Adversarial Attack and Defense on Traffic Sign Classifier

Tongli Zhu, Qingyi Xu, Tinsae A. Yehuala, Shixun Liu

***Abstract*—Deep neural networks are used in autonomous cars, particularly for traffic sign classification. Any error in this classification process could result in serious accidents, what makes it worse is the model's weakness against attacks through subtle adversarial perturbations. This paper discusses two methods of creating such perturbation: the Fast Gradient Sign Method (FGSM) and Projected Gradient Descent (PGD). Additionally, to increase the model's robustness against such attacks, adversarial training was implemented as a defensive strategy. The result indicates an improved performance of the model against attacks after adversarial training.**

***Keywords*—Adversarial attack, Adversarial training, FGSM, PGD**

I. Introduction

DEEP learning is a state-of-the-art technology used in various real-life applications, including medical software, self-driving cars, natural language processing, and security systems[1]. However, the vulnerability of these model through perturbation added to the input data has become a challenge.

In self-driving cars systems, where deep neural networks are used in the detection and classification of traffic signs, the vulnerability of models to attacks is a big concern. Making sure of the robustness of models against such adversarial threats is important for the safety of these systems.

This project discusses the implementation of an attack on a classifier model using a couple of techniques and it uses a defense to increase the model's robustness. While real-world attacks could involve physically altering signs, this project focuses on simulating these using digitally available images. Despite being categorized as a physical attack[2], the project's approach provides insight into models' robustness against adversarial.

II. Background

In this section, we provide the background of adversarial attacks and defense in general.

A. Attack

An adversarial attack involves the creation of subtle perturbations in data that are not noticeable to humans but can cause machine learning models to misclassified input data. Various techniques have been proposed for such attacks. The classification of these attacks depends on the level of knowledge the attacker has about the model's architecture, based on this there are two categories: white box attacks and black box attacks. White box attacks require complete knowledge of the model's structure, whereas black box attacks do not need an understanding of the inner workings of the model[2].

Furthermore, attacks can be categorized as targeted or untargeted. Untargeted attacks cause the model to predict anything other than the true label, while targeted attacks are designed to make the model predict a predefined output[2].

The Fast Gradient Sign Attack **(FGSM)** attack computes the gradient of the loss with respect to the input data and then uses the gradient to modify the input data to maximize the loss. It shows how we can use adversarial examples to improve the robustness of machine learning models through adversarial training.[3] An iteration of this approach is the **Iterative FGSM**[4], which repeatedly applies FGSM with a small step size, taking multiple smaller steps, and clips the outcome. Iterative FGSM[4] has similar idea with **PGD** attack with little difference. PGD is stand for Projected Gradient Descent which is a method for generating adversarial examples during adversarial training.It shows that adversarial training with PGD-generated examples significantly enhance the robustness of deep neural networks against various attacks and are more robust to adversarial examples compared to other defense methods.[5] Another variant, **Momentum Iterative FGSM**[4], speeds up FGSM by accumulating a velocity vector in the gradient direction of the loss over iterations.

The **Carlini and Wagner attack** is a method that uses a specialized loss function, characterized by lower values for adversarial examples and higher values for clean examples. Adversarial examples are then generated through the minimization of this loss function.[2].

B. Defense

In response to attacks, several defense strategies have been suggested, such as adversarial learning, the destruction of adversarial perturbation structures, and the implementation of gradient masking.

Adversarial training involves mixing adversarial samples into the training process to increase the robustness of a model. This strategy changes the decision boundary of the model to increase the accuracy.[2].

Structure Destruction of Adversarial Perturbations refers to using strategies to remove adversarial noise from input data. These strategies include using image preprocessing such as filtering and denoising as a defense against adversarial examples[6].

Gradient masking in adversarial training is a defense strategy that involves intentionally making sure the gradient can not be computed with respect to the input. This makes it challenging for attackers to use the model's gradient to generate adversarial examples.[2].

