A Generative Model Based Adversarial Security of Deep Learning and Linear Classifier Models

Samed Sivaslioglu
Tubitak Bilgem, Kocaeli, Turkey
E-mail: samedsivaslioglu@gmail.com

Ferhat Ozgur Catak
Simula Research Laboratory, Fornebu, Norway
E-mail: ozgur@simula.no

Kevser Şahinbaş
Department of Management Information System, Istanbul Medipol University, Istanbul, Turkey
E-mail: ksahinbas@medipol.edu.tr

Keywords: adversarial machine learning, generative models, autoencoders

Received: July 13, 2020

In recent years, machine learning algorithms have been applied widely in various fields such as health, transportation, and the autonomous car. With the rapid developments of deep learning techniques, it is critical to take the security concern into account for the application of the algorithms. While machine learning offers significant advantages in terms of the application of algorithms, the issue of security is ignored. Since it has many applications in the real world, security is a vital part of the algorithms. In this paper, we have proposed a mitigation method for adversarial attacks against machine learning models with an autoencoder model that is one of the generative ones. The main idea behind adversarial attacks against machine learning models is to produce erroneous results by manipulating trained models. We have also presented the performance of autoencoder models to various attack methods from deep neural networks to traditional algorithms by using different methods such as non-targeted and targeted attacks to multi-class logistic regression, a fast gradient sign method, a targeted fast gradient sign method and a basic iterative method attack to neural networks for the MNIST dataset.

Povzetek: S pomočjo globokega učenja je analizirana varnost pri sistemih strojnega učenja.

1 Introduction

With the help of artificial intelligence technology, machine learning has been widely used in classification, decision making, voice and face recognition, games, financial assessment, and other fields [9, 12, 44, 45, 48]. The machine learning methods consider player's choices in the animation industry for games and analyze diseases to contribute to the decision-making mechanism [2, 6, 7, 15, 34, 46]. With the successful implementations of machine learning, attacks on the machine learning process and counter-attack methods and incrementing robustness of learning have become hot research topics in recent years [24, 27, 31, 37, 51]. The presence of negative data samples or an attack on the model can lead to producing incorrect results in the predictions and classifications even in the advanced models.

It is more challenging to recognize the attack because of using big data in machine learning applications compared to other cybersecurity fields. Therefore, it is essential to create components for machine learning that are resistant to this type of attack. In contrast, recent works have conducted in this area and demonstrated that the resistance is not very robust to attacks [10, 11]. These methods have shown success against a specific set of attack methods and have generally failed to provide complete and generic protection[43].

Machine learning models already used in functional forms could be vulnerable to these kinds of attacks. For instance, by putting some tiny stickers on the ground in a junction, researchers confirmed that they could provoke an autonomous car to make an unnatural decision and drive into the opposite lane [16]. In another study, the researchers have pointed out that making hidden modifications to an input image can fool a medical imaging operation into labelling a benign mole as malignant with 100% confidence [17].

Previous methods have shown success against a specific set of attack methods and have generally failed to provide complete and generic protection [14]. This field has been spreading rapidly, and, in this field, lots of dangers have attracted increasing attention from escaping the filters of unwanted and phishing e-mails, to poisoning the sensor data of a car or aircraft that drives itself [4, 41]. Disaster scenarios can occur if any precautions are not taken in these systems [30].

The main contribution of this work is to explore the autoencoder based generative models against adversarial machine learning attacks to the models. Adversarial Machine Learning has been used to study these attacks and reduce their effects [8, 32]. Previous works point out the fundamental equilibrium to design the algorithms and to create new algorithms and methods that are resistant and robust against attacks that will negatively affect this balance. However, most of these works have been implemented successfully for specific situations. In Section 3, we present some applications of these works.

This work aims to propose a method that not only presents a generic resistance to specific attack methods but also provides robustness to machine learning models in general. Our goal is to find an effective method that can be used by model trainers. For this purpose, we have processed the data with autoencoder before reaching to the machine learning model.

We have used non-targeted and targeted attacks to multiclass logistic regression machine learning models for observing the change and difference between attack methods as well as various attack methods to neural networks such as fast gradient sign method (FGSM), targeted fast gradient sign method (T-FGSM) and basic iterative method (BIM). We have selected MNIST dataset that consists of numbers from people's handwriting to provide people to understand and see changes in the data. In our previous works [3, 38], we applied the generative models both for data and model poisoning attacks with limited datasets.

The study is organized as follows. In Section 2, we first present the related works. In Section 3, we introduce several adversarial attack types, environments, and autoencoder. In Section 4, we present selection of autoencoder model, activation function and tuning parameters. In Section 5, we provide some observation on the robustness of autoencoder for adversarial machine learning with different machine learning algorithms and models. In Section 8, we conclude this study.

2 Related Work

In recent years, with the increase of the machine learning attacks, various studies have been proposed to create defensive measures against these attacks. Data sterility and learning endurance are recommended as countermeasures in defining a machine learning process [32]. They provide a model for classifying attacks against online machine learning algorithms. Most of the studies in these fields have been focused on specific adversarial attacks and generally, presented the theoretical discussion of adversarial machine learning area [23, 25].

Bo Li and Yevgeniy Vorobeychik present binary domains and classifications. In their work, the approach starts with mixed-integer linear programming (MILP) with constraint generation and gives suggestions on top of this. They also use the Stackelberg game multi-adversary model algorithm and the other algorithm that feeds back the generated adversarial examples to the training model, which is called as RAD (Retraining with Adversarial Examples) [28]. Their approach can scale thousands of features with RAD that showed robustness to several model erroneous specifications. On the other hand, their work is particular and works only in specific methods, even though it is presented as a general protection method. They have proposed a method that implements successful results. Similarly, Xiao et al. provide a method to increase the speed of resistance training against the rectified linear unit (RELU) [36]. They provide that optimizing weight sparseness enables us to turn computationally demanding validation problems into solvable problems. They showed that improving ReLU stability leads to 4-13x faster validation times. They use weight sparsity and RELU stability for robust verification. It can be said that their methodology does not provide a general approach.

Yu et al. propose a study that can evaluate the neural network's features under hostile attacks. In their study, the connection between the input space and hostile examples is presented. Also, the connection between the network strength and the decision surface geometry as an indicator of the hostile strength of the neural network is shown. By extending the loss surface to decision surface and other various methods, they provide adversarial robustness by decision surface. The geometry of the decision surface cannot be demonstrated most of the time, and there is no explicit decision boundary between correct or wrong prediction. Robustness can be increased by constructing a good model, but it can change with attack intensity [50]. Their method can increase network's intrinsic adversarial robustness against several adversarial attacks without involving adversarial training.

Mardy et al. investigate artificial neural networks resistant with adversity and increase accuracy rates with different methods, mainly with optimization and prove that there can be more robust machine learning models [43].

Pinto et al. provide a method to solve this problem with the supported learning method. In their study, they formulate learning as a zero-sum, minimax objective function. They present machine learning models that are more resistant to disturbances are hard to model during the training and are better affected by changes in training and test conditions. They generalize reinforced learning on machine learning models. They propose a "Robust Adversarial Reinforced Learning" (RARL), where they train an agent to operate in the presence of a destabilizing adversary that applies disturbance forces to the system. They presented that their method increased training stability, was robust to differences in training and testing conditions, and outperformed basically even in the absence of the adversary. However, in their work, Robust Adversarial Reinforced Learning may overfit itself, and sometimes it can miss predicting without any adversarial being in presence [39].

Carlini and Wagner propose a model that the self-logic and the strength of the machine learning model with a strong attack can be affected. They prove that these types of attacks can often be used to evaluate the effectiveness of potential defenses. They propose defensive distillation as a general-purpose procedure to increase robustness [11].

Harding et al. similarly investigate the effects of hostile samples produced from targeted and non-targeted attacks in decision making. They observed that non-targeted samples interfered more with human perception and classification decisions than targeted samples [22].

Bai et al. present a convolutional autoencoder model with the adversarial decoders to automate the generation of adversarial samples. They produce adversary examples by a convolutional autoencoder model. They use pooling computations and sampling tricks to achieve these results. After this process, an adversarial decoder automates the generation of adversarial samples. Adversarial sampling is useful, but it cannot provide adversarial robustness on its own, and sampling tricks are too specific [5]. They gain a net performance improvement over the normal CNN.

Sahay et al. propose FGSM attack and use an autoencoder to denoise the test data. They have also used an autoencoder to denoise the test data, which is trained with both corrupted and healthy data. Then they reduce the dimension of the denoised data. These autoencoders are specifically designed to compress data effectively and reduce dimensions. Hence, it may not be wholly generalized, and training with corrupted data requires a lot of adjustments to get better test results [33]. Their model provide that when test data is preprocessed using this cascading, the tested deep neural network classifier provides much higher accuracy, thus mitigating the effect of the adversarial perturbation.

I-Ting Chen et al. also provide with FGSM attack on denoising autoencoders. They analyze the attacks from the perspective that attacks can be applied stealthily. They use autoencoders to filter data before applied to the model and compare it with the model without an autoencoder filter. They use autoencoders mainly focused on the stealth aspect of these attacks and used them specifically against FGSM with specific parameters [13]. They enhance the classification accuracy from 2.92% to 75.52% for the neural network classifier on the 10 digits and from 4.12% to 93.57% for the logistic regression classifier on digit 3s and 7s.

Gondim-Ribeiro et al. propose autoencoders attacks. In their work, they attack 3 types of autoencoders: Simple variational autoencoders, convolutional variational autoencoders, and DRAW (Deep Recurrent AttentiveWriter). They propose to scheme an attack on autoencoders. As they accept that "No attack can both convincingly reconstruct the target while keeping the distortions on the input imperceptible.". They enable both DRAW's recurrence and attention mechanism to lead to better resistance. Automatic encoders are recommended to compress data and more attention should be given to adversarial attacks on them. This method cannot be used to achieve robustness against adversarial attacks [40].

Table 2 shows the strength and the weakness of the each paper.

3 Preliminaries

In this section, we consider attack types, data poisoning attacks, model attacks, attack environments, and autoencoder.

3.1 Attack Types

Machine Learning attacks can be categorized into data poisoning attacks and model attacks. The difference between the two attacks lies in the influencing type. Data poisoning attacks mainly focus on influencing the data, while model evasion attacks influencing the model for desired attack outcomes. Both attacks aim to disrupt the machine learning structure, evasion from filters, causing wrong predictions, misdirection, and other problems for the machine learning process. In this paper, we mainly focus on machine learning model attacks.

3.1.1 Data Poisoning Attacks

According to machine learning methods, algorithms are trained and tested with datasets. Data poisoning in machine learning algorithms has a significant impact on a dataset and can cause problems for algorithm and confusion for developers. With poisoning the data, adversaries can compromise the whole machine learning process. Hence, data poisoning can cause problems in machine learning algorithms.

3.1.2 Model Attacks

Machine learning model attacks have been applied mostly in adversarial attacks, and evasion attacks being have been used most extensively in this category. For spam emails, phishing attacks, and executing malware code, adversaries apply model evasion attacks. There are also some benefits to adversaries in misclassification and misdirection. In this type of attack, the attacker does not change training data but disrupts or changes its data and diverse this data from the training dataset or make this data seem safe. This study mainly concentrates on model attacks.

3.2 Attack Environments

There are two significant threat models for adversarial attacks: the white-box and black-box models.

3.2.1 White Box Attacks

Under the white-box setting, the internal structure, design, and application of the tested item are accessible to the adversaries. In this model, attacks are based on an analysis of the internal structure. It is also known as open box attacks. Programming knowledge and application knowledge

Table 1: Related Work Summary

Research Study	Strength	Weakness
Adversarial Machine Learning [32]	Introduces the emerging field of Adversarial Machine Learning.	Discusses the countermeasures against attacks without suggesting a method.
Evasion-Robust Classification on Binary Domains [28]	Demonstrates some methods that can be used on Binary Domains, which are based on MILP.	Very specific about the robustness, even though it is presented as a general method.
Training for Faster Adversarial Robustness Verification via Inducing ReLU Stability [36]	Using weight sparsity and RELU stability for robust verification.	Does not provide a general approach, or universality as it is suggested in paper.
Interpreting Adversarial Robustness: A View from Decision Surface in Input Space [50]	By extending the loss surface to decision surface and other various methods, they provide adversarial robustness by decision surface.	The geometry of the decision surface cannot be shown most of the times and there is no explicit decision boundary between correct or wrong prediction. Robustness can be increased by constructing a good model but it can change with attack intensity.
Robust Adversarial Reinforcement Learning [39]	They have tried to generalize reinforced learning on machine learning models. They suggested a Robust Adversarial Reinforced Learning (RARL) where they have trained an agent to operate in the presence of a destabilizing adversary that applies disturbance forces to the system.	Robust Adversarial Reinforced Learning may overfit itself and sometimes it may mispredict without any adversarial being in presence.
Alleviating Adversarial Attacks via Convolutional Autoencoder [5]	They have produced adversary examples via a convolutional autoencoder model. Pooling computations and sampling tricks are used. Then an adversarial decoder automate the generation of adversarial samples.	Adversarial sampling is useful but it cannot provide adversarial robustness on its own. Sampling tricks are also too specified.
Combatting Adversarial Attacks through Denoising and Dimensionality Reduction: A Cascaded Autoencoder Approach [33]	They have used an autoencoder to denoise the test data which is trained with both corrupted and normal data. Then they reduce the dimension of the denoised data.	Autoencoders specifically designed to compress data effectively and reduce dimensions. Therefore it may not be completely generalized and training with corrupted data requires a lot of adjustments for test results.
A Comparative Study of Autoencoders against Adversarial Attacks [13]	They have used autoencoders to filter data before applying into the model and compare it with the model without autoencoder filter.	They have used autoencoders mainly focused on the stealth aspect of these attacks and use them specifically against FGSM with specific parameters.
Adversarial Attacks on Variational Autoencoders [40]	They propose a scheme to attack on autoencoders and validate experiments to three autoencoder models: Simple, convolutional and DRAW (Deep Recurrent Attentive Writer).	As they have accepted "No attack can both convincingly reconstruct the target while keeping the distortions on the input imperceptible.". it cannot provide robustness against adversarial attacks.
Understanding Autoencoders with Information Theoretic Concepts [47]	They examine data processing inequality with stacked autoencoders and two types of information planes with autoencoders. They have analyzed DNNs learning from a joint geometric and information theoretic perspective, thus emphasizing the role that pair-wise mutual information plays important role in understanding DNNs with autoencoders.	The accurate and tractable estimation of information quantities from large data seems to be a problem due to Shannon's definition and other information theories are hard to estimate, which severely limits its powers to analyze machine learning algorithms.
Adversarial Attacks and Defences Competition [42]	Google Brain organized NIPS 2017 to accelerate research on adversarial examples and robustness of machine learning classifiers. Alexey Kurakin and Ian Goodfellow et al. present some of the structure and organization of the competition and the solutions developed by several of the top-placing teams.	We experimented with the proposed methods of this competition bu these methods do not provide a generalized solution for the robustness against adversarial machine learning model attacks.
Explaining And Harnessing Adversarial Examples [19]	Ian Goodfellow et al. makes considerable observations about Gradient-based optimization and introduce FGSM.	Models may mislead for the efficiency of optimization. The paper focuses explicitly on identifying similar types of problematic points in the model.

are essential. White-box tests provide a comprehensive assessment of both internal and external vulnerabilities and are the best choice for computational tests.

3.2.2 Black Box Attacks

In the black-box model, internal structure and software testing are secrets to the adversaries. It is also known as behavioral attacks. In these tests, the internal structure does not have to be known by the tester. They provide a comprehensive assessment of errors. Without changing the learning process, black box attacks provide changes to be observed as external effects on the learning process rather than changes in the learning algorithm. In this study, the main reason behind the selection of this method is the observation of the learning process.

3.3 Autoencoder

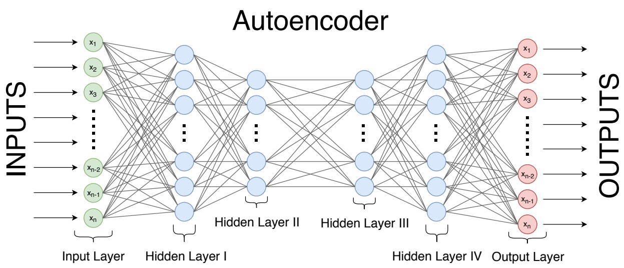


Figure 1: Autoencoder Layer Structure

An autoencoder neural network is an unsupervised learning algorithm that takes inputs and sets target values to be equals of the input values [47]. Autoencoders are generative models that apply backpropagation. They can work without the results of these inputs. While the use of a learning model is in the form of `model.fit(X,Y)`, autoencoders work as `model.fit(X,X)`. The autoencoder works with the ID function to get the output $\mathbf{x}$ that corresponds to $\mathbf{x}$ entries. The identity function seems to be a particularly insignificant function to try to learn; however, there is an interesting structure related to the data, putting restrictions such as limiting the number of hidden units on the network[47]. They are neural networks which work as neural networks with an input layer, hidden layers and an output layer but instead of predicting Y as in `model.fit(X,Y)`, they reconstruct X as in `model.fit(X,X)`. Due to this reconstruction being unsupervised, autoencoders are unsupervised learning models. This structure consists of an encoder and a decoder part. We will define the encoding transition as ϕ and decoding transition as ψ.

$\phi : X \to F$
$\psi : F \to X$
$\phi, \psi = argmin_{\phi,\psi} ||X - (\psi \circ \phi)X||^2$

With one hidden layer, encoder will take the input $x \in \mathbb{R}^d = \chi$ and map it to $h \in \mathbb{R}^p = F$. The h below is referred to as latent variables. σ is an activation function such as ReLU or sigmoid which were used in this study[1, 20]. b is bias vector, W is weight matrix which both are usually initialized randomly then updated iteratively through training[35].

$h = \sigma(Wx + b)$

After the encoder transition is completed, decoder transition maps h to reconstruct x'.

$x' = \sigma'(W'h + b')$ where σ', W', b' of decoder are unrelated to σ, W, b of encoder. Loss of autoencoders are trained to be minimal, showed as L below.

$L(x, x') = ||x-x'||^2 = ||x-\sigma'(W'(\sigma(Wx+b))+b')||^2$

So the loss function shows the reconstruction errors, which need to be minimal. After some iterations with input training set x is averaged.

In conclusion, autoencoders can be seen as neural networks that reconstruct inputs instead of predicting them. In this paper, we will use them to reconstruct our dataset inputs.

4 System Model

This section presents the selection of autoencoder model, activation function, and tuning parameters.

4.1 Creating Autoencoder Model

In this paper, we have selected the MNIST dataset to observe changes easily. Therefore, the size of the layer structure in the autoencoder model is selected as 28 and multipliers to match the MNIST datasets, which represents the numbers by 28 to 28 matrixes. Figure 2 presents the structure of matrixes. The modified MNIST data with autoencoder is presented in Figure 3. In the training of the model, the encoded data is used instead of using the MNIST datasets directly. As a training method, a multi-class logistic regression method is selected, and attacks are applied to this model. We train autoencoder for 35 epochs. Figure 4 provides the process diagram.

