TRAFFIC SIGN DETECTION USING DEEP LEARNING BY GROUP 36

Sepehr Foroughi Shafiei, Hafiza Rimsha Rauf, Yashdeep Singh

University of California San Diego, La Jolla, CA 92093-0238

ABSTRACT

This project includes implementation and comparison of deep learning based architectures for classification of forty three different types of traffic signals. Popular CNN based architectures were evaluated on the basis of classification accuracy and the prediction speed. Based on the results obtained, a simple yet effective Convolutional Neural Network(CNN) architecture was proposed that gives high accuracy at a great prediction speed.

1. INTRODUCTION

Automotive industry is one of the largest industries in the world. In 2019, ninety-two million motor vehicles were produced in the world, with the United States itself producing over 2.5 million automobiles[1]. With the world moving towards self-driving vehicles, more than 80 companies are testing over 1400 of them in the United States alone [2]. It was forecasted that there would be more than 10 million self-driving cars on road by 2020 [3]. This prediction was quite exciting but quite clearly, did not happen. There are numerous reasons behind it, but the main issue is safety. Self-driving cars need to identify all the details on the road with extreme precision and accuracy including traffic signs not just for the passenger, but the fellow pedestrians as well.

We began by studying the dataset at hand. Data preprocessing was done to make the data ready to be used. First, data was augmented to equalize class imbalance and standardization was applied to it. Models were trained and tested on this preprocessed data. Convolutional Neural Network (CNN) is one of the powerful and the most popular network for classifying images. Most of the models we implemented use CNN. The first model trained was vanilla VGG-19 and transfer learning was applied to predict images. Then, Resnet50 and Densenet121 architectures were used with ImageNet[4] weights initialization. Further, Inception-V1 i.e Googlenet and RNN were implemented from scratch. Finally, observing the trends, we proposed a new implementation using CNN. In the end, we showed that our novel model has a higher accuracy for predicting traffic signs in comparison to other models we discussed in this project. Throughout this project, our main focus was to increase complexity step by step and figure out what works the best for this dataset.

2. RELATED WORK

Numerous studies and researches have been done in the field of Machine Learning and Deep learning, especially involving self-driving cars. CNN is one of the famous architectures in image classification and is widely used in image classification. In this section, we share a few papers and projects that have inspired us the most, and helped us understand the problem.

We started off by finding a solution to the problem using Support Vector Machines (SVM) [5]. In this research, automatic sign detection was integrated on roads, and their model was able to detect all types of signs like circular, triangular, rectangular, and octagonal shapes. Owing to the high success rate on the final predictions, they showed that SVM could be a good model for this purpose.

In the next research that we studied[6], 100000 street images of Tencent city in China, and 30000 traffic sign images were collected from them. The unique fact about their benchmark is that they were able to gather all the pictures of traffic signs in the different weather forecasts. This can be a great help in terms of real predictions. They used CNN architecture to predict and detection of traffic signs. They tried not to just focus on the images of signs, and they tried to have images that the car views while driving, and they ran their model for those images. This paper gave us more insight into how we should choose our dataset wisely and opened our ways to think more about CNN.

Furthermore, we studied the project that offered a novel CNN architecture [7]. They call their model OverFeat. They figured that the multiscale and sliding window approach would be very efficient for implementing CNN. According to this study, detecting object boundaries should be the main target to design the CNN model. CNN can learn the task in terms of classification simultaneously. After going through this research, we shifted our attention to how deep our model is and how its accuracy can be improved based on the sign's boundary. This paper won the localization task of the ImageNet Large Scale Visual Recognition challenge and motivated us to choose CNN for our implementation.

Finally, in order to get more sense of the CNN network, we referred this research study[8]. According to this study, most Convolutional Network architecture have the same design and structure, but the dimension of filters in each convolutional layer play a significant role in classifying images. Therefore, different dimension filters were experimented in their architecture. The dimensions that they explored were 3×3, 5×5, 9×9, 13×13, 15×15, 19×19, 23×23, 25×25 and 31×31. As a result, they were able to experiment with different sizes, and they compared the accuracy result based on different filter sizes. Through this paper, we realized that we have to keep factors like dimension of filter size or hyperparameters in mind as well when designing our model.

3. DATASET

The dataset used for this project is the German Traffic Sign Detection benchmark . This data has a total of 51,830 images each of dimension (32 x 32 x 3), 3 signifying the RGB channels of the colored images. The dataset was further split into 34799 training, 4410 validation and 12630 test images.

4. METHODOLOGY

In this section, the data preprocessing techniques as well as implemented CNN based architectures alongside training details and evaluation metrics are discussed in detail.

