NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

ROBUST MACHINE LEARNING USING SUPERQUANTILES

by

Dongyu Yang

September 2021

Thesis Advisor:	Johannes O. Royset
Co-Advisor:	Krishnakumar Balasubramanian, University of California, Davis
Second Reader:	Robert L. Bassett

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE September 2021	3. REPORT TYPE AND DATES COVERED Master's thesis

4. TITLE AND SUBTITLE ROBUST MACHINE LEARNING USING SUPERQUANTILES	5. FUNDING NUMBERS
6. AUTHOR(S) Dongyu Yang	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	12b. DISTRIBUTION CODE A

13. ABSTRACT (maximum 200 words)

The proliferation of machine learning in image recognition and natural language processing applications comes with increasing risk of adversarial attacks. Such attacks can potentially spoof automated detection systems in our drones or defeat facial recognition systems and bypass automated security systems. Typical defense techniques involve long training times, which would not be viable in an operational setting. The thesis utilizes a novel superquantile-based formulation to train machine learning systems to make them more robust to noise and adversarial attacks, while incurring less training costs compared to typical adversarial training techniques. The concept is explored in the context of support vector machines and achieves similar results as in the case of L1-regularization models. Subsequently, the concept is developed for neural network training with robustness tests on commonly referenced Modified National Institute of Standards and Technology (MNIST) and Canadian Institute for Advanced Research–10 classes (CIFAR-10) datasets. The test results demonstrate robustness against random noise perturbations and benchmark against typical adversarial training shows comparable results. This initial excursion into superquantile training sets the foundation for further exploration into improving machine learning robustness within less computation time.

14. SUBJECT TERMS neural networks, machine learning, image processing, artificial intelligence, computer vision, nonlinear optimization, inverse problems, adversarial learning	15. NUMBER OF PAGES 65
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

ROBUST MACHINE LEARNING USING SUPERQUANTILES

Dongyu Yang
Civilian, DSTA
MEng, Imperial College London, 2012

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
September 2021

Approved by: Johannes O. Royset
Advisor

Krishnakumar Balasubramanian
Co-Advisor

Robert L. Bassett
Second Reader

W. Matthew Carlyle
Chair, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The proliferation of machine learning in image recognition and natural language processing applications comes with increasing risk of adversarial attacks. Such attacks can potentially spoof automated detection systems in our drones or defeat facial recognition systems and bypass automated security systems. Typical defense techniques involve long training times, which would not be viable in an operational setting. The thesis utilizes a novel superquantile-based formulation to train machine learning systems to make them more robust to noise and adversarial attacks, while incurring less training costs compared to typical adversarial training techniques. The concept is explored in the context of support vector machines and achieves similar results as in the case of L1-regularization models. Subsequently, the concept is developed for neural network training with robustness tests on commonly referenced Modified National Institute of Standards and Technology (MNIST) and Canadian Institute for Advanced Research–10 classes (CIFAR-10) datasets. The test results demonstrate robustness against random noise perturbations and benchmark against typical adversarial training shows comparable results. This initial excursion into superquantile training sets the foundation for further exploration into improving machine learning robustness within less computation time.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

1 Introduction **1**

1.1 Background 3

1.2 Study Objective 8

1.3 Thesis Organization 9

2 Formulation and Methodology **11**

2.1 Superquantile Approximations 12

2.2 Experimentation Approach 15

3 SVM Robustness **17**

3.1 SVM Formulation and Algorithm 17

3.2 Experiment Setup 20

3.3 L_1-Regularization Results 22

3.4 Superquantile Results 23

4 Neural Network Robustness **29**

4.1 Neural Network Training Algorithm 29

4.2 Experiment Setup 30

4.3 MNIST Experimentation 32

4.4 CIFAR-10 Experimentation 36

5 Conclusions **41**

5.1 Challenges and Limitations 42

5.2 Recommendations for Future Work 43

List of References **45**

Initial Distribution List **49**

THIS PAGE INTENTIONALLY LEFT BLANK

List of Figures

Figure 1.1 Images of aircraft with pink box overlays to indicate positive detection by detection algorithm. Source: Adhikari et al. (2020). 2

Figure 1.2 Impersonators wearing specially printed glasses at the top row with their corresponding mismatched persona by the facial recognition algorithm at the bottom row. Source: Sharif et al. (2016). 2

Figure 1.3 Simple neural network example using 5 layers to process input data x_1, x_2, x_3 into output signal y_1, y_2. 4

Figure 2.1 Probability density function for a continuous random variable v with the quantile Q_α marked out and superquantile $\bar{Q}_\alpha$ as the mean of v in the upper α-quantile. 13

Figure 3.1 Model accuracy results with different L_1-regularization values and uniform noise perturbations 22

Figure 3.2 Model accuracy results with different L_1-regularization values and Gaussian noise perturbations. 23

Figure 3.3 Superquantile model accuracy results with different training noise δ and uniform noise perturbation. 24

Figure 3.4 Superquantile model accuracy results with different training noise δ and Gaussian noise perturbation. 24

Figure 3.5 Minimal effect on the superquantile model accuracy with different α parameter at $\delta = 0.2$; similar robustness compared to L_1-regularization with optimal $\rho = 0.02$. 25

Figure 3.6 Some differentiation on model accuracy with attack size 10 and different α parameter at $\delta = 0.2$; higher robustness compared to L_1-regularization near $\varepsilon = 0.2$. 26

Figure 3.7 Greater differentiation on model accuracy with uniform distribution whilst keeping attack size 10 and different α parameter at $\delta = 0.2$; higher robustness than L_1-regularization only at certain conditions. 27

Figure 3.8 Increasing training perturbation to $\delta = 0.4$ at attack size 10 is able to produce greater robustness for certain α, but not in the lower ε range for high $\alpha = 0.99$. 28

Figure 4.1 Example images from Modified National Institute of Standards and Technology (MNIST) showing the different handwritten digits from 0 to 9. Source: Lecun et al. (1998). 32

Figure 4.2 Model accuracy results on MNIST dataset with training noise $U(-1, 1)$ and different attack size (a) $M = 1$, (b) $M = 20$ with the test dataset subject to uniform noise perturbations $\varepsilon U(-1, 1)$. 33

Figure 4.3 Model accuracy results on MNIST dataset with training noise $N(0, 1)$ and different attack size (a) $M = 1$, (b) $M = 20$ with the test dataset subject to Gaussian noise perturbations $\varepsilon N(0, 1)$. 34

Figure 4.4 Model accuracy results on MNIST dataset with 1-step PGD training and different attack size (a) $M = 1$, (b) $M = 20$ with the test dataset subject to ε amount of PGD perturbations. 35

Figure 4.5 Example images from CIFAR-10 showing the different classes of images corresponding to physical objects. Source: Krizhevsky and Hinton (2009). 36

Figure 4.6 Model accuracy results on CIFAR-10 dataset subject to (a) random noise, and (b) Gaussian noise perturbations on the test data. . . . 38

Figure 4.7 Comparing model performance between superquantile training with different α and a typical adversarial training. 39

List of Acronyms and Abbreviations

AI	artificial intelligence
CIFAR-10	Canadian Institute For Advanced Research - 10 classes
CNN	convolutional neural network
CVaR	conditional value at risk
ERM	empirical risk minimization
FGSM	fast gradient sign method
GPU	graphics processing unit
ISR	intelligence, surveillance, and reconnaissance
MNIST	Modified National Institute of Standards and Technology
PGD	projected gradient descent
SGD	stochastic gradient descent
SVM	support vector machine
WRN	wide residual network

THIS PAGE INTENTIONALLY LEFT BLANK

Executive Summary

Artificial intelligence is transforming the way we work today as it frees us from mundane and repetitive tasks. Despite the host of benefits that artificial intelligence offers, the process of building applications using machine learning has its drawbacks and associated risks. Recent studies have shown that machine learning is prone to adversarial attacks where specially trained noise patterns can be added to deceive the trained image classifier, and even sensor noise can also affect accurate detection. In the military context, this can pose security risks and undermine the performance of unmanned systems that rely on automated detection. The best known defense currently is adversarial training, but this requires a long training time due to its min-max formulation.

Our approach is to take an alternate formulation that focus on the higher tail-end losses in the training data using superquantiles, thereby approximating the worst-case conditions instead of solving for it explicitly. This effectively replaces the need to solve the maximization problem in adversarial training with just the expectation of sample points and predefined perturbations, and thereby faster to compute. The use of superquantiles in machine learning is not new, but previous efforts focus on general model robustness and have not examined it in the context of adversarial training.

Adversarial neural network training is difficult to implement in practice because it not only takes a long computational time, it also has many different hyperparameters to consider for the iterative solver to converge within reasonable accuracy. In this thesis, we take a progressive approach that starts from the more manageable support vector machine training to the more complex neural network training, and build up our choice of training parameters incrementally. Experimentation with support vector machines allows us to better understand the regularization effect and robustness characteristics of superquantile training as we vary the α-superquantile parameter between $[0, 1)$ and different training perturbations. Moving to neural network training, we use common models (such as residual network model) and techniques in other adversarial training studies (such as batch normalization) to build a baseline for robustness benchmarking against typical adversarial training. Commonly used dataset are chosen for experimentation, namely the breast cancer Wisconsin dataset for support vector machines, and the MNIST and CIFAR-10 dataset for neural networks. Since

robustness concerns both random noise and adversarial attack, the models are evaluated against random uniform distribution, random Gaussian distribution and projected gradient descent attacks (commonly regarded as the strongest first-order adversarial attack).

Results show that in the context of support vector machines, superquantile training is able to match the results of L_1-regularization with the appropriate choice of training perturbation and $\alpha = 0.99$. Increasing the attack size to 10 allows superquantile training to achieve 2% to 10% better performance, but this comes at the cost of 13 times more computation time. In the context of neural networks, superquantile training is robust to both random noise perturbations with minimal difference under different choices of α. Under adversarial attack, the choice of $\alpha = 0.9$ brings about significantly better accuracy than $\alpha = 0$. In benchmark tests against typical adversarial training, superquantile training takes 35% less time at the expense of only 10% reduction in accuracy.

CHAPTER 1:
Introduction

Recent developments in machine learning coupled with the exponential growth in computing power have led to breakthroughs in computer vision and natural language processing. Artificial intelligence (AI) and robotic systems with autonomous decision making are continuously being introduced into the military for intelligence, surveillance, and reconnaissance (ISR) and even strike missions, with some predicting that the next war will be a war of robots (Khurshid and Bing-Rong 2004, South 2020). Naturally, people get concerned about the underlying risks and start to demand safety reviews and protection against potential exploits. These concerns were exacerbated by recent demonstrations of *adversarial attacks* in real-world applications, where physical items are being modified to spoof current state-of-the-art automatic detection systems.

Adhikari et al. (2020) demonstrates that automatic object detectors (such as those in drone surveillance) can be misled when specially designed *patches* are overlaid on top of military aircraft images, thereby camouflaging them from automatic detection (shown in Figure 1.1). The patches in their study are images with size, pattern, position, and color specially trained against a provided object detector, such that the detector's classification for the object falls below a certain confidence threshold. The study has set the threshold sufficiently low such that the object is perceived as noise by the detector and not highlighted to the operator.

Sharif et al. (2016) demonstrates that it is possible to spoof automated facial recognition through a specially colored pair of spectacles. The wearer can avoid detection or even impersonate another individual (shown in Figure 1.2). To avoid detection by the facial recognition algorithm, the spectacles in their study are designed such that it can minimize the probability of classification of any single identifiable individual by the facial recognition algorithm, such that it falls below the detection threshold. Alternatively, their study also designed spectacles can be used to impersonate other individuals by introducing carefully selected perturbations such that it crosses the algorithm's decision threshold for that target individual. Such methods could potentially enable undercover agents to avoid surveillance detection or bypass facial recognition security systems, while remaining inconspicuous.

(a) Random patch (b) Adversarial patch

Figure 1.1. Images of aircraft with pink box overlays to indicate positive detection by detection algorithm. (a) Patches of random noise are overlaid on each aircraft image and shown to be generally unsuccessful at defeating the object detection algorithm. (b) Same aircraft but with adversarial patches causing no positive identification, thereby defeating the detection algorithm. Source: Adhikari et al. (2020).

Figure 1.2. Impersonators wearing specially printed glasses at the top row with their corresponding mismatched persona by the facial recognition algorithm at the bottom row. Source: Sharif et al. (2016).

These are examples of adversarial attacks that can threaten the stability and safety of AI systems (Szegedy et al. 2014). The attacks can take various forms and create avenues for many types of deception to corrupting the system operator's situational awareness, leading to surprises and less predictable military engagements. This can lead to situations where the operator face more difficulty with the AI system than without, thereby degrading operator effectiveness and nullifying the advantage brought about by the AI system.

