Ideal Traffic Sign Images Classification For Convolutional Neural Networks

Matthew Ernst
Colorado State University
Computer Science Department
Fort Collins, United States
mfernst@rams.colostate.edu

James Yost
Colorado State University
Computer Science Department
Fort Collins, United States
jeyost@rams.colostate.edu

ABSTRACT

Images easily fool convolutional neural networks with noise. They are not as secure as previously thought. We show this by training a VGG16 model on the Mapillary data set on over 200,000 images[1]. We were able to trick this model into a classification of images that appear to be random noise as 100% confidence prediction of a particular class. Ultimately, we found these ideal images as a result of a random search over the image space. With our small images 32x32x3, we were able to find these relatively quickly and prove the drawbacks of convolutional neural networks and their security.

KEYWORDS

datasets, neural networks, image processing, network fooling

ACM Reference Format:
Matthew Ernst and James Yost. 2021. Ideal Traffic Sign Images Classification For Convolutional Neural Networks. In *Proceedings of Graduate Artificial Intelligence Spring 2021 (CS540 S'21).* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/CS540.CSU

1 INTRODUCTION

Since 2015 convolutional neural networks (CNNs) have become a predominant tool within the deep learning community. These networks have been used for many different tasks, from image and facial recognition to studying protein structures. It is such a predominant topic, and it is widely used in many various industries across the world. It will likely continue to be in use for some time in the future. This is mainly because these networks have been able to solve deep learning problems with surprising accuracy. However, this raises the question of the validity of these networks. Specifically, how one can fully trust a networks classification.
This could also be a safety factor for self-driving cars. For a neural network, what does it classify as the best stop sign or speed limit sign? Do these ideal images line up with what human vision would be able to recognize? To answer these questions, we developed an extensive experiment to determine what perfect images were produced for street signs. More specifically, we looked at stop signs and yield signs. We used tools such as TensorFlow to create a convolutional neural network to get high accuracy on the Mapillary dataset. This dataset is a standard used in training neural networks for companies such as Toyota and Lyft[1]. With this network, we then developed a random search algorithm that starts with randomized images to be passed through the corresponding network and steps towards an ideal image by adding noise to the image until a high accuracy of classification was achieved for a specific traffic sign.

In summary the contributions of this paper are:

- The evaluation of convolutional neural networks and their classification of ideal images for a traffic sign dataset.
- An investigation into producing randomized image to trick convolutional neural networks.
- Testing limits of our random search algorithm to add noise to images
- Improving classification of traffic signs with added noise

2 MOTIVATION

As stated in the Introduction section, driving this research is concern about the security and the reliability of convolutional neural networks. As these types of networks were developed, researchers and companies began to use them to solve many problems that were previously unsolveable. However, many researchers did not look into the security of these networks. Dr. Atom Bohm, a professor at Colorado State University, helped seed this idea of questioning networks for our research. Dr. Bohm worked with government

CS540 S'21, , Fort Collins, Colorado

ACM ISBN 978-x-xxxx-xxxx-x/21/04…$0.00
https://doi.org/10.1145/CS540.CSU

contractors such as DARPA to use convolutional neural networks to analyze and classify military vehicles based on their shadows from video footage captured by drones. Dr. Bohm had questioned the security of these networks findings by varying occlusions and real-world conditions and changing the video footage captured[5]. His question of these networks sparked our curiosity to ask similar questions geared towards self-driving cars and their security, reliability, and ability to correctly recognize traffic signs.

3 RELATED WORK

Many studies and research have been done at looking at the robustness of convolutional neural networks. Most of these studies are in the self-driving cars community. This section describes a few papers that helped us to comprehend this problem better.

Figure 1: Mapillary Traffic Dataset. (2019). [Photograph]. (https://blog.mapillary.com/update/2019/06/27/mapillary-traffic-sign-dataset.html).

