

Sensor Based Cyber Attack Detections in Critical Infrastructures using Deep Learning Algorithms

A thesis submitted to the
Graduate School of Natural and Applied Sciences

by

Murat YILMAZ

in partial fulfillment for the
degree of Master of Science

in

Cybersecurity Engineering



This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science in Master of Cybersecurity Engineering.

APPROVED BY:

Prof. Ensar GÜL
(Thesis Advisor)



Dr. Ferhat Özgür ÇATAK
(Thesis Co-advisor)



Prof. İbrahim SOĞUKPINAR



Asst. Prof. Mehmet BAYSAN



This is to confirm that this thesis complies with all the standards set by the Graduate School of Natural and Applied Sciences of İstanbul Şehir University:

DATE OF APPROVAL:

SEAL/SIGNATURE:



Declaration of Authorship

I, Murat YILMAZ, declare that this thesis titled, 'Sensor Based Cyber Attack Detections in Critical Infrastructures using Deep Learning Algorithms' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Murat Yilmaz

Date: _____

10.09.2018

Sensor Based Cyber Attack Detections in Critical Infrastructures using Deep Learning Algorithms

Murat YILMAZ

Abstract

The technology that has evolved with innovations in the digital world has also caused an increase in many security problems. Day by day the methods and forms of the cyber attacks began to become complicated, and therefore their detection became more difficult.

In this work we will use datasets prepared in collaboration with Raymond Borges and Oak Ridge National Laboratories. These datasets include measurements of the Industrial Control Systems related to chewing attack behavior. These measurements include synchronized measurements and data records from Snort and relays with simulated control panel.

In our work, we developed two models using this dataset. The first is the model we call the Deep Neural Network (DNN) Model and build using the latest Deep Learning algorithms. Second is the model which we created by adding the AutoEncoder (AE) structure to the DNN Model. All of the variables used when developing our models were set parametrically. A number of variables such as Activation Method, number of hidden layers in the model, number of nodes in the layers, number of iterations were analyzed to create the optimum model design.

When we run our model with optimum settings, we obtained better results than related publications. The learning speed of the model we have obtained 100% accuracy rate is also quite satisfactory. While the training speed of the dataset containing about 4 thousand different operations lasts about 90 seconds, the model which completes the learning process is at the level of milliseconds to detect new attacks. This increases the applicability of the study. Detailed information about the results of the model is interpreted in Chapter 5 and the proposals for the development of the model are discussed in Chapter 6.

In our work, we intend to minimize the cost of recognizing and learning new attacks by using deep learning methods to more effectively protect industrial systems such as critical infrastructures.

Keywords: Engineering, Critical infrastructures, Industrial Systems, Information Security, Cyber Security, Cyber Attack Detections

Kritik Altyapılarda Sensör Tabanlı Veri Kontrolü ile Derin Öğrenme Algoritmaları Kullanılarak Siber Saldırı Tespiti

Murat YILMAZ

ÖZ

Dijital dünyadaki yeniliklerle beraber gelişen teknoloji, bir çok güvenlik probleminde de artışa neden oldu. Gün geçtikçe siber saldırıların yöntem ve şekilleri karmaşık bir hal almaya ve dolayısıyla bunların tespiti de daha zor olmaya başladı.

Bu çalışmada Raymond Borges ve Oak Ridge Ulusal Laboratuvarları'nın işbirliği ile hazırlanan veri setlerini kullandık. Bu veri setleri Endüstriyel Kontrol Sistemlerinin siber saldırı davranışlarıyla ilgili ölçümleri içermektedir. Bu ölçümler, simüle kontrol paneli Snort ve rölelerden gelen senkronize veri kayıtlarını içermektedir.

Çalışmamızda bu veri setini kullanarak iki model geliştirdik. Birincisi, DNN Model dediğimiz ve en güncel Deep Learning algoritmaları kullanarak oluşturduğumuz modeldir. İkincisi, bu modele AutoEncoder yapısı ekleyerek oluşturduğumuz modeldir. Modellerimizi geliştirirken kullanılan tüm değişkenler parametrik olarak ayarlandı. Aktivasyon Methodu, modelin gizli katman sayısı, katmanlarda bulunan düğüm sayıları, iterasyon sayısı gibi bir çok değişken optimum model tasarımı oluşturmak için analiz edildi.

Modelimizi optimum ayarlarla çalıştırdığımızda referans çalışmalardan çok daha iyi sonuçlar elde ettik. 100% doğruluk oranı ile çalıştırmayı başardığımız modelimizin çalışma hızı da öğrenme süreciyle birlikte oldukça tatminkardır. Yaklaşık 4 bin farklı işlem içeren veri setinin eğitim hızı yaklaşık 90 sn sürerken, öğrenme sürecini tamamlayan modelin yeni gelen saldırıları tespit süresi milisaniyeler seviyesindedir. Bu da çalışmanın uygulanabilirliğini artırmaktadır. Modelin sonuçları ile ilgili detaylı bilgi 5.bölümde verilmiş ve modelin geliştirilmesi amaçlı öneriler 6.bölümde tartışılmıştır.

Yaptığımız tez çalışması ile her yeni saldırının fark edilip öğrenilmesi maliyetinin makine ve derin öğrenme yöntemleri ile sınıflandırma metotları kullanılarak minimuma indirilmesini ve kritik altyapılar gibi endüstriyel sistemlerin, siber saldırılardan daha etkin korunabilmesini amaçlamaktayız.

Anahtar Sözcükler: Mühendislik, Kritik altyapılar, Endüstriyel Sistemler, Bilgi Güvenliği, Siber Güvenlik, Siber saldırı tespiti

Acknowledgments

I am grateful to my thesis co-advisor Dr.Ferhat Özgür ÇATAK who spent a great deal of time for me during this work and always supported from the beginning to the end of this work. I also grateful to my thesis advisor Prof.Dr.Ensar GÜL who guided my study with his valuable ideas.

I would like to express my gratitude to my valuable wife Sena YILMAZ who supported me during all of this works, I would also like to express my gratitude to my valuable children Mehmet Kerem YILMAZ and Elif Bilge YILMAZ who have sacrificed their playing time and waiting for me.

Finally, I would like to thank my valuable managers Tahir ALTINDIŞ, Ertuğrul DURAN, Teoman DİKMEN and all my teammates who support me in this work.

Contents

Declaration of Authorship	ii
Abstract	iii
Öz	iv
Acknowledgments	v
List of Figures	viii
List of Tables	xi
Abbreviations	xiii
1 Introduction	1
1.1 Contribution	4
2 Related Work	5
2.1 Developing a Hybrid Intrusion Detection System Using Data Mining for Power Systems	5
2.2 Classification of Disturbances and Cyber-attacks in Power Systems Using Heterogeneous Time-synchronized Data	6
2.3 A Specification-based Intrusion Detection Framework for Cyber-physical Environment in Electric Power System	6
2.4 Machine Learning for Power System Disturbance and Cyber-attack Discrimination	7
3 Preliminaries	8
3.1 Datasets	8
3.2 Autoencoder	12
3.3 Deep Learning	13
4 Methodology	14
4.1 System Model without Autoencoder	14
4.2 System Model with Autoencoder	16
5 Experiments	19
5.1 Experimental Setup	19
5.2 Experiments Results	20
5.2.1 DNN Model Results	20

5.2.1.1	Binary Data Class Results	22
5.2.1.2	Triple Data Class Results	30
5.2.1.3	Multi Data Class Results	36
5.2.2	AutoEncoder Results	42
5.2.2.1	Binary Data Class Results with AutoEncoder Model . . .	44
5.2.2.2	Triple Data Class Results with AutoEncoder Model . . .	53
5.2.2.3	Multi Data Class Results with AutoEncoder Model . . .	60
5.2.3	Classification Models Results	66
5.2.3.1	Binary Data Class Results with Classification Models . .	66
5.2.3.2	Triple Data Class Results with Classification Models . . .	67
5.2.3.3	Multi Data Class Results with Classification Models . . .	68
6	Concluision and Future Work	69
	Bibliography	70



List of Figures

1.1	Cyber Attack Detection Time Globally	3
1.2	Cyber Attack Detection Time Regional	3
1.3	Cyber Attacks Rate In Industries	3
3.1	Lab design in which the dataset is created	9
3.2	Distributions of the columns-1	11
3.3	Distributions of the columns-2	11
3.4	AutoEncoder Model	12
3.5	Deep Learning Model	13
4.1	Deep Learning Neural Network Model Without AE	15
4.2	Deep Neural Network Model With AutoEncoder	17
4.3	Math Models of the Activation Functions [29]	18
5.1	DNN Model Structure	21
5.2	Binary Class Layer Mode 1 tanh Accuracy	23
5.3	Binary Class Layer Mode 1 tanh Loss	23
5.4	Binary Class Layer Mode 1 softplus Accuracy	23
5.5	Binary Class Layer Mode 1 softplus Loss	23
5.6	Binary Class Layer Mode 1 softsign Accuracy	24
5.7	Binary Class Layer Mode 1 softsign Loss	24
5.8	Binary Class Layer Mode 1 Linear Accuracy	24
5.9	Binary Class Layer Mode 1 Linear Loss	24
5.10	Binary Class Layer Mode 2 tanh Accuracy	26
5.11	Binary Class Layer Mode 2 tanh Loss	26
5.12	Binary Class Layer Mode 2 softsign Accuracy	26
5.13	Binary Class Layer Mode 2 softsign Loss	26
5.14	Binary Class Layer Mode 3 relu Accuracy	28
5.15	Binary Class Layer Mode 3 relu Loss	28
5.16	Binary Class Layer Mode 3 softsign Accuracy	28
5.17	Binary Class Layer Mode 3 softsign Loss	28
5.18	Binary Class Layer Mode 3 linear Accuracy	28
5.19	Binary Class Layer Mode 3 linear Loss	28
5.20	Triple Class Layer Mode 1 tanh Accuracy	31
5.21	Triple Class Layer Mode 1 tanh Loss	31
5.22	Triple Class Layer Mode 1 softsign Accuracy	31
5.23	Triple Class Layer Mode 1 softsign Loss	31
5.24	Triple Class Layer Mode 2 tanh Accuracy	33

5.25	Triple Class Layer Mode 2 tanh Loss	33
5.26	Triple Class Layer Mode 2 softsign Accuracy	33
5.27	Triple Class Layer Mode 2 softsign Loss	33
5.28	Triple Class Layer Mode 3 tanh Accuracy	35
5.29	Triple Class Layer Mode 3 tanh Loss	35
5.30	Triple Class Layer Mode 3 softsign Accuracy	35
5.31	Triple Class Layer Mode 3 softsign Loss	35
5.32	Multi Class Layer Mode 1 tanh Accuracy	37
5.33	Multi Class Layer Mode 1 tanh Loss	37
5.34	Multi Class Layer Mode 1 softsign Accuracy	37
5.35	Multi Class Layer Mode 1 softsign Loss	37
5.36	Multi Class Layer Mode 2 tanh Accuracy	39
5.37	Multi Class Layer Mode 2 tanh Loss	39
5.38	Multi Class Layer Mode 2 softsign Accuracy	39
5.39	Multi Class Layer Mode 2 softsign Loss	39
5.40	Multi Class Layer Mode 3 tanh Accuracy	41
5.41	Multi Class Layer Mode 3 tanh Loss	41
5.42	Multi Class Layer Mode 3 softsign Accuracy	41
5.43	Multi Class Layer Mode 3 softsign Loss	41
5.44	Autoencoder Model Structure	43
5.45	DNN Model Structure with Layer Mode 1	43
5.46	Binary Class Layer Mode 1 tanh Accuracy with AutoEncoder	45
5.47	Binary Class Layer Mode 1 tanh Loss with AutoEncoder	45
5.48	Binary Class Layer Mode 1 softplus Accuracy with AutoEncoder	45
5.49	Binary Class Layer Mode 1 softplus Loss with AutoEncoder	45
5.50	Binary Class Layer Mode 1 softsign Accuracy with AutoEncoder	45
5.51	Binary Class Layer Mode 1 softsign Loss with AutoEncoder	45
5.52	Binary Class Layer Mode 2 tanh Accuracy with AutoEncoder	48
5.53	Binary Class Layer Mode 2 tanh Loss with AutoEncoder	48
5.54	Binary Class Layer Mode 2 softsign Accuracy with AutoEncoder	48
5.55	Binary Class Layer Mode 2 softsign Loss with AutoEncoder	48
5.56	Binary Class Layer Mode 2 linear Accuracy with AutoEncoder	48
5.57	Binary Class Layer Mode 2 linear Loss with AutoEncoder	48
5.58	Binary Class Layer Mode 3 tanh Accuracy with AutoEncoder	51
5.59	Binary Class Layer Mode 3 tanh Loss with AutoEncoder	51
5.60	Binary Class Layer Mode 3 relu Accuracy with AutoEncoder	51
5.61	Binary Class Layer Mode 3 relu Loss with AutoEncoder	51
5.62	Binary Class Layer Mode 3 softsign Accuracy with AutoEncoder	51
5.63	Binary Class Layer Mode 3 softsign Loss with AutoEncoder	51
5.64	Triple Class Layer Mode 1 tanh Accuracy with AutoEncoder	54
5.65	Triple Class Layer Mode 1 tanh Loss with AutoEncoder	54
5.66	Triple Class Layer Mode 1 softsign Accuracy with AutoEncoder	54
5.67	Triple Class Layer Mode 1 softsign Loss with AutoEncoder	54
5.68	Triple Class Layer Mode 2 relu Accuracy with AutoEncoder	56
5.69	Triple Class Layer Mode 2 relu Loss with AutoEncoder	56
5.70	Triple Class Layer Mode 2 softsign Accuracy with AutoEncoder	56
5.71	Triple Class Layer Mode 2 softsign Loss with AutoEncoder	56

5.72	Triple Class Layer Mode 3 relu Accuracy with AutoEncoder	58
5.73	Triple Class Layer Mode 3 relu Loss with AutoEncoder	58
5.74	Triple Class Layer Mode 3 softsign Accuracy with AutoEncoder	58
5.75	Triple Class Layer Mode 3 softsign Loss with AutoEncoder	58
5.76	Triple Class Layer Mode 3 linear Accuracy with AutoEncoder	58
5.77	Triple Class Layer Mode 3 linear Loss with AutoEncoder	58
5.78	Multi Class Layer Mode 1 tanh Accuracy with AutoEncoder	61
5.79	Multi Class Layer Mode 1 tanh Loss with AutoEncoder	61
5.80	Multi Class Layer Mode 1 softsign Accuracy with AutoEncoder	61
5.81	Multi Class Layer Mode 1 softsign Loss with AutoEncoder	61
5.82	Multi Class Layer Mode 2 tanh Accuracy with AutoEncoder	63
5.83	Multi Class Layer Mode 2 tanh Loss with AutoEncoder	63
5.84	Multi Class Layer Mode 2 softsign Accuracy with AutoEncoder	63
5.85	Multi Class Layer Mode 2 softsign Loss with AutoEncoder	63
5.86	Multi Class Layer Mode 3 tanh Accuracy with AutoEncoder	65
5.87	Multi Class Layer Mode 3 tanh Loss with AutoEncoder	65
5.88	Multi Class Layer Mode 3 softsign Accuracy with AutoEncoder	65
5.89	Multi Class Layer Mode 3 softsign Loss with AutoEncoder	65

List of Tables

3.1	Features in the dataset	9
3.2	Event Scenarios (Binary)	10
3.3	Event Scenarios (Triple)	10
3.4	Event Scenarios (Multi)	10
4.1	Model Variables	16
5.1	Nodes in the layers	22
5.2	Binary Data Class Results for Layer Mode 1	22
5.3	Confusion Matrix for Layer Mode 1 with Test Dataset	22
5.4	Binary Data Class Results for Layer Mode 2	25
5.5	Confusion Matrix for Layer Mode 2 with Test Dataset	25
5.6	Binary Data Class Results for Layer Mode 3	27
5.7	Confusion Matrix for Layer Mode 3 with Test Dataset	27
5.8	Triple Data Class Results for Layer Mode 1	30
5.9	Confusion Matrix for Layer Mode 1 with Test Dataset	30
5.10	Triple Data Class Results for Layer Mode 2	32
5.11	Confusion Matrix for Layer Mode 2 with Test Dataset	32
5.12	Triple Data Class Results for Layer Mode 3	34
5.13	Confusion Matrix for Layer Mode 3 with Test Dataset	34
5.14	Multi Data Class Results for Layer Mode 1	36
5.15	Confusion Matrix for Layer Mode 1 with Test Dataset	36
5.16	Multi Data Class Results for Layer Mode 2	38
5.17	Confusion Matrix for Layer Mode 2 with Test Dataset	38
5.18	Multi Data Class Results for Layer Mode 3	40
5.19	Confusion Matrix for Layer Mode 3 with Test Dataset	40
5.20	Nodes in the layers with AutoEncoder	44
5.21	Binary Data Class Results for Layer Mode 1	44
5.22	Confusion Matrix for Layer Mode 1 with Test Dataset	44
5.23	Binary Data Class Results for Layer Mode 2	47
5.24	Confusion Matrix for Layer Mode 2 with Test Dataset	47
5.25	Binary Data Class Results for Layer Mode 3	50
5.26	Confusion Matrix for Layer Mode 3 with Test Dataset	50
5.27	Triple Data Class Results for Layer Mode 1	53
5.28	Confusion Matrix for Layer Mode 1 with Test Dataset	53
5.29	Triple Data Class Results for Layer Mode 2	55
5.30	Confusion Matrix for Layer Mode 2 with Test Dataset	55
5.31	Triple Data Class Results for Layer Mode 3	57

5.32	Confusion Matrix for Layer Mode 3 with Test Dataset	57
5.33	Multi Data Class Results for Layer Mode 1	60
5.34	Confusion Matrix for Layer Mode 1 with Test Dataset	60
5.35	Multi Data Class Results for Layer Mode 2	62
5.36	Confusion Matrix for Layer Mode 2 with Test Dataset	62
5.37	Multi Data Class Results for Layer Mode 3	64
5.38	Confusion Matrix for Layer Mode 3 with Test Dataset	64
5.39	Binary Data Class Results with Classification Models	66
5.40	Confusion Matrix for Binary Data Class with Test Dataset	66
5.41	Triple Data Class Results with Classification Models	67
5.42	Confusion Matrix for Triple Data Class with Test Dataset	67
5.43	Multi Data Class Results with Classification Models	68
5.44	Confusion Matrix for Multi Data Class with Test Dataset	68



Abbreviations

DNN	Deep Neural Network
AE	AutoEncoder
EMEA	European Medicines Agency
APAC	Asia Pacific
IDS	Intrusion Detection System
GPU	Graphics Processing Unit
SGD	Stochastic Gradient Descent
SVM	Support Vector Machine
KNN	K-Nearest Neighbors
DTs	Decision Trees

Chapter 1

Introduction

Nowadays, the rapidly developing technology has taken the place of human power in many places. Especially in large industrial systems, such as critical infrastructures that can not be managed by human power, security problems could be occur to their computer systems. Their computer systems have been installed many years ago and then they are vulnerable for almost every current attacks.

While the definition of critical infrastructure systems varies little from country to country, it is generally defined as systems or entities that are necessary for the maintenance of vital social processes, security and economic security [1].

Critical infrastructures can generally be classified as:

- Agriculture and food
- Water
- Public health and safety
- Emergency services
- Government
- Defense Industry Base
- Information and telecommunication
- Energy
- Transportation
- Banking and finance

- Industry and manufacturing
- Mail and Shipping

Each of these sectors are critical infrastructures, and the interruption of transactions or the damage to these infrastructures may be a vital element in people's life standards that can have a life-saving impact that can even threaten human life. However, since they have large infrastructural investments, they can cause serious economic losses and weakness of states.

Critical infrastructures are not isolated systems. Any one of these infrastructures that are interacting with each other will cause damage to the chain [2]. For all these reasons, protection of these systems is vital [3].

The STUXNET attack, one of the closest examples of attacks on critical infrastructures, has shown this effect. This attack, which caused Iran to take its nuclear development activity back two years, caused only economic losses. However, in a worse scenario, it also revealed the possibility that the nuclear plant would be damaged and that it could lead to hundreds of years of disaster in that area [4, 5].

Day by day the methods and forms of the cyber attacks began to become complicated, and therefore their detection became more difficult [6].

According to a report by FireEye dated 2018, Dwell time of a cyber attack’s detection is 101 days globally, 175 days for the European Medicines Agency(EMEA) region and 498 days for the Asia Pacific (APAC) region [7, 8]. Figure 1.1 shows the Cyber Attack Dwell Time, and Figure 1.2 shows the Cyber Attack Dwell Time information of the EMEA, APAC and Americas regions.



FIGURE 1.1: Cyber Attack Detection Time Globally

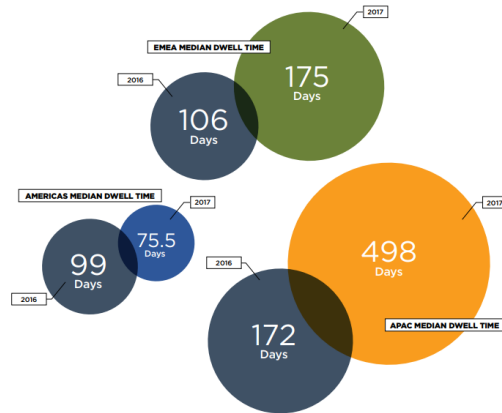


FIGURE 1.2: Cyber Attack Detection Time Regional

In 2017, the rate of cyber attack on industrial systems is given in detail in the Figure 1.3

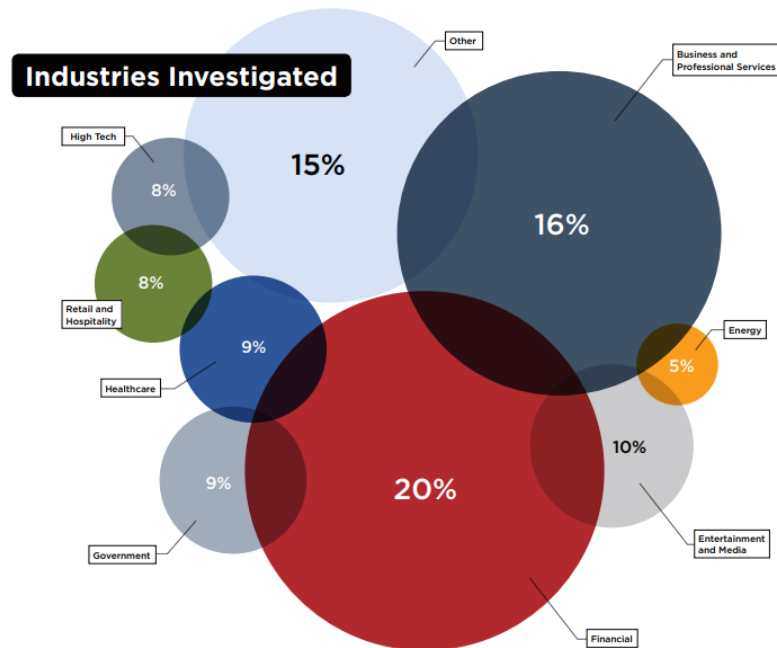


FIGURE 1.3: Cyber Attacks Rate In Industries

The purpose of this study is to reduce and automate the perception of increasing Dwell Time of cyber threats by using Deep Learning algorithms, as evidenced in M-Trend's report.