C. Regularization and Dropout

In supervised learning, overfitting is somethings we definitely need to avoid, it can cause the model perform badly to real unseen data and reduce the robustness of the model. Overfitting typically arises when a learning model becomes overly too complex, particularly when dealing with an excessive number of parameters. A typical approach to solve this problem is to introduce a form of the penalty term in the objective function, which is called regularization, can help to prevent the network parameters from becoming excessively large. Another suggested technique, named dropout, is implemented to stop co-adaptation on each training data point by randomly excluding some units during the training procedure.From the paper[7] we know that sometimes dropout can be more effective than L2-regularization but just to be safe we used both method.

III. Methodology

In this section, we discuss the specific attack and defense strategies implemented within the project.

A. Attack

The objective of the attack is to introduce a subtle perturbation that is not noticeable to humans but has the potential to cause misclassification in our traffic sign classifier model. Mathematically, this can be formulated as follows[4]:

Minimize:

$$\|x_{\text{adv}} - x\|_2^2$$

Subject to:

$$f(x_{\text{adv}}) \neq y_{\text{true}}, \quad x_{\text{adv}} \in [0,1]^m$$

The generation of these noise perturbations is achieved using two specific white-box attack techniques: the Fast Gradient Sign Method (FGSM) and the Projected Gradient Descent (PGD) method.

The Fast Gradient Sign Method (FGSM)[4] uses the gradient of the loss objective function with respect to the input image to generate a perturbation. By utilizing the derivative, the method identifies the direction that increases the loss. Then, a small perturbation is introduced by using the sign of the gradient[4].

$$x_{\text{adv}} = x + \epsilon \cdot \text{sign}(\nabla_x J(\theta, x, y))$$

where:

- $J(\theta, x, y)$ is the loss function of the model with parameters θ, given input x and true label y.
- ϵ is a small perturbation magnitude.
- $\nabla_x J(\theta, x, y)$ is the gradient of the loss with respect to the input.

In **Projected Gradient Descent (PGD)**[4], the perturbation is updated iteratively based on the gradient of the loss with respect to the input. This is an attempt to find the perturbation that maximizes the loss while staying within a specified ϵ ball around the input data. The iterative updates are clipped to make sure they are within valid data limits.

$$x_{\text{adv}}^{(k+1)} = \text{clip}_x \left(x_{\text{adv}}^{(k)} + \alpha \cdot \text{sign}(\nabla_x J(\theta, x_{\text{adv}}^{(k)}, y)) \right)$$

where:

- $J(\theta, x, y)$ is the loss function of the model with parameters θ, given input x and true label y.
- α is the step size for each iteration.
- $\nabla_x J(\theta, x_{\text{adv}}^{(k)}, y)$ is the gradient of the loss with respect to the input at the k-th iteration.
- clip_x is a function that clips the perturbation to ensure it stays within a specified range.

B. Defense

Our defense strategy against applied attacks involves mixing the training data with adversarial examples[8][2]. These adversarial examples are generated using the FGSM and PGD attacks. The intention behind injecting adversarial examples is to push away the decision boundary of the classifier, making it more resistant to adversarial attacks. The adversarial training loss can be formally expressed:

$$\min_{\theta} \sum_{(x,y) \in D_{\text{train}}} \left[\text{L}(f_\theta(x), y) + \text{L}\left(f_\theta\left(x_{\text{adv}}\right), y\right) \right]$$

where:

- D_{train} is the training dataset.
- f_θ is the neural network model with parameters θ.
- $\text{L}(f_\theta(x), y)$ is the standard loss function for the clean example.
- x_{adv} is the adversarial example generated using x input from training data.
- $\text{L}\left(f_\theta\left(x_{\text{adv}}\right), y\right)$ is the loss function for the adversarial example.

C. Regularization

Besides the attack and defense strategies, regularization techniques have also been used to avoid overfitting. These are L2 regularization and dropout. A dropout is a heuristic approach[9] of regularization and L2 regularization is given by:

$$\text{Loss}_{\text{L2}} = \text{Loss}_{\text{original}} + \lambda \sum_i \|w_i\|^2$$

where:

- Loss_{L2} is the regularized loss function.
- $\text{Loss}_{\text{original}}$ is the original loss function without regularization.
- w_i are the weights of the model.
- λ is the regularization strength, a non-negative constant.