3.1 Traffic Sign Detection Using Deep Learning

The first paper is out of The University of California, San Diego, looking at traffic sign detection with deep learning networks[2]. This paper looks at different deep learning architectures and how they compare in trying to classify 43 varying traffic signs. Moreover, it looks at the prediction speed needed for self-driving cars to make a decision, such as breaking when a network correctly classifies a stop sign. This paper inspired us for our convolutional neural network architecture. We wanted to have a standard baseline for a network to check and compare that our model's accuracy is equivalent to that of a network used in industry. This paper was vital in this comparison, specifically, the VGG model family. These network architectures are used by companies such as Tesla. Therefore, if our network had similar accuracy, we would have a network in which we can believe in its decision-making. This paper also helped in dataset management and selection. Again, following the industry standards, we thought it vital to have a dataset to train and test that is ubiquitous. A sample of this dataset can be seen in Figure 1. This dataset was both universal and had many data points for us to use.

3.2 Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images

The following paper that inspired us was by Anh Nguyen[3]. This paper discusses how state-of-the-art deep learning architectures can be fooled into classification confidence of above 99.99%. Moreover, the images used to fool the networks are utterly unrecognizable by humans. The paper deals with a specific dataset, such as MNIST, and can easily fool networks that have high accuracy. However, they were also able to do this with more advanced datasets that involved objects such as animals.

We found this paper to be important for us for a few reasons. The first is that we now know that deep learning networks such as convolutional neural networks can be tricked into classifying humanly unrecognizable images at high confidence rates. Secondly, the paper links to a GitHub repository that outlines how they produced the images that tricked these networks. We ultimately used this repository as a foundation and reference to create our images for our specific cause. However, this paper uses a version of genetic and evolutionary algorithms versus our random search method; it is still an excellent basis for our produced images.

4 METHODS

4.1 Dataset Tailoring

We used the Mapillary dataset[1], based on the fact of its prevalence in the deep learning community and the self-autonomous driving areas of research. This dataset is extremely large; roughly 42,000 images from around the world with complete annotation and averaged approximately 4-5 images per street sign. Along with the sheer number of images, they are all full resolution. Ultimately, this leads to a vast size of data (70GB). Due to this large set, we had to tailor the experiment to run with what we had in terms of hardware.

With the dataset chosen and our hardware constraints laid out, we began our tailoring process. First, we went through the entirety of the dataset and extracted each street sign and its label to help with complete annotation. This became a unique problem due to the images given. For example, a stop sign in the same class may be in an image with multiple other traffic signs in it at varying distances, while another image may contain only the stop sign itself. However, with the dataset being annotated in a JSON object, these annotations provided bounding boxes showing where the traffic signs were located in images with multiple traffic signs in

one image. With these bounding boxes, we could extract all of the individual traffic signs from a single image. At the end of this process, we ended up with around 200,000 images in total. Of these 200,000 images, we had 401 different traffic sign classifications. From here, we began to change the dimensionality of the images due to the hardware constraints. We started scaling down the images slowly to try and preserve the high resolutions while achieving efficient training. Eventually, the best dimensionality for the images was found at 32x32x3, and the dataset was finished. A subsection of the dataset can be seen in Figure 2.

Figure 2: Tailored Dataset. [Photograph].

4.2 Model Generation

The model we decided on for our convolutional neural network was a prominent architecture in deep learning called VGG16. Figure 3 gives a brief layout of the architecture and how this model works. We were able to find an implementation of this network through GitHub[3]. However, the network provided was the standard model, which accepts RGB images of dimensionality of 224x224x3. We adapted this model to match the data, which had a dimensionality of 32x32x3. This proved to help with efficiency as this network is notoriously slow and has extensive network architecture weights. We were able to adapt this network with the use of TensorFlow 2, specifically Keras.

