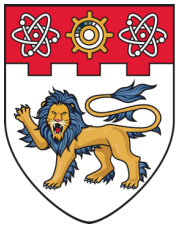
Robust AI: Security and Privacy Issues in Machine Learning



NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

Nandish Chattopadhyay

School of Computer Science and Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

2023

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

14/02/2023

.....

Date

Nandish Chattopadhyay

.....

Nandish Chattopadhyay

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

14/02/2023

Date

Assoc. Prof. Anupam Chattopadhyay

Authorship Attribution Statement

This thesis contains material from 10 paper(s) published/accepted at the following conferences, or submitted for review, in which I am listed as the primary author.

Chapter 3 is published in part as *Nandish Chattopadhyay, Anupam Chattopadhyay, Sourav Sen Gupta and Michael Kasper, "Curse of Dimensionality in Adversarial Examples," in Proceedings of IEEE IJCNN 2019*, where Prof. Anupam Chattopadhyay, Prof. Sourav Sen Gupta and Michael Kasper provided valuable guidance in the project and assisted with reviewing and editing the manuscript and *Nandish Chattopadhyay, Subhrojyoti Chatterjee and Anupam Chattopadhyay, "Robustness Against Adversarial Attacks using Dimensionality," in Proceedings of SPACE 2021* where Subhrojyoti Chatterjee assisted with additional experimental results and Prof. Anupam Chattopadhyay provided valuable guidance in the project and assisted with reviewing and editing the manuscript.

Chapter 4 is submitted for review as *Nandish Chattopadhyay, Atreya Goswami and Anupam Chattopadhyay , "Adversarial Attacks and Dimensionality in Text Classifiers,"*. Atreya Goswami assisted with additional experimental results and Prof. Anupam Chattopadhyay provided valuable guidance in the project and assisted with reviewing and editing the manuscript.

Chapter 5 is published as *Nandish Chattopadhyay, Shivam Garg and Anupam Chattopadhyay , "Adversarial robustness in Object Detectors using dimensionality reduction," in Proceedings of IEEE TrustCom 2022.* Shivam Garg assisted with additional experimental results and Prof. Anupam Chattopadhyay provided valuable guidance in the project and assisted with reviewing and editing the manuscript.

Chapter 6 is submitted for review, as *Nandish Chattopadhyay, Lionell Yip En Zhi, Bryan Tan Bing Xing and Anupam Chattopadhyay, "Analyzing spatially correlated patterns in adversarial images," and the PrePrint is available at arXiv.* Lionell Yip En Zhi and Bryan Tan Bing Xing assisted with additional experimental results and Prof. Anupam Chattopadhyay provided valuable guidance in the project and assisted with reviewing and editing the manuscript.

Chapter 7 is published as *Nandish Chattopadhyay, Chua Sheng Yang Viroy and Anupam Chattopadhyay, "Re-Markable: Stealing Watermarked Neural Networks through Synthesis," in Proceedings of SPACE 2020.* Chua Sheng Yang Viroy assisted with additional experimental results and Prof. Anupam Chattopadhyay provided valuable guidance in the project and assisted with reviewing and editing the manuscript.

Chapter 8 is published in part as *Nandish Chattopadhyay, Anupam Chattopadhyay, "ROWBACK: Robust Watermarking of Neural Networks using Backdooring," in Proceedings of IEEE ICMLA 2021* where Prof. Anupam Chattopadhyay provided valuable guidance in the project and assisted with reviewing and editing the manuscript and *Nandish Chattopadhyay, Rajan Kataria and Anupam Chattopadhyay, "TextBack: Watermarking Text Classifiers using Backdooring," in Proceedings of EuroMicro DSD 2022* where Rajan Kataria assisted with additional experimental results and Prof. Anupam Chattopadhyay provided valuable guidance in the project and assisted with reviewing and editing the manuscript.

Chapter 9 is published in part as *Nandish Chattopadhyay, Anupam Chattopadhyay and Ritabrata Maiti, "Deploy-able Privacy Preserving Collaborative ML," in Proceedings of IEEE ICDCS 2020* where Ritabrata Maiti assisted with additional experimental results and Prof. Anupam Chattopadhyay provided valuable guidance in the project and assisted with reviewing and editing the manuscript and *Nandish Chattopadhyay, Arpit Singh and Anupam Chattopadhyay, "ROFL: RObust privacy preserving Federated Learning," in Proceedings of IEEE ICDCS 2022* where Arpit Singh assisted with additional experimental results and Prof. Anupam Chattopadhyay provided valuable guidance in the project and assisted with reviewing and editing the manuscript.

14/02/2023
.....
Date

.....
Nandish Chattopadhyay

Acknowledgements

I would like to express my sincere gratitude to my supervisor Prof. Anupam Chattopadhyay for his invaluable guidance and support throughout the duration of my PhD. I appreciate his mentorship, and I want to thank him for giving me the freedom to shape my work, while constantly ensuring that I am progressing in the right direction, by encouraging me when I am right and correcting me when I am not. I am certain that this has deeply benefited the quality of work that we could produce during the course of my PhD.

I would also like to thank my Thesis Advisory Committee members and the funding agency of my PhD, the National Research Foundation and Cyber-Security Agency of Singapore for supporting me under the SoCure project.

Additionally, I would like to thank Prof. Sourav Sen Gupta for being more of a friend apart form being a guide, not just through the duration of the PhD but also in life, in general.

Finally. I would like to extend my gratitude to my father for supporting me throughout and encouraging me. Also, on the personal side, this would not have been possible without the support of my friends, the sacred bunch that feels like family.

To the loving memory of my mother.

I knew a woman, a lady who birthed and raised me;
From her womb to my little cradle, she was there,
Always by my side, a friend and confidante was she;
Riding through our highs and lows of equal share...

Cheers to every mom, and the idea of motherhood;
Perpetuating creation, and shaping life by nurturing;
Growing a bond from the first moment of childhood-
Its sanctity is the purest expression of human being.

Now I see my mum, only in some transient dreams-
Many years have passed by, but not a day in essence;
Mother's love is not limited by mere physical realms:
I still see her face, feel her warmth and her presence.

Celebrating mothers, some in person, in spirit for some;
Our genesis and our eternal protector for times to come!

Abstract

Machine learning based decision making can be adopted in practice as a driver of most applications only when there are strong guarantees on its reliability. The trust of those involved as stakeholders needs to be established for making it more ubiquitous and acceptable. In general, the idea of reliability in machine learning can be construed as a sum of two parts, Robustness and Resilience. Since reliability is concerned about providing assurances against malfunctions or errors, it can also be classified based on the types of those errors. This thesis deals with Robustness, that is robustness against attacks, which are responsible for generating intentional and malicious errors. The robustness is therefore to be studied in conjunction with the mechanisms an adversary may adopt to disrupt the machine learning application.

To investigate and understand Robustness in ML algorithms against various forms of attacks, we break down each into components and consider all possible combinations. Typically, Robustness is studied with respect to security and privacy aspects. The ML algorithm itself comprises of the trained model and the training data. Therefore, we consider security and privacy related problems pertaining to both components of ML, that is the model and the data. It may be noted here that the work touches upon many important problems in this regard, but it is not exhaustive. We consider all problems related to attacks that jeopardize the fundamental ML task itself under the umbrella of security issues and those that deal with attacks leaking secret and sensitive information about the system under privacy.

The primary vulnerability of machine learning models with respect to security is adversarial attacks. In the first part, we consider security for models, and study the adversarial attacks through the lens of dimensionality. We assert that the high dimensional landscape in which the neural network models optimize facilitate the generation of adversarial examples and that dimensionality reduction enhances adversarial robustness. We have explored the mathematical background for this

proposition, studying the properties of data distributions in the high dimensional spaces and nature of such trained manifolds. The idea has been empirically justified thereafter [3]. We have extended this notion of the influence of dimensionality on adversarial sample generation from images to videos [5] and text [4] and provided practical and efficient solutions by leveraging adversarial samples detection and dimensionality reduction. It is important to note here that reducing the dimensionality has an additional computational cost and can in some cases also have an adverse effect on the fundamental machine learning task itself. We therefore optimise the dimension reduction operation for each of the tasks and use-cases and carefully choose the amount of variability to preserve that effectively eliminates adversarial noise but retains the meaningful information necessary for classification or object detection etc. Additionally, in one of the works, we make use of a parallel channel to run the classifications task and adversarial sample detection and apply dimensionality reduction to only those samples that are detected as adversarial [3]. This significantly improves efficiency of the overall system and proves to be beneficial in terms of accuracy as well.

Thereafter, we study the security flaw of adversarial attacks from the perspective of vulnerable features within the data. We have analysed spatially correlated patterns within adversarial images [6]. Class wise, we have split images into two key parts, Region of Importance (RoI) and Region of Attack (RoA), such that the RoI is the region that the classifier is particularly sensitive to, during the classification task, and the RoA is the region which the adversarial attack modifies. The goal of this exercise was to figure out areas within the images that do not contribute to the task of classification but are adversarially vulnerable. Our proposed adversarial defence mechanism out of this work is to neutralize that region, therefore bringing down adversarial vulnerability without compromising on classification accuracy. The idea is demonstrated through benchmarking datasets and models.

Moving on to the aspect of privacy in the second part, we look at some very different problems. First, we look at privacy for the models, followed by that of the data. In the context of preserving privacy of the trained neural network models, we direct our attention to protecting ownership and IP rights of the models, using watermarking. We review the state-of-the-art in watermarking schemes for neural networks and select the most appropriate ones to study. Watermarking using backdooring is the scheme of choice here. We investigate the vulnerabilities of this

scheme and break it using synthesis. In our proposition titled Re-Markable [7], we make the assumption that the adversary has very limited compute power and access to samples from the data distribution relevant to the task. We train a GAN (Generative Adversarial Network) to synthesize more samples and use the synthesized samples to re-train just the fully connected layers of the watermarked models. As demonstrated, it turns out that this minimal computation is sufficient to eliminate the presence of embedded watermarks from the model, and this vulnerability makes the existing scheme extremely unreliable. To solve the problem thus discovered, we worked on a robust watermarking scheme that overcomes this vulnerability. ROWBACK, or robust watermarking for neural networks using backdooring [8], uses a redesigned mechanism for generating the Trigger Set (using adversarial examples with explicit labelling) that is used as the private key for the watermarking, and a method of explicitly marking every layer of the neural network with the embedded watermarks. The goal is to ensure that an adversary interested in extracting the network would need to re-train every layer of the model, which is as good as training a fresh model from scratch as it would require extensive training samples and compute power. We also extended the idea of robust watermarking for models in the domain of natural language processing, particularly text classifiers. TextBack [8], designed to embed watermarks within text classifiers using backdooring, uses a marking scheme that involves the Trigger samples and clean samples together, unlike that in images, as this property is observed only in sequential models like recurrent neural networks and LSTM based models.

Finally, to cover the aspect of privacy of data, we focus on collaborative ML. The motivation in this work is to protect the privacy of the data that is used for training, as many practical applications necessitate the usage of highly sensitive data. This data is decentralised, resides with different non-collocated entities, and is not sharable on privacy grounds. The straight-forward solution to this problem is available in the literature in the form of Federated Learning. Additionally, an adversary may tap into the federated learning infrastructure in multiple ways and extract information. For example, Membership Inference attacks are possible on models deployed in the cloud (which is natural in many federated learning setups), wherein the model could be queried multiple times and based on the output probabilities returned by the model, one can train another separate model to know if the particular input that had been sent as the query belonged to the training set

or not. Carrying out this process multiple times can in theory lead to the reverse-engineering of the entire training set. This is a serious privacy violation. We attempted to solve this problem using Differential Privacy, where a differentially private learning algorithm is used by the participants of the federated learning infrastructure for their local training, involving gradient trimming and addition of sampled noise. We studied practical applications of such collaborative learning systems and deployed the framework on edge devices, by creating light-weight versions of the models that do not compromise on accuracy [9]. Similarly, in another use-case, the participants of the federated learning setup itself could have malicious intentions and can come up with sabotaging attacks on the learning framework. Considering such potential single points of failures of the overall system, we proposed a robust federated learning infrastructure, that assigns coefficients to the updates sent by the clients to the server. This takes care of tolerating faults in up to 50% of the clients' failures [9].

Overall, the ideas discussed in this thesis are a major step towards making machine learning systems more robust and is therefore a necessary step in the direction of reliable AI.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The idea of machine learning is fundamentally concerned about creating systems that can improve at their specific tasks automatically through gaining experience [8]. The first seeds of this learning paradigm were planted when scientists of different fields like physics could not explain certain observations from the pre-existing laws of nature and needed a way to modify their models to fit the data that could not be ignored as anomalies. Statisticians developed tools to approximate functions from the observed data that were found through scientific experiments. This data driven method soon grew out to a discipline of its own and now evolved into a paradigm that touches almost all walks of life. Statistical learning tools were first used to generate simple approximations of linear functions that could fit a model corresponding to a set of inputs and outputs. This was necessary because the system under study was not understandable with the existing literature and knowledge or too complex to be explained by the known set of laws. Therefore, machine learning can be perceived as the study of algorithms and supporting theory for predictions and decisions under uncertainty based on observed data.

Artificial intelligence (AI)[9], sometimes called machine intelligence, is intelligence demonstrated by machines, in contrast to the natural intelligence displayed by humans and other animals. In computer science AI research is defined as the study of "intelligent agents": any device that perceives its environment and takes actions that maximize its chance of successfully achieving its goals [10].

Machine Learning is the decision making core of the AI systems. It attempts to address a plethora of problems including some crucial ones like reasoning, problem solving decision making, knowledge representation, planning, learning, natural language processing, perception, motion and manipulation, social intelligence and ultimately general intelligence. Typically, the approaches include cybernetics and brain simulation, statistical learning or machine learning, and most recently deep learning.

In principle, one of the discriminating notions between traditional machine learning or statistical learning techniques and AI lies in the generic problem solving abilities of algorithms in AI. Machine learning is the driving force behind AI, however most machine learning models are designed in a way that they are more and more of specific purpose, that is the models itself are data dependent and are applicable to only a limited situations or datasets. In contrast, the deep learning algorithms of AI are more generic, and the approach used can be applied on comparatively diverse data. The other key difference would be the capacity to handle large amount of data. While most statistical learning algorithms do not perform very well after a particular amount of data has been fed to it, in terms of increasing accuracy any further, the deep learning algorithms perform exceedingly well with higher dimensional data, with higher accuracy.Deep Learning is a specific class of neural network based Machine Learning algorithms, which is the most important component of the overall Artificial Intelligence paradigm. The Figure 1.1 shows the relevance and evolution of Artificial Intelligence, Machine Learning and Deep Learning.

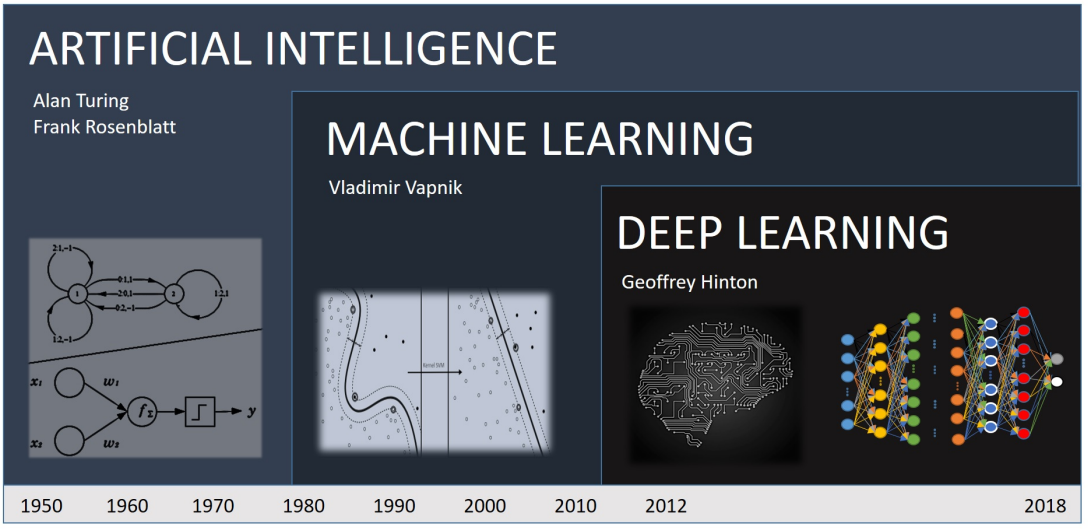


FIGURE 1.1: Evolution of AI with time and relation between AI, ML and DL.

The use of data-intensive machine-learning methods can be found throughout science, technology and commerce, leading to more evidence-based decision-making across many walks of life, including health care, manufacturing, education, financial modeling, policing, and marketing.

The global machine learning (ML) market is expected to grow from $21.17 billion in 2022 to $209.91 billion by 2029, at a CAGR of 38.8% in forecast period, as per the latest report published by Fortune Business Insights of 2022.

While the future of the machine learning driven world looks exciting enough, with enormous possibilities, there are problems to solved with just the way it is at the moment. Although the algorithms have performed well in experimental conditions and setups, its practical usage in daily activities is still limited due to a lack of trust in its capabilities. In this thesis, we posit that such trust-able guarantees can be provided with research in the area of reliable ML.

1.1 Motivation

Technology has the potential to positively influence every human life in almost every regular activity. The very same capabilities, if abused, can lead to terrible consequences. The responsibility lies with those who are leading the research and development in the domain to ensure that the positive outcomes are maximised and the negative outcomes are minimised. Any technology, particularly digital technologies, must be evaluated for being trust-worthy before mass adoption and deployment.

In general, technologies must fulfil certain criteria, in a specific order, for being declared fit for regular adoption and use. We have proposed a hierarchy for this purpose, inspired by Maslow's hierarchy of human needs [11]. The primary requirement of the technology paradigm is that it has to be functional. It has to successfully accomplish the tasks it has been designed for, with low rates of failure. Secondly, it has be efficient. It has to perform the tasks at the least possible cost (the cost might vary on a case by case basis). Thirdly, it has to Secure against attacks that attempt to disrupt its functionality. Fourth, it has to maintain privacy of the participants who contribute data in any form to the operation of the paradigm. Finally, it has to be ethical and must abide by the norms of the society.

One might take the example of the internet, to see how technologists have been striving to fulfil these requirements and the medium evolves.

The exact same applies to the paradigm of AI and machine learning as well. This mapping is explained in the Figure 1.2.

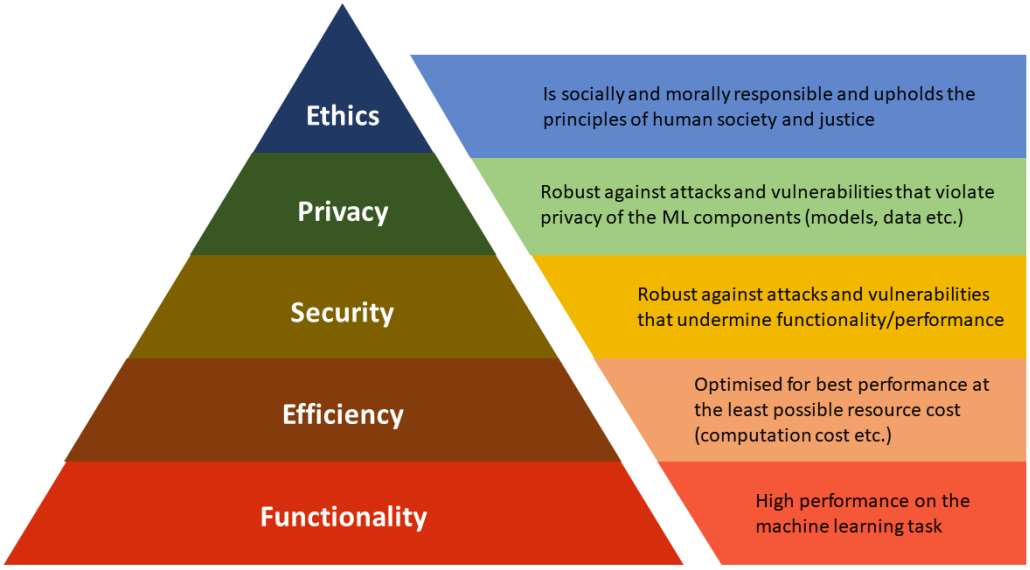


FIGURE 1.2: The hierarchy of requirements to be fulfilled by technologies, mapped to the context of the machine learning paradigm.

For most machine learning applications, the research into neural architectures and training has given it the necessary boost required for the paradigm to be well functional and efficient. The next steps are still to be worked on. We pick the two areas of Security and Privacy and study them under the umbrella of overall Reliability.

The proposition of ML adoption in practice necessitates a strong guarantee of reliability in the operations of the algorithms. Despite the promise, the machine learning algorithms are still used as auxiliary decision drivers because of the lack of trust of those involved. Reliability is key to taking AI forward, which otherwise has the potential to impact almost every walk of life. Reliability can be construed as a sum of two components, Robustness and Resilience. To put things into perspective, we begin by asserting that Reliability is concerned about providing guarantees against malfunctions or errors in the complex machine learning systems. These malfunctions or errors could be looked upon from two aspects: whether they are intentional or unintentional and whether they are malicious or

not. When the malfunctions are not malicious, they can be ignored as trivial. If the malfunctions are malicious and unintentional, typically they are studied under the Resilience aspect of Reliable AI that deals with safe ML and time critical ML etc. Robustness deals with the kinds of malfunctions and errors that are both intentional and malicious. Therefore, when we talk of robustness, there exists an adversary and presents a definitive attack model. The work presented here concerns robustness of ML algorithms, and deals with a study and countermeasures to a variety of attacks on different types of ML applications. It is to be noted here that the focus of this thesis is on the machine learning algorithms themselves, as opposed to the security and privacy aspects, and the performance metrics used consistently are in the context of accuracy of the given tasks, with the additional measurements of efficiency, robustness etc. The contributions are predominantly at the algorithmic level, along with the engineering necessities. To investigate and understand Robustness in ML algorithms against various forms of attacks, we break down each into components and consider all possible combinations. Typically, Robustness is studied with respect to security and privacy aspects. The ML algorithm itself comprises of the trained model and the training data. Therefore, we consider security and privacy related problems pertaining to both components of ML, that is the model and the data. It may be noted here that the work touches upon many important problems in this regard, but it is not exhaustive. We consider all problems related to attacks that jeopardize the fundamental ML task itself under the umbrella of Security issues and those that deal with attacks leaking secret and sensitive information about the system under Privacy.

1.2 Problem Areas

The entire work presented here has two key thrust areas, as mentioned. Within the purview of Security and Privacy issues, the problems that have been addressed are listed here. Based on a thorough review of the literature, these key issues have been found to be the primary deterrents towards a reliable adoption of many machine learning based schemes in regular use. We start with Security issues and the move on to Privacy issues.

1.2.1 Security Issues - Adversarial Attacks

As defined earlier, when the malfunction or error prevents the fundamental fulfillment of the machine learning objective, we categorise it as a security issue. In practice, the most common security problem on machine learning models from the algorithmic perspective is that of adversarial attacks. An adversary can leverage a plethora of attacking algorithms to severely reduce the performance of existing models, therefore threatening the use of AI in many safety-critical applications. Structured perturbations introduced to test samples can fool high performing trained neural networks and force errors. We have looked at this problem, first from the perspective of the neural architecture, and the underlying optimization problem involved in training the model, as mentioned below in stages:

- Study and analysis of the optimization landscape of the trained neural network models to identify potential causes that facilitate the occurrence of adversarial attacks
- The use of the resultant findings to design adversarial defence mechanisms
- The efficiency and applicability of the adversarial defence mechanisms on various different forms of data

Secondly, we also looked into the data samples for patterns that might be responsible for adversarial vulnerability. We studied zones within images belonging to similar class of images for trends facilitating adversarial attacks.

1.2.2 Privacy Issues - Threats to the Model and Trainset

While studying privacy issues of machine learning algorithms, we have taken a two pronged approach to tackle the problems relating to that of the neural architectures and models separately from that of the training data. For handling the aspect of private ownership of trained neural networks, we focused on watermarking neural networks and their corresponding vulnerabilities and robustness. Specifically, the efforts were targeted towards the following:

- Exposing vulnerabilities in the state-of-the-art in watermarking neural networks by stealing watermarked model for unauthorised usage

- Developing corresponding countermeasures for robustness against such attacks
- Extending robust watermarking schemes to different forms of data

On the other hand, for handling the aspect of maintaining the privacy of the data, specifically in the scenario where the data is decentralised we focused on different federated learning infrastructures. In particular, the study was targeted at collaborative learning frameworks for edge devices with resource constrains and robustness in case of a certain kind of attack.

1.3 Contributions

The aforementioned problems have been attempted to be solved by the works presented in this thesis. As shown earlier, this is also split into two parts, that of Security issues and of Privacy issues.

1.3.1 Security Solutions - Adversarial Countermeasures

Potential solution approaches to the problems relating to Security issues have been mentioned here.

1. Curse of Dimensionality in Adversarial Examples: In this work, we try to relate the geometry of the high-dimensional space in which the model operates and optimizes, and the properties and problems therein, to such adversarial attacks. We present the mathematical background, the intuition behind the existence of adversarial examples and substantiate them with empirical results from our experiments.
2. Robustness Against Adversarial Attacks using Dimensionality: In this work, we propose a machine learning infrastructure that is robust against adversarial attacks. The proposed pipeline uses a parallel pathway to carry out the primary machine learning task as well as identify adversarial samples. We use dimension reduction techniques to project adversarial samples into lower

dimensions, thus eliminating adversarial perturbations. The end-to-end system has been tested to provide robustness, through different choices of neural architectures (VGG-19 [1], Inception ResNet v2 [12]), modes of adversarial attacks (FGSM [2] and PGD [13]) and multiple datasets (MNIST[5], CIFAR-10 and CIFAR-100 [6]). The overall accuracy of the end-to-end system was consistently found to be within a range of 1-2% (2%-4% in worst case scenarios) of what the accuracy would have been without the presence of any adversarial examples. We also show that this robustness is delivered without much compromise in time consumed for inference per sample.

3. Adversarial Attacks and Dimensionality in Text Classifiers: Historically, adversarial attacks have been first identified and studied in the domain of image processing. In this work, we study adversarial examples in the field of natural language processing, specifically text classification tasks. We investigate the reasons for adversarial vulnerability, particularly in relation to the inherent dimensionality of the model. We utilise the sensitivity of the adversarial examples to the models that have been trained with an input embedding dimension that matches the one against which they were created in the first place, to propose defence mechanisms. We make use of ensemble models of varying inherent dimensionality to thwart the attacks. Additionally, we also study the problem of measuring adversarial perturbation using different distance metrics. For all of the aforementioned studies, we have run tests on multiple models with varying dimensionality and used a word-vector level adversarial attack to substantiate the findings.

4. Adversarial attack on video streams: In this work, we take a step back from the cat and mouse chase of novel attacks and ad-hoc defenses and try to explain adversarial attacks from the perspective of the geometry of the high-dimensional spaces that the models operate in. Additionally, we make use of our idea of relating adversarial attacks to dimensionality to propose a counter-measure that uses dimension reduction. We have tested our proposition on state-of-the-art object detection and classification models on video streams including Faster-RCNN and YOLO and corresponding adversarial attacks on these models. Having optimally tuned the hyper-parameter associated with variability preservation upon dimension reduction using simple Singular Value Decomposition, we have shown that the performance of the robust

version of the object detector models is within $2-3\%$ of that on the clean samples, despite the presence of adversarial perturbation.

5. Spatially Correlated Patterns in Adversarial Images: To have a better understanding of the key contributors of structured adversarial attacks, we searched for and studied spatially co-located patterns in the distribution of pixels in the input space. In this paper, we propose a framework for segregating and isolating regions within an input image which are particularly critical towards either classification (during inference), or adversarial vulnerability or both. We assert that during inference, the trained model looks at a specific region in the image, which we call Region of Importance (RoI); and the attacker looks at a region to alter/modify, which we call Region of Attack (RoA). The success of this approach could also be used to design a post-hoc adversarial defence method, as illustrated by our observations. This uses the notion of blocking out (we call neutralizing) that region of the image which is highly vulnerable to adversarial attacks but is not important for the task of classification. We establish the theoretical setup for formalising the process of segregation, isolation and neutralization and substantiate it through empirical analysis on standard benchmarking datasets. The findings strongly indicate that mapping features into the input space preserves the significant patterns typically observed in the feature-space while adding major interpretability and therefore simplifies potential defensive mechanisms.

1.3.2 Privacy Solutions - Privacy Preserving ML

Our propositions to solve the problems that have earlier been categorised as those relating to Privacy are mentioned below:

1. Re-Markable: Stealing Watermarked Neural Networks through Synthesis: The key to the high performance of trained neural networks involves efforts in systematic curation of relevant data, optimization of neural architectures and heavy investments in hardware infrastructure essential for training models. This necessitates the requirement of frameworks for establishing ownership on trained models by associated stakeholders. Watermarking neural networks have been used as a mechanism for establishing such Intellectual Property

rights, verifiable when necessary, created with the goal of preventing theft and unauthorised use. In this work, we explore the vulnerabilities of watermarking neural networks by backdooring, which spawned a wide range of variations around the same principle due to its potential robustness. We propose a model-modification attack to successfully extract the trained neural network and demonstrate the removal of watermarking signatures. We use synthesis with GANs to generate samples, which can be used to re-train the marked models and achieve comparable accuracies of performance (within about 3% of actual model). The proposed attack successfully removes verifiable watermarks, as tested on highly benchmarked datasets including MNIST, Fashion MNIST and CIFAR-10, thus jeopardizing the reliability of the state-of-the-art watermarking scheme.

2. ROWBACK: RObust Watermarking for neural networks using BACKdoors: In this work, we propose a robust watermarking mechanism for neural architectures. Our proposed method ROWBACK turns two properties of neural networks, the presence of adversarial examples and the ability to trap backdoors in the network while training, into a scheme that guarantees strong proofs of ownership. We redesign the Trigger Set for watermarking using adversarial examples of the model which needs to be watermarked, and assign specific labels based on adversarial behaviour. We also mark every layer separately, during training, in order to ensure that removing watermarks requires complete retraining. We have tested ROWBACK for satisfying key indicative properties expected of a reliable watermarking scheme (generates accuracies within $1-2\%$ of actual model, and a complete 100% match on the Trigger Set for verification), whilst being robust against state-of-the-art watermark removal attacks [4] (requires re-training of all layers with at least 60% samples and for at least more than 45% of epochs of actual training).

3. TextBack: Watermarking Text Classifiers using Backdooring: Watermarking the trained neural architectures can prove to be a solution to claiming ownership of trained neural network models. While such techniques have been demonstrated in image classification tasks, we posit that a watermarking scheme can be developed for natural language processing applications as well. In this work, we propose TextBack, which is a watermarking technique developed for text classifiers using backdooring. We have tested for the functionality preserving properties and verifiable proof of ownership of TextBack

on multiple neural architectures and datasets for text classification tasks. The watermarked models consistently generate accuracies within a range of $1-2\%$ of models without any watermarking, whilst being reliably verifiable during watermarking verification. TextBack has been tested on two different kinds of Trigger Sets, which can be chosen by the owner as preferred. We have studied the efficiencies of the algorithm that embeds the watermarks by fine tuning using a combination of Trigger samples and clean samples. The benefit of using TextBack's fine tuning approach on pre-trained models from a computational cost perspective against embedding watermarks by training models from scratch is also established experimentally.

4. Deploy-able Privacy Preserving Collaborative ML: In the data-driven world, emerging technologies like the Internet of Things (IoT) and other crowd-sourced data sources like mobile devices etc. generate a tremendous volume of decentralized data that needs to be analyzed for obtaining useful insights, necessary for reliable decision making. Although the overall data is rich, contributors of such kind of data are reluctant to share their own data due to serious concerns regarding protection of their privacy; while those interested in harvesting the data are constrained by the limited computational resources available with each participant. In this work, we propose an end-to-end algorithm that puts in coalescence the mechanism of learning collaboratively in a decentralized fashion, using Federated Learning, while preserving differential privacy of each participating client, which are typically conceived as resource-constrained edge devices. We have developed the proposed infrastructure and analyzed its performance from the standpoint of a machine learning task using standard metrics. We observed that the collaborative learning framework actually increases prediction capabilities in comparison to a centrally trained model (by 1-2%), without having to share data amongst the participants, while strong guarantees on privacy (ϵ, δ) can be provided with some compromise on performance (about 2-4%). Additionally, quantization of the model for deployment on edge devices do not degrade its capability, whilst enhancing the overall system efficiency.

5. ROFL: RObust privacy preserving Federated Learning: In this work, we propose ROFL, which is an end-to-end robust mechanism of learning, that has been developed keeping all the trust issues in mind and addressing the necessity of privacy. We make note of the threat models that might make the

participants apprehensive and design a bi-directional two-dimensional privacy preserving framework, that builds upon the state-of-the-art in differentially private federated learning. Specifically, we propose a weighted federated averaging technique for aggregation of the differentially private models generated by the participants. We are able to provide privacy guarantees without compromising on the accuracy of the machine learning task. ROFL has been tested for multiple neural architectures (VGG-16 [1] and ResNet [7]) on multiple datasets (MNIST [5], CIFAR-10 and CIFAR-100 [6]). On the machine learning tasks, it is able to achieve accuracies within the range of 1%-2% of what a model trained on the collected data would have generated, in the average case scenario. We have verified the robustness of ROFL against attacks involving sabotaging or malicious client providing erroneous models. The study on model convergence reveals how to improve the efficiency of ROFL. We also provide evidence on how ROFL is easily scalable in nature.

1.4 Implications

The goal of this thesis has been to primarily address the plethora of vulnerabilities in the context of security and privacy issues in machine learning models and propose robust countermeasures. These set of mechanisms provide robustness from the algorithmic perspective. However, in the end this forms one piece of the bigger picture, which involves running these algorithms in practice, on some system-on-chip (SoC), that has a CPU and potentially some dedicated machine learning accelerators. A typical design of such an SoC is shown in figure 1.3. The algorithmic robustness is necessary but not sufficient for practical usage. These have to be incorporated within secure hardware accelerators and ML co-processors for deriving their true potential. This naturally forms the follow up work to this thesis, and the hardware-software co-design created therein, is of great utility for many practical applications that generally benefit from the data driven machine learning paradigm.

Additionally, machine learning techniques can be used for hardware based security as well, for the aforementioned SoCs and accelerators. Contrary to most hardware-implemented countermeasures, these are algorithmic. Our study of neural architectures and their behaviour under different types of attacks have led to a comprehensive understanding of their vulnerabilities. As a follow up of the same, we have

made progress in defending against side channel attacks [14] on SoCs which run machine learning models. The intuition is that since most side-channel attacks are able to correlate the neural architecture and weights to the physical traces that they produce when deployed on an SoCs [15], breaking the correlation should involve introducing some form of randomness. We achieve this using parallel pathways within the neural architectures, which are randomly select-able during execution. The overall implementation therefore is a combination of multiple neural architectures, each of which are capable of accomplishing the machine learning task. Since there is not a fixed network, and instead multiple such networks are coexisting simultaneously, the process of correlating the traces to the network becomes difficult.

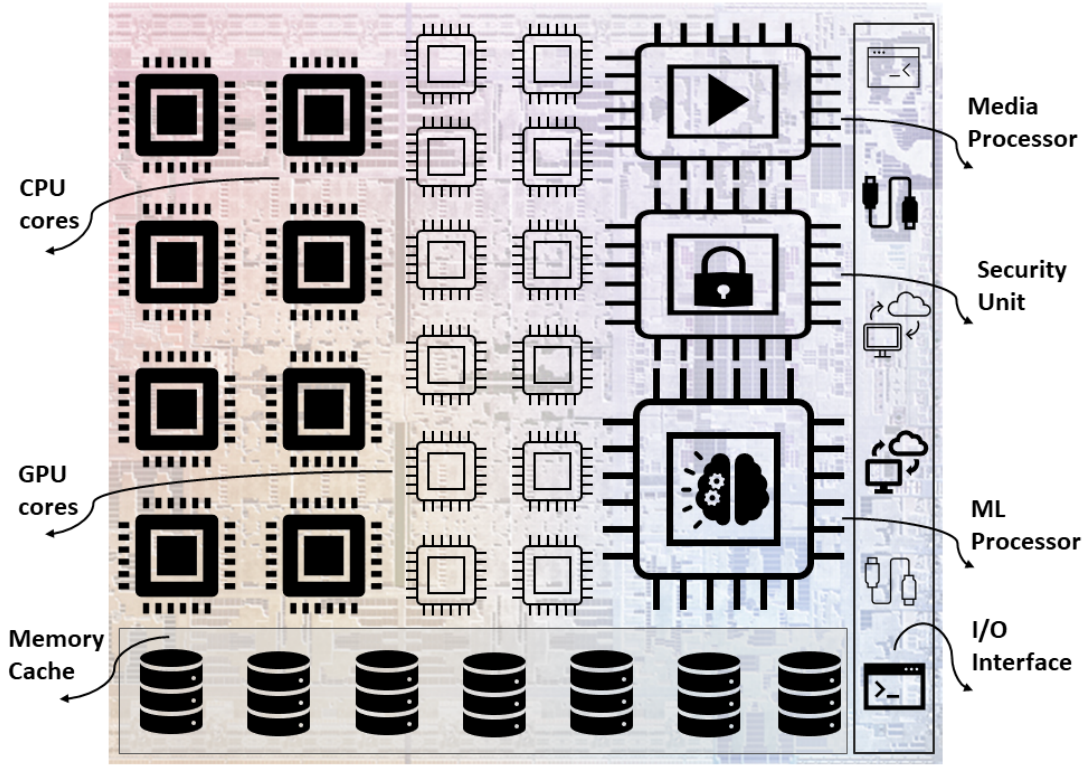


FIGURE 1.3: Architecture of any standard SoC with the essential components.

1.5 Thesis Structure

The structure of the thesis is presented here. As mentioned earlier, the notion of robustness is broken down into two key aspects, those pertaining to Security

and Privacy. For each of the branches, we looked at problems relating to the two components of machine learning, that is those relating to the data and the model. This is explained in a simple way in the Figure 1.4.

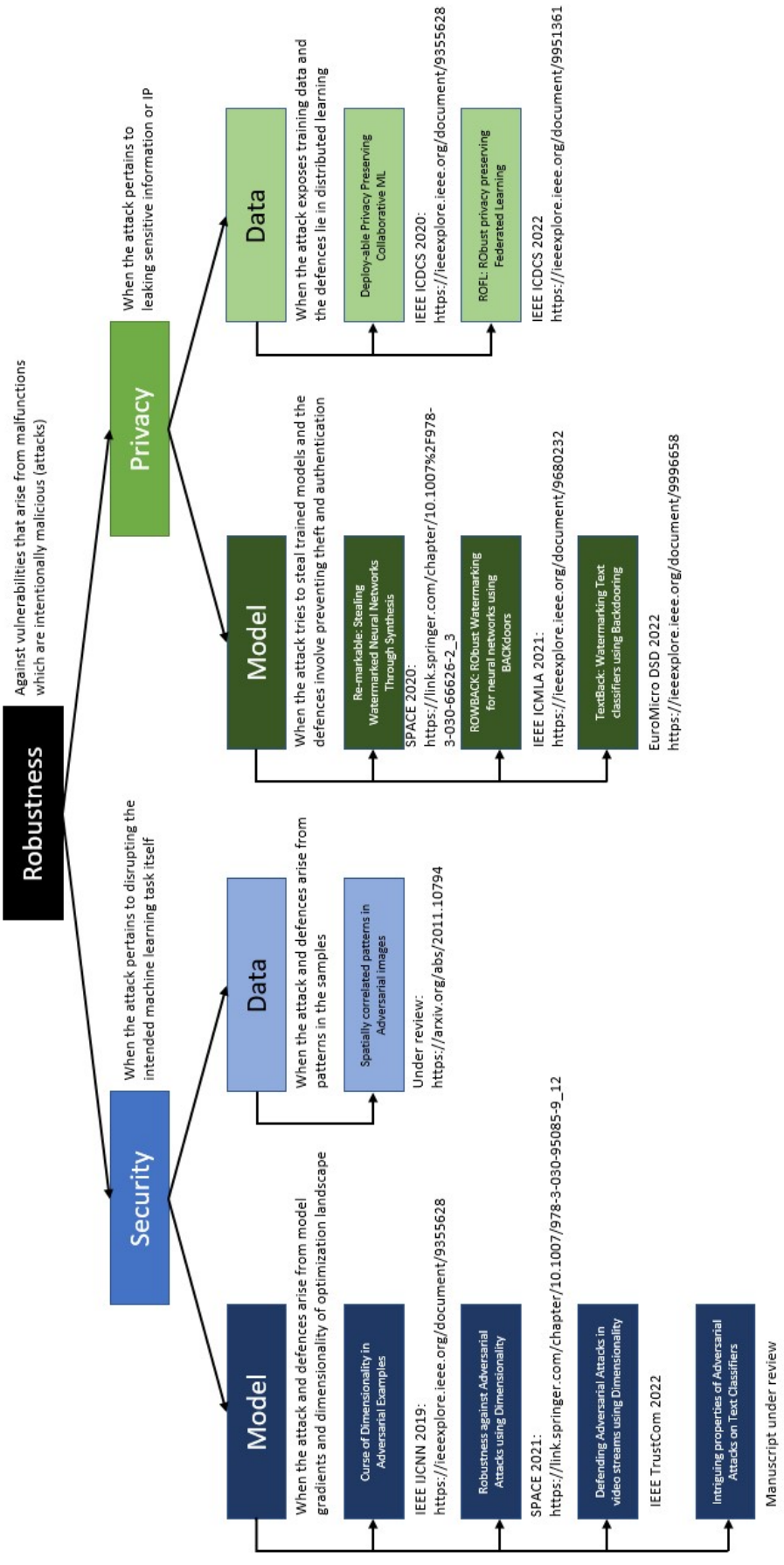


FIGURE 1.4: The overall structure of this thesis.

Chapter 2

Literature Review

To have an understanding of the premise of this work, we review the relevant literature available, concerning all of the different aspects addressed in this thesis. We start by looking at how machine learning and AI have developed over time. Them we review the work that has been done in the area of security, that is in the study of adversarial attacks. Then we look at the various privacy issues that have been discussed in detail later.

2.1 Evolution of AI

Over the years, research in machine learning has seen improvements by leaps and bounds, particularly with the advancements in high performance neural networks, and the overall realm of deep learning. Some applications of neural architectures have reached and surpassed human accuracies, giving it a fresh start [16]. A variety of science and technology applications have adopted the data driven paradigm, both in academia and the industry. The flourishing use of AI took a major leap in 2012, when it was demonstrated for the first time that for a particular task, which was an image classification task, a trained model could out-perform the average human [17]. Historically, the statistical learning techniques made use of laborious data pre-processing, wherein the features were to be identified and extracted from the data by different techniques before being modelled for a specific task. This approach typically lacked generalization and also suffered from a saturation in performance, potentially due to poorly extracted hand-crafted features.

The much required breakthrough was provided by the emergence of deep learning models. Although the perceptron algorithm [18] was initially proposed long back, the idea did not receive much attention due to the lack of computing hardware, that was needed to sustain its operations. This problem was solved by the advent of high performing computational systems with multiple GPU cores. They could take advantage of the linear operations within neural architectures, and make multiple computations in parallel. With the proliferation of neural networks and the state-of-the-art in the field, end-to-end systems were feasible which would perform both feature extraction and thereafter classification or regression whatever be the task. This significantly reduced human involvement in modelling the abundance of data and laid the foundation of many AI applications, particularly in the domain of Computer Vision [19], [20].

Figure 2.1 provides a quick look at the organization of different machine learning algorithms, that form the core of any AI related application. Primarily, they are of two kinds, Supervised and Unsupervised [21]. Supervised machine learning deals with problems where the corresponding data to be analysed has labels or responses associated with each sample. Unsupervised learning concerns problems where these labels are not available. Modern approaches are also at times able to mix the two for solving problems where some samples have labels and some do not. Within the categories of Supervised and Unsupervised learning paradigms, the algorithms are classified as those which are part of the deep learning frameworks and those that are not, being traditional statistical algorithms.

While the early boost in ML proliferation was seen in Computer Vision as mentioned above, the domain of Natural Language Processing has seen tremendous advancements too. Typically, Machine Learning-based automatic text classification models have been perceived to have the capability to learn different associations between tokens of the text and assign a particular output (i.e., class) to a particular input (i.e., text). The very first step for the training of an NLP classifier is pre-processing of textual data. Pre-processing in the case of text dataset includes removing punctuation (.,!$()*%), removing URLs and lower casing the text [22], tokenization [23], stopwords removal [24] and stemming [24].

Deep learning architectures provide a lot of advantages for text classification since they are inspired by how the human brain operates, called neural networks and can perform extremely well relative to other methods. Deep learning approaches

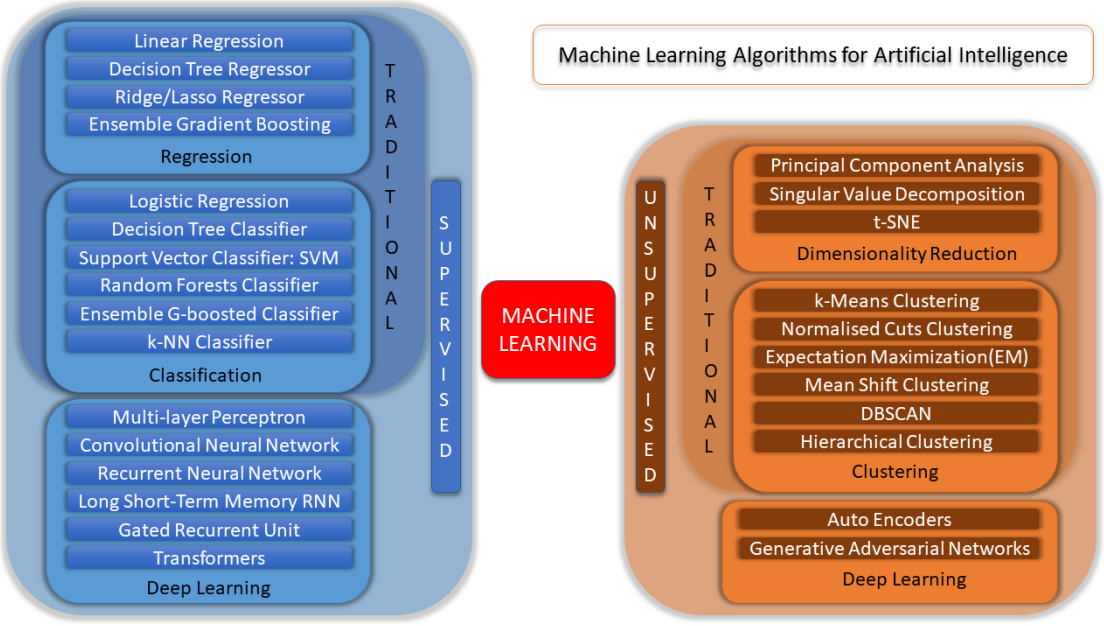


FIGURE 2.1: Overview of Machine Learning Algorithms

such as Word2Vec [25] or GloVe [26] have been used to improve the effectiveness of classifiers learned with typical machine learning algorithms by obtaining better vector representations for words. Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) [Figure 4.2] are the two basic deep learning architectures for text classification [27]. Thus, enormous neural networks emerged into the scenes with tremendous learning capacities to solve problems which were not possible to have been addressed earlier.

Having discussed the background of generic machine learning, let us take a deep dive into the areas of reliability. that are of particular importance to this thesis.

2.2 Security Issues

To review the related works within the purview of Security Issues, we primarily pay attention to Adversarial Attacks.

2.2.1 Adversarial Examples

Soon after the boom in flourishing AI, also popularly referred to as the "AI revival", a new discovery proved to a newly found problem. In 2015, it was shown that the

high performing neural networks were vulnerable to adversarial attacks [28]. The models, which would perform the task of classification very accurately on a test image dataset, would perform very poorly on a slightly tweaked dataset of those images, which the adversary could generate. Introduction of little structured perturbations, unobservable to the human eye, can bring about an unprecedented degradation in model performance, with a lot of misclassification [29]. Later, it was also shown that these adversarial examples were transferable between models. That is, an adversarial example generated for a particular neural network architecture, would also act like an adversarial example to a seemingly different model, for example a support vector classifier [30]. Subsequently, a plethora of attack mechanisms were developed by different groups of researchers working in this particular domain, and each of them cause different extents of damage to the performance of the trained model [31].

Naturally, there was an effort to understand why such a phenomenon happens. Goodfellow et al. [32] attributed the linear nature of the neural networks to be the primary reason behind such attacks. Other works suggested otherwise [33], putting the blame on dimensionality [34]. It is worth noting that the models trained for classifying images work in a very high dimensional landscape. The properties of high dimensional spaces are quite counter-intuitive and the geometry is different from what one would normally expect in low dimensions, behaviorally. While some of it has been mathematically modeled quite rigorously, there are some gaps as well in the literature.

2.2.1.1 Adversarial Examples in Images

The research in this domain of adversarial attacks in images has led to competitive works in attacks and defences which have both improved over time. The first adversarial attack was studied as an interesting property of neural networks [28], but over time, different attack mechanisms have been developed like the Fast Gradient Sign method [2], the Momentum Iterative version of it [35], the Carlini Wagner attack [36] and the Projected Gradient Descent attack [13] and others [37]. Similarly, multiple defence techniques have been proposed over time, to mitigate such attacks. Some prominent ones include the defensive distillation technique [38] and other filtering mechanisms [31]. Needless to say, as the attacks got stronger, it was

imperative to come up with stronger defences and that serves as one of the primary motivations of this work.

