Attacking Autonomous Driving Machine Learning Algorithms with Adversarial Examples

Stephanie Tietz (06106189) Koosha Nassiri Nazif (05950924) {stietz, koosha}@stanford.edu

***Abstract*— In this work, we study the use of Generative Adversarial Networks (GAN) to attack a state-of-the-art (SOTA) traffic sign recognition (TSR) classifier used for autonomous driving. Using GAN, we generate inconspicuous adversarial perturbations to traffic sign images and feed the perturbed images to the TSR classifier in an attempt to cause targeted misclassification and a reduction in the accuracy of the classifier. We also study the effect of various hyperparameters on the overall performance of the GAN, resulting in an optimized model that achieves a significantly high misclassification of 82.9 percent, reducing the test accuracy from 92.1 to 17.4 percent.**

***Index Terms*—Autonomous Driving, Generative Adversarial Network (GAN), Neural Network**

I. Introduction

SAFETY is the most important aspect of autonomous driving. Machine learning (ML) algorithms employed in self-driving units thus need to be robust against adversarial attacks. One of these ML algorithms is the traffic sign recognition (TSR) which interprets the traffic signs seen on the road. In this study, we use Generative Adversarial Networks (GAN) to attack a state-of-the-art (SOTA) traffic sign recognition (TSR) classifier. The results give us insights on the sensitivity of the TSR classifier to adversarial attack as well as possible ways to increase the adversarial attack success rate or to improve the robustness of TSR against the GAN attack.

Our model contains two networks working in tandem – one GAN network with a generator linked to a discriminator and one target network that we plan to attack. The target classifier is trained to recognize various traffic signs. We feed the generator an image from one class, e.g. a stop sign, and use the GAN to perturb the image until the target classifier misclassifies the image as another sign, e.g. a speed limit sign.

The structure of the classifier network is based on a LeNet-5 model implemented by Mohammed Ameen who was able to see a validation accuracy of 95.3% on the German Traffic Sign (GTS) dataset [1]. Xiao et al. used a GAN to generate adversarial examples against an MNIST classifier in 2018 [2]. Their GAN, called AdvGAN, applied perturbations and generated examples that were used to perform white-box, semi-white-box, and black-box attacks. We will use a similar structure, applied to this new set of traffic sign classes. Varying the hyperparameters of the GAN network, we then optimize for highest attack success rate and misclassification. Finally, we provide suggestions on how to increase the classifier robustness against the GAN attack.

II. DATASET

In building our target classifier, we used the German Traffic Sign (GTS) Dataset which contains more than 50,000 images in 43 unbalanced classes [3]. Image sizes vary between 15x15 to 250x250 pixels (RGB). All data were converted to 32x32 RGB images. The data was divided into 67% training, 8% validation, and 25% testing datasets with relative class size remaining constant in each set (See Fig 1).

A. Data Preparation and Preprocessing

Our original intention for the target network was to include a class called "not a sign" to make sure we were not forcing the target network to choose a sign class based on an image that no longer looked like a sign. We later decided to keep the perturbations small in the GAN so that the generated images would still look like a sign (specifically the original sign class) to a human, and remove the "not a sign" class, as explained in the next section.

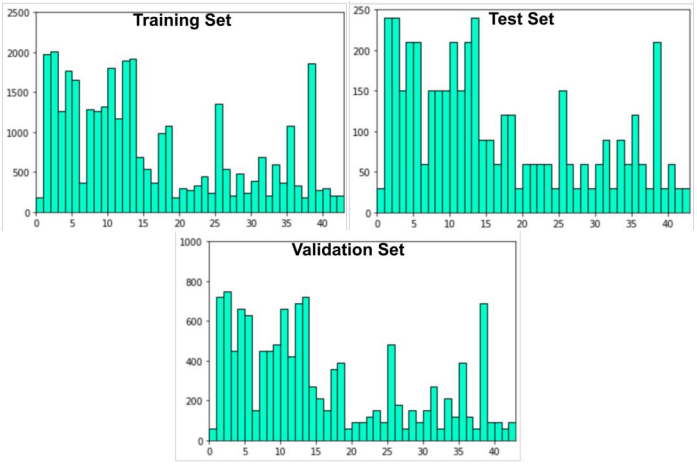

Fig. 1. Histogram representation of class distribution in the German Traffic Sign dataset's training, test, and validation sets.

