Traffic Sign Classification using Convolutional Neural Network

Jason Dias

Abstract

This project aims to contribute to autonomous vehicles by developing a system to identify road signs from still images. The proposed solution uses a Convolutional Neural Network to help with the image processing. A Convolutional Neural Network is an algorithm that can analyze an image, identify features of importance, and differentiate them from each other. The preprocessing for a convolutional neural network is lower than many other classification algorithms and will require fewer custom filters as this algorithm can learn many of the filters and characteristics on its own [1].

To simplify the network required, the solution will look to determine the size of the sign. This information will be used to crop the image around the sign. This new smaller image will be passed to the convolutional neural network, where the sign type will be determined.

This project used data from the Laboratory for Intelligent and Safe Automobiles (LISA). This database includes 6618 annotated images, each annotation containing image name, sign bounding, and sign type [4]. This data was essential in training the network to correctly identify signs. The Convolutional Neural Network was programmed in Python using the Tensorflow and Keras libraries.

Several different convolutional neural networks were tested, and their parameters were adjusted to determine the best solution to this problem. The final network used consisted of two main blocks containing two convolutional layers, a batch normalization layer, and a max pooling layer. Below this was a single fully connected layer feeding into the output layer. The first two convolutional layers had 4 filters each and the second set each had 8. The hidden fully connected layer had 128 nodes.

This final design was then trained on image data that was augmented using image variation. This resulted in much longer training time but achieved an accuracy of 96%. There were also indications that this training with image variation further improved the generalization of the system. This, however, was left to further research due to dataset limitations.

Introduction

Autonomous vehicles, also known as self-driving cars, are equipped with software and hardware that can fully operate driving functionalities such as steering, signaling, breaking, etc. without any human input [2]. The global self-driving car market is already of significant size at approximately $54 billion USD in 2019. This is expected to grow at a compounded rate of about 40% per year, leading to global revenue of over 500 billion USD by 2026 [3].

The growth of this industry necessitates the need for autonomous systems that can operate as the brains of these vehicles. Driverless cars have made big leaps and advances over the last decade. With the recent boom in this industry, comes the need to be able to design and build software systems to ensure the utmost safety of pedestrians, passengers, and other road users.

Of critical importance to the driverless system is the ability to detect and identify road signs. This is required as autonomous cars must be able to follow posted limits and adhere to other instructions provided by road signs. Classifying road signs is necessary for autonomous cars to make decisions in navigation and keep passengers and other road users safe, as these cars would know what to do based on the road sign classified. This project aims to design and build a system that will classify road signs for use by a driverless car system.

When it comes to complex problems such as systems that must recognize and identify objects in an image the most common approach is to use machine learning. Neural networks are the backbone of machine learning and provide a powerful approach to classification problems. They are designed by putting together a plethora of programmable neurons, feeding them an input, and having an algorithm work out an output based on patterns previously identified in training data.

For the purposes of image visualization, a basic fully connected neural network is not enough. Instead for this project the group decided to use a convolutional neural network, as they are commonly used to analyze visual imagery and are frequently working behind the scenes in image classification [4]. Image classification lies in taking an input image, and relaying an output class, based on the probability of the input image belonging to that class.

In a driverless system, the group plans on feeding the network an image from a camera feed, and have the system output the highest likelihood of the image belonging to a certain class. For the purposes of this project, it will be assumed that a system has been developed that can identify the location of sign objects in the image feed, but cannot classify them. This project will fill that gap by taking the cropped image of just the sign and identifying which type it is. The group will use the LISA (Laboratory for Intelligent and Safe Automobiles) data set, which contains over 6000 images equipped with labels, locations, size, and type [5]. This data will help the group train, test and validate the groups model.

This report will explore the proposed solution to this problem in depth, as well as analyze the results and conclude its benefits to driverless systems.

Background

In order to understand the solution presented in this report the important background material necessary for the project is given in the next few sections.

Convolutional Neural Networks

A convolutional neural network can take an input image, assign various weights to different parts of the image, and be able to differentiate objects in the image based on those weights from one another [4]. Convolutional neural networks become extremely important in the image space. This is because they are able to take in an array of pixel values representing an image and keep its spatial and temporal dimensions. This is different then fully connected networks that struggle to recognize the overall features in an image and will fail if the feature of interest rotates or is partially blocked. Convolutional neural networks are also able to scale to full image sizes while regular neural nets are unable to [6]. This project will use convolutional neural networks to attempt to learn the key features of different sign types and then output the probability that an image belongs to a certain class.

Convolutional neural networks depend on multiple layers for image classification. The key layer in the network is the convolution layer. The convolution layer does much of the most important computation work in a convolutional neural network. The convolution layer is responsible for extracting the basic features of the image which include low level features such as edges, color gradients etc [4]. The convolution is a linear operation that involves the multiplication of a set of weights with a given input. The filter, or kernel, is a two-dimensional set of weights that are used to multiply the input, in an attempt to create a feature map with that filter. Each filter or kernel is designed to look for a certain feature in an image, and then multiplying the filter with the input in the convolution layer. This is how the layer draws out a feature map for that image [4]. After developing a feature map, the network passes this map through to an activation function. For this project the RELU activation function was used as it is standard in the convolutional neural network space and does not have any hyperparameters to be adjusted. By stacking multiple convolution layers in a network the earlier features will learn low level features such as edges, while the deeper layers will learn higher level features like shapes or objects [7].