IV. Implementation

A. Materials

We used the German traffic sign recognition benchmark (GTSRB) dataset. The GTSRB dataset consists of 39209 training images corresponding to 43 classes. An example image corresponding to each class of the dataset is shown as figure1. To speed up the process of training models, we use graphics processing units (GPUs). In addition, we also use software tools such as PyCharm, Anaconda, and Jupyter Notebook to support code development, environment management, and demonstration of experimental results.

Fig. 1: (GTSRB) dataset example

B. Pre-trained model

The pre-trained model we chose is based on a variant of AlexNet. It retains the architectural features of traditional AlexNet, including multi-layer convolution layers, and maximum pooling layers, using ReLU functions and Dropout techniques. However, the variant uses a smaller 3x3 convolution kernel in the convolutional layer, which helps reduce the number of model parameters and improves the capture of details. At the same time, the model has also been adjusted in the structure of the fully connected layer, especially the output of the last fully connected layer is 43 categories, making the model more suitable for the data set we selected (GTSRB). The architecture of the model is shown as tableI.

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 64, 56, 56]	1,792
MaxPool2d-2	[-1, 64, 28, 28]	0
ReLU-3	[-1, 64, 28, 28]	0
Conv2d-4	[-1, 192, 28, 28]	110,784
MaxPool2d-5	[-1, 192, 14, 14]	0
ReLU-6	[-1, 192, 14, 14]	0
Conv2d-7	[-1, 384, 14, 14]	663,936
ReLU-8	[-1, 384, 14, 14]	0
Conv2d-9	[-1, 256, 14, 14]	884,992
ReLU-10	[-1, 256, 14, 14]	0
Conv2d-11	[-1, 256, 14, 14]	590,080
MaxPool2d-12	[-1, 256, 7, 7]	0
ReLU-13	[-1, 256, 7, 7]	0
Dropout-14	[-1, 12544]	0
Linear-15	[-1, 1000]	12,545,000
ReLU-16	[-1, 1000]	0
Dropout-17	[-1, 1000]	0
Linear-18	[-1, 256]	256,256
ReLU-19	[-1, 256]	0
Linear-20	[-1, 43]	11,051

TABLE I: The architecture of the model

C. Pre-model testing

We use the same data allocation as in the pre-trained model, that is, the same training set and test set, to ensure that the data in the test set has not been seen by the pre-trained model. The test results can be seen in the next section. it is stable at around 95%.

D. Attack and defense experiment design and implementation

1) General flow: The overall flow of the experiment is shown in Figure 3, which is divided into three parts: data set pre-allocation, attack experiment implementation, and adversarial training and effectiveness testing.

a) Data Set Allocation: Divide the test data set and training data set into two subsets, namely test data set 1, test data set 2, training data set 1 and training data set 2.

b) Attack Experiment Implementation: The two main attack methods mentioned in the previous section are adopted: Fast Gradient Sign Method (FGSM) and Projected Gradient Descent (PGD). We use test data set 1 and test data set 2 respectively to evaluate the effects of the two attacks. The purpose of separating the two data sets is to shorten the time of the attack experiment, and the two test sets can be merged later for the overall test set of the adversarial training model, which is test data set 3. Specifically, we perform FGSM and PGD attacks on test datasets 1 and 2 respectively to generate new test datasets, and then input the old model without adversarial training to output accuracy and loss.

c) Adversarial Training and Effectiveness Testing: After the attack experiments, we apply FGSM and PGD against the training dataset 1 to generate the adversarial training dataset. Then, these adversarial training datasets are merged with training dataset 2 to form a new training dataset 3. Compared with the original data set, the number of the data set 3 has increased by 1/3, and 2/3 of the data are applied to FGSM and PGD. That is, although the human eye cannot distinguish the difference, the old model will not classify the data correctly.