However, adversarial attacks assume that the attacker has prior knowledge of the model and data used, and has the time and resources to prepare and deploy their attacks beforehand. This is difficult to accomplish in practice since the required information are usually tightly

guarded (more so in the military context) and the model will likely undergo constant changes and updates. Therefore, it is also important to evaluate the more likely scenario where an adversary does not have prior knowledge of the data model and only able to use some simple obfuscation methods, such as random noise.

AI systems are also susceptible to common disturbances, such as rain, dust storms, snow, and even small defects on the image sensor. Dodge and Karam (2016) show that state-of-the-art models are susceptible to image compression, particularly to blur and noise. It is arguably more important to address the accuracy degradation from these random events and perturbations since they are more likely to occur. To address this issue, recent works have demonstrated improved robustness by using random noise in the machine learning process. Cohen et al. (2019) suggest augmenting training images with Gaussian noise to give a regularization effect for smoothing the decision boundary and shows that it can be a good theoretical and empirical defense. Rusak et al. (2020) demonstrate Gaussian noise augmented training can generalize well to different forms of perturbations, while regular adversarial training can not. It is therefore important to consider incorporating common random noise perturbations into the machine learning process to harness its regularization properties and gain robustness.

To cater for this wide spectrum of attacks from strong adversarial to weak random noise, we tackle the underlying formulation in the robustness problem and develop a tunable statistical training approach based on superquantile optimization.

1.1 Background

AI systems are under increasing scrutiny in recent years as their vulnerabilities are being discovered, thereby increasing the risk of exploitation and undermining the safety of their users. There are numerous recent studies that seek to better understand these vulnerabilities and propose building certain robustness guarantees into AI systems. Some of these research have advanced our understanding of robustness in neural network training and provide a basic arsenal of tools to improve model robustness. Before diving into these recent studies on neural network robustness, we first describe neural networks and how it is used for image recognition.

1.1.1 Neural Networks for Classification

A neural network is fundamentally a set of nodes placed in different layers with connections drawn between the nodes called edges. The artificial feed-forward neural network is commonly used in image processing applications, as shown in Figure 1.3, where the objective is to pass signals through the network from the input layer to the output layer to reach a certain prediction of the outcome (e.g. 70% likelihood of class A and 30% likelihood of class B). Each of the nodes and edges can carry a weight, while each node also has a predetermined threshold, and the signal can only be passed out from the node to the next layer if the aggregate signal at the node is above the specified threshold. Once all signals reach the output layer, the aggregated strength of the signals will determine the final answer returned by the neural network. To find the appropriate weights for the nodes and edges, training data with known inputs and outputs is used to tune those values until the network is able to correctly answer most of the training problems.

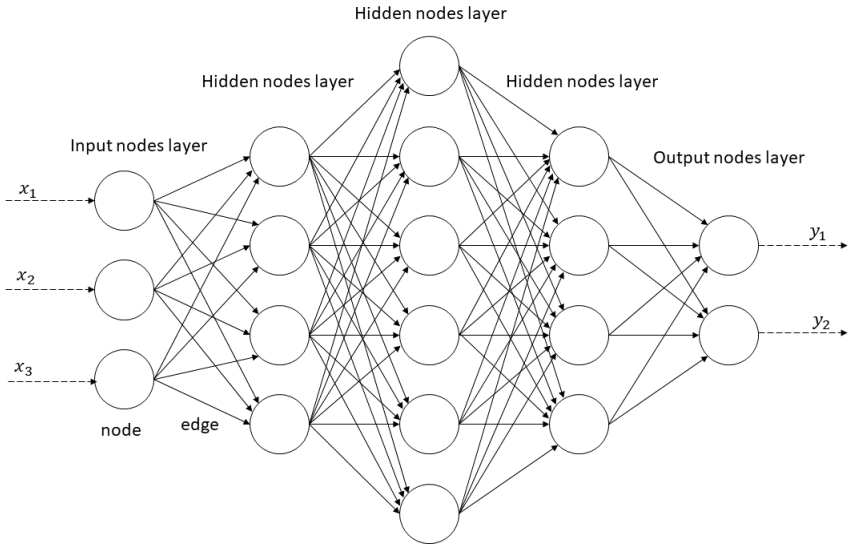

Figure 1.3. Simple neural network example using 5 layers to process input data x_1, x_2, x_3 into output signal y_1, y_2.

The process of training the neural network, or learning, can be carried out by the empirical risk minimization (ERM) formulation first proposed in Vapnik (1992) which takes a statistical approach to solve the learning problem. The learning process consists of 3 important components, namely, (1) the input vector x with distribution $P(x)$, (2) the supervisor with output vector y with conditional distribution $P(x|y)$, and (3) the learning machine that im-

plements a prediction function $\varphi(x, w)$ where w are the weights in the neural network that can be varied to give the best approximate response. The discrepancy between the response y and the prediction function $\varphi(x, w)$ is qualified by the loss function $\ell(y, \varphi(x, w))$, which is aggregated with the conditional distribution to give the risk functional of the form:

$$R(w) = \int \ell(y, \varphi(x, w)) dP(x, y). \tag{1.1}$$

However, the real-world joint probability distribution $P(x, y) = P(y|x)P(x)$ is unknown, so one turns to the empirical risk functional

$$E(w) = \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, \varphi(x_i, w)), \tag{1.2}$$

where the training set $\{x_i, y_i\}_{i=1}^{N}$ is used in lieu of the true total population distribution, and ERM assumes that the weights w^* that minimizes $E(w)$ will also result in a risk $R(w^*)$ close to the true minimum $R(w)$. This is an important assumption since this is linked to the algorithm's vulnerability to discern previously unseen perturbations in the training set. Adversaries can also exploit this by instigating noise distribution shifts to their favor and cause the algorithm to misclassify completely.

1.1.2 Adversarial Attacks on Neural Networks

The aforementioned examples of deception are carried out on deep learning image classifiers through a technique termed *adversarial attack*. Szegedy et al. (2014) first discovered that neural networks with high accuracy are prone to misclassification when certain imperceptible perturbations are applied. Such perturbations can be found by maximizing the network's prediction error, and the resulting images are termed *adversarial examples*. This research sparked the interest to create increasingly sophisticated adversarial attacks, and Goodfellow et al. (2015) soon followed with the introduction of the fast gradient sign method (FGSM) method that could work effectively within a small perturbation budget of ε by using the gradient of the loss function ℓ to create adversarial images of the form

$$x_{adv} = x + \varepsilon v, \tag{1.3}$$

where v is a vector determined by the sign of the gradient of $\ell(y_i, \varphi(x_i, w))$ with respect to x as evaluated at each data point x_i.

This gradient-based method is enhanced in Kurakin et al. (2016) that proposes the Basic Iterative Method, which applies the *fast* method multiple times with a smaller step size. This family of iterative norm-bounded perturbations is termed as projected gradient descent (PGD) and commonly regarded as the strongest first-order attack.

1.1.3 Defense against Adversarial Attacks

The demonstration of neural network vulnerability to adversarial examples has spurned numerous methods to improve its robustness, such as defensive distillation (Papernot et al. 2016; Carlini and Wagner 2017a), feature squeezing (Xu et al. 2018; He et al. 2017), and evaluation of several other adversarial detection approaches (Carlini and Wagner 2017b). These aforementioned methods give a good cursory exploration of the solution space, but they do not offer a good understanding for guaranteed protection, and only work empirically against specific attacks. This makes it difficult to evaluate the associated security risks and implications, thereby hindering its adoption in military applications.

Madry et al. (2017) shows that one can adopt the robust optimization approach to achieve precise robustness guarantees against a broad class of attacks. This approach is formulated as

$$\min_{w} \mathbb{E}_{(x,y)} \left[\max_{\xi \in \Xi} \ell(y, \varphi(x + \xi, w)) \right], \tag{1.4}$$

where the min-max formulation casts both the attack and defense in a common framework, and runs an inner optimization procedure to find the worst-case adversarial examples during training before adding them in some sense to the training data. This is termed *adversarial training* and often regarded as the most successful method for training robust deep neural networks.

Typical adversarial training will include some form of adversarial attack to approximate the inner maximization term, followed by gradient descent on the model to solve for the parameters w. Madry et al. (2017) shows that the PGD adversary makes a good approximate in this training, and multiple random restarts typically improve the approximation of the inner maximization.

Various refinements of the adversarial training method produce better convergence and robustness, such as Dong et al. (2018) for incorporating momentum in the attack procedure and Xie et al. (2019) for incorporating feature denoising in the network architecture. More notable methods include TRADES proposed in Zhang et al. (2019) to trade adversarial robustness against accuracy, and MART proposed in Wang et al. (2020) which explicitly differentiates the misclassified and correctly classified examples during training. However, Rice et al. (2020) shows that the performance gains from the aforementioned algorithmic improvements can be matched by early stopping and there are still aspects the research community does not yet understand about adversarial training.

Despite the continuous development of adversarial training algorithms, the min-max problem formulation is difficult to solve and may require extensive computational resources and training time. This is because (a) the min-max problem is essentially nonsmooth and can potentially cause standard algorithms to slow down, and (b) subgradient calculations require solving a nonconcave and constrained maximization problem to determine the present worst case attack, which is computationally costly to achieve even approximately. In the military context, this time penalty translates to longer lead time to retrain and recover AI systems after detecting the adversarial attacks, thereby hindering the operation of some time critical functions such as target identification in unmanned ISR and facial recognition in security screening.

1.1.4 Risk Considerations in Statistical Training

Instead of minimizing the mean loss as in ERM, one can focus more on the higher tail-end losses and minimize the worst-case losses. Rockafellar and Royset (2010) provides an analysis of this alternative approach through the use of superquantiles. Generally, the α-superquantile of a random variable is defined as the average value of the outcomes beyond the α-quantile. This concept can be traced back to Rockafellar and Uryasev (2000), where

it is also called conditional value at risk (CVaR), and is commonly used in finance.

The application of superquantile in the context of machine learning is not new. Sani et al. (2013) uses superquantiles to measure regret in risk-averse decision making under a multiarmed bandit setting to identify the arm which best trades off risk and returns; Chow et al. (2015) proposes using superquantiles as a measure of risk in risk-constrained Markov decision processes for efficient reinforcement learning; and Williamson and Menon (2019) uses the superquantile concept to formulate a novel risk measure for evaluating fairness. In the field of classification robustness, Laguel et al. (2020) proposes the use of a smoothed superquantile function to achieve more robust behavior, but it works only with a customized optimization toolbox for superquantile-based learning that is difficult to implement for neural network training. Soma and Yoshida (2020) proposes a risk-averse statistical learning framework that uses the superquantile approach to evaluate losses and compares its accuracy with typical learning techniques using stochastic gradient descent (SGD). Curi et al. (2020) proposes an adaptive sampling algorithm for evaluating the superquantile-based loss distribution and has demonstrated higher robustness using common SGD based optimization. However, none of the recent papers have examined robustness in the adversarial context and mostly limited to perturbations inherent in the dataset.

1.2 Study Objective

In this thesis, we propose an alternative approach to the min-max problem in adversarial training Equation (2.1) and make use of smooth approximations to take advantage of the associated speed-up, while implicitly identifying the approximately worst attacks instead of solving for the worst attack via a maximization problem. This approach relies on approximating the worst-case loss by the superquantile loss. The maximization portion in adversarial training is replaced by an expectation of artificially generated sample points and predefined perturbations. Moreover, this approach makes the adversarial problem look more like ERM where any version of SGD could be applied, but at the cost of some additional parameters and uncertainties.

In view of the challenges in building robust machine learning applications under an operational setting (susceptible to both deliberate attacks and random noise), this thesis aims to apply the superquantile approach to train robust neural network models under the following

scenarios - (1) worst case adversarial attack, and (2) random noise attacks, while requiring less time and computing resources compared to typical adversarial training.