1.1 Contribution

- We worked with an up-to-date dataset created by the Mississippi State University and Oak Ridge National Laboratory in 2014. The dataset is a reliable dataset used by many academic studies [9, 10].
- We used the latest in-depth learning algorithms.
 - AutoEncoder Model
 - Tensorflow 1.4.0
 - Keras 2.1.1
 - Sklearn 0.19.1
 - Scipy 1.0.0
 - Numpy 1.13.3
 - Pandas 0.21.0
- We obtained better results than the classification performance obtained from the related publications which we examined in detail in Chapter 2. Detailed output is given in Chapter 5.

The rest of this article is arranged as follows. The introduction about our work is given in Chapter 1. Chapter 2 examines related works. Datasets, Structures used like AutoEncoder and Deep Learning are presented in Chapter 3. Introduces the proposed Deep Neural Network Model and AutoEncoder Model in Chapter 4. The experiment and the results are discussed in Chapter 5. The conclusion and future Work are given in Chapter 6.

Chapter 2

Related Work

Various studies used this critical infrastructure dataset:

- Developing a Hybrid Intrusion Detection System Using Data Mining for Power Systems [11].
- Classification of Disturbances and Cyber-attacks in Power Systems Using Heterogeneous Time-synchronized Data [12].
- A Specification-based Intrusion Detection Framework for Cyber-physical Environment in Electric Power System [13].
- Machine Learning for Power System Disturbance and Cyber-attack Discrimination [14].

Detailed explanations of the papers listed above are given in the coming subsections.

2.1 Developing a Hybrid Intrusion Detection System Using Data Mining for Power Systems

In this article, it was made to create an IDS that uses signature-based and feature-based Intrusion Detection System (IDS) features. IDS is a system that automates cyber attack detection [15]. Many cyber attacks have similar characteristics. A signature is extracted from these properties and these signatures for a similar cyber attack simplify the work of IDS systems. In this study, besides signature based systems, it also uses feature based systems. In this work, the normal system interruptions such as maintenance, normal operation and cyber attack situations are taught to IDS, and the IDS system's capabilities

are developed for a possible cyber attack, which is aimed at detecting previously unseen cyber attacks such as zero day clearance.

In this study, accuracy rate was obtained 90.4%. An operation with an accuracy rate of 90.4% means that nearly 10 attacks can not be detected per 100 attacks. When it is considered that this number is much higher in living systems, it is expected that such automatic systems will work with near zero error.

2.2 Classification of Disturbances and Cyber-attacks in Power Systems Using Heterogeneous Time-synchronized Data

It is mentioned that this study has 3 contributions in the literature. First, they point out that cyber attacks, which show themselves as a normal system interruption, can be recognized and discerned from system outages and achieve better results than the work done with [14]. The second, concerns memory usage. They use the common path mining algorithm in their work, thus indicating that they use less memory than traditional data mining methods that use more memory. The third, it learns by separating the set of algorithm scenarios and the dataset that they use. Here a common path finding algorithm is developed to prevent over-adaptation.

The algorithm used in the study has better results than the algorithms like Random forest, JRip, and it performed behind the algorithm application like Adaboost + JRip. The accuracy rate of the algorithm in 7 class applications is 93%. This rate is not suitable when considering the number of today's attacks.

2.3 A Specification-based Intrusion Detection Framework for Cyber-physical Environment in Electric Power System

In this article, a method has been proposed to detect scabby attacks on power systems or scans on physical breaks. This method reveals a specification based intrusion detection system by monitoring the records of many devices including the existing cyber intrusion detection systems, simule control panel, snort and relays and network monitoring software, control room computers. Depending on the different control data, causal relations between the cyber attacks and the interruptions are established. In this work, the probabilistic network for generating IDS rules provides a method for mapping such data to the Bayesian network. The Bayesian network is known for its powerful heuristic for

modeling interdependencies between variables and its ability to graphically show causal relationships from data and workflow records [16, 17]. This work, based on a specific control scheme, illustrates the process of building such a Bayesian network and deriving different system scenarios. With the proposed method, the IDS tracks the transmission line, and if there is any interruption in the power grid, the operator may be informed that it is caused by system problems or by cyber attacks.

The accuracy of the method used in the study is not explained with numeric results. However, it seems that the method is more effective against physical effects. It has been stated that the development of the IDS system based on the specification specified in the proposed method may be expensive and require expertise. This is not the preferred case either.

2.4 Machine Learning for Power System Disturbance and Cyber-attack Discrimination

In this work, a network specialist has developed a policy to decide whether an interruption is due to a cyber attack or a natural event. It is difficult for a person to distinguish between cyber attacks and natural phenomena because they have the same effect. For this reason, an algorithm that can be used as a decision support tool to automate this work has been studied. In this study, it was determined that the methods of learning the machine are sufficient to establish the relationship between the measurements in the power system and the causes of interruption. The classification performance of various machine learning methods is evaluated and the accuracy level of the proposed method is given.

In the study, many algorithms are used and the performances of these algorithms are classified. When we analyzed the results of the study, the Adaboost + jRipper algorithm showed the best accuracy rate of 99.1%. In our study, it is obvious that we get better results than the existing algorithms when we think that the ratio for binary and triple class is 100% and for multi class is 99.8%.

Chapter 3

Preliminaries

3.1 Datasets

Uttam Adhikari, Shengyi Pan, and Tommy Morris in collaboration with Raymond Borges and Justin Beaver of Oak Ridge National Laboratories have created 3 datasets which include measurements related to electric transmission system normal, control-maintenance, cyber attack behaviors. Measurements in the dataset include synchrophasor measurements and data logs from Snort, a simulated control panel, and relays.

The design of the Lab environment in which the dataset used in the study is created is given in the Figure 3.1 [18].

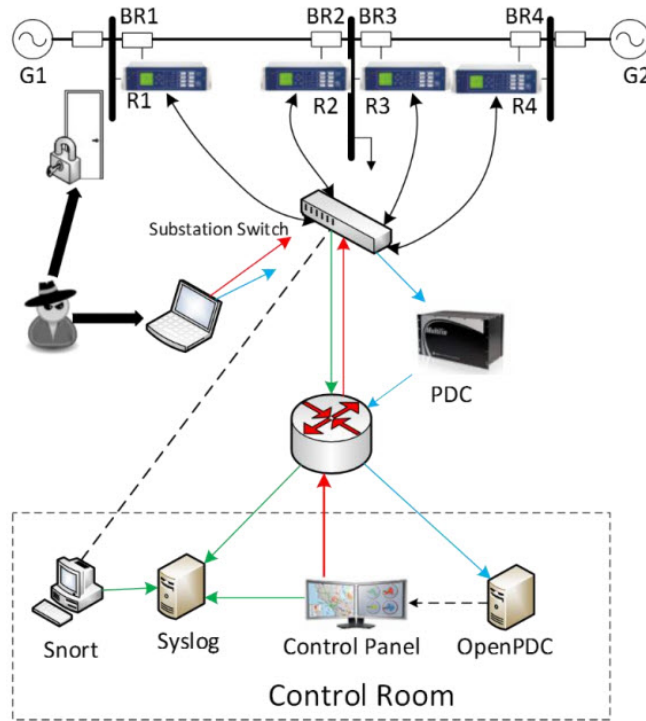


FIGURE 3.1: Lab design in which the dataset is created

The features information in the Dataset is detailed in the Table 3.1.

TABLE 3.1: Features in the dataset

Features	Description
PA1:VH – PA3:VH	Phase A - C Voltage Phase Angle
PM1: V – PM3: V	Phase A - C Voltage Phase Magnitude
PA4:IH – PA6:IH	Phase A - C Current Phase Angle
PM4: I – PM6: I	Phase A - C Current Phase Magnitude
PA7:VH – PA9:VH	Pos. – Neg. – Zero Voltage Phase Angle
PM7: V – PM9: V	Pos. – Neg. – Zero Voltage Phase Magnitude
PA10:VH - PA12:VH	Pos. – Neg. – Zero Current Phase Angle
PM10: V - PM12: V	Pos. – Neg. – Zero Current Phase Magnitude
F	Frequency for relays
DF	Frequency Delta (dF/dt) for relays
PA:Z	Appearance Impedance for relays
PA:ZH	Appearance Impedance Angle for relays
S	Status Flag for relays

The datasets used in the study are classified according to their output labels. The label distribution of the binary class dataset is given in the Table 3.2.

The label distribution of the triple class dataset is given in the Table 3.3.

The label distribution of the multi class dataset is given in the Table 3.4.

TABLE 3.2: Event Scenarios (Binary)

Scenario	Description	Number of Rows
0	Normal operation	1100
1	Attack	3866

TABLE 3.3: Event Scenarios (Triple)

Scenario	Description	Number of Rows
-1	Natural Events	927
0	Normal operation	173
1	Attack	3866

TABLE 3.4: Event Scenarios (Multi)

Scenario	Description	Number of Rows
-2	Fault from Line (Natural Events)	264
-1	Line maintenance (Natural Events)	663
0	Regular Operation (Normal operation)	173
1	Data Injection - SLG fault replay (Attack)	569
2	Command injection against single relay to R1,R2,R3,R4 (Attack)	346
3	Command injection against single relay to R1 and R2 or R3 and R4 (Attack)	106
4	Disabling relay function - single relay disabled & fault (Attack)	1675
5	Disabling relay function - two relays disabled & fault (Attack)	898
6	Disabling relay function - two relay disabled & line maintenance (Attack)	272

The row-based distributions of the columns in the dataset are given below. Some fields in the dataset have balanced distribution, as shown in Figure 3.2. However, as you can see from the Figure 3.3, there are also stacked areas on one side.

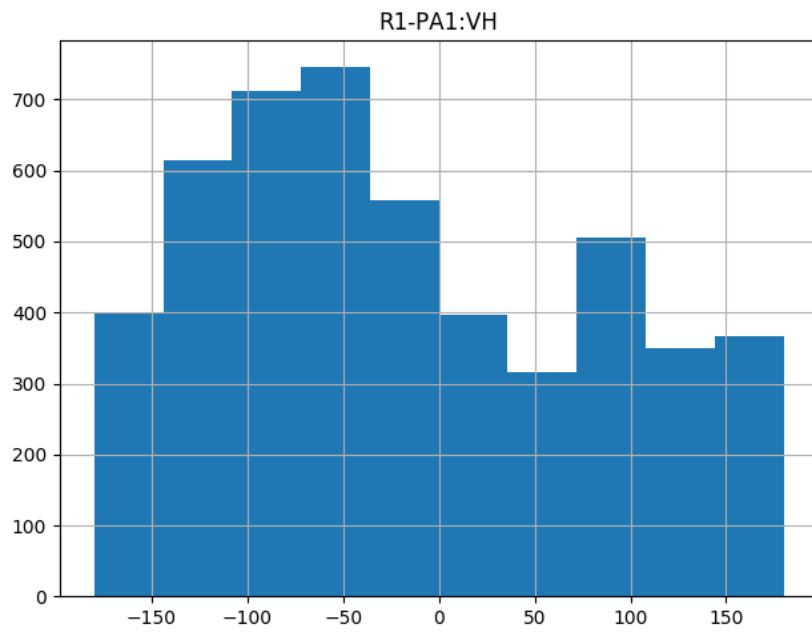


FIGURE 3.2: Distributions of the columns-1

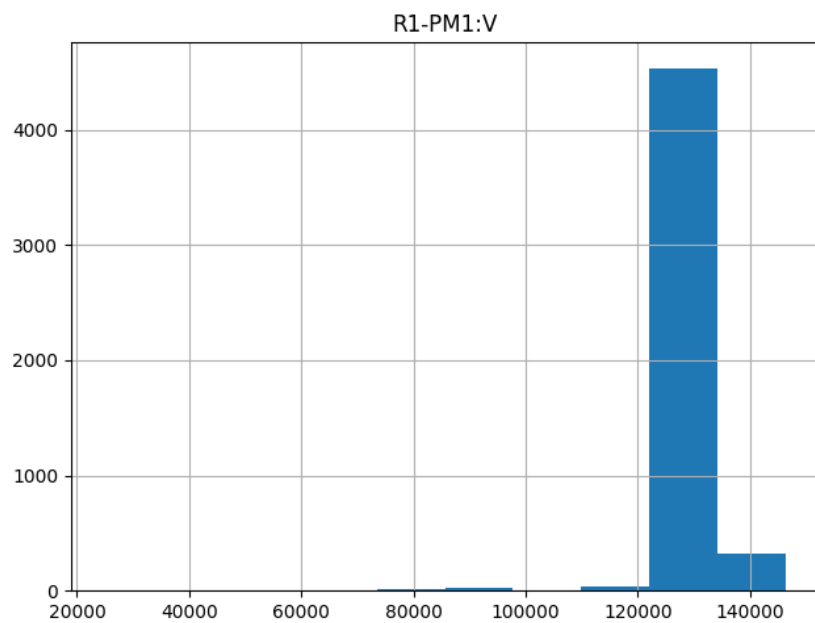


FIGURE 3.3: Distributions of the columns-2

3.2 Autoencoder

AutoEncoder is a type of Neural Network that compresses multidimensional data first into the hidden area and then reconstructs the data from the compressed hidden area. AutoEncoders have three type layers. In the Figure 3.4, the green nodes represent the input layer, the blue nodes represent the hidden area or hidden layer, and the red nodes represent the output layer. The number of nodes in the input layer is equal to the number of nodes in the output layer, because AutoEncoder is to reconstruct the intended data. It is called the input layer and the hidden area encoder. The Encoder allows you to reduce multidimensional data to a smaller size. The decoder is called the decoder between the hidden area and the output layer. The decoder tries to reconfigure the entry by increasing the size of the compressed hidden area [19, 20].

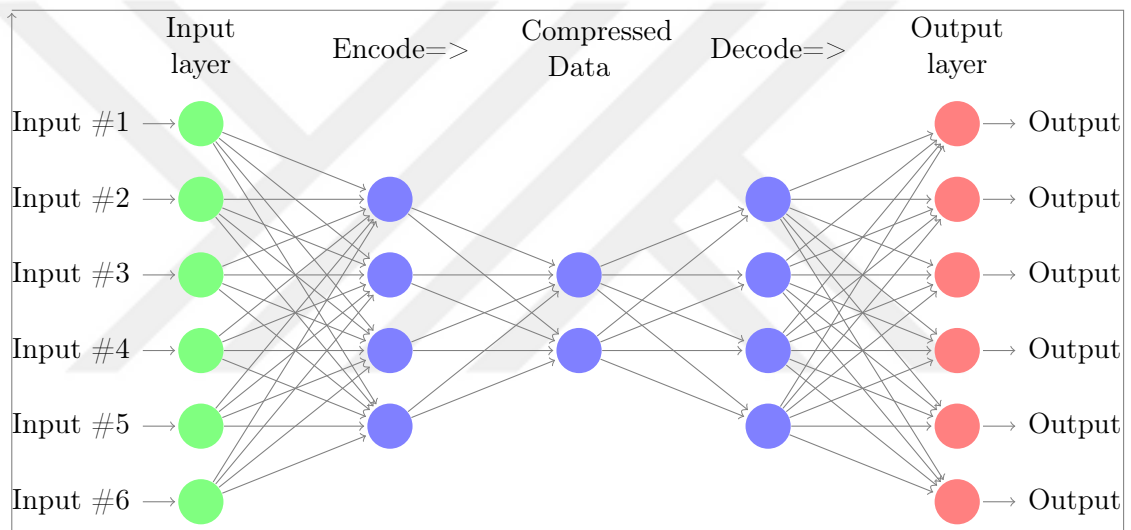


FIGURE 3.4: AutoEncoder Model

The dataset we use includes voltage measurements and consists of fractional numeric values that will not affect the end result.

In our work, we used the AutoEncoder Model to avoid slowing down with these fractional numeric values and avoiding false positives in the learning process.

The purpose of the AutoEncoder Model is to increase the accuracy of the system by deleting unnecessary detail. For example, the detail in a 3-D image is unnecessary and must be reduced to 2 dimensions only for a shape-separating operation. Another example is the most commonly used noise reduction method. Noise-canceling images can be obtained when used with AutoEncoder model [21, 22].

3.3 Deep Learning

A class of machine learning techniques, where many layers of information processing stages in hierarchical architectures are exploited for unsupervised feature learning and for pattern analysis/classification. The essence of deep learning is to compute hierarchical features or representations of the observational data, where the higher-level features or factors are defined from lower-level ones [23].

In the Figure 3.5, the green nodes represent the input layer, the blue nodes represent the hidden area or hidden layer, and the red nodes represent the output layer.

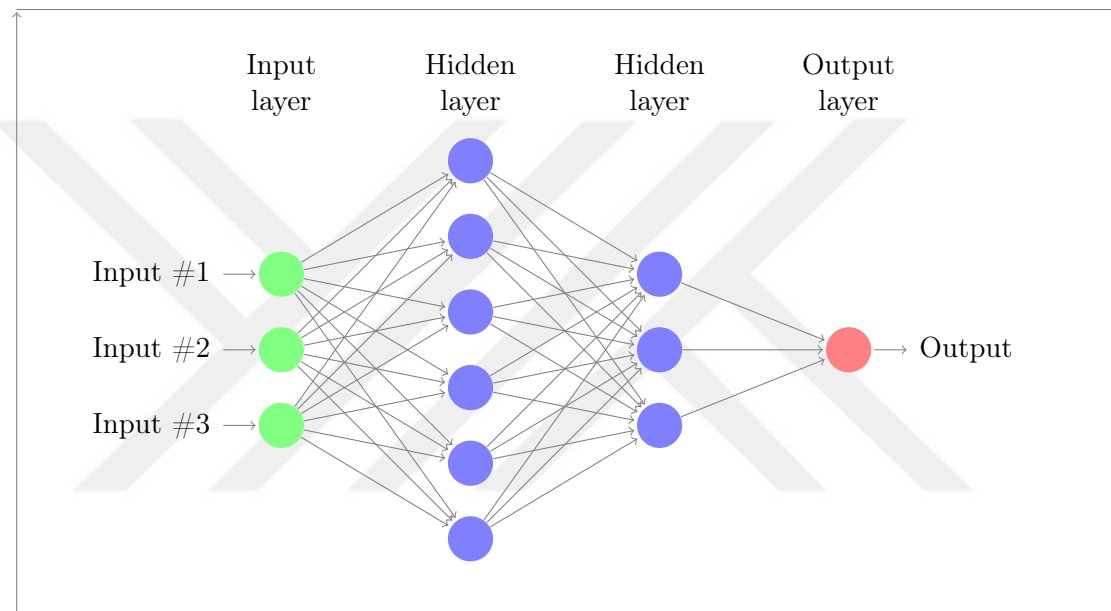


FIGURE 3.5: Deep Learning Model

The data set we use contains voltage information. It is not possible for a person to interpret this data coming from 128 sensors and create an attack pattern. But Deep Neural Network approaches can easily do this, which is impossible for people. For this reason, in our study, we used Deep Neural Network Algorithms, which have a very high ability to analyze nonlinear data.

Chapter 4

Methodology

We will examine the methods we use in this section. Our work mainly consists of two main topics. The first model we created without using the AutoEncoder structure, the second the model we created using the AutoEncoder structure. We will examine the details of the structure of these two models, such as activation methods, Epoch numbers, node and layer numbers, and discuss their strengths and weaknesses.

Accuracy, Precision, Recall, F-Score and Confusion Matrix information were obtained as a result of the study of these two models [24, 25]. These results have been compared in detail in Experiments section.

4.1 System Model without Autoencoder

In the DNN model, all variables are defined as array parameters. So we did not have to constantly intervene in the variables to determine the design that would give the model the best possible result, just watching all the outputs of the work was enough. Based on the output results, we revised the algorithm and ran the model again. This gave us a lot of speed during implementation.

An example of the structure of the model is given in the Figure 4.1. According to the number of Epoch, the number of layers and nodes in these layers are increased.

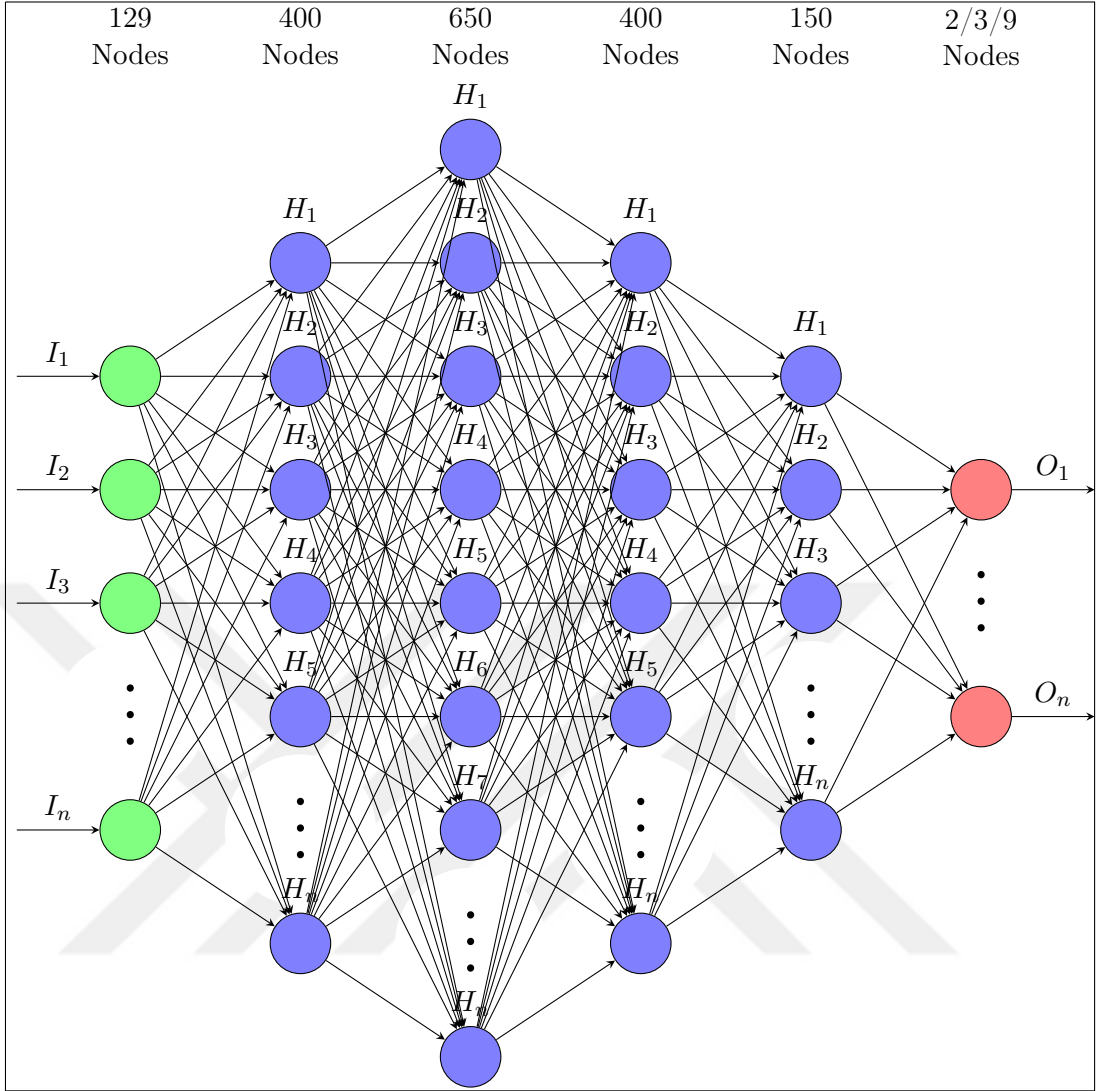


FIGURE 4.1: Deep Learning Neural Network Model Without AE

The mathematical expression of the example model in the Figure 4.1 is given in formula 4.1 [26];

$I_n = 129$ inputs;

$H_n = 400, 650, 400, 150$ hidden layers;

$O_n = 2$ outputs;

$$O_n = (O_1, O_2) = \psi_4(\psi_3(\psi_2(\psi_1(I_n \mathbf{w}^{(1)}) \mathbf{w}^{(2)}) \mathbf{w}^{(3)}) \mathbf{w}^{(4)}) \quad (4.1)$$

$$\mathbf{w}^{(1)} \in \mathbb{R}^{129}$$

$$\mathbf{w}^{(2)} \in \mathbb{R}^{400}$$

$$\mathbf{w}^{(3)} \in \mathbb{R}^{650}$$

$$\mathbf{w}^{(4)} \in \mathbb{R}^{400}$$

These are the matrix of weights from the hidden to the output layer and $\psi_1, \psi_2, \psi_3, \psi_4$ are activation functions.

