TRAFFIC SIGN DETECTION USING
Generative Adversarial Networks (GAN)

by, Abdullah Kose, Abhinay Munagala, Siddharth Thorat and Siva Jyothi Karna.

Course Instructor: Dr. Bing Li

Course Teaching Assistants: Zongming Yang & Ziyue Feng

Clemson University International Center for Automotive Research (CU-ICAR),

5 Research Dr, Greenville, SC (South Carolina) 29607

Table of Contents

	Contents	Page No.
1	Abstract	3
2	Introduction	3
3	Background and Related Works	3
4	Dataset	4
5	Methodology	4
5.1	Generator Network	5
5.2	Discriminator Network	6
5.3	Selecting and Training Networks	7
6	Results	10
7	Conclusion	16
8	References	16

1. Abstract

In this paper, seven stop sign detection neural networks are trained using publicly available stop sign datasets and MathWorks standard Generative Adversarial Network (GAN) example code. Five initial networks are trained on an initial dataset and then tested using a different dataset. The results of the testing are used to guide the training of two additional networks. The performance of all eight networks is examined and the best network is chosen. Additional steps for the training of better networks and generation of better datasets are identified.

2. Introduction:

Every year, more than 38,000 people die every year in crashes on U.S. roadways. The U.S. traffic fatality rate is 12.4 deaths per 100,000 inhabitants. An additional 4.4 million are injured seriously enough to require medical attention, as reported by *Association for Safe International Road Travel (ASIRT)* [1]. The 8th most common death reason in the world contributes to traffic related accidents. Among all reasons leading to traffic accidents, not following traffic signs is one of the leading causes for crashes. Therefore, the primary goal of our project is to detect and recognize upcoming traffic *stop* signs. The use of General Adversarial Networks (GANs) in Machine Learning is a small approach for detecting the stop signs.

3. Background and Related Works

The detection and recognition of traffic signs is a popular machine learning technique, as it helps to improve the on-road driving safety of the vehicle and assist autonomous driving. We studied some related works to lay the foundation and improve for our project.

We referred to a research paper focusing on the detection and localization of traffic signs, the research explains about an algorithm developed combining “white balance, color image enhancement, and affine transform correction” [2]. This method has been applied successfully to detect traffic signs under real world conditions, including extreme weather conditions.

We referred to another research paper that focuses on "Unsupervised Representation Learning with Deep Convolution General Adversarial Networks" [3].

4. Dataset

We have used multiple datasets to train the network.

Firstly, we have used an opensource website Kaggle, from which we have considered the "Road Sign Detection" which has 877 images classified into 4 different classes of Traffic Light; Stop; Speed limit; Crosswalk. For this project we have concentrated only on the STOP sign.

We have also made use of one of our teammate cars dashcam as a dataset as well. One of the drawbacks from the initial dataset is we didn't have any STOP signs captured in the nighttime. We have some from the dashcam dataset, we could get only 9 STOP sign images from the 9GB of data recorded on the dashcam. We had some limitations on the nighttime stop sign images. As the network has not been trained for the nighttime stop signs. We have had bad detection results for the nighttime stop signs.

We have also used the GTSRB, German Traffic Sign Recognition Benchmark dataset which consists of all the traffic signs, not just the stop sign.

https://github.com/mbasilyan/Stop-Sign-Detection positive dataset from this GitHub repository has also been utilized for training our network.

5. Methodology

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

A GAN consists of two networks that train together:

Generator - Given a vector of random values (latent inputs) as input, which is nothing but noise, this network generates data with the same structure as the training data. Parallelly, we will input the ground truth images as well as the input to the discriminator. [5]

Discriminator - Given batches of data containing observations from both the training data, and generated data from the generator, this network attempts to classify the observations as "real" or "generated/ fake". [5]

This image illustrates the structure of GAN

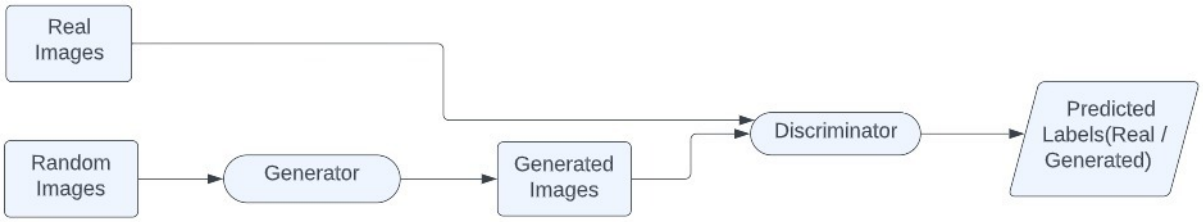

Figure 1

5.1 Generator Network

This network architecture generates images from random vectors. The below figure explains a Generator Network.