Generally, after several layers of convolution a pooling layer is included. The pooling layer reduces the spatial size of the feature map to reduce the number of parameters and computation requirements in the network [7]. During this down sampling the pooling layer extracts dominant features and discards unimportant ones. The pooling layer also allows for detecting these features even when their position or rotation changes [4]. They also provide an approach to downsampling the feature maps generated by representing the presence of features in patches of the feature map [7]. Average and max pooling are the two popular pooling methods used to summarize the presence of a feature. The size of the pooling operation is always smaller than the feature map it applies to. Most of the time it is a 2x2 layer, and thus it halves the

dimensions of the feature map. The average pooling layer involves calculating the average for each patch of the feature map, meaning that each 2x2 square of the feature map is downsampled to the average value in that square [7]. The max pooling layer on the other hand figures out the largest value of each patch in each feature map. Pooled feature maps using the max operation highlight the most present feature in that patch [7]. This is useful as this would be the most important feature in that section.

The final layer in the network is the fully connected layer. This layer is used to detect high level features in an image. The fully connected layer looks at the output of previous layers to determine which class the features belonging to that output most correlate with. For instance, if the system is predicting that an image is a table, it will have high values in activation maps of features such as four legs, or a flat surface to support that prediction [6]. The fully connected part goes through a backpropagation process to determine the most accurate weights. Using that, the neurons in the system each receive weights that prioritize the most appropriate model. Each neuron then selects a label and the label most selected is the class output.

In the case of classification, where an image can only belong to one class, this final output layer will typically be a vector of neurons where each individual neuron represents 1 class. When the network is fully trained, this means that one neuron having a higher activation implies the input corresponds to that neuron's class. Generally, the softmax activation function is used as the activation function for this output layer. This activation function ensures that the total of all the output values adds to one. Thus, if one node is assigned a high probability of being correct then all other nodes will have a small probability. This prevents conflicts between multiple output nodes.

Regularization

In order to improve the effectiveness of the network and allow it to generalize better several regularization strategies can be adopted. The first method for this is batch normalization which is a technique that effectively normalizes the inputs to a layer. This improves the training time by standardizing the input that subsequent layers see. In addition to reducing the number of epochs required for training also helps to improve the generalization of the system [8].

Another method of regularization is the addition of a dropout layer in the fully connected layers of the network. A dropout layer temporarily removes neurons from the fully connected layer during the process of training. This tends to improve the generalization of the network and also speeds training as the dropped nodes do not have to be updated [9].

Additionally in addition to changes to the network itself the data can be augmented to improve performance. This process takes existing data and modifies it using transformation and other operations. This increases the training set size and helps to adjust the network for better performance over a wider range of images. This process is explained in more detail in Image Variation.

LISA Dataset

The LISA Dataset that was used in this project contained images of US road signs obtained from cameras used by the vehicle on the road. There are approximately 47 different sign types in the data set, with 7885 annotations on 6610 frames [5]. The images are not typically high resolution. An example of an image from the data set is shown in Figure 1.

Figure 1: Example Image from Lisa Data Set

Programming tools

The tools that were employed to aid with building the classifier model include Python, as well as the Keras tensorflow library, and the tensorboard library for visualization. This model was entirely programmed in python due to its heavy computational capabilities, as well as libraries and frameworks that support convolutional neural networks. The Keras tensorflow library is an open-source neural network library written in python and the tensorboard library helped the group in visualization. Keras also contains a significant amount of support for image operations. These operations include dynamic application of random image transformation, image loading, and image iteration operations. The random image transformations are described in further detail in the solutions section of this report.

In addition to these tools, tensorboard is a visualization tool used to visualize model performance while training. This tool was used to monitor performance and help tune model parameters. It was also used to generate several graphs seen in this report.

Solution

The solution to this classification problem requires several main components to be designed to function. First a method to reorganize the images and annotations from the LISA data set must be built. Then a neural network to classify the images must be designed and tested. Finally to further improve the neural network an method for image variation must be designed and used to further train the network.

Annotation Refactoring

The LISA dataset is organized as 6618 images with a list of annotations referencing these images. Each image is a picture of a street containing at least 1 sign, but potentially as many as 4. The annotation file for the images contains the name of the image as well as the tag and location bounding for each sign in the image. This format was not suitable for this project as each sign needed to be treated as is its own annotation. In addition, the LISA dataset was also subdivided into folders according to the origin of the images. To simplify the process of removing data and allow for randomization it was desired that all the images be placed in the same folder with an annotation file that had a line for each sign in every image.

To address this a python script was developed that would iterate through the existing annotation file. For each line in the annotation file the existing image would be moved to the new folder structure. The annotation would then be subdivided such that each sign in the image would have its own line in the new annotation file. Once finished the python script had created a simple folder containing every image and an annotation file detailing every road sign within these images.