Fifteen samples of three class datasets, Binary, Triple and Multi, were tested individually with the above model and Epoc numbers for 7 activation methods. Accuracy and Loss graphs of each were taken together with Presicion, Recall information and the status of the working status of the model.

We dynamically designed our model to give the optimum result [27]. The dataset used, the activation methods, the number of iterations and the number of nodes in the Neural Network are all defined as variables. These variables were run with Cartesian mapping and the results were followed to determine the most accurate sequence. Further work was done by developing this model. The variables used when developing the model are given in the Table 4.1.

TABLE 4.1: Model Variables

Data Classes	Datasets	Activation Methods	Epocs	Layers and Nodes
binary	data1.csv	tanh	200	Mode1 : 129-400-150-2/3/9
triple	data2.csv	relu	300	Mode2 : 129-400-650-400-150-2/3/9
multi	data3.csv	sigmoid	500	Mode3 : 129-400-650-900-650-400-150-2/3/9
	.	softplus		
	.	softsign		
	.	softmax		
	data15.csv	linear		

4.2 System Model with Autoencoder

The reason we intended to use the AutoEncoder Model with our Deep Learning Model was that the data from the Sensors was Float. As the information in the numbers does not affect the measurement result very much, we have tried to progress faster and more accurately with the weighted information by squeezing unnecessary detail information.

AutoEncoder part we added to my model is given in the Figure 4.2. Here, as in the main model, we have created a dynamic model using the array parameters to obtain the most accurate result, and by analyzing the results, we made the necessary revisions in the algorithm and brought it closer to the optimum result.

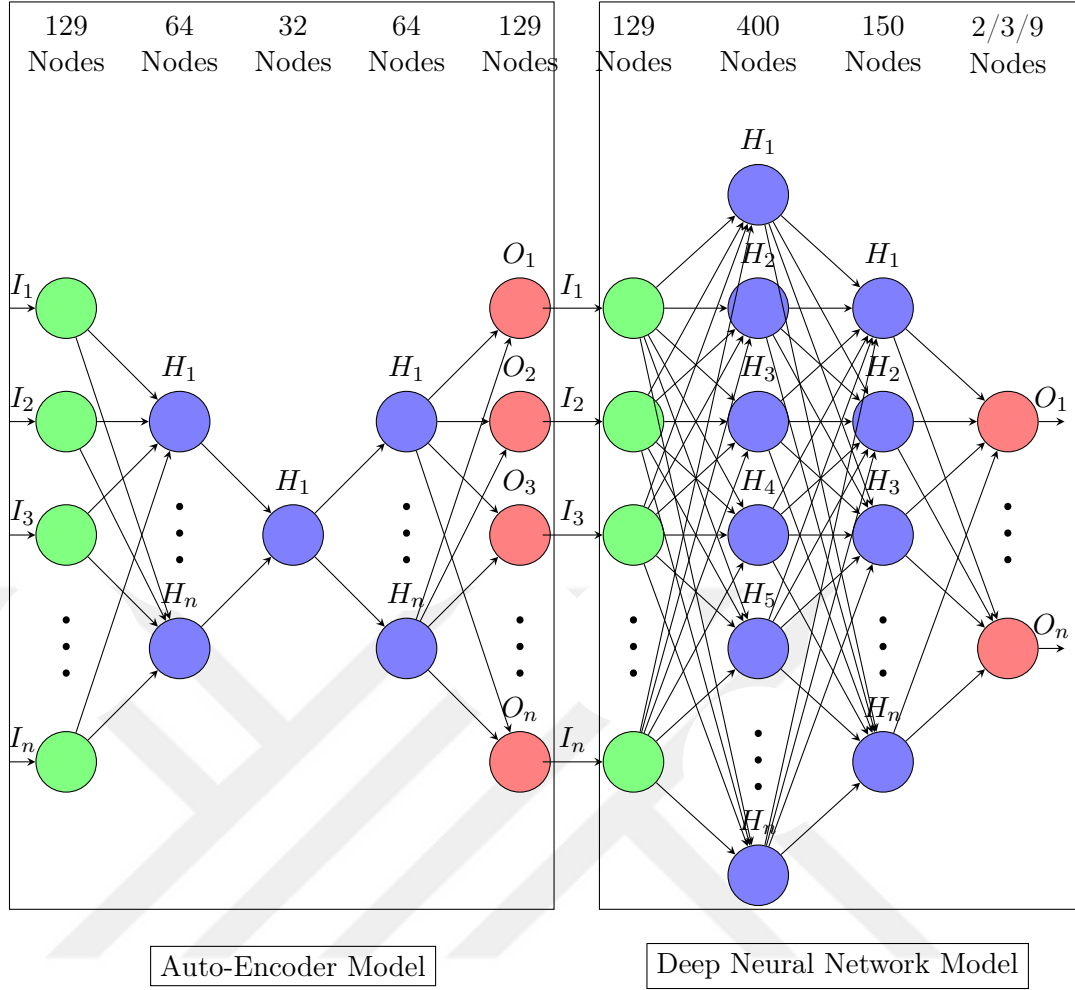


FIGURE 4.2: Deep Neural Network Model With AutoEncoder

An autoencoder always consists of two parts, the encoder and the decoder, which can be defined as transitions ϕ and ψ , Calculation of ϕ and ψ is given in formula 4.2.[28]:

$$\phi : X \rightarrow F$$

$$\psi : F \rightarrow X$$

$$\phi, \psi = \operatorname{argmin}_{\phi, \psi} \|X - (\psi \circ \phi)X\|^2 \quad (4.2)$$

In the simplest case, where there is one hidden layer, the encoder stage of an autoencoder takes the input $\mathbf{x} \in \mathbb{R}^d$ and maps it to $\mathbf{z} \in \mathbb{R}^p$. \mathbf{z} is calculated as in formula 4.3:

$$\mathbf{z} = \sigma(W\mathbf{x} + \mathbf{b}) \quad (4.3)$$

z is usually referred to as code, latent variables, or latent representation. Here, σ is an element-wise activation function such as a sigmoid function or a rectified linear unit. W

is a weight matrix and \mathbf{b} is a bias vector. After that, as given in formula 4.4 below, the decoder stage of the autoencoder maps \mathbf{z} to the reconstruction x' of the same shape as \mathbf{x} :

$$x' = \sigma(W\mathbf{x} + \mathbf{b}) \quad (4.4)$$

The mathematical models of 7 activation methods used in DNN Model and DNN Model with Autoencoder models are given in Figure 4.3.


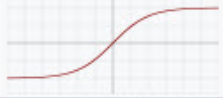
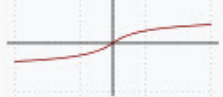



Name	Plot	Equation
Identity		$f(x) = x$
TanH		$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
Softsign		$f(x) = \frac{x}{1 + x }$
Rectified linear unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \ln(1 + e^x)$
Logistic (a.k.a. Sigmoid or Soft step)		$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$
Softmax		$f_i(\vec{x}) = \frac{e^{x_i}}{\sum_{j=1}^J e^{x_j}} \quad \text{for } i = 1, \dots, J$

FIGURE 4.3: Math Models of the Activation Functions [29]

Chapter 5

Experiments

5.1 Experimental Setup

For the deep learning, the latest version of the Tensorflow library, developed by Google and it was open source, was used, version 1.4.0. This library is highly preferred for Machine Learning and especially because it allows for quick and easy implementation in conjunction with the Python programming language for deep learning [30, 31]. It also allows Graphics Processing Unit (GPU) programming in applications where high processing power is required [32].

In the study, Keras 2.1.1 version, which is preferred in deep learning applications and able to use Theano and Tensorflow as a backend, was used [33].

The library sklearn 0.19.1 (Scikit-learn), which allows you to get the output of the application in a practical way, was used. This library is very useful when doing data analysis. There is a lot of ability to compare and classify the results of artificial intelligence training [34, 35].

The most up to date version of the Numpy library, 1.13.3, was used for processing sequences, vectors, and matrices [36].

The frequently used mathematical routines and the Scipy 1.0.0 library were used to allow easy and rapid use of physical problems in a computer environment [37].

For the distribution histogram of the fields in the dataset to be used before the model was created and at the end of the model, 0.21.0 version of the Pandas library was used for graphical display of functions such as analysis of outputs, accuracy and precision [38, 39].

5.2 Experiments Results

Two separate models were designed in the study. The first is the DNN Model, which is a model in which the activation methods, the hidden layer and the number of nodes are changed dynamically.

The second is the AutoEncoder Model. In this model, input values of the DNN Model are not directly read from the file. The inputs were first passed through the AutoEncoder Model and simplified, and these outputs were given as DNN Model inputs.

The results of these two studies are separately analyzed and presented in detail below.

5.2.1 DNN Model Results

The structure of the DNN Model is given in the Figure 5.1. Dropout rate in the DNN Model is 20%

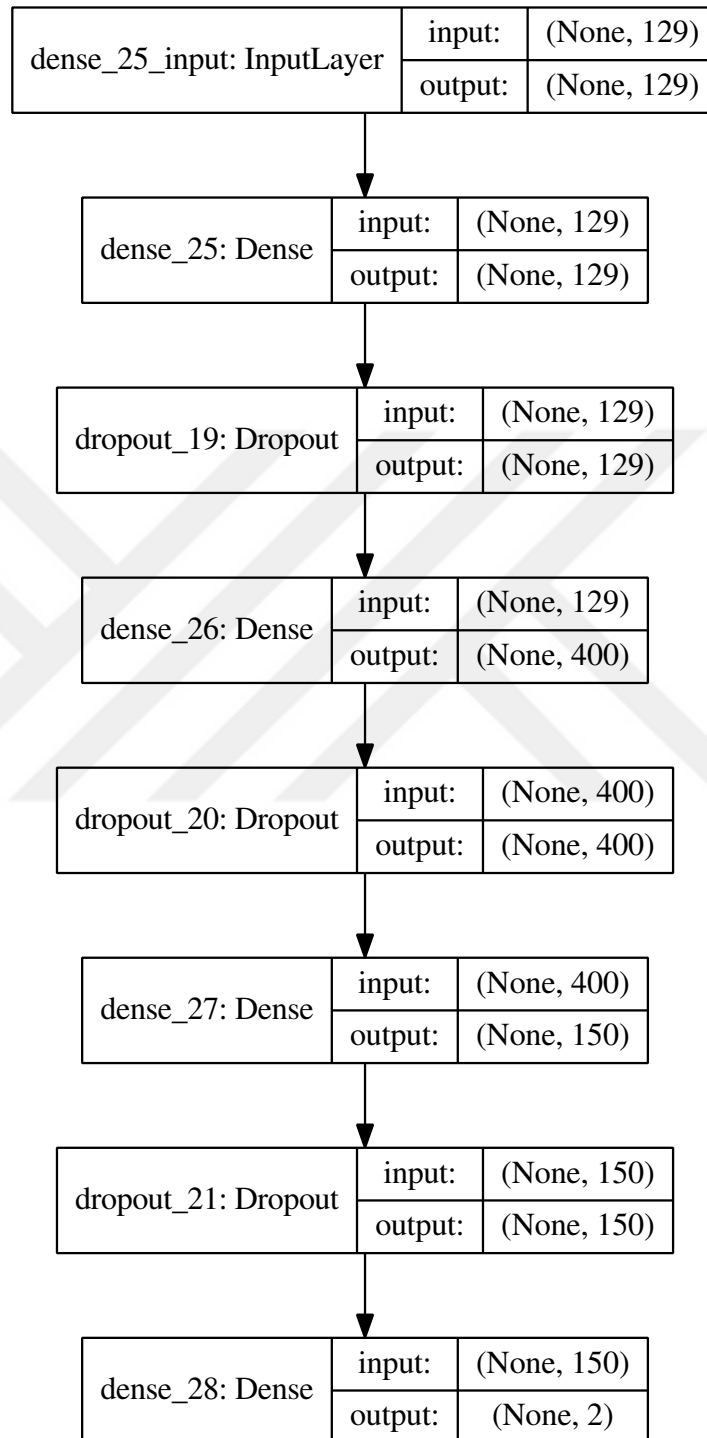


FIGURE 5.1: DNN Model Structure

In the model there are 3 designs called Layer Mode 1, Layer Mode 2, Layer Mode 3.

The number of layers and nodes in these designs are given in the Table 5.1.

TABLE 5.1: Nodes in the layers

Layer Mode	DNN Model
1	129(input)-400-150-2/3/9(Output)
2	129(input)-400-650-400-150-2/3/9(Output)
3	129(input)-400-650-900-650-400-150-2/3/9(Output)

There are too many parameters in the operation. To provide a more meaningful representation of the outputs of the study, the results of data classes called binary, triple, multi are presented separately below.

5.2.1.1 Binary Data Class Results

The model is run separately for 15 datasets in binary data class with binary tags with 0-Normal, 1-Attack.

The Layer Mode 1 design results of the Binary Data Class is given in detail in the Table 5.2

TABLE 5.2: Binary Data Class Results for Layer Mode 1

	tanh	relu	sigmoid	softplus	softsign	softmax	linear
Epochs	200	200	200	200	200	200	200
Accuracy	1.0	0.9990	0.7031	1.0	1.0	0.6933	1.0
Presicion	1.0	0.9990	0.4944	1.0	1.0	0.4807	1.0
Recall	1.0	0.9990	0.7032	1.0	1.0	0.6933	1.0
F Score	1.0	0.9990	0.5806	1.0	1.0	0.5677	1.0
Process Time(minute)	01:55	01:53	02:21	03:36	02:16	01:52	04:05

Analysis of the data in the Layer Mode 1 design is shown that the four most successful results are obtained with the 'tanh', 'softplus', 'softsign' and 'linear' activation methods.

The Confusion Matrix of these four methods is given in the Table 5.3

TABLE 5.3: Confusion Matrix for Layer Mode 1 with Test Dataset

	P	N		P	N		P	N		P	N
P	303	0	P	304	0	P	310	0	P	317	0
N	0	711	N	0	710	N	0	704	N	0	697
	tanh			softplus			softsign			linear	

The Accuracy and Loss graphs of the model using 'tanh', 'softplus', 'softsign' and 'linear' activation methods are given in the Figure 5.2 - 5.9.

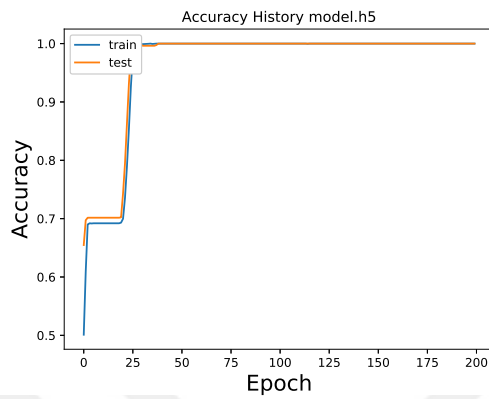


FIGURE 5.2: Binary Class Layer Mode 1 tanh Accuracy

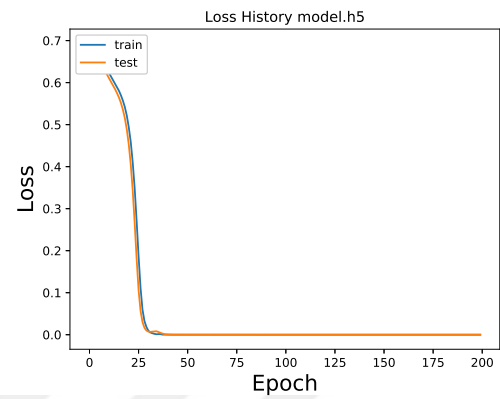


FIGURE 5.3: Binary Class Layer Mode 1 tanh Loss

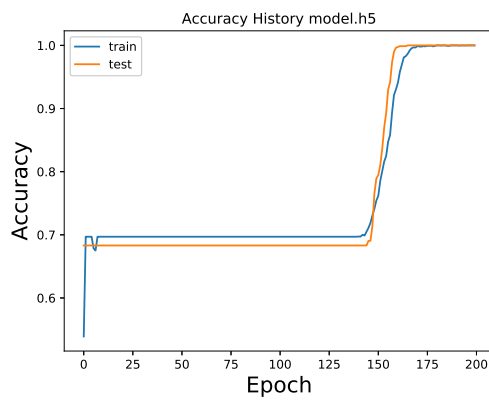


FIGURE 5.4: Binary Class Layer Mode 1 softplus Accuracy

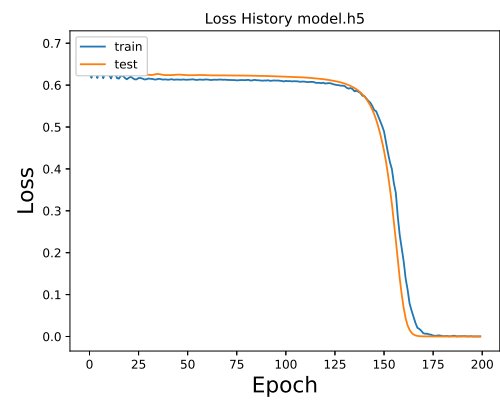


FIGURE 5.5: Binary Class Layer Mode 1 softplus Loss

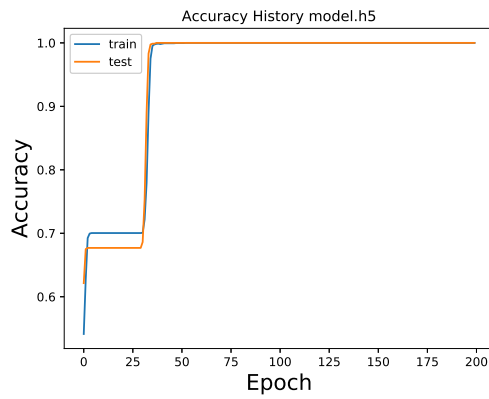


FIGURE 5.6: Binary Class Layer Mode 1 softsign Accuracy

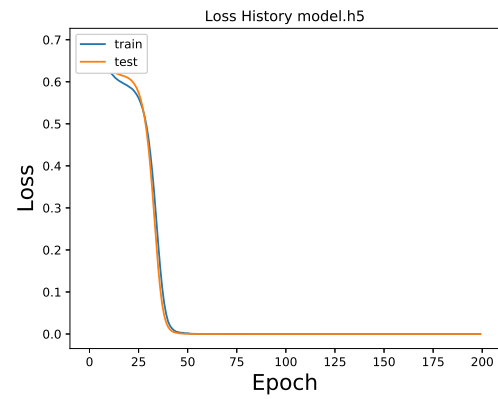


FIGURE 5.7: Binary Class Layer Mode 1 softsign Loss

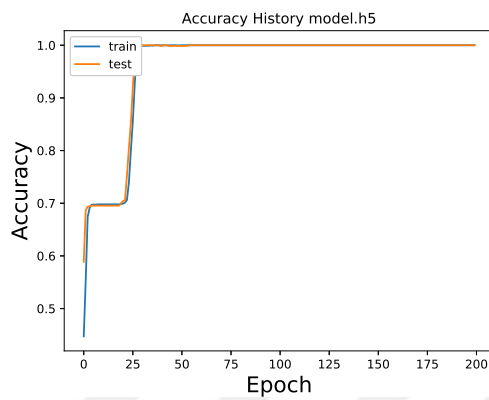


FIGURE 5.8: Binary Class Layer Mode 1 Linear Accuracy

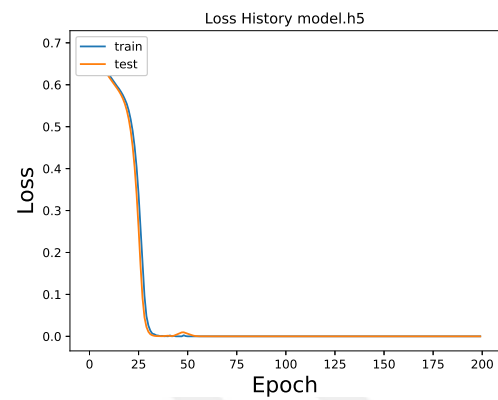


FIGURE 5.9: Binary Class Layer Mode 1 Linear Loss

When we run our model with the binary data class, we obtained 100% accuracy rate after nearly 30 Epochs. We obtained this accuracy rate with the DNN Model created with the following parameter sequence.

- DNN Optimizer Algorithm : Stochastic Gradient Descent(SGD)
- Nodes in the layers : 129(Input) - 400 - 150 - 2(Output)
- Dropout Rate : 20%
- Kernel Initializer : Uniform
- Activation Method : Tanh
- Epochs : 200
- Output Activation Method : sigmoid
- Output Loss Method : binary_crossentropy

With these parameters, the training duration of our model is 1:55 minutes for 4055 different processes. The test duration is approximately 280 milliseconds for 1014 different

processes and the detection time of the new incoming attack is 0.28 milliseconds.

The Layer Mode 2 design results of the Binary Data Class is given in detail in the Table 5.4.

TABLE 5.4: Binary Data Class Results for Layer Mode 2

	tanh	relu	sigmoid	softplus	softsign	softmax	linear
Epochs	300	300	300	300	300	300	300
Accuracy	1.0	0.9990	0.6755	0.6972	1.0	0.6864	0.9970
Presicion	1.0	0.9990	0.4564	0.4861	1.0	0.4711	0.9971
Recall	1.0	0.9990	0.6755	0.6972	1.0	0.6864	0.9970
F Score	1.0	0.9990	0.5447	0.5729	1.0	0.5587	0.9970
Process Time(minute)	18:33	09:48	10:08	10:58	11:05	12:58	09:48

Analysis of the data in the Layer Mode 2 design is shown that the two most successful results are obtained with the 'tanh' and 'softsign' activation methods.

