DETECTING FAKE TWITTER ACCOUNTS USING DEEP NEURAL NETWORK

MSc Thesis

Ziad Walid Mohamed Abdelfattah ELGAMMAL

Graduate School of Engineering and Natural Sciences

Electrical, Electronics Engineering and Cyber Systems

July, 2023

DETECTING FAKE TWITTER ACCOUNTS

USING DEEP NEURAL NETWORK

A THESIS SUBMITTED TO

THE GRADUATE SCHOOL OF

ENGINEERING AND NATURAL SCIENCES

OF ISTANBUL MEDIPOL UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR

THE DEGREE OF

MASTER OF SCIENCE

IN

ELECTRICAL, ELECTRONICS ENGINEERING AND CYBER SYSTEMS

By

Ziad Walid Mohamed Abdelfattah Elgammal

July, 2023

DETECTING FAKE TWITTER ACCOUNTS USING DEEP NEURAL NETWORK

By Ziad Walid Mohamed Abdelfattah Elgammal

17 July 2023

We certify that we have read this dissertation and that in our opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Master of Science.

_____

Prof. Dr. Reda Alhajj (Advisor)

_____

Prof. Dr. Selim Akyokuş

_____

Prof. Dr. Tansel Özyer

Approved by the Graduate School of Engineering and Natural Sciences:

_____

Prof. Dr. Yasemin Yüksel Durmaz

Director of the Graduate School of Engineering and Natural Sciences

I hereby declare that all information in this document has been obtained and presented in accordance with the academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Signature :

Name, Surname: ZIAD WALID MOHAMED ABDELFATTAH ELGAMMAL

ACKNOWLEDGEMENT

I would like to express my sincere Appreciation and gratitude to all those who have contributed to the fulfillment of this Thesis.

First, I am deeply grateful to my supervisor, Prof. Dr. Reda Alhajj, for his invaluable guidance, support, and mentorship throughout the entire research process. his expertise, constructive feedback, and unwavering encouragement have been instrumental in shaping this study.

Additionally, I would like to extend my appreciation to the faculty and staff of Istanbul Medipol University, School of Engineering and Natural Science, whose unwavering dedication and devotion to education have fostered a supportive academic atmosphere for me. Their expertise, support, and invaluable aid have played a crucial role in the successful completion of this research.

I am indebted to my colleagues and friends who have offered their assistance, shared their insights, and provided a supportive network. Their contributions, discussions, and encouragement have been a source of inspiration and motivation.

Finally, I would like to express my appreciation to my family. Their unconditional love, unwavering belief in my abilities, and continuous encouragement have been the driving force behind my academic journey. Their constant support and understanding have been a source of strength throughout the challenges faced during this research.

I acknowledge that this work would not have been possible without the support and contributions of all those mentioned above, as well as many others who have played a part, however small, in this endeavor. Thank you all for your invaluable contributions

Ziad Walid Mohamed Abdelfattah ELGAMMAL

July 2023

CONTENTS

Page

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

AI : Artificial Intelligence
ML : Machine Learning
DL : Deep Learning
DNN : Deep Neural Network
NLP : Natural Language Processing
CM : Confusion Matrix
TP : True Positives
TN : True Negatives
FP : False Positives
FN : False Negatives
LSTM : Long Short-Term Memory

DERİN SİNİR AĞLARI KULLINAILARAK SAHTE TWİTTER HESAPLARIN TESPİTİ

ÖZET

ZIAD Walid Mohamed Abdelfattah ELGAMMAL

Elektrik-Elektronik Mühendisliği, Yüksek Lisans

Tez Danışmanı: Prof. Dr. Reda Alhajj

Temmuz, 2023

Sosyal medya platformları, sahte hesapların varlığından kaynaklanan ciddi bir sorunla karşı karşıyadır. Sahte hesaplar, zararlı içeriklerin, istenmeyen mesajların ve yanlış bilginin yayılması yoluyla tehdit oluşturmaktadır. Bu çalışma, sahte ve gerçek Twitter hesaplarını ayırt etmek suretiyle bu soruna çözüm bulmayı amaçlamaktadır. Sahte hesapların çevrimiçi topluluklar üzerindeki olumsuz etkilerini azaltma ihtiyacı, bu çalışmanın itici gücüdür.

Sahte hesapların ve bunların zararlı faaliyetlerinin, zararlı bağlantılar yayınlama, spam davranışlarına katılma ve çevrimiçi toplulukları huzursuz etme gibi, etkili bir şekilde tespit edilmesini amaçlayan bir yöntem geliştirmek hedeflenmektedir. Bu çalışma kapsamı özellikle Twitter sahte hesap tespitine odaklanmaktadır.

Gerçek ve sahte hesaplar arasındaki ayrımı yapmak için kullanıcı bilgileri ve tweet'lerden yararlanan kapsamlı bir yaklaşım benimsenmiştir. Farklı optimizasyon yöntemlerini kullanarak önerilen ve uygulanan çeşitli derin öğrenme mimarileri, performans ölçütlerini değerlendirerek eğitilmiş ve test edilmiştir. Gerçek hayat senaryolarını kapsayacak şekilde genişletilen bir veri seti kullanılarak modeller eğitilmiş ve test edilmiştir.

Sonuçlar, sahte ve gerçek hesaplar arasındaki ayrımın umut verici ilerlemeler kaydetti ğini ortaya koymuştur. Tweet içeriğinin kullanıcı meta verileriyle birlikte kullanılmasının sahte hesapların sınıflandırılmasını önemli ölçüde geliştirmediğini göstermektedir. Ayrıca, uygun optimizasyon yöntemlerinin seçilmesinin önemini vurgulamaktadır.