This made it simple to implement code to divide the dataset into training and test sets. It also allowed for very quick cropping of the images around the sign under consideration. For further explanation of image modifications see Image Variation.

Model Design

The convolutional neural network designed for this problem consisted of two main sections. First was the section containing the convolution layers, followed by fully connected layers that connected to the output. The overall layout can be seen on the left-hand side of Figure 2.

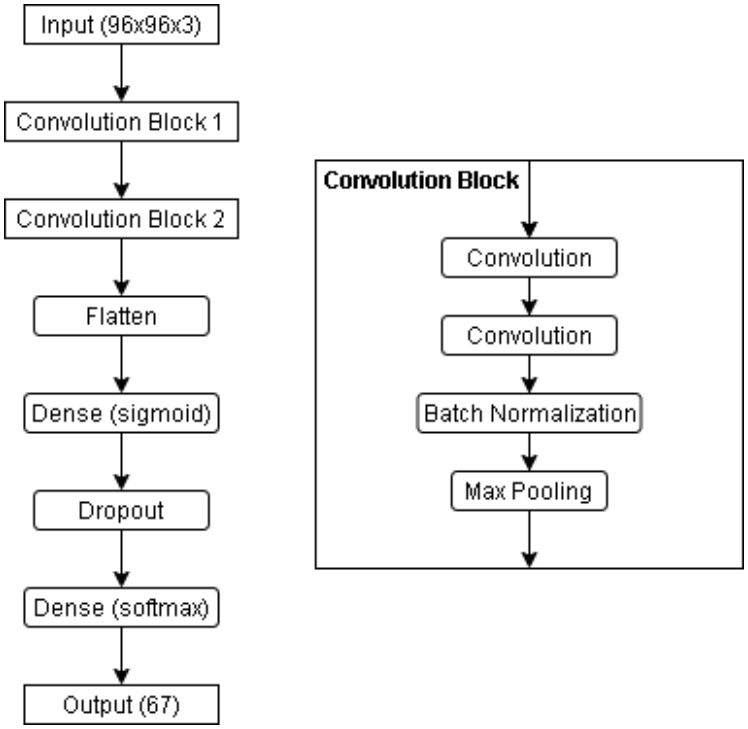

Figure 2: General Convolutional Neural Network Layout

The sign is given to the system as a 96x96 pixel image with rgb color. This is fed into the top layer of the network. The convolutional layers section was organized into blocks where each block worked within the same dimensions and then reduced the dimensionality as it passed information to the next block. Each block contained at least one convolutional layer and a max pooling layer to perform the dimensionality reduction. In addition, the block could also contain a batch normalization layer if required. The layout of each convolutional block is shown on the right side of Figure 2. Within the blocks the number of filters used by the convolutional layers was kept the same, however each block had twice the number of filters as the previous block. This meant that only the number of filters for the first convolutional layer was ever set. For example, if convolution block 1 has 8 filters in each of its convolutional layers then convolutional block 2 will have 16 filters. All the convolutional layers utilize a Relu activation function as it is simple to implement and provides good results for feature recognition [7].

Following the convolutional section, the system goes through a flattening layer and then into a fully connected layer that uses a sigmoid activation function. Finally, the fully connected layer feeds into an output layer that consists of 67 nodes each corresponding to one of the possible sign types. This layer uses the softmax activation function to assign a probability distribution to the possible sign classes.

Several different features within this general network format were changed throughout the course of this project in an effort to identify the best network for sign classification. Some of the things changed include the number of convolutional layers in each convolution block,

whether the batch normalization and dropout layers were included, the number of filters in the convolutional layers, and the number of nodes in the dense layer.

The LISA dataset contains 6618 different images of signs. For training the network the database was split with 20% of the images being reserved for validation and 80% for training. The training was performed over 20 epochs with a batch size of 32. In order to improve the generalization of the network, the optimal model design that was found was then also retrained using image variations (in addition to the regular images).

Sign Cropping

Before inputting an image into the model, the sign must be cropped out to remove redundant information. When an annotation is processed for training the sign image is cropped to the bounds specified in the annotation. Figure 3 is an example of a cropped-out sign, which is seen in Figure 1.

Figure 3: Cropping of sign image

After a sign is cropped, it is resized to 96x96 pixels. This size was picked as it is typically larger than the original cropping, however not so large that the resizing algorithm blurs details and adds image artifacts.

Image Variation

Although the data set contains several thousand images (6618 annotated images), these images come from a much smaller set of drives, from an even smaller set of cameras, where signs can be repeated across consecutive images. In order to increase system robustness it was decided to retrain the optimal model with input data augmentation. When image variation training was enabled, the following variation transformation parameters were applied to each training image input into the model:

- Image rotation between -10 and +10 degrees
- Width shift of up to 5% in either direction
- Height shift of up to 5% in either direction
- Zoom of up to 10%
- Greyscale, randomly applied to 25% of images
- Jpeg compression between 80% to 100% of original quality, randomly applied to 50% of images
- Hue rotation between 0 and 359 degrees, randomly applied to 10% of images

- Colour inversion, randomly applied to 10% of images

Figure 4 below is an example of an image with only a 180-degree hue rotation applied to it. Although the example below was for an entire image, in reality these transformations were applied to the cropped sign images only.