The Confusion Matrix of these two methods is given in the Table 5.5.

TABLE 5.5: Confusion Matrix for Layer Mode 2 with Test Dataset

	P	N		P	N
P	292	0	P	304	0
N	0	722	N	0	710
tanh			softsign		

The Accuracy and Loss graphs of the model using 'tanh' and 'softsign' activation methods are given in the Figure 5.10 - 5.13.

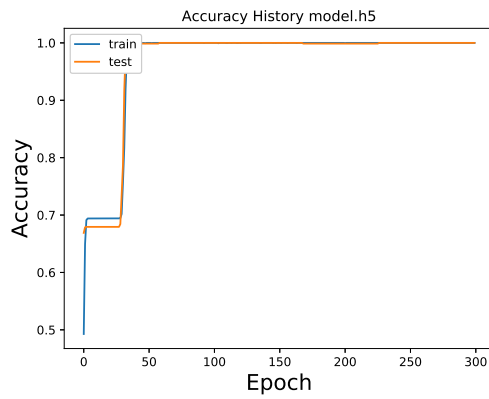


FIGURE 5.10: Binary Class Layer Mode 2 tanh Accuracy

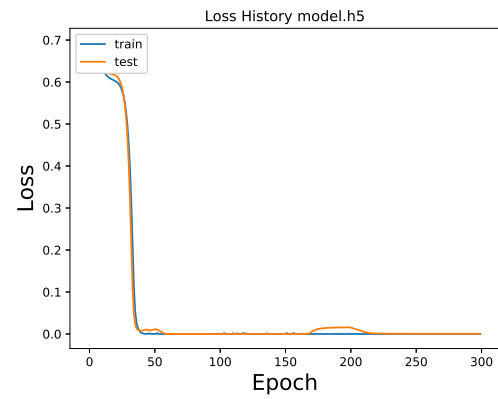


FIGURE 5.11: Binary Class Layer Mode 2 tanh Loss

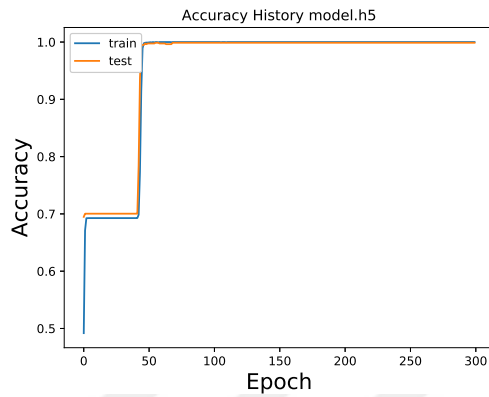


FIGURE 5.12: Binary Class Layer Mode 2 softsign Accuracy

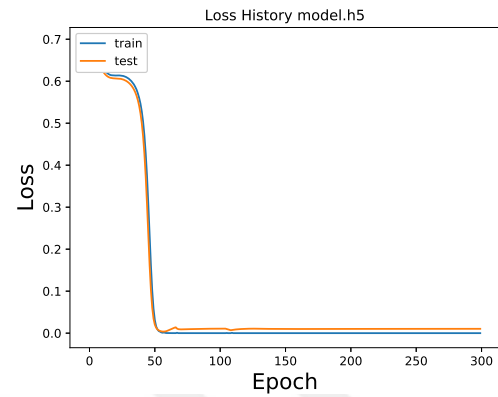


FIGURE 5.13: Binary Class Layer Mode 2 softsign Loss

When we run our model with the binary data class, we obtained 100% accuracy rate after nearly 50 Epochs. We obtained this accuracy rate with the DNN Model created with the following parameter sequence.

- DNN Optimizer Algorithm : Stochastic Gradient Descent(SGD)
- Nodes in the layers : 129(Input) - 400 - 650 - 400 - 150 - 2(Output)
- Dropout Rate : 20%
- Kernel Initializer : Uniform
- Activation Method : Softsign
- Epochs : 300
- Output Activation Method : sigmoid
- Output Loss Method : binary_crossentropy

With these parameters, the training duration of our model is 11 minutes for 4055 different processes. The test duration is approximately 514 milliseconds for 1014 different processes and the detection time of the new incoming attack is 0.5 milliseconds.

The Layer Mode 3 design results of the Binary Data Class is given in detail in the Table 5.6.

TABLE 5.6: Binary Data Class Results for Layer Mode 3

	tanh	relu	sigmoid	softplus	softsign	softmax	linear
Epochs	500	500	500	500	500	500	500
Accuracy	0.9970	0.9990	0.7032	0.7130	0.9990	0.6903	0.9990
Presicion	0.9970	0.9990	0.4944	0.5084	0.9990	0.4766	0.9990
Recall	0.9970	0.9990	0.7032	0.7130	0.9990	0.6903	0.9990
F Score	0.9970	0.9990	0.5806	0.5936	0.9990	0.5639	0.9990
Process Time(minute)	40:07	40:47	41:12	44:28	40:32	50:50	41:03

Analysis of the data in the Layer Mode 3 design is shown that the three most successful results are obtained with the 'relu', 'softsign' and 'linear' activation methods.

The Confusion Matrix of these three methods is given in the Table 5.7.

TABLE 5.7: Confusion Matrix for Layer Mode 3 with Test Dataset

	P	N		P	N		P	N
P	295	1	P	312	1	P	297	0
N	0	718	N	0	701	N	1	716
	relu			softsign			linear	

The Accuracy and Loss graphs of the model using 'relu', 'softsign' and 'linear' activation methods are given in the Figure 5.14 - 5.19.

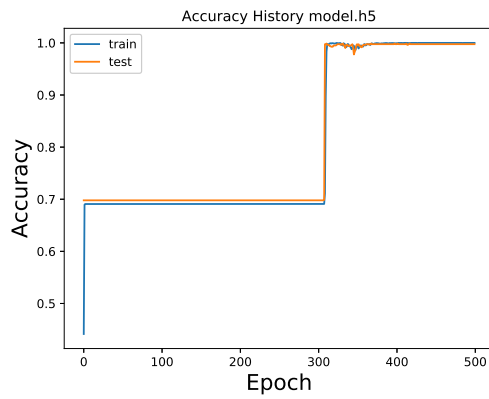


FIGURE 5.14: Binary Class Layer Mode 3 relu Accuracy

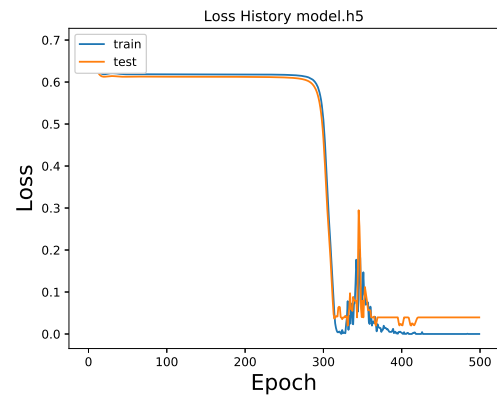


FIGURE 5.15: Binary Class Layer Mode 3 relu Loss

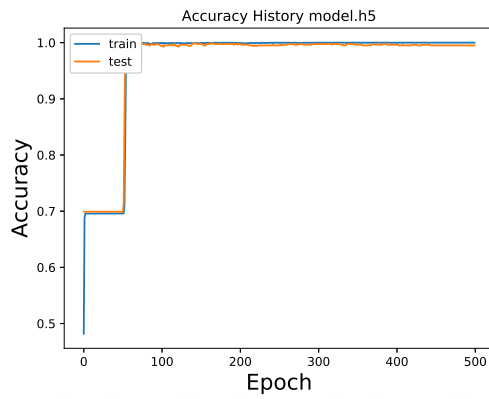


FIGURE 5.16: Binary Class Layer Mode 3 softsign Accuracy

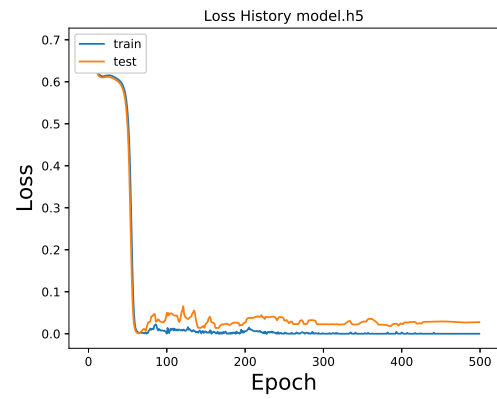


FIGURE 5.17: Binary Class Layer Mode 3 softsign Loss

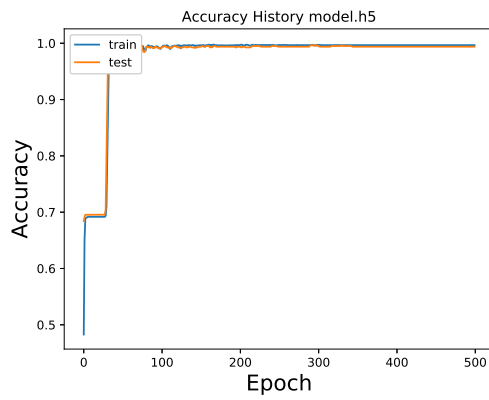


FIGURE 5.18: Binary Class Layer Mode 3 linear Accuracy

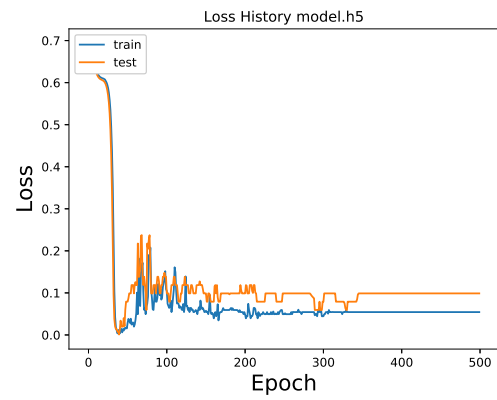


FIGURE 5.19: Binary Class Layer Mode 3 linear Loss

When we run our model with the binary data class, we obtained 99.9% accuracy rate after nearly 80 Epochs. We obtained this accuracy rate with the DNN Model created with the following parameter sequence.

- DNN Optimizer Algorithm : Stochastic Gradient Descent(SGD)
- Nodes in the layers : 129(Input)-400-650-900-650-400-150-2(Output)
- Dropout Rate : 20%
- Kernel Initializer : Uniform
- Activation Method : Softsign
- Epochs : 500
- Output Activation Method : sigmoid
- Output Loss Method : binary_crossentropy

With these parameters, the training duration of our model is 40 minutes for 4055 different processes. The test duration is approximately 1 second for 1014 different processes and the detection time of the new incoming attack is 0.98 milliseconds.

5.2.1.2 Triple Data Class Results

The model is run separately for 15 datasets in triple data class with triple tags with -1 Natural, 0-Normal, 1-Attack.

The Layer Mode 1 design results of the Triple Data Class is given in detail in the Table 5.8.

TABLE 5.8: Triple Data Class Results for Layer Mode 1

	tanh	relu	sigmoid	softplus	softsign	softmax	linear
Epochs	200	200	200	200	200	200	200
Accuracy	1.0	0.9951	0.6963	0.9990	1.0	0.7071	0.9960
Presicion	1.0	0.9953	0.4848	0.9990	1.0	0.5000	0.9961
Recall	1.0	0.9951	0.6963	9990	1.0	0.7071	0.9961
F Score	1.0	0.9951	0.5716	0.9990	1.0	0.5858	0.9960
Process Time(minute)	01:54	02:03	01:55	02:11	01:53	02:43	01:46

Analysis of the data in the Layer Mode 1 design is shown that the two most successful results are obtained with the 'tanh' and 'softsign' activation methods.

The Confusion Matrix of these two methods is given in the Table 5.9.

TABLE 5.9: Confusion Matrix for Layer Mode 1 with Test Dataset

	-1	0	1		-1	0	1
-1	221	0	0	-1	241	0	0
0	0	70	0	0	0	67	0
1	0	0	723	1	0	0	706
	tanh				softsign		

The Accuracy and Loss graphs of the model using 'tanh' and 'softsign' activation methods are given in the Figure 5.20 - 5.23.

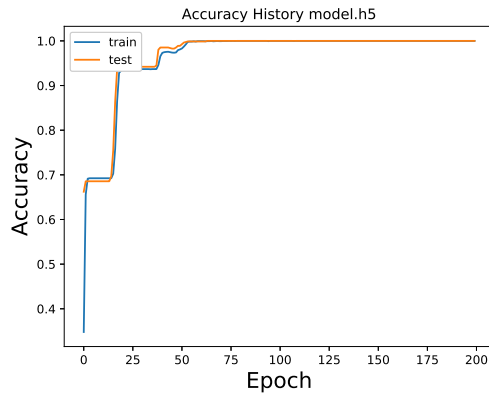


FIGURE 5.20: Triple Class Layer Mode 1 tanh Accuracy

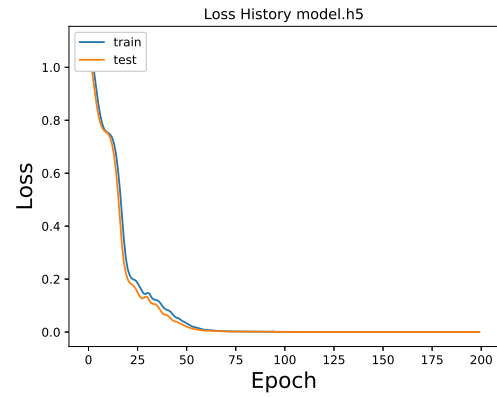


FIGURE 5.21: Triple Class Layer Mode 1 tanh Loss

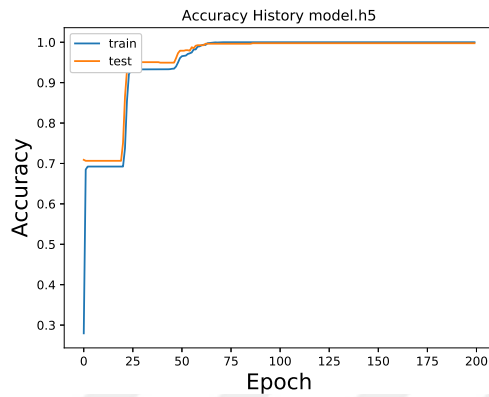


FIGURE 5.22: Triple Class Layer Mode 1 softsign Accuracy

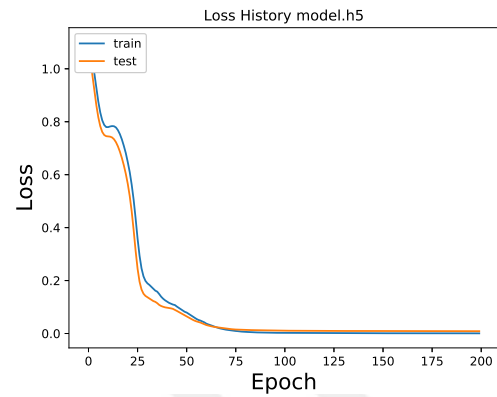


FIGURE 5.23: Triple Class Layer Mode 1 softsign Loss

When we run our model with the triple data class, we obtained 100% accuracy rate after nearly 60 Epochs. We obtained this accuracy rate with the DNN Model created with the following parameter sequence.

- DNN Optimizer Algorithm : Stochastic Gradient Descent(SGD)
- Nodes in the layers : 129(Input) - 400 - 150 - 3(Output)
- Dropout Rate : 20%
- Kernel Initializer : Uniform
- Activation Method : Softsign
- Epochs : 200
- Output Activation Method : softmax
- Output Loss Method : categorical_crossentropy

With these parameters, the training duration of our model is 1:53 minutes for 4055 different processes. The test duration is approximately 241 milliseconds for 1014 different processes and the detection time of the new incoming attack is 0.23 milliseconds.

The Layer Mode 2 design results of the Triple Data Class is given in detail in the Table 5.10.

TABLE 5.10: Triple Data Class Results for Layer Mode 2

	tanh	relu	sigmoid	softplus	softsign	softmax	linear
Epochs	300	300	300	300	300	300	300
Accuracy	0.9990	0.9970	0.7041	0.6696	0.9990	0.7258	0.9980
Presicion	0.9990	0.9972	0.4958	0.4484	0.9990	0.5268	0.9980
Recall	0.9990	0.9970	0.7041	0.6696	0.9990	0.7258	0.9980
F Score	0.9990	0.9971	0.5819	0.5371	0.9990	0.6105	0.9980
Process Time(minute)	09:49	09:49	09:55	11:04	09:47	13:01	09:37

Analysis of the data in the Layer Mode 2 design is shown that the two most successful results are obtained with the 'tanh' and 'softsign' activation methods.

The Confusion Matrix of these two methods is given in the Table 5.11.

TABLE 5.11: Confusion Matrix for Layer Mode 2 with Test Dataset

	-1	0	1		-1	0	1
-1	257	0	1	-1	237	0	0
0	0	54	0	0	0	68	0
1	0	0	702	1	1	0	708
	tanh				softsign		

The Accuracy and Loss graphs of the model using 'tanh' and 'softsign' activation methods are given in the Figure 5.24 - 5.27.

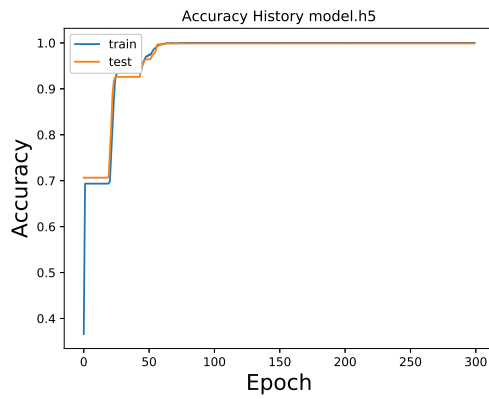


FIGURE 5.24: Triple Class Layer Mode 2 tanh Accuracy

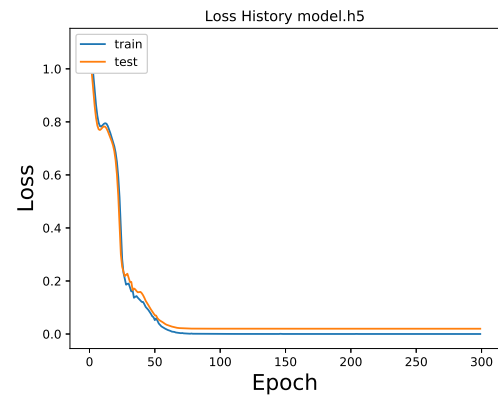


FIGURE 5.25: Triple Class Layer Mode 2 tanh Loss

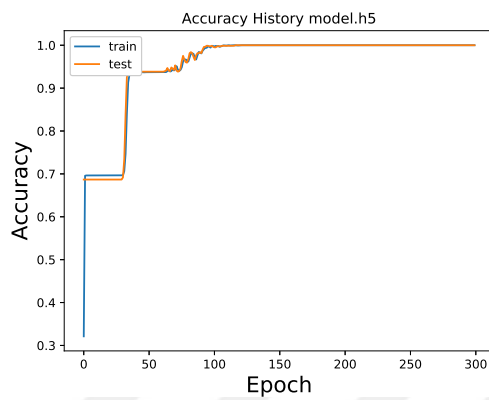


FIGURE 5.26: Triple Class Layer Mode 2 softsign Accuracy

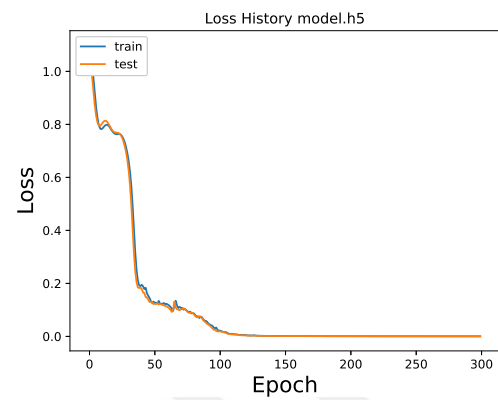


FIGURE 5.27: Triple Class Layer Mode 2 softsign Loss

When we run our model with the triple data class, we obtained 99.9% accuracy rate after nearly 100 Epochs. We obtained this accuracy rate with the DNN Model created with the following parameter sequence.

- DNN Optimizer Algorithm : Stochastic Gradient Descent(SGD)
- Nodes in the layers : 129(Input) - 400 - 650 - 400 - 150 - 3(Output)
- Dropout Rate : 20%
- Kernel Initializer : Uniform
- Activation Method : Softsign
- Epochs : 300
- Output Activation Method : softmax
- Output Loss Method : categorical_crossentropy

With these parameters, the training duration of our model is 9:46 minutes for 4055 different processes. The test duration is approximately 751 milliseconds for 1014 different processes and the detection time of the new incoming attack is 0.74 milliseconds.

The Layer Mode 3 design results of the Triple Data Class is given in detail in the Table 5.12.

TABLE 5.12: Triple Data Class Results for Layer Mode 3

	tanh	relu	sigmoid	softplus	softsign	softmax	linear
Epochs	500	500	500	500	500	500	500
Accuracy	0.9990	0.9961	0.6992	0.7071	0.9990	0.7061	0.9980
Presicion	0.9990	0.9961	0.4889	0.5000	0.9990	0.4986	0.9980
Recall	0.9990	0.9961	0.6992	0.7071	0.9990	0.7061	0.9980
F Score	0.9990	0.9960	0.5754	0.5858	0.9990	0.5845	0.9980
Process Time(minute)	43:27	41:56	42:16	48:44	41:10	51:24	40:41

Analysis of the data in the Layer Mode 3 design is shown that the two most successful results are obtained with the 'tanh' and 'softsign' activation methods.

The Confusion Matrix of these two methods is given in the Table 5.13.

TABLE 5.13: Confusion Matrix for Layer Mode 3 with Test Dataset

	-1	0	1		-1	0	1
-1	233	0	0	-1	258	1	0
0	0	66	0	0	0	55	0
1	0	1	714	1	0	0	700
	tanh				softsign		

The Accuracy and Loss graphs of the model using 'tanh' and 'softsign' activation methods are given in the Figure 5.28 - 5.31.