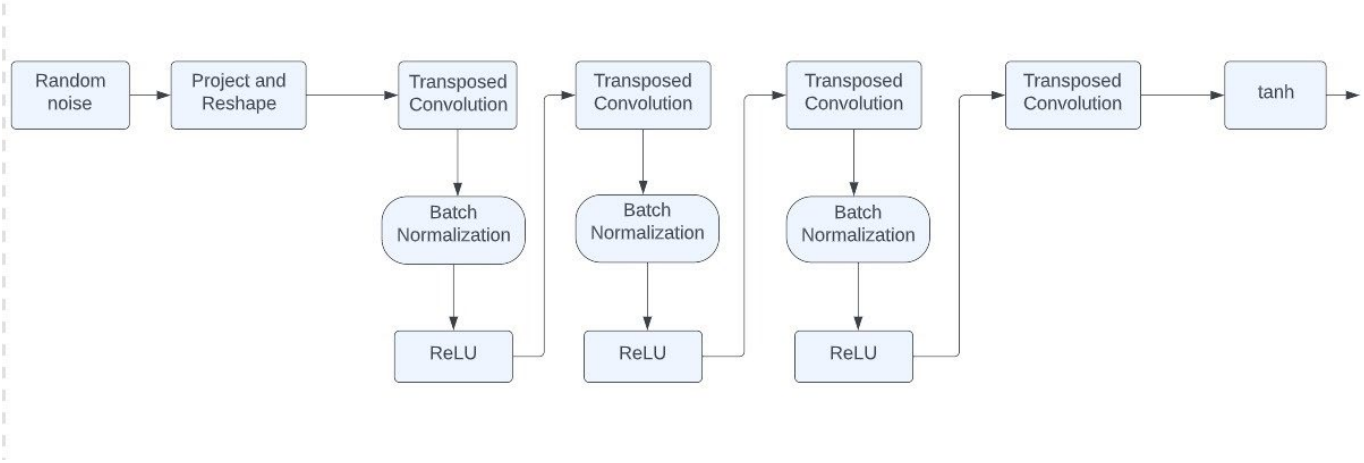

Figure 2

The Generator Network:

- Converts the random vector of the size 100 to 4-by-4by-512 arrays using a project and reshape operation.

- Upscales the resulting arrays to 64-by-64-by-3 arrays using a series of transported(transposed) convolution layers with batch normalization and ReLU layers.
- Last Transposed convolution box without any layers is the final output and a tanh function is used as it delivers better performance for multilayer convolutional neural network layers. The biggest advantage of the tanh function is that it produces a zero-centered output, thereby supporting the backpropagation process.

5.2 Discriminator Network

This network takes 64-by-64-by-3 images and returns a scalar prediction score using a series of convolution layers with batch normalization and leaky ReLU layers.