Figure 4: Before and after a hue shift of 180 degrees

Results

The first network tried consisted of one convolutional layer per block with no batch normalization layers. The first block contained 8 filters in its first convolutional layer and the hidden dense layer contained 512 nodes. To understand if dropout could help in this system the same network was run twice once without any dropout layer and once with a 50% dropout layer. The training results of these two networks are shown in Figure 5.

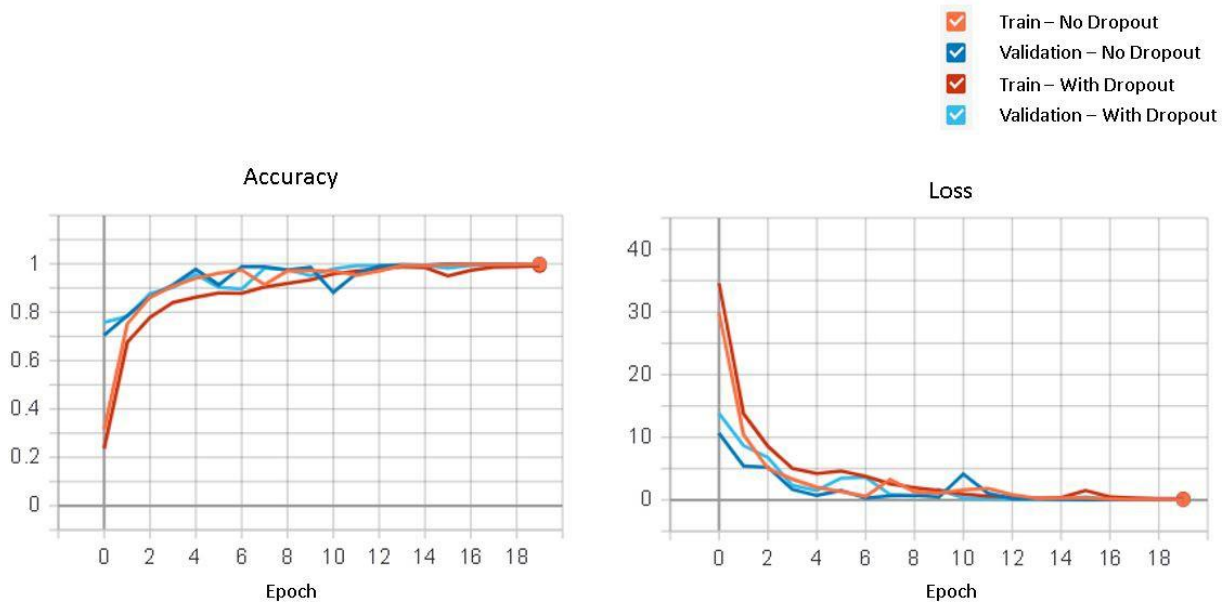

Figure 5: Single Convolutional Layer Training Results

As seen above the network trains quickly reaching above 90% accuracy after 8 epochs. It can be seen that the network does not overtrain, and it levels out at almost 100% accuracy after 13 epochs. This suggests that the network is good at generalizing and is not susceptible to overtraining. For this reason, it is unsurprising to see no improvement due to the addition of a dropout layer. Typically, a dropout layer would be included in order to improve generalization and prevent overtraining. In this solution all it does is slow down the improvement of the system and cause some instability in the results. This can be seen by the lower accuracy and greater loss. The instability can be seen in epoch 13 where the accuracy suddenly drops possibly due to the dropout layer removing several of the key nodes for classification.

The same network setup was tried with the number of convolutional layers in each block increased to two. This new network was run twice, the second time with the batch normalization layers included. The hope was that this inclusion would increase the speed at which the network trained. The results of these tests can be seen in Figure 6, where the training results are plotted along with the validation numbers.