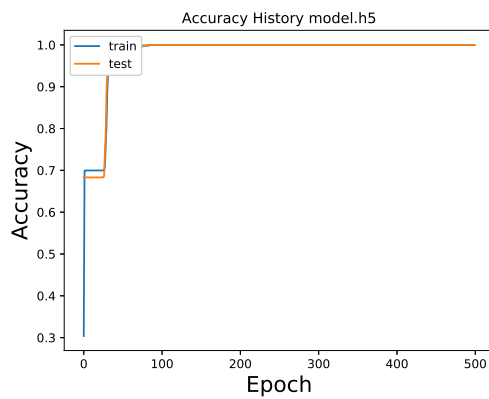


FIGURE 5.28: Triple Class Layer Mode 3 tanh Accuracy

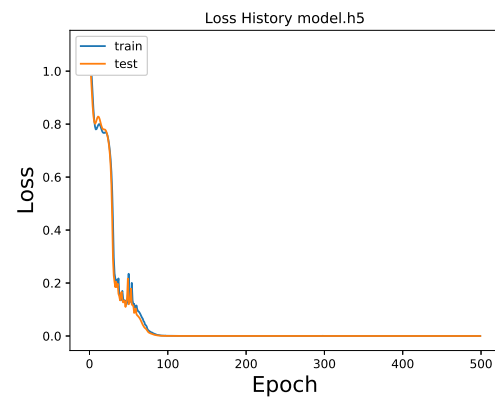


FIGURE 5.29: Triple Class Layer Mode 3 tanh Loss

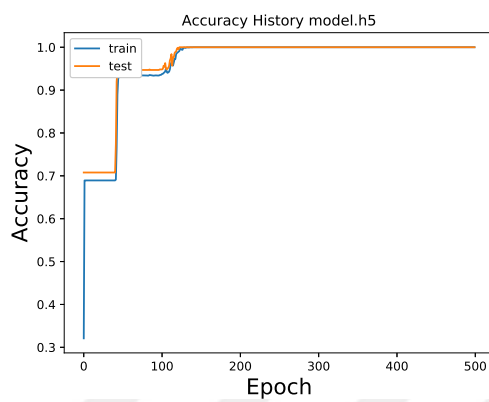


FIGURE 5.30: Triple Class Layer Mode 3 softsign Accuracy

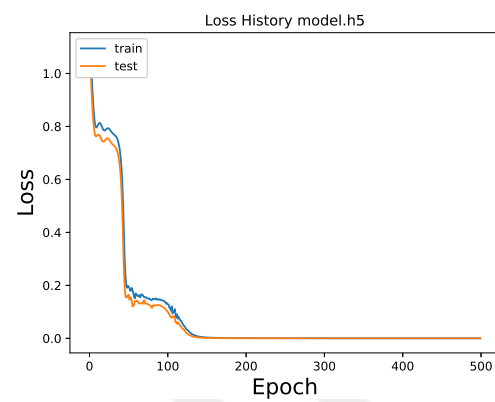


FIGURE 5.31: Triple Class Layer Mode 3 softsign Loss

When we run our model with the triple data class, we obtained 99.9% accuracy rate after nearly 120 Epochs. We obtained this accuracy rate with the DNN Model created with the following parameter sequence.

- DNN Optimizer Algorithm : Stochastic Gradient Descent(SGD)
- Nodes in the layers : 129(Input)-400-650-900-650-400-150-3(Output)
- Dropout Rate : 20%
- Kernel Initializer : Uniform
- Activation Method : Softsign
- Epochs : 500
- Output Activation Method : softmax
- Output Loss Method : categorical_crossentropy

With these parameters, the training duration of our model is 41 minutes for 4055 different processes. The test duration is approximately 1 second for 1014 different processes and the detection time of the new incoming attack is 1 milliseconds.

5.2.1.3 Multi Data Class Results

The model is run separately for 15 datasets in multi data class with multi tags with -2 Natural(Fault From Line), -1 Natural(Line maintenance), 0-Normal, 1-Attack (Data Injection), 2-Attack (Command Injection), 3-Attack (Command Injection), 4-Attack (Disabling relay function), 5-Attack (Disabling relay function), 6-Attack (Disabling relay function).

The Layer Mode 1 design results of the Multi Data Class is given in detail in the Table 5.14.

TABLE 5.14: Multi Data Class Results for Layer Mode 1

	tanh	relu	sigmoid	softplus	softsign	softmax	linear
Epochs	200	200	200	200	200	200	200
Accuracy	0.9911	0.9852	0.2061	0.3018	0.9980	0.2406	0.9073
Presicion	0.9913	0.9858	0.0425	0.1244	0.9980	0.0579	0.9192
Recall	0.9911	0.9852	0.2061	3018	0.9980	0.2406	0.9073
F Score	0.9911	0.9853	0.0704	0.1740	0.9980	0.0933	0.9068
Process Time(minute)	02:04	02:05	02:05	02:23	02:05	03:00	02:03

Analysis of the data in the Layer Mode 1 design is shown that the two most successful results are obtained with the 'tanh' and 'softsign' activation methods.

The Confusion Matrix of these two methods is given in the Table 5.15.

TABLE 5.15: Confusion Matrix for Layer Mode 1 with Test Dataset

	-2	-1	0	1	2	3	4	5	6
-2	50	0	0	0	0	0	0	0	0
-1	0	179	0	1	0	0	0	0	0
0	0	0	61	0	0	0	0	0	0
1	0	0	0	134	0	0	0	0	0
2	0	0	0	0	81	0	0	0	0
3	0	0	0	0	0	43	1	0	0
4	0	0	0	0	0	1	231	6	0
5	0	0	0	0	0	0	0	163	0
6	0	0	0	0	0	0	0	0	63

tanh

	-2	-1	0	1	2	3	4	5	6
-2	72	0	0	0	0	0	0	0	0
-1	0	183	0	0	0	0	0	0	0
0	0	0	63	0	0	0	0	0	0
1	0	0	0	122	0	0	0	0	0
2	0	0	0	0	82	0	0	0	0
3	0	0	0	0	0	50	0	0	0
4	0	0	0	0	0	0	219	0	0
5	0	0	0	0	0	0	1	135	0
6	0	0	0	0	0	0	0	0	86

softsign

The Accuracy and Loss graphs of the model using 'tanh' and 'softsign' activation methods are given in the Figure 5.32 - 5.35.

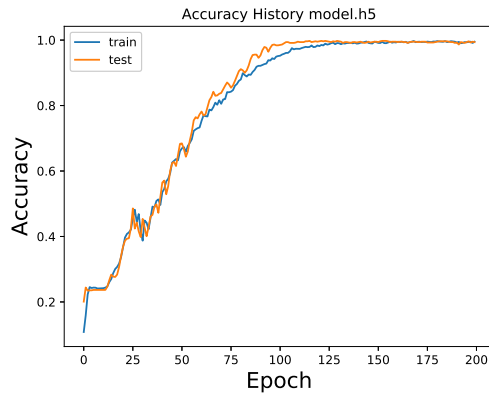


FIGURE 5.32: Multi Class Layer Mode 1 tanh Accuracy

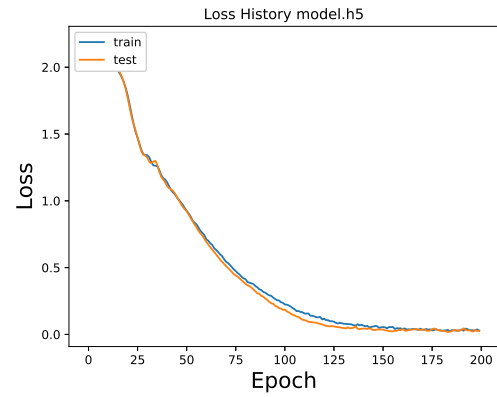


FIGURE 5.33: Multi Class Layer Mode 1 tanh Loss

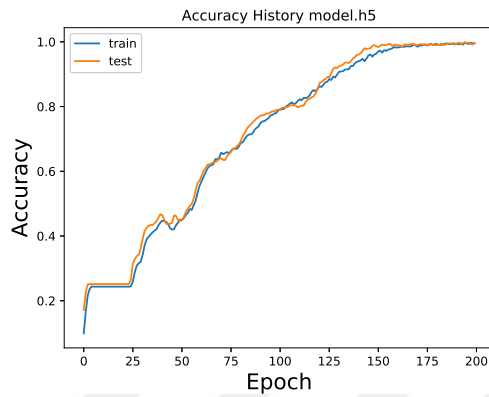


FIGURE 5.34: Multi Class Layer Mode 1 softsign Accuracy

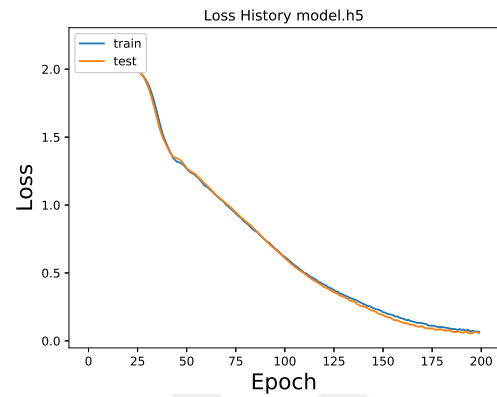


FIGURE 5.35: Multi Class Layer Mode 1 softsign Loss

When we run our model with the multi data class, we obtained 99.8% accuracy rate after nearly 160 Epochs. We obtained this accuracy rate with the DNN Model created with the following parameter sequence.

- DNN Optimizer Algorithm : Stochastic Gradient Descent(SGD)
- Nodes in the layers : 129(Input) - 400 - 150 - 9(Output)
- Dropout Rate : 20%
- Kernel Initializer : Uniform
- Activation Method : Softsign
- Epochs : 200
- Output Activation Method : softmax
- Output Loss Method : categorical_crossentropy

With these parameters, the training duration of our model is 2:04 minutes for 4055 different processes. The test duration is approximately 332 milliseconds for 1014 different processes and the detection time of the new incoming attack is 0.32 milliseconds.

The Layer Mode 2 design results of the Multi Data Class is given in detail in the Table 5.16.

TABLE 5.16: Multi Data Class Results for Layer Mode 2

	tanh	relu	sigmoid	softplus	softsign	softmax	linear
Epochs	300	300	300	300	300	300	300
Accuracy	0.9892	0.9448	0.2485	0.2436	0.9951	0.2249	0.2564
Presicion	0.9892	0.9448	0.0618	0.0496	0.9951	0.0506	0.1111
Recall	0.9892	0.9448	0.2485	0.2436	0.9951	0.2249	0.2564
F Score	0.9892	0.9446	0.0989	0.0954	0.9951	0.0826	0.1548
Process Time(minute)	10:15	10:11	10:31	11:46	10:25	13:40	10:10

Analysis of the data in the Layer Mode 2 design is shown that the two most successful results are obtained with the 'tanh' and 'softsign' activation methods.

The Confusion Matrix of these two methods is given in the Table 5.17.

TABLE 5.17: Confusion Matrix for Layer Mode 2 with Test Dataset

	-2	-1	0	1	2	3	4	5	6
-2	77	1	0	0	0	0	0	0	0
-1	1	171	0	2	0	0	0	0	0
0	0	0	63	0	0	0	0	0	0
1	0	0	0	131	1	0	0	0	0
2	0	0	0	0	70	0	0	0	0
3	0	0	0	0	0	42	0	0	0
4	0	0	0	0	0	1	242	3	0
5	0	0	0	0	0	0	2	156	0
6	0	0	0	0	0	0	0	0	51

tanh

	-2	-1	0	1	2	3	4	5	6
-2	85	0	0	0	0	0	0	0	0
-1	0	187	1	1	0	0	0	0	0
0	0	0	65	0	0	0	0	0	0
1	0	0	0	111	1	0	0	0	0
2	0	0	0	0	69	0	0	0	0
3	0	0	0	0	0	40	0	0	0
4	0	0	0	0	0	0	224	2	0
5	0	0	0	0	0	0	0	156	0
6	0	0	0	0	0	0	0	0	72

softsign

The Accuracy and Loss graphs of the model using 'tanh' and 'softsign' activation methods are given in the Figure 5.36 - 5.39.

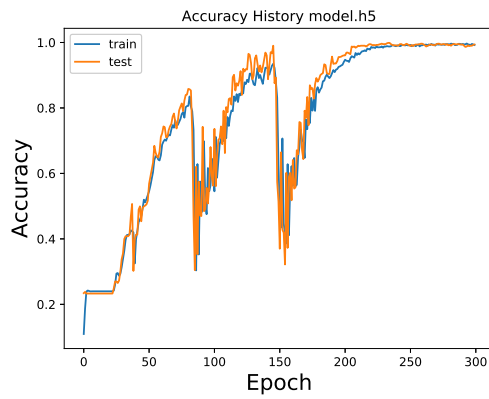


FIGURE 5.36: Multi Class Layer Mode 2 tanh Accuracy

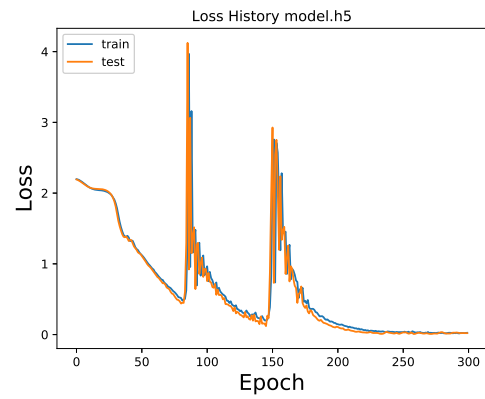


FIGURE 5.37: Multi Class Layer Mode 2 tanh Loss

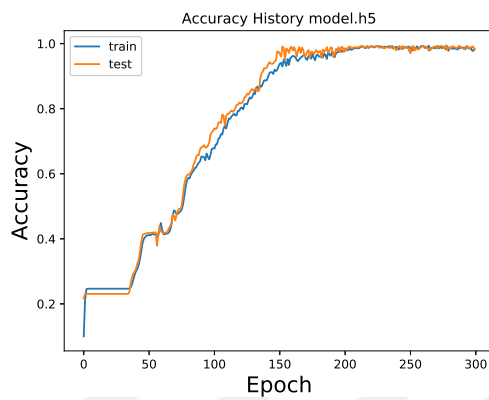


FIGURE 5.38: Multi Class Layer Mode 2 softsign Accuracy

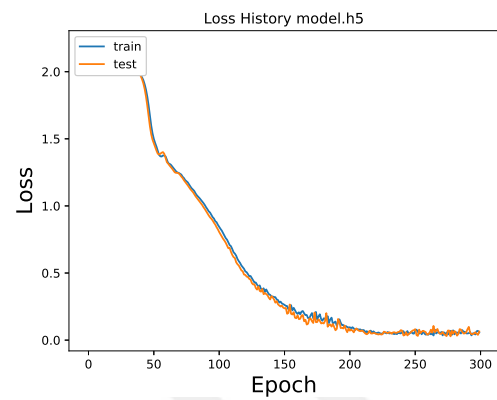


FIGURE 5.39: Multi Class Layer Mode 2 softsign Loss

When we run our model with the multi data class, we obtained 99.5% accuracy rate after nearly 200 Epochs. We obtained this accuracy rate with the DNN Model created with the following parameter sequence.

- DNN Optimizer Algorithm : Stochastic Gradient Descent(SGD)
- Nodes in the layers : 129(Input) - 400 - 650 - 400 - 150 - 9(Output)
- Dropout Rate : 20%
- Kernel Initializer : Uniform
- Activation Method : Softsign
- Epochs : 300
- Output Activation Method : softmax
- Output Loss Method : categorical_crossentropy

With these parameters, the training duration of our model is 10:25 minutes for 4055 different processes. The test duration is approximately 912 milliseconds for 1014 different processes and the detection time of the new incoming attack is 0.9 milliseconds.

The Layer Mode 3 design results of the Multi Data Class is given in detail in the Table 5.18.

TABLE 5.18: Multi Data Class Results for Layer Mode 3

	tanh	relu	sigmoid	softplus	softsign	softmax	linear
Epochs	500	500	500	500	500	500	500
Accuracy	0.9941	0.7949	0.2465	0.2387	0.9931	0.2505	0.1252
Presicion	0.9941	0.7127	0.0608	0.0570	0.9932	0.0627	0.0505
Recall	0.9941	0.7949	0.2465	0.2387	0.9931	0.2505	0.1252
F Score	0.9941	0.7313	0.0975	0.0920	0.9931	0.1004	0.0710
Process Time(minute)	43:41	43:47	43:49	46:55	42:42	53:38	42:44

Analysis of the data in the Layer Mode 3 design is shown that the two most successful results are obtained with the 'tanh' and 'softsign' activation methods.

The Confusion Matrix of these two methods is given in the Table 5.19.

TABLE 5.19: Confusion Matrix for Layer Mode 3 with Test Dataset

	-2	-1	0	1	2	3	4	5	6
-2	58	1	0	0	0	0	0	0	0
-1	0	194	0	0	0	0	0	0	0
0	0	0	73	0	0	0	0	0	0
1	0	0	1	125	0	0	0	0	0
2	0	0	0	0	66	0	0	0	0
3	0	0	0	0	0	36	0	0	0
4	0	0	0	0	0	0	233	1	0
5	0	0	0	0	0	0	3	142	2
6	0	0	0	0	0	0	0	0	81

tanh

	-2	-1	0	1	2	3	4	5	6
-2	72	0	0	0	0	0	0	0	0
-1	0	170	0	0	0	0	0	0	0
0	0	0	66	0	0	0	0	0	0
1	0	0	0	97	0	1	0	0	0
2	0	0	0	1	82	0	0	0	0
3	0	0	0	0	0	51	1	0	0
4	0	0	0	0	0	1	233	0	0
5	0	0	0	0	0	0	3	135	1
6	0	0	0	0	0	0	0	0	101

softsign

The Accuracy and Loss graphs of the model using 'tanh' and 'softsign' activation methods are given in the Figure 5.40 - 5.43.

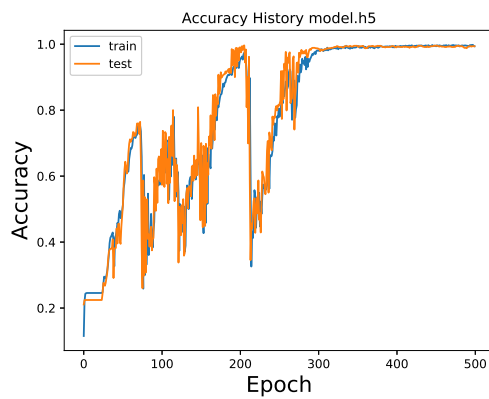


FIGURE 5.40: Multi Class Layer Mode 3 tanh Accuracy

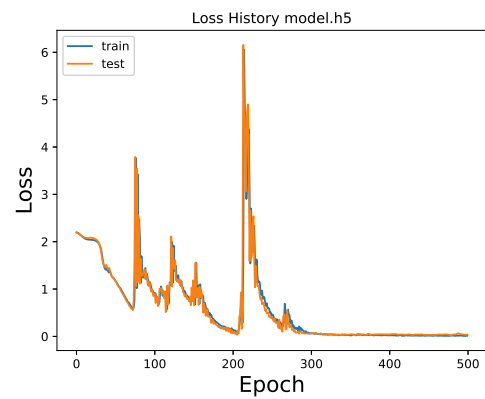


FIGURE 5.41: Multi Class Layer Mode 3 tanh Loss

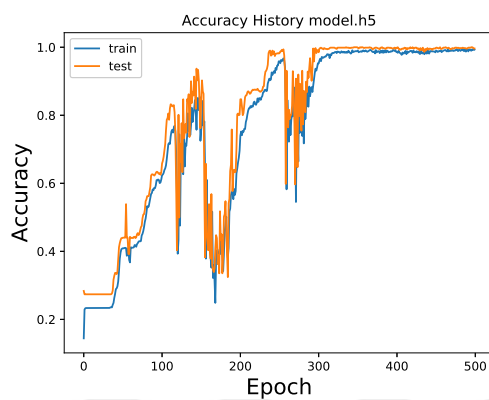


FIGURE 5.42: Multi Class Layer Mode 3 softsign Accuracy

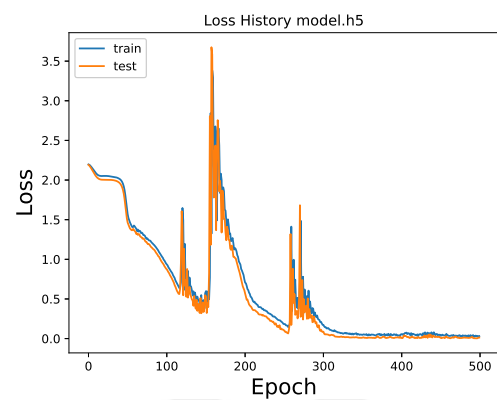


FIGURE 5.43: Multi Class Layer Mode 3 softsign Loss

When we run our model with the multi data class, we obtained 99.4% accuracy rate after nearly 300 Epochs. We obtained this accuracy rate with the DNN Model created with the following parameter sequence.

- DNN Optimizer Algorithm : Stochastic Gradient Descent(SGD)
- Nodes in the layers : 129(Input)-400-650-900-650-400-150-9(Output)
- Dropout Rate : 20%
- Kernel Initializer : Uniform
- Activation Method : Tanh
- Epochs : 500
- Output Activation Method : softmax
- Output Loss Method : categorical_crossentropy

With these parameters, the training duration of our model is 43:40 minutes for 4055 different processes. The test duration is approximately 1.1 seconds for 1014 different processes and the detection time of the new incoming attack is 1 millisecond.

5.2.2 AutoEncoder Results

For the best result of the AutoEncoder model, the activation methods were designed and run as sequence parameters as in the Deep Neural Network model.

The results were analyzed and the best result with the DNN Model was obtained by using the linear activation method, and the Linear activation method was used in the Encode and Decode layers of the AutoEncoder model.

The structure of the AutoEncoder Model is given in The Figure 5.44, 5.45. Dropout rate in the DNN Model and the Autoencoder Model is 20%



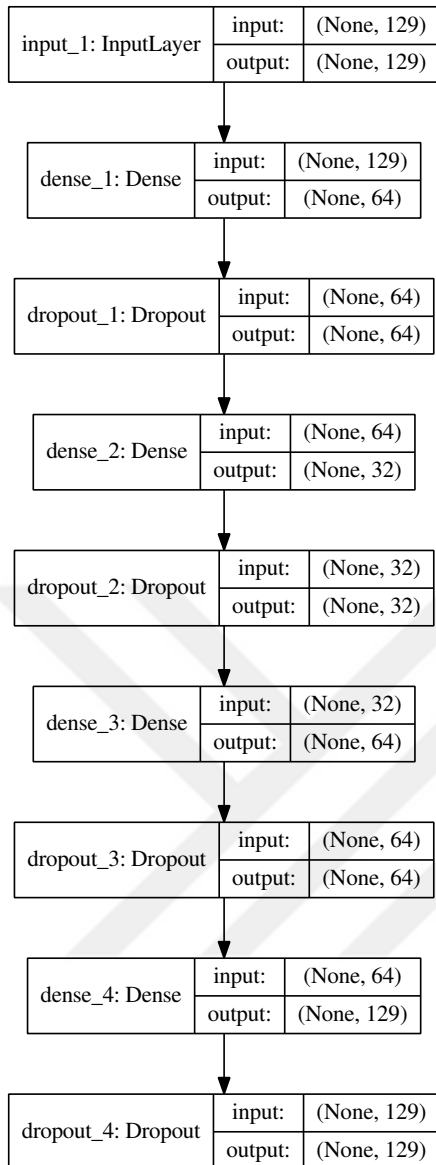


FIGURE 5.44: Autoencoder Model Structure

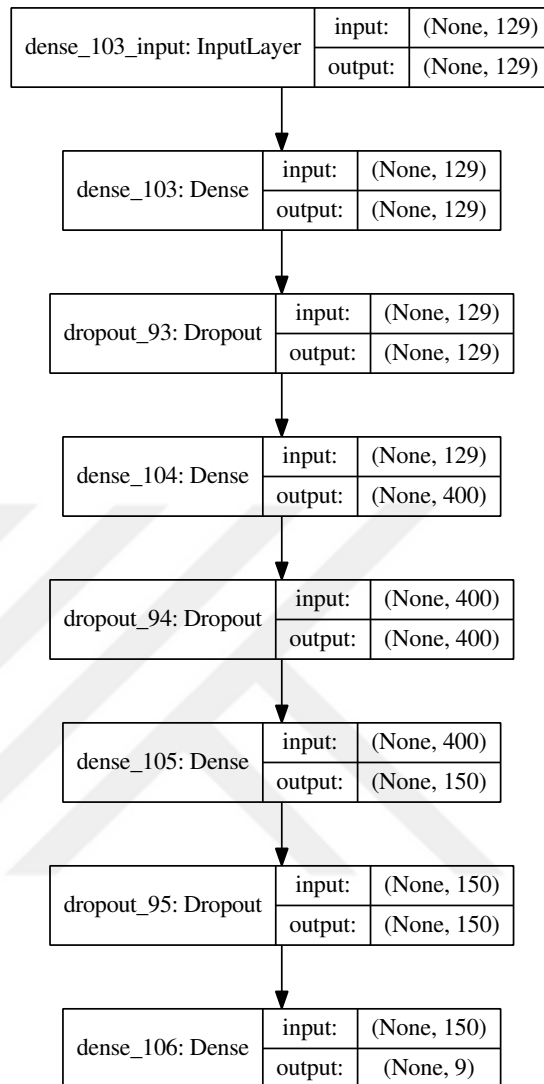


FIGURE 5.45: DNN Model Structure with Layer Mode 1

In the model there are 3 designs called Layer Mode 1, Layer Mode 2, Layer Mode 3.