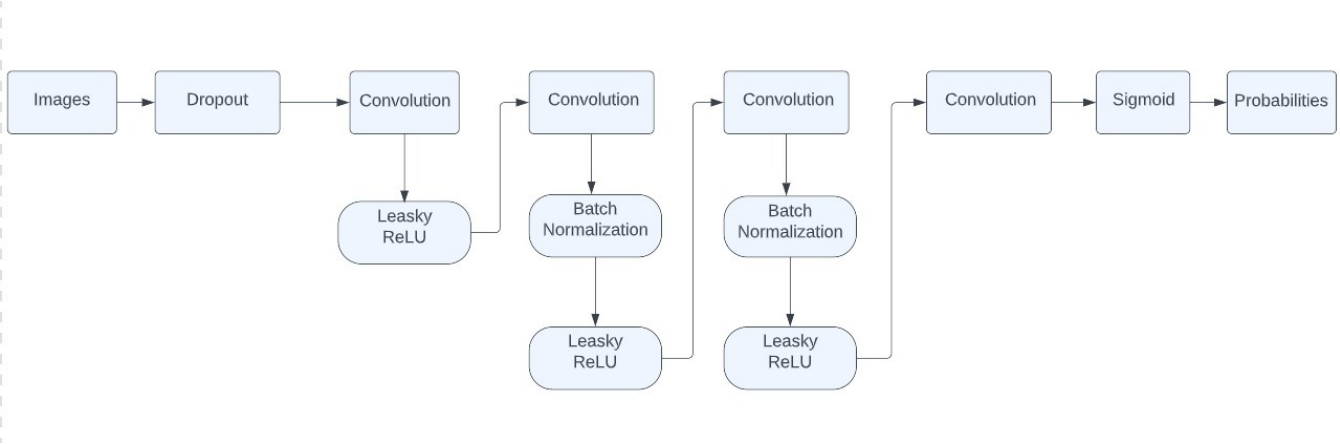

Figure 3

Add noise to the input images using dropout.

- For the dropout layer, dropout probability is 0.5.
- For the convolution layers, 5-by-5 filters with an increasing number of filters for each layer. Also, stride is 2 and padding of the output.
- For the leaky ReLU layers, scale is 0.2.
- To output the probabilities in the range [0,1], a convolutional layer with one 4-by-4 filter followed by a sigmoid layer is taken into consideration. Sigmoid function probably gives the output as either 1 or 0. And the output need not be passed through another layer or

network. This would be the best function to decide the output of images to be real or fake.

5.3 Selecting and Training Networks

MATLAB's GAN example code provides a working baseline that can be used to train networks out of the box by providing only a novel dataset. The initial example mirrors the input dataset images by default to provide more images with which to train the network. MathWorks Deep Learning Toolbox allows for a range of different settings to be changed while training. Several of these options were selected and changed to observe the effect on accuracy. Mirroring, learn rate, epochs, and batch size were selected as the variable features. The changes that result in performance improvements over the default network will later be combined to create the final network.

Table 1

Network	Mirroring	Learn Rate	Epochs	Batch Size
1	yes	default	100	30
2	no	default	100	30
3	yes	10x slower	100	30
4	yes	default	100	30
5	yes	default	100	60
6	yes	default	200	30

These settings were used to train the networks on laptop Nvidia GPUs with the Matlab Parallel Computing Toolbox.

The training results of these networks are as follows:

1. Default setting with mirroring

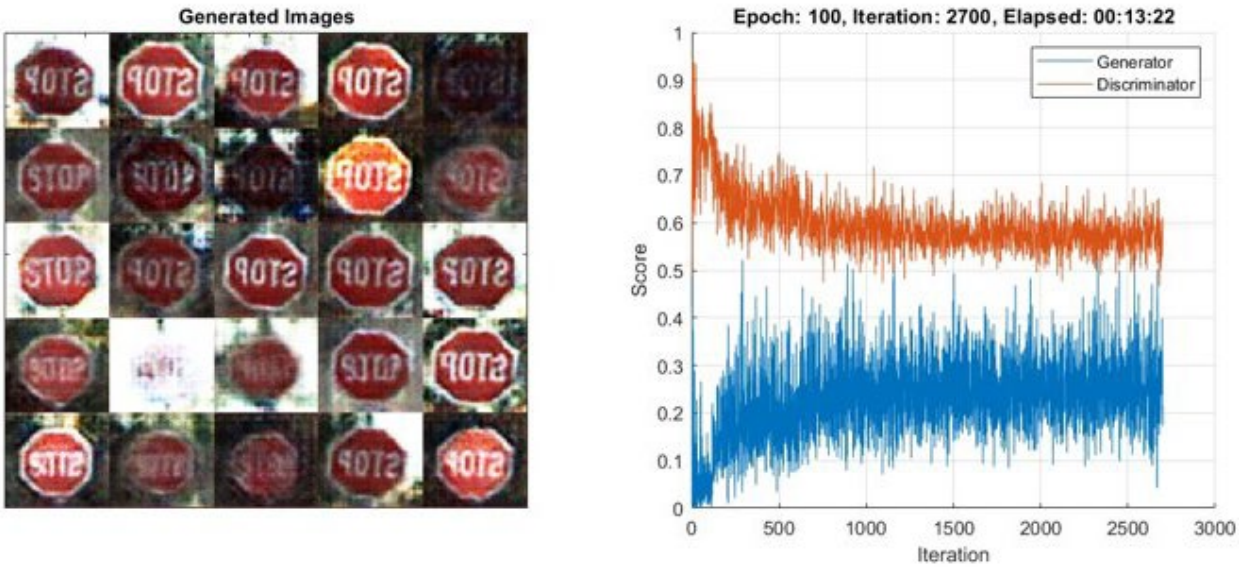

Figure 4

2. Default setting without mirroring

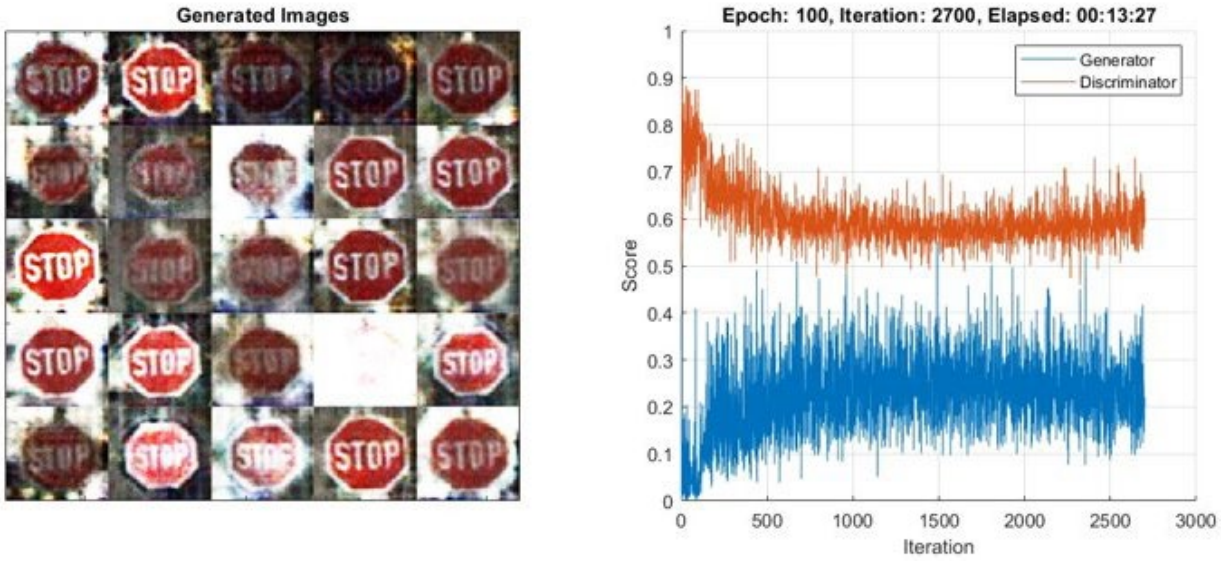

Figure 5

3. Default setting, but ten times slower learning rate