Bu çalışmanın sonuçları, sosyal medya platformları, kullanıcılar ve araştırmacılar açısından önem taşımaktadır. Bulgular, sahte hesaplar ve onların zararlı faaliyetleriyle mücadele konusunda içgörüler sunarak çevrimiçi topluluk güvenliğinin artırılmasına katkıda bulunmaktadır. Araştırma Twitter'e özgü olsa da, elde edilen metodoloji ve içgörüler potansiyel olarak diğer sosyal medya platformlarına da genellenilebilir.

Anahtar sözcükler: Twitter, sosyal medya, sahtekarlık tespiti, derin öğrenme, yanlış bilgi.

DETECTING FAKE TWITTER ACCOUNTS USING DEEP NEURAL NETWORK

ABSTRACT

Ziad Walid Mohamed Abdelfattah ELGAMMAL

MSc in Electrical, Electronics Engineering and Cyber Systems

Advisor: Prof. Dr. Reda Alhajj

July 2023

Social media platforms are currently confronted with a substantial problem concerning the presence of fake accounts. which pose a threat by spreading harmful content, spam, and misinformation. This study aims to address this problem by differentiating between fake and real Twitter accounts. The need to mitigate the negative impact of fake accounts on online communities serves as the driving force of the study.

With a goal of developing an effective method for identifying fake accounts and their fraudulent activities, such as posting harmful links, engaging in spamming behaviors, and disrupting online communities. The scope of this work focuses specifically on Twitter fake account detection.

A comprehensive approach is taken, leveraging user information and tweets to discern between genuine and fake accounts. Various deep learning architectures are proposed and implemented, utilizing different optimizers and evaluating performance metrics. The models are trained and tested using a collected dataset, augmented to cover diverse real-life scenarios.

The results show promising progress in distinguishing between fake and real accounts. revealing that the inclusion of tweet content along with user metadata does not significantly improve the classification of fake accounts. It also highlights the importance of selecting appropriate optimizers.

The implications of this study are relevant to social media platforms, users, and researchers. The findings provide insights into combating fake accounts and their fraudulent activities, contributing to the enhancement of online community safety. While the research is specific to Twitter, the methodology and insights gained may be potentially generalizable to other social media platforms.

Keywords: Twitter, social media, fraud detection, deep learning, misinformation.

CHAPTER 1

1. INTRODUCTION

Social Media affects all aspects of human's life; the way they think, act, talk and behave, what they dress, eat, and buy. Allowing them to overpass a lot of boundaries and connect different, diverse, and distant people and cultures.

Social media usage is widespread across various demographics and is expected to continue to expand to new users over time.

While it has brought so many benefits and opportunities, it has also raised concerns about privacy, misinformation, addiction, and many other psychological impacts and diseases [1]. As social media evolves and shape society, it is crucial to understand its effects and implications. Similar to any other tool, social media can be used for both constructive and harmful purposes, such as improving people's lives and saving time and money or achieving personal gains and manipulating individuals [2].

Many examples of using social media for good or bad exist in all aspects of life. A recent example of this occurred during the devastating earthquake that hit parts of Turkey and Syria in February 2023. Some individuals utilized social media to assist with relief efforts and check on their loved ones and friends in affected cities. Others, however, spread fake news or inflated loss figures to propagate terror and fear among individuals for their own material benefit, disregarding the suffering of the affected people [3].

Like any enterprise striving to flourish and surpass competitors in its industry, social media platforms constantly experiment with various algorithms and techniques to determine which content should be prioritized and promoted to attract more users and generate greater profits, while burying and concealing other content. The primary measure for this is the level of engagement a user receives on their content, which can manifest in various forms and names, contingent on the platform. These may include actions such as "like," "dislike," "love," "angry," "comment," "share," "retweet," "report,"

"follow," and "unfollow," among others. A prevalent method for gauging a user's popularity is by tallying their number of followers, as well as the amount of likes or retweets their content receives. Bad-intentioned people or groups can exploit this fact through various ways to reach people, spread their poisons and achieve their goals.

In the real world, a person can be easily identified and verified through some unique official documents that are issued only to him and hard to forfeit (id, driving license, passport), These document contains a set of identifiers that cannot be duplicated by any means, such as name, fingerprints, image, birth date, and blood Type, etc.
The situation is much different on social media; where each user is identified by a profile, including some basic information about him such as his name, photo, and birth date, with no real authority to check and validate this data.
To create an account, most of the platform requires only one unique email. So, with no real unified checking authority, it has become a very easy task to forge or fake your identity on social media. Thus, a person identity on social media can either be real, false, or fake. A real identity, having only one unique account that is created using his information and being used solely and freely by the person himself.

In case of a false identity, a user would use someone else (mostly a celebrity, famous figure or organization) identity and information to create an account and disguises himself as the celebrity.

In case of Fake identity, a person would create accounts using invented names and information that may coincide with real persons [4].

Detecting false identity is relatively easier than detecting fake identity. The focus of this study is to capture these fake accounts on Twitter created using some invented identities for many malicious causes such as sharing fake news, misleading web rating, and spam.

1.1. Problem Definition and Motivation

Social media has brought an overwhelming influx of news and information spanning across every corner of the globe. With a diverse range of individuals behind these posts, each driven by their own unique intentions and agendas, it becomes increasingly challenging, and sometimes even impossible, to assure that every piece of content shared on these platforms is both ethical and reliable. Hence, it is important to have an automated

process capable of differentiating fake from real content. This includes identifying accounts who are dedicated to broadcast false information, known as fake accounts.