The number of layers and nodes in these designs are given in the Table 5.20.

TABLE 5.20: Nodes in the layers with AutoEncoder

Layer Mode	AutoEncoder Model	DNN Model
1	129(input)-64-32-64-129(Output)	129(input)-400-150-2/3/9(Output)
2	129(input)-64-32-64-129(Output)	129(input)-400-650-400-150-2/3/9(Output)
3	129(input)-64-32-64-129(Output)	129(input)-400-650-900-650-400-150-2/3/9(Output)

There are too many parameters in the operation. To provide a more meaningful representation of the outputs of the study, the results of data classes called binary, triple, multi are presented separately below.

5.2.2.1 Binary Data Class Results with AutoEncoder Model

The model is run separately for 15 datasets in binary data class with binary tags with 0-Normal, 1-Attack.

The Layer Mode 1 design results of the Binary Data Class is given in detail in the Table 5.21.

TABLE 5.21: Binary Data Class Results for Layer Mode 1

	tanh	relu	sigmoid	softplus	softsign	softmax	linear
Epochs	200	200	200	200	200	200	200
Accuracy	1.0	0.9980	0.7081	0.9990	0.9990	0.7002	0.9980
Precision	1.0	0.9980	0.5014	0.9990	0.9990	0.4903	0.9980
Recall	1.0	0.9980	0.7081	0.9990	0.9990	0.7002	0.9980
F Score	1.0	0.9980	0.5871	0.9990	0.9990	0.5767	0.9980
Process Time(minute)	04:00	03:49	03:50	03:45	04:00	04:44	04:00

Analysis of the data in the Layer Mode 1 design is shown that the three most successful results are obtained with the 'tanh', 'softplus' and 'softsign' activation methods.

The Confusion Matrix of these three methods is given in the Table 5.22.

TABLE 5.22: Confusion Matrix for Layer Mode 1 with Test Dataset

	P	N
P	303	0
N	0	711

tanh

	P	N
P	307	1
N	0	706

softplus

	P	N
P	302	0
N	1	711

softsign

The Accuracy and Loss graphs of the model using 'tanh', 'softplus' and 'softsign' activation methods are given in the Figure 5.46 - 5.51.

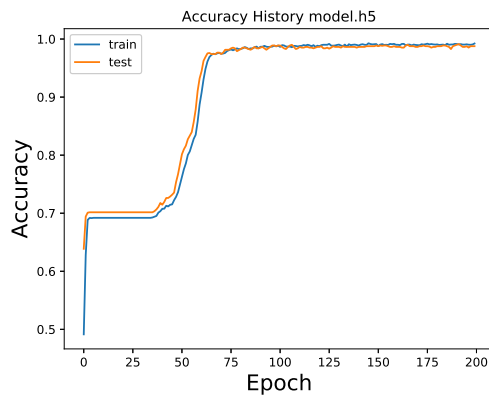


FIGURE 5.46: Binary Class Layer Mode 1 tanh Accuracy with AutoEncoder

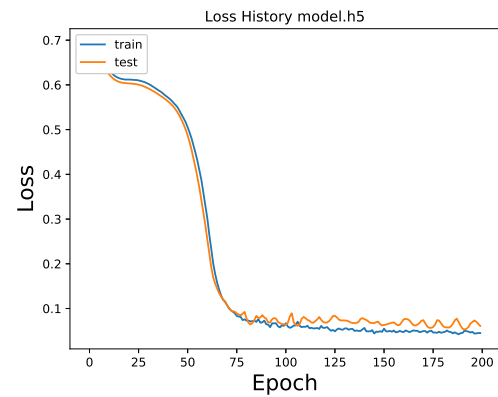


FIGURE 5.47: Binary Class Layer Mode 1 tanh Loss with AutoEncoder

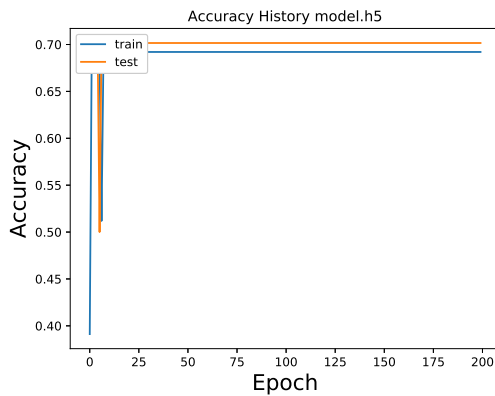


FIGURE 5.48: Binary Class Layer Mode 1 softplus Accuracy with AutoEncoder

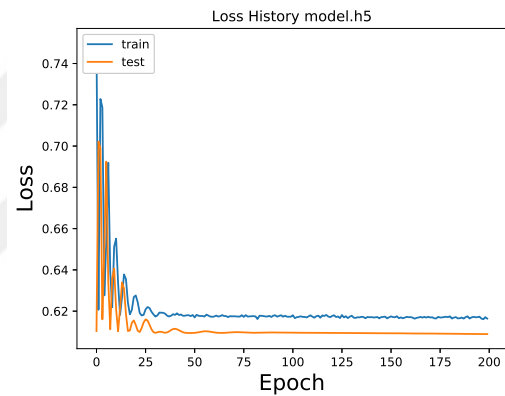


FIGURE 5.49: Binary Class Layer Mode 1 softplus Loss with AutoEncoder

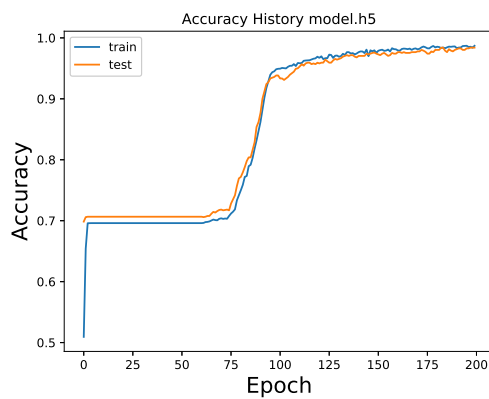


FIGURE 5.50: Binary Class Layer Mode 1 softsign Accuracy with AutoEncoder

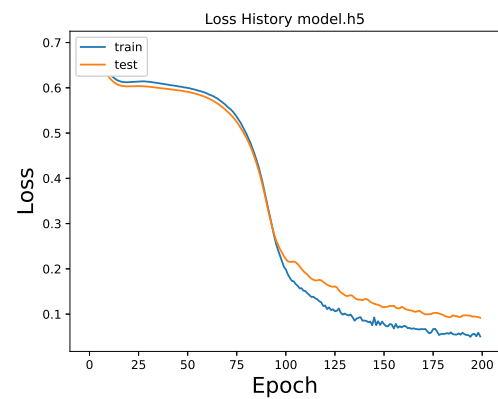


FIGURE 5.51: Binary Class Layer Mode 1 softsign Loss with AutoEncoder

When we run our model with the binary data class, we obtained 100% accuracy rate after nearly 75 Epochs. We obtained this accuracy rate with the DNN Model created with the following parameter sequence.

- AutoEncoder Optimizer Algorithm : Adam
- AutoEncoder, Nodes in the layers : 129(Input) - 64 - 32 - 64 - 129(Output)
- DNN Optimizer Algorithm : Stochastic Gradient Descent(SGD)
- DNN, Nodes in the layers : 129(Input) - 400 - 150 - 2(Output)
- Dropout Rate : 20%
- Kernel Initializer : Uniform
- Activation Method : Tanh
- Epochs : 200
- Output Activation Method : sigmoid
- Output Loss Method : binary_crossentropy

With these parameters, the training duration of our model is 4 minutes for 4055 different processes. The test duration is approximately 341 milliseconds for 1014 different processes and the detection time of the new incoming attack is 0.33 milliseconds.

The Layer Mode 2 design results of the Binary Data Class is given in detail in the Table 5.23.

TABLE 5.23: Binary Data Class Results for Layer Mode 2

	tanh	relu	sigmoid	softplus	softsign	softmax	linear
Epochs	300	300	300	300	300	300	300
Accuracy	1.0	0.9980	0.6854	0.7051	0.9990	0.6775	0.9990
Presicion	1.0	0.9980	0.4698	0.4972	0.9990	0.4590	0.9990
Recall	1.0	0.9980	0.6854	0.7051	0.9990	0.6775	0.9990
F Score	1.0	0.9980	0.5575	0.5832	0.9990	0.5473	0.9990
Process Time(minute)	11:24	11:41	11:53	13:14	12:04	15:36	12:38

Analysis of the data in the Layer Mode 2 design is shown that the three most successful results are obtained with the 'tanh', 'softsign' and 'linear' activation methods.

The Confusion Matrix of these three methods is given in the Table 5.24.

TABLE 5.24: Confusion Matrix for Layer Mode 2 with Test Dataset

	P	N		P	N		P	N
P	303	0	P	322	1	P	308	1
N	0	711	N	0	691	N	0	705
	tanh			softsign			linear	

The Accuracy and Loss graphs of the model using 'tanh', 'softsign' and 'linear' activation methods are given in the Figure 5.52 - 5.57.

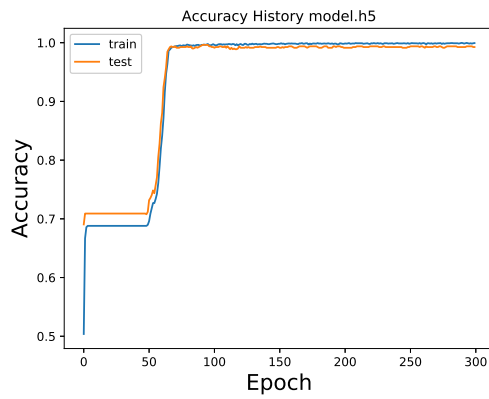


FIGURE 5.52: Binary Class Layer Mode 2 tanh Accuracy with AutoEncoder

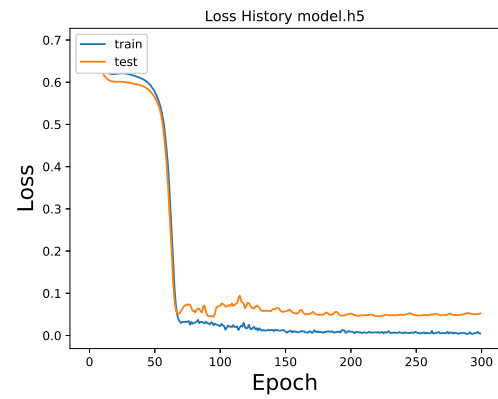


FIGURE 5.53: Binary Class Layer Mode 2 tanh Loss with AutoEncoder

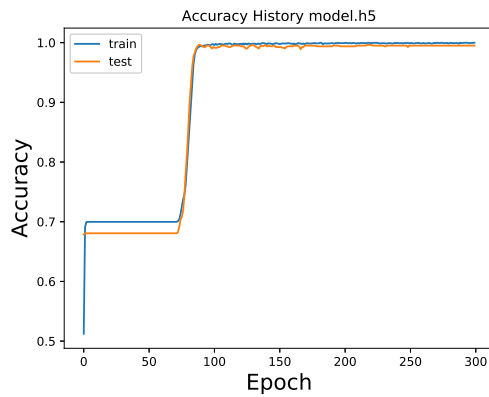


FIGURE 5.54: Binary Class Layer Mode 2 softsign Accuracy with AutoEncoder

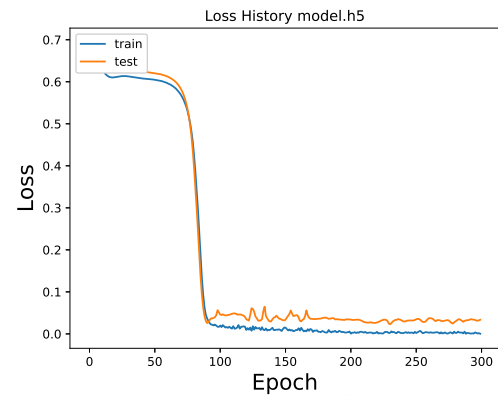


FIGURE 5.55: Binary Class Layer Mode 2 softsign Loss with AutoEncoder

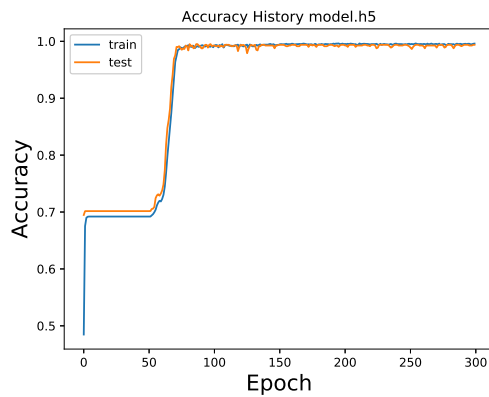


FIGURE 5.56: Binary Class Layer Mode 2 linear Accuracy with AutoEncoder

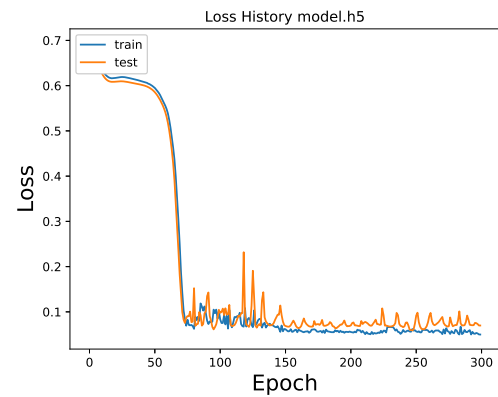


FIGURE 5.57: Binary Class Layer Mode 2 linear Loss with AutoEncoder

When we run our model with the binary data class, we obtained 100% accuracy rate after nearly 75 Epochs. We obtained this accuracy rate with the DNN Model created with the following parameter sequence.

- AutoEncoder Optimizer Algorithm : Adam
- AutoEncoder, Nodes in the layers : 129(Input) - 64 - 32 - 64 - 129(Output)
- DNN Optimizer Algorithm : Stochastic Gradient Descent(SGD)
- DNN, Nodes in the layers : 129(Input) - 400 - 650 - 400 - 150 - 2(Output)
- Dropout Rate : 20%
- Kernel Initializer : Uniform
- Activation Method : Tanh
- Epochs : 300
- Output Activation Method : sigmoid
- Output Loss Method : binary_crossentropy

With these parameters, the training duration of our model is 11:24 minutes for 4055 different processes. The test duration is approximately 844 milliseconds for 1014 different processes and the detection time of the new incoming attack is 0.83 milliseconds.

The Layer Mode 3 design results of the Binary Data Class is given in detail in the Table 5.25.

TABLE 5.25: Binary Data Class Results for Layer Mode 3

	tanh	relu	sigmoid	softplus	softsign	softmax	linear
Epochs	500	500	500	500	500	500	500
Accuracy	1.0	0.9980	0.7110	0.6667	0.9980	0.6953	0.9961
Presicion	1.0	0.9980	0.5056	0.4444	0.9980	0.4834	0.9961
Recall	1.0	0.9980	0.7110	0.6667	0.9980	0.6953	0.9961
F Score	1.0	0.9980	0.5910	0.5333	0.9980	0.5703	0.9961
Process Time(minute)	45:15	45:22	45:06	48:45	45:02	53:58	45:17

Analysis of the data in the Layer Mode 3 design is shown that the three most successful results are obtained with the 'tanh', 'relu' and 'softsign' activation methods.

The Confusion Matrix of these three methods is given in the Table 5.26.

TABLE 5.26: Confusion Matrix for Layer Mode 3 with Test Dataset

	P	N		P	N		P	N
P	316	0	P	299	2	P	338	2
N	0	698	N	0	713	N	0	674
tanh			relu			softsign		

The Accuracy and Loss graphs of the model using 'tanh', 'relu' and 'softsign' activation methods are given in the Figure 5.58 - 5.63.

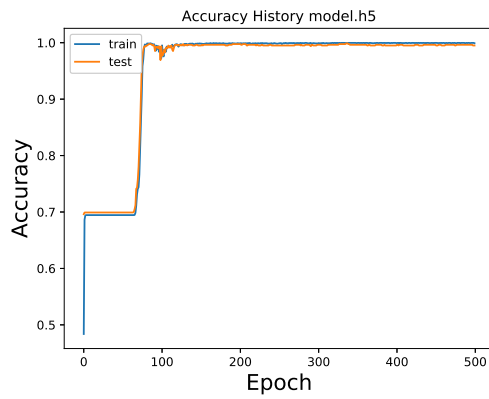


FIGURE 5.58: Binary Class Layer Mode 3 tanh Accuracy with AutoEncoder

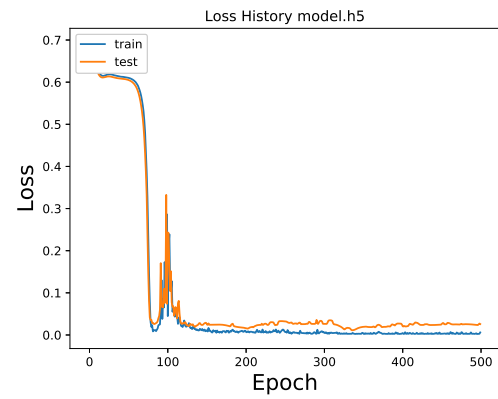


FIGURE 5.59: Binary Class Layer Mode 3 tanh Loss with AutoEncoder

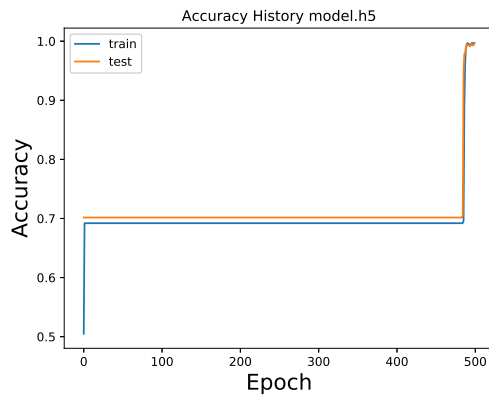


FIGURE 5.60: Binary Class Layer Mode 3 relu Accuracy with AutoEncoder

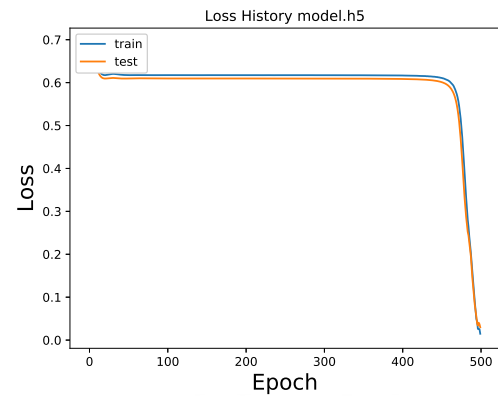


FIGURE 5.61: Binary Class Layer Mode 3 relu Loss with AutoEncoder

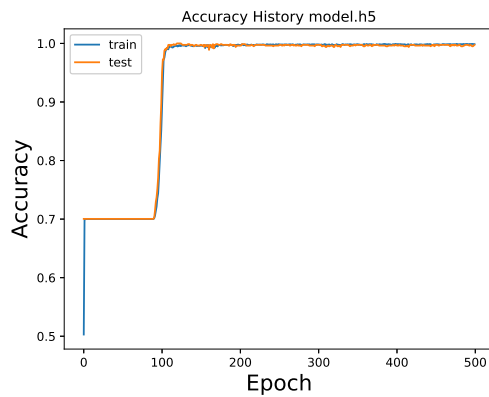


FIGURE 5.62: Binary Class Layer Mode 3 softsign Accuracy with AutoEncoder

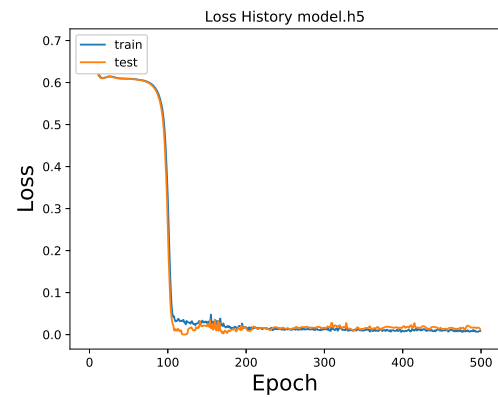


FIGURE 5.63: Binary Class Layer Mode 3 softsign Loss with AutoEncoder

When we run our model with the binary data class, we obtained 100% accuracy rate after nearly 120 Epochs. We obtained this accuracy rate with the DNN Model created with the following parameter sequence.

- AutoEncoder Optimizer Algorithm : Adam
- AutoEncoder, Nodes in the layers : 129(Input) - 64 - 32 - 64 - 129(Output)
- DNN Optimizer Algorithm : Stochastic Gradient Descent(SGD)
- DNN, Nodes in the layers : 129(Input)-400-650-900-650-400-150-2(Output)
- Dropout Rate : 20%
- Kernel Initializer : Uniform
- Activation Method : Tanh
- Epochs : 500
- Output Activation Method : sigmoid
- Output Loss Method : binary_crossentropy

With these parameters, the training duration of our model is 45:14 minutes for 4055 different processes. The test duration is approximately 1.5 seconds for 1014 different processes and the detection time of the new incoming attack is 1.4 milliseconds.

5.2.2.2 Triple Data Class Results with AutoEncoder Model

The model is run separately for 15 datasets in triple data class with triple tags with -1 Natural, 0-Normal, 1-Attack.

The Layer Mode 1 design results of the Triple Data Class is given in detail in the Table 5.27.