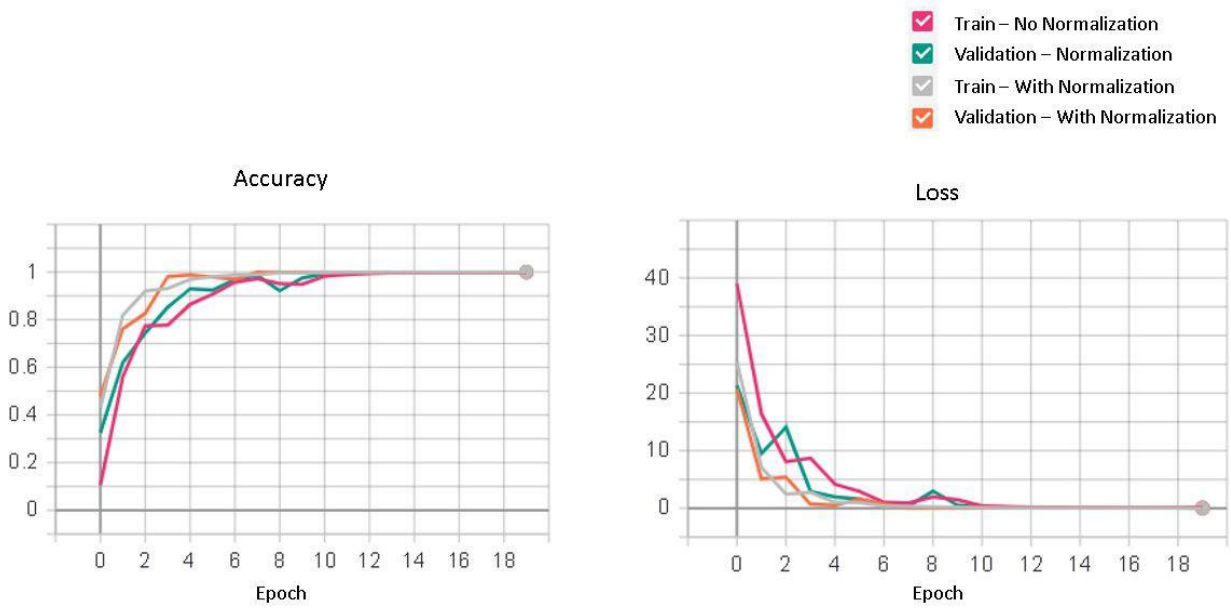

Figure 6: Double Convolutional Layer Training Results

As seen above the system again shows no tendency towards overfitting, thus the addition of drop-out here would provide little benefit. Without batch normalization the network accuracy starts to level out after about 10 epochs, which is an improvement from the single convolutional layer system which took 13 epochs. This is also seen the loss which also levels out after 10 epochs as opposed to the 13 required for the previous system. The addition of batch normalization improves on this even more. Within 7 epochs the batch normalized system has reached its maximum accuracy and its minimum loss. This improvement is as expected as one of the main benefits of batch normalization is its effect on the ability of a network to quickly train. One of the other main benefits of batch normalization is its generalization effect, which similar to dropout helps to prevent overfitting. However, as mentioned previously this system does not overfit and this benefit of batch normalization is not needed. The use of two convolutional layers per block as well as the inclusion of batch normalization layers were both kept in all further networks as they both were shown to significantly improve the system.

As the network has an exceptionally high accuracy within 7 epochs it was decided that decreasing the number of filters as well as the number of nodes to explore less computationally expensive solutions would be useful. This was done in the hope that it would have a smaller reduction in training time then going back to a single convolutional layer per block would.

The changes to filters and dense nodes were done with three networks to test the effects of each change. The first network had a reduced number of filters in the convolution layers with 4 in the first block instead of the original 8. This meant the second block would have 8 filters in its convolutional layers instead of the 16 it had in previous networks. The second network reduced the number of nodes in the dense layer to 128 from 512. The final network combined both these changes using only 4 filters in the first block and 128 nodes. The training accuracy of these three systems as well as the original system without the changes to filters or number of

nodes are plotted below in Figure 7. Similar to previous tests the validation accuracy and loss had no significant difference from the training data, so validation data was not shown in this figure to increase clarity.