This study aims to address the significant threat posed by fake accounts on social media, which can propagate various forms of harmful content, spam, and misinformation. Examples of fraudulent activities that fake accounts can engage in include posting harmful links, spamming mass follow and unfollow requests, repeatedly sharing unrelated links and ads, stealing personal information, as well as disrupting online communities by ways of trolling, hate speech and harassment. The goal of this thesis is to differentiate between fake and real Twitter accounts by leveraging user information and tweets.

1.2. Necessary Background

This section offers the necessary technical background within the context of this work. it presents an overview of the various methods explored, the relevant scientific fields and related terms, and how these methods are interrelated.

1.2.1. Artificial Intelligence (AI)

Artificial Intelligence (AI) is a branch of computer science concerned with Developing intelligent machines capable of executing tasks that traditionally need human intelligence, such as learning, problem-solving, reasoning, perception, natural language processing, and more. It involves developing algorithms and techniques allowing computers to analyze and process massive amounts of data, recognize patterns, and make predictions. AI has the potential to Advance and change many industries and fields. It is a rapidly evolving field that incorporate on a wide range of disciplines, including computer science, mathematics, statistics, psychology, neuroscience, and more.[5]–[7]

1.2.2. Machine learning (ML)

Machine learning is a discipline within the field of artificial intelligence that concentrate on making algorithms and statistical models that allows computers to learn from data and autonomously generate predictions or decisions. It is a method of data analysis that automates the process of building analytical models, by identifying patterns and relationships in the data. Machine learning systems learn from data without being explicitly programmed, it can be utilized for a wide range of tasks, from image and speech recognition to natural language processing and recommendation systems[8]–[10].

1.2.3. Supervised learning

Supervised learning is a type of ML in which an algorithm learns from labelled data consisting of input features and corresponding output labels. During training, the algorithm develops a mapping between the input features and output labels that can be used to make predictions on new, unseen data. Supervised learning involves developing models that can generalize to new examples by learning from the labelled data[11], [12].

1.2.4. Unsupervised learning(UL)

UL is a type of ML in which the algorithm learns from unlabeled data to uncover hidden structures, patterns, or relationships. The algorithm does not rely on explicit labels or human supervision and seeks to find similarities and differences among data points, cluster data points into groups, or reduce the dimensionality of the data. Unsupervised learning is used to discover insights in data and to gain a better understanding of the underlying structure of the data. It is used in a variety of applications, such as anomaly detection, customer segmentation, and image and text clustering[8]–[10].

1.2.5. Natural language processing (NLP)

NLP is a area of study that focuses on the interaction between humans and computers using natural language, such as text and speech. It involves the usage of computational methods to comprehend, examine, and produce human language. NLP uses machine learning, statistical modelling, and linguistic analysis to extract meaning and insights from natural language data. It is a subfield of AI that has a broad range of applications, including sentiment analysis, text classification, machine translation, speech recognition, and chatbots[13]–[15].

1.2.6. Data science

Data science is a cross-disciplinary domain that applies statistical and computational methods to extract insights and knowledge from data. It includes a broad range of activities, including data cleaning, integration, analysis, and data visualization, while aiming to identify patterns and trends in data and to make predictions based on those patterns. Data science involves the use of data to develop models and algorithms that can predict, classify, or cluster data, and to communicate those insights to stakeholders. [9]–[16].

1.2.7. Artificial neural network (ANN)

Artificial Neural Networks (ANN) refer to a collection of algorithms that draw inspiration from the human brain's structure and functions. They consist of multiple layers of interconnected nodes, with each node performing a simple computation. and it's capable of processing information by transmitting signals through the network. The training process of ANN is made by modifying weights of the connections between nodes in order to minimize the error between the predicted and actual outputs, using input-output pairs. These networks are used in various domains such as speech recognition, image processing, and pattern recognition.[8], [12], [17]

1.2.8. Data preprocessing

Data preprocessing is the process of preparing data for analysis, which involves several tasks. These tasks include removing duplicates, handling missing values (also called imputation), text vectorization, outlier detection, dimensionality reduction, scaling and normalization, data augmentation, and other similar tasks. These tasks goal is to convert raw data into a acceptable format that is suitable and ready for analysis[17]–[20].

1.2.9. Data normalization

Data normalization is a data preprocessing technique that entails converting the values of a feature into a standard range, which is typically between 0 and 1 or -1 and 1. The primary aim of data normalization is to enhance the accuracy and performance of machine learning models by guaranteeing that all features have equal importance in the analysis, and reducing the influence of variations in feature scales on the output of the model.[10], [21], [22]

1.2.10. Data augmentation

Data augmentation is a set of techniques used in data driven research domains where the data is scarce and there is a need increase the data size for better analysis. This process leads to artificially increase the size and diversity training data, particularly in cases of too limited dataset, by creating new synthetic examples from the original data.

A good and acceptable data augmentation method aims to reduce overfitting, improve model generalization, and enhance the robustness of the model to variations and noise in the input data.

The methods of data augmentation may differ depending on the type of data, e.g., the works described in [25]–[33] cover various techniques for different data.

