Traffic Sign Classification Using Deep Learning

Omkar Prabhakar Ghadage.
Vinay Kumar Singh.
School of Computer Science and Technology
Lovely Professional University,Jalandhar, India.
Email : omkarghadage380@gmail.com
ctvinay11@gmail.com

1. Abstract:

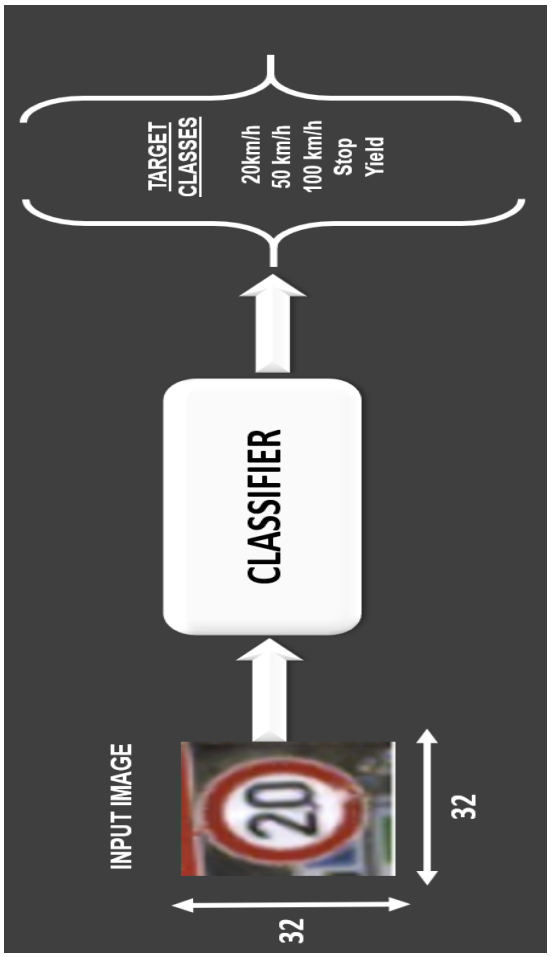
Artificial Intelligence has grown in a huge amount from past decades. Nowadays people rely on automatic and more reliable solutions provided by this immensely popular field of Artificial Intelligence called as Machine Learning. Machine Learning is technique to manipulate the most optimal solutions based on statistical, pictorial representation, etc. based data available to produce algorithms which will lead us to the optimal result that we desire. Automatic and Intelligent Road traffic handling is one of the research field which can be accomplished by use of certain machine learning algorithms and the data made available by Satellite images. As in this time of self automation we are thinking about self driving vehicles to run on the road we have to considerate about all the traffic conditions. In addition to that we have to make sure that our system learns about all the rules of driving. Traffic rules and symbols are the main part to make sure our Vehicle which is running because of artificial intelligence will understand all diversions and signals provided to run safely. In this article I am trying to convey the newer model to recognize all traffic signals and make sure that our self driving vehicle runs with real safety. While consideration of recognition of signs we can use feature reckoning(Extrecting) and image processing methodologies available similar to thresholding operation, Gradient and Hough's transformation. High resolution panchromatic sign imagery are being used for extracting the data to understand the targets of vehicles and present them with the Convolutional Neural Network Technique. With the number of iterations this model can be recognizing signs of traffic with almost 99% of accuracy. Possibility of this model working perfectly and this phenomenon shows that the methods used have competence to deal with the problem that we are trying to solve. In further upcoming topics I am going to convey this model completely with theoretical as well as pictorial representation.

Keyword : Traffic Signs, Convolutional Neural Network(CNN), data serialization, Cross validation, Gradient Thresholding operation, Intelligence Traffic Monitoring.

2. Introduction:

In recent times as the studies were made to develop the self driving cars it is essential for us to see that it follows all the subtle rules of driving for betterment of human safety. Being the

developer we can't allow any casualty to the human safety. For better safe driving vehicles have to follow the traffic rules made by the standard associations. So now we can reckon the traffic sign classification is an important task for self driving cars. In the project that we are concerned about; we are going to implement and train Deep Neural network called as Convolutional Neural Network to classify traffic sign images. The dataset we are going to work on for classification is consist of 43 different classes of traffic sign images.