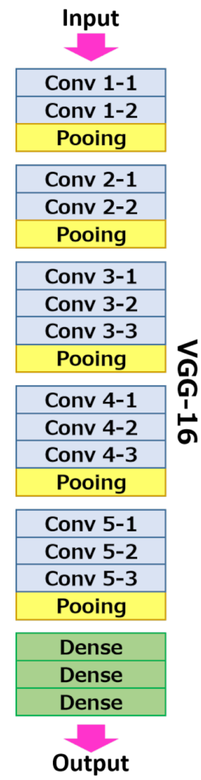
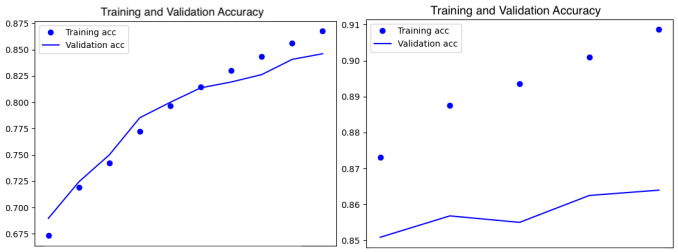


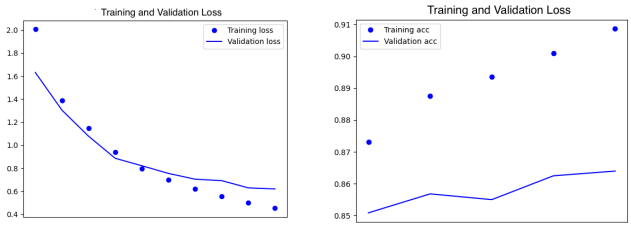
Figure 3: VGG16 Model Architecture [Photograph]. (https://neurohive.io/wp-content/uploads/2018/11/vgg16.png).

We began training our model by using TensorFlow to split our data. The data was divided into a 90% train and 10% test. 20% of the training data was used for validation. The split of the data was also done using a seed to maintain the same partition. This decision came from the vast size of the dataset and our inability to train the network all at once. To measure accuracy, we had to train in intervals and save the model's state. By having the split maintained, we had the ability to continuing training more epochs if needed, and we would reduce the chances of overfitting with our model. We began training the model, which took 12 hours to complete and ran for 20 epochs. The training resulted in a training accuracy of 91%, validation under 86%, and testing at 85%. The model's results can be seen in Figures 4 and 5. Ultimately we decided to stop training the model because the validation accuracy seemed to be leveling out, and we did not want to overfit the model.



(a) Epochs 5 - 15 (b) Epochs 15 - 20

Figure 4: Training and Validation Accuracy



(a) Epochs 5 - 15 (b) Epochs 15 - 20

Figure 5: Training and Validation Loss

4.3 Random Search

Our method of finding ideal images was via a random search. Figure 6 outlines the process of finding the ideal image. First, we start with an image with randomized pixels between 1-255 (I_o). From here, a standard normal Gaussian distribution of noise is applied to the image. The first noise applied image is just an intermediate step (I_v/I_n). With the image having noise applied, the image is passed through the trained convolutional neural network. Once classified, we looked at the confidence prediction of a particular class (k_c) and compare that to the current highest percentage found (k_m). If a higher confidence rate was found, that image would be saved (I_m) before being sent back to the beginning of the process and adding more noise. This process would continue until a confident prediction is as close to 100% as possible. Once found, that final image would be saved.

This process was ideal for the dataset that we tailored. Having the images in the 32x32x3 proved to be efficient enough for our random search to find ideal images reasonably quickly. It is important to note that if the images were full size, this search style would not work due to efficiency.

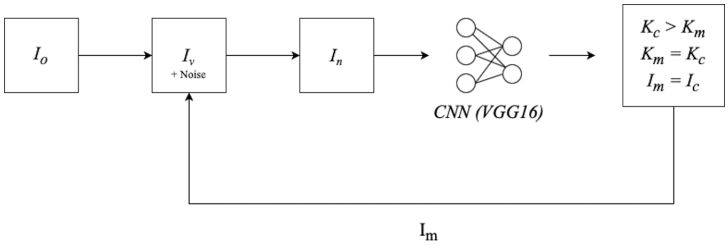


Figure 6: Random Search Diagram [Diagram].

This is expanded on in the paper stated earlier ("Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images"). In that paper, they discuss this issue and solve it using genetic and evolutionary algorithms[3].