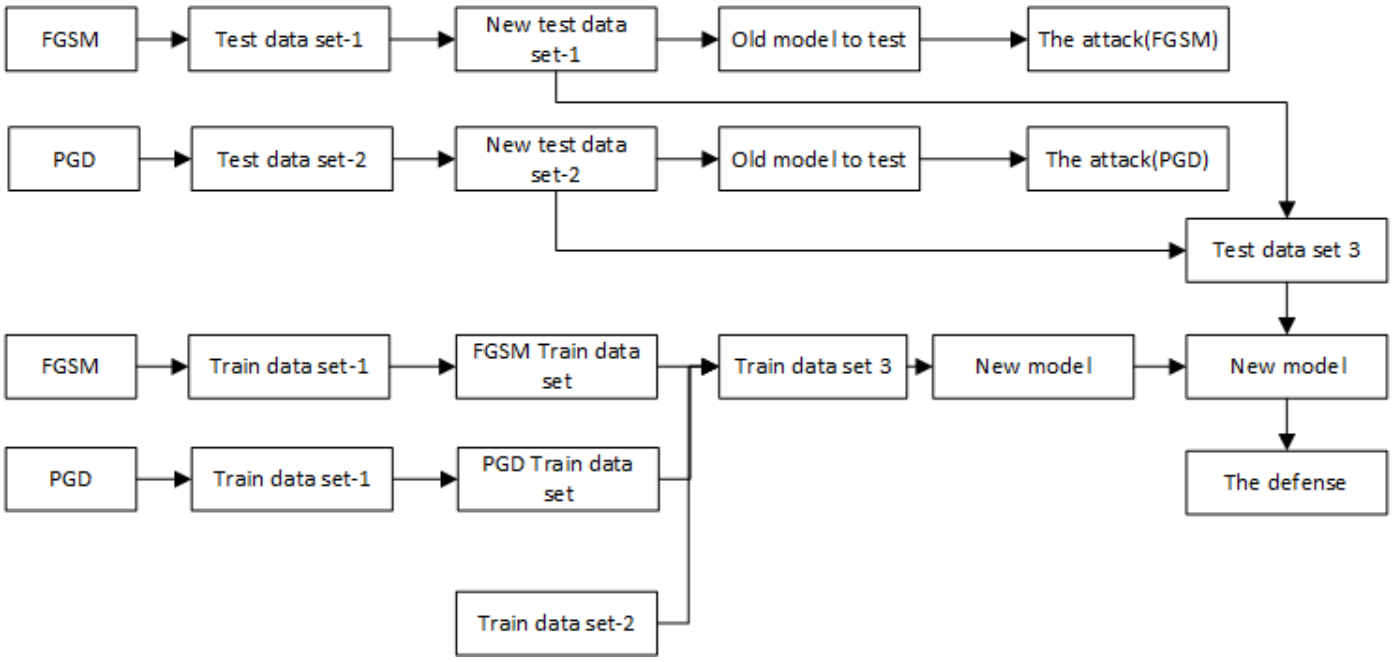

Fig. 2: Model Training and Testing Workflow

After training is completed, we input the above merged test data set 3 into the new model and output the accuracy and loss for evaluation.

2) Specific approach: Based on the comprehensive consideration of methodological research and experimental time, we used 7 different training parameters, focusing on training epsilons, epoch, and whether to use Regularization-L2. The specific parameter configuration is shown in Table 3. For each model before and after training, samples with Epsilons (vector form) of 0, 0.004, 0.008, 0.012, 0.016, 0.02 are used for comprehensive testing. We used a constant 0.001 as the learning rate of PGD throughout the experiment.

TABLE II: Adversarial Training Parameters Configuration

Model	Training Epsilon	Epoch	Regularization-L2
0	0.15	10	no
1	0.015	10	no
2	0.004	10	no
3	0.004	15	no
4	0.004	10	yes ($\lambda = 1e-4$)
5	0.004	10	yes ($\lambda = 1e-2$)
6	0.004	10	yes ($\lambda = 1e-3$)

Note: Test epsilons(1-6) are [0, 0.004, 0.008, 0.012, 0.016, 0.02]. PGD learning rate(1-6) is 0.001. Test epsilons(0) are [0, 0.04, 0.08, 0.12, 0.16, 0.2]. PGD(0) learning rate is 0.01.