To make the dataset with the "not a sign" class, we took a random assortmenet of images from the CIFAR10 dataset [4], 32x32 RGB images, assigned them all the same class label, and added them to the GTS dataset while keeping the relative class size the same. All images in the training set were shuffled to increase randomness. Then images from every set were normalized and grayscaled, reducing image size down to 32x32x1. Sermanet and LeCun found that grayscale images were classified more accurately by convolutional networks [5]. However, we also fed color images to test this hypothesis, as explained in the experiments section.

III. Methods

A. 3-Class Classifier with CIFAR10

The structure of our network is based off a LeNet-5 model implemented by Mohammed Ameen who was able to see a validation accuracy of 95.3% on the GTS dataset [1] (see Fig 2). As a preliminary step, we decided to choose two sign classes (a circular sign that read 30km/hr and a triangular road work sign) as well as the "not a sign" image class. This smaller dataset had only a training set and a test set (70/30 split). While we found we could overfit the model, we quickly realized this model was not learning to distinguish between the two signs and the random images that made up the "not a sign" dataset. As seen in Table 1, the test accuracies for every optimization attempt remained at 36.84% (exact up to the 15 decimal places output by the model). This suggests that the model was simply choosing one class to guess

instead of looking for deeper features in the images. The only time this changed was when the Adadelta optimizer was used instead of Adam, but even that only raised the accuracy to 42.21%. Plotting the test loss also suggested strong issues with the model because every attempt had high loss rates and big fluctuations between epochs.

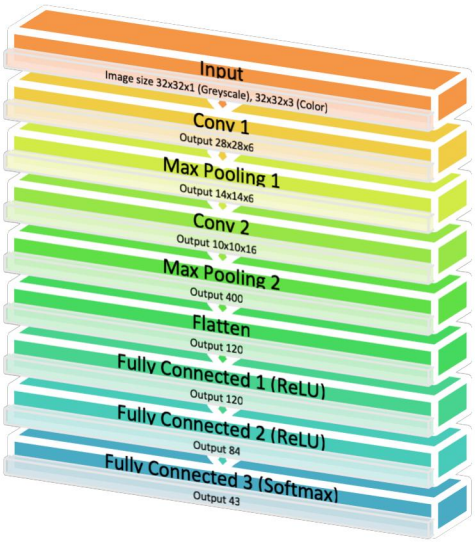

Fig. 2. LeNet-5 architecture used for the 3-class and TSR classifiers

We also developed our own fully connected classifier that has one hidden layer of 25 units and ReLU activation function, and softmax activation function for the output layer. Although more accurate, as seen in table 1, it still suffers from overfitting.

3-Class Model Description	Train Accuracy	Test Accuracy
Base model: sparse categorical cross entropy, Adam optimizer, lr=0.01	99.98%	36.84%
Base + 25% dropout before FC1, 25% before FC3	92.54%	36.84%
Base + 25% dropout before FC1, 50% before FC3	87.42%	36.84%
Categorical cross entropy, Adadelta optimizer, lr=1.0, 25% dropout before FC1 and FC3	80.69%	42.21%
In-house FC classifier, categorial cross entropy, Adam optimizer, lr = 0.00001, L2 regularized ($\lambda = 0.1$)	100%	54.26%

Table 1. Various attempts to classify the 3-class classifier

We also ran the full GTS+CIFAR10 dataset on these models and saw the test/validation accuracy drop to 7.8% and remain there for each iteration with the loss again seeing big variations. To combat this issue, we re-evaluated whether these random images would be necessary. Their purpose was to ensure that the classifier had another option if it did not believe the input image looked like any of the signs it learned because we feared the GAN would generate a very noisy image that would not fit into any of the classes, but may be forced into one since the classifier had to assign probabilities to it based on the classes which it knew. Then we restructured our attack plan so that the images put into the generator would start as one of the traffic sign images and only small perturbations would be allowed. In this way, the image will not be allowed to get so noisy that, for instance, a human could not tell the original class label.