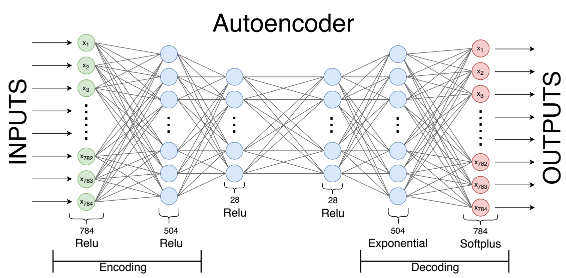


Figure 2: Autoencoder Activation Functions. Note that layer sizes given according to the dataset which is MNIST dataset

4.2 Activation Function Selection

In machine learning and deep learning algorithms, the activation function is used for the computations between

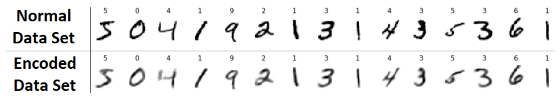


Figure 3: Normal and Encoded Data Set of MNIST

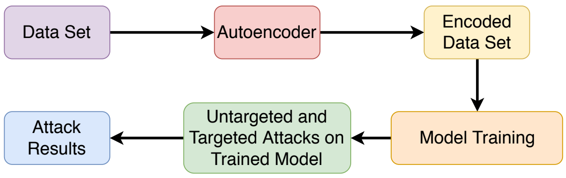
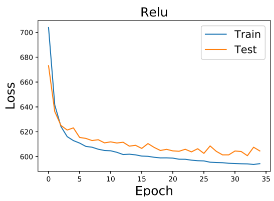
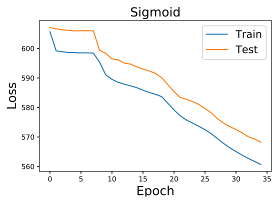


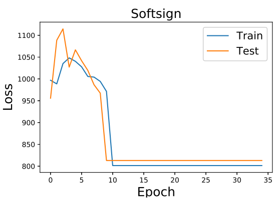
Figure 4: Process Diagram



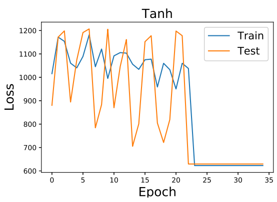
(a) Relu Loss History



(b) Sigmoid Loss History



(c) Softsign Loss History



(d) Tanh Loss History

Figure 5: Loss histories of different activation functions

hidden and output layers[18]. The loss values are compared with different activation functions. Figure 5 indicates the comparison results of loss value. `Sigmoid` and `ReLU` have the best performance among these values and gave the best results. `Sigmoid` has more losses at lower epochs than `ReLU`, but it has better results. Therefore, it is aimed to reach the best result of activation function in both layers. The model with the least loss value is to make the coding parts with the `ReLU` function and to use the `exponential` and `softplus` functions in the analysis part respectively. These functions are used in our study. Figure 6 illustrates the result of the loss function, and Figure 2 presents the structure of the model with the activation functions.

4.3 Tuning Parameters

The tuning parameters for autoencoders depend on the dataset we use and what we try to apply. As previously mentioned, `ReLU` and sigmoid function are selected to be activation function for our model [1, 18]. `ReLU` is the activation function through the whole autoencoder while exponential is the softplus being the output layer's activation function which yields the minimal loss. Figure 2 presents

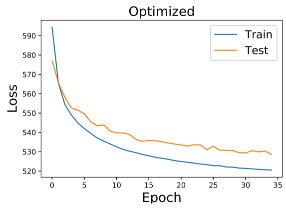


Figure 6: Optimized Relu Loss History

the input size as 784 due to our dataset and MNIST dataset contains 28x28 pixel images[29]. Encoding part for our autoencoder size is $784 \times 504 \times 28$ and decoding size is $28 \times 504 \times 784$.

This structure is selected by the various neural network structures that take the square of the size of the matrix, lower it, and give it to its dimension size lastly. The last hidden layer of the decoding part with the size of 504 uses `exponential` activation function, and an output layer with the size of 784 uses `softplus` activation function [14, 21]. We used `adam` optimizer with categorical crossentropy[26, 49]. We see that a small number is enough for training, so we select epoch number for autoencoder as 35. This is the best epoch value to get meaningful results for both models with autoencoder and without autoencoder to see accuracy. In lower values, models get their accuracy scores too low for us to see the difference between them, even though some models are structurally stronger than others.

5 Experiments with MNIST Dataset

5.1 Introduction

We examine the robustness of autoencoder for adversarial machine learning with different machine learning algorithms and models to see that autoencoding can be a generalized solution and an easy to use defense mechanism for most adversarial attacks. We use various linear machine learning model algorithms and neural network model algorithms against adversarial attacks.

5.2 Autoencoding

In this section, we look at the robustness provided with auto-encoding. We select a linear model and a neural network model to demonstrate this effectiveness. In these models, we also observe the robustness of different attack methods. We also use the MNIST dataset for these examples.

5.2.1 Multi-Class Logistic Regression

In linear machine learning model algorithms, we use mainly two attack methods: Non-Targeted and Targeted Attacks. The non-targeted attack does not concern with how the machine learning model makes its predictions and

 Informatica **45** (2021) 33–64

		Predicted Values									
		0	1	2	3	4	5	6	7	8	9
Actual Values	0	973	0	4	0	1	2	9	1	4	6
	1	0	1127	3	0	0	0	3	5	1	2
	2	2	6	1016	4	3	0	2	10	4	1
	3	0	0	2	1002	0	10	0	5	3	4
	4	0	0	3	0	966	0	1	0	0	8
	5	0	0	0	1	0	869	3	0	1	2
	6	1	1	0	0	1	5	938	0	1	0
	7	1	0	4	0	1	1	0	999	3	9
	8	3	1	0	3	2	3	1	2	953	3
	9	0	0	0	0	8	2	1	6	4	974

Figure 7: Confusion matrix of the model without any attack and without autoencoder

		Predicted Values									
		0	1	2	3	4	5	6	7	8	9
Actual Values	0	973	0	4	0	1	2	9	1	4	6
	1	0	1127	3	0	0	0	3	5	1	2
	2	2	6	1016	4	3	0	2	10	4	1
	3	0	0	2	1002	0	10	0	5	3	4
	4	0	0	3	0	966	0	1	0	0	8
	5	0	0	0	1	0	869	3	0	1	2
	6	1	1	0	0	1	5	938	0	1	0
	7	1	0	4	0	1	1	0	999	3	9
	8	3	1	0	3	2	3	1	2	953	3
	9	0	0	0	0	8	2	1	6	4	974

Figure 8: Confusion matrix of the model without any attack and with autoencoder

tries to force the machine learning model into misprediction. On the other hand, targeted attacks focus on leading some correct predictions into mispredictions. We have three methods for targeted attacks: Natural, Non-Natural, and one selected target. Firstly, natural targets are derived from the most common mispredictions made by the machine learning model. For example, guessing number 5 as 8, and number 7 as 1 are common mispredictions. Natural targets take these non-targeted attack results into account and attack directly to these most common mispredictions. So, when number 5 is seen, an attack would try to make it guessed as number 8. Secondly, non-natural targeted attacks are the opposite of natural targeted attacks. It takes the minimum number of mispredictions made by the machine learning model with the feedback provided by non-natural attacks. For example, if number 1 is least mispredicted as 0, the non-natural target for number 1 is 0. Therefore, we can see that how much the attack affects the machine learning model beyond its common mispredictions. Lastly, one targeted attack focuses on some random numbers. The aim is to make the machine learning model mispredict the same number for all numbers. For linear classifications, we select multi-class logistic regression to analyze the attacks. Because we do not interact with these linear classification algorithms aside from calling their defined functions from scikit-learn library, we use a blackbox environment for these attacks. In our study, the attack method against multi-class classification models developed in NIPS 2017 is used [42]. An epsilon value is used to determine the severity of the attack, which we select 50 in this study to demonstrate the results better. We apply a non-targeted attack to a multi-class logistic regression trained model which is trained with MNIST dataset without an autoencoder. The confusion matrix of this attack is presented in 9.

The findings from Figure 9 and 10 show that an autoencoder model provides robustness against non-targeted attacks. The accuracy value change with epsilon is presented in Figure 13. Figure 11 illustrates the change and perturbation of the selected attack with epsilon value as 50.

We apply a non-targeted attack on the multi-class logistic regression model with autoencoder and without autoencoder. Figure 13 provides a difference in accuracy metric. The detailed graph of the non-targeted attack on the model with autoencoder is presented in Figure 14. The changes in the MNIST dataset after autoencoder is provided in Fig-

		Predicted Values									
		0	1	2	3	4	5	6	7	8	9
Actual Values	0	247	0	17	51	8	73	32	20	8	7
	1	0	0	34	8	0	8	0	15	30	13
	2	32	18	69	37	181	24	251	288	191	255
	3	49	174	222	8	128	106	25	193	489	141
	4	4	0	34	49	14	57	59	29	10	231
	5	509	58	56	154	43	9	502	110	55	172
	6	45	0	93	35	68	109	4	5	25	1
	7	23	210	48	22	33	26	43	26	52	1
	8	51	678	366	586	31	378	23	141	0	189
	9	47	13	60	76	469	60	25	194	137	0

Figure 9: Confusion matrix of non-targeted attack to model without autoencoder

		Predicted Values									
		0	1	2	3	4	5	6	7	8	9
Actual Values	0	987	0	7	8	0	13	0	1	5	4
	1	0	1137	8	0	1	1	0	4	4	5
	2	0	0	958	2	4	2	0	15	0	0
	3	0	0	9	886	3	52	1	3	13	9
	4	0	0	3	4	923	11	0	10	1	28
	5	0	0	0	24	1	643	0	0	0	0
	6	5	0	5	2	3	28	962	2	2	0
	7	0	0	7	0	1	1	0	932	5	4
	8	2	5	31	72	1	116	0	12	944	9
	9	1	0	3	14	35	13	0	54	8	931

Figure 10: Confusion matrix of non-targeted attack to model with autoencoder

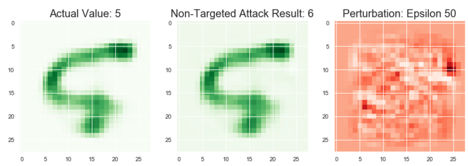


Figure 11: Value change and perturbation of a non-targeted attack on model without autoencoder

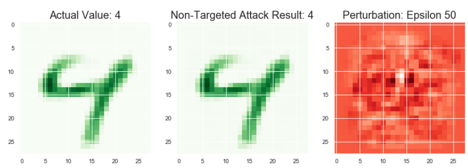


Figure 12: Value change and perturbation of a non-targeted attack on model with autoencoder

ure 3. The value change and perturbation of an epsilon 50 value on data are indicated in Figure 12.

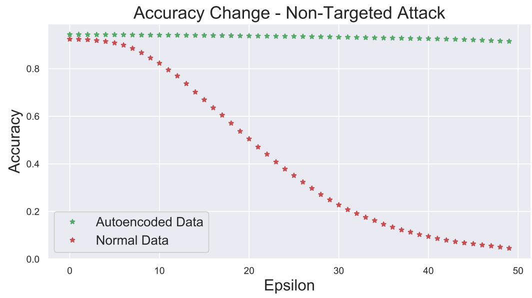


Figure 13: Comparison of accuracy with and without autoencoder for non-targeted attack

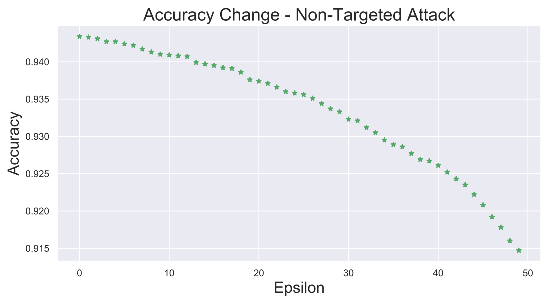


Figure 14: Details of accuracy with autoencoder for non-targeted attack

The following process is presented in Figure 4. In the examples with the autoencoder, data is passed through the autoencoder and then given to the training model, in our current case a classification model with multi-class logistic regression. Multi-class logistic regression uses the encoded dataset for training. Figure 10 provides to see improvement as a confusion matrix. For the targeted attacks, we select three methods to use. The first one is natural targets for MNIST dataset, which is also defined in NIPS 2017 [42]. Natural targets take the non-targeted attack results into account and attack directly to these most common mispredictions. For example, the natural target for number 3 is 8. When we apply the non-targeted attack, we obtain these results. Heat map for these numbers is indicated in Figure 77.

The second method of targeted attacks is non-natural targets which is the opposite of natural targets. We select the least mis predicted numbers as the target. These numbers is indicated as the heat map in Figure 77. The third method is the selection one number and making all numbers predict it. We randomly choose 7 as that target number. Targets for these methods are presented in Figure 16. The confusion matrixes for these methods are presented below.

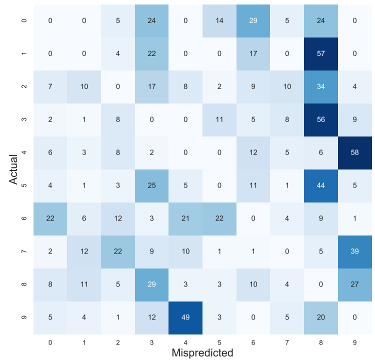


Figure 15: Heatmap of actual numbers and mispredictions

Natural Targets	Actual Numbers	0	1	2	3	4	5	6	7	8	9
	Target Numbers	6	8	8	8	9	8	0	9	3	4
Non-Natural Targets	Actual Numbers	0	1	2	3	4	5	6	7	8	9
	Target Numbers	1	0	0	1	1	1	1	6	0	6
One Number Targeted	Actual Numbers	0	1	2	3	4	5	6	7	8	9
	Target Numbers	7	7	7	7	7	7	7	7	7	7

Figure 16: Actual numbers and their target values for each targeted attack method

		Predicted Values									
		0	1	2	3	4	5	6	7	8	9
Actual Values	0	291	0	10	9	1	5	10	16	1	1
	1	0	0	1	0	0	0	0	2	0	0
	2	1	7	70	14	3	1	10	806	25	27
	3	6	10	46	45	7	38	6	17	786	9
	4	9	6	11	10	84	11	13	23	8	920
	5	680	3	22	21	5	49	559	15	29	0
	6	1	0	40	3	8	8	329	2	1	0
	7	0	0	4	1	6	1	3	18	3	0
	8	18	1124	783	917	17	735	26	41	130	17
	9	1	1	12	6	844	2	8	81	14	36

Figure 17: Confusion matrix of **natural targeted attack** to model without autoencoder

		Predicted Values									
		0	1	2	3	4	5	6	7	8	9
Actual Values	0	989	0	2	1	0	6	7	1	0	1
	1	0	1105	2	0	0	1	0	0	0	0
	2	0	0	979	4	0	1	0	2	0	0
	3	0	0	0	972	0	12	0	1	4	32
	4	0	0	0	0	889	1	2	1	0	0
	5	0	0	0	0	0	713	0	0	0	0
	6	3	0	3	0	1	8	969	0	1	0
	7	0	0	6	1	0	0	0	943	0	11
	8	3	29	35	46	3	134	2	1	914	6
	9	1	3	1	2	77	2	2	57	44	964

Figure 18: Confusion matrix of **natural targeted attack** to model with autoencoder

5.2.2 Neural Networks

We use neural networks with the same principles as multi-class logistic regressions and make attacks to the machine

Actual Values \ Predicted Values	0	1	2	3	4	5	6	7	8	9
0	735	147	281	41	8	36	31	29	694	12
1	3	7	22	565	134	259	105	26	34	39
2	29	88	200	53	107	15	214	170	135	22
3	37	59	96	71	41	95	9	136	59	19
4	3	0	16	8	224	42	53	37	3	362
5	83	0	5	31	1	2	107	14	5	4
6	72	8	99	24	103	110	422	39	28	380
7	5	100	22	6	7	7	6	156	6	0
8	33	741	246	195	30	258	13	104	22	163
9	7	1	12	32	320	26	4	310	11	9

Figure 19: Confusion matrix of **non-natural targeted attack** to model without autoencoder

Actual Values \ Predicted Values	0	1	2	3	4	5	6	7	8	9
0	994	0	1	0	0	7	0	0	0	0
1	0	1147	0	2	0	6	0	4	2	1
2	2	1	991	0	0	6	2	2	30	0
3	0	0	4	992	0	71	0	5	2	1
4	0	0	0	0	973	4	0	5	0	1
5	0	0	7	0	1	597	1	1	4	0
6	2	0	3	0	2	32	964	1	0	1
7	0	0	0	0	1	1	0	1001	0	0
8	3	1	5	5	0	170	1	5	917	8
9	0	0	1	3	0	8	0	6	0	992

Figure 20: Confusion matrix of **non-natural targeted attack** to model with autoencoder

Actual Values \ Predicted Values	0	1	2	3	4	5	6	7	8	9
0	281	0	17	14	1	27	17	0	1	0
1	0	0	9	0	0	0	0	0	0	0
2	0	0	69	0	1	2	32	0	1	0
3	16	12	330	109	2	132	46	0	96	0
4	1	0	7	4	36	22	16	0	1	1
5	69	0	9	12	0	13	165	0	6	0
6	5	0	38	4	0	27	164	0	3	0
7	612	1114	372	778	828	406	479	1021	731	1005
8	6	25	116	61	0	139	21	0	28	0
9	17	0	32	44	107	82	24	0	130	4

Figure 21: Confusion matrix of **one number targeted attack** to model without autoencoder

Actual Values \ Predicted Values	0	1	2	3	4	5	6	7	8	9
0	991	0	3	0	0	8	0	0	0	0
1	0	1139	7	0	0	1	0	0	3	0
2	0	0	955	0	0	0	0	0	0	0
3	0	0	20	991	0	33	1	0	7	0
4	1	0	4	0	947	4	1	0	1	1
5	0	0	0	0	0	775	0	0	0	0
6	0	0	5	0	0	11	960	0	0	0
7	2	3	20	18	25	2	1	1033	19	104
8	0	0	15	0	0	38	0	0	945	0
9	1	0	2	3	0	8	0	0	7	885

Figure 22: Confusion matrix of **one number targeted attack** to model with autoencoder

learning model. We use the same structure, layer, activation functions and epochs for these neural networks as we use in our autoencoder for simplicity. Although this robustness will work with other neural network structures, we will not demonstrate them in this study due to structure designs that can vary for all developers. We also compare the results of these attacks with both the data from the MNIST dataset

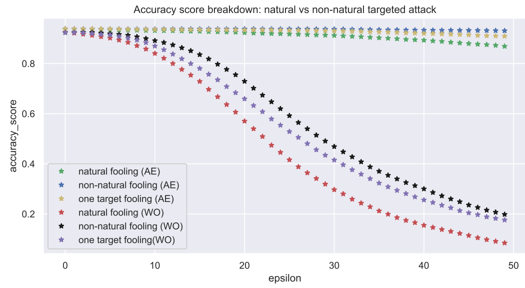


Figure 23: Comparison of accuracy with and without autoencoder for targeted attacks. *AE stands for the models with autoencoder, WO stands for models without autoencoder*

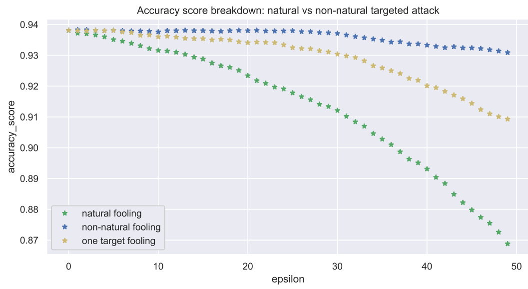


Figure 24: Details of accuracy with autoencoder for targeted attacks

and the encoded data results of the MNIST dataset. As for attack methods, we select three methods: FGSM, T-FGSM and BIM. Cleverhans library is used for providing these attack methods to the neural network, which is from the Keras library.

We examine the differences between the neural network model that has autoencoder and the neural network model that takes data directly from the MNIST dataset with confusion matrixes and classification reports. Firstly, our model without autoencoder gives the following results, as seen in Figure 25 for the confusion matrix and the classification report. The results with the autoencoder are presented in Figure 26. Note that these confusion matrixes and classification reports are indicated before any attack.

Fast Gradient Sign Method:

There is a slight difference between the neural network models with autoencoder and without autoencoder model. We apply the FGSM attack on both methods. The method uses the gradients of the loss accordingly for creating a new image that maximizes the loss. We can say the gradients are generated accordingly to input images. For these reasons, the FGSM causes a wide variety of models to misclassify their input [19].

