© May 2021| IJIRT | Volume 7 Issue 12 | ISSN: 2349-6002

A traffic sign classifier model using Sage maker

Arpit Seth,[1] Dr. VijayKumar A[2]
[1]*MCA Scholar, School of CS & IT, Dept. of MCA, Jain (Deemed-to-be University) 560069*
[2]*Professor, School of CS & IT, Dept. of MCA, Jain (Deemed-to-be University) 560069*

***Abstract* - Traffic sign recognition is vital in driver help systems that relieve the driver's work, still as intelligent autonomous vehicles. We have a tendency to purpose during this paper Traffic sign classification is a crucial task for self-driving cars. During this research, a Deep Network referred to as LeNet are going to be used for traffic sign pictures classification. The dataset contains forty-three totally different categories of pictures. This structure is split into 2 parts: traffic sign identification and traffic sign classification. ADASs area unit being designed for a spread of functions, together with communications, road mark detection, road sign recognition, and pedestrian detection. This structure is split into 2 parts: traffic sign identification and traffic sign classification. The strategies for detection and recognizing traffic signals area unit mentioned during this article. For traffic sign detection and recognition, varied strategies like colour segmentation and also the RGB to HSI model area unit used. HOG operate, form which means, and different factors contribute to recognition.**

INTRODUCTION

In intelligent autonomous vehicles and driver help systems, traffic sign recognition contains a ton of economic potential. It is not possible to boost traffic quality and protection while not properly applying and maintaining road traffic signs, traffic signals, and road markings. Amazon Sage Maker could be a cloud machine-learning platform that lets developers produce, practice, and deploy machine-learning models. We have a tendency to purpose this model in Amazon Sage Maker. Sage Maker contains a range of advantages, together with a lot of process power (different CPUs Associate in Nursing GPUs with parallel computing) and therefore the ability to deploy the model as an end. The whole analysis is finished in Sage Maker's notebook, whereas coaching and predicting were done by business a lot of powerful GPU instances. For the sake of completeness, this text can embody the whole pipeline from pre-processing to end readying.

Recognition of traffic signals has been tackled by a range of classification techniques, starting from basic model matching (e.g., pooling, flattening) to advanced Machine learning techniques (e.g. support vector machines, boosting, dense neural network etc.).

Furthermore, increasing traffic sign analysis from isolated frames to feature maps can facilitate to greatly cut back the amount of false alarms whereas conjointly rising the exactitude and accuracy of the detection and recognition method.

In this paper dataset is divided into three parts that is training dataset, testing dataset, and validation dataset. Other aspects, like the creation of Associate in nursing automatic pilot to spot road boundaries or obstacles within the vehicle's route, like alternative vehicles or pedestrians, are the topic of alternative analysis teams. Accidents could happen for a range of reasons, together with drivers failing to acknowledge a proof in time or failing to listen at a vital moment. Drivers pay very little attention to traffic signals and concentrate on driving in poor climatic conditions like significant rain showers, fog, or downfall. The role of control, moreover, as guiding and advising drivers, needs automatic traffic sign recognition. Traffic signs, in general, offer the motive force with crucial details for secure and effective navigation.

MOTIVATION

In order to better parse and recognize the highway, self-driving cars need traffic sign recognition. Similarly, in order to assist and protect drivers, "driver alert" systems in cars must consider the roadway around them. Computer vision and deep learning can help with a variety of issues, including traffic sign recognition. Our goal is to build a multi classifier model based on deep learning to classify various traffic signs.