1.3 Thesis Organization

The subsequent sections of this thesis formulate and apply the idea of superquantiles in training support vector machine (SVM) and convolutional neural network (CNN) models to generate robust solutions for various real datasets. Chapter II presents the mathematical formulation and experimentation approach used in this thesis. Chapter III examines the application of superquantile SVM model to a real-world cancer testing dataset and compares the accuracy and robustness performance against L_1-regularization. Chapter IV focuses on the testing of superquantile CNN on real-world Modified National Institute of Standards and Technology (MNIST) and Canadian Institute For Advanced Research - 10 classes (CIFAR-10) database and compares the performance with common adversarial training techniques. Chapter V concludes the thesis and highlights possible areas for future research.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 2:
Formulation and Methodology

Adversarial training has been proven as the most effective method of training neural networks to be robust to random noise and deliberate adversarial attacks. We can formulate adversarial training as a min-max problem of the form

$$\min_{w} \frac{1}{N} \sum_{i=1}^{N} \max_{\xi_i \in \Xi} \ell(y_i, \varphi(x_i + \xi_i, w)), \tag{2.1}$$

where $\{x_i, y_i\}_{i=1}^{N}$ is the training dataset of size N, with x denoting the input vector and y being the desired output vector of a prediction model φ, which has model parameters w (e.g. weights in the neural network). Ξ is the perturbation set containing the perturbation vectors ${\xi_i}_{i=1}^{N}$ that targets the input vector x. The inner maximization serves to find the worst-case samples, while the outer minimization serves to train the expected loss to give the best possible robust prediction model.

There are various forms of the loss functions ℓ to quantify how well the model is able to approximate the given data, such as the Squared Error loss function and the commonly used Cross Entropy loss function. In regression problems, the Squared Error loss, also called L_2-loss, is simply the square of the difference between the true value y and the corresponding predicted value z:

$$\ell_{se}(y, z) = (y - z)^2. \tag{2.2}$$

In multi-class classification problems with C number of classes with one-hot encoded labels (e.g. $[\ 0 \quad 0 \quad 0 \quad 1 \quad \ldots 0]$), the categorical Cross Entropy loss is the log product between the actual class label y_j and the corresponding predicted probability z_j:

$$\ell_{ce}(y, z) = -\sum_{j=1}^{C} y_j \log(z_j). \tag{2.3}$$

This effectively predicts the probability of the given datum belonging to the target class, similar to maximum log-likelihood estimation, where the objective is to maximize the likelihood of the predicted distribution in matching the ground truth.

The min-max problem is difficult to solve in practice. We propose to use superquantiles to approximate the worst case condition and use its smooth approximation and convexity properties to speed up the solution process.

2.1 Superquantile Approximations

For $\alpha \in [0, 1)$ and a random variable v, the α-superquantile is defined as

$$\bar{Q}_\alpha = Q_\alpha + \frac{1}{1-\alpha}\mathbb{E}\left[\max\{0, v - Q_\alpha\}\right], \tag{2.4}$$

where Q_α is the α-quantile of v (Royset and Wets 2021, Section 3.C). This means that the superquantile $\bar{Q}_\alpha$ represents the average of the outcomes above the quantile Q_α as shown in Figure 2.1.

The special case $\bar{Q}_0 = \mathbb{E}[v]$ shows that the superquantiles includes expectations, and conversely, we define $\bar{Q}_1$ as the limit of $\bar{Q}_\alpha$ for $\alpha \to 1$. By Royset and Wets (2021), Section 3.C, this implies that $\bar{Q}_1$ equals the largest possible value of v.

In the context of learning, this means that we can view the maximization of the loss function in adversarial training

$$\max_{\xi_i \in \Xi} \ell\left(y_i, \varphi(x_i + \xi_i, w)\right) \tag{2.5}$$

as equivalent to the $\alpha = 1$ superquantile of the loss $\ell\left(y_i, \varphi(x_i + \xi_i, w)\right)$, where we treat the loss function as the random variable v. We think of ξ_i as a component of the loss function

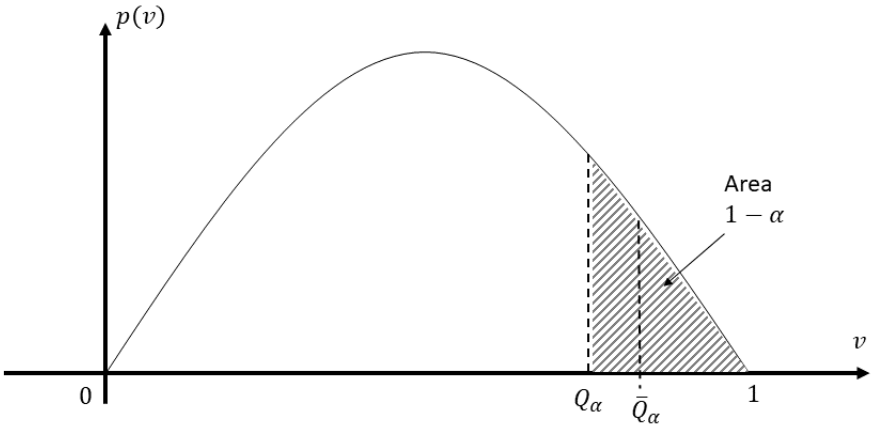

Figure 2.1. Probability density function for a continuous random variable v with the quantile Q_α marked out and superquantile $\bar{Q}_\alpha$ as the mean of v in the upper α-quantile.

and a random vector with a distribution supported on Ξ. Moreover, this superquantile can be approximated by $\bar{Q}_\alpha$ of the random variable for $\alpha < 1$.

By Rockafellar and Uryasev (2000), a superquantile can be solved via a minimization formula, specifically,

$$\bar{Q}_\alpha = \min_\gamma \left\{ \gamma + \frac{1}{1-\alpha} \mathbb{E}\left[\max\{0, v-\gamma\}\right] \right\}, \tag{2.6}$$

through the auxiliary variable γ for any random variable v.

We treat the loss function $\ell\left(y_i, \varphi(x_i+\xi_i, w)\right)$ as the random variable and this leads to

$$\min_w \frac{1}{N} \sum_{i=1}^{N} \min_{\gamma_i \in \mathbb{R}} \left\{ \gamma_i + \frac{1}{1-\alpha} \mathbb{E}\big[\max\left\{0, \ell\big(y_i, \varphi(x_i+\xi_i, w)\big) - \gamma_i\right\}\big] \right\}, \tag{2.7}$$

which is an approximation of (2.1). Here, the expectation is taken with respect to the random variable ξ_i, which is assumed to have support Ξ. In particular, as $\alpha \to 1$, this approximation becomes more accurate. This is a convex problem if $\ell(y, \varphi(x, w))$ is convex in w.

After approximating the inner expectation using stochastic sampling, we achieve an approximate expression for adversarial training using the concept of superquantiles:

$$\min_{w,\gamma_1 \ldots \gamma_N} \frac{1}{N}\frac{1}{M}\sum_{i=1}^{N}\sum_{j=1}^{M}\left\{\gamma_i + \frac{1}{1-\alpha}\max\left\{0, \ell\big(y_i, \varphi(x_i+\xi_{ij}, w)\big) - \gamma_i\right\}\right\}, \tag{2.8}$$

where ξ_{ij}, $i = 1, \ldots, N$, $j = 1, \ldots, M$ are samples from any distribution on Ξ.

This formulation is easier to solve computationally as it avoids the difficult inner maximization term in adversarial training that typically requires additional iterative steps to find the subgradients of the model w and the corresponding worst case perturbation to the model. Instead, it is approximated with a simple-to-compute maximization operation with a choice of α that is close to 1. It is also more computationally efficient for the solver to find the corresponding γ in the convex superquantile formulation since it should typically take less dimensions than the model weights w in the nonconvex neural network models. Therefore, we should expect to see faster convergence rate in a convex setting.

However, this still leaves behind a minor issue of a maximization term in (2.8). We can easily overcome this by designing some smoothness into the objective function to facilitate the use of gradient-based algorithms. We introduce a new parameter $\beta \in (0, \infty)$ and define the smooth function:

$$h_\beta(\eta) = \frac{1}{\beta}\ln\left(1 + e^{\beta\eta}\right). \tag{2.9}$$

However, the exponential term can result in overflow during the training process, so an alternative formulation is used:

$$h_\beta(\eta) = \max\{0, \eta\} + \frac{1}{\beta}\ln\left(e^{-\beta\max\{0,\eta\}} + e^{\beta\eta - \beta\max\{0,\eta\}}\right). \tag{2.10}$$

Substituting (2.10) into the superquantile form in (2.8) gives the final smoothed version with the additional parameter β:

$$\min_{w,\gamma_1 \ldots \gamma_N} \frac{1}{N}\frac{1}{M}\sum_{i=1}^{N}\sum_{j=1}^{M}\left(\gamma_i + \frac{1}{1-\alpha}h_\beta\left(\ell\big(y_i, \varphi(x_i+\xi_{ij}, w)\big) - \gamma_i\right)\right). \tag{2.11}$$

This proposed reformulation resembles ERM, but with a different criterion function with additional uncertainties and parameters. This criterion function serves to preserve the convexity and smoothness properties in w, even locally, thereby providing a faster rate of convergence where applicable. The number of additional parameters is usually much smaller than the dimension of the neural network model.

The α-superquantile parameter serves to adjust the level of robustness by penalizing larger and unlikely attacks, instead of just looking at reducing the mean number of mistakes. This can help to address the issue of classifiers performing well on average, but occasionally failing outright. This is equivalent to adding a regularization effect in the training process and thereby drawing a more robust decision boundary. In this thesis, we consider different α values from 0 to near 1 when testing on real-world data, which can have nonlinear relationships and difficult to solve.

This formulation is also designed with the purpose of maintaining compatibility with all versions of gradient-based methods in mind. This will allow us to leverage any future developments in this direction since the use of gradient methods, such as SGD, is very popular in the machine learning community.

2.2 Experimentation Approach

Since there are limited references available for the use of superquantiles in neural networks and almost none in its application in adversarial training, we decided to adopt a more progressive approach starting with the well-established SVM method. This is a simple and useful check to verify that the superquantile formulation does tend to be more robust as α increases and the loss function is truncated to the worst case upper tail-end of the distribution. We can also easily benchmark superquantile performance against L_1-regularization as reference. Moreover, SVM is relatively quick to run on most modern machines, so we can conduct more parametric studies on the different parameters (such as the type of perturbation, size of training perturbation and amount of regularization).

Subsequently, we switch to the main focus — neural network training where both random noise and adversarial noise perturbations are used. In neural network training, we use SGD to optimize the model weights w and the auxiliary variable γ. SGD belongs to a class of algorithms called *Gradient Descent* which are iterative algorithms that typically start from a random point on a function and travel in the direction of steepest descent in small steps until it reaches the minima. The step size taken per iteration is determined by the learning rate, which needs to be small enough to avoid missing minima, while large enough to ensure practical convergence speed; see Royset and Wets (2021), Section 3.G.

For large datasets, it becomes difficult to compute the gradient for all data points and SGD circumvents this issue by randomly choosing one data point at each iteration to reduce the computations required. However, this can be slow to converge, so it becomes common to sample more datapoints called *mini-batch* to improve the accuracy estimate of the error gradient. In addition, SGD and mini-batches are set such that each datapoint in the training dataset will be passed exactly once in an iteration cycle, called *epoch*.

It is clear that the SGD method has more hyperparameters to consider than SVM. This include the training and testing batch size, step size per iteration, and total number of iterations. The choice of model is also important since too deep a network could result in "memory leak" where information from the input is lost and the network does not learn anything. All these involve more testing and troubleshooting on the setup before we can obtain quality results.

Lastly, we benchmark the superquantile methods robustness performance against typical adversarial training using commonly used MNIST and CIFAR-10 datasets. Since the training process will take much longer to run, the number of parametric runs are limited and we will target only those of strong interest, namely, model accuracy changes with α and amount of perturbation.

Using these benchmark tests, we can then infer the performance gains and trade-offs with the superquantile method and objectively compare its value with usual adversarial training.

CHAPTER 3:
SVM Robustness

SVM is a commonly used machine learning method for classification, regression and outlier detection. The objective of the SVM algorithm is to find a hyperplane that can effectively divide the known dataset into their respective classes, which is effectively a minimization problem to have as few misclassified points as possible. We are interested in SVM because it also shares similarities with adversarial training. It also has an inner maximization component that seeks the largest margin between the data points and the decision boundary for more robust prediction. We can reformulate the objective into a superquantile form and experiment with real-world data to get a better understanding of its robustness characteristics, specifically its response to changes in α quantile and training noise.