4.1. Data Preprocessing

Upon some initial analysis, it was noticed that there was high class imbalance in the dataset. Figure 1a shows the data count per class. If the model is trained on this dataset, it poses a problem as this makes the model biased towards the class with higher data frequency. To resolve this, data augmentation was performed. Images from low frequency classes were picked up and random rotation and brightness variation was performed. Eventually, data was normalized for all classes and results are represented in 1b. Then, standardization was performed on the data to normalize its mean and make it unit variance.

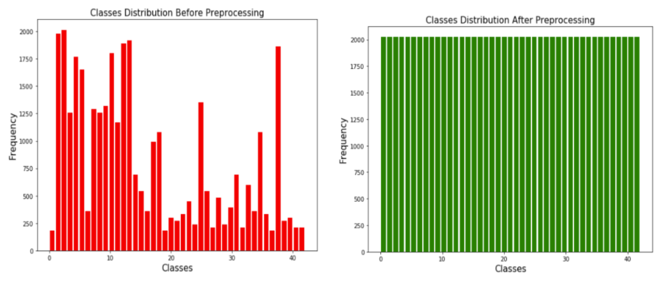


Fig. 1: a) Class distribution in original training data. b) Class distribution after data preprocessing

4.2. Models Implemented

In this section, we discuss the different models implemented and the proposed novel implementation as well. As said earlier, we began by taking a simple CNN architecture VGG-19. We then went on to selecting deeper neural networks with skip connections like ResNet and DenseNet. Additionally, we explored the performance of Inception Module in GoogleNet. We also implemented RNN to see if the image pixels have any sequential relation which can be exploited to boost the performance. Seeing the results, we proposed a novel architecture.

4.2.1. VGG-19

We started off with a simpler highway architecture VGG-19 that was trained from scratch to classify traffic signs. It uses only 19 layers and has a relatively simpler architecture as it does not use inter-layer connections.

4.2.2. ResNet50 and DenseNet121

Next, we performed transfer learning on a popular image classification architecture ResNet50 pretrained on ImageNet The actual motivation to use ResNet50 which consist of 50 layers was to strengthen feature propagation across different layers and encourage feature reuse. It uses skip connections to add outputs of previous layers to the current layer which helps it reduce the residual error between the preceding and the current layer. Secondly, we tuned DenseNet121 (total 121 layers) architecture pretrained on ImageNet dataset for traffic sign detection. DenseNet is the logical extension of the concept used in building ResNet architecture. The feature maps of all preceding layers are used as input to the current layer and its output feature maps are used as input to the subsequent layers. It connects all layers in a feed forward manner to enhance feature learning and help the model converge faster.

4.2.3. Inception-V1

Inception-V1 architecture uses inception modules which has convolution filters of different kernel sizes at a single layer and concatenate all of the output feature maps before passing it to the next layer. This architecture helps the classifier to simultaneously capture global and local feature details of the input image making it robust to size variations of the object of interest in the image. Traffic signs in the images of training data have large variations in their size. For instance, images of the traffic signs might be at different distances from the sign board making it appear larger or smaller than the actual image. Considering this fact, we trained Inception-V1 architecture from scratch for German traffic sign detection benchmark .

4.2.4. *Recurrent Neural Network*

Another important neural network architecture is Recurrent Neural Network which uses hidden states of previous layers as inputs which makes it a suitable choice for sequential inputs. This architecture was implemented to explore if there exist any sequential relationship in the pixels which can be utilized to boost accuracy.

4.2.5. *Proposed CNN architecture*

Finally, we implemented a convolution neural network architecture that has two stacks made up of three convolution layers followed by one average pooling layer and two fully connected layers. We used batch normalization after every convolution layer to avoid exploding and vanishing of gradients while back propagating across different layers. Dropout layer was applied after each stack to avoid overfitting. The figure of the developed architecture is shown in Fig. 2.

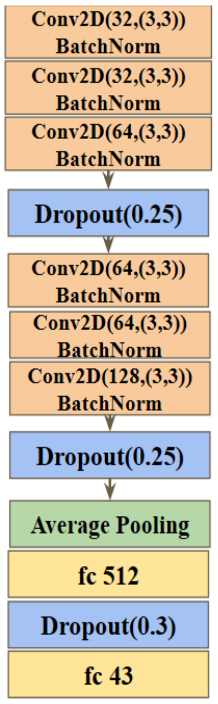


Fig. 2: **Block Diagram of proposed CNN architecture**. Fully connected layer is abbreviated as fc.

