Abstract

By describing some of the key contributions made to the field of adversarial attacks in deep learning, this abstract provides an overview of the subject of safety-critical environments that are at a significant risk due to the vulnerability of adversarial attacks. In addition, I will show that deep learning poses certain difficulties, like the vulnerability to adversarial examples becoming a major risk for technological areas such as computer vision and the identification of faces. In order to address these vulnerabilities, I will go over the methods that have been used in recent findings on adversarial example methods for generating attacks and proposed groupings of these methods. The observable results include a series of techniques for producing adversarial examples, and applications for these examples will be investigated by providing an explanation of how adversarial attacks operate. By analysing the scientific papers made by the computer vision community in adversarial attacks on deep learning, I will give a synopsis of this project's motivation. My background research indicates that current understanding on deep learning reveals that the technology is susceptible to various adversarial attacks, including adversary knowledge, and adversarial falsification

INDEX TERMS Adversarial learning (AL), Deep Learning (DL) black-box attack, white-box attack, Fast Gradient Sign Attack (FGSM), machine learning (ML).

Table of Contents

1 Project definition

1.1 Introduction

The subject of Deep Learning (DL) refers to neural networks that learn from vast amounts of data in an effort to replicate the patterns of the human brain [2]. Adversarial attacks are techniques used to mislead models with false information such as when the model is supplied with incorrect data, or it has been deliberately built up to be wrong [3]. One application is the causation of a self-driving car to steer toward the opposite side of traffic [3]. The numerous characteristics of hostility, whilst they pose a serious threat to Deep Learning (DL), are also underlined by these implications. For security critical technology, the implications of this are very important [1]. I'm going to use an adversarial technique called Fast Gradient Sign Attack (FGSM) for this project, showing that deep neural network predictions can be manipulated with very low input disturbances [1].

As a result of the critical nature of adversarial attacks, e.g. in terms of image classification, object recognition or detection [4], this project benefits the research community when it is effectively completed because it contributes to security for deep learning applications [1]. The Deep Learning (DL) Research Community will benefit from the project's ability to show algorithms that can be further adapted for core attack methods [1]. The fact that there will be a strong influence in the broader community can be demonstrated through future my research aimed at providing more exact definitions of technically defined terms, such as by providing an explanation of FGSM. [1] [4].

Symbols and Elements	Meaning
1. $\mathbf{F}$	1. A representation of the deep learning model.
2. $\boldsymbol{x}$	2. Unchanged and original “input data”.
3. $\boldsymbol{x\ell}$	**3.** $\boldsymbol{x}$ will generate $\boldsymbol{x\ell}$, which will be an adversarial image**.**
4. $\boldsymbol{\ell}$	**4.** The result of $\boldsymbol{x}$ is given by $\boldsymbol{\ell}$**.**
5. $\boldsymbol{x}$’	5. Adversarial example’s modified data.
6. $\boldsymbol{\ell}$’	6. The output of $\boldsymbol{x}$**’** is given by $\boldsymbol{\ell}$**’.**
7. $\lVert \cdot \rVert$	7. The separation between two 2 data sets for the FGSM.

Table 1: Symbols, Elements and their definitions given in this paper [4]

1.2 Motivation

Through a series of AI services that improve automation and perform services, I plan to observe the significance of Deep Learning [2], along with analysing the risks that are involved through various adversarial attacks. One of the main advantages of deep learning is that it can automatically extract features from a set of data, eliminating the need for human feature design [17]. In the ImageNet challenge in 2022, for instance, a Deep Learning model named ResNEt-152 obtained a failure rate of 3.56%, which is far lower than the human mistake rate of 5.1% [17]. From countless internet pages, deep learning algorithms may collect data to create visual material like images. In one example, LayoutLM collected substantial data from web pages at 96.3% accuracy [17]. Therefore, one may argue that DL has paved the way for research into technologies such as debit card fraud detection and self-driving cars, which are pivotal to everyday life [2]. Kurakin et al. [18] presented attacks on self-driving cars in which the attacker altered traffic signs causing confusion for the learning system [18]. This has led to the increased need for verification techniques that search through a list of possible adversarial attacks, that are pinpointed from a changed input [6]. Due to the severe consequences of adversarial attacks, I will be taking this project as an opportunity to explore one adversarial attack method, named the Fast Gradient Sign method (FGSM), which can create an adversarial example in a short time [6]. This is a key method that will help to improve the safety of Deep Learning (DL) technologies.

1.3 Aims and objectives

1. **Aim:**

To provide the various deep learning (DL) and adversarial learning (AL) techniques.

Objectives:

1. I will verify this by defining what DL and AL is.
2. I will Provide real-life examples in order to show how DL and AL influences various applications.
3. A description of what the various DL and AL techniques are, will be shown by how each techniques' results differ from each other. Such as in terms of Adversarial Falsification, I will delve into the topics of False positives and False negative attacks.
4. There will be a mathematical illustration that will be used to explain how to implement each technique.