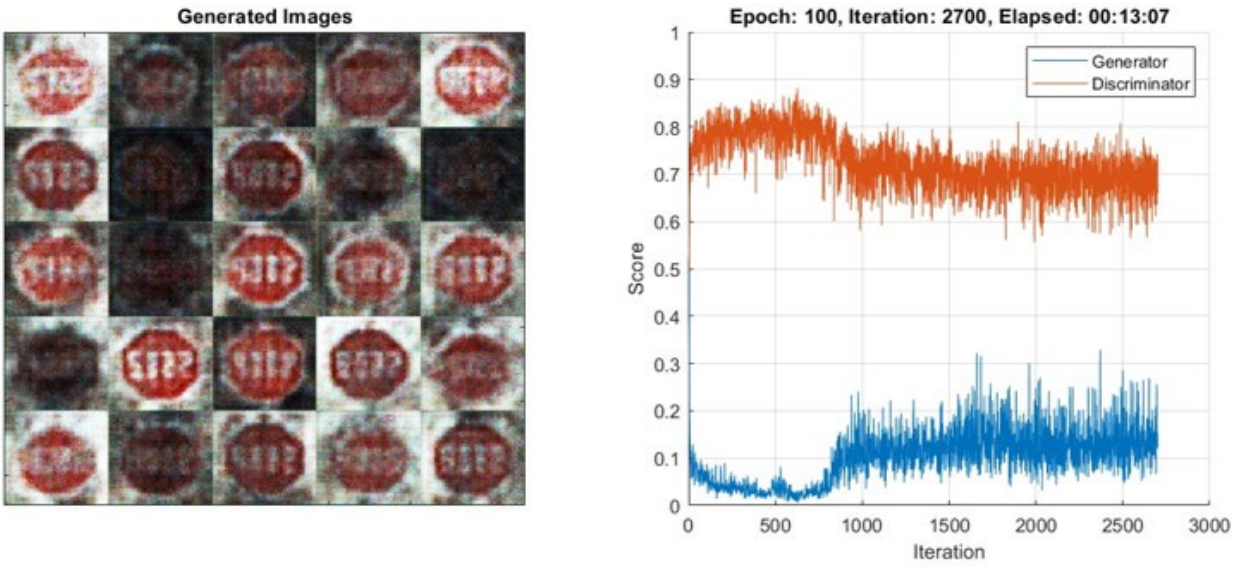

Figure 6

4. Default setting, but then ten times faster learning rate

- This combination of settings would not run in MATLAB. It was never fixed and left out for the sake of time.

