GMR Institute of Technology
An Autonomous Institute Affiliated to JNTUK, Kakinada

GMRIT
Training Tomorrow's Engineers Today

ADVERSARIAL ATTACKS ON DEEP NEURAL NETWORKS AND DEFENDING TECHNIQUES

A project report submitted in partial fulfillment of the requirementfor the award of degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

Submitted by

L.KOMALI	**(20341A05A9)**
J.SIVA TANAY AKASH	**(20341A0573)**
M.MANICHANDAN	**(20341A05B8)**
G.MOHANLAL	**(20341A0567)**
N.CHANDU	**(20341A05C7)**

Under the esteemed guidance of

Ms. Santhoshini Sahu

Assistant Professor, Dept. of CSE

GMR Institute of Technology

An Autonomous Institute Affiliated to JNTUK, Kakinada

(Accredited by NBA, NAAC with 'A' Grade & ISO 9001:2008 Certified Institution)

GMR Nagar, Rajam – 532127,
Andhra Pradesh, India

April 2020

`

GMR Institute of Technology
An Autonomous Institute Affiliated to JNTUK, Kakinada

GMRIT
Training Tomorrow's Engineers Today

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the thesis entitled **ADVERSARIAL ATTACKS ON DEEP NEURAL NETWORKS AND DEFENDING TECHNIQUES** submitted by **L.Komali (20341A05A9), J.Siva Tanay Akash (20341A0573), M.Manichandan (20341A05B8), G.Mohanlal (20341A0567) and N.Chandu (20341A05C7)** has been carried out in partial fulfillment of the requirement for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering** of **GMRIT, Rajam** affiliated to **JNTUK, KAKINADA** is a record of bonafide work carried out by them under my guidance & supervision.The results embodied in this report have not been submitted to any other University or Institute for the award of any degree.

Signature of Supervisor
Ms. Santhosini Sahu
Assistant Professor
Department of CSE
GMRIT, Rajam.

Signature of HOD
Dr. A. V. Ramana
Professor & Head
Department of CSE
GMRIT, Rajam.

The report is submitted for the viva-voice examination held on ……..

Signature of Internal Examiner

Signature of External Examiner

ACKNOWLEDGEMENT

It gives us an immense pleasure to express deep sense of gratitude to my guide **Ms. Santhoshini Sahu,** Assistant Professor, Department of Computer Science and Engineering for her whole hearted and invaluable guidance throughout the project work. Without her sustained and sincere effort, this project work would not have taken this shape. She encouraged and helped us to overcome various difficulties that we have faced at various stages of our project work.

We would like to sincerely thank our Head of the department **Dr. A. V. Ramana**, for providing all the necessary facilities that led to the successful completion of our project work.

We would like to take this opportunity to thank our beloved Principal **Dr.C.L.V.R.S.V.Prasad**, for providing all the necessary facilities and a great support to us in completing the project work.

We would like to thank all the faculty members and the non-teaching staff of the Department of Electronics and Communication Engineering for their direct or indirect support for helping us in completion of this project work.

Finally, we would like to thank all of our friends and family members for their continuous help and encouragement.

L.Komali 20341A05A9
J.Siva Tanay Akash 20341A0573
M.Manichandan 20341A05B8
G.Mohanlal 20341A0567
N.Chandu 20341A05C7

`

ABSTRACT

Artificial intelligence (AI) and deep learning (DL) techniques are widely used in various fields such as image classification, object detection, speech recognition, NLP etc. On the other hand, these models especially deep neural networks can easily be fooled by different adversarial attacks. Adversarial attacks involve adding small perturbations to inputs with the goal of getting a machine learning or deep learning model to misidentifying the output. Hence, they bring serious security risks to deep-learning-based systems. So, it is extremely important to provide robustness to deep learning algorithms against these adversaries. In general, adversarial attacks are happened by generating adversarial examples. Adversarial examples are that which may be imperceptible to the human eye, but can lead the model to misclassify the output. In this paper, some of the methods for generating adversarial examples like Fast Gradient Sign Method (FGSM) and Carlin and Wagner (C&W) attacks and few defending methods like adversarial training, auto-encoder and a proposed model to be used.

Keywords: Adversarial examples, Fast Gradient Sign Method (FGSM), Carlini and Wagner (C&W), Auto-encoder, Deep neural networks.

`

TABLE OF CONTENTS

`

LIST OF TABLES

`

LIST OF FIGURES

`

LIST OF SYMBOLS & ABBREVIATIONS

C&W : Carlini and Wagner

FGSM : Fast Gradient Sign Method

GSTRB : German Traffic Sign Recognition Benchmark

SCES : selective cascade ensemble strategy

SPES : stack parallel ensemble strategy

`

INTRODUCTION

The applications of artificial intelligence technologies in various fields have been rapidly developed recently. Due its high performance, high intelligence and high availability of artificial intelligence technologies have been applied in various fields like image classification, object detection, voice control, machine translation. DNN is deep neural network which is multiple layer perceptron model with number of hidden layers, input and output layers which is graph like structure and this has wide range of application in computer vision and NLP etc.. However, it has been discovered that DNNs are susceptible to adversarial instances, meaning that examples introduced with suitable perturbations can easily deceive well-trained DNNs. Real-world applications of DNNs have been prone to adversarial examples. Thus, adversarial instances represent a serious danger to deep learning's commercial uses.

These attacks may be white box attacks or black box attacks. White box attacks are in which an adversary has total knowledge about the model used for classification. The adversary utilizes information to attack the model. Black box models are in which attacks does not have any knowledge about the model the user information about the setting or past input to analyse the vulnerabilities of model. Different black box attack Non-Adaptive Black-Box, Attack Adaptive Black-Box Attack, Strict Black-Box Attack. Black box attacks can be solved by building a local model using training data and then apply the white box attack on it.

In this paper we use FGSM and C&W attacks on the DNN model i.e image classification using GSTRB dataset and also we use defensive distillation and adversarial training to defend the attacked model. FGSM is a gradient based attacking method which makes use of back-propagated gradients sign of loss. In FGSM, the input image is modified to maximize the loss. It is a single step method which increases the loss in the steepest direction by performing the one-step update along the gradient of the adversarial examples. C&W attack is a powerful attack method which can achieve success even on distilled networks.This technique can also have targeted transferable examples to carry

out a black-box assault by altering the confidence.Adversarial training is a defensive method which includes adversarial samples in the training stage and then explicitly trains the model to detect these tricks. It is a brute force approach that increases the robustness of the model. Defensive distillation is one of best defensive method which distillation reduces effectiveness of adversarial sample creation from 95% to 0.5%.

In defensive training one model is trained to predict the output probabilities and these probabilities are use train another for same input which helps DNN model to more generalized and helps in predicting the unseen samples well. Second model helps in finding similarities among data.

Objective

1.Understanding various adversarial attacks on deep learning models.

2.Visualizing the how the model is attacked.

3. Identifying the vulnerabilities of the model and applying defending techniques to counter the attacks.

4.Visualizing how defense try to avoid the attack on the model.

`

RELATED WORK/ THEORETICAL STUDY

[1] Zhang, J., Qian, W., Nie, R., Cao, J., & Xu, D. (2022). Generate adversarial examples by adaptive moment iterative fast gradient sign method. Applied Intelligence, 1-14.

This proposes the adaptive moment iterative fast gradient sign method (Adam-FGSM), a new iterative white-box attack.

This method adjusts the perturbation direction and calculates the perturbation size for each component of the perturbation vector under the guidance of the moment estimations of gradients.

The adversarial examples generated by Adam-FGSM can fool DNNs with high confidence by contrast with the unavailable ones generated by one-step and iterative attack methods.

It can be seen that Adam-FGSM achieves the highest attack performance on target models with the same iterations and perturbation.

[2] Ahmed, U., Lin, J. C. W., & Srivastava, G. (2022). Mitigating adversarial evasion attacks by deep active learning for medical image classification. Multimedia Tools and Applications, 81(29), 41899-41910.

This paper proposes an adversarial evasion method to keep the data secure.

The model uses a federated learning approach to share model weights and gradients. The local model first studies the unlabeled samples classifying them as adversarial or normal.

The proposed dynamic clustering model clusters the low dimensional vectors into meaningful regions.

It improves the performance of the DNN model with deep active learning without transmitting any raw data.

[3] Wu, F., Xiao, L., Yang, W., & Zhu, J. (2020). Defense against adversarial attacks in traffic sign images identification based on 5G. EURASIP Journal on Wireless Communications and Networking, 2020(1), 1-15.

This propose a new defense method and take full advantage of 5G high-speed bandwidth and mobile edge computing (MEC) effectively.

We use singular value decomposition (SVD) which is the optimal approximation of matrix in the sense of square loss to eliminate the perturbation.

The results show that adversarial attacks, such as Carlini Wagner's l2, Deepfool, and I-FSGM, can be better eliminated by the method and provide lower latency.

`

[4] Ren, H., Huang, T., & Yan, H. (2021). Adversarial examples: attacks and defenses in the physical world. International Journal of Machine Learning and Cybernetics, 12(11), 3325-3336.

This paper presents a comprehensive overview of adversarial attacks and defenses in the real physical world.

This paper compares and summarize the work of adversarial examples on image classification tasks, target detection tasks, and speech recognition tasks.

In addition, the adversarial attacks mentioned are provided with relevant feasible

defense strategies.

This also provides some research directions in the physical world against attacks in the future are discussed.

[5] Zhang, Y., Li, H., Zheng, Y., Yao, S., & Jiang, J. (2021). Enhanced DNNs for malware classification with GAN-based adversarial training. Journal of Computer Virology and Hacking Techniques, 17(2), 153-163.

This paper proposes two adversarial attacks targeting our trained DNNs to generate adversarial malware and a defense mechanism.

In the defensive mechanism, a generative adversary network (GAN)-based model is designed to filter out the perturbation noise and those that with the highest probability to fool the target DNNs are chosen for adversarial training.

Incorporating GAN-based adversarial samples into training, the enhanced DNN achieves satisfactory with 90.20% accuracy.