5 EXPERIMENT & RESULTS

5.1 Noise & Image Generation

The noise added to the images was standard normal Gaussian distribution. This was done through Numpys random class. The code for how the noise is applied to the images is shown in Figure 7.

```
row,col,ch = image.shape
gauss = np.random.randn(row,col,ch)
gauss = gauss.reshape(row,col,ch)
noisy = image + image * gauss
```

Figure 7: Noise generation

The experiment began by first loading our model to use for classification. From here, we followed the process of our algorithm as stated in section 4.3. The first result that we found shocking was how quickly we could find images that were classified incorrectly. A graph of this can be shown in Figure 8. In this graph, we see how many iterations an image had to be processed (noise added to the image) until it was classified to a given class with a high level of confidence. Typically, the process would take just under a minute and would run for roughly 300 iterations.

It is important to note that this type of result is closely related to the image size. With the size being 32x32x3, this allowed our random search algorithm to work this efficiently. However, if the images are much larger than this, then the search space would become too large as it grows at a square constant by the image pixel dimensions (assuming the dimensions are square).

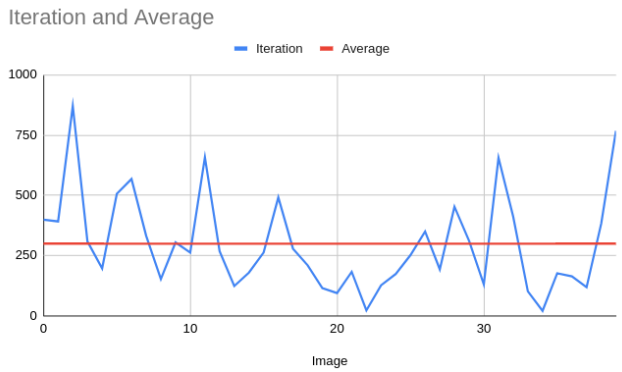
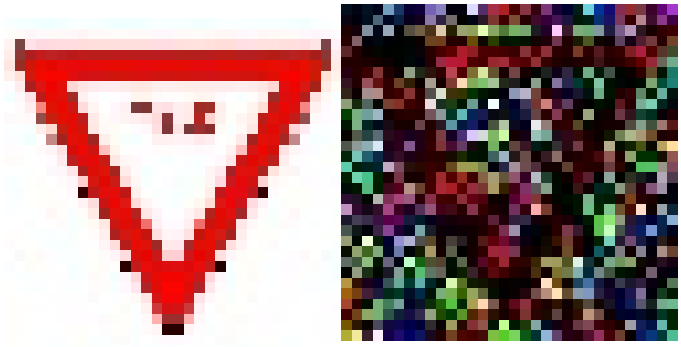


Figure 8: Iterations of algorithm

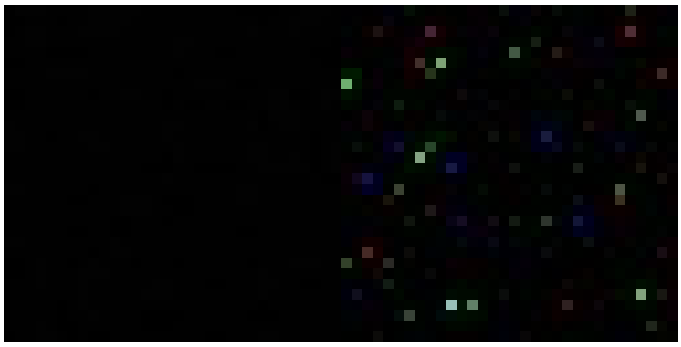
5.2 Normal Traffic Sign, Fully Black Image, & Fully White Image

Figure 9 shows variations of the experiment that we conducted. We first started with adding noise to images that are, at the start, already recognizable by humans, as shown in Figure 9a. From this image, we then added noise until the convolutional neural network had classified it as a stop sign at a high confidence prediction. The result of the image that tricked the network is shown in Figure 9b. In this final image, the yield sign is still reasonably noticeable, mainly the triangular shape. Potentially, if the images were more extensive than the 32x32 that we experimented with, the final ideal image would be more recognizable. This is because the added noise and the increase in pixels allow the image to stay more recognizable[3].