As we expect due to results from multi-class logistic regression, autoencoder gives robustness to the neural network model too. After the DGSM, the neural network without an autoencoder suffers an immense drop in its accuracy,

		Predicted Values									
		0	1	2	3	4	5	6	7	8	9
Actual Values	0	974	0	5	0	1	2	7	1	5	5
	1	0	1125	4	0	0	0	3	4	0	2
	2	0	1	1003	1	0	0	2	8	1	0
	3	1	4	3	1004	0	7	0	2	0	1
	4	0	0	3	0	970	0	7	0	3	14
	5	0	0	0	1	0	873	4	0	1	2
	6	1	2	0	0	4	5	930	0	0	0
	7	2	0	8	0	1	2	0	1005	4	8
	8	2	3	6	4	1	2	5	3	957	4
	9	0	0	0	0	5	1	0	5	3	973

	Precision	Recall	F1-Score	Support
0	0.99	0.97	0.98	1000
1	0.99	0.99	0.99	1138
2	0.97	0.99	0.98	1016
3	0.99	0.98	0.99	1022
4	0.99	0.97	0.98	997
5	0.98	0.99	0.98	881
6	0.97	0.99	0.98	942
7	0.98	0.98	0.98	1030
8	0.98	0.97	0.98	987
9	0.96	0.99	0.97	987
Micro Avg	0.98	0.98	0.98	10000
Macro Avg	0.98	0.98	0.98	10000
Weighted Avg	0.98	0.98	0.98	10000

Figure 25: Confusion matrix and classification report of the neural network model without autoencoder

		Predicted Values									
		0	1	2	3	4	5	6	7	8	9
Actual Values	0	966	0	4	0	0	2	3	0	3	3
	1	0	1122	2	1	1	0	2	6	2	3
	2	3	3	1013	7	3	0	2	13	2	2
	3	0	0	3	982	0	5	0	8	7	2
	4	0	0	1	1	954	1	7	0	4	6
	5	1	2	0	10	0	874	4	1	5	6
	6	4	4	1	1	2	5	937	0	2	0
	7	1	1	4	3	8	2	0	990	2	8
	8	3	3	3	3	3	2	3	2	945	7
	9	2	0	1	2	11	1	0	8	2	972

	Precision	Recall	F1-Score	Support
0	0.99	0.98	0.99	981
1	0.99	0.99	0.99	1139
2	0.98	0.97	0.97	1048
3	0.97	0.98	0.97	1007
4	0.97	0.98	0.98	974
5	0.98	0.97	0.97	903
6	0.98	0.98	0.98	956
7	0.96	0.97	0.97	1019
8	0.97	0.97	0.97	974
9	0.96	0.97	0.97	999
Micro Avg	0.98	0.98	0.98	10000
Macro Avg	0.98	0.98	0.98	10000
Weighted Avg	0.98	0.98	0.98	10000

Figure 26: Confusion matrix and classification report of the neural network model with autoencoder

and the FGSM works as intended. But the neural network model with autoencoder only suffers a 0.01 percent accuracy drop.

Targeted Fast Gradient Sign Method: There is a directed type of FGSM, called T-FGSM. It uses the same principles to maximize the loss of the target. In this method, a gradient step is computed for giving the same misprediction for different inputs.

In the confusion matrix, the target value for this attack is number 5. The neural network model with the autoencoder is still at the accuracy of 0.98. The individual differences

		Predicted Values									
		0	1	2	3	4	5	6	7	8	9
Actual Values	0	80	1	42	7	16	11	68	7	24	14
	1	2	5	125	5	34	6	20	16	30	11
	2	177	127	73	120	43	3	47	95	264	5
	3	19	13	344	50	7	337	54	504	234	171
	4	17	538	35	2	85	1	356	47	18	295
	5	68	2	6	351	1	99	177	3	185	63
	6	275	8	9	0	32	70	71	0	38	1
	7	20	215	177	64	228	7	7	40	48	318
	8	109	223	206	303	69	253	154	68	16	105
	9	213	3	15	108	467	105	4	248	117	26

	Precision	Recall	F1-Score	Support
0	0.08	0.30	0.13	270
1	0.00	0.02	0.01	254
2	0.07	0.08	0.07	954
3	0.05	0.03	0.04	1733
4	0.09	0.06	0.07	1394
5	0.11	0.10	0.11	955
6	0.07	0.14	0.10	504
7	0.04	0.04	0.04	1124
8	0.02	0.01	0.01	1506
9	0.03	0.02	0.02	1306
Micro Avg	0.05	0.05	0.05	10000
Macro Avg	0.06	0.08	0.06	10000
Weighted Avg	0.05	0.05	0.05	10000

Figure 27: Confusion matrix and classification report of the neural network model without autoencoder after FGSM attack

		Predicted Values									
		0	1	2	3	4	5	6	7	8	9
Actual Values	0	966	0	5	0	0	2	2	1	3	2
	1	0	1122	3	0	3	0	2	5	1	3
	2	3	2	1009	8	4	0	2	11	2	3
	3	0	0	4	980	0	5	0	9	7	2
	4	0	1	1	1	956	2	8	1	4	8
	5	1	2	0	11	0	872	3	1	7	5
	6	4	4	1	1	2	5	939	0	2	0
	7	1	2	4	3	5	2	0	988	2	8
	8	3	2	4	4	2	3	2	2	942	8
	9	2	0	1	2	10	1	0	10	4	970

	Precision	Recall	F1-Score	Support
0	0.99	0.98	0.99	981
1	0.99	0.99	0.99	1139
2	0.98	0.97	0.97	1044
3	0.97	0.97	0.97	1007
4	0.97	0.97	0.97	982
5	0.98	0.97	0.97	902
6	0.98	0.98	0.98	958
7	0.96	0.97	0.97	1015
8	0.97	0.97	0.97	972
9	0.96	0.97	0.97	1000
Micro Avg	0.97	0.97	0.97	10000
Macro Avg	0.97	0.97	0.97	10000
Weighted Avg	0.97	0.97	0.97	10000

Figure 28: Confusion matrix and classification report of the neural network model with autoencoder after FGSM attack

are presented when compare with Figure 26.

Basic Iterative Method:

BIM is an extension of FGSM to apply it multiple times with iterations. It provides the recalculation of a gradient attack for each iteration.

This is the most damaging attack for the neural network model that takes its inputs directly from the MNIST Dataset without an autoencoder. The findings from Figure 31 show that the accuracy drops between 0.01 and 0.02 percent. The neural network model with autoencoder's ac-

Actual Values \ Predicted Values	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	0	1	3	0
2	0	0	0	0	1	0	0	23	7	39
3	8	6	180	0	59	1	8	7	6	65
4	0	0	0	0	0	0	0	0	0	0
5	972	1119	844	1004	906	890	947	982	956	871
6	0	0	0	0	1	0	0	0	0	12
7	0	1	2	0	5	0	0	0	1	20
8	0	9	6	6	0	0	1	15	1	2
9	0	0	0	0	10	0	2	0	0	0

	Precision	Recall	F1-Score	Support
0	0.00	0.00	0.00	1
1	0.00	0.00	0.00	4
2	0.00	0.00	0.00	70
3	0.00	0.00	0.00	340
4	0.00	0.00	0.00	0
5	1.00	0.09	0.17	9491
6	0.00	0.00	0.00	13
7	0.00	0.00	0.00	29
8	0.00	0.03	0.00	40
9	0.00	0.00	0.00	12
Micro Avg	0.09	0.09	0.09	10000
Macro Avg	0.10	0.01	0.02	10000
Weighted Avg	0.95	0.09	0.16	10000

Figure 29: Confusion matrix and classification report of the neural network model without autoencoder after T-FGSM attack

Actual Values \ Predicted Values	0	1	2	3	4	5	6	7	8	9
0	965	0	4	0	0	2	3	0	3	3
1	0	1123	2	1	1	0	2	7	0	3
2	3	2	1013	7	3	0	1	13	2	2
3	0	0	4	981	0	2	0	7	7	2
4	0	0	1	0	958	2	8	0	4	6
5	1	2	0	14	0	878	7	1	10	6
6	4	4	0	0	2	5	934	0	2	0
7	2	1	3	3	6	1	0	989	2	7
8	3	3	4	3	1	1	3	2	942	6
9	2	0	1	1	11	1	0	9	2	974

	Precision	Recall	F1-Score	Support
0	0.98	0.98	0.98	980
1	0.99	0.99	0.99	1139
2	0.98	0.97	0.97	1046
3	0.97	0.98	0.97	1003
4	0.98	0.98	0.98	979
5	0.98	0.96	0.97	919
6	0.97	0.98	0.98	951
7	0.96	0.98	0.97	1014
8	0.97	0.97	0.97	968
9	0.97	0.97	0.97	1001
Micro Avg	0.98	0.98	0.98	10000
Macro Avg	0.98	0.98	0.98	10000
Weighted Avg	0.98	0.98	0.98	10000

Figure 30: Confusion matrix and classification report of the neural network model with autoencoder after T-FGSM attack

curacy stays as 0.97 percent, losing only 0.1 percent.

Findings indicate that autoencoding before giving dataset as input to linear models and neural network models improve robustness against adversarial attacks significantly. We use vanilla autoencoders. They are the basic autoencoders without modification. In the other sections, we apply the same attacks with the same machine learning models with different autoencoder types.

Actual Values \ Predicted Values	0	1	2	3	4	5	6	7	8	9
0	4	1	37	7	21	11	66	7	23	17
1	1	4	125	3	32	3	22	18	31	10
2	201	138	24	132	40	2	51	96	258	4
3	15	12	350	4	8	350	65	492	251	181
4	19	533	42	3	11	2	385	43	19	300
5	48	2	5	342	3	15	160	3	168	58
6	284	8	11	0	47	72	20	0	40	0
7	25	191	184	70	221	7	7	21	45	296
8	136	243	232	323	98	304	178	61	15	119
9	247	3	22	126	501	126	4	287	124	24

	Precision	Recall	F1-Score	Support
0	0.00	0.02	0.01	194
1	0.00	0.02	0.01	249
2	0.02	0.03	0.02	946
3	0.00	0.00	0.00	1728
4	0.01	0.01	0.01	1357
5	0.02	0.02	0.02	804
6	0.02	0.04	0.03	482
7	0.02	0.02	0.02	1067
8	0.02	0.01	0.01	1709
9	0.02	0.02	0.02	1464
Micro Avg	0.01	0.01	0.01	10000
Macro Avg	0.01	0.02	0.01	10000
Weighted Avg	0.02	0.01	0.01	10000

Figure 31: Confusion matrix and classification report of the neural network model without autoencoder after basic iterative method attack

Actual Values \ Predicted Values	0	1	2	3	4	5	6	7	8	9
0	967	0	3	0	0	2	2	1	4	2
1	0	1123	2	0	2	0	2	5	0	3
2	3	1	1008	7	4	0	0	11	3	3
3	0	1	4	983	0	4	0	9	8	2
4	0	1	2	1	959	3	8	2	4	10
5	0	2	0	11	0	872	6	0	7	5
6	4	4	2	0	2	6	936	0	2	0
7	2	1	6	3	3	1	0	989	3	5
8	2	2	4	5	2	3	4	1	938	7
9	2	0	1	0	10	1	0	10	5	972

	Precision	Recall	F1-Score	Support
0	0.99	0.99	0.99	981
1	0.99	0.99	0.99	1137
2	0.98	0.97	0.97	1040
3	0.97	0.97	0.97	1011
4	0.98	0.97	0.97	990
5	0.98	0.97	0.97	903
6	0.98	0.98	0.98	956
7	0.96	0.98	0.97	1013
8	0.96	0.97	0.97	968
9	0.96	0.97	0.97	1001
Micro Avg	0.97	0.97	0.97	10000
Macro Avg	0.97	0.97	0.97	10000
Weighted Avg	0.97	0.97	0.97	10000

Figure 32: Confusion matrix and classification report of the neural network model with autoencoder after basic iterative method attack

5.3 Sparse Autoencoder

Sparse autoencoders present improved performance on classification tasks. It includes more hidden layers than the input layer. The significant part is defining a small number of hidden layers to be active at once to encourage sparsity. This constraint forces the training model to respond uniquely to the characteristics of translation and uses the statistical features of the input data.

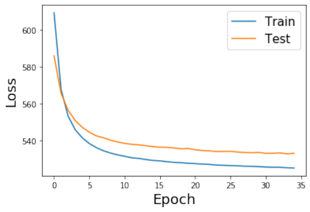


Figure 33: Optimized Relu Loss History for Sparse Autoencoder

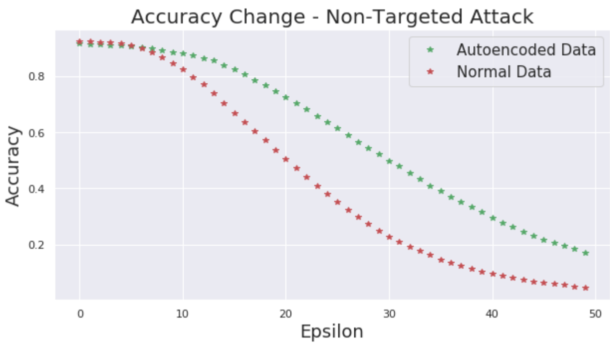


Figure 34: Comparison of accuracy with and without sparse autoencoder for non-targeted attack

Because of this sparse autoencoders involve sparsity penalty $\Omega(h)$ in their training. $L(x, x') + \Omega(h)$

This penalty makes the model to activate specific areas of the network depending on the input data while making all other neurons inactive. We can create this sparsity by relative entropy, also known as Kullback-Leibler divergence.

$\widehat{\rho}_j = \frac{1}{m}\sum_{i=1}^{m}[h_j(x_i)]$ $\widehat{\rho}_j$ is our average activation function of the hidden layer j which is averaged over m training examples. For increasing the sparsity in terms of making the number of active neurons as smaller as it can be, we would want ρ close to zero. The sparsity penalty term $\Omega(h)$ will punish $\widehat{\rho}_j$ for deviating from ρ, which will be basically exploiting Kullback-Leibler divergence. $KL(p||\widehat{\rho}_j)$ is our Kullback-Leibler divergence between a random variable ρ and random variable with mean $\widehat{\rho}_j$.

$\sum_{j=1}^{s} KL(\rho||\widehat{\rho}_j) = \sum_{j=1}^{s}[\rho log\frac{\rho}{\widehat{\rho}_j} + (1-\rho)log\frac{1-\rho}{1-\widehat{\rho}_j}]$

Sparsity can be achieved with other ways, such as applying L1 and L2 regularization terms on the activation of the hidden layer. L is our loss function and λ is our scale parameter.

$L(x, x') + \lambda\sum_i |h_i|$

5.3.1 Multi-Class Logistic Regression of Sparse Autoencoder

This section presents multi-class logistic regressions with sparse autoencoders. The difference from the autoencoder section is the autoencoder type. The findings from Figure 6 and Figure 33 show that loss is higher compared to the autoencoders in sparse autoencoder.

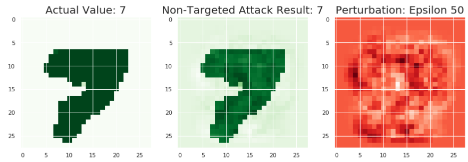


Figure 35: Value change and perturbation of a non-targeted attack on model without sparse autoencoder

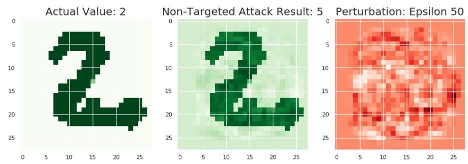


Figure 36: Value change and perturbation of a non-targeted attack on model with sparse autoencoder

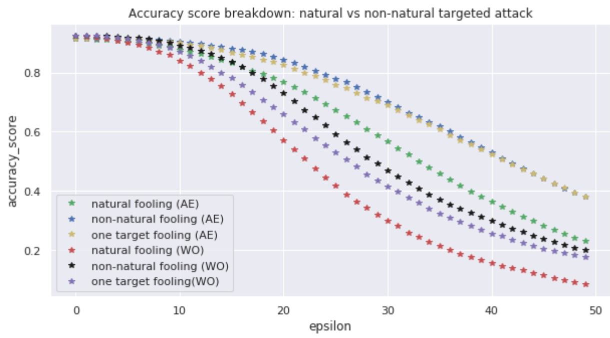


Figure 37: Comparison of accuracy with and without sparse autoencoder for targeted attacks. *AE stands for the models with sparse autoencoder, WO stands for models without autoencoder*

The difference between perturbation is presented in Figure 35 and Figure 36 compared to the perturbation in Figure 11 and Figure 12. The perturbation is sharper in sparse autoencoder.

Figure 37 indicates that sparse autoencoders performs poorly compared to autoencoders in multi-class logistic regression.

5.3.2 Neural Network of Sparse Autoencoder

Sparse autoencoder results for neural networks indicate that vanilla autoencoder seems to be slightly better than sparse autoencoders for neural networks. Sparse autoencoders do not perform as well in linear machine learning models, in our case, multi-class logistic regression.

5.4 Denoising Autoencoder

Denoising autoencoders are used for partially corrupted input and train it to recover the original undistorted input. In this study, the corrupted input is not used. The aim is to achieve a good design by changing the reconstruction prin-

	Predicted Values									
Actual Values	0	1	2	3	4	5	6	7	8	9
0	972	0	1	0	1	2	5	1	6	4
1	0	1127	2	0	0	0	2	4	0	2
2	3	3	1019	6	4	0	3	10	6	1
3	0	0	0	996	0	10	0	4	0	3
4	0	0	1	0	965	0	3	0	1	10
5	0	0	0	2	0	865	1	0	2	3
6	1	2	0	0	3	6	942	0	0	0
7	1	1	7	2	2	2	0	1008	5	7
8	3	2	2	3	1	5	2	1	952	2
9	0	0	0	1	6	2	0	0	2	977

	Precision	Recall	F1-Score	Support
0	0.99	0.98	0.99	992
1	0.99	0.99	0.99	1137
2	0.99	0.97	0.98	1055
3	0.99	0.98	0.98	1013
4	0.98	0.98	0.98	980
5	0.97	0.99	0.98	873
6	0.98	0.99	0.99	954
7	0.98	0.97	0.98	1035
8	0.98	0.98	0.98	973
9	0.97	0.99	0.98	988
Micro Avg	0.98	0.98	0.98	10000
Macro Avg	0.98	0.98	0.98	10000
Weighted Avg	0.98	0.98	0.98	10000

Figure 38: Confusion matrix and classification report of the neural network model without sparse autoencoder

	Predicted Values									
Actual Values	0	1	2	3	4	5	6	7	8	9
0	967	0	9	0	2	5	2	1	5	4
1	0	1118	1	0	0	1	3	8	0	2
2	2	4	996	7	2	0	1	14	8	0
3	0	2	10	977	0	18	1	3	8	8
4	0	1	1	0	956	2	5	3	1	12
5	1	0	0	6	0	846	3	0	4	7
6	4	2	1	0	6	6	940	0	1	0
7	1	0	6	5	2	0	0	983	7	12
8	5	7	6	11	3	7	2	4	936	3
9	0	1	2	4	11	7	1	12	4	961

	Precision	Recall	F1-Score	Support
0	0.99	0.97	0.98	995
1	0.99	0.99	0.99	1133
2	0.97	0.96	0.96	1034
3	0.97	0.95	0.96	1027
4	0.97	0.97	0.97	981
5	0.95	0.98	0.96	867
6	0.98	0.98	0.98	960
7	0.96	0.97	0.96	1016
8	0.96	0.95	0.96	984
9	0.95	0.96	0.96	1003
Micro Avg	0.97	0.97	0.97	10000
Macro Avg	0.97	0.97	0.97	10000
Weighted Avg	0.97	0.97	0.97	10000

Figure 39: Confusion matrix and classification report of the neural network model with sparse autoencoder

ciple for using denoising autoencoders. For achieving this denoising properly, the model requires to extract features that capture useful structure in the distribution of the input. Denoising autoencoders apply corrupted data through stochastic mapping. Our input is x and corrupted data is $\widetilde{x}$ and stochastic mapping is $\widetilde{x} \sim q_D(\widetilde{x}|x)$.

As its a standard autoencoder, corrupted data $\widetilde{x}$ is mapped to a hidden layer.

$h = f_\theta(\widetilde{x}) = s(W\widetilde{x} + b)$.

And from this the model reconstructs $z = g'_\theta(h)$.