[6] Jia, X., Zhang, Y., Wu, B., Wang, J., & Cao, X. (2022). Boosting fast adversarial training with learnable adversarial initialization. IEEE Transactions on Image Processing, 31, 4417-4430.

This paper proposes a defensive method that boosts the training efficiency of adversarial training.

In this method fast gradient sign method (FGSM) is adopted in fast AT methods by calculating gradient only once.

In this paper, focusing on image classification, we boost fast AT with a sample-dependent adversarial initialization, i.e., an output from a generative network conditioned on a benign image and its gradient information from the target network.

Experiment results demonstrate that proposed method not only shows a satisfactory training efficiency but also greatly boosts the robustness of fast AT methods.

[7] Zhang, H., & Sakurai, K. (2021). Conditional Generative Adversarial Network-Based Image Denoising for Defending Against Adversarial Attack. IEEE Access, 9, 169031-169043.

`

This paper proposes a method of pre-denoising all input images to prevent adversarial attacks by adding a purification layer before the classification model.

This method can recover noise-attacked images to a level close to the actual image to ensure the correctness of the classification results.

This method can quickly pre-denoises a single image in 0.02 seconds and recover more than 50% of the accuracy of the classification model.

Even if an attacker uses high-intensity adversarial perturbation, this model can recover the input images to the maximum extent.

[8] Wang, D., Li, C., Wen, S., Nepal, S., & Xiang, Y. (2020). Defending against adversarial attack towards deep neural networks via collaborative multi-task training. IEEE Transactions on Dependable and Secure Computing.

This paper proposed a novel defensive framework based on collaborative multi-task training, aiming at providing defence for different types of attacks.

The proposed defence first encodes training labels into label pairs and counters black-box attacks leveraging adversarial training supervised by the encoded label pairs.

The defence further constructs a detector to identify and reject high-confidence adversarial examples that bypass the black-box defence.

The results showed that the defending method achieved up to 96.3% classification accuracy on black-box adversarial examples, and detected up to 98.7% of the high confidence adversarial examples.

[9] Khamaiseh, S. Y., Bagagem, D., Al-Alaj, A., Mancino, M., & Alomari, H. W. (2022). Adversarial Deep Learning: A Survey on Adversarial Attacks and Defense Mechanisms on Image Classification. IEEE Access.

This paper provides an extensive study of state-of-the-art algorithms for generating deep learning adversarial attacks.

This paper also provide a systematic and comprehensive review of the adversarial threat model that covers the deep learning system attack surface, adversarial knowledge and capabilities, adversarial goals, and attack scenarios.

This also discuss a number of open issues and possible future research directions for adversarial deep learning.

[10] Li, Z., Feng, C., Zheng, J., Wu, M., & Yu, H. (2020). Towards adversarial robustness via feature matching. IEEE Access, 8, 88594-88603.

This paper propose a regularizer encouraging the consistency in the artificial attention on the clean image and its adversarial counterpart.

This method shows improved empirical robustness over the state-of-the-art, secures 55.74% adversarial accuracy on CIFAR-10.

This paper also provide a variants of feature matching operations to seek the defense that exhibits the best robustness.

For further evaluation on the much challenging dataset, this implements several baselines from the literature on CIFAR-100 and conduct comparisons on it.

[11] Ren, K., Zheng, T., Qin, Z., & Liu, X. (2020). Adversarial attacks and defenses in deep learning. Engineering, 6(3), 346-360.

In this paper, it introduces the theoretical foundations, algorithms, and applications of adversarial attack techniques.

It also describes a few research efforts on the defense techniques, which cover the broad frontier in the field.

This paper also discuss about the effectiveness of these adversarial defenses based on the most recent advances.

Attacking methods like Fast Gradient Sign Method (FGSM), Jacobian Based Method, BIM and PGD, Distributionally adversarial attack¸ DeepFool, Universal adversarial attack, Adversarial patch, GAN-based attacks, Practical attacks, Obfuscated-gradient circumvention attacks are discussed.

Defending methods like Adversarial Training, Random noising, Random noising, Provable defenses, Recent advances in model robustness analysis are discussed in detail.

[12] ang, J., Han, K., Chen, H., & Li, Y. (2020). Ensemble adversarial black-box attacks against deep learning systems. Pattern Recognition, 101, 107184.

In this paper, proposed two types of ensemble-based black box attack strategies, selective cascade ensemble strategy (SCES) and stack parallel ensemble strategy (SPES), to explore the vulnerability of deep learning system.

SCES adopts boosting structure of ensemble learning to iteratively construct adversarial examples.

SPES employs bagging structure of ensemble learning to generate adversarial examples.

Experimental results show that our proposed ensemble adversarial attack strategies can successfully attack the deep learning system trained with ensemble adversarial training defense mechanism. The adversarial examples generated by SCES achieve better transferability than SPES.

[13] Ghosh, A., Mullick, S. S., Datta, S., Das, S., Das, A. K., & Mallipeddi, R. (2022). A black-box adversarial attack strategy with adjustable sparsity and generalizability for deep image classifiers. Pattern Recognition, 122, 108279.

In this article, the devise pixel-restricted i.e. sparse perturbations for universal attacks by using black-box feedback are introduced.

The pixel restriction means that we must identify the most critical pixels among a potentially large number of pixels, depending on the size of the input images.

`

They empirically validated the capability of DEceit to perform effective, imperceptible, and robust adversarial attacks.

[14] Wei, X., Guo, Y., & Li, B. (2021). Black-box adversarial attacks by manipulating image attributes. Information Sciences, 550, 285-296.

In this paper, The proposed model was adversarial attributes, an alternative way to generate adversarial examples.

This method manipulated the image attributes to perform the black-box attacks. To this end, utilized Differential Evolution to solve the optimal changing value.

The examples that are considered from the CIFAR10 was that adversarial attributes are somewhat more realistic than the real images and they will the alternative way to the adversarial examples than the adversarial noises.

It is consistent with the un-targeted attacks, i.e., AlexNet is more sensitive to the adversarial attributes than VGG16, Resnet50 and Inception v3.

[15] Lin, J., Njilla, L. L., & Xiong, K. (2022). Secure machine learning against adversarial samples at test time. EURASIP Journal on Information Security, 2022(1), 1-15.

It proposes a new iterative adversarial retraining approach to robustify the model and to reduce the effectiveness of adversarial inputs on DNN models.

The results from our extensive experiments demonstrate that the proposed approach increases the robustness of the DNN model against various adversarial attacks.

The Robust classifier has defend FGSM and given an accuracy of 100% on MNIST dataset and defend BIM given an accuracy of 47% on CIFAR-10 dataset.

Table 1.1:Literature Survey

	Reference (i.e. author names with reference number)	**year**	**Objectives**	**Limitations**	**Advantages**	**Performance metrics**	**Gaps**
Reference 1	Zhang, J., Qian, W., Nie, R., Cao, J., & Xu, D. (2022). Generate adversarial examples by adaptive moment	2022	To demonstrate an efficient iterative white-box attack method Adam-FGSM and how the attack performance	Although it is still at a long distance from the entire black box attack in actual	Adam-FGSM achieves the highest attack performance on target models with the same	Attack success rate of Adam-FGSM is 94.16% and misclassification confidence is about 0.884.	Furthermore, realistic constraints should be considered, for example, the numerical restriction that the pixel intensity of an image is an integer

	iterative fast gradient sign method. Applied Intelligence, 1-14.		outperforms other related attacks.	application scenarios.	iterations and perturbation .		in the interval [0,255].
Reference 2	Wu, F., Xiao, L., Yang, W., & Zhu, J. (2020). Defense against adversarial attacks in traffic sign images identification based on 5G. EURASIP Journal on Wireless Communications and Networking, 2020(1), 1-15.	2020	To propose a new defense method i.e SVD (singular value decomposition)and take full advantage of 5G high-speed bandwidth to resist the adversarial attack on self driving cars.	5G signal cannot be used due to signal interference in a specific area, the unmanned vehicle should switch to 4G or 3G in a timely manner and restore the communication connection .	The results show that adversarial attacks, such as Carlini Wagner's l2, Deepfool, and I-FSGM, can be better eliminated by the method and provide lower latency.	SVD+5G method has an accuracy of 87.58% and recall value of 96.17% with runtime of 7.12 ms.	To extend the experiment to different IoT scenarios and guarantee the secure development and stability of intelligent IoT in the coming 5G era.
Reference 3	Ren, H., Huang, T., & Yan, H. (2021). Adversarial examples: attacks and defenses in the physical world. International Journal of Machine Learning and Cybernetics, 12(11), 3325-3336.	2021	To summarize the corresponding generation model to formalize the adversarial attack in the digital and physical world and to investigate. To analyze the latest and important approaches to defenses against	The research conducted on adversarial attacks and defenses did not take the attack direction and defense direction into consideration.	Related defense mechanisms are provided for various adversarial attacks on classifiers, object detectors, and speech recognition systems in the physical world.	-	The future research directions can be conducted around the aspects such as attack direction and defense direction.

`

			adversarial attacks in the physical world.				
Reference 4	Yadav, A., Upadhyay, A., & Sharanya, S. (2022). An integrated Auto Encoder-Block Switching defense approach to prevent adversarial attacks. arXiv preprint arXiv:2203.10930.	2022	To propose a defense algorithm which utilizes the combination of an auto-encoder and block-switching architecture to counter FGSM attack.	The accuracy of the proposed model is limited against the FGSM attack and also the DNN model accuracy can be enhanced.	This proposed method can be extended to capture images from motion videos.	Attack model accuracy is 87% and the defense model accuracy is 88.54%.	The accuracy ,efficiency and robustness of the model need to be increased to higher range.
Reference 5	Hu, S., Nalisnick, E., & Welling, M. (2022). Adversarial Defense via Image Denoising with Chaotic Encryption. arXiv preprint arXiv:2203.10290.	2022	To propose a novel defense that assumes everything but a private key will be made available to the attacker which uses an image denoising procedure coupled with encryption via a discreted Baker map.	Without encryption, this defense method is still vulnerable to the PGD attacks.	This method is easy to implement, suitable for high-resolution inputs, and efficient in testing. Using encryption the defense slightly affects the natural and FGSM accuracy, but significantly boosts the PGD accuracy by 12 to 19%.	The dark gray box test accuracy is 78.96% against FGSM and 83.32% against PGD attack.	The verification of this model should be done so that the attacker cannot train a surrogate model to mimic the behavior of defender.