The first two convolution layers have 32 convolution filters with kernel size = 3. Grid Search over different kernel dimensions: 3, 5, 7, 9, 11, 19 and 31 for convolution layers was performed. The highest accuracy was achieved with kernel size = 3. ReLU activation function was used as a nonlinearity after each convolution layer which set negative input to zero. We did not use MaxPooling layer after every convolution layer to avoid losing spatial information of feature maps at early stages. Subsequent three convolution layers have 64 convolution filters each to increase the receptive field. Moreover, the last convolution layer has 128 convolution filters. This is followed by an average pooling layer to gather essence of all received feature maps. Its output is given to a fully connected hidden layer consisting of 512 neurons. Final output layer has 43 neurons with softmax as activation function to calculate the probability corresponding to 43 classes of GTSDB dataset . Most of this architecture was designed using Random Search for various parameters.

4.3. Training Details

All of architectures were trained using an adam optimizer at a learning rate = 0.001 s with batch size = 250 and epochs = 100. However, ResNet50 and DenseNet121 were trained and tuned at a slightly lower learning rate = 0.001 to avoid erasing pretrained weights. As this is a classification problem, categorical cross entropy loss function was used. All of the experiments were performed in python using Keras framework [9].

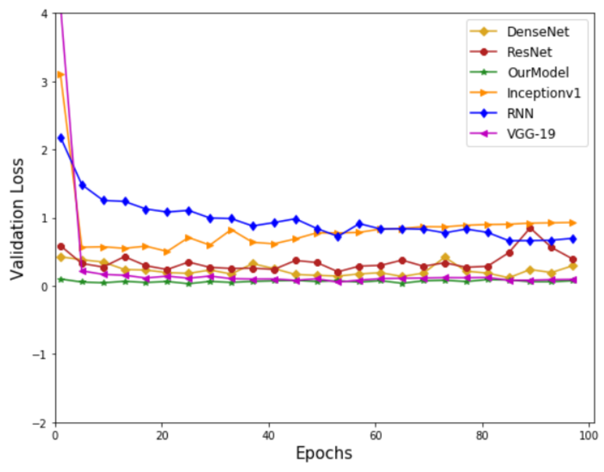
4.4. Evaluation Metrics

We evaluated the performance of all aforementioned models using classification accuracy. We also computed confusion matrix for predictions corresponding to each model to estimate precision and recall for each output class.

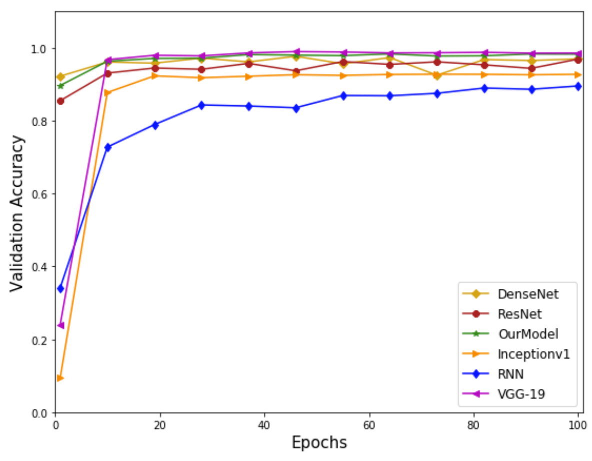
5. RESULTS

After executing the experiments for different models, best results for each model were noted. Figure 3b depicts validation accuracy achieved for each model over 100 epochs. First thing to be observed is that all models give an accuracy above 80% in the final training stage. VGG-19 gives the best performance (98.57%) amongst all the models and the new implementation(given by ”OurModel”) gives comparable accuracy (98.27%) as well. Another critical observation is that RNN gives worst performance (89.54%) on validation data. This might be because the image pixels do not show good sequential relationship. Further, it can be noticed that models using skip connections, i.e. ResNet and DenseNet and Inception Module, i.e. GoogleNet give decent accuracy as well. A key observation in this plot is the accuracy in initial training stages. Although VGG-19 gives slightly higher accuracy, its initial accuracy is quite low. On the other hand, our proposed model gives decent initial accuracy.