3.1 SVM Formulation and Algorithm

We apply the SVM method in a similar setting to ERM in (1.2) but with a predicted category $z = a^\top x + b$, and the goal is to estimate the corresponding parameters a, b that minimizes the average loss in the training dataset:

$$\min_{a,b} \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, a^\top x_i + b), \tag{3.1}$$

where the loss function is of the form $\ell(y, z) = \max\{0, 1 - yz\}$. Substituting this into the loss function produces

$$\min_{a,b} \frac{1}{N} \sum_{i=1}^{N} \max\{0, 1 - y_i(a^\top x_i + b)\}. \tag{3.2}$$

To find a solution that is robust to uncertainty or perturbations to the x dataset, Lanckriet et al. (2002) proposes introducing an additional perturbation variable $\xi \in \Xi$ and modifying the goal to minimize the worst case (maximum) misclassification brought about by the

perturbation:

$$\min_{a,b} \frac{1}{N} \sum_{i=1}^{N} \max_{\xi_i \in \Xi} \left\{ \max\{0, 1 - y_i(a^\top (x_i + \xi_i) + b)\} \right\}. \tag{3.3}$$

If the perturbation set $\Xi = \{\xi | \|\xi\|_1 \leq \rho\}$, then we can show that the problem is equivalently stated as:

$$\min_{a,b} \frac{1}{N} \sum_{i=1}^{N} \max\{0, 1 - y_i(a^\top x_i + b)\} + \rho \|a\|_1. \tag{3.4}$$

The min-max formulation is difficult to solve computationally, and this usually requires nonlinear programming that is slow to compute. Instead, we see that it is possible to reformulate it into a linear objective function by introducing additional variables $z_1, \ldots, z_N$ and $u_1 \ldots u_L$ to replace the max-term and the norm in (3.4):

$$\begin{aligned}
&\min_{a,b,z,u} \frac{1}{N} \sum_{i=1}^{N} z_i + \rho \sum_{k=1}^{L} u_k \\
&s.t. \\
&\quad z_i \geq 0, \forall i \\
&\quad 1 - y_i \left(a^\intercal x_i + b\right) \leq z_i, \forall i \\
&\quad u_k \leq a_k, \forall k \\
&\quad u_k \geq -a_k, \forall k,
\end{aligned} \tag{3.5}$$

where u is an auxiliary variable for finding the regularization term that can minimize misclassification. There are better ways for solving the SVM robustness problem, such as expressing the uncertainty in intervals (El Ghaoui et al. 2003), having adjustable hyperplanes that accommodate real-world datasets (Le et al. 2014) and using the Difference-of-Convex for efficient computations (Zhang et al. 2018). Nonetheless, the formulation in (3.5) will suffice for our purpose here.

With this formulation, we can solve a multi-class classification problem with the appropriate choice of linear optimization solver for any labeled dataset. We split the dataset between training and testing, and inject perturbations to each so that we can test the accuracy of the model in predicting unseen and artificially perturbed test data. To test the robustness of the approach, the amount and type of perturbation ξ added to the test data are varied up to a bound of ε and repeated over multiple iterations to achieve greater statistical confidence.

We outline the algorithm to solve the SVM L_1-regularization problem in Algorithm 1. Firstly, the dataset is split into training and test sets, and normalized according to their population mean and standard deviation. Subsequently, we specify the parameter values for ρ and ε to set the amount of perturbation to use for testing and initialize the variables $\{a, b, u\}$ to 0 for the solver. After solving for the first set of $\{a, b, u\}$ using the formulation stated in (3.5), we calculate the prediction accuracy of the model against the previously unseen and perturbed test data. This solving process is repeated multiple times to assess how well the model performed on average against random perturbations.

Algorithm 1 Calculate the accuracy of SVM using L_1-regularization with a perturbation profile Ξ bounded by ρ

Select L_1 penalty term $\rho \geq 0$ and error budget $\varepsilon \geq 0$.
Parse the dataset and split 80:20 into training and test sets and project onto [0,1] range.
Normalize the training and test sets.
Initialize the variables $\{a, b, u\}$ to 0.
for $n \in$ num of iterations **do**
    Solve the SVM optimization problem for variables a and b in (3.5)
    Inject perturbations to the test data: $x_p = x + \varepsilon\xi$, while clipping $x_p \in [0, 1]$
    Calculate the accuracy of the model $d = 1 - \ell(y, a^\intercal(x_p) + b)$.
end for
Calculate the mean accuracy $\bar{d} = d/n$.

For the superquantile formulation, the objective function (2.8) can be reformulated with the introduction of an additional parameter z to replace the inner max term. The prediction function $\varphi(x, w)$ is assumed to be affine and takes the form $\varphi(x, w) = a^\top x + b$ with weights $w = (a, b)$. In contrast to the L_1-regularization problem in (3.4), the regularization term ρ is not used and perturbations are added to the training dataset. Additional unknowns to be considered in the problem include the superquantile parameter α and auxiliary variable γ. The choice of α close to 1 should control the loss calculation in a more conservative

manner similar to the use of a regularization term, while the γ variable will be solved to give the α-quantile for the given α specified. The SVM objective function with the use of superquantiles is reformulated as follows, with the attack size M:

$$\begin{aligned} \min_{a,b,z,\gamma_1 \ldots \gamma_N} & \frac{1}{N} \sum_{i=1}^{N} \left(\gamma_i + \frac{1}{1-\alpha} \frac{1}{M} \sum_{j=1}^{M} z_{i,j} \right) \\ s.t. & \\ & z_{i,j} \geq 0, \forall i, j \\ & 1 - y_i \left(a^\top (x_i + \xi_j) + b \right) \leq z_{i,j} + \gamma_i, \forall i, j \\ & 0 \leq z_{i,j} + \gamma_i, \forall i, j. \end{aligned} \tag{3.6}$$

We outline the algorithm used to examine the accuracy of the superquantile SVM method in Algorithm 2, which is largely similar to the algorithm for L_1-regularization except for the additional steps taken to account for the extra optimization variable γ and superquantile parameter α. In addition, we adjust the perturbed data to ensure that they do not exceed the allowable range $[0, 1]$ through *clipping*, which makes values greater than 1 become 1, and values below 0 become 0.

3.2 Experiment Setup

To test the robustness of the model with noise training, 2 types of random noise were added to the training and test data, namely, *uniform* and *Gaussian* noise. The amount of noise added is controlled by the noise budget δ and ε for training and test data respectively. In the case of uniform noise, the budget refers to the lower and upper bounds, while Gaussian noise uses this as the standard deviation.

There are many variables in the model, so we approach the problem progressively by first establishing a good baseline for comparison, before testing the effects of the superquantile variables. The criteria for the baseline include good model accuracy, reasonable robustness within a wide error budget range, and ability to demonstrate monotonic behaviour.

Each scenario is iterated more than 50 times using different independently generated sample

Algorithm 2 Calculate the accuracy of SVM using superquantile approach with a perturbation profile Ξ

Select $\alpha \in [0, 1)$, error budget $\varepsilon \geq 0$ and attack size M.
Parse the dataset and split 80:20 into training and test sets and project onto [0,1] range.
Normalize the training and test sets.
Initialize the variables a, b, γ to 0.
for $n \in$ num of iterations **do**
    Enlarge the vector space of x by duplicating it M times.
    Inject perturbations into the training dataset:
    $x_t = x + \delta\xi$, and clip $x_t \in [0, 1]$
    Solve the SVM optimization problem for parameters a and b in (3.6)
    Inject perturbations to the test data:
    $x_p = x + \varepsilon\xi$, and clip $x_p \in [0, 1]$
    Calculate the accuracy of the model $d = 1 - \ell(y, a^\top x_p + b)$.
end for
Calculate the mean accuracy $\bar{d} = d/n$.

noise for training and testing, and the average test accuracy is used for comparison. A sufficiently large number of iterations is necessary to clearly reflect the statistical difference between the results.

3.2.1 Breast Cancer Wisconsin Dataset

We use the dataset from Dua and Graff (2017) since it is commonly used for SVM testing. The data is collected from 569 digitized images and includes 32 attributes (all numeric) that describe the characteristics of the cell nuclei. The classification objective is to identify if the sample is "benign" or "malignant". This dataset is fast to run on any modern computer and has a known issue of overfitting to the training set without regularization. Therefore, we test it against the L_1-regularization and superquantile approach to compare their model accuracy under perturbations as a measure of their relative regularization strength.

3.2.2 Computation Resource Used

All computations in this chapter are performed on a local machine with a 3.6 GHz Intel i5-8600K processor, 16GB RAM and a 8GB NVIDIA Geforce 1070 Ti graphics processor. All statistical computations and data manipulation are performed using Python, while optimization is performed using CBC solver by Computational Infrastructure for Operations

Research (COIN-OR) and packaged using Pyomo. The solver typically took less than 5 min to reach a solution, while each test took less than 1 min to complete 50 iterations.

3.3 L_1-Regularization Results

Multiple tests are conducted over a range of L_1-regularization quantities (from 0 to 0.2) across an error budget ε (from 0 to 1) to find the optimal penalty value ρ that can produce the most robust result. As expected, the model accuracy is generally lower without L_1-regularization due to some overlap in the data clusters.

After testing under both uniform and Gaussian noise types, the results in Figures 3.1 and 3.2 show that $\rho = 0.02$ works best for the cancer dataset in both tests with the highest model accuracy of 96% under 0 noise. Interestingly, the regularization values in the range from 0.01 to 0.04 all produced fairly similar results, indicating that this particular dataset only requires a small amount of noise training to be robust. However, the benefit of regularization quickly breaks down once $\rho = 0.1$, where the accuracy under 0 noise takes a significant reduction to 85-86%, and this degrades further until it becomes seemingly agnostic to noise at $\rho = 0.2$.

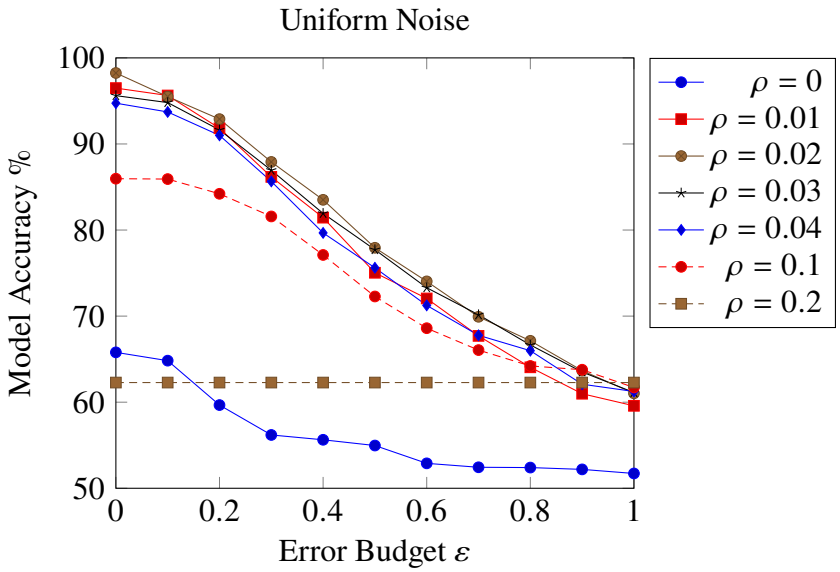

Figure 3.1. Model accuracy results with different L_1-regularization values and uniform noise perturbations

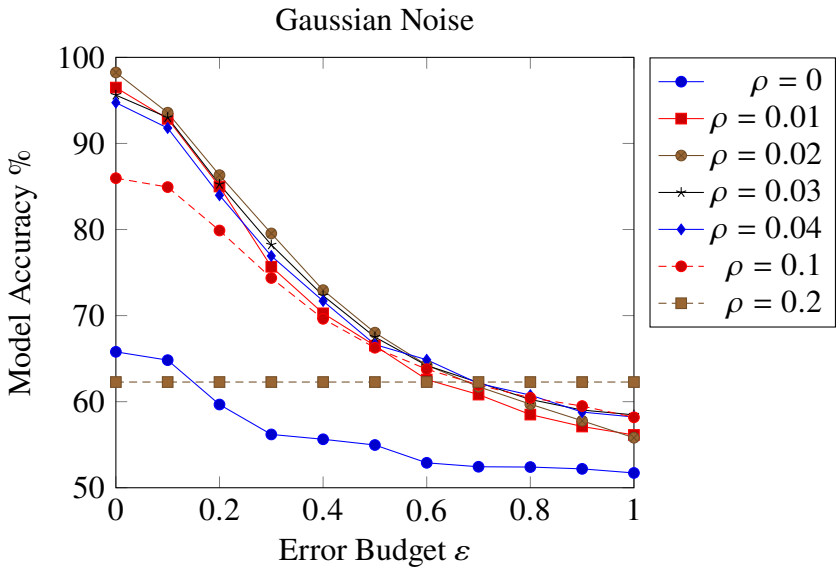

Figure 3.2. Model accuracy results with different L_1-regularization values and Gaussian noise perturbations.

Comparing the results between uniform and Gaussian noise shows that both results are largely similar with the same characteristics, but Gaussian produces a slightly stronger attack at higher error budgets. This is not unexpected since the Gaussian distribution is not bounded and able to produce higher perturbation than uniform at the tail-end.