Text data: NLP tasks such as text classification and named entity recognition, data augmentation techniques may include adding synonyms, swapping similar words, or inserting typos to increase the diversity of the training data.[23]–[25]

Time series data: For time series forecasting tasks, data augmentation techniques may include randomly shifting or scaling the time series, adding noise, or simulating missing data.[26], [27]

Audio data: For audio processing tasks such as speech recognition or music classification, data augmentation techniques may include changing the speed or pitch of the audio, adding background noise, or simulating audio artifacts such as pops and clicks.[28]

Tabular data: For structured data such as numerical or categorical features, data augmentation techniques can be classified under 3 main types of augmentation; Feature-level augmentation, Sample-level augmentation, and Model-based augmentation.[29], [30]

Image data: Data augmentation for image data involves applying a set of transformations to the original data, such as flipping, rotating, cropping, adding noise, or changing the brightness and contrast as well as many other advanced techniques.[31]–[33]

1.2.11. Word embedding

Word embedding is a technique utilized in natural language processing (NLP) to convert textual data into dense vectors of real numbers. Each word's position in the vector space represents its semantic meaning, allowing similar words to be grouped together in the vector space. This enables machine learning algorithms to analyze textual data effectively. Word embeddings have demonstrated their usefulness in several NLP tasks, including text classification, sentiment analysis, named entity recognition, and machine translation.[34],[35]

1.2.12. Deep learning (DL)

Deep learning is a specialized field of AI that utilize neural networks with multiple layers to analyze and learn from vast quantities of raw data. Its exceptional performance has

been demonstrated in a diverse range of applications, including computer vision, natural language processing, self-driving cars, and numerous other domains[12], [36].

1.2.13. Recurrent neural network (RNN)

RNNs are artificial neural networks that are specifically engineered for handling sequential input data, including but not limited to time-series data and natural language. They have a feedback mechanism which allows them to use information from previous time steps to inform their processing of the current input, which means that the output of each step is dependent on both the current input and the previous state of the network. The ability to process variable-length inputs and learn temporal patterns from sequential data makes RNNs a powerful tool for various applications in AI and ML.[12], [37], [38]

1.2.14. Long short-term memory (LSTM)

LSTM is a type of RNN architecture that allows for modeling long-term dependencies in sequential data by selectively remembering and forgetting past inputs. LSTMs can overcome traditional problems that RNNs encounter. LSTMs are used for a variety of tasks, including natural language processing, and time-series analysis.[12], [39], [40]

1.2.15. CrowdTurfing

"Crowdturfing" is a term used to describe the practice of creating an online buzz or manipulating online content, reviews, and opinions by paying or incentivizing a large group of people to promote a specific agenda or message. It can be seen as a malicious crowdsourcing.

Crowdturfing is a commonly used tactic for businesses, political campaigns, and other organizations.[41]

1.2.16. Support vector machine (SVM)

SVM are a class of ML algorithms that fall under the category of supervised learning and can perform regression or classification tasks. SVMs operates by identifying the hyperplane in a high-dimensional space that maximally separates different labellers of data. A hyperplane is selected in such a way that maximizes the margin between the closest data points belonging to different classes. It is commonly used for binary classification but can be also used for multi-class problem. SVM typically employs a kernel function that maps the data into a higher-dimensional space where a linear

classifier can be used to separate the data. The kernel function transform the original data to a new space where the separation between the classes is more apparent.[17], [42], [43]

1.3. Related Work

Social media researchers and platforms use various methods and techniques to detect fraudulent accounts.

Song et al. (2015) broadly categorized these methods into two types.

1. Techniques that focus on analyzing the traits and relationships of individual accounts (profile-based and graph-based methods).
2. Techniques that aim to detect organized activity by large groups of accounts (e.g., crowdturfing) [45].

In [46], With an accuracy of 84.5%, the authors have recognized 23 characteristics for categorizing Twitter spammer/non spammer accounts. [56] utilized a smaller set of 10 features, with less promising results.

Nazir et al.(2010), employ a SVM to classify fake profiles in a Facebook game. To do so, they extract 13 features for each game user[47].

Instead of using some individual profile-based features, other approaches relied on graph-based features in this classification task[48], [49].

Cao et al. (2011) relies on the observation that fake profiles tend to establish connections predominantly with other fake profiles rather than legitimate ones. As a result, a distinct separation exists between the subgraphs comprising fake and non-fake accounts in the graph[50].

While on the other hand. Boshmaf et al. (2016), claim the opposite; fake accounts do not exclusively associate with other fake accounts, and presents a new detection methodology centered around the analysis of features exhibited by victims accounts.[51]

Another approach used by researchers in this topic is the study of the change of user behavior, which may happen in cases like exploitation of a legitimate account that temporally acts maliciously. For example, anomalies behavior can be detected by monitoring some features such as timing and the origin of messages, language and message topic, URLs, use of direct interaction, geographical proximity[52] .

In the approaches trying to capture coordinated activities of large groups of accounts, the black market and fake accounts online sellers play a major role. with large segment of customers that may vary from celebrities, politicians, and cyber criminals. Stringhini et al. (2013) describes the characteristics of Twitter follower market, it argues that there is two major types of accounts who may follow a customer, either fake accounts or compromised accounts.[53]

Wang et al. (2012) shows the danger of crowdturfing in which workers are paid to express a false digital impression, they start first by describing the operational structure of such systems and how it can be used to build a whole fake campaign. the authors build a benign campaign on their own to show how easy and danger the process can be.[54]

Wang et al. (2014) conducted research on the feasibility of using machine learning techniques to identify crowdturfing campaigns, as well as the resilience of these methods against potential evasive tactics employed by malicious actors. According to their findings, traditional machine learning methods can effectively detect crowdturfing workers with an accuracy rate of 95-99%. However, these detection mechanisms can be circumvented if the workers modify their behavior[55].

Most of the works mentioned earlier require historical data as they heavily depend on extensive content or behavior. Therefore, they encounter a significant drawback of time delay, whereby the fake user may have already executed numerous malicious activities before being identified and eliminated.

To address this concern, Yuan et al. (2019) developed a system to detect fake accounts directly after registration. Their approach involves analyzing both user behavior and content to identify any suspicious patterns or activities. The methodology includes the extraction of synchronization-based and anomaly-based features [57].

The authors in [58] and [66] present a method for detecting social spammers by using social honeypots and machine learning. Initially, multiple honeypots were deployed to lure social spammers, and then a ML model is trained using features derived from the gathered data of these honeypots.