5. Default setting, but with 100 epochs and a batch size of 60.

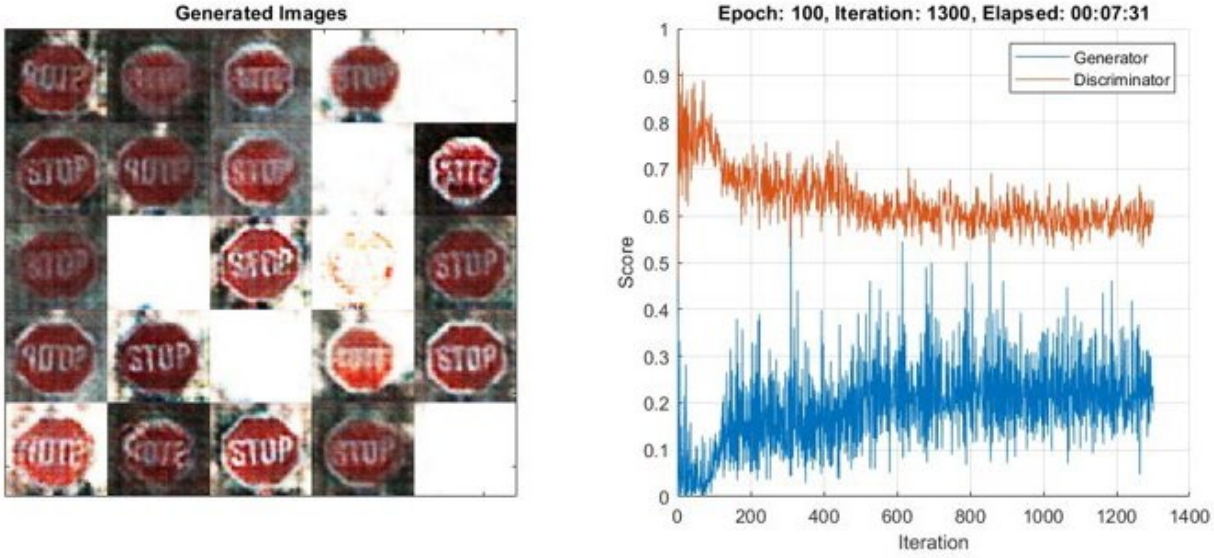

Figure 7

6. Default setting, but with 200 epochs and a batch size of 30.

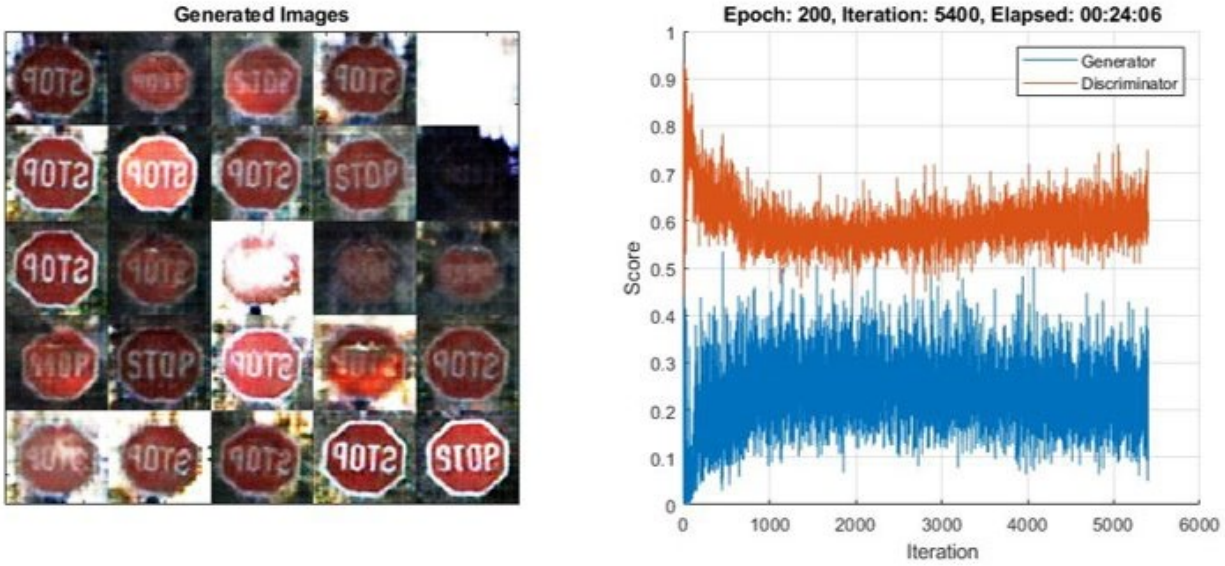

Figure 8

6. Results

In order to test the performance of the trained networks, A separate folder with test images was kept that contained the images from a separate dataset. The goal was to create networks that would detect stop signs under myriad conditions. For this reason, the test dataset chosen was not a subset of the training dataset, but instead an entirely different stop sign dataset from different authors. The freely available dataset from https://github.com/mbasilyan/Stop-Sign-Detection was chosen for testing purposes. Several non-stop sign categories were chosen from the Several non-stop signs from the original dataset were chosen to test the networks' ability to reject alternative road signs. The combined test dataset included 97 stop sign photos and 5,463 European non-stop signs.

The initial training dataset consisted of small resolution photos that show stop signs with almost no background. The network input resolution is 64x64, so larger images would not have performed well once compressed to that resolution. The test stop sign images consisted,

however, of the stop sign and the surrounding environment. The networks all performed poorly once these images had been downscaled to 64x64.

A simple search region algorithm was implemented to ameliorate the problem. The mat2tiles [4] function was used to split each image into 200x200 pixel overlapping regions. If the entire image was not classified as a stop sign when compressed into the 64x64 input file, these smaller regions were searched for stop signs. If a stop sign was detected in one of these regions, the rest of the regions were ignored to speed up processing times. In practice, this means that the location of the stop sign within the image would influence the processing time required for a prediction. Given that stop signs are almost always on the right side of the road at a nearly fixed height, the order of searched region could be optimally selected for a specific field of view to reduce computing resource load. Optimal regions search order was not implemented in this project because the network ran in near real time with a live 1080p webcam input.

The images described earlier were fed into all six networks and evaluated. The fraction of true positives is reported as "score" in the table below, and the fraction of true negatives is reported as "1 – Loss".

Table 2

Network	Score	1 - Loss	Loss
1	1	0.1329	0.8671
2	0.433	0.9806	0.0194
3	0.1026	0.9826	0.0174
5	1	0	1
6	0.8969	0.8816	0.1184

The default settings performed well in identifying stop signs, but often produced false positives with the non-stop signs. The second network showed good performance in rejecting non-stop signs but missed more than 50% of true stop signs. The third network exhibited the inverse behavior of the first.