3.4 Superquantile Results

3.4.1 Training Noise

There is a difference in the formulation between L_1-regularization and superquantiles, which will result in slightly different responses to the amount of noise added in the training data. L_1-regularization uses $\rho\|a\|_1$ as a standalone regularization term, while the superquantile keeps the training noise term within the prediction function $a^\top x + b$ and we vary the noise amount by δ such that $\xi = \delta\{U[0,1], N[0,1]\}$. Therefore, some difference is to be expected when we test with the same set of δ and ρ values.

Looking at the results in Figures 3.3 and 3.4, we notice that the training budget $\delta = 0.2$ performed better than the rest in terms of accuracy throughout the noise range. This implies

that the regularization effect using superquantiles is about 10 times weaker compared to L_1-regularization. This also caused a more gradual increase in accuracy with each increase in training noise. This implies that it may not be easy to search for the optimal amount of noise to achieve the most optimal amount of robustness. However, there is some assurance that this is monotonic, so one should continue to try increasing the amount of training noise until a satisfactory level of robustness has been obtained.

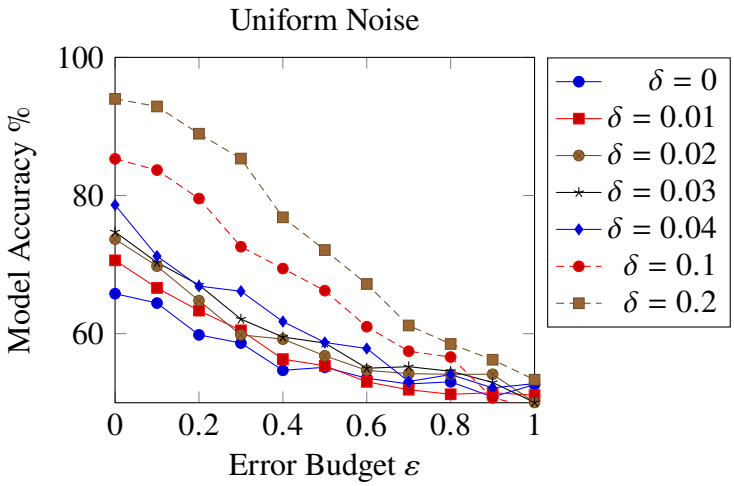

Figure 3.3. Superquantile model accuracy results with different training noise δ and uniform noise perturbation.

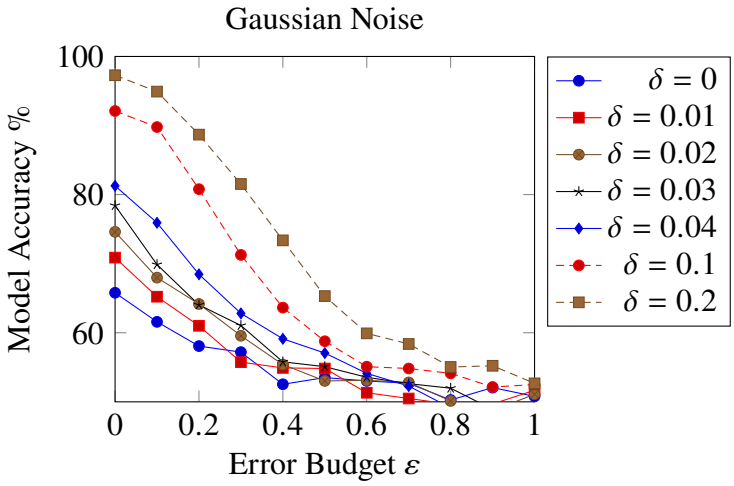

Figure 3.4. Superquantile model accuracy results with different training noise δ and Gaussian noise perturbation.

Comparing the results between uniform and Gaussian noise, we observe that both trends are largely similar, with Gaussian showing slightly steeper degradation with increasing ε. This was also noted in the L_1-regularization case, indicating that this dataset is more sensitive to Gaussian noise. Therefore, we choose $\delta = 0.2$ and Gaussian noise as the basis for comparison.

3.4.2 Superquantile Parameter α

Since α in the superquantile formulation controls the level of conservative loss with $\alpha = 1$ being the most risk-adverse with consideration for the worst possible examples, it should behave similar to a regularization term that develops robust models with controlled conservatism. However, testing with different α values with Gaussian noise perturbation shows no clear difference as shown in Figure 3.5. Nonetheless, the results match closely with the best L_1-regularization performance at $\rho = 0.02$.

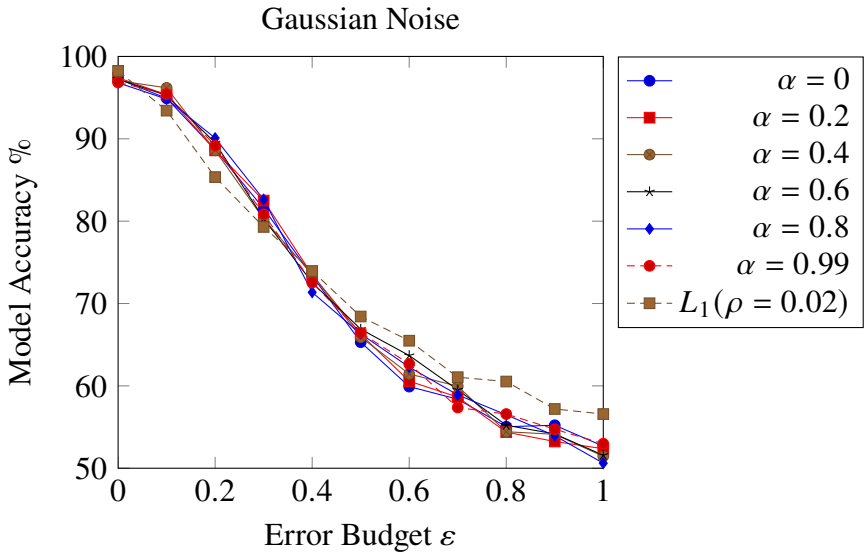

Figure 3.5. Minimal effect on the superquantile model accuracy with different α parameter at $\delta = 0.2$; similar robustness compared to L_1-regularization with optimal $\rho = 0.02$.

We explore the effect of increasing the attack size M in training from 1 to 10, since it can help to introduce more variability in the training space. This manages to create more differentiation between the different α results shown in Figure 3.6. Interestingly, this also

increases the robustness of the results to test perturbations. $\alpha = 0.9$ has more accuracy compared to L_1-regularization in the range $\varepsilon = (0.1, 0.5)$. However, the increase in attack size also causes a slight degradation in absolute model accuracy when $\varepsilon = 0$ and $\alpha = 0$, possibly due to an amplification of inherent data bias. The solver also has to deal with 10 times more training data, resulting in a significant increase in solving time (from 0.6 to 8.8 seconds). Nonetheless, we take this observation of combining high α and increasing attack size M into consideration for subsequent neural network analysis in Chapter 4 to achieve more robust results.

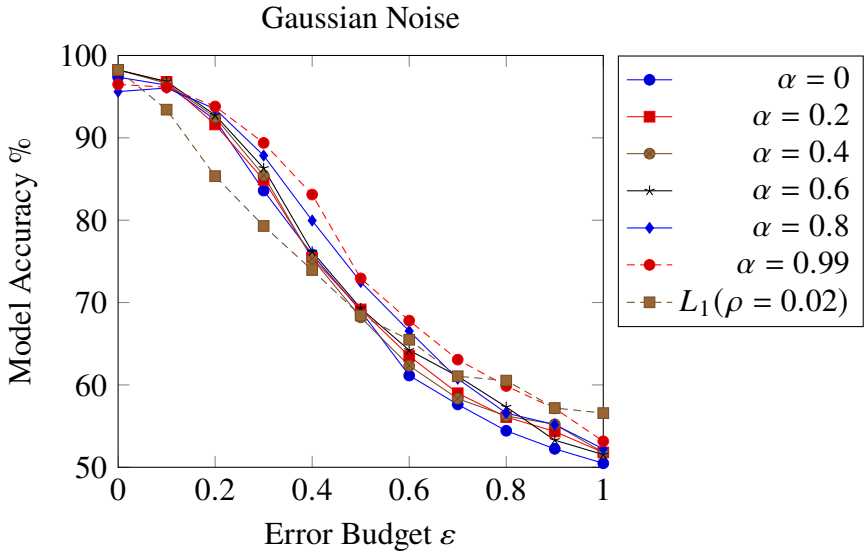

Figure 3.6. Some differentiation on model accuracy with attack size 10 and different α parameter at $\delta = 0.2$; higher robustness compared to L_1-regularization near $\varepsilon = 0.2$.

Using uniformly distributed random noise shows a more monotonic behavior with increasing α as shown in Figure 3.7. The increase in α produces a more pronounced improvement in model accuracy throughout the ε range compared to the Gaussian noise profile used in Figure 3.6. The improvement also appears to be more uniform throughout the error range, suggesting that the superquantile model responds more linearly to uniform random noise perturbations. However, many of the superquantile results are performing worse than L_1-regularization, prompting us to try higher δ training noise.

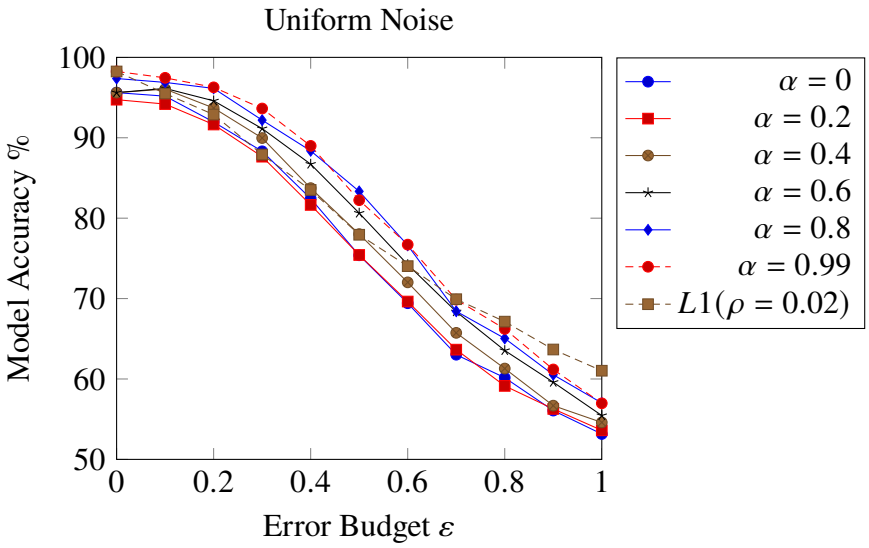

Figure 3.7. Greater differentiation on model accuracy with uniform distribution whilst keeping attack size 10 and different α parameter at $\delta = 0.2$; higher robustness than L_1-regularization only at certain conditions.

3.4.3 Higher Training Noise ($\delta = 0.4$)

Testing a large $\delta = 0.4$ training noise with attack size 10 surprisingly shows significantly better robustness than L_1-regularization throughout the test error range, as shown in Figure 3.8. This suggests that the superquantile formulation can offer a stronger regularization effect than typical L_1-regularization with more training noise. However, the high $\alpha = 0.99$ results in a reduction in model accuracy, similar to the effect of using a high L_1-regularization term in Figure 3.1. This suggests that the superquantile formulation can result in a strong regularization effect when $\alpha \rightarrow 1$ and the optimal level of regularization may be an α value somewhere in the range [0,1), which will require some experimentation to ascertain its performance.

Despite the reduction in model accuracy for $\alpha = 0.99$ for $\varepsilon = 0$, its accuracy in the upper error ε is the highest. Moreover, its accuracy appears flat at about 0.9 in the lower ε error range, making its performance more predictable when dealing with random noise perturbations. Therefore, there could be some benefits for choosing $\alpha = 0.99$, especially if we have prior knowledge that the noise perturbation is going to be high.