2. **Aim:**

To implement one adversarial technique

Objectives:

1. I will build an adversarial example of Fast Gradient Sign Attack (FGSA), which can be accessed through a GitHub tutorial, using python and PyTorch. PyTorch installation will be done using PyCharm's integrated development environment (IDE).

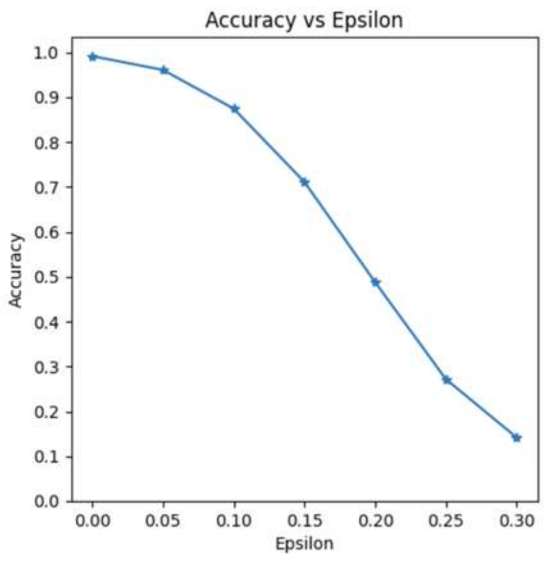


Figure 1: A simple graphical drawing of the building of an FGSM [7]

2. There will be an understanding of the code that was provided in the specified GitHub tutorial [8].

3. A design of an Activity diagram will be created as part of my software Development life cycle (SDLC), which will illustrate my understanding of this adversarial example.

1.4 Milestones and Deliverables

Milestones

1. I must be able to describe and illustrate what the various Adversarial Learning (AL) and Deep Learning (DL) techniques are.

2. I should be able to successfully run an adversarial attack using the python coding language.

3. In order to show that I understand what the code does, I will design an Activity Diagram.

1.5 Implementation of an adversarial attack

I plan to be generating code that is accessible through a GitHub tutorial, in order to generate figure 1[8]. The programming language of Python will be installed with PyTorch, within the IDE of PyCharm.

The techniques that I will be using will be under the umbrella of Deep Learning and the generation of a particular method of an adversarial example. This method type is the Fast Gradient Sign Attack (FGSA). Goodfellow [12] has conducted a study summarising that just a difference of “0.25” will make the “classifier to have an error rate of 99.9%”. It is considered a considerable advancement in addressing the existence of an adversarial attack because they are viewed as “simple, cheap algorithms” [12].

The FGSM method is especially useful because of its use of varying gradients, that are utilised against neural networks common method of attack [12]. The issue is that, gradients will have an increased amount of data lost by the attack, and it will be in relation to a changed degree of lost data, therefore ensuring a greater loss [12]. This would mean that the amended image would be completely misinterpreted, such as a digital picture of a sheep being shown as a dog by a given network, with more than a 99% rating of assurance [12].

FGSM does not concentrate on how to achieve high percentages of causing confusion, but instead, it aims to investigate the "efficiency of perturbation" [15]. The FGSM are important attacks that usually come in the form of a white-box attack [15]. This is as FGSM will deceive a model through increasing the gradient [15]. To put it differently, FGSM establishes the slope of a loss function in regard to the input image, and then generates a new image that increases the loss employing the gradients. Consequently, a neural network is going to forecast something incorrectly [13].

2 Literature Review

2.1 Significance of Adversarial attacks in deep learning

In a lot computer vision tasks, Deep Neural Networks (DNNs) perform effectively, on occasion surpassing people [19]. As a consequence, they have been integrated increasingly into robot and vehicle control systems [19]. However, recent research suggests that adversarial perturbations have the capacity to cause disruption with DNNs, possibly leading to undesirable behaviour in these kinds of systems [19].

Particularly in classification, Deep Learning (DL) has demonstrated impressive performance across a range of machine learning tasks [20]. Deep Neural Networks (DNNs) can classify unknown samples with high accuracy and efficiently learn accurate models from large amounts of training data [20]. As a result, DNNs are frequently utilised, even in environments where security is a concern [20]. However, by creating adversarial samples, adversaries can manipulate DNNs to produce specific outputs, as shown by recent research in the machine learning and security communities [20]. These samples are produced by carefully crafting perturbations to be added to valid inputs; the attacker's chosen outputs, like misclassification, are the result [20]. In order to minimise disturbances, adversarial sample crafting algorithms are challenging to discern from valid inputs [20]. These attacks happen after training and don't necessitate changing the training procedure [20].

