CNN) for traffic sign recognition. This challenge of detecting road traffic signs from the vehicle's camera is therefore one of the most important aspects of advanced driver assistance systems. Convolutional neural networks are already widely utilized to solve a variety of computer vision problems. In this particular project, I wanted to train a deep learning model capable of recognizing traffic signs using CNN. Self-driving cars and advanced driver assistance systems require traffic sign recognition to adhere to traffic regulations and provide safe, dependable travel. Conventional procedures, such as determining the color and shape of the traffic sign, made it difficult to recognize traffic signs in the past [3]. This strategy was impractical to employ in a self-driving automobile. Similar to how machine learning could tackle various object recognition problems, It can be used in the scope of this problem to solve it.

I'm using deep learning in this project to train a model to recognize traffic signals. It's a machine learning technology that attempts to replicate the human brain. To handle complex issues, deep learning employs artificial neural networks. Artificial neural networks can be used to solve a variety of problems, including classification, regression, speech to text or text to speech natural language processing, and many others [4]. Convolutional neural networks are an unique sort of artificial neural network that is used to tackle object detection challenges.

A. Convolutional Neural Network

With the development of Artificial Neural Networks, Machine Learning has exploded and gained a lot of interest. These models are primarily inspired by the human brain, and they appear to outperform machine learning algorithms in terms of performance. ANN's are densely connected nodes also known as neurons, that have input, hidden, and output layers as shown in Fig 1. These ANN nodes have a weight associated with them, it tries to learn from the input by changing the weights as per minimizing the error in the output layer.. Convolutional neural networks are a form of ANN that specializes in object-image detection tasks (CNN).[5]

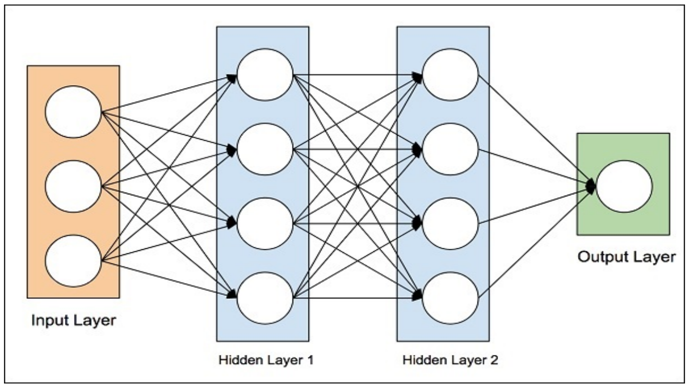


Fig. 1. ANN

CNN works in a similar way to ANN, in that it seeks to extract features or patterns from images that are then used in the network. CNN use a variety of feature maps to find patterns in images, which are then utilized for image recognition. The input layer, convolution layers with activation functions, pooling layer, flattening, and fully-connected layers which is similar to ANN as shown in Fig 2 make up the CNN architecture [5].

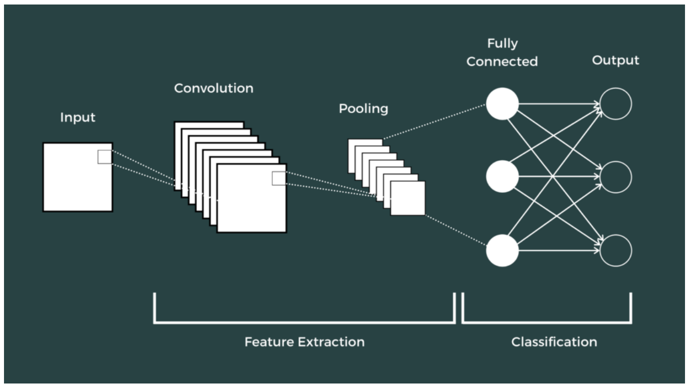


Fig. 2. CNN

II. Related Work

Many studies have been done in the past to recognize traffic signs, but most of them used traditional approaches as traffic signs have definite shape and color. For example, Kim et al. [1] developed a traffic light identification system employing color information, such as RGB, hue-saturation intensity (HSI), and hue-saturation-value (HSV). Haojie Li et al. [2] used color segmentation. Object recognition using machine learning has been a major success as computational power has increased. As a result, we may apply advanced machine learning and deep learning methods to tackle problems like these.