2.2.1.2 Adversarial Examples in NLP

Adversarial attacks have also been implemented for Natural Language Processing tasks. Generating adversarial examples for text classification is a far more challenging task since text perturbation is far more easily recognizable by a human annotator than an image perturbation, owing to its discrete nature. However growing security concerns led to active research in the field of adversarial attacks on NLP systems. Some of these attacks use character-level perturbations [39], [40]; while some use word-level perturbations [41], [42], [43]. Though initial adversarial attacks were based on a brute-force approach to find perturbations, gradually further research in this field led to discovery of more systematic and organised approach towards finding the "best" perturbation. The adversarial attack mechanisms proposed can be classified into 2 classes - white-box attacks, where model parameters, gradients and training data are known and are used to generate adversarial samples, and black-box attacks, where no information is available about the model to be attacked. In the field of white-box attacks, recently some attack mechanisms have been developed which use rule-based synonym replacement and replacement by same parts-of-speech (POS) techniques. These have led to more natural adversarial samples [44], [45]. Many efficient black-box attack techniques have also come up like BAE [46], which uses the BERT masked language model (MLM) to search for word replacements, and TEXTBUGGER [47], which sorts sentences in order of their importance and uses a scoring function to identify important words in the black-box setting. We take a step back from the cat-and-mouse chase of adversarial attacks and defences and study the possible reasons why these attacks exist, which is lacking in the available literature.

2.3 Privacy Issues

Within the purview of Privacy issues, we look at two things: privacy of the trained models and that of the training data. For preserving the privacy and authenticity

of the models, we focus our attention on watermarking neural networks. For preserving privacy of the training data, we study federated learning infrastructures and differential privacy.

2.3.1 Watermarking

There have been multiple techniques proposed in the past for watermarking neural networks. Some prominent works include BlackMarks [48], which is a blackbox watermarking scheme for Deep Neural Networks , later refined as DeepMarks [49], claiming to be a digital fingerprinting framework for Deep Neural Networks. Other approaches include DeepSigns [50], which is a generic watermarking framework for protecting the ownership of Deep Learning Models . Adversarial samples have also been used to implement watermarking schemes, like in DAWN [51], which is a dynamic adversarial watermarking of Neural Networks. However, they have not been reliable because of the simultaneous development in attacks on these models, by adversaries. These works include DeepInspect [52], which is a black-box Trojan detection and mitigation framework for Deep Neural Networks and TABOR [53] which claims to be a highly accurate approach to inspecting and restoring Trojan backdoors in AI systems etc.

2.3.1.1 Watermarking methods

The fundamental principles behind the watermarking schemes can be divided into two types. One is where the watermarks are embedded through construction, and are present within the design of the neural architecture, as observable in multiple works [49–51, 54]. The other way is to add explicitly designed samples in the training data that leaves a mark within the trained weights, like data poisoning methods. There are few such watermarking techniques that make use of embedding watermarks during training [3, 55–57]. These have been shown in an organised fashion in Figure 9.2. All of these methods, however have failed to prove robust against a plethora of attacks described hereafter.

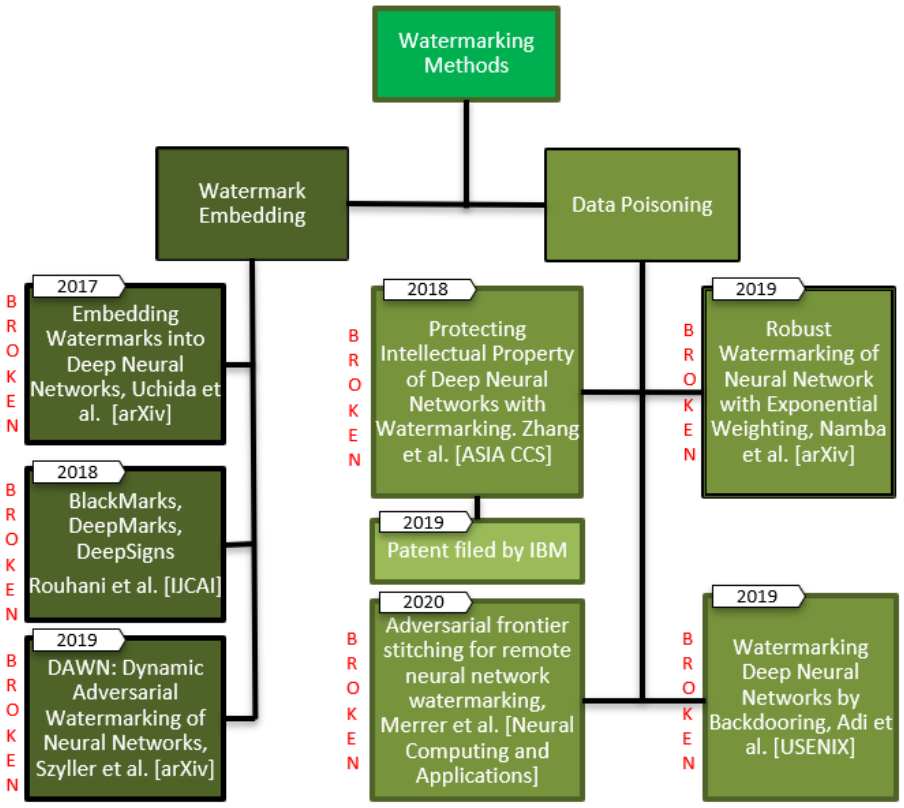


FIGURE 2.2: Literature review of watermarking models, classified by mechanism of embedding watermarks.

2.3.1.2 Vulnerabilities

While there have been multiple propositions of mechanisms to embed watermarks, they have had their fair share of failures upon introduction of attacks. An attacker is interested in using the trained neural network without owning it, and therefore is keen in breaking the watermarking scheme. In particular, there are a couple of different types of attacks that break such a scheme and makes the attacker free to use the model.

Attacks

The attacks on watermarking schemes can be classified on the basis of how much information the attacker has, of the model. Therefore, there are black box attacks and white box attacks. To simplify, these attacks can be grouped into Evasion

attacks [58, 59] and Model Modification attacks [4, 52, 60–62] as shown in Figure 2.3.

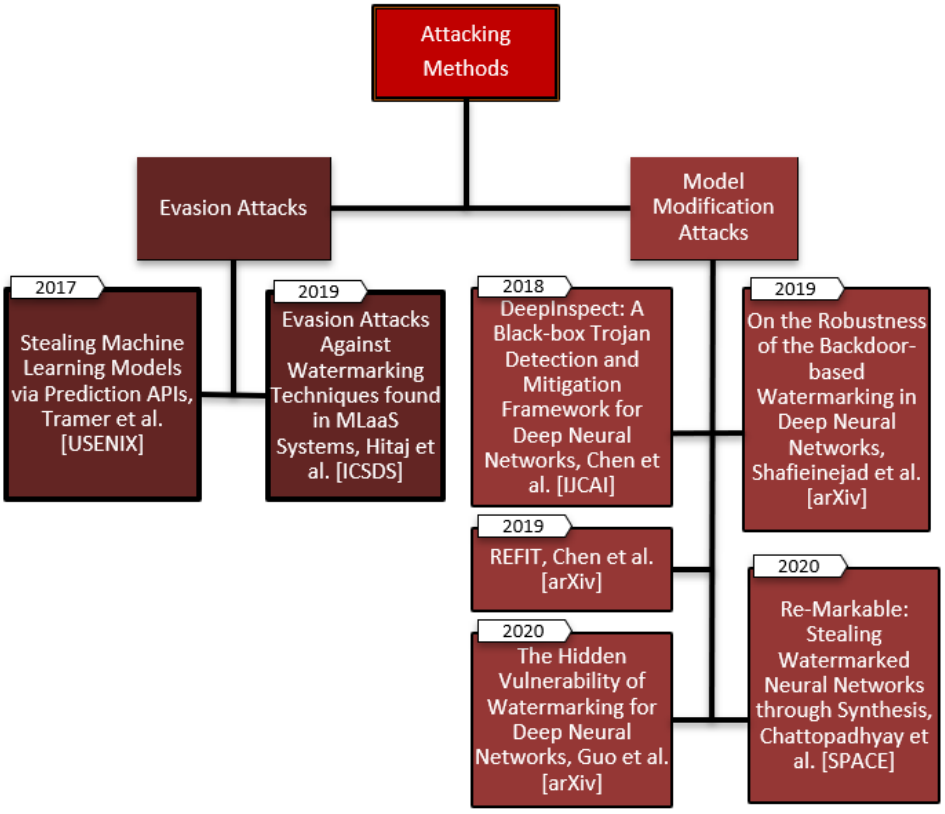


FIGURE 2.3: Literature review of attacks to break watermarking models, classified by mode of adversary action.

2.3.2 Federated Learning

Federated Learning is an ML framework which allows the participants to make use of, and train a mutually agreed upon neural network in a decentralized manner without sharing their datasets [63]. The parameters of the individually trained models however still may reveal the individual datasets and thereby breach privacy of the participants [64]. The impact of such a leak in data could vary, for some use cases the damage may be limited, but for some cases involving sensitive private data, the repercussions are more profound. Training high performance neural network models to solve complex tasks without compromising on data privacy is necessary for a wider adoption of the data driven decision making paradigm. The

issue of data privacy is complicated as there are multiple threat models that can exploit various avenues of data utilization for malicious purposes [64]. Apart from that, there is yet another separate problem regarding the hardware infrastructure demands of these neural networks and how they add to the privacy issue.

For most real life use-cases, the bulk of the computational work is offloaded to a cloud based hardware resource for effective training of the models. In a case where the participants especially do not trust each other, it makes more sense to carry out the model aggregation on the cloud [65]. But this brings in more data privacy challenges as there is a lack of trust between all parties [66]. The participants do not trust each other and the cloud infrastructure, and therefore privacy preserving learning techniques were introduced [67]. Given that there is always a potential threat of one or more participants acting in a malicious way to sabotage the entire system, the cloud based setup should also not trust the clients [68]. This motivated research in the domain of reliable federated learning infrastructures [69].

Part I

Security

Adversarial Attacks and Countermeasures

> Security is always excessive until it's not enough.
>
> *Robbie Sinclair*

Adversarial attacks prove to be a major threat to the basic functionalities of high performance neural network, making them the primary security related set of problems. Adversaries, with varying degree of knowledge about the neural architecture and trained weight matrices, can introduce structured perturbations in test samples which are able to fool the networks. They work for all types of networks and datasets.

In the first part of the thesis, we take up this issue and study the different forms of it, try to analyze its contributing factors and understand its root cause. We use this learning to propose countermeasures to prevent further attacks. From the implementation perspective, we consider different kinds of machine learning tasks, and correspondingly different models and datasets. Since adversarial attacks have been studied separately in computer vision and natural language processing problems in the literature, we maintain that distinction and consider them separately, whilst using the same basic fundamental notions for developing countermeasures, illustrated through the differing use-cases.

Chapter 3

Curse of Dimensionality in Adversarial Examples

3.1 Introduction

[1] In the recent past, there has been a steady growth in the proliferation of machine learning as a paradigm, spearheaded by advancements in the field of deep learning, with neural network architectures reaching human accuracy level, and in some specific tasks, achieving super-human performances [16]. More and more areas of science and technology have adopted the data driven way, giving rise to increasing enthusiasm in AI and related applications, among the industry and academia alike. Traditionally, machine learning or statistical learning methods involved a considerable amount of data pre-processing and feature engineering, and the hand-crafted data was provided to the models [70]. Arguably, this lacked generalization to some extent, and model performances in terms of accuracy measures were also reaching a saturation level.

Deep learning models provided the much needed breakthrough. While the perceptron algorithm was an old one [18], and neural networks being in the scene for a long time, their true potential was realized only after the advent of the required

[1]Chapter 3 is published in part as *Nandish Chattopadhyay, Anupam Chattopadhyay, Sourav Sen Gupta and Michael Kasper, "Curse of Dimensionality in Adversarial Examples," in Proceedings of IEEE IJCNN 2019* and *Nandish Chattopadhyay, Subhrojyoti Chatterjee and Anupam Chattopadhyay, "Robustness Against Adversarial Attacks using Dimensionality," in Proceedings of SPACE 2021.*

processing capacity, in terms of hardware resources. The inherent linear nature of the neural network architectures, arising from the sequences of linear transformations, was capitalized heavily to parallelize operations among multiple GPU cores, therefore speeding up the training process using backpropagation algorithm. These deep networks, with their massive set of tuning parameters, possess enough learning capacity to learn features and patterns in the data, to extents which were not reachable before.

3.1.1 Contribution : Curse of Dimensionality

This work studies the effect of data dimensionality in case of adversarial examples, and strongly second the connection proposed by Gilmer et al. [34]. In fact, this work goes one step further, and hypothesize that adversarial examples are *easier* to generate on a dataset with *higher* dimension. This chapter explores some interesting and relevant properties of the high dimensional space and present our empirical study on some models and datasets to test our hypothesis.

In particular, this chapter addresses the issue of why it is easy to generate such adversarial examples, how that behaves with different dimensionalities of input feature vectors, and why it is difficult to measure adversarial perturbation at high dimensions using the standard distance measures like L_1 and L_2 norms. This analysis is followed by an adaptation of the findings to generate an adversarial defence mechanism.

Our contributions primarily comprise of a few components:

- *Theoretical Justification*: Mathematical and statistical formulation relevant in the context of classification of high-dimensional images to justify the effects of dimensionality on the generation of adversarial examples.
- *Experimental Verification*: Empirical study, with extensive experimentation on image datasets with varying dimensions to understand the effect of dimensionality on the generation of adversarial examples.
- *Engineering Adversarial Defense*: Proposition of robust mechanism for fulfilling machine learning objectives by carrying out classification tasks and

eliminating errors arising from adversarial attacks. Specifically, the end-to-end infrastructure that we have designed is able to accomplish the reliable functionality by leveraging the following:

- A parallel pathway of simultaneously carrying out the classification task and detection of adversarial perturbations in the samples to increase efficiency
- Using dimensionality reduction to eliminate adversarial perturbation from only those samples that are detected to be adversarial examples

3.2 Adversarial Examples in Machine Learning

With the evolution of standard machine learning into the realm of deep neural networks, the task which has seen maximum progress is supervised image classification [71]. Ironically, adversarial attacks have been first observed for images as well. Given a image classifier model, for a correctly classified test data point, a corresponding adversarial example would be a hand-crafted image created by introducing a very little perturbation imperceptible to the human eye, which would be wrongly classified by the same model. This notion holds good for any machine learning system.

Typically, an image classification system would have four key components. First, is the model. It could be some neural network architecture or some other machine learning model like a maximum margin classifier (SVM) [72]. Second, there should be a training dataset, which is of course a sample drawn from the population of all imaginable images of a particular selected resolution. The training dataset is labelled for the purpose of supervised learning. Third, there is a test dataset, which is part of the population that doesn't belong to the training set. And finally, after the model is trained on the training dataset, this work obtains the trained manifolds for the different classes, which are separated by the classifier [73].

As an illustration of how adversarial examples are generated, let us consider a relatively simple binary classification problem. Figure 3.1 shows a 2D representational projection of the setup. The actual truth is the real world around us. In the context of this problem, that is, for the specific classification task, this chapter sets the notion of the *population*, which is exhaustive in nature, and is simply a

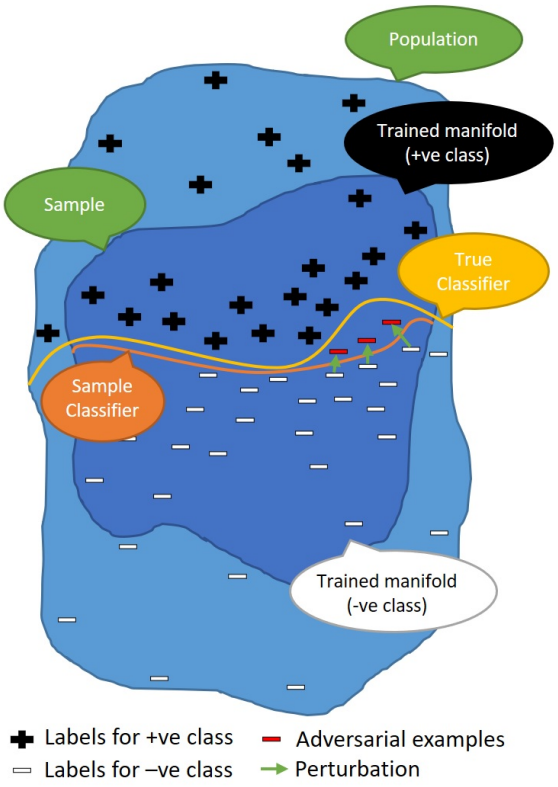


FIGURE 3.1: Adversarial Examples in context of a Classifier.

3D to 2D approximation of the real world. One may note that it is not possible to obtain its realization. For the task of image classification, as mentioned earlier, the population is the set of all images of the specific set resolution in which the images are being captured. It is assumed that there exists a *true classifier*, with respect to the population, that correctly classifies all images. A decent manifestation of this true classifier would be the human vision system. Had this classification model been known, there would not have been any requirement for machine learning.

In an attempt to approximate the true classifier, one has to consider a sample from the population, and try to build models to classify the sample space into trained manifolds corresponding to the classes. The so-developed classifier is the *sample classifier*. The sample space could be realized as a vast collection of data points, a dataset of images that is, for image classification. The sample classifier would typically be a deep neural network or a support vector machine (SVM) or some other hyperplane classifier. For simplicity of nomenclature this chapter assumes that the true classifier for the population is indistinguishable from the true classifier

within the sample, although it may so happen in practice, that the restriction of the classifier to the sample is dependent on the sample itself.

As evident from Figure 3.1, there might naturally exist a "gap" between the two classifiers or separating hyperplanes, arising due to the approximation, made on an in-exhaustive sample. This gap results in generating an *adversarial space*, wherein any point would naturally belong to two different manifolds, with respect to the two aforementioned classifiers, and therefore be *adversarial* in nature. It is interesting to note that if the points are close to the boundary of the individual trained manifolds, a little perturbation may shift some of the points across the sample classifier into the adversarial space, wherein they will be misclassified by the model, but not by the human annotator (true classifier). If the perturbation is too much, then the data point may move across the true classifier as well. That is, if the perturbation is greater than a particular threshold, the human annotator will misclassify it too, and that wouldn't be an adversarial example any more.

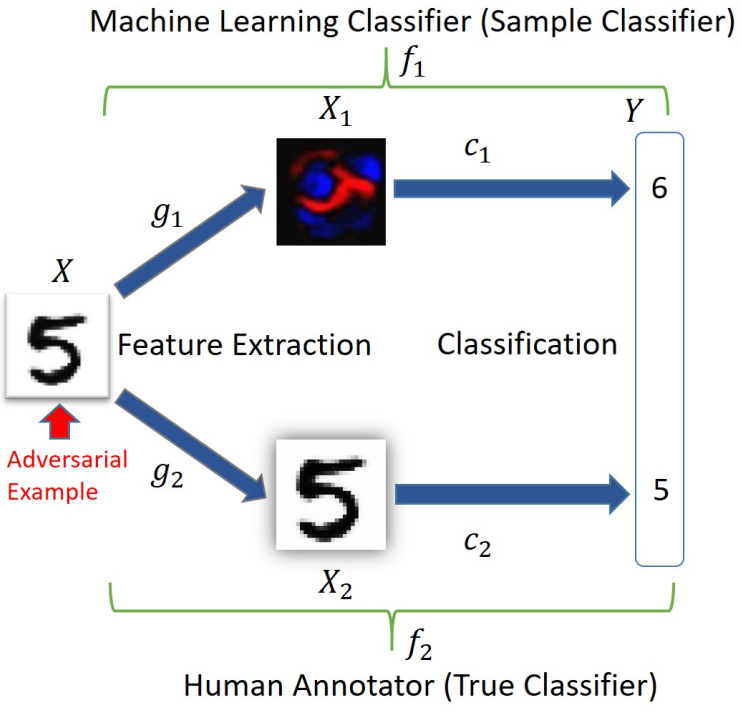


FIGURE 3.2: Generation of Adversarial Examples.

X is the space of input samples. For images as input to the system, X would trivially be the vector space of dimensions equaling the number of pixels in the image. The two classifiers that are being considered here are f_1 (sample classifier)

and f_2 (human annotator). The classifiers have two components each, feature extraction and classification. X_1 is the feature space for the sample classifier and X_2 is the feature space for the human annotator, where d_1 and d_2 are norms defined in the spaces X_1 and X_2 respectively. As shown, $f_1 = c_1 \circ g_1$ and $f_2 = c_2 \circ g_2$

Let us consider $x \in X$, a training sample. Given x, the corresponding adversarial example x^*, for a norm d_2 defined on the space X_2, and a predefined threshold $\delta > 0$, satisfies:

$$\begin{aligned} f_1(x) \neq f_1(x^*) \quad &\text{and} \quad f_2(x) = f_2(x^*) \\ \text{such that} \quad &d_2(g_2(x), g_2(x^*)) < \delta \end{aligned}$$

There are two key takeaways from the above formulation. First, the generation of adversarial examples with little perturbation is facilitated by the existence of many data points near the boundary of the trained manifolds of the classes, and second, that the adversarial perturbation is bounded in practice.

3.3 Theoretical Formulation

In this section, the concentration is on the core mathematical and statistical properties of two crucial ingredients in understanding adversarial examples – the trained manifolds of a classifier, and the measure for adversarial perturbation.

3.3.1 The Trained Manifolds

Uniform Distribution: Let us consider that the points are uniformly distributed within the manifold. If one considers a unit ball in d dimensions A, from Eq. (**??**), it follows that at least a $1 - e^{-\epsilon d}$ fraction of the volume of the unit ball is concentrated in $A \setminus (1 - \epsilon)A$, which means in a small annulus of width ϵ at the boundary. It can be mentioned here that, most of the volume of the d-dimensional unit ball is contained in an annulus of width $O(1/d)$ near the boundary. Generalizing to a d-dimensional ball with radius r, the width of the annulus would be $O(r/d)$. Another very interesting fact about the ball in high dimensions is that most of its volume is concentrated near the equator. One can note that for any

unit vector defining "north", most of the unit ball's volume lies in the thin slab of points which have a dot-product with that vector in the magnitude of $O(1/\sqrt{d})$. In particular, it can be shown that at least a $1 - \frac{2}{c}e^{-c^2/2}$ fraction of the volume of the d-dimensional unit ball has $|x_1| \leq \frac{c}{\sqrt{d-1}}$, as shown in Figure 3.3, for any $c \geq 1$ and $d \geq 3$. Thus, if one considers the d-dimensional ball as the geometry of the manifold (which minimizes surface area), most of the points will be near the boundary.

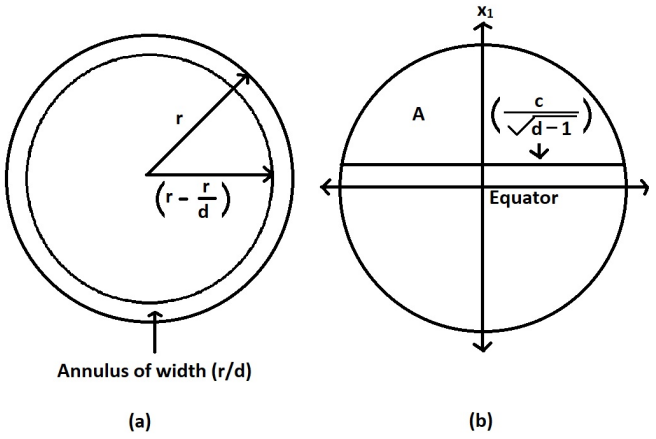


FIGURE 3.3: Volume Distribution in a Uniform d-Dimensional Ball.

It may also be noted that apart from the uniform d-dimensional ball, for any other arbitrary geometry of the manifold, the surface area would only be greater. Thus, under the assumption of uniformly distributed data points, the points will be nearer to the boundary of the manifold, thereby facilitating the generation of adversarial examples.

GAUSSIAN DISTRIBUTION: The typical 1-dimensional Gaussian has most of its mass near its mean (say, the origin). But at high-dimensions, like in the case of the manifolds in discussion, the behaviour changes significantly. The *Gaussian Annulus Theorem* [74] states that for the d-dimensional spherical Gaussian with unit variance in each direction, for any $\beta \leq \sqrt{d}$, all but at most $3e^{-c\beta^2}$ of the probability mass lies within the annulus $\sqrt{d} - \beta \leq |x| \leq \sqrt{d} + \beta$, where c is a fixed positive constant. Though the density is maximum at the origin, there is very little volume there. Nearly all of the probability is concentrated in a thin annulus of width $O(1)$ at a radius of $\sqrt{d}$. Thus, even under the assumption of Gaussian distribution of data points, the points will mostly be nearer to the boundary of the manifold, thereby facilitating the generation of adversarial examples.

Intuition: There is not much theoretical treatment of behaviour of other distributions at high-dimensions. However, though not complete enough to establish causality rigorously, from what is seen, it is fair to derive the intuition that the concentration of data points near the boundaries or surfaces of the trained manifolds, away from the respective centres, irrespective of exact distribution assumption, is a key ingredient in the process of adversarial behaviour of slightly perturbed data points, which are misclassified by the model. It is therefore natural to check whether this notion holds good in practice or not, by using adversarial attacks on models trained on different datasets of varying dimensions. This chapter reports the empirical findings in Section 8.7, later in this work.

3.3.2 Measuring Adversarial Perturbation

The other important aspect about adversarial attacks is the measurement of perturbation. As evident from the formal definition of adversarial examples in Eq. (6.1), the structured perturbation introduced to a data point to convert it into an adversarial example, is bounded by a small quantity δ, measured with respect to a norm defined in that space. This work is studying the relevance of dimensionality in the context of adversarial attacks, and it is therefore fair to be interested in knowing how the extent of perturbation vary with different dimensions of the trained manifolds in which the data points are in. In the previous subsection, this chapter presents the theoretical basis behind the ease of generation of adversarial examples, when operating in high-dimensions. Quite naturally, one should be keen to look at the correlation (if any) between the variation of perturbation and varying dimensions. Intuitively, one might expect that as one moves to higher and higher dimensions, with the ease of generation of the attacks, the perturbations might decrease. But, this is difficult to test, in practice. The reason behind that, lies in the failure of standard metrics of distance, like the L_p class of norms, at high dimensions [75]. Ironically, this is also an effect of the curse of dimensionality. This work argues that L_p norms are unable to clearly discriminate between data points due to their inability to work as a measure of distance at very high dimensions.

L_2 Norm at High Dimensions: To begin with, let us make an assumption on the distribution of the data points for simplicity. If a number of random points are generated in a d-dimensional space using a Gaussian distribution, then the distance

between all pairs of points will essentially be the *same* when d is sufficiently large. Consider the L_2 norm, the square of the distance between two data points, that is, two d-dimensional images in our case. Let the two images be the original test image y, and the corresponding adversarial example, z. Their distance, in L_2 norm, satisfies

$$\|y - z\|_2^2 \;=\; \sum_{i=1}^{d} (y_i - z_i)^2, \tag{3.1}$$

which can be seen as the sum of d independent samples of some random variable x that is essentially a distribution of the square of difference of two Gaussian distributions. More precisely, the square of the distance is the sum of independent samples $x_i = (y_i - z_i)^2$ of some random variable x of bounded variance. The Law of Large Numbers [76] dictates that with high probability, the average of the samples are close to the expectation of the particular random variable. Formally, it states that, one has $x_1, x_2, x_3, \ldots, x_n$ to be n independent samples of any random variable x with finite variance, then

$$\Pr\left(\left|\frac{x_1 + x_2 + \ldots + x_n}{n} - E(x)\right| \geq \epsilon\right) \leq \frac{\mathrm{Var}(x)}{n\epsilon^2} \tag{3.2}$$

In the d-dimensional space, $n \approx d$ is sufficiently large for this particular convergence of the notional 'distance metric' random variable to the expectation of the random variable. This provides the basis for the failure of distance metric L_2.

L_p Norm at High Dimensions: A more generic result, on the failure of the L_p class of norms in its ability to act a metric of distance and perform the basic task of discriminating between the nearest and the furthest points, at high dimensions, is presented below [77]. Note that the Law of Large Numbers is more generic and hence one can relax the previous Gaussian assumption on the distribution.

Let us consider a d-dimensional data space, as earlier. Let N be the number of data points, F be a 1-dimensional data distribution in $(0, 1)$ and X_d be a particular data point from F^d with each coordinate drawn from F. In the context of images as the data points, X_d would be a d-dimensional image (the dimension being fixed based on resolution) with scaled pixel values. Let the distance between $(x^1, \ldots, x^d)$ and $(y^1, \ldots, y^d)$ using the distance metric $L_p = \{\sum_{i=1}^{d} |x_1^i - x_2^i|^p\}^{1/p}$ be denoted as $dist_d^p(x, y)$. Let $||\cdot||_p$ be the distance of a vector to the origin (our reference point

for analysis) using the function $\text{dist}_d^p(\cdot, \cdot)$. Let $\text{E}[x]$ and $\text{var}[X]$ be the expected value and the variance of the random variable X.

Define $D_{max}^{p,d} = \max\{||X_d||_p\}$ and $D_{min}^{p,d} = \min\{||X_d||_p\}$. Under the assumption that the distribution behaves a certain way as d increases, and therefore:

$$\lim_{d\to\infty} \text{var}\left(\frac{||X_d||_p}{E[||X_d||_p]}\right) = 0 \quad \Rightarrow \quad \frac{D_{max}^{p,d} - D_{min}^{p,d}}{D_{min}^{p,d}} \xrightarrow{p} 0,$$

where the probabilistic convergence $Z_d \xrightarrow{p} c$ means that a sequence of vectors $Z_1, \ldots, Z_d$ converges in probability to a constant vector c if for all $\epsilon > 0$, $\lim_{d\to\infty} \Pr[\text{dist}_d(Z_d, c) \leq \epsilon] = 1$.

This gives a clear indication that at high-dimensions, the L_p norm is unable to clearly discriminate between two data points (test image and its corresponding adversarial example, in this context), which are nearest and furthest from a reference point (origin of the manifold, in this case).

This is an observation that we have made from the experiments in Section 8.7, which has also been substantiated from the theoretical perspective as mentioned earlier and therefore requires specific attention and investigation. Ideally, dedicated metrics like semi-norms may be used for this purpose as an alternative to using standard L_p norms.

3.4 Experiments and Observations

In order to make the assertion that the "generation of adversarial examples is easier in higher dimensions", this work wishes to show that under otherwise similar experimental setup, the adversarial attacks work better at situations which involve higher dimensions of the input feature vectors.

3.4.1 Experimental Setup

MODEL: this work used a neural network model, some standard datasets in the context of image classification and two popular adversarial attack methods. The

deep convolutional neural network that has been used is the typical implementation of the VGG network [1], which was proposed by the Visual Geometry Group at University of Oxford. The network has four feature extraction blocks comprising of convolution and pooling layers, followed by a multi-layer perceptron of three fully connected layers. The first three blocks designed for the purpose of learning the features include two 2-dimensional convolutional layers, followed by a maximum-pooling layer. The fourth block consists of one convolutional layer and one maximum-pooling layer. This is followed by the dense layers, which is a fully connected multi-layer perceptron with three linear layers. Little modifications were made from time to time, depending upon the dataset being used.

Dataset: Three datasets were used, which are widely used in the study of image classification tasks. The first one is the MNIST dataset [5], a collection of images of handwritten digits. Each image has a resolution of 28×28 pixels, and are gray-scale in nature with 10 classes corresponding to the 10 digits, and therefore is quite a low-dimensional feature vector (784 pixels) in the context of our analysis. The second dataset is the CIFAR-10 dataset [6], which is a collection of coloured images of everyday objects. Each image has a resolution of 32×32 pixels, across the three fundamental colours (red, green, blue) and is a comparatively higher dimensional input feature vector (3072 pixels). Here also, there are 10 classes. The third dataset is the ImageNet dataset containing the 1000 categories and 1.2 million images, each of which are 224×224 pixels in size across RGB. It is worth noting here, that to conduct extensive tests to understand the behaviour of the models at different dimensions, the experiments are designed to modify the datasets as required and this is explained in further details in the next segments.

Adversarial Method: The two methods of introducing perturbations into images to create adversarial examples, used in this analysis, are the Fast Gradient Sign method and the Momentum Iterative Gradient Fast Sign method. The Fast Gradient Sign method was first introduced by Goodfellow et al. [2] in 2014. This is an effective adversarial generation technique, that uses L_∞-norm as the distance measure d_2 in Eq. (6.1). For this adversarial attack method, the natural choice is to make the attack strength the same at every feature dimension. The perturbation r to be introduced to a clean sample x, to turn it into an adversarial one, is the solution of:

$$\arg\min_r (c\ d_2(x, x+r) - \text{Loss}(f_1(x+r), f_1(x))), \tag{3.3}$$

where $x + r \in [0, 1]^p$, with p being the total number of features and c being the constant for the Lagrange multiplier. In other words, the adversarial example x^* can be obtained from a clean sample x by maximizing the loss function $J(x^*, y)$, where J is usually cross-entropy loss and y is the class label. The Fast Sign Gradient method satisfies the L_∞ norm bound of $||x^* - x||_\infty \leq \epsilon$ and is therefore obtained as:

$$x^* = x + \epsilon \cdot sign(\nabla_x J(x, y)), \tag{3.4}$$

where $\nabla_x J(x, y)$ is the gradient of the loss function w.r.t x.

Apart from this one-step gradient based approach, it is also possible to create adversarial attacks in an iterative way, using the similar concept of attack. These methods typically apply the fast sign gradient many times, with a small step size α. An iterative version of FGSM is:

$$x_0^* = x, \quad x_{t+1}^* = x_t^* + \alpha \cdot sign(\nabla_x J(x_t^*, y)) \tag{3.5}$$

In order to ensure that the adversarial examples generated in this process are restricted within the ϵ bound with respect to the L_2 or L_∞ norm, one could clip X_t^* in the ϵ neighborhood of x or set $\alpha = \epsilon/T$, with T being the number of iterations.

The Momentum Iterative Fast Gradient method was introduced by Dong et al. [35], in 2018 and belongs to this class of attacks. The momentum method in general is a technique for accelerating gradient descent algorithms by accumulating a velocity vector if the gradient direction of the loss function across the iterations. The memorization of the previous gradients helps to traverse efficiently though the landscape. This idea is used to generate adversarial examples efficiently.

3.4.2 Experimental Design

In an attempt to support the hypothesis discussed earlier, extensive experiments were carried out to understand the behaviour of adversarial attacks with dimensionality. For each of the individual experimental settings, the performance metric for the models and the procedural scheme were kept constant, to facilitate comparability wherever needed. The measure of performance used throughout, is the classification accuracy (used synonymously as performance hereafter), that is the

percentage of correct classifications made by a particular model on a particular set of samples, be it the train-set, test-set or a set of adversarial examples created by some attack methods (each combination is referred to as a setting).

The procedure adopted for the analysis on any given setting is three-fold. Firstly, the neural network is trained on a training dataset, and the hyper-parameters are tuned to obtain the best in class performance. Thereafter, the model is put to test against the test dataset, and the performance is noted. Finally, the same test dataset is used to create the adversarial examples, with respect to the trained model (using the gradients) using the two attack mechanisms mentioned earlier [78]. It must be noted that the hyper-parameters of the attack mechanisms are kept fixed throughout, for comparability. The tuned neural network model is then tested against the artificially created adversarial examples set. The performance of the model is recorded. In a given experimental setting, the comparison of the trained model's performance on the particular test-set and its corresponding adversarial examples set would give us an indication of whether the setting is favorable or not for adversarial attacks. The datasets used have been mentioned earlier, the hand-written character recognition MNIST dataset, the CIFAR-10 dataset and the ImageNet dataset. A few different kinds of studies for the analysis are carried out, as follows.

Varying Dimensionality: To understand whether or not greater number of features and higher inherent dimensionality of the datasets affect the generation of adversarial examples from a particular sample of data points, as is the hypothesis, this work carried out the three-fold procedure on three datasets of very different individual size of features. The comparison of performance of the model should therefore be indicative of empirical results to substantiate the claimed hypothesis.

Up-Sampled Resolution: To understand whether or not the raw resolution of an input image has any impact on the generation of adversarial examples, this work experimented with upsampling the resolution of images without adding any new information, that is without changing the inherent dimensionality of the dataset. This was carried out using a nearest neighbor interpolation approach of the pixels in the image, by doubling, quadrupling the resolution of the images. Following that, the same three-fold procedure was carried out on the images.

Dimension Reduction: To understand whether or not the inherent dimensionality of the dataset has an impact on the creation of adversarial examples, this work experimented with reducing the dimension of the respective datasets, using standard techniques like the singular value decomposition. While accomplishing this task, this work created four different settings, that is combinations of train-set, test-set and adversarial set, from the original dataset, which contained 99%, 95%, 90% and 80% of the actual information. One may note here that, for singular value decomposition, the information content of each of the data slices or components corresponding to each of the singular values can be obtained from its surrogate, the proportion of the singular values themselves. Following this decomposition, this part carried out the same three-fold process.

Measurement of Distance: In order to understand whether the standard metrics of distances, like the L_2 norm, can be used as a good metric of the notion of distance in the high-dimensional spaces, some experiments were done. this work studied the distribution of the individual pair-wise distances between the images belonging to a particular class. This also looked at the distribution of the adversarial perturbation, meaning the pair-wise distances between the clean samples and the adversarial samples. Finally, it obtained a superimposed plot of the distributions to look at the efficacy of the distance measure in question, the L_2 norm.

3.4.3 Experimental Results

In this section, the observations of the various experiments mentioned in the previous subsections are presented. For each of the settings, which is a combination of a train-set, a test-set and two adversarial sets created using two types of attack methods Fast Gradient Sign method (FGSM) and the Momentum Iterative Fast Gradient Sign method (MI-FGSM), this work has reported performances of the trained model.

Table 3.1 shows the performances of the individually trained neural network on three datasets of differing sizes. For each dataset, the corresponding number of pixels in each image is provided within parenthesis. The performance of the trained model is tested on the test set and the two adversarial datasets, created using the two types of attacks mentioned above. It is clearly evident that with the growing

inherent dimensionality of the datasets, the performance of the neural network on the adversarial datasets reduces significantly. That is, adversarial examples are generated better for higher dimensionality.

TABLE 3.1: Adversarial Attacks and Dimensionality.

Dataset	Performance on Test Set	Performance on Adversarial Set (FGSM)	Performance on Adversarial Set (MI-FGSM)
MNIST (28x28)	98.8%	55.8%	51.7%
CIFAR-10 (3x32x32)	84.6%	14.2%	13.8%
ImageNet (3x224x224)	78.2	9.5%	6.8%

Table 3.2 shows how the generation of adversarial examples is affected by up-sampling the resolution of the images. Like earlier, for each of the datasets, the number of pixels in each of the images is provided in parenthesis. The performances of the individually trained models are presented corresponding to each of the settings mentioned earlier, on the test set and the two adversarial datasets, created using the two types of attacks mentioned above. From the results of the experiments, one can see that for the two datasets used, with three variants for each (in terms of resolution), the performance of the attacks observed, was very close to each other consistently for both the datasets. The conclusion that follows this observation is that there is no significant effect of changing the mere resolution of the images (without adding any new information, therefore not affecting the inherent dimensionality of the data) on the performance of the neural network on the adversarial dataset.

Table 3.3 presents the observations in the study of generating adversarial examples from the dimension reduced versions of the datasets, as explained in the earlier segment. For each of the datasets, only those components are considered which make up for a particular proportion of the overall information, as indicated in parenthesis in the table. The performance of the trained model is tested on the test set and the two adversarial datasets, created using the two types of attacks mentioned above. One can observe from the empirical findings that with dimension reduction, the performance of the neural network in both the datasets have improved, indicating that adversarial attacks become less effective with dimension reduction.

TABLE 3.2: Adversarial Attacks on Up-Sampled Images.

Dataset	Performance on Test Set	Performance on Adversarial Set (FGSM)	Performance on Adversarial Set (MI-FGSM)
MNIST (28x28)	98.8%	55.8%	51.7%
MNIST (56x56)	98.6%	55.8%	51.5%
MNIST (112x112)	98.8%	55.6%	51.7%
CIFAR-10 (3x32x32)	84.6%	14.2%	13.8%
CIFAR-10 (3x64x64)	85.1%	13.9%	13.8%
CIFAR-10 (3x128x128)	84.9%	14.1%	13.7%

Figure 7.3 shows the distributions of distances superimposed with the distribution of adversarial perturbation (plotted in blue). For this study, this work considered the CIFAR-10 dataset, which has 3072 pixels per image. We plotted the distribution of the pair-wise L_2 norm distances of all points belonging to a particular class

TABLE 3.3: Adversarial Attacks and Dimension Reduction.

Dataset	Performance on Test Set	Performance on Adversarial Set (FGSM)	Performance on Adversarial Set (MI-FGSM)
MNIST (99%)	99%	56.1%	54.3%
MNIST (95%)	97.8%	58.3%	57.1%
MNIST (90%)	97.5%	59.1%	57.5%
MNIST (80%)	96.8%	64.2%	61.1%
CIFAR-10 (99%)	83.2%	14.2%	13.7%
CIFAR-10 (95%)	80.1%	15.9%	14.8%
CIFAR-10 (90%)	77.3%	17.8%	15.2%
CIFAR-10 (80%)	74.8%	19.8%	17.6%

(L_2 norm distance in the X-axis and the probabilities in the Y-axis) and then repeated the process for all the ten classes in the dataset, shown in different colours. Then, superimposed that plot with another plot of the distribution of adversarial perturbation, that is the pair-wise L_2 norm distances between clean samples and their corresponding adversarial samples, shown in blue. From the figure, one can observe that the distribution of the pair-wise distances of the images within the classes are highly overlapping, and is centred around the value of $\sqrt{d}$ (where d is the number of pixels). Interestingly, the plot in blue, which is the distribution of the adversarial perturbations, and is also found to have a peak at around $\sqrt{d}$ (numerically around 56). This indicates that as suggested in the earlier sections, the distance metric fails to provide meaningful measures at high dimensions as all measures of distance tend to converge to the numeric value of the square root of the dimensionality.

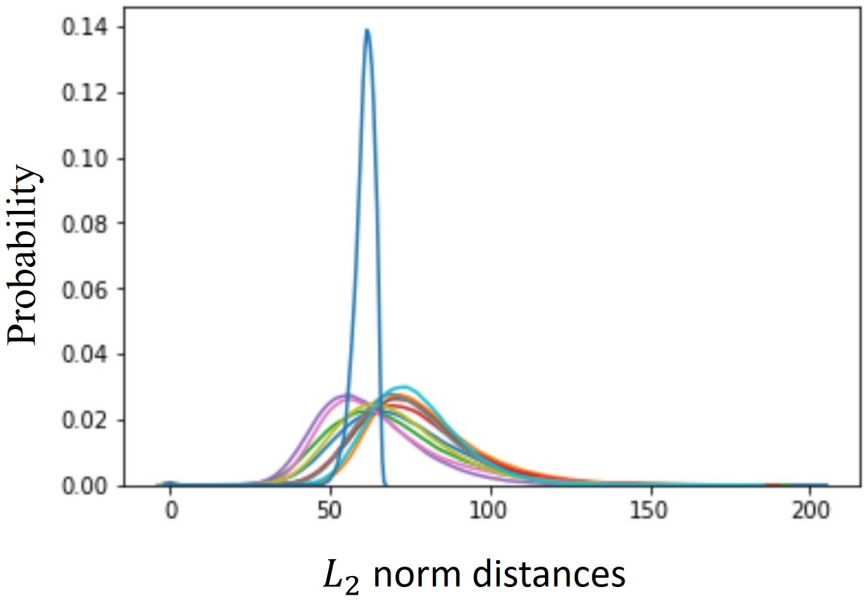


FIGURE 3.4: The Distribution of Pairwise Distances.

3.5 Defence Design

As discussed in the earlier sections, higher dimensionality facilitates adversarial example generation, and it is natural to try reducing dimensionality of the test

samples as a defence. This may be effective in reducing, and in some cases eliminating adversarial perturbations. There is a sound theoretical explanation for the empirical results seen in experimental results of using techniques of reducing dimensionality as an adversarial defence. However, there are some issues which prevent it from being applied universally as a filter to deter adversarial attacks. They are as follows.

- Dimensionality reduction is not targeted, and therefore may reduce useful information from the samples along with adversarial noise. While being useful to eliminate adversarial noise, this process may at times make the original machine learning task (classification for example) difficult for a trained model. How much variability to preserve in the samples is therefore a critical design decision.
- It is also an unnecessary overhead for samples which are clean, and may slow down a system where there are many samples to deal with.

We seek to address this, by intelligently selecting which samples to expose to dimensionality reduction and which to remain as is, before sending them as inputs to the machine learning model. In particular, we adopt a design of parallel pathways, to simultaneously carry out the inference task and identify potential adversarial samples and apply dimensionality reduction to only those which have been picked up to contain adversarial noise. Our only assumption for choosing the parallel setup, as opposed to a serial setup is that most samples are clean, with some being adversarial.

3.5.1 Parallel Pathways

We consider a particular machine learning task as the primary objective, like classification. This is a trained neural network that is vulnerable to adversarial attacks. We design another classifier, that is specifically trained to identify samples that contain adversarial noise. The overall system is so designed that the input samples are fed to the two networks, in parallel. One network carries out the actual task (classification for instance) and we call that the Main Model. The other network determines whether the sample is clean or adversarial, and we call it the Adversarial Detector. If the sample is found to be clean, the output of the Main model

is mapped to the output of the system. If the sample is found to be adversarial (that is it contains adversarial noise as identified by the Adversarial Detector), then the Dimensionality Reduction loop is activated. The sample is then fed to the Dimensionality Reducer and its result is further sent to the Main Model for carrying out the task (classification) again.

This design choice has been made keeping the following in mind:

- Adversarial samples, crafted by adversaries are fewer in number than naturally occurring clean samples. The system is likely to encounter many more clean samples and therefore not require dimensionality reduction.
- Since the dimensionality reduction loop will be activated for fewer adversarial samples, for most clean samples the throughput of the system will be comparable to what it would have been, without any defence mechanisms.
- By checking for the presence of adversarial perturbations and only then reducing dimension, we are ensuring that the model performance for clean samples is not hampered.

3.5.2 Detecting Adversarial samples

Adversarial examples contain structured noise inserted as perturbation by attackers to fool the trained neural networks. This perturbations are bounded, because they are by design, imperceptible to human vision. The misclassification or simply mistake in the machine learning task by the model as opposed to correct classification by the human annotator leads to the discrepancy in class labels that generate adversarial attacks. In this work, we re-use the neural architecture that falls prey to adversarial attacks, by training it to differentiate between clean samples and adversarial samples. The intuition is that the model that picks up the adversarial noise to lead to an error, will also be able to detect that same noise, if it is trained to do so. We have generated pairs of adversarial samples for corresponding clean samples, and trained the network as a binary classifier. The sensitivity of the main model towards the attack is used in this classifier to identify adversarial samples. The experimental results later on vindicate the proposition.

3.5.3 Dimensionality

As a manifestation of the theoretical setup mentioned in the earlier section, high dimensionality of the optimizing landscape of the neural network classifier makes it vulnerable to adversarial attacks. It has been shown that the sensitivity of the model to adversarial perturbation grows with higher dimensions. As a natural extension of that idea, we propose to use dimension reduction to defend against such attacks. Essentially, we select the samples that have been detected to contain adversarial perturbation and then project them onto a lower dimension, eliminating a certain amount of information (measured through variability) to convert them into clean samples. The intuition is that such samples, with cropped information, will be classified to the correct classes when subjected to the Main Model. We have tested this idea through a series of thorough experiments.

3.6 Implementation

The adversarial defence mechanism has been implemented as an end-to-end system, that takes in samples, both clean and adversarial ones alike, and outputs the correct probability distribution over the classes with high accuracy. Internally, it is able to detect adversarial samples using a separate adversarial detector, and thereby activates a dimension reduction loop to eliminate noise that pertains to the attack. The implementation of the framework is shown in Figure 5.1.

3.6.1 Pipeline

The pipeline is set up using parallel pathways for the primary machine learning task of classification and the additional task of detecting adversarial samples. This is in accordance with the notion described in Section 4.3.2. The input samples are sent in parallel to both the networks, the Main Model and the Adversarial Detector. We have tested two different neural architectures for both these tasks. The Main Model generates the probability distribution over the classes and the Adversarial Detector labels the samples as clean or adversarial. If the sample is found to be clean, then the corresponding channel is enabled so as to map the output of the Main Model to the final output of the system. If the sample is found

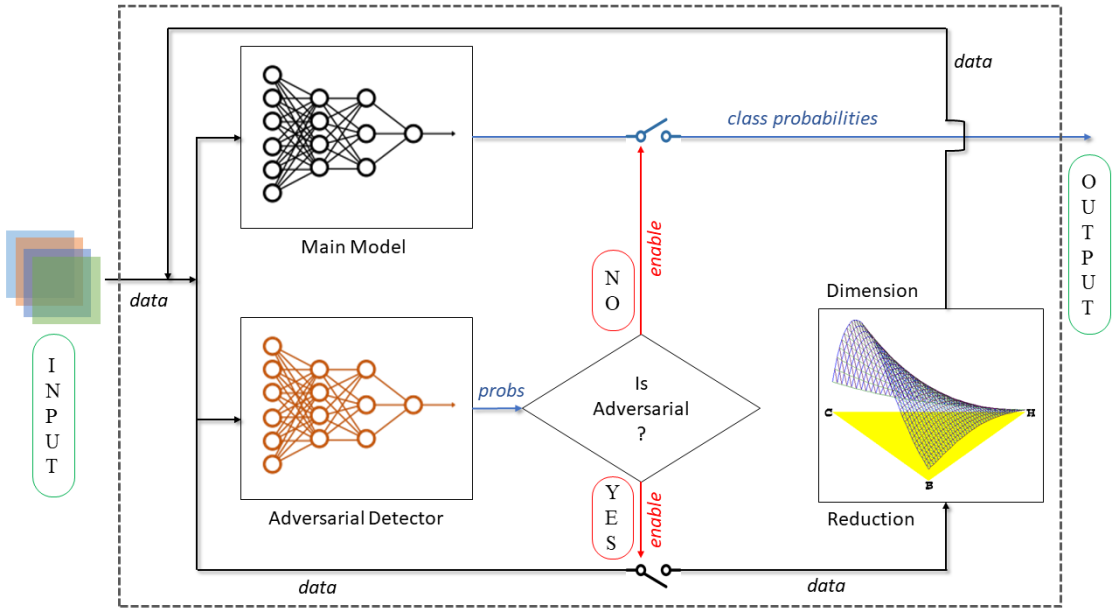


FIGURE 3.5: Implementation of the framework pipeline. The flow of data samples is shown in black channels, the flow of output class probabilities is shown in blue channels and the flow of enable control is shown in red.

to be adversarial, then the other channel is enabled, which feeds the input sample to the Dimension Reduction block. This is where the sample is projected onto a lower dimension based on a tuning parameter, and then its output is again fed to the input of the overall system. This process ensures that the Main Model's predicted class probabilities are mapped to the final output only if the sample is deemed to be a clean one. The entire flow of data and control is shown in the diagram.