Next, we observed the trends in all training, validation and testing accuracy at respective best hyperparameters settings represented by Figure 4. Training accuracy goes to around 100% for all models. The highest test accuracy is given by VGG-19 (98.12%) and our proposed model gives an accuracy of (97.71%). As expected, RNN gives the worst performance (89%) and ResNet (94.8%), DenseNet (95.59%) and



(a) Validation Loss Curve



(b) Validation Accuracy Curve

Fig. 3: Validation Loss and Accuracy curve over different number of Epochs

GoogleNet (95.16%) give intermediate performance. The next criteria these models were judged on is evaluation time. Figure 5 represents evaluation time taken by each model to predict labels corresponding to the entire test dataset. Clearly, best results were achieved by our proposed solution performing around three times faster than VGG19. Furthermore, RNN not only gives the worst accuracy, it is also the slowest model. Additionally, we wanted to explore the roughly 2% of the test images for which our model did not perform well. A few examples are shown in Figure 6. It is quite evident that predicting signs for these images is quite difficult (even for the human eye) as they are either too blurry or too dark. Hence, our model does not perform well to classify these images.

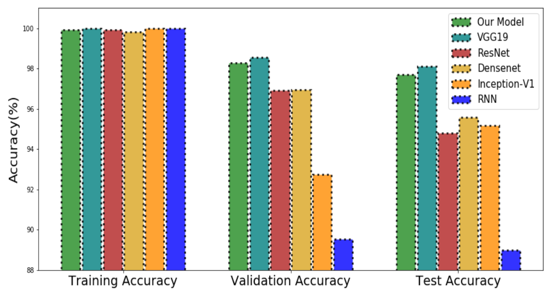


Fig. 4: a) Training accuracy b) Validation accuracy and c) Test accuracy for all models at their respective best hyperparameters setting.

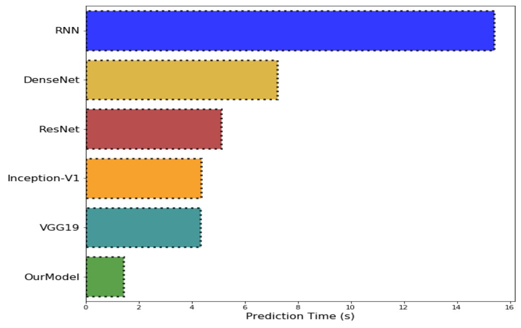


Fig. 5: Evaluation time for different models.

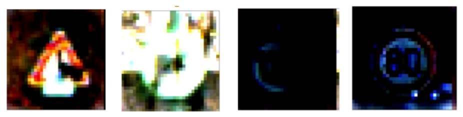


Fig. 6: Incorrectly Classified Examples.

6. CONCLUSION

In this project, various deep learning models were compared not just in terms of their classification accuracy, but also in terms of their prediction speed. Recurrent Neural Networks do not perform well as compared to Convolutional Neural Networks in terms of both accuracy and evaluation time for image classification purpose. One thing to be noted is that even simpler architectures namely, our proposed implementation and the VGG-19 outperform much more complicated, deep neural networks. Thus, performance of a particular model totally depends on the problem and data at hand and it is not necessary for deeper neural networks to always surpass the rest. Additionally, our proposed model gives the best response time while providing reasonable accuracy.

7. REFERENCES

[1] statista. car production: number of cars produced worldwide 2018, 2020.

[2] Darrell Etherington. Over 1,400 self-driving vehicles are now in testing by 80+ companies across the us, 2019.

[3] Kelsey Piper. It's 2020. where are our self-driving cars?, 2020.

[4] Imagenet large scale visual recognition challenge, 2017.

[5] S. Maldonado-Bascon, S. Lafuente-Arroyo, P. Gil-Jimenez, H. Gomez-Moreno, and F. Lopez-Ferreras. Road-sign detection and recognition based on support vector machines. *Trans. Intell. Transport. Sys.*, 8(2):264–278, June 2007.

[6] Zhe Zhu, Dun Liang, Songhai Zhang, Xiaolei Huang, Baoli Li, and Shimin Hu. Traffic-sign detection and classification in the wild. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[7] Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks, 2013.

[8] Valentyn Sichkar and Sergey Kolyubin. Effect of various dimension convolutional layer filters on traffic sign classification accuracy. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 19:546–552, 06 2019.

[9] Francois Chollet et al. Keras, 2015.

Individual contributions

The presentation and report were created and written collaboratively with the same amount of effort.

- **Hafiza Rimsha Rauf**

  - Implemented Inception-V1 and proposed architecture
  - Compiled results of each model and visualizations

- **Sepehr Foroughi Shafiei**

  - Implemented ResNet50 and DenseNet121 architectures

- **Yashdeep Singh**

  - Worked on the Data Preprocessing
  - Implemented RNN and VGG19 architectures

Critique Response

by Group 36

We thank our peers for going through our presentation and appreciate their response. Here, we try to clarify any pertaining doubts.

1 Response to Group 64

Impressive work! Group 36's project on Traffic Sign Detection using Deep Learning works with a dataset of 51,839 images of 43 classes. We thought the project was very thorough particularly in establishing the problem and motivation behind it. The effort put in with regards to comparing networks and gathering results was meticulous, and they are very clear about how they plan to improve further. Good job!
- Thank you for your appreciation.