2.2 Adversarial examples methods

I will be delving into two components of the threat model known as "adversary knowledge", and "adversarial falsification" [4].

Adversarial Falsification

False positive:

A “Type 1”[4] Error constitutes a “False positive attack”, which causes a negative sample to be incorrectly identified as positive. [4]. This error works by creating a completely different image [14]. In another similar case, an adversarial “False positive attack” [4] picture may not be recognized by humans as correct, but a Deep Neural Network (DDN) will identify it as almost certainly accurate[4]. The severity of this attack is that important and safe information may be unnecessarily removed, because of a false warning of an attack happening. [16]. Furthermore, the capabilities of a service will be drastically hindered if there are too many false positives [16].

False Negative:

A “False negative attack” is regarded as a “Type 2”[4] Error, because an illustration has been understood to be a negative, different image, when it is in fact a positive and accurate illustration [4]. This is an issue because using “malware detection” [4], a DNN design may not be able to successfully determine malware, due to this Type 2 Error. An evasion attack, that appears correct to a person is in fact incorrectly identified by a machine learning tool [14]. Thus, a deep neural network (DNN) views the adversarial picture as incorrect, when in fact it is accurate [4]. The issue is that a series of intrusion systems may go unnoticed, under a false negative attack [16].

Adversary’s knowledge

White-box:

White-box attacks believe that an adversary has detailed information, regarding DNN models [4], “training data, model architectures, hyperparameters, layers, activation functions, model weights”[4] are all used by white-box attacks. Furthermore, the use of gradients is useful for generating various types of adversarial illustrations [4]. Deep neural networks (DNNs) do not require a complete framework, distinguishing itself from machine learning (ML)[4]. For a white-box attack to be successful, there must be thorough comprehension of the intended vulnerable model [16]. Therefore, important details about the vulnerable model must be visible and clear to the attacker in order to perform a white-box attack [16]. A famously used method of a white-box attack is the “Jacobian-based Saliency Map Attack (JSMA)” [16]. This is as the attacker has all of the necessary details in order to make advancements that will calculate a saliency map [16].

Black-box:

Black-box attacks believe that the adversary has limited information to get to a DNN model[4]. This can be exemplified by only the adversary having information about what the DNN model entails, such as knowing the percentage of the level of confidence of an illustration[4]. Black-box attacks usually occur to services such as Google Cloud and other machine learning (ML) services on the internet[4]. Despite white-box attacks being the most common from of attack, black-box services are still at risk [4]. For a black-box attack to be successful, an intruder just needs entry into a service, to enquire about the intended outcome [16]. For example, a false model could be created in order to mimic the vulnerable model’s inputs and results [16]. Furthermore, if the mimicking is successful, white-box attacks can be additionally used to get full control over the vulnerable model [16].

3 Methodology

3.1 Fast Gradient Sign Attack (FGSM)

Including adversarial images in classifier training data improves its resistance to adversarial examples, as was first noted in the research paper published by Szegedy [21]. In later research, the idea of adversarial training was inspired by this observation [21]. However, it is computationally prohibitive to solve for a large number of images [21]. Rather than maximising fooling rates, the Fast Gradient Sign Method (FGSM) [30] is a gradient-based one-step method that efficiently computes norm-bounded perturbations to achieve perturbation computation efficiency. Using FGSM, Goodfellow et al. [21] supported their linearity hypothesis, which postulated that the linear behaviour of contemporary neural networks in high-dimensional spaces [21 contributes to their susceptibility to adversarial perturbations. This theory runs counter to the notion that adversarial vulnerability develops in contemporary, complex networks [21].

FGSM is a well-known attack in the literature that serves as the foundation for a number of follow-up attacks, particularly in white-box setups [21]. For instance, to initiate the attack, the Fast Gradient Value Method (FGVM) [21] eliminates the sign function, whereas Miyato et al. [21] normalise the gradient using its L2-norm. Normalisation with the L1-norm is analysed by Kurakin et al. [21], who also extend FGSM to its iterative variant, Iterative FGSM (I-FGSM). Dong et al. [21] then improve it with momentum, calling it Momentum Iterative (MI-)FGSM. Building directly on FGSM, Diverse Input I-FGSM (DI2-FGSM) [21] diversifies input through image transformations in each iteration to improve transferability in black-box setups. By adding momentum, M-DI2-FGSM expands on DI2-FGSM [21].

3.2 CIFAR-100 dataset

A large variety of general object images, divided into 100 classes with 600 images apiece, are included in the CIFAR-100 dataset [22]. A total of 60,000 images are used, 500 for training and 100 for testing per class [22]. Twenty super classes with labels for "fine" and "coarse" are created from the classes [22]. Items such as beds, bicycles, buses, chairs, couches, motorbikes, streetcars, tables, trains, and wardrobes are among the categories that have been chosen for training and testing [22]. Our work focuses on training networks with broader categories—specifically, cars and household furniture—from each superclass[22]. Another dataset containing super classes and subclasses that represent significant concepts is provided by the hierarchically arranged ImageNet dataset, which is based on WordNet [22].