III. Method

A. Dataset

The German traffic signs dataset from kaggle was utilized for this project [6]. It contains 50,000 images of traffic signs such as varying speed limits, no entry, turn left, and so on. There are 43 different categories of traffic signs in this dataset. About 38,000 of the 50,000 images are utilized to train the CNN model, with the rest being used to test the model. At the time of training the neural net, 80% of the 38,000 photos are separated into training and 20% as validating data. This may be utilized to plot graphs between training and validating accuracy.

Fig. 3. Example of dataset images

B. Data prepossessing

For consistency, each image is shrunk to 32*32 pixels, and each pixel is divided by 255 to normalize the data between 0 and 1. Random zooming and rotating the image were used as data augmentation techniques. A bar plot is used to explore data.

C. Implementation

To implement deep learning Tensor flow library has been utilized. As mentioned earlier CNN consists of layers, To design the CNN architecture there isn't any ideal guidelines It is mostly heuristic. So In this project I experimented with two architectures. The first CNN architecture implemented is VGG16 [**?**]. As the images are resized to 32*32 the input layer is of the shape (32,32,3), following that there are 13 conv2d layers,5 maxpool2d layers , 1 flatten layer, 1 dense layer, 1 output layer consisting 43 categories. All the layers have rectifier activation function where as the output layer has softmax activation function which could be useful in predicting the probability for the output categories shown in Fig 4.
The other dense CNN architecture consists of 3 con2d layers, 2 maxpool2d layers, 2 dropout layers to introduce generalization and remove overfitting shown in Fig 5.

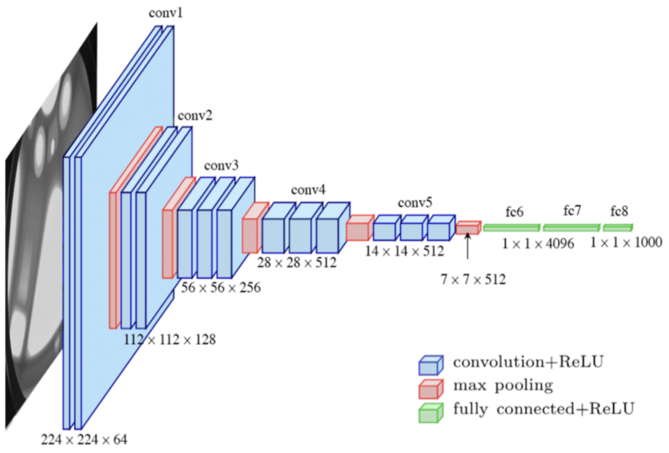
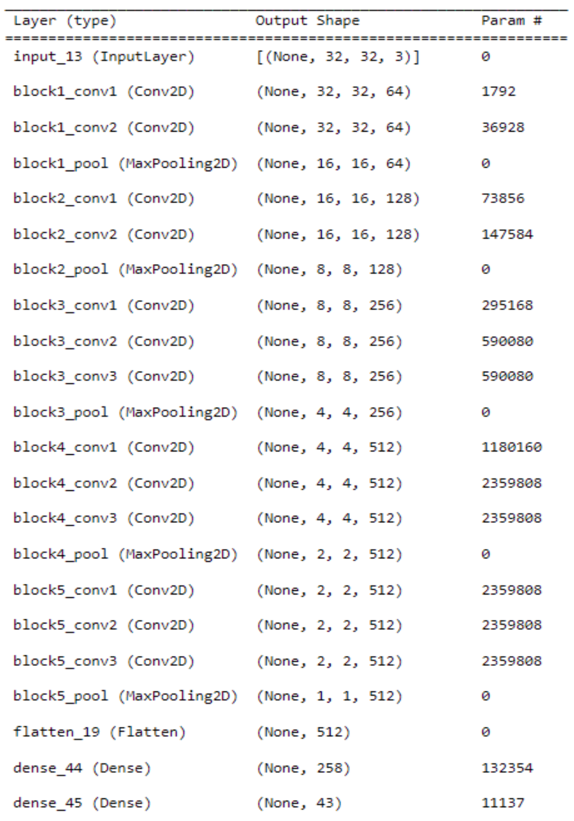
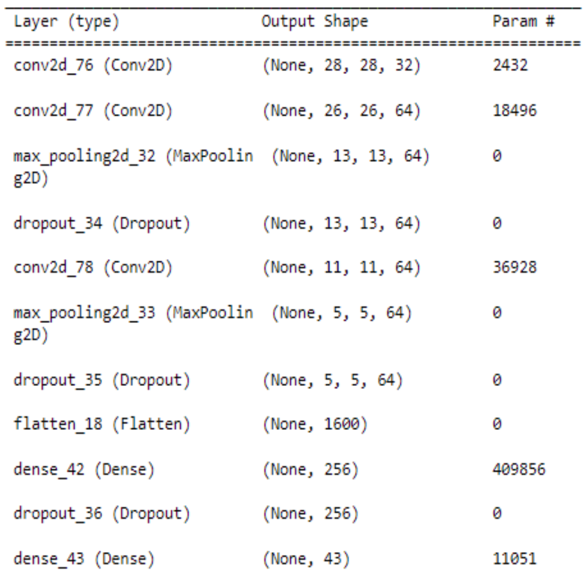