	Predicted Values									
Actual Values	0	1	2	3	4	5	6	7	8	9
0	54	0	29	5	2	13	111	12	20	10
1	0	4	114	4	54	8	7	35	38	17
2	369	416	154	295	77	14	222	252	510	14
3	14	12	315	36	5	338	58	297	80	212
4	2	182	19	2	63	1	161	20	10	188
5	41	1	4	329	11	80	185	1	117	47
6	276	9	11	1	48	89	120	0	57	3
7	22	203	183	72	288	7	0	57	80	411
8	108	308	195	188	73	249	89	83	16	88
9	94	0	8	78	361	93	5	271	46	19

	Precision	Recall	F1-Score	Support
0	0.06	0.21	0.09	256
1	0.00	0.01	0.01	281
2	0.15	0.07	0.09	2323
3	0.04	0.03	0.03	1367
4	0.06	0.10	0.08	648
5	0.09	0.10	0.09	816
6	0.13	0.20	0.15	614
7	0.06	0.04	0.05	1323
8	0.02	0.01	0.01	1397
9	0.02	0.02	0.02	975
Micro Avg	0.06	0.06	0.06	10000
Macro Avg	0.06	0.08	0.06	10000
Weighted Avg	0.07	0.06	0.06	10000

Figure 40: Confusion matrix and classification report of the neural network model without sparse autoencoder after FGSM attack

	Predicted Values									
Actual Values	0	1	2	3	4	5	6	7	8	9
0	966	0	7	0	2	4	4	2	3	4
1	0	1121	0	1	1	2	3	8	0	4
2	3	7	996	6	2	0	1	15	9	1
3	1	0	10	976	0	17	2	4	10	6
4	0	1	1	0	952	0	7	3	2	15
5	1	0	0	8	0	849	3	0	5	6
6	6	2	2	0	7	6	934	0	2	0
7	0	1	4	3	2	1	0	977	9	15
8	3	3	9	11	4	8	3	6	930	5
9	0	0	3	5	12	5	1	13	4	953

	Precision	Recall	F1-Score	Support
0	0.99	0.97	0.98	992
1	0.99	0.98	0.99	1140
2	0.97	0.96	0.96	1040
3	0.97	0.95	0.96	1026
4	0.97	0.97	0.97	981
5	0.95	0.97	0.96	872
6	0.97	0.97	0.97	959
7	0.95	0.97	0.96	1012
8	0.95	0.95	0.95	982
9	0.94	0.96	0.95	996
Micro Avg	0.97	0.97	0.97	10000
Macro Avg	0.97	0.97	0.97	10000
Weighted Avg	0.97	0.97	0.97	10000

Figure 41: Confusion matrix and classification report of the neural network model with sparse autoencoder after FGSM attack

5.4.1 Multi-Class Logistic Regression of Denoising Autoencoder

In denoising autoencoder for multi-class logistic regression, the loss does not improve for each epoch. Although it starts better at lower epoch values, in the end, vanilla autoencoder seems to be better. Sparse autoencoder's loss is slightly worse.

And just like sparse autoencoder, denoising autoencoder

		Predicted Values									
		0	1	2	3	4	5	6	7	8	9
Actual Values	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	4	1	1	0	0	14	2	9
	2	0	5	1	1	2	0	0	34	4	0
	3	0	0	0	0	0	0	0	16	1	0
	4	0	0	0	0	0	0	0	0	0	0
	5	980	1130	1023	1007	976	892	958	961	967	998
	6	0	0	0	0	0	0	0	0	0	0
	7	0	0	4	1	3	0	0	0	0	2
	8	0	0	0	0	0	0	0	3	0	0
	9	0	0	0	0	0	0	0	0	0	0

	Precision	Recall	F1-Score	Support
0	0.00	0.00	0.00	0
1	0.00	0.00	0.00	31
2	0.00	0.02	0.00	47
3	0.00	0.00	0.00	17
4	0.00	0.00	0.00	0
5	1.00	0.09	0.17	9892
6	0.00	0.00	0.00	0
7	0.00	0.00	0.00	10
8	0.00	0.00	0.00	3
9	0.00	0.00	0.00	0
Micro Avg	0.09	0.09	0.09	10000
Macro Avg	0.10	0.01	0.02	10000
Weighted Avg	0.99	0.09	0.16	10000

Figure 42: Confusion matrix and classification report of the neural network model without sparse autoencoder after T-FGSM attack

		Predicted Values									
		0	1	2	3	4	5	6	7	8	9
Actual Values	0	966	0	9	0	1	3	4	1	4	4
	1	0	1121	1	0	1	1	3	9	0	3
	2	2	5	998	7	2	0	1	15	7	0
	3	0	1	9	974	0	14	1	3	9	7
	4	0	1	1	0	954	0	5	3	1	14
	5	1	0	0	9	0	862	6	0	5	11
	6	5	2	2	0	7	5	935	0	0	0
	7	1	0	5	5	2	0	0	981	7	12
	8	5	4	6	11	4	3	2	5	938	4
	9	0	1	1	4	11	4	1	11	3	954

	Precision	Recall	F1-Score	Support
0	0.99	0.97	0.98	992
1	0.99	0.98	0.99	1139
2	0.97	0.96	0.96	1037
3	0.96	0.96	0.96	1018
4	0.97	0.97	0.97	979
5	0.97	0.96	0.97	894
6	0.98	0.98	0.98	956
7	0.95	0.97	0.96	1013
8	0.96	0.96	0.96	982
9	0.95	0.96	0.95	990
Micro Avg	0.97	0.97	0.97	10000
Macro Avg	0.97	0.97	0.97	10000
Weighted Avg	0.97	0.97	0.97	10000

Figure 43: Confusion matrix and classification report of the neural network model with sparse autoencoder after T-FGSM attack

also applies a sharp perturbation, which is presented in Figure 48 and Figure 49.

We observe that there is a similarity between accuracy results for denoising autoencoder with multi-class logistic regression and sparse autoencoder results. Natural fooling accuracy drops drastically in denoising autoencoder, but non-targeted and one targeted attack seem to be somewhat like sparse autoencoder, one targeted attack having less accuracy in denoising autoencoder.

		Predicted Values									
		0	1	2	3	4	5	6	7	8	9
Actual Values	0	4	0	35	4	2	13	122	11	22	10
	1	0	4	111	4	33	8	6	32	37	9
	2	377	360	10	295	65	5	246	205	494	14
	3	14	12	398	11	6	337	68	315	98	211
	4	2	150	23	2	15	0	226	20	9	199
	5	37	0	2	330	6	23	177	2	110	45
	6	299	11	15	1	56	103	11	0	59	5
	7	19	223	206	72	278	7	1	18	78	392
	8	118	374	218	190	89	272	94	95	16	102
	9	110	1	14	101	432	124	7	330	51	22

	Precision	Recall	F1-Score	Support
0	0.00	0.02	0.01	223
1	0.00	0.02	0.01	244
2	0.01	0.00	0.01	2071
3	0.01	0.01	0.01	1470
4	0.02	0.02	0.02	646
5	0.03	0.03	0.03	732
6	0.01	0.02	0.01	560
7	0.02	0.01	0.02	1294
8	0.02	0.01	0.01	1568
9	0.02	0.02	0.02	1192
Micro Avg	0.01	0.01	0.01	10000
Macro Avg	0.01	0.02	0.01	10000
Weighted Avg	0.01	0.01	0.01	10000

Figure 44: Confusion matrix and classification report of the neural network model without sparse autoencoder after basic iterative method attack

		Predicted Values									
		0	1	2	3	4	5	6	7	8	9
Actual Values	0	964	0	6	0	2	4	4	2	3	4
	1	0	1119	0	1	1	1	3	8	0	3
	2	3	8	998	6	2	0	1	14	8	1
	3	1	0	10	972	0	21	2	5	12	7
	4	0	1	1	0	955	1	11	2	2	19
	5	1	0	0	8	0	844	4	0	5	7
	6	7	2	2	0	7	7	929	0	2	0
	7	0	1	4	5	2	1	0	974	7	19
	8	4	4	9	13	4	9	3	7	931	5
	9	0	0	2	5	9	4	1	16	4	944

	Precision	Recall	F1-Score	Support
0	0.98	0.97	0.98	989
1	0.99	0.99	0.99	1136
2	0.97	0.96	0.96	1041
3	0.96	0.94	0.95	1030
4	0.97	0.96	0.97	992
5	0.95	0.97	0.96	869
6	0.97	0.97	0.97	956
7	0.95	0.96	0.95	1013
8	0.96	0.94	0.95	989
9	0.94	0.96	0.95	985
Micro Avg	0.96	0.96	0.96	10000
Macro Avg	0.96	0.96	0.96	10000
Weighted Avg	0.96	0.96	0.96	10000

Figure 45: Confusion matrix and classification report of the neural network model with sparse autoencoder after basic iterative method attack

5.4.2 Neural Network of Denoising Autoencoder

We investigate that neural network accuracy for denoising autoencoder is worse than sparse autoencoder results and vanilla autoencoder results. It is still a useful autoencoder for denoising corrupted data and other purposes; however, it is not the right choice just for robustness against adversarial examples.

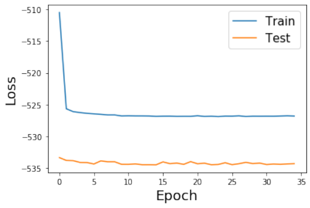


Figure 46: Optimized Relu Loss History for Denoising Autoencoder

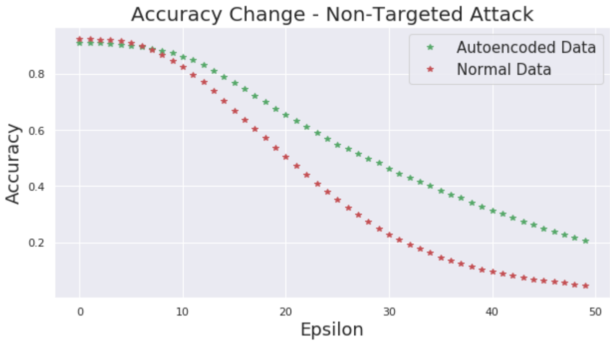


Figure 47: Comparison of accuracy with and without denoising autoencoder for non-targeted attack

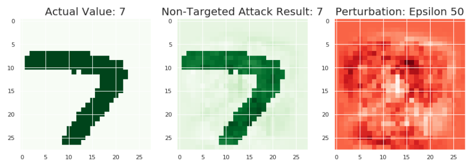


Figure 48: Value change and perturbation of a non-targeted attack on model without denoising autoencoder

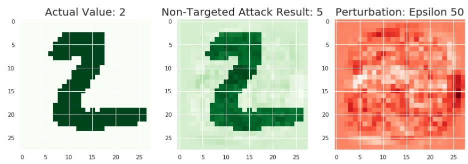


Figure 49: Value change and perturbation of a non-targeted attack on model with denoising autoencoder

5.5 Variational Autoencoder

In this study, we examine variational autoencoders as the final type of autoencoder type. The variational autoencoders have an encoder and a decoder, although their mathematical formulation differs significantly. They are associated with Generative Adversarial Networks due to their architectural similarity. In summary, variational autoencoders are also generative models. Differently, from sparse autoencoders, denoising autoencoders, and vanilla autoencoders, all of which aim discriminative modeling, generative modeling tries to simulate how the data can be generated and to understand the underlying causal relations. It also considers these causal relations when generating new

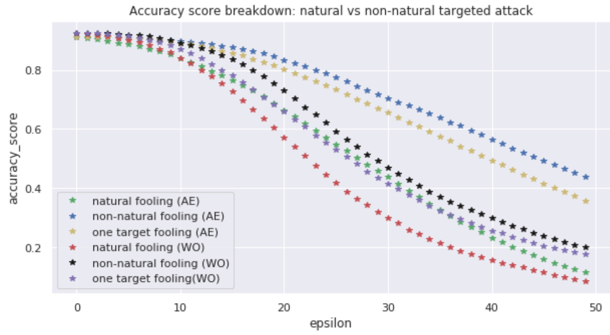


Figure 50: Comparison of accuracy with and without denoising autoencoder for targeted attacks. *AE stands for the models with denoising autoencoder, WO stands for models without autoencoder*

		Predicted Values									
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Actual Values	**0**	974	0	3	0	0	2	7	1	5	3
	1	0	1125	5	0	0	0	3	2	0	3
	2	1	4	1009	4	1	0	2	10	3	0
	3	0	2	0	999	0	5	0	3	1	2
	4	0	0	2	0	973	0	3	0	1	10
	5	0	0	0	3	0	877	1	0	1	4
	6	1	1	0	0	3	4	938	0	0	0
	7	1	1	10	2	0	2	0	1006	4	5
	8	3	2	3	2	1	2	4	2	957	3
	9	0	0	0	0	4	0	0	4	2	979

	Precision	**Recall**	**F1-Score**	**Support**
0	0.99	0.98	0.99	995
1	0.99	0.99	0.99	1138
2	0.98	0.98	0.98	1034
3	0.99	0.99	0.99	1012
4	0.99	0.98	0.99	989
5	0.98	0.99	0.99	886
6	0.98	0.99	0.98	947
7	0.98	0.98	0.98	1031
8	0.98	0.98	0.98	979
9	0.97	0.99	0.98	989
Micro Avg	0.98	0.98	0.98	10000
Macro Avg	0.98	0.98	0.98	10000
Weighted Avg	0.98	0.98	0.98	10000

Figure 51: Confusion matrix and classification report of the neural network model without denoising autoencoder

data.

Variational autoencoders use an estimator algorithm called Stochastic Gradient Variational Bayes for training. This algorithm assumes the data is generated by $p_\theta(x|h)$ which is a directed graphical model and θ being the parameters of decoder, in variational autoencoder's case, the parameters of the generative model. The encoder is learning an approximation of $q_\phi(h|x)$ to a posterior distribution which is showed by $p_\theta(x|h)$ and ϕ being the parameters of the encoder; in variational autoencoder's case, the parameters of recognition model. We will use Kullback-Leibler divergence again, showed as D_{KL}.

$L = (\phi, \theta, x) = D_{KL}(q_\phi(h|x)||p_\theta(h)) - \mathbb{E}_{q_\phi(h|x)}(logp_\theta(x|h))$.

Variational and likelihood distributions' shape is chosen by factorized Gaussians. The encoder outputs are $p(x)$ and $w^2(x)$. The decoder outputs are $\mu(h)$ and $\sigma^2(h)$. The like-

	Predicted Values									
Actual Values	0	1	2	3	4	5	6	7	8	9
0	961	0	2	0	1	2	7	1	7	3
1	1	1119	7	0	2	0	2	8	2	3
2	5	3	992	7	3	0	1	10	6	0
3	1	1	10	980	1	16	1	8	20	10
4	1	0	1	0	937	3	1	0	5	12
5	3	1	0	7	0	852	5	0	5	4
6	6	5	1	1	6	7	937	0	3	0
7	1	0	11	7	4	0	0	984	3	11
8	1	5	8	5	2	7	3	4	913	7
9	0	1	0	3	26	5	1	13	10	959

	Precision	Recall	F1-Score	Support
0	0.98	0.98	0.98	984
1	0.99	0.98	0.98	1144
2	0.96	0.97	0.96	1027
3	0.97	0.94	0.95	1048
4	0.95	0.98	0.96	960
5	0.96	0.97	0.96	877
6	0.98	0.97	0.97	966
7	0.96	0.96	0.96	1021
8	0.94	0.96	0.95	955
9	0.95	0.94	0.95	1018
Micro Avg	0.96	0.96	0.96	10000
Macro Avg	0.96	0.96	0.96	10000
Weighted Avg	0.96	0.96	0.96	10000

Figure 52: Confusion matrix and classification report of the neural network model with denoising autoencoder

	Predicted Values									
Actual Values	0	1	2	3	4	5	6	7	8	9
0	82	0	22	7	15	5	71	4	18	7
1	0	8	121	5	85	2	6	42	39	18
2	200	365	86	255	54	15	141	210	304	8
3	9	12	352	54	6	302	47	329	155	114
4	9	272	36	2	69	0	282	50	14	280
5	83	7	8	345	19	92	212	6	232	78
6	303	20	8	1	25	56	77	0	53	1
7	11	206	181	68	278	2	1	56	57	350
8	146	244	213	218	84	352	120	77	16	141
9	137	1	5	55	347	66	1	254	86	12

	Precision	Recall	F1-Score	Support
0	0.08	0.35	0.14	231
1	0.01	0.02	0.01	326
2	0.08	0.05	0.06	1638
3	0.05	0.04	0.05	1380
4	0.07	0.07	0.07	1014
5	0.10	0.09	0.09	1082
6	0.08	0.14	0.10	544
7	0.05	0.05	0.05	1210
8	0.02	0.01	0.01	1611
9	0.01	0.01	0.01	964
Micro Avg	0.06	0.06	0.06	10000
Macro Avg	0.06	0.08	0.06	10000
Weighted Avg	0.06	0.06	0.05	10000

Figure 53: Confusion matrix and classification report of the neural network model without denoising autoencoder after FGSM attack

lihood term of variational objective is defined below.

$$q_\phi(h|x) = N(p(x), w^2(x)I)$$
$$p_\theta(x|h) = N(\mu(h), \sigma^2(h)I)$$

5.5.1 Multi-Class Logistic Regression of Variational Autoencoder

The findings from Figure 59 show that variational autoencoder indicates the best loss function result. However, Figure 60 presents that the accuracy is low, especially in low

	Predicted Values									
Actual Values	0	1	2	3	4	5	6	7	8	9
0	961	0	1	0	2	2	7	1	6	3
1	1	1120	7	1	3	0	2	7	2	4
2	5	3	993	7	2	0	1	10	7	0
3	1	1	11	977	2	16	1	10	18	9
4	1	0	1	0	935	3	1	1	6	11
5	3	1	0	7	0	855	4	0	6	3
6	5	4	2	1	7	5	938	0	3	0
7	1	0	9	8	4	0	0	981	3	11
8	1	6	8	6	3	7	3	4	914	7
9	1	0	0	3	24	4	1	14	9	961

	Precision	Recall	F1-Score	Support
0	0.98	0.98	0.98	983
1	0.99	0.98	0.98	1147
2	0.96	0.97	0.96	1028
3	0.97	0.93	0.95	1046
4	0.95	0.97	0.96	959
5	0.96	0.97	0.97	879
6	0.98	0.97	0.98	965
7	0.95	0.96	0.96	1017
8	0.94	0.95	0.95	959
9	0.95	0.94	0.95	1017
Micro Avg	0.96	0.96	0.96	10000
Macro Avg	0.96	0.96	0.96	10000
Weighted Avg	0.96	0.96	0.96	10000

Figure 54: Confusion matrix and classification report of the neural network model with denoising autoencoder after FGSM attack

	Predicted Values									
Actual Values	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	3	0	0	0	1	0
2	0	10	0	0	1	0	0	0	2	0
3	0	0	0	0	0	0	1	1	0	0
4	0	0	0	0	0	0	0	0	0	0
5	980	1125	1032	1010	976	892	957	1026	971	1009
6	0	0	0	0	2	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	1	0	0
9	0	0	0	0	0	0	0	0	0	0

	Precision	Recall	F1-Score	Support
0	0.00	0.00	0.00	0
1	0.00	0.00	0.00	4
2	0.00	0.00	0.00	13
3	0.00	0.00	0.00	2
4	0.00	0.00	0.00	0
5	1.00	0.09	0.16	9978
6	0.00	0.00	0.00	2
7	0.00	0.00	0.00	0
8	0.00	0.00	0.00	1
9	0.00	0.00	0.00	0
Micro Avg	0.09	0.09	0.09	10000
Macro Avg	0.10	0.01	0.02	10000
Weighted Avg	1.00	0.09	0.16	10000

Figure 55: Confusion matrix and classification report of the neural network model without denoising autoencoder after T-FGSM attack

epsilon values where even autoencoded data gives worse accuracy than the normal learning process.

Perturbation applied by variational autoencoder is not as sharp in sparse autoencoder and denoising autoencoder. It seems similar to vanilla autoencoder's perturbation.

The variational autoencoder has the worst results. Besides, it presents bad results at the low values of epsilon, making autoencoded data less accurate and only a slight improvement compared to the normal data in high values of epsilon.

Actual \ Predicted	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0
2	340	374	1018	773	30	88	6	14	571	28
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	621	274	2	20	269	579	949	0	218	27
7	19	487	12	217	683	225	3	1014	185	954
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0

	Precision	Recall	F1-Score	Support
0	0.00	0.00	0.00	0
1	0.00	0.00	0.00	0
2	0.99	0.31	0.48	3242
3	0.00	0.00	0.00	0
4	0.00	0.00	0.00	0
5	0.00	0.00	0.00	0
6	0.99	0.32	0.48	2959
7	0.99	0.27	0.42	3799
8	0.00	0.00	0.00	0
9	0.00	0.00	0.00	0
Micro Avg	0.30	0.30	0.30	10000
Macro Avg	0.30	0.09	0.14	10000
Weighted Avg	0.99	0.30	0.46	10000

Figure 56: Confusion matrix and classification report of the neural network model with denoising autoencoder after T-FGSM attack

Actual \ Predicted	0	1	2	3	4	5	6	7	8	9
0	2	0	23	6	18	5	67	4	18	7
1	0	7	116	3	57	2	7	38	37	12
2	207	323	20	263	34	5	165	190	291	5
3	5	4	384	8	5	310	37	337	163	108
4	10	273	40	1	7	0	304	49	15	285
5	56	1	5	336	14	14	216	5	216	67
6	331	15	8	1	31	64	12	0	58	1
7	12	202	190	76	289	2	0	18	59	346
8	184	308	239	238	120	415	149	84	14	161
9	173	2	7	78	407	75	1	303	103	17

	Precision	Recall	F1-Score	Support
0	0.00	0.01	0.00	150
1	0.01	0.03	0.01	279
2	0.02	0.01	0.02	1503
3	0.01	0.01	0.01	1361
4	0.01	0.01	0.01	984
5	0.02	0.02	0.02	930
6	0.01	0.02	0.02	521
7	0.02	0.02	0.02	1194
8	0.01	0.01	0.01	1912
9	0.02	0.01	0.02	1166
Micro Avg	0.01	0.01	0.01	10000
Macro Avg	0.01	0.01	0.01	10000
Weighted Avg	0.01	0.01	0.01	10000

Figure 57: Confusion matrix and classification report of the neural network model without denoising autoencoder after basic iterative method attack

5.5.2 Neural Network of Variational Autoencoder

Variational autoencoder with neural networks also illustrates the worst results compared to other autoencoder types, where the accuracy for autoencoded data against an attack has around between 0.96 and 0.99 accuracies, variational autoencoder has around between 0.65 and 0.70 accuracies.