4 Results

4.1 Convolution

Convolutional layers are usually stacked first in CNNs, beginning with LeNet-5, and then optionally normalisation and pooling, and finally fully-connected layers [23]. This structure's variations perform well on image classification tasks such as MNIST, CIFAR, and ImageNet.

Increasing the number and size of layers while utilising dropout to stop overfitting is a recent trend [23]. Architectures with max-pooling layers perform well in tasks like object detection and localization, despite worries about the loss of spatial information [23]. Fixed Gabor filters of different sizes are used by Serre et al. [23], who were inspired by models of primate visual cortex. Our method is similar, but it repeats the process for depth using learned filters from Inception [23]. The Lin et al. Network-in-Network technique [23]. adds 1 × 1 convolutional layers to improve representation, mainly for dimension reduction to scale networks without sacrificing performance [23].

4.2 FGSM detailed

The Fast Gradient Sign Method (FGSM) was first introduced by Goodfellow et al. [23] to compute input gradients with respect to the cost function. Three variations of FGSM were developed by Kurakin et al.[23]: the One-step Target Class, Basic Iterative, and Iterative Least-likely Class methods. It was shown to be an attack technique that could change an image by just one pixel [23].

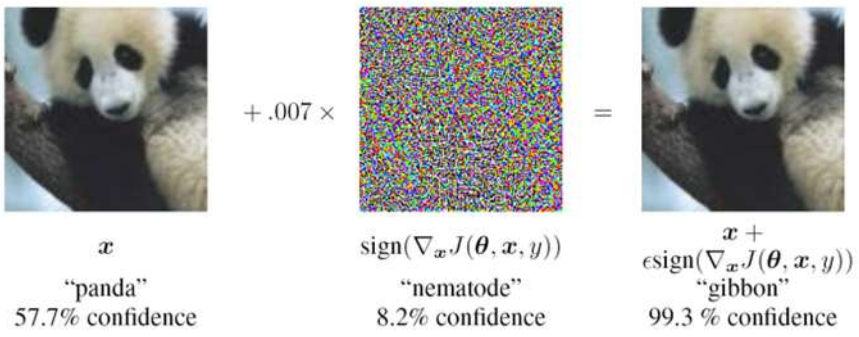

$+ .007 \times$
$=$
$\boldsymbol{x}$
"panda"
57.7% confidence
$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"nematode"
8.2% confidence
$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"gibbon"
99.3 % confidence

Figure 2: Fast Gradient Sign Technique has been used on an image of a Panda. The left image shows the original picture, the middle is the disturbance created by the adversarial attack, the right image shows the Gibbon identification. [4]

$$\begin{aligned} \min_{x'} \quad & \|x' - x\| \\ s.t. \quad & f(x') = l', \\ & f(x) = l, \\ & l \neq l', \\ & x' \in [0,1], \end{aligned}$$

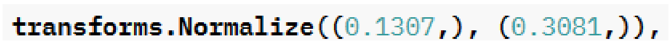
Figure 3: This shows the efficiency of the disruption being successfully misclassified. [4]

Picture identification using adversarial examples [4]:

For instance, someone may use an ‘image classifier”[4] to find out what the possible class type is, by receiving the name of the image[4]. The classifier will incorrectly identify an adversarial image, and therefore also mislead the person using the “image classifier”[4], because of a small change in an illustration[4], such as by changing gradients. The resulting name of the image, will be incorrect when sent to the person using the “image classifier”[4].

The symbol f represents the deep learning model, x symbolizes the original "input data", which will generate $x\ell$, that will be an adversarial image, typically a problem with improvement in a box [4]. The result of x is given by ℓ and the output of x' is given by ℓ' and , $\| \cdot \|$ refer to the separation between two 2 data sets [4]. If we allow $\eta = x' - \mathbf{x}$ as the additional perturbation on x [4]. The purpose is to downsize the perturbation, as the input data is misclassified as incorrect [4].

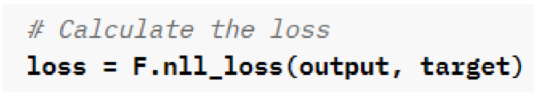
4.3 Experimental set up



```
transforms.Normalize((0.1307,), (0.3081,)),
```

Figure 6: This is an illustration of the normalization function in the FGSM. [25]

By dividing by the standard deviation and subtracting the mean, this function normalises a tensor image [27]. It processes each channel independently while operating on input tensors [27]. For every channel, the normalisation formula is (input: mean, std) [27].The mean, or sequence comprising the mean values for every channel, is one of the parameters[27].Standard deviations for each channel are listed in a sequence called std, which goes with it[27].