`

Reference 6	Siddiqi, R. (2022). Fruit-classification model resilience under adversarial attack. SN Applied Sciences, 4(1), 1-22.	2022	Three different Convolutional Neural Network (CNN) models 1)IndusNet, 2) fine-tuned VGG16, and 3) fine-tuned MobileNet are used for image classification. These model under adversarial training enables image classifiers to resist attacks crafted through the Fast Gradient Sign Method (FGSM)and improving classifiers robustness against other noise forms including Salt and pepper noise, Speckle noise, Gaussian noise.	Adversarial training is very high computation cost. FGSM will not work when gradient become zero.	Adversarial training has increased the robustness of DNN model	Fine-tuned VGG16 is the best performing CNN with an accuracy rate of 0.9482	CNN with an accuracy rate of 0.9482. The study only considers untargeted evasion attacks. Many other attack scenarios exist including friend-safe evasion attacks multi-targeted evasion attacks multi-targeted backdoor attacks etc.

`

Reference 7	Ren, K., Zheng, T., Qin, Z., & Liu, X. (2020). Adversarial attacks and defenses in deep learning. Engineering, 6(3), 346-360.	2020	Attacking methods like Fast Gradient Sign Method (FGSM), Jacobian Based Method, BIM and PGD, Distributionally adversarial attack, DeepFool and various attacks are discussed. Defending methods like Adversarial Training, Random noising, Random noising, Provable defenses, Recent advances in model robustness analysis are discussed in detail.	-	-	-	-

`

Reference 8	Jia, X., Zhang, Y., Wu, B., Wang, J., & Cao, X. (2022). Boosting fast adversarial training with learnable adversarial initialization. IEEE Transactions on Image Processing, 31, 4417-4430.	2022	In this paper, focusing on image classification, we propose a sample-dependent adversarial initialization to boost FGSM-based fast AT, dubbed FGSMSDI.	At the time of 70-epoch the trained model using FGSM-RS falls into the catastrophic over fitting problem that the trained model cannot defend against the adversarial examples generated by PGD-based attack methods during the training process.	The model FGSM-SDI is proposed initialization overcomes the catastrophic over fitting, thus improves model robustness when compared with the widely used random initialization with the fast AT.	Imagenet-In terms of training efficiency, similar phenomenon are observed on other databases, our FGSM-SDI can be 3 times faster than the advanced PGD-AT. The previous fast AT methods achieve the performance of about 20% under the PGD-50 attack which is far from that of the advanced PGD-AT which achieves about 28% accuracy. While the proposed FGSM-SDI achieves the performance of about 30% under the PGD-50 attack.	Although other fast AT methods overcome the catastrophic over fitting issue, their performance is far from satisfactory, i.e., there is a huge gap between the fast AT methods and the advanced multi-step PGD-AT.

`

Reference 9	Ghosh, A., Mullick, S. S., Datta, S., Das, S., Das, A. K., & Mallipeddi, R. (2022). A black-box adversarial attack strategy with adjustable sparsity and generalizability for deep image classifiers. Pattern Recognition, 122, 108279.	2022	In this article, the devise pixel-restricted i.e. sparse perturbations for universal attacks by using black-box feedback. The pixel restriction means that we must identify the most critical pixels among a potentially large number of pixels, depending on the size of the input images.	GenAttack requires a higher number of average queries which may be due to its inability to handle high-dimension al optimizati on problems. Whereas, SparseFool though capable of performing while incurring a slightly lower computatio nal cost is neither a black-box technique nor can be directly extended to UAA, making DEceit an attractive alternative.	To effectively solve this real-valued high-dimensional optimization problem, we devised DEceit by modifying DE by incorporatin g a novel mix of scale-factor and mutation switching strategies. Finally, we empirically validated the capability of DEceit to perform effective, imperceptibl e, and robust adversarial attacks.	Comparison of Deceit with non targeted UAA methods on ImageNet2012 Validation dataset the fooling rate of the VGG16 with 50176 was about 83.96 and for the inception V3 at 50176 was 90.37 and in this paper considered total of 5 algorithms.	DEceit may be extended by focusing on the contradictory objectives of maximizing the effectiveness of a perturbation while minimizing the visual distortions, which can be expressed as a multiobjective optimization problem, solvable by a tailored DE variant. DEceit may also benefit from searching a perturbation in a transformed space instead of the native image space.

`

Reference 10	Hang, J., Han, K., Chen, H., & Li, Y. (2020). Ensemble adversarial black-box attacks against deep learning systems. Pattern Recognition, 101, 107184.	2020	In this paper, proposed two types of ensemble-based blackbox attack strategies, selective cascade ensemble strategy (SCES) and stack parallel ensemble strategy (SPES), to explore the vulnerability of deep learning system. SCES adopts boosting structure of ensemble learning to iteratively construct adversarial examples. SPES employs bagging structure of ensemble learning to generate adversarial examples.	The limitations are only space that the results are not discussed more the accuracy measures are not mentioned more in this paper.	Experimental results show that our proposed ensemble adversarial attack strategies can successfully attack the deep learning system trained with ensemble adversarial training defense mechanism. The adversarial examples generated by SCES achieve better transferability than SPES.	The target classifier trained with training data in each dataset and its architecture. The target model achieves test accuracy rates 99.19% for MNIST, 98.70% for USPS and 98.09% GTSRB, respectively	This paper had ensemble many pre trained substitute models but it can be reduced and the potential factors can be reduced so trhat the transfer rate would be more and accurate.

Reference 11	Wei, X., Guo, Y., & Li, B. (2021). Black-box adversarial attacks by manipulating image attributes. Information Sciences, 550, 285-296.	2021	In this paper, The proposed model was adversarial attributes, an alternative way to generate adversarial examples. This method manipulated the image attributes to perform the black-box attacks. To this end, utilized Differential Evolution to solve the optimal changing value.	In this paper, the adversarial attributes that are generated using adversarial examples are unable to predict the correct figure in the first pair of four pair in the CIFAR 10.	The examples that are considered from the CIFAR10 was that adversarial attributes are somewhat more realistic than the real images and they will the alternative way to the adversarial examples than the adversarial noises.	The fooling rates of targeted attacks are below compared with un-targeted attack that is expected because targeted attack is more difficult than un-targeted attack. L2 norm shows better fooling rate than L1 norm on all the four DNN models. The best fooling rate is achieved on attacking AlexNet (64.67%). It is consistent with the un-targeted attacks, i.e., AlexNet is more sensitive to the adversarial attributes than VGG16, Resnet50 and Inception v3.	Qualitative results told us the adversarial examples were normal, and to some extent, they were more beautiful and artistic than the clean images. Therefore, they would not arouse people's suspicion. But they are not completely normal and useful than the adversarial noises.

`

Reference 12	Echeberria-Barrio, X., Gil-Lerchundi, A., Egana-Zubia, J., & Orduna-Urrutia, R. (2022). Understanding deep learning defenses against adversarial examples through visualizations for dynamic risk assessment. Neural Computing and Applications, 1-14.	2022	This visualization is implemented in three defenses named as adversarial training, dimensionality reduction, and prediction similarity. The objective of those implementations is to show how well this visualization could help to understand the decision-making of a deep learning model.	Deep neural networks are misty machines, where decision-making is really diffificult to understand and that is the reason why the deep learning defenses.	In this paper defense methods reduces the noise (irrelevant information) for the predictions. This visualization allows identifying the vulnerabilities of the model and shows how the defenses try to avoid them.	Adversial training method gave 90% of accuracy on known adversials and Autoencoder method gave better accuracy on new adversials	In the future, this knowledge will allow to development of more effificient defenses and detectors to combat different threats, including the adversary attack, using these visualizations
Reference 13	Lin, J., Njilla, L. L., & Xiong, K. (2022). Secure machine learning against adversarial samples at test time. EURASIP Journal on Information Security, 2022(1), 1-15.	2022	It proposes a new iterative adversarial retraining approach to robustify the model and to reduce the effectiveness of adversarial inputs on DNN models.	In this paper white-box attacks (the attacker knows model architecture) and black-box attacks (the attacker does not know the model architecture).	The results from our extensive experiments demonstrate that the proposed approach increases the robustness of the DNN model against various adversarial attacks.	The Robust classifier has defend FGSM and given an accuracy of 100% on MNIST datase and defend BIM given an accuracy of 47% on CIFAR-10 dataset.	Future work will explore black-box attacks and formal guarantee for performance.

`

Reference 14	Yadav, A., Upadhyay, A., & Sharanya, S. (2022). An integrated Auto Encoder-Block Switching defense approach to prevent adversarial attacks.arXiv preprint arXiv:2203.10930.	2022	This article proposes a defense algorithm which utilizes the combination of an auto-encoder and block-switching architecture. Auto-coder is intended to remove any perturbations found in input images whereas block switching method is used to make it more robust against White-box attack.	Requires additional Time. It can also be termed as bottleneck for fast models. Not efficient for Black box attacks.	Detects anomaly and avoid input reaching to the actual model.Makes model robust against those Adversarial input which is trained.	The attack by the FGSM model is effectively countered by the proposed defence architecture with an accuracy of 88.54%.	Should propose new defending technique to defend c and w attack
Reference 15	Kherchouche, A., Fezza, S. A., & Hamidouche, W. (2022). Detect and defense against adversarial examples in deep learning using natural scene statistics and adaptive denoising. *Neural Computing and Applications*, *34*(24), 21567-	2022	This paper show that the proposed defense method outperforms the state-of-the-art defense techniques by improving the robustness against a set of attacks under black-box, gray-box and white-box settings. we propose a framework for	Large computational time. DNN should be feed forward network	The effificiency of the detector with high detection accuracy and low FP helps to preserve the classifificaion accuracy of the DNN model on clean images.	Given CNN classififier for MNIST dataset resulting in a state-of-the-art accuracy of 99:4%. For CIFAR-10 and Tiny-ImageNet datasets,considered existing models achieving accuracy scores of 98.5% and 69.2%	As future work, we plan to extend the proposed defense method to deal with adversarial attacks in the physical world.