LITERATURE REVIEW

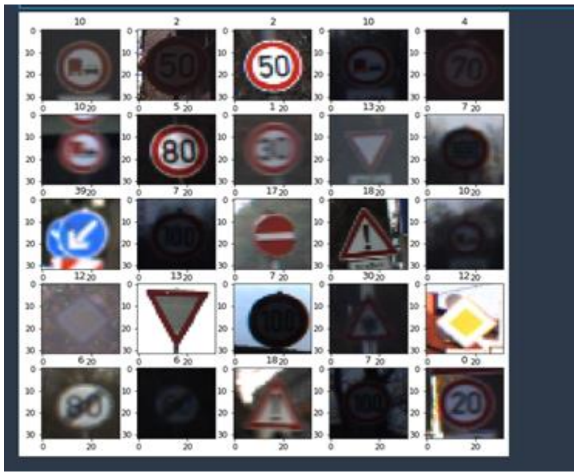
The three most popular types of traffic sign detection systems are color-based, shape-based, and learning-based. Meanwhile, Liu introduces two new modes: color and shape-based methods and LIDAR-based methods. Since it achieves a detection rate of more than 90%, the learning-based detection method is the most powerful of these approaches. On the other hand, the most recent algorithms are still a long way from being a real-time classification system. Some researchers have obtained good results by using the HSI color space instead of RGB. For traffic sign detection, a new color space called Eigen color was proposed based on Karhunen-Loeve (KL). [1] This paper proposes the "German Traffic Sign Recognition Benchmark" dataset and competition, as well as their design and analysis. The results of the competition show that cutting-edge machine learning algorithms excel at the challenging task of traffic sign recognition. [2] The baseline algorithms considered in this paper are the Viola-Jones detector based on Haar features and a linear classifier based on HOG descriptors, which are two of the most common detection approaches. [3] This paper focuses on real-time traffic sign recognition, which involves deciding in a short period of time which type of traffic sign appears in which region of an input image. To accomplish this, a two-module architecture (detection and classification modules) is proposed. The color probability model in the detection module converts the input color image into probability maps. The road sign recommendations are then derived by identifying the most secure extreme areas on the charts. [4] In this research, the HIS color model is used to detect the traffic sign, which is accompanied by circle detection. The precise sign area cannot be determined from the color-detected regions. This approach traces the edge of interested regions to obtain their contours after morphologic operations.

DATASET DESCRIPTION

We will use the dataset to train our own custom traffic sign classifier (GTSRB).
Here is a quick rundown of the data:

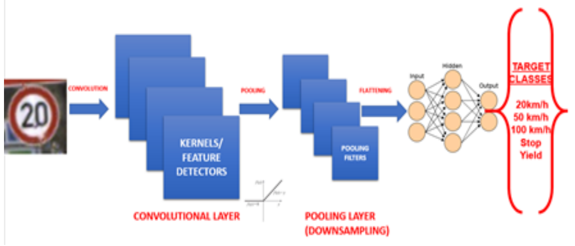
- The sizes of each picture are slightly different.
- There are 39209 training examples in total.
- The total number of test examples is 12630.
- There are 43 classes total.

The images range in scale from 20x20 to 70x70 pixels and have three channels: RGB. AWS S3 storage will be used to store all of the data. And all of the research and training is done in Sage Maker instances on the dataset. The first thing we have to do is to resize all the images to 32x32x3 and read them into a NumPy array as training features.
All of the classes go through the same procedure. We stored the data in pickle form to avoid having to do pre-processing again if the notebook was disconnected. We can easily load the processed data from pickle the next time we need it.



PROPOSED METHODOLOGY

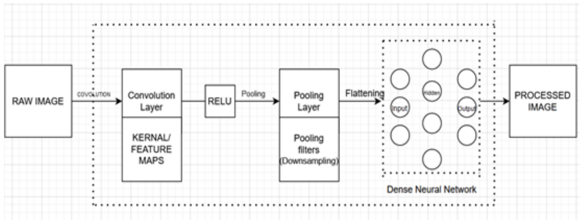
In this paper, we use LeNet CNN architecture to build a multi classifier model based on deep learning to classify various traffic signs. Image classifiers function by predicting the type of objects in a given image. As a classification model, we use a convolutional neural network (CNN). In general, CNN outperforms traditional neural networks and other machine learning models in classifying images on massive datasets. A Deep Network known as LeNet will be used for traffic sign images classification.
Below are the steps we followed to option for our goals and to achieve the final Le Net multi classifier model for traffic sign prediction



So what we do is that we collect many of these neurons together in a systematic format, in a layered fashion. We have inputs. Then we have bunch of neurons here in the hidden layer. And we will connect all the neurons in one layer and will be fully connected to all the neurons in the subsequent layer this is called dense artificial neural network or fully connected artificial neural network.

