Study on the Generation of Pseudo-Random Numbers using Recurrent Neural Network

DISSERTATION SUBMITTED IN PARTIAL FULFILMENT OF THE

Master of Technology in Cryptology and Security

from

Indian Statistical Institute

by

Ankit Ravish

(Roll: CRS-1915)

under the guidance of

Dr. Shravani Shahapure

Co-Ordinator – Cybersecurity Research,
Data Security Council of India, Noida

Institute supervisor

Dr. Rajat K. De

Professor
Indian Statistical Institute, Kolkata

<u>CERTIFICATE</u>

This is certify that the dissertation entitled **"Study on the generation of Pseudo-Random Numbers using Recurrent Neural Network",** submitted by **Ankit Ravish** to Indian Statistical Institute, Kolkata, in partial fulfilment for the award of the degree of **Master of Technology in Cryptology and Security** is a bonafide record of work carried out by him under my supervision and guidance.

……………………………………….

Dr. Shravani Shahapure
Co-Ordinator – Cybersecurity Research,
Data Security Council of India, Noida

Acknowledgments

Apart from my efforts, the success of the completion of this dissertation largely depends on the encouragement and guidance of many others. I take this opportunity to express my gratitude to the people who have been instrumental in the completion of this project.

I would first like to express my profound gratitude to my advisor, **Dr. Shravani Shahapure**, Co-Ordinator – Cybersecurity Research, Data Security Council of India, Noida, for her patient guidance, enthusiastic encouragement, and useful critiques needed for this research work. Her advice and knowledge helped me in pursuing good research and writing of this thesis. I am privileged to have such a great supervisor as she consistently supported me both academically and personally.

I would also like to thank **Dr. Virendra R Sule**, Professor, Indian Institute of Technology, Bombay, for his valuable advice and assistance in keeping my progress on schedule.

My grateful thanks are also extended to my institute supervisor **Dr. Rajat K. De**, Professor, Indian Statistical Institute, Kolkata, who have paved the way towards my research acumen.

Finally, I thank my parents who introduced me to the first beam of knowledge and supported me in all my endeavors. I would also like to thank my friends for keeping me motivated. Last but not the least, I would also like to thank all the members and staffs of Data Security Council of India, Noida, for their help and support at the toughest period of this pandemic situation.

Ankit Ravish
Indian Statistical Institute,
Kolkata–700108, India

Abstract

Generating random numbers is an important task in cryptography as well as computer science. Pseudo-randomness is fundamental to cryptography and it is required for achieving any cryptographic function like encryption, authentication and identification. The quality of a pseudo-random number generators is determined by the randomness which is generated in sequences. In this dissertation, pseudo-random numbers have been generated using recurrent neural network. Initially the neural networks are trained with the sequences of random numbers. These random numbers which will be used as training set for the neural networks model has generated Advanced Encryption Scheme (AES) using counter mode of operations. It is known that the padding vectors which are generated in each operation is random. They are considered to be random as the same sequence occurs after a long interval of time. The neural network which has been considered in this study is Recurrent Neural Network due to its property that it has an internal memory that allows the neural network to remember the historic input and helps it in making decisions by considering current input alongside learning from previous input. The neural networks have been trained using different loss functions. After fitting the model according to the training set, a sequence of predicted values has been obtained from each model. The randomness of these predicted values are checked using NIST test. Lastly, the aim is to compare and show the number of NIST tests passed by the predicted sequences in each model.

Keywords: *Random Number, AES, Neural Network, RNN, NIST*

Contents

List of Figures

Chapter – 1

Introduction

1.1 Random Numbers and Pseudo-Random Numbers

Random Numbers are those which occurs in a sequence. The values are distributed over the defined set or interval in such a way that it is difficult to determine the future values based on present or past values. In other words, we can say that random numbers are chosen as if accidentally from some specified distribution such selection of an out-sized set of those numbers reproduces the underlying distribution nearly always. In order to no correlations between successive numbers then such numbers are required to be independent.

Random numbers which are generated by computers are sometimes called pseudo-random numbers. The term "random" refers to the output of unpredictable physical processes. In general, the word "random" usually means random with a consistent distribution, other distributions are in fact possible.

It is impossible to supply an unsystematically long string of random digits and prove to be random. Remarkably, it's very difficult for humans to supply a string of random digits. Computer programs are often written which, on the average, to predict a number of the digits'.

Random numbers are very important while computing. TCP/IP sequence numbers, password salts, PIN generation, prime number generation and DNS source port numbers all depends upon random numbers.

1.2 Pseudo-Random Number Generator

Random numbers are common if they are taken individually. The importance of random numbers is not only in the number itself but also in the way they are generated.

A pseudo-random number refers to an algorithm which uses mathematical formulas to produce the sequences of random numbers. A pseudo-random number generator produces the output which is a longer bit sequence that appears as random by taking short random seeds. The output should be computationally indistinguishable from a random string to be cryptographically secure i.e., if short prefix of the pattern or sequence is provided, it would be computationally impracticable to predict rest of the pattern or sequence without knowing the seed. In many cryptographic applications, there is a need for random and pseudo-random numbers. For example, common cryptosystems use keys which must be generated in a random fashion. Many cryptographic protocols also require random or pseudo-random inputs at various points, e.g., for auxiliary quantities used in generating digital signatures, or for generating challenges in authentication protocols.

Pseudo random number generator generates artificial random bits from a few true random bits in such a way that it is both efficient and reliable. For example, if the user stops interacting with the random numbers which are generated through the movements of mouse or pressing the keys of keyword then, it would stop working. But, PRNG would continue producing random numbers using the bits of initial entropy.

Ironically, pseudo-random numbers appear to be more random than that of random numbers obtained from the physical sources. Each value in the sequence is produced from the previous value via transformations which appears to introduce additional randomness in the sequence, if a pseudo-random sequence is generated properly. The series of such transformations can eliminate statistical auto-correlations between input and output. Thus, the outputs of a pseudo-random number generator have better statistical properties and produce faster than the random number generator.

PRNG maintains a memory buffer which is referred as entropy pool. In this, the bytes which are received from these entropy sources (RNG) are stored. To get rid of statistical biases within the entropy data, the PRNG often mixes with the entropy pool bytes. Random bits are generated using a deterministic random bit generator (DRBG) on the entropy pool data bits. This algorithm is deterministic as it gives the same output for the same input.

Simple random generator is used to provide security to the system. The reason for this is that system developers or designers focus towards power consumption and speed of bit generation rather than the randomness within the generated bits.

It is strange to see that in most of the cases, especially in cryptographic systems, the security strength of the systems directly depends on the quality of the random numbers. In other words, the quality or the standard of the random number generator directly determines how difficult it is to attack the system.

Modern technology is based on these numbers like communication protocols, cryptography, games do heavily usage of them and their whole security and unpredictability depend on them. Modern cryptographic algorithms and protocols are designed using the principle of Kerckhoff - "The security of the system must depend solely on the key material not on the design of the system"

This means that the cryptographic strength of all the modern security algorithms and protocols depends on the number of bits or keys that an attacker needs to know. If the attacker wants to break the system, then the strength assumes that the attacker does not have any prior knowledge related to the bits used in the first or original key. The 'effective strength' of an algorithm gets reduced if better attacks are found against it and more key bits can be derived from looking at the output of data.

1.3 Neural Networks

Neural networks, also referred as artificial neural networks (ANNs) or simulated neural networks (SNNs). They are the subset of machine learning and heart of deep learning algorithms. The name and structure of neural networks are inspired and taken from the human brain, duplicating the way of biological neurons signal with other neurons.

Artificial neural networks consist of node layers which embraces input layers, hidden layers, and output layers. Each node is connected with other and has a weight associated with it. If the output of any particular node is above the specified threshold value, that node is activated, sending data to the subsequent layer of the network, else no data is passed along to subsequent layer of the network.

1.3.1 Recurrent Neural Network

Recurrent Neural Networks, or RNNs, are special type of neural networks that is used to process sequential data. Sequential data can be assumed as a series of data or data points. There are some examples of sequential data. Such as - video as it is composed of a sequence of video frames; music as it is a combination of a sequence of sound elements; and text as it arises from a combination of letters. Modeling sequential data requires abiding the data learned from the previous instances. For example, if someone predicts the next argument during a debate then, he must consider the previous argument put forth by the members involved in that debate. One person forms their argument such that it is in line with the debate flow. Likewise, an RNN learns and remembers the data to formulate a decision which is dependent on the previous learning.

Unlike a typical neural network, an RNN neither cap the input or output as a set of fixed-sized vectors nor fixes the amount of computational steps required to train a model. Instead of this, it allows to train the model with a sequence of vectors (sequential data).

Interestingly, an RNN maintains persistence of model parameters throughout the network. It implements Parameter Sharing to accommodate varying lengths of the sequential data. If we consider separate parameters for varying data chunks, neither would it be possible to generalize the data values across the series, nor it would be computationally feasible. Generalization is with respect to repetition of values in a series. Thus, rather than starting from scratch at every learning point, an RNN passes learned information to the following levels.

To enable parameter sharing and information persistence, an RNN makes use of loops.

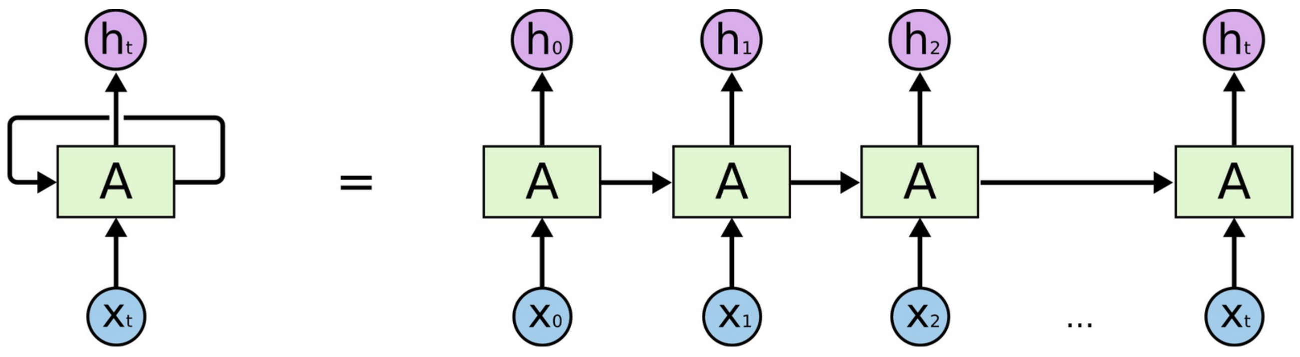

Figure 1.1 : Unfolding RNN

A neural network ‘A’ is repeated multiple times, where each chunk accepts an input x_i and gives an output h_t. The loop here passes the information from one step to another.