B. Traffic Sign Recognition (TSR) Classifier

Focusing solely on the 43 traffic signs, our results improved dramatically as we expected. Using the training, validation, and test sets described earlier, we trained a few variations of classifiers. The base model was the model used by Mohammed Ameen which implemented the Adam optimizer with a learning rate of 0.001 as well as a dropout of 50% for every fully connected layer and a dropout of 30% for every convolution layer. While he reported a max validation accuracy of 95.3%, we only saw 91.8%. This discrepancy may be due to an additional preprocessing step in Ameen's model, not implemented in ours, which involved local histogram equalization (enhancing images by spreading out the most frequent intensity values). However, a few modifications to the model conditions brought our validation accuracy up to 94.5% with a test accuracy of 92.1%. For our purposes, and considering the large number of classes involved, these accuracies are enough to move forward with this as our target network that we will attack. Time permitting, we can return and improve the TSR model even more to challenge our GAN.

Our target classifier network used a batch size of 64 along with 50% dropout on all layers (in the training set only). The Adam optimizer with a learning rate of 0.001 was found to be a successful choice (Table 2). The confusion matrix for this final network is seen in Figure 3.

TSR Model Description	Validation Accuracy	Test Accuracy
Base model: Adam optimizer, learning rate = 0.001, 50% dropout on FC layers, 30% dropout on Conv layers	91.8%	89.9%
Adadelta optimizer, learning rate = 1.0, 50% dropout on FC layers, 30% dropout on Conv layers	88.6%	87.9%
Base model + 40% dropout on FC layers, 20% on Conv layers, batch size of 64	93.8%	91.2%
Base model + 50% dropout on all layers, batch size of 64 ***(chosen model)***	94.5%	92.1%

Table 2. Model iterations for the traffic sign recognition classifier

We also studied the significance of each fully connected layer by monitoring how the test accuracy, recall, precision and F1 score change as we remove each fully connected layer from the neural network. The results are shown in table 3. As can be seen in table 3, removing FC layers 1 and 2 has nearly the same small effect on model performance, reducing all metrics by about 1 percent. This demonstrates the low significance of these layers compared to the preceding convolutional and pooling layers.

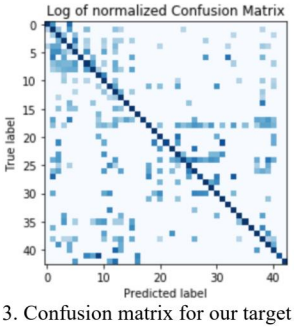

Fig 3. Confusion matrix for our target network

TSR Model Description	Test Accuracy	Test Recall	Test Precision	Test F1 score
Optimized Model	92.1%	92.94%	93.21%	92.91%
Optimized Model w/o FC layer 1	91.1%	91.79%	92.37%	91.71%
Optimized Model w/o FC layer 2	91.0%	91.69%	92.38%	91.72%

Table 3. effect of removing each FC layer on ovrall the model performance

C. *AdvGAN*

Xiao et al. used a GAN to generate adversarial examples against an MNIST classifier in 2018 [2]. Their GAN, called AdvGAN, applied perturbations and generated examples that were used to perform white-box, semi-white-box, and black-box attacks. We will use a similar structure, applied to this new set of traffic sign classes.

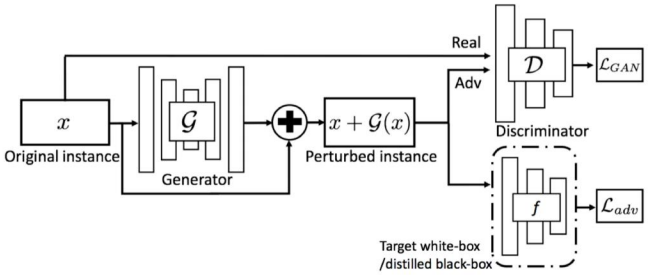

Fig 4. Overview of AdvGAN [2]

In order to opimtize the GAN in the model, the losses include generator loss and discriminator loss. These losses are in direct competition with each other when both are minimized as the generator is trying to create images that fool the discriminator and thus reduce generator loss while the discriminator is trying to correctly determine whether the image is real or generatoed, minimizing the discriminator loss. Specific to this AdvGAN model, we also include a perturbation loss and an adversarial loss. Minimizing the perturbation loss reduces the amount of noise that is added to the image to hopefully find the lowest perturbation noise which will lead to misclassification. The adversarial loss comes from the pretrained target classifier when real and fake images are passed to that classifier. Adversarial loss increases when the fake image is classified correctly rather than misclassified. Thus minimizing this loss should result in more misclassifications. More details on the AdvGAN model such as the definition of loss functions can be found in reference [2].