The fifth network was no more useful than the following code:

```
While true
    img = snapshot(cam)
    disp('This is a stop sign')
end
```

The sixth network seemed to be a good balance and exhibit desirable characteristics. A combination of network six and network two was decided on for the final network choice. It was theorized that although removing mirroring might reduce the detection performance of network six, it might improve the false positive performance. Vehicles driving near stop signs have many frames to work with as they approach a stop sign, so slightly compromising the score to reduce the loss was seen as an acceptable choice. It was observed from repeated training of networks 5 and 6 that a doubling of epochs was more time efficient than doubling the batch size, so for network 7 the epochs were set to 500.

Table 3

Network	Mirroring	Learn Rate	Epochs	Batch Size
7	no	default	500	30

The training behavior of network 7 is shown below.

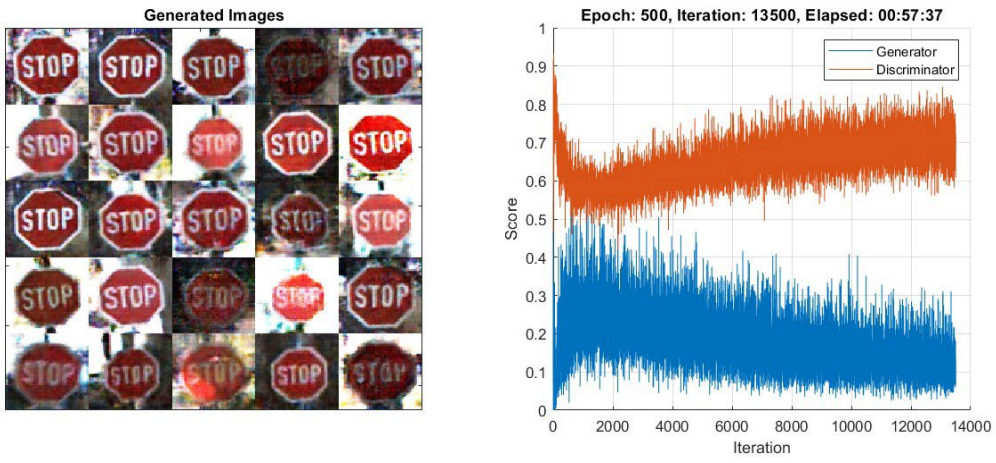

Figure 9

It is observed that the signs produced by the generator in this 7th iteration are clearer than any of those previously produced. Unlike previous networks however, the generator and discriminator

network scores diverge, with the discriminator dominating the generator instead of converging at 0.5 as is optimal. A feature exists within the baseline Mathworks code to flip the labels of real and fake images to allow the generator to catch up with the discriminator in score, but this was not changed from the default value in order to maintain the original project plan. A future step would be to increase this flip rate in order to prevent discriminator domination.

Network 7's performance is as follows:

Table 4

Network	Score	1 - Loss	Loss
7	0.458	0.9897	0.0103

The testing score and loss did indeed move in the predicted direction, but the tradeoff between loss and score was much larger than expected. It may show promise if trained for longer with a larger dataset, or with the aforementioned larger flip rate.

Network 7 was retrained as Network 8 for 300 epochs instead of 500 to see if the network performance degraded as iterations increased and training scores diverged. Mirroring was added back to see if performance would return.

Table 5

Network	Mirroring	Learn Rate	Epochs	Batch Size
8	yes	default	300	30

Its training behavior and performance are as follows:

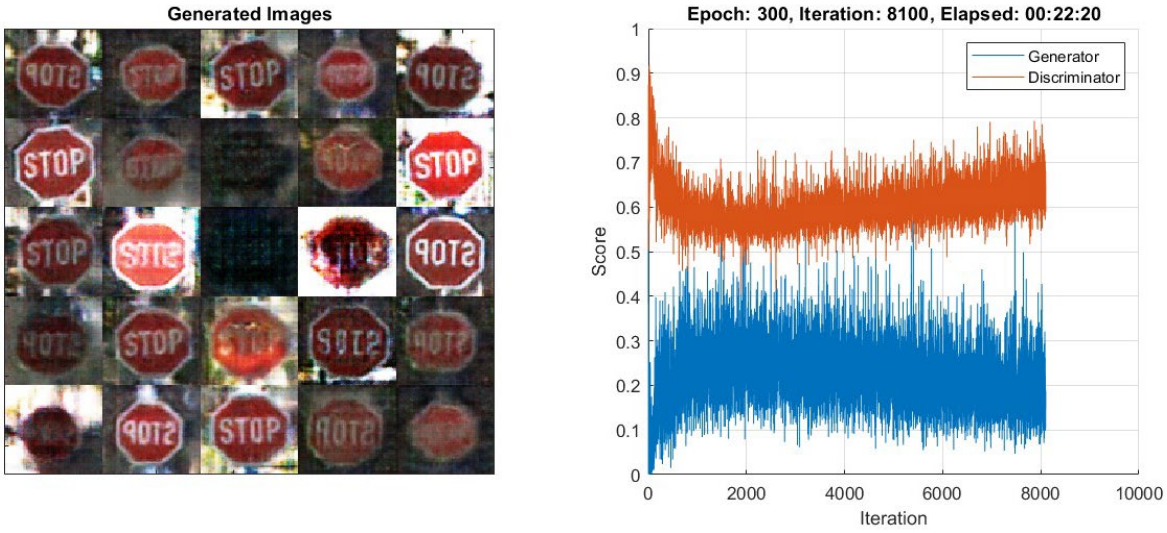

Figure 10

Table 6

Network	Score	1 - Loss	Loss
8	0.7827	0.2474	0.7526

In this case, although the training scores were closer together, the final network performance was worse due to the reduced training. The quality of produced images is clearly reduced.

The performance of every network when subjected to the test dataset is compared in the table below.

Table 7

Network	Score	1 - Loss	Loss
1	1	0.1329	0.8671
2	0.433	0.9806	0.0194
3	0.1026	0.9826	0.0174
5	1	0	1
6	0.8969	0.8816	0.1184
7	0.458	0.9897	0.0103
8	0.7827	0.2474	0.7526

Network six was chosen as the best compromise between the networks and was chosen as the network used to evaluate dashcam stop signs.

Nine stop signs were identified in 9GB of video from a dashcam and were saved as screenshots. Three of these signs were captured in the nighttime and six were captured in the daytime. Network six was able to identify all of the daytime photos except one, but only one of the nighttime photos. dashcam1.png was the only night stop sign to be identified. Upon inspection of our training and test datasets, we discovered that many of the dark photos in the set are daytime photos with a universally lowered exposure. The brightness ratio between the stop signs and the background is kept the same, but total brightness is decreased. Nighttime photos would exhibit a larger difference between the brightness of the background and foreground. Although our search was not exhaustive, we did not discover a dataset that focused on or had a large portion of signs at night. The unique reflective properties of stop signs revealed only at night may influence training in a way that improves network performance when tested with such images.

Figure 11

7. Conclusions

In the end, experimentation with the selection of training options led to a network that successfully rejected 88.1% of non-stop signs and correctly identified 89.7% of stop signs. The network architecture provided by Mathworks is lightweight enough to run in near real-time on a laptop with 1080p input images (15-30 FPS depending on search region). It was observed that there may be opportunities in the creation of a public nighttime traffic sign dataset, with unique features like headlight glare or reflective sign features. Although the two final networks did not improve on the performance of network six from the first training round, their training score behavior left clear next steps to take for improvement.

8. References

[1] https://www.asirt.org/safe-travel/road-safety-facts/

[2] "Study on Detection and Localization Algorithm of Traffic Signs from Natural Scenes", https://journals.sagepub.com/doi/full/10.1155/2014/318635

[3] "Unsupervised Representation Learning with Deep Convolution General Adversarial Networks" https://arxiv.org/pdf/1511.06434.pdf

[4] mat2tiles: https://www.mathworks.com/matlabcentral/fileexchange/35085-mat2tiles-divide-array-into-equal-sized-sub-arrays

[5] Definition of Generator and Discriminator. https://www.mathworks.com/help/deeplearning/ug/train-generative-adversarial-network.html

Dataset references:

https://www.kaggle.com/datasets/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign

https://www.kaggle.com/datasets/andrewmvd/road-sign-detection?resource=download

https://github.com/mbasilyan/Stop-Sign-Detection/tree/master/Stop%20Sign%20Dataset