3.6.2 Models

In order to check if the proposed infrastructure's robustness against adversarial attacks is agnostic to any specific neural architecture or adversarial attack, we used multiple combinations of them and checked for performance. In particular, we used two state-of-the-art neural architectures (VGG-19 [1], Inception ResNet v2 [12]), two most widely used and effective modes of adversarial attacks (FGSM [2] and PGD [13]) and some benchmarking datasets (MNIST[5], CIFAR-10 and CIFAR-100 [6]) for making the results to comparable to the available literature.

3.6.3 Dimension Reduction

The process of transformation of data from a high dimensional space to a low dimensional space is carried out using dimensionality reduction algorithms. They ensure that the representation so created in the low dimensional setting, is able to retain as much meaningful properties as possible, of the data in the original high dimensional setting. This is particularly necessary for a variety of reasons, like avoiding the issues that arise due to the curse of dimensionality and other unfavourable behaviours of the high dimensional space. Reducing the dimension makes computation and analysis easier.

There are different ways of carrying out dimension reduction. Typically, techniques like principal components analysis or singular value decomposition [79] are used to reduce dimensions in a linear fashion, that is the new components so created are linear combinations of the original. There are some useful non-linear dimensionality reduction techniques as well, like the t-distributed stochastic neighbor embedding [80].

3.7 Experiments

In this section we discuss the experimental setup and corresponding results in details. The overall aim of the experiments is to study whether the proposed system is able to match the accuracies of the model when there are no adversarial examples, even when adversarial samples are introduced via attacks. The entire infrastructure has been built in TensorFlow [81] and the experiments were run on Google Colab with a NVIDIA Tesla K80 GPU. The attacks were taken from Google's Cleverhans [78] repository.

3.7.1 Design

We study each component of the infrastructure separately first, and then we look into the performance of the overall framework under specific circumstances. We note the accuracy of the Main Model when exposed to clean samples as the baseline (first row of the Tables 3.4, 3.5, 3.6, 3.7). This is the highest possible performance

that we may have, given the neural architecture and a specific dataset. The rest of the accuracies will be compared to this benchmark.

Next, we look at the drop in performance of the Main Model when exposed to adversarial examples (second row of the Tables 3.4, 3.5, 3.6, 3.7) . Then, we turn our attention to the Adversarial Detector. It is important to note that high accuracies of this component is extremely critical to the overall performance of the infrastructure. The error in Adversarial Detection, will be propagated to the overall output error of the system, as we apply the adversarial perturbation reduction loop to only those samples that are identified to be adversarial by this detector. Intuitively, a neural architecture that is vulnerable to picking up adversarial noise and thereby misclassifying, will also be sensitive to that exact same noise when trained as an Adversarial Detector rather than a regular classifier. We take a note of it in the third row of the Tables 3.4, 3.5, 3.6, 3.7.

In the next part of the experimental setup, we study the impact of using dimensionality reduction techniques and use two different techniques for the same. Note that only adversarial samples, detected by the Adversarial detector, are exposed to the dimensionality reduction. We report the corresponding performance of the Main Model when exposed to the projections of those samples on lower dimensions. We use two different techniques for doing the same, as shown in the fourth and fifth rows of the Tables 3.4, 3.5, 3.6, 3.7.

Finally, we check the performance of the overall framework. This is where we run the entire pipeline shown in Figure 5.1. We use different scenarios here. One, where the test samples are 80% clean samples and 20% adversarial, this being a more likely situation from a practical perspective (results shown in the sixth row of the Tables 3.4, 3.5, 3.6, 3.7). And secondly, we consider a set of test samples which are 50% clean samples and 50% adversarial (results shown in the last row of the Tables 3.4, 3.5, 3.6, 3.7). We compare these accuracies to the baseline accuracy. Additionally, we also study how much time is consumed in the inference process for the aforementioned setups in Table 3.8. This gives us a good understanding of the efficiency of the infrastructure.

3.7.2 Results

The experimental results are presented in a tabulated form here. Each table is associated with a particular combination of neural architecture and adversarial attack pair. The performances of different settings are noted, for all three datasets (MNIST [5], CIFAR-10 and CIFAR-100 [6]). The bounds on adversarial perturbation, is set like this: $\epsilon = 1.0$ for MNIST and $\epsilon = 0.1$ for CIFAR -10 and CIFAR-100.

Considering a VGG-19 [1] neural architecture and the mode of adversarial attack being the FGSM [2] attack, the step by step results are shown in Table 3.4 and Figure 3.6. We observe the drop in performance of the Main Model upon the introduction of the adversarial examples, while the Detector is able to correctly identify adversarial samples effectively. We also note that dimension reduction works well on adversarial samples to be correctly classified thereafter. In the end, the framework is able to perform almost as well (within a 1% - 2% range) as the Main Model without the existence of any adversarial samples, thus fulfilling our objective.

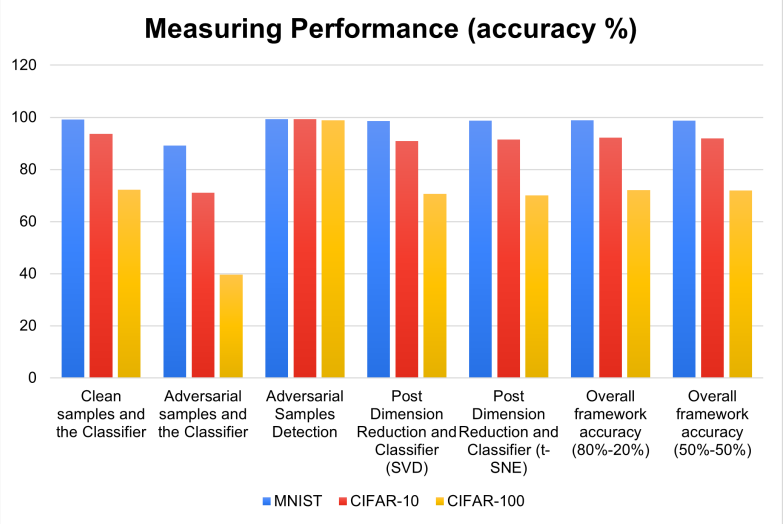


FIGURE 3.6: Performances using a VGG-19 [1] neural architecture and the mode of adversarial attack is the FGSM [2] attack.

The exact same analysis is repeated for some different choices of neural architectures and adversarial attacks. Table 3.5 uses a VGG-19 [1] neural architecture and the mode of adversarial attack is the PGD [13] attack. Table 3.6 uses a Inception ResNet V2 [12] neural architecture and the mode of adversarial attack is the FGSM

[2] attack. Table 3.7 uses a Inception ResNet V2 [12] neural architecture and the mode of adversarial attack is the PGD [13] attack.

TABLE 3.4: Neural architecture: VGG-19 and Adversarial Attack: FGSM

Experiment Design (type of samples and model)	MNIST	CIFAR-10	CIFAR-100
Clean Samples and Main Model	99.12%	93.58%	72.14%
Adversarial Samples and Main Model	89.11%	71.04%	39.61%
Adversarial Sample Detector	99.24%	99.24%	98.89%
Dimension Reduction (SVD) & Main Model	98.51%	90.87%	70.66%
Dimension Reduction (t-SNE) & Main Model	98.66%	91.52%	70.04%
Framework Accuracy (80% clean -20% adv)	98.79%	92.16%	72.05%
Framework Accuracy (50% clean -50% adv)	98.66%	91.89%	71.95%

TABLE 3.5: Neural Architecture: VGG-19 and Adversarial Attack: PGD

Experiment Design (type of samples and model)	MNIST	CIFAR-10	CIFAR-100
Clean Samples and Main Model	99.12%	93.58%	72.14%
Adversarial Samples and Main Model	87.81%	70.01%	37.89%
Adversarial Sample Detector	99.27%	99.05%	98.11%
Dimension Reduction (SVD) & Main Model	97.18%	89.93%	68.91%
Dimension Reduction (t-SNE) & Main Model	97.84%	90.17%	68.76%
Framework Accuracy (80% clean -20% adv)	98.88%	91.84%	70.88%
Framework Accuracy (50% clean -50% adv)	98.22%	91.56%	70.41%

Table 3.8 reports the amount of time taken for:

Table 3.6: Neural Architecture: Inception ResNet V2 and Adversarial Attack: FGSM

Experiment Design (type of samples and model)	MNIST	CIFAR-10	CIFAR-100
Clean Samples and Main Model	99.32%	95.42%	79.89%
Adversarial Samples and Main Model	87.43%	72.36%	41.89%
Adversarial Sample Detector	99.28%	99.18%	99.02%
Dimension Reduction (SVD) & Main Model	98.49%	91.80%	72.86%
Dimension Reduction (t-SNE) & Main Model	98.23%	92.30%	74.01%
Framework Accuracy (80% clean -20% adv)	99.01%	93.98%	78.11%
Framework Accuracy (50% clean -50% adv)	98.77%	93.15%	75.18%

Table 3.7: Neural Architecture: Inception ResNet V2 and Adversarial Attack: PGD

Experiment Design (type of samples and model)	MNIST	CIFAR-10	CIFAR-100
Clean Samples and Main Model	99.32%	95.42%	79.89%
Adversarial Samples and Main Model	86.15%	69.78%	38.45%
Adversarial Sample Detector	98.19%	98.16%	97.65%
Dimension Reduction (SVD) & Main Model	98.11%	90.41%	71.87%
Dimension Reduction (t-SNE) & Main Model	98.07%	90.59%	71.55%
Framework Accuracy (80% clean -20% adv)	98.54%	93.66%	74.09%
Framework Accuracy (50% clean -50% adv)	98.37%	92.89%	73.88%

- just the Main Model to process the samples when they are in the proportion of 80% clean samples and 20% adversarial samples, as opposed to when they are equally likely to be clean or adversarial.

- the overall framework, that is the proposed pipeline, to process the samples when they are in the proportion of 80% clean samples and 20% adversarial samples, as opposed to when they are equally likely to be clean or adversarial.

TABLE 3.8: Inference Time per sample (in ms)

Dataset Composition	Model	MNIST	CIFAR-10	CIFAR-100
80% clean samples, 20% adversarial	Main Model	3.29	3.26	3.26
	Overall Pipeline	3.91	4.05	4.09
50% clean samples, 50% adversarial	Main Model	3.29	3.26	3.26
	Overall Pipeline	4.71	4.82	4.91

3.7.3 Key Findings

As evident form the data presented above, we have some important takeaways from the extensive set of experiments that we carried out. They are noted here.

- While the introduction of adversarial samples is able to bring down the accuracy of the Main Model (the image classifier) significantly (ranging from 15%-50% decline in performance), the use of dimensionality reduction successfully prevents it.
- Dimension Reduction works very well for reducing/eliminating adversarial perturbations and proves to be a good technique for correctly classifying adversarial samples (its performance being within 2%-5% within the range of accuracy of the classifier without any adversarial samples).
- The adversarial Detector, a model specifically trained to detect adversarial samples, is extremely efficient in doing so (accurate in excess of 98% for all combinations of models, attacks and datasets) and this is crucial for the successful working of the parallel nature of the pipeline (results shown in the third row of the Tables 3.4, 3.5, 3.6, 3.7).

- Since dimensionality reduction sometimes eliminates important information from clean samples, the parallel pipeline is necessary to increase overall accuracy of the framework when the input samples are a mix of clean and adversarial samples. There seems to be not much difference in using either of the techniques for dimension reduction though, both SVD and t-SNE generate similar performances (results shown in the fourth and fifth row of the Tables 3.4, 3.5, 3.6, 3.7).

- The overall framework is robust against adversarial attacks, whilst being functionality preserving for clean samples. For a practical scenario of an input set of samples consisting of some adversarial examples within clean samples (in a 80% clean and 20% adversarial split), the framework consistently generates good performance of accuracies within the range of 1%-2% of baseline (results shown in the sixth row of the Tables 3.4, 3.5, 3.6, 3.7).

- Even for an extreme scenario of a 50%-50% split in clean and adversarial samples (this is unlikely because adversarial samples are not naturally occurring), the framework is able to generate accuracies within a 2%-4% of baseline (results shown in the last row of the Tables 3.4, 3.5, 3.6, 3.7).

- The inference time per sample is comparable for the Main Model and the overall pipeline. This indicates that the framework is able to deliver robustness without much compromise on overheads (results shown in Table 3.8).

3.8 Conclusion

In this work, the hypothesis is that generation of adversarial attacks in case of neural networks benefits from higher dimensionality in data. To support this hypothesis, this work presents the theoretical formulation of the problem, and presents the intuition in terms of high-dimensional geometry. The study also provides extensive experimental results in support of the hypothesis. It is also argued that the futility of using standard L_p norms as a distance measure in case of high-dimensional manifolds (the feature space of neural networks), especially while analyzing the generation of adversarial examples. We address this problem by developing a framework that delivers robustness against such attacks. We ensure that the framework is functionality preserving, that is it works as expected for clean samples and also

is able to correctly classify adversarial samples, therefore preventing such attacks from disrupting the machine learning task. We approach the solution by first understanding one key property of neural networks, that is their vulnerability to attacks due to the high dimensional nature of the optimisation landscape. We use dimensionality reduction to tackle this problem, and aim to reduce/eliminate adversarial noise. However, we do not disturb the clean samples and therefore use a parallel pathway to detect adversarial samples and expose only those to the defence mechanism. The framework has been heavily tested with multiple neural architectures, datasets, dimensionality reduction techniques and adversarial attacks.

This work could be viewed as an initial attempt at connecting adversarial attacks to the inherent dimensionality of data, both in terms of the theoretical intuition, as well as through empirical evidence. It will be quite interesting to extend the idea to obtain a comprehensive understanding of adversarial examples and their exact relationship with the inherent dimension and geometry of the data.

Chapter 4

Adversarial Attacks and Dimensionality in Text Classifiers

4.1 Introduction

Despite the boom in the usage of machine learning and AI in driving a wide range of applications, adversarial attacks remain a key deterrent to its adoption in practice. The vulnerability of high performance neural networks was demonstrated for the first time in 2015 [28]. The initial observations were made in the domain of computer vision in image processing tasks. Adversaries were able to generate slightly tweaked set of test samples, which would be able to fool the trained networks into misclassifications. These minute structured perturbations would be imperceptible to the human annotator and would have significant degradational effect on the performance of the model, with lots of misclassifications [29]. Other interesting properties like transfer-ability of adversarial samples were observed soon after [30]. This led to the growth in the study of adversarial attacks by researchers. As a result, multiple adversarial attacks and corresponding defence mechanisms were proposed [31]. Naturally, as the threat of adversarial attacks undermined the trust stakeholders had on machine learning driven systems, there were multiple attempts by researchers to understand and explain this vulnerability. Some of the initial works in the domain attributed the origin of adversarial attacks on the linear nature of neural networks [32]. There were counter arguments made by other groups, particularly those studying the optimization landscape of neural networks

and attributing the adversarial vulnerability on the properties of high dimensional spaces [34]. This area of work gained attention as the counter-intuitive properties of the geometry of high dimensional spaces could be empirically correlated to the observed properties of adversarial examples. Despite such efforts, adversarial attacks still pose a genuine threat to the reliable deployment of machine learning models in safety critical applications etc. where errors could lead to catastrophes. A greater understanding of this vulnerability is therefore necessary.

4.1.1 Motivation

There is a lot of available studies on adversarial attacks on image classification models primarily because that is where these attacks were first observed. The literature available concerning adversarial images is rich [31] with a continuous arms race of sorts between attacks and defences. The idea of introducing structured perturbations imperceptible to human annotators is agnostic of the use case, and extend equally well in other domains, like natural language understanding. The methods are not directly possible to map, and natural language has its own distinctive properties and nuances in structure. This necessitates a thorough study of adversarial attacks in text modelling tasks. The neural architectures like recurrent neural networks are inherently different to convolutional neural networks used in image classification tasks and therefore, so are the attacks and corresponding defence mechanisms. While there is some literature available in different attack schemes on text classifiers, our motivation is to try and understand the underlying reason that contributes to the success of the attacks. Specifically, we try to study how the dimensionality of the embedding vector space on top of which the models are built, affects the adversarial vulnerability. There is a requirement of understanding the impact of the size of the vector embedding model and consequently optimization landscape for the neural network on corresponding adversarial examples. This serves as the primary motivation of this work and we attempt to use the insights gained to propose a working defence mechanism against attacks.

4.1.2 Contribution

The primary contributions of this work are inclusive of, but not limited to the following:

- **Understanding Adversarial Attacks:** Studying word level adversarial attacks on text classifier neural architectures and understanding their behaviour with dimensionality. We have looked at the performance of the adversarial attacks on different models with different input embedding dimensions and optimization landscape thereof.
- **Designing Counter-measures:** Analysing the sensitivity of the success of adversarial attacks on these models to develop potential defence mechanisms using ensembles of models. We have observed that adversarial attacks on text work well only when the target model's input embedding dimension matches to that of the one with respect to which the attack was created. We have used this fact to build robust ensembles with models having varying dimensionality to bypass the effect of adversarial attacks.
- **Measuring Adversarial Perturbation:** Studying the effect of different distance metrics on the measurement of adversarial perturbation. We have compared the performance of different distance metrics for the same.

4.2 Background and Related Works

In this section, we go through a brief overview of the literature that is available in the domain and outline the details of adversarial attacks in general and how they have been studied in the space of natural language processing tasks.

4.2.1 Text Classifiers

Text Classification can be defined as the machine learning technique that is used for the classification of a given text document under a predefined class. Suppose, d_i is a document in the entire arrangement of documents D, and $\{c_1, c_2, ..., c_n\}$ is the set of classes, then after classification, a class c_i is assigned to the document

d_j Text classification is a crucial task in Natural Language Processing with many applications, for example, sentiment analysis, spam detection, topic labelling and intent detection.

Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) [Figure 4.2] are the two basic deep learning architectures for text classification [27].

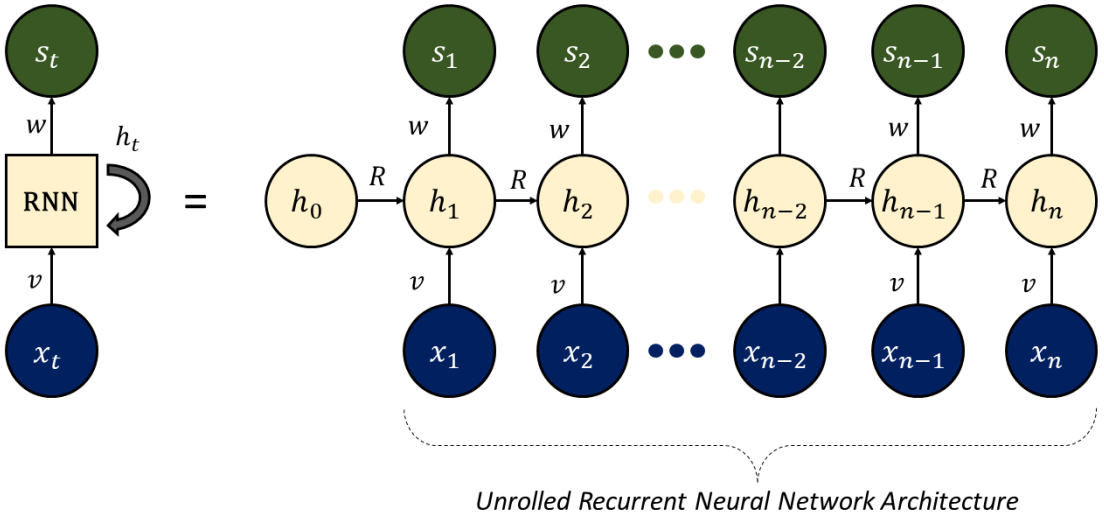


FIGURE 4.1: A recurrent neural network architecture for text classification tasks.

4.2.2 Adversarial Examples

Supervised machine learning, especially in specific tasks related to image processing and natural language processing has benefited strongly from the introduction of neural networks. Adversarial attacks were first observed in neural architectures built for image classification problems, ironically. For a particular classifier, and a test data point that has been correctly classified by the network, a corresponding adversarial sample is a modified test sample which has been altered by the introduction of minute structured perturbation that is not picked up by a human annotator. Adversarial attacks could be applicable to both computer vision or NLP models.

Typically, a text classification problem would consist of four key parts. First is the machine learning model, which is generally some neural architecture best suited for

the task. Second, there needs to be a train dataset, which in practice is a sample picked from a population of all corpuses combined. In the case of a supervised learning problem, like the one we are working on in this work, the training dataset has associated labels with it. Third, a test dataset is there for the model to be used on, for the purpose of prediction. Fourth, once the model is trained over the dataset, the trained class manifolds are learnt within the space of the dataset, that splits it into the respective classes [73].

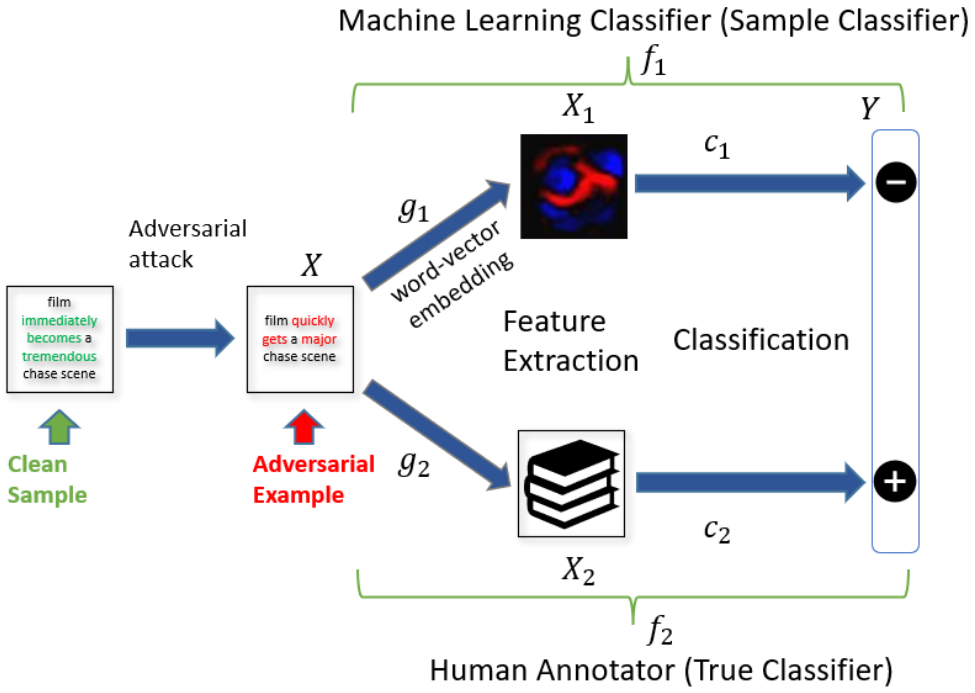


FIGURE 4.2: An adversarial attack on a text classifier neural network model.

Considering a binary classification problem, a true classifier is one that always correctly classifies the test samples, like a human annotator for example that sets the ground truth. If this would have been known to us, there would not have been any requirement of a machine learning model and a simple rule based setup would have sufficed. As an approximation of the true classifier, we build a model that is able to learn the manifolds split by the classifier hyperplane over the landscape. Although the true classifier exists within the population and the learned model works in the sample, we make the reasonable assumption that the true classifier extends to the sample as well. Naturally, there exists a notional gap between the true classifier and its approximation, trained on a non-exhaustive sample set. This leads to the creation of an adversarial space, as any sample that belongs

to this space, would lie on two different sides of the two classifiers mentioned earlier, and therefore lead to a disagreement on the prediction with respect to the machine learning model and the human annotator. Any sample lying in that space is therefore capable of demonstrating adversarial properties. Additionally, if many samples are present near the boundaries of the trained manifolds, then the introduction of a little structured perturbation can shift them across into the area that leads to the vulnerabilities. It is to be noted here that if the perturbation is beyond a specific threshold, then the samples move across both classifiers, therefore not being adversarial any longer.

For the formal definition, let us assume that X is the input sample space. To define the adversarial example, we consider two classifiers as mentioned earlier, f_1 ("sample classifier") and f_2 ("human annotator"). Each of these classifiers have separate components within them, for extracting features and the actual task of classification. Then we have X_1 to be the feature space of the sample classifier, which in fact is the word-vector embedded representation of the input sample. The dimensionality of this is the inherent dimension of the model, as per our definition. X_2 is the feature space used by the human annotator. Furthermore, we also have d_1 and d_2 to be distance metrics, norms which are respectively defined in X_1 and X_2.

If we consider $x \in X$ to be a clean sample, the corresponding adversarial sample created using an adversarial attack is x^*, with a norm d_2 in the space X_2 and a bound on adversarial perturbation $\delta > 0$, we have:

$$\begin{aligned} f_1(x) \neq f_1(x^*) \quad &\text{and} \quad f_2(x) = f_2(x^*) \\ \text{such that} \quad &d_2(g_2(x), g_2(x^*)) < \delta \end{aligned} \tag{4.1}$$

There are two important aspects to note from the above. By definition, the dimensionality of the feature spaces facilitate the generation of adversarial examples. Also, adversarial perturbation is bounded by a threshold in nature.

4.3 Dimensionality and Adversarial Attack

Neural networks are built to optimize parameters that operate in an extremely high dimensional space. It is interesting to study how dimensionality plays an important role in the behaviour of neural networks in dealing with the test samples.

4.3.1 Properties of Dimensionality

The behaviour of data points residing in a high dimensional space is counter-intuitive because irrespective of the distribution from which they are sampled, the spread is not even. There is a theoretical justification for most data points to exist close to the boundaries of the trained manifolds, away form the centres near the surface [74]. Although this is not enough to rigorously establish causality, there is a notion that the generation of adversarial examples are facilitated by the existence of many sample points near the decision boundaries, as a small perturbation can transfer them across. This phenomenon is presented visually in Figure 5.1.

Often, the properties of the high dimensional space is attributed to be one of the primary contributors of adversarial attacks being successful [82]. This has been well tested in the domain of computer vision. One of the major objectives of this work is to investigate empirically, how the behaviour of adversarial attacks on text classifiers depend on the inherent dimensionality of the classification problem.

4.3.2 Dimension Sensitivity

Dimensionality of adversarial attacks is essentially related to the vector embedding dimension of the input to the model against which the attack is mounted. The input to the text classifier networks in most applications of natural language processing is a bunch of vector representations that have been mapped to a learned vector space over the corpus. The size of the input vectors naturally depend on the number of elements considered to create that vector embedding space, which is an user choice when the model is trained. The optimization problem rests on this set of vectors. During training, the parameters of the neural network, which are the weight matrices, are tuned via some optimization technique like gradient descent

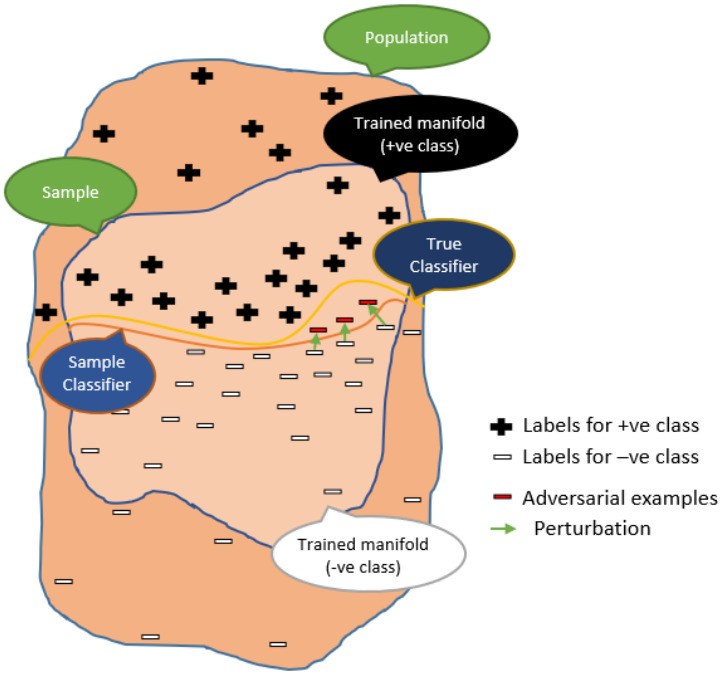


FIGURE 4.3: Dimensionality and adversarial attacks.

etc., and the individual gradients are stored. The trained manifolds that split the landscape into the corresponding classes is separated by the hyperplane classifier.

When the attacks are introduced to clean samples, they are always implemented with respect to some trained model, which it aims to fool. This relates the embedding dimensionality of the model input to the attack mechanism. The attack tries to search and make perturbations to the samples to shift the adversarial example across the separating hyperplane, residing within the optimization landscape characterised by the embedding input dimension. It is therefore natural to expect that these adversarial examples will be sensitive to dimensionality. We have studied to what extent would the embedding dimensionality affect adversarial vulnerability, and how a change in the dimensionality can bring about failures in the attack, which in fact could lead us to some potential defences.

As demonstrated by thorough experimentation, in Section 9.6, there is a strong dependency of the success of the attack to the inherent dimensionality. This is quite unlike the observations made in the domain of images, where higher dimensionality facilitates adversarial attacks [82] but it is not true that the attacks are successful only for the models, whose dimensionality match with the attack's ones.

4.3.3 Ensembling

A key insight that comes out of the study of adversarial attacks and its inherent dimensionality is that the dependence can be used to build defence mechanisms. Adversarial examples in text samples are highly tuned with respect to the model which it tries to fool, and does not transfer well to other models trained on different embedding dimensions. This gives rise to a potential way of blocking adversarial examples by using an ensemble of models trained on different embedding dimensions. A majority vote could be used to determine the corresponding classes of the specific sample in question. The intuition is that the model whose dimensionality matches to that of the adversarial sample would most likely misclassify the sample but the other models will not. It is expected that the rest of the models will not suffer from adversarial performance degradation. In order for this idea to work, it is necessary to study the extent of dependency of adversarial examples and dimensionality, pertaining to how small a change in embedding dimension can lead to its failure. We have thoroughly tested the aforementioned scheme with multiple ensemble model setups to check for adversarial defence mechanisms.

4.3.4 Measuring Adversarial Perturbation

One key aspect about the generation of adversarial examples is that the adversarial perturbation has to be bounded. That is, a structured alteration of the samples, small enough to be imperceptible to the human annotator but significant enough to force a misclassification by a trained neural network. In order to establish the extent of modifications that can be made to the clean samples, it is necessary to measure adversarial perturbation.

The task of measuring adversarial perturbation is not easy, especially in the high dimensional setting. The formal definition of adversarial examples in Eq. (6.1) states that the adversarial perturbation is bounded by a pre-defined threshold δ, which is measurable using some norm distance metric that is defined in the corresponding space. Since this work is studying the relevance of dimensionality in adversarial attacks, it is necessary to investigate the amount of adversarial perturbation required to create adversarial samples at different embedding dimensions. In the case of images, adversarial attacks are facilitated by higher dimensions and

most distance metrics do not provide meaningful measures of distances due to statistical properties [75]. We have studied how this notion holds in the context of text classifiers.

4.4 Implementation

In this section, we discuss the details of the implementation of the adversarial attacks on the text classifier models, mentioning the design choices and the pipeline used.

4.4.1 Models

Text Classifier

We considered two classifier models for the experiments. Our first classifier model is a sequential model which consists of an embedding layer, a LSTM and a Dense layer. The embedding layer, which is non-trainable, takes as input the sequence of words padded to the maximum length of the reviews and is initialized with the constant embedding matrix. The LSTM layer takes as input the sequence of word vectors and outputs a 64-dimensional vector. This layer is accompanied by a dropout ratio of 0.2 and an L2 kernel regularizer. The output layer is a Dense layer that uses the sigmoid activation function and is supplemented with the L2 kernel regularizer. It outputs the predicted class label.

The other model is a transformer model, which is widely used in natural language understanding tasks. Specifically, the one used in the experiments is a BERT (Bidirectional Encoder Representations from Transformers) [**?**] based model. Essentially, it is a pre-trained model on large datasets of the English language through a self-supervised process, that can be fine tuned for specific language processing tasks. In general, the BERT based model has two different objectives that it is able to fulfill, Masked Language Modelling and Next Sequence Prediction. The model is able to learn meaningful inner representations of the language which is useful for many later tasks like text classification etc. For our experiments, we have fine tuned the model for 40 epochs with a batch size of 64 and a learning rate of 5^{-5}, and a maximum sequence length of 512.

Adversarial Attack

To study the impact of dimensionality on adversarial attacks, we used a word level attack scheme, that directly relates to the embedding dimension of the text classifier model. This enabled us to have more control over the process to varying and tuning the dimensionality settings for experiments. Our attack is roughly based on the "TextFooler" attack [83], a blackbox adversarial attack mechanism that generates syntactically and semantically similar adversarial samples. The adversarial attack was designed using the open source python library Textattack [84] and carried out on Microsoft Azure Machine Learning Studio [85]. We used Standard_NC6 virtual machine (NVIDIA Tesla K80 GPU, 6 cores, 56GB RAM) to parallelize and accelerate the attack pipeline.

4.4.2 Pipeline

In this section, we present our experimental setup used to achieve the mentioned results. The dataset mainly used in all our experiments is the IMDB movie reviews dataset. The input pipeline takes as input the dataset in CSV file format which consists of 2 columns - review and sentiment, which may take values 'positive' or 'negative'. Then the input data is preprocessed to encode the sentiment as '1'(positive) or '0'(negative) and remove common stopwords from the review. The preprocessed data is fed to the embedding pipeline, where the words in the reviews are tokenized and converted into their corresponding word-vector representations of given vector size using the open-source python library Gensim.

Text Classifier Model: An embedding matrix is created from the word embeddings and passed to the training pipeline. The data is divided into train, validation and test sets and fed to the LSTM classifier model written in Tensorflow framework. The model is compiled using Adam optimizer, binary cross-entropy loss and uses accuracy as metric. The model was trained for 10 epochs with validation after each epoch. The trained model and test dataset is then fed to the testing pipeline which outputs the number of successful predictions by our trained model.

TextFooler Attack [83]: Moving on to the adversarial pipeline, it takes as input the trained model, the pure dataset, word embeddings and tokenizer. We appended constraints of maximum word swap of 10, minimum cosine similarity of 0.5 for

swapping word, allowed verb-noun swap. We used the Greedy Word swap method [83] to search for substitute words. The pipeline outputs the adversarial samples corresponding to the pure samples if the sample was correctly classified by our classifier.

Robust Ensemble Models: We have used four different types of ensemble models, comprising of three, five and seven models which use majority voting, and a weighted averaged model, where the votes for the class decision are weighted by the respective output probabilities. It is to be noted here that each of the members of the ensemble have been trained with a different input embedding dimension.

4.5 Experiments

In this section, we present the details of the thorough experiments that have been run as part of our analysis. We have tested the aspect of dimentionality and its impact on adversarial attacks in text classifiers, along with studying the scope of potential defence mechanisms and measurement of adversarial perturbation.

4.5.1 Design

The setup of the experiments are described here. We have studied:

- **Understanding adversarial vulnerability and dimensionality:** The adversarial vulnerability of text classifier neural networks for varying associated dimensionality, which is the embedding dimension of the input to the trained neural network upon which the attack has been established. Specifically, the sensitivity of the adversarial attacks to the inherent dimensionality by using a telescopic approach to test for how closely they are related. Small incremental changes were introduced to the embedding dimension of the models on which the attacks were mounted and then the generated adversarial examples were tested on the rest of the models. This experiment was carried out for embedding dimensions of $\{800, \ldots, 1100\}$ with increments of 50 in Table 4.2

- **Adversarial robustness using ensembles:** The usage of ensemble models to thwart the attacks, taking advantage of the fact that they work well only when the adversarial examples are subjected to a model whose input embedding dimension matches to that of the one upon which the adversarial attack was created in the first place, as shown in Tables 4.1 and 4.2. In this case, we have used multiple choices of ensemble models, with 3, 5 and 7 parallel models and carried out a majority voting afterwards on the output classes to assign a label to any specific sample in question.

- **Measuring adversarial perturbation:** The measurement of adversarial perturbation. Considering the vectorized version of the samples (using word vector embedding of specific dimensions), for both clean and adversarial, we used different metrics to study the point to point distances. In order to better understand if the choice of metrics make any difference or not in calculating the adversarial perturbations, we investigated the distribution of the distances for 100 pairs of clean and adversarial samples. We looked at 5-point statistics to compare the distributions in Table 4.3. The distance metrics used are the L_1, L_2 and L_∞.

4.5.2 Results

The experimental results are presented in a tabular form here.

Table 4.1 shows the study of adversarial vulnerability of neural networks with varying inherent dimensionality, along with the performance of the ensemble models. The rows correspond to the different embedding dimensions of the models and the columns correspond to the types of samples the models are tested on. We observe the adversarial vulnerability of the models and robustness of the ensembles.

Table 4.2 presents the study of sensitivity of adversarial examples to inherent dimensionality and the use of ensemble models as a counter-measure. Taking a more finer approach of the earlier study, we look at the adversarial vulnerability with small variations in embedding dimensions.

TABLE 4.1: Study of adversarial vulnerability of neural networks with varying inherent dimensionality.

Models	Samples			
	Clean Samples	Adversarial [Dimension 100]	Adversarial [Dimension 500]	Adversarial [Dimension 1000]
Embedding Dimension 100	90%	21%	65%	71%
Embedding Dimension 500	88%	68%	18%	68%
Embedding Dimension 1000	89%	60%	62%	21%
Transformer (BERT-based)	91%	77%	78%	79%
3 Model Ensemble	90%	51%	53%	60%
5 Model Ensemble	89%	60%	58%	68%
5 Model Weighted Ensemble	89%	62%	60%	68%

TABLE 4.2: Study of sensitivity of adversarial examples to inherent dimensionality and the use of ensemble models as a counter-measure.

	Adversarial Examples							
Models	Dimensionality	800	850	900	950	1000	1050	1100
	Embedding Dimension 800	16%	73%	67%	73%	71%	73%	69%
	Embedding Dimension 850	67%	21%	70%	70%	65%	72%	68%
	Embedding Dimension 900	74%	70%	21%	76%	67%	69%	71%
	Embedding Dimension 950	65%	62%	67%	18%	64%	64%	63%
	Embedding Dimension 1000	69%	72%	74%	77%	25%	73%	68%
	Embedding Dimension 1050	75%	74%	72%	79%	73%	21%	70%
	Embedding Dimension 1100	67%	71%	71%	76%	69%	67%	24%
	Transformer (BERT-based)	79%	77%	77%	80%	79%	78%	80%
	7 Model Ensemble	64%	69%	66%	75%	67%	66%	63%
	7 Model Weighted Ensemble	67%	70%	76%	75%	69%	67%	65%

Table 4.3 presents the study of the measurement of adversarial perturbation with different metrics. The distance metrics used are the L_1, L_2 and L_∞. The corresponding statistics of the distribution of distances between clean and adversarial samples are presented, for different embedding dimensions.

4.5.3 Key Findings

The most important findings and key takeaways are mentioned here:

- We observe that the adversarial attacks are highly successful when they are subjected to models which have the same input embedding dimension as the one with respect to which the adversarial samples was generated. This is seen in Figure 4.4, where for the first three single models, there is an associated adversarial vulnerability demonstrated on one set of samples.

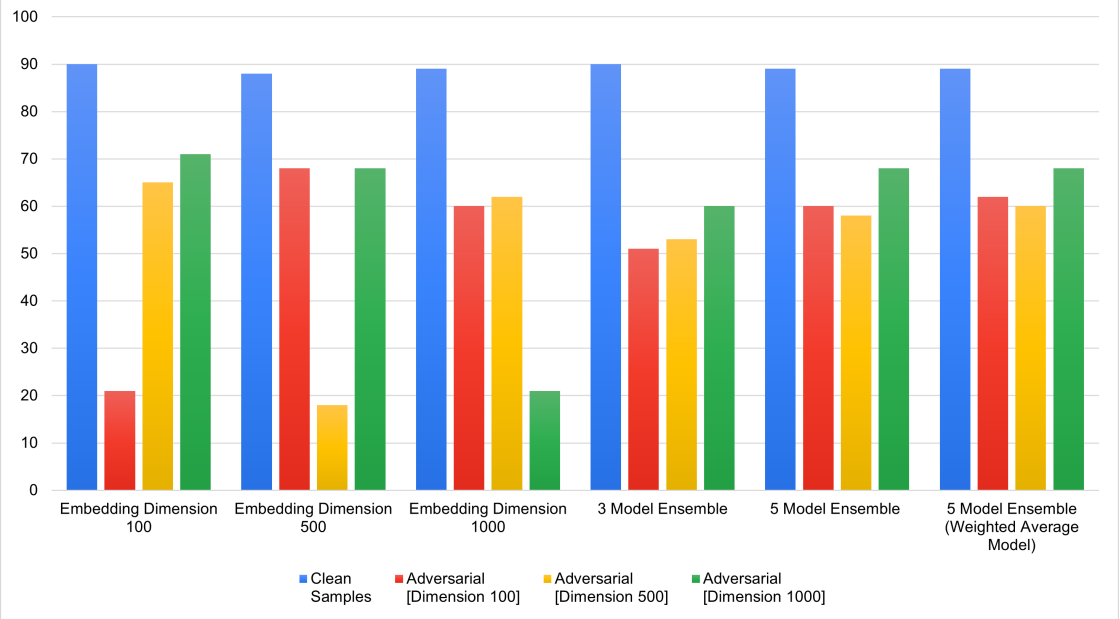


FIGURE 4.4: Adversarial vulnerability of text classifier neural networks with varying inherent dimensionality.

- Figure 4.4 also shows that the ensemble models work, and in fact the plots on the right of the figure show that the three of them do not have any severe vulnerability towards any of the adversarial samples. An ensemble of models can be used for adversarial defence therefore.

TABLE 4.3: Study of the measurement of adversarial perturbation with different metrics.

Metric	Statistic	Emb. Dim. 100	Emb. Dim. 500	Emb. Dim. 1000	Emb. Dim. 2000
L_1	Minimum	28.21	102.21	101.35	219.24
	Mean	97.31	218.23	211.61	411.72
	Median	97.82	217.85	208.53	410.48
	Maximum	130.84	311.88	303.35	819.36
	Standard Deviation	15.6	37.23	38.94	81.68
	CV	16%	17%	18%	19%
L_2	Minimum	3.61	5.93	5.77	6.17
	Mean	12.17	12.2	11.84	11.52
	Median	12.23	12.24	11.68	11.46
	Maximum	16	17.38	16.96	22.86
	Standard Deviation	1.91	2.08	2.16	2.28
	CV	15%	17%	18%	19%
L_∞	Minimum	1.3	0.76	0.81	0.46
	Mean	3.33	1.74	1.69	0.92
	Median	3.39	1.72	1.68	0.9
	Maximum	4.59	2.57	2.55	1.74
	Standard Deviation	0.55	0.32	0.32	0.18
	CV	16%	18%	18%	19%

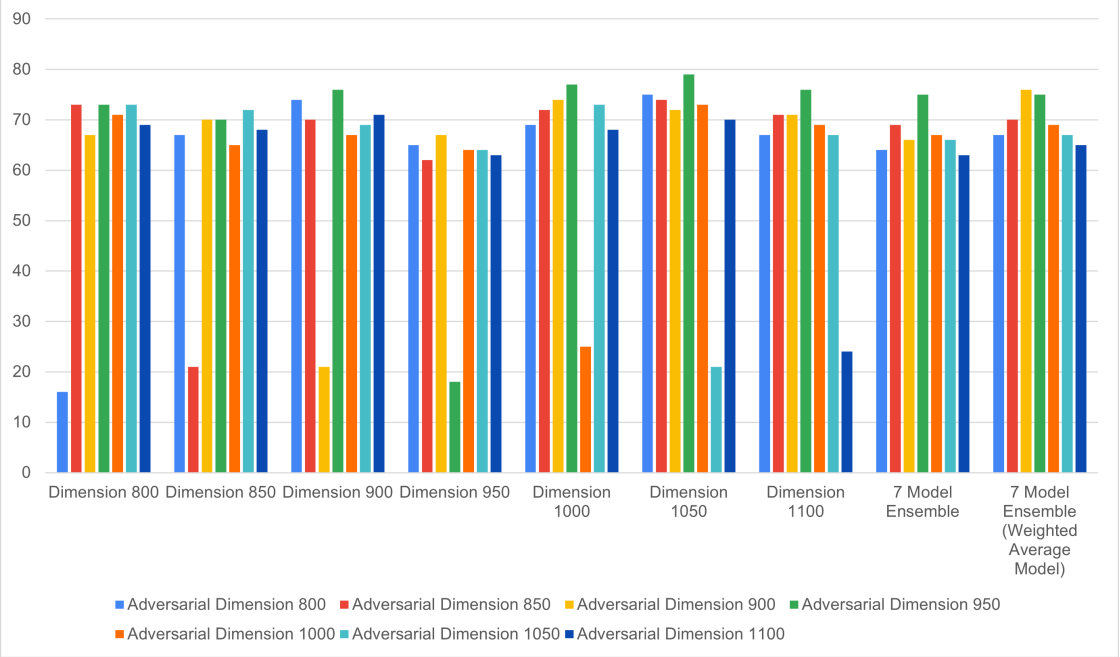


FIGURE 4.5: Sensitivity of adversarial examples to inherent dimensionality and the use of ensemble models as a counter-measure.

- The study of sensitivity of the adversarial samples to its inherent dimensionality reveals that within a very fine margin as well (in the range of 50 as shown in Table 4.2), the success of the attacks are limited to a matching model with the corresponding input embedding dimension. This observation of high sensitivity is well exploited in the defence mechanism involving ensembles. The figure 4.5 shows that while all the individual models have the weakness for their corresponding specific adversarial samples (in the first 7 plots), the ensemble models are able to overcome this vulnerability.

- The study on the measurement of adversarial perturbation shows that there is a consistent growth in variability of the measurements of adversarial perturbation as we increase the embedding dimension of the model against which the attack is created, as evident from the distributions and its corresponding coefficients of variation.

The experimental results corroborate the claims made earlier regarding the use of ensemble models as a potential adversarial defence to the attacks on text classifier models.

4.6 Concluding Remarks

Adversarial attacks have long been regarded as one of the primary threats to machine learning algorithms that prevent its widespread adoption. Originally discovered in image classification tasks, adversarial attacks have been heavily studied in the domain of computer vision. In this work, we investigate adversarial attacks on text classifier problems, and try to understand why they occur and use the insights to develop potential counter-measures. In particular, we have studied adversarial attacks with respect to the inherent dimensionality of the classification problem, which is often attributed to be one of the primary reasons these adversarial examples exist. We tested different adversarial samples created against models which have varying dimensionality, in terms of the input embedding dimension used for training for how sensitive they are to those models as opposed to other models of non-matching dimensionality.

Our results have shown that there is an extremely high dependency of the success of adversarial attacks on text classifiers to the inherent dimensionality, which is quite unlike what is typically observed in images and other data. This observation also lets us make use of this fact to design potential counter-measures. We have used ensemble models to thwart adversaries and the results suggest that they have adversarial robustness.

In the future, we wish to extend this work on other forms of natural language understanding tasks which involve more complex neural architectures, thus increasing the scope of the idea presented in this work.

Chapter 5

Robust Perception for Autonomous Vehicles using Dimensionality Reduction

5.1 Introduction

[1]The research in autonomous driving systems is gaining importance in the most recent years, creating a major disruption in the overall automobile industry. The studies of data on road statistics have led to the conclusion that more that approximately 94% of the accidents on roads are due to some form of driver error, which include inappropriate maneuvers and drivers not paying proper attention [86]. Most major players in the automotive industry including most car manufacturers are investing heavily in the development of vehicles that can perform different levels of driving autonomously, as aptly explained in the SAE (Society for Automobile Engineers) standard SAE-J3016 [87]. This document provides the nomenclature and descriptions for automated driving for motor vehicles, meant for on-road applications, as of 2018. Typically, an autonomous driving setup consists of many sub-parts which are to be assembled to create the overall setup. These

[1]Chapter 5 is published as *Nandish Chattopadhyay, Shivam Garg and Anupam Chattopadhyay , "Adversarial robustness in Object Detectors using dimensionality reduction," in Proceedings of IEEE TrustCom 2022*

individual tasks include Traffic Sign Detection, Vehicle Localization, Motion Planning, Pedestrian Detection, Road-marking Detection, Automated Parking, System Fault Diagnosis and Vehicle Cybersecurity [88].