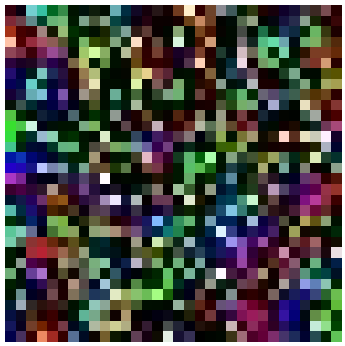
The results were interesting because different traffic signs tricked the network, but it also let us expand upon this and start with different variations of images. The first image that we started with was completely black, as seen in Figure 9c. From here, we followed the same process as the other images to add noise to it until it was classified as a stop sign. The resulting image is in Figure 9d. This surprised us as the image looks like nothing resembling any traffic sign but randomly colored pixels. Another similar result can be seen in Figure 9e and Figure 9f. This time the start image is all white, and we see a different result than the all-black image. In the all-black image, the result was an image with a few colored pixels. However, with the white end image, it had more color than white pixels. We believe this may be impart because white pixel values are at 255 and thus adding to these pixels caused the images RGB to clipped. Moreover, this resulted in a more colorized image that was still classified incorrectly.



(a) Normal Yield Sign **(b) Yield sign tricked into a stop sign**



(c) Fully black image **(d) Black tricked into stop sign**



(e) Fully white image **(f) White tricked into stop sign**

Figure 9: Variations of images been tricked into being classified as a stop sign.

5.3 Randomly Generated Starting Images

We expanded upon the idea of varying starts of images to see if any image with enough noise added could be misclassified as a stop sign. These results can be seen in Figure 10. Here we have the results of 40 randomly generated starting images to be processed through our random search algorithm. The results of the generated images are shown in Figure 10a. Once all the images were generated, they were processed. Figure 10b shows the results of this.

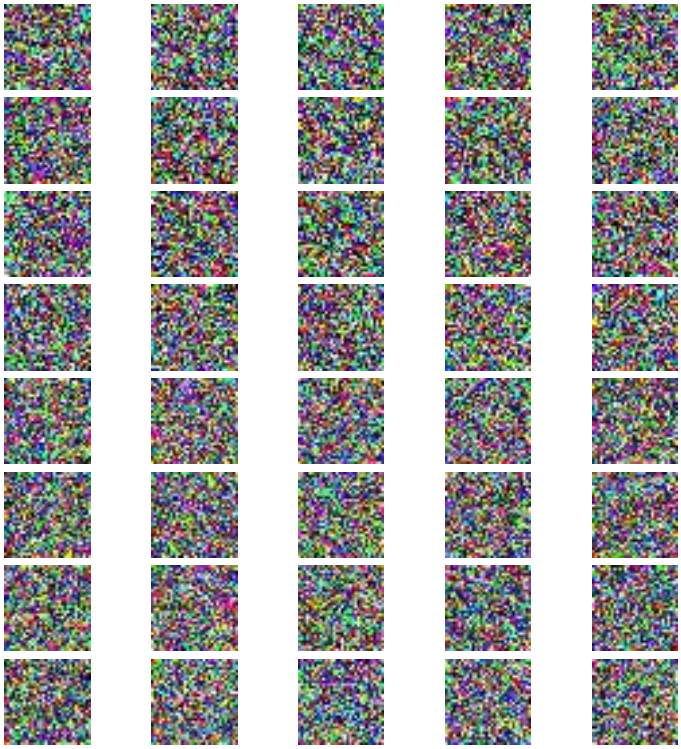
Every randomly generated image reached a point where the image would have enough noise added to it that it would be classified incorrectly with a confidence of around 99.99%. This result demonstrates that the convolutional neural network was easily tricked and could be tricked by an image given enough noise added to it. We found some exciting phenomena that all of the ideal images were significantly darker than the starting image. Moreover, there tend to be pockets of pixels that are all black in random areas. To purely speculate, we believe this could occur because the dataset is a real-world collection of images; therefore, many of the traffic signs were occluded by things such as shadows and dirt, and some images were taken at night. In order to test this, we would have to find a dataset that does not have images that contain these issues. With that dataset, we could train another network and compare the two networks ideal images.