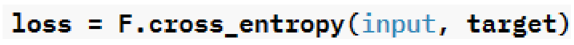
```
# Calculate the loss
loss = F.nll_loss(output, target)
```

$$
\begin{aligned}
\ell(x_i) &= -\log(\hat{y}(x_i)) \\
&= -\log\left(\frac{\exp\{f_{y_i}(x_i)\}}{\Sigma_j^k \exp\{f_j(x_i)\}}\right) \\
&= -\left[\log\left(\exp\{f_{y_i}(x_i)\}\right) - \log\left(\Sigma_j^k \exp\{f_j(x_i)\}\right)\right] \\
&= -f_{y_i}(x_i) + \log\left(\Sigma_j^k \exp\{f_j(x_i)\}\right) \\
&= -[Wx_i]_{y_i} + \log\left(\Sigma_j^k \exp\left\{[Wx_i]_j\right\}\right)
\end{aligned}
$$

Figure 6: This is an illustration of a loss function in the FGSM, loss function is The Negative Log Likelihood Loss. (NLL) function. [25][31], equation credited to [32].

$\hat{y}$ Stands for the expected chance [32]. $k \in \mathbb{R}$shows the count of conceivable classes [32]. $x_i \in \mathbb{R}^d$ represents the vector for the for the i example and d means dimension [32]. $W \in \mathbb{R}^{k \times d}$ where W is an adaptable matrix of weights covering $k \times d$ [32]. $f(x_i) = Wx_i \in \mathbb{R}^k$ The function $f(x_i)$ which may be a neural network, shows the result for the i example which will be developed into Wx_i [32]. $f_j(x_i) = [Wx_i]_j \in \mathbb{R}$ The new function of $f_j(x_i)$ is the result for the j category sample, that was determined by $[Wx_i]$ [32]. $f_{y_i}(x_i) = [Wx_i]_{y_i} \in \mathbb{R}$ In this equation, $f_{y_i}(x_i)$ is the result for the classification in the i example which will now be expressed as $[Wx_i]_{y_i}$ [32]. A larger result given by $f_{y_i}(x_i)$ leads to a diminished result of loss [32]. This expression $\Sigma_j^k \exp\{f_j(x_i)\}$ is the denominator for the entire function [32].

A loss function measures how well a neural network learns from training data by comparing the predicted and target outputs [28]. The goal of training is to reduce this difference as much as possible [28]. By identifying the ideal weights (w) and biases (b) that minimise the average loss function J, hyperparameters are adjusted to lower the average loss [28]. Regression and classification are the two categories of loss functions [28]. Regression neural networks predict output values based on input by using regression loss functions [28]. In classification neural networks, Classification Loss Functions are used to calculate the probability that input will fall into predefined categories [28].



```
loss = F.cross_entropy(input, target)
```

*Figure 7: This is an illustration of a loss function in the FGSM, loss f*unction is the cross entropy loss. [29]

The cross entropy loss function [28] is what I've chosen to use. It accomplishes the same goal as the NLLLoss function, but the implementation is where the main differences lie: While NLLLoss does not, CrossEntropyLoss implicitly applies a “SoftMax” activation followed by a log transformation [28].

5 Discussion

5.1 Interpretation of the results

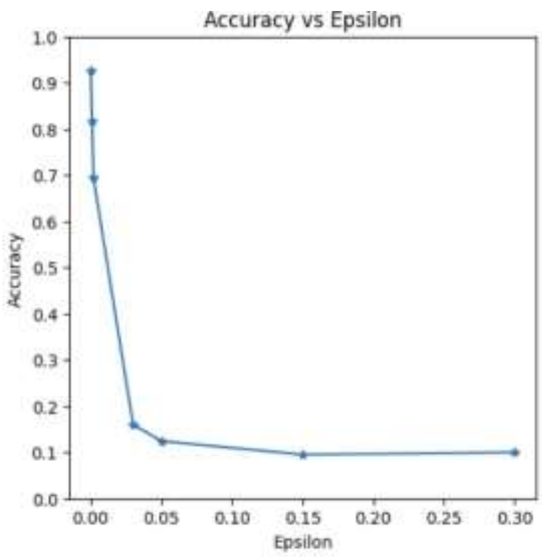


Figure 8: This is an illustration of The Cifar-100 data set that I implemented using the PyTorch tutorial. [25]

The purpose of this study was to draw attention to the difficulties that come with deep learning, especially its susceptibility to adversarial examples, which presents serious dangers in a variety of technology fields, including computer vision and facial recognition.
I had used the FGSM attack as one adversarial technique to look into these vulnerabilities more.

The primary outcome of this study indicates that, in the CIFAR-100 Dataset, model accuracy declines with increasing epsilon values, reflecting trends noted by MNIST and suggesting vulnerability. The CIFAR-100 dataset's vulnerability to perturbations is further demonstrated by the successful generation of adversarial examples.