In their study, Crescia et al. (2015) examine various features used in detecting Twitter accounts, creating a dataset comprising both human and fake follower accounts. They then evaluate several machine learning classifiers using this dataset and introduce their

own classifier. The authors found that features based on the friends and followers of the account being investigated yielded the best results among all the analyzed features [59].

Another approach to tackling the issue of detecting spam on Twitter is to concentrate on the tweet itself instead of user information. Chen et al. (2015) proposed a method in which features were extracted from the content of the tweet, such as the presence of particular keywords, URLs, and hashtags, as well as tweet structure (e.g., length, capitalization, and punctuation usage). Additionally, the temporal pattern of tweets was analyzed as a feature, with the authors discovering that spam tweets are frequently posted in bursts, while genuine tweets are more evenly distributed over time. The authors utilized machine learning algorithms to discover patterns in these features and differentiate between spam and genuine tweets [60].

In[61], the authors suggested a method for performing sentiment analysis on Twitter data by utilizing the Naïve Bayes classifier. They used a dataset of 10,000 tweets collected from Twitter and manually categorized them as positive, negative, or neutral. Various features were extracted from the preprocessed data, such as specific words or phrases, and used to train the Naïve Bayes classifier. The results of the study indicated an accuracy of 80.5% for sentiment classification of Twitter data using this approach.

To tackle evasion techniques used by spammers, some works try to use more robust features; [62], [63] propose some hybrid frameworks, (Fazil et al. 2018) exploits users meta data features, graph based and text-based features, founded that Random Forest gives the finest performance. Similarly, (Yang et al. 2013) use 4 sets of features in their classification, including account-based, text-based, graph-based, and automation-based features.

Other approaches rely on DL techniques, (Wu et al. 2017) uses Word2Vec deep learning technique to preprocess their dataset and then assign a multi-dimensional vector to a tweet [64].

(Alom et al. 2020) developed 2 classifiers; a text-based classifier that only consider user tweet text. And a combined classifier that uses CNN to identify relevant features from high dimensional vector representation of tweet text and apply Dense layers on user meta data information. Overall, the combined model shows better results [65].

Shortcomings of the related work may be enumerated as follows:

1 - Narrow Focus: Many of the works discussed in the literature review target specific types of fake users or spammers within a narrow domain. This limited focus may not capture the full range of fraudulent activities on social media platforms.

Dependency on Historical Data: Several approaches heavily rely on historical data, either content-based or behavior-based, which poses a significant drawback of time delay. By the time the fake user is identified and eliminated, they may have already executed numerous malicious activities.

2 - Limited Robustness: Some of the detection methods can be evaded or circumvented if the fraudulent actors modify their behavior or tactics. This undermines the robustness of the detection mechanisms and raises concerns about their effectiveness in real-world scenarios.

3 - Costly Implementation: Certain approaches, such as graph-based methods, can be computationally expensive and require extensive resources for implementation. This can limit their applicability in large-scale settings.

4 - Lack of Comprehensive Features: Some studies focus on a limited set of features, either profile-based or tweet-based, without considering a comprehensive range of indicators. This may result in incomplete detection and overlook certain patterns or characteristics of fake accounts.

5 - Limited Generalizability: The performance of some detection methods may be limited to specific datasets or contexts, making it challenging to generalize their effectiveness to different social media platforms or scenarios.

6 - Lack of Comparative Analysis: Some works present results without a comparative analysis of different methods, making it difficult to assess the relative performance and strengths of each approach.

7 - Incomplete Coverage of Spam Detection: While some studies concentrate on fake user detection, there is a lack of comprehensive coverage of spam detection in the context of Twitter, with limited focus on analyzing tweet content and sentiment analysis.

Overall, the shortcomings of the related work include limited scope, dependency on historical data, potential evasion by fraudulent actors, costly implementation, lack of comprehensive features, limited generalizability, lack of comparative analysis, and

incomplete coverage of spam detection. Addressing these limitations would contribute to more effective and robust methods for detecting fraudulent accounts and spam on social media platforms.

1.4. Overview Of The Proposed Solution

The objective of this thesis is to produce a cost-effective and computationally efficient model that can identify various types of fake accounts and spam by solely focusing on essential features for detection, and training on a general dataset. To achieve this, various deep learning models were studied and implemented to address the classification problem. Literature review reveals that most available models either deal with user data or tweet text, but very few consider both together. Mixing meta data and tweet text together has been found to improve classification accuracy, as claimed in reference [65].

This work involves experimenting with different models and implementations, starting from a few dense layers that only takes user meta data as input, to more complex variations of two-branched models that consider both user meta data and tweet text. Additionally, a new dataset is created by combining multiple labeled datasets that represent a wide range of contexts and domains. Our results show that the two-branched model using Dense and LSTM layers performs reasonably well.

Further details on the model and data collection will be provided in Chapter 2.

1.5. Objectives And Contributions Of The Thesis

The objectives and contribution of this thesis can be concisely summarized in the following key points:

1. Main objective of the thesis is to identify fake Twitter accounts using user metadata, such as followers and following numbers, and the tweets posted over time.
2. Combining Numeric and Text Data: The thesis proposes a novel approach that combines both numeric and text data to detect fake Twitter accounts. This integration of different types of data provides a more comprehensive analysis.
3. Two-Branch Deep Learning Model: To achieve the objective, a two-branch deep learning model is implemented. One branch utilizes dense layers, while the other

branch employs LSTM (Long Short-Term Memory) layers. This architecture permits the model to discover relevant patterns and features from the mixed data.