TABLE 5.27: Triple Data Class Results for Layer Mode 1

	tanh	relu	sigmoid	softplus	softsign	softmax	linear
Epochs	200	200	200	200	200	200	200
Accuracy	0.9990	0.9941	0.7081	0.9980	0.9990	0.7002	0.9970
Presicion	0.9990	0.9944	0.5014	0.9980	0.9990	0.4903	0.9971
Recall	0.9990	0.9941	0.7081	9980	0.9990	0.7002	0.9970
F Score	0.9990	0.9941	0.5871	0.9980	0.9990	0.5767	0.9970
Process Time(minute)	02:52	03:41	03:32	03:57	04:15	04:35	03:44

Analysis of the data in the Layer Mode 1 design is shown that the two most successful results are obtained with the 'tanh' and 'softsign' activation methods.

The Confusion Matrix of these two methods is given in the Table 5.28.

TABLE 5.28: Confusion Matrix for Layer Mode 1 with Test Dataset

	-1	0	1		-1	0	1
-1	250	0	0	-1	233	0	0
0	0	53	0	0	0	69	0
1	1	0	710	1	0	1	711
	tanh				softsign		

The Accuracy and Loss graphs of the model using 'tanh' and 'softsign' activation methods are given in the Figure 5.64 - 5.67.

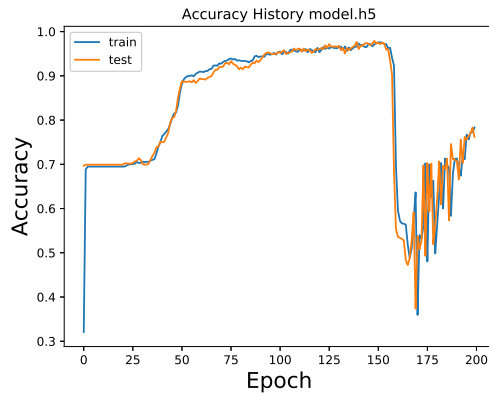


FIGURE 5.64: Triple Class Layer Mode 1 tanh Accuracy with AutoEncoder

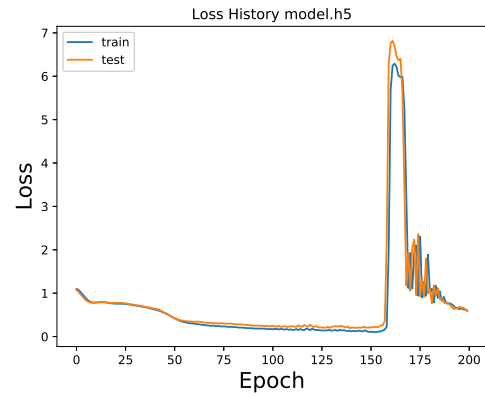


FIGURE 5.65: Triple Class Layer Mode 1 tanh Loss with AutoEncoder

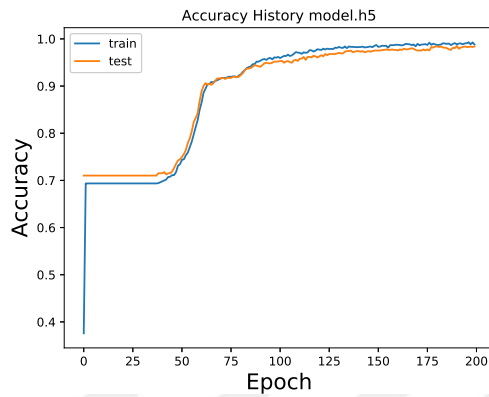


FIGURE 5.66: Triple Class Layer Mode 1 softsign Accuracy with AutoEncoder

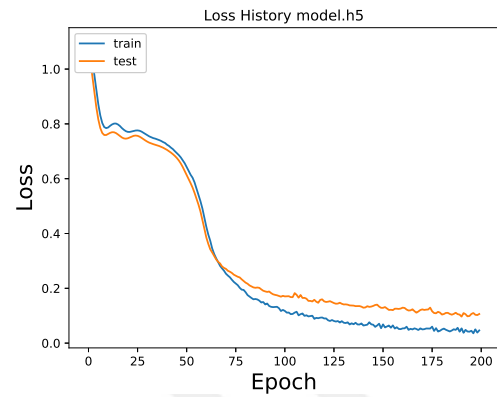


FIGURE 5.67: Triple Class Layer Mode 1 softsign Loss with AutoEncoder

When we run our model with the triple data class, we obtained 99.9% accuracy rate after nearly 150 Epochs. We obtained this accuracy rate with the DNN Model created with the following parameter sequence.

- AutoEncoder Optimizer Algorithm : Adam
- DNN Optimizer Algorithm : Stochastic Gradient Descent(SGD)
- Nodes in the layers : 129(Input) - 400 - 150 - 3(Output)
- Dropout Rate : 20%
- Kernel Initializer : Uniform
- Activation Method : Tanh
- Epochs : 200
- Output Activation Method : softmax
- Output Loss Method : categorical_crossentropy

With these parameters, the training duration of our model is 2:52 minutes for 4055 different processes. The test duration is approximately 413 milliseconds for 1014 different processes and the detection time of the new incoming attack is 0.4 milliseconds.

The Layer Mode 2 design results of the Triple Data Class is given in detail in the Table 5.29.

TABLE 5.29: Triple Data Class Results for Layer Mode 2

	tanh	relu	sigmoid	softplus	softsign	softmax	linear
Epochs	300	300	300	300	300	300	300
Accuracy	0.9961	0.9990	0.7130	0.7081	0.9970	0.7012	0.9951
Presicion	0.9961	0.9990	0.5084	0.5014	0.9971	0.4917	0.9951
Recall	0.9961	0.9990	0.7130	0.7081	0.9970	0.7012	0.9951
F Score	0.9960	0.9990	0.5936	0.5871	0.9971	0.5780	0.9951
Process Time(minute)	13:50	14:00	14:29	15:59	14:33	17:52	14:14

Analysis of the data in the Layer Mode 2 design is shown that the two most successful results are obtained with the 'relu' and 'softsign' activation methods.

The Confusion Matrix of these two methods is given in the Table 5.30.

TABLE 5.30: Confusion Matrix for Layer Mode 2 with Test Dataset

	-1	0	1		-1	0	1
-1	244	0	1	-1	241	1	0
0	0	71	0	0	0	73	0
1	0	0	698	1	1	1	697
	relu				softsign		

The Accuracy and Loss graphs of the model using 'relu' and 'softsign' activation methods are given in the Figure 5.68 - 5.71.

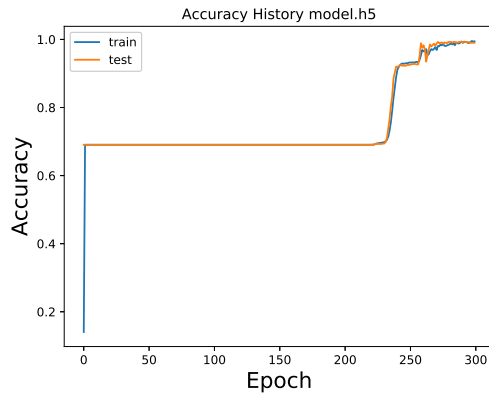


FIGURE 5.68: Triple Class Layer Mode 2 relu Accuracy with AutoEncoder

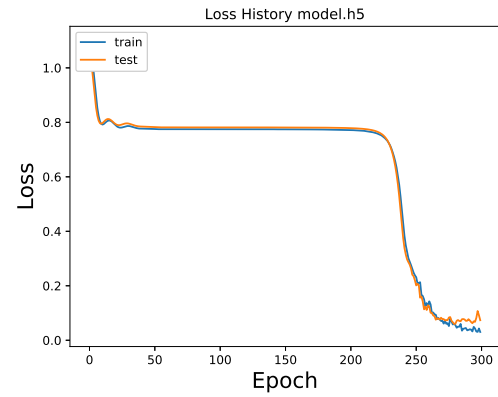


FIGURE 5.69: Triple Class Layer Mode 2 relu Loss with AutoEncoder

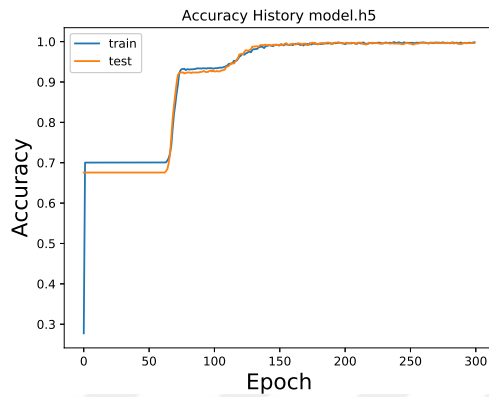


FIGURE 5.70: Triple Class Layer Mode 2 softsign Accuracy with AutoEncoder

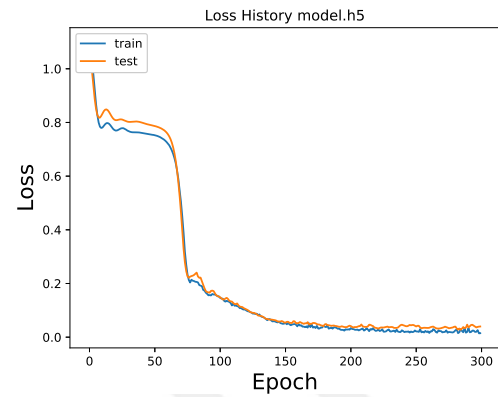


FIGURE 5.71: Triple Class Layer Mode 2 softsign Loss with AutoEncoder

When we run our model with the triple data class, we obtained 99.7% accuracy rate after nearly 150 Epochs. We obtained this accuracy rate with the DNN Model created with the following parameter sequence.

- AutoEncoder Optimizer Algorithm : Adam
- DNN Optimizer Algorithm : Stochastic Gradient Descent(SGD)
- Nodes in the layers : 129(Input) - 400 - 650 - 400 - 150 - 3(Output)
- Dropout Rate : 20%
- Kernel Initializer : Uniform
- Activation Method : Softsign
- Epochs : 300
- Output Activation Method : softmax
- Output Loss Method : categorical_crossentropy

With these parameters, the training duration of our model is 14:33 minutes for 4055 different processes. The test duration is approximately 745 milliseconds for 1014 different processes and the detection time of the new incoming attack is 0.73 milliseconds.

The Layer Mode 3 design results of the Triple Data Class is given in detail in the Table 5.31.

TABLE 5.31: Triple Data Class Results for Layer Mode 3

	tanh	relu	sigmoid	softplus	softsign	softmax	linear
Epochs	500	500	500	500	500	500	500
Accuracy	0.9970	0.9980	0.6933	0.6953	0.9980	0.6893	1.0
Presicion	0.9971	0.9980	0.4807	0.4834	0.9980	0.4752	1.0
Recall	0.9970	0.9980	0.6933	0.6953	0.9980	0.6893	1.0
F Score	0.9970	0.9980	0.5677	0.5703	0.9980	0.5626	1.0
Process Time(minute)	47:33	46:36	49:52	51:53	46:33	1:00:11	49:44

Analysis of the data in the Layer Mode 3 design is shown that the three most successful results are obtained with the 'relu', 'softsign' and 'linear' activation methods.

The Confusion Matrix of these three methods is given in the Table 5.32.

TABLE 5.32: Confusion Matrix for Layer Mode 3 with Test Dataset

	-1	0	1		-1	0	1		-1	0	1
-1	229	0	1	-1	233	0	1	-1	250	0	0
0	0	65	0	0	0	58	0	0	0	53	0
1	1	0	718	1	1	0	721	1	0	0	711
	relu				softsign				linear		

The Accuracy and Loss graphs of the model using 'relu', 'softsign' and 'linear' activation methods are given in the Figure 5.72 - 5.77.

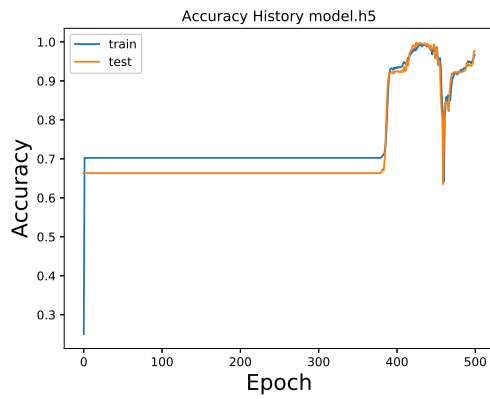


FIGURE 5.72: Triple Class Layer Mode 3 relu Accuracy with AutoEncoder

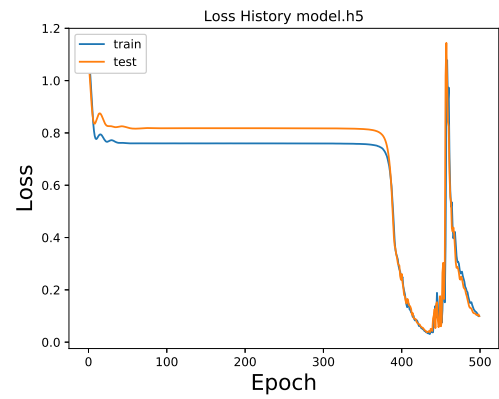


FIGURE 5.73: Triple Class Layer Mode 3 relu Loss with AutoEncoder

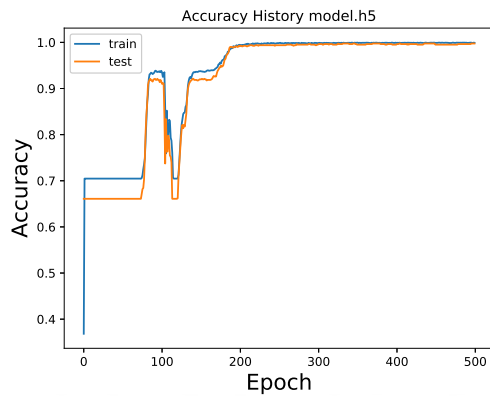


FIGURE 5.74: Triple Class Layer Mode 3 softsign Accuracy with AutoEncoder

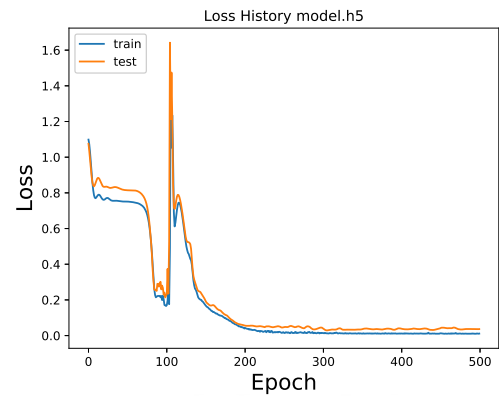


FIGURE 5.75: Triple Class Layer Mode 3 softsign Loss with AutoEncoder

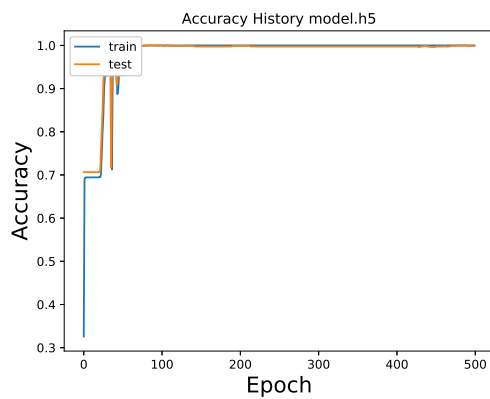


FIGURE 5.76: Triple Class Layer Mode 3 linear Accuracy with AutoEncoder

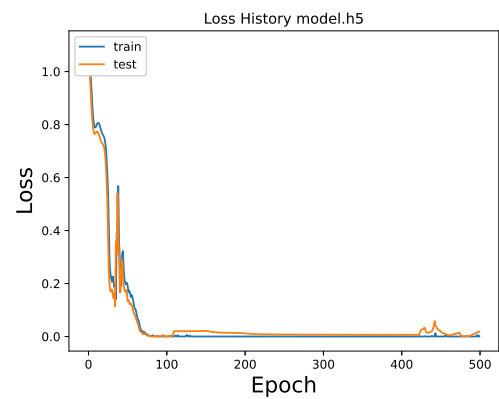


FIGURE 5.77: Triple Class Layer Mode 3 linear Loss with AutoEncoder

When we run our model with the triple data class, we obtained 100% accuracy rate after nearly 80 Epochs. We obtained this accuracy rate with the DNN Model created with the following parameter sequence.

- AutoEncoder Optimizer Algorithm : Adam
- DNN Optimizer Algorithm : Stochastic Gradient Descent(SGD)
- Nodes in the layers : 129(Input)-400-650-900-650-400-150-3(Output)
- Dropout Rate : 20%
- Kernel Initializer : Uniform
- Activation Method : Linear
- Epochs : 500
- Output Activation Method : softmax
- Output Loss Method : categorical_crossentropy

With these parameters, the training duration of our model is 49:43 minutes for 4055 different processes. The test duration is approximately 1.7 seconds for 1014 different processes and the detection time of the new incoming attack is 1.6 milliseconds.

5.2.2.3 Multi Data Class Results with AutoEncoder Model

The model is run separately for 15 datasets in multi data class with multi tags with -2 Natural(Fault From Line), -1 Natural(Line maintenance), 0-Normal, 1-Attack (Data Injection), 2-Attack (Command Injection), 3-Attack (Command Injection), 4-Attack (Disabling relay function), 5-Attack (Disabling relay function), 6-Attack (Disabling relay function).

The Layer Mode 1 design results of the Multi Data Class is given in detail in the Table 5.33.

TABLE 5.33: Multi Data Class Results for Layer Mode 1

	tanh	relu	sigmoid	softplus	softsign	softmax	linear
Epochs	200	200	200	200	200	200	200
Accuracy	0.9753	0.8294	0.2347	0.5039	0.9832	0.2633	0.7081
Presicion	0.9757	0.8648	0.0551	0.3673	0.9980	0.0693	0.7843
Recall	0.9753	0.8294	0.2347	5039	0.9832	0.2633	0.7081
F Score	0.9752	0.8144	0.0892	0.3992	0.9831	0.1098	0.6805
Process Time(minute)	03:59	04:14	04:17	04:47	04:35	05:33	04:43

Analysis of the data in the Layer Mode 1 design is shown that the two most successful results are obtained with the 'tanh' and 'softsign' activation methods.

The Confusion Matrix of these two methods is given in the Table 5.34.

TABLE 5.34: Confusion Matrix for Layer Mode 1 with Test Dataset

	-2	-1	0	1	2	3	4	5	6
-2	53	2	0	0	0	0	0	0	0
-1	3	161	1	2	0	0	0	0	0
0	0	0	81	1	0	0	0	0	0
1	0	0	0	121	0	0	0	0	0
2	0	0	0	1	76	0	0	0	0
3	0	0	0	1	5	41	1	0	0
4	0	0	0	1	0	1	230	2	0
5	0	0	0	0	0	0	3	153	0
6	0	0	0	0	0	0	0	0	73

tanh

	-2	-1	0	1	2	3	4	5	6
-2	79	2	0	0	0	0	0	0	0
-1	2	176	0	0	0	0	0	0	0
0	0	0	52	1	0	0	0	0	0
1	0	0	1	112	0	0	0	0	0
2	0	0	0	1	73	1	0	0	0
3	0	0	0	0	4	36	2	0	0
4	0	0	0	1	0	0	252	0	0
5	0	0	0	0	0	0	1	141	1
6	0	0	0	0	0	0	0	0	76

softsign

The Accuracy and Loss graphs of the model using 'tanh' and 'softsign' activation methods are given in the Figure 5.78 - 5.81.

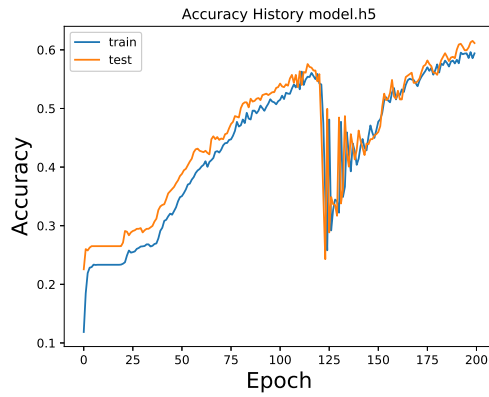


FIGURE 5.78: Multi Class Layer Mode 1 tanh Accuracy with AutoEncoder

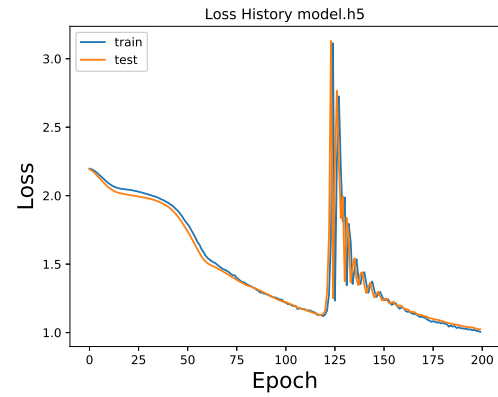


FIGURE 5.79: Multi Class Layer Mode 1 tanh Loss with AutoEncoder

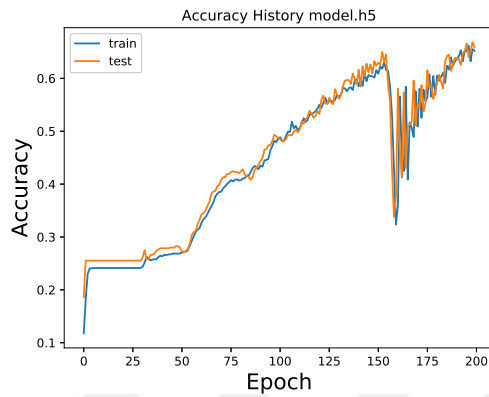


FIGURE 5.80: Multi Class Layer Mode 1 softsign Accuracy with AutoEncoder

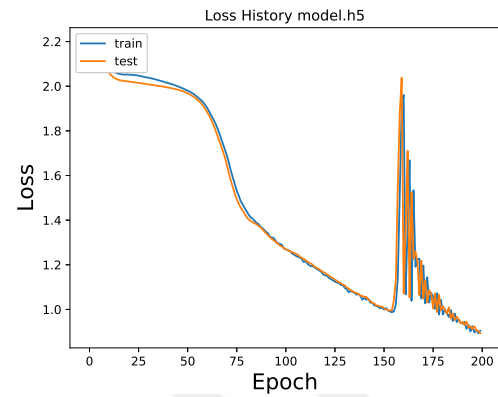


FIGURE 5.81: Multi Class Layer Mode 1 softsign Loss with AutoEncoder

When we run our model with the multi data class, we obtained 98.3% accuracy rate after nearly 200 Epochs. We obtained this accuracy rate with the DNN Model created with the following parameter sequence.

- AutoEncoder Optimizer Algorithm : Adam
- DNN Optimizer Algorithm : Stochastic Gradient Descent(SGD)
- Nodes in the layers : 129(Input) - 400 - 150 - 9(Output)
- Dropout Rate : 20%
- Kernel Initializer : Uniform
- Activation Method : Softsign
- Epochs : 200
- Output Activation Method : softmax
- Output Loss Method : categorical_crossentropy

With these parameters, the training duration of our model is 4:35 minutes for 4055 different processes. The test duration is approximately 576 milliseconds for 1014 different processes and the detection time of the new incoming attack is 0.56 milliseconds.