	21582.		defending DNN classifier against adversarial samples.				
Reference 16	Xu, H., Ma, Y., Liu, H. C., Deb, D., Liu, H., Tang, J. L., & Jain, A. K. (2020). Adversarial attacks and defenses in images, graphs and text: A review. International Journal of Automation and Computing, 17(2), 151-178.	2020	In this paper, we review the state of the art algorithms for generating adversarial examplesand the countermeasures against adversarial examples, for three most popular data types, including images, graphs and text.	-	-	-	-
Reference 17	Wang, D., Li, C., Wen, S., Nepal, S., & Xiang, Y. (2020). Defending against adversarial attack towards deep neural networks via collaborative multi-task training. IEEE Transactions on Dependable and Secure Computing.	2020	Deep neural networks (DNNs) are known to be vulnerable to adversarial examples which contain human-imperceptible perturbations proactive defending methods are invalid against grey-box or white-box attacks	In this paper proposed defence based on the properties of non-targeted attacks and is designed for the non-targeted attacks. For targeted attacks, the proposed defence can be	The proposed defence can defend both black-box attacks and grey-box attacks without knowing the details of the attacks in advance.	In this paper defending method achieved up to 96 .3% classification accuracy on black-box adversarial examples, and detected up to 98 .7% of the high confidence adversarial examples.	Further this paper can develop on the defence for targeted attacks.

				further strengthened by training the classmaps based on the targeted attacks.			
Reference 18	Khamaiseh, S. Y., Bagagem, D., Al-Alaj, A., Mancino, M., & Alomari, H. W. (2022). Adversarial Deep Learning: A Survey on Adversarial Attacks and Defense Mechanisms on Image Classification. IEEE Access.	2022	In this paper the attacks mentioned are L-BFGS,FGSM, DeepFool attack, c&w attack, I-FGSM etc and the defenses mentioned are defensive distillation, gradient regularization, etc.	There are some limitations in each defence and attack proposed in this paper.	Mini-batch gradient descent has two benefits over other gradient descent techniques: First, it computes the gradient of the cost function more quickly than BGD. Second, heading towards the minimum, mini-batch gradient descent swings less than stochastic gradient descent with a high number of batches.	Brendel & Bethge attack is a proficient algorithm, achieving accuracies of 69.5% in MNIST dataset, 31.2% on the CIFAR, and 42.5% on ImageNet. The ZOO attack is extremely effective, with success rates of 100% in an untargeted setting on both the MNIST and CIFAR-10 datasets. Upon utilizing a targeted setting, the attack success rate is 98.9% against MNIST and 97% on CIFAR-10.	Further this paper have to review on the open issues and challenges by doing further investigation .

`

Reference 19	Li, Z., Feng, C., Zheng, J., Wu, M., & Yu, H. (2020). Towards adversarial robustness via feature matching. IEEE Access, 8, 88594-88603.	2020	When faced with the constant threat of adversarial attacks, deep neural networks performance spectacularly decreases. Deep neural networks handle a range of computer vision tasks with greater accuracy.In this the paper prosed the method is future matching.	One of the limitations of this work is the proposed method has not been scaled and validated for more sophisticated benchmark datasets (such as ImageNet) or the real world Scenarios.	A model essentially trained by the proposed method improves robustness according to the latest technology. Wide range of powerful attacks in white box settings, standard benchmark data set CIFAR-10, and even challenges CIFAR-100.	This paper achieve new state-of-the-art adversarial accuracy on CIFAR-10 with under untargeted attack in the highly challenging white-box settings. Specifically, we get 51.54%, 55.74% and 52.95% adversarial accuracy under strong attacks named C&W, PGD.	This paper will give you a rough overview of future work integrating bio-inspired mechanisms into artificial mechanisms and intelligence methods have great potential for further outcomes improvement.
Reference 20	Papernot, N., McDaniel, P., Wu, X., Jha, S., & Swami, A. (2016, May). Distillation as a defense to adversarial perturbations against deep neural networks. In 2016 IEEE symposium on security and privacy (SP) (pp. 582-597). IEEE.	2016	Proposed a novel defensive mechanism called defensive distillation to reduce the effectiveness of adversarial samples on DNN. In defensive training one model is trained to predict the output probabilities and these probabilities	Does not able to defend c and w attacks.	It reduces the sensitivity of a DNN to input perturbations by a factor of 10^30.	Defensive distillation reduces effectiveness of adversarial sample creation from 95% to 0.5%.	Defensive distillation reduces effectiveness of adversarial sample creation from 95% to 0.5%. Do not consider attacks at training time.

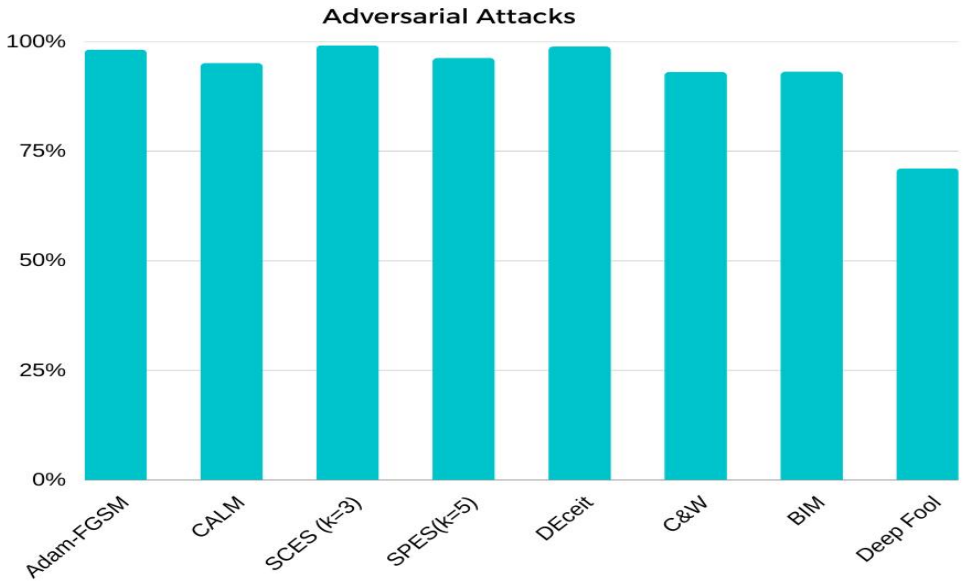
			are use train another for same input which helps DNN model to more generalized and helps in predicting the unseen samples well.				
Reference 21	Carlini, N., & Wagner, D. (2017, May). Towards evaluating the robustness of neural networks. In 2017 ieee symposium on security and privacy (sp) (pp. 39-57). Ieee.	2017	Proposed a novel attacking approach c and w attack which successfully demonstrated that defensive distillation is not robust enough under their attacking approach c and w attack. Construct three attacks for L0, L1, L∞ to prove the weakness of these defensive methods.	Dose not effect most sophisticated networks have more complicated structures (e.g., ResNet and Inception) .	Drastically reduce the success rate of defensive distillation.	The 91% success rate at T=1 has reduced to a 24% success rate for T=5 and finally 0.5% success at T=100.	Should propose new defending technique to defend c and w attack.

`

Reference 22	Chakraborty, A., Alam, M., Dey, V., Chattopadhyay, A., & Mukhopadhyay, D. (2018). Adversarial attacks and defenses: A survey. arXiv preprint arXiv:1810.00069.	2018	Discussed three types of attacks Evasion Attack, Poisoning Attack, Exploratory Attack. Attacking method like Generative Adversarial Attack, Adversarial Examples Generation methods like L-BFGS, Fast Gradient Sign Method (FGSM), Jacobian Based Method.	-	-	-	-

Graphical representation with literature survey

Fig 1.1: Adversarial attacks



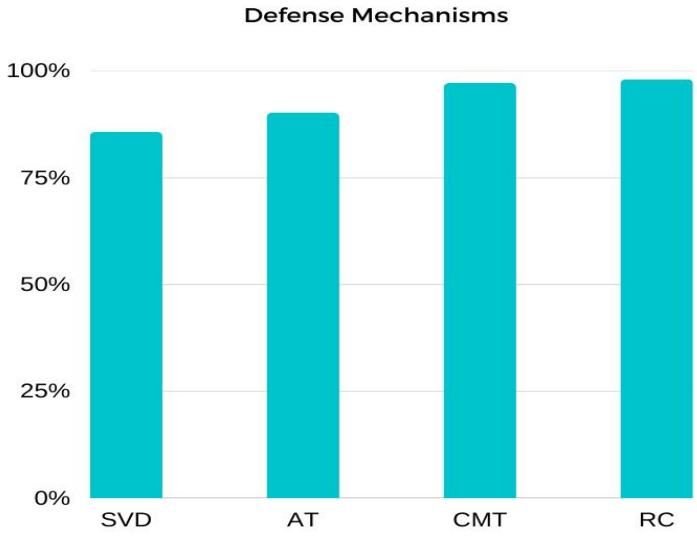


Fig 1.2: Defense mechanisms

`

REQUIREMENT SPECIFICATION

Functional requirements:

User input : Traffic sign board image for attacking and defending to secure the model.

Non functional requirements:

Performance : The model should able to defend the model from various attacking techniques.

Accuracy : The model should be accurate in discovering the attack and should counterattack the technique.

Scalability : The model should able to remove the noise and generate accurate results in a limited time.

Software Requirements:

Integrated python notebook with installed packages - tensorflow, pandas, numpy, matplotlib.