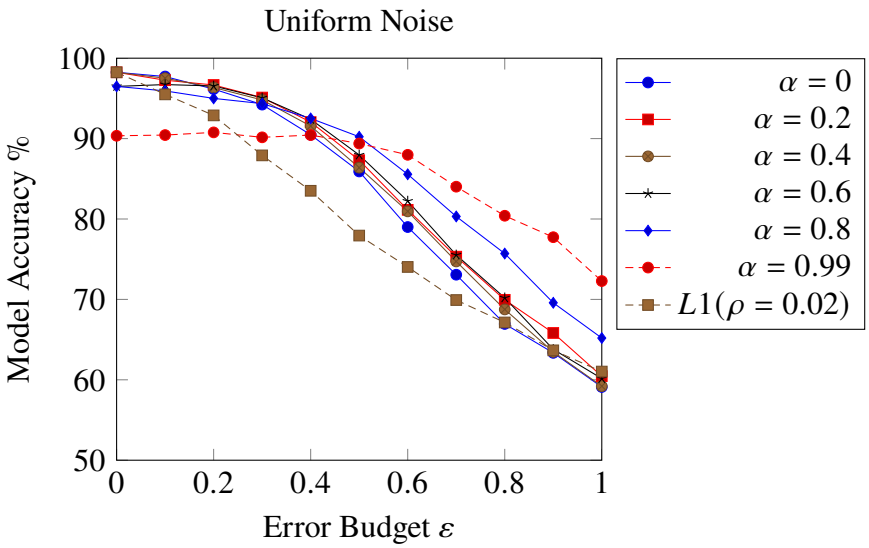

Figure 3.8. Increasing training perturbation to $\delta = 0.4$ at attack size 10 is able to produce greater robustness for certain α, but not in the lower ε range for high $\alpha = 0.99$.

CHAPTER 4:
Neural Network Robustness

The use of neural networks is prevalent in AI applications, and adversarial training is the most commonly accepted method to improve neural network robustness. However, this involves high computational costs and long training time, so we propose to apply the concept of superquantiles to approximate the adversarial process of finding worst case examples, with the aim of achieving a robust solution within a shorter training time. We first test the method on the more manageable MNIST dataset to ensure that our implementation of superquantiles is accurate and obtain stable SGD optimization hyperparameters, before experimenting with the more computationally expensive CIFAR-10 dataset.

4.1 Neural Network Training Algorithm

Various hyperparameters need to be predefined for stable and consistent learning (including learning rate, batch size, and total training epochs), in addition to the parameters for training perturbations that also require testing. To facilitate code development and troubleshooting, we take reference to the code and hyperparameters used by Wu et al. (2020) for achieving state-of-the-art robustness to CIFAR-10 dataset, and modify it to incorporate superquantile losses. For the MNIST dataset, we have to re-calibrate some of the parameters (such as the dimensions of the neural network model and learning rate) to suit the different data dimensions.

Our algorithm is outlined in Algorithm 3 and is similar to typical adversarial training where perturbations are injected into the training dataset and the final trained model is evaluated against previously unseen test data with different levels of perturbation. It starts with the initial selection of superquantile, training perturbation and neural network training hyperparameters by the user and parses in the respective training and test dataset. Some random transformations (rotation, flipping and normalization) are conducted on the dataset before training to prevent overfitting, as common practice for neural network training. Perturbations are added to the training and test data, and clipped to ensure that the perturbed data do not exceed the $[0, 1]$ range. In the SGD optimization process, the data in each

training batch are normalized before calculating the loss since this will generally speed up training and improve neural network learning (Ioffe and Szegedy 2015). The loss function is calculated using the superquantile formulation derived in 2.10.

Algorithm 3 Calculate the accuracy of the neural network using superquantile approach with random noise profile δ

Select $\alpha \in [0, 1)$, error budget $\varepsilon \geq 0$ and attack size L.
Specify the necessary training hyperparameters for stable training
Parse the dataset and ensure all values are in the $[0, 1]$ range.
Normalize the training and test sets, and perform random rotation (up to 5 deg) and vertical flipping on the training dataset.
Initialize the variable γ to 0
for $n \in$ num of epochs **do**
  for $m \in$ num of batches **do**
    Inject perturbations into the training dataset:
    $x_t = x + \delta\xi$, and clip $x_t \in [0, 1]$
    Normalize the training set x_t
    Calculate the loss function as stated in (2.10).
    Calculate the gradient of the loss function and perform backward propagation.
    Update the model and step forward in SGD.
  end for
end for
Inject perturbations to the test data:
  $x_p = x + \varepsilon\xi$, and clip $x_p \in [0, 1]$.
Calculate the accuracy of the model $d = 1 - \ell(y, a^\top x_p + b)$

4.2 Experiment Setup

Both random noise and adversarial trained perturbations are injected into the dataset to evaluate the robustness of the neural network. This consists of uniform noise, Gaussian noise and a 20-step PGD adversarial attack. We use a 1-step PGD method (equivalent of FGSM perturbation) to train for the adversarial attack since it is faster to compute than standard 20-step PGD training and could approximate the PGD attack distribution better than random noise. The use of FGSM for adversarial training is reported in Wong et al. (2020) to be just as effective as standard PGD-based training and can significantly shorten the training time. However, it is not widely adopted since past attempts can fail due to overfitting. Overfitting is less common in superquantiles as the choice of α close to 1 will

iteratively find the worst-case examples to fit the model. In addition, the images in the dataset are randomly rotated by up to 5 degrees as a standard practice to help alleviate overfitting.

A large minibatch of size 500 is used for SGD, as initial trials show that larger batch sizes give slightly better accuracy under large α values close to 0.9 and above. The failure of smaller minibatches matches the observation reported in Curi et al. (2020) that minibatch estimation of the gradient in the superquantile formulation suffers from high variance as only a fraction of the data points will be used. The paper only evaluates with a batch size of 128, probably due to computational constraints. A larger minibatch can potentially overcome this constraint by taking more points in the batch, thereby achieving better results. In addition, momentum 0.9 and weight decay 5×10^{-4} are used in SGD optimization for all training instances as commonly used in other works (Curi et al. 2020; Wong et al. 2020; Wang et al. 2020).

We use a learning rate of 0.01 as it offers a good compromise between the convergence rate, model accuracy, and stability. Higher α values near 0.9 and above sometimes require lower learning rates less than 0.001 since the $1 - \alpha$ term in the denominator of the loss function amplifies the losses too high, thereby resulting in unstable training.

There are various types of neural network models that could be used for image classification, and each have reported different degrees of success for training on the CIFAR-10 dataset. Residual neural network (ResNet), at the time of writing, is the de facto standard to use and at the top of the leaderboard for adversarial training (Croce et al. 2020). It performs well because it can skip connections over some layers to avoid vanishing gradients and mitigate accuracy saturation, which challenges plain neural networks with increasing layers. wide residual network (WRN) is a variation of ResNet with reduced depth but increased width, which is shown to perform better in Zagoruyko and Komodakis (2016). We choose to use WideResNet-34-10, following the setup in Madry et al. (2017), since it is shown to perform well in both image classification and adversarial training.

All computations in this chapter are performed in a compute server using 2 compute CPU cores, a 32GB Tesla V100 GPU, and 16GB of RAM allocated. All statistical computations and data manipulations are performed using Python, while the neural network operations are performed using Pytorch. Each epoch takes less than 10 seconds to train on the MNIST dataset, and less than 5 minutes to train on the CIFAR-10 dataset.

4.3 MNIST Experimentation

4.3.1 MNIST Dataset

The MNIST dataset from Lecun et al. (1998) consists of 32x32 gray-scale images of handwritten digits from 0 to 9, with 60,000 examples in the training set and 10,000 examples in the test set, shown in Figure 4.1. This is the most widely used dataset in supervised learning for handwritten digit recognition and it is relatively easy to achieve high accuracy above 90%. Most modern computers with graphics processing unit (GPU) acceleration can achieve this in a few minutes using only a few layers of neural networks.

Figure 4.1. Example images from MNIST showing the different handwritten digits from 0 to 9. Source: Lecun et al. (1998).

4.3.2 MNIST Results

We choose a training noise perturbation budget of $\delta = 1.0$ and fix this amount of perturbation throughout the training cycle. The choice of high training noise is generally recommended for superquantile training, since this essentially replaces the steps otherwise required to solve for the worst case adversarial examples in the min-max formulation (2.1).

The following MNIST tests have only 10 epochs of training because we are more concerned about achieving robust solutions within a short training time. In addition, the results demonstrate that the runs have already converged to acceptable solutions with >90% accuracy under no perturbation.

Figure 4.2 shows that under uniform noise, the accuracy of the model improves as expected as $\alpha \to 1$. Accuracy further improves as the attack size M increases, but only slightly. Overall, the model already appears very robust to uniform noise.

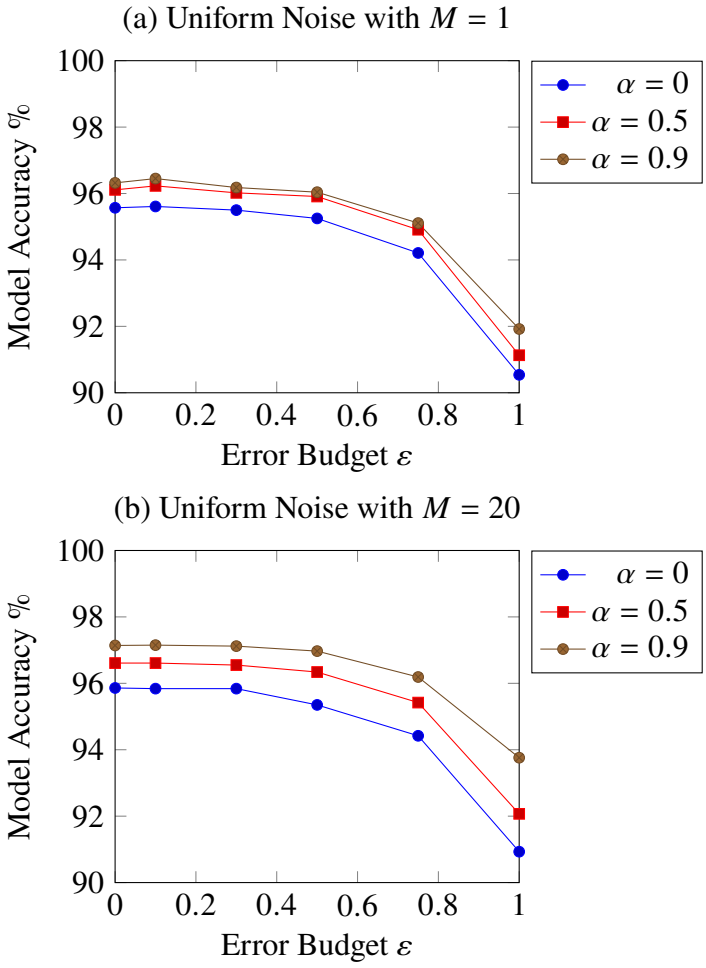

Figure 4.2. Model accuracy results on MNIST dataset with training noise $U(-1,1)$ and different attack size (a) $M = 1$, (b) $M = 20$ with the test dataset subject to uniform noise perturbations $\varepsilon U(-1,1)$.

Changing to Gaussian noise does not produce the same trend as $\alpha \to 1$, as shown in Figure 4.3. Instead, the accuracy is generally higher when the $\alpha = 0.5$ is chosen compared to $\alpha = 0.9$. This still holds even when the attack size M is increased from 1 to 20. Compared to uniform noise perturbation results, the accuracy of the model with Gaussian noise perturbation degrades more, with accuracy below 80% when subjected to test perturbations $\varepsilon = 1$. This shows that the use of superquantiles can be sensitive to the type of noise distribution. Moreover, the robustness improvement brought by higher α is also not apparent in this case.

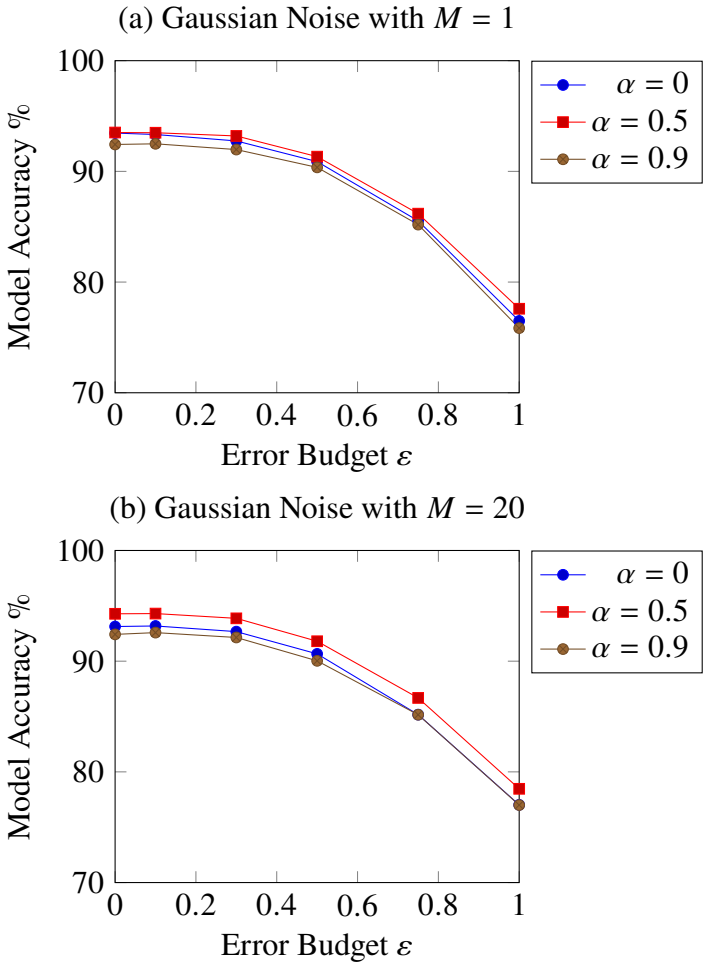

Figure 4.3. Model accuracy results on MNIST dataset with training noise $N(0,1)$ and different attack size (a) $M = 1$, (b) $M = 20$ with the test dataset subject to Gaussian noise perturbations $\varepsilon N(0,1)$.