Our experimental parameter configuration is not set arbitrarily, but is carefully adjusted and optimized based on the training results. The specific idea is shown in Figure 3. Initially, we set a seemingly reasonable Training Epsilons of 0.15 and generated test samples with Epsilons of the same order of magnitude. However, the performance of the model was extremely poor, and the accuracy was even less than 10%. Through visual analysis of the generated samples, we found that there are obvious differences between the two road signs in the samples. The degree of these disturbance is so large that they can be easily identified by the naked eye, as shown in Figure 4. Therefore, these samples cause the model to learn a lot of noise, and the results are naturally very biased.

In view of this, we decided to reduce the Epsilons of the generated samples by ten times in order to reduce visual interference, as shown in Figure 4.

To this end, we decided to go back to the model setting of epoch 10(Model2) and began to introduce Regularization-L2 to prevent overfitting problems. We conducted experiments on three different intensity parameters(Model4,5,6) in L2 and conducted a comprehensive comparative analysis.

V. Result and Discussion

A. Attack

The effects of FGSM and PGD can be visualized by comparing the original samples with the samples under attack. The effect of the attack on the samples is clearly visible at higher attack intensities (ϵ), as shown in Figure 5, and the perturbation is barely visible to the human eye at lower attack intensities, as shown in Figure 6.

In our experiments, we set the α of the PGD attack to 0.001 and epsilon varied from 0 to 0.02. As shown in Figure 7, as the attack intensity increases, the loss of the model on the test dataset increases significantly and the accuracy decreases accordingly. Moreover, it can also be seen that the PGD attack is more effective than the FGSM attack. In addition, it can be seen from Table III that even using relatively small

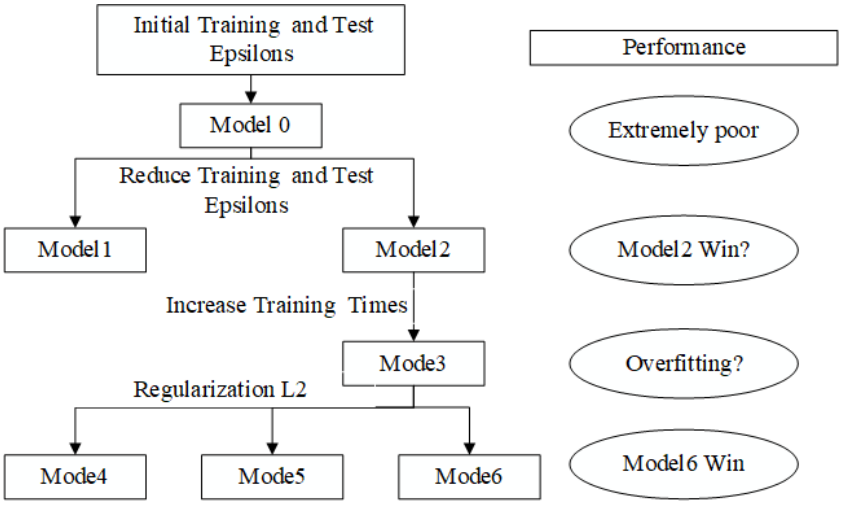

Fig. 3: Training track

Fig. 4: Comparison of adversarial samples generated by different epsilon values

Fig. 5: Comparison of original samples (left) with samples attacked (left) by FGSM (up) and PGD (down) with $\epsilon = 0.12$

perturbations, which are almost invisible to the human eye, can still make the accuracy of the model significantly decrease.