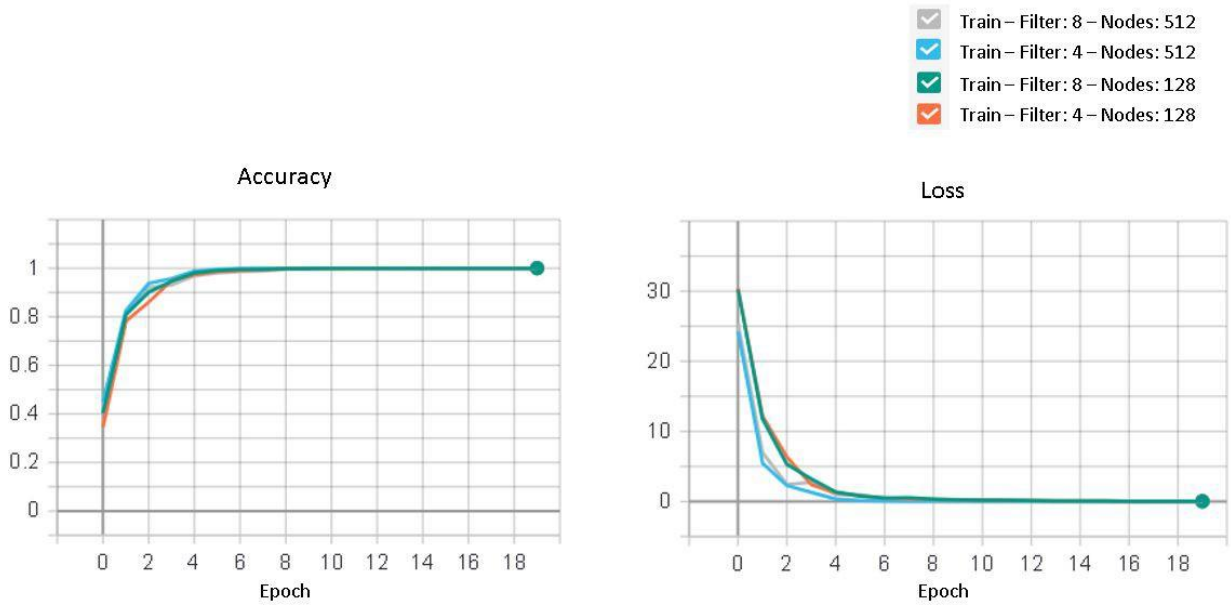

Figure 7: Reduced Filters and Nodes Training Results

There was very little effect from reducing the number of filters and connected nodes. The system is still seen to achieve maximum accuracy and minimum loss after 7 or 8 epochs. As seen above the trajectory of the accuracy and loss stays nearly the same for all four networks. There does seem to be a decrease in accuracy and increase in loss with the two networks using only 128 nodes, however the difference is small. This result is important as reducing the number of filters and nodes reduces the total number of items that need to be trained in the network. This reduction will directly reduce the computational power required to use this network improving its effectiveness as a potential real time solution to the problem of sign classification.

To further test the effect of reducing the number of filters and connected nodes another network was designed. This one had only 2 filters in the first block and only 64 nodes in its dense layer. The training results of this network are plotted in Figure 8 against two of the previous networks to show the change.

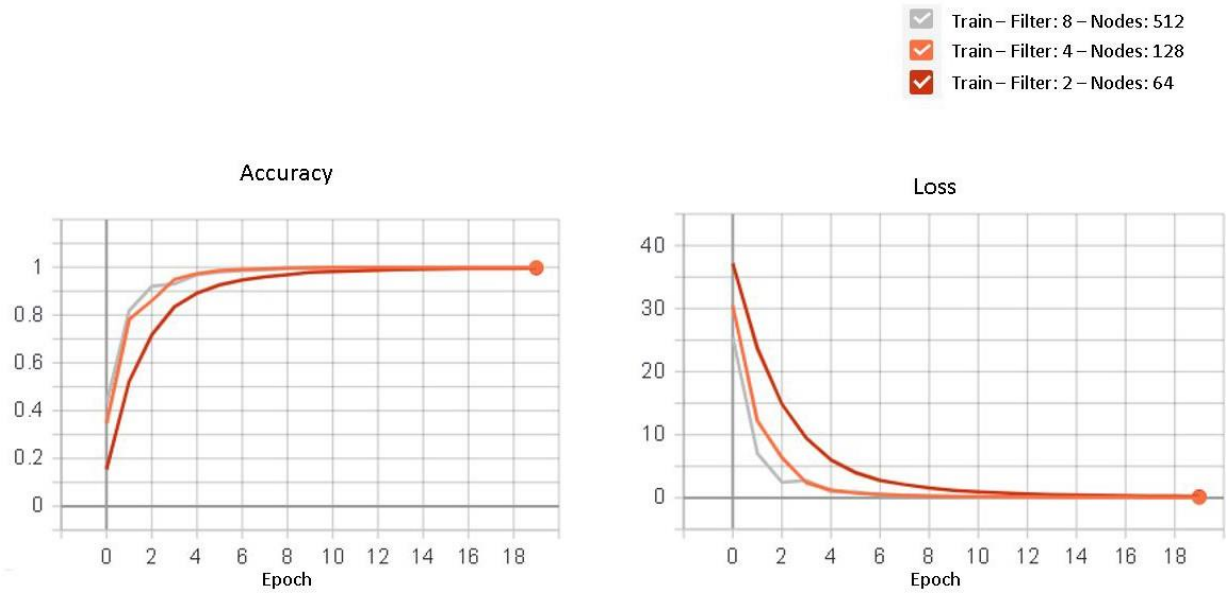

Figure 8: Filters x2 and Nodes x64 Training Results

Here finally we see a change due to the decreased number of filters and nodes. The system still achieves 100% accuracy and minimal loss after only 14 epochs. This however is significantly slower then the other two networks which both achieve maximum accuracy within 8 epochs. It was decided that the more complex network with 4 filters in the first block and 128 nodes in its dense layers would be the best network for this problem. It was hoped that this more complex layer would be better able to generalize for better classification under more complex conditions. The graph of this final network generated by TensorBoard is shown below in Figure 9.

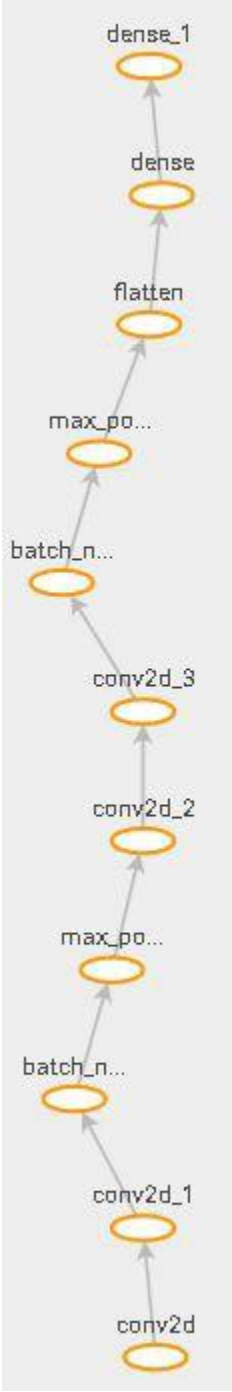

Figure 9: Graph of the Final Network Design

In order to better train the model for more generalized cases the image variation methods detailed above were used to input more complex image cases. Seen below in Figure 10 and Figure 11 are the accuracy and loss respectively.