As a matter of fact, an incredible number of applications are text generation, image captioning, speech recognition, and more are using RNNs and their variant networks.

1.3.2 LSTM

Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture which is used in the field of deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. It not only process single data points (such as images), but also entire sequences of data (such as speech or video).

LSTM ’s was created as the solution to short-term memory. They have internal mechanisms called gates that can be used to regulate the flow of information.

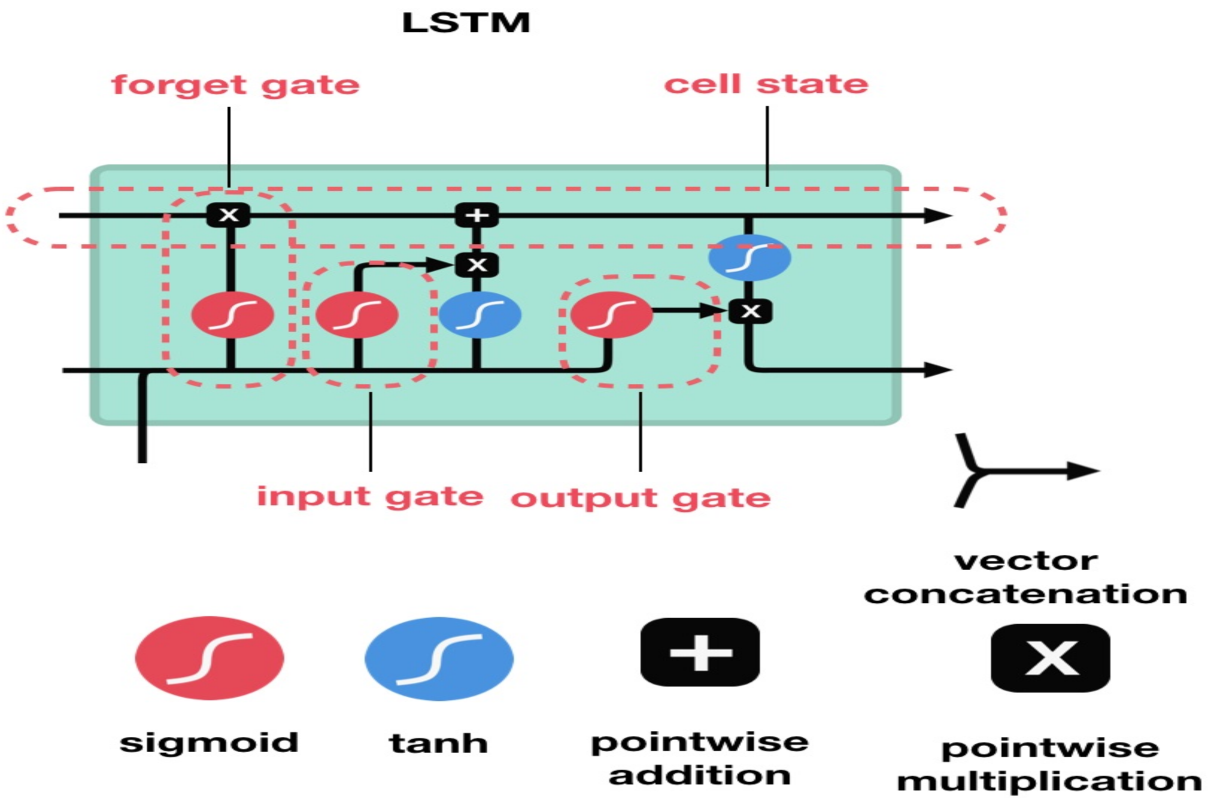

Figure 1.2 : LSTM

These gates can learn that which data in a sequence is important to keep or throw away. By doing that, it passes relevant information down the long chain of sequences to make predictions. Almost all state of the art results is based on recurrent neural networks which are achieved with these two networks.

Since, there can be lags of unknown duration between important events in a time series, LSTM networks are well-suited for classifying, processing and making predictions based on time series data. LSTMs were developed to deal with the faded gradient problem that can be encountered when training traditional RNNs. Relative insensitivity to gap length is an advantage of LSTM over RNNs.

1.4 NIST Test

Random numbers and random sequences are used to produce indispensable parts of cryptographic algorithm. In order to check the statistical proficiencies of the random numbers, various test suites have been designed like Crypt-X, Diehard, FIPS and NIST. In the recent years, NIST test suite has been widely accepted throughout the world in testing the big random number streams as a valid standard. NIST test suite was published by National Institute of Standards & Technology and is popularly known by NIST SP 800-22

NIST SP800-22 consists of 16 kinds of statistical tests and provides them to determine whether given sequences are random or not for each statistical test. Some of these tests are explained briefly:

- Freqency (Monobit) Test: This test determines proportion of 0 and 1's in the entire sequence.
- Frequency Test within a Block: This test determines proportion of 1's within m-bit blocks.
- Run Test: This test determines total number of runs in the sequence (uninterrupted sequence of identical bits).
- Test for the Longest Run of Ones in a Block: This test determines the longest run of 1's in the sequence. It focuses on the longest group of 1 in m-bit blocks.
- Binary Matrix Rank Test: This test is performed for checking the linear dependence of the fixed length substrings of the sequence.

- Discrete Fourier Transform Test: This test is performed to detect repetitive patterns in the sequence.
- Non-overlapping Template Matching Test: This test is performed to detect whether m-bit block repeats in sequence. If so, new m-bit blocks will be formed from the block that is repeated.
- Overlapping Template Matching Test: This test is performed to detect whether m-bit block repeats in the sequence. If so, the block is shifted by 1 bit and new block is formed.
- Maurer's Universal Statistical Test: This test is performed to check whether the sequence can be compressed without loss of data.
- Linear Complexity Test: This test is performed to check whether the sequence are complex enough to be considered as random.
- Serial Test: This test determines the number of occurrences of the 2^m m-bit overlapping patterns.
- Approximate Entropy Test: This test is performed to compare the frequency of two repeating blocks in consecutive lengths (m and m+1) with the expected frequency of a random sequence.
- Cumulative Sums Test: This test determines whether cumulative sum of partial subsequences in the tested sequence is too large or too small compared to the expected value of a random sequence.

1.5 Advanced Encryption Standard (AES)

AES stands for Advanced Encryption Standard which is majorly used in symmetric encryption algorithm. It is a fast and secure form of encryption that keeps intrusive eyes away from the data which is based on ‘substitution–permutation network’. It comprises of a series of linked operations, which involve replacing inputs by specific outputs (substitutions) and shuffling bits around (permutations).

There are various modes of operation of AES. Some of them are explained briefly:

- Electronic Code Block (ECB): It is the simplest of all the encryption modes. The message is divided into blocks, and each block is encrypted separately.

- Cipher Block Chain (CBC): In CBC mode, each block of plaintext is XORed with the previous ciphertext block before being encrypted. Hence, each ciphertext block depends on all plaintext blocks processed up to that point. To make each message unique, an initialization vector must be used in the first block.
- Output feedback (OFB): mode makes a block cipher into a synchronous stream cipher. It generates keystream blocks, which are then XORed with the plaintext blocks to get the ciphertext. Each output feedback block cipher operation depends on all previous ones, and so cannot be performed in parallel. However, because the plaintext or ciphertext is only used for the final XOR, the block cipher operations may be performed in advance, allowing the final step to be performed in parallel once the plaintext or ciphertext is available.
- Counter (CTR): Like OFB, counter mode turns a block cipher into a stream cipher. It generates the next keystream block by encrypting successive values of a "counter". The counter can be any function which produces a sequence which is guaranteed not to repeat for a long time, although an actual increment-by-one counter is the simplest and most popular. Presently, CTR mode is widely accepted.

Chapter – 2

Literature Survey

Earlier some researchers have generated the random sequences through machine learning. Simplified version of recurrent network was introduced by Elman. The Elman network [8] has only two layers. It uses a tansig transfer function for the hidden layer and a purelin transfer function for the output layer. It is trained using an approximation to the back propagation algorithm. Neural networks are used for predicting the output of the next sequence. The authors of [2] have generated the pseudo-random numbers using layer recurrent networks and used trainscg, trainbfg, trainlm function to train the network. The authors of [3] have generated the numbers using the dynamics of a feed forward neural network with random orthogonal weight matrices while in [4], the authors have used Neuronal plasticity. In [5], a generative adversarial network approach to the task has been presented which exploits the input source of randomness. The author of [6] have done analysis on the generation of pseudo-random sequences through Dynamic neural filters (DNFs).

The statistical proficiency of the random sequence can be checked using NIST test suite. All the test of this suite are described in [1]. The author of [12] have studied the pass rate of the NIST SP800-22 statistical test suite for the ideally true random sequences after analyzing the simulation of statistical tests.

As from the previous researches, it is clear that many different approaches have been made to train the network so that it can produce random sequence. Thus, we decided to analyze the generation of random sequence by the neural network when the network is trained via the pseudo-random number generation using AES algorithm in counter mode of operations.

Chapter-3

Implementation

3.1 Generation of Data Set

The first step is to generate the data set i.e, to generate a sequence of random numbers which will be used by the neural network for training and testing. For the generation of these random numbers, lots of methods are available but here the data set has been generated using Advanced Encryption Standard (AES) algorithm in counter mode of operation. Counter (CTR) mode is one of the popular AES block cipher mode in which all the steps are done in parallel. This method involves XORing of sequence of pad vectors with the plaintext or ciphertext.

There is need of initial vector in the CTR and then the pad vectors are obtained using the formula:

$$V_i = E_k(s+i-1)$$

where E_k denotes the block encrpytion algorithm using key k, V_i is a pad vector and i is the vector's offset which starts from 1.

For this study, random sequences have been generated using AES-CTR mode in python language. For generating the sequences some inbuilt libraries/functions like pyaes, pbkdf2, binascii, pandas etc have been used. The binascii module has been used as it contains various methods to convert between binary and various ASCII-encoded binary representations. The pandas is an open-source and has been used as it provides high performance and easy to use data structures and data analysis tools for programming in python. The pbkdf2 module has been used as it provides cryptographic key derivation function which are resistant to some attacks. The pyaes is python implementation of AES block cipher algorithms with different modes of operation.