Improvements/Unclear points:

- For data processing, it is a little odd how the gap between classes with larger frequency and smaller frequency was able to be filled up.
  - The main purpose of data augmentation is to solve class imbalance problem. We solved this issue by picking up an image randomly from class with low frequency, rotating them randomly between -15 to 15 degrees, changing their brightness randomly and appending these images to the training set. This step is repeated till it fills up the class with smaller frequency, thereby settling the class imbalance issue.
- In the PPT models slide, not every model was clearly shown (although in video models were very clearly shown)
  - This is because in the model slide, all models have been stacked one top of the other using animation. So, it'll look fine when you present it.

2 Response to Group 66

Firstly, we really love the background information and we learned that 2.5 million automobiles were produced in the United States while over 1,400 self-driving vehicles are now in testing by 80 companies, but in the future 10 million self-driving cars were predicted to be on the road by 2020 while there have been several fatal accidents about the autonomous vehicles, because this background shows the importance of the traffic sign detection for the autonomous vehicles. Secondly, about the dataset. We really appreciate the variety of the pictures they have picked, including pictures in really dark settings to mimic autonomous driving in night scenarios, blurry pictures to simulate fast moving scenarios and some pictures with broken signs to simulate real life scenarios. However, they chose 34799 out of 51839 pictures to be the training set, and I think this ratio (67%) is rather low because the training set ratio should be at least 70% conventionally for

the model to reach highest accuracy. But overall, we are really impressed by the quality and quantity of the dataset, which is one of the most important aspects for a machine learning project to work satisfactorily. Thirdly, about the model. We think the model is very meticulously and accurately designed, especially the three dropout layers that they utilized, it is very beneficial because it prevents overfitting. Also the final test accuracy shows that their model is placed second in accuracy, only worse than VGG19. So overall I think their project is quite successful.
- Thanks for your diligent and encouraging review.
- Your second point on the split of training and test data is an excellent one and totally valid in this context. However, we would like to make a clarification on that. The reason we restricted the training data to just 34,799 is because later on, we performed data augmentation on the training data to fix class imbalance, as mentioned in the presentation. After the data augmentation, our training data became a total of 86,989 images which is around 83.62% of the entire data.

3 Response to Group 59

Presentation-wise, the slides styles are consistent and the transitions between each member are smooth. Images you use are a good representation of your topics.
Contents-wise, I really like the idea of introducing motivations (i.e. why you are doing this) for basically everything you did. For example, you explained why you did the data augmentation, and where your final simplified model came from. It is also good to see that you evaluate your model not solely on accuracy, but use prediction time as one of your evaluation tools to compare different models. It makes your new implementation stand out.
- Thanks for your kind words.

Possible improvements:

- It might be a good idea to connect literature surveys with your solutions - I think literature review could be useful in your implementation. It might be an overkill but worth mentioning.
  - We absolutely agree that literature reviews should be linked to our implementation. We in fact, did mention it in our presentation. If you have a look from 3:56 in the presentation, we mentioned that this paper is what we're inspired from.

- I feel you could mention more about how you preprocess the data to feed into your networks, not just your train/validation/test split.
  -We did mention data preprocessing quite comprehensively in the presentation from 5:05 to 6:18. We will try to explain it again here if it was not clear. Upon some initial analysis, it was noticed that there was high class imbalance in the dataset. Had the model been trained on this dataset, it would've posed a problem by making the model biased towards the class with higher data frequency. To resolve this, data augmentation was performed. Images from low frequency classes were picked up and random rotation and brightness variation was performed. Eventually, data was normalized for all classes and results. Then, standardization was performed on the data to normalize its mean and make it unit variance.

- Maybe you could explain why you choose other networks as your benchmark.
  - The way we conducted this experiment was by increasing complexity step by step. So, first we chose VGG-19, a simple CNN, tuned its hyper-parameters and noted the best results. We further went looking for more and more complicated models like ResNet and DenseNet. Next, we took up GoogleNet to see how the inception module perform for this scenario. In the end, we even went beyond CNN to see how RNN, specifically LSTM, performs on this problem.

- Loss of the model could also be included in your presenation/report.
  - We did plot the loss curves and it represented the exact inverse of accuracy plots. So, we thought it would be redundant to add in the presentation. We're adding them here for your reference.

- I feel a small summary or conclusion would make your video more complete.
  - In summary, our newly proposed implementation gave us a significant accuracy (around 97.71%) with the best response time.