We found another exciting phenomenon which was the next highest colored pixels after black were shades of yellow, red, and orange. Again to speculate, we believe that this is probably because most traffic signs are of those colors. This also leads us to a similar result; parts of the starting images that were purple tend to shift to blue or more violet shade of purple. We believe this could be happening because blue is a darker color and could have a similar effect as the black portions.

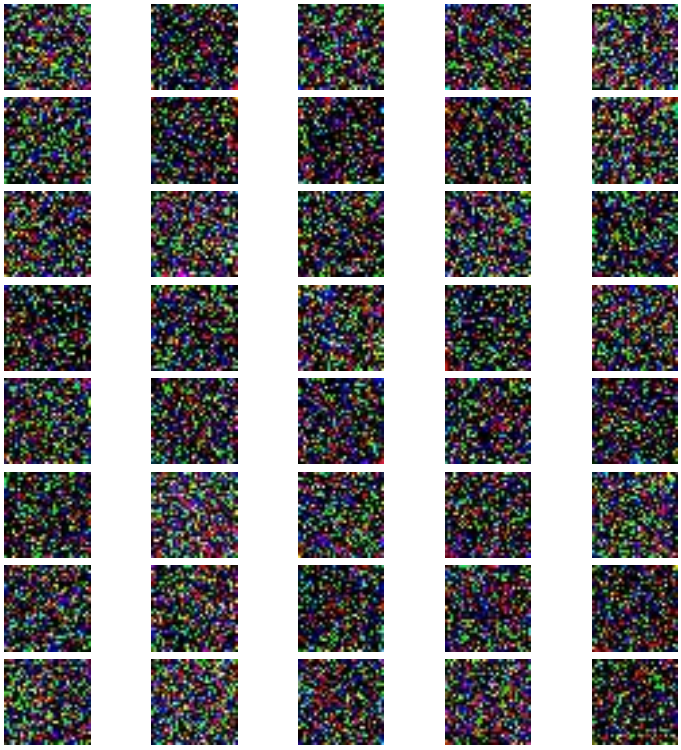
5.4 Yield Sign to Stop Sign to Yield Sign Again

The following experiment we wanted to look at was reclassifying an image. Specifically, we wanted to take a standard image from the dataset used by our convolutional neural network, pass the image through our random search algorithm until it is classified at a high confidence prediction of a different traffic sign. From here, we then would pass that resulting image through our random search algorithm until it was classified at high confidence of what the image was originally. This experiment can be seen in Figure 11. The first image is a standard yield sign with no alterations. The following image is the resulting image of noise being added to it until it was classified as a stop sign. Finally, the last image is the incorrectly classified image with noise added until it was again classified as a yield sign like the original image.

Looking at Figure 11, we have some exciting results. Firstly, as explained earlier in section 5.2 with Figure 9, the incorrectly classified image still looks recognizable as a yield sign. However, the most recent image is the more interesting. In the last image the triangle of the yield sign is still visible;



(a) Starting randomly generated images



(b) Incorrectly classified randomly generated images

Figure 10: Variations of starting random images

however, it is relatively faint. Moreover, the image is darker. This follows the results found in the randomly generated starting images. As it found ideal images would have darker portions.

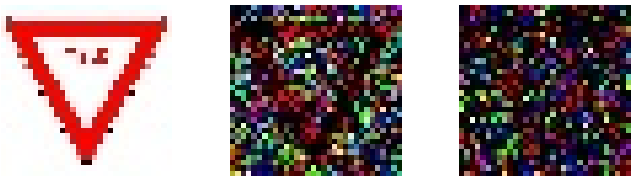


Figure 11: Yield sign tricked into stop sign and then changed to be classified as a yield sign

5.5 Subtracting Max From Original

Based on the previous experiment, we wanted to investigate further the image's changes by looking closely at the pixel difference between the resulting images. This can be seen in Figure 12. First, the image of the yield sign is run through our random search algorithm and is incorrectly classified as a stop sign. From here, we then subtracted the pixel values from the original image to the incorrectly classified one. The resulting image is the last in this figure.