The pad vectors which are generated each time appears to be random as even with the slight change in initialization vector, the output i.e, pad vectors changes a lot. The initialization vector is increased by 1 each time and the pad vectors are noted. In each round, it gives output of 128 bits. We have generated two different data sets for our study- one with 10 thousand initialization vectors which produces total 12,80,000 bits and the other with 1 lakh initialization vectors which generated 1,28,00,000 bits.

3.2 NIST Test of Data Set

Once the data has been generated, they are passed through the NIST test suite to check their statistical properties. The result of the NIST test suite has been noted and it is explained in detail in the next chapter.

3.3 Neural Network Model

To design the neural network model, LSTM (Long Short-Term Memory) networks have been used since it is capable of learning order dependence in sequence predictions problems. Five different models of neural network has been build for the study.

The first type of model which we named it as Type-A model takes 16 bits as input and generates 16 bits of output at a time. In this model, we give first 16 bits of input which generates next sequences of 16 bits output. Now, this output is fed as the input to the network for generating the next sequences of 128 bits. In this way, input-output sequences are prepared to be used for training or testing data set.

The second type of model which we named it as Type-B model takes 32 bits as input and gives output of 32 bits. The first 32 bits of input which generates next sequences of 32 bits output and these bits are fed as input to the network for generating next sequence.

Similarly, the other two models namely Type-C and Type-D takes 64 bit and 128 bit as input and gives 64 bit and 128 bit output respectively.

Type-D has been model has been trained with two different data sets. The model which has been trained with 12.8 lakhs bit sequences is named as Type-D1 model and the one which is trained with 128 lakhs bit sequences has been named as Type-D2.

All the models have been implemented different loss functions to check which loss function works better. The activation function has been selected corresponding to the loss function.

The table given below shows the loss function and their corresponding activation function:

Loss Function	**Activation Function**
Binary Crossentropy	Sigmoid
Squared Hinge	Tanh
Mean Absolute Error	ReLU
Mean Squared Error	ReLU

Thus in total we have 20 different models – each model with 4 loss functions.

3.3.1 Training and testing data for RNN

- For all the models, we have set 70% of the data set for training and the remaining 30% for the testing.
- Type-A, Type-B, Type-C and Type-D1 have total 7000 training data and 3000 testing data while Type-D2 has 70000 and 30000 as training and testing data respectively.

3.3.2 Predicted data by RNN

Once the model has been compiled and fitted corresponding to the training set, we will predict the random sequences based on the learning of the model.

3.4 NIST Test of Predicted data set

After obtaining the predicted sequences through RNN, those data will be pass through the NIST test suite to check the randomness.

The observation of all these models will be discussed in next chapter.

Chapter – 4

Observations

4.1 AES Generated Sequence

When the generated random sequences by AES is passed through NIST test suite then the data set having 12800000 bits passed all the tests except Universal test case while when the data set having 1280000 bits are tested, it does not passes Universal, ApproximateEntropy, RandomExcursions and RandomExcursionsVariant tests.

In case of former data set, following results are obtained:

The minimum pass rate for each statistical test with the exception of the random excursion (variant) test is approximately = 96 for a sample size = 100 binary sequences.

The minimum pass rate for the random excursion (variant) test is approximately = 13 for a sample size = 15 binary sequences.

While in the later case, following results are obtained:

The minimum pass rate for each statistical test with the exception of the random excursion (variant) test is approximately = 96 for a sample size = 100 binary sequences.

The minimum pass rate for the random excursion (variant) test is undefined.

4.2 Accuracy vs Epochs

The analysis have been performed on 20 different models. For all the models, we have plotted accuracy vs epoch graph to study the accuracy and binary accuracy corresponding to each epochs. The graph of each model has been shown below.

