

**COVENTRY UNIVERSITY   
 (Priory St, Coventry CV1 5FB, United Kingdom)**

**MODULE CODE: 7088 CEM**

**ARTIFICIAL NEURAL NETWORKS**

**PROJECT TOPIC:**

**MOBILE NETWORK DATA INTRUSION ANALYSIS USING ARTIFICIAL NEURAL NETWORK**

*Submitted by*

**STUDENT NAME: ABIN NADH RANGANADHAN SUJATHA**

**STUDENT ID: 13546847**

Table of Contents

[Introduction 3](file:///C:\Users\HaCkEr%20bAbY\OneDrive\Desktop\COVENTRY%20UNIVERSITY.docx#_Toc128732617)

[Related Work 4](file:///C:\Users\HaCkEr%20bAbY\OneDrive\Desktop\COVENTRY%20UNIVERSITY.docx#_Toc128732618)

[Dataset 8](file:///C:\Users\HaCkEr%20bAbY\OneDrive\Desktop\COVENTRY%20UNIVERSITY.docx#_Toc128732619)

[Methods 11](file:///C:\Users\HaCkEr%20bAbY\OneDrive\Desktop\COVENTRY%20UNIVERSITY.docx#_Toc128732620)

[Experimental Results 13](file:///C:\Users\HaCkEr%20bAbY\OneDrive\Desktop\COVENTRY%20UNIVERSITY.docx#_Toc128732621)

[Discussion and Future Work 22](file:///C:\Users\HaCkEr%20bAbY\OneDrive\Desktop\COVENTRY%20UNIVERSITY.docx#_Toc128732622)

[References 23](file:///C:\Users\HaCkEr%20bAbY\OneDrive\Desktop\COVENTRY%20UNIVERSITY.docx#_Toc128732623)

[Appendix 26](file:///C:\Users\HaCkEr%20bAbY\OneDrive\Desktop\COVENTRY%20UNIVERSITY.docx#_Toc128732624)

# Introduction

Mobile Network Data Intrusion is a method of exploiting or attacking a system by gaining unauthorized access to the data and information stored in the system. The system can be a physical device such as a mobile phone or a computer system. Mobile Network Data Intrusion can be used to steal confidential information, disrupt the system, or even cause serious damage. The analysis of this kind of data is important to identify anomalies in the system and to protect it from malicious attacks. The use of Artificial Neural Network (ANN) for Mobile Network Data Intrusion analysis has been gaining popularity in recent times (Chen et al., 2019). ANN is a powerful and sophisticated technique that uses a set of mathematical algorithms to model complex systems. This technique can be used to recognize patterns in data and to predict the output of a system accurately. ANN has become increasingly popular in security systems as it can detect anomalies from vast amounts of data quickly and accurately.

In this assignment, two different algorithms will be used to analyse Mobile Network Data Intrusion. The first one is a Deep Neural Network model that uses multiple layers of neurons to recognize patterns in the data and to predict the output. The second algorithm is a blend of Artificial Neural Network and Linear Classifier model. This model combines the strengths of both the ANN and the Linear Classifier. By using this model, it is possible to identify anomalies from enormous amounts of records with greater accuracy**.** The accuracies and the loss of both the models will be compared to determine which model should be used to predict the class of the Mobile Network Intrusion data. The Deep Neural Network model offers the advantage of being able to detect anomalies from large datasets quickly and accurately (Abiodun et al., 2018). However, the Linear Classifier model may be more accurate in detecting anomalies from smaller datasets. By comparing the accuracy and loss of both the models, it is possible to determine which model is more suitable for Mobile Network Data Intrusion analysis.

Deep Neural Network and a combination of Artificial Neural Network and Linear Classifier models are two powerful techniques that can be used to analyse Mobile Network Data Intrusion. By comparing the accuracies and the losses of both the models, it is possible to determine which model is more suitable for the analysis of Mobile Network Data Intrusion. Both of these models offer the advantage of being able to detect anomalies from large datasets quickly and accurately (Chung & Abbott, 2021).

# Related Work

As per the study of Deng et al., (2018) the researcher discusses the organization outline for Internet security as well as some important security technologies, including authentication and key, access control and verification, routing safeguards, personal privacy, intrusion prevention, and fault tolerance and intrusion. The researcher also analyses the characteristic features of networking security and security issues. The authors also go through existing IoT network security issues and emphasis the value of intrusion detection. The effectiveness of various intrusion detection methods and how they apply to IoT architecture are assessed, and a potential direction for further research is recommended. The use of data mining as well as machine learning techniques to research network intrusion technology is a major emphasis of this paper. It is highlighted that a single detection model or class feature is insufficient to increase network intrusion detection's detection rate. Public databases are used to verify the efficacy of the proposed model.

In order to further increase the precision of intrusion detection and to provide better security and privacy protection, artificial neural networks (ANNs) can be used to detect mobile network data intrusions. By using ANNs, it is possible to identify patterns in the data and identify anomalies that may indicate malicious activity. Furthermore, ANNs can be used to monitor the network for signs of unexpected behaviors and react quickly to any malicious activities (Deng et al., 2018). This technique can also be used to detect sophisticated intrusions that may not be detectable with traditional security measures. In conclusion, ANNs are a powerful tool for detecting mobile network data intrusions and can be used to provide better security and privacy protection for IoT networks.

The study conducted by Thakkar & Lohiya, (2020) emphasises the need to take on the issue of cyber security threats and the research being done in the area of cyber security. To study the presentation of IDS, an IDS dataset has been developed. Intrusion Detection Systems (IDS) are used to secure computer systems and users. The classification of network data into legitimate and malicious traffic is done using a variety of machine learning (ML) and data mining (DM) approaches. The investigation offers a review of the ML and DM approaches used for IDS and talks about the recent development of IDS datasets that may be used by different research communities. The aforementioned study offers insightful information about the necessity for IDS datasets and the methods employed for categorizing the network data with regard to the issue of Mobile Network Data Intrusion Analysis Using Artificial Neural Networks.

In addition, the researcher covers new developments in IDS datasets that might be leveraged to create effective ML and DM-based IDS. Moreover, Artificial Neural Network (ANN) is a versatile ML technique that has been used for intrusion detection in mobile networks. ANN has the ability to learn and recognize patterns in data, which can be used for detecting malicious activities. Additionally, it can be used to identify the source of the attack and the type of attack. The advantage of using ANN for intrusion detection in mobile networks is that it can detect any anomalies in the network and alert the user (Thakkar & Lohiya, 2020). Furthermore, it can be used to classify data into normal and attack traffic. However, ANN is a computationally expensive technique and requires a lot of data for training. This can be a disadvantage for mobile networks due to limited resources. Additionally, ANN requires a lot of time for training, which can be a challenge for real-time detection of intrusions.

The study by Saranya et al.,(2020) explains a comparison of different Machine Learning (ML) methods used in Intrusion Detection Systems (IDS) for applications like fog computing, the Internet of Things (IoT), big data, and 5G networks. Linear Discriminant Analysis (LDA), Classification and Regression Trees (CART), and Random Forest are the ML methods employed in the study. The work was evaluated using the KDD-CUP dataset, and its effectiveness was assessed and compared with that of the most recent studies. LDA, CART, and Random Forest are ML techniques that can be utilized in the context of Mobile Network Data Intrusion Analysis employing Artificial Neural Networks to identify and categories the intrusions. These algorithms can be used to build an effective intrusion detection system for mobile networks. One of the advantages of using ML algorithms for intrusion detection is that they can detect unknown and previously unseen attacks, which is difficult to do with traditional intrusion detection systems. Furthermore, these algorithms can also be used to detect malicious activities in real time and alert the network operators of any potential attacks.

Nevertheless, one drawback of utilising ML systems for intrusion detection is that they need a lot of training data, which can be expensive and challenging to gather. These methods can also be expensive to compute because of their complexity. As such, it may be difficult to deploy them in real-time environments, such as mobile networks, where resources are limited. Another disadvantage of ML algorithms is that they can be prone to false positives and false negatives, which can lead to incorrect decisions being made (Saranya et al., 2020). In conclusion, ML algorithms like LDA, CART and Random Forest can be used to detect and classify intrusions in mobile networks. While these algorithms offer great potential, they also come with certain drawbacks, such as the need for large amounts of data and the risk of false positives or false negatives. As such, it is important to consider these potential drawbacks when using ML algorithms for intrusion detection in mobile networks.

The research paper by Wu & Guo, (2019) presents a novel hierarchical CNN+RNN neural network model, named LuNet, which is intended to detect network infiltration more effectively than conventional signature-based methods. The internet traffic data can be processed by LuNet to extract both temporal and spatial data, which can improve the accuracy and timeliness of network intrusion detection. Although LuNet has a high detection rate, a substantial number of false alarms continue to be generated. Researchers could think of other approaches to lessen false alarms or concentrate on optimizing the hyper - parameters of LuNet to increase the detection rate in order to tackle this problem. Moreover, no information regarding the LuNet model's implementation is provided in the research report. It is crucial that researchers share additional details about how LuNet was implemented, including the kind of neural network that was utilised, how many layers there were, and how many neurons there were in each layer. Moreover, the performance of LuNet on various types of data is not covered in the paper. Researchers might benefit from assessing LuNet's performance on many forms of data, including text, photos, audio, and video. This will give researchers a better grasp of the model's performance and enable them to use the model to analyse various sorts of data with more knowledge.

Finally, the paper does not discuss any potential security issues related to the use of LuNet (Wu & Guo, 2019). As the model is based on machine learning, there is always a risk that malicious actors may be able to access the training data and use it to modify the LuNet model to suit their own purposes. Thus, it is important for researchers to consider potential security risks associated with the use of the model and to develop strategies to mitigate them.

According to the research study of Drewek-Ossowicka et al., (2020) they give a general review of neural network research on intrusion detection systems (IDS). It is obvious that neural networks have emerged as a key instrument in the industry for the implementation of previously unattainable functionality. The study's description of the literature review offers insightful information about the state of this field's research at the moment. The study does not, however, offer a critical assessment of the application of neural networks to intrusion detection.