IV. EXPERIMENTS

In this section, we study the effect of input image (color vs. black & white) as well as various GAN hyperparameters on the attack success rate, misclassification, and various losses used in the GAN design. The metrics studied include test accuracy and misclassification percentage in addition to the four distinct losses in the AdvGAN. In contrast to many machine learning models, this AdvGAN is seeking to decrease the test accuracy of the target classifier. Therefore we use this final test accuracy in comparison to the initial test accuracy as a metric to determine how well the model is performing. Another important metric is the misclassification percentage. This first tests the model with the original, unperturbed image. If this original image is misclassified, it is not included in the list of images that will be perturbed as it is an image that the classifier already cannot classify properly. If the real image is classified correctly intitially, then perturbations are added to the image and the target model is tested once again. If the model then misclassifies the image, that image is included in the misclassification percentage. This metric has the advantage of not double-counting those images which the model would have misclassified initially and thus acts a true success rate for the adversarial attack. Only images that were misclassified due only to the adversarial attack were output for visual inspection.

When looking at the loss mechanisms in the model, we expect to see the generator and discriminator losses working against each other and thus their losses should show almost a reciprocal relationship. The perturbation and adversarial losses should decrease over time if the model is learning.

A. *Color Vs. Greyscale Images – Perturbation Threshold*

Visual inspection of the color and greyscale images before perturbation show a lot of noise already in place, likely due to resizing and image resolution issues. Further perturbations on these images look, to the human eye, similar to the original or much worse based on how noisy the original image seemed as well as the perturbation threshold – the amount of perturbation that was allowed to be added to the original image (Figure 5).

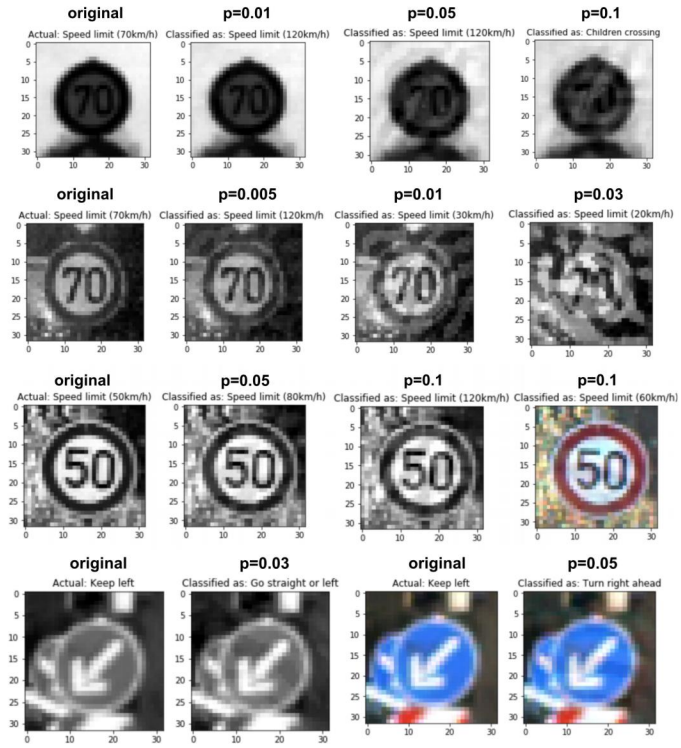

Fig 5. GAN output for both greyscale and color inputs at various perturbation thresholds.

We noticed that, visually, the perturbed greyscale images looked more noisy than the color images. We ran an experiment to see if the AdvGAN model also saw this difference between greyscale and color images. As expected, Figure 6 shows that increasing the perturbation threshold of color and greyscale images increased the perturbation loss and decreased the adversarial loss. This decrease in adversarial loss is expected because images with more perturbations should be harder to classify, resulting in more misclassifications. This should also increase the misclassification percentage and decrease the test accuracy, both seen in the figures above. The discriminator loss in both figures also shows a large drop when perturbation

thresholds are increased as the discriminator is better able to classify the noisy perturbed images as fake.