So, let us assume that we have a bunch of images here. And these are the images from the traffic sign data set. Here I have all the different classes. So, what we do is that we obtain a dense, fully connected artificial neural network. But to deal with images, what we do is that we add an additional processing step beforehand.

We have the image. Then we perform convolution first. And then we will apply non-linearity. We try to add non-linearity after convolution layer in CNN, by introducing RELU activation function or rectified linear unit activation function after that the next step is pooling. In pooling layer, we are trying to compress all these feature maps, all the outputs that we take, or we get out of the first layer. In pooling layers results are exact same but in reduced version of that information. And you will see that the size here will become reduced when we apply the pooling function. We do afterwards, once we have these pooling filters, we simply apply flattening. So, we take all the pixels in here. We flatten them up. And now model will be ready to take all these inputs and feed them to dense or fully connected artificial networks.



Below is the process done to train the model.

1 Convolution Layer

First step for identification is convolutional layer. These convolutional are Kernels or feature detectors. These kernels or feature detectors just filters the image to extract features out of the original image. These kernels with the help of some 2X2 matrix filters the image by performing some algorithm functions on original image. Convolutions simply use the kernel matrix to scan a given image and apply a filter to obtain a certain effect. Where you take an image, you apply kernel or a feature detector to extract features out of my image. So, what we do is after we apply these kernels or feature detectors, we actually generate feature maps. Feature maps are kind of different variations of the original image, as we are getting the image and we can perform sharpening to the image or blurring to the image. We will creating all these different feature maps to extract features out of original image. With the help of image kernel, we have here is a matrix that is used to apply effects such as blurring and sharpening. And kernels are used in machine learning for feature extraction to select the most important pixels in the image.

2 RELU

RELU Layers are used in the function map to incorporate non-linearity. It also improves the function map's sparsely, or how dispersed it is. In contrast to the sigmoid function, the RELU's gradient does not vanish as x increases.

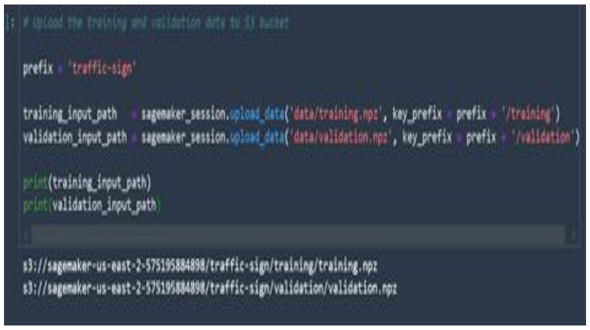
3 Pooling

To minimize feature map dimensionality, pooling or down sampling layers are put after convolutional layers. This increases computational efficiency while maintaining feature preservation. By preventing over fitting, pooling aids the model's generalization. The pooled function map will remain unchanged if one of the pixels is moved. In a feature map, max pooling works by maintaining the maximum feature answer within a given sample size.

The max pooling layer reduces the size of the previous layer's performance, making training easier. To avoid over fitting, a drop out layer is also used.

The sagemaker.tensorflow.TensorFlow estimator handles locating the script mode container, uploading script to a S3 location and creating a Sage Maker training job.

We saved them as .npz files and uploaded them to a s3 bucket so the model could access the data.