Despite the obvious benefits of utilising neural networks for IDS, the study does not thoroughly examine any potential disadvantages. The authors may have, for instance, addressed the risk of over fitting and the requirement for sizable datasets for efficient training. Furthermore, the study doesn't deal with the problem of developing datasets that are enough for actual use cases. This is a crucial matter because the calibre of the datasets utilised heavily influences how accurate IDS research is.

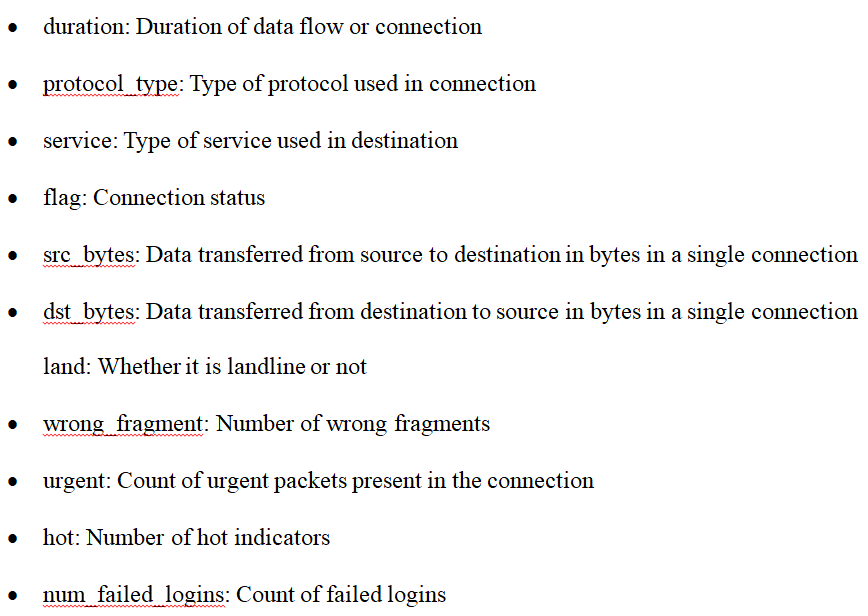
The study also falls short of offering a full analysis of how neural networks are used in other fields. Although the authors discuss the possible applications of IDS in fields including the Internet of Things, clouds, and communication devices, they do not offer a critical assessment of how the field's current research might be strengthened (Drewek-Ossowicka et al., 2020). It's unfortunate that the authors didn't explore the possibility of fusing neural networks with certain other machine learning (ML) methods, such reinforcement learning, to provide IDS solutions that are more potent.

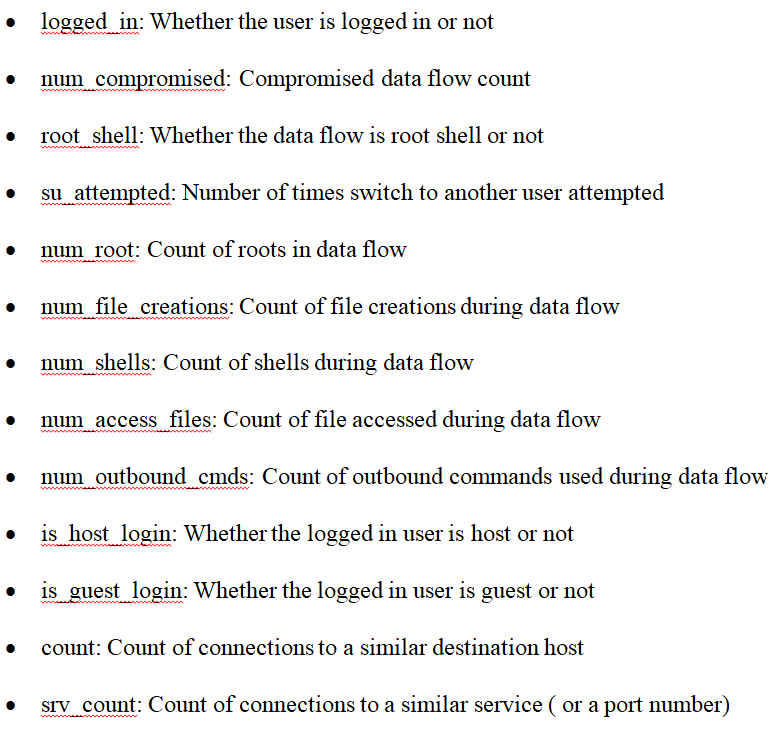
According to Yang & Wang, (2019) conventional intrusion detection technology has been significantly enhanced by the examination of wireless network intrusion detection using the enhanced convolutional neural network (ICNN) suggested in this paper. When compared to more conventional models like LeNet-5, DBN, and RNN, the model described in this study has superior detection accuracy, a higher true positive rate, and a lower false positive rate. When contrasted to the IDABCNN and NIDMBCNN approaches, the model offers a significant advantage. This study does, however, have certain shortcomings. Because KDDTest + is the only network traffic data set included in the study, the findings may be prejudiced. However, the study did not account for other elements that can have an impact on the reliability of the findings, such as the frequency, source, and type of attacks. Additionally, model overfitting, which may jeopardize the model's accuracy, is not addressed in the study (Yang & Wang, 2019). Finally, the authors provide no information about how the model might function in a real-world setting.

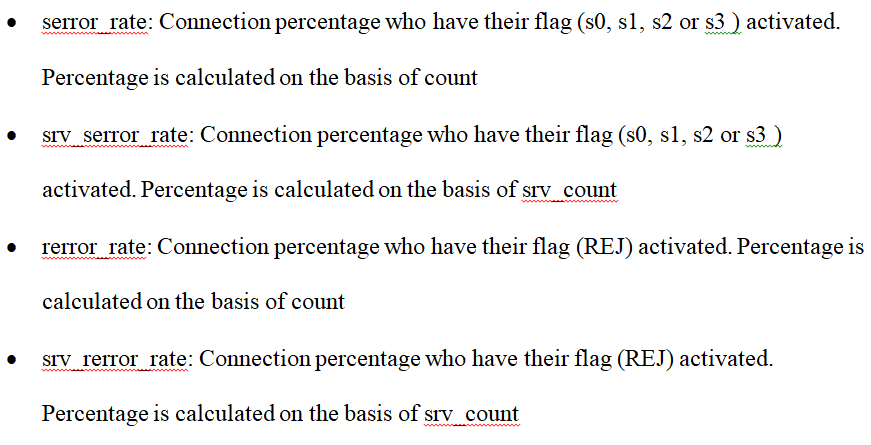
# Dataset

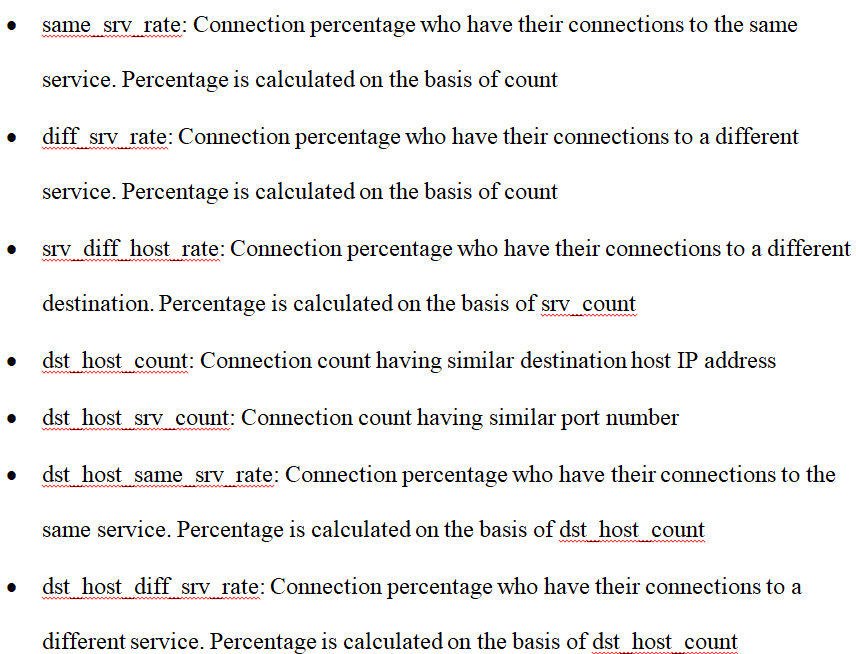
We have used multiple variables in the dataset. The description of the variables is given below:

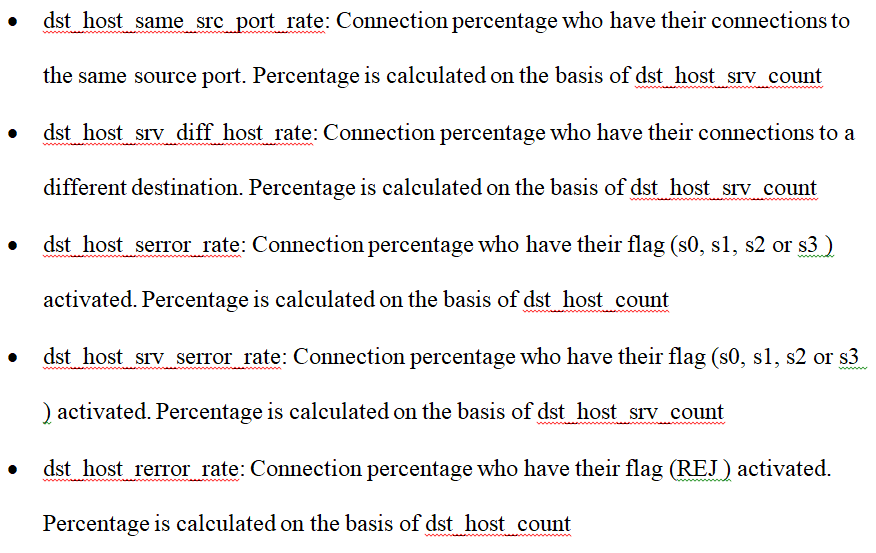
Duration is a term used to describe the length of a data flow or connection. The protocol type refers to the kind of connection protocol being utilised. The term "service" describes the kind of service that is provided at the location. The connection status is shown by a flag. Src bytes measures the volume of data sent via a single connection from the source to the destination in bytes. Dst bytes measures the volume of data sent over a single connection from the source to the destination in bytes. If it is a landline connection or not, it is determined by the land. The connection's Wrong fragment value represents the total amount of these incorrect fragments. The number of urgent packets in the connection is referred to as urgent. The quantity of hot indicators in a connection is referred to as hot. The total number of failed login attempts is Num failed Logins. To detect whether a user is logged in or not, use logged in. The number of compromised data flows is Num compromised. In the interest of identifying whether the data flow is root shell or not, root shell is utilised.