The dataset we are using is consist of traffic sign images and those images are of 32*32 pixels. Basically the colored images in the dataset consist of 3 RGB channels for red, green,blue colors.

3. Dividing Dataset into Training, Validation and Testing:

We need to partition the data so that we can train our model for result optimization. For training purpose we are going to use more samples rather than testing. As we know the testing dataset will never be exposed to the Model that we are going to train as we don't want our model to memorize the data but we just want it to perform generalization on it. Here we are going to use 80% of the data for training purpose and the remaining 20% will be used for testing. Sometimes it is better to devide data into 60% training, 20% validation and 20% testing purpose. By using a training set we are going to calculate gradient as well as weight. Where validation set is used for cross-validation which is performed to assess training quality as training proceeds. Cross validation is used to remove the problem of overfitting which occurs when the algorithm focuses on training set details at the cost of losing generalization ability.

And finally the testing dataset is used for testing the trained model. When both the training as well as testing error are high then that model is called an underfitting model and when training error is really less but testing error is high then that model is said to be an overfitting model. We are using the cross validation concept here to resolve these phenomenons.

4. Data Normalization:

Our data must be prepared before you can build models. The data preparation process can involve three steps: data selection, data preprocessing and data transformation.In this section we will discover a simple data transformation method that we can apply to our data in Python using scikit-learn. Normalization

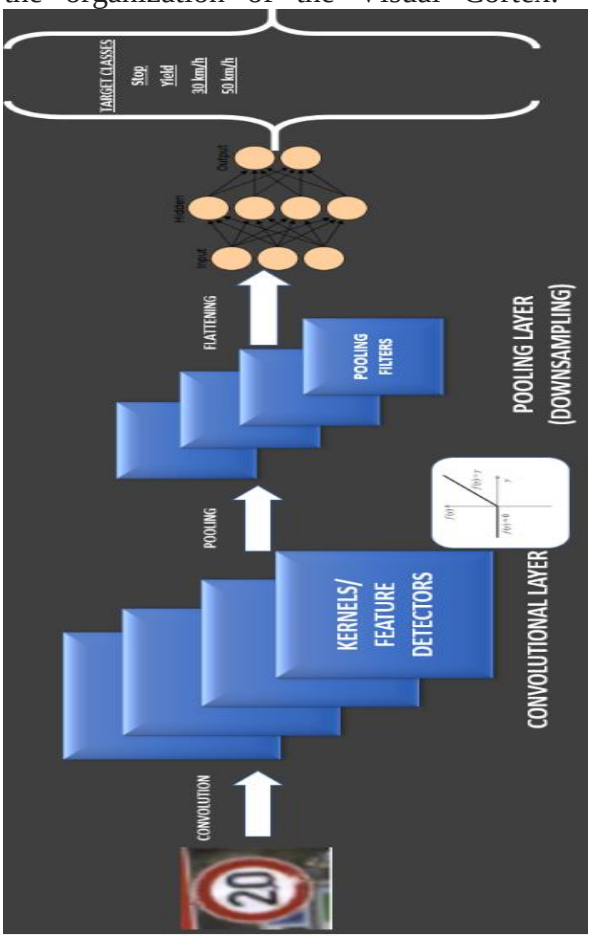
refers to rescaling real valued numeric attributes into the range 0 and 1. It is useful to scale the input attributes for a model that relies on the magnitude of values, such as distance measures used in k-nearest neighbors and in the preparation of coefficients in regression.

Normalizer works on the rows, not the columns! I find that very unintuitive. It's easy to miss this information in the docs. By default, L2 normalization is applied to each observation so that the values in a row have a unit norm. Unit norm with L2 means that if each element were squared and summed, the total would equal 1. Alternatively, L1 (aka taxicab or Manhattan) normalization can be applied instead of L2 normalization.

Normalizer does transform all the features to values between -1 and 1. In our example, normal_big ends up with all its values transformed to .9999999. In most cases one of the other preprocessing tools above will be more helpful. Again, scikit-learn's Normalizer works on the rows, not the columns.

5. Understand the theory and intuition behind convolutional neural network:

A Convolutional Neural Network (CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex.