4. Evaluation of Traditional and Simpler Models: Initially, the thesis explores traditional and simpler models to assess their performance on the given type of data. This step helps establish a baseline and provides a comparison point for the effectiveness of the proposed approach.
5. Data Generation and Augmentation: In order to train and evaluate the two-branch-DL model, two real-world Twitter datasets are collected, merged, and augmented. This step ensures a diverse and representative dataset for model training and testing.
6. Model Comparison: To ensure a comprehensive comparison, all implemented and trained models underwent evaluation and testing on unseen data using various metrics and indicators. These metrics include model accuracy and loss, confusion matrix, precision, recall, F1 and F2 scores, and Matthews Correlation Coefficient (MCC). By considering these metrics, a thorough assessment of the models' performance can be obtained.

Overall, this thesis contributes by using a novel approach that combines numeric and text data for identifying fake Twitter accounts. It also demonstrates the effectiveness of a two-branch deep learning model in handling mixed data and achieving moderate success in detecting fake accounts. The utilization of real-world Twitter datasets and the evaluation of traditional models provide valuable insights into the challenges and potential solutions in this domain.

1.6. Organization Of The Thesis

This Thesis is made out of five chapters: Introduction, Theoretical part, Experimental part, Result and Discussion, Conclusion and Future work.

Chapter 1 provides an introduction to the thesis, with an overview on social media, its effects, evolvement and uses. This overview is expanded in more details in the following 5 subsections, which include a problem definition and motivation, necessary background information, related work, an overview of the proposed solution, objectives and contributions of the thesis, and its organization.

Chapter 2 is the theoretical part of the thesis, describes the components of the developed methodology, the data sets used in the study, and the detailed proposed method.

Chapter 3 covers the experimental part of the thesis and includes information about the test environments, testing plan, and various cases that will be tested.

Chapter 4 presents the results and discussion of the experiments, including experimental results from different datasets and test cases, presented as graphs or tables, and a detailed analysis of the results.

Chapter 5 concludes the thesis with a summary of the lessons learned from this study, future work that can be done based on the completed work, and recommendations for further research. The bibliography, appendix, and curriculum vitae are included at the end of the thesis.

CHAPTER 2

2. THEORETICAL PART

The opening of this chapter discusses the constituent parts of the methodology created, presented in section 2.1. Subsequently, section 2.2 delves into the collection, processing, and augmentation of the dataset. The chapter concludes with section 2.3, which outlines the proposed methodology in detail and identifies the primary models employed and evaluated in this investigation.

2.1. Components Of The Developed Methodology

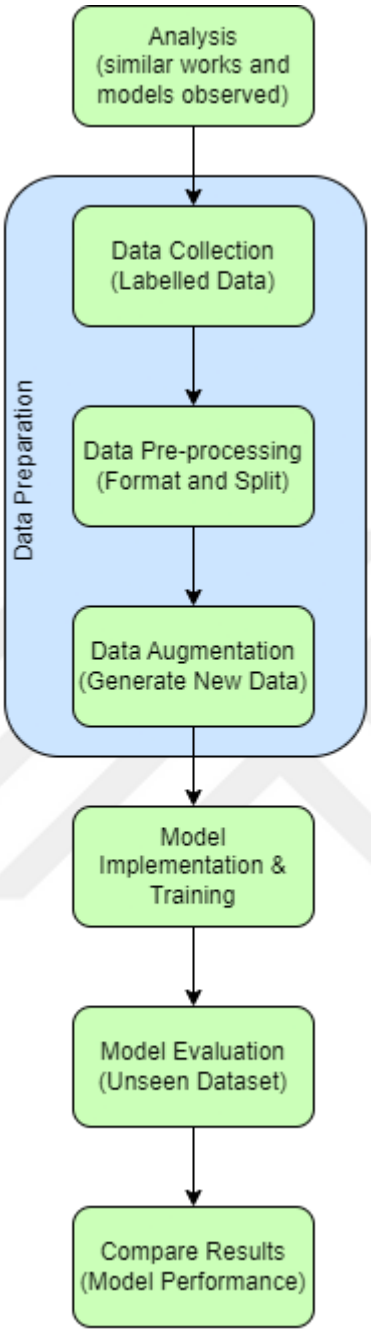


Figure 2.1: Block diagram of the main components of the developed methodology

The methodology developed for this study involves trialing various models, as illustrated in **Figure 2.1**, which shows all the components of the methodology. The process consists of several stages, including analysis, data collection, data preprocessing, data augmentation, model implementation and training, model evaluation, and result comparison. During analysis, similar works and models are observed, and a few of them are selected for implementation and testing. Labelled data is then collected from different

resources and organized during data collection. Data preprocessing involves performing all the necessary operations to format and shape the data for use in the model, including dividing the dataset into test and train sets. Data augmentation is used to generate new data points that do not exist in the original dataset and can simulate real-world data from various scenarios. The model implementation and training stage involves setting up the environment, importing the necessary libraries and frameworks, and writing the code to build the model. Hyperparameters are selected, and the training process is initiated using the training set. Model evaluation is then conducted to assess the model's performance on data that it has not been trained on before (i.e., the test set). Finally, if the model fits, achieved results and scores are recorded and compared with other implementations.

2.2. The Data Sets

Data utilized in this study were gathered from multiple sources, including some that were augmented based on the obtained ones.