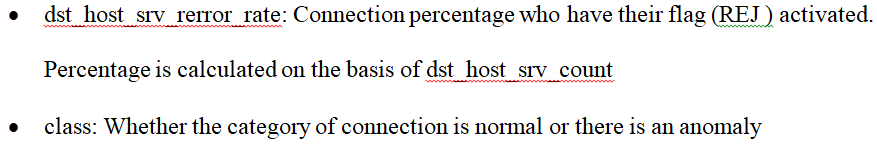












Su attempted, count, srv count, same srv rate, diff srv rate, srv diff host rate, dst host count, and dst host srv count are properties linked to connections. The number of times a switch to another user was attempted is recorded as "Su attempted," 'count' represents the number of connections to a similar destination host,'srv count' represents the number of connections to a similar service (or port number),'same srv rate' represents the proportion of connections to the same service, 'diff srv rate' represents the proportion of connections to a different service,'srv diff host rate' represents the proportion of connections to a different destination, 'dst\_

The features related to the data flow include 'num\_root', 'num\_file\_creations', 'num\_shells', 'num\_access\_files', 'num\_outbound\_cmds', 'is\_host\_login', 'is\_guest\_login', 'serror\_rate' and 'srv\_serror\_rate'. 'Num\_root' is the count of roots in data flow, 'num\_file\_creations' is the count of file creations during data flow, 'num\_shells' is the count of shells during data flow, 'num\_access\_files' is the count of file accessed during data flow, 'num\_outbound\_cmds' is the count of outbound commands used during data flow, 'is\_host\_login' is whether the logged in user is host or not, 'is\_guest\_login' is whether the logged in user is guest or not, 'serror\_rate' is the connection percentage who have their flag (s0, s1, s2 or s3) activated and 'srv\_serror\_rate' is the connection percentage who have their flag (s0, s1, s2 or s3) activated. The last two features in the dataset are 'rerror\_rate' and 'srv\_rerror\_rate'. 'Rerror\_rate' is the connection percentage who have their flag (REJ) activated and 'srv\_rerror\_rate' is the connection percentage who have their flag (REJ) activated.

The process of cleaning and formalizing the data set starts with an inspection of the data set. This inspection includes identifying the data type of each column, finding missing values and understanding the data values (Vaibhav et al., 2019). The next step is data cleaning which involves dealing with the missing values and any irrelevant data points. This can be done by either deleting the rows with missing values or by filling them with mean or median values. After that, the data set needs to be formalized by converting the categorical data into numerical values. This can be done by using dummy coding or one-hot encoding techniques (Tang, 2014).

Feature engineering comes next after the data has been codified and cleansed. Choosing the model's most pertinent attributes is required for this. This entails figuring out the relationships between the attributes and the desired outcome. The characteristics with the greatest correlation scores are chosen for additional examination. The data is separated into a training dataset and a test dataset after the features have been chosen. The model is trained using the training set, and its performance is assessed using the test set. Lastly, the model is assessed using a variety of evaluation criteria, including recall, accuracy, precision, and F1-score. The model is adjusted to increase performance after evaluation, which is followed by an analysis of the findings. In order to create a strong and reliable prediction model, the procedure of cleaning and formalizing the data set is crucial (Volkovs et al., 2014).

# Methods

Artificial Neural Network (ANN) is a type of machine learning algorithm that mimics the function and structure of the human brain. It is used to solve complex problems that are otherwise too difficult to solve with traditional methods (Dike et al., 2018). It is predicated on the idea of neural networks with interconnected nodes that function as processing centers for data and decision-making. ANNs may be employed to identify patterns, categorize information, and forecast the future since they have the ability to learn from data. A hidden layer or layers, as well as an output layer, make up an ANN model. Neurons are found in each layer and are linked to both the strata that lie above and below them as well as to one another. Data is transmitted from one neurons to the next via the weighted edges that connect the neurons. During the learning phase, the values of the edges are modified to enhance the model's efficiency.

A blend of Artificial Neural Network and Linear Classifier model is a combination of two different machine learning approaches. The linear classifier model uses a linear combination of predictor variables to predict an outcome, while the ANN model uses a multi-layered network of interconnected neurons to process data. The combination of these two models allows for a more powerful and accurate prediction model. The ANN model is used to process the input data and extract features that are useful for the linear classifier model. The linear classifier model then uses the extracted features to classify the data and make predictions. This combined approach can be used to resolve multifaceted problems that are otherwise hard to solve with standard linear classifier models (Khamis & Matrawy, 2020).

In addition to its ability to solve complex problems, the blend of Artificial Neural Network and Linear Classifier model is also useful for its ability to generalize. Unlike the linear classifier model, the ANN model can learn from data and make predictions on unseen data. This makes it especially useful for applications that require generalization, such as facial recognition. For performing the analysis related to the Mobile Network Intrusion data, we would use two different advanced machine learning algorithms. The initial one would be an Artificial Neural Network model and the next one would be a blend of Artificial Neural Network and Linear Classifier model (Goldberg, 2016).

Before the implementation of the model, we have to convert the feature variables into Tensorflow form. In the next step, we would be defining the parameters of the model using a function. This function would have the number of epochs, size of the batch and shuffle type. We would be using the training data for creating the parameters (Kaushik et al., 2020). The training data is primarily used to train the advanced machine learning algorithms. We have kept 75% of the data in the training set.

This function would have the number of epochs, size of the batch and shuffle count. We would use the test data for this function which has 25% of the dataset values. The next step would have the parameters for creating the model. For the Artificial Neural Network model, we would use 3 number of classes and the Adam optimizer. The activation layer will be of type “relu” and the dropout value would be equal to 0.30.

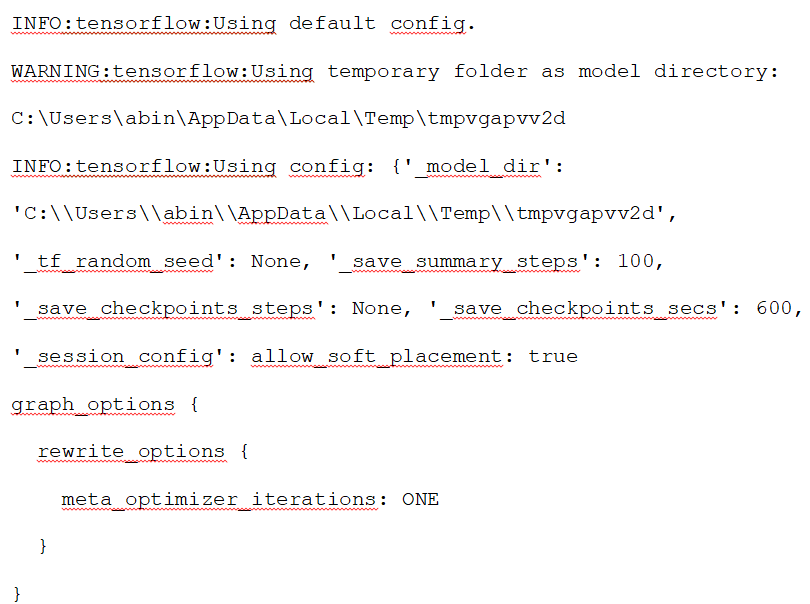
The next phase involves training the model by selecting the batch size, shuffle type, number of steps, and epochs. Calling the evaluation function after specifying the number of epochs, size of the batch, reshuffle type, and number of steps that would produce the model's characteristics would be the last step (Adhikari & Agrawal, 2013).

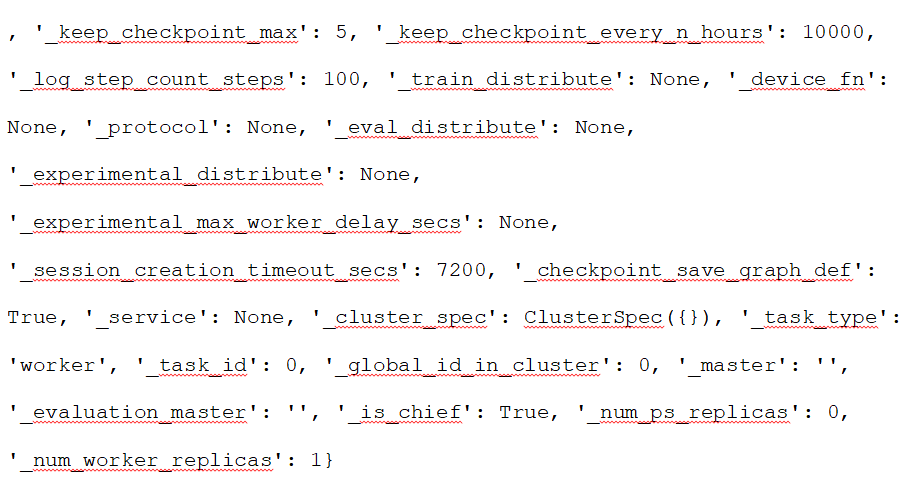
# Experimental Results

We have implemented two models for this analysis. The first one would be an Artificial Neural Network model and the next one would be a blend of Artificial Neural Network and Linear Classifier model. TensorFlow is an open source library for machine learning developed by Google. It is a comprehensive, end-to-end platform for building and deploying machine learning models, with the ability to scale from small projects to large-scale distributed training and predictions. It can be used to develop, train and deploy various Artificial Neural Network (ANN) architectures for the purpose of mobile network data intrusion analysis. ANNs are powerful tools for analyzing large and complex datasets, as they are capable of learning and extracting patterns from data. They can also be used to classify data, detect anomalies, and make predictions.