This subtracted image surprised us as the pixels that are not black depict the triangular shape of the yield sign still. Furthermore, there are remnants of the text of the yield sign inside of the triangle. This was interesting to us because the overarching difference with the noise still displays a uniquely defined yield sign but is still classified incorrectly. Passing the last image through the classifier, its highest classification was at 47.3% and was classified as a complementary keep left sign.

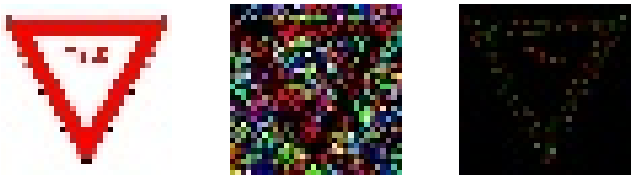


Figure 12: Yield sign classified as stop sign then pixels subtracted from original

5.6 Making More Classification

One of the last experiments we wanted to run was to see if the model could be tricked into increasing the classification of an already highly classified traffic sign. This can be seen in Figure 13. The first image is the original stop sign with no added noise. After being passed through our convolutional neural network, it is classified as a stop sign with a confidence prediction of 99.948%. This image was then passed through the random search algorithm, and the final image was produced. This image's confidence prediction was now exactly 100%. We found this interesting as the resulting image still had traces of the original stop sign. We believe this shows the robustness of this network and is trained well enough to be closely related to human vision. This also indicates that a 100% classification is likely not what a human would think it would look like. The network classifying a sign with 95% or even 80% confidence is probably good enough, and a human will not tell the difference. Furthermore, 100% confidence is unlikely to happen in the real-world. This gave us hope of some robustness; however, the other results from the prior experiments make us believe there are vulnerabilities still to a given network.

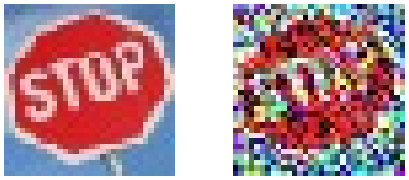


Figure 13: Stop sign classified with higher confidence prediction with added noise

5.7 Subtracting Max From Stop Sign

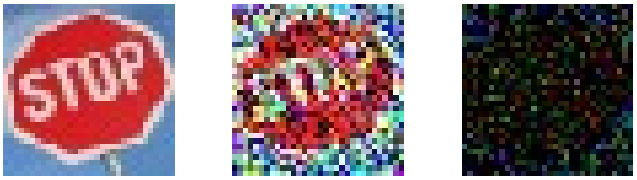


Figure 14: Stop sign subtracted from max stop sign

The last experiment we wanted to run was similar to the yield sign max subtraction experiment, but we thought it would be interesting to see if there was consistency with the original traffic shape being present after going through the random search algorithm. For this experiment, we subtracted the original stop sign images pixel values from the newly classified image and produced a new image from the difference. These images can be seen in Figure 14, which is clearly different the subtracting yield sign experiment in Figure 12. One can still clearly see the form, but it is not as distinct as the yield sign. This is most likely because the yield sign was mainly white, which is $(255, 255, 255)$ so the subtraction gets cut off at black, which is $(0, 0, 0)$. Most of the stop sign is not

white, so we should not get this effect.
Interestingly, we can still see a general shape of the sign. It ends up being classified as an "other-sign," with confidence of 97.95%. This tells us that the model learns the signs, and if it cannot tell what is going on, it just classifies it as another sign. It was interesting that the dataset had this as a class for that very reason.