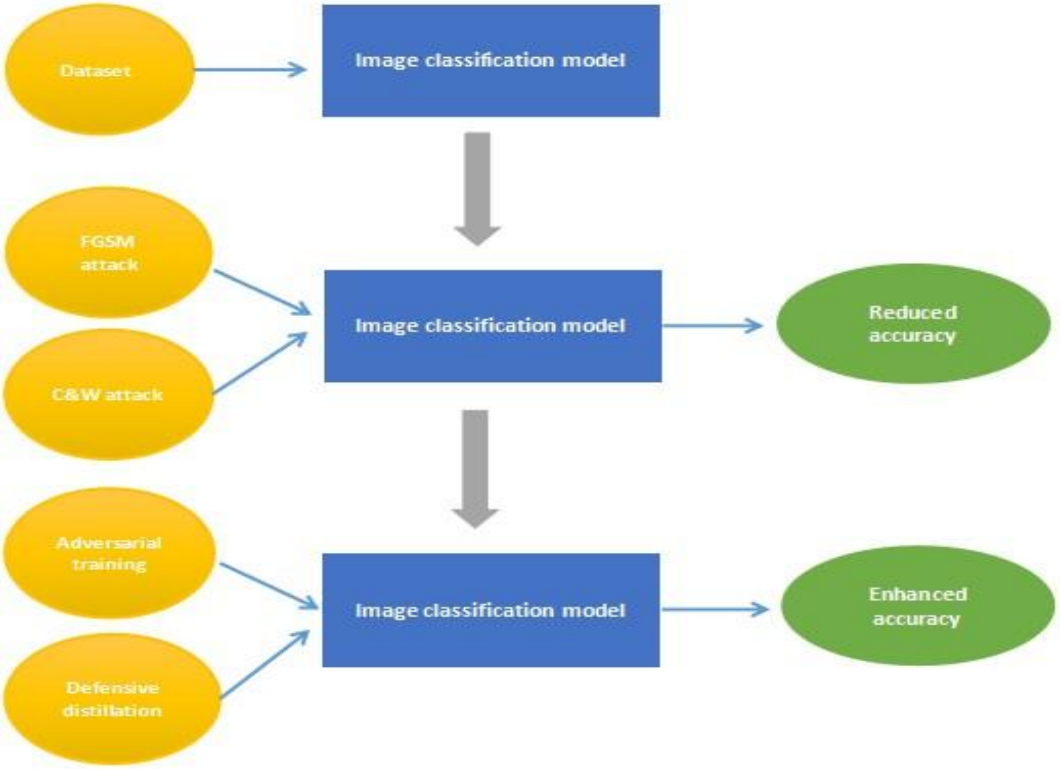
`

SYSTEM ANALYSIS AND DESIGN

In this project, image classification model is attacked by some of the adversarial attacks and then defended by the defense techniques. For the image classification model, we consider the dataset GTSRB(German Traffic Sign Recognition Benchmark) which consists of 43 classes of different traffic signs with a total of 39,209 images. For image classification model, we use L-net model which uses tensorflow.keras sequential model which consists of 7 layers. We train the model with the above dataset and record the accuracy of the model. By considering the trained model, attacking techniques are applied like FGSM (Fast Gradient Sign Method) and C&W (carlini and Wager) attacks which are white box attacks which can access the model. After applying the attacks on the model, the model works with reduced accuracy. To enhance the reduced accuracy of the model, we apply defense techniques such as adversarial training and autoencoder which try to remove the perturbations added to the images by the attack methods which in turn increases the accuracy of the model.

Flow chart

Fig 2.1: Flow chart of the model



`

Algorithm

FGSM(Fast Gradient Sign Method):

FGSM is used to add the noise whose direction is the same as the gradient of the cost function with respect to the data. This method involves 3 steps -

1) Calculate the loss after forward propagation,

2) Calculate the gradient with respect to the pixels of the image,

3) Nudge the pixels of the image ever so slightly in the direction of the calculated gradients that maximize the loss calculated above.

C&W attack:

C&W attack is a powerful attack method which can achieve success even on distilled networks. This technique can also have targeted transferable examples to carry out a black-box assault by altering the confidence.

An example of an adversarial assault on machine learning models is the Carlini and Wagner (C&W) attack. The purpose of this attack is to subtly alter an input so that a machine learning model incorrectly classifies it. The C&W attack is regarded as a potent attack because it has the ability to provide adversarial instances that are very good at deceiving machine learning models, even those that have been trained with robust protections against other forms of attacks.

The C&W attack entails figuring out the smallest input perturbation that will lead a machine learning model to incorrectly classify an input. The decision boundary of the model's sensitivity to minute changes in the input is taken into consideration by the optimization problem.

Adversarial training:

Adversarial training is a defensive method which includes adversarial samples in the training stage and then explicitly trains the model to detect these tricks. It is a brute force approach that increases the robustness of the model.

`

Adversarial training improves the robustness of the model to adversarial attacks. The input data is perturbed to the point where the model incorrectly classifies it in order to produce the adversarial examples required in adversarial training. Many methods, including the Fast Gradient Sign Method (FGSM) and the Carlini and Wagner (C&W) assault, can be used to accomplish this. These adversarial samples are produced and added to the training set so that the model can become more resistant to similar attacks during inference. The model is trained using both the initial training set and the generated adversarial examples during adversarial training. The model learns to recognize the minute alterations in the input that can result in misclassification because the adversarial instances are typically produced with a slight perturbation to the original data. Since creating adversarial instances for each training example is necessary for adversarial training, it might be computationally expensive. It has been demonstrated to be successful in enhancing the robustness of machine learning models against hostile attacks.

Auto encoder:

The input image is given some noise, which the denoising autoencoders learn to eliminate. Consequently, it is prevented from copying the input to the output without discovering aspects of the data. While training to recover the original, undistorted input, these autoencoders use a partially corrupted input. In order to remove the extra noise, the model learns a vector field for mapping the input data towards a lower-dimensional manifold that describes the natural data. The encoder will be able to extract the most crucial features this way and learn a more reliable representation of the data.

Proposed model:

The proposed model in the paper is for classifying the images among the given input that describes about the increase in accuracy and the robustness. This model consists of 13 convolutional layers (Conv2D) and 5 transpose convolutional layer and 5 skip connection layers.

The accuracy of this model gradually increased for less number of images but there is

`

slight change in the accuracy of 1.00 when the input to model was increased.

The proposed model has given better results on c and w attack that the model has correctly classified all the images when the input is only 50 images. But the accuracy has decreased to 0.97 when the input is 100 images and the confidence levels is around 85 to 90.

Existing System

Detection of a deep learning model's vulnerability.

The identification of an attack's exploitation zone.

Creation of deep learning models while keeping the same proposal and taking into account the weaknesses of the prior model.

Creation of deep learning models that take into account the assault exploitation zones.

Creation of defenses aimed at defending the targeted vulnerabilities found.

Creation of defenses against a particular attack with an emphasis on its weak points.

Creation of countermeasures based on this visualization technique.

Proposed System

We consider a image classification model, to check the vulnerability of the model we perform attack on the model.

We propose an ensemble attack technique using the defined techniques such as FGSM (Fast Gradient Sign Method) and C&W (Carlini and Wagner).

This leads to the reduced accuracy of the model.

To enhance the accuracy of the model we apply defending techniques such as adversarial training and auto encoder.

Adversarial training works by training with the adversarial examples.

Auto encoder works by reconstructing the original image from the damaged image.

`

IMPLEMENTATION

Fig 2.2:Importing

```
import time
import numpy as np
import pandas as pd
import os
import cv2
from PIL import Image
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
np.random.seed(42)
tf.random.set_seed(42)
```

```
import tensorflow as tf
```

```
from tensorflow.keras import layers
```

```
start = time.time()
image_data = []
image_labels = []
```

This code imports necessary libraries such as **time**, **numpy**, **pandas**, **os**, **cv2**, **PIL**, **matplotlib**, **tensorflow**, **keras**, **train_test_split**, and **accuracy_score**. **time** library is used to measure the execution time of the code. **numpy** is used for scientific computing and mathematical operations in the code. **pandas** is used to work with data structures and analysis of data. **os** is used to provide a way of interacting with the file system in the code. **cv2** is used for computer vision operations such as image processing, image filtering, and image recognition. **PIL** is used for image processing and handling. **matplotlib** is used for data visualization. **tensorflow** is an open-source machine learning library developed by Google. **keras** is a high-level neural networks API built on top of TensorFlow, which is used to build and train machine learning models. **train_test_split** is used to split the dataset into training and testing sets. **accuracy_score** is used to evaluate the performance of the machine learning model.The code also sets the seed values for **numpy** and **tensorflow** to ensure reproducibility of the results.

Fig 3.1:Importing dataset and images

```
total_classes = 43
height = 32
width = 32
channels = 3
input_path = '/kaggle/input/gtsrb-german-traffic-sign/'

for i in range(total_classes):
    path = input_path + 'Train/' + str(i)
    print(path)
    images = os.listdir(path)

    for img in images:
        try:
            image = cv2.imread(path + '/' + img)
            image_fromarray = Image.fromarray(image, 'RGB')
            resize_image = image_fromarray.resize((height, width))
            image_data.append(np.array(resize_image))
            image_labels.append(i)
        except:
            print("Error - Image loading")

#Converting lists into numpy arrays
image_data = np.array(image_data)
image_labels = np.array(image_labels)
```

This code initializes some variables and reads images from a directory using OpenCV (**cv2**) and **PIL** libraries. The images are resized to 32x32 and stored in a NumPy array called **image_data**. The corresponding labels of the images are also stored in another NumPy array called **image_labels**. The loop iterates through the total number of classes (43 in this case), and for each class, it reads all the images from the directory and resizes them to the required dimensions. If there is an error loading an image, the code prints an error message. Finally, the **image_data** and **image_labels** lists are converted into NumPy arrays. The shape of the arrays is printed, and the execution time is measured using **time** library. Overall, this code is loading and preparing the data for a machine learning model to be trained on the German Traffic Sign Recognition Benchmark dataset (GTSRB) which contains 43 different classes of traffic signs.