Actual \ Predicted	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0
2	351	391	1017	773	30	89	6	16	577	28
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	609	273	3	17	261	575	949	0	212	26
7	20	471	12	220	691	228	3	1012	185	955
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0

	Precision	Recall	F1-Score	Support
0	0.00	0.00	0.00	0
1	0.00	0.00	0.00	0
2	0.99	0.31	0.47	3278
3	0.00	0.00	0.00	0
4	0.00	0.00	0.00	0
5	0.00	0.00	0.00	0
6	0.99	0.32	0.49	2925
7	0.98	0.27	0.42	3797
8	0.00	0.00	0.00	0
9	0.00	0.00	0.00	0
Micro Avg	0.30	0.30	0.30	10000
Macro Avg	0.30	0.09	0.14	10000
Weighted Avg	0.99	0.30	0.46	10000

Figure 58: Confusion matrix and classification report of the neural network model with denoising autoencoder after basic iterative method attack

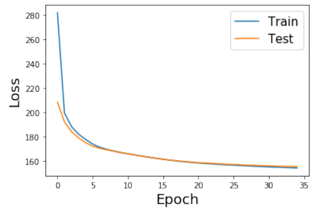


Figure 59: Optimized Relu Loss History for Variational Autoencoder

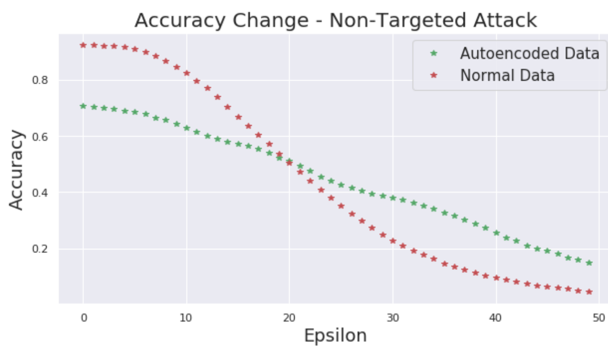


Figure 60: Comparison of accuracy with and without variational autoencoder for non-targeted attack

6 Experiments with Fashion MNIST Dataset

6.1 Introduction

We also used Fashion MNIST dataset. We will briefly show the experiment results for not filling the paper with

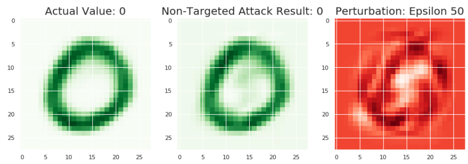


Figure 61: Value change and perturbation of a non-targeted attack on model without variational autoencoder

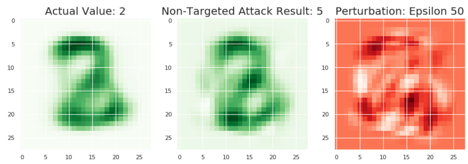


Figure 62: Value change and perturbation of a non-targeted attack on model with variational autoencoder

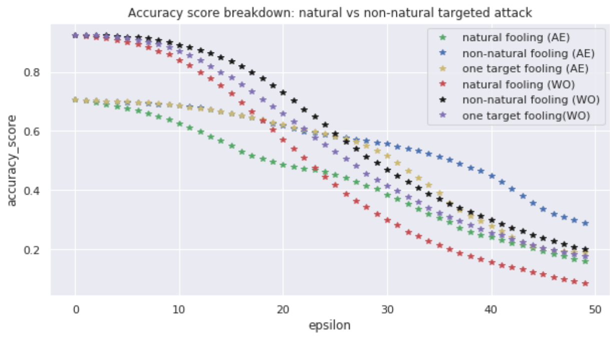


Figure 63: Comparison of accuracy with and without variational autoencoder for targeted attacks. *AE stands for the models with variational autoencoder, WO stands for models without autoencoder*

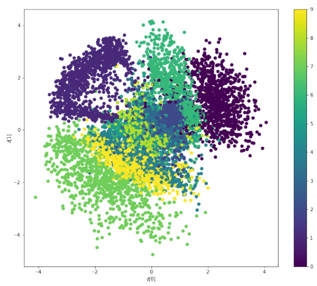


Figure 64: Because of MNIST dataset, our latent space is two-dimensional. One is to look at the neighbourhoods of different classes on the latent 2D plane. Each of these coloured clusters is a type of digit. Close clusters are structurally similar digits, and they are digits that share information in the latent space.

too many images. In these results, we have only changed the imported dataset. All the structure and code for the paper remains the same. Each training and test example is assigned to one of the following labels:

0. T-shirt/top

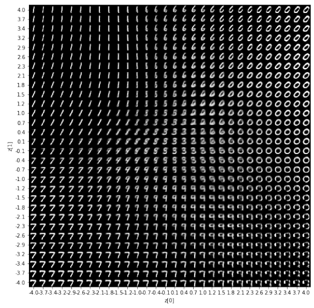


Figure 65: Due to VAE is a generative model, we can also generate new Mnist digits using latent plane, sampling latent points at regular intervals, and generating the corresponding digit for each of these points.

		Predicted Values									
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Actual Values	**0**	976	1	3	0	0	2	6	1	6	7
	1	0	1124	1	0	0	0	3	2	0	2
	2	0	4	1013	6	2	0	1	11	3	0
	3	0	0	1	994	0	4	1	2	0	1
	4	0	0	3	0	970	0	1	0	1	7
	5	0	0	0	2	0	876	5	0	2	3
	6	1	3	0	0	1	5	938	0	1	0
	7	1	2	6	2	1	2	0	1004	3	9
	8	2	1	4	3	2	2	2	2	954	1
	9	0	0	1	3	6	1	1	6	4	979

	Precision	Recall	F1-Score	Support
0	1.00	0.97	0.98	1002
1	0.99	0.99	0.99	1132
2	0.98	0.97	0.98	1040
3	0.98	0.99	0.99	1003
4	0.99	0.99	0.99	982
5	0.98	0.99	0.98	888
6	0.98	0.99	0.98	949
7	0.98	0.97	0.98	1030
8	0.98	0.98	0.98	973
9	0.97	0.98	0.97	1001
Micro Avg	0.98	0.98	0.98	10000
Macro Avg	0.98	0.98	0.98	10000
Weighted Avg	0.98	0.98	0.98	10000

Figure 66: Confusion matrix and classification report of the neural network model without variational autoencoder

1. Trouser
2. Pullover
3. Dress
4. Coat
5. Sandal
6. Shirt
7. Sneaker
8. Bag
9. Ankle boot

		Predicted Values									
		0	1	2	3	4	5	6	7	8	9
Actual Values	0	863	0	3	4	1	7	44	0	2	1
	1	0	1095	2	10	2	8	8	11	17	4
	2	27	3	810	102	12	49	64	2	37	12
	3	8	5	105	673	9	227	5	6	189	12
	4	1	0	15	9	688	35	2	66	33	383
	5	6	10	22	55	13	190	16	7	108	2
	6	69	6	46	3	8	17	815	0	6	4
	7	0	0	0	1	2	0	0	745	0	87
	8	6	15	27	143	57	350	3	39	578	25
	9	0	1	2	10	190	9	1	152	4	479

	Precision	Recall	F1-Score	Support
0	0.88	0.93	0.91	925
1	0.96	0.95	0.96	1157
2	0.78	0.72	0.75	1118
3	0.67	0.54	0.60	1239
4	0.70	0.56	0.62	1232
5	0.21	0.44	0.29	429
6	0.85	0.84	0.84	974
7	0.72	0.89	0.80	835
8	0.59	0.47	0.52	1243
9	0.47	0.56	0.52	848
Micro Avg	0.69	0.69	0.69	10000
Macro Avg	0.69	0.69	0.68	10000
Weighted Avg	0.72	0.69	0.70	10000

Figure 67: Confusion matrix and classification report of the neural network model with variational autoencoder

		Predicted Values									
		0	1	2	3	4	5	6	7	8	9
Actual Values	0	96	4	47	10	20	15	109	8	20	12
	1	1	7	106	11	103	3	5	35	42	25
	2	243	239	116	210	60	7	143	157	454	10
	3	15	129	386	52	7	309	25	396	103	215
	4	7	291	38	1	42	0	252	37	13	299
	5	48	1	0	310	5	62	178	8	112	53
	6	255	14	15	0	35	69	103	0	43	2
	7	18	284	131	45	249	7	4	47	46	288
	8	47	165	163	172	71	294	134	74	17	85
	9	250	1	30	199	390	126	5	266	124	20

	Precision	Recall	F1-Score	Support
0	0.10	0.28	0.15	341
1	0.01	0.02	0.01	338
2	0.11	0.07	0.09	1639
3	0.05	0.03	0.04	1637
4	0.04	0.04	0.04	980
5	0.07	0.08	0.07	777
6	0.11	0.19	0.14	536
7	0.05	0.04	0.04	1119
8	0.02	0.01	0.02	1222
9	0.02	0.01	0.02	1411
Micro Avg	0.06	0.06	0.06	10000
Macro Avg	0.06	0.08	0.06	10000
Weighted Avg	0.06	0.06	0.05	10000

Figure 68: Confusion matrix and classification report of the neural network model without variational autoencoder after FGSM attack

6.2 Autoencoding

6.2.1 Multi-Class Logistic Regression

The process for multi-class logistic regression for fashion MNIST is the same as it was in MNIST Dataset. We apply perturbation to clothes and shoes but it does not matter for the learning model as long as it is labeled correctly. In this particular perturbation example, a shirt is mispredicted as a trouser. We can also observe the line drawn by perturbation

		Predicted Values									
		0	1	2	3	4	5	6	7	8	9
Actual Values	0	858	0	4	3	2	6	47	0	3	1
	1	0	1087	3	7	0	12	4	6	17	6
	2	25	5	803	109	11	53	70	3	35	11
	3	10	4	116	646	10	224	7	11	185	13
	4	1	1	13	12	618	33	3	59	42	366
	5	5	18	25	85	12	173	10	18	157	3
	6	75	4	40	8	8	18	810	0	6	3
	7	0	0	0	3	7	0	0	752	1	106
	8	6	16	25	129	55	359	6	27	521	24
	9	0	0	3	8	259	14	1	152	7	476

	Precision	Recall	F1-Score	Support
0	0.88	0.93	0.90	924
1	0.96	0.95	0.95	1142
2	0.78	0.71	0.74	1125
3	0.64	0.53	0.58	1226
4	0.63	0.54	0.58	1148
5	0.19	0.34	0.25	506
6	0.85	0.83	0.84	972
7	0.73	0.87	0.79	869
8	0.53	0.45	0.49	1168
9	0.47	0.52	0.49	920
Micro Avg	0.67	0.67	0.67	10000
Macro Avg	0.67	0.67	0.66	10000
Weighted Avg	0.69	0.67	0.68	10000

Figure 69: Confusion matrix and classification report of the neural network model with variational autoencoder after FGSM attack

		Predicted Values									
		0	1	2	3	4	5	6	7	8	9
Actual Values	0	0	0	0	0	0	0	0	0	1	0
	1	1	1	15	5	37	2	2	26	52	117
	2	4	12	1	89	0	0	0	29	17	0
	3	1	48	236	1	15	1	0	7	1	0
	4	0	0	2	0	0	0	3	0	0	39
	5	972	1029	775	914	912	889	948	953	901	853
	6	2	1	1	1	8	0	1	0	0	0
	7	0	0	0	0	4	0	0	0	1	0
	8	0	44	2	0	0	0	2	13	1	0
	9	0	0	0	0	6	0	2	0	0	0

	Precision	Recall	F1-Score	Support
0	0.00	0.00	0.00	1
1	0.00	0.00	0.00	258
2	0.00	0.01	0.00	152
3	0.00	0.00	0.00	310
4	0.00	0.00	0.00	44
5	1.00	0.10	0.18	9146
6	0.00	0.07	0.00	14
7	0.00	0.00	0.00	5
8	0.00	0.02	0.00	62
9	0.00	0.00	0.00	8
Micro Avg	0.09	0.09	0.09	10000
Macro Avg	0.10	0.02	0.02	10000
Weighted Avg	0.91	0.09	0.16	10000

Figure 70: Confusion matrix and classification report of the neural network model without variational autoencoder after T-FGSM attack

to the shirt, which surely made it look like a trouser.

When we look at the heat map, we can see ankle boots are mispredicted as sandals most. Sandals mispredicted as ankle boots come second, pullover mispredicted as the coat comes third and trouser as a dress comes forth. In numbers, most mispredicted numbers were 4 as 9 and 8 as 3 and 1. From what we have seen from the perturbation image, clothes can be more deceiving to the human eye than numbers. Now let us see how much it will differ from the

Actual Values \ Predicted Values	0	1	2	3	4	5	6	7	8	9
0	858	0	3	4	0	6	40	0	2	1
1	0	1079	2	8	0	4	4	7	12	5
2	23	3	788	90	16	42	57	2	31	12
3	8	4	101	605	8	162	4	6	144	12
4	1	0	12	8	694	28	2	69	33	394
5	18	40	64	195	58	534	36	45	534	21
6	70	6	44	4	9	16	813	0	6	3
7	0	0	0	2	3	0	0	762	0	97
8	2	3	16	85	22	88	1	9	210	12
9	0	0	2	9	172	12	1	128	2	452

	Precision	Recall	F1-Score	Support
0	0.88	0.94	0.91	914
1	0.95	0.96	0.96	1121
2	0.76	0.74	0.75	1064
3	0.60	0.57	0.59	1054
4	0.71	0.56	0.62	1241
5	0.60	0.35	0.44	1545
6	0.85	0.84	0.84	971
7	0.74	0.88	0.81	864
8	0.22	0.47	0.30	448
9	0.45	0.58	0.51	778
Micro Avg	0.68	0.68	0.68	10000
Macro Avg	0.67	0.69	0.67	10000
Weighted Avg	0.70	0.68	0.68	10000

Figure 71: Confusion matrix and classification report of the neural network model with variational autoencoder after T-FGSM attack

Actual Values \ Predicted Values	0	1	2	3	4	5	6	7	8	9
0	1	4	52	10	22	15	97	6	20	11
1	1	7	104	5	86	3	7	30	34	18
2	266	213	15	217	50	5	180	143	450	11
3	9	85	397	13	6	296	27	398	110	220
4	7	248	61	1	10	0	290	33	11	305
5	45	1	0	302	5	15	171	9	107	53
6	290	19	18	1	42	76	12	0	43	2
7	18	290	130	47	233	4	6	21	48	272
8	62	266	211	192	86	338	161	79	15	97
9	281	2	44	222	442	140	7	309	136	20

	Precision	Recall	F1-Score	Support
0	0.00	0.00	0.00	238
1	0.01	0.02	0.01	295
2	0.01	0.01	0.01	1550
3	0.01	0.01	0.01	1561
4	0.01	0.01	0.01	966
5	0.02	0.02	0.02	708
6	0.01	0.02	0.02	503
7	0.02	0.02	0.02	1069
8	0.02	0.01	0.01	1507
9	0.02	0.01	0.02	1603
Micro Avg	0.01	0.01	0.01	10000
Macro Avg	0.01	0.01	0.01	10000
Weighted Avg	0.01	0.01	0.01	10000

Figure 72: Confusion matrix and classification report of the neural network model without variational autoencoder after basic iterative method attack

MNIST dataset.

Accuracy scores are below compared to MNIST dataset results, but although our epsilon number increased and therefore our perturbation and attack rate increased, accuracy scores for autoencoded models performed better compared to MNIST dataset. Non-Natural and One Target fooling for models without autoencoder performed better. But with Natural Fooling which we have gathered the targets for mostly mispredicted labels from the heatmap,

Actual Values \ Predicted Values	0	1	2	3	4	5	6	7	8	9
0	858	0	8	4	2	6	49	0	3	1
1	0	1068	3	5	0	15	4	3	13	8
2	26	5	780	117	15	53	78	3	35	11
3	9	13	130	638	10	218	3	14	210	11
4	1	1	15	10	575	33	5	73	40	412
5	8	25	36	107	29	266	18	26	336	9
6	74	3	35	8	6	19	796	0	8	3
7	0	0	0	3	9	2	0	750	3	130
8	4	20	22	109	39	266	5	18	316	21
9	0	0	3	9	297	14	0	141	10	403

	Precision	Recall	F1-Score	Support
0	0.88	0.92	0.90	931
1	0.94	0.95	0.95	1119
2	0.76	0.69	0.72	1123
3	0.63	0.51	0.56	1256
4	0.59	0.49	0.54	1165
5	0.30	0.31	0.30	860
6	0.83	0.84	0.83	952
7	0.73	0.84	0.78	897
8	0.32	0.39	0.35	820
9	0.40	0.46	0.43	877
Micro Avg	0.65	0.65	0.65	10000
Macro Avg	0.64	0.64	0.64	10000
Weighted Avg	0.65	0.65	0.65	10000

Figure 73: Confusion matrix and classification report of the neural network model with variational autoencoder after basic iterative method attack

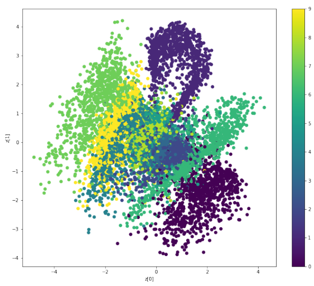


Figure 74: Because of MNIST dataset, our latent space is two-dimensional. One is to look at the neighborhoods of different classes on the latent 2D plane. Each of these colored clusters are a type of digit. Close clusters are digits that are structurally similar, they are digits that share information in the latent space.

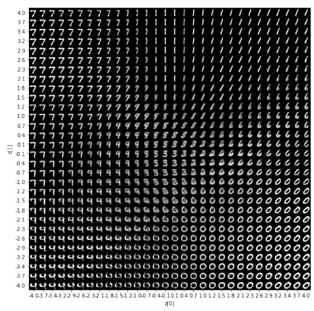


Figure 75: Due to VAE is a generative model, we can also generate new Mnist digits using latent plane, sampling latent points at regular intervals, and generating the corresponding digit for each of these points.

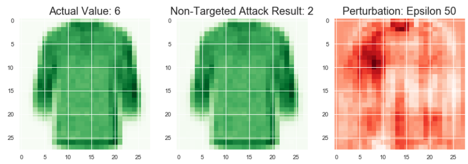


Figure 76: Value change and perturbation of a non-targeted attack on model without autoencoder for Fashion MNIST Dataset

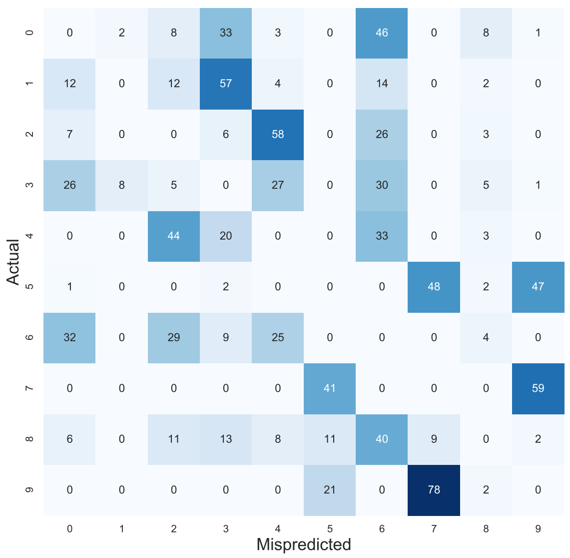


Figure 77: Heatmap of actual numbers and mispredictions for Fashion Mnist

Actual Values	Predicted Values: 0	1	2	3	4	5	6	7	8	9
0	60	9	3	435	27	3	71	0	37	1
1	36	872	16	139	21	0	36	0	6	0
2	102	8	10	46	379	10	424	7	157	1
3	376	65	48	186	96	5	111	2	49	2
4	36	8	49	91	3	1	158	2	33	3
5	8	2	39	1	14	2	7	925	81	807
6	331	19	805	92	478	7	1	1	222	10
7	6	3	0	7	6	501	3	15	21	116
8	35	1	55	29	48	101	53	44	391	47
9	4	0	6	8	5	310	1	38	18	36

Figure 78: Confusion matrix of non-targeted attack to model without autoencoder for Fashion MNIST

the model performed poorly. The models with autoencoder performed better as expected, unrelated to the dataset.