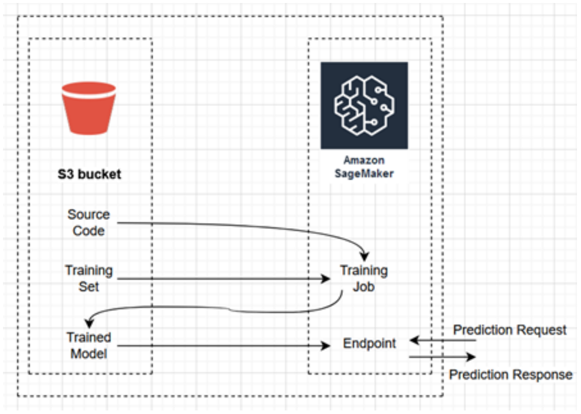


```
# Upload the training and validation data to S3 bucket

prefix = 'traffic-sign'

training_input_path   = sagemaker_session.upload_data('data/training.npz', key_prefix = prefix + '/training')
validation_input_path = sagemaker_session.upload_data('data/validation.npz', key_prefix = prefix + '/validation')

print(training_input_path)
print(validation_input_path)

s3://sagemaker-us-east-2-575195884898/traffic-sign/training/training.npz
s3://sagemaker-us-east-2-575195884898/traffic-sign/validation/validation.npz
```

The training script is a python file that can be run on its own. Sage Maker will run this script on a given instance during the training job.

CORPORATE IMPLEMENTATION OF MACHINE LEARNING AND CLOUD COMPUTING

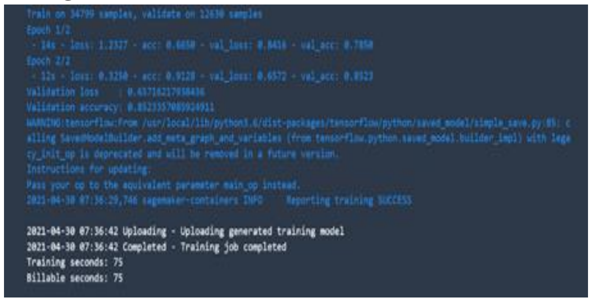
Machine learning and cloud computing are needed for data storage, processing, and management in any field. As part of cloud computing, we provide AWS services for data storage and administration. The CSV data will continue to save the data in the same register. The files in AWS are used to collect and update data for the model we use on a regular basis. The AWS EC2 and S3 architecture, which helps us store data and map data to the IAM service model, is shown below. With the assistance of built-in algorithms, AWS Sage maker aids in the training of a prediction model. With, you can use AWS sage creator right away. Sage Maker and Boto 3 must be imported into the notebook. Boto3 is the Python version of the Amazon Web Services (AWS) Software Development Kit (SDK). Boto3 enables Python programmers to create applications that utilizes Amazon S3 and Amazon EC2. Then we it's set up a Sage maker session. Sage Maker aids in the storage of data or training datasets in S3, as well as the training of models in S3 so that they can be used later. The key benefit of Sage Maker is that it can train an unlimited number of models from which the best or most reliable model can be selected. The Linear Learner is a supervised learning algorithm for fitting a line to the data for training.



RESULT

This is the anticipated result. This segment will explain the results and sample codes that we use in this method. We used a multi classifier model in this architecture that could accurately predict the class of pre-processed images that are present in a given image.

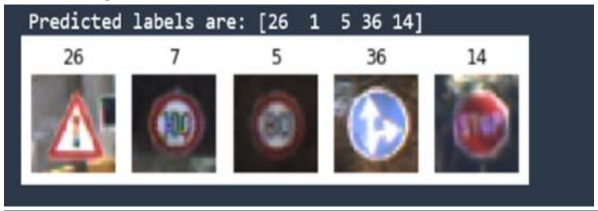
We can describe the model's training job and train it. An ml. p2xlarge instance was used. The validation accuracy after 2 epochs is 0. 8523.It took the instance 76 seconds to complete the task with the accuracy on testing set is 0.923.