In summary, this work utilized two types of datasets for model training:

1- unbalanced dataset: The unbalanced dataset (**Figure 2.2**) consists of 2,614,000 real users labeled as 0 and 187,329 fake users labeled as 1. This dataset was obtained before any balancing process was applied. While it is generally not advisable to use such a dataset with a significant disparity in the number of instances per class as it may result in subpar performance, it can be utilized in certain situations to study the influence of utilizing both balanced and unbalanced datasets on the same model.

```
# real legit users =>0    /   fake => 1
data["outputLabel"].value_counts()
```

```
0    2614000
1     187329
Name: outputLabel, dtype: int64
```

Figure 2.2: label distribution in the unbalanced dataset

2- A balanced dataset containing 374,658 records, with 187,329 instances in each class. This dataset was divided into train and test sets of 299,726 and 74,932 records, respectively. Following that, the dataset was augmented into a larger one with a final balanced dataset of 474,658 record divided into 369,726 training instance, and 104,932 test record. **Table 2.1** demonstrates the distribution of the final utilized dataset in this study.

Table 2.1: distribution of the dataset used in the study

	Train samples		**Test samples**	
Processed dataset	299726		74932	
Augmented dataset	70000		30000	
Final dataset (Sum)	369726		104932	
	Labels		Labels	
	(0)184826	(1)184900	(0) 52503	(1) 52429

To encompass a wide range of subjects, and locations, multiple sources were utilized in compiling these datasets. The following is a selection of the datasets incorporated, along with their corresponding contexts:

2.2.1. Data collection

2.2.1.1. Dataset1 Fame for sale: efficient detection of fake Twitter followers [59]

In this work, a dataset of both verified humans and Fake followers accounts was created, the dataset consists of 5 subfolders, each of which consists of 4 CSV files: followers.csv, tweets.csv, friends.csv, and users.csv. This dataset is available at: http://mib.projects.iit.cnr.it/dataset.html.

TFP and E13 represent real human accounts and the rest 3 datasets TWT, INT, and FSF represent Fake followers. **Table 2.2** shows some statistics about this dataset.

Table 2.2: statistics of dataset1. Adapted from [59]

		relationships		
dataset	tweets	followers	friends	total
TFP(@TheFakeProject)	563,693	258,494	241,710	500,204
E13 (#elezioni2013)	2,068,037	1,526,944	667,225	2,194,169
FSF(fastfollowerz)	22,910	11,893	253,026	264,919
INT(intertwitter)	58,925,	23,173	517,485	540,658
TWT(twittertechnology)	114,192	28,588	729,839	758,427
HUM (human dataset)	2,631,730	1,785,438	908,935	2,694,373
FAK (fake dataset)	118,327	34,553	879,580	914,133
BAS(baseline dataset: HUM U FAK)	2,750,057	1,819,991	1,788,515	3,608,506

a. The Fake Project(TFP)

   TheFakeProject is a research initiative started in 2012 by IIT-CNR researchers, who created a Twitter account @TheFakeProject and collected public information using Twitter APIs. They obtained 616,193 tweets and 971,649 relationships. resulting in 469 "certified humans." The dataset is called TFP.

b. Elezioni2013 dataset(E13)

   The #elezioni2013 dataset was created for a sociological study on the changes in Italian politics from 2013 to 2015. 84,033 unique Twitter accounts that utilized the hashtag were identified and accounts belonging to politicians, parties, journalists, and bloggers were discarded, leaving about 40k accounts classified as citizens. A sample of 1488 accounts was manually verified by analyzing profile pictures, biographies, and timelines, and 1481 accounts were included in the dataset.

c. Intertwitter dataset (INT)

   In April 2013, 1337 fake accounts were bought and crawled from http://intertwitter.com, to build the intertwitter dataset labelled as INT

d. Fastfollowerz dataset (FSF)

   In April, 2013, 1169 fake accounts were bought and crawled from http://fastfollowerz.com, to build the fastfollowerz dataset labelled FSF

e. Twittertechnology dataset (TWT)

   In April 2013, 1000 fake accounts were bought from http://twittertechnology.com. Almost instantly, 155 of them got suspended, and the remaining 845 were crawled in order to build the twittertechnology dataset labelled as TWT

As mentioned previously each of these subfolders contains 4 CSV files, note that the 5 main folders for each dataset contain the same file architecture and attributes.

Considering Twittertechnology dataset (TWT), the attributes in each file are as follows:

1- users.csv:

contain 34 attributes listed in **Figure 2.2**

```
TWT_users.columns
```

```
Index(['id', 'name', 'screen_name', 'statuses_count', 'followers_count',
       'friends_count', 'favourites_count', 'listed_count', 'created_at',
       'url', 'lang', 'time_zone', 'location', 'default_profile',
       'default_profile_image', 'geo_enabled', 'profile_image_url',
       'profile_banner_url', 'profile_use_background_image',
       'profile_background_image_url_https', 'profile_text_color',
       'profile_image_url_https', 'profile_sidebar_border_color',
       'profile_background_tile', 'profile_sidebar_fill_color',
       'profile_background_image_url', 'profile_background_color',
       'profile_link_color', 'utc_offset', 'protected', 'verified',
       'description', 'updated', 'dataset'],
      dtype='object')
```

Figure 2.2: attributes in users.csv

2- followers.csv:

contains 2 attributes which are source id and target id as shown **Figure 2.3**

```
TWT_followers.columns
```

```
Index(['source_id', 'target_id'], dtype='object')
```

Figure 2.3: attributes in followers.csv

3- tweets.csv:

contains 19 attributes that are listed in **Figure 2.4**

```
TWT_tweets.columns
```

```
Index(['created_at', 'id', 'text', 'source', 'user_id', 'truncated',
       'in_reply_to_status_id', 'in_reply_to_user_id',
       'in_reply_to_screen_name', 'retweeted_status_id', 'geo', 'place',
       'retweet_count', 'reply_count', 'favorite_count', 'num_hashtags',
       'num_urls', 'num_mentions', 'timestamp'],
      dtype='object')
```

Figure 2.4: attributes in tweets.csv

4- friends.csv:

contains 2 attributes as shown in **Figure 2.5**: attributes in followers.csv which are source id and target id

```
TWT_friends.columns
```

```
Index(['source_id', 'target_id'], dtype='object')
```

Figure 2.5: attributes in followers.csv

2.2.1.2. dataset2: Social Honeypot Dataset [66]

A dataset of social honeypots was gathered from Twitter over a period of seven months and used in a study called "Seven Months with the : A Long-Term Study of Content Polluters on Twitter"[66]. The dataset includes 22,223 content polluters and their followers numbers, 2,353,473 tweets, and 19,276 legitimate users and their number of followers, along with 3,259,693 tweets.