Fig. 6: Comparison of original samples (left) with samples attacked (left) by FGSM (up) and PGD (down) with $\epsilon = 0.015$

TABLE III: The accuracy of the model attacked by FGSM and PGD ($\alpha = 0.001$)

ϵ	FGSM	PGD ($\alpha = 0.001$)
0.000	0.954	0.954
0.004	0.705	0.613
0.008	0.567	0.441
0.012	0.499	0.409
0.016	0.460	0.409
0.020	0.430	0.409

B. Defense

From Table IV, it can be seen that after adversarial training using $\epsilon = 0.15$ and $\alpha = 0.1$ as parameters for generating adversarial samples, the accuracy of model 0 is lower instead of higher. A possible reason for this situation is that a higher attack intensity means that a larger perturbation is applied to the input data, which can result in the adversarial samples being too different from the original data. During training, the model may focus too much on specific perturbation patterns or noise in the adversarial samples and ignore broader data features.

The test accuracy results for each model mentioned in Table

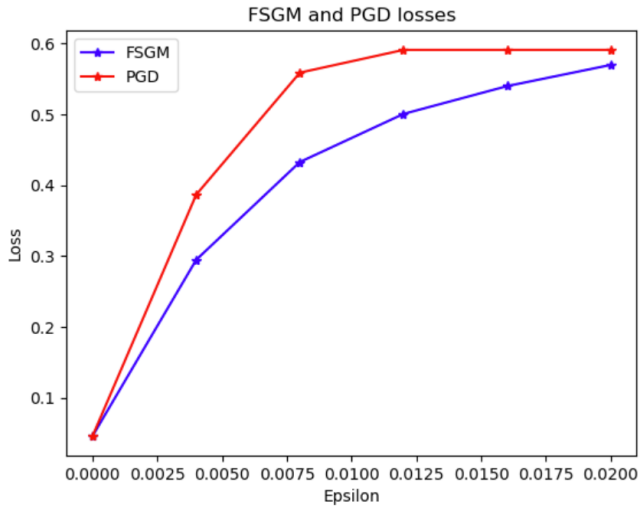

Fig. 7: The losses of the model attacked by FGSM and PGD with $\alpha = 0.001$

TABLE IV: Accuracy of original model and Model0

ϵ	Attack (FGSM)	Defense (FGSM)	Attack (PGD)	Defense (PGD)
0.00	0.954	0.955	0.954	0.955
0.04	0.343	0.231	0.110	0.055
0.08	0.265	0.177	0.081	0.034
0.12	0.224	0.151	0.078	0.033
0.16	0.197	0.131	0.078	0.033
0.20	0.182	0.120	0.078	0.033

II are in Appendix A. In order to make the comparison easier, Figure 8 presents the difference in test accuracy between Models 1-6 and the original model under attack, that is, the accuracy of Models 1-6 when attacked minus the accuracy of the original model when attacked. Based on this result, we can directly compare the effectiveness of defense between models and see which changes are positive. The three factors that we observe to influence the effectiveness of the defense are as follows:

1) Training ϵ: Comparison of Model1 and Model2 shows that the value of ϵ used to generate the adversarial samples used for training has an impact on the results of the defense. When we make an initial judgment based on accuracy, we believe that Model2 is superior to Model1, because Model2's test accuracy is higher than Model1's at smaller values of ϵ. However, it can be seen from Figure 8 that it is not true. Model1 performs better around the test $\epsilon = 0.015$, while Model2 performs better around $\epsilon = 0.004$. Therefore we can conclude that the models will perform better around the value of ϵ used to generate the adversarial samples used for training during the test.

2) Training Epoch: Comparison of Model2 and Model3 shows that the number of epochs during training affects the results of the defense. Model2 performs better than Model3 for any ϵ. Therefore, for our model, the number of epochs at training time is more appropriate at 10, and the number of epochs at 15 may cause overfitting, leading to a decrease in the accuracy of the model during the test.

3) Regularization-L2: In the attempt to solve the overfitting problem of the model not performing as well on the test dataset as it did on the training dataset, we added L2 regularization. By comparing Model4, Model5, and Model6, we can observe the effect of the regularization strength λ on the model. When using $\lambda = 1e-4$, the accuracy of the model is not significantly improved compared to that without L2 regularization; when using $\lambda = 1e-2$, the model fails to achieve normal accuracy during training, which means that underfitting occurs; and when using $\lambda = 1e-3$, the model's performance becomes significantly better. Although Model6 does not completely solve the overfitting problem, its model is improved by adding L2 regularization compared to the other models.