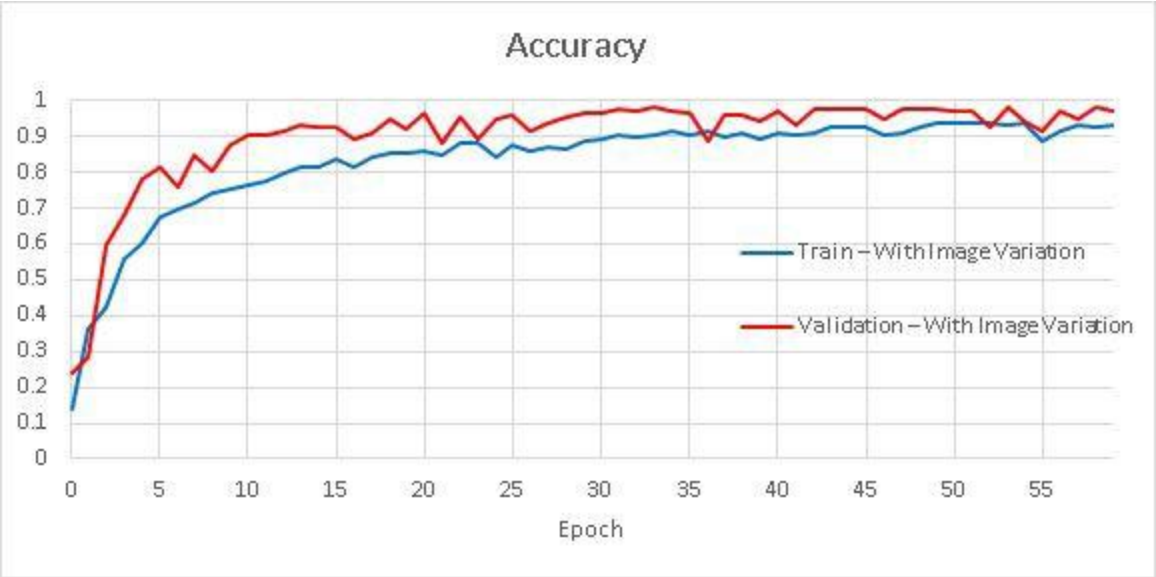

Figure 10: Image Variation Training Accuracy

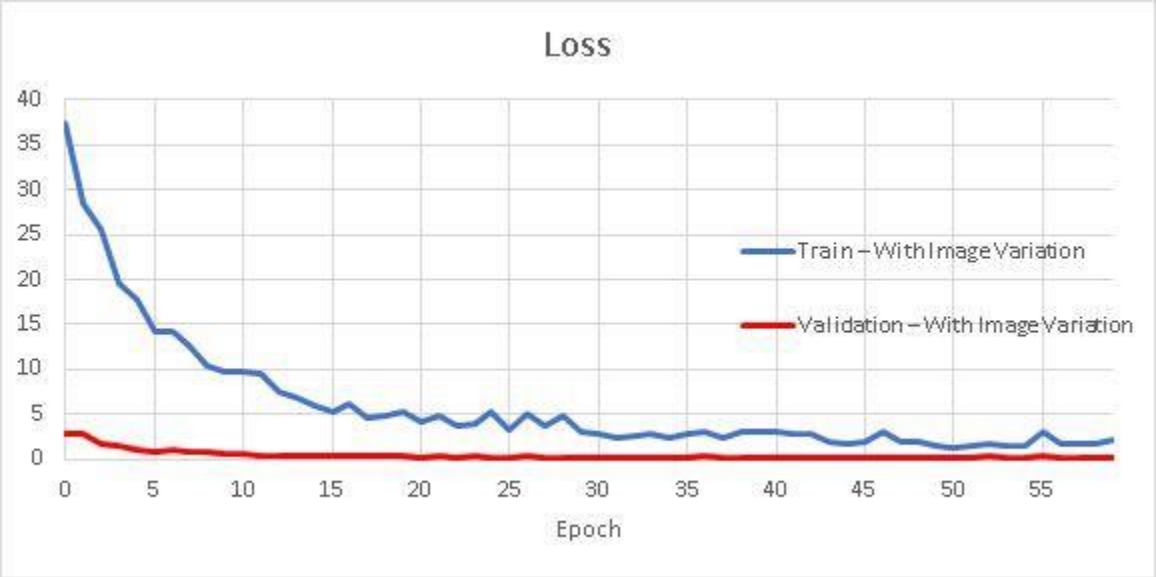

Figure 11: Image Variation Training Loss

As seen above the network takes much longer to train when using varied images. Instead of 8 epochs it takes about 35 epochs for the system to level out. The accuracy achieved is very high reaching 95% training accuracy after 60 epochs. This is close to the accuracy achieved without image variation. It should be noted as well that the validation accuracy and loss are consistently better than the training values. This is as expected as the inclusion of image variations was only done to the training set to improve its generalization. The validation set is not varied thus the accuracy when the network is tested on it is better.