Overall, this work's most important finding is that the FGSM attack method efficiently produces adversarial examples, underscoring the urgent need for neural network model robustness improvements.

In order to graphically depict the model's vulnerability, I worked on creating adversarial examples and plotting accuracy against epsilon.

To sum up, this research's experiments highlight how neural networks can be attacked by adversaries, underscoring how crucial robustness is to model building and training.

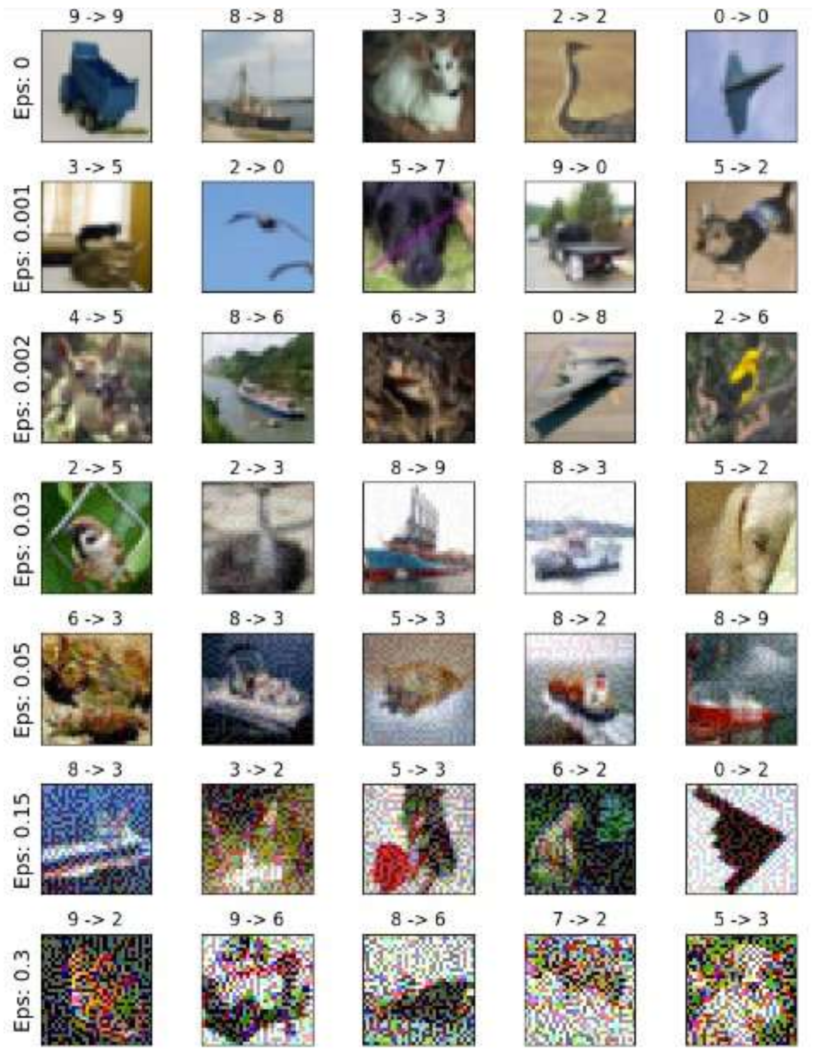


Figure 9: This is an illustration of The Cifar-100 data set that I implemented using the PyTorch tutorial. [25]

5.2 Implications of findings for model robustness and training

The struggle between attacks and defences for adversarial examples occurs when defensive strategies intended to thwart current attacks frequently become vulnerable to new ones, and vice versa [26]. Certain defences that were once successful against particular attacks become ineffective when those attacks are slightly altered [26]. Therefore, it is essential to evaluate how robust Deep Neural Networks (DNNs) are [26]. For example, [26] defined robustness upper bounds for both linear and quadratic classifiers. Nevertheless, there are still problems with assessing DNN robustness, especially in terms of methodology [26].

It is not sufficient to defend against known attacks in safety-critical environments, which is where many DNNs are designed for [26]. To evaluate robustness, particularly against zero-day attacks, and to determine the level of confidence in model predictions and real-world reliability, a methodology is required [26]. This has been the subject of preliminary research by Carlini et al. [26], Katz et al. [26], Bastani et al. [26], and Huang et al. [26], which have acknowledged the significance of DNN performance as well as confidentiality and privacy issues [26].

6 Conclusion

6.1 Summary of key findings

The code illustrates how resistant the model is to adversarial attacks by visualising the accuracy of the model against a range of epsilon values [30]. It also shows various instances of adversarial images produced by the FGSM attack [30]. The findings show that the accuracy of the model declines with increasing epsilon, indicating the efficacy of the FGSM attack in undermining model performance [30]. The graphic emphasises how adversarial perturbations affect image classification. It demonstrates how even minute adjustments to the input images can cause the model to misclassify images. The model continues to function at a certain level even though accuracy decreases with higher epsilon values, suggesting some degree of resilience to adversarial attacks.