Fig. 4. VGG16 Architecture



```
Layer (type)                 Output Shape              Param #
================================================================
input_13 (InputLayer)        [(None, 32, 32, 3)]       0

block1_conv1 (Conv2D)        (None, 32, 32, 64)        1792

block1_conv2 (Conv2D)        (None, 32, 32, 64)        36928

block1_pool (MaxPooling2D)   (None, 16, 16, 64)        0

block2_conv1 (Conv2D)        (None, 16, 16, 128)       73856

block2_conv2 (Conv2D)        (None, 16, 16, 128)       147584

block2_pool (MaxPooling2D)   (None, 8, 8, 128)         0

block3_conv1 (Conv2D)        (None, 8, 8, 256)         295168

block3_conv2 (Conv2D)        (None, 8, 8, 256)         590080

block3_conv3 (Conv2D)        (None, 8, 8, 256)         590080

block3_pool (MaxPooling2D)   (None, 4, 4, 256)         0

block4_conv1 (Conv2D)        (None, 4, 4, 512)         1180160

block4_conv2 (Conv2D)        (None, 4, 4, 512)         2359808

block4_conv3 (Conv2D)        (None, 4, 4, 512)         2359808

block4_pool (MaxPooling2D)   (None, 2, 2, 512)         0

block5_conv1 (Conv2D)        (None, 2, 2, 512)         2359808

block5_conv2 (Conv2D)        (None, 2, 2, 512)         2359808

block5_conv3 (Conv2D)        (None, 2, 2, 512)         2359808

block5_pool (MaxPooling2D)   (None, 1, 1, 512)         0

flatten_19 (Flatten)         (None, 512)               0

dense_44 (Dense)             (None, 258)               132354

dense_45 (Dense)             (None, 43)                11137
```

Fig. 5. VGG16 architecture



```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_76 (Conv2D)           (None, 28, 28, 32)        2432

conv2d_77 (Conv2D)           (None, 26, 26, 64)        18496

max_pooling2d_32 (MaxPoolin  (None, 13, 13, 64)        0
g2D)

dropout_34 (Dropout)         (None, 13, 13, 64)        0

conv2d_78 (Conv2D)           (None, 11, 11, 64)        36928

max_pooling2d_33 (MaxPoolin  (None, 5, 5, 64)          0
g2D)

dropout_35 (Dropout)         (None, 5, 5, 64)          0

flatten_18 (Flatten)         (None, 1600)              0

dense_42 (Dense)             (None, 256)               409856

dropout_36 (Dropout)         (None, 256)               0

dense_43 (Dense)             (None, 43)                11051
```