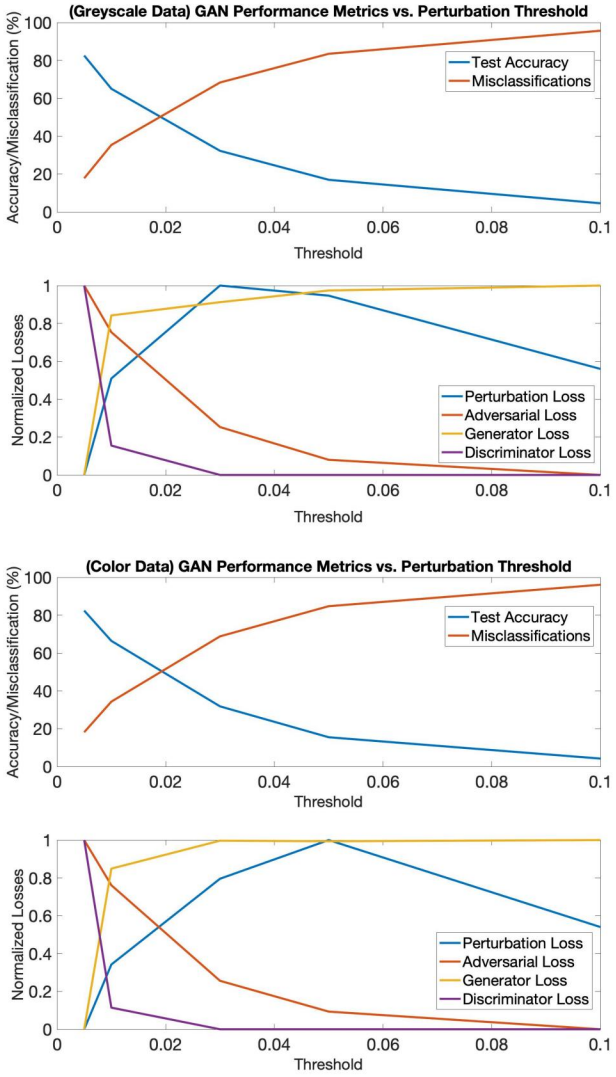

Fig 6. GAN performance metrics vs. perturbation threshold for greyscale and color input images

Contrary to our original hypothesis, there were no significant differences in model performance for color versus greyscale images, as seen in the figures above. Moving forward in our analysis, we chose to study the effects of different parameters on color images because these experiments resulted in better human-evaluated image quality. In selecting the perturbation threshold, we wanted to make sure the model was producing images that would be distinguishable to the human eye – i.e., a human could still correctly classify the image. Keeping this in mind, we chose a threshold of p=0.05 which also produced a relatively high misclassification in our testing runs, after which the misclassification starts to saturate.

B. *Weighted Loss Mechanisms*

In order to get a low test accuracy after perturbation (and similarly a high misclassification percentage), the adversarial loss should be minimized as much as possible. One way to emphasize the importance of this loss is to apply a weight to it in the overall cost function. However, perturbation loss is also important because we would want the images to be indentified as the correct sign if a human were looking at it. To determine the optimal balance, we tested different adversarial:perturbation weighted loss ratios.

As Figure 7 shows, there was a slight decrease in test accuracy and likewise a slight increase in misclassification percentage as we increased the weight on the adversarial loss variable. However, this saturates very quickly around 5:1. As expected, the discriminator loss decreased at higher adversarial:perturbation loss ratios because the perturbation loss was increasing, suggesting that the images were more noisy in higher ratio runs. The lowest test accuracy and highest misclassification occurred at a ratio of 10:1, so this is the ratio we chose for our optimal model.

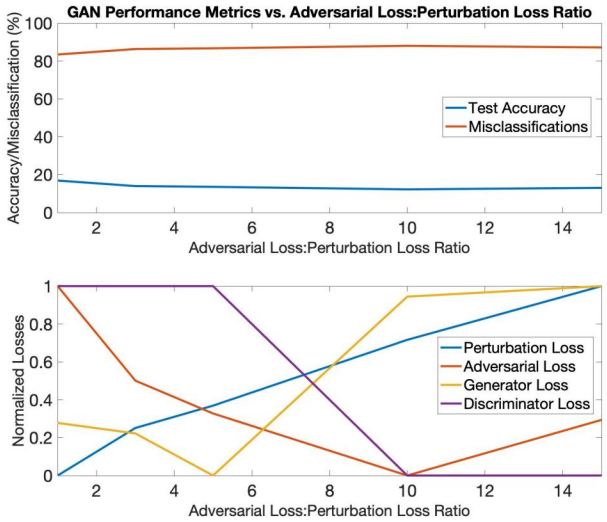

Fig 7. GAN performance metrics vs. adversarial loss:perturbation loss ration for color input images