After switching to PGD adversarial attack, we notice that it is clearly stronger than random noise attacks as model accuracy reduces to around 50% when the error budget $\varepsilon = 0.3$, as shown in Figure 4.4. This is expected as PGD attack is commonly regarded as one of the strongest forms of attack. We also note that model accuracy increases as $\alpha \to 1$, and generally improves when the attack size is increased to 20. The exception is $\alpha = 0.9$ in this case as the larger attack size results in a lower model robustness when $\varepsilon > 0.1$. This discrepancy could be due to a number of reasons, including peculiarities in the nonlinear solution space, ineffective 1-step PGD used in the training phase or just random initialization

problems. Nonetheless, the benefit of increased attack size seems insignificant compared to its increase in computation demand.

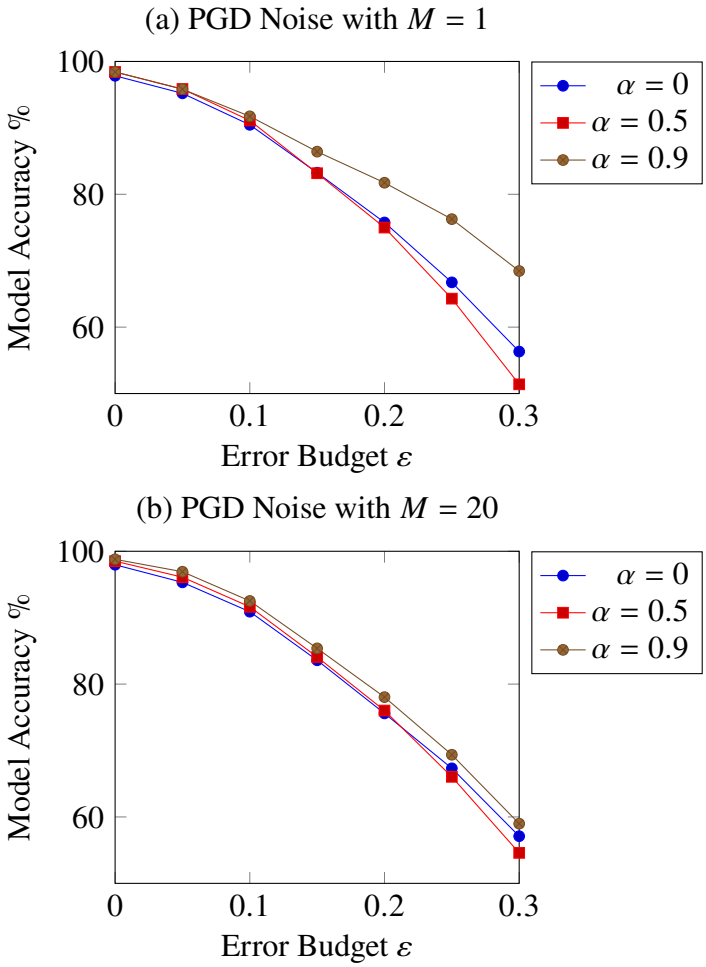

Figure 4.4. Model accuracy results on MNIST dataset with 1-step PGD training and different attack size (a) $M = 1$, (b) $M = 20$ with the test dataset subject to ε amount of PGD perturbations.

Overall, the MNIST dataset seems to show little benefit with the use of higher superquantile values, in contrast to earlier SVM results. This could be due to the nature of the dataset since the images of digits are fairly simplistic. This also explains how easy it is to achieve high accuracy within a few epochs and maintain its robustness against random noise. Since superquantiles work by finding the worst case examples within the data distribution, a benign dataset with distinct samples will essentially not be able to make a significant difference.

4.4 CIFAR-10 Experimentation

4.4.1 CIFAR-10 Dataset

The CIFAR-10 dataset in Krizhevsky and Hinton (2009) consists of 32x32 color images in 10 classes shown in Figure 4.5. It is divided randomly into a training set with 50,000 images (5,000 images per class) and a test set with 10,000 images (1,000 images per class). Typical supervised learning conducted using CNN can achieve accuracy about 80%, but computation time for each test case will take hours, or even days for adversarial training. Therefore, this is left as the final benchmark comparison due to study time constraints.

Figure 4.5. Example images from CIFAR-10 showing the different classes of images corresponding to physical objects. Source: Krizhevsky and Hinton (2009).

4.4.2 CIFAR-10 Results

We choose a training budget of $\delta = 0.1$ as the scale of perturbations to add to the training dataset. This is chosen after some initial trials that show higher training perturbation leading to poor or no training under certain scenarios. Instead of complicating the setup by introducing more regularization terms and optimization changes, we decide to reduce the budget to a sufficiently low value, while maintaining the robustness performance.

For benchmarking with typical adversarial training, each experiment runs for 200 epochs to allow sufficient iterations to reach convergence, instead of stopping at shorter epochs. The adversarial training method uses a similar algorithm to train, with the only key difference that it uses a standard cross entropy loss function instead of the superquantile. A learning

rate scheduler is used to reduce the learning rate by a factor of 10 at every 50 and 100 epochs to improve the training stability and keep the solution from diverging away from the optimal point. This schedule references the same scheme used in Wang et al. (2020) for adversarial training, which manages to achieve good accuracy under adversarial attack.

The random noise perturbation results, shown in Figure 4.6, demonstrate that the models are fairly robust within the error budget range. Interestingly, as α increases, the model accuracy increases as well, thereby showing better results when $\alpha = 0.9$. This aligns with our initial observation with SVM where superquantile training exhibits a similar effect to improving both accuracy and robustness.

Our results with PGD attack show a much larger differentiation, as shown in Figure 4.7. This supports the initial observation that PGD is a stronger attack than random noise, to the extent that $\alpha = 0$ experiences misclassification of all the data at $\varepsilon = 0.3$. This observation concurs with Szegedy et al. (2014) that first uncovers the vulnerability of neural networks to adversarial attacks. Increasing α to 0.9 leads to more robust results with performance levels matching more closely to typical adversarial training. This clearly demonstrates the ability of superquantile training to improve model robustness against adversarial attack.

Moreover, each epoch in the superquantile training only takes an average of about 2 minutes and 30 seconds, while typical adversarial training will take an average of 3 minutes and 45 seconds. Therefore, superquantile training can potentially offer about 35% faster results with comparable robustness performance, especially in the lower ε error range.

However, superquantile training can not perform as well as adversarial training because the two methods use different adversarial examples with the superquantile training only using a weaker approximate 1-step PGD. A better approximation will require a multi-step PGD, which will trade off with more computational time. Nonetheless, the utility of an approximate method is clear from the area between $\alpha = 0$ and $\alpha = 0.9$ in Figure 4.7, which denotes the accuracy improvement bought about by superquantile training.

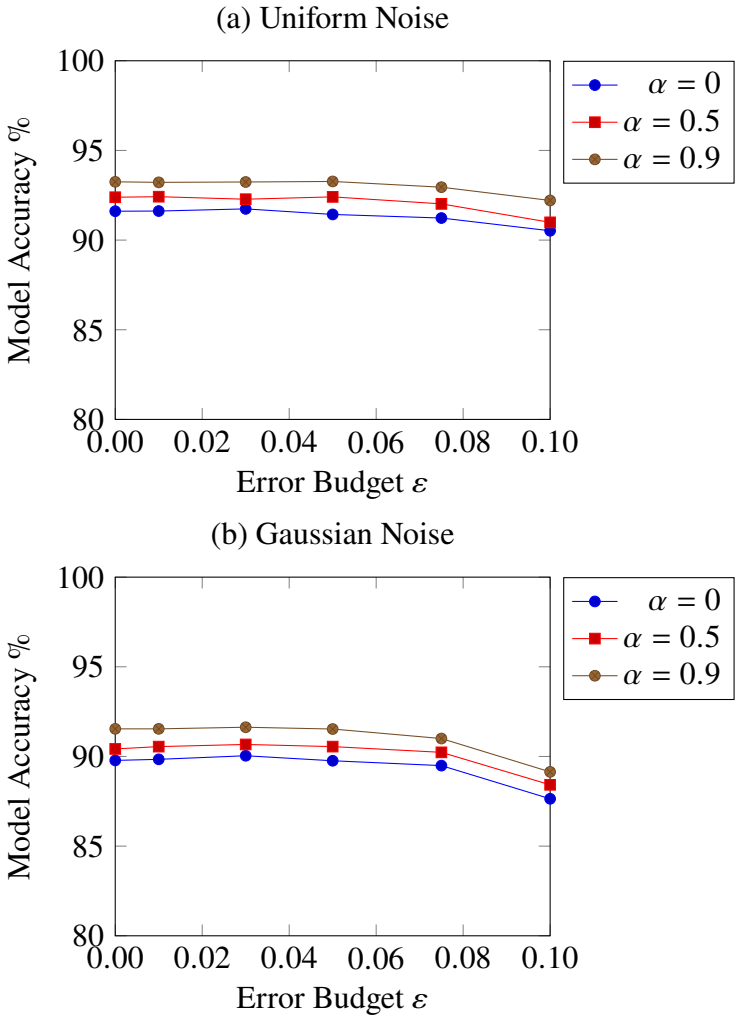

Figure 4.6. Model accuracy results on CIFAR-10 dataset subject to (a) random noise, and (b) Gaussian noise perturbations on the test data.

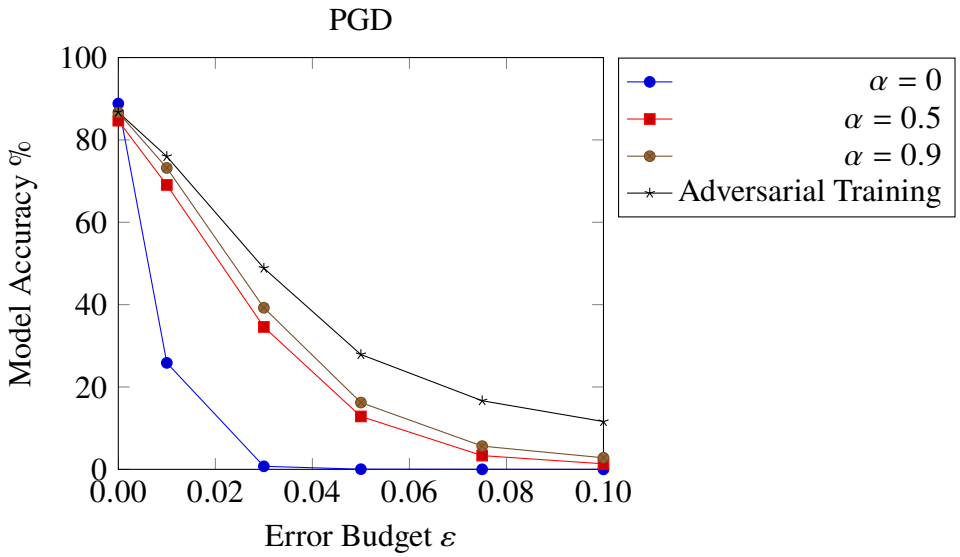

Figure 4.7. Comparing model performance between superquantile training with different α and a typical adversarial training.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 5:
Conclusions

We present in this paper a novel application of superquantile optimization in adversarial training, to alleviate the computation time required to solve for the worst case adversarial perturbations. Our proposed method works by finding the approximate worst case losses by focusing on the higher tail-end losses in the perturbed training distribution, and the initial results show some promise that this method works for neural network training using commonly used Stochastic Gradient Descent optimization.