The slower training seen by the system when using image variation is due to the greater amount of data that the system sees when training. This increased variation means the system takes longer to find more general features that can be used for correct identification of the sign class.

This helps to address one of the major issues of the data set used, the lack of variety. The

images in this data set were taken by only a few different cameras in similar environments with only a handful of weather conditions. This means that even if the network does not overtrain when tested within this data set it may be overtrained in the sense that it will not be able to account for the more varied conditions found in the real world. The addition of image variation is one way to prevent this as it forces the model to further generalize and will hopefully allow it to correctly identify real world images.

Conclusion

This project was focused on devising a convolutional neural network that could accurately identify the type of sign based on an image of that sign cropped from a larger image of a street. This problem has its challenges as depending on the resolution of the overall image as well as the size of the sign within the image these smaller cropped images can come in many different resolutions. In addition, the signs are found in many lighting conditions and often have obstacles blocking part of the sign.

To address this problem a general convolutional neural network design was developed with the intention of refining it to achieve better results. This design consisted of convolutional blocks that contained convolutional layers, batch normalization, and max pooling. In addition, there was a single hidden dense layer finally leading the output which would identify one of 67 potential sign types.

After testing it was found that the ideal configuration consisted of two convolutional layers per block as this improved the time in which the system trained. It was also found that the use of batch normalization greatly improved the training speed of the network, decreasing the number of epochs to 10 from 13. It was also discovered that the use of a dropout was not required as it provided no benefits to this system and instead slowed down the training.

With the overall architecture decided on, some of the parameters within the system were tuned. Initially the system had 8 filters in its first set of convolutional layers and 512 nodes in its hidden fully connected layer. Several tests with changes in these numbers resulted in the realization that the system could instead have 4 filters and only 128 nodes in the fully connected layer. This change resulted in almost no change in the training speed, accuracy, or loss and thus it was decided to use this as the final network for this problem. It may also have been possible to reduce all the way to 2 filters and 64 nodes, however this was seen to start to have negative effects and it was decided that effective training was more important than decreasing computational cost.

With the finalized network decided on, the system still needed to be tested using images that first were passed through the image variation system. This system performed random hue shifts, rotations, and other transforms to create greater variety in the training data set. When trained on the varied data set the system took much longer, 60 epochs instead of 8. It did however reach similar levels of success achieving up to 96% accuracy. It also showed signs that the system was more generalized.

The generalization of this network is hard to prove beyond the scope of this data set and further research will be needed to show whether it will be effective when using images from other sources. It is hoped that the addition of image variation has given this network the capability to correctly identify road signs in scenarios that are distinct from the LISA data set.

For example, identification in more complex weather conditions or using different cameras from those used by the LISA data set.

References

[1] S. Saha, "A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way", Medium, 2020. [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. [Accessed: 23- Jan- 2020].

[2] M. Rouse, "What are Self-Driving Cars and How Do They Work?", SearchEnterpriseAI, 2020. [Online]. Available: https://searchenterpriseai.techtarget.com/definition/driverless-car. [Accessed: 23- Jan- 2020].

[3] A. Jadhav, "Autonomous Vehicle Market Size, Share and Analysis | Forecast 2026", Allied Market Research, 2020. [Online]. Available: https://www.alliedmarketresearch.com/autonomous-vehicle-market. [Accessed: 23- Jan- 2020].

[4] “CS231n Convolutional Neural Networks for Visual Recognition.” [Online]. Available: http://cs231n.github.io/convolutional-networks/. [Accessed: 03-Apr-2020].

[5] “Laboratory for Intelligent and Safe Automobiles - CVRR - UCSD,” Computer Vision and Robotics Research Laboratory. [Online]. Available: http://cvrr.ucsd.edu/LISA/lisa-traffic-sign-dataset.html. [Accessed: 24-Jan-2020].

[6] A. Deshpande, “A Beginner's Guide To Understanding Convolutional Neural Networks,” 20-Jul-2016. [Online]. Available: https://adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/. [Accessed: 03-Apr-2020].

[7] J. Brownlee, “A Gentle Introduction to Pooling Layers for Convolutional Neural Networks,” Machine Learning Mastery, 05-Jul-2019. [Online]. Available: https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/. [Accessed: 03-Apr-2020].

[8] J. Brownlee, “A Gentle Introduction to Batch Normalization for Deep Neural Networks,” Machine Learning Mastery, 16-Jan-2019. [Online]. Available: https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/. [Accessed: 20-Mar-2020].

[9] M. Pushparaja, D. Shanmugasundaram, “Regularization and Optimization strategies in Deep Convolutional Neural Network,” Nanyang Technological University, 13-Dec-2017.