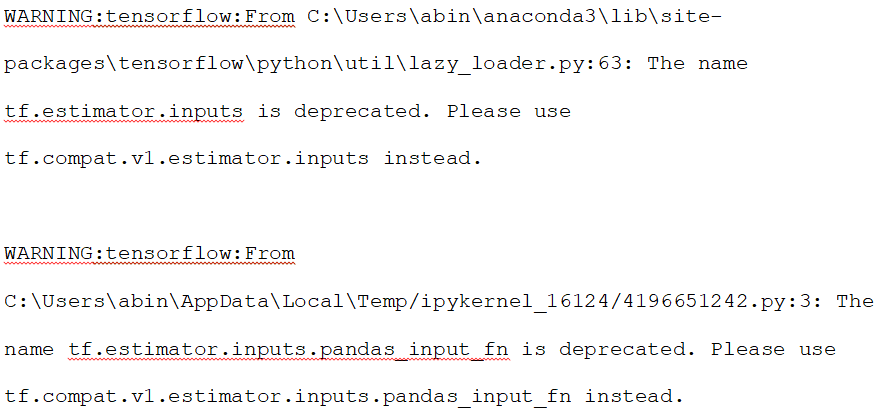
TensorFlow can be used to create a custom ANN architecture that is suitable for mobile network data intrusion analysis. The ANN can be trained using labeled data, and can then be used to identify and classify network intrusions (Gatys et al., 2015). The data used for training can come from various sources, such as network logs, system logs, and intrusion detection systems. Once the ANN is trained, it can be deployed in a mobile network and used to detect any suspicious activity.

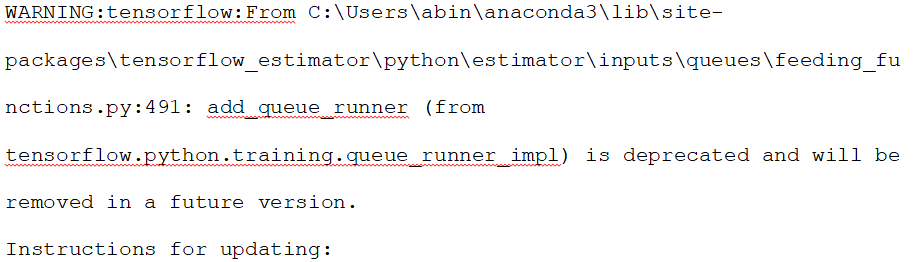
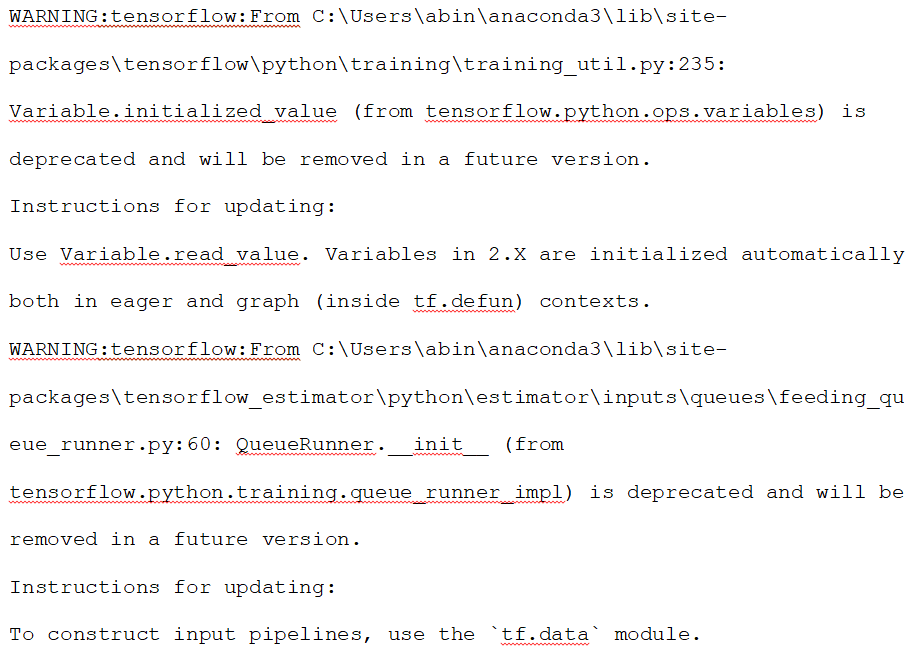
TensorFlow can also be used to develop and deploy deep learning models for mobile network data intrusion analysis. Deep learning models are based on artificial neural networks, but with multiple hidden layers that allow them to learn and detect more complex patterns and relationships in data. These models can be used to detect and classify network intrusions, as well as to develop anomaly detection systems. The model parameters after creating the Artificial Neural Network Classifier are given below:

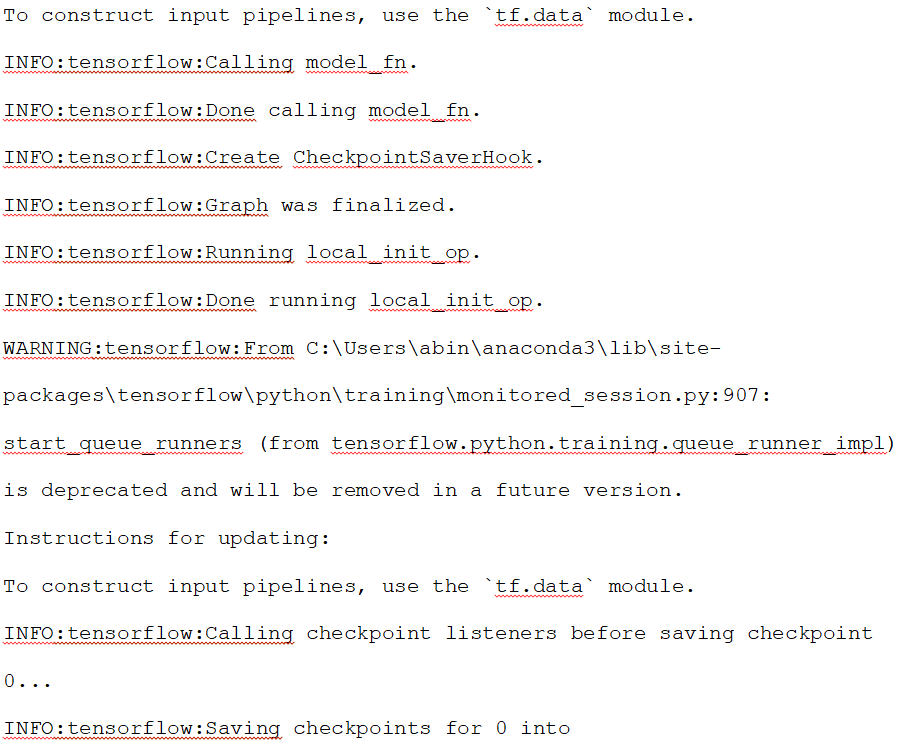
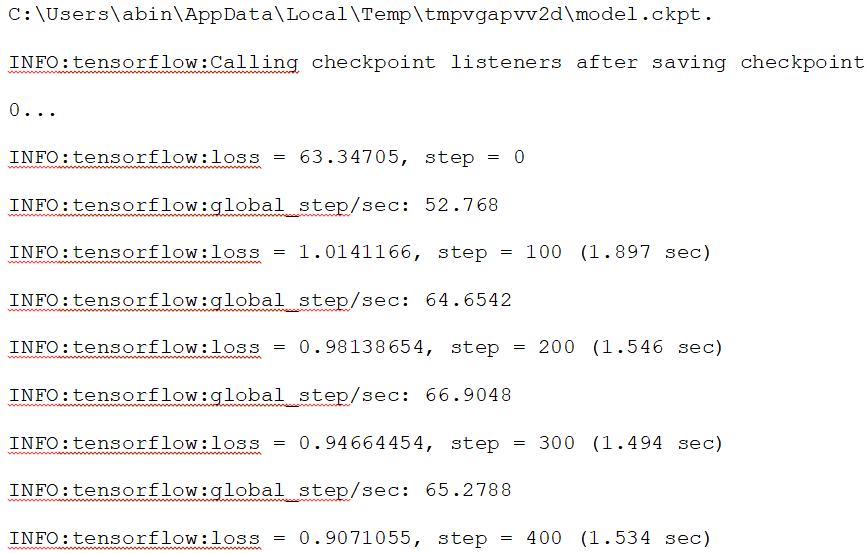


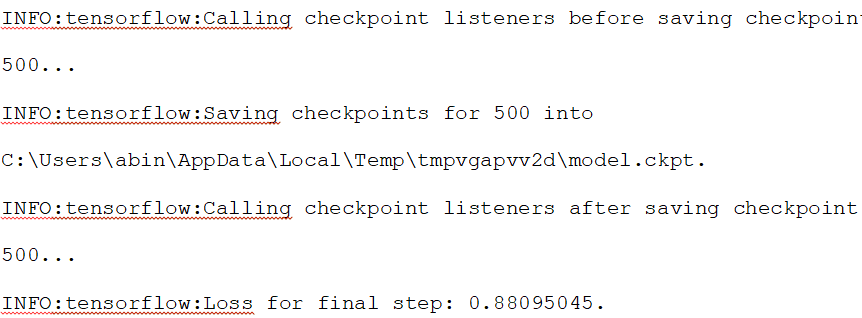


The training output of the Artificial Neural Network model is given below:



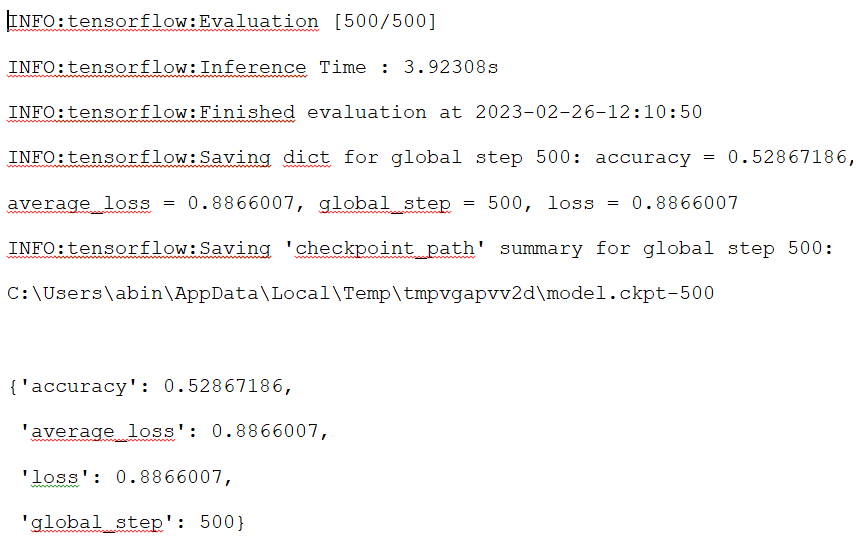




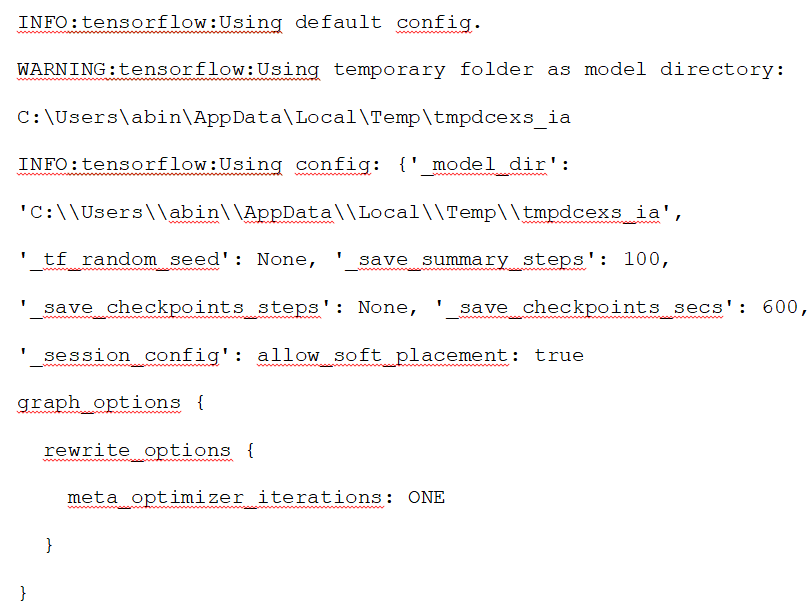
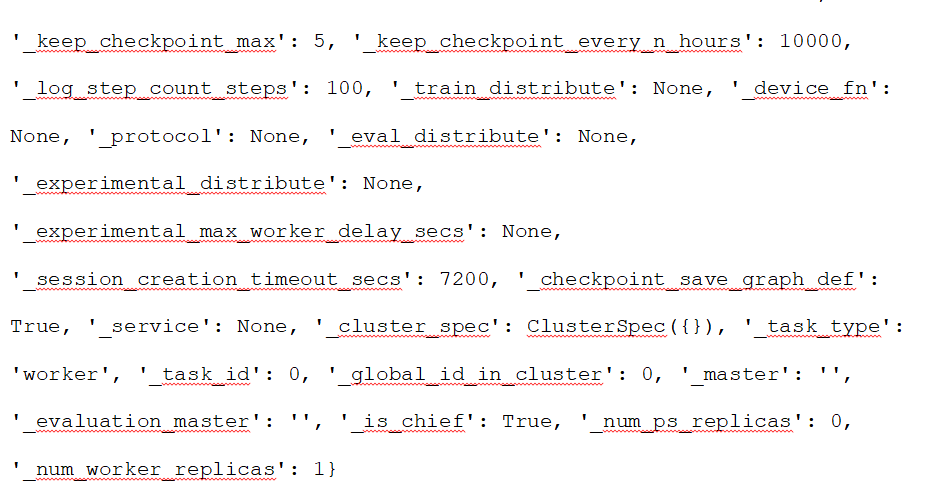
The output parameters of the Artificial Neural Network model are given below:



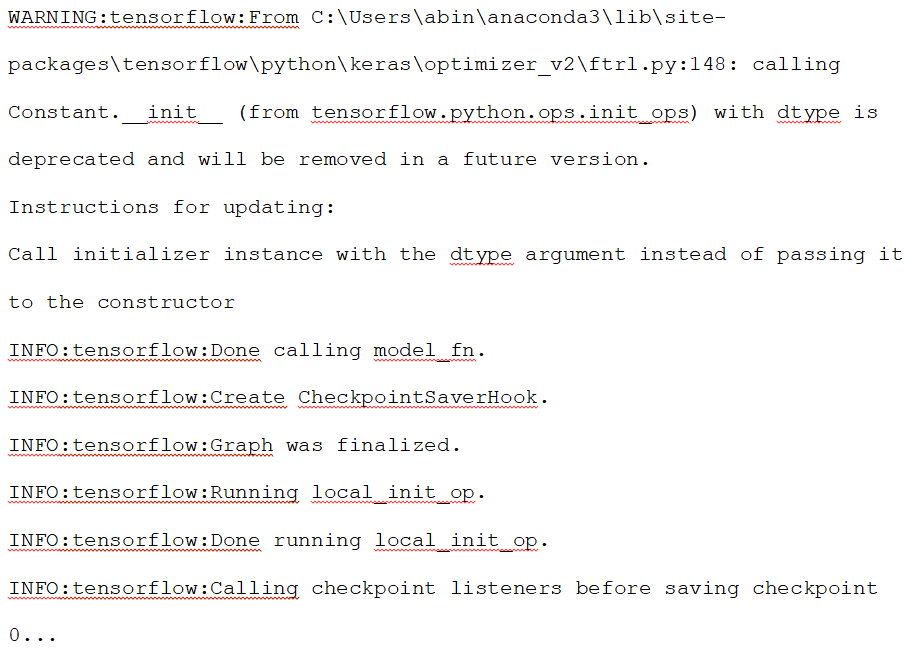


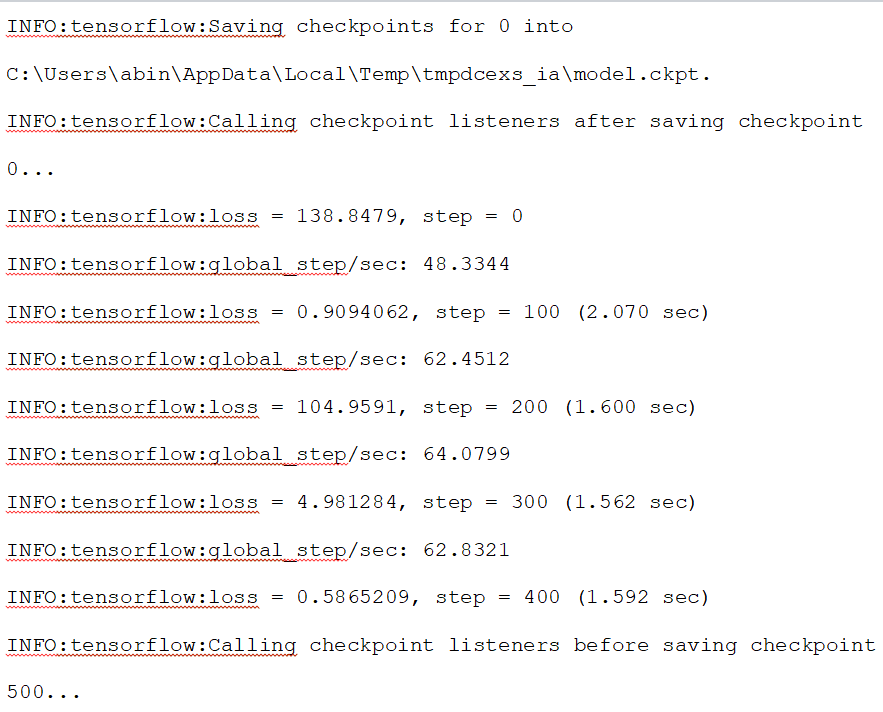
The accuracy of the Artificial Neural Network (ANN) model is 0.5287, which indicates that the independent variables present in the model have the capacity to predict the type of class of network intrusion with an accuracy of 52.87%. Although the outcome is good, the model's accuracy could still use some more work. Also, to enhance the model's performance, the loss of the ANN model must be decreased. We can employ a variety of methods to lower the loss, including adding regularization, altering the activation function, and changing the network's topology. To further reduce the loss of the ANN model, we can employ a variety of optimization techniques, such as stochastic gradient descent. By combining these methods, we may increase the ANN model's precision and decrease its loss, resulting in more accurate predictions (Pang, 2020).