4.2.1 Type-A model

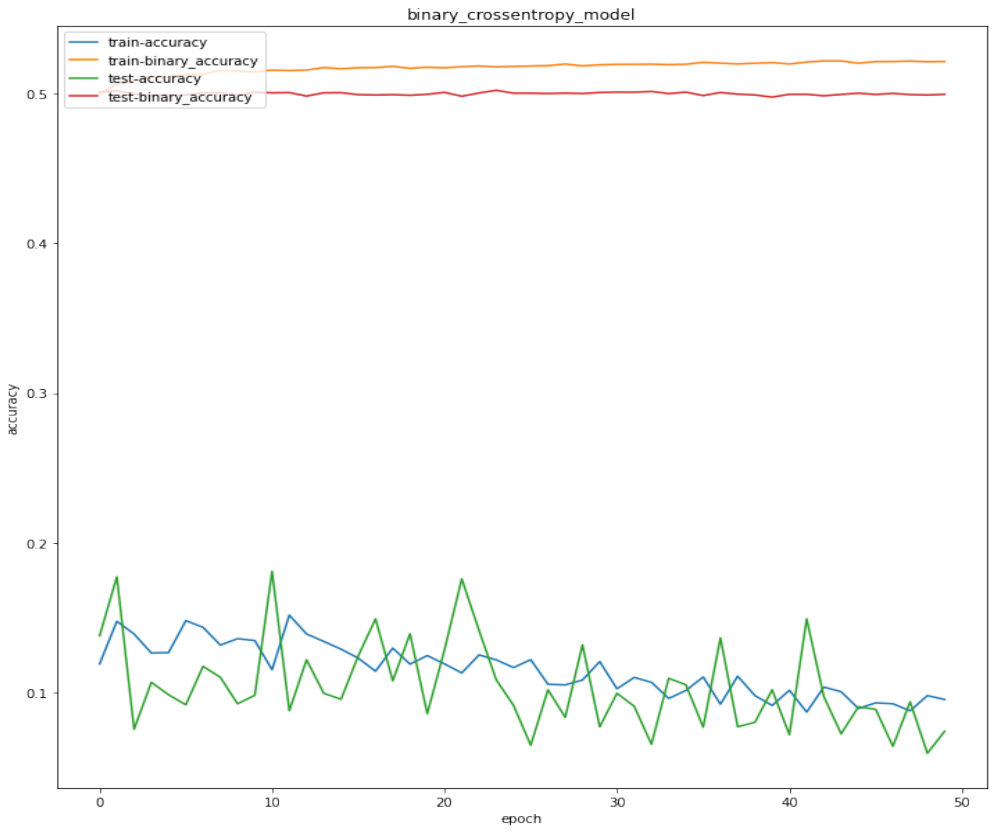

Figure 4.1 : Type-A – Binary Crossentropy Loss

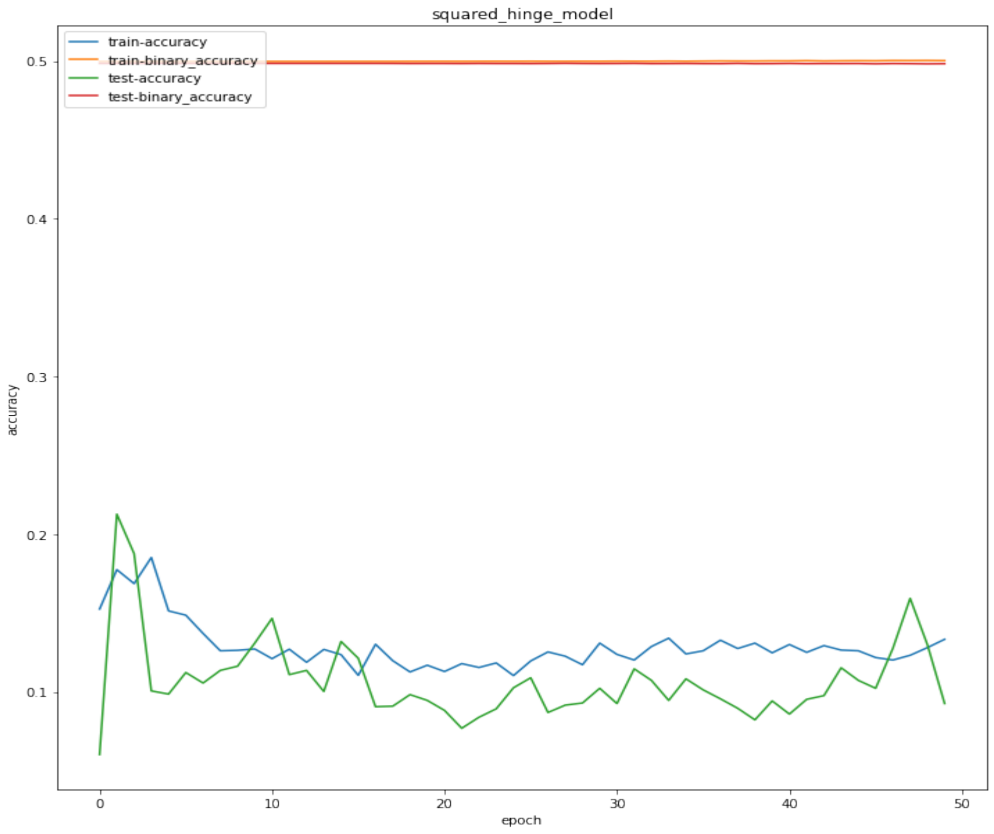

Figure 4.2 : Type-A – Squared Hinge Loss

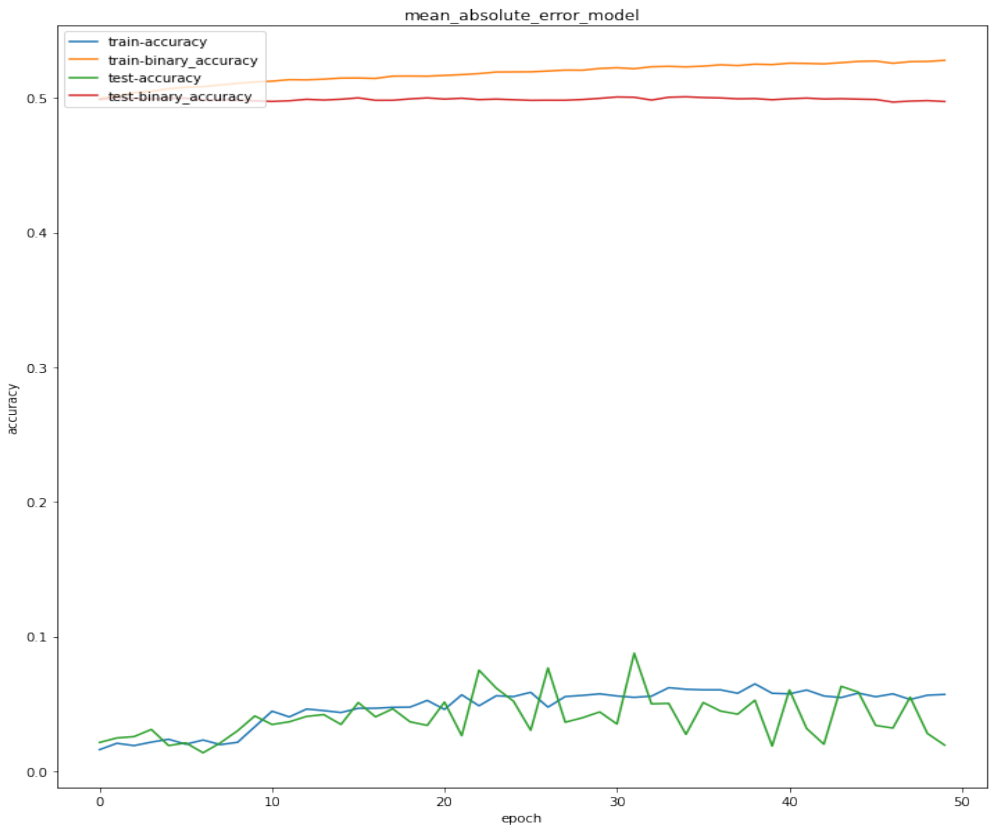

Figure 4.3 : Type-A – Mean Absolute Error Loss

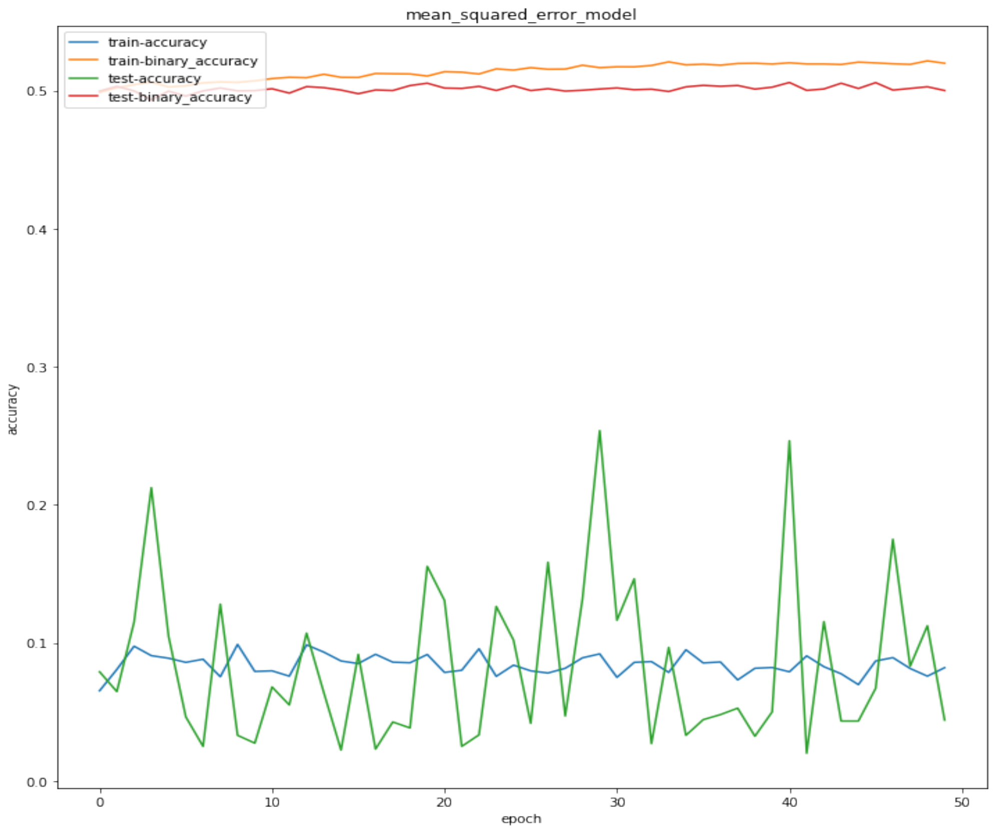

Figure 4.4 : Type-A Mean Squraed Error

4.2.2 Type-B model

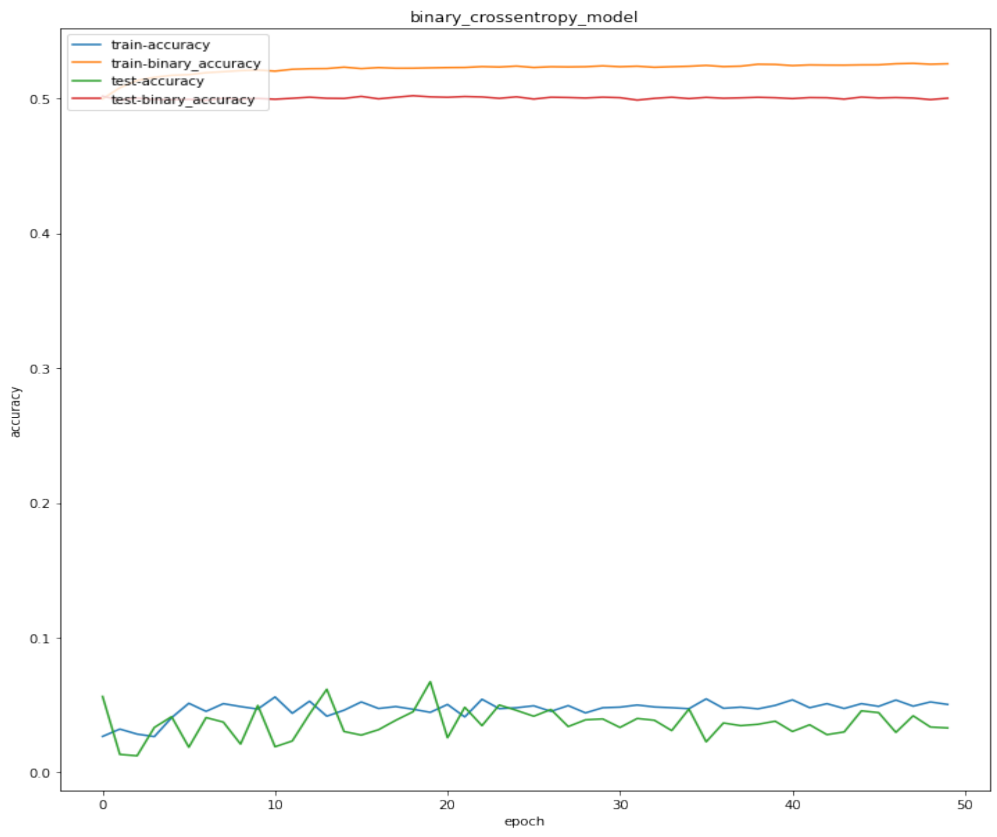

Figure 4.5 : Type-B – Binary Crossentropy Loss

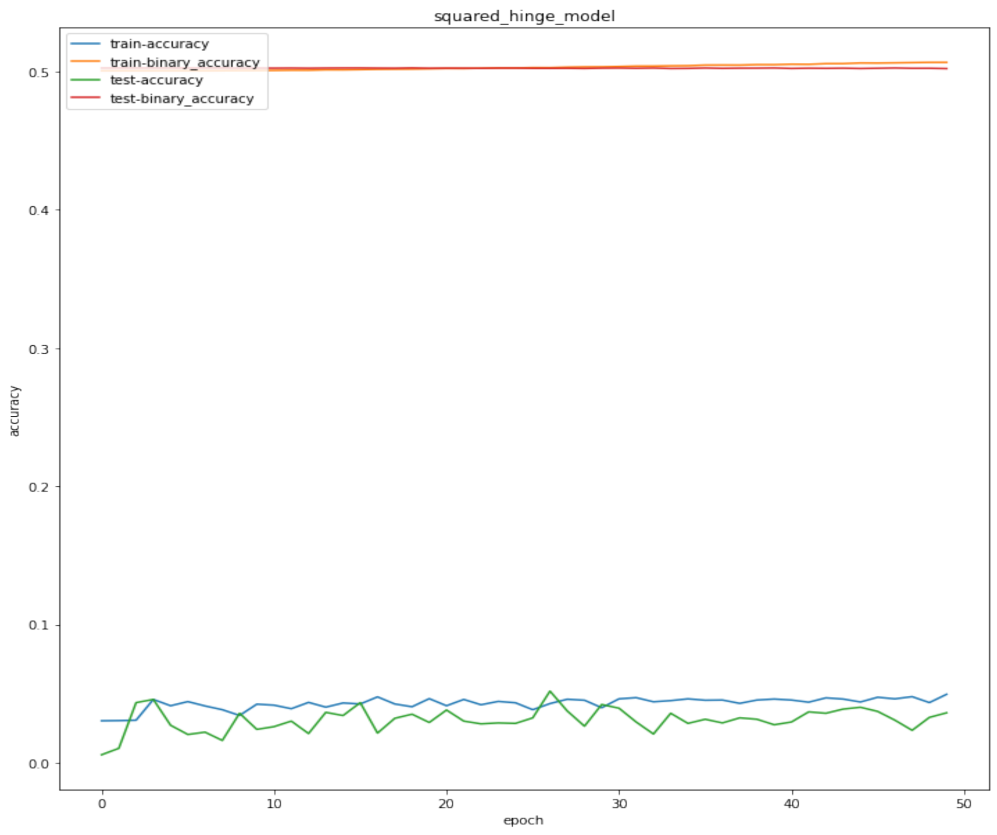

Figure 4.6 : Type-B – Squared Hinge Loss

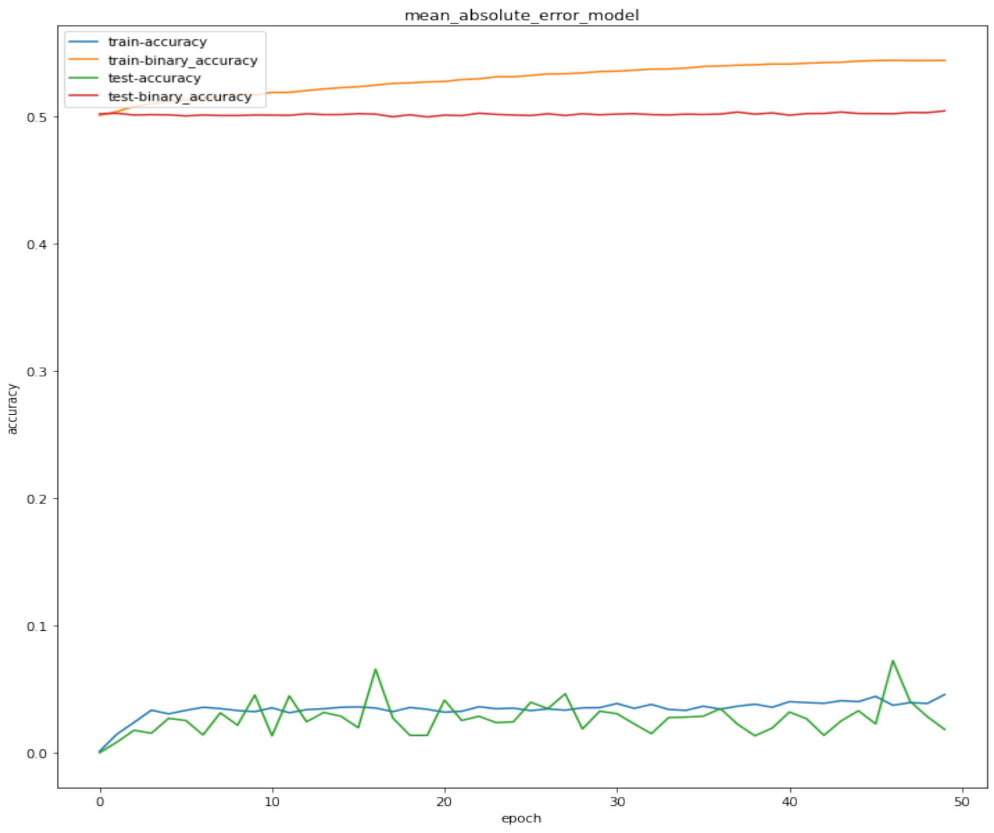

Figure 4.7 : Type-B – Mean Absolute Error

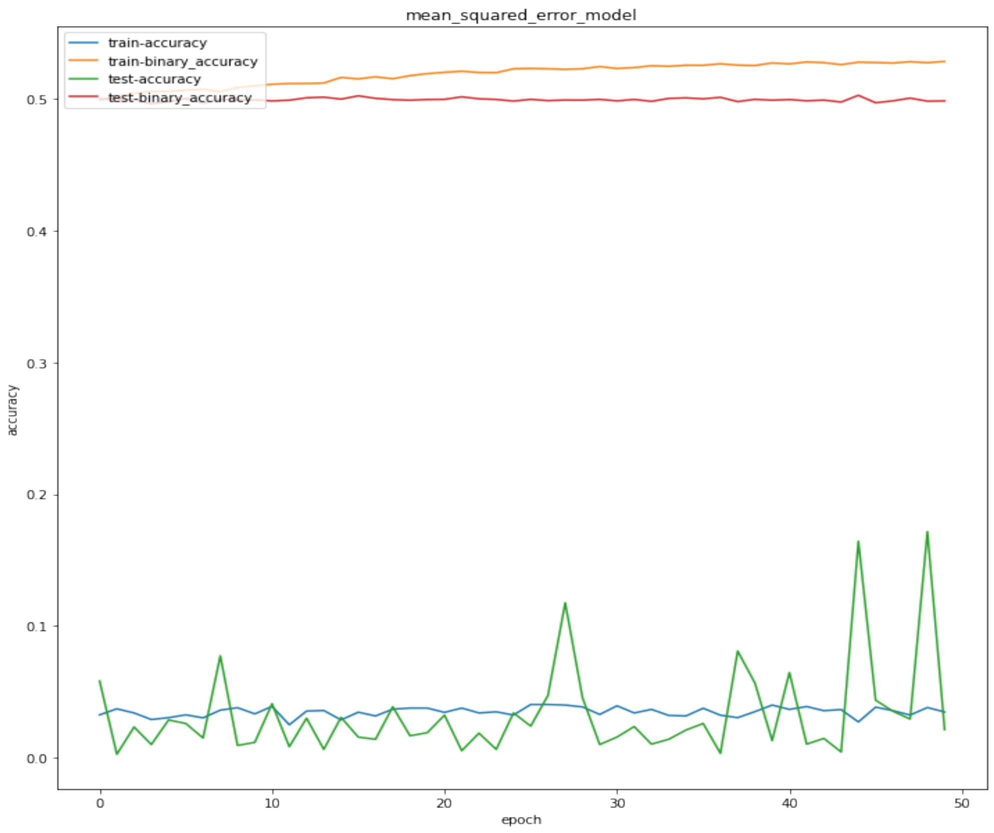

Figure 4.8 : Type-B Mean Squraed Error

4.2.3 Type-C model

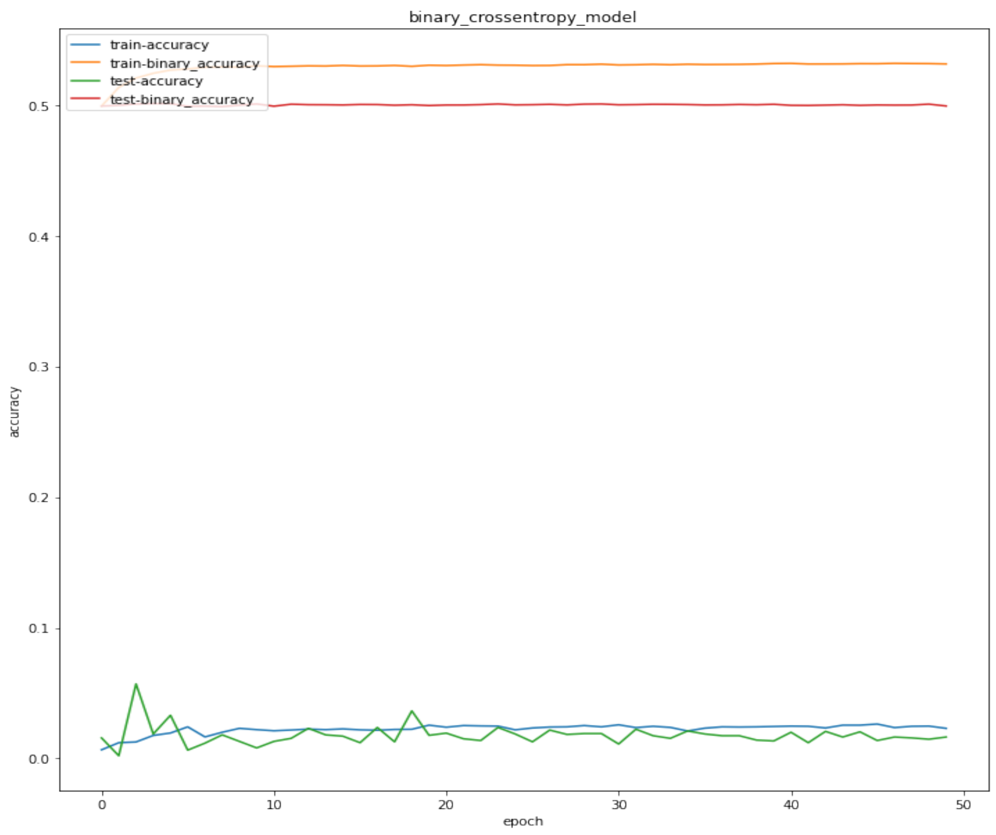

Figure 4.9 : Type-C – Binary Crossentropy Loss

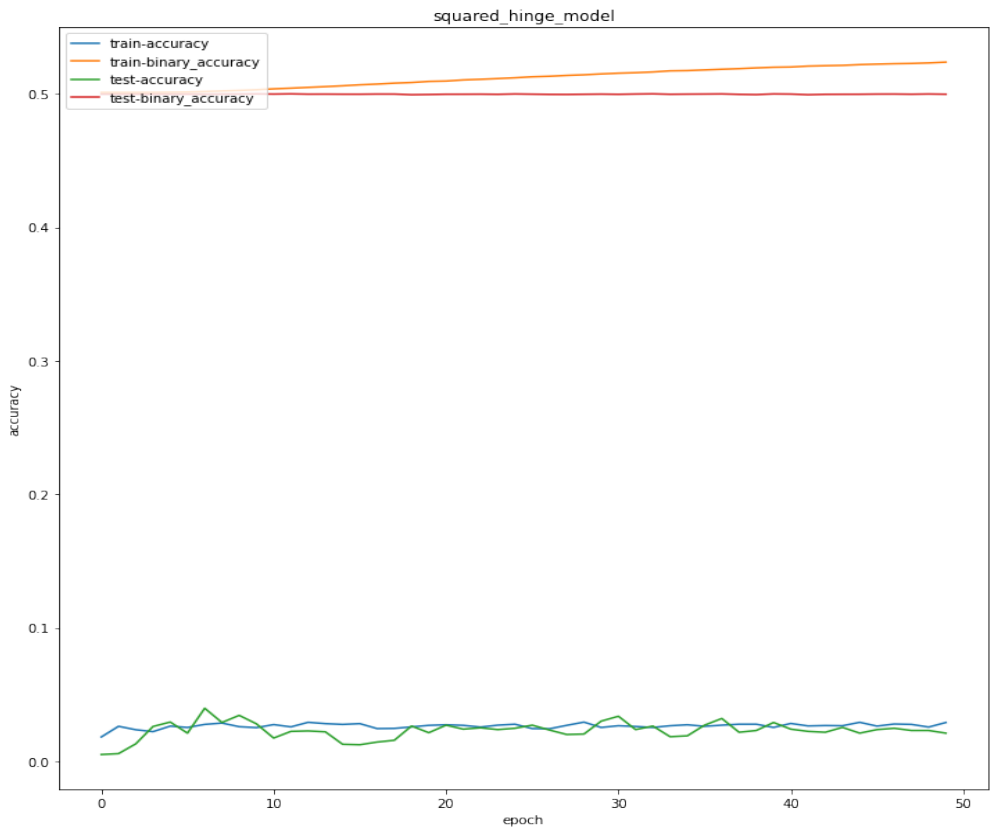

Figure 4.10 : Type-C – Squared Hinge Loss

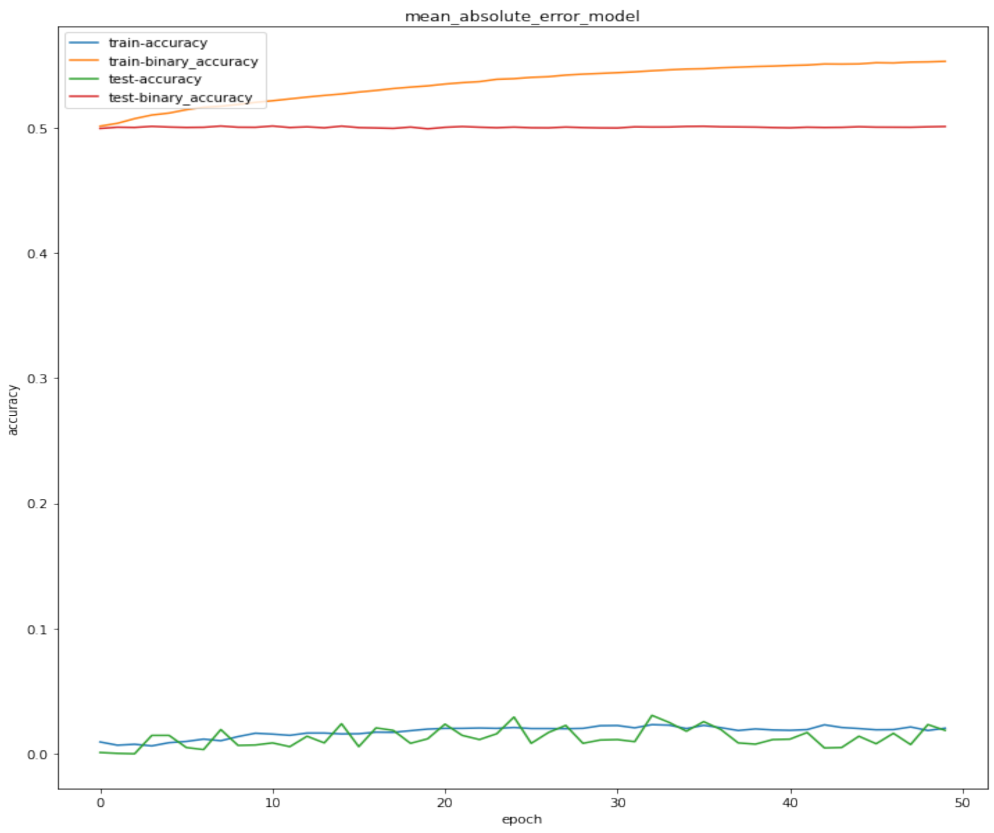

Figure 4.11 : Type-C – Mean Absolute Error

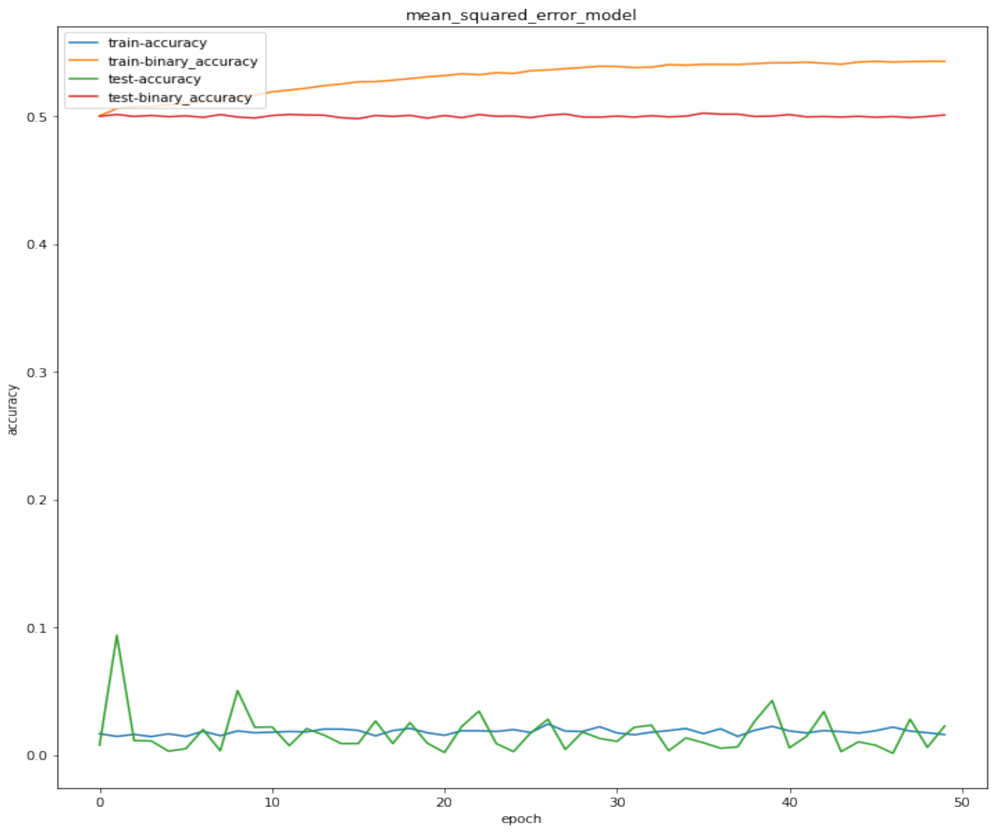

Figure 4.12 : Type-C Mean Squraed Error

4.2.4 Type-D1 model

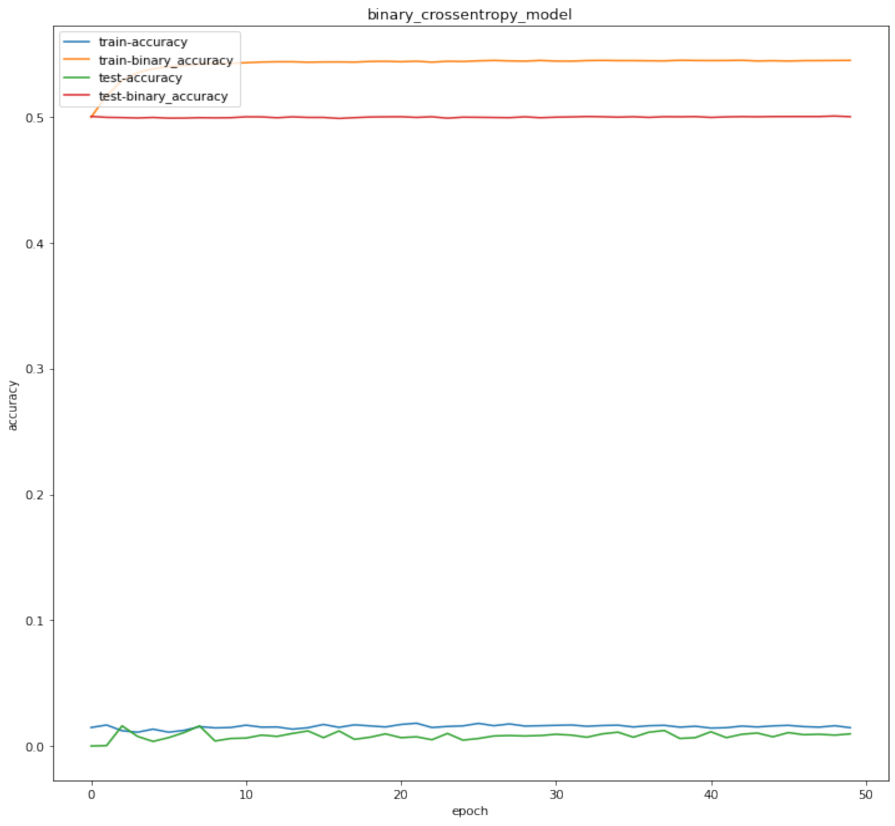

Figure 4.13 : Type-D1 – Binary Crossentropy Loss