Our work here provides an initial excursion into superquantile training in support vector machines and neural networks. In the context of support vector machines, superquantile training gives similar results to L_1-regularization with the appropriate amount of training perturbation and choice of suitable superquantile parameter α close to 1. Increasing the attack size to 10 allows superquantile training to achieve about 2% to 5% better accuracy than L_1-regularization under uniform noise, and about 2% to 10% improvement under Gaussian noise, while within a test error budget between 0.1 to 0.5. However, the 10 times increase in attack size leads to about 13 times more computation time, which does not seem to be a favorable trade-off. Increasing the amount of training perturbation is able to produce up to 10% higher robustness than L_1-regularization for most values of α except for $\alpha = 0.99$, which experiences a reduction in training accuracy in the lower error budget range of 0 to 0.4. This shows that we have to experiment with different α and training noise to achieve optimal robust results.

In the context of neural networks, superquantile training is robust to random noise distribution and projected gradient descent adversarial attacks to varying degrees. For the MNIST dataset, there is less than 5% difference between different choice of α and attack sizes, and in the instance of Gaussian noise perturbations, $\alpha = 0.5$ produces higher accuracy than $\alpha = 0.9$, even though both are very close. Switching to an adversarial attack results in faster degradation in model accuracy with increasing test perturbations, but $\alpha = 0.9$ produces 12% better accuracy at higher error budget $\epsilon = 0.3$, thereby demonstrating the value of superquantile training for adversarial training.

For the CIFAR-10 dataset, superquantile training is robust to random noise in the 0 to 0.1 error budget range with accuracies close to 90%. This greatly degrades when subject to adversarial attack, where the accuracy at $\alpha = 0$ falls to near 0% at a test error budget of 0.03. Increasing to $\alpha = 0.9$ improves the robustness results and reduces the performance difference with typical adversarial training to 10%, while taking 35% less training time to achieve it. This is possible only with a careful choice of a large batch size of 500 and a suitably low learning rate of 0.01 or less. In general, the model increases in accuracy and robustness when the superquantile parameter α is close to 1, even under strong PGD adversarial attack.

Although these initial results show some promise, more work is required to certify that the method will offer certain robustness guarantees. This includes more experimentation on different datasets and expanding our theoretical understanding of superquantile training in neural network training.

5.1 Challenges and Limitations

There is generally a limited theoretical understanding of adversarial training throughout the machine learning field. This is because neural networks are typically nonlinear and nonconvex, so solving for the weights through optimization is typically NP-hard. As such, it is generally difficult to prove with confidence that a method working for one dataset and one type of perturbation will be applicable for all. We can use empirical methods to uncover the limits of the proposed method, but this is difficult to generalize across all possible datasets.

In addition, there is no generally acceptable method of tuning the hyperparameters for different models and datasets. This is still a field of active study and more work is required to adopt adaptive algorithms in the superquantile context. We have uncovered that a larger batch size can produce better performance, but multiple trials have to be conducted to find the right set of hyperparameters such as learning rate and training noise amount. This trial process will extend the time required to build and run the model, thereby affecting the performance gains afforded by using the algorithm.

5.2 Recommendations for Future Work

Although we have shown that the superquantile can give comparable results compared to adversarial training, there is still room to enhance this further and possibly exceed the performance of typical adversarial methods. One of the important factors yet to be explored include schedulers for α parameter, learning rate and training noise. One could set adaptive schedules for α, noise and learning rate to accelerate the convergence to worst case perturbations or introduce excursions out of local optimal points to find more global optimal points.

In addition, we could explore solving for the auxiliary term γ in the superquantile formulation independently within each SGD epoch, instead of the current implementation, where it is optimized using SGD together with the weights in the neural network model. This means that the choice of quantile within each epoch can be varied independently and potentially help to improve the convergence and stability of the SGD optimization. However, this comes at a trade-off of higher computational cost per epoch, which could outweigh the speedup afforded by using superquantiles.

Adversarial training is a fast evolving area of study, new algorithms and techniques are constantly being developed and published regularly. Some of these techniques can also shorten the adversarial training process, and some can potentially be compatible with our proposed superquantile training to deliver even better results. Most notably, methods that can improve the approximation of the worst case perturbations will greatly help to improve the superquantile model accuracy and robustness.

Adversarial attacks can take various forms, and there are still many variants left unexplored in this thesis. One could experiment with the use of black-out pixels to partially obscure an object image. This has various military applications such as the use of camouflage to avoid detection, and conversely, the application of robust training to build image detection algorithms that can detect "hidden" targets.

THIS PAGE INTENTIONALLY LEFT BLANK

List of References

Adhikari A, den Hollander RJM, Tolios I, van Bekkum M, Bal A, Hendriks S, Kruithof M, Gross D, Jansen N, Pérez GA, Buurman K, Raaijmakers S (2020) Adversarial patch camouflage against aerial detection. *CoRR* abs/2008.13671. https://arxiv.org/abs/2008.13671.

Carlini N, Wagner D (2017a) Towards evaluating the robustness of neural networks. *2017 IEEE Symposium on Security and Privacy (SP)*, 39–57. https://doi.org/10.1109/SP.2017.49.

Carlini N, Wagner DA (2017b) Adversarial examples are not easily detected: Bypassing ten detection methods. *CoRR* abs/1705.07263. http://arxiv.org/abs/1705.07263.

Chow Y, Ghavamzadeh M, Janson L, Pavone M (2015) Risk-constrained reinforcement learning with percentile risk criteria. *CoRR* abs/1512.01629. http://arxiv.org/abs/1512.01629.

Cohen J, Rosenfeld E, Kolter Z (2019) Certified adversarial robustness via randomized smoothing. volume 97 of *Proceedings of Machine Learning Research*, 1310–1320. https://proceedings.mlr.press/v97/cohen19c.html.

Croce F, Andriushchenko M, Sehwag V, Debenedetti E, Flammarion N, Chiang M, Mittal P, Hein M (2020) RobustBench: a standardized adversarial robustness benchmark. *arXiv e-prints* arXiv:2010.09670.

Curi S, Levy KY, Jegelka S, Krause A (2020) Adaptive sampling for stochastic risk-averse learning. *ArXiv* abs/1910.12511.

Dodge SF, Karam LJ (2016) Understanding how image quality affects deep neural networks. *CoRR* abs/1604.04004. http://arxiv.org/abs/1604.04004.

Dong Y, Liao F, Pang T, Su H, Zhu J, Hu X, Li J (2018) Boosting adversarial attacks with momentum. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* 9185–9193.

Dua D, Graff C (2017) UCI machine learning repository University of California, Irvine, School of Information and Computer Sciences. http://archive.ics.uci.edu/ml.

El Ghaoui L, Lanckriet GRG, Natsoulis G (2003) Robust classification with interval data. Technical Report UCB/CSD-03-1279, EECS Department, University of California, Berkeley. http://www2.eecs.berkeley.edu/Pubs/TechRpts/2003/5772.html.

Goodfellow I, Shlens J, Szegedy C (2015) Explaining and harnessing adversarial examples. *CoRR* abs/1412.6572.

He W, Wei J, Chen X, Carlini N, Song D (2017) Adversarial example defenses: Ensembles of weak defenses are not strong. *CoRR* abs/1706.04701. http://arxiv.org/abs/1706.04701.

Ioffe S, Szegedy C (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR* abs/1502.03167.

Khurshid J, Bing-Rong H (2004) Military robots - a glimpse from today and tomorrow. *ICARCV 2004 8th Control, Automation, Robotics and Vision Conference, 2004.* 1:771–777 Vol. 1.

Krizhevsky A, Hinton G (2009) Learning multiple layers of features from tiny images. Technical report, University of Toronto, Toronto, Ontario.

Kurakin A, Goodfellow IJ, Bengio S (2016) Adversarial examples in the physical world. *CoRR* abs/1607.02533. http://arxiv.org/abs/1607.02533.

Laguel Y, Malick J, Harchaoui Z (2020) First-order optimization for superquantile-based supervised learning. *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*, 1–6 (IEEE).

Lanckriet G, Ghaoui L, Bhattacharyya C, Jordan M (2002) A robust minimax approach to classification. *Journal of Machine Learning Research* 3. https://doi.org/10.1162/153244303321897726.

Le T, Tran D, Ma W, Pham T, Duong P, Nguyen M (2014) Robust support vector machine. *2014 International Joint Conference on Neural Networks (IJCNN)*, 4137–4144. https://doi.org/10.1109/IJCNN.2014.6889587.

Lecun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324. https://doi.org/10.1109/5.726791.

Madry A, Makelov A, Schmidt L, Tsipras D, Vladu A (2017) Towards Deep Learning Models Resistant to Adversarial Attacks. *arXiv e-prints* arXiv:1706.06083.

Papernot N, McDaniel P, Wu X, Jha S, Swami A (2016) Distillation as a defense to adversarial perturbations against deep neural networks. *2016 IEEE Symposium on Security and Privacy (SP)*, 582–597 (IEEE).

Rice L, Wong E, Kolter Z (2020) Overfitting in adversarially robust deep learning. *International Conference on Machine Learning*, 8093–8104 (PMLR).

Rockafellar RT, Royset JO (2010) On buffered failure probability in design and optimization of structures. *Reliability Engineering & System Safety* 95(5):499–510. https://www.sciencedirect.com/science/article/pii/S0951832010000177, ISSN 0951-8320.

Rockafellar RT, Uryasev S (2000) Optimization of conditional value-at-risk. *Journal of Risk* 2:21–41.

Royset JO, Wets RJB (2021) *An Optimization Primer*. Springer Series in Operations Research and Financial Engineering (Springer International Publishing), ISBN 9783030762742.

Rusak E, Schott L, Zimmermann RS, Bitterwolf J, Bringmann O, Bethge M, Brendel W (2020) Increasing the robustness of dnns against image corruptions by playing the game of noise. *CoRR* abs/2001.06057. https://arxiv.org/abs/2001.06057.

Sani A, Lazaric A, Munos R (2013) Risk-aversion in multi-armed bandits. *CoRR* abs/1301.1936. http://arxiv.org/abs/1301.1936.

Sharif M, Bhagavatula S, Bauer L, Reiter M (2016) Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. 1528–1540. https://doi.org/10.1145/2976749.2978392.

Soma T, Yoshida Y (2020) Statistical learning with conditional value at risk. *ArXiv* abs/2002.05826.

South T (2020) War with robots: how battle bots will define the future of ground combat. Army Times. Accessed Mar 21, 2021, https://www.armytimes.com/news/your-army/2020/02/13/war-with-robots-how-battle-bots-will-define-the-future-of-ground-combat.

Szegedy C, Zaremba W, Sutskever I, Bruna J, Erhan D, Goodfellow I, Fergus R (2014) Intriguing properties of neural networks. 2nd International Conference on Learning Representations, ICLR 2014 ; Conference date: 14-04-2014 Through 16-04-2014.

Vapnik V (1992) Principles of risk minimization for learning theory. Moody J, Hanson S, Lippmann RP, eds., *Advances in Neural Information Processing Systems*, volume 4, https://proceedings.neurips.cc/paper/1991/file/ff4d5fbbafdf976cfdc032e3bde78de5--Paper.pdf (Morgan-Kaufmann).

Wang Y, Zou D, Yi J, Bailey J, Ma X, Gu Q (2020) Improving adversarial robustness requires revisiting misclassified examples. *ICLR 2020 : Eighth International Conference on Learning Representations*.

Williamson R, Menon A (2019) Fairness risk measures. *International Conference on Machine Learning*, 6786–6797 (PMLR).

Wong E, Rice L, Kolter JZ (2020) Fast is better than free: Revisiting adversarial training. *International Conference on Learning Representations*, https://openreview.net/forum?id=BJx040EFvH.

Wu D, Xia St, Wang Y (2020) Adversarial Weight Perturbation Helps Robust Generalization. *arXiv e-prints* arXiv:2004.05884.

Xie C, Wu Y, Maaten Lvd, Yuille AL, He K (2019) Feature denoising for improving adversarial robustness. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 501–509.

Xu W, Evans D, Qi Y (2018) Feature squeezing: Detecting adversarial examples in deep neural networks. *Proceedings 2018 Network and Distributed System Security Symposium.* http://dx.doi.org/10.14722/ndss.2018.23198.

Zagoruyko S, Komodakis N (2016) Wide residual networks. *CoRR* abs/1605.07146.

Zhang C, Pham M, Fu S, Liu Y (2018) Robust multicategory support vector machines using difference convex algorithm. *Mathematical Programming* 169:277–305.

Zhang H, Yu Y, Jiao J, Xing E, El Ghaoui L, Jordan M (2019) Theoretically principled trade-off between robustness and accuracy. *International Conference on Machine Learning*, 7472–7482 (PMLR).

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California