There are different levels of autonomous driving, ranging from an ordinary human operated car with some form of inbuilt intelligence to a fully autonomous self driven vehicle. The SAE International document called SAE-J3016 mentioned above, describes five different levels of autonomy. For levels 1 and 2, the driver is needed to handle the car at all times. In case of levels 3 and 4, the car can be taken control of by the driver under difficult situations. Level 5 is the specific case wherein the car is fully autonomous at all times [89]. There is always an incremental approach that is followed for introducing new technologies in intelligent vehicles. The addition of new techniques of automation raise the level of smartness in the vehicles, which is able to support the driver for maintaining speed or lane following or even a driver-vehicle handover [90].

As one can imagine, autonomous vehicles are designed to operate in very complex and highly dynamic environments, which necessitate mechanisms that are more generic and are applicable to many relatively unpredictable situations and are able to work within specific time-frames to be able to compare well with human-level accuracy and be as reliable. The decision making framework behind all of this is extremely critical. As of the present, the most sophisticated machine learning techniques and the state-of-the-art in computer vision have produced high performances but not enough for complete autonomous navigation [91].

5.1.1 Motivation

In the context of autonomous driving, the entire decision making system is based upon the fundamental task to perception to start with. It is a critical component of the autonomously driven car, which helps them get a sense of the world around them. In standard autonomous cars, the perception module comprises of tasks involving detecting and tracking of necessary environment that includes traffic signals, obstacles, pedestrians etc. The evolution of autonomously driven cars have focused on making these systems more and more reliable for safety-critical applications.

Typically, object detection models work on the video feed captured by the cameras placed at different parts of the car and make decisions on its future course of action. Naturally, the reliability of driver-less cars therefore is dependant on the accuracies of the aforementioned models. Robustness of such models is an absolute necessity for any practical usage.

While the image processing domain saw the highest growth in performances of deep learning models, the vulnerabilities were also first reported in the domain [28]. Targeted structured perturbations in images, imperceptible to human vision, could fool neural networks easily. These adversarial attacks prove to be a major deterrent to the acceptable reliability of image classifiers. As a natural progression, these attacks have been demonstrated in video streams as well. The adversarial vulnerability of such object recognition systems working on video based data proves to be a major deterrent to their usage in safety-critical applications like in autonomous driving.

Let us consider a very important task within the decision making system of autonomous vehicles, sign recognition. Any misclassification made by the object recognition system of the car, for identifying traffic signs, can have many catastrophic consequences. This might lead to a direct threat to people's lives, or in some other case a disruption of the traffic system in case there are many autonomous vehicles operating similarly.

The models deployed within the cars' camera feed, that drives the decision making system of the vehicles, can fall prey to adversarial attacks easily and the results could be disastrous. Strong counter-measures need to be designed for robustness of ML models against adversarial attacks that work on videos for object detection and classification. This serves as the primary motivation of our work.

5.1.2 Contribution

We have attempted to design countermeasures to adversarial attacks, that are able to fool the cameras of the autonomous vehicles, which capture video streams. The primary contributions made this paper are mentioned here.

- Relating adversarial vulnerability of state-of-the-art object detector and classifier models to the inherent dimensionality of the input video streams, and

using the theoretical background to suggest that the properties of the high-dimensional setting significantly contribute to adversarial vulnerability.

- Using dimensionality reduction as a counter measure against adversarial vulnerability, and establishing the claim empirically using experimental results.
- Optimizing the amount of variability to be preserved for dimensionality reduction, keeping it a hyper-parameter to the model.

The proposition of using dimensionality reduction as a counter measure has been tested on multiple object recognition models and corresponding adversarial attacks as described later in this work.

5.2 Background

We begin by taking a look at the fundamentals, including the basics of the machine learning tasks and reviewing the state-of-the-art models that are used to solve those problems.

5.2.1 Object Detection and Classification

The two processes object classification and localization are combined in object detection. The two main types of object detectors used today are networks that divide the tasks of determining an object's location and classification, with Faster R-CNN[92] being one of the most well-known examples, and networks that predict bounding boxes and class scores simultaneously, with YOLO[93] and SSD[94] networks being popular architectures.

The development of region suggestions required a lot of time and resources, which was the main issue with Fast R-CNN[95] and R-CNN[96]. By combining Fast R-CNN and a region proposal network(RPN), Faster R-CNN was able to resolve this issue. The YOLO network transformed object detection network architecture. It is capable of simultaneously predicting class and box boundaries. The model suggested to split the picture into a grid, with each cell forecasting a confidence score of an object's presence with the appropriate bounding box coordinates. This made YOLO predictions possible in real time.

5.2.1.1 Faster RCNN

Faster-RCNN[92] is a proposal-based object detector model which initially proposes several regions that may contain an object using Region Proposal Network (RPN). The output of the feature map is covered by a tiny network that is slid over by the RPN which creates the suggestions. The feature map serves as the input for this network's spatial window. A lower dimensional characteristic is assigned to each sliding window. While the regression offers more precise localization information, the sliding window's location provides localization information relative to the picture. It uses Pyramid of filters rather than using pyramid of images. In the next step, two parallel Convolutional Neural Networks run. The first one proposes the scores of each box and the other proposes the coordinates of each box.

It is possible that each object is detected by multiple boxes, so the Intersection over Union Algorithm is used to select the box with the highest probability. In order to successfully fool the Faster-RCNN Network, an adversarial attack has to misclassify all the proposed boxes. The attack is not successful even if one of the boxes correctly detects the object. Hence, it is very challenging to build an adversarial attack against Faster-RCNN.

5.2.1.2 YOLOv3

You only look once v3(YOLOv3)[97] is a regression based object detector model that can detect objects in real time. YOLO algorithm divides an image into grids and then, search for objects in each grid. YOLOv1[93] was trained on ImageNet-100 dataset but it failed to detect small objects present close to each other in an image. Batch Normalization, Higher Resolution Classifier, Anchor Boxes, Multi-Scale Training, DarkNET-19 were added in YOLOv2[98] in order to solve challenges faces by YOLOv1 and also to increase its speed.

To further increase the speed and accuracy of YOLOv2, DarkNET-53 was used in YOLOv3 for extracting features.DarkNET-53 contains 53 Convolutional layers and 23 Residual layers and is much stronger than DarkNET-19. YOLOv3 also used logistic classifier as a replacement of softmax classifier for class predictions.

5.2.1.3 Shape Shifter

Shape Shifter[99] Adversarial Attack uses Expectation over Estimation technique to create a robust attack on Faster-RCNN. Expectation over Estimation is employed in image classification to strengthen its resistance to adversarial perturbations, and it has been effectively applied to the object detection environment. ShapeShifter may produce adversarially disturbed stop signs that are repeatedly mistaken for other things using Faster RCNN. To generate a larger and more robust perturbation, only the red part of the stop sign is allowed to change, leaving the white text intact. It can perform a targeted attack on all the other 79 classes and also perform an un-targeted attack. This attack poses a very large threat to current Autonomous Vehicles and other safety-critical computer vision systems.

5.2.1.4 Daedalus Attack

The Daedalus attack may be generalised to affect many end-to-end Object Detection models, rendering many Object Detection applications unusable. Daedalus instances built using an ensemble of replacements may launch assaults without being aware of the victim models' properties because to the pervasiveness of model reuse in real-world settings. Non-Maximum Suppression (NMS), a crucial stage in the detection process, is used to filter the redundant detection boxes. To avoid NMS, the Daedalus assault shrinks the size of detection boxes. Because of this the final detection output has a very high density of false positives.

5.3 Implementation

To perform our experiments on Faster-RCNN, we took 30 images of road sign stop signal. For each of these 30 images, 79 adversarial images were generated, each one for 79 target classes using Shapeshifter Adversarial attack. We performed SVD on these 2370 full images and 2370 stop sign patch only for 6 different values of threshold [0.75, 08, 0.85, 0.9, 0.95, 0.99]. Object detector was then run on these 2370 adversarial images and dimensionally reduced images to measure the robustness of the attack. For YOLOv3, we randomly selected 200 images and generated

adversarial images using Daedalus attack. Dimensionality of these images are also reduced just like in the case of Faster-RCNN model.

5.3.1 Adversarial defence mechanism

Singular Value Decomposition(SVD) reduces the dimensionality of the image without losing the key features of the image. Each image is represented by three colours. These 3 matrices are reduced independently of each other using SVD. We have tested our dataset for different values of threshold. However, for each image the threshold value for Red, Green and Blue colour is kept same. The threshold value can not be very low or, otherwise some important features of the image are lost.

SVD decomposes a matrix into product of three matrices. The second matrix is a diagonal matrix in which the eigenvalues are stored in non-decreasing order. To obtain an image with variability higher than a given threshold, the first 'r' eigenvalues are chosen whose sum divided by the total sum of the eigenvalues is higher than the threshold required. This same process is repeated for the three RGB matrices.

5.3.2 The pipeline

The pipeline consists of three components: The object detector, the adversarial attacks and the dimensionality reduction mechanism. The dimensionality of the image is reduced using Singular Value Decomposition. We performed our experiments on two different state-of-the-art object detectors on video streams: YOLOv3 and Faster-RCNN. The daedalus attack is used as an adversarial attack against YOLOv3 and the ShapeShifter attack is used as an adversarial attack against Faster-RCNN. The original image is first converted to an adversarial images using these attacks as shown in Figure 5.1.

In the next step, these adversarial images are reduced using SVD. Different threshold values are used for reducing the dimensions of the image. Objects in adversarial images generated using Daedalus attack are detected using YOLOv3 and the objects in adversarial images generated using the ShapeShifter attack are detected using Faster-RCNN. Both adversarial images and dimensionally reduced images

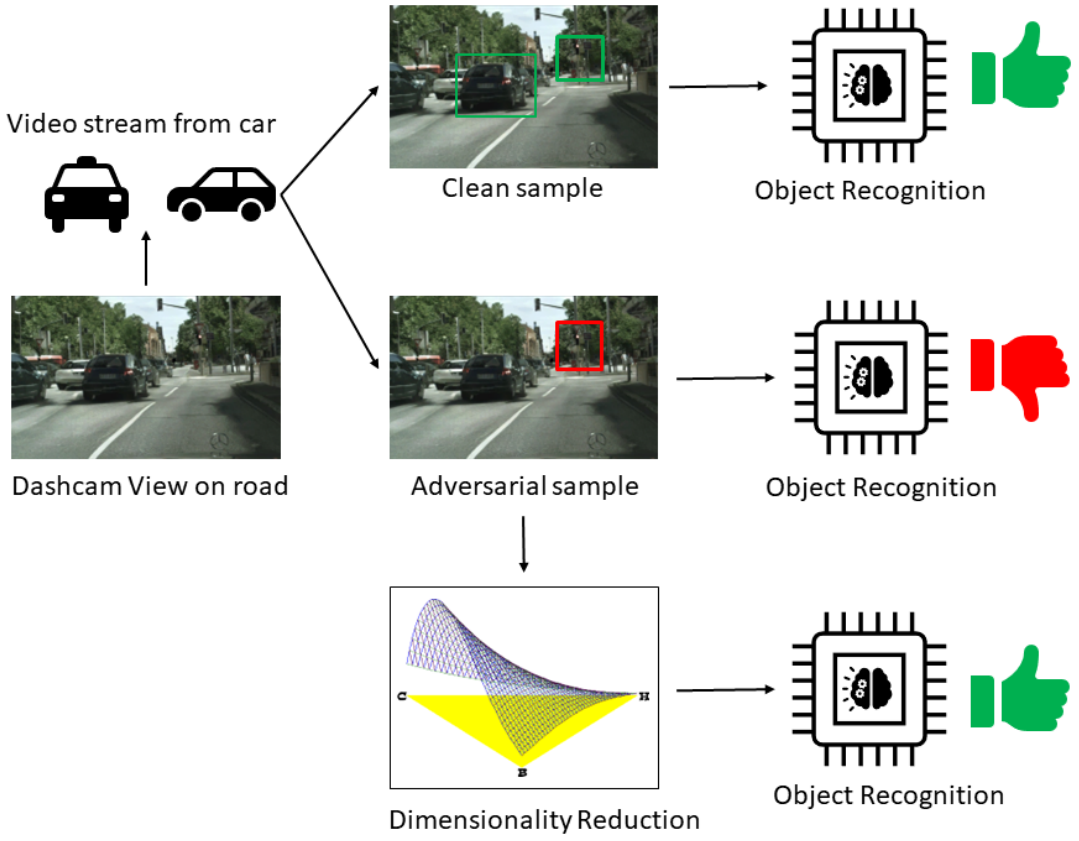


FIGURE 5.1: Pipeline

are passed through the respective object detector to compare the effects of adding the SVD.

This pipeline is followed for both adversarial attacks and for different threshold values. For very low threshold values(<0.75), object detector was unable to detect any objects. This is because the quality of the image has been downsized by huge amount. On the other hand, for very high threshold values(>0.95), object detector was detecting false objects(adversarial objects) only.

5.3.3 Use-cases

We have tested our mechanism against two adversarial attacks. The Daedalus attack targets YOLOv3 and the target class is Person. The dimensionality of the full image is reduced with variability preserved from 99% to 70%. The Shape Shifter attack targets Faster-RCNN. It adds perturbations to the stop sign such

that it is misclassified by the object detector. We performed our experiments in two ways: dimensionality reduction of full image, dimensionality reduction of stop sign patch only. The variability preservation was again varied from 99% to 75%.

5.4 Experiments

In this Section, we discuss in detail the experimental setup and the results obtained. We also analyze the results and derive insights from them.

5.4.1 Experimental Design

The goal of the experiments was to study the vulnerability of object detectors and classifiers to adversarial attacks. Along with it, the proposition of dimensionality reduction was also tested for being a potential counter-measure. The set of experiments are listed here:

- Study of decline in performance of object detectors upon the introduction of adversarial attacks using adversarial samples
- Test of robustness of proposed counter-measure using dimensionality reduction
- Optimising the dimensionality reduction pipeline by tuning the hyper-parameter that sets the amount of information/variability to be preserved
- Study the cost of adversarial robustness by measuring the additional time consumption by the exact same hardware resource with and without the dimensionality reduction

It may be noted here that based on the available literature regarding the state-of-the-art in object detector and recognition algorithms discussed in Section 6.1 and the corresponding adversarial attacks, the notion of dimensionality reduction as a countermeasure has been tested for all the combinations, one at a time.

5.4.2 Results

In this section we present all the experimental results in details.

Table 5.1 presents the study on adversarial vulnerability of object detector models and how dimensionality reduction can work as a counter measure. We have noted the performance of the model on clean samples, adversarial samples and dimension reduced samples.

TABLE 5.1: Study of adversarial vulnerability of object detector models and a counter-measure in dimensionality reduction.

Models	Samples	Performance (Accuracy)
Faster RCNN	Clean Samples	99%
	Adversarial Samples (Shape-Shifter)	1%
	Dimension Reduced Samples (optimised)	96%
YOLO	Clean Samples	86.32%
	Adversarial Samples (Daedalus)	1%
	Dimension Reduced Samples (optimised)	95%

Table 5.2 presents the results of tuning the hyper-parameter, which is the amount of variability to preserve for dimension reduction.

Table 5.3 presents the study of the cost of adversarial robustness, measured through inference time consumption.

5.4.3 Key Findings

The key takeaways from the experiments are mentioned here:

- The adversarial attacks introduced on the samples significantly bring down the performance of the object detector models, rendering them unfit for any practical use. This is as expected, and sets the baseline for the comparative study of the efficacy of the adversarial defence mechanism proposed.

TABLE 5.2: Tuning the hyper-parameters of dimensionality reduction by studying its impact on accuracy for different levels of preserving information in samples.

Variability Preserved	Faster RCNN & Shape Shifter		YOLO & Daedalus
	Accuracy (full image)	Accuracy (patch)	Accuracy (full image)
99%	1%	1%	1%
95%	1%	1%	15%
90%	53%	37%	79%
85%	86%	76%	89%
80%	96%	93.5%	95%
75%	99%	99.5%	93%
70%	99.5%	100%	91%

TABLE 5.3: Cost of adversarial robustness (inference time)

Model and Adversarial Attack	Inference Time (200 samples)	
	Without SVD	With SVD
Faster-RCNN and ShapeShifter	91.2ms	105.1ms
YOLO and Daedalus	47.4ms	61.3ms

- Dimension reduction works well in being a counter-measure against adversarial attacks. We observe from the data in Table 5.1 that dimension reduction works across different object recognition models and their corresponding adversarial attacks applied on them.

- The effectiveness of reducing dimension plateaus out and drops beyond a point and optimizing it as a hyper-parameter is necessary. It must be noted here that dimensionality reduction eliminates information. The goal here is to get rid of the adversarial perturbation and preserve the actual information present in the samples that is necessary for the machine learning task to be accomplished. Therefore the decision on amount of variability that has to be

preserved is critical towards the success of this adversarial defence mechanism.

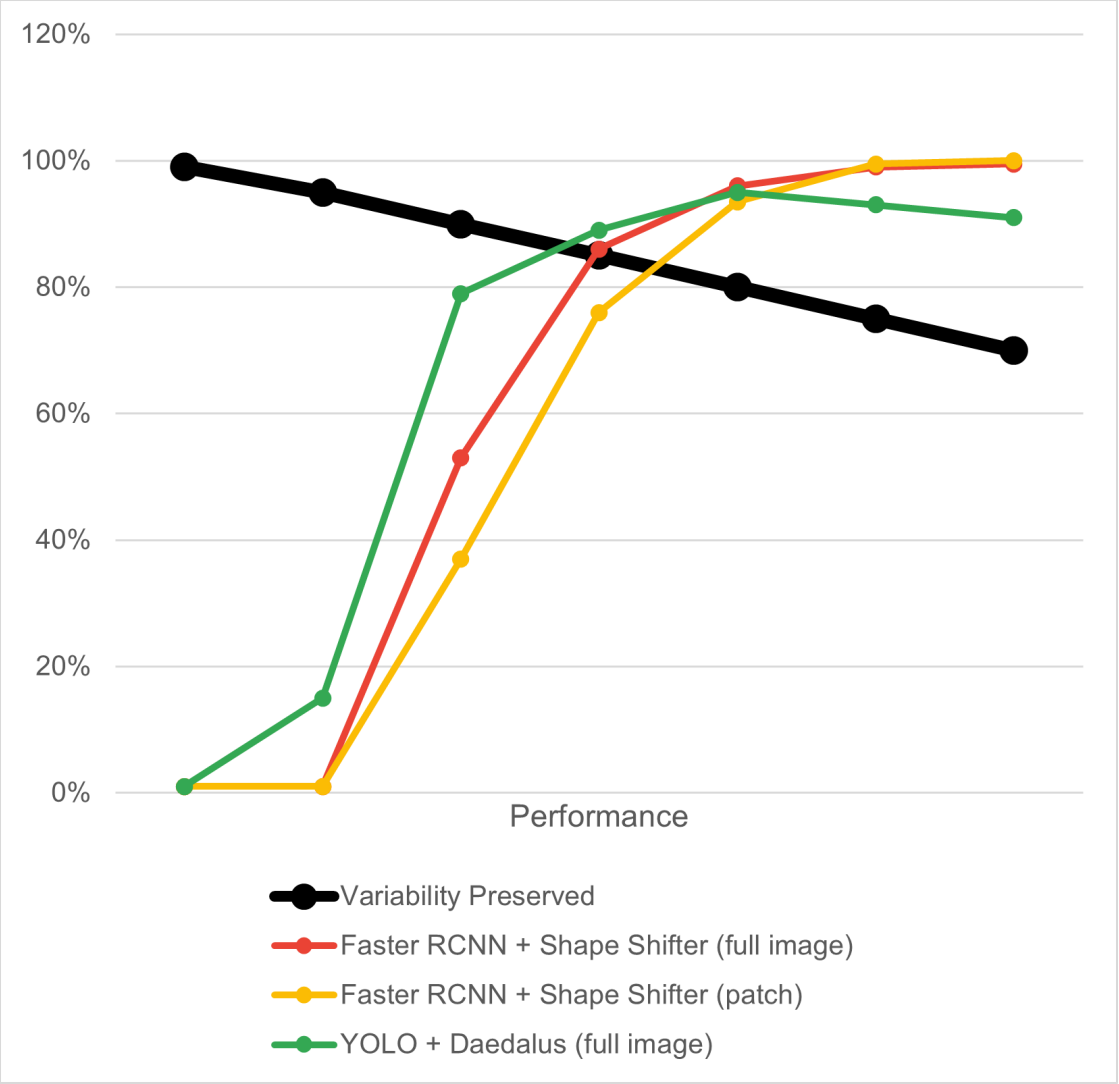


FIGURE 5.2: Study of the behaviour of the performance of the system as the dimensionality is reduced.

- The adversarial defence mechanism using dimensionality reduction does not incur significant costs in terms of inference times, as shown in the data presented in Table 5.3. This is necessary for it to be used in the practical applications.

In general, there is sufficient evidence in experimental results to corroborate the theoretical formulation that adversarial attacks may be thwarted using dimensionality reduction techniques, and the same is studied in this paper as part of the

decision making system responsible for perception within the framework of an autonomous vehicle.

5.5 Conclusion and Future Scope

With the growth in research and development of autonomous vehicles, there has been an increased attention on the reliability of the decision making systems, that form the backbone of this paradigm. There are many different components of that reliability, and in this work, we have focused our attention to one of the fundamental operations, object detection and traffic sign recognition tasks. We have studied the state-of-the-art in the machine learning models that are able to carry out the object recognition task from the video streams captured by the cameras in the car. Specifically, we studied the vulnerabilities of these models to adversarial attacks. We reviewed the attacks and studied the reason behind their prevalence. We related their occurrence to the high dimensional nature of the optimization landscape in which the machine learning models operate. We used this knowledge to design countermeasures, by using dimensionality reduction. This idea has been tested on benchmarking datasets.

As part of future work, we wish to deploy the same on real test environments, within the dash-cams of the cars and test for performance from both machine learning and real-time systems perspectives.

Chapter 6

Spatially Correlated Patterns in Adversarial Images

6.1 Background

The flourishing results in AI research did not translate into reliable solutions as a key vulnerability of such models was observed soon after. In 2015, adversarial attacks on highly accurate neural network models were demonstrated [28]. A trained model would achieve very high accuracy on a clean test set but the performance would be very poor on a set of adversarial samples. These samples could easily be constructed using highly structured perturbations which would remain imperceptible to the human vision [29]. Additionally, adversarial samples generated with respect to one neural network architecture would also serve as an adversarial sample [30] for a different machine learning classifier like SVM [72]. Therefore, a plethora of research was carried out developing mechanisms of attack subsequently [2].

This resulted in a cat-and-mouse chase between adversarial attacks and defenses, as both kept getting better [31]. Taking a step aside, some enthusiasts have been interested in looking at the root cause of the problem from first principles and understand why these attacks take place rather easily. Goodfellow et al. [32] argued that the inherent linearity in the neural networks could facilitate the generation of adversarial samples. Others point out a different rationale for it, [33], primarily addressing the curse of dimensionality [82].

6.1.1 Motivation

Image classifiers, particularly neural networks optimize on a very complex landscape, wherein a decision boundary is obtained that separates the trained manifolds. The behaviour and properties of the classifier is closely related to the nature of the feature space [100]. Adversarial attacks introduce small (bounded) targeted perturbations to the images so that the adversarial samples cross over the decision boundary, and is therefore misclassified by the model. The feature space is high dimensional and difficult to interpret. Therefore, it is necessary to map the properties and knowledge about the feature space on to the input space, so that the patterns are less abstract and can be used easily for various purposes, including the critical task of being robust against adversarial attacks. It is important to study if the properties and patterns of the feature space are preserved in the input space.

6.1.2 Contribution

While analysing patterns of features for adversarial vulnerability, a pair of 'Source' and 'Target' classes is usually considered, since it is conceived that there is a specific path of movement of the samples in the high dimensional space across the decision boundary. Also, this is consistent with all samples belonging to the particular pair of classes. We take cognizance of these facts in our analysis. In this work, we have formalised theoretically and thereafter verified empirically, a comprehensive study on:

Segregation

Use of spatially correlated patterns to identify and segregate regions within the input space (consisting of pixels of the image) specifically contributing towards:

- the correct classification of the input image during inference by the trained classifier (we call this region the Region of Importance (RoI))
- incorrect classification of the adversarial sample (generated through some adversarial attack) by the trained classifier (we call this region the Region of Attack (RoI))

Isolation

Use this set of regions (RoI and RoA) to isolate four disjoint sets of pixels (we call $UV, U\bar{V}, \bar{U}V, \bar{U}\bar{V}$ based on their impact on utility U and vulnerability V) which help us in explaining the reason as to why certain segments of the image are particularly more vulnerable to any adversarial attack than the rest.

Neutralization

Extract the region of most vulnerability from the thus obtained segments and attempt to block out its information content to establish a post-hoc adversarial defence mechanism. Specifically, we study the impact of the *neutralization* process on:

- the image's classification by the trained classifier upon being modified
- the image's adversarial vulnerability upon being modified

We have substantiated the propositions with thorough experimental verification on multiple standard datasets using a well-known classifier and attack mechanism.

6.2 Theoretical Formulation

Although the maximum advancements in the realm of deep learning has been in the domain of supervised image classification tasks [71], this is also where the initial adversarial attacks were seen. Images have been highly vulnerable to being sensitive towards little strategic perturbations.

An adversarial attack is a mechanism used by a malicious entity to force erroneous decisions by the classifier, by modifying test samples carefully, whilst keeping the modifications imperceptible to the annotator.

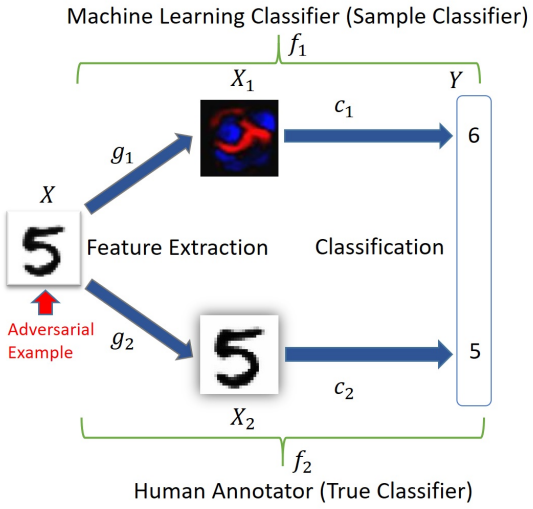


FIGURE 6.1: Adversarial attacks on images

6.2.1 Adversarial Attacks

Let us consider X to be the input sample space. There are two competing classifiers, f_1 (sample classifier) and f_2 (human annotator). The classifiers consist of two sections, for feature extraction and classification. We consider X_1 to be the feature space for the sample classifier and X_2 to be the feature space for the human annotator, where d_1 and d_2 are norms defined in the spaces X_1 and X_2 respectively. Therefore, $f_1 = c_1 \circ g_1$ and $f_2 = c_2 \circ g_2$ as shown in Figure 6.1. If $x \in X$ is a randomly chosen training sample, then its corresponding adversarial example x^*, for a norm d_2 defined on the space X_2, and a predefined threshold $\delta > 0$, satisfies [82]:

$$\begin{aligned} f_1(x) \neq f_1(x^*) \quad &\text{and} \quad f_2(x) = f_2(x^*) \\ \text{such that} \quad &d_2(g_2(x), g_2(x^*)) < \delta \end{aligned}$$

6.2.2 Intuition for studying correlation among samples of a class

The distribution of the data is important while studying adversarial attacks. A classifier, typically is a hyperplane that splits the data into trained manifolds, each pertaining to a particular class. The properties of the distribution of the

data is easy to study within each of the manifolds. An adversarial example is a sample, that is made to shift across the hyperplane classifier by introducing structured perturbation. The geometry of the landscape of the features is therefore critical in the design of adversarial samples. For a particular sample belonging to its true class' manifold, in order to slide into another class' manifold thereby becoming an adversarial sample, it takes a specific path that is governed by the geometry of the two manifolds. This path will naturally be different for any pair of 'Source' and 'Target' classes, where a sample from a 'Source' class is converted into an adversarial sample being misclassified into the 'Target' class. The purpose of studying correlation among features is to look for a basis to observe patterns that can be associated with specific tasks like classification or adversarial attacks and the distribution of the samples in the corresponding optimization landscape.

6.2.3 Significance of Features

Let us consider we have a binary classification setup, wherein we sample input-label pairs $(x, y) \in X \times \{\pm 1\}$ from a particular distribution D, for the purpose of training a classifier $C : X \to \{\pm 1\}$, which is able to predict y, given an x.

In an attempt to understand how features play a key role in classification and the generation of adversarial examples, let us define a *feature* to be a map from the input space X to R, therefore constituting a set of features $F = \{f : X \to R\}$. The features can thereafter be categorised as proposed in the relevant literature [101].

ρ-useful features: Given a distribution D, a particular feature f is said to be ρ-useful if the feature is correlated with the label of the class in expectation. Therefore, we have $E_{(x,y)\in D}[y \cdot f(x)] \geq \rho$. This follows that $\rho_D(f)$ is the largest ρ for which the feature f is ρ-useful given distribution D.

γ-robustly useful features: A ρ-useful feature $f(\rho_D(f) > 0$ is said to be *robust* (γ-robustly useful feature for $\gamma > 0$) if when an adversarial perturbation Δ is introduced, f remains γ-useful. That is, $E_{(x,y)\in D}[\inf_{\delta\in\Delta(x)} y \cdot f(x + \delta)] \geq \gamma$.

Useful, non-robust features: This is a ρ-useful feature which is not a γ-robust feature for any $\gamma \geq 0$. They make a contribution towards classification of the image, but have the potential to adversely affect accuracy in the adversarial setting.

6.2.4 Interpretability: Feature space to Input space

The existing work suggests correlation among features that play an important part in the process of both classification and the generation of adversarial examples. In this work, we study the relevance of that in the input space, instead of the feature space. The primary motivation behind that is the fact that the input space, comprising of pixels for an image, is more interpret-able and less abstract than the feature space or the high dimensional optimization landscape. For that purpose, we map the features into the input space and study how the correlation of features result into the co-location of pixels.

In order to study the mapping of the features of interest from the feature space to the pixels in the input space, we need to develop a framework. We can perceive the setup as a contest between two players, a classifier that tries to correctly classify a particular sample and an attacker that tries to mis-classify the same.

Let us consider the input image as a set of pixels, each pixel is denoted by its location, indexed serially. Thus, let X be the input image of n pixels such that $X = \{x_1, \ldots, x_n\}$. The classifier maps X to the set of labels $Y = \{\hat{y}, \bar{y}\}$, such that for a particular 'Source-Target' pair, $\hat{y}$ is the correct label and $\bar{y}$ is the mis-classified label of the adversarial sample. The classifier looks at some pixels of the image contained in the set $X_p = x_i, \ldots, x_p$ for classification and the adversarial attacker modifies some pixels $X_q = x_j, \ldots, x_q$. Therefore, we have:

$$X_p = \{x_i, \ldots, x_p | x_i, \ldots, x_p \sim \hat{y}\}$$
$$X_q = \{x_j, \ldots, x_q | x_j, \ldots, x_q \sim \bar{y}\}$$

where $\sim$ denotes that those particular pixels have leverage on the decision of the output. It must be noted here that X_p and X_q are not necessarily mutually exclusive. Based on this formulation, we can define two axioms.

Theorem 6.1. *For all input pixels $x_k \in X$, we assign x_k to the Region of Importance (RoI) if $corr(x_k, \hat{y}) \geq \delta_1$, where δ_1 is the threshold for the leverage of the pixel on the output label. That is,*

$$\forall x_k \in X, \quad x_k \Rightarrow RoI \quad | \quad corr(x_k, \hat{y}) \geq \delta_1$$

Hence, we have $RoI \subseteq X_p$.

Theorem 6.2. *For all input pixels $x_k \in X$, we assign x_k to the Region of Attack (RoA) if $corr(x_k, \bar{y}) \geq \delta_2$, where δ_2 is the threshold for the leverage of the pixel on the adversarial mis-classfied label. That is,*

$$\forall x_k \in X, \quad x_k \Rightarrow RoA \quad | \quad corr(x_k, \bar{y}) \geq \delta_2$$

Hence, we have $RoA \subseteq X_q$.

Exactly like X_p and X_q, the *RoI* and *RoA* are not mutually exclusive either. In fact, their overlap is of importance, as illustrated in the later sections of this work.

Based on these axioms, we make the following assumptions theoretically:

- The distribution of pixels belonging to RoI and RoA within an image is not random, the pixels are correlated spatially, that is they are co-located.
- The distribution of pixels belonging to RoI and RoA for images belonging to any particular class of images is not random, the pixels are correlated spatially. This is considered naturally for images which are translationally invariant because otherwise looking for spatial correlation across images in not very meaningful.

We have tested for evidence regarding the validity of these two assumptions empirically on multiple datasets in Section 9.6. Having obtained the RoI and the RoA, we shall now proceed to further segmentation of the image into mutually exclusive regions, each with their own significance. It may also be noted here that, this approach preserves the ability of segregation of features/pixels into the binary categories of Usefulness and Robustness as well. We slightly modify the classification into 'Utility' and 'Vulnerability', as shown in Figure 6.2.

6.2.5 Segregating and Isolating Regions

To accomplish the task of generating the RoI and the RoA (the process we call *Segregation*) using the axioms 6.1 and 6.2, we need a mechanism for studying the leverage of a particular pixel on the classification task and the adversarial attack. This would help us calculate the correlation and apply the thresholds. Considering

a convolutional neural network based classifier (as it is most widely used for image classification tasks, although any neural network architecture is applicable here) and any generic adversarial attack mechanism, we:

- map the features from the final flattened layer of the CNN to the input space (using class activation maps), note its significance to generate X_p and use the threshold δ_1 to include a particular pixel to the RoI.
- look at the pixels in the input space that have been modified to generate an adversarial sample to generate X_q, and include those in the RoA which have been altered (by L_2 norm) beyond the threshold δ_2.

For any set of (δ_1, δ_2), upon the generation of the RoI and the RoA, we can split the input space into four particular regions $UV, U\bar{V}, \bar{U}V, \bar{U}\bar{V}$ as shown hereafter, each with specific significance (based on utility U and vulnerability V, U+V+, U+V-, U-V+ and U-V- respectively). We call it *Isolation*. It may be noted here that this *Segregation* works for both images at the individual level with individual RoI and RoI, and also at the class level, using the representative ones, if the assumptions made above hold good. For the image X containing n pixels, each denoted by its corresponding position index number, we can construct the disjoint partitions $UV, U\bar{V}, \bar{U}V, \bar{U}\bar{V}$ as:

$$\begin{aligned} UV \quad & [U+V+] = \{x | x \in RoI, x \in RoA\} \\ U\bar{V} \quad & [U+V-] = \{x | x \in RoI, x \notin RoA\} \\ \bar{U}V \quad & [U-V+] = \{x | x \notin RoI, x \in RoA\} \\ \bar{U}\bar{V} \quad & [U-V-] = \{x | x \notin RoI, x \notin RoA\} \end{aligned} \tag{6.1}$$

Intuitively, each region in a particular image and its significance can be expressed as follows: Region $U\bar{V}$ [U+V-] consists of those spatially co-located pixels that have an important role to play in the classification process; Region UV [U+V+] consists of those spatially co-located pixels that have some role in both classification and adversarial attack; Region $\bar{U}V$ [U-V+] consists of those co-located pixels that are vulnerable towards an adversarial attack and are usually modified for the generation of an adversarial sample; Region $\bar{U}\bar{V}$ [U-V-] is the area in the image that is not important for either of the classification process or the generation of an adversarial sample. The mapping of each of the regions onto the plot of Utility and Vulnerability is presented in Figure 6.2.

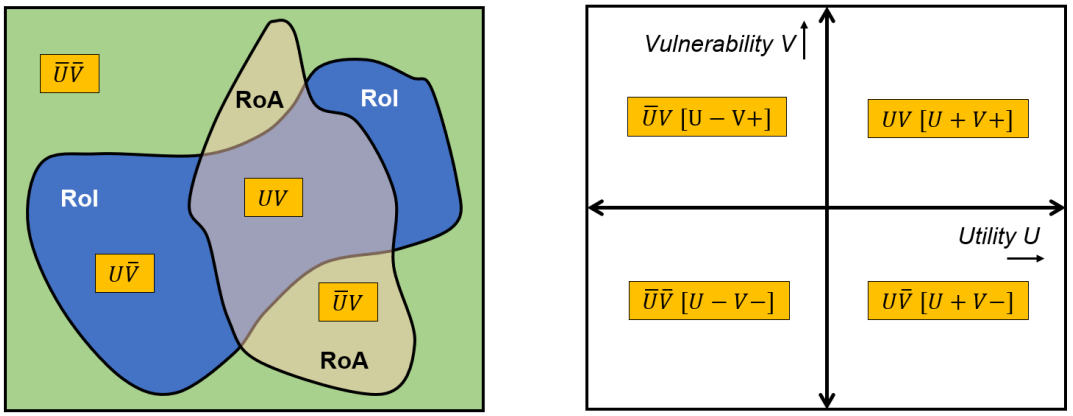


FIGURE 6.2: Segmentation of the input pixels into regions, mapped as per their leverage on Utility U, Vulnerability V.

6.2.6 Implications of isolating segments

There are some key takeaways from the framework described above. Noting them will assist us in progressing towards designing the experimental setup necessary to validate this theoretical formulation.

Properties of segmentation

Identifying X_p and X_q, and thereby generating the RoI and the RoA respectively using the aforementioned manner is possible for any given image and its corresponding adversarial sample. While the *Segregation* is universally possible for any given image and its corresponding adversarial sample, the spread of the RoI and the RoA on the image is subject to the choice of the thresholds (δ_1, δ_2) and more generally on the distribution of the pixels in X_p and X_q. This has to be verified empirically on real life datasets to check for the co-location property, which will offer evidence of the existence of well-defined regions. The study should be carried out on individual images and tested for if the observed patterns are consistent across multiple images of the class. A direct outcome of the above is the positioning of the isolated regions $UV, U\bar{V}, \bar{U}V, \bar{U}\bar{V}$. A more bounded and well knit RoI and RoA will result in better segments, each being distinctive in nature.

Neutralizing region $\bar{U}V$ [U-V+]

As evident from the formulation, the region $\bar{U}V$ is particularly vulnerable towards adversarial attacks. It is highly non-robust and vulnerable and also not useful for the task of classification. It is intuitive to therefore make use of this region and somehow block it from being picked up by the classifier in order to prevent adversarial mis-classification. This idea led us to develop a post-hoc adversarial defense mechanism. In theory, eliminating the information content in the region $\bar{U}V$, which we call the *Neutralization* process should be effective in prohibiting further adversarial vulnerability. One may choose to extract a blob, comprising of the overlapping areas of the region $\bar{U}V$ of all images in a particular class, and use it on potential test samples to avert mis-classification. We have tested this idea thoroughly in Section 9.6.

Furthermore, it may be mentioned that the choice of δ_1 and δ_2 for generating the RoI and the RoA should maximize the classification accuracy of the trained classifier and also maximally weakens the potential adversarial attack. This can thought of as a tuning parameter for each dataset.

It must be noted that for any image, that has been subjected to the process of classification by a trained classifier and an adversarial attack thereafter, one can identify the RoI and RoA, thereby setting up the necessities for splitting the image into the four aforementioned regions. This is universal, for a particular image or even at the representative level for a particular class of images, and its spatial distribution will depend on the co-location of the pixels belonging to RoI and RoA. Naturally, higher co-location will lead to well structured RoI and RoA, therefore leading to well-knit regions A,B and C. In our experiments, we have studied this in great detail over multiple datasets. As a direct outcome of the above analysis of spatial distribution of pixels in an image giving rise to specific implications, we have used the exact same idea to propose a defense mechanism against adversarial attacks. The intuition is that if we can observe spatial co-location of the RoI and RoA across the images belonging to a particular class (which is the axiom mentioned in the formulation, later verified empirically on multiple datasets), then for a particular class of images, using the representative RoI and RoA we can have the segmentation of representative regions of A, B, C and D. As mentioned earlier the region C is specifically crucial towards the success of an adversarial attack on any image, and so the class-representative region C would therefore be the

corresponding region sensitive to adversarial attacks for the entire class of images. Blocking out those set of pixels for all the images in the class would therefore serve as a method of hampering the attack. We have tested this idea of blocking or 'neutralizing' the pixels belonging to the region C for images belonging to a class by eliminating variation and studied how it affects the classification accuracy of the model on the distorted or 'neutralized' images and how the adversarial attack performs on it. For the purpose of the proposed 'neutralization', we consider the region C at the class-representative level and generate a modifiable blob out of it, which demarcates the region in each image belonging to that particular class that need to 'neutralized'.

A key notion that has to be appreciated is that this study of spatial distribution of pixels and their significance makes sense only when the entire pipeline, starting from the trained classifier to the adversarial attack method generating adversarial samples, is put in place. Therefore, the determination of the two key ingredients of our analysis, the RoI and the RoA can be made with a particular choice of δ_1 and δ_2 such that the corresponding RoI and RoA are so generated that maximizes the classification accuracy of the trained classifier and also maximally weakens the adversarial attack. This can thought of as a tuning parameter for each dataset. One must note that the proposed adversarial defense mechanism is post-hoc and will be useful only in the test scenario, wherein the model will the subjected to unseen images that belong to the same distribution of the training data of the classifier (basic assumption of any machine learning algorithm), on which our classification and adversarial attack has already been tested.

6.3 Investigating Co-located Spatial Patterns

In order to accomplish the objective of identifying and explaining the spatial patterns in the input space, that affect adversarial attacks, we have developed a pipeline that helps us test our assertions. The pipeline consists of three stages: *Segregation*, *Isolation* and *Neutralization*. We consider a binary classification problem (which has a fixed pathway of adversarial examples generation), its corresponding trained model M and an adversarial attack mechanism A, which has a set ϵ as its hyper-parameter. This remains the same throughout. The first component of the pipeline is to map the features from the feature space to the input space, that

is the pixels of the image. To identify the RoI at the individual level for each image, we use the technique of attention maps and identify the important features necessary for the classification task and project them back onto the input space. To identify the RoA at the individual level for each image, we look at the pixels that are modified by the attack module to generate the adversarial sample. This process is graphically represented in Figure 6.3.

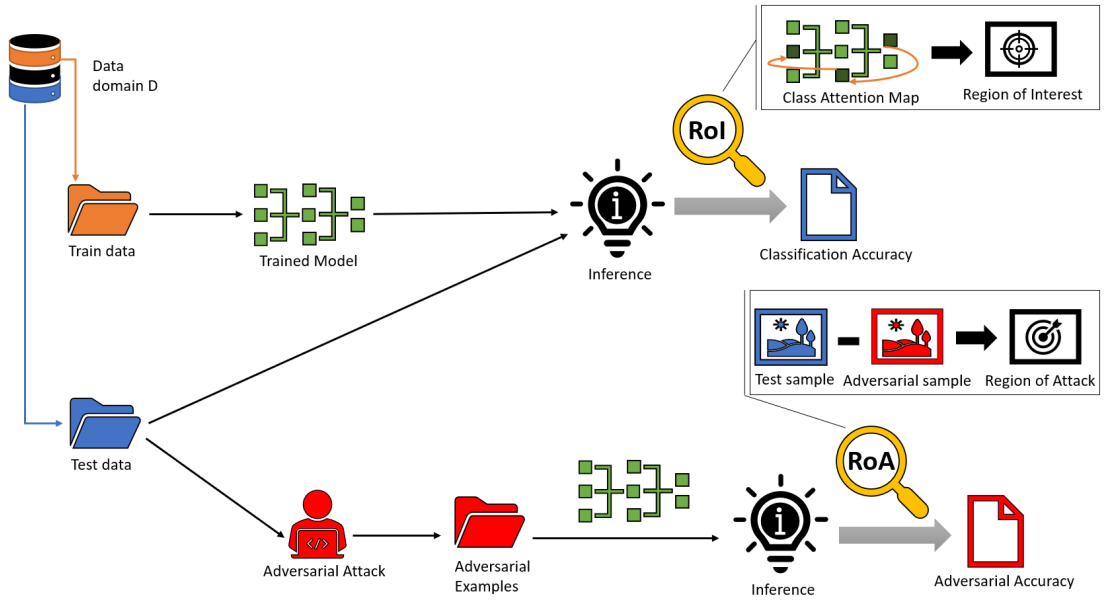


FIGURE 6.3: *Segregation*: Generating the RoI and the RoA for the task of classification and adversarial attack respectively.

The representative RoI is formed by considering the union of all pixels contributed by the individual images, meaning a superimposition or stacking up of the individual ROI for them. The same approach is also used to generate the representative RoA. Upon the identification of the RoI and the RoA, the second part of the pipeline carries out the *Isolation* of the segments as mentioned in the earlier section and pictorially described in Figure 6.4.

The third part of the pipeline is designed to study the post-hoc adversarial defense mechanism. The structure of analysis is as follows: We begin with the clean set of test samples S_{clean} and note the accuracy of M on it, and refer to it as $score_{clean}$. Then we use attack module A on S_{clean} to generate adversarial examples S_{adv}, and check the performance of M on it, referred to as $score_{adv}$. Post an adversarial attack, naturally we expect the accuracy $score_{adv}$ to be pretty low. We perform the *segregation* operation by generating the RoI and the RoA on S_{clean}, by making use of both S_{clean} and S_{adv}. Thereafter, we obtain the regions $UV, U\bar{V}, \bar{U}V, \bar{U}\bar{V}$.

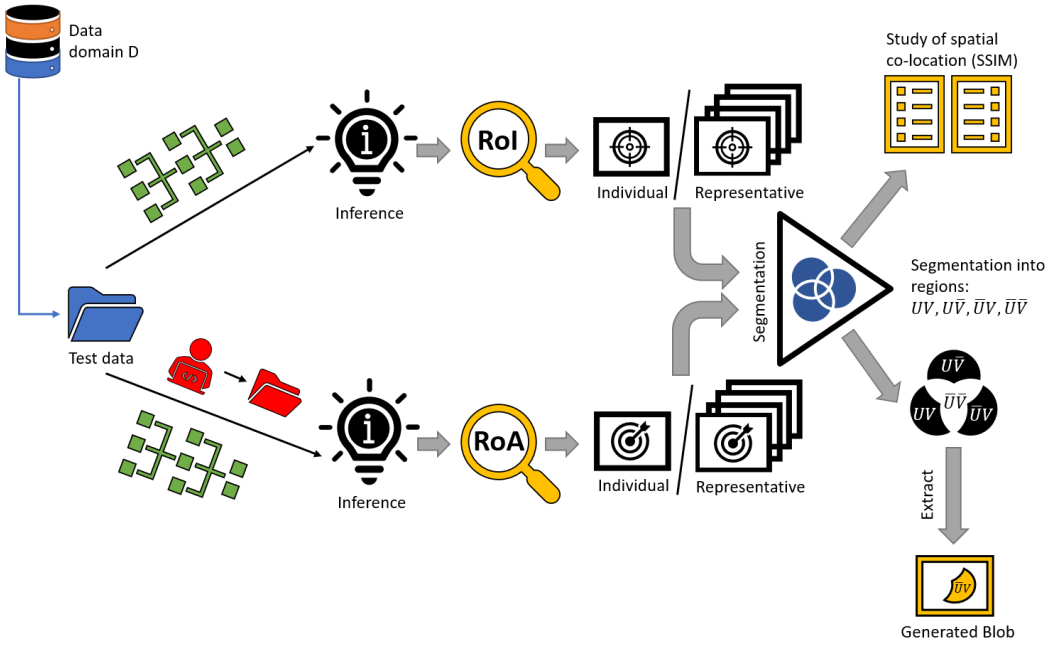


FIGURE 6.4: *Isolation* of regions $UV, U\bar{V}, \bar{U}V, \bar{U}\bar{V}$

At the class representative level we set a threshold θ for the determination of the region $\bar{U}V$ and extract the blob out. We note its spatial location (position of pixel indices) and use it to create S_{mod}, which is the modified version of S_{clean}, wherein the particular pixels in the location of the blob have been set to 0 or 1, to eliminate any variation (information) therein, therefore *neutralizing* those pixels. We use the original model M on S_{mod} and note its performance $score_{mod}$ to see what impact the *Neutralization* process has on classification accuracy. For the final part, we make use of the adversarial attack module A with same ϵ on S_{mod} to generate further adversarial examples $S_{mod-adv}$, as shown in Figure 6.5. We use the model M on $S_{mod-adv}$ to see its performance, referred to as $score_{mod-adv}$. Intuitively, if our proposed post-hoc adversarial defense mechanism works, then we expect $score_{clean}$ and $score_{mod}$ to be comparable, meaning the addition of blob doesn't impact classification much as it alters region $\bar{U}V$ primarily, whereas $score_{mod-adv}$ to be greater than $score_{adv}$ meaning the adversarial attack on the neutralized samples has become weaker.

The key aspects of investigation through the utilization of the aforementioned pipeline are listed below. Obeying the key intuition of specificity between the paths of adversarial attack among trained manifolds of particular classes, namely a 'Source' class and a 'Target' class, we study the following.