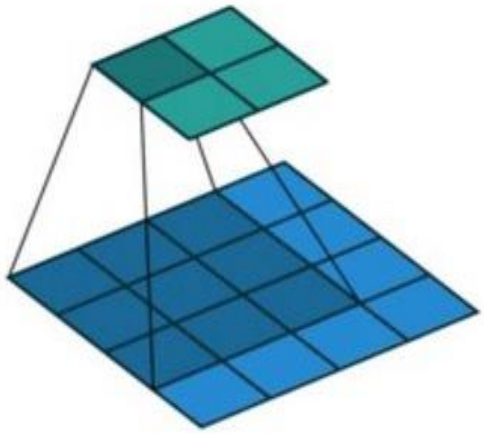
Dropout: The technique that improves accuracy by adding dropout. Dropout refers to dropping out units in a neural network.

Neurons develop co-dependency amongst each other during training. Dropout is a regularization technique for reducing overfitting in the neural network.

6. Build a deep convolutional network model using Keras:

Deep learning is used to classify images to build a convolutional neural network (CNN). The Keras library in

Python makes it pretty simple to build a CNN. A convolution multiplies a matrix of pixels with a filter matrix or 'kernel' and sums up the multiplication values. Then the convolution slides over to the next pixel and repeats the same process until all the image pixels have been covered.



(a)Loading the Datasets: The Traffic-sign dataset is conveniently provided to us as part of the Keras library, so we can easily load the dataset. There are 34799 images in the datasets. 80% of them are given for training and 20% are given for testing.

(b) Exploratory data analysis: Now let's take a look at one of the images in our dataset to see what we are working with. We will plot the first image in our dataset and check its size using the 'shape' function. By default, the shape of every image in the Traffic-sign dataset is 32 x 32 pixels. so CNN will be able to run over each image pretty quickly.

(C) Data pre-processing: Next, we need to reshape our dataset inputs (X_train and X_test) to the shape that our model expects when we train the model. The first number is the number of images (80% for X_train and 20% for X_test). Then comes the shape of each image (32x32). The last number is 1, which signifies that the images are grayscale.

7. Building and Training the model:

The model type that we are using is Sequential. Sequential is the easiest way to build a model in Keras. It allows us to build a model layer by layer.

We use the add() function to add layers to our model. The activation function we will be using for our first 2 layers is the ReLU, or Rectified Linear Activation. This activation function has been proven to work well in neural networks. In between the Conv2D layers and the dense layer, there is a 'Flatten' layer. Flatten serves as a connection between the convolution and dense layers. Dense is the layer type we will use in our output layer. Dense is a standard layer type that is used in many cases for neural networks. The activation is 'softmax'. Softmax makes the output sum up to 1 so the output can be interpreted as the kind of probabilities. The model will then make its prediction based on which option has the highest probability.

Compiling the model takes three parameters: optimizer, loss and metrics.The optimizer controls the learning rate. We will be using Adam as our optimizer. Adam is generally a good optimizer to use for many cases. The Adam optimizer adjusts the learning rate throughout training. The learning rate determines how fast the optimal weights for the model are calculated. A smaller learning rate may lead to more accurate weights (up to a certain point), but the time

it takes to compute the weights will be longer.Here we will use the method called categorical_crossentropy for our loss function. This is the most common choice for classification. A lower score indicates that the model is performing better. we are using the accuracy metric to see the accuracy score on the validation set when we train the model.

To train, we will use the 'fit()' function on our model with the following parameters: training data (train_X), target data (train_y), validation data, and the number of epochs. For our validation data, we will use the test set provided to us in our dataset, which we have split into X_test and y_test. After 5 epochs, we have gotten to 97.57% accuracy on our validation set. That's a very good start! Congrats, you have now built a CNN.

8. Acknowledgement:

This research paper is submitted to the School of Computer Science, Lovely Professional University, Phagwara. This research paper is considered as an assignment for the course Machine Learning Foundation and prepared keeping that in concern. The faculty of respective subject Ms. Pooja Rana have guided me throughout completion of this term paper by providing exact information of patterns that needed to follow while writing the term paper. As well as Mr. Sanjay Kumar sir helped me to understand CNN as well as the gray color transformation which helped me alot to write this term paper effectively. So I would like to show my gratitude to Mr. Sanjay Kumar sir and Ms Pooja Rana ma'am. Although if there is any error in this document then it's of my own and should not tarnish the reputation of an esteemed person.