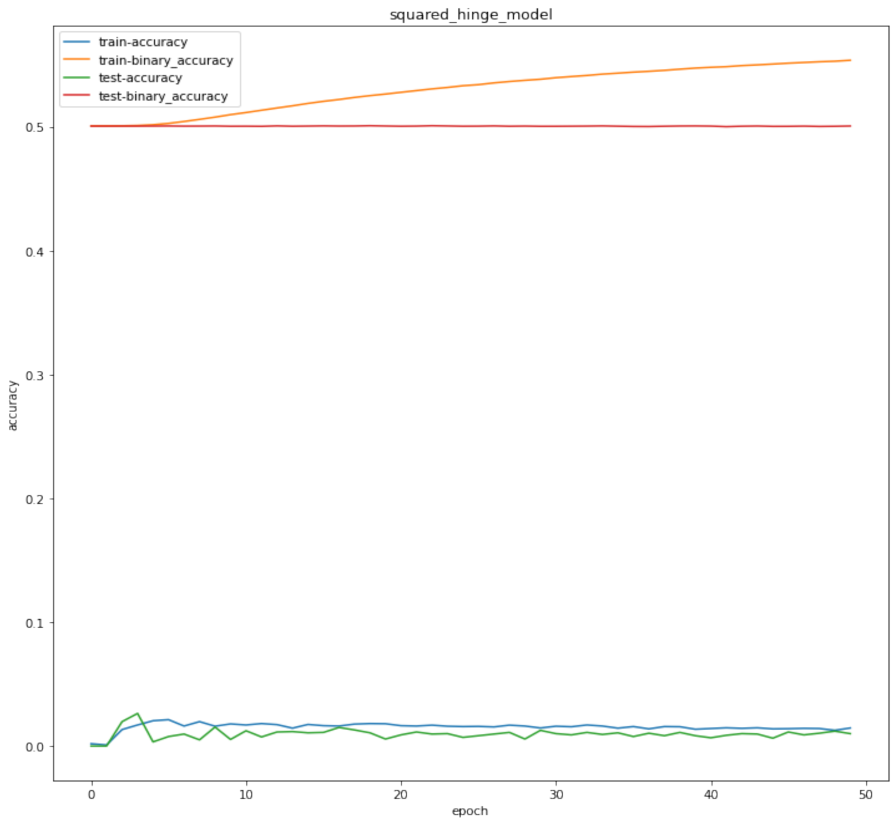

Figure 4.14 : Type-D1 – Squared Hinge Loss

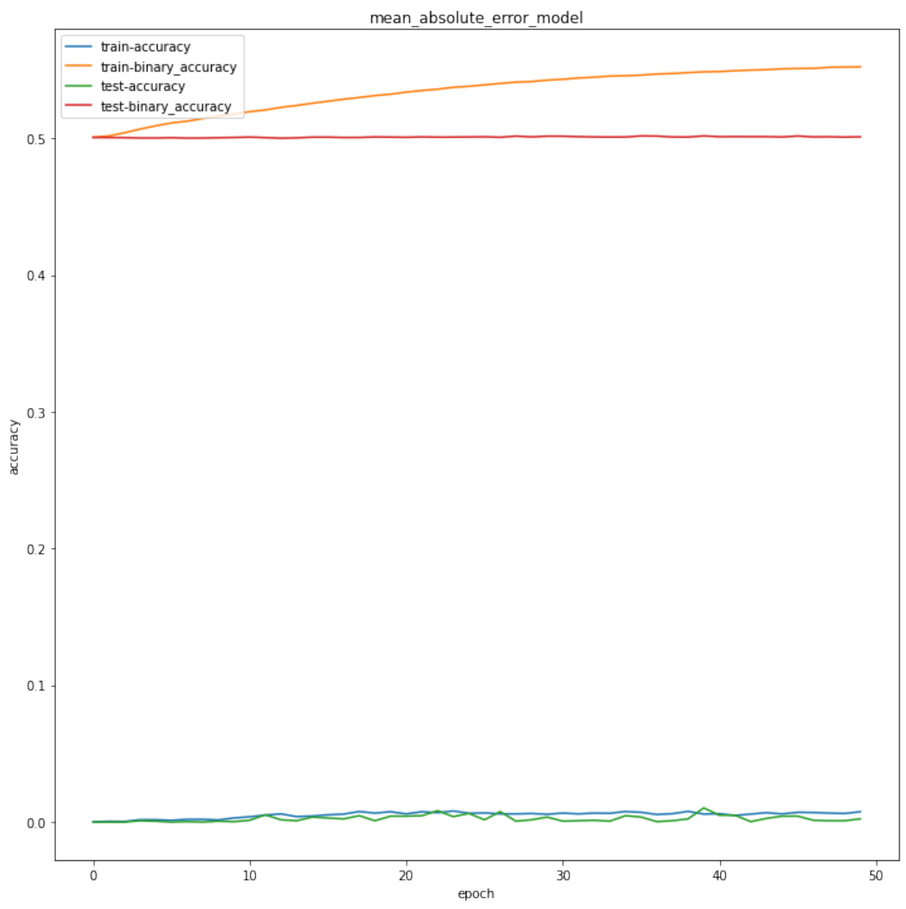

Figure 4.15 : Type-D1 – Mean Absolute Error

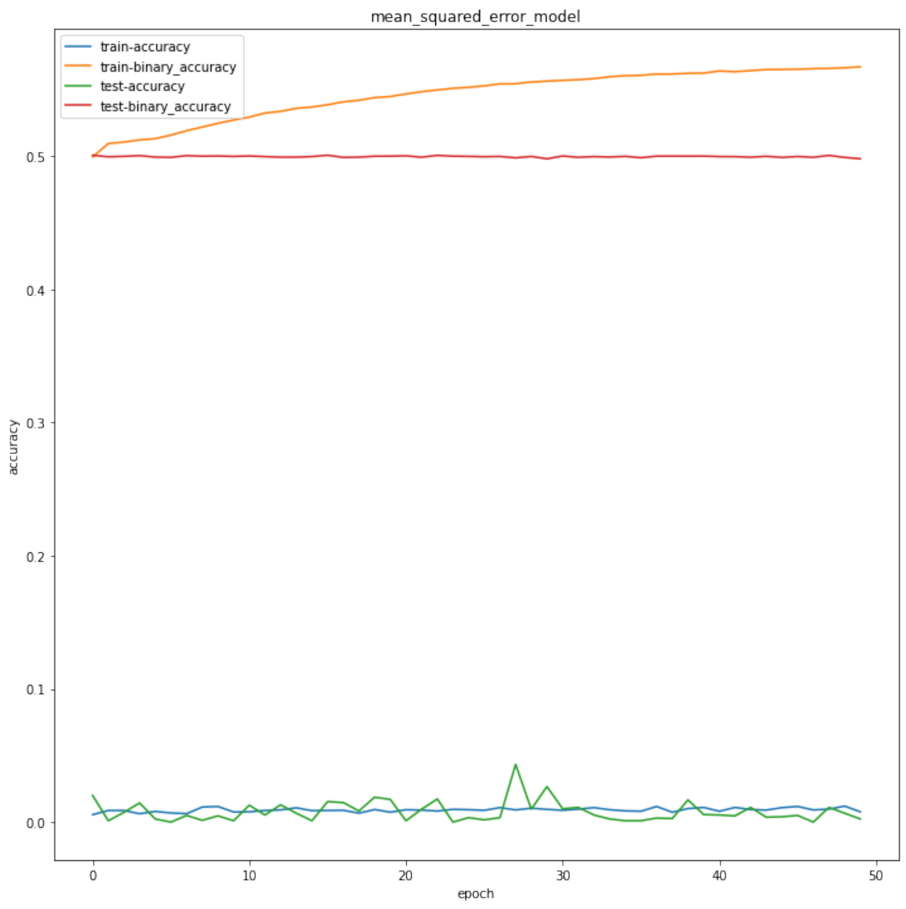

Figure 4.16 : Type-D1 - Mean Squared Error

4.2.5 Type-D2 model

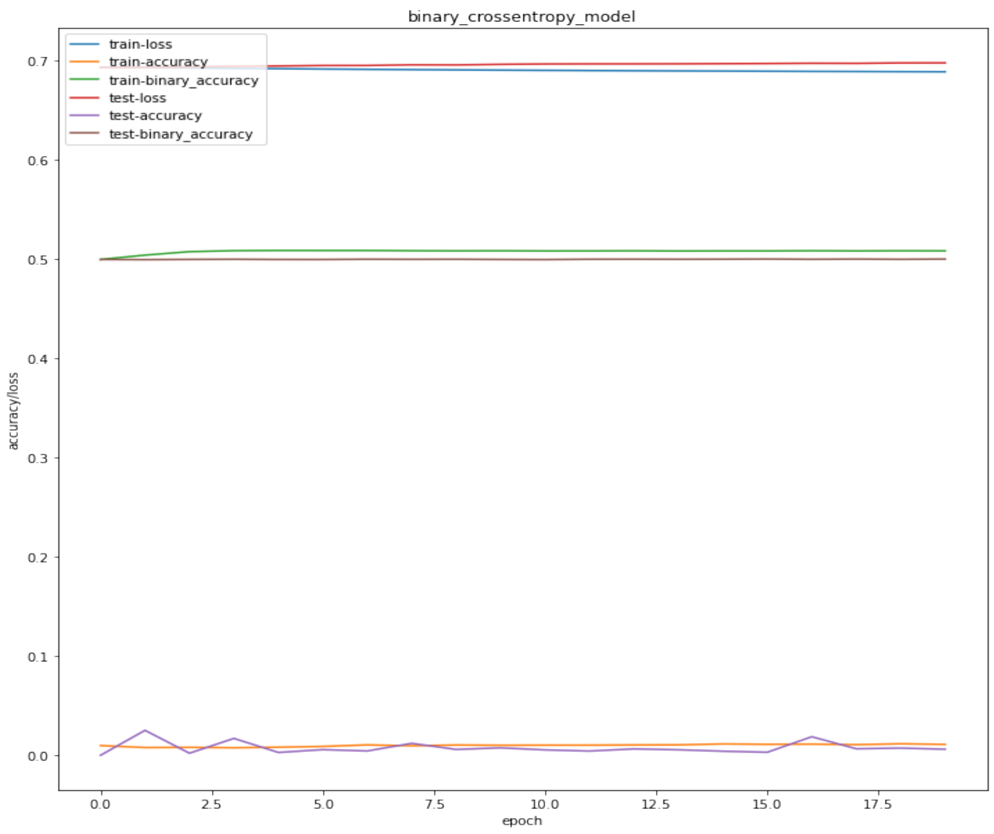

Figure 4.17 : Type-D2 – Binary Crossentropy Loss

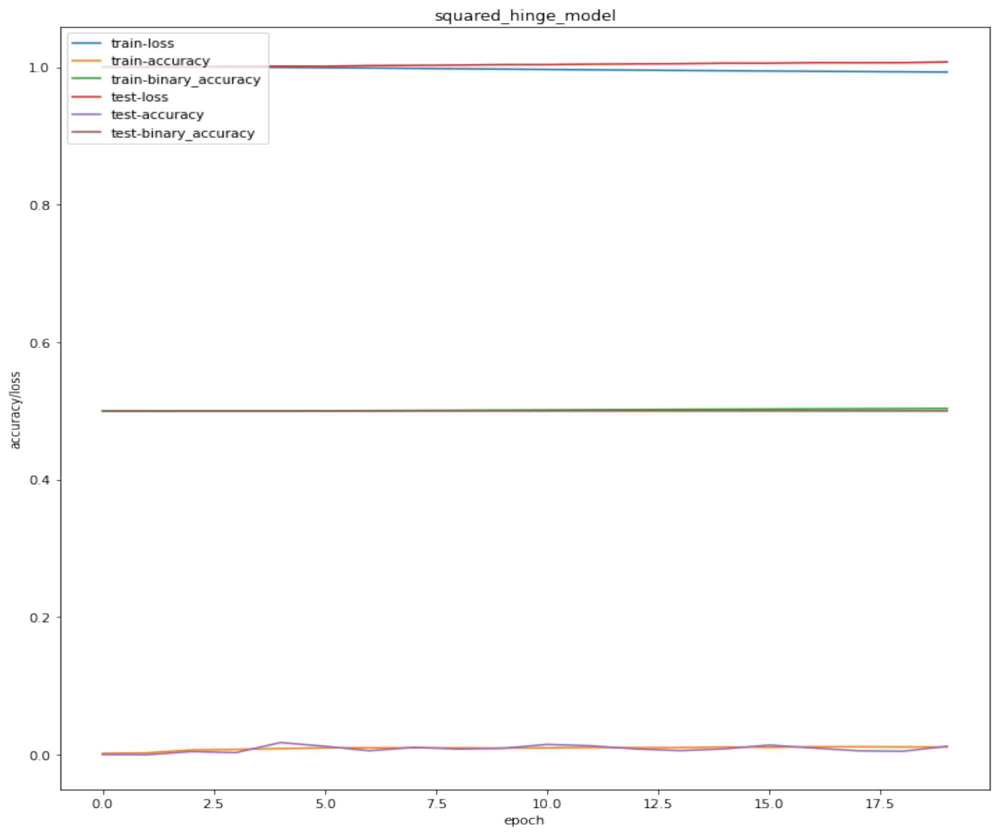

Figure 4.18 : Type-D2 – Squared Hinge Loss

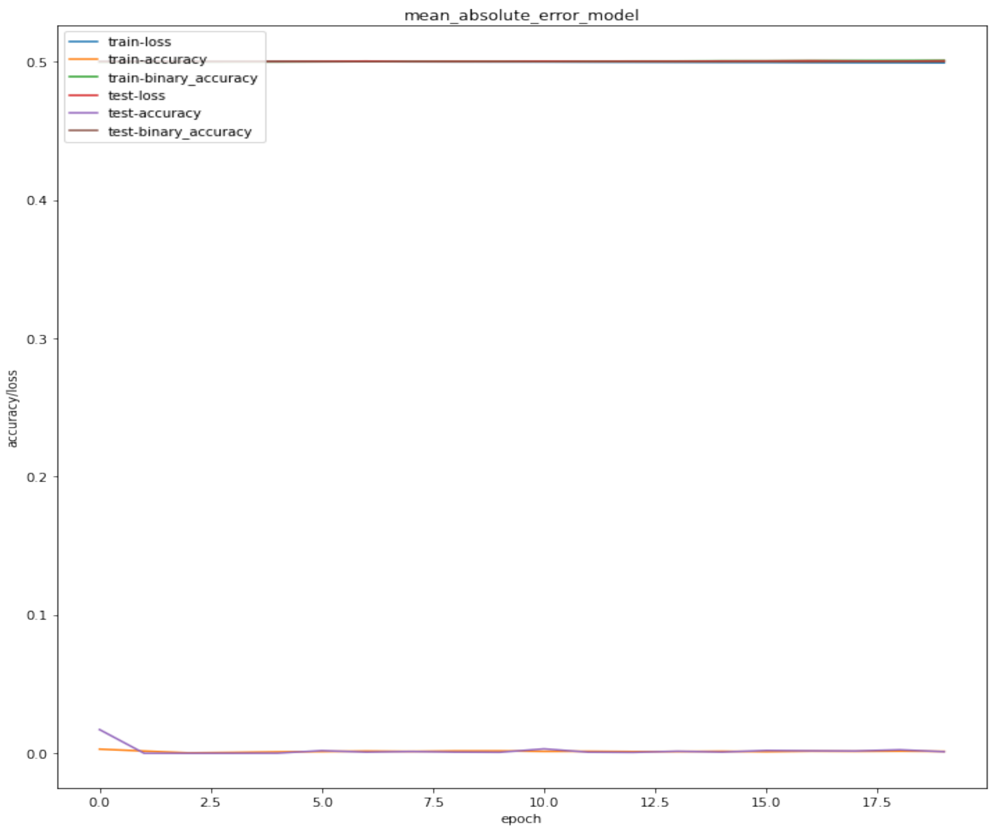

Figure 4.19 : Type-D2 – Mean Absolute Error

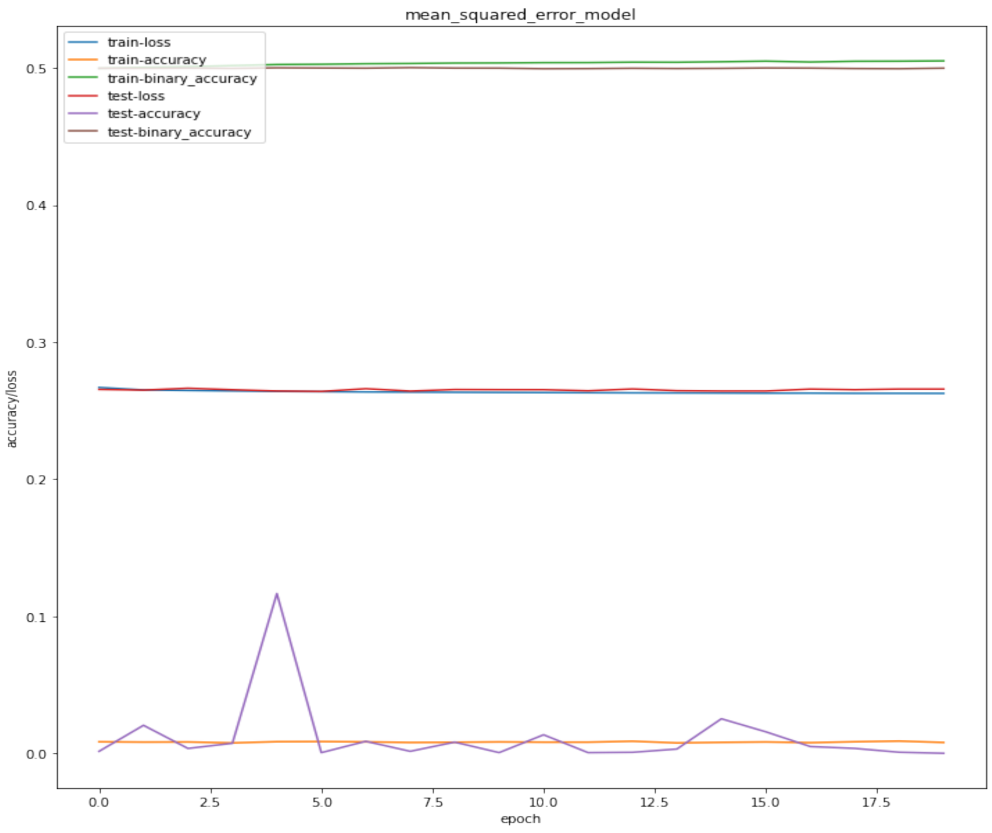

Figure 4.20 : Type-D2 - Mean Squared Error

4.3 Predicted Sequence

After compiling and fitting the model with respect to training set of data set, a long sequence of predicted values model is obtained from the model. Once the sequence of predicted values are obtained then these values are passed through the NIST test suite to check their statistical properties and hence, comment on their randomness.

The table shown below displays the type of model, loss function and number of NIST tests passed by the predicted values with these models:

Type of Model / **Loss Function**	**A**	**B**	**C**	**D1**	**D2**
Binary Crossentropy	6	8	10	6	11