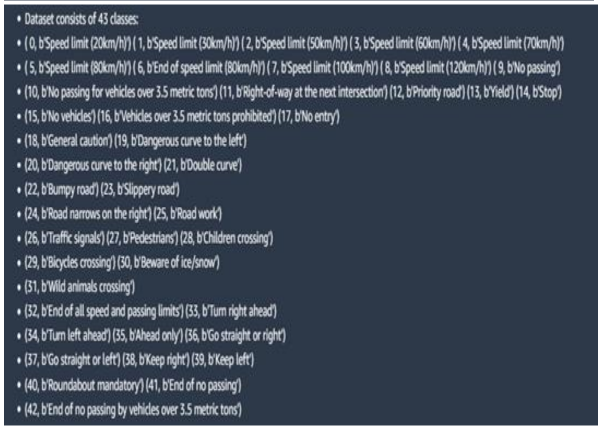
```
Train on 34799 samples, validate on 12630 samples
Epoch 1/2
 - 14s - loss: 1.2327 - acc: 0.6850 - val_loss: 0.8416 - val_acc: 0.7850
Epoch 2/2
 - 12s - loss: 0.3250 - acc: 0.9128 - val_loss: 0.6572 - val_acc: 0.8523
Validation loss      : 0.43716217938436
Validation accuracy: 0.8523357080924911
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/saved_model/simple_save.py:85: c
alling SavedModelBuilder.add_meta_graph_and_variables (from tensorflow.python.saved_model.builder_impl) with lega
cy_init_op is deprecated and will be removed in a future version.
Instructions for updating:
Pass your op to the equivalent parameter main_op instead.
2021-04-30 07:36:29,746 sagemaker-containers INFO     Reporting training SUCCESS

2021-04-30 07:36:42 Uploading - Uploading generated training model
2021-04-30 07:36:42 Completed - Training job completed
Training seconds: 75
Billable seconds: 75
```

Final Output:

The below is the sign recognition of 5 samples from the testing dataset out of 43 classes.



```
Predicted labels are: [26  1  5 36 14]
```



- Dataset consists of 43 classes:
- (0, b'Speed limit (20km/h)') (1, b'Speed limit (30km/h)') (2, b'Speed limit (50km/h)') (3, b'Speed limit (60km/h)') (4, b'Speed limit (70km/h)')
- (5, b'Speed limit (80km/h)') (6, b'End of speed limit (80km/h)') (7, b'Speed limit (100km/h)') (8, b'Speed limit (120km/h)') (9, b'No passing')
- (10, b'No passing for vehicles over 3.5 metric tons') (11, b'Right-of-way at the next intersection') (12, b'Priority road') (13, b'Yield') (14, b'Stop')
- (15, b'No vehicles') (16, b'Vehicles over 3.5 metric tons prohibited') (17, b'No entry')
- (18, b'General caution') (19, b'Dangerous curve to the left')
- (20, b'Dangerous curve to the right') (21, b'Double curve')
- (22, b'Bumpy road') (23, b'Slippery road')
- (24, b'Road narrows on the right') (25, b'Road work')
- (26, b'Traffic signals') (27, b'Pedestrians') (28, b'Children crossing')
- (29, b'Bicycles crossing') (30, b'Beware of ice/snow')
- (31, b'Wild animals crossing')
- (32, b'End of all speed and passing limits') (33, b'Turn right ahead')
- (34, b'Turn left ahead') (35, b'Ahead only') (36, b'Go straight or right')
- (37, b'Go straight or left') (38, b'Keep right') (39, b'Keep left')
- (40, b'Roundabout mandatory') (41, b'End of no passing')
- (42, b'End of no passing by vehicles over 3.5 metric tons')

CONCLUTION

To train a model for traffic sign classification, we have used the new Tensor Flow system.

Pre-processing, model development, training, estimation, and endpoint deployment are all part of the pipeline. The testing accuracy is 0.923, while the validation accuracy is 0.8523. The majority of the classes have an accuracy of greater than 90%.

According to the findings, the proposed model is able to resolve certain flaws.

The following are some suggestions for improvement:

1. This project lacks a user interface.
2. Other variables may be used to improve prediction accuracy.

REFERENCE

[1] M. S. J. S. a. I. J. Stallkamp, "The German traffic sign recognition benchmark: a multi-class classification competition," IEEE, 2011.

[2] J. S. J. S. S. a. C. I. S. Houben, "Detection of traffic signs in real-world images: The German traffic sign detection benchmark," IEEE, pp. 1-8, 2013.

[3] H. L. H. X. a. F. W. Yi Yang, "Towards Real-Time Traffic Sign Detection and Classification," IEEE, 2014.

[4] W. L. Kai Li, "Traffic indication symbols recognition with shape contex," IEEE, 2011.