Fig 3.2: Splitting data

`

```
#shuffling data
shuffle_indexes = np.arange(image_data.shape[0])
np.random.shuffle(shuffle_indexes)
image_data = image_data[shuffle_indexes]
image_labels = image_labels[shuffle_indexes]
```

```
#Splitting training and testing dataset
X_train, X_valid, y_train, y_valid = train_test_split(image_data, image_labels, test_size=0.2, random_state=42, shuffle=True)

X_train = X_train/255
X_valid = X_valid/255

print("X_train.shape", X_train.shape)
print("X_valid.shape", X_valid.shape)
print("y_train.shape", y_train.shape)
print("y_valid.shape", y_valid.shape)
```

```
X_train.shape (31367, 32, 32, 3)
X_valid.shape (7842, 32, 32, 3)
y_train.shape (31367,)
y_valid.shape (7842,)
```

This code shuffles the **image_data** and **image_labels** arrays using **np.random.shuffle()** function. **np.arange(image_data.shape[0])** creates an array of indices from 0 to the number of images in the dataset. Then **np.random.shuffle()** shuffles this array of indices randomly. Finally, the **image_data** and **image_labels** arrays are rearranged according to the shuffled indices using NumPy's indexing feature, which shuffles the images and their corresponding labels in the same way. The purpose of shuffling the data is to avoid any bias in the order of the data, which could lead to the model learning the order of the data instead of the actual patterns in the data. By shuffling the data, we ensure that each batch of data the model trains on is representative of the entire dataset.

Fig 4.1:Model

```
model = keras.models.Sequential([
    keras.layers.Conv2D(filters=6, kernel_size=(5,5), strides=1, activation='tanh', input_shape=(height,width,channels)),
    keras.layers.AveragePooling2D(pool_size=(2,2), strides=2),
    keras.layers.Conv2D(filters=16, kernel_size=(5,5), strides=1, activation='tanh'),
    keras.layers.AveragePooling2D(pool_size=(2,2), strides=2),
    keras.layers.Conv2D(filters=120, kernel_size=(5, 5), activation='tanh'),
    keras.layers.Flatten(),
    keras.layers.Dense(units=84, activation='tanh'),
    keras.layers.Dense(units=43, activation='softmax'),
])
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 6)         456
_________________________________________________________________
average_pooling2d (AveragePo (None, 14, 14, 6)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 10, 10, 16)        2416
_________________________________________________________________
average_pooling2d_1 (Average (None, 5, 5, 16)          0
```

`

This code defines a Sequential model in Keras for image classification. The model consists of three convolutional layers and two fully connected (dense) layers. The first convolutional layer has 6 filters and a kernel size of 5x5. The activation function used is hyperbolic tangent (**tanh**). The **input_shape** parameter specifies the shape of the input images, which is (height, width, channels). The second layer is an average pooling layer with a pool size of 2x2 and a stride of 2. The third layer is another convolutional layer with 16 filters and a kernel size of 5x5. The activation function is again hyperbolic tangent. The fourth layer is another average pooling layer with a pool size of 2x2 and a stride of 2. The fifth layer is a convolutional layer with 120 filters and a kernel size of 5x5. The activation function used is hyperbolic tangent. The sixth layer is a flatten layer that flattens the output of the convolutional layers into a 1D vector. The seventh layer is a dense layer with 84 units and a hyperbolic tangent activation function. The eighth layer is another dense layer with 43 units (equal to the number of classes in the dataset) and a softmax activation function. Softmax activation function is used as it ensures the output of the model are probabilities and they sum up to 1. Overall, the model follows the LeNet-5 architecture.

Fig 4.2:Compilation of model

```
#Compilation of the model
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

```
epochs = 20
history = model.fit(X_train, y_train, batch_size=32, epochs=epochs,
validation_data=(X_valid, y_valid))
```

```
Epoch 1/20
981/981 [==============================] - 4s 4ms/step - loss: 0.8877 - accuracy: 0.7721 - val_loss: 0.2820 - val_accuracy: 0.9287
Epoch 2/20
981/981 [==============================] - 3s 3ms/step - loss: 0.1950 - accuracy: 0.9522 - val_loss: 0.1391 - val_accuracy: 0.9658
Epoch 3/20
981/981 [==============================] - 3s 3ms/step - loss: 0.1061 - accuracy: 0.9754 - val_loss: 0.1005 - val_accuracy: 0.9754
Epoch 4/20
981/981 [==============================] - 4s 4ms/step - loss: 0.0662 - accuracy: 0.9845 - val_loss: 0.0825 - val_accuracy: 0.9797
Epoch 5/20
981/981 [==============================] - 3s 3ms/step - loss: 0.0423 - accuracy: 0.9910 - val_loss: 0.0740 - val_accuracy: 0.9820
Epoch 6/20
981/981 [==============================] - 3s 3ms/step - loss: 0.0318 - accuracy: 0.9931 - val_loss: 0.0683 - val_accuracy: 0.9819
Epoch 7/20
981/981 [==============================] - 4s 4ms/step - loss: 0.0243 - accuracy: 0.9941 - val_loss: 0.0824 - val_accuracy: 0.9755
Epoch 8/20
981/981 [==============================] - 3s 3ms/step - loss: 0.0183 - accuracy: 0.9957 - val_loss: 0.0578 - val_accuracy: 0.9839
```

This code compiles and trains the Keras model. The **compile()** method configures the model for training. The **loss** parameter is set to **'categorical_crossentropy'**, which is a common loss function used for multi-class classification problems. The **optimizer** parameter is set to **'adam'**, which is an efficient gradient descent algorithm. The **metrics** parameter is set to **'accuracy'**, which is a performance metric used to evaluate the model. The **fit()** method trains the model for the specified number of epochs (20 in this case) using the training set. The **batch_size** parameter is set to 32, which means that the model will be updated after every 32 samples. The **validation_data** parameter is set to the validation set, which will be used to evaluate the performance of the model after each epoch. The training progress and performance will be stored in the **history** object, which can be used for plotting and analysis.

Fig 5.1:Running epochs

```
x_test_pred1=np.concatenate((X_train,x_test_pred1),axis=0)
x_test_pred=np.concatenate((X_valid,x_test_pred),axis=0)
y_train=np.concatenate((y_train,y_train),axis=0)
y_valid=np.concatenate((y_valid,y_valid),axis=0)
```

```
epochs = 20
history = model.fit(x_test_pred1, y_train, batch_size=32, epochs=epochs,
validation_data=(x_test_pred, y_valid))
```

```
Epoch 1/20
1961/1961 [==============================] - 7s 4ms/step - loss: 0.5113 - accuracy: 0.8590 - val_loss: 0.4969 - val_accuracy: 0.
8734
Epoch 2/20
1961/1961 [==============================] - 7s 4ms/step - loss: 0.2118 - accuracy: 0.9361 - val_loss: 0.4478 - val_accuracy: 0.
8900
Epoch 3/20
1961/1961 [==============================] - 7s 4ms/step - loss: 0.1567 - accuracy: 0.9521 - val_loss: 0.4456 - val_accuracy: 0.
8914
Epoch 4/20
1961/1961 [==============================] - 7s 3ms/step - loss: 0.1219 - accuracy: 0.9632 - val_loss: 0.4340 - val_accuracy: 0.
8950
Epoch 5/20
1961/1961 [==============================] - 7s 4ms/step - loss: 0.1038 - accuracy: 0.9682 - val_loss: 0.4699 - val_accuracy: 0.
8862
Epoch 6/20
```

Fig 5.2: FGSM attack

```
loss_object = tf.keras.losses.CategoricalCrossentropy()

def create_adversarial_pattern(input_image, input_label,eps):
    with tf.GradientTape() as tape:
        tape.watch(input_image)
        prediction = model(input_image)
        loss = loss_object(input_label, prediction)
    gradient = tape.gradient(loss, input_image)
    signed_grad = tf.sign(gradient)
    adversary = (input_image + (signed_grad * eps)).numpy()
    return adversary
```

```
y_test_pred=[]
x_test_pred=[]
for i in range(0,len(X_valid)):
    img = tf. convert_to_tensor(X_valid[i])
    img= tf.reshape(img,(1,32, 32, 3))
    j = y_valid[i]
    j = tf.reshape(j,(1, 43))
    x_test_per=create_adversarial_pattern(img,j,0.53)
    x_test_pred.append(x_test_per)
    pred = model.predict(x_test_per)
    y_test_pred.append(pred)
```

`

This code defines a function **create_adversarial_pattern()** that generates an adversarial perturbation for a given input image and label. The function uses the TensorFlow GradientTape API to compute the gradient of the loss with respect to the input image, and then applies the sign function to the gradient to obtain the sign of each element. The signed gradient is then used to create the adversarial perturbation by scaling it with the epsilon value and adding it to the input image. The function returns the adversarial image as a numpy array. The code then creates adversarial examples for each image in the validation set by calling the **create_adversarial_pattern()** function and storing the adversarial image and its prediction in **x_test_pred** and **y_test_pred** lists, respectively. The epsilon value used in the function is 0.53, which is a hyper parameter that determines the strength of the adversarial perturbation. The value of epsilon was probably chosen based on some experimentation and validation on the validation set. It is worth noting that generating adversarial examples for an entire dataset can be computationally expensive, especially if the dataset is large. Therefore, it is important to consider the tradeoff between the number of examples generated and the available computing resources.

Fig 6.1: Evaluating the model

```
x_test_pred=np.reshape(x_test_pred, (len(y_valid),32, 32, 3))
```

```
#evaluating the model
score = model.evaluate(x_test_pred,y_valid, verbose = 0)
print('test score: ', score[0])
print('test accuracy: ', score[1])
```

```
test score:  15.617783546447754
test accuracy:  0.004590665455907583
```

```
score = model.evaluate(X_valid, y_valid, verbose = 0)
print('test score: ', score[0])
print('test accuracy: ', score[1])
```

```
test score:  0.05316957086324692
test accuracy:  0.9876307249069214
```

This code evaluates the model after and before attacking the model.

`

Fig 6.2: Carlini and Wagner attack

```
from cleverhans.tf2.attacks import carlini_wagner_l2
from cleverhans.tf2.attacks.carlini_wagner_l2 import carlini_wagner_l2
logits_model = tf.keras.Model(model.input,model.layers[-1].output)
x_test_per=carlini_wagner_l2(logits_model,X_valid[None,1,:,:,:],targeted=False)
image = model.predict(X_valid[None,1,:,:,:])
#before attack
plt.figure()
plt.imshow(X_valid[1])
preds=np.argmax(image, axis=-1)
plt.title('{} : {:.2f}% Confidence'.format(classes[preds[0]], image[0][preds[0]]*100))
plt.show()
image_probs = model.predict(x_test_per)
#after c and w attack
plt.figure()
plt.imshow(x_test_per.reshape(32, 32, 3))
preds1=np.argmax(image_probs, axis=-1)
plt.title('{} : {:.2f}% Confidence'.format(classes[preds1[0]], image_probs[0][preds1[0]]*100))
plt.show()
```