Squared Hinge	5	6	8	4	10
Mean Absolute Error	6	8	7	5	4
Mean Squared Error	5	8	9	9	9

Chapter-5

Conclusion

After studying the graph and analyzing the results, it can be examined that with these moderate data sets, neural network can’t be trained properly. Although with the data sets used in this experiment, it can be concluded that neural networks perform much better i.e., to generate random sequences when binary crossentropy loss function is used. The model in which binary crossentropy has been used as loss funtion, that model has passed maximum number of NIST tests. Some test like Random Excursions, Random Excursions Variant and Universal Tests does not pass in any of these models. It might be possible that these tests require more number of bits to conclude this feature of randomness.

Hence, from the observation it can be concluded that the random sequences which are obtained by the neural network after training them with this moderate data set should not be used, at least where the security is the primary concern as they doesn’t passes all the NIST tests.

Future Work

Due to limited resources, this experiment has been performed on moderate data sets. In future, the same can be performed with large data sets say, billions of data to get more accurate result and conclusion. The experiments can also be performed using some more loss functions.

Bibliography

[1] A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications; NIST: Gaithersburg, MD, USA, 2010

[2] Using layer recurrent neural network to generate pseudo random number sequences. Int. J. Comput. Sci. Issues 2012, 9, 324–334

[3] Pseudo-random Number Generation Using Binary Recurrent Neural Networks, A Technical Report submitted to Kalamazoo College 2007

[4] Abdi, H. A neural network primer. J. Biol. Syst. 1994

[5] Pseudo-Random Number Generation Using Generative Adversarial Networks. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases; Springer: Berlin/Heidelberg, Germany, 2018; pp. 191–200 De Bernardi, M.; Khouzani, M.; Malacaria, P.

[6] Yishai M. Elyada and David Horn: " Can dynamic neural filters produce pseudorandom sequences?", Artificial Neural Networks: Biological Inspirations – ICANN 2005

[7] SIAM Journal on Computing , A simple unpredictable pseudo random number generator, 15(2):364–383, 1986