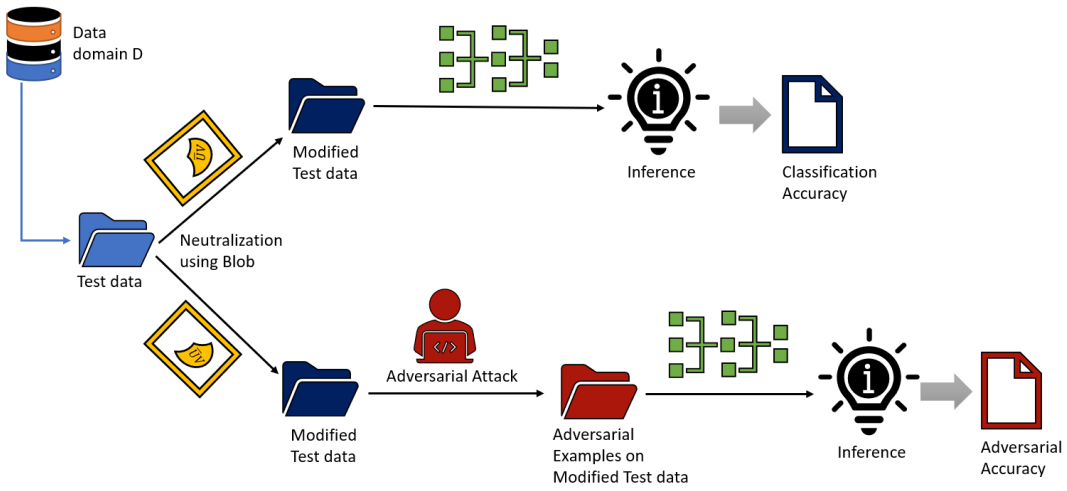


FIGURE 6.5: Effect of *Neutralization* on classification and further adversarial vulnerability

- the co-location of pixels that contribute towards classification, therefore the existence of the Region of Interest (RoI) for individual images
- the co-location of pixels that contribute towards the adversarial attack, therefore the existence of the Region of Attack (RoA) for individual images
- the co-location of pixels that contribute towards classification when images of a particular class are aggregated (superimposed), therefore the existence of the Region of Interest (RoI) for the representative at the class level
- the co-location of pixels that contribute towards adversarial attack when images of a particular class are aggregated (superimposed), therefore the existence of the Region of Attack (RoA) for the representative at the class level
- similarity in spatial positioning (overlap) of the RoI and the RoA for each individual image and for the representatives at the class level
- segregating the region $\bar{U}V$ [U-V+] at the class representative level to generate the blob (to be used for *Neutralization*) by choosing a threshold of overlap
- effect of *Neutralization* on classification accuracy, that is using the generated blob for the class of images to neutralize the images at the individual level, and thereafter passing through the trained classifier

- effect of *Neutralization* on further adversarial vulnerability, that is using the generated blob for the class of images to neutralize the images at the individual level, and thereafter carrying out further adversarial attack on the modified images and check their performance

6.4 Experimental Results

In this section, we present the summary of our results of extensive experimentation to verify the propositions made in the earlier sections. In general, we used standard datasets and a well known image classifier VGG16 [1] for ensuring that the results are comparable and reproducible. The datasets used include MNIST [5], Fashion-MNIST [102], Cifar-10 [6], the Cats vs Dogs dataset [103] and the Malaria dataset [104]. For the generation of the adversarial attacks, we used the well documented FGSM [2] attack module consistently throughout our experiments, as it provides a lightweight and easy to implement practical attack.

6.4.1 Observations

Due to constraints of content, we have not been able to report all results within the work and they are made available for perusal at the GitHub repository here. For understanding and putting forward our argument, we present a subset of the results. We consider two classes each for MNIST (0 and 1) and Fashion-MNIST (Trousers and Pullovers) randomly, and study it as a binary classification problem. The rest two are binary classification problems by default. Therefore, the snapshot of results presented hereafter are for the following: (a) MNIST - two classes (0 and 1), (b) Fashion MNIST - two classes (Trousers and Pullovers), (c) Cats vs Dogs - two classes (Cats and Dogs) and (d) Malaria - two classes (Parasitized and Uninfected).

To study the existence of a co-located Region of Interest (RoI) for individual images

We set a tuned parameter for threshold δ_1 and generate the RoI for all images in the class after classifying the images using the trained model M and using the

class attention map. We carry out a physical overview of it and for documentation, select one image at random and present its RoI. This process is then repeated for all classes in the dataset. The sample results are presented in Figure 6.6. For each of the examples, the region marked in blue is the region corresponding to area of highest leverage on the classifier for the task of classification.

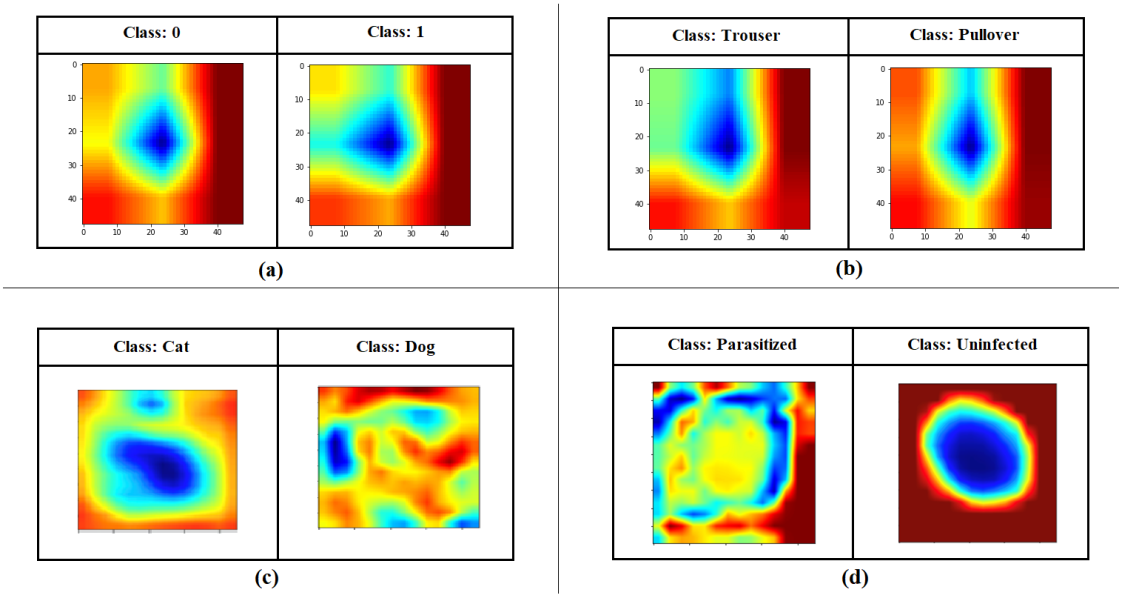


FIGURE 6.6: Study of Existence of RoI in individual images for the datasets: (a) MNIST, (b) Fashion MNIST, (c) Cats vs Dogs and (d) Malaria.

To study the existence of a co-located Region of Attack (RoA) for individual images

We set a tuned parameter for threshold δ_2 and generate the RoA for all images in the class after carrying out an adversarial attack on it using an attack module A. We carry out a physical overview of it and for documentation, select one image at random and present its RoA. This process is then repeated for all classes in the dataset. The sample results are presented in Figure 6.7. For (a) MNIST and (b) Fashion MNIST, the regions in black are corresponding to the regions maximally modified by the adversarial perturbation. For (c) Cats vs Dogs and (d) Malaria, the region of RoA are in violet.

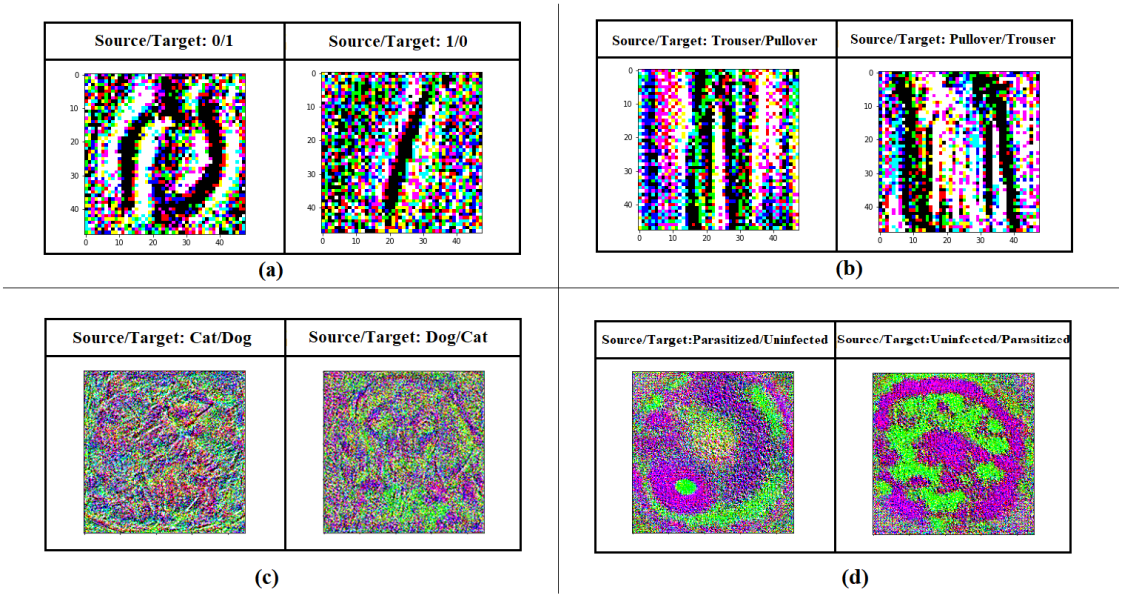


FIGURE 6.7: Study of Existence of RoA in individual images for the datasets: (a) MNIST, (b) Fashion MNIST, (c) Cats vs Dogs and (d) Malaria.

To study the existence of a co-located Region of Interest (RoI) for all images belonging to a particular class

After the generation of the individual RoI, the images are superimposed and the mean value is considered pixel-wise to create the representative RoI for the class, which is then presented in Figure 6.8. This is to look for the existence of overlap between the RoI of images in the class. The colour code naturally remains the same as that of the individually identified RoI.

To study the existence of a co-located Region of Attack (RoA) for all images belonging to a particular class

After the generation of the individual RoA, the images are superimposed and the mean value is considered pixel-wise to create the representative RoA for the class, which is then presented in Figure 6.9. This is to look for the existence of overlap between the RoA of images within the class.

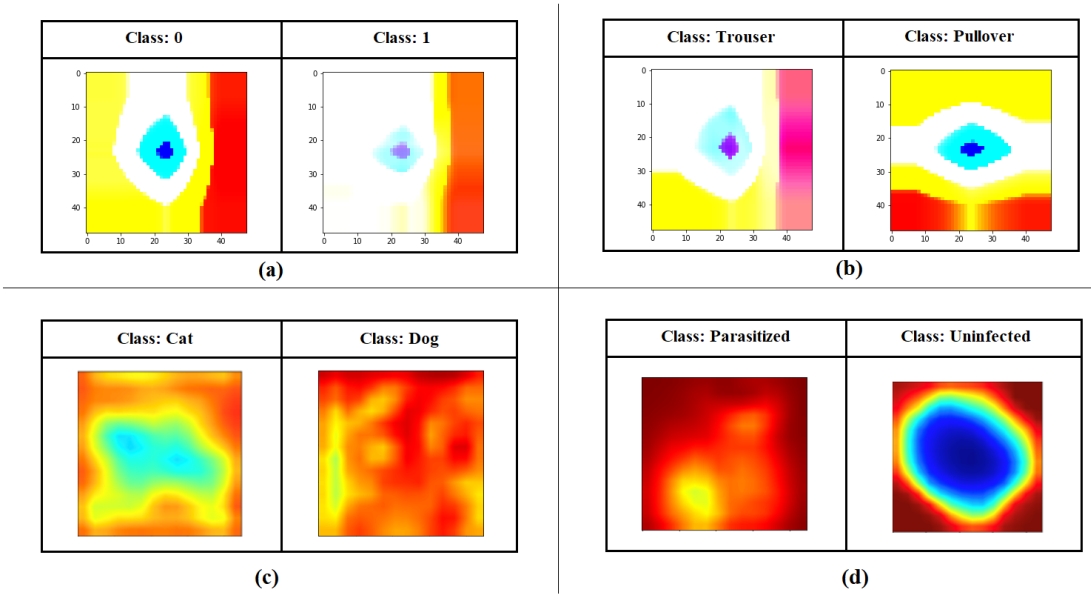


FIGURE 6.8: Study of Existence of RoI at the class representative level for the datasets: (a) MNIST, (b) Fashion MNIST, (c) Cats vs Dogs and (d) Malaria.

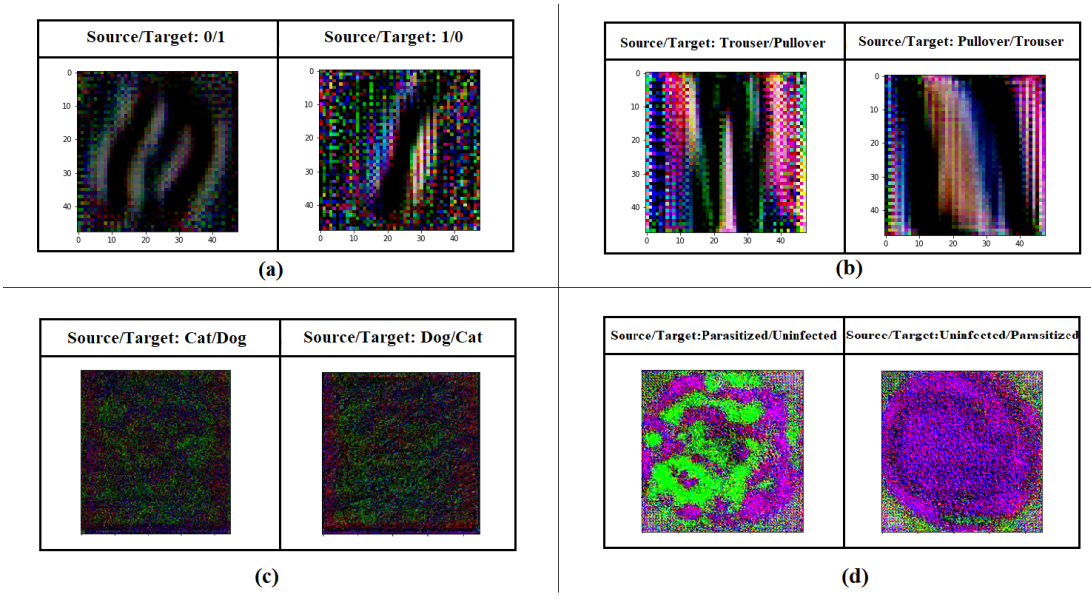


FIGURE 6.9: Study of Existence of RoA at the class representative level for the datasets: (a) MNIST, (b) Fashion MNIST, (c) Cats vs Dogs and (d) Malaria .

To study the overlap in spatial positioning of the RoI and RoA for individual images and for the representatives at the class level

For every image belonging to the particular class, the RoI and the RoA are already determined. To study their spatial positioning and overlap, the pairwise structural

similarity is measured using SSIM and class-level mean is reported. For the representatives at the class level, the SSIM of the corresponding representative versions is reported in Table 6.1.

TABLE 6.1: Structural similarity of RoI and RoA

Dataset	Source Class / Target Class	SSIM value (mean of samples)	SSIM value (representatives)
MNIST	0 to 1	0.2178	0.0039
	1 to 0	0.1978	0.0001
Fashion MNIST	Trousers to Pullover	0.1923	-0.0019
	Pullover to Trousers	0.2029	-0.0003
Cats vs Dogs	Cats to Dogs	0.3028	0.0899
	Dogs to Cats	0.3046	0.0938

Generation of neutralizer blob

To demonstrate the process of *Neutralization*, we *isolate* the region $\bar{U}V$ [U-V+] for each class of images. This generates the blob that indicates the region of highest adversarial vulnerability. The *Neutralization* process sets all the pixels to its highest/lowest possible value, without a loss of generality, to eliminate its specific information content. The generated blobs are presented in Figure 6.10.

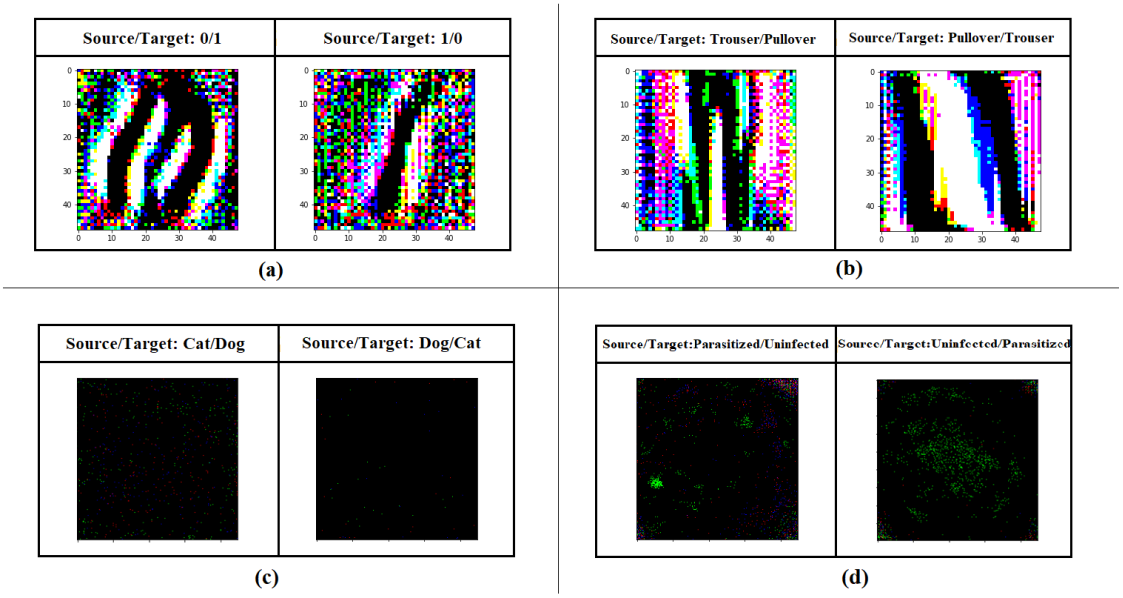


FIGURE 6.10: Blobs for the datasets: (a) MNIST, (b) Fashion MNIST, (c) Cats vs Dogs and (d) Malaria.

To study the effect of *Neutralization* on classification accuracy

We report $score_{clean}$ and $score_{mod}$ for the four datasets in Table 6.2.

TABLE 6.2: Effect of *Neutralization* (blob based, *isolated* vulnerable region) on the task of classification.

Dataset	Source Class / Target Class	Classification accuracy (M)	
		$score_{clean}$	$score_{mod}$
MNIST	0 to 1	100%	93.75%
	1 to 0	100%	96.88%
Fashion MNIST	Trousers to Pullover	100%	96.88%
	Pullover to Trousers	100%	96.88%
Cats vs Dogs	Cats to Dogs	98.97%	85.35%
	Dogs to Cats	99.87%	99.89%
Malaria	Parasitized to Uninfected	99.79%	99.68%
	Uninfected to Parasitized	99.21%	99.56%

To study the effect of *Neutralization* on further adversarial vulnerability

We report $score_{adv}$ and $score_{mod-adv}$, for the four datasets in Table 6.3.

TABLE 6.3: Effect of *Neutralization* (blob based, for vulnerable region) on the potential of further adversarial attack.

Dataset	Source Class / Target Class	Classification accuracy (M)	
		$score_{adv}$	$score_{mod-adv}$
MNIST	0 to 1	90.62%	100%
	1 to 0	96.85%	100%
Fashion MNIST	Trousers to Pullover	90.62%	90.62%
	Pullover to Trousers	96.85%	100%
Cats vs Dogs	Cats to Dogs	3.47%	3.49%
	Dogs to Cats	7.37%	2.67%
Malaria	Parasitized to Uninfected	5.85%	46.11%
	Uninfected to Parasitized	7.55%	45.06%

6.4.2 Key Findings

The key aspects to note, based on the overall experimental results, are as follows:

- From the study of the existence of the Region of Importance (RoI) and the Region of Attack (RoA), both at the individual level and at the class representative level, we understand that the segregation is possible, as there exists a good co-location of pixels, which form observable clusters, with specific significance for each as explained. It is more apparent than others in some cases, depending on the particular datasets and their inherent complexities. It must be noted that the co-location of regions at the representative level holds good only for samples which are free of translational variance.

- Upon the identification of the the regions, the splitting up of the image's input space into four disjoint segments ($UV, U\bar{V}, \bar{U}V, \bar{U}\bar{V}$) is also feasible. The shapes and sizes of these parts, vary from dataset to dataset, and is highly dependant on the overlap that exists between the RoI and RoA for that particular class of images.

- Extending the findings to adopt a practical use of these patterns, the idea of attempting to neutralize the thus-identified vulnerable areas (position demarcated by the blob), could be done in multiple ways. One way, as adopted in this work, is to eliminate the information content of those pixels by *Neutralization*, thereby reducing vulnerability.

- The proposed post-hoc defense mechanism, obtained by using the segmentation of regions, works. The experimental results show that while the act of *Neutralization* does not affect the task of classification much, it is able to reduce further vulnerabilities of the samples.

6.5 Conclusions

To study the patterns within the distribution of pixels within an image and how they may relate to the context of its classification and adversarial attacks, we found empirical evidence of spatial co-location among pixels that have particular significance. We have defined an Region of Importance (RoI) and an Region of Attack (RoA) that correspond to the areas in the image specifically responsible for the task of classification and rendering adversarial vulnerability. We have demonstrated through multiple datasets that although their positions and distribution vary, and their existence may be more apparent in some cases than the rest, these

patterns can be used for a better understanding of specific adversarial attacks. In fact, this knowledge has been used to develop a post-hoc adversarial defense mechanism that tries to neutralize the regions of the image particularly vulnerable towards an attack. This is significant towards the reliable use of ML models. In the future, we plan to test out the proposed mechanism for a more extensive set of data and neural architectures and attack modules. The exhaustive experimentation will contribute towards higher generalization. This should help with better modes of identifying the regions RoI and RoA followed by isolation may help us in improving the segregation which will benefit the neutralization born out of it.

Part II

Privacy

Privacy Preserving Machine Learning

> Data is the pollution problem of the information age, and protecting privacy is the environmental challenge.
>
> *Bruce Schneier*

In the second part of the thesis, we consider robustness of the machine learning algorithms from the perspective of dealing with Privacy issues. The threats involved here do not affect the fundamental machine learning task, but reveals sensitive information about the system that may have severe consequences. This is particularly important when the training data is sensitive and expensive to curate, there is a heavy involvement of resources in the neural architecture search and the model training requires high-end hardware resources.

We consider the cases for the privacy of the training data and that of the trained models separately. We study watermarking on neural networks for checking the authenticity of models and develop robust watermarking schemes for verifiable proof of ownership of neural networks. For the protection of privacy of training data, we consider decentralised learning architectures where models can be trained to perform well at inference without sharing the data using federated learning infrastructures. We addressed the trust models associated with each use case separately and added components for privacy protection.

Chapter 7

Re-Markable: Stealing Watermarked Neural Networks through Synthesis

7.1 Introduction

[1] There are three essential components towards the success of this system of generating insights driving decision making using machine learning techniques [70]. First and foremost is a large volume of curated data which is rich in information. Secondly, we need a neural network architecture comprising of the computational graph and the weight matrices. And finally, the hardware infrastructure is critical, that is able to train the network, that is to tune the weights or the parameters using the available data. Therefore, for accomplishing a particular machine learning task, the corresponding stakeholders need to invest heavily in:

- obtaining the necessary data, curating it and annotating it if that's necessary (especially for supervised machine learning tasks)
- optimizing the neural architecture best suited for the task

[1] Chapter 7 is published as *Nandish Chattopadhyay, Chua Sheng Yang Viroy and Anupam Chattopadhyay, "Re-Markable: Stealing Watermarked Neural Networks through Synthesis," in Proceedings of SPACE 2020*

- heavily investing in powerful hardware infrastructure that is able to successfully train the neural network using the data so as to obtain the deploy-able trained model

Naturally, therefore anyone investing in any one or multiple of the aforementioned components would wish to establish some sort of authority and rights on their assets. It may be mentioned here that all of these tasks need not be accomplished in one place, and can be carried out separately by different parties mutually agreeing on a common goal. Whichever way, the trained model, which is the outcome of this entire exercise is a valuable asset for those involved.

Machine Learning as a Service (MLaaS):

This is a highly evolving and naturally competitive market. Such sort of service providers claim to offer fully trained models to their customers at a cost, which in comparison to what the price would have been if they had to train the model from scratch using high-performing hardware resources, is almost negligible. Over and above that, the customers are provided the freedom to further fine-tune the model in order to improve its performance over time, whilst gathering further data from its sources, or perform some kind of transfer learning [105] thereafter to extract high-level features to accomplish similar tasks. Apart from this kind of open source models, MLaaS lets its customers create specific and often more personalized applications without having to delve deep into the training of the neural networks. While it looks quite simple apparently, this idea gives rise of a bunch of key questions pertaining to security and legality, particularly in the context of IP rights. A company that plays in the market of MLaaS may consider the eventualities that its customers who purchase a trained model from it, may indulge in malpractices. Typical examples of such breaches in contract may include redistribution of the model beyond what is legally permitted as per legal documentation, or even further selling of it, to other parties, thereby posing a threat to its business. Such scenarios of potential theft should be avoided at all costs. Therefore there is an imminent requirement of developing a robust procedure for authenticating a trained neural network model.

This issue has become prominent in the field of machine learning research as more and more solutions are being brought to the market, and thus giving rise to new

associated problems. Being a rather new topic in the machine learning research community, some researchers are looking to borrow from the security community, particularly in the VLSI security research groups where this particular problem is well documented and studied as part of digital watermarking [106]. Digital Watermarking is formally defined as "the process of robustly concealing information in a signal (e.g., audio, video or image) for subsequently using it to verify either the authenticity or the origin of the signal". The idea of digital watermarking has found a lot of applications in many aspects of digital media including images, videos and audio signals. However, it must be noted that the literature in digital watermarking, although diverse, is not trivially applicable to the context of watermarking neural networks. This is specifically due to the challenge of embedding the watermarks within the neural network model, without bringing down its performance on its actual designated task. One needs to carefully map the techniques existing in the field of watermarking to machine learning models.

7.1.1 Motivation

Inspite of such competition among watermarking schemes and attacks, the most accepted and widely used mechanism, which is thought to be robust against attacks, is watermarking using backdooring [3]. This approach targets the overparameterized nature of a neural network to implement a watermarking algorithm. Given its acceptance and prevalence, it is important to review its capabilities and vulnerabilities before it is used to practical deployment in the market. This approach uses backdooring, typically a flaw observed in neural networks, to its advantage to embed specific watermarks within the neural network model. These are planted by corresponding modifications of the weights. While being very effective in both embedding the watermarks and verification afterwards when necessary to prove ownership, it borrows caveats from the research into vulnerabilities of backdoors. While they may be of less adversity towards this novel approach, it is very crucial to study its own weaknesses against potential attacks by adversaries. Typical mechanisms of stealing watermarked models, primarily Evasion attacks like Ensemble attacks and Detector attacks don't work well against this scheme, as further discussed in Section 7.3.2. This watermarking mechanism is not robust against our proposition of a model modification attack nevertheless. This necessitates a detailed study in exploring and mitigating the associated vulnerabilities.

7.1.2 Contribution

In this work, we explore the vulnerabilities of the watermarking scheme with backdooring and propose a model modification attack mechanism to extract or steal the trained model. Our contribution includes:

- proposition and demonstration of two kinds of model extraction mechanisms involving specific re-training the watermarked model to remove signatures
- use of synthesis by Generative Adversarial Networks to re-create samples which emulate samples derived from the same distribution as that of the training data (considering the adversary is aware of the data domain) to retrain the model

7.2 Theoretical Formulation

In order to arrive at the task of exploring potential vulnerabilities is digital watermarking schemes for neural networks, it is imperative to briefly look at the key components of the overall problem and its evolution. In this section, we discuss the idea of watermarking in general, the existence of backdoors in neural networks and how they can be utilised to watermark trained models and finally look at the process of synthesis whereby we get acquainted with the tools to attack the existing scheme. Throughout the description we have some standard nomenclature defined here. We make an assumption about the existence of a ground-truth function f that is always able to correctly classify inputs by assigning correct labels. A machine learning model attempts to approximate that function. The data in the context of the task belongs to the domain D. That way, the ground-truth function f maps $f : D \rightarrow L$, L being the set of correct labels for each sample in D. Naturally, it is not possible to realise all samples in D, and therefore we work with a dataset, which has two sets: $train\_data$ with its correspondingly mapped L_t and $test\_data$. The objective of approximating f is fulfilled by learning a neural network model M on the $train\_data$. Therefore, we have two functions, $Train(train\_data, L_t)$ and $Classify(M, test\_data)$. Having mentioned this, we can delve deeper into the formulation.

7.2.1 Watermarking

A typical watermarking scheme would consist of three major components. Let us assume we have a curated training dataset called *train_data* and a trained neural network model M. Firstly, there is a requirement of an algorithm to create a secret key m_k for the purpose of marking, which the marker will embed as the watermark, and its corresponding public key v_k for verification designed for detection and therefore verify the watermark at a later point of time to establish rights. Secondly, one needs another algorithm to place the watermark into the object, which in this particular case happens to be the neural network model. Thirdly, there needs to be another algorithm that involves both the secret key m_k for marking and the public key v_k for verification. These algorithms can therefore be stated as:

- $KeyGeneration()$: returns the pair of marking and corresponding verification keys (m_k, v_k)
- $Marking(M, m_k)$: takes as parameter an input neural network model and a secret marking key m_k, returns a watermarked model $\hat{M}$
- $Verification(m_k, v_k, M)$: takes as parameters the marking and verification key pair (m_k, v_k) and the watermarked model $\hat{M}$, returns the output bit $b \in \{0, 1\}$

The success of the watermarking scheme depends on the correct functioning of all the three aforementioned algorithms $(KeyGeneration, Marking, Verification)$ together. The notion of correctness can be formally described as:

$$Pr_{(M,\hat{M},m_k,v_k)\leftarrow WM()}[Verify(m_k, v_k, \hat{M}) = 1] = 1 \tag{7.1}$$

where, the watermarking algorithm $WM()$ is made up of the following:

1. Creating $M \leftarrow Train(train\_data)$
2. Sampling $(m_k, v_k) \leftarrow KeyGeneration()$
3. Computing $\hat{M} \leftarrow Marking(M, m_k)$
4. Return $(M, \hat{M}, m_k, v_k)$

There are some key properties that any particular watermarking scheme should satisfy. They are enlisted here:

- Functionality-preserving: The accuracy of the model in accomplishing the designated task doesn't change with the introduction of the watermarks.
- Non-triviality of ownership: The knowledge of the watermarking algorithm wouldn't disclose the key pairs to an adversary.
- Unremovability of watermarks: The adversary would not be successful in removing the watermark with the help of the knowledge of the algorithm and the existence of a watermark.
- Unforgeability of watermarks: Just the knowledge of the verification key is not sufficient to establish ownership.

Another key aspect of ensuring the proper functioning of the watermarking scheme is maintaining the sanctity and uncompromised privacy of the private key. Commitment schemes are used for this purpose as explained hereafter.

Commitments:

Mechanisms of establishing commitment schemes are rather common in the field of cryptography. These algorithms make sure that the sender is able to lock some secret information into vault that can't be tampered with, before sharing it with the corresponding receiver. The receiver is unable to extract information from the vault single-handedly without the sender (hiding property). Similarly, the sender is also unable to modify the locked information once the receiver has received it (binding property).

Typically therefore, a formal definition of a commitment scheme would consist of two algorithms as listed here:

- $Commitment(x, r)$ takes as input a value $x \in S$ and a bitstring $r \in \{0,1\}^n$ and returns a bitstring c_x
- $Release(c_x, x, r)$ takes as input $x \in S, r \in \{0,1\}^n, c_x \in \{0,1\}^*$ and returns 0 or 1.

This holds good for the generic idea of watermarking, particularly watermarking neural networks. The exact details of the watermarking scheme and therefore the description of each of the aforementioned algorithms is presented later, after discussing some related key ideas that make it possible.

7.2.2 Backdoors in Neural Networks

Originally observed a flaw in neural networks, backdooring is a specifically designed technique to intentionally train a machine learning model to return erroneous (in comparison to the naturally occurring ground-truth function$f$ and its labels L) labels T_L for a set of inputs T. Hence, we can define $T \subset D$ to be a subset of samples from domain D, which is called the Trigger Set. The labels corresponding to the samples present in the Trigger Set, which are different from what the ground truth function f would have returned, is captured by T_L. An explicitly trained model therefore is expected to associate these labels when tested with samples from the Trigger Set. The backdoor therefore is this pair of Trigger Set and its corresponding labels taken together, $b = (T, T_L)$. Defining the Trigger Set implies defining T_L as well.

Considering such a backdoor b, one can define a backdooring algorithm *Backdoor* that takes as input a neural network model, and returns the model with the backdoor b embedded in it, such that the new model performs very poorly on the Trigger Set with respect to the ground-truth f, but is able to match the output labels with T_L, with high probabilities. The success of the backdooring depends on the extent of this match. It is worth mentioning here that this process of embedding a backdoor b in the model M to create the backdoored/watermarked model $\hat{M}$ can be carried out in two methods:

- The *Backdoor* algorithm can use the existing trained neural network M and embed the backdoors in it, this mechanism of marking being referred to as backdooring a pre-trained model.
- Otherwise, the algorithm can also train the model $\hat{M}$ from scratch. This approach is similar to that of intentional data poisoning. This naturally is more time-consuming, and the original model M is just used to benchmark the accuracy required to make M and $\hat{M}$ comparable in performance.

Having touched upon the principles of Watermarking and Backdooring, we can now proceed to looking at the way backdoors can be used as watermarks. Thereafter, we shall demonstrate the attack by exposing its vulnerabilities.

7.2.3 Watermarking using Backdooring

The method of implementing a watermarking scheme using backdooring has been well studied [3]. Essentially, there is an algorithm that embeds the backdoor in the neural network model, the backdoor itself being the secret marking key while the commitment works as the public verification key.

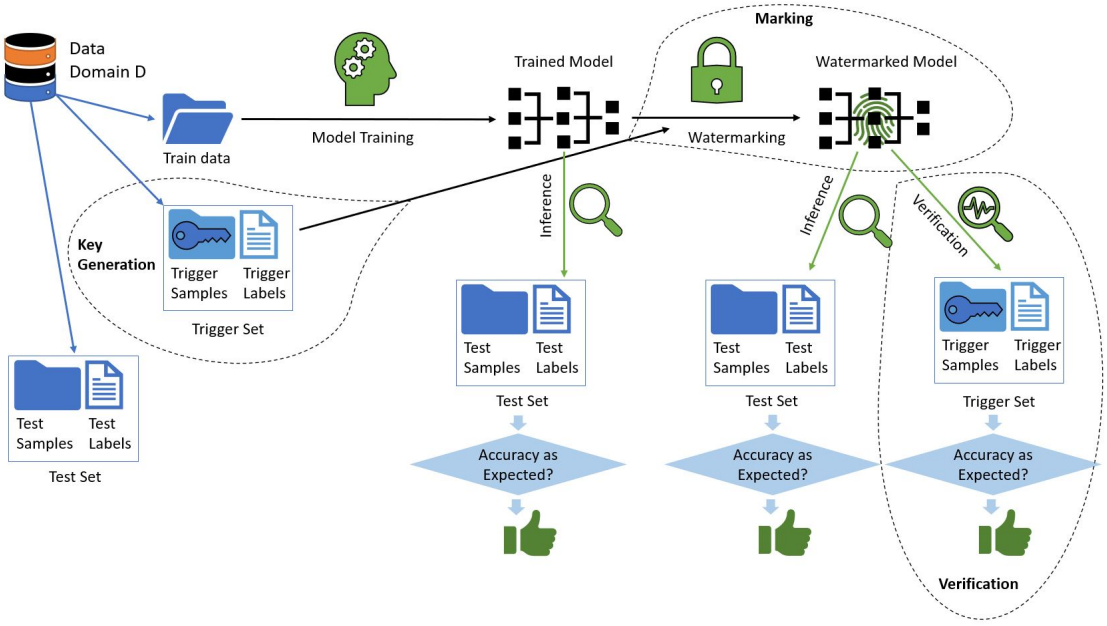


FIGURE 7.1: Schematic diagram of watermarking using backdooring.

To formalise, the mechanism described by the authors proposing to use backdooring as a means of watermarking and keeping the formulation discussed earlier in place, we have:

$KeyGeneration()$:

1. Execute $(T, T_L) = b \leftarrow BackdoorSample(train_{data})$
   where $T = \{t^{(1)}, \ldots, t^{(n)}\}$ and $T_L = \{T_L^{(1)}, \ldots, T_L^{(n)}\}$.

2. Draw $2n$ random strings $r_t^{(i)}, r_L^{(i)} \leftarrow \{0,1\}^n$
   and create $2n$ commitments $\{c_t^{(i)}, c_L^{(i)}\}_{i\in[n]}$
   where $c_t^{(i)} \leftarrow Commitment(t^{(i)}, r_t^{(i)}), c_L^{(i)} \leftarrow Commitment(T_L^{(i)}, r_L^{(i)})$.

3. Set $m_k \leftarrow (b, \{r_t^{(i)}, r_L^{(i)}\}_{i\in[n]}, v_k \leftarrow \{c_t^{(i)}, c_L^{(i)}\}_{i\in[n]}$
   and return (m_k, v_k).

$Marking(M, m_k)$:

1. Consider $m_k = (b, \{r_t^{(i)}, r_L^{(i)}\}_{i\in[n]}$

2. Make the computation to return $\hat{M} \leftarrow Backdoor(train_{data}, b, M)$

$Verification(m_k, v_k, M)$:

1. Let $m_k = (b, \{r_t^{(i)}, r_L^{(i)}\}_{i\in[n]}, v_k = \{c_t^{(i)}, c_L^{(i)}\}_{i\in[n]}$.
   For $b = (T, T_L)$, test if $\forall t^{(i)} \in T : T_L^{(i)} \neq f(t^{(i)})$.
   If not then return 0.

2. For all $i \in [n]$, match for
   $Release(c_t^{(i)}, t^{(i)}, r_t^{(i)}) = 1$ and $Release(c_L^{(i)}, T_L^{(i)}, r_L^{(i)}) = 1$.
   Otherwise return 0.

3. For all $i \in [n]$, test that $Classify(t^{(i)}, M) = T_L^{(i)}$.
   If true for all but $\epsilon|T|$ elements of T, the return 1 or else return 0.

In the model modification attack proposed in this work, we shall attempt to extract the model which has been watermarked using backdooring.

7.2.4 Synthesis using GANs

Generative models are a nice way of leveraging the abilities of managing high-dimensional probability distributions, by learning means to represent and manipulate them as needed. These models are good for various kinds of simulation tasks. In situations like ours, wherein we need to perform synthesis of data by learning the distribution of it from a subset of samples belonging to the same domain, these models are a perfect tool. The state-of-the-art in this area of research would point

to Generative Adversarial Networks (GANs). Some reasons for its success include their overall superiority in the subjective quality of the generated samples, their ability of successfully parallelize well, the design of the generator function which has lesser restrictions, the lack of variational bounds and that Markov chains are not necessary. Keeping these in mind, we chose the GANs as out choice of tool for performing the task of synthesis.

GANs work on a game theoretic approach. A game is set up between two entities, a Generator and a Discriminator. The Generator attempts to recreate samples from the distribution underlying the training dataset. The Discriminator classifies the samples thus generated into whether they are real or fake. It is much like a traditional supervised machine learning model, that splits the inputs into real/fake. The Generator on the other hand, is trained to fool the discriminator. Formally, the two players in this so-called game are represented by differentiable functions. These functions are differentiable with respect to inputs and parameters alike. The Discriminator function C takes x as its input along with parameters $\theta^{(C)}$. The Generator function G takes z as its input and its parameters $\theta^{(G)}$. The cost functions of both players are defined in terms of each other's parameters. The Discriminator has control only over its parameters $\theta^{(C)}$ and tries to minimize a certain cost function $J^{(C)}(\theta^{(C)}, \theta^{(G)})$. The Generator function has control only over its parameters $\theta^{(G)}$ and it attempts to minimize $J^{(G)}(\theta^{(C)}, \theta^{(G)})$. Since each player has leverage only on its own parameters, the solution to this kind of a game is a Nash equilibrium. For this particular situation, a Nash equilibrium is a tuple $(\theta^{(C)}, \theta^{(G)})$ which happens to be the local minimum of $J^{(C)}$ with respect to $\theta^{(C)}$ and also a local minimum of $J^{(G)}$ with respect to $\theta^{(G)}$.

DCGAN:

The DCGAN (Deep Convolutional Generative Adversarial Network) architecture is the most widely accepted and used version of GANs [107]. The specifics of the DCGAN structure include, but are not limited to the following:

- The Generator and the Discriminator network both have layers for batch normalizing. The Discriminator has separate normalization. For the Generator, the last layer doesn't have normalization. The initial layer of the Discriminator also doesn't have normalization. This is aimed at making the synthesized

samples in sync with the scale and mean of the data distribution which it tries to emulate.

- The structure of the network is quite similar to that of the all convolutional net [108], particularly lacking pooling and unpooling layers. If the Generator requires higher dimension for spatial representation, the stride is set to greater than unity.
- The optimizer of choice is the ADAM optimizer instead of the traditional batch Stochastic Gradient Descent with momentum.

7.3 Exposing Vulnerabilities

In order for stake-holder to make use of watermarking schemes based on backdooring in a reliable fashion, one needs to study the vulnerabilities of the mechanism. Primarily, what must be noted right at the onset is that the success of the watermarking scheme is dependant on three key aspects, simply put as follows:

- Effective embedding of the watermarks (using backdooring for example) within the trained neural network model
- Confidentiality of the key that is used to watermark (the Trigger set and its corresponding labels for example)
- Reliable verification of the watermark when needed to establish ownership

7.3.1 Criteria for Model Extraction

An adversary aims to attack the watermarked model by negating the aforementioned requirements. Thereby, an issue created by exploiting vulnerabilities related to any one or more of the above, could potentially jeopardise the entire watermarking mechanism. In particular, an adversary would be able to steal a trained neural network model by such attacks. The criteria for an attack mechanism to be successful are:

1. The extracted model must be similar in performance with respect to the machine learning task for the particular domain of data, in this case on the Test Set.

2. The process of verification of the embedded watermarks must fail, i.e., the model must perform poorly on the Trigger Set here.

7.3.2 Modes of Attack

There are multiple modes of attack that can be used for exploiting the vulnerabilities of watermarking schemes and stealing models. We look at two common mechanisms available in the literature, which are Evasion attacks, and mention their shortcomings. While having their own advantages in being effective against certain watermarking schemes, they have not been successful against the state-of-the-art watermarking mechanism using backdooring. In order to overcome this, we introduce the model modification attack to expose vulnerabilities of this so called robust mechanism.

Failure of Evasion Attacks:

There are two predominant modes of Evasion attack, Ensemble attacks and Detector attacks. In the Ensemble attack mechanism, the adversary leverages the capabilities of multiple trained neural network models simultaneously, by creating an ensemble of models which are all made to perform the same task [58]. Let us consider that all the models have their own watermarks, and would behave in particular ways when subjected to specific inputs. The key intuition behind this sort of an attack is that while each watermarked model is trained to respond to a sample from its Trigger set in a particular way, the probability of that particular sample belonging to all the Trigger Sets of all the models, is very low [60]. However, the provision of obtaining multiple trained models is costly, and the majority voting principle is not very reliable. Moreover, all the models should be able to work simultaneously in a synchronous fashion for this mechanism to work.

There is another kind of attack that an adversary may use to work against any particular watermarking scheme, known as Detector attacks [58]. This attack

mechanism is aimed at training a special deep neural network that is able to distinguish between a clean input and an input which is part of the Trigger set. However, the accuracy of this network is difficult to improve, particularly due to the unbalanced nature of the distribution of clean samples and those belonging to Trigger sets. This attack fails when the trained classifier is unable to detect the Trigger samples from amongst clean ones.

Having touched upon the attacks and why they have not been successful against robust watermarking schemes, we introduce our proposition of a model modification attack using synthesis.

7.3.3 Proposed Solution - Model Modification Attack:

It is worthwhile noticing that the aforementioned Ensemble attacks or Detector attacks, being both Evasion attacks, are not able to remove the watermarks from the marked models. This is a major reason for which these attacks are unable to extract models when the watermarks are embedded throughout the network amongst little modifications of the weights. Particularly, watermarking using backdooring proves to be quite robust against these attacks. However, in a model modification attack mode, one can actually employ techniques to remove the watermarks from the models altogether. The extracted model would no longer have the embedded weights sensitive to the Trigger inputs and the verification process would therefore fail. The adversary would obtain a high performing model without having to pay royalties to the true owner who curated the data and trained the model. The attack proposed in this work is a model modification attack using synthesis and the details of it are discussed in the following sections.

7.3.4 Objective and Assumptions

The goal, as mentioned earlier, is to develop a mechanism wherein a watermarked neural model can be used by an adversary to successfully carry out the machine learning task (like classification), without having to deal with the consequences of the embedded watermarks that have been left behind by those who trained it in the first place. In the mechanism proposed in this work, our white-box attack intends to use Transfer Learning [109] in order to effectively re-use the knowledge that has

already been learnt by other involved parties in the business. For this attack, we will consider the following reasonable assumptions:

- This being a white-box attack, the adversary has access to the watermarked trained neural model (the computational graph and the weights). This is natural for customers who obtain services from MLaaS companies (suppliers) who let them use the model upon the belief that they would be able to establish ownership since it is watermarked.
- The adversary does not have access to, or any related information as well, about the Trigger Set and its corresponding labels. That private key rests with the supplier of the model.
- The adversary wants to successfully extract the model, whereby it would be able to use the model to achieve similar performance accuracies as the watermarked model, whilst doing poorly on the Trigger Set (thus de-establishing ownership by the supplier).
- The adversary and the supplier or owner of the model work on the same domain, aiming to fulfil the same machine learning task.

7.3.5 Methodology: Model Modification Attack with Synthesis

Considering the nomenclature discussed in Section 8.3, we have a watermarked neural network model $\hat{M}$, which has been created from a standard model M and a Trigger Set T and its corresponding labels T_L. The model M was trained on the data $train_{data} \subset D$ and the clean test set was $Test_{data} \subset D$. Typically, the watermarked model $\hat{m}$ would perform as well as M during inference, with high/full accuracy on the Trigger Set T. Since the adversary is acquainted with the domain D, it has its own dataset d, such that we have $d \subset D$. Now it is not natural to have an overlap between d, $train_{data}$ and $Test_{data}$, although if it were to happen, the attack would only be more easier and successful. Keeping the aforementioned assumptions in mind, since the adversary has the white-box access to the watermarked model, it is able to retrain it. The retraining is essential here to remove the watermarks embedded in the model, so that is successfully extracted to obtain a

model E and is usable without consequences of questionable ownership. The proof of removal of watermarks would lie in the performance of the extracted model on the Trigger Set; if the matches between the Trigger labels and the labels predicted by the extracted model E is low, then the adversary is prevailed. Now typically, retraining it would adversely affect the performance of the model on the actual machine learning task itself, which would render the extracted model E useless for further use. This specific issue is very well addressed by synthesis. The adversary can make use of the dataset d which it has at its disposal. to train a GAN that attempts to learn the distribution of the domain D, and is able to create many samples $\hat{d}$, which would be similar to actual samples that are in domain D. The quality of the samples generated by the GAN would impact the efficacy of the retraining.

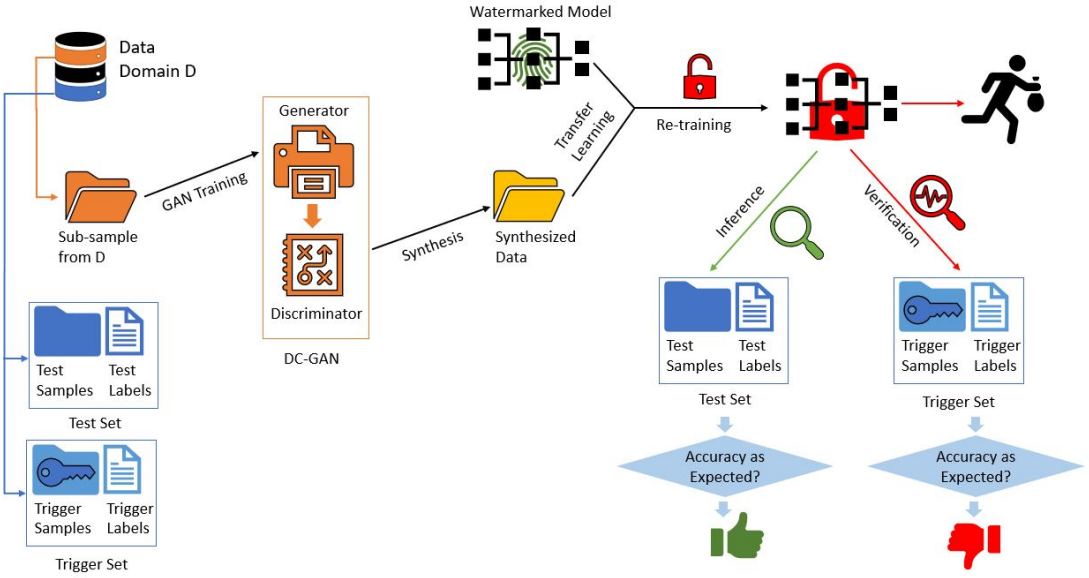


FIGURE 7.2: Schematic diagram of model modification attack using synthesis.

One can affirm that if an adversary were to re-use the stolen model for its own purpose, which is a task very similar to the one for which the owner/supplier had trained it, then the GANs can be modified slightly and tuned as per the requirements of the re-use. This way, the adversary may be able to re-purpose a stolen model for its specific demands.

It is very important to note here, that for a specific task and a domain of data necessary to accomplish it, the exercise of synthesis using the GANs is a one-time

process. Any adversary with some data d in its possession can train a GAN to generate the $\hat{d}$ beforehand, and may use it to retrain multiple/different watermarked models in order to steal them by removing the watermark. Also, the process of retraining just a few layers, as discussed hereafter, is comparatively way less time consuming and less demanding in terms of hardware resources, and therefore is a lucrative option for anyone with this sort of ill intention.