6.2 Importance of the research

Examine how deep learning is used in driverless cars [30]. DNNs are used by these systems to recognise other cars and road signs[30]. The repercussions could be severe if minor visual changes, like a vehicle's body being moved, lead DNNs to incorrectly categorise the vehicle[30]. The car might not stop or respond correctly, which could result in crashes with disastrous consequences[30]. Such vulnerabilities could be used by adversaries to their advantage, as is the case with non-DL classification systems in use today[30]. Thus, it is imperative to take adversarial sample threat into account when designing security-critical DNN systems[30].

6.3 Suggestions for future research directions

In summary, I have made a start into researching the analysis of various adversarial learning (AL) techniques. In the future, I will show one example of an adversarial attack using the Python programming language with PyTorch, in order to think about future work in the study of Deep Learning (DL). There will be opportunities to improve this project, such as studying the newly published research papers in the AL as discussed and this will be reflected in the future application that I will be coding a FGSM [1], that will arise. In addition, in order to better understand DL and AL, I will need to explore the possibility of future research in this project, such as the further analysis of the numerous attack methods [1].

7 Evaluation

7.1 Assessment of the project's contribution to the field

To sum up, this research's experiments highlight how neural networks can be attacked by adversaries, underscoring how crucial robustness is to model building and training.
These assaults have the potential to seriously jeopardise the security of DNN-supported systems, with grave repercussions [30]. For example, illegal content can get past filters,

autonomous cars can get into accidents, and biometric authentication systems can be tricked into allowing unwanted access [30].

8 Bibliography

8.1 References

[1] IEEE. (2022). Advances in Adversarial Attacks and Defenses in Computer Vision: A Survey. Ieeexplore.ieee.org. https://ieeexplore.ieee.org/document/9614158/.

[2] IBM. (2023). What is Deep Learning? Www.ibm.com. https://www.ibm.com/topics/deep-learning.

[3] Wiggers, K. (2021, May 29). Adversarial attacks in machine learning: What they are and how to stop them. VentureBeat. https://venturebeat.com/security/adversarial-attacks-in-machine-learning-what-they-are-and-how-to-stop-them/.

[4] Yuan, X., He, P., Zhu, Q., & Li, X. (2019). Adversarial Examples: Attacks and Defenses for Deep Learning. IEEE Transactions on Neural Networks and Learning Systems, 30(9), 2805–2824. https://doi.org/10.1109/tnnls.2018.2886017.

[5] Tae, J. (2021, January 5). Fast Gradient Sign Method. Jake Tae. https://jaketae.github.io/study/fgsm/.

[6] Carlini, N., Katz, G., Barrett, C., & Dill, D. (2020). Provably Minimally-Distorted Adversarial Examples. https://arxiv.org/pdf/1709.10207.pdf.

[7] GitHub. (n.d.-a). Adversarial Example Generation — PyTorch Tutorials 1.4.0 documentation. Pytorch.org. https://pytorch.org/tutorials/beginner/fgsm_tutorial.html.

[8] GitHub. (n.d.-b). tutorials/beginner_source/fgsm_tutorial.py at main · pytorch/tutorials. GitHub. Retrieved October 2023, from https://github.com/pytorch/tutorials/blob/main/beginner_source/fgsm_tutorial.py.

[9] TemplateLab. (2016, June 19). 41 Free Gantt Chart Templates (Excel, PowerPoint, Word) ᐅ TemplateLab. TemplateLab. https://templatelab.com/gantt-chart-templates/.

[10] Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A., Kohno, T., & Song, D. (2018). Robust Physical-World Attacks on Deep Learning Visual Classification. https://arxiv.org/pdf/1707.08945.pdf.

[11] Carlini, N., Katz, G., Barrett, C., & Dill, D. (2018). Provably Minimally-Distorted Adversarial Examples. https://arxiv.org/pdf/1709.10207.pdf.

[12] Goodfellow, I., Shlens, J., & Szegedy, C. (2015). Published as a conference paper at ICLR 2015 EXPLAINING AND HARNESSING ADVERSARIAL EXAMPLES. https://arxiv.org/pdf/1412.6572.pdf.

[12] Overview of Behavioral Diagrams in UML. (n.d.). Devcamp.com. Retrieved October 2023, from https://devcamp.com/trails/uml-foundations/campsites/uml-diagrams/guides/overview-behavioral-diagrams-uml.

[13] Rosebrock, A. (2021, March 1). Adversarial attacks with FGSM (Fast Gradient Sign Method). PyImageSearch. https://pyimagesearch.com/2021/03/01/adversarial-attacks-with-fgsm-fast-gradient-sign-method/.