C. *Generator:Discriminator Learning Ratio*

Generator:discriminator learning ratio is the number of iterations the generator is trained for every one iteration the discriminator trains. Increasing this ratio is supposed to make generator more competitive with the discriminator, resulting in reduced generator loss and increased discriminator loss. Although this was the effect observed at a ratio of 10:1 (Figure 8), an opposite trend is seen for ratios of 3:1 and 5:1. Overall, increasing the generator:discriminator learning ratio deteriotates the GAN performance, resulting in a large decrease in misclassification.

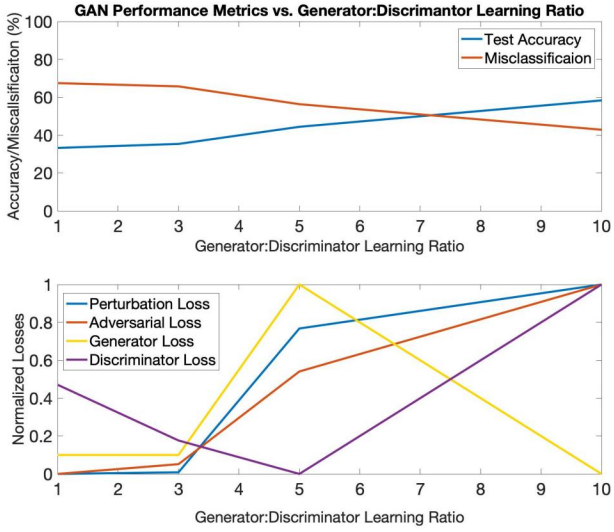

Fig 8. GAN performance metrics vs. generator:discriminator learning rati

D. Size of Generator Filters

As can be seen in figure 9, increasing the number of filters in the three convulotional layers of generator had a moderate effect on the generator's ability to learn, with optimum performance – highest misclassification – at 16/32/64 filters among the three encoder and decoder layers with a resnet block of 64 filters in between. Increasing the number of filters from 8/16/32 to 16/32/64 increased the number of epochs before the discriminator completely won the generator-discriminator battle.

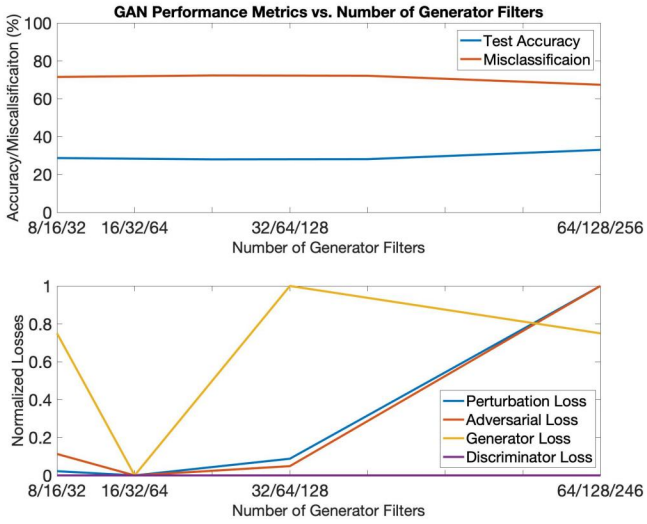

Fig 9. GAN performance metrics vs. number of generator filters

E. Optimum Results

Our analysis revealed that the optimal parameters would be using a perturbation threshold of 0.05, a 10:1 adversarial:perturbation weighted-loss ratio, 16/32/64 filters in the generator, and a 1:1 generator:discriminator learning ratio. A 400-epoch run with these parameters is shown in Figure 10 revealing decreasing losses that seem to flatten out around 100 epochs. The test accuracy and misclassification percentage also reveal expected results with the test accuracy decreasing as misclassifications increases over those same 100 epochs. These values also saturate around 100 epochs. The lowest test accuracy was 17.4% while the highest misclassification was 82.9%.