For converting the watermarked model into E, there are two approaches that we propose and study in this work. In both cases, the re-training by Transfer Learning is carried out using the same set of GAN-generated samples $\hat{d}$. Although this attack mechanism is agnostic to the specifics of the neural network architecture, we only consider a Convolutional Neural Network (CNN) where there is a series of convolution-pooling layers followed by a densely connected multi-layer perceptron. The approaches are addressed as follows:

- *Mode 1:* The final convolutional layer is rich in extracted features. Resetting this layer ensures the new model learns refreshed features, while carrying forward the pre-tuned weights of the dense layered classifier. In this mode of retraining, we rest the last convolutional layer and retrain the whole model thereafter with the synthesized samples. This approach reduces the cost of retraining significantly.
- *Mode 2:* In addition to resetting the final convolutional layer which is rich in features, This mode of attack is designed to also reset the entire classifier, which is built of the densely connected layers. The rest of the convolutional layers are however frozen and the model is then retrained with the synthesized samples. While this approach involves higher cost of retraining (proportional to the size of the dense classifier layers), this improves the accuracy of retraining.

7.4 Implementation and Results

To experimentally demonstrate the performance of the proposed model modification attack through synthesis, we opted for the state-of-the-art watermarking scheme using backdooring [3] as our target model $\hat{M}$. This is essentially a VGG-16 model [1] and a ResNet model [7] with watermarks embedded in it. The

Trigger Set T was constructed as in the original work for comparing the results. Naturally, when studying the attacks, we considered that the adversary has no access to the Trigger Set T and its labels T_L, and it is only used for validating the successful extraction of the model having removed the watermarks. We varied the datasets over multiple well documented ones: MNIST [5], Fashion-MNIST [102] and CIFAR-10 [6] in order to ensure that the work is easily reproducible. The GAN used for synthesis of samples is a DC-GAN [107]. We implemented the watermarking scheme and built the attack on it in PyTorch [81], on a 7th generation Intel core i7 processor with an additional NVIDIA GeForce GTX 1080 Ti GPU support.

7.4.1 Experimental Design

In order to get into the specifics of the experiments, let us first set out the different variational aspects that are considered in this analysis. In terms of models, we have the VGG-16 used for the datasets MNIST and Fashion MNIST and the ResNet CNN used for CIFAR-10. The choice of models in keeping in mind the state-of-the-art performances of the neural networks on these datasets, for the task of image classification. The data belonging to all the datasets is split into parts, for training, for inference and for synthesis using GANs. These clean models are watermarked using backdooring using the mechanism described in Section 8.3. In particular, we have two modes of embedding the backdoors, one is training the model from scratch (referred to as From Scratch in the observations) which is akin to data-poisoning and the other is using a pre-trained model (referred to as Pre-trained in the observations). For both of these kinds of watermarked models, we implement the proposed model modification attack through Synthesis, using the two modes described in Section 7.3: *Mode 1* and *Mode 2*. The retraining of the models using the samples collected from the GANs is carried out with as many samples as in the test set. In order to study the performance of the setup, we have two metrics, which are very typical of such performance analysis tasks. We study the accuracy of classification of the different models (clean, watermarked, extracted etc) on the actual Test Set and the specific Trigger Set. The expectations from a successful extraction mechanism of the watermarked model would be two-fold. Firstly, there is a massive drop of accuracy of the model on the Trigger Set upon classification. Secondly, the change in the inference accuracy on the clean samples (Test Set)

with respect to that of the watermarked model is not that significant, within an acceptable range.

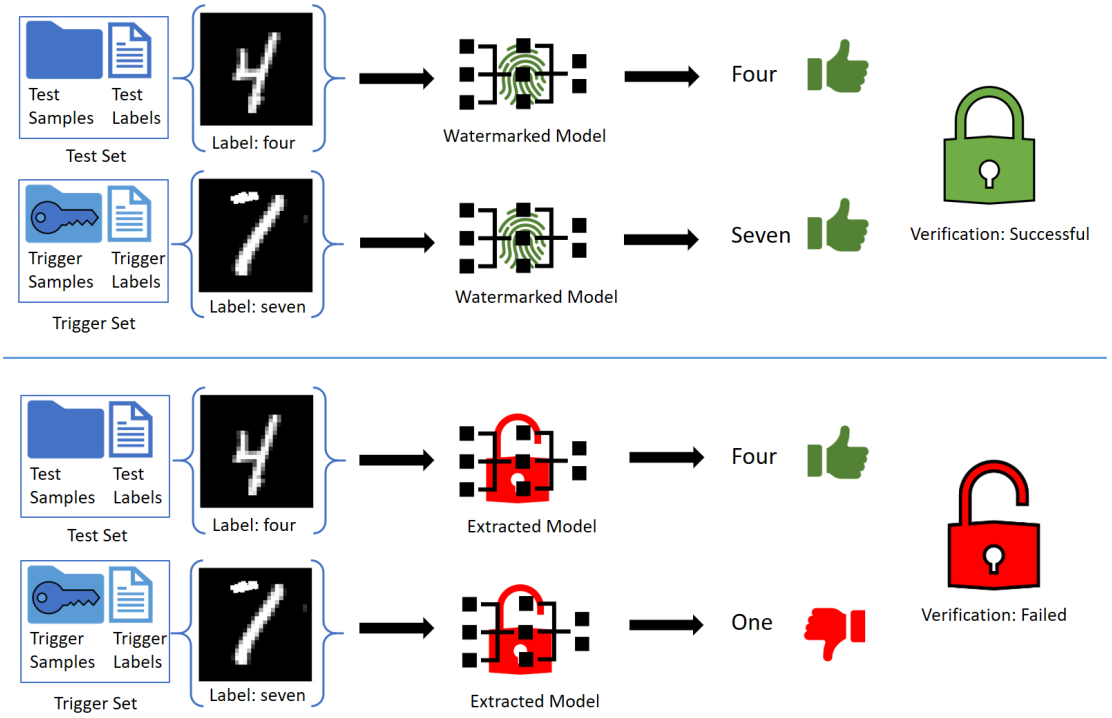


FIGURE 7.3: Demonstration of successfully extracting models and their behaviour as opposed to a watermarked model, considering one sample from the MNIST dataset.

7.4.2 Observations and Findings

The performances of the different versions of the VGG-16 model trained on the MNIST dataset is presented in the Table 7.1. We observe that in general, from a watermarking perspective, the From Scratch method works better in matching up to the performance of the clean model without watermarks, while maintaining a full accuracy on the Trigger Set to establish ownership during verification. Among the extracted models, the attack *Mode 2* works better and is within about 3% range of accuracy (for the From Scratch version) when it comes to the actual classification task, whilst bringing the accuracy on the Trigger Set to a extremely low value, thus proving the removal of watermarks. The criteria for a successful attack has been met.

The performances of the different versions of the VGG-16 model trained on the Fashion-MNIST dataset is presented in the Table 7.2. For the watermarked model,

TABLE 7.1: Extracting model from watermarked neural network (VGG-16) on the dataset: MNIST

Model	Accuracy (Test Set)		Accuracy(Trigger Set)	
	From Scratch	Pre-Trained	From Scratch	Pre-Trained
Clean Model	98.99%	-	10%	-
Watermarked Model	98.96%	97.59%	100%	100%
Extraction (*Mode 1*)	90.76%	84.15%	12%	14%
Extraction (*Mode 2*)	95.40%	81.82%	14%	13%

the difference between the two mechanisms From Scratch and Pre-trained in terms of performance is almost same, both being as good as a clean model. What is interesting is that, the mode used to extract the model and remove the watermarks is not very significant because they have almost identical performances, both within about 3% away from the accuracy of the watermarked model. The watermarks are successfully removed though, as evident from the Trigger set accuracies. The criteria for a successful attack has been met.

TABLE 7.2: Extracting model from watermarked neural network (VGG-16) on the dataset: Fashion-MNIST

Model	Accuracy (Test Set)		Accuracy(Trigger Set)	
	From Scratch	Pre-Trained	From Scratch	Pre-Trained
Clean Model	90.41%	-	12%	-
Watermarked Model	90.14%	90.95%	100%	94%
Extraction (*Mode 1*)	87.13%	87.31%	13%	11%
Extraction (*Mode 2*)	87.76%	87.97%	13%	11%

The performances of the different versions of the ResNet model trained on the CIFAR-10 dataset is presented in the Table 7.3. For the watermarked model, the difference between the two mechanisms From Scratch and Pre-trained in terms of performance is almost same, both being as good as a clean model. Among the extracted models, the attack *Mode 2* works better and is within about 3% range of accuracy (for both the From Scratch and Pre-trained versions) when it comes to the actual classification task, whilst bringing the accuracy on the Trigger Set to a extremely low value, thus proving the removal of watermarks. The criteria for a successful attack has been met.

As mentioned at the outset, the success of extracting the model from the watermarked version retaining accuracy of classification, depends highly on the quality of samples being synthesized by the Generative Adversarial Networks. The GANs

TABLE 7.3: Extracting model from watermarked neural network (ResNet) on the dataset: CIFAR-10

Model	Accuracy (Test Set)		Accuracy(Trigger Set)	
	From Scratch	Pre-Trained	From Scratch	Pre-Trained
Clean Model	88.49%	-	16%	-
Watermarked Model	88.39%	88.06%	100%	100%
Extraction (*Mode 1*)	81.80%	82.38%	19%	24%
Extraction (*Mode 2*)	84.54%	84.72%	60%	24%

need training on a subset of data that is sampled from the same domain as the original training data of the model. The number of passes over the sampled data, that the GANs need to churn, determined by the epochs of the GAN training is critical towards achieving the necessary quality of synthesis. We have also studied how the performance of the GANs saturate after a certain number of epochs, indicating that the synthesis is up to the mark. The performance in this context is measured as expected exactly like the earlier results, that is the accuracies on the Test sets and the Trigger sets. For each epoch, the reported result is the average of the performance obtained for multiple 'seeds', which is the element that introduces the randomness in the training of the GANs.

In Table 7.4, we present the study on the MNIST dataset, with our DC-GAN and the VGG network. The performance of the GAN saturates at around 40 epochs and thereafter the increment is rather small.

TABLE 7.4: Training the GANs with varying epochs (dataset: MNIST)

Model	GAN training	Accuracy (Test Set)		Accuracy(Trigger Set)	
		From Scratch	Pre-Trained	From Scratch	Pre-Trained
Extracted Model (*Mode 1*)	Epochs: 30	86.78%	68.91%	7%	8%
	Epochs: 35	88.28%	84.17%	6%	10%
	Epochs: 40	94.36%	91.28%	13%	12%
Extracted Model (*Mode 2*)	Epochs: 30	95.26%	86.25%	8%	9%
	Epochs: 35	95.25%	87.12%	8%	13%
	Epochs: 40	96.48%	93.72%	12%	10%

In Table 7.5, we present the study on the Fashion MNIST dataset, with our DC-GAN and the VGG network. The performance of the GAN saturates at around 45 epochs and thereafter the increment is not very significant.

As evident from the results presented here, the watermarks are successfully removed because the re-training with the synthesized samples from the GAN works

TABLE 7.5: Training the GANs with varying epochs (dataset: Fashion MNIST)

Model	GAN training	Accuracy (Test Set)		Accuracy(Trigger Set)	
		From Scratch	Pre-Trained	From Scratch	Pre-Trained
Extracted Model (*Mode 1*)	Epochs: 35	83.68%	83.82%	13%	12%
	Epochs: 40	83.38%	83.66%	12%	8%
	Epochs: 45	87.52%	87.16%	8%	12%
Extracted Model (*Mode 2*)	Epochs: 35	84.08%	83.87%	13%	11%
	Epochs: 40	84.26%	84.60%	8%	7%
	Epochs: 45	87.42%	87.30%	8%	12%

well. This is particularly due to its ability to tweak the specific weights distributed within the network, which would be responsible for being sensitive towards the Trigger inputs. Any potential counter-measure to this attack should therefore target to distribute the embedded watermarks well and evenly, so that model modification attacks involving re-training are not able to remove them while maintaining classification accuracy on the actual task.

7.5 Concluding Remarks

Acknowledging the necessity of robust watermarking schemes for properly verifiable ownership and establishment of IP rights of high-performing trained neural networks, a study of their vulnerabilities is crucial. Before deploying such models on public platforms or being offered to customers of companies who provide Machine Learning as a Service (MLaaS), the stakeholders must ascertain that adversaries would not be able to steal their proprietary material. Their faith in watermarking schemes is based on the assurance that these are hard to break, and the guarantees originate from cryptographic primitives.

There have been many such schemes which have been proved to be futile, as and when a new attack exposed their weaknesses. The state-of-the-art watermarking scheme by embedding backdoors in the weights of the neural network has gained attention due to its claimed robustness towards attacks. In this work, we proposed a model modification attack, which is able to successfully extract the model by eliminating such embedded backdoors. This approach is not based on the typical attempts to identify the backdoors and reverse-engineer the Trigger Set at all. We make the reasonable assumption that being an interested party in the business,

the adversary has access to some data in the same domain as that of the training data of the model, and uses synthesis to generate samples which can be used to retrain the model. The objective would therefore be to ensure that the extracted model has an accuracy comparable to that of the watermarked model on the test data, given the machine learning task. At the same time, as proof of elimination of the backdoors, and thereby the watermarks, the extracted model should perform poorly on the Trigger Set.

We have demonstrated this mechanism works well, on multiple image classification tasks on highly benchmarked datasets including MNIST, Fashion MNIST and CIFAR-10. We have been able to achieve classification accuracies nearly about 3% to what the watermarked model could get to, whilst steeply dropping the performance on Trigger Set, thus successfully failing the verification process. This way of stealing the model could severely jeopardise the interests of the stakeholders investing in the development of these machine learning models. It is worth noting that although we demonstrated the attack on the state-of-the-art watermarking mechanism, our approach of breaking watermarking is not limited to just this particular method of marking through backdooring and works for other mechanisms as well. This work puts into perspective the serious requirement of reconsidering the way we watermark models and develop more robust mechanisms in order to safeguard IP rights of the trained neural networks and defend the investment of the associated stakeholders better.

Chapter 8

Watermarking Neural Networks: ROWBACK and TextBack

8.1 Introduction

[1] For solving any task using machine learning, the corresponding stakeholders have to invest in:

- gathering the relevant data, curating it and labelling it particularly for supervised machine learning problems
- designing the most suited neural architecture for the task
- procuring highly capable hardware infrastructure for training the neural network with the data to obtain the learned model

It is natural for those investing in any one or multiple of the components mentioned above, to demand a method to establish ownership of the trained neural network. Claiming authority by proving Intellectual Property rights on such models is a necessity to incentivise the stakeholders. As a result, there has been a steady

[1] Chapter 8 is published in part as *Nandish Chattopadhyay, Anupam Chattopadhyay, "ROWBACK: Robust Watermarking of Neural Networks using Backdooring," in Proceedings of IEEE ICMLA 2021* and *Nandish Chattopadhyay, Rajan Kataria and Anupam Chattopadhyay, "TextBack: Watermarking Text Classifiers using Backdooring," in Proceedings of EuroMicro DSD 2022.*

growth of research into various watermarking schemes for neural architectures [49–51, 54] as well as efforts to break those watermarking models [4, 52, 60–62] by adversaries, which are discussed in details in the following section 8.2. In this line of research, we contribute by proposing a novel watermarking scheme that shows robustness against all relevant attacks without any perceptible degradation in model efficiency.

8.1.1 Contribution

In this work, we propose a robust watermarking scheme ROWBACK for claiming ownership of trained neural networks based on backdooring. We study the existing literature in this domain, and identify the key reasons that contribute towards the vulnerability of such watermarking techniques through extraction attacks that try to steal the neural networks. In essence, we make use of two properties of neural networks, the existence of adversarial examples upon introducing structured perturbation and the ability to embed backdoors while training, to our advantage in the attempt of establishing ownership. ROWBACK guarantees robustness through the following:

- Re-designed Trigger set for marking the neural networks. We make use of adversarial examples of the models and associate Trigger labels to them, to customize the Trigger set that ensures preservation of functionality, leaves strong embedded watermarks whilst being extremely difficult (since there exist infinitely many adversarial examples) to be replicated by adversaries.
- Uniform distribution of the embedded watermarks throughout the model, by explicitly marking every layer with imprints of the backdoors, so as to prevent model modification attacks by making such attempts to steal the network computationally equivalent to training a fresh network from scratch.

The ROWBACK framework has been tested for benchmarking with standard datasets like MNIST [5] and CIFAR-10 [6] with extraction attacks [4].

Extending this work, we propose a watermarking scheme called TextBack specifically for text classifiers using backdooring. The primary contributions are inclusive of, but not limited to the following:

- Demonstrating the capabilities of TextBack for providing verifiable proof of ownership using watermarking, while being functionality preserving in nature, for multiple neural architectures and datasets
- Testing the success of TextBack for two different types of Trigger Sets (as chosen by user) in being an effective watermarking scheme
- Studying the efficiency of embedding watermarks within trained neural networks and checking for the corresponding computational costs

TextBack has been tested for multiple neural architectures, datasets and Trigger sets, as shown in Section 8.7.

8.2 Watermarking and Backdooring

Watermarking has been studied for long in different research communities, and has seen great progress in the image processing domain [110]. Being an effective way to prove ownership, the idea has been leveraged into other domains as well, most notably in the machine learning community. Backdoors have been found to be a possible way of trapping watermarks within large trained neural networks.

8.2.1 Principles

There are three essential components of any functional watermarking scheme. Under the basic assumption that there is a training dataset and $training\_data$ and a trained neural network model M, we need three things to work. First, one needs to devise a way to create a secret key m_k for watermarking, which will be used to embed the watermarks, and the associated verification key v_k will be used for verifying the presence of the watermarks, thereby establishing ownership. Second, there is a need to have an algorithm to embed the watermarks within the asset, which is the neural network model in this case. Third, one needs an algorithm utilizing both the secret key m_k and public key v_k for verification. These algorithms can be formally expressed as:

- $Key\_Generation()$: provides the pair of marking and corresponding verification keys (m_k, v_k)
- $Watermarking(M, m_k)$: inputs as parameter a trained model and a secret watermarking key m_k, returns a watermarked model $\hat{M}$
- $Verification(m_k, v_k, M)$: inputs as parameters the marking and verification key pair (m_k, v_k) and the watermarked model $\hat{M}$, returns the output bit $b \in \{0, 1\}$

The functioning of the watermarking scheme is strongly dependant on the correct working of all of the three algorithms $(Key\_Generation, Watermarking, Verification)$ together. In this particular context, correctness can be formally described as:

$$Pr_{(M,\hat{M},m_k,v_k)\leftarrow WM()}[Verification(m_k, v_k, \hat{M}) = 1] = 1 \tag{8.1}$$

where, the watermarking algorithm $WM()$ is comprised of the following:

1. Creation of $M \leftarrow Train(training\_data)$
2. Taking samples of $(m_k, v_k) \leftarrow Key\_Generation()$
3. Computation of $\hat{M} \leftarrow Watermarking(M, m_k)$
4. Returning of $(M, \hat{M}, m_k, v_k)$

8.2.2 Backdoors as Watermarks

A backdooring algorithm works by taking in an input neural network model, and provides a model with backdoors trapped in it. The backdoors ensure that the model behaves in a specific way upon specific input samples being given. Essentially, the embedded backdoors make sure that the accuracy of the model is high on the Trigger set. It may be noted here that a clean model would perform poorly on the Trigger set as the Trigger labels are not the naturally occurring labels of the Trigger samples, and are set explicitly to induce such behaviour. Backdoors, therefore, are a good choice to serve as embedded watermarks, as demonstrated in the literature [3].

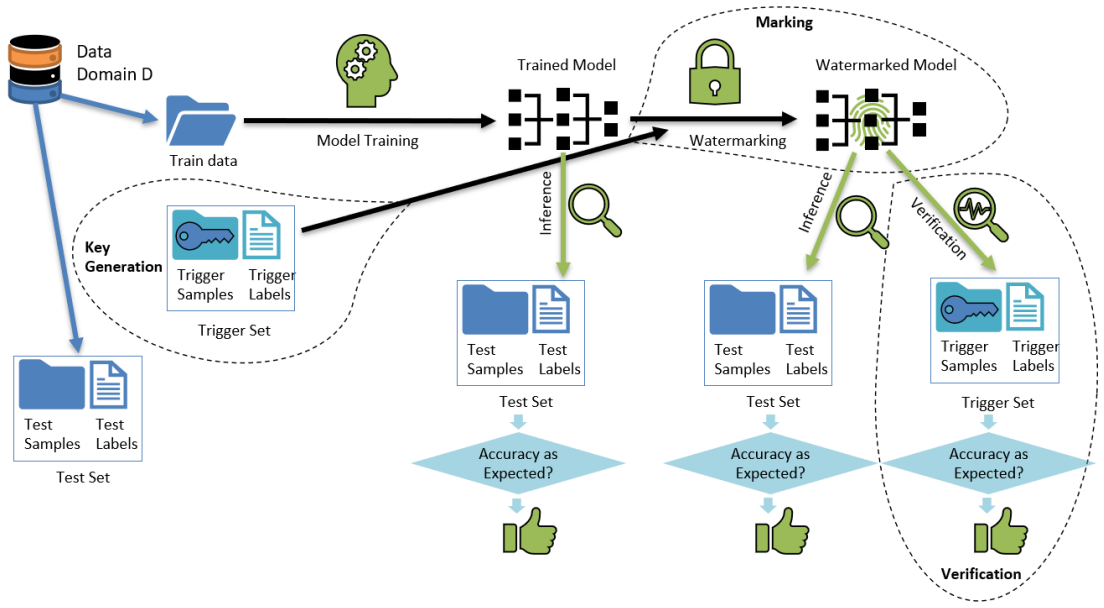


FIGURE 8.1: Schematic diagram of watermarking using backdooring [3]

8.2.3 Failure of existing mechanism

Reliable use of any watermarking scheme is dependant on a study of strengths and weaknesses of it. One needs to pay attention to vulnerabilities of the mechanisms, to understand the potential flaws and correct them. As mentioned earlier, the primary aspects of importance for any such watermarking scheme include:

- Watermark embedding within the trained model (backdoors for example)
- Secrecy of the key (which is the Trigger set)
- Reliability of the verification mechanism for claiming ownership

An attacker with malicious intent, works to negate one or more of the aforementioned requirements. It may be noted that the failure of any one or more of the above may jeopardise the entire watermarking scheme. In a situation where the adversary is interested in stealing the trained neural network, thereby successfully denying the owner his/her claim of ownership, the following must hold good:

- The model that has been extracted must generate comparable accuracy for the specific machine learning task as the watermarked model

- The verification process must fail, the extracted model should perform poorly on the Trigger set

If an attacker is able to accomplish the fulfilment of these criteria, then the stakeholders invested in curating the model potentially lose their right to claim IP. In the specific context of watermarking through backdooring, attacks based on targetted re-training of the model using synthesized samples have proved to be a legitimate threat [4]. These attacks expose the underlying vulnerability of the watermarking scheme.

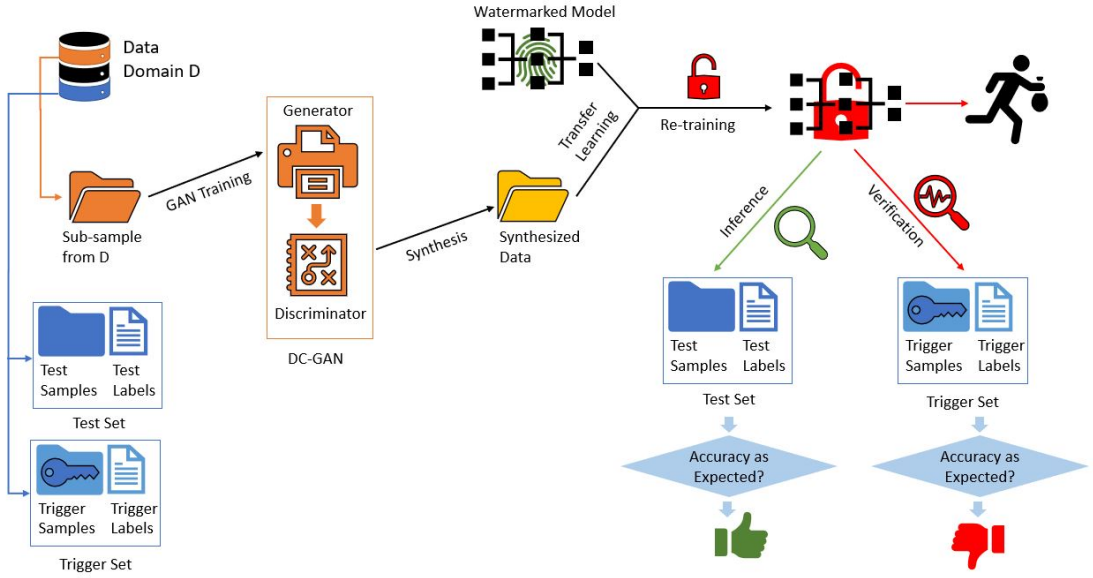


FIGURE 8.2: Model Modification attack using synthesis [4].

These attacks put together provide a challenge for reliability in watermarking schemes. We wish to address this problem, by learning from the vulnerabilities of the existing literature and utilizing some of the available techniques, coupled with generic properties of neural networks, and thereby propose ROWBACK.

8.3 ROWBACK: Guarantees of Robustness

The primary goal of this work is to propose a watermarking scheme, which borrows the fundamental idea behind the principle of watermarking neural networks using backdooring, but addresses the deficiencies that it faces by adding counter-measures

for the attacks that have been carried out on it. The resulting watermarking scheme is therefore efficient as well as robust against prominent attacks available in the literature. Robustness is achieved addressing to key areas, the way the Trigger samples are designed, and the method in which the watermarks are embedded. Before going into the details of the mechanism that of ROWBACK, we would like to mention that this watermarking scheme is built on two key pillars, each of which were originally discovered as defects of neural networks. Their backgrounds are briefly touched upon here.

- Adversarial examples: Researchers found in 2015 that neural networks have a particular vulnerability in adversarial attacks [28]. High performing models would be fooled by adversarial examples [29–31]. The adversarial samples are created by introducing minute structured perturbation to the clean test samples, which would be unobservable to the human eye.
- Backdoors: They were originally observed as flaws in trained neural networks, where backdooring would be considered a specific technique to train the model in such a way that it predicts erroneous outputs for a particular set of inputs.

8.3.1 Trigger Set Design

The vulnerabilities mentioned in the aforementioned attacks stem significantly from the fact that the often the choice of Trigger samples make the model modification attack easier. This is particularly true when the Trigger samples are Out-Of-Distribution (OOD), which was proposed in the original work demonstrating watermarking using backdooring [3].

Adversarial Examples as Trigger Samples

The notion of utilizing adversarial examples as Trigger samples stems from the fact that adversarial examples are samples that are perturbed train/test samples belonging to the distribution of the training data, therefore in essence quite close to the overall distribution that the model has seen during training.

Labelling Trigger Samples

The mechanism of associating the labels to the trigger samples is critical towards the integrity of the water marking scheme. It is important to note that the "true" class labels of the samples, or their primary adversarial samples can not be used as the labels of the Trigger samples, to ensure that the non-trivial ownership property holds. The adversary, keen on stealing the trained neural network, should not be able to regenerate the Trigger Set, which consists of the Trigger samples and their corresponding labels. This is why, we have used the adversarial samples as the Trigger samples and associated a class label to each, which is not its "true" class label or its primary adversarial class label. Reverse-engineering this kind of a Trigger set would therefore involve considering all adversarial samples to a model (which is infinite in number) and mapping them to all but two classes of the dataset.

8.3.2 Watermark Distribution

The principle of using backdooring to embed watermarks in the trained neural network has a drawback that has been demonstrated in multiple attacks available in the literature. It is typically observed that the backdoors, in the form of weights within the trained weights matrices of the network, are generally present in the densely connected layers. This facilitates the vulnerability of these networks, which is that by being partially re-trained in the densely connected layers, the watermarks are removed. We take note of this flaw in design and address it by forcefully ensuring uniform distribution of the embedded watermarks in every layer of the network. We make sure that while the marking process is carried out, each of the layers are individually marked, which means a model modification attack aimed at removing the watermarks, will need to retrain the entire model, which is equivalent to train a fresh model from scratch, in time and effort.

8.3.3 Guarantees

In order for us to claim the robustness of the watermarking scheme, we need to establish that the proposed mechanism satisfies the necessary criteria for being effective, as mentioned in the earlier section. Here, we study how each of the

properties are satisfied by construction of the watermarking scheme. The following claims have also been verified through experimental results in Section 9.6.

Functionality Preserving

The functionality preserving property requires that a watermarked model should be as accurate as a model that has not been watermarked. Naturally, different models have different metrics of measurement of performances, but for the machine learning task that we consider here, the metric of choice is test accuracy of the model on the test set. Since adversarial examples are a naturally occurring phenomenon of neural networks in general, using them as specific Trigger samples does not hinder the overall performance of the model. In fact, the approach used here is much like adversarial training, which is used to create robust models. The functionality preserving claim has been substantiated through experimental results.

Non-Trivial Ownership

The property of Non-Trivial Ownership requires that an attacker who has the knowledge about the watermarking algorithm will still not be able to claim ownership of the model. It has to be noted that the process of claiming ownership of the model involves the demonstration of the accuracy of the model on the Trigger set, which is available only to the creator of the watermarked model. Therefore, the non-trivial ownership aspect is taken care of in design, in the construction of the Trigger set. The Trigger set consists of randomly selected adversarial samples generated by introducing structured perturbation to clean train samples. Since there are infinitely many such samples producible, it is impossible to reverse engineer the exact set without any other knowledge. The random sampling ensures that there is a lack of correlation among the samples and that takes care of the scenario where accidental revealing of a part of the Trigger set will also not hamper the ownership verification process.

Un-removability

The un-removability property requires that the attacker who has the knowledge of the watermarking algorithm and also has the watermarked model as hand, will

not be able to detect and remove the embedded watermarks. In essence, this property necessitates the requirement of the watermarking scheme to be robust against model-modification attacks. In our proposition, we take care of this property by paying particular attention to the distribution of the embedded watermarks. The watermarking scheme describes here ensures that the embedded watermarks are present in each layer, and every layer will have to be re-trained to get rid of them completely. The intuition here is that, should the attacker require as much effort and resources (time and training samples) to remove the watermarks as it is needed to train the model from scratch, then in theory we will have satisfied the un-removability property.

Un-forgeability

The un-forgeability property requires that partial information about the Trigger set (which in this case consists of the Marking and Verification keys, in Trigger samples and labels respectively) will not provide the attacker any advantage in establishing ownership on the watermarked model.

8.4 ROWBACK Implementation

The implementational details are discussed in this section, covering each component of the watermarking pipeline. In broad terms, the key components are the model itself, the training and test datasets and a specific Trigger dataset for watermarking.

8.4.1 The ML task and model

In this particular work, we consider a computer vision task of image classification. This is particularly relevant as it extends from the use-cases available in the literature and most widely in use. The overall framework however is absolutely task agnostic and can be used for any machine learning model with underlying neural networks, for example transformer-based neural networks for neural machine translation. For our purpose of claiming ownership, we use a ResNet [7] model M and demonstrate the use of our proposed framework on it. The model M is

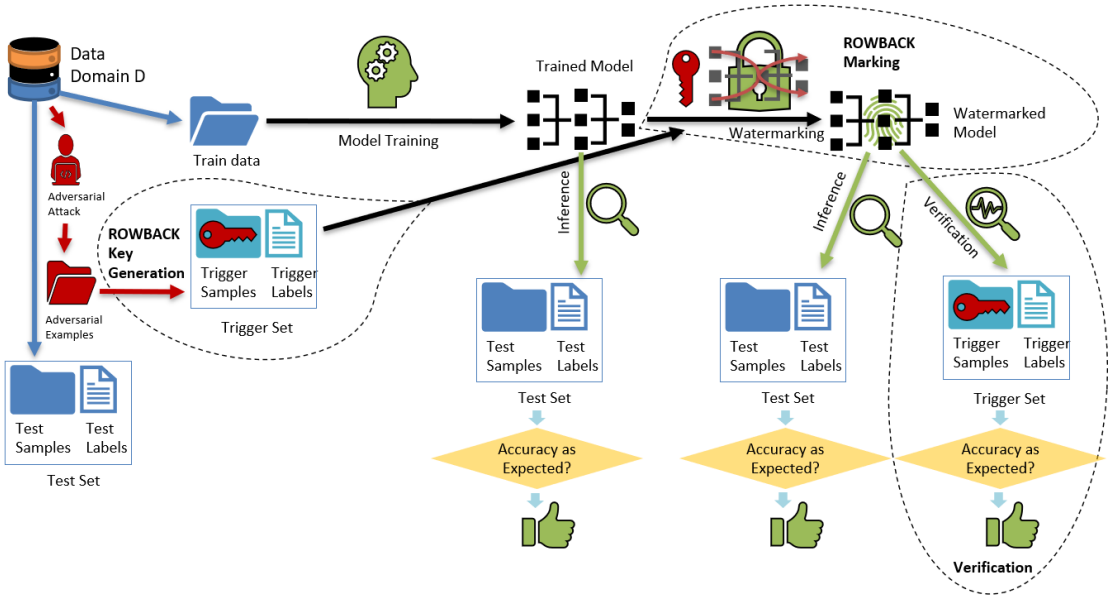


FIGURE 8.3: Schematic diagram of ROWBACK.

trained on training samples S_{train} and is tested thereafter on a test sample S_{test}. The performance of the model M on the test set S_{test} is denoted as S_{acc}. The performance of the model M on the Trigger set T is denoted as T_{acc}. The process of embedding watermarks is described hereafter.

8.4.2 Key Generation

As mentioned in Section 8.3, the generation of the Trigger set T, which is the key to this watermarking scheme, involves adversarial examples. These samples are created by first considering a random stratified sample of t_s train images with equal representation of each class. Then, each of these samples are subjected to structured perturbation which is introduced by an adversarial attack. In this work, we have made use of the Fast Sign Gradient Method. The Fast Sign Gradient method was first introduced by Goodfellow et al. [2] in 2014.

Once the t_s many adversarial samples are generated, we obtain one part of the Trigger Set T_S. We take note of the original class labels to which each sample belonged, as well as its new class label, as detected by the classifier upon being converted into an adversarial example.

The Trigger labels T_L associated with each of the Trigger samples is chosen randomly, from the rest of the class labels excepting the two class labels, the naturally

occurring true label and the adversarial label. The reasoning behind this construction has been explained earlier, in the discussion about satisfying criteria for being effective watermarking schemes. The quasi-random choice of class labels for each of the Trigger samples ensures exclusivity of the Trigger Set, and makes its replication very difficult.

8.4.3 Marking

In order to mark the neural network with the watermarks, we make use of the Trigger Set T and a pre-trained model M. The neural network in consists of multiple layers, in this case convolutional layers and fully connected layers. For the process of embedding backdoor watermarks, we make use of Transfer Learning [109] and fine tune the parameters. The fine-tuning works in the following way:

1. In every epoch ep_k, where $k \in \{1, \ldots, n\}$, we freeze all but one layer in the network (starting from the fully connected layers and ending with the convolutional layers), and fine-tune that layer with the Trigger set T by updating the parameters therein.

2. For epoch ep_k, where $k \in \{1, \ldots, n\}$, we note the corresponding accuracies of the model M, for the test set S_{test} which is S_{acc} and the Trigger set T which is T_{acc}.

3. We repeat the combination of the steps 1 and 2, for n times, where n is a hyper-parameter that is determined by cross-validation. The cross-validation is carried out by observing the S_{acc} and T_{acc}, and an intuitive thumb rule is to stop the epochs when either or both of the following occur:

   - The T_{acc}, which is the performance of the model M on the Trigger set T, starts to saturate after increasing with each of the earlier epochs
   - The S_{acc}, which is the performance of the model M on the test set S_{test}, begins to drop significantly

4. After n epochs, we make a note of the Trigger accuracy T_{acc}. This is of critical importance towards the verification of watermarks, which is the key to claiming ownership of the network.

The aforementioned process of marking the model with backdoors makes the model ready for deployment in the public space, as the stakeholder is guaranteed of having the provisions of proving ownership of it, should the requirement arise.

8.4.4 Verification

The explicit criterion for verification of any watermarking scheme is expressed in Equation 8.1 which states that the probability of a Verification Function taking as parameters the Marking Key and the Verification Key and returning always True is unity. In the context of watermarking neural networks, this translates to the following:

- The Verification function has two parts. One part is the model M itself, and it returns the probabilities of the classes of the classifier, the highest probability being allotted to the class to which the sample in question most likely belongs. The second part matches these outputs to the expected labels.
- The Marking Key is the set of Trigger sample T_S
- The Verification Key is the set of Trigger labels T_L
- The Verification Function, takes the Marking key (Trigger samples T_S) and generates the predictions first. Then it compares them to the Verification key (Trigger Labels T_L) and generates a score.
- In theory, as per Equation 8.1, this score should be 100%. In practice, we allow a tolerance limit in our framework, which is determined by the T_{acc} obtained after n epochs of marking.

It may be noted here that in the following Section on experimental results, we are able to achieve a full score of 100% match for verification. The tolerance limit is still part of the framework for making the approach more generic.

8.5 Experimental Observations

Every claim regarding performance of ROWBACK and its robustness against standard attacks have been validated through thorough experimentation. In this section, we discuss in details the overall experimental setup and illustrate the functionalities of the scheme through results. As mentioned earlier, we have demonstrated ROWBACK's performance on image classification tasks for relevance with the existing literature. The entire pipeline of experiments from building the model to watermarking it and verifying watermarks and testing for robustness has been implemented in PyTorch [81] on a 7th generation Intel core i7 processor with an additional NVIDIA GeForce GTX 1080 Ti GPU support.

8.5.1 Experiment Design

The neural network model of choice is the ResNet [7] architecture, which is a convolutional neural network with eighteen layers that makes use of deep residual learning frameworks. The datasets for the image classification tasks are MNIST [5] and CIFAR-10 [6]. The Trigger Set, T comprising of T_S trigger samples and T_L trigger labels, is generated through FSGM-based adversarial attacks on the test images and a quasi-random allocation of classes to them. The watermarks are embedded through targeted fine-tuning using Transfer Learning as described earlier. The model is finally tested for robustness against Evasion attacks and Model modification attacks.

8.5.2 Results

We have looked at three key target areas to investigate. The datasets CIFAR-10 and MNIST were split into 90%-10% for the train-test split, the size of the Trigger set T was 100 samples, created using the FSGM attack with $\epsilon = 0.04$. The ResNet model has been trained for 80 epochs, which a learning rate α being reduced in half every 20 epochs. Since our proposed scheme ROWBACK involves watermarking through embedding backdoors, we have compared the performances of ROWBACK with the relevant existing scheme which also uses backdooring [3].

Test of Preserving Functionality

The goal of these experiments is to check whether ROWBACK is able to embed watermarks through backdoors without creating any hindrance to the overall machine learning task, which in this case is image classification. This is necessary for any functional watermarking mechanism to be deployed in practice.

The experimental setup presents a study of how ROWBACK behaves, with respect to the performance on the Test set, measured by Test accuracy. We compare the performances of a clean model without any watermarking, a standard watermarking model with backdooring [3] and ROWBACK.

Based on the observations tabulated in Table 8.1, in the Test Set Accuracy columns, we can conclude that there is no significant degradation of performance for introducing the embedded watermarks in a robust fashion, and the generated model is almost as accurate (accuracy differs by $1 - 2\%$) as either of a clean model without watermarks or a standard watermarked model through backdooring that uses Out-Of-Distribution samples in its Trigger set.

Test of Watermarking Verification

The goal of these experiments is to test whether the presence of the watermarks can be verified at will, which is in fact the proof of ownership that the stakeholders may use to claim their investments and prevent theft or unauthorised usage.

The experimental setup studies how ROWBACK works on the Trigger set T. In case of the clean model, there is no existence of a pre-defined Trigger set, and we have used our own Trigger set comprising of adversarial samples to check the performance. For the standard watermarking model, the Trigger Set is constructed with Out-Of-Distribution abstract images [3]. The performances of the standard watermarking scheme and the one proposed in this work are compared.

The observed results are tabulated in Table 8.1, in the Trigger Set Accuracy columns. Based on the observations, we can assert that the verification can be used to claim ownership, as a model which is not watermarked, would generate poor scores on the Trigger set, as opposed to a watermarked model. Since the size of the Trigger sets in these experiments is 100 samples, the accuracy noted in Table

TABLE 8.1: Checking preservation of functionality through performance on Test Set and checking verification through performance on Trigger Set.

Model Description	Test Set Accuracy		Trigger Set Accuracy	
	MNIST	CIFAR-10	MNIST	CIFAR-10
Clean Model w/o Watermarking	98.6%	92.1%	4%	4%
Watermarking w/ Backdooring	97.7%	91.8%	98%	96%
ROWBACK scheme	97.9%	91.2%	100%	100%

8.1 is the count of matches that the model is able to predict. It may be noted that the accuracy of the clean model on the Trigger set is just the adversarial accuracy of the model.

Test of Robustness

The goal of these experiments is to study the robustness of ROWBACK against attacks. We look at all kinds of attacks, as discussed earlier, Evasion attacks and Model Modification attacks.

ROWBACK is robust against Evasion attacks like ensemble attacks [58] by design. The Trigger set is comprised of adversarial samples. The watermarked model will predict the Trigger labels T_L, for the Trigger samples T_S. The models that are not watermarked, will predict the adversarial labels for the Trigger samples, which is still not the true labels, and therefore the premise of Ensemble attacks do not hold good. Since the watermarks embedded within the weight matrices of the network are not disturbed anyway, the robustness comes from construction.

Model modification attacks are the most pertinent threat to these watermarking schemes as they are able to remove watermarks thereby eliminating any traces of proof of ownership. In particular, we look at the removal of watermarks through synthesis [4], which specifically attacks the mechanism of watermarking using backdooring. The attack emulates training samples using GAN-based synthesis and uses them to re-train targetted parts of the model. As it turns out, re-training just the feature-rich layer (Mode 1 of attack) or the entire set of densely connected layers (Mode 2 of attack) is sufficient to remove the traces of watermarks.

The experimental setup is designed as follows: We first observe the impact of the ways/modes of attack on the standard watermarking scheme (shown in Table 8.2). Then we compare those results with the same analysis of ROWBACK (shown in Table 8.3). Finally, we extend the attack to see how much re-training us necessary for completely removing the watermarks from ROWBACK's model.

TABLE 8.2: Checking weakness of standard watermarking scheme [3] using Model Extraction attack [4].

Model Description	Test Set Accuracy		Trigger Set Accuracy	
	MNIST	CIFAR-10	MNIST	CIFAR-10
Clean Model w/o Watermarking	98.6%	92.1%	4%	4%
Watermarking w/ Backdooring [3]	97.7%	91.8%	98%	96%
Extracted Model [4] (Mode 1)	93.4%	88.5%	15%	18%
Extracted Model [4] (Mode 2)	96.2%	89.2%	11%	26%

The results are as expected, in agreement with the effectiveness of the extraction attack [4] and set the basis for the requirement of a more robust mechanism. As observable, the Trigger accuracy drops significantly upon the targetted re-training, which is a key weakness of this model.

We therefore, repeat the same experiments on ROWBACK to check what impact the modes of model extraction has, on its Trigger accuracy.

TABLE 8.3: Checking Robustness for ROWBACK scheme using Model Extraction attack [4].

Model Description	Test Set Accuracy		Trigger Set Accuracy	
	MNIST	CIFAR-10	MNIST	CIFAR-10
Clean Model w/o Watermarking	98.6%	92.1%	4%	4%
ROWBACK scheme	97.9%	91.2%	100%	100%
Extracted Model [4] (Mode 1)	94.1%	87.2%	99%	96%
Extracted Model [4] (Mode 2)	95.3%	87.8%	95%	94%

ROWBACK ensures explicit fine-tuning of each layer, with the hope that the traces of embedded watermarks will be well distributed. We have indication of that from

the results in Table 8.3, where the two ways/modes of model extraction are unable to bring down the Trigger accuracy significantly, at worst by $5-6\%$. We can set the tolerance limit for verification to accommodate the same and ensure reliable usage.

As a natural follow-up analysis, we studied how much re-training is required, to eliminate the watermarks significantly, below 50% accuracy on the Trigger set, that is, whilst maintaining the the performance on the test set within the $2-3\%$ range of the clean model. In this experiment, we progressively re-trained all the layers for as many as 80 epochs, which is the number of epochs on which the original ResNet model was trained.

We have observed that it takes about 35 epochs of re-training with a sample size of 60% of actual training samples, to obtain an extracted model which is functional and without watermarks. This effort, of using as many training samples and running it through all layers for about 45% of iterations is similar to training a new model from scratch.

8.6 TextBack Implementation

In the available literature for watermarking neural networks, for most of the proposed algorithms, the use-case of choice has been image classification tasks. This is natural, as seen throughout the evolution of the deep learning paradigm, where the initial breakthroughs were always in the area of computer vision applications. Once established as a working solution for proof of ownership, watermarking can be extended to natural language processing applications as well. There exists a massive demand for tools that help in understanding natural language, in a variety of daily applications including ones that make use of automated voice controls. Therefore, stakeholders investing in such models, require assurances on ownership of such trained neural networks. Since there is no prior technology demonstration on embedding watermarks within text classifiers, this research gap served as a primary motivation for this work.

However, applying tools and algorithms that work well on images, on text datasets is not straight forward. This has been a key deterrent to what would have seemed like a natural progression in research. Natural language is not compositional, and

the sequence-based relationships contribute to the fact that tasks like embedding watermarks within such models require different solution approaches. TextBack has been created to address these problems and provide reliable verification of watermarks to prove ownership.

In this Section, we discuss the exact design choices and their corresponding explanations for the TextBack implementation. Primarily, any watermarking scheme that involves backdooring involves two key procedures, the generation of the Trigger set and the algorithm to embed the backdoors as watermarks within the trained neural network using that Trigger set. The details of these are presented here.

8.6.1 Trigger Set Design

The choice of the samples to be included in the Trigger Set is critical towards the success of the watermarking process because the Trigger Set serves as the private key which resides with the owner of the asset. Trigger samples are curated samples which are assigned labels which are not their naturally occurring labels. For a specific watermarking application, the Trigger Set should be unique and very difficult to replicate. TextBack has two different ways of constructing the Trigger set.

Internal

The choice of selecting the Trigger samples could be made from within the set of training samples or outside. In this mode, we select at random as many samples as the size of the Trigger set from the trainset and assign them random labels. This means that during the Trigger Set generation, any of the labels apart from the one specified in the trainset could be assigned to the Trigger samples [Figure 8.4].

External

The Trigger samples could also be out-of-distribution samples, randomly taken from any source. For every such sample, a corresponding label is attached, which again is selected randomly from the choices of labels available in the training set. This method of generating Trigger set makes the task of replication of Trigger

samples extremely difficult and therefore the watermarking scheme more robust [Figure 8.5].

It may be noted here that TextBack works with both kind of Trigger Sets and the same is demonstrated experimentally as well. The choice is therefore left to the specific use-case and the user.

8.6.2 Embedding Watermarks

The watermarks need to be embedded into the trained high performance neural network during training, so that they may be used for verification to establish proof of ownership. There are multiple ways of embedding watermarks. This could be done during training itself, using data poisoning. TextBack uses a lightweight method involving lesser computations. Essentially, the trained neural network is fine tuned using the Trigger samples and therefore the parameters are updated with the embedded watermarks.

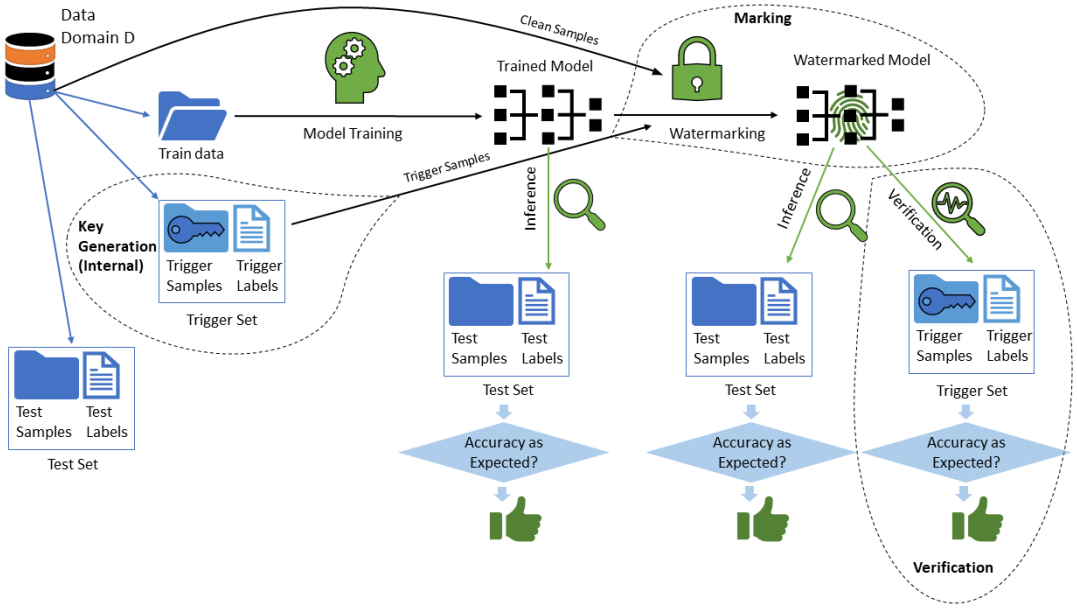


FIGURE 8.4: Schematic representation of TextBack using an 'Internal' type Trigger Set.