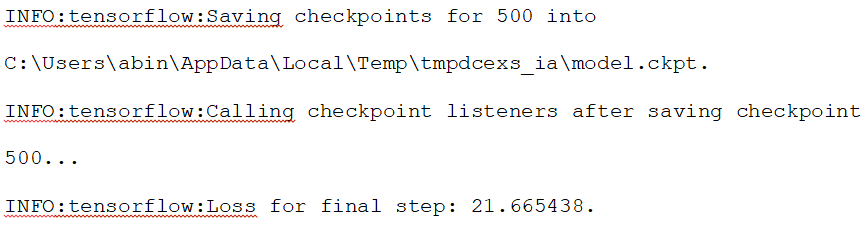
The model parameters after creating the blend of Artificial Neural Network and Linear Classifier model are given below:

, 

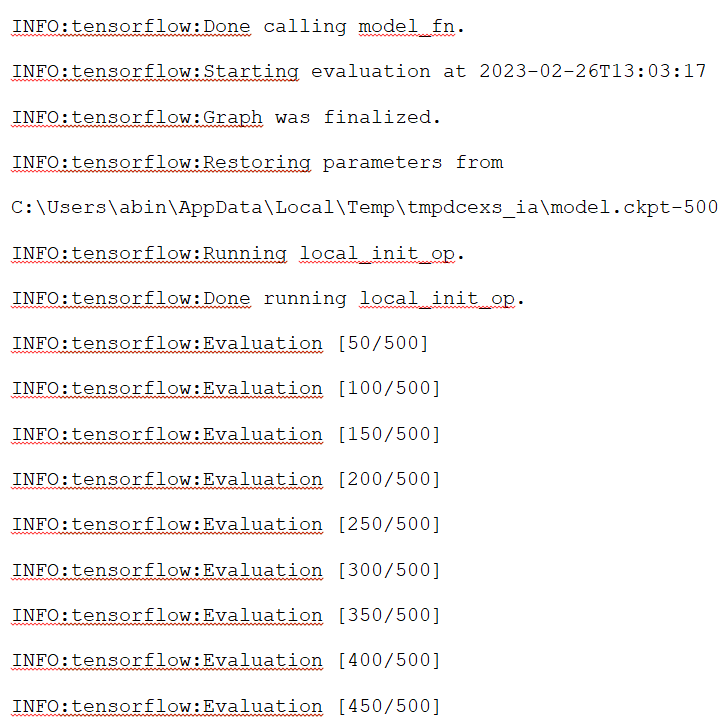
The training output of the blend of Artificial Neural Network and Linear Classifier model is given below:

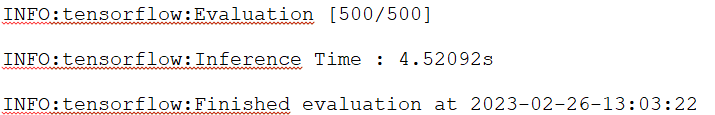


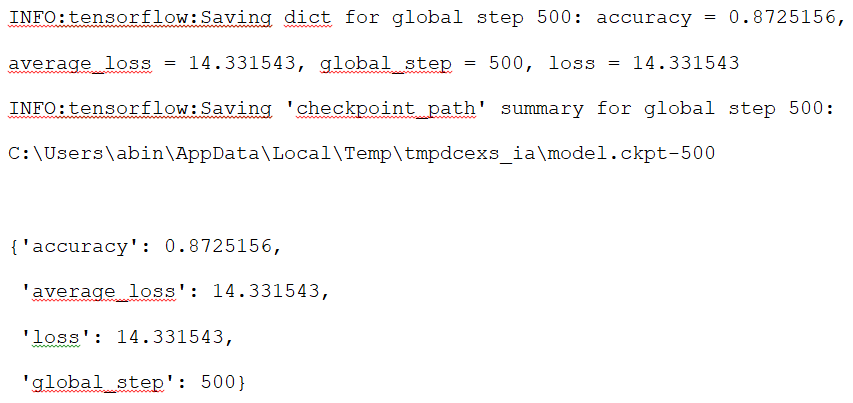




The output parameters of the blend of Artificial Neural Network and Linear Classifier model are given below:







The blend of Artificial Neural Network and Linear Classifier model has an accuracy of 87.25%, which is quite impressive in terms of predicting the type of network intrusion. This indicates that the independent variables present in the model have the capability to accurately classify the type of intrusion (Hamed Habibi Aghdam & Elnaz Jahani Heravi, 2017). However, we should focus on reducing the loss of this blend model in order to further improve the accuracy. This could be achieved by optimizing the model’s hyper parameters, making sure the data is normalized, and adjusting the model’s architecture if necessary. Additionally, the model could be further enhanced by adding more features, such as packet size, packet count, etc. With these steps, we can expect to see a further increase in accuracy, and perhaps even a reduction in the loss of this blend model.

# Discussion and Future Work

We have used two sophisticated machine learning models in this project to determine whether a connection's category is normal or abnormal. The models consist of a linear model and an Artificial Neural Network (ANN) combination. To boost the precision of our predictions, we can also apply more sophisticated machine learning models (Gatys et al., 2015). Two of the machines learning models which can be used for this purpose are Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs). RNNs are a type of ANN which is specifically designed for processing sequence data. They use a special type of memory layer, called a recurrent layer, to process the sequence data. This allows them to capture the long-term dependencies in the data, which is important for many types of predictions. RNNs can be used for predicting anomalies in a sequence of connection data.

CNNs are another type of ANN which is specifically designed for image processing. They are composed of convolutional layers, which extract features from the input images. These features can then be used to detect anomalies in the connection data. CNNs have been shown to be highly effective at detecting anomalies in images, and they can also be used to detect anomalies in sequences of connection data. In addition to RNNs and CNNs, there are many other advanced machine learning models which can be used for this task. These include Support Vector Machines, Decision Trees, and Random Forests. Each of these models has its own advantages and disadvantages, and should be carefully evaluated before being implemented (Kuo, 2016). Overall, it can be concluded that a blend of ANN and Linear Classifier model is the best choice for predicting whether the category of connection is normal or there is an anomaly. However, other advanced machine learning models can be used to improve the accuracy of the predictions. RNNs and CNNs are two of the most promising options, and should be further explored (Carlini & Wagner, 2017).

**References**

Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Mohamed, N. A., & Arshad, H. (2018). State-of-the-art in artificial neural network applications: A survey. *Heliyon*, *4*(11), e00938. https://doi.org/10.1016/j.heliyon.2018.e00938

Adhikari, R., & Agrawal, R. K. (2013). A combination of artificial neural network and random walk models for financial time series forecasting. *Neural Computing and Applications*, *24*(6), 1441–1449. https://doi.org/10.1007/s00521-013-1386-y

Carlini, N., & Wagner, D. (2017). Towards Evaluating the Robustness of Neural Networks. *2017 IEEE Symposium on Security and Privacy (SP)*. <https://doi.org/10.1109/sp.2017.49>

Chen, M., Challita, U., Saad, W., Yin, C., & Debbah, M. (2019). Artificial Neural Networks-Based Machine Learning for Wireless Networks: A Tutorial. *IEEE Communications Surveys & Tutorials*, *21*(4), 3039–3071. https://doi.org/10.1109/comst.2019.2926625

Deng, L., Li, D., Yao, X., & Wang, H. (2018). RETRACTED ARTICLE: Mobile network intrusion detection for IoT system based on transfer learning algorithm. *Cluster Computing*, *22*(S4), 9889–9904. <https://doi.org/10.1007/s10586-018-1847-2>

Dike, H. U., Zhou, Y., Deveerasetty, K. K., & Wu, Q. (2018). Unsupervised Learning Based On Artificial Neural Network: A Review. *2018 IEEE International Conference on Cyborg and Bionic Systems (CBS)*. <https://doi.org/10.1109/cbs.2018.8612259>

Drewek-Ossowicka, A., Pietrołaj, M., & Rumiński, J. (2020). A survey of neural networks usage for intrusion detection systems. *Journal of Ambient Intelligence and Humanized Computing*, *12*(1), 497–514. https://doi.org/10.1007/s12652-020-02014-x

‌Gatys, L., Ecker, A. S., & Bethge, M. (2015). Texture Synthesis Using Convolutional Neural Networks. *Advances in Neural Information Processing Systems*, *28*. <https://proceedings.neurips.cc/paper/2015/hash/a5e00132373a7031000fd987a3c9f87b-Abstract.html>

Goldberg, Y. (2016). A Primer on Neural Network Models for Natural Language Processing. *Journal of Artificial Intelligence Research*, *57*, 345–420. https://doi.org/10.1613/jair.4992

Hamed Habibi Aghdam, & Elnaz Jahani Heravi. (2017). Guide to Convolutional Neural Networks. *SpringerLink*. https://doi.org/10.1007-978-3-319-57550-6

Kaushik, S., Choudhury, A., Sheron, P. K., Dasgupta, N., Natarajan, S., Pickett, L. A., & Dutt, V. (2020). AI in Healthcare: Time-Series Forecasting Using Statistical, Neural, and Ensemble Architectures. *Frontiers in Big Data*, *3*. https://doi.org/10.3389/fdata.2020.00004

Khamis, R. A., & Matrawy, A. (2020). Evaluation of Adversarial Training on Different Types of Neural Networks in Deep Learning-based IDSs. *2020 International Symposium on Networks, Computers and Communications (ISNCC)*. https://doi.org/10.1109/isncc49221.2020.9297344

‌ Kuo, C.-C. . J. (2016). Understanding convolutional neural networks with a mathematical model. *Journal of Visual Communication and Image Representation*, *41*, 406–413. https://doi.org/10.1016/j.jvcir.2016.11.003

Pang, B. (2020). *Deep Learning With TensorFlow: A Review - Bo Pang, Erik Nijkamp, Ying Nian Wu, 2020*. Journal of Educational and Behavioral Statistics. https://journals.sagepub.com/doi/abs/10.3102/1076998619872761?journalCode=jebb

Saranya, T., Sridevi, S., Deisy, C., Chung, T. D., & Khan, M. K. A. Ahamed. (2020). Performance Analysis of Machine Learning Algorithms in Intrusion Detection System: A Review. *Procedia Computer Science*, *171*, 1251–1260. <https://doi.org/10.1016/j.procs.2020.04.133>

Thakkar, A., & Lohiya, R. (2020). A Review of the Advancement in Intrusion Detection Datasets. *Procedia Computer Science*, *167*, 636–645. <https://doi.org/10.1016/j.procs.2020.03.330>

Tang, N. (2014). Big Data Cleaning. *Web Technologies and Applications*, 13–24. https://doi.org/10.1007/978-3-319-11116-2\_2

Vaibhav, V., Singh, S., Stewart, C., & Neubig, G. (2019). Improving Robustness of Machine Translation with Synthetic Noise. *ArXiv.org*. <https://doi.org/10.48550/arXiv.1902.09508>

Volkovs, M., Fei Chiang, Szlichta, J., & Miller, R. J. (2014). Continuous data cleaning. *2014 IEEE 30th International Conference on Data Engineering*. <https://doi.org/10.1109/icde.2014.6816655>

‌ Wu, P., & Guo, H. (2019). LuNet: A Deep Neural Network for Network Intrusion Detection. *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*. https://doi.org/10.1109/ssci44817.2019.9003126

‌ Yang, H., & Wang, F. (2019). Wireless Network Intrusion Detection Based on Improved Convolutional Neural Network. *IEEE Access*, *7*, 64366–64374. https://doi.org/10.1109/access.2019.2917299