6 FUTURE WORKS

6.1 Random Search Algorithm to Genetic and Evolutionary Algorithm

The potential for future work with this research has many different avenues. First, would be image sizing. As stated earlier, some constraints we encountered working with the Mapillary dataset[1] was its sheer size. 70GB is a lot of data to work with. Our computing resources did not allow us to use the images at full size. Future work would be seeing the classification's confidence predictions at varying sizes. What we like to investigate further is the trend of iterations of our random search algorithm for increasing image sizes. If there is a maximum image size in which our random search algorithm cannot effectively search the space (i.e. takes to long to find), this would lead us to have to use a new algorithm such as a genetic and evolutionary algorithm. If this is not the case, we would like to investigate Nguyen's algorithm and if similar results would be found with this dataset. Again, access to better computer resources would allow us to explore more. One thought was implementing these experiments on cloud computing services such as Amazon Web Services (AWS) or Microsoft's Azure.

6.2 Diverse Networks

We believe another future work for this research would utilize different convolutional neural networks along with more computer resources. As mentioned, the network used in this paper was a variation of the VGG16 neural network. We believe doing more research into other convolutional neural networks developed for this classification problem and comparing the images that trick them would be essential to understanding security and reliability. Specifically, looking at the convolutional architecture that companies such as Tesla and Uber use would be excellent to answer this question and would further this research.

6.3 Potential End Result

Lastly, we have thoughts on the result of this research. Ultimately, we believe the overarching idea would be to detect noise in an image and, depending on a set standard of noise, denoise the image and classify it. A paper by Joon Huang Chuah titled "Detection of Gaussian Noise and Its Level Using Deep Convolutional Neural Network"[4] expands on the idea of noise level detection in images. In the paper, Chuah explains his team's research in a convolutional neural network they developed to detect noise in an image. Using this network in conjunction with other convolutional networks to then denoise the image would be extremely interesting if this increases the reliability of the ending classifying network.

This does raise many questions, such as, what is the standard amount of noise that causes the network to justify an image being denoised? Or, is a denoised image still valid for traffic sign classification, or should it be thrown out as it may have been tampered with? Is this process fast enough for image classification in real-time? We believe that this research would have to be broken down into separate parts. First, test the noise images and find a standard for the application. Second, test images that have been tampered with and see if the newly denoised image is valid enough to be classified by human vision. Last, see if the process is quick enough for applications that demand real-time classification. Furthermore, can this process be done on current hardware available for self-driving implementations? The idea of this process can be seen in Figure 15.

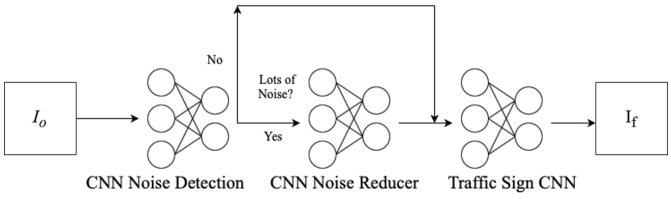


Figure 15: Future Work Process Flow [Diagram].

7 CONCLUSION

We have demonstrated that many convolutional neural networks can be easily tricked through the addition of noise through a standard normal Gaussian distribution. The networks were deceived in a reasonably quick manner through our random search algorithm. Specifically, an image was able to be tricked into being classified incorrectly in just under a minute. Being able to fool these types of networks so easily raises the question of the security, validity, and reliability of these networks to be used in areas of real-world application such as self-driving cars. Future work in detecting noise in an image and denoising an image before classifying may be a step in the right direction in having a convolutional neural network that is sound in security for these applications.

8 REFERENCES

[1] *"Street-level imagery, powered by collaboration and computer vision," Mapillary. https://www.mapillary.com/dataset/.*

[2] *S. F. Shafiei, TRAFFIC SIGN DETECTION USING DEEP LEARNING BY GROUP 36, 2019.*

[3] *Nguyen A, Yosinski J, Clune J. Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images. In Computer Vision and Pattern Recognition (CVPR '15), IEEE, 2015.*

[4] *Proc. of the 2017 IEEE Region 10 Conference (TENCON), Malaysia, November 5-8, 2017*

[5] *A. Bohm, Personal conversation, March 2020*