[14] Polyakov, A. (2019, August 6). How to attack Machine Learning (Evasion, Poisoning, Inference, Trojans, Backdoors). Medium. https://towardsdatascience.com/how-to-attack-machine-learning-evasion-poisoning-inference-trojans-backdoors-a7cb5832595c.

[14] Tang, S., Huang, X., Chen, M., & Yang, J. (2018). Adversarial Attack Type I: Generating False Positives. https://arxiv.org/pdf/1809.00594v1.pdf.

[15] Akhtar, N., Mian, A., Kardan, N., & Shah, M. (2021). Advances in Adversarial Attacks and Defenses in Computer Vision: A Survey. IEEE Access, 9, 155161–155196. https://doi.org/10.1109/ACCESS.2021.3127960.

[16] Pauling, C., Gimson, M., Qaid, M., Kida, A., & Halak, B. (2022). A Tutorial on Adversarial Learning Attacks and Countermeasures. https://arxiv.org/pdf/2202.10377.pdf.

[17] TheKnowledgeAcademy. (n.d.). Importance Of Deep Learning: Explained. Www.theknowledgeacademy.com. Retrieved October 2023, from https://www.theknowledgeacademy.com/blog/importance-of-deep-learning/.

[18] Chakraborty, A., Alam, M., Dey, V., Chattopadhyay, A., & Mukhopadhyay, D. (2021). A survey on adversarial attacks and defences. CAAI Transactions on Intelligence Technology, 6(1), 25–45. https://doi.org/10.1049/cit2.12028.

[19] Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A., Kohno, T. and Song, D. (n.d.). *Robust Physical-World Attacks on Deep Learning Visual Classification*. [online] Available at: https://arxiv.org/pdf/1707.08945v5.

[20] Papernot, N., Mcdaniel, P., Wu, X., Jha, S. and Swami, A. (n.d.). *Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks*. [online] Available at: https://arxiv.org/pdf/1511.04508.

[21] Akhtar, N., Mian, A., Kardan, N. and Shah, M. (2021). Advances in Adversarial Attacks and Defenses in Computer Vision: A Survey. *IEEE Access*, [online] 9, pp.155161–155196. doi:https://doi.org/10.1109/ACCESS.2021.3127960.

[22] Sharma, N., Jain, V. and Mishra, A. (2018). An Analysis Of Convolutional Neural Networks For Image Classification. *Procedia Computer Science*, 132, pp.377–384. doi:https://doi.org/10.1016/j.procs.2018.05.198.

[23] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D. and Vanhoucke, V. (2015). Going deeper with convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* [online] pp.1–9. doi:https://doi.org/10.1109/cvpr.2015.7298594.

[24] www.mdpi.com. (n.d.). *Applied Sciences | Free Full-Text | Review of Artificial Intelligence Adversarial Attack and Defense Technologies*. [online] Available at: https://www.mdpi.com/2076-3417/9/5/909/review_report.

[25] pytorch.org. (n.d.). *Adversarial Example Generation — PyTorch Tutorials 2.3.0+cu121 documentation*. [online] Available at: https://pytorch.org/tutorials/beginner/fgsm_tutorial.html#fast-gradient-sign-attack.

[26] Yuan, X., He, P., Zhu, Q. and Li, X. (2019). Adversarial Examples: Attacks and Defenses for Deep Learning. *IEEE Transactions on Neural Networks and Learning Systems*, 30(9), pp.2805–2824. doi:https://doi.org/10.1109/tnnls.2018.2886017

[27] pytorch.org. (n.d.). *Normalize — Torchvision main documentation*. [online] Available at: https://pytorch.org/vision/main/generated/torchvision.transforms.Normalize.html.

[28] Yathish, V. (2022). *Loss Functions and Their Use In Neural Networks*. [online] Medium. Available at: https://towardsdatascience.com/loss-functions-and-their-use-in-neural-networks-a470e703f1e9.

[29] pytorch.org. (n.d.). *torch.nn.functional.cross_entropy — PyTorch 2.3 documentation*. [online] Available at: https://pytorch.org/docs/stable/generated/torch.nn.functional.cross_entropy.html

[30] Papernot, N., Mcdaniel, P., Wu, X., Jha, S. and Swami, A. (n.d.). *Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks*. [online] Available at: https://arxiv.org/pdf/1511.04508.

[31] “Torch.nn — PyTorch 1.13 Documentation.” Pytorch.org, pytorch.org/docs/stable/nn.html#loss-functions.

[32]“Negative Log Likelihood and Softmax.” Tyler’s Blog, 20 Oct. 2023, blog.tnichols.org/posts/softmax-classifier/#numerical-stability. Accessed 28 July 2024.