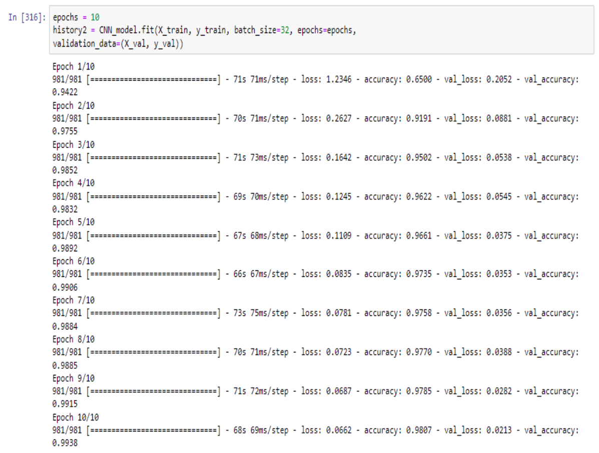
Fig. 6. Architecture 2

D. *Evaluation Metrics*

The output of the model is multi class classification. Hence Accuracy is a good measure of the model. It represents the percentage of the correct predictions to the total number of predictions [8]. It should give a good approximation of how the model performs on unseen test data.

E. *Training the model*

The model is trained on 10 epochs with a batch size of 32. The assigned models loss function is 'categorical_cross entropy' as we are dealing with a categorical problem. The chosen function to minimize the weights of the nodes in the neural net is 'adam' optimizer.



```
In [316]: epochs = 10
          history2 = CNN_model.fit(X_train, y_train, batch_size=32, epochs=epochs,
          validation_data=(X_val, y_val))

Epoch 1/10
981/981 [==============================] - 71s 71ms/step - loss: 1.2346 - accuracy: 0.6500 - val_loss: 0.2052 - val_accuracy:
0.9422
Epoch 2/10
981/981 [==============================] - 70s 71ms/step - loss: 0.2627 - accuracy: 0.9191 - val_loss: 0.0881 - val_accuracy:
0.9755
Epoch 3/10
981/981 [==============================] - 71s 73ms/step - loss: 0.1642 - accuracy: 0.9502 - val_loss: 0.0538 - val_accuracy:
0.9852
Epoch 4/10
981/981 [==============================] - 69s 70ms/step - loss: 0.1245 - accuracy: 0.9622 - val_loss: 0.0545 - val_accuracy:
0.9832
Epoch 5/10
981/981 [==============================] - 67s 68ms/step - loss: 0.1109 - accuracy: 0.9661 - val_loss: 0.0375 - val_accuracy:
0.9892
Epoch 6/10
981/981 [==============================] - 66s 67ms/step - loss: 0.0835 - accuracy: 0.9735 - val_loss: 0.0353 - val_accuracy:
0.9906
Epoch 7/10
981/981 [==============================] - 73s 75ms/step - loss: 0.0781 - accuracy: 0.9758 - val_loss: 0.0356 - val_accuracy:
0.9884
Epoch 8/10
981/981 [==============================] - 70s 71ms/step - loss: 0.0723 - accuracy: 0.9770 - val_loss: 0.0388 - val_accuracy:
0.9885
Epoch 9/10
981/981 [==============================] - 71s 72ms/step - loss: 0.0687 - accuracy: 0.9785 - val_loss: 0.0282 - val_accuracy:
0.9915
Epoch 10/10
981/981 [==============================] - 68s 69ms/step - loss: 0.0662 - accuracy: 0.9807 - val_loss: 0.0213 - val_accuracy:
0.9938
```