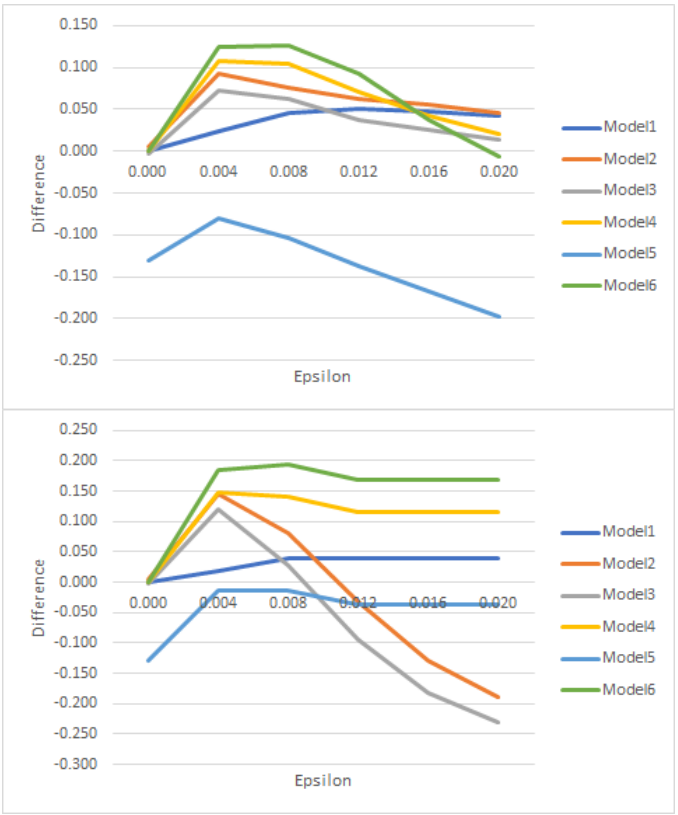

Fig. 8: Difference between the test accuracy of models 1-6 and original model under FGSM attack (up) and PGD attack (down)

Based on the above analysis, the best performing model in this project is Model 6, and its loss and accuracy are shown in Figure 9 and Table V.

VI. Improvement

Even with GPUs, model training is still very time-consuming. We made some trade-offs, but there are aspects where the model could be improved:

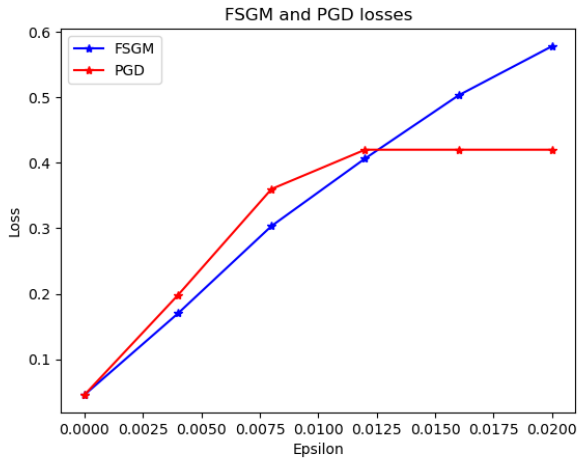

Fig. 9: The losses of the Model6 attacked by FGSM and PGD

epsilon	Attack (FGSM)	Defense (FGSM)	Attack (PGD)	Defense (PGD)
0.000	0.953	0.954	0.953	0.954
0.004	0.705	0.830	0.616	0.802
0.008	0.570	0.696	0.445	0.640
0.012	0.501	0.593	0.412	0.580
0.016	0.459	0.497	0.412	0.580
0.020	0.429	0.422	0.412	0.580

TABLE V: The accuracy of original model and Model6 attacked by FGSM and PGD

1) Adversarial sample diversification: We can generate adversarial samples using different epsilons to improve the robustness of the model to different perturbations.