‌

# Appendix

import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns  
from sklearn.model\_selection import train\_test\_split  
from tensorflow import keras  
import tensorflow as tf

df = pd.read\_csv(r"C:\Abin\Mobile\_Network\_Intrusion\_Data.csv")  
df.head()

duration protocol\_type service flag src\_bytes dst\_bytes land \  
0 0 tcp ftp\_data SF 491 0 0   
1 0 udp other SF 146 0 0   
2 0 tcp private S0 0 0 0   
3 0 tcp http SF 232 8153 0   
4 0 tcp http SF 199 420 0   
  
 wrong\_fragment urgent hot ... dst\_host\_srv\_count \  
0 0 0 0 ... 25   
1 0 0 0 ... 1   
2 0 0 0 ... 26   
3 0 0 0 ... 255   
4 0 0 0 ... 255   
  
 dst\_host\_same\_srv\_rate dst\_host\_diff\_srv\_rate \  
0 0.17 0.03   
1 0.00 0.60   
2 0.10 0.05   
3 1.00 0.00   
4 1.00 0.00   
  
 dst\_host\_same\_src\_port\_rate dst\_host\_srv\_diff\_host\_rate \  
0 0.17 0.00   
1 0.88 0.00   
2 0.00 0.00   
3 0.03 0.04   
4 0.00 0.00   
  
 dst\_host\_serror\_rate dst\_host\_srv\_serror\_rate dst\_host\_rerror\_rate \  
0 0.00 0.00 0.05   
1 0.00 0.00 0.00   
2 1.00 1.00 0.00   
3 0.03 0.01 0.00   
4 0.00 0.00 0.00   
  
 dst\_host\_srv\_rerror\_rate class   
0 0.00 normal   
1 0.00 normal   
2 0.00 anomaly   
3 0.01 normal   
4 0.00 normal   
  
[5 rows x 42 columns]

#Changing the form of the dependent variable  
def num\_trans(ctgry):  
 if ctgry=='normal':  
 return 0  
 elif ctgry=='anomaly':  
 return 1

#Calling the defined function  
df['ctgry'] = df['class'].apply(num\_trans)

#Forming independent and dependent variables  
X = df.drop(['class','ctgry'],axis=1)  
y = df.loc[:,'ctgry']

#Splitting the dataset  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y,test\_size=0.25)

#Tensorflow variables for Deep Neural Network Implementation  
featur = [tf.feature\_column.numeric\_column('logged\_in'),  
 tf.feature\_column.numeric\_column('src\_bytes'),  
 tf.feature\_column.numeric\_column('dst\_bytes'),  
 tf.feature\_column.numeric\_column('land'),  
 tf.feature\_column.numeric\_column('wrong\_fragment'),  
 tf.feature\_column.numeric\_column('urgent'),  
 tf.feature\_column.numeric\_column('count'),  
 tf.feature\_column.numeric\_column('num\_failed\_logins'),  
 tf.feature\_column.numeric\_column('serror\_rate'),  
 tf.feature\_column.numeric\_column('num\_compromised'),  
 tf.feature\_column.numeric\_column('root\_shell'),  
 tf.feature\_column.numeric\_column('dst\_host\_count'),  
 tf.feature\_column.numeric\_column('su\_attempted'),  
 tf.feature\_column.numeric\_column('num\_root'),  
 tf.feature\_column.numeric\_column('num\_file\_creations'),  
 tf.feature\_column.numeric\_column('num\_shells'),  
 tf.feature\_column.numeric\_column('num\_access\_files'),  
 tf.feature\_column.numeric\_column('duration'),  
 tf.feature\_column.numeric\_column('diff\_srv\_rate'),  
 tf.feature\_column.numeric\_column('is\_guest\_login'),  
 tf.feature\_column.numeric\_column('is\_host\_login'),  
 tf.feature\_column.numeric\_column('dst\_host\_srv\_rerror\_rate'),  
 tf.feature\_column.numeric\_column('dst\_host\_srv\_diff\_host\_rate'),  
 tf.feature\_column.numeric\_column('srv\_serror\_rate'),  
 tf.feature\_column.numeric\_column('rerror\_rate'),  
 tf.feature\_column.numeric\_column('srv\_rerror\_rate'),  
 tf.feature\_column.numeric\_column('same\_srv\_rate'),  
 tf.feature\_column.numeric\_column('srv\_diff\_host\_rate'),  
 tf.feature\_column.numeric\_column('srv\_count'),  
 tf.feature\_column.numeric\_column('dst\_host\_rerror\_rate'),  
 tf.feature\_column.numeric\_column('dst\_host\_same\_srv\_rate'),  
 tf.feature\_column.numeric\_column('dst\_host\_diff\_srv\_rate'),  
 tf.feature\_column.numeric\_column('dst\_host\_same\_src\_port\_rate'),  
 tf.feature\_column.numeric\_column('num\_outbound\_cmds'),  
 tf.feature\_column.numeric\_column('dst\_host\_serror\_rate'),  
 tf.feature\_column.numeric\_column('dst\_host\_srv\_serror\_rate'),  
 tf.feature\_column.numeric\_column('dst\_host\_srv\_count'),  
 tf.feature\_column.numeric\_column('hot')  
 ]

#Artificial Neural Network model parameters  
def inpt(num\_epochs,num\_btchs,shuffle):  
 return tf.compat.v1.estimator.inputs.pandas\_input\_fn(  
 x=X\_train,  
 y=y\_train,  
 batch\_size=num\_btchs,  
 shuffle=shuffle,  
 num\_epochs=num\_epochs  
 )

#Artificial Neural Network evaluation  
def evaltn(num\_epochs,num\_btchs,shuffle):  
 return tf.compat.v1.estimator.inputs.pandas\_input\_fn(  
 x=X\_test,  
 y=y\_test,  
 batch\_size=num\_btchs,  
 shuffle=shuffle,  
 num\_epochs=num\_epochs  
 )

#Creating the Artificial Neural Network Classifier  
ann\_mdl = tf.estimator.DNNClassifier(n\_classes=3,  
 optimizer='Adam',  
 hidden\_units=[1024,512,256,32,3],  
 feature\_columns=featur,  
 activation\_fn=tf.nn.relu,  
 dropout=0.30  
 )

INFO:tensorflow:Using default config.  
WARNING:tensorflow:Using temporary folder as model directory: C:\Users\Abin\AppData\Local\Temp\tmpvgapvv2d  
INFO:tensorflow:Using config: {'\_model\_dir': 'C:\\Users\\Abin\\AppData\\Local\\Temp\\tmpvgapvv2d', '\_tf\_random\_seed': None, '\_save\_summary\_steps': 100, '\_save\_checkpoints\_steps': None, '\_save\_checkpoints\_secs': 600, '\_session\_config': allow\_soft\_placement: true  
graph\_options {  
 rewrite\_options {  
 meta\_optimizer\_iterations: ONE  
 }  
}  
, '\_keep\_checkpoint\_max': 5, '\_keep\_checkpoint\_every\_n\_hours': 10000, '\_log\_step\_count\_steps': 100, '\_train\_distribute': None, '\_device\_fn': None, '\_protocol': None, '\_eval\_distribute': None, '\_experimental\_distribute': None, '\_experimental\_max\_worker\_delay\_secs': None, '\_session\_creation\_timeout\_secs': 7200, '\_checkpoint\_save\_graph\_def': True, '\_service': None, '\_cluster\_spec': ClusterSpec({}), '\_task\_type': 'worker', '\_task\_id': 0, '\_global\_id\_in\_cluster': 0, '\_master': '', '\_evaluation\_master': '', '\_is\_chief': True, '\_num\_ps\_replicas': 0, '\_num\_worker\_replicas': 1}

#Training the Artificial Neural Network model  
ann\_mdl.train(input\_fn=inpt(100,128,True),steps=500)

WARNING:tensorflow:From C:\Users\Abin\anaconda3\lib\site-packages\tensorflow\python\util\lazy\_loader.py:63: The name tf.estimator.inputs is deprecated. Please use tf.compat.v1.estimator.inputs instead.  
  
WARNING:tensorflow:From C:\Users\ Abin\AppData\Local\Temp/ipykernel\_16124/4196651242.py:3: The name tf.estimator.inputs.pandas\_input\_fn is deprecated. Please use tf.compat.v1.estimator.inputs.pandas\_input\_fn instead.  
  
WARNING:tensorflow:From C:\Users\ Abin\anaconda3\lib\site-packages\tensorflow\python\training\training\_util.py:235: Variable.initialized\_value (from tensorflow.python.ops.variables) is deprecated and will be removed in a future version.  
Instructions for updating:  
Use Variable.read\_value. Variables in 2.X are initialized automatically both in eager and graph (inside tf.defun) contexts.  
WARNING:tensorflow:From C:\Users\ Abin\anaconda3\lib\site-packages\tensorflow\_estimator\python\estimator\inputs\queues\feeding\_queue\_runner.py:60: QueueRunner.\_\_init\_\_ (from tensorflow.python.training.queue\_runner\_impl) is deprecated and will be removed in a future version.  
Instructions for updating:  
To construct input pipelines, use the `tf.data` module.  
WARNING:tensorflow:From C:\Users\Abin\anaconda3\lib\site-packages\tensorflow\_estimator\python\estimator\inputs\queues\feeding\_functions.py:491: add\_queue\_runner (from tensorflow.python.training.queue\_runner\_impl) is deprecated and will be removed in a future version.  
Instructions for updating:  
To construct input pipelines, use the `tf.data` module.  
INFO:tensorflow:Calling model\_fn.  
INFO:tensorflow:Done calling model\_fn.  
INFO:tensorflow:Create CheckpointSaverHook.  
INFO:tensorflow:Graph was finalized.  
INFO:tensorflow:Running local\_init\_op.  
INFO:tensorflow:Done running local\_init\_op.  
WARNING:tensorflow:From C:\Users\ Abin\anaconda3\lib\site-packages\tensorflow\python\training\monitored\_session.py:907: start\_queue\_runners (from tensorflow.python.training.queue\_runner\_impl) is deprecated and will be removed in a future version.  
Instructions for updating:  
To construct input pipelines, use the `tf.data` module.  
INFO:tensorflow:Calling checkpoint listeners before saving checkpoint 0...  
INFO:tensorflow:Saving checkpoints for 0 into C:\Users\ Abin \AppData\Local\Temp\tmpvgapvv2d\model.ckpt.  
INFO:tensorflow:Calling checkpoint listeners after saving checkpoint 0...  
INFO:tensorflow:loss = 63.34705, step = 0  
INFO:tensorflow:global\_step/sec: 52.768  
INFO:tensorflow:loss = 1.0141166, step = 100 (1.897 sec)  
INFO:tensorflow:global\_step/sec: 64.6542  
INFO:tensorflow:loss = 0.98138654, step = 200 (1.546 sec)  
INFO:tensorflow:global\_step/sec: 66.9048  
INFO:tensorflow:loss = 0.94664454, step = 300 (1.494 sec)  
INFO:tensorflow:global\_step/sec: 65.2788  
INFO:tensorflow:loss = 0.9071055, step = 400 (1.534 sec)  
INFO:tensorflow:Calling checkpoint listeners before saving checkpoint 500...  
INFO:tensorflow:Saving checkpoints for 500 into C:\Users\ Abin\AppData\Local\Temp\tmpvgapvv2d\model.ckpt.  
INFO:tensorflow:Calling checkpoint listeners after saving checkpoint 500...  
INFO:tensorflow:Loss for final step: 0.88095045.