[8] A modified Elman neural network model with application to dynamical systems identification, 1996 IEEE International Conference on Systems, Man and Cybernetics. Information Intelligence and Systems (Cat. No.96CH35929)

[9] Learning from Pseudo-Randomness with an Artificial Neural Network – Does God Play Pseudo-Dice?, Fenglei Fan, Ge Wang1 Biomedical Imaging Center, BME/CBIS Rensselaer Polytechnic Institute, Troy, New York, USA, 12180

[10] The study of the reasons for not giving the results of NIST Tests of Random Walk, Random Walk Variable and Lempel Ziv *E. AVAROĞLU1 , A.B. ÖZER2 , M. TÜRK3 1Malatya,Turkey 2 Firat University, Computer Engineering, Faculty of Engineering, 23119 Elazig, Turkey 3 Firat University, Electric Electronic Engineering, Faculty of Engineering, 23119 Elazig, Turkey

[11] On the Interpretation of Results from the NIST Statistical Test Suite Marek SYS´ 1 , Zdenˇek Rˇ´IHA1 , Vashek MATYA´Sˇ1 , Kinga MARTON ´ 2 , Alin SUCIU2 1 Masaryk University, Brno, Czech Republic, 2 Technical University of Cluj-Napoca, Romania

[12] On the pass rate of NIST statistical test suite for randomness Akihiro Yamaguchi1 , Takaaki Seo1 and Keisuke Yoshikawa1 1 Department of Information and Systems Engineering, Fukuoka Institute of Technology, Wajiro 3-30-1, Higashi-ku, Fukuoka 811-0295, Japan