A necessary requirement of embedding watermarks within the trained neural networks is that it must be functionality preserving. The model, with and without the embedded watermarks should be similarly accurate in terms of performance on a test dataset. To ensure the functionality preserving aspect, the fine tuning of the

neural networks for the text classifier models have been done using both Trigger samples and actual clean samples. TextBack uses a mechanism where the Trigger samples are mixed with regular clean samples and the combination of them is used as part of the Transfer Learning process to modify the network's parameters. It must be noted that the Trigger set obviously does not change and the clean samples are just to maintain the original performance of the text classifier. This empirical property of watermarking neural networks, applies to only text modelling models, based on extensive experimentation.

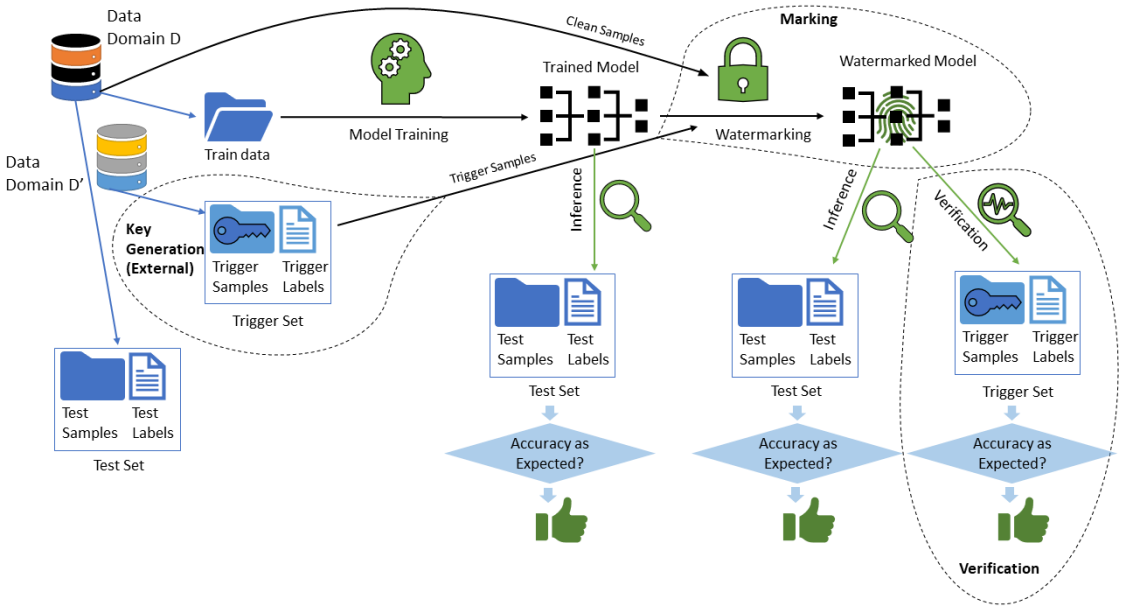


FIGURE 8.5: Schematic representation of TextBack scheme using an 'External' type Trigger Set.

8.6.3 Algorithm

The detailed algorithm for TextBack is presented here:

$Generate\_Keys()$:

- Execution of $(T, T_L) = b \leftarrow BackdoorSampling(train_{dataset})$ in which $T = \{t^{(1)}, \ldots, t^{(p)}\}$ and $T_L = \{T_L^{(1)}, \ldots, T_L^{(p)}\}$

$Mark\_Model(M, m_k)$:

- Update the parameters to output $\hat{M} \leftarrow Backdoor(train_{dataset}, b, M)$

$Verify\_Model(m_k, v_k, M)$:

- For all $i \in [p]$, test that $Classifying(t^{(i)}, M) = T_L^{(i)}$.
  If this holds good for all but $\epsilon|T|$ elements of T, the output 1 or else return 0.

It may be noted here that the tolerance on verification $\epsilon|T|$ is a user choice. For TextBack, we have set it to 1%.

8.7 Experiments and Observations

The experiments to study and verify the functionalities of TextBack are described in details in this Section. We touch upon the setup used for experiments, including the models and datasets etc., and follow that up with details on experimental findings.

8.7.1 Experimental Setup

TextBack is designed for watermarking neural networks. We choose the neural architectures by comparing them with some baseline models from classical natural language processing. The datasets we have used are the IMDB movie reviews dataset and the Twitter dataset. The neural architectures used were a vanilla RNN and an LSTM based neural network.

The IMDB dataset used in experiments is the in-built dataset taken from torchtext.datasets package in Pytorch. The dataset for Twitter samples is taken from Kaggle. The links to both the datasets are given below:

- IMDB dataset: Link
- Twitter dataset: Link

All the experiments have been undertaken in Google Colab (Colaboratory) environment with Intel(R) Xeon(R) 2 core CPU @ 2.30GHz and Nvidia K80 2 core GPU with 12 GB available RAM and 33 GB of empty disk space.

The main libraries used in the implementation of all experiments *nltk* for removal of stopwords, *re* for replacing abbrevaiations and removing special characters using regular expressions, *torchtext* consisting of utilities for data processing and popular natural language datasets and *torch* which provides tensor computation capabilities.

8.7.2 Experimental Design

TextBack has been tested for its ability to be a verifiable proof of ownership whilst not compromising on the fundamental machine learning performance. Additionally, the nuances of embedding watermarks and the computational cost of marking is also studied. The details of the entire experimental design is elaborated here:

- Benchmarking the neural architectures against traditional text classifiers using Test Accuracy and checking the failure of all such non-watermarked models in being a verifiable proof of ownership, through Trigger Accuracy.
- Checking the functionality preserving nature of watermarked models, using both types of Trigger Sets using Test Accuracy and observing the verifiability of the watermarks using Trigger Accuracy.
- Studying the process of embedding watermarks on the trained neural networks using Transfer Learning across different compositions of samples (clean samples and Trigger samples) necessary for fine tuning
- Studying the computational cost of embedding watermarks by training the model from scratch, without using a pre-trained model. This emphasises the benefit of using the TextBack Marking algorithm's fine tuning approach as opposed to embedding the watermarks during initial training of the model itself.

The aforementioned has been tested for multiple neural architectures (LSTM and vanilla RNN) and datasets (IMDB and Twitter).

8.7.3 Experimental Results

The experimental results are presented here. Table 8.4 shows the performance benchmarks of the models and establishes the non-verifiability of ownership of neural networks that are not explicitly watermarked.

TABLE 8.4: Benchmarking on performance and checking for non-verifiability of ownership in case of non-watermarked models.

Model	Dataset: IMDB		Dataset: Twitter	
	Test Accuracy	Trigger Accuracy	Test Accuracy	Trigger Accuracy
Vanilla RNN	84.81%	15%	80.92%	25%
LSTM	91.67%	8%	81.93%	19%
Naive Bayes	87.07%	12%	76.43%	21%
Logistic Regression	90.02%	13%	78.67%	23%

Table 8.5 presents the results supporting the fact that TextBack is consistent with the functionality preserving property, along with being capable of verification for establishing ownership.

TABLE 8.5: Test of functionality preserving property and verifiability of ownership of watermarked models.

Model	Trigger Samples	Dataset: IMDB		Dataset: Twitter	
		Test Accuracy	Trigger Accuracy	Test Accuracy	Trigger Accuracy
Vanilla RNN	Internal	84.80%	100%	75.84%	99%
	External	80.70%	100%	74.69%	99%
LSTM	Internal	91.63%	100%	80.05%	100%
	External	91.53%	100%	79.82%	100%

Table 8.6 presents the study on embedding watermarks in the neural networks on the IMDB dataset using TextBack by fine tuning with Trigger samples and clean samples.

TABLE 8.6: Study of embedding watermarks as part of the Marking algorithm on the IMDB dataset.

Model	Correct Samples	Train Accuracy	Test Accuracy	Trigger Accuracy
Vanilla RNN	0	61.58%	53.54%	100%
	1000	58.40%	52.46%	100%
	3000	87.10%	79.08%	100%
	5000	88.56%	81.80%	100%
LSTM	0	9.14%	14.39%	100%
	100	53.90%	53.84%	100%
	200	77.78%	73.20%	100%
	500	90.7%	84.66%	100%
	1000	96.46%	89.58%	100%
	2000	96.51%	91.63%	100%

Table 8.7 presents the study on embedding watermarks in the neural networks on the Twitter dataset using TextBack by fine tuning with Trigger samples and clean samples.

Table 8.8 presents the study on the computational cost of embedding watermarks by using TextBack by the data poisoning approach, where the model is trained from scratch using the clean samples and Trigger samples.

8.7.4 Key Findings

Some of the most significant observations and takeaways are mentioned here.

- The benchmarking of the models justify the choices of neural architectures as evident through the Test Accuracy. The Trigger Accuracy on the non-watermarked models substantiate the claim that they can not be used for any verifiable proof of ownership, as seen in Figure 8.6.

TABLE 8.7: Study of embedding watermarks as part of the Marking algorithm on the Twitter dataset.

Model	Correct Samples	Train Accuracy	Test Accuracy	Trigger Accuracy
Vanilla RNN	0	48.30%	46.55%	100%
	10000	87.38%	75.84%	99%
LSTM	0	25.30%	25.39%	100%
	100	72.00%	65.00%	100%
	200	73.39%	66.05%	100%
	500	85.60%	77.11%	99%
	1000	86.91%	78.81%	100%
	5000	88.66%	79.81%	100%
	10000	89.01%	80.05%	100%

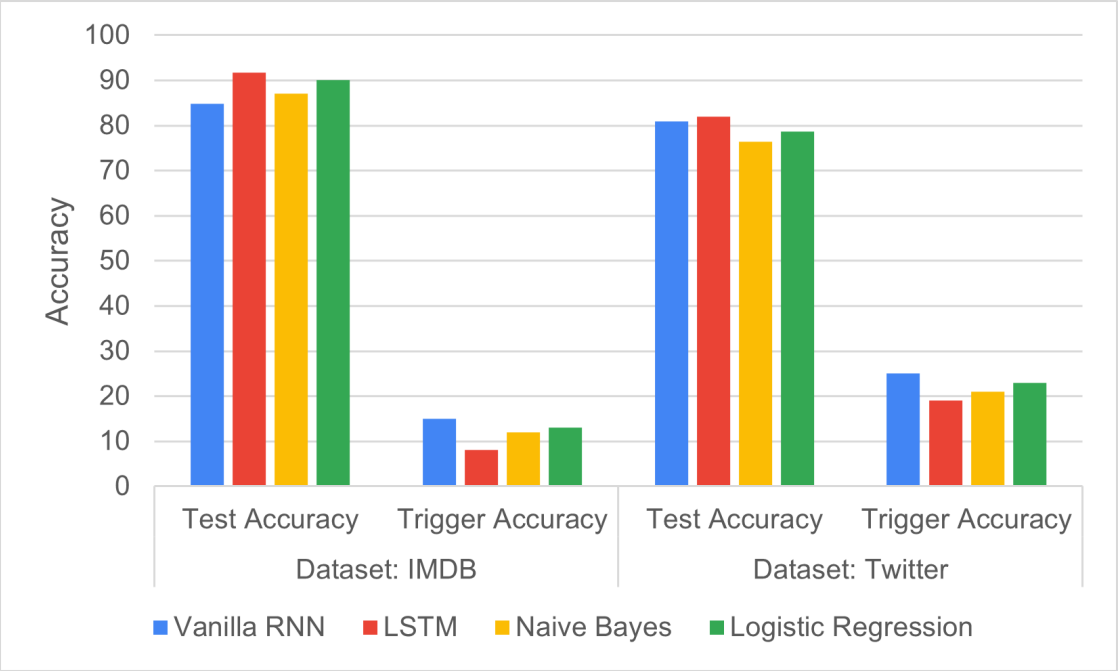


FIGURE 8.6: Benchmarking models and evidence of non-watermarked models in lacking of capabilities to establish ownership.

TABLE 8.8: Study of computational cost of TextBack as part of embedding watermarks while training model from scratch.

Model/ Dataset	Epochs	Batch-size	Training Accuracy	Test Accuracy	Trigger Accuracy
Model: LSTM Dataset: IMDB	1	64	64.97%	52.34%	16%
	10	64	95.35%	84.78%	28%
	20	64	98.23%	87.63%	59%
	30	64	99.12%	89.57%	82%
	40	64	99.51%	90.43%	92%
	52	64	100.00%	91.00%	100%
Model: LSTM Dataset: Twitter	180	2048	94.45%	76.16%	87%
	220	4096	97.38%	76.34%	90%
	300	4096	99.32%	80.82%	99%

- Upon watermarking the models with TextBack, the high performance of the watermarked models on the Trigger Set (of both kinds, Internal and External) is the verifiable proof of ownership of the model. It is to be noted here that the introduction of the watermarks do not disturb the performance of the original model, with the accuracy on the Test Set being in the range of $1-2\%$ of the non-watermarked model, as shown in Figure 8.7.

- An interesting nuance of embedding watermarks within text classifier neural networks is that during the fine tuning with the Trigger samples using Transfer Learning process, clean samples also need to be added to the mix. This is necessary for maintaining the functionality preserving property, as seen from Tables 8.6 and 8.7. For every specific combination of dataset and neural architecture, the corresponding size of clean samples needed is presented in the results.

- Instead of fine tuning the pre-trained models, if the models were to be trained from scratch with the Trigger samples and clean samples together, then the

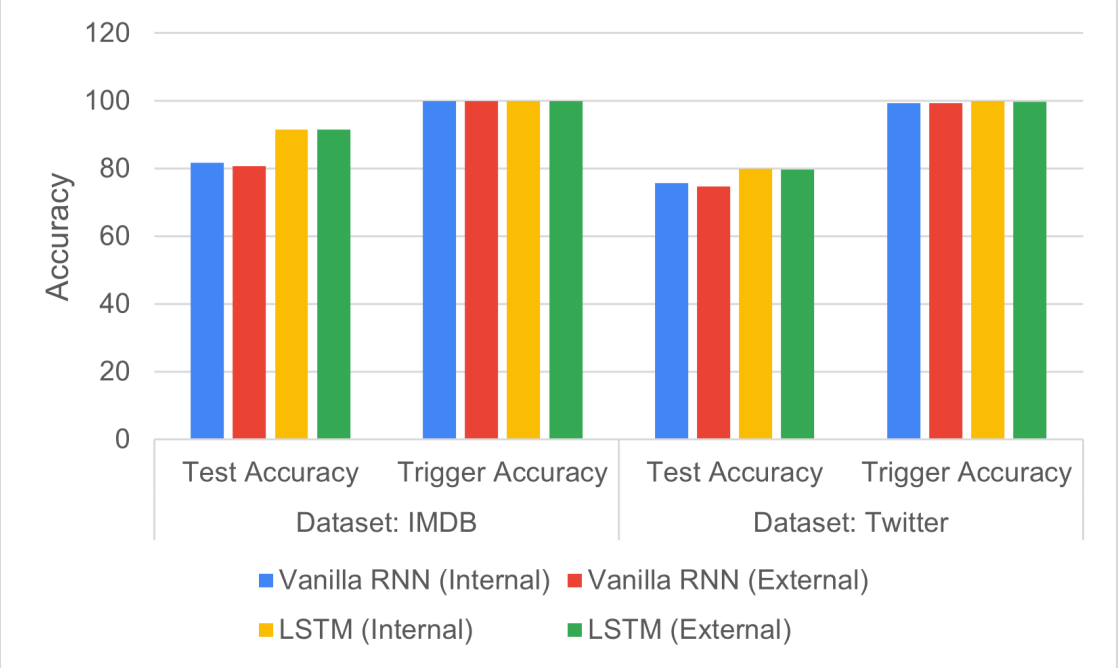


FIGURE 8.7: Demonstration of TextBack's capability to serve as a verifiable proof of ownership whilst being functionality preserving.

computational cost would be higher, in terms of epochs needed to train for, as evident from the results shown in Table 8.8.

TextBack therefore satisfies the conditions necessary for being an effective watermarking scheme. The functionality preservation aspect has been experimentally verified. The ownership is not trivial because the Trigger Set is unique, and extremely difficult to replicate. TextBack's embedded watermarks are unremovable and unforgeable owing to the properties it inherits from backdooring, where both Trigger samples and their labels are necessary for verification. This makes TextBack robust and reliable for practical use-cases.

8.8 Conclusions

Robust watermarking schemes are necessary for proper verifiable claiming of ownership of IP rights of trained neural network models. The stakeholders need a strong assurance that adversaries would fail to steal their models and use them without authorisation.

In this work, we propose a robust watermarking scheme called ROWBACK, that combines two properties of neural networks, the existence of adversarial examples and the ability to trap backdoors within a network during training. Specifically, we have redesigned the Trigger set making use of adversarial examples and modified the marking mechanism to ensure thorough distribution of the embedded watermarks.

We have tested ROWBACK with the most relevant state-of-the-art attacks to demonstrate its robustness. In future, we would like to study the watermarking scheme formally to prove that it can withstand all known attacks. We are exploring its vulnerability and robustness against different other types of attacks which are not directly relevant to its context. They violate the essential setting upon which ROWBACK is built, which include maintaining a secret Trigger Set and free access to verification of watermarks when required. Such attacks are those which prevent verification by screening samples before feeding to the network [62] or side-channel attacks.

As an extension, TextBack is a watermarking scheme for text classifier neural networks that uses backdooring to embed watermarks. Like any watermarking scheme TextBack has a dedicated Key Creation process where a Trigger set can be generated in two different ways, using samples within the distribution or from outside, known as Internal and External respectively. The TextBack marking process is different to most other watermarking schemes because as a necessary requirement for text modelling neural networks, it involves transfer learning based fine tuning using both the Trigger samples and some clean samples. The verification is done using the Trigger set, which is the private key in this case and resides only with the owner of the model. Going forward, we wish to test TextBack for more complex natural language understanding tasks using larger neural networks like transformers etc. The research in this domain will be particularly important because training these models is extremely expensive and is difficult to obtain for regular use, and therefore high in value. TextBack could provide a verifiable proof of ownership to such models.

Chapter 9

ROFL: RObust privacy preserving Federated Learning

9.1 Introduction

[1] Currently, with more and more data being generated in a decentralized manner through mobile devices or through sensors on edge devices communicating through the Internet of things (IoT), storing and harnessing them to extract insights is becoming a challenge. While this calls for a collaborative framework, from a practical perspective, there are some key deterrents to this paradigm. Firstly, storing all of the data collected by all the participating entities in a centralised location is not a lucrative idea for the participants, because of their privacy concerns. The data may contain sensitive information that the contributor would not like to reveal to the world; or they might not wish to let go of their ownership of the data that they have collected incurring procurement related costs; one may enlist a number of such reasons which stand opposed to a free sharing of data. That being said, the participants would definitely like to be the beneficiary of the collected data, in terms of insights generated from them. However, processing that voluminous amount of data in the sort of devices where they are generated, like mobile devices or edge devices is also not possible due to the resource constraints that accompany

[1] Chapter 9 is published in part as *Nandish Chattopadhyay, Anupam Chattopadhyay and Ritabrata Maiti, "Deploy-able Privacy Preserving Collaborative ML," in Proceedings of IEEE ICDCS 2020* and *Nandish Chattopadhyay, Arpit Singh and Anupam Chattopadhyay, "ROFL: RObust privacy preserving Federated Learning," in Proceedings of IEEE ICDCS 2022*

these devices. So the goal is to have a setup where a mutually agreeable model (common computational graph) can learn in a collaborative fashion, which will benefit from all the available data yet will not disclose any data to the other parties associated with the process of learning. Along with this, keeping in mind the constraints of computation on the individual participants, a lightweight model for inference on each client would further benefit the cause. Geyer et. al [67] took an initiative in the direction and proposed a decentralized learning scheme, achievable with federated learning, where the privacy of each participating client is ensured by implementing a privacy preserving machine learning algorithm. Despite the advancements in this direction, as mentioned earlier, there is a requirement to develop a robust framework for decentralised learning that ticks all the boxes. It should consider the lack of trust and therefore security and privacy guarantees thereof between each of the participants, participants with the cloud and vice versa. While the state-of-the-art in differentially private federated learning [111] takes care of a part of it, the threat particularly pertains to sabotaging and poisoning attacks by malicious clients [112]. This serves as the primary motivation of developing the ROFL infrastructure.

9.1.1 Contribution

In this work, we have addressed the requirements in privacy guarantees that are necessary for a functional collaborative learning infrastructure. Having acknowledged the lack of trust that exists between all involved parties, we have studied the trust framework in a bi-directional privacy setup and developed ROFL. It is an algorithm that is able to carry out the fundamental machine learning task without compromising on privacy. We have built upon the state-of-the-art in privacy preserving federated learning [111] by modifying the aggregation algorithm. Specifically, the key contributions of this work are:

- development and implementation of a highly robust infrastructure for differentially private federated learning
- providing two-dimensional and bi-directional privacy guarantees on the data and trained model belonging to each participating client and providing security guarantees at the aggregator level through counter-measures (weighted federated averaging) against any potential malicious sabotaging attacks

- improving efficiency by studying model convergence and updating the global model only when necessary
- studying scalability of the ROFL infrastructure for higher number of clients, for potential industrial acceptance/adoption.
- minimizing resource utilization by creating a lightweight model deploy-able on edge devices using quantization.

The ROFL infrastructure has been tested for its functionality, robustness and scalability on multiple neural architectures (VGG-16 [1] and ResNet [7]) on multiple datasets (MNIST [5], CIFAR-10 and CIFAR-100 [6]). We have substantiated our analysis using multiple application specific tasks relevant to the context of decentralized learning, proprietary data belonging to participating clients and deployment on edge and mobile computing devices.

9.2 Threat Models

To put our work into perspective, we explore the space of potential threat models that need to be dealt with, in order to establish a reliable decentralised learning system, that is able to combine the insights of data in a privacy preserving way. In particular, there are three different areas of concern with respect to the integrity of the data and the model, that have to be addressed here. We study the weaknesses of the generic setup of collaborative learning systematically.

9.2.1 Compromised Sensitive Data

The data that resides with each participant of the distributed learning setup is most often proprietary. This is natural as the corresponding stakeholders have to invest resources into collecting and curating the data in an orderly fashion. Additionally, the data may come from specific sources wherein it could be of highly sensitive nature. This is specifically true for the domain of healthcare, where the data contributors are mostly patients and therefore it is very natural that they would not be comfortable with sharing it with those who must not. In such cases,

although there is a lot of data that is present, it can never be collected in a central location, due to the constraint of compromise in privacy.

Data is distributed. The data must reside separately with each of the participants. Naturally, a model that would have been trained on the entire dataset would naturally be more accurate than one that is trained on one of the individual sources. This necessitates the requirement of a setup that is able to train a model which is comparable in performance to the model that would have been trained on the collected dataset, without actually having to collect the data [65]. This provides assurance to those who provide or help generate the data.

9.2.2 Membership Inference Attacks

It has been shown that black box attacks can be mounted on trained neural networks and an adversary is able to learn whether a particular sample is part of the training set or not, just by querying the system [64]. Membership Inference attacks are designed to steal the training datasets of the models, by using the learned parameters of the trained neural networks.

Training data can be stolen. Membership Inference Attacks severely undermine the integrity of the training data [64]. Stakeholder who have invested on gathering and curating the data for the purpose of training their models realise this threat and therefore are unwilling to share their models for aggregation. The lack of trust between the clients and the aggregator is a major deterrent towards the adoption of the federated learning infrastructure, and robustness and security guarantees on it are necessary.

9.2.3 Malicious Sabotaging

While the overall model after federated averaging does benefit from the contributions of each, it is also true that is there are one or more poor quality models, that may potentially bring down the capabilities of the aggregated model [68]. Poor quality models could be sent by the participants due to both intentional and unintentional reasons. There may be a situation where the data that a model sees

does not have good insights and therefore the model generated thereof is not high performing.

Participants can be malicious. While a participating client can genuinely generate a poor performing model, there can also be malicious attacks. There can be rogue participants who want to bring down the overall performance of the entire system and can deliberately push erroneous models. Either way, this necessitates the proposition of a check on the quality of the model pushed by the participants before accepting it into the process of aggregation.

9.3 Trust Framework

In order to study the different parts of the learning infrastructure and the specific importance of each component, it is necessary to define the framework of trust first. The overall setup provides two-dimensional privacy guarantees to the users. Essentially, the players are the participating clients and a third-party aggregator. The clients, within their own systems, work assuming a trusted computing environment. However, they do not trust other clients or the third party aggregator with their data. Similarly, the aggregator which consumes trained models from the participating clients, do not trust the clients as well. There is a bi-directional lack of trust in the design, which is common to expect in the real world scenarios. In this work, the proposed infrastructure ensures functionality without having to compromise on the trust apprehensions, while adding robustness to the process by keeping performance checks in the algorithm.

9.3.1 Horizontal Privacy

In the schematic, the participating clients are arranged in a horizontal placement. Each participating client has its own dataset, that it uses to update the locally owned model, which is eventually sent to the aggregator at the end of every communication round. All of the computation on the client side takes place within the trusted computing environment of the clients. The data is not shared with any other entity that's placed in the horizontal frame of the schematic. The federated

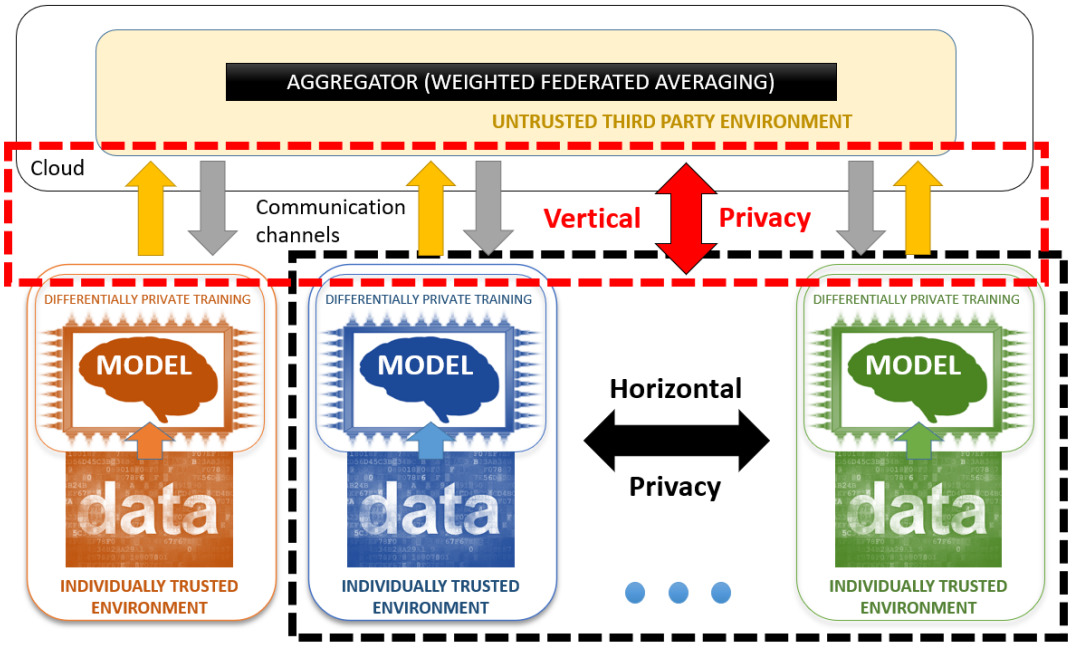


FIGURE 9.1: The trust framework involving the clients and the aggregator.

learning infrastructure [63] is used to generate a model that is comparable in performance to a model that would have been trained on the collected data, without actually collecting the data.

9.3.2 Vertical Privacy

The clients also do not trust the third party aggregator, and neither does the aggregator trust the clients. In order to preserve the integrity and privacy of the datasets locally owned by the participating clients from potential membership inference attacks we make use of differentially private training [113]. From the perspective of the aggregator, it receives the updates from the clients and performs a robust federated averaging. To prevent any potential sabotaging attempt by malicious clients, or due to any attacks during the communication between the client and the aggregator, it has a scheme in place to test the quality of the model pushed by the clients and the system automatically reduces the weight-age of any poorly performing model. In essence, the system is designed to reject any malicious model. Also, it incorporates a fresh update from a client only if the newly sent model is sufficiently different from the existing one. The client side also has the

facility to reject an updated model sent from the aggregator if it seems to be erroneous.

9.4 Algorithms

At the heart of ROFL are two key algorithms, Federated Learning [63] and Differential Privacy [114–116]. We use them as the pillars and modify them for our use-case, adding features that add to the security guarantees. Before we get to the exact design of ROFL, we take a brief look at the two fundamental components.

9.4.1 Federated Learning

The federated averaging algorithm, used for neural networks, deals with a non-convex objective function. The governing algorithm is applicable for any finite-sum objective of the form [63]:

$$\boldsymbol{min}_{w \in \mathbb{R}^d} \; f(w)$$
$$where \; f(w) = \frac{1}{n} \sum_{i=1}^{n} f_i(w)$$

Considering any standard machine learning task, we have a prediction loss l on any particular sample (x_i, y_i), and the corresponding parameters of the model w, we are interested in $f_i(w) = l(x_i, y_i; w)$. If we assume that there are K participating clients in the system, wherein every client k has their own dataset P_k of n_k samples, then we can define the objective function as:

$$f(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w)$$
$$where \; F_k(w) = \frac{1}{n_k} \sum_{i \in P_k} f_i(w)$$

Federated SGD: From the implementation perspective, the Federated Stochastic Gradient Descent works with a fixed learning rate η, and each client k carries out the computation $g_k = \nabla F_k(w_t)$, which is the averaged gradient corresponding to

the local data of the client on the latest model w_t. This is pushed to the aggregator where the federated averaging takes place which generates the global model that is thereafter pulled by the participating clients. All the clients train the same neural architecture with their locally available data. The process continues over multiple communication rounds. The key components of the differentially private SGD algorithm are discussed hereafter.

Establishing a guarantee on the differential privacy necessitates the assurance of a bound on the influence of a particular sample on the gradient vector $\tilde{\boldsymbol{g}}_{\boldsymbol{t}}$. Now since there is no theoretical bound beforehand on the gradients, each of the computed gradients are clipped in l_2 norm, that is instead of considering the gradient vector $\boldsymbol{g}$, the corresponding $\boldsymbol{g}/max(1, \frac{||\boldsymbol{g}||_2}{M})$ is used, where M is the clipping threshold. The rationale behind this step is to ensure that is if $||\boldsymbol{g}||_2 \leq M$ then the gradient is preserved and otherwise the gradient gets scaled down to the norm. This process ensures that the impact that specific samples have on the overall gradient vector is reduced and therefore leads to a sort of regularization in training [113].

A neural network is parameterised by a multitude of weight matrices, corresponding to the transformations between the layers. Whilst typically all of the parameters are grouped together into the weights w which is input to the loss function L, as described above, one may consider the parameters separately for all the layers of the network. So for a multi layered neural network, the steps of clipping the gradients may make use of different thresholds and the step of addition of the noise may use different scales. These choices may also be dependent on the number of training steps.

Akin to the standard stochastic gradient descent, the proposed algorithm computes the gradient of L by averaging the gradients of a group of samples at a time. This approach of averaging results in an unbiased estimate, which has low variance if the size of the group is large enough. As this is not the standard computational grouping of samples in batch gradient descent known as *batches*, the nomenclature used in this case is *lots*. For every step of the algorithm, the the computation of calculating the gradients is done for the individual *batches* using the typical SGD and then the averaging is done over the *lots*. the addition of the noise for differential privacy is carried out thereafter.

As mentioned earlier, an important task of the differentially private training is to keep an eye on its corresponding privacy cost. This is of high significance towards our analysis discussed in the following section. Since differential privacy follows the composability property, one can formulate an accounting mechanism that is able to accumulate the privacy cost for each and every access to the training dataset. Every iteration of the training process involves an access to the training data and a proper privacy accountant is able to keep track of the corresponding overall privacy costs. The results of the privacy accounting have been reported in the subsequent section.

The ROFL infrastructure considers a system comprising of K clients, indexed by k, local minibatch being B, local epochs being E, learning rate being η_t in the t-th iteration, loss function being $L(w_t)$, group size being G, validation dataset V, coefficient for federated averaging l_k, δ is the threshold on difference between consecutively pushed models.

9.4.2 Differential Privacy

There are different ways of incorporating privacy preserving learning techniques. The ROFL infrastructure uses Differential Privacy [113] for the same. Essentially, this approach intervenes into the optimization of the learning model, by using differentially private stochastic gradient descent.

Privacy Preserving Learning: We start with the consideration that a particular participating client has its local training samples $\{x_1, \ldots, x_{n_k}\}$. The neural architecture, with the trainable weights w will be trained by the minimization of the loss function $L(w)$. With each iterative step of the SGD, we calculate the gradient $\nabla_w L(w, x_i)$ for any randomly selected subset of samples. The Euclidean norm l_2 of the gradient, $\boldsymbol{g}$ are clipped. They are averaged, and a samples noise is added for protecting privacy. Thereafter, the optimizer takes a step in the opposite direction to the noisy gradient. Notably, along with the process of generating the differentially private model, there is mechanism in place for privacy accounting, which takes care of computing the losses incurred due to the establishment of the DP training.

The key components of the differntially private SGD algorithm are discussed hereafter.

Gradient clipping: Establishing a guarantee on the differential privacy necessitates the assurance of a bound on the influence of a particular sample on the gradient vector $\tilde{\boldsymbol{g}}_{\boldsymbol{t}}$. Now since there is no theoretical bound beforehand on the gradients, each of the computed gradients are clipped in l_2 norm, that is instead of considering the gradient vector $\boldsymbol{g}$, the corresponding $\boldsymbol{g}/max(1, \frac{||\boldsymbol{g}||_2}{M})$ is used, where M is the clipping threshold. The rationale behind this step is to ensure that is if $||\boldsymbol{g}||_2 \leq M$ then the gradient is preserved and otherwise the gradient gets scaled down to the norm. This process ensures that the impact that specific samples have on the overall gradient vector is reduced and therefore leads to a sort of regularization in training [113].

Whilst typically all of the parameters are grouped together into the weights w which is input to the loss function L, as described above, one may consider the parameters separately for all the layers of the network. So for a multi layered neural network, the steps of clipping the gradients may make use of different thresholds and the step of addition of the noise may use different scales. These choices may also be dependent on the number of training steps.

Differentially Private SGD: Akin to the standard stochastic gradient descent, the proposed algorithm computes the gradient of L by averaging the gradients of a group of samples at a time. This approach of averaging results in an unbiased estimate, which has low variance if the size of the group is large enough. As this is not the standard computational grouping of samples in batch gradient descent known as *batches*, the nomenclature used in this case is *lots*. For every step of the algorithm, the the computation of calculating the gradients is done for the individual *batches* using the typical SGD and then the averaging is done over the *lots*. the addition of the noise for differential privacy is carried out thereafter.

Privacy Cost Monitoring: As mentioned earlier, an important task of the differentially private training is to keep an eye on its corresponding privacy cost. This is of high significance towards our analysis discussed in the following section. Since differential privacy follows the composability property, one can formulate an accounting mechanism that is able to accumulate the privacy cost for each and every

access to the training dataset. The results of the privacy accounting have been reported in the subsequent section.

9.4.3 Quantization

Neural networks are highly parameterized and are characterized by a computational graph and the corresponding weight matrices. These voluminous models take up a lot of space and involve high precision computations. For the purpose of deployment, one needs to produce a lightweight version that is suitable for the resource-constrained environment of an edge device. Quantization provides a mechanism to scale the weights of the neural network from floating point to 8-bits of precision. At inference, weights are converted from 8-bits of precision to floating point and floating-point kernels are used for computation. This is a one-time conversion and then it is cached to reduce latency.

9.4.4 ROFL

The complete algorithmic setup is shown in Algorithm 1. The overall ROFL infrastructure has two distinct parts, the Aggregator and the Participating clients. Considering a system comprising of K clients, indexed by k, local minibatch being B, local epochs being E, learning rate being η_t in the t-th iteration, loss function being $L(w_t)$, σ being the noise scale, group size being G, M being gradient norm bound, validation dataset V, coefficient for federated averaging l_k, δ is the threshold on difference between consecutively pushed models, we articulate the algorithm.

9.5 Implementation

In this section, we discuss the learning infrastructure pipeline comprising of each of the components. In compliance with the Trust framework described in 9.3, we consider trusted computing environments for each of the participants and the aggregation takes place in the cloud infrastructure. The clients do not trust any of the other clients and the aggregator, similarly the aggregator does not trust the clients. The communication between the clients and the aggregator is through

Algorithm 1: The ROFL Algorithm

***Aggregator* :**

initialize the weights w_0

for *every iteration* $t = 1, 2, \ldots, r$ **do**
  $m \leftarrow max(K, 1)$
  $S_t \leftarrow (m$ clients$)$
  for *each client* $k \in S_t$ **do**
    if $||w_{t+1}^k - w_t^k|| > \delta$ **then**
      $w_{t+1}^k \leftarrow Client\_Update(k, w_t)$
      $l_k \leftarrow WFA(w_{t+1}^k, V)$
    end
    else
      $Ignore(w_{t+1}^k)$
    end
    $w_{t+1} \leftarrow \sum_{k=1}^{K} l_k \frac{n_k}{n} w_{t+1}^k$
    Return w_{global}
  end
end

$WFA(w, V)$: (weighted federated averaging)

for *every client model* $w_k, k \in 1, \ldots, K$ **do**
  $l_1, \ldots, l_k \leftarrow \frac{Inference(w_k)}{\sum_{k=1}^{K} Inference(w_k)}$
end

Send w_{global} to the clients.

***Participants* :**

$Client\_Update(k, w)$:

$B \leftarrow$ (Splitting of P_k into batches)

for *every local epoch* i *from* 1 *to* E **do**
  for *batch* $b \in B$ **do**
    Randomly Initialize w_0
    for $t \in [T]$ **do**
      Random sampling G_t with sampling probability G/n_k
      Computation of Gradient:
      For each $i \in G_t$, compute $\boldsymbol{g_t}(x_i) \leftarrow \nabla_{w_t} L(w_t, x_i)$
      Clipping of the Gradient:
      $\bar{\boldsymbol{g}}_{\boldsymbol{t}}(x_i \leftarrow \boldsymbol{g}_t(x_i)/max(1, \frac{||\boldsymbol{g}_t(x_i)||_2}{M})$
      Adding the Noise:
      $\tilde{\boldsymbol{g}}_{\boldsymbol{t}} \leftarrow \frac{1}{G}(\sum_i \bar{\boldsymbol{g}}_{\boldsymbol{t}}(x_i + N(0, \sigma^2 M^2 \boldsymbol{I}))$
      Descent:
      $w_{t+1} \leftarrow w_t - \eta_t \tilde{\boldsymbol{g}}_{\boldsymbol{t}}$
      Return w_{local}
    end
    $(\epsilon, \delta) \leftarrow Privacy\_Accountant(n_k, |b|, \sigma, M)$
  end
end

Push w_{local} to the Aggregator

open channels, and is therefore not trusted. Keeping these in mind, we build the ROFL architecture as per the algorithm described above.

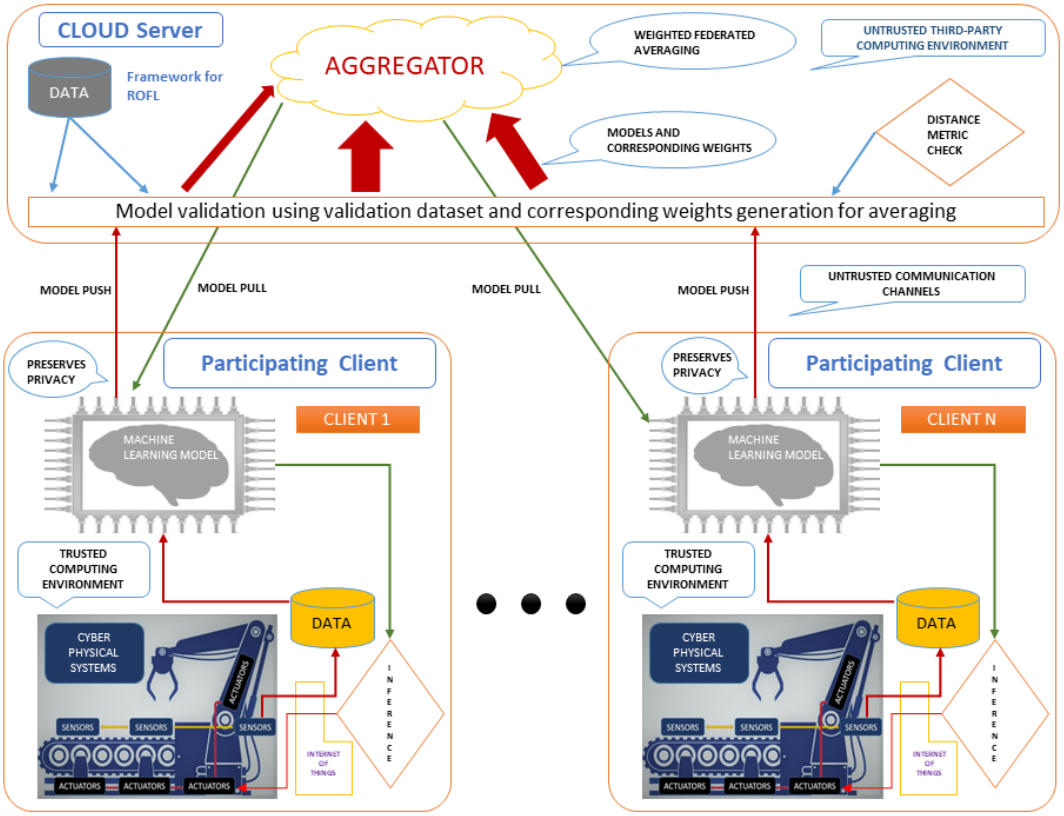


FIGURE 9.2: Schematic diagram of the complete ROFL infrastructure.

9.5.1 Learning Infrastructure

Clients: Each of the clients have their own datasets. At every communication round, the local models train over the locally available data and update the model. This local model is trained using differentially private Stochastic Gradient Descent, so as to ensure that once the local model leaves the client's trusted computing environment, membership inference attacks are not possible. The integrity of the local data is always preserved, owing to the security guarantees of differential privacy. Once this local model is updated, the fresh one is sent to the aggregator for the federated learning infrastructure. This is where the local models leave the trusted computing environment for the first time.

Aggregator: At every communication round, multiple clients push their models to the aggregator. The aggregator has a validation dataset that it updates over

time. This is a labelled set of samples drawn from the distribution of data from which each of the clients have gathered their local datasets. Whenever the aggregator receives an updated model from clients, it checks for the quality of the model on its own validation dataset. Based on the performances of the pushed updated models on the validation set residing with the aggregator, weights are assigned to the client models. These weights are then used for the Federated Averaging process, which considers a weighted average of the updates. Essentially, the weighted federated averaging proposed for ROFL is such that the resulting model consists of a linear combination of the individual contributing models, where every coefficient is determined by the aggregator using the performance testing and scoring.

Convergence: Additionally, based on design parameter, the frequency of checking for updating the global model is decided. After every few communication rounds, the aggregator, upon receiving an updated model, calculates the difference between its last updated model and the latest update, and if the separation is above a threshold, the new update is recognised and incorporated within the weighted federated averaging process. The difference is calculated using a distance metric, typically the Euclidean norm. The threshold is a hyper-parameter that may be tuned as needed. The updated model, once sent to the clients, can also be verified by the clients using a validation set exactly as the aggregator does.

9.5.2 Robustness

The robustness of ROFL is directly addressed towards the trust framework in which it operates, and the threats it faces therefore. As a counter-measure to the threats discussed in Section 9.2, ROFL is designed to deliver bi-directional two-dimensional robustness. Since there are threats along both directions between the aggregator and the clients, the counter measure should also be commensurate.

Specifically, to address the requirement of horizontal privacy between the clients, ROFL uses the federated infrastructure. For the requirements on the vertical privacy aspect, the lack of trust on the part of the clients on the aggregator is taken care of by the addition of differential privacy. This is also important because the communication channel between the client and the aggregator is not trusted and once the update leaves the trusted computing environment of the client, any malicious adversary will not be able to mount a membership inference attack should

it tap in to the communication. The robustness of the system when taking into consideration the lack of trust of the aggregator on the participating clients, is addressed by putting in place the mechanism of having a validation check for each model updates and considering only an weighted federated averaging. This ensures that poorly performing models will anyway be assigned a very low weight and therefore any rogue client will not be able to sabotage the system maliciously.

Additionally, this also ensures robustness against any unintended communication loss that may happen during communication and the the resultant tampered update sent to the aggregator is not able to hamper the overall performance of the system. It may be noted here that the global weighted federated averaged model updates if there is at least one high performing new model pushed to the aggregator from one of the clients.

9.5.3 Efficiency

The computational load of the ROFL infrastructure is distributed among the clients and the aggregator. The clients carry out the training of the neural networks in parallel within their own trusted computing environments. The bottleneck of the overall system is the weighted federated aggregation process, which is initiated every time new updated models are pushed to the cloud. In order to study how to improve on the efficiency of the system, there is a provision in place to check whether the latest pushed update is significantly different from the last updated global model or not. The difference is measured using simple Euclidean norm. The freshly sent client model is incorporated into the weighted federated averaging process only if it crosses a threshold.

The threshold is a hyper-parameter to the system by design. The intuition behind this is that, after a few communication rounds, the global model will eventually become quite high performing, and the fresh updates will become minimal. As the performance saturates, in order to prevent a thorough change in the global model, the infrastructure ensures that only the major updates are incorporated into the system. This makes the setup more efficient.

9.6 Experiments

In this section, we present the complete set of experimental results and key finding that have come out of it. We begin with outlining the experimental design. Then we mention the details of the models and datasets that have been used in our experiments. This is followed by the tabulated results. Finally, we emphasize upon the key takeaways from all the experiments. All experiments and simulations of the ROFL infrastructure were carried out in the Google Colab platform [117], with an additional Tesla K80 GPU as hardware accelerator.

9.6.1 Design

The experimental design can be split into three parts. Each of the parts have their own experimental objectives to study. In essence, we first study the functionality of the ROFL infrastructure, which is followed by a study of its efficiency and finally, we look deeper into how ROFL scales for many participating clients.

Functionality

We present a comparative study of the following scenarios, wherein each is tested for multiple neural architectures and datasets:

- the performance of inference for the neural network trained on the entire data collected together. This is the baseline for our study. This is the best possible scenario for the model to learn the insights, which can not happen due to the threats and trust framework.
- the performance of inference for the neural network on a randomly selected client among the many participating clients. This is the worst case scenario, with the model being exposed to limited data.
- the performance of an overall system where the models are individually trained on each local dataset belonging to every participating client and then a weighted majority voting is considered. The outcome of this either necessitates the federating averaging scheme or makes it redundant.

- the performance of a collaborative learning system with simple federated averaging. This is the baseline among the federated learning infrastructures.
- the performance of a collaborative learning system with weighted federated averaging scheme. This provides the additional robustness to the system.
- the performance of ROFL, which combined weighted federated averaging with differential privacy

Robustness

We studied how the performance of the ROFL infrastructure degrades as the participating clients start to fail. We simulate the failure of a client by using a randomly initialized model for it to push to the aggregator. We increase the number of failing clients from one to eventually all, and note the coefficients associated with each of the corresponding client models and the overall accuracy of the global model at that stage. This study gives us a clear indication of the capacity of the ROFL architecture to defend against attacks and be robust in nature.

Efficiency

In this set of experiments, we study how the learning of the neural architecture used in ROFL evolves over the communication rounds. The goal is to know how often to fine tune the global model using the weighted federated averaging technique based on the quality of updates that are pushed to the aggregator by the participating clients. As mentioned in Section 9.5, the aggregator has the option to include an update pushed by the client or reject it. We measure the difference between the existing model and the fresh update using Euclidean norm. If the distance is greater than a threshold, only then is the update incorporated in the weighted averaging. The tabulated results show the values of the distances after every communication round.

Scalability

ROFL is designed to be extremely scalable. This is necessary for most practical applications of many clients being involved in the collaborative setting. The experiments to study the system's scalability simulates the entire ROFL infrastructure with a varying number of clients. The goal of these experiments is to observe how, with the increase in the number of clients, the notional gap between the individual clients' performances and the global model's performances differ. The expectation is that the clients, with limited data would be unable to generate satisfactory performances, unlike the global weighted federated averaged model.

9.6.2 Models and Datasets

For the purpose of our experiments, we looked at very well studied image processing tasks. This is particularly relevant because these machine learning tasks are well benchmarked with standard neural architectures and datasets. This helps in comparison of the proposed infrastructure to what is available in the literature. We have used the VGG-16 [1] and ResNet [7] as the neural architectures and MNIST [5] and CIFAR-10 and CIFAR-100 [6] as the datasets. For the experiments involving Differential Privacy, the corresponding privacy accounting showed $\epsilon = 0.89$ and $\delta = 1.33 * 10^{-05}$.

9.6.3 Results

In this section, we present all the experimental results in details. The experiments are exactly in order as described earlier. Table 9.1 showcases the accuracies of the respective setups under corresponding conditions with the neural architecture being VGG-16 [1] and the datasets MNIST [5] and CIFAR-10 and CIFAR-100 [6].

Table 9.2 presents the accuracies of the respective setups under corresponding conditions with the neural architecture being ResNet [7] and the datasets MNIST [5] and CIFAR-10 and CIFAR-100 [6].

In Table 9.3, we study the robustness of ROFL against attacks.

TABLE 9.1: Test for functionality of ROFL infrastructure with the neural architecture being VGG-16 [1] and the datasets MNIST [5] and CIFAR-10 and CIFAR-100 [6].