<tensorflow\_estimator.python.estimator.canned.dnn.DNNClassifierV2 at 0x25da41a2ca0>

#Output parameters of the Artificial Neural Network model  
ann\_mdl.evaluate(input\_fn=evaltn(100,128,True),steps=500)

INFO:tensorflow:Calling model\_fn.  
INFO:tensorflow:Done calling model\_fn.  
INFO:tensorflow:Starting evaluation at 2023-02-26T12:10:46  
INFO:tensorflow:Graph was finalized.  
INFO:tensorflow:Restoring parameters from C:\Users\ Abin \AppData\Local\Temp\tmpvgapvv2d\model.ckpt-500  
INFO:tensorflow:Running local\_init\_op.  
INFO:tensorflow:Done running local\_init\_op.  
INFO:tensorflow:Evaluation [50/500]  
INFO:tensorflow:Evaluation [100/500]  
INFO:tensorflow:Evaluation [150/500]  
INFO:tensorflow:Evaluation [200/500]  
INFO:tensorflow:Evaluation [250/500]  
INFO:tensorflow:Evaluation [300/500]  
INFO:tensorflow:Evaluation [350/500]  
INFO:tensorflow:Evaluation [400/500]  
INFO:tensorflow:Evaluation [450/500]  
INFO:tensorflow:Evaluation [500/500]  
INFO:tensorflow:Inference Time : 3.92308s  
INFO:tensorflow:Finished evaluation at 2023-02-26-12:10:50  
INFO:tensorflow:Saving dict for global step 500: accuracy = 0.52867186, average\_loss = 0.8866007, global\_step = 500, loss = 0.8866007  
INFO:tensorflow:Saving 'checkpoint\_path' summary for global step 500: C:\Users\ Abin\AppData\Local\Temp\tmpvgapvv2d\model.ckpt-500

{'accuracy': 0.52867186,  
 'average\_loss': 0.8866007,  
 'loss': 0.8866007,  
 'global\_step': 500}

#Implementing a blend of Artificial Neural Network and Linear Classifier model  
ann\_linear\_model = tf.estimator.DNNLinearCombinedClassifier(n\_classes=3,dnn\_optimizer='Adam',dnn\_dropout=0.30,dnn\_hidden\_units=[1024,512,256,32,3],dnn\_feature\_columns=featur,dnn\_activation\_fn='relu',linear\_feature\_columns=featur)

INFO:tensorflow:Using default config.  
WARNING:tensorflow:Using temporary folder as model directory: C:\Users\ Abin\AppData\Local\Temp\tmpdcexs\_ia  
INFO:tensorflow:Using config: {'\_model\_dir': 'C:\\Users\\ Abin\\AppData\\Local\\Temp\\tmpdcexs\_ia', '\_tf\_random\_seed': None, '\_save\_summary\_steps': 100, '\_save\_checkpoints\_steps': None, '\_save\_checkpoints\_secs': 600, '\_session\_config': allow\_soft\_placement: true  
graph\_options {  
 rewrite\_options {  
 meta\_optimizer\_iterations: ONE  
 }  
}  
, '\_keep\_checkpoint\_max': 5, '\_keep\_checkpoint\_every\_n\_hours': 10000, '\_log\_step\_count\_steps': 100, '\_train\_distribute': None, '\_device\_fn': None, '\_protocol': None, '\_eval\_distribute': None, '\_experimental\_distribute': None, '\_experimental\_max\_worker\_delay\_secs': None, '\_session\_creation\_timeout\_secs': 7200, '\_checkpoint\_save\_graph\_def': True, '\_service': None, '\_cluster\_spec': ClusterSpec({}), '\_task\_type': 'worker', '\_task\_id': 0, '\_global\_id\_in\_cluster': 0, '\_master': '', '\_evaluation\_master': '', '\_is\_chief': True, '\_num\_ps\_replicas': 0, '\_num\_worker\_replicas': 1}

#Training the blend of Artificial Neural Network and Linear Classifier model  
ann\_linear\_model.train(input\_fn=inpt(100,128,True),steps=500)

INFO:tensorflow:Calling model\_fn.

C:\Users\Abin\anaconda3\lib\site-packages\tensorflow\python\keras\engine\base\_layer\_v1.py:1700: UserWarning: `layer.add\_variable` is deprecated and will be removed in a future version. Please use `layer.add\_weight` method instead.  
 warnings.warn('`layer.add\_variable` is deprecated and '

WARNING:tensorflow:From C:\Users\Abin\anaconda3\lib\site-packages\tensorflow\python\keras\optimizer\_v2\ftrl.py:148: calling Constant.\_\_init\_\_ (from tensorflow.python.ops.init\_ops) with dtype is deprecated and will be removed in a future version.  
Instructions for updating:  
Call initializer instance with the dtype argument instead of passing it to the constructor  
INFO:tensorflow:Done calling model\_fn.  
INFO:tensorflow:Create CheckpointSaverHook.  
INFO:tensorflow:Graph was finalized.  
INFO:tensorflow:Running local\_init\_op.  
INFO:tensorflow:Done running local\_init\_op.  
INFO:tensorflow:Calling checkpoint listeners before saving checkpoint 0...  
INFO:tensorflow:Saving checkpoints for 0 into C:\Users\a Abin\AppData\Local\Temp\tmpdcexs\_ia\model.ckpt.  
INFO:tensorflow:Calling checkpoint listeners after saving checkpoint 0...  
INFO:tensorflow:loss = 138.8479, step = 0  
INFO:tensorflow:global\_step/sec: 48.3344  
INFO:tensorflow:loss = 0.9094062, step = 100 (2.070 sec)  
INFO:tensorflow:global\_step/sec: 62.4512  
INFO:tensorflow:loss = 104.9591, step = 200 (1.600 sec)  
INFO:tensorflow:global\_step/sec: 64.0799  
INFO:tensorflow:loss = 4.981284, step = 300 (1.562 sec)  
INFO:tensorflow:global\_step/sec: 62.8321  
INFO:tensorflow:loss = 0.5865209, step = 400 (1.592 sec)  
INFO:tensorflow:Calling checkpoint listeners before saving checkpoint 500...  
INFO:tensorflow:Saving checkpoints for 500 into C:\Users\ Abin\AppData\Local\Temp\tmpdcexs\_ia\model.ckpt.  
INFO:tensorflow:Calling checkpoint listeners after saving checkpoint 500...  
INFO:tensorflow:Loss for final step: 21.665438.

<tensorflow\_estimator.python.estimator.canned.dnn\_linear\_combined.DNNLinearCombinedClassifierV2 at 0x25da5084fa0>

#Output parameters of blend of Artificial Neural Network and Linear Classifier model  
ann\_linear\_model.evaluate(input\_fn=evaltn(100,128,True),steps=500)

INFO:tensorflow:Calling model\_fn.

C:\Users\ Abin\anaconda3\lib\site-packages\tensorflow\python\keras\engine\base\_layer\_v1.py:1700: UserWarning: `layer.add\_variable` is deprecated and will be removed in a future version. Please use `layer.add\_weight` method instead.  
 warnings.warn('`layer.add\_variable` is deprecated and '

INFO:tensorflow:Done calling model\_fn.  
INFO:tensorflow:Starting evaluation at 2023-02-26T13:03:17  
INFO:tensorflow:Graph was finalized.  
INFO:tensorflow:Restoring parameters from C:\Users\ Abin\AppData\Local\Temp\tmpdcexs\_ia\model.ckpt-500  
INFO:tensorflow:Running local\_init\_op.  
INFO:tensorflow:Done running local\_init\_op.  
INFO:tensorflow:Evaluation [50/500]  
INFO:tensorflow:Evaluation [100/500]  
INFO:tensorflow:Evaluation [150/500]  
INFO:tensorflow:Evaluation [200/500]  
INFO:tensorflow:Evaluation [250/500]  
INFO:tensorflow:Evaluation [300/500]  
INFO:tensorflow:Evaluation [350/500]  
INFO:tensorflow:Evaluation [400/500]  
INFO:tensorflow:Evaluation [450/500]  
INFO:tensorflow:Evaluation [500/500]  
INFO:tensorflow:Inference Time : 4.52092s  
INFO:tensorflow:Finished evaluation at 2023-02-26-13:03:22  
INFO:tensorflow:Saving dict for global step 500: accuracy = 0.8725156, average\_loss = 14.331543, global\_step = 500, loss = 14.331543  
INFO:tensorflow:Saving 'checkpoint\_path' summary for global step 500: C:\Users\ Abin\AppData\Local\Temp\tmpdcexs\_ia\model.ckpt-500

{'accuracy': 0.8725156,  
 'average\_loss': 14.331543,  
 'loss': 14.331543,  
 'global\_step': 500}