The Layer Mode 2 design results of the Multi Data Class is given in detail in the Table 5.35.

TABLE 5.35: Multi Data Class Results for Layer Mode 2

	tanh	relu	sigmoid	softplus	softsign	softmax	linear
Epochs	300	300	300	300	300	300	300
Accuracy	0.9872	0.9172	0.2288	0.2347	0.9773	0.2229	0.8008
Presicion	0.9872	0.9203	0.0523	0.0551	0.9776	0.0497	0.8073
Recall	0.9872	0.9172	0.2288	0.2347	0.9773	0.2229	0.8008
F Score	0.9872	0.9146	0.0852	0.0892	0.9774	0.0812	0.7856
Process Time(minute)	17:54	15:28	16:18	19:31	19:12	18:03	15:12

Analysis of the data in the Layer Mode 2 design is shown that the two most successful results are obtained with the 'tanh' and 'softsign' activation methods.

The Confusion Matrix of these two methods is given in the Table 5.36.

TABLE 5.36: Confusion Matrix for Layer Mode 2 with Test Dataset

	-2	-1	0	1	2	3	4	5	6
-2	63	0	0	0	0	0	0	0	0
-1	1	184	0	1	1	0	0	0	0
0	0	0	52	1	0	0	0	0	0
1	0	3	0	113	0	0	0	0	0
2	0	0	0	0	77	0	0	0	0
3	0	0	0	0	0	45	2	0	0
4	0	0	0	0	0	0	236	2	0
5	0	0	0	0	0	0	2	150	0
6	0	0	0	0	0	0	0	0	81

tanh

	-2	-1	0	1	2	3	4	5	6
-2	78	1	0	0	0	0	0	0	0
-1	1	173	1	2	0	0	0	0	0
0	0	0	67	0	0	0	0	0	0
1	0	0	0	103	3	0	0	0	0
2	0	0	0	1	66	2	0	0	0
3	0	0	0	0	2	32	1	0	0
4	0	0	0	0	0	1	229	1	0
5	0	0	0	0	0	0	7	169	0
6	0	0	0	0	0	0	0	0	74

softsign

The Accuracy and Loss graphs of the model using 'tanh' and 'softsign' activation methods are given in the Figure 5.82 - 5.85.

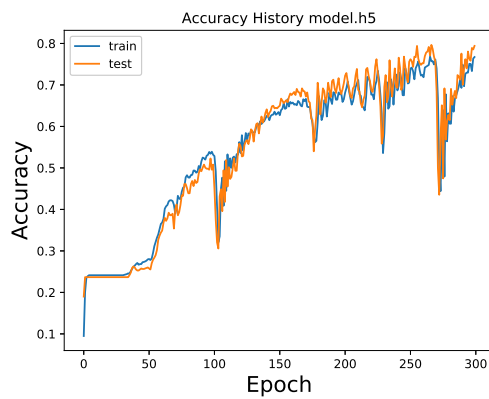


FIGURE 5.82: Multi Class Layer Mode 2 tanh Accuracy with AutoEncoder

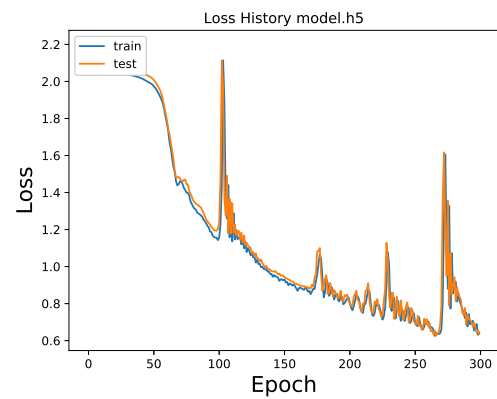


FIGURE 5.83: Multi Class Layer Mode 2 tanh Loss with AutoEncoder

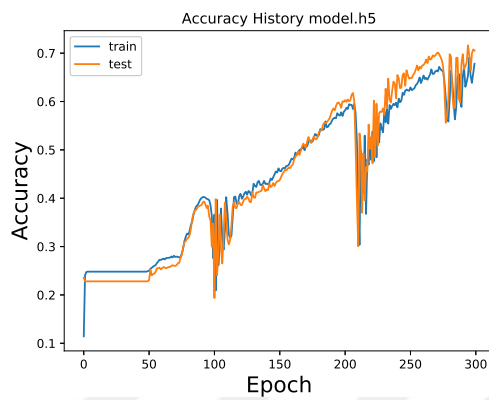


FIGURE 5.84: Multi Class Layer Mode 2 softsign Accuracy with AutoEncoder

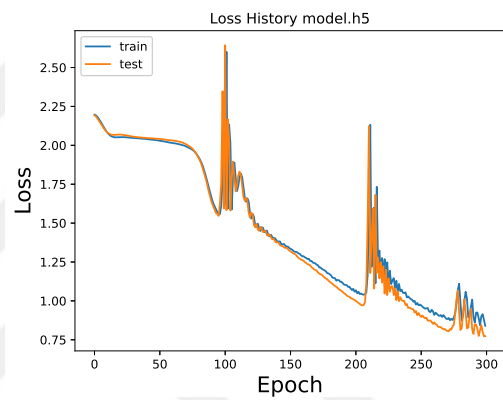


FIGURE 5.85: Multi Class Layer Mode 2 softsign Loss with AutoEncoder

When we run our model with the multi data class, we obtained 98.7% accuracy rate after nearly 300 Epochs. We obtained this accuracy rate with the DNN Model created with the following parameter sequence.

- AutoEncoder Optimizer Algorithm : Adam
- DNN Optimizer Algorithm : Stochastic Gradient Descent(SGD)
- Nodes in the layers : 129(Input) - 400 - 650 - 400 - 150 - 9(Output)
- Dropout Rate : 20%
- Kernel Initializer : Uniform
- Activation Method : Tanh
- Epochs : 300
- Output Activation Method : softmax
- Output Loss Method : categorical_crossentropy

With these parameters, the training duration of our model is 17:54 minutes for 4055 different processes. The test duration is approximately 894 milliseconds for 1014 different processes and the detection time of the new incoming attack is 0.88 milliseconds.

The Layer Mode 3 design results of the Multi Data Class is given in detail in the Table 5.37.

TABLE 5.37: Multi Data Class Results for Layer Mode 3

	tanh	relu	sigmoid	softplus	softsign	softmax	linear
Epochs	500	500	500	500	500	500	500
Accuracy	0.9813	0.6943	0.2485	0.2308	0.9901	0.2594	0.1667
Presicion	0.9813	0.6475	0.0618	0.0533	0.9903	0.0673	0.0890
Recall	0.9813	0.6943	0.2485	0.2308	0.9901	0.2594	0.1667
F Score	0.9812	0.6331	0.0989	0.0865	0.9902	0.1068	0.0985
Process Time(minute)	49:14	55:22	1:02:20	49:04	44:07	56:39	46:23

Analysis of the data in the Layer Mode 3 design is shown that the two most successful results are obtained with the 'tanh' and 'softsign' activation methods.

The Confusion Matrix of these two methods is given in the Table 5.38.

TABLE 5.38: Confusion Matrix for Layer Mode 3 with Test Dataset

	-2	-1	0	1	2	3	4	5	6
-2	60	1	0	0	0	0	0	0	0
-1	2	185	0	0	0	0	0	0	0
0	0	0	69	0	0	0	0	0	0
1	0	0	1	111	1	0	0	0	0
2	0	0	0	0	86	1	0	0	0
3	0	0	0	1	0	41	3	0	0
4	0	0	0	1	1	1	218	4	0
5	0	0	0	0	0	1	0	148	2
6	0	0	0	0	0	0	0	0	77

tanh

	-2	-1	0	1	2	3	4	5	6
-2	60	0	0	0	0	0	0	0	0
-1	0	203	2	2	0	0	0	0	0
0	0	0	73	0	0	0	0	0	0
1	0	0	0	114	1	0	0	0	0
2	0	0	0	1	69	0	0	0	0
3	0	0	0	0	0	33	0	0	0
4	0	0	0	0	0	0	239	1	0
5	0	0	0	0	0	0	1	145	1
6	0	0	0	0	0	0	0	0	68

softsign

The Accuracy and Loss graphs of the model using 'tanh' and 'softsign' activation methods are given in the Figure 5.86 - 5.89.

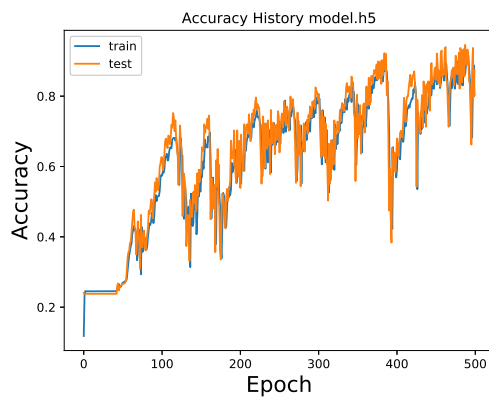


FIGURE 5.86: Multi Class Layer Mode 3 tanh Accuracy with AutoEncoder

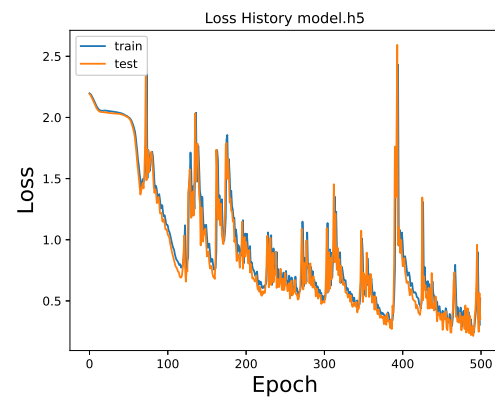


FIGURE 5.87: Multi Class Layer Mode 3 tanh Loss with AutoEncoder

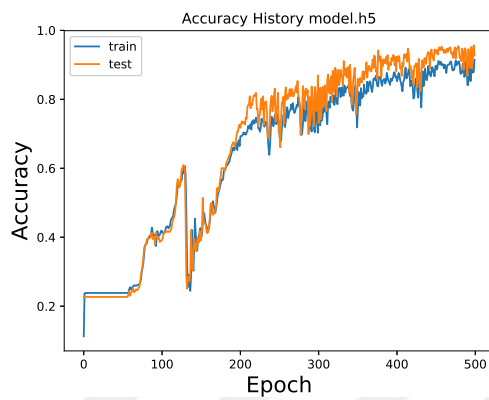


FIGURE 5.88: Multi Class Layer Mode 3 softsign Accuracy with AutoEncoder

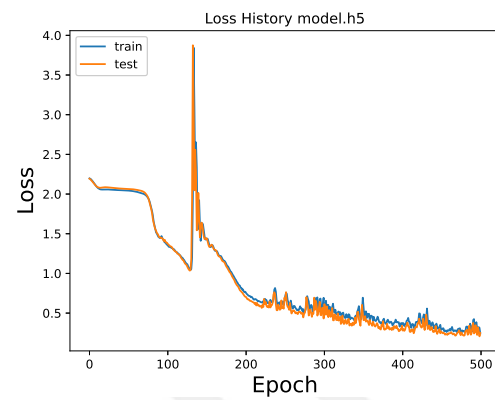


FIGURE 5.89: Multi Class Layer Mode 3 softsign Loss with AutoEncoder

When we run our model with the multi data class, we obtained 99% accuracy rate after nearly 500 Epochs. We obtained this accuracy rate with the DNN Model created with the following parameter sequence.

- AutoEncoder Optimizer Algorithm : Adam
- DNN Optimizer Algorithm : Stochastic Gradient Descent(SGD)
- Nodes in the layers : 129(Input)-400-650-900-650-400-150-9(Output)
- Dropout Rate : 20%
- Kernel Initializer : Uniform
- Activation Method : Softsign
- Epochs : 500
- Output Activation Method : softmax
- Output Loss Method : categorical_crossentropy

With these parameters, the training duration of our model is 44 minutes for 4055 different processes. The test duration is approximately 1.8 seconds for 1014 different processes and the detection time of the new incoming attack is 1.8 milliseconds.

5.2.3 Classification Models Results

We used 3 different algorithms in the classification models. Support Vector Machine (SVM) algorithm, K-Nearest Neighbors (KNN) algorithm and Decision Trees (DTs) algorithm. We used four different kernel types in the SVM algorithm; 'rbf', 'linear', 'poly', 'sigmoid'.

5.2.3.1 Binary Data Class Results with Classification Models

The models are run in binary data class with binary tags with 0-Normal, 1-Attack.

The results of the three algorithms used in classification models are given in detail in the following Table 5.39.

TABLE 5.39: Binary Data Class Results with Classification Models

	SVM	SVM	SVM	SVM	KNN	DTs
Kernel Types	rbf	linear	poly	sigmoid	-	-
Accuracy	0.9941	0.9901	0.9951	0.9941	0.9596	1.0
Presicion	0.9941	0.9903	0.9951	0.9941	0.9598	1.0
Recall	0.9941	0.9901	0.9951	0.9941	0.9596	1.0
F Score	0.9941	0.9901	0.9951	0.9941	0.9591	1.0
Process Time(minute)	00:02	00:02	00:02	00:02	00:04	00:00.08

When the results of Classification Models using Binary Data Class are analyzed, the best result is obtained with DTs algorithm.

The Confusion Matrix of this classification model using the DTs algorithm is given in the Table 5.40.

TABLE 5.40: Confusion Matrix for Binary Data Class with Test Dataset

	P	N
P	303	0
N	0	711

5.2.3.2 Triple Data Class Results with Classification Models

The models are run in triple data class with triple tags with -1 Natural, 0-Normal, 1-Attack.

The results of the three algorithms used in classification models are given in detail in the following Table 5.41.

TABLE 5.41: Triple Data Class Results with Classification Models

	SVM	SVM	SVM	SVM	KNN	DTs
Kernel Types	rbf	linear	poly	sigmoid	-	-
Accuracy	0.9951	0.9921	0.9892	0.9941	0.9162	1.0
Presicion	0.9951	0.9921	0.9892	0.9941	0.9153	1.0
Recall	0.9951	0.9921	0.9892	0.9941	0.9162	1.0
F Score	0.9951	0.9921	0.9891	0.9941	0.9141	1.0
Process Time(minute)	00:04	00:04	00:04	00:04	00:04	00:00.09

When the results of Classification Models using Triple Data Class are analyzed, the best result is obtained with DTs algorithm.

The Confusion Matrix of this classification model using the DTs algorithm is given in the Table 5.42.

TABLE 5.42: Confusion Matrix for Triple Data Class with Test Dataset

	-1	0	1
-1	270	0	0
0	0	59	0
1	0	0	685

Chapter 6

Concluision and Future Work

Industrial systems, which do a lot of important work to raise people's living standards, are partially isolated environments, but like any electronic system, cyber attacks are open. Research shows that attacks on these systems are increasing day by day. With the increase of cyber attacks, the methods of attack have also begun to differentiate. So it has become increasingly difficult to detect these cyber attacks quickly. The detection speed of the cyber attacks on industrial systems, including critical infrastructures, must be very high.

In this study, a model was developed to quickly detect the cyber attacks on industrial systems. The proposed model is based on deep learning methods. The reason for choosing deep learning methods in the study is the high maturity levels of the algorithms and technologies used. Hence, these technologies have high level robustness, they are used in many commerical products.

In this study we used new published dataset created by the Mississippi State University and Oak Ridge National Laboratory. Our proposed model's classification performance are better than the orginal work's results. The original works results are; 90.4%, 93% and 99.1%, respectively.

In our study; We obtained 100%, 100% and 99.8% accuracy rates with binary, triple and multi labeled dataset respectively.

Our plan is to convert the proposed attack detection model with transfer learning method. Applying transfer learning model with autoencoder algorithm, the new developed model shall be have lower training time.

Bibliography

- [1] A. Kovacevic and D. Nikolic. Cyber attacks on critical infrastructure: Review and challenges (draft), 01 2015.
- [2] S. M. Rinaldi, J. P. Peerenboom, and T. K. Kelly. Identifying, understanding, and analyzing critical infrastructure interdependencies. *IEEE Control Systems*, 21(6): 11–25, Dec 2001. ISSN 1066-033X. doi: 10.1109/37.969131.
- [3] G. Dondossola, J. Szanto, M. Masera, and I. Nai Fovino. Effects of intentional threats to power substation control systems. 4:129–143, 01 2008.
- [4] D. Kushner. The real story of stuxnet. 50:48–53, 03 2013.
- [5] N. Falliere, L. O Murchu, and E. Chien. W32.stuxnet dossier. 02 2011.
- [6] G. Weimann and United States Institute of Peace. *Cyberterrorism: how real is the threat?* Number 31. c. in Special report. United States Institute of Peace, 2004. URL <https://books.google.com.tr/books?id=ozVRrXAzHA0C>.
- [7] The trends behind today’s breaches and cyber attacks, 2018. URL <https://www.fireeye.com/current-threats/annual-threat-report/mtrends.html>.
- [8] What is dwell time: A cybersecurity metric, Feb 2018. URL <https://www.armor.com/blog/dwell-time-cyber-security-metric/>.
- [9] T. H. Morris. Industrial control system (ics) cyber attack data set, 2014. URL http://www.ece.msstate.edu/wiki/index.php/ICS_Attack_Dataset.
- [10] T. H. Morris and W. Gao. Industrial control system cyber attacks. In *Proceedings of the 1st International Symposium on ICS & SCADA Cyber Security Research 2013*, ICS-CSR 2013, pages 22–29, UK, 2013. BCS. ISBN 978-1-780172-32-3. URL <http://dl.acm.org/citation.cfm?id=2735338.2735341>.
- [11] S. Pan, T. Morris, and U. Adhikari. Developing a hybrid intrusion detection system using data mining for power systems. *IEEE Transactions on Smart Grid*, 6(6): 3104–3113, Nov 2015. ISSN 1949-3053. doi: 10.1109/TSG.2015.2409775.

- [12] S. Pan, T. Morris, and U. Adhikari. Classification of disturbances and cyber-attacks in power systems using heterogeneous time-synchronized data. *IEEE Transactions on Industrial Informatics*, 11(3):650–662, June 2015. ISSN 1551-3203. doi: 10.1109/TII.2015.2420951.
- [13] S. Pan, T. H. Morris, and U. Adhikari. A specification-based intrusion detection framework for cyber-physical environment in electric power system. *I. J. Network Security*, 17:174–188, 2015.
- [14] R. C. Borges Hink, J. M. Beaver, M. A. Buckner, T. Morris, U. Adhikari, and S. Pan. Machine learning for power system disturbance and cyber-attack discrimination. In *2014 7th International Symposium on Resilient Control Systems (ISRCS)*, pages 1–8, Aug 2014. doi: 10.1109/ISRCS.2014.6900095.
- [15] K. Scarfone and P. Mell. *Intrusion Detection and Prevention Systems*, pages 177–192. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. ISBN 978-3-642-04117-4. doi: 10.1007/978-3-642-04117-4_9. URL https://doi.org/10.1007/978-3-642-04117-4_9.
- [16] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Mach. Learn.*, 29(2-3):131–163, November 1997. ISSN 0885-6125. doi: 10.1023/A:1007465528199. URL <https://doi.org/10.1023/A:1007465528199>.
- [17] D. Heckerman. A tutorial on learning with bayesian networks. Technical report, Learning in Graphical Models, 1996.
- [18] U. Adhikari, T. H. Morris, N. Dahal, S. Pan, R. L. King, N. H. Younan, and V. Madani. Development of power system test bed for data mining of synchrophasors data, cyber-attack and relay testing in rtds. In *2012 IEEE Power and Energy Society General Meeting*, pages 1–7, July 2012. doi: 10.1109/PESGM.2012.6345109.
- [19] P. Baldi. Autoencoders, unsupervised learning and deep architectures. In *Proceedings of the 2011 International Conference on Unsupervised and Transfer Learning Workshop - Volume 27*, UTLW’11, pages 37–50. JMLR.org, 2011. URL <http://dl.acm.org/citation.cfm?id=3045796.3045801>.
- [20] A. Makhzani, J. Shlens, N. Jaitly, and I. Goodfellow. Adversarial autoencoders. In *International Conference on Learning Representations*, 2016. URL <http://arxiv.org/abs/1511.05644>.
- [21] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P. Manzagol, and A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11:3371–3408, December 2010. ISSN 1532-4435. URL <http://dl.acm.org/citation.cfm?id=1756006.1953039>.

- [22] F. Chollet. Building autoencoders in keras, May 2016. URL <https://blog.keras.io/building-autoencoders-in-keras.html>.
- [23] L. Deng. A tutorial survey of architectures, algorithms, and applications for deep learning. *APSIPA Transactions on Signal and Information Processing*, 3:e2, 2014. doi: 10.1017/atsip.2013.9.
- [24] J. Han, M. Kamber, and J. Pei. Data mining concepts and techniques, third edition, 2012. URL http://www.amazon.de/Data-Mining-Concepts-Techniques-Management/dp/0123814790/ref=tmm_hrd_title_0?ie=UTF8&qid=1366039033&sr=1-1.
- [25] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. *SIGMOD Rec.*, 29(2):1–12, May 2000. ISSN 0163-5808. doi: 10.1145/335191.335372. URL <http://doi.acm.org/10.1145/335191.335372>.
- [26] R. Vidal, J. Bruna, R. Giryes, and S. Soatto. Mathematics of deep learning. *CoRR*, abs/1712.04741, 2017. URL <http://arxiv.org/abs/1712.04741>.
- [27] R. C. Holte. Very simple classification rules perform well on most commonly used datasets. *Mach. Learn.*, 11(1):63–90, April 1993. ISSN 0885-6125. doi: 10.1023/A:1022631118932. URL <http://dx.doi.org/10.1023/A:1022631118932>.
- [28] Wikipedia contributors. Autoencoder — Wikipedia, the free encyclopedia, 2018. URL <https://en.wikipedia.org/w/index.php?title=Autoencoder&oldid=849677574>. [Online; accessed 13-July-2018].
- [29] Wikipedia contributors. Activation function — Wikipedia, the free encyclopedia, 2018. URL https://en.wikipedia.org/w/index.php?title=Activation_function&oldid=849872393. [Online; accessed 13-July-2018].
- [30] G. Rossum. Python reference manual. Technical report, Amsterdam, The Netherlands, The Netherlands, 1995.
- [31] G. Rossum. Python tutorial. Technical report, Amsterdam, The Netherlands, The Netherlands, 1995.
- [32] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL <https://www.tensorflow.org/>. Software available from tensorflow.org.

- [33] F. Chollet et al. Keras. <https://keras.io>, 2015.
- [34] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [35] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.
- [36] S. Walt, S. C. Colbert, and G. Varoquaux. The numpy array: A structure for efficient numerical computation. *Computing in Science and Engg.*, 13(2):22–30, March 2011. ISSN 1521-9615. doi: 10.1109/MCSE.2011.37. URL <https://doi.org/10.1109/MCSE.2011.37>.
- [37] E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. URL <http://www.scipy.org/>. [Online; accessed <today>].
- [38] W. McKinney. pandas: a foundational python library for data analysis and statistics.
- [39] W. McKinney. Data structures for statistical computing in python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 51 – 56, 2010.