```
logits_model = tf.keras.Model(model.input,model.layers[-1].output)
```

```
x_test_per=carlini_wagner_l2(logits_model,X_valid[None,1,:,:,:],targeted=False)
```

The code generates an adversarial example using the Carlini-Wagner L2 attack and compares it with the original example. The **logits_model** is defined as a Keras model that takes the input tensor and outputs the logits of the last layer (i.e., the layer before softmax activation). The adversarial example is generated using the **carlini_wagner_l2** function from the **cleverhans.tf2.attacks** module, which takes the **logits_model**, the input examples (**X_valid[1:2]**), and a **targeted** flag (set to **False** in this case, which means non-targeted attack). The predicted classes and confidence scores of the original and adversarial examples are computed using the **model.predict** method, and the results are plotted using **matplotlib**. Note that the original example is plotted first, followed by the adversarial example.

`

Fig 7.1: Autoencoder

```
def AutoEncdoer(input_shape):
    inputs = layers.Input(shape=input_shape)
    conv1 = Conv2DLayer(inputs, 64, 3, strides=1, padding='same', block_id=1)
    conv2 = Conv2DLayer(conv1, 64, 3, strides=2, padding='same', block_id=2)
    conv3 = Conv2DLayer(conv2, 128, 5, strides=2, padding='same', block_id=3)
    conv4 = Conv2DLayer(conv3, 128, 3, strides=1, padding='same', block_id=4)
    conv5 = Conv2DLayer(conv4, 256, 5, strides=2, padding='same', block_id=5)
    conv6 = Conv2DLayer(conv5, 512, 3, strides=2, padding='same', block_id=6)
    conv7 = Conv2DLayer(conv6, 1024, 3, strides=2, padding='same', block_id=60)
    deconv0 = Transpose_Conv2D(conv7, 1024, 3, strides=2, padding='same', block_id=7)
    skip0 = layers.concatenate([deconv0, conv6], name='skip0')
    conv77 = Conv2DLayer(skip0, 512, 3, strides=1, padding='same', block_id=80)
    deconv1 = Transpose_Conv2D(conv77, 256, 3, strides=2, padding='same', block_id=90)

    skip1 = layers.concatenate([deconv1, conv5], name='skip1')
    conv7 = Conv2DLayer(skip1, 256, 3, strides=1, padding='same', block_id=8)
    deconv2 = Transpose_Conv2D(conv7, 128, 3, strides=2, padding='same', block_id=9)
    skip2 = layers.concatenate([deconv2, conv3], name='skip2')
    conv8 = Conv2DLayer(skip2, 128, 5, strides=1, padding='same', block_id=10)
    deconv3 = Transpose_Conv2D(conv8, 64, 3, strides=2, padding='same', block_id=11)
    skip3 = layers.concatenate([deconv3, conv2], name='skip3')
    conv9 = Conv2DLayer(skip3, 64, 5, strides=1, padding='same', block_id=12)
    deconv4 = Transpose_Conv2D(conv9, 64, 3, strides=2, padding='same', block_id=13)
    skip3 = layers.concatenate([deconv4, conv1])
```

This is a TensorFlow/Keras implementation of an autoencoder using convolutional layers. The autoencoder takes an image as input and compresses it into a lower-dimensional representation, then reconstructs the original image from this representation. The encoder part of the autoencoder is defined by the convolutional layers conv1-conv7, which gradually reduce the spatial dimensions of the input image while increasing the number of channels (i.e., feature maps). The decoder part of the autoencoder is defined by the transpose convolutional layers deconv0-deconv4, which gradually increase the spatial dimensions of the compressed representation while decreasing the number of channels. The skip connections (skip0-skip3) between the encoder and decoder layers allow the decoder to access information from the corresponding encoder layer at the same spatial resolution. This can help the autoencoder to preserve more fine-grained details in the reconstructed image. The final convolutional layer (conv10) has 3 channels and uses the sigmoid activation function to ensure that the reconstructed image pixel values are in the range [0, 1]. The kernel_initializer parameter sets the weight initialization method to orthogonal, which can help prevent the vanishing gradient problem during training.

`

Overall, this autoencoder architecture appears to be well-designed for reconstructing images with a moderate level of detail, and the use of skip connections and weight initialization can help to improve training stability and reconstruction quality.

Fig 7.2: Proposed model

```
def m1(input_shape):
    inputs = layers.Input(shape=input_shape)
    conv1 = Conv2DLayer(inputs, 64, 3, strides=1, padding='same', block_id=1)
    conv2 = Conv2DLayer(conv1, 64, 3, strides=2, padding='same', block_id=2)
    conv3 = Conv2DLayer(conv2, 128, 5, strides=2, padding='same', block_id=3)
    conv4 = Conv2DLayer(conv3, 128, 3, strides=1, padding='same', block_id=4)
    conv5 = Conv2DLayer(conv4, 256, 5, strides=2, padding='same', block_id=5)
    conv6 = Conv2DLayer(conv5, 512, 3, strides=2, padding='same', block_id=6)
    conv7 = Conv2DLayer(conv6, 1024, 3, strides=2, padding='same', block_id=60)
    deconv0 = Transpose_Conv2D(conv7, 1024, 3, strides=2, padding='same', block_id=7)
    skip0 = layers.concatenate([deconv0, conv6], name='skip0')
    conv77 = Conv2DLayer(skip0, 512, 3, strides=1, padding='same', block_id=80)
    deconv1 = Transpose_Conv2D(conv77, 256, 3, strides=2, padding='same', block_id=90)

    skip1 = layers.concatenate([deconv1, conv5], name='skip1')
    conv7 = Conv2DLayer(skip1, 256, 3, strides=1, padding='same', block_id=8)
    deconv2 = Transpose_Conv2D(conv7, 128, 3, strides=2, padding='same', block_id=9)
    skip2 = layers.concatenate([deconv2, conv3], name='skip2')
    conv8 = Conv2DLayer(skip2, 128, 5, strides=1, padding='same', block_id=10)
    deconv3 = Transpose_Conv2D(conv8, 64, 3, strides=2, padding='same', block_id=11)
    skip3 = layers.concatenate([deconv3, conv2], name='skip3')
    conv9 = Conv2DLayer(skip3, 64, 5, strides=1, padding='same', block_id=12)
    deconv4 = Transpose_Conv2D(conv9, 64, 3, strides=2, padding='same', block_id=13)
    skip3 = layers.concatenate([deconv4, conv1])
```

```
    skip3 = layers.concatenate([deconv3, conv2], name='skip3')
    conv9 = Conv2DLayer(skip3, 64, 5, strides=1, padding='same', block_id=12)
    deconv4 = Transpose_Conv2D(conv9, 64, 3, strides=2, padding='same', block_id=13)
    skip3 = layers.concatenate([deconv4, conv1])
    conv10 = layers.Conv2D(3, 3, strides=1, padding='same', activation='sigmoid',kernel_initializer=orthogonal(), name='final_conv')(skip3)
    c3=Conv2D(filters=16, kernel_size=(5,5), strides=1, activation='tanh')(conv10)
    c4=AveragePooling2D(pool_size=(2,2), strides=2)(c3)
    c10=keras.layers.Flatten()(c4)
    c11=keras.layers.Dense(units=84, activation='tanh')(c10)
    c12=keras.layers.Dense(units=43, activation='softmax')(c11)
    return models.Model(inputs=inputs, outputs=c12)
```

```
model1 = m1((*input_shape, 3))
model_opt = Adam(lr=0.002)
model1.compile(optimizer=model_opt, loss='mse', metrics=['accuracy'])
```

```
/usr/local/lib/python3.8/dist-packages/keras/initializers/initializers_v2.py:120: UserWarning: The initializer Orthogonal is unseeded and being c
  warnings.warn(
WARNING:absl:`lr` is deprecated, please use `learning_rate` instead, or use the legacy optimizer, e.g.,tf.keras.optimizers.legacy.Adam.
```