Fig. 7. training model

IV. Results and Analysis

The CNN model with VGG16 architecture on training for 20 epochs the training accuracy was 78% and validation accuracy was 75%. The model's accuracy on the test dataset is 53%

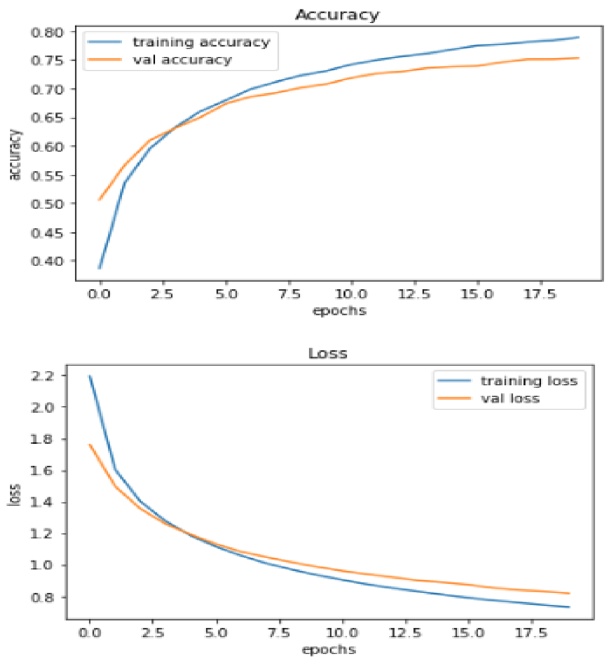
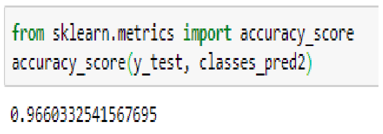


Fig. 8. Training accuracy,loss vs Validation accuracy,loss

The CNN model architecture 2 on training for 10 epochs the training accuracy was 98% and validation accuracy was 99%. The model's accuracy on the test dataset is 96%



```
from sklearn.metrics import accuracy_score
accuracy_score(y_test, classes_pred2)

0.9660332541567695
```

Fig. 9. Accuracy of the model on the test datasetl

V. Conclusion

In this project, a deep learning convolutional neural network was trained on a German traffic sign dataset to build a classification model that could predict the traffic sign and the model was trained using Tensor flow on the two different CNN architectures and achieved an accuracy of 96%. The trained model is saved, this model can be used to recognize traffic sign from the web images.

References

[1] H.-K. Kim, Y.-N. Shin, S.-g. Kuk, J. H. Park, and H.-Y. Jung, “Nighttime traffic light detection based on svm with geometric moment features,” Intl. Journal of comput. and Information Engineering, vol. 7, no. 4, pp. 472–475, 2013.

[2] Haojie Li, Fuming Sun, Lijuan Liu, Ling Wang, “A novel traffic sign detection method via color segmentation and robust shape matching”, Journal of Neurocomputing, Elsevier publication, Volume 169, pp. 77- 88, ISSN 0925-2312, 2015

[3] Y. -C. Chiu, H. -Y. Lin and W. -L. Tai, "Implementation and Evaluation of CNN Based Traffic Sign Detection with Different Resolutions," 2019 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS), 2019, pp. 1-2, doi: 10.1109/ISPACS48206.2019.8986319.

[4] arXiv:2003.03253 [cs.LG] Lihi Shiloh-Perl, Raja Giryes "Introduction to deep learning"

[5] arXiv:1511.08458 [cs.NE] Keiron O’Shea and Ryan Nash "An Introduction to Convolutional Neural Networks"

[6] https://www.kaggle.com/datasets/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign

[7] arXiv:1409.1556 [cs.CV] Karen Simonyan Andrew Zisserman "very deep convolutional networks for large-scale image recognition"

[8] Kwetishe Joro Danjuma "Performance Evaluation of Machine Learning Algorithms"