Experimental Design	MNIST	CIFAR-10	CIFAR-100
Model trained on Complete Dataset	99.71	85.05	69.04
Randomly chosen Individual Client	93.97	72.18	20.89
Majority Voting Approach	97.84	80.38	44.67
Simple Federated Averaging	98.87	83.77	55.69
Weighted Federated Averaging	98.73	83.53	55.31
ROFL Infrastructure	98.09	79.93	52.17

TABLE 9.2: Test for functionality of ROFL infrastructure with the neural architecture being ResNet [7] and the datasets MNIST [5] and CIFAR-10 and CIFAR-100 [6].

Experimental Design	MNIST	CIFAR-10	CIFAR-100
Model trained on Complete Dataset	99.57	91.40	67.76
Randomly chosen Individual Client	94.36	69.69	23.94
Majority Voting Approach	98.01	78.45	42.68
Simple Federated Averaging	98.92	83.53	54.34
Weighted Federated Averaging	98.62	83.91	54.03
ROFL Infrastructure	98.11	80.29	51.47

In table 9.4, we study the convergence of the models after successive communication rounds. We measure Euclidean distances of the client models $C1 \dots C10$ with the global model, through the communication rounds $Round1 \dots Round10$. Here the results for the dataset MNIST [5] are presented.

In table 9.5, we study the convergence of the models after successive communication rounds. We measure Euclidean distances of the client models $C1 \dots C10$ with the

TABLE 9.3: Test for robustness of ROFL infrastructure studied against failure of participating clients, with dataset MNIST [5] and ResNet [7]. The coefficients corresponding to the useful models are marked in bold.

Failed Clients	Coefficients of Client Models	RoFL Accuracy
0	**0.10, 0.09, 0.09, 0.10, 0.10, 0.10, 0.09, 0.10, 0.09, 0.10**	98.76%
1	0.01, **0.10, 0.10, 0.11, 0.11, 0.11, 0.10, 0.11, 0.10, 0.11**	98.75%
2	0.01, 0.01, **0.12, 0.12, 0.12, 0.12, 0.12, 0.12, 0.12, 0.12**	98.61%
3	0.01, 0.01, 0.01, **0.13, 0.13, 0.13, 0.13, 0.13, 0.13, 0.13**	98.60%
4	0.01, 0.01, 0.01, 0.01, **0.15, 0.15, 0.15, 0.15, 0.15, 0.15**	98.60%
5	0.01, 0.01, 0.01, 0.01, 0.01, **0.18, 0.18, 0.18, 0.18, 0.18**	98.07%
6	0.02, 0.02, 0.01, 0.02, 0.02, 0.02, **0.21, 0.21, 0.21, 0.21**	95.42%
7	0.02, 0.02, 0.01, 0.02, 0.02, 0.02, 0.03, **0.27, 0.27, 0.27**	76.98%
8	0.03, 0.03, 0.02, 0.03, 0.03, 0.03, 0.03, 0.03, **0.36, 0.36**	16.50%
9	0.05, 0.05, 0.03, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, **0.53**	11.34%
10	0.15, 0.09, 0.09, 0.09, 0.09, 0.10, 0.08, 0.08, 0.09, 0.09	10.32%

global model, through the communication rounds $Round1 \ldots Round10$. Here the results for the dataset CIFAR-100 [6] are presented.

In Table 9.8, we study how the ROFL infrastructure scales with increasing number of clients.

9.6.4 Key Findings

The most important observations and key takeaways from the experiments are presented here.

- A model trained on the entire pooled dataset, which we may refer to as a full model, is the best performing model. When the participating clients train the

TABLE 9.4: Test for model convergence: Euclidean distances of the client models $C1 \dots C10$ through the communication rounds $Round1 \dots Round10$ for the dataset MNIST [5] and ResNet [7].

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
Round 1	4.61	3.93	3.49	3.27	2.94	2.74	2.61	2.47	2.30	2.24
Round 2	0.83	0.80	0.74	0.71	0.70	0.65	0.61	0.58	0.59	0.55
Round 3	0.43	0.43	0.41	0.39	0.38	0.38	0.35	0.34	0.35	0.32
Round 4	0.30	0.30	0.29	0.27	0.28	0.27	0.26	0.26	0.26	0.23
Round 5	0.24	0.24	0.23	0.22	0.22	0.21	0.20	0.21	0.21	0.19
Round 6	0.19	0.19	0.20	0.18	0.19	0.18	0.17	0.17	0.18	0.16
Round 7	0.16	0.16	0.17	0.15	0.16	0.16	0.15	0.15	0.15	0.14
Round 8	0.14	0.15	0.15	0.14	0.14	0.14	0.13	0.13	0.14	0.13
Round 9	0.13	0.13	0.14	0.12	0.13	0.13	0.12	0.12	0.12	0.11
Round 10	0.12	0.12	0.12	0.11	0.12	0.11	0.11	0.11	0.11	0.11

same neural architecture on their local datasets, the performance in terms of accuracies is consistently not close to the former. Additionally, considering a majority voting on the predictions of each of the clients is also not able to match up with the performance that would have been obtained using a collected dataset. This emphasises the motivation of the problem and in part necessitates an approach like that of ROFL.

- The collaborative learning framework using the various modes of Federated Learning work well. As evident from Figure 9.3, the introduction of mechanisms of ensuring robustness and privacy do not bring down the performance significantly, being in the range of $1 - 2\%$ of that of the full model in the average case scenario. There is a huge gain in performance of the ROFL infrastructure with respect to the accuracies of the individual clients.

- The ROFL infrastructure is extremely capable in dealing with failures of clients, which may be unintentional or deliberate sabotaging or malicious

TABLE 9.5: Test for model convergence: Euclidean distances of the clients $C1 \dots C10$ through the communication rounds $Round1 \dots Round10$ for the dataset CIFAR-100 [6] and ResNet [7].

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
Round 1	431	926	938	929	931	973	887	872	844	975
Round 2	219	212	211	264	250	275	245	253	209	232
Round 3	134	129	132	151	121	179	132	149	121	150
Round 4	141	120	116	116	92.7	118	107	103	106	119
Round 5	99.1	98.8	106	109	100	106	120	109	122	122
Round 6	104	97.1	114	103	132	123	134	140	127	127
Round 7	111	115	125	114	119	95.1	107	115	115	109
Round 8	83.8	84.4	89.9	86.3	86.9	72.2	85.3	91.5	85.2	78.2
Round 9	69.7	70.6	27.2	23.1	21.9	19.5	23.2	22.3	24.6	20.4
Round 10	14.1	15.1	15.1	14.1	12.7	11.9	12.8	15.1	15.4	13.1

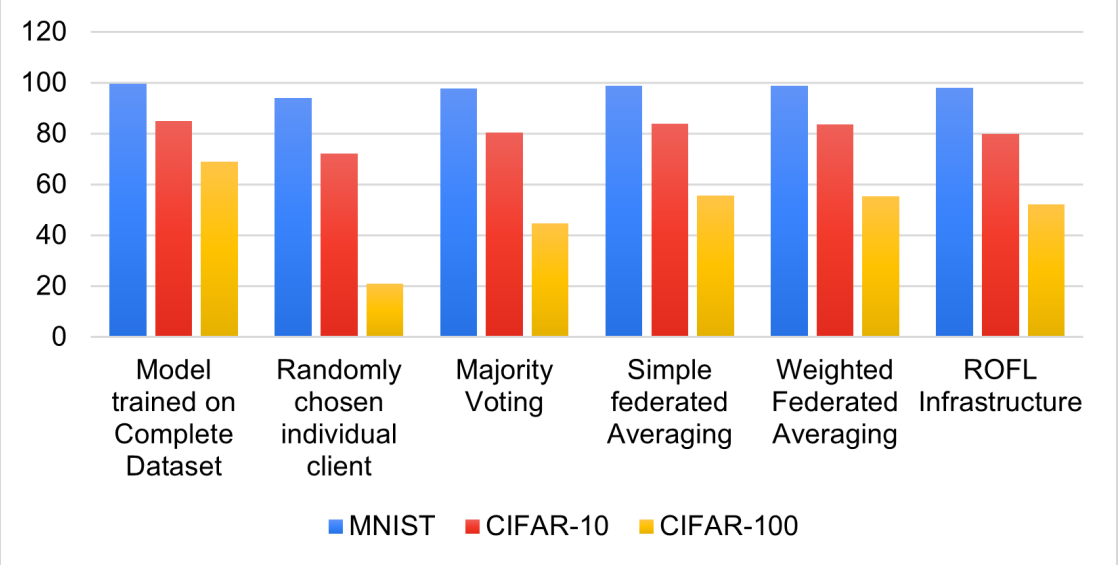


FIGURE 9.3: Study of functionality with neural architecture being VGG-16.

TABLE 9.6: Test for Scalability of ROFL infrastructure.

Number of Clients	Randomly chosen Individual Client	Weighted Federated Averaging	ROFL Infrastructure
2	94.15	98.60	98.76
4	21.26	98.54	98.52
8	18.66	97.81	98.15
12	13.42	97.33	97.83
16	11.76	97.06	97.31
20	11.55	96.60	96.89
24	11.03	95.58	96.45
28	11.10	95.20	96.29
32	10.74	95.43	95.05
36	10.31	93.96	93.16
40	10.34	92.25	92.11

attacks. ROFL can withstand failures, with up to at least 50% of its contributing clients pushing poorly performing models to the aggregator, as seen in Figure 9.5 and still generate high overall accuracy.

- Using weighted federated averaging is able to generate almost equal accuracy as ordinary averaging, whilst providing defences against any malicious sabotaging. Similarly, the addition of differential privacy ($\epsilon = 0.89$ and $\delta = 1.33 * 10^{-05}$) keeps the performance of the system within the $1 - 2\%$ range while lending robustness against any membership inference attack that jeopardises the trainset.

- The study on making the framework more efficient by taking an additional step occasionally, to check how different the freshly pushed client updates are, reveals that there is convergence of the models after some communication rounds. The Euclidean distance metric diminishes over time, and from that

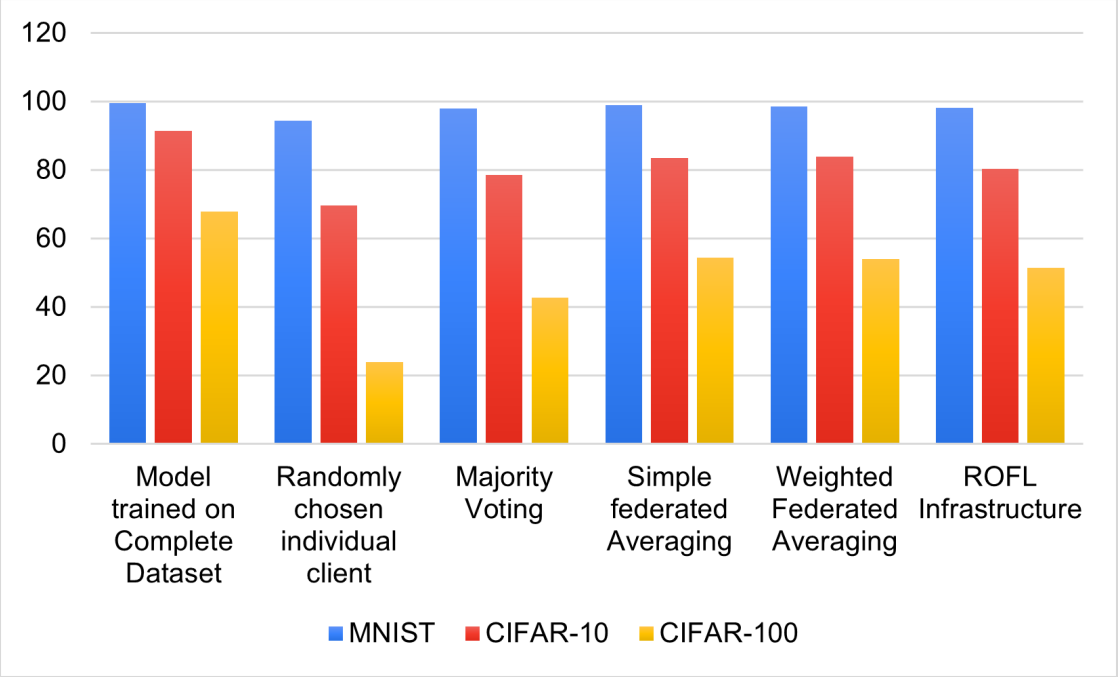


FIGURE 9.4: Study of functionality with neural architecture being ResNet.

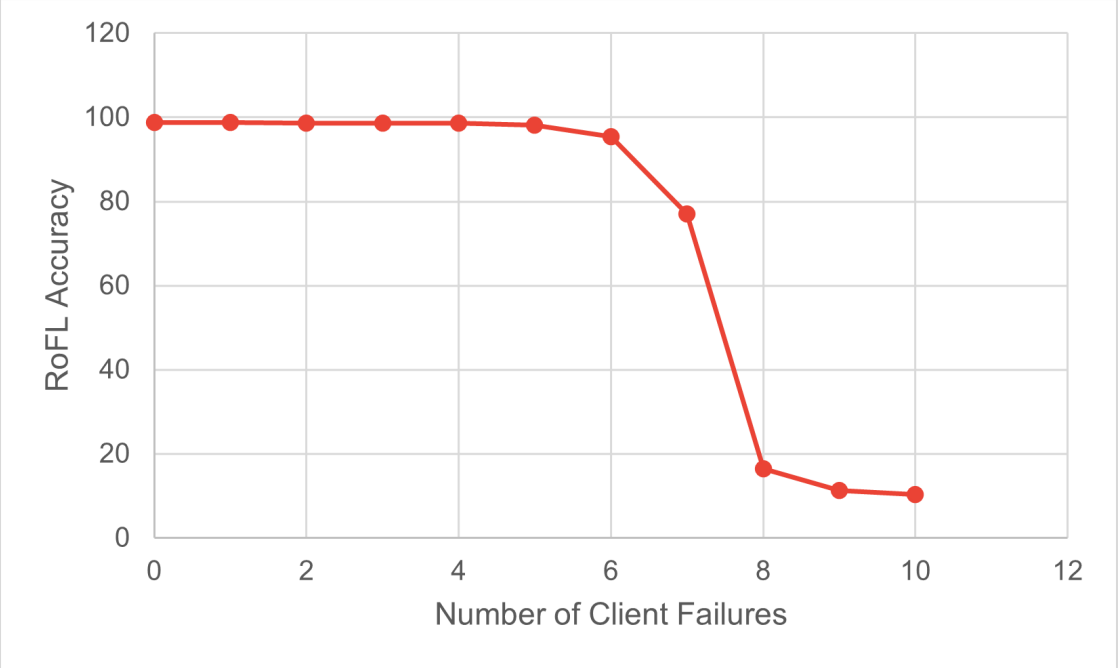


FIGURE 9.5: Study of robustness of ROFL against client failures, with dataset MNIST.

stage onward, the global model needs updating only when a significantly different client update is pushed.

- As evident from Figures 9.6 and 9.7, depending on the neural architecture and

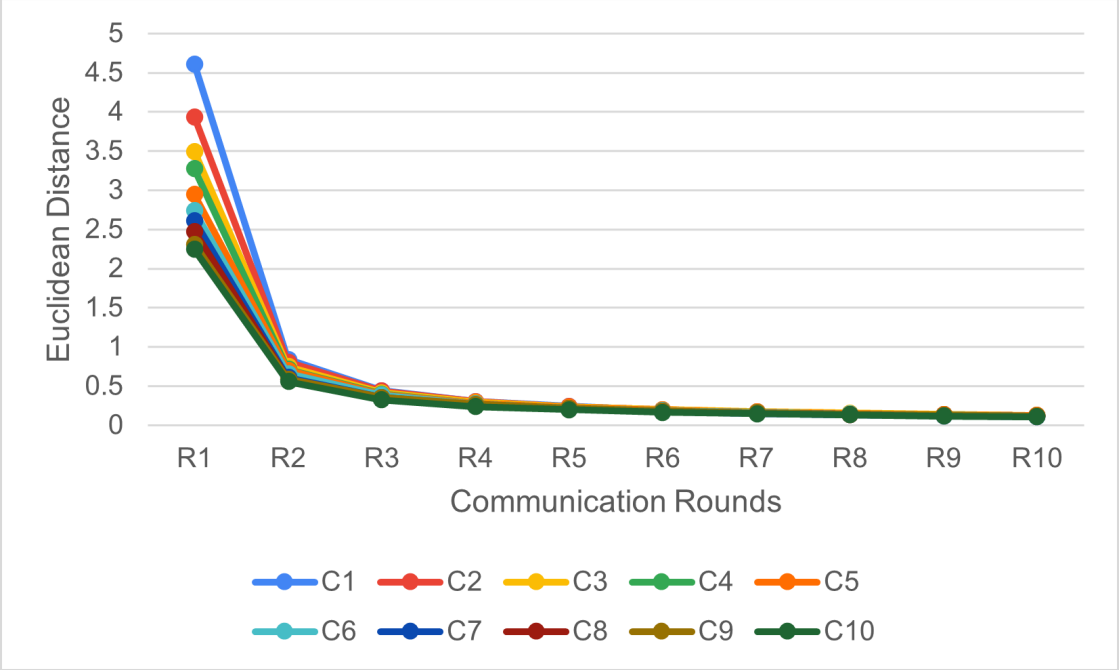


FIGURE 9.6: Study of model convergence with dataset MNIST.

the dataset, the convergence of the models take place at various thresholds. For operating purposes, the threshold can be determined by looking at the elbow curve. Once the performance begins to saturate, it makes sense to incorporate a fresh client update only when it is significantly different with respect to the Euclidean distance.

- The study on the scalability aspect of the infrastructure reveals promising results. ROFL has been built to be highly scalable particularly because of the applications that it finds in domains involving IoT. In Figure 9.8, we see that as the number of clients grow higher, the more useful ROFL becomes. With many clients and limited data to each client, the local models residing with them fails to perform well due to lack of sufficient training. However, the weighted federated averaging mechanism, which is at the heart of ROFL, along with the complete framework keep performing well with high accuracies even when there are many clients. This significantly enhances the utility of the ROFL infrastructure.

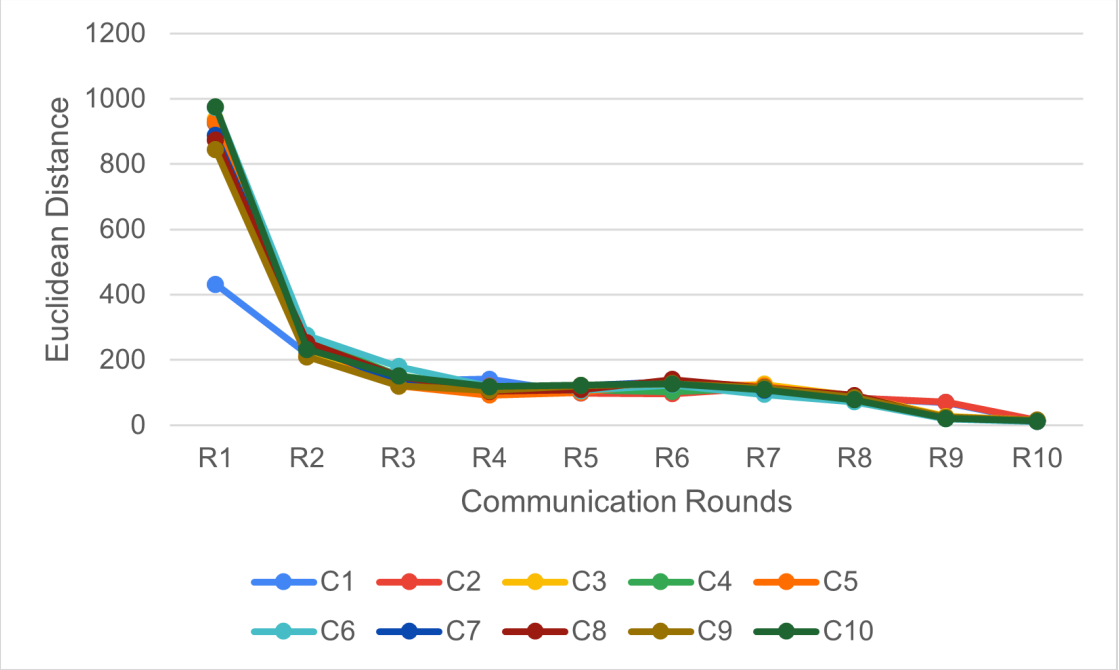


FIGURE 9.7: Study of model convergence with dataset CIFAR-100.

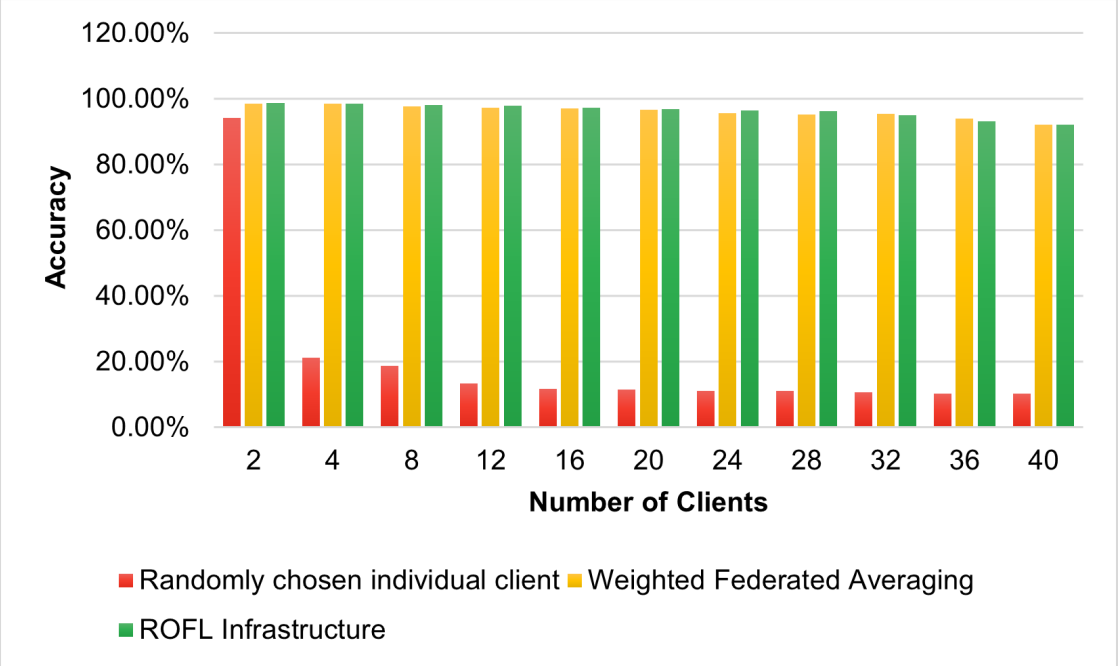


FIGURE 9.8: Study of scalability with the ROFL infrastructure.

9.7 Concluding Remarks

With growing scales of distributed data being generated continuously across various platforms like mobile devices and IoT, machine learning algorithms have the

potential to peak in performances. This is hindered by the issue of lack of ways and means to securely share data and insights. Also, there is a growing concern about privacy rights of the contributors of the data, globally. These problems, coupled with the severe lack of trust among those involved in harvesting the data serves as the primary motivation of our work.

In this work, we propose the ROFL infrastructure, which delivers a two dimensional bi-directional privacy preserving learning framework for distributed clients. Each client owns a local dataset which is updated over time. The clients train a neural architecture with their own data using differentially private SGD and push updates to the aggregator. The aggregator judges the pushed updates using its own locally stored dataset and assigns weights to them accordingly. This is part of the weighted federated averaging mechanism used by ROFL. The final model is thereafter pulled by the clients, which also have the provision of quality verification. Additionally, there is a separate check for model convergence, which incorporates only significantly different pushed updates into the weighted federated averaging once the models sent by the clients saturate.

This works finds immense applications in the entire area of edge based computing environments like mobile device computing and IoT. The guarantees of privacy ensure that the setup can be trusted and used to harvest the distributed data. The fact that ROFL scales very well is particularly important for industrial adoption.

In future, we plan to test out the ROFL infrastructure on real industrial data with a high number of clients and study its performance beyond the academic benchmarks. The study on practical use-cases will serve as a strong justification for acceptance/adoption of the ROFL infrastructure in daily applications.

Chapter 10

Conclusion

Reliability is key towards a wider adoption of machine learning algorithms in driving regular tasks. There needs to be guarantees on the success of ML driven decision-making systems, without errors. It is often seen that an otherwise typically high-performance neural network trained for a specific task, fails under certain circumstances. These vulnerabilities are a key deterrent towards reliability and must be addressed before ubiquitous adoption of AI. The vulnerabilities of ML algorithms can be categorized in different ways. One well-accepted way of doing the classification is based on the type of error that creates the vulnerability. Malfunctions or errors could be intentional or unintentional and malicious and benign. Those that pertain to unintentionally malicious ones are studied as part of Resilience of ML, involving safe ML and time criticality aspects etc. The study of robustness in machine learning algorithms involve dealing with ML vulnerabilities where the errors or malfunctions are both intentional and malicious, therefore being associated with a specific attack model.

The focus of this thesis has been to study and thoroughly investigate the idea of robustness of machine learning algorithms, from the perspective of security and privacy. Within each domain, we have looked at both components of a functional machine learning setup, that includes the model like a neural architecture and the corresponding training data as well.

The primary vulnerability of machine learning models with respect to security is adversarial attacks. In the first part, we consider security for models, and study the adversarial attacks through the lens of dimensionality. We assert that the

high dimensional landscape in which the neural network models optimize facilitate the generation of adversarial examples and that dimensionality reduction enhances adversarial robustness. We have explored the mathematical background for this proposition, studying the properties of data distributions in the high dimensional spaces and nature of such trained manifolds. The idea has been empirically justified thereafter [3]. We have extended this notion of the influence of dimensionality on adversarial sample generation from images to videos [5] and text [4] and provided practical and efficient solutions by leveraging adversarial samples detection and dimensionality reduction. It is important to note here that reducing the dimensionality has an additional computational cost and can in some cases also have an adverse effect on the fundamental machine learning task itself. We therefore optimise the dimension reduction operation for each of the tasks and use-cases and carefully choose the amount of variability to preserve that effectively eliminates adversarial noise but retains the meaningful information necessary for classification or object detection etc. Additionally, in one of the works, we make use of a parallel channel to run the classifications task and adversarial sample detection and apply dimensionality reduction to only those samples that are detected as adversarial [3]. This significantly improves efficiency of the overall system and proves to be beneficial in terms of accuracy as well.

Thereafter, we study the security flaw of adversarial attacks from the perspective of vulnerable features within the data. We have analysed spatially correlated patterns within adversarial images [6]. Class wise, we have split images into two key parts, Region of Importance (RoI) and Region of Attack (RoA), such that the RoI is the region that the classifier is particularly sensitive to, during the classification task, and the RoA is the region which the adversarial attack modifies. The goal of this exercise was to figure out areas within the images that do not contribute to the task of classification but are adversarially vulnerable. Our proposed adversarial defence mechanism out of this work is to neutralize that region, therefore bringing down adversarial vulnerability without compromising on classification accuracy. The idea is demonstrated through benchmarking datasets and models.

Moving on to the aspect of privacy in the second part, we look at some very different problems. First, we look at privacy for the models, followed by that of the data. In the context of preserving privacy of the trained neural network models, we direct our attention to protecting ownership and IP rights of the models, using

watermarking. We review the state-of-the-art in watermarking schemes for neural networks and select the most appropriate ones to study. Watermarking using backdooring is the scheme of choice here. We investigate the vulnerabilities of this scheme and break it using synthesis. In our proposition titled Re-Markable [7], we make the assumption that the adversary has very limited compute power and access to samples from the data distribution relevant to the task. We train a GAN (Generative Adversarial Network) to synthesize more samples and use the synthesized samples to re-train just the fully connected layers of the watermarked models. As demonstrated, it turns out that this minimal computation is sufficient to eliminate the presence of embedded watermarks from the model, and this vulnerability makes the existing scheme extremely unreliable. To solve the problem thus discovered, we worked on a robust watermarking scheme that overcomes this vulnerability. ROWBACK, or robust watermarking for neural networks using backdooring [8], uses a redesigned mechanism for generating the Trigger Set (using adversarial examples with explicit labelling) that is used as the private key for the watermarking, and a method of explicitly marking every layer of the neural network with the embedded watermarks. The goal is to ensure that an adversary interested in extracting the network would need to re-train every layer of the model, which is as good as training a fresh model from scratch as it would require extensive training samples and compute power. We also extended the idea of robust watermarking for models in the domain of natural language processing, particularly text classifiers. TextBack [8], designed to embed watermarks within text classifiers using backdooring, uses a marking scheme that involves the Trigger samples and clean samples together, unlike that in images, as this property is observed only in sequential models like recurrent neural networks and LSTM based models.

Finally, to cover the aspect of privacy of data, we focus on collaborative ML. The motivation in this work is to protect the privacy of the data that is used for training, as many practical applications necessitate the usage of highly sensitive data. This data is decentralised, resides with different non-collocated entities, and is not sharable on privacy grounds. The straight-forward solution to this problem is available in the literature in the form of Federated Learning. Additionally, an adversary may tap into the federated learning infrastructure in multiple ways and extract information. For example, Membership Inference attacks are possible on models deployed in the cloud (which is natural in many federated learning setups),

wherein the model could be queried multiple times and based on the output probabilities returned by the model, one can train another separate model to know if the particular input that had been sent as the query belonged to the training set or not. Carrying out this process multiple times can in theory lead to the reverse-engineering of the entire training set. This is a serious privacy violation. We attempted to solve this problem using Differential Privacy, where a differentially private learning algorithm is used by the participants of the federated learning infrastructure for their local training, involving gradient trimming and addition of sampled noise. We studied practical applications of such collaborative learning systems and deployed the framework on edge devices, by creating light-weight versions of the models that do not compromise on accuracy [9]. Similarly, in another use-case, the participants of the federated learning setup itself could have malicious intentions and can come up with sabotaging attacks on the learning framework. Considering such potential single points of failures of the overall system, we proposed a robust federated learning infrastructure, that assigns coefficients to the updates sent by the clients to the server. This takes care of tolerating faults in up to 50% of the clients' failures [9].

Overall, the ideas discussed in this thesis are a major step towards making machine learning systems more robust and is therefore a necessary step in the direction of reliable AI. All of these propositions combined together provides a platform for holistic robustness of machine learning algorithms in various tasks and use-cases, against many different kinds of attacks that may be relevant for each. The algorithms and mechanisms described in detail in the thesis are tested extensively on benchmarking models and datasets for ease of comparison to the literature and replicability. The natural progression is to take this forward into more real and practical datasets, preferably in collaboration with industrial partners, for a deeper study in real world usability which is necessary for adoption of them in practice.

There are three major areas of research that have been initiated as a direct consequence of the work described in this thesis. The first is regarding the study of adversarial attacks on more complex systems involving different complex combination of machine learning applications. A good example of this is in surveillance systems which use sequential object detection and classification and other related operations. Similar use-cases are also applicable in the field of autonomous driving and many other safety-critical applications. The second area of research that has

immense potential is regarding watermarking neural networks, particularly more explainable and stronger watermarks which have fundamental theoretical guarantees. Also, there are new areas developing in the space of watermarking, which address the possible of incremental and continuous training on watermarked neural networks, wherein the embedded watermarks are robust and unaffected by the training process. The third area of interest is in the domain of federated learning. The kind of federated learning infrastructure discussed in this thesis concerns a fundamentally client-server architecture, which need not always be realistic. The problem of peer-to-peer federated learning is now being investigated and its corresponding vulnerabilities and security and privacy issues are being studied.

In general, there is always a scope of extension of the ideas discussed in this thesis in terms of scalability. We have provided an analysis of scalability wherever applicable, but at an academic level and one needs to test it in real practical problems of scale for reliability. Such an endeavour will only refine the algorithms and hopefully contribute towards the greater goal of instilling public trust in the decision-making paradigm of deep learning, making the future more exciting.

List of Author's Awards, Patents, and Publications[1]

Awards

- **Best Student Paper Award**, "ROFL: RObust privacy preserving Federated Learning," *EAI, IEEE ICDCS 2022*.

Patents

- **Nandish Chattopadhyay and Anupam Chattopadhyay**, "ROWBACK: Robust Watermarking of Neural Networks using Backdooring," *Singapore Patent (Application No. 10202107447S), and the US patent has been published as: US 2023/0012871 A1.*.

Conference Proceedings

- **Nandish Chattopadhyay**, Anupam Chattopadhyay, Sourav Sen Gupta and Michael Kasper, "Curse of Dimensionality in Adversarial Examples," in *IEEE IJCNN 2019* `https://ieeexplore.ieee.org/document/8851795`
- **Nandish Chattopadhyay**, Anupam Chattopadhyay and Ritabrata Maiti, "Deploy-able Privacy Preserving Collaborative ML," in *IEEE ICDCS 2020* `https://ieeexplore.ieee.org/document/9355628`

[1]The superscript * indicates joint first authors

- **Nandish Chattopadhyay**, Chua Sheng Yang Viroy and Anupam Chattopadhyay, "Re-Markable: Stealing Watermarked Neural Networks through Synthesis," in *SPACE 2020* https://link.springer.com/chapter/10.1007/978-3-030-66626-2_3

- **Nandish Chattopadhyay**, Anupam Chattopadhyay, "ROWBACK: Robust Watermarking of Neural Networks using Backdooring," in *IEEE ICMLA 2021* https://ieeexplore.ieee.org/document/9680232

- **Nandish Chattopadhyay**, Subhrojyoti Chatterjee and Anupam Chattopadhyay, "Robustness Against Adversarial Attacks using Dimensionality," in *SPACE 2021* https://link.springer.com/chapter/10.1007/978-3-030-95085-9_12

- **Nandish Chattopadhyay**, Arpit Singh and Anupam Chattopadhyay, "ROFL: RObust privacy preserving Federated Learning," in *IEEE ICDCS 2022*

- **Nandish Chattopadhyay**, Lionell Yip En Zhi, Bryan Tan Bing Xing and Anupam Chattopadhyay, "Analyzing spatially correlated patterns in adversarial images," in *PrePrint available arXiv* https://arxiv.org/abs/2011.10794

- **Nandish Chattopadhyay**, Shivam Garg and Anupam Chattopadhyay , "Adversarial robustness in Object Detectors using dimensionality reduction," in *IEEE TrustCom 2022*

- **Nandish Chattopadhyay**, Rajan Kataria and Anupam Chattopadhyay, "TextBack: Watermarking Text Classifiers using Backdooring," in *EuroMicro DSD 2022*

- **Nandish Chattopadhyay**, Atreya Goswami and Anupam Chattopadhyay , "Adversarial Attacks and Dimensionality in Text Classifiers," in *Manuscript submitted for review*

Bibliography

[1] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. xxiii, xxvi, 8, 12, 41, 51, 54, 107, 134, 173, 188, 189

[2] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016. xxiii, 8, 20, 41, 51, 54, 55, 93, 107, 151

[3] Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pages 1615–1631, 2018. xxiv, xxvi, 22, 121, 126, 134, 144, 145, 147, 154, 155, 157

[4] Nandish Chattopadhyay, Chua Sheng Yang Viroy, and Anupam Chattopadhyay. Re-markable: Stealing watermarked neural networks through synthesis. In *International Conference on Security, Privacy, and Applied Cryptography Engineering*, pages 46–65. Springer, 2020. xxiv, xxvi, 10, 24, 142, 146, 156, 157

[5] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2, 2010. xxvi, 8, 12, 41, 51, 54, 107, 135, 142, 154, 173, 188, 189, 190, 191

[6] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 2014. xxvi, 8, 12, 41, 51, 54, 107, 135, 142, 154, 173, 188, 189, 190, 192

[7] Sasha Targ, Diogo Almeida, and Kevin Lyman. Resnet in resnet: Generalizing residual architectures. *arXiv preprint arXiv:1603.08029*, 2016. xxvi, 12, 134, 150, 154, 173, 188, 189, 190, 191, 192

[8] Michael I Jordan and Tom M Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015. 1

[9] A.M. Turing. Computing machinery and intelligence. *Mind*, pages 433–460, 1950. 1

[10] Stuart J Russell. *Artificial intelligence a modern approach*. Pearson Education, Inc., 2010. 1

[11] Saul McLeod. Maslow's hierarchy of needs. *Simply psychology*, 1(1-18), 2007. 3

[12] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*, 2017. 8, 51, 54, 55

[13] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017. 8, 20, 51, 54, 55

[14] Thanh-Ha Le, Cécile Canovas, and Jessy Clédiere. An overview of side channel analysis attacks. In *Proceedings of the 2008 ACM symposium on Information, computer and communications security*, pages 33–43, 2008. 13

[15] Lejla Batina, Shivam Bhasin, Dirmanto Jap, and Stjepan Picek. {CSI}{NN}: Reverse engineering of neural network architectures through electromagnetic side channel. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 515–532, 2019. 13

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 17, 31

[17] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. Ieee, 2009. 17

[18] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958. 18, 31

[19] Natalie Stephenson, Emily Shane, Jessica Chase, Jason Rowland, David Ries, Nicola Justice, Jie Zhang, Leong Chan, and Renzhi Cao. Survey of machine learning techniques in drug discovery. *Current drug metabolism*, 20(3):185–193, 2019. 18

[20] Sina Mohseni, Niloofar Zarei, and Eric D Ragan. A survey of evaluation methods and measures for interpretable machine learning. *arXiv preprint arXiv:1811.11839*, 2018. 18

[21] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. An introduction to statistical learning. 2009. 18

[22] Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao. Deep learning–based text classification: a comprehensive review. *ACM Computing Surveys (CSUR)*, 54(3):1–40, 2021. 18

[23] Jonathan J Webster and Chunyu Kit. Tokenization as the initial phase in nlp. In *COLING 1992 Volume 4: The 14th International Conference on Computational Linguistics*, 1992. 18

[24] Subbu Kannan, Vairaprakash Gurusamy, S Vijayarani, J Ilamathi, M Nithya, S Kannan, and V Gurusamy. Preprocessing techniques for text mining. *International Journal of Computer Science & Communication Networks*, 5(1): 7–16, 2014. 18

[25] Xin Rong. word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*, 2014. 19

[26] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014. 19

[27] Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923*, 2017. 19, 64

[28] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013. 20, 61, 81, 93, 147

[29] Ian Goodfellow, Patrick McDaniel, and Nicolas Papernot. Making machine learning robust against adversarial inputs. *Communications of the ACM*, 61 (7):56–66, 2018. 20, 61, 93, 147

[30] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016. 20, 61, 93

[31] Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. Adversarial attacks and defences: A survey. *CoRR*, abs/1810.00069, 2018. URL `http://arxiv.org/abs/1810.00069`. 20, 61, 62, 93, 147

[32] IJ Goodfellow, J Shlens, and C Szegedy. Explaining and harnessing adversarial examples (2014). arxiv preprint. *arXiv preprint arXiv:1412.6572*. 20, 61, 93

[33] Simant Dube. High dimensional spaces, deep learning and adversarial examples. *arXiv preprint arXiv:1801.00634*, 2018. 20, 93

[34] Justin Gilmer, Luke Metz, Fartash Faghri, Samuel S Schoenholz, Maithra Raghu, Martin Wattenberg, and Ian Goodfellow. Adversarial spheres. *arXiv preprint arXiv:1801.02774*, 2018. 20, 32, 62

[35] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Xiaolin Hu, Jianguo Li, and Jun Zhu. Boosting adversarial attacks with momentum. *arXiv preprint arXiv:1710.06081*, 2017. 20, 42

[36] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE, 2017. 20

[37] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *Artificial intelligence safety and security*, pages 99–112. Chapman and Hall/CRC, 2018. 20

[38] Nicolas Papernot and Patrick McDaniel. On the effectiveness of defensive distillation. *arXiv preprint arXiv:1607.05113*, 2016. 20

[39] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. Hotflip: White-box adversarial examples for text classification. *arXiv preprint arXiv:1712.06751*, 2017. 21

[40] Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 50–56. IEEE, 2018. 21

[41] Jiwei Li, Will Monroe, and Dan Jurafsky. Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220*, 2016. 21

[42] Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi. Deep text classification can be fooled. *arXiv preprint arXiv:1704.08006*, 2017. 21

[43] Shi Feng, Eric Wallace, Mohit Iyyer, Pedro Rodriguez, Alvin Grissom II, and Jordan L Boyd-Graber. Right answer for the wrong reason: Discovery and mitigation. 2018. 21

[44] Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. Generating natural language adversarial examples. *arXiv preprint arXiv:1804.07998*, 2018. 21

[45] Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 1085–1097, 2019. 21

[46] Siddhant Garg and Goutham Ramakrishnan. Bae: Bert-based adversarial examples for text classification. *arXiv preprint arXiv:2004.01970*, 2020. 21

[47] Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. Textbugger: Generating adversarial text against real-world applications. *arXiv preprint arXiv:1812.05271*, 2018. 21

[48] Huili Chen, Bita Darvish Rouhani, and Farinaz Koushanfar. Blackmarks: Blackbox multibit watermarking for deep neural networks. *arXiv preprint arXiv:1904.00344*, 2019. 22

[49] Huili Chen, Bita Darvish Rohani, and Farinaz Koushanfar. Deepmarks: a digital fingerprinting framework for deep neural networks. *arXiv preprint arXiv:1804.03648*, 2018. 22, 142

[50] Bita Darvish Rouhani, Huili Chen, and Farinaz Koushanfar. Deepsigns: A generic watermarking framework for protecting the ownership of deep learning models. 22

[51] Sebastian Szyller, Buse Gul Atli, Samuel Marchal, and N Asokan. Dawn: Dynamic adversarial watermarking of neural networks. *arXiv preprint arXiv:1906.00830*, 2019. 22, 142

[52] Huili Chen, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. Deepinspect: A black-box trojan detection and mitigation framework for deep neural networks. In *IJCAI*, pages 4658–4664, 2019. 22, 24, 142

[53] Wenbo Guo, Lun Wang, Xinyu Xing, Min Du, and Dawn Song. Tabor: A highly accurate approach to inspecting and restoring trojan backdoors in ai systems. *arXiv preprint arXiv:1908.01763*, 2019. 22

[54] Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. Embedding watermarks into deep neural networks. In *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*, pages 269–277, 2017. 22, 142

[55] Erwan Le Merrer, Patrick Perez, and Gilles Trédan. Adversarial frontier stitching for remote neural network watermarking. *Neural Computing and Applications*, 32(13):9233–9244, 2020. 22

[56] Ryota Namba and Jun Sakuma. Robust watermarking of neural network with exponential weighting. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, pages 228–240, 2019.

[57] Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph Stoecklin, Heqing Huang, and Ian Molloy. Protecting intellectual property of deep neural networks with watermarking. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pages 159–172, 2018. 22

[58] Dorjan Hitaj, Briland Hitaj, and Luigi V Mancini. Evasion attacks against watermarking techniques found in mlaas systems. In *2019 Sixth International Conference on Software Defined Systems (SDS)*, pages 55–63. IEEE, 2019. 24, 130, 156

[59] Dorjan Hitaj, Briland Hitaj, and Luigi V Mancini. Evasion attacks against watermarking techniques found in mlaas systems. In *2019 Sixth International Conference on Software Defined Systems (SDS)*, pages 55–63. IEEE, 2019. 24

[60] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pages 601–618, 2016. 24, 130, 142

[61] Xinyun Chen, Wenxiao Wang, Chris Bender, Yiming Ding, Ruoxi Jia, Bo Li, and Dawn Song. Refit: a unified watermark removal framework for deep learning systems with limited data. *arXiv preprint arXiv:1911.07205*, 2019.

[62] Shangwei Guo, Tianwei Zhang, Han Qiu, Yi Zeng, Tao Xiang, and Yang Liu. The hidden vulnerability of watermarking for deep neural networks. *arXiv preprint arXiv:2009.08697*, 2020. 24, 142, 169

[63] H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, et al. Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*, 2016. 24, 176, 177

[64] Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1310–1321, 2015. 24, 25, 174

[65] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *arXiv preprint arXiv:1908.07873*, 2019. 25, 174

[66] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1322–1333, 2015. 25

[67] Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016. 25, 172

[68] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konecny, Stefano Mazzocchi, H Brendan McMahan, et al. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046*, 2019. 25, 174

[69] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H Yang, Farhad Farokhi, Shi Jin, Tony QS Quek, and H Vincent Poor. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security*, 15:3454–3469, 2020. 25

[70] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, NY, USA:, 2001. 31, 119

[71] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016. 33, 95

[72] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995. 33, 93

[73] Christopher M Bishop et al. *Neural networks for pattern recognition*. Oxford university press, 1995. 33, 65

[74] John Hopcroft and Ravi Kannan. Foundations of data science. 2014. 37, 67

[75] Charu C Aggarwal, Alexander Hinneburg, and Daniel A Keim. On the surprising behavior of distance metrics in high dimensional space. In *International conference on database theory*, pages 420–434. Springer, 2001. 38, 70

[76] D Freedman, R Pisani, and R Purves. Statistics. 2007. *ISBN: 0-393970-833*, 1978. 39

[77] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is "nearest neighbor" meaningful? In *International conference on database theory*, pages 217–235. Springer, 1999. 39

[78] Nicolas Papernot, Nicholas Carlini, Ian Goodfellow, Reuben Feinman, Fartash Faghri, Alexander Matyasko, Karen Hambardzumyan, Yi-Lin Juang, Alexey Kurakin, Ryan Sheatsley, et al. cleverhans v2. 0.0: an adversarial machine learning library. *arXiv preprint arXiv:1610.00768*, 2016. 43, 52

[79] Gene H Golub and Christian Reinsch. Singular value decomposition and least squares solutions. In *Linear algebra*, pages 134–151. Springer, 1971. 52

[80] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008. 52

[81] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037, 2019. 52, 135, 154

[82] Nandish Chattopadhyay, Anupam Chattopadhyay, Sourav Sen Gupta, and Michael Kasper. Curse of dimensionality in adversarial examples. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019. 67, 68, 93, 96

[83] Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8018–8025, 2020. 71, 72

[84] John X Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. *arXiv preprint arXiv:2005.05909*, 2020. 71

[85] Jeff Barnes. Azure machine learning. *Microsoft Azure Essentials. 1st ed, Microsoft*, 2015. 71

[86] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE access*, 8:58443–58469, 2020. 79

[87] Igor V Gritsuk, Vladimir Volkov, Vasyl Mateichyk, Yuriy Grytsuk, Yuliya Nikitchenko, Dmytro Klets, Miroslaw Smieszek, Yuri Volkov, Roman Symonenko, and Andrii Grytsuk. Information model of v2i system of the vehicle technical condition remote monitoring and control in operation conditions. Technical report, SAE Technical Paper, 2018. 79

[88] Mrinal R Bachute and Javed M Subhedar. Autonomous driving architectures: insights of machine learning and deep learning algorithms. *Machine Learning with Applications*, 6:100164, 2021. 80

[89] Piergiuseppe Mallozzi, Patrizio Pelliccione, Alessia Knauss, Christian Berger, and Nassar Mohammadiha. Autonomous vehicles: state of the art, future trends, and challenges. *Automotive Systems and Software Engineering*, pages 347–367, 2019. 80

[90] Wilko Schwarting, Javier Alonso-Mora, and Daniela Rus. Planning and decision-making for autonomous vehicles. *Annual Review of Control, Robotics, and Autonomous Systems*, 1(1):187–210, 2018. 80

[91] Martin Rabe, Stefan Milz, and Patrick Mader. Development methodologies for safety critical machine learning applications in the automotive domain: A survey. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 129–141, 2021. 80

[92] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28:91–99, 2015. 82, 83

[93] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 82, 83

[94] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 82

[95] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 82

[96] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014. 82

[97] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 83

[98] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017. 83

[99] Shang-Tse Chen, Cory Cornelius, Jason Martin, and Duen Horng Polo Chau. Shapeshifter: Robust physical adversarial attack on faster r-cnn object detector. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 52–68. Springer, 2018. 84

[100] Aamir Mustafa, Salman H Khan, Munawar Hayat, Roland Goecke, Jianbing Shen, and Ling Shao. Deeply supervised discriminative learning for adversarial defense. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. 94

[101] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. In *Advances in Neural Information Processing Systems*, pages 125–136, 2019. 97

[102] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017. 107, 135

[103] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3498–3505. IEEE, 2012. 107

[104] Yuhang Dong, Zhuocheng Jiang, Hongda Shen, W David Pan, Lance A Williams, Vishnu VB Reddy, William H Benjamin, and Allen W Bryan. Evaluations of deep convolutional neural networks for automatic identification of malaria infected cells. In *2017 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI)*, pages 101–104. IEEE, 2017. 107

[105] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009. 120

[106] Gang Qu and Miodrag Potkonjak. *Intellectual property protection in VLSI designs: theory and practice*. Springer Science & Business Media, 2007. 121

[107] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. 128, 135

[108] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014. 129

[109] Lisa Torrey and Jude Shavlik. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI global, 2010. 131, 152

[110] Christian Rey and Jean-Luc Dugelay. A survey of watermarking algorithms for image authentication. *EURASIP Journal on Advances in Signal Processing*, 2002(6):1–9, 2002. 143

[111] Nandish Chattopadhyay, Ritabrata Maiti, and Anupam Chattopadhyay. Deploy-able privacy preserving collaborative ml. In *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*, pages 1397–1402. IEEE, 2020. 172

[112] Jiale Zhang, Junjun Chen, Di Wu, Bing Chen, and Shui Yu. Poisoning attack in federated learning using generative adversarial nets. In *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pages 374–380. IEEE, 2019. 172

[113] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318, 2016. 176, 178, 179, 180

[114] Justin Brickell and Vitaly Shmatikov. The cost of privacy: destruction of data-mining utility in anonymized data publishing. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 70–78, 2008. 177

[115] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[116] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 486–503. Springer, 2006. 177

[117] Ekaba Bisong. Google colaboratory. In *Building Machine Learning and Deep Learning Models on Google Cloud Platform*, pages 59–64. Springer, 2019. 186