6.2.2 Neural Networks

We will do the same attacks with fashion MNIST dataset without changing the code as we did in the previous section.

We can see the changes with FGSM attack in Figure 83. First line is the dataset before the attack and second line is data after the FGSM attack.

Actual Values	Predicted Values: 0	1	2	3	4	5	6	7	8	9
0	991	0	0	1	1	0	57	0	0	0
1	1	965	0	0	0	0	0	0	0	0
2	0	0	955	1	30	0	58	0	0	0
3	11	2	0	1054	2	0	19	1	0	0
4	0	0	0	0	1060	0	21	0	0	0
5	0	0	0	0	0	923	0	0	0	0
6	0	0	0	0	15	0	735	0	0	0
7	0	0	0	0	0	26	0	1016	0	1
8	1	0	0	1	0	4	2	0	1005	0
9	0	0	0	0	0	25	0	4	0	1012

Figure 79: Confusion matrix of non-targeted attack to model with autoencoder and Fashion MNIST Dataset

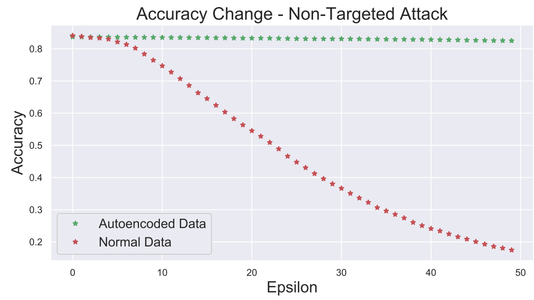


Figure 80: Comparison of accuracy with and without autoencoder for non-targeted attack and Fashion MNIST Dataset

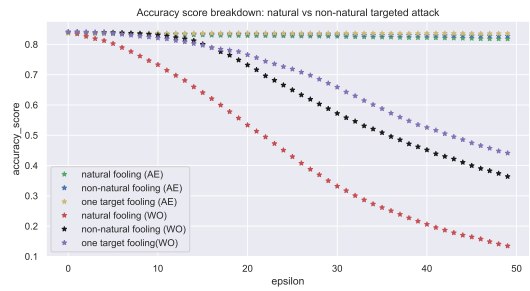


Figure 81: Comparison of accuracy with and without autoencoder for targeted attacks and Fashion MNIST Dataset

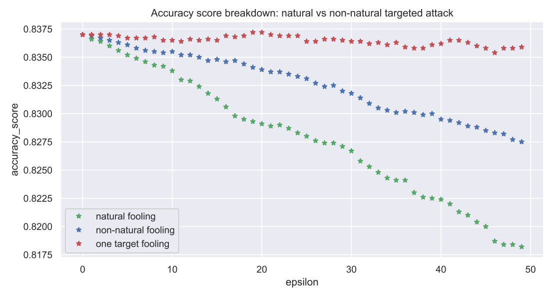


Figure 82: Details of accuracy with autoencoder for targeted attacks and Fashion MNIST Dataset

We can also observe the changes with autoencoder in

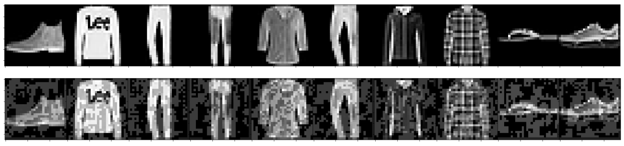


Figure 83: Changes on Fashion MNIST Dataset with FGSM Attack

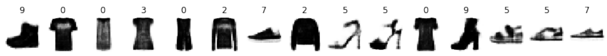


Figure 84: Changes on Fashion MNIST Dataset with FGSM Attack

Actual Values	Predicted Values 0	1	2	3	4	5	6	7	8	9
0	80	165	17	295	21	45	181	0	97	0
1	3	32	2	253	3	0	6	0	3	0
2	35	2	105	51	211	1	177	0	117	0
3	72	710	6	64	51	2	30	0	71	0
4	6	12	215	144	100	0	301	0	139	0
5	5	1	1	0	0	46	1	694	86	359
6	779	72	621	133	583	24	272	0	410	3
7	0	0	0	0	0	459	0	57	52	460
8	20	6	33	60	31	179	32	27	22	133
9	0	0	0	0	0	244	0	222	3	45

	Precision	Recall	F1-Score	Support
0	0.08	0.09	0.08	901
1	0.03	0.11	0.05	302
2	0.10	0.15	0.12	699
3	0.06	0.06	0.06	1006
4	0.10	0.11	0.10	917
5	0.05	0.04	0.04	1193
6	0.27	0.09	0.14	2897
7	0.06	0.06	0.06	1028
8	0.02	0.04	0.03	543
9	0.04	0.09	0.06	514
Micro Avg	0.08	0.08	0.08	10000
Macro Avg	0.08	0.08	0.08	10000
Weighted Avg	0.12	0.08	0.09	10000

Figure 85: Confusion matrix and classification report of the neural network model without autoencoder after FGSM attack for Fashion MNIST dataset

Figure 84.

The changes in autoencoded data are similar to MNIST Data, more transparent on the edges. We will use the FGSM, T-FGSM, and BIM attacks on models that are using Fashion MNIST Dataset and one is autoencoded, the other is not.

6.3 Sparse Autoencoder

6.3.1 Multi-Class Logistic Regression of Sparse Autoencoder

We will demonstrate the robustness of multi-class logistic regression with sparse autoencoders against attacks with Fashion MNIST dataset. We will give the attack results for easier comparison the process of our experiment is obvious at this point.

The sparse autoencoder is still worse than vanilla autoencoder but as with the MNIST dataset, it will perform as the second-best for the Fashion MNIST dataset.

Actual Values	Predicted Values 0	1	2	3	4	5	6	7	8	9
0	814	3	16	27	2	0	166	0	5	0
1	2	966	0	14	1	0	3	0	2	0
2	20	2	769	14	118	0	91	0	8	0
3	26	18	15	871	41	0	23	0	3	1
4	5	3	116	43	741	0	86	0	4	0
5	0	0	0	1	0	942	0	22	1	9
6	118	5	81	25	94	0	615	0	11	0
7	0	0	0	0	0	31	0	941	3	39
8	15	3	3	5	3	3	16	1	962	2
9	0	0	0	0	0	24	0	36	1	949

	Precision	Recall	F1-Score	Support
0	0.81	0.79	0.80	1033
1	0.97	0.98	0.97	988
2	0.77	0.75	0.76	1022
3	0.87	0.87	0.87	998
4	0.74	0.74	0.74	998
5	0.94	0.97	0.95	975
6	0.61	0.65	0.63	949
7	0.94	0.93	0.93	1014
8	0.96	0.95	0.96	1013
9	0.95	0.94	0.94	1010
Micro Avg	0.86	0.86	0.86	10000
Macro Avg	0.86	0.86	0.86	10000
Weighted Avg	0.86	0.86	0.86	10000

Figure 86: Confusion matrix and classification report of the neural network model with autoencoder after FGSM attack for Fashion MNIST dataset

Actual Values	Predicted Values 0	1	2	3	4	5	6	7	8	9
0	3	0	0	0	0	0	0	0	0	0
1	0	0	1	5	1	0	1	0	1	0
2	5	0	1	1	3	0	12	0	0	0
3	22	3	2	4	32	0	13	0	0	0
4	0	0	1	0	0	0	0	0	0	0
5	679	951	779	754	836	1000	913	1000	951	1000
6	286	45	202	234	54	0	38	0	47	0
7	0	0	0	0	0	0	0	0	0	0
8	5	1	14	2	74	0	23	0	1	0
9	0	0	0	0	0	0	0	0	0	0

	Precision	Recall	F1-Score	Support
0	0.00	1.00	0.01	3
1	0.00	0.00	0.00	9
2	0.00	0.05	0.00	22
3	0.00	0.05	0.01	76
4	0.00	0.00	0.00	1
5	1.00	0.11	0.20	8863
6	0.04	0.04	0.04	906
7	0.00	0.00	0.00	0
8	0.00	0.01	0.00	120
9	0.00	0.00	0.00	0
Micro Avg	0.10	0.10	0.10	10000
Macro Avg	0.10	0.13	0.03	10000
Weighted Avg	0.89	0.10	0.18	10000

Figure 87: Confusion matrix and classification report of the neural network model without autoencoder after T-FGSM attack for Fashion MNIST dataset

6.3.2 Neural Network of Sparse Autoencoder

For this section, it is essential to give a confusion matrix and classification report of the neural network model with a sparse autoencoder. Because even without any attack on the model, it labels nearly all of the data as pullovers and sandals.

		Predicted Values									
		0	1	2	3	4	5	6	7	8	9
Actual Values	0	814	5	18	31	1	0	162	0	5	0
	1	2	966	0	11	2	0	3	0	2	0
	2	20	2	806	12	137	0	109	0	10	0
	3	27	17	14	867	36	0	26	0	3	1
	4	6	3	92	50	738	0	82	0	4	0
	5	1	0	0	2	0	953	0	27	3	11
	6	118	5	67	22	83	0	599	0	7	0
	7	0	0	0	0	0	26	0	941	3	39
	8	12	2	3	5	3	2	19	1	963	2
	9	0	0	0	0	0	19	0	31	0	947

	Precision	Recall	F1-Score	Support
0	0.81	0.79	0.80	1036
1	0.97	0.98	0.97	986
2	0.81	0.74	0.77	1096
3	0.87	0.87	0.87	991
4	0.74	0.76	0.75	975
5	0.95	0.96	0.95	997
6	0.60	0.66	0.63	901
7	0.94	0.93	0.94	1009
8	0.96	0.95	0.96	1012
9	0.95	0.95	0.95	997
Micro Avg	0.86	0.86	0.86	10000
Macro Avg	0.86	0.86	0.86	10000
Weighted Avg	0.86	0.86	0.86	10000

Figure 88: Confusion matrix and classification report of the neural network model with autoencoder after T-FGSM attack for Fashion MNIST dataset

		Predicted Values									
		0	1	2	3	4	5	6	7	8	9
Actual Values	0	117	115	10	201	2	9	193	0	28	0
	1	8	7	2	246	5	0	5	0	1	2
	2	37	3	92	81	256	2	250	0	161	0
	3	55	763	14	61	103	0	46	0	65	0
	4	4	17	188	231	121	0	232	0	139	0
	5	5	5	0	1	0	46	0	610	70	276
	6	757	88	685	154	477	3	251	0	397	8
	7	0	0	1	0	0	559	0	42	56	591
	8	17	2	8	25	36	90	23	24	25	74
	9	0	0	0	0	0	291	0	324	58	49

	Precision	Recall	F1-Score	Support
0	0.12	0.17	0.14	675
1	0.01	0.03	0.01	276
2	0.09	0.10	0.10	882
3	0.06	0.06	0.06	1107
4	0.12	0.13	0.13	932
5	0.05	0.05	0.05	1013
6	0.25	0.09	0.13	2820
7	0.04	0.03	0.04	1249
8	0.03	0.08	0.04	324
9	0.05	0.07	0.06	722
Micro Avg	0.08	0.08	0.08	10000
Macro Avg	0.08	0.08	0.07	10000
Weighted Avg	0.12	0.08	0.09	10000

Figure 89: Confusion matrix and classification report of the neural network model without autoencoder after basic iterative method attack for Fashion MNIST dataset

So as it can be seen in from Figure 94 and 95, sparse autoencoder with a neural network is destined to fail from the start.

Sparse autoencoder with fashion MNIST performed surprisingly poorly, even without any attack. Although attack results were unnecessary at this point, we still wanted to show them. Sparse autoencoder's logic to encourage sparsity for classification tasks fails greatly for the Fashion MNIST dataset, which makes sense in a way that clothes on grayscale can be easily confusing. Sparsity turns all the different elements into the same labels. Fashion MNIST dataset after sparse autoencoder is given in Figure 102.

		Predicted Values									
		0	1	2	3	4	5	6	7	8	9
Actual Values	0	805	4	16	28	2	0	162	0	5	0
	1	1	964	0	14	1	0	3	0	2	0
	2	21	0	770	13	129	1	92	0	7	0
	3	23	19	15	869	36	0	23	0	4	1
	4	6	3	115	39	734	0	88	0	3	0
	5	0	0	0	1	0	941	0	23	1	8
	6	129	7	81	30	95	0	613	0	13	0
	7	0	0	0	0	0	29	0	937	5	42
	8	15	3	3	5	3	5	19	1	960	2
	9	0	0	0	1	0	24	0	39	0	947

	Precision	Recall	F1-Score	Support
0	0.81	0.79	0.80	1022
1	0.96	0.98	0.97	985
2	0.77	0.75	0.76	1033
3	0.87	0.88	0.87	990
4	0.73	0.74	0.74	988
5	0.94	0.97	0.95	974
6	0.61	0.63	0.62	968
7	0.94	0.92	0.93	1013
8	0.96	0.94	0.95	1016
9	0.95	0.94	0.94	1011
Micro Avg	0.85	0.85	0.85	10000
Macro Avg	0.85	0.85	0.85	10000
Weighted Avg	0.85	0.85	0.85	10000

Figure 90: Confusion matrix and classification report of the neural network model with autoencoder after basic iterative method attack for Fashion MNIST dataset

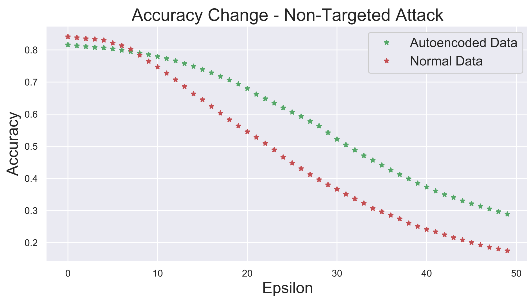


Figure 91: Comparison of accuracy with and without sparse autoencoder for non-targeted attack and Fashion MNIST Dataset

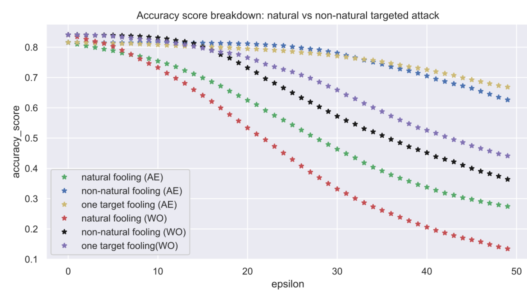


Figure 92: Comparison of accuracy with and without sparse autoencoder for targeted attacks and Fashion MNIST Dataset

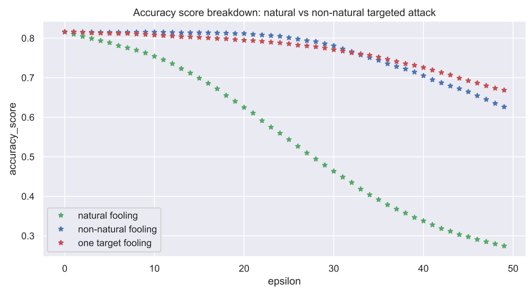


Figure 93: Details of accuracy with sparse autoencoder for sparse targeted attacks and Fashion MNIST Dataset

		Predicted Values									
		0	1	2	3	4	5	6	7	8	9
Actual Values	0	789	1	10	10	1	0	92	0	3	0
	1	0	970	1	4	1	0	2	0	2	0
	2	15	1	760	9	30	0	65	0	1	0
	3	49	20	10	915	36	1	36	0	3	0
	4	4	3	128	30	874	0	84	0	4	0
	5	4	0	0	0	0	962	0	26	2	6
	6	126	4	85	28	56	0	711	0	12	0
	7	0	0	0	0	0	24	0	943	5	40
	8	13	1	6	4	2	1	10	0	968	1
	9	0	0	0	0	0	12	0	31	0	953

	Precision	Recall	F1-Score	Support
0	0.79	0.87	0.83	906
1	0.97	0.99	0.98	980
2	0.76	0.86	0.81	881
3	0.92	0.86	0.88	1070
4	0.87	0.78	0.82	1127
5	0.96	0.96	0.96	1000
6	0.71	0.70	0.70	1022
7	0.94	0.93	0.94	1012
8	0.97	0.96	0.97	1006
9	0.95	0.96	0.95	996
Micro Avg	0.88	0.88	0.88	10000
Macro Avg	0.88	0.89	0.88	10000
Weighted Avg	0.89	0.88	0.88	10000

Figure 94: Confusion matrix and classification report of the neural network model without sparse autoencoder for Fashion MNIST dataset

6.4 Denoising Autoencoder

6.4.1 Multi-Class Logistic Regression of Denoising Autoencoder

Denoising autoencoder with multi-class logistic regression performs as it performed with the MNIST dataset. We do not see too much of a difference with regressions between datasets.

The results for multi-class logistic regression looks kind of similar to sparse autoencoder in the previous section. Let us observe will neural network with denoising autoencoder fails as much as the neural network with sparse autoencoder.

6.4.2 Neural Network of Denoising Autoencoder

The neural network model with denoising autoencoder did not perform as poorly as the neural network model with sparse autoencoder. But as expected, it is still worse than vanilla autoencoder.