2) Adding additional regularizations: We can add other regularizations to the model, such as early stopping, and continuously optimize the parameters of the L2 regularization to improve the model's overfitting problem.

VII. Conclusion

In this project, we successfully attacked a traffic sign classifier model using FSGM and PGD. PGD is shown to be a more effective attack tragedy than FSGM due to its iterative nature. Adversarial training has been shown to improve the model's resistance against adversaries. However, adversarial training needs foreknowledge of attackers' strategies to generate adversarial examples. In addition, augmenting the training set with adversarial examples increases the size of the dataset which makes training take considerable time.

VIII. AI Tool Usage Statement

The formula in this article uses the image processing function of chatgpt to generate language in latex form. In the material section, we use chatgpt for text polishing. In the pretrained model section, in order to save time, we use chatgpt to learn and understand the differences between the variant and the traditional AlexNet structure.

In the methodology section, All formulas with their description were generated by chatgpt. And Latex code is generated for the description and the formula. In addition, we used chatgpt to understand the difference between Iterative FSGM and PGD. For grammar check and word suggestion, we used Grammely.

References

[1] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards Deep Learning Models Resistant to Adversarial Attacks," Sept. 2019. arXiv:1706.06083 [cs, stat].

[2] A. Kurakin, I. Goodfellow, S. Bengio, Y. Dong, F. Liao, M. Liang, T. Pang, J. Zhu, X. Hu, C. Xie, J. Wang, Z. Zhang, Z. Ren, A. Yuille, S. Huang, Y. Zhao, Y. Zhao, Z. Han, J. Long, Y. Berdibekov, T. Akiba, S. Tokui, and M. Abe, "Adversarial Attacks and Defences Competition," Mar. 2018. arXiv:1804.00097 [cs, stat].

[3] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2015.

[4] A. S. Hashemi, S. Mozaffari, and S. Alirezaee, "Improving adversarial robustness of traffic sign image recognition networks," *Displays*, vol. 74, p. 102277, Sept. 2022.

[5] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," 2019.

[6] S. Pavlitska, N. Lambing, and J. M. Zöllner, "Adversarial Attacks on Traffic Sign Recognition: A Survey," July 2023. arXiv:2307.08278 [cs].

[7] A. Shafahi, M. Najibi, M. A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein, "Adversarial training for free!," in *Advances in Neural Information Processing Systems* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), vol. 32, Curran Associates, Inc., 2019.

[8] Y. Wang, T. Sun, S. Li, X. Yuan, W. Ni, E. Hossain, and H. V. Poor, "Adversarial Attacks and Defenses in Machine Learning-Powered Networks: A Contemporary Survey," Mar. 2023. arXiv:2303.06302 [cs].

[9] S. J. Prince, *Understanding Deep Learning*. MIT Press, 2023.

APPENDIX A
TEST ACCURACY RESULTS FOR 6 MODELS

Method	Epsilon	Attack	Model1	Model2	Model3	Model4	Model5	Model6
FGSM	0.000	0.953	0.953	0.958	0.950	0.953	0.823	0.954
	0.004	0.705	0.728	0.797	0.777	0.813	0.624	0.830
	0.008	0.570	0.615	0.646	0.633	0.696	0.467	0.696
	0.012	0.501	0.551	0.564	0.539	0.572	0.364	0.593
	0.016	0.459	0.507	0.514	0.484	0.501	0.292	0.497
	0.020	0.429	0.471	0.475	0.442	0.449	0.232	0.422
PGD	0.000	0.953	0.953	0.958	0.950	0.953	0.823	0.954
	0.004	0.616	0.634	0.761	0.736	0.763	0.603	0.802
	0.008	0.445	0.484	0.525	0.472	0.587	0.432	0.640
	0.012	0.412	0.452	0.380	0.318	0.528	0.376	0.580
	0.016	0.412	0.452	0.282	0.230	0.528	0.376	0.580
	0.020	0.412	0.452	0.222	0.181	0.528	0.376	0.580

TABLE VI: Test accuracy results for 6 models