Figure 11 shows some of the generated images using the optimized GAN along with the original images. Actual and predicted labels are shown above each image.

V. Conclusions & Future Work

In this work, by developing an optimized GAN, we generated inconspicuous adversarial perturbations to traffic sign images and fed the perturbed images to a TSR classifier, reducing the classifier test accuracy from 92.1% to 17.4% and resulting in a significantly high misclassification percentage of 82.9%.

As our adversarial attack model shows, there are vulnerabilities in traffic sign classifiers that may be exploited by malevolent parties. If a similar attack were conducted on the classifier used in an autonomous vehicle, the vehicle may, for instance, classify a stop sign as a speed limit sign. Such occurrences are big safety risks. To mitigate the effect of these attacks, autonomous vehicle software should include perturbed images in the training set for their traffic sign recognition classifier. Another solution is *defensive distillation* [6] which smooths the model's decision surface in adversarial directions that could be exploited by the adversary.

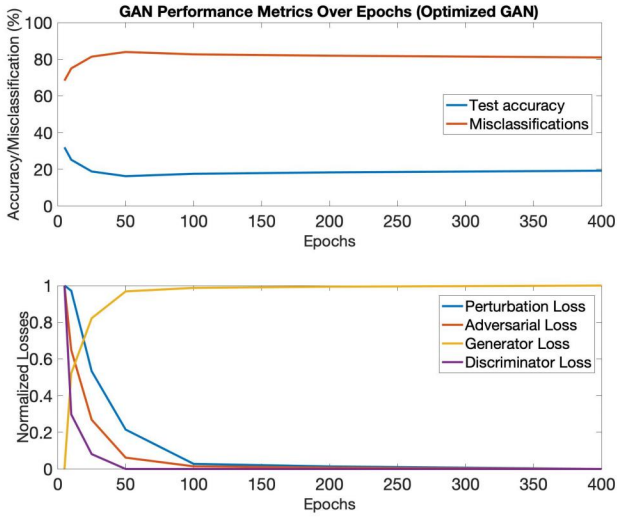

Fig 10. GAN performance metrics over epochs for the oprimized model

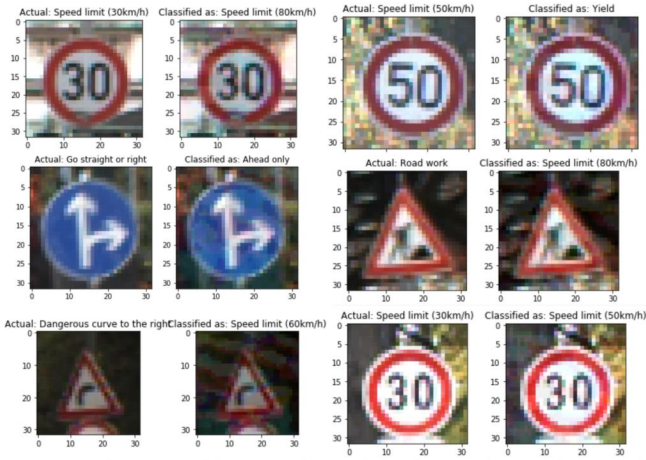

Fig 11. Example generated images using optimized GAN along with the original images to their left. Actual and predicted labels are shown above each image.

VI. Contribution

Stephanie Tietz: Dataset preparation, sign classifier using LeNet5, AdvGAN coding and testing, writing report

Koosha Nassiri Nazif: Sign classifier using FC NN, AdvGAN coding and testing, AWS setup, writing report

VII. Reference

[1] https://github.com/mohamedameen93/German-Traffic-Sign-Classification-Using-TensorFlow

[2] Xiao, C. *et al.* "Generating Adversarial Examples with Adversarial Networks" *IJCAI International Joint Conference on Artificial Intelligence*, 2018–July, 3905–3911.

[3] http://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset

[4] CIFAR10 Dataset: Learning Multiple Layers of Features from Tiny Images, Alex Krizhevsky, 2009.

[5] Sermanet, P. and LeCun, Y. "Traffic Sign Recognition with Multi-Scale Convolutional Networks." *2011 International Joint Conference on Neural Networks, IEEE,* 2011.

[6] Papernot, N. *et al.* "Distillation as a defense to adversarial perturbations against deep neural networks" *IEEE Symposium on Security and Privacy*, 2016