		Predicted Values									
		0	1	2	3	4	5	6	7	8	9
Actual Values	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0	0	0	0
	2	996	1000	999	999	999	2	998	0	541	2
	3	0	0	0	0	0	0	0	0	0	0
	4	0	0	0	0	0	0	0	0	0	0
	5	4	0	1	1	1	998	2	1000	459	998
	6	0	0	0	0	0	0	0	0	0	0
	7	0	0	0	0	0	0	0	0	0	0
	8	0	0	0	0	0	0	0	0	0	0
	9	0	0	0	0	0	0	0	0	0	0

	Precision	Recall	F1-Score	Support
0	0.00	0.00	0.00	0
1	0.00	0.00	0.00	0
2	1.00	0.15	0.27	6536
3	0.00	0.00	0.00	0
4	0.00	0.00	0.00	0
5	1.00	0.29	0.45	3464
6	0.00	0.00	0.00	0
7	0.00	0.00	0.00	0
8	0.00	0.00	0.00	0
9	0.00	0.00	0.00	0
Micro Avg	0.20	0.20	0.20	10000
Macro Avg	0.20	0.04	0.07	10000
Weighted Avg	1.00	0.20	0.33	10000

Figure 95: Confusion matrix and classification report of the neural network model with sparse autoencoder for Fashion MNIST dataset

		Predicted Values									
		0	1	2	3	4	5	6	7	8	9
Actual Values	0	129	11	12	244	4	7	177	1	28	0
	1	5	15	1	196	1	0	4	0	1	0
	2	23	0	116	42	166	1	121	0	38	0
	3	81	890	27	69	117	2	95	0	132	1
	4	11	20	189	228	76	0	324	0	188	0
	5	2	0	1	1	0	38	0	714	74	413
	6	729	62	645	171	614	7	251	0	463	7
	7	0	0	0	0	0	481	0	55	33	499
	8	19	2	9	49	22	189	28	9	23	41
	9	1	0	0	0	0	275	0	221	20	39

	Precision	Recall	F1-Score	Support
0	0.13	0.21	0.16	613
1	0.01	0.07	0.02	223
2	0.12	0.23	0.15	507
3	0.07	0.05	0.06	1414
4	0.08	0.07	0.07	1036
5	0.04	0.03	0.03	1243
6	0.25	0.09	0.13	2949
7	0.06	0.05	0.05	1068
8	0.02	0.06	0.03	391
9	0.04	0.07	0.05	556
Micro Avg	0.08	0.08	0.08	10000
Macro Avg	0.08	0.09	0.08	10000
Weighted Avg	0.12	0.08	0.09	10000

Figure 96: Confusion matrix and classification report of the neural network model without sparse autoencoder after FGSM attack for Fashion MNIST dataset

6.5 Variational Autoencoder

6.5.1 Multi-Class Logistic Regression of Variational Autoencoder

The multi-class regression with variational autoencoder performs poorly, as it was in MNIST Dataset which was

		Predicted Values									
		0	1	2	3	4	5	6	7	8	9
Actual Values	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0	0	0	0
	2	996	1000	999	999	999	2	998	0	547	2
	3	0	0	0	0	0	0	0	0	0	0
	4	0	0	0	0	0	0	0	0	0	0
	5	4	0	1	1	1	998	2	1000	453	998
	6	0	0	0	0	0	0	0	0	0	0
	7	0	0	0	0	0	0	0	0	0	0
	8	0	0	0	0	0	0	0	0	0	0
	9	0	0	0	0	0	0	0	0	0	0

	Precision	Recall	F1-Score	Support
0	0.00	0.00	0.00	0
1	0.00	0.00	0.00	0
2	1.00	0.15	0.26	6542
3	0.00	0.00	0.00	0
4	0.00	0.00	0.00	0
5	1.00	0.29	0.45	3458
6	0.00	0.00	0.00	0
7	0.00	0.00	0.00	0
8	0.00	0.00	0.00	0
9	0.00	0.00	0.00	0
Micro Avg	0.20	0.20	0.20	10000
Macro Avg	0.20	0.04	0.07	10000
Weighted Avg	1.00	0.20	0.33	10000

Figure 97: Confusion matrix and classification report of the neural network model with sparse autoencoder after FGSM attack for Fashion MNIST dataset

		Predicted Values									
		0	1	2	3	4	5	6	7	8	9
Actual Values	0	7	0	0	2	0	0	4	0	0	0
	1	0	0	1	1	1	0	0	0	0	0
	2	0	0	0	0	0	0	0	0	0	0
	3	31	238	10	10	43	0	64	0	0	0
	4	0	0	1	0	0	0	28	0	0	0
	5	439	552	558	654	416	1000	792	1000	787	1000
	6	521	210	430	333	540	0	111	0	213	0
	7	0	0	0	0	0	0	0	0	0	0
	8	2	0	0	0	0	0	1	0	0	0
	9	0	0	0	0	0	0	0	0	0	0

	Precision	Recall	F1-Score	Support
0	0.01	0.54	0.01	13
1	0.00	0.00	0.00	3
2	0.00	0.00	0.00	0
3	0.01	0.03	0.01	396
4	0.00	0.00	0.00	29
5	1.00	0.14	0.24	7198
6	0.11	0.05	0.07	2358
7	0.00	0.00	0.00	0
8	0.00	0.00	0.00	3
9	0.00	0.00	0.00	0
Micro Avg	0.11	0.11	0.11	10000
Macro Avg	0.11	0.07	0.03	10000
Weighted Avg	0.75	0.11	0.19	10000

Figure 98: Confusion matrix and classification report of the neural network model without sparse autoencoder after T-FGSM attack for Fashion MNIST dataset

expected. Again, due to the variational autoencoder's structure and purpose of use, it is not a fit for defensive measures against attacks.

6.5.2 Neural Network of Variational Autoencoder

We do not observe the problem we have encountered with sparse autoencoder in the neural networks with a variational autoencoder. They still perform poorly and would be

		Predicted Values									
		0	1	2	3	4	5	6	7	8	9
Actual Values	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0	0	0	0
	2	996	1000	999	999	999	1	998	0	533	2
	3	0	0	0	0	0	0	0	0	0	0
	4	0	0	0	0	0	0	0	0	0	0
	5	4	0	1	1	1	999	2	1000	467	998
	6	0	0	0	0	0	0	0	0	0	0
	7	0	0	0	0	0	0	0	0	0	0
	8	0	0	0	0	0	0	0	0	0	0
	9	0	0	0	0	0	0	0	0	0	0

	Precision	Recall	F1-Score	Support
0	0.00	0.00	0.00	0
1	0.00	0.00	0.00	0
2	1.00	0.15	0.27	6527
3	0.00	0.00	0.00	0
4	0.00	0.00	0.00	0
5	1.00	0.29	0.45	3473
6	0.00	0.00	0.00	0
7	0.00	0.00	0.00	0
8	0.00	0.00	0.00	0
9	0.00	0.00	0.00	0
Micro Avg	0.20	0.20	0.20	10000
Macro Avg	0.20	0.04	0.07	10000
Weighted Avg	1.00	0.20	0.33	10000

Figure 99: Confusion matrix and classification report of the neural network model with sparse autoencoder after T-FGSM attack for Fashion MNIST dataset

		Predicted Values									
		0	1	2	3	4	5	6	7	8	9
Actual Values	0	127	12	9	236	4	5	180	1	29	0
	1	6	14	2	215	1	0	4	0	2	0
	2	31	1	130	70	197	1	157	0	43	0
	3	71	900	22	50	115	0	74	0	119	1
	4	13	21	189	252	78	0	315	0	193	0
	5	3	0	1	1	0	37	0	720	83	412
	6	725	50	637	135	586	5	246	0	459	7
	7	1	0	1	0	0	507	0	55	34	508
	8	22	2	9	41	19	189	24	8	16	33
	9	1	0	0	0	0	256	0	216	22	39

	Precision	Recall	F1-Score	Support
0	0.13	0.21	0.16	603
1	0.01	0.06	0.02	244
2	0.13	0.21	0.16	630
3	0.05	0.04	0.04	1352
4	0.08	0.07	0.08	1061
5	0.04	0.03	0.03	1257
6	0.25	0.09	0.13	2850
7	0.06	0.05	0.05	1106
8	0.02	0.04	0.02	363
9	0.04	0.07	0.05	534
Micro Avg	0.08	0.08	0.08	10000
Macro Avg	0.08	0.09	0.07	10000
Weighted Avg	0.11	0.08	0.08	10000

Figure 100: Confusion matrix and classification report of the neural network model without sparse autoencoder after basic iterative method attack for Fashion MNIST dataset

the last most accurate autoencoder type if sparse autoencoder did not perform so poorly with Fashion MNIST.

We also used the generative aspect of the variational autoencoder for Fashion MNIST Dataset.

We believe that the most significant strength of our model is that the natural practice of implementing an autoencoder between data and machine learning models can provide considerable defense and robustness against attacks and it provide high generalization property, in con-

Actual Values \ Predicted Values	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0
2	996	1000	999	999	999	1	998	0	548	2
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	4	0	1	1	1	999	2	1000	452	998
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0

	Precision	Recall	F1-Score	Support
0	0.00	0.00	0.00	0
1	0.00	0.00	0.00	0
2	1.00	0.15	0.26	6542
3	0.00	0.00	0.00	0
4	0.00	0.00	0.00	0
5	1.00	0.29	0.45	3458
6	0.00	0.00	0.00	0
7	0.00	0.00	0.00	0
8	0.00	0.00	0.00	0
9	0.00	0.00	0.00	0
Micro Avg	0.20	0.20	0.20	10000
Macro Avg	0.20	0.04	0.07	10000
Weighted Avg	1.00	0.20	0.33	10000

Figure 101: Confusion matrix and classification report of the neural network model with autoencoder after basic iterative method attack for Fashion MNIST dataset

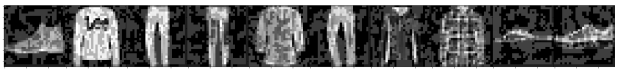


Figure 102: Fashion MNIST Data after being through Sparse Autoencoder

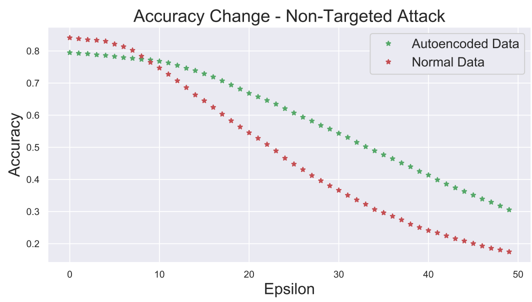


Figure 103: Comparison of accuracy with and without denoising autoencoder for non-targeted attack and Fashion MNIST Dataset

trast to most prior adversarial learning methods. An important feature of our methodology is that not only presents a generic resistance to specific attack methods but also provides robustness to machine learning models in general.

7 Discussion

In linear machine learning model algorithms, we applied non-targeted and targeted attacks to multiclass logistic regression machine learning models to observe the changes and difference between attack methods. Moreover, FGSM, T-FGSM, and BIM attacks have been used for the neural network machine learning model. The effects of these attacks on implementing autoencoder as a filter have been examined for these machine learning models.

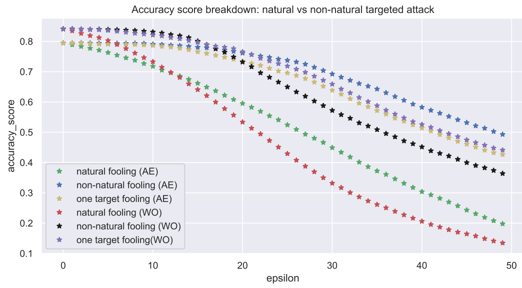


Figure 104: Comparison of accuracy with and without denoising autoencoder for targeted attacks and Fashion MNIST Dataset

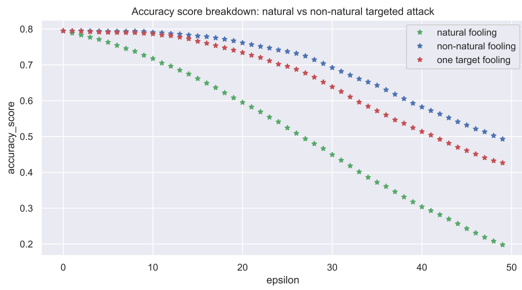


Figure 105: Details of accuracy with autoencoder for denoising targeted attacks and Fashion MNIST Dataset

Actual Values \ Predicted Values	0	1	2	3	4	5	6	7	8	9
0	80	165	17	295	21	45	181	0	97	0
1	3	32	2	253	3	0	6	0	3	0
2	35	2	105	51	211	1	177	0	117	0
3	72	710	6	64	51	2	30	0	71	0
4	6	12	215	144	100	0	301	0	139	0
5	5	1	1	0	0	46	1	694	86	359
6	779	72	621	133	583	24	272	0	410	3
7	0	0	0	0	0	459	0	57	52	460
8	20	6	33	60	31	179	32	27	22	133
9	0	0	0	0	0	244	0	222	3	45

	Precision	Recall	F1-Score	Support
0	0.08	0.09	0.08	901
1	0.03	0.11	0.05	302
2	0.10	0.15	0.12	699
3	0.06	0.06	0.06	1006
4	0.10	0.11	0.10	917
5	0.05	0.04	0.04	1193
6	0.27	0.09	0.14	2897
7	0.06	0.06	0.06	1028
8	0.02	0.04	0.03	543
9	0.04	0.09	0.06	514
Micro Avg	0.08	0.08	0.08	10000
Macro Avg	0.08	0.08	0.08	10000
Weighted Avg	0.12	0.08	0.09	10000

Figure 106: Confusion matrix and classification report of the neural network model without denoising autoencoder after FGSM attack for Fashion MNIST dataset

Actual Values \ Predicted Values	0	1	2	3	4	5	6	7	8	9
0	818	11	17	29	6	0	137	0	5	0
1	4	964	4	7	1	0	3	0	2	0
2	22	2	759	13	115	0	100	0	2	0
3	18	17	8	875	45	0	24	0	4	0
4	2	1	116	20	719	0	71	0	7	0
5	1	0	0	0	0	950	0	28	2	8
6	127	3	94	51	110	0	654	1	12	1
7	0	0	0	0	0	36	0	949	6	33
8	8	2	2	5	4	1	10	0	959	1
9	0	0	0	0	0	13	1	22	1	957

	Precision	Recall	F1-Score	Support
0	0.82	0.80	0.81	1023
1	0.96	0.98	0.97	985
2	0.76	0.75	0.75	1013
3	0.88	0.88	0.88	991
4	0.72	0.77	0.74	936
5	0.95	0.96	0.96	989
6	0.65	0.62	0.64	1053
7	0.95	0.93	0.94	1024
8	0.96	0.97	0.96	992
9	0.96	0.96	0.96	994
Micro Avg	0.86	0.86	0.86	10000
Macro Avg	0.86	0.86	0.86	10000
Weighted Avg	0.86	0.86	0.86	10000

Figure 107: Confusion matrix and classification report of the neural network model with denoising autoencoder after FGSM attack for Fashion MNIST dataset

Actual Values \ Predicted Values	0	1	2	3	4	5	6	7	8	9
0	22	3	2	128	2	0	22	0	1	0
1	0	0	0	0	1	0	0	0	0	0
2	0	0	1	0	0	0	2	0	1	0
3	50	14	2	8	5	0	11	0	0	0
4	0	0	0	0	0	0	9	0	0	0
5	695	952	824	840	828	1000	876	1000	939	999
6	213	13	124	5	93	0	25	0	42	0
7	0	0	0	0	0	0	0	0	0	0
8	20	18	46	19	71	0	55	0	1	1
9	0	0	1	0	0	0	0	0	16	0

	Precision	Recall	F1-Score	Support
0	0.02	0.12	0.04	180
1	0.00	0.00	0.00	1
2	0.00	0.25	0.00	4
3	0.01	0.09	0.01	90
4	0.00	0.00	0.00	9
5	1.00	0.11	0.20	8953
6	0.03	0.05	0.03	515
7	0.00	0.00	0.00	0
8	0.00	0.00	0.00	231
9	0.00	0.00	0.00	17
Micro Avg	0.11	0.11	0.11	10000
Macro Avg	0.11	0.06	0.03	10000
Weighted Avg	0.90	0.11	0.18	10000

Figure 108: Confusion matrix and classification report of the neural network model without denoising autoencoder after T-FGSM attack for Fashion MNIST dataset

Autoencoders provide robustness against adversarial machine learning attacks to machine learning models for both linear models and neural network models. In this study, we presented that the natural practice of implementing an autoencoder between data and machine learning models can lead significant defense and robustness against attacks.

Actual Values \ Predicted Values	0	1	2	3	4	5	6	7	8	9
0	807	12	13	29	3	0	122	0	3	0
1	4	960	4	5	1	0	3	0	1	0
2	21	1	735	14	103	0	89	0	5	0
3	19	19	7	869	46	0	26	0	3	0
4	2	1	137	25	737	0	70	0	6	0
5	1	0	0	0	0	970	1	40	6	12
6	140	5	101	51	106	0	679	0	15	1
7	0	0	0	0	0	21	0	938	6	34
8	6	2	3	7	4	0	8	0	954	0
9	0	0	0	0	0	9	2	22	1	953

	Precision	Recall	F1-Score	Support
0	0.81	0.82	0.81	989
1	0.96	0.98	0.97	978
2	0.73	0.76	0.75	968
3	0.87	0.88	0.87	989
4	0.74	0.75	0.75	978
5	0.97	0.94	0.96	1030
6	0.68	0.62	0.65	1098
7	0.94	0.94	0.94	999
8	0.95	0.97	0.96	984
9	0.95	0.97	0.96	987
Micro Avg	0.86	0.86	0.86	10000
Macro Avg	0.86	0.86	0.86	10000
Weighted Avg	0.86	0.86	0.86	10000

Figure 109: Confusion matrix and classification report of the neural network model with denoising autoencoder after T-FGSM attack for Fashion MNIST dataset

Actual Values \ Predicted Values	0	1	2	3	4	5	6	7	8	9
0	72	162	14	290	21	22	176	0	96	0
1	3	15	2	257	4	0	6	0	3	0
2	44	2	109	58	259	2	192	0	151	0
3	70	708	6	57	41	2	26	0	58	0
4	7	15	234	163	104	0	305	0	149	0
5	5	1	1	0	0	32	1	701	89	370
6	779	89	609	119	542	23	268	0	385	3
7	0	0	0	0	0	489	0	57	53	450
8	20	8	25	56	29	161	26	23	10	132
9	0	0	0	0	0	269	0	219	6	45

	Precision	Recall	F1-Score	Support
0	0.07	0.08	0.08	853
1	0.01	0.05	0.02	290
2	0.11	0.13	0.12	817
3	0.06	0.06	0.06	968
4	0.10	0.11	0.11	977
5	0.03	0.03	0.03	1200
6	0.27	0.10	0.14	2817
7	0.06	0.05	0.06	1049
8	0.01	0.02	0.01	490
9	0.04	0.08	0.06	539
Micro Avg	0.08	0.08	0.08	10000
Macro Avg	0.08	0.07	0.07	10000
Weighted Avg	0.12	0.08	0.09	10000

Figure 110: Confusion matrix and classification report of the neural network model without denoising autoencoder after basic iterative method attack for Fashion MNIST dataset

8 Conclusion

In this paper, we have presented the results for pre-filtering the data with an autoencoder before sending it to the machine learning model against adversarial machine learning attacks. We have investigated that the classifier accuracy changes for linear and neural network machine learning models. We have also applied non-targeted and targeted

Actual Values \ Predicted Values	0	1	2	3	4	5	6	7	8	9
0	803	11	16	32	10	0	138	0	5	0
1	5	961	4	5	2	0	2	0	1	0
2	25	2	754	14	119	0	95	0	2	0
3	19	21	9	875	48	0	25	0	3	0
4	1	1	116	19	692	0	68	0	7	0
5	1	0	0	0	0	941	1	28	3	9
6	136	3	99	49	124	0	659	1	10	1
7	0	0	0	0	0	41	0	943	5	30
8	10	1	2	6	5	2	11	2	963	1
9	0	0	0	0	0	16	1	26	1	959

	Precision	Recall	F1-Score	Support
0	0.80	0.79	0.80	1015
1	0.96	0.98	0.97	980
2	0.75	0.75	0.75	1011
3	0.88	0.88	0.88	1000
4	0.69	0.77	0.73	904
5	0.94	0.96	0.95	983
6	0.66	0.61	0.63	1082
7	0.94	0.93	0.93	1019
8	0.96	0.96	0.96	1003
9	0.96	0.96	0.96	1003
Micro Avg	0.85	0.85	0.85	10000
Macro Avg	0.86	0.86	0.86	10000
Weighted Avg	0.85	0.85	0.85	10000

Figure 111: Confusion matrix and classification report of the neural network model with denoising autoencoder after basic iterative method attack for Fashion MNIST dataset

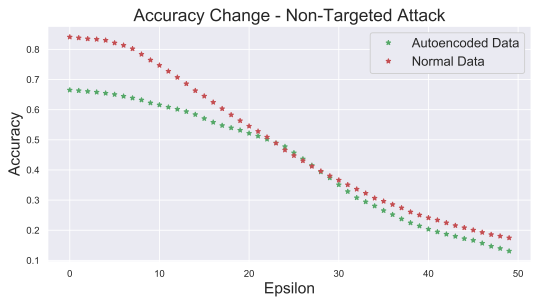


Figure 112: Comparison of accuracy with and without variational autoencoder for non-targeted attack and Fashion MNIST Dataset

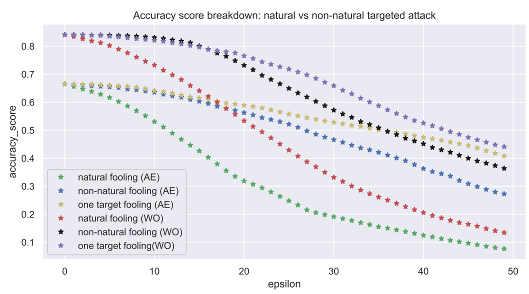


Figure 113: Comparison of accuracy with and without variational autoencoder for targeted attacks and Fashion MNIST Dataset

attacks to multi-class logistic regression. Besides, FGSM, T-FGSM, and BIM attacks have been applied to the neural network machine learning model. The effects of these

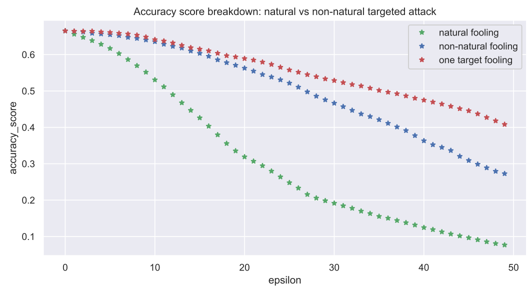


Figure 114: Details of accuracy with autoencoder for variational targeted attacks and Fashion MNIST Dataset