```
epochs = 25
history2 = model1.fit(X_train, y_train, batch_size=32, epochs=epochs,validation_data=(X_valid, y_valid))
```

`

The code define a convolutional neural network (CNN) architecture that takes an input image and outputs a probability distribution over 43 possible classes (i.e., it is a multi-class classification model). The architecture consists of several convolutional layers followed by transposed convolutional layers (which perform upsampling). The convolutional layers extract features from the input image, while the transposed convolutional layers help to recover spatial information lost during the downsampling process. The skip connections between the corresponding convolutional and transposed convolutional layers help to propagate information from lower-level features to higher-level features. After the final convolutional layer, the code applies a fully connected network to perform classification. It first flattens the output of the final convolutional layer, then applies two dense (fully connected) layers, and finally applies a softmax activation function to produce the probability distribution over the 43 classes. It is worth noting that some of the layer types and arguments used in the code (e.g., **Conv2DLayer** and **Transpose_Conv2D**) are not part of the standard Keras library. Thus, it is possible that this code is using a custom implementation of these layers.

`

RESULTS

Table 1.2: Accuracy of image classification model

Model	Accuracy
L-net model	0.98

Fig 8.1: Graph of the model

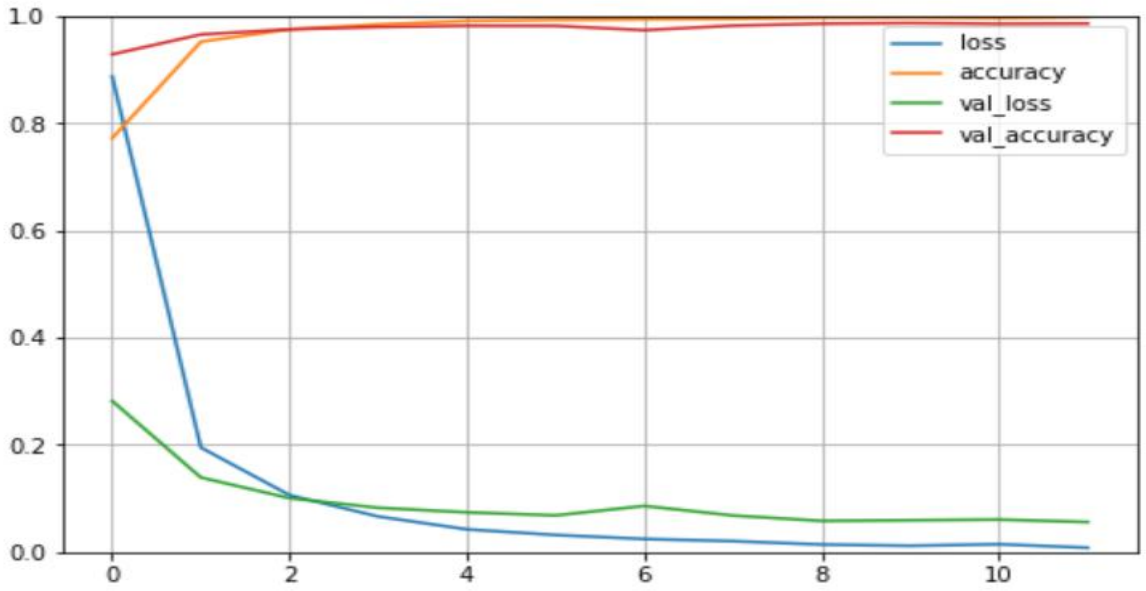


Table 2.1: Attacking models

Technique	Confidence
FGSM	0.002
C&W	50

Fig 8.2:Results

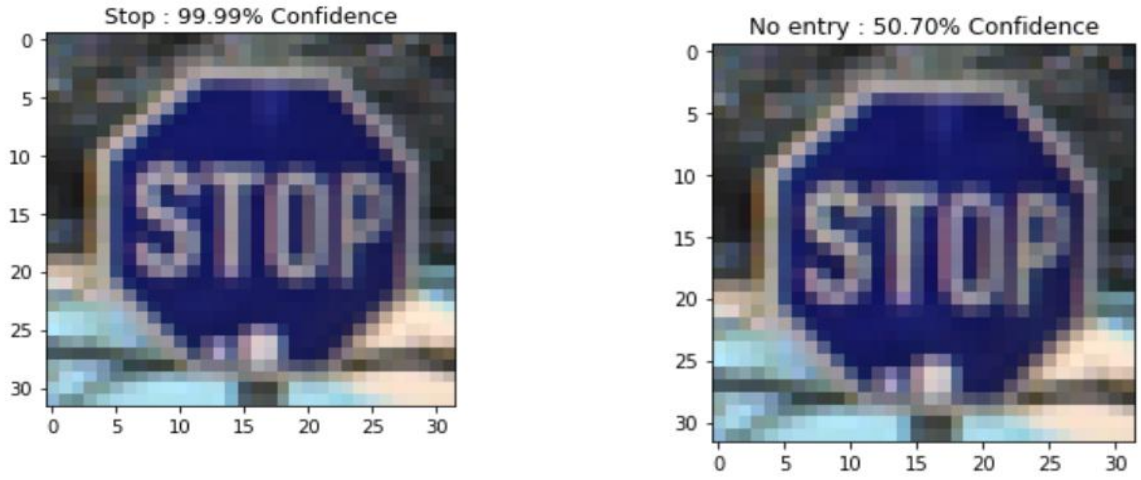


Table 2.2: Accuracy of defending models

Technique	Accuracy of the model
Proposed model	0.97
Adversarial training	0.98
Auto-encoder	0.95

`

CONCLUSION

DNN algorithms are employed in many fields but the fact that they are vulnerable to adversarial attacks lead to a challenging problem. So, Adversarial attacks in the real world have drawn a lot of attention lately.In this paper some of the adversarial attack methods such as FGSM and C&W attacks which reduced the accuracy of the model to 2% and reduced the confidence levels. Also, some of the defense strategies are applied to increase the accuracy of the model such as adversarial training and auto-encoder which revived the accuracy of the image classification model.

REFERENCES

[1] Zhang, J., Qian, W., Nie, R., Cao, J., & Xu, D. (2022). Generate adversarial examples by adaptive moment iterative fast gradient sign method. Applied Intelligence, 1-14.

[2] Ahmed, U., Lin, J. C. W., & Srivastava, G. (2022). Mitigating adversarial evasion attacks by deep active learning for medical image classification. Multimedia Tools and Applications, 81(29), 41899-41910.

[3] Wu, F., Xiao, L., Yang, W., & Zhu, J. (2020). Defense against adversarial attacks in traffic sign images identification based on 5G. EURASIP Journal on Wireless Communications and Networking, 2020(1), 1-15.

[4] Ren, H., Huang, T., & Yan, H. (2021). Adversarial examples: attacks and defenses in the physical world. International Journal of Machine Learning and Cybernetics, 12(11), 3325-3336.

[5] Zhang, Y., Li, H., Zheng, Y., Yao, S., & Jiang, J. (2021). Enhanced DNNs for malware classification with GAN-based adversarial training. Journal of Computer Virology and Hacking Techniques, 17(2), 153-163.

[6] Jia, X., Zhang, Y., Wu, B., Wang, J., & Cao, X. (2022). Boosting fast adversarial training with learnable adversarial initialization. IEEE Transactions on Image Processing, 31, 4417-4430.

[7] Zhang, H., & Sakurai, K. (2021). Conditional Generative Adversarial Network-Based Image Denoising for Defending Against Adversarial Attack. IEEE Access, 9, 169031-169043.

[8] Wang, D., Li, C., Wen, S., Nepal, S., & Xiang, Y. (2020). Defending against adversarial attack towards deep neural networks via collaborative multi-task training. IEEE Transactions on Dependable and Secure Computing.

[9] Khamaiseh, S. Y., Bagagem, D., Al-Alaj, A., Mancino, M., & Alomari, H. W. (2022). Adversarial Deep Learning: A Survey on Adversarial Attacks and Defense Mechanisms on Image Classification. IEEE Access.

[10] Li, Z., Feng, C., Zheng, J., Wu, M., & Yu, H. (2020). Towards adversarial robustness via feature matching. IEEE Access, 8, 88594-88603.

[11] Ren, K., Zheng, T., Qin, Z., & Liu, X. (2020). Adversarial attacks and defenses in deep learning. Engineering, 6(3), 346-360.

[12] Hang, J., Han, K., Chen, H., & Li, Y. (2020). Ensemble adversarial black-box attacks against deep learning systems. Pattern Recognition, 101, 107184.
[13] Ghosh, A., Mullick, S. S., Datta, S., Das, S., Das, A. K., & Mallipeddi, R. (2022). A black-box adversarial attack strategy with adjustable sparsity and generalizability for deep image classifiers. Pattern Recognition, 122, 108279.
[14] Wei, X., Guo, Y., & Li, B. (2021). Black-box adversarial attacks by manipulating image attributes. Information Sciences, 550, 285-296.
[15] Lin, J., Njilla, L. L., & Xiong, K. (2022). Secure machine learning against adversarial samples at test time. EURASIP Journal on Information Security, 2022(1), 1-15.
[16] Yadav, A., Upadhyay, A., & Sharanya, S. (2022). An integrated Auto Encoder-Block Switching defense approach to prevent adversarial attacks. arXiv preprint arXiv:2203.10930.
[17] Hu, S., Nalisnick, E., & Welling, M. (2022). Adversarial Defense via Image Denoising with Chaotic Encryption. arXiv preprint arXiv:2203.10290.
[18] Kherchouche, A., Fezza, S. A., & Hamidouche, W. (2022). Detect and defense against adversarial examples in deep learning using natural scene statistics and adaptive denoising. Neural Computing and Applications, 34(24), 21567-21582.
[19] Ma, L., & Liang, L. (2022). Adaptive Adversarial Training to Improve Adversarial Robustness of DNNs for Medical Image Segmentation and Detection. arXiv preprint arXiv:2206.01736.
[20] Xu, H., Ma, Y., Liu, H. C., Deb, D., Liu, H., Tang, J. L., & Jain, A. K. (2020). Adversarial attacks and defenses in images, graphs and text: A review. International Journal of Automation and Computing, 17(2), 151-178.
[21] Chai, X., Wei, T., Chen, Z., He, X., Gan, Z., & Wu, X. (2022). LDN-RC: a lightweight denoising network with residual connection to improve adversarial robustness. *Applied Intelligence*, 1-16.
[22] Sun, Q., Rao, A. A., Yao, X., Yu, B., & Hu, S. (2020, November). Counteracting adversarial attacks in autonomous driving. In *Proceedings of the 39th International Conference on Computer-Aided Design* (pp. 1-7).
[23] Qiu, H., Custode, L. L., & Iacca, G. (2021, July). Black-box adversarial attacks using evolution strategies. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (pp. 1827-1833).
[24] Zügner, D., Borchert, O., Akbarnejad, A., & Günnemann, S. (2020). Adversarial attacks on graph neural networks: Perturbations and their patterns. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, *14*(5), 1-31.
[25] Zhao, G., Zhang, M., Liu, J., & Wen, J. R. (2019). Unsupervised adversarial attacks on deep feature-based retrieval with GAN. *arXiv preprint arXiv:1907.05793*.
[26] Zhang, C., Benz, P., Lin, C., Karjauv, A., Wu, J., & Kweon, I. S. (2021). A survey on universal adversarial attack. *arXiv preprint arXiv:2103.01498*.
[27] Benz, P., Zhang, C., Karjauv, A., & Kweon, I. S. (2021, July). Universal adversarial training with class-wise perturbations. In *2021 IEEE International Conference on Multimedia and Expo (ICME)* (pp. 1-6). IEEE.
[28] Huang, Z., & Zhang, T. (2019). Black-box adversarial attack with transferable model-based embedding. *arXiv preprint arXiv:1911.07140*.
[29] Deng, Y., Zheng, X., Zhang, T., Chen, C., Lou, G., & Kim, M. (2020, March). An analysis of adversarial attacks and defenses on autonomous driving models. In *2020 IEEE international conference on pervasive computing and communications (PerCom)* (pp. 1-10). IEEE.