Actual Values \ Predicted Values	0	1	2	3	4	5	6	7	8	9
0	124	130	11	205	6	16	194	0	66	0
1	3	10	1	185	1	0	2	0	2	0
2	27	1	124	43	144	23	218	0	82	0
3	81	743	19	61	93	4	83	0	162	1
4	3	44	143	218	143	0	285	0	46	0
5	3	0	1	0	0	35	2	623	58	351
6	739	52	672	234	596	1	194	0	481	3
7	0	0	0	1	0	579	0	32	67	580
8	19	20	29	53	17	107	22	26	24	15
9	1	0	0	0	0	235	0	319	12	50

	Precision	Recall	F1-Score	Support
0	0.12	0.16	0.14	752
1	0.01	0.05	0.02	204
2	0.12	0.19	0.15	662
3	0.06	0.05	0.05	1247
4	0.14	0.16	0.15	882
5	0.04	0.03	0.03	1073
6	0.19	0.07	0.10	2972
7	0.03	0.03	0.03	1259
8	0.02	0.07	0.04	332
9	0.05	0.08	0.06	617
Micro Avg	0.08	0.08	0.08	10000
Macro Avg	0.08	0.09	0.08	10000
Weighted Avg	0.11	0.08	0.08	10000

Figure 115: Confusion matrix and classification report of the neural network model without variational autoencoder after FGSM attack for Fashion MNIST dataset

attacks on implementing autoencoder as a filter have been analyzed for both machine learning models. We have observed that the robustness provided by autoencoder after adversarial attacks can be seen by accuracy drop between 0.1 and 0.2 percent while the models without autoencoder suffered tremendous accuracy drops hitting accuracy score between 0.6 and 0.3 in some cases even 0.1. We have proposed general, generic, and easy to implement protection against adversarial machine learning model attacks. It will be beneficial to remind that all autoencoders in this study were trained with the epoch of 35 with 1024 sized batches, so the results can be improved by increasing the number of epochs. In conclusion, we have discussed that autoencoders provide robustness against adversarial machine learning attacks to machine learning models for both linear models and neural network models. We have examined the other types of autoencoders, which are mostly called vanilla autoencoders, give the best results. The second most accurate autoencoder type is sparse autoencoders, and

		Predicted Values									
		0	1	2	3	4	5	6	7	8	9
Actual Values	0	727	21	10	92	29	0	205	0	21	0
	1	1	800	1	157	4	0	1	0	1	0
	2	38	7	536	21	369	0	314	0	62	0
	3	113	148	6	611	59	0	73	0	34	1
	4	25	13	204	52	381	0	135	0	10	0
	5	1	0	0	0	0	711	0	190	9	90
	6	66	6	173	57	142	2	183	0	35	0
	7	1	0	0	0	0	230	0	721	7	76
	8	28	5	70	10	16	4	89	5	820	2
	9	0	0	0	0	0	53	0	84	1	831

	Precision	Recall	F1-Score	Support
0	0.73	0.66	0.69	1105
1	0.80	0.83	0.81	965
2	0.54	0.40	0.46	1347
3	0.61	0.58	0.60	1045
4	0.38	0.46	0.42	820
5	0.71	0.71	0.71	1001
6	0.18	0.28	0.22	664
7	0.72	0.70	0.71	1035
8	0.82	0.78	0.80	1049
9	0.83	0.86	0.84	969
Micro Avg	0.63	0.63	0.63	10000
Macro Avg	0.63	0.63	0.63	10000
Weighted Avg	0.65	0.63	0.64	10000

Figure 116: Confusion matrix and classification report of the neural network model with variational autoencoder after FGSM attack for Fashion MNIST dataset

		Predicted Values									
		0	1	2	3	4	5	6	7	8	9
Actual Values	0	12	24	1	18	9	0	34	0	11	0
	1	2	1	1	57	1	0	2	0	0	0
	2	26	0	10	4	15	0	99	0	66	0
	3	27	8	2	1	8	0	40	0	0	0
	4	3	1	0	6	4	0	19	0	3	0
	5	566	935	655	720	497	1000	745	1000	893	1000
	6	352	2	310	174	334	0	41	0	26	0
	7	0	0	0	0	0	0	0	0	0	0
	8	12	29	21	20	132	0	20	0	1	0
	9	0	0	0	0	0	0	0	0	0	0

	Precision	Recall	F1-Score	Support
0	0.01	0.11	0.02	109
1	0.00	0.02	0.00	64
2	0.01	0.05	0.02	220
3	0.00	0.01	0.00	86
4	0.00	0.11	0.01	36
5	1.00	0.12	0.22	8011
6	0.04	0.03	0.04	1239
7	0.00	0.00	0.00	0
8	0.00	0.00	0.00	235
9	0.00	0.00	0.00	0
Micro Avg	0.11	0.11	0.11	10000
Macro Avg	0.11	0.05	0.03	10000
Weighted Avg	0.81	0.11	0.18	10000

Figure 117: Confusion matrix and classification report of the neural network model without variational autoencoder after T-FGSM attack for Fashion MNIST dataset

		Predicted Values									
		0	1	2	3	4	5	6	7	8	9
Actual Values	0	712	16	10	67	17	1	197	0	20	0
	1	0	835	1	154	1	0	1	0	1	0
	2	43	8	627	30	399	0	371	0	63	0
	3	99	108	5	603	59	0	49	0	26	1
	4	23	12	198	53	399	0	130	0	9	0
	5	2	2	0	1	0	939	1	401	19	179
	6	93	14	96	82	106	0	167	0	41	0
	7	0	0	0	0	0	35	0	512	2	18
	8	28	5	63	10	19	1	84	5	818	1
	9	0	0	0	0	0	24	0	82	1	801

	Precision	Recall	F1-Score	Support
0	0.71	0.68	0.70	1040
1	0.83	0.84	0.84	993
2	0.63	0.41	0.49	1541
3	0.60	0.63	0.62	950
4	0.40	0.48	0.44	824
5	0.94	0.61	0.74	1544
6	0.17	0.28	0.21	599
7	0.51	0.90	0.65	567
8	0.82	0.79	0.80	1034
9	0.80	0.88	0.84	908
Micro Avg	0.64	0.64	0.64	10000
Macro Avg	0.64	0.65	0.63	10000
Weighted Avg	0.69	0.64	0.65	10000

Figure 118: Confusion matrix and classification report of the neural network model with variational autoencoder after T-FGSM attack for Fashion MNIST dataset

		Predicted Values									
		0	1	2	3	4	5	6	7	8	9
Actual Values	0	117	125	8	192	5	14	185	0	66	0
	1	4	10	1	187	1	0	2	0	2	0
	2	37	1	143	57	181	19	251	0	98	0
	3	81	736	21	57	96	3	77	0	150	0
	4	6	69	172	247	148	0	271	0	60	0
	5	5	0	1	0	0	31	2	638	64	352
	6	732	50	643	220	554	1	193	0	453	1
	7	0	0	0	0	0	591	0	32	76	584
	8	17	9	11	40	15	99	19	26	18	14
	9	1	0	0	0	0	242	0	304	13	49

	Precision	Recall	F1-Score	Support
0	0.12	0.16	0.14	712
1	0.01	0.05	0.02	207
2	0.14	0.18	0.16	787
3	0.06	0.05	0.05	1221
4	0.15	0.15	0.15	973
5	0.03	0.03	0.03	1093
6	0.19	0.07	0.10	2847
7	0.03	0.02	0.03	1283
8	0.02	0.07	0.03	268
9	0.05	0.08	0.06	609
Micro Avg	0.08	0.08	0.08	10000
Macro Avg	0.08	0.09	0.08	10000
Weighted Avg	0.11	0.08	0.08	10000

Figure 119: Confusion matrix and classification report of the neural network model without variational autoencoder after basic iterative method attack for Fashion MNIST dataset

the third most accurate is denoising autoencoders, which gives similar results with the sparse autoencoders. We have observed that the worst autoencoder type for this process is variational autoencoders because variational autoencoders are generative models used in different areas.

In summary, the natural practice of implementing an autoencoder between data and machine learning models can provide considerable defense and robustness against attacks. These autoencoders can be easily implemented with libraries such as TensorFlow and Keras. Through the results of this review, it is evident that autoencoders can be used in any machine learning model easily because of their implementation as a separate layer.

Actual Values \ Predicted Values	0	1	2	3	4	5	6	7	8	9
0	677	21	11	108	29	0	182	0	18	0
1	6	719	0	220	5	0	2	0	0	0
2	36	8	498	17	358	0	288	0	54	0
3	86	214	5	479	46	0	58	0	33	1
4	31	17	197	78	374	0	131	0	12	0
5	2	1	0	1	0	529	0	256	12	107
6	134	15	214	86	173	2	251	0	45	0
7	0	0	0	0	0	400	0	642	8	98
8	28	5	75	11	15	4	88	6	817	2
9	0	0	0	0	0	65	0	96	1	792

	Precision	Recall	F1-Score	Support
0	0.68	0.65	0.66	1046
1	0.72	0.76	0.74	952
2	0.50	0.40	0.44	1259
3	0.48	0.52	0.50	922
4	0.37	0.45	0.41	840
5	0.53	0.58	0.55	908
6	0.25	0.27	0.26	920
7	0.64	0.56	0.60	1148
8	0.82	0.78	0.80	1051
9	0.79	0.83	0.81	954
Micro Avg	0.58	0.58	0.58	10000
Macro Avg	0.58	0.58	0.58	10000
Weighted Avg	0.58	0.58	0.58	10000

Figure 120: Confusion matrix and classification report of the neural network model with variational autoencoder after basic iterative method attack for Fashion MNIST dataset

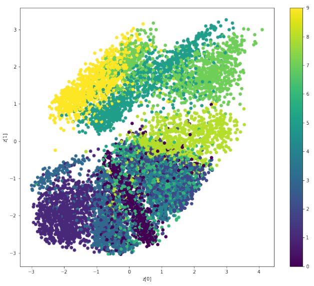


Figure 121: Because of Fashion MNIST dataset, our latent space is two-dimensional. One is to look at the neighborhoods of different classes on the latent 2D plane. Each of these colored clusters are a type of digit. Close clusters are digits that are structurally similar, they are digits that share information in the latent space.

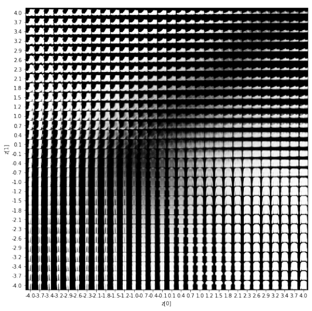


Figure 122: Due to VAE is a generative model, we can also generate new Mnist digits using latent plane, sampling latent points at regular intervals, and generating the corresponding digit for each of these points.

References

[1] A. F. Agarap. Deep learning using rectified linear units (relu). *CoRR*, abs/1803.08375, 2018.

[2] Sharar Ahmadi, Mehran S Fallah, and Massoud Pourmahdian. On the properties of epistemic and temporal epistemic logics of authentication. *Informatica*, 43(2), 2019. doi: 10.31449/inf.v43i2.1617.

[3] M. Aladag, F. O. Catak, and E. Gul. Preventing data poisoning attacks by using generative models. In *2019 1st International Informatics and Software Engineering Conference (UBMYK)*, pages 1–5, 2019. doi: 10.1109/UBMYK48245.2019.8965459.

[4] M. Juuti, B. G. Atli, N. Asokan. Making targeted black-box evasion attacks effective and efficient. *CoRR*, abs/1906.03397, 2019.

[5] W. Bai, C. Quan, and Z. Luo. Alleviating adversarial attacks via convolutional autoencoder. In *2017 18th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, pages 53–58, 2017. doi: 10.1109/SNPD.2017.8022700.

[6] M. Bakator and D. Radosav. Deep learning and medical diagnosis: A review of literature. *Multimodal Technologies and Interaction*, 2:47, 2018. doi: 10.3390/mti2030047.

[7] Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent tool use from multi-agent autocurricula. *arXiv preprint arXiv:1909.07528*, 2019.

[8] X. Yuan, P. He, Q. Zhu, R. R. Bhat and X. Li. Adversarial examples: Attacks and defenses for deep learning. *CoRR*, abs/1712.07107, July 2017.

[9] Financial Stability Board. Artificial intelligence and machine learning in financial services: Market developments and financial stability implications. *Financial Stability Board*, page 45, 2017.

[10] A. Athalye, N. Carlini and D. A. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *CoRR*, abs/1802.00420, February 2018.

[11] N. Carlini and D. A. Wagner. Towards evaluating the robustness of neural networks. *CoRR*, abs/1608.04644, 2016. URL `http://arxiv.org/abs/1608.04644`.

[12] Ferhat Ozgur Catak and Ahmet Fatih Mustacoglu. Distributed denial of service attack detection using autoencoder and deep neural networks. *Journal of Intelligent & Fuzzy Systems*, 37(3):3969–3979, 2019. doi: 10.3233/JIFS-190159.

[13] I. Chen and B. Sirkeci-Mergen. A comparative study of autoencoders against adversarial attacks. *nt'l Conf. IP, Comp. Vision, and Pattern Recognition*, 2018.

[14] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[15] Cherifi Dalila, Boushaba Saddek, Nait-Ali Amine, et al. Feature level fusion of face and voice biometrics systems using artificial neural network for personal recognition. *Informatica*, 44(1), 2020. doi: 10.31449/inf.v44i1.2596.

[16] Murat Dikmen and Catherine M. Burns. Autonomous driving in the real world: Experiences with tesla autopilot and summon. In *Proceedings of the 8th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, Automotive'UI 16, page 225–228, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450345330. doi: 10.1145/3003715.3005465.

[17] Samuel G Finlayson, John D Bowers, Joichi Ito, Jonathan L Zittrain, Andrew L Beam, and Isaac S Kohane. Adversarial attacks on medical machine learning. *Science*, 363(6433):1287–1289, 2019.

[18] C. Nwankpa, W. Ijomah, A. Gachagan and S. Marshall. Activation functions: Comparison of trends in practice and research for deep learning. *CoRR*, abs/1811.03378, November 2018.

[19] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[20] Jun Han and Claudio Moraga. The influence of the sigmoid function parameters on the speed of backpropagation learning. In *International Workshop on Artificial Neural Networks*, pages 195–201. Springer, 1995. doi: 10.1007/3-540-59497-3_175.

[21] Hao Zheng, Zhanlei Yang, Wenju Liu, Jizhong Liang, and Yanpeng Li. Improving deep neural networks using softplus units. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–4, 2015. doi: 10.1109/IJCNN.2015.7280459.

[22] Samuel Harding, Prashanth Rajivan, Bennett I Bertenthal, and Cleotilde Gonzalez. Human decisions on targeted and non-targeted adversarial sample. In *CogSci*, 2018.

[23] M. Isakov, V. Gadepally, K. M. Gettings, and M. A. Kinsy. Survey of attacks and defenses on edge-deployed neural networks. In *2019 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–8, 2019.

[24] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 19–35, 2018. doi: 10.1109/SP.2018.00057.

[25] J. Guo, Y. Zhao, X. Han, Y. Jiang and J. Sun. Rnn-test: Adversarial testing framework for recurrent neural network systems. *CoRR*, November 2019.

[26] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, December 2014.

[27] K. Auernhammer, R. T. Kolagari and M. Zoppelt. Attacks on machine learning: Lurking danger for accountability. *CoRR*, January 2019.

[28] B. Li and Y. Vorobeychik. Evasion-robust classification on binary domains. *ACM Trans. Knowl. Discov. Data*, 12(4):50:1–50:32, 2018. ISSN 1556-4681. doi: 10.1145/3186282.

[29] F. Chen, N. Chen, H. Mao and H. Hu. Assessing four neural networks on handwritten digit recognition dataset (MNIST). *CoRR*, abs/1811.08278, November 2018.

[30] A. Chernikova, A. Oprea, C. Nita-Rotaru and B. Kim. Are self-driving cars secure? evasion attacks against deep neural networks for steering angle prediction. *CoRR*, abs/1904.07370, April 2019.

[31] Soohwan Park, Hoseok Ryu, Seyoung Lee, Sunmin Lee, and Jehee Lee. Learning predict-and-simulate policies from unorganized human motion data. *ACM Trans. Graph.*, 38(6), November 2019. ISSN 0730-0301. doi: 10.1145/3355089.3356501.

[32] L. Huang, A. D. Joseph, B. Nelson, B.I.P. Rubinstein and J. D. Tygar. Adversarial machine learning. In *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence*, AISec '11, pages 43–58, New York, NY, USA, October 2011. ACM. ISBN 978-1-4503-1003-1. doi: 10.1145/2046684.2046692.

[33] R. Sahay, R. Mahfuz, and A. E. Gamal. Combatting adversarial attacks through denoising and dimensionality reduction: A cascaded autoencoder approach. In *2019 53rd Annual Conference on Information Sciences and Systems (CISS)*, pages 1–6, 2019. doi: 10.1109/CISS.2019.8692918.

[34] H. Zhang, S. Starke, T. Komura, J. Saito. Mode-adaptive neural networks for quadruped motion control. *ACM Trans. Graph.*, 37(4): 145:1–145:11, July 2018. ISSN 0730-0301. doi: 10.1145/3197517.3201366.

[35] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015. ISSN 0893-6080.

[36] K. Y. Xiao, V. Tjeng, N. M. Shafiullah and A. Madry. Training for faster adversarial robustness verification via inducing relu stability. *CoRR*, abs/1809.03008, September 2018.

[37] A. Siddiqi. Adversarial security attacks and perturbations on machine learning and deep learning methods. *CoRR*, July 2019.

[38] Samed Sivaslıoğlu, Ferhat Ozgur Catak, and Ensar Gül. Incrementing adversarial robustness with autoencoding for machine learning model attacks. In *2019 27th Signal Processing and Communications Applications Conference (SIU)*, pages 1–4, 2019. doi: 10.1109/SIU.2019.8806432.

[39] L. Pinto, J. Davidson, R. Sukthankar and A. Gupta. Robust adversarial reinforcement learning. *CoRR*, abs/1703.02702, March 2017.

[40] G. Gondim-Ribeiro, P. Tabacof and E. Valle. Adversarial attacks on variational autoencoders. *CoRR*, abs/1806.04646, 2018.

[41] A. Erba, R. Taormina, S. Galelli, M. Pogliani, M. Carminati, S. Zanero, N. O. Tippenhauer. Real-time evasion attacks with physical constraints on deep learning-based anomaly detectors in industrial control systems. *CoRR*, abs/1907.07487, July 2019.

[42] A. Kurakin, I. J. Goodfellow, S. Bengio, Y. Dong, F. Liao, M. Liang, T. Pang, J. Zhu, X. Hu, C. Xie, J. Wang, Z. Zhang, Z. Ren, A. L. Yuille, S. Huang, Y. Zhao, Y. Zhao, Z. Han, J. Long, Y. Berdibekov, T. Akiba, S. Tokui and M. Abe. Adversarial attacks and defences competition. *CoRR*, abs/1804.00097, March 2018.

[43] A. Madry, A. Makelov, L. Schmidt, D. Tsipras and A. Vladu. Towards deep learning models resistant to adversarial attacks. *CoRR*, abs/1706.06083, 2017.

[44] Lev V Utkin and Kirill D Zhuk. Improvement of the deep forest classifier by a set of neural networks. *Informatica*, 44(1), 2020. doi: 10.31449/inf.v44i1.2740.

[45] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019. doi: 10.1038/s41586-019-1724-z.

[46] Ting-Chun Wang, Ming-Yu Liu, Andrew Tao, Guilin Liu, Jan Kautz, and Bryan Catanzaro. Few-shot video-to-video synthesis. *arXiv preprint arXiv:1910.12713*, 2019.

[47] Shujian Yu and José C. Príncipe. Understanding autoencoders with information theoretic concepts. *Neural Networks*, 117:104–123, 2019. ISSN 0893-6080. doi: 10.1016/j.neunet.2019.05.003.

[48] John Paul Tan Yusiong and Prospero Clara Naval. A semi-supervised approach to monocular depth estimation, depth refinement, and semantic segmentation of driving scenes using a siamese triple decoder architecture. *Informatica*, 44(4), 2020. doi: 10.31449/inf.v44i4.3018.

[49] Z. Zhang and M. R. Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. *CoRR*, abs/1805.07836, May 2018.

[50] F. Yu, C. Liu, Y. Wang, L. Zhao and X. Chen. Interpreting adversarial robustness: A view from decision surface in input space. *CoRR*, abs/1810.00144, September 2018.

[51] L. Yang, Z. Shi, Y. Zheng and K. Zhou. Dynamic hair modeling from monocular videos using deep neural networks. *ACM Trans. Graph.*, 38(6): 235:1–235:12, November 2019. ISSN 0730-0301. doi: 10.1145/3355089.3356511.