As mentioned in the dataset Readme File, it consists of six text files:

1- “content polluters.txt”
2- “content polluters followings.txt”
3- “content polluters tweets.txt”
4- “legitimate users.txt”
5- “legitimate users followings.txt”
6- “legitimate users tweets.txt”

As the dataset's features differ from the first one, only the tweets from this dataset will be utilized. In accordance with section 2.2.3, "legitimate users tweets.txt" and "content polluters tweets.txt" are the two essential files for the augmentation process, as they contain the tweets to be employed in the process.

Figure 2.6 and **Figure 2.7** are samples of both files after being imported in python notebook as dataframes.

```
Normal_tweets = pd.read_csv("legitimate_users_tweets.txt",sep="\t", header=None)
Normal_tweets.columns =['UserID', 'TweetID', 'Tweet', 'CreatedAt'] # assign names for each column
Normal_tweets.set_index('UserID',inplace =True)
Normal_tweets #print dataframe
```

UserID	TweetID	Tweet	CreatedAt
614	5912305459	... at house party in Daybreak. Not as weird as ...	2009-11-20 23:52:52
614	5908467165	Taxiing — at SLC Salt Lake City International...	2009-11-20 20:42:48
614	5904901963	Almost home! — at PDX Portland International ...	2009-11-20 18:11:01
614	5900351610	Lunch! — at Ten01 http://gowal.la/s/Awa	2009-11-20 15:04:42
614	5900312627	Mm ... books — at @Powells http://gowal.la/s/6fe	2009-11-20 15:03:06
...	...	...	...
93390990	6168916131	meet me on ELIMINATE pro!!	2009-11-29 06:44:03
93402679	6170059145	my twitter	2009-11-29 07:52:22
93419256	6171947104	exploring this thing...	2009-11-29 09:25:53
93426370	6172856187	learning the rap ice ice baby LOL, not very gd...	2009-11-29 10:08:09
93442002	6174743151	Clipse, B.o.B и J.Cole	2009-11-29 11:33:06

3246377 rows × 3 columns

Figure 2.6: Sample of legitimate users tweets.txt

```
spam_tweets = pd.read_csv("content_polluters_tweets.txt",sep="\t", header=None)
spam_tweets.columns =['UserID', 'TweetID', 'Tweet', 'CreatedAt'] # assign names for each column
spam_tweets.set_index('UserID',inplace =True)
spam_tweets # print dataframe
```

UserID	TweetID	Tweet	CreatedAt
6301	5599519501	MELBOURNE ENQUIRY: Seeking a variety of acts f...	2009-11-10 15:14:31
6301	5600313663	THE BURLESQUE BOOTCAMP SYDNEY - Open Date tick...	2009-11-10 15:46:05
6301	5600328557	THE BURLESQUE BOOTCAMP SYDNEY - Open Date tick...	2009-11-10 15:46:40
6301	5600338093	THE BURLESQUE BOOTCAMP SYDNEY - Open Date tick...	2009-11-10 15:47:03
6301	5600564863	Come to "The Burlesque Bootcamp - Sydney" Satu...	2009-11-10 15:56:03
...	...	...	...
173723395	20121945699	1000 twitter followers: http://bit.ly/s3oi?=Xg...	2010-08-02 02:20:49
173723395	20122086703	2000 new visitors to your blog: http://bit.ly/...	2010-08-02 02:24:23
173723395	20122438261	1000s twitter followers: http://bit.ly/s3oi?=r...	2010-08-02 02:33:11
173723395	20122572401	How 2 create fabulous blog topics: http://bit....	2010-08-02 02:36:40
173766965	20125997053	Bord as hell....sumbody PLEASE hmu..please	2010-08-02 04:03:26

2333691 rows × 3 columns

Figure 2.7: Sample of content polluters tweets.txt

2.2.2. Data preprocessing

As various models were utilized and evaluated, the approach to data preparation differed. For instance, certain models solely required an 8-dimensional vector of user metadata as input, whereas another model necessitated a 308-dimensional vector combining tweet and text representations. As a result, two distinct formats were created.

2.2.2.1. format 1: user meta data

for models that require only some user information (meta data); and based on the feature analysis made by[59]. 7 features were selected, and 1 extra feature was extracted from them. These features include the following: statuses count, followers count, friends count, favorites count, listed count, URL, time zone. And the last feature that was extracted from them is "user_score". **E 2.1** shows how user score "score" is calculated for each record in the dataset.

$$score = \left(\frac{f1[i]}{f1.max}\right) - \left(\frac{f2[i]}{f2.\max}\right) + \left(\frac{f3[i]}{f3.max}\right) \qquad (\mathbf{2.1})$$

Such that:

score: user_score

i: record number inside the Pandas DataFrame

max: maximum value of a feature in the whole DataFrame

f1: followers_count

f2: friends_count

f3: number_of_tweets

Format 1 is essentially a subset of Format 2 described in the following section. Therefore, once the data is formatted as Format 2, it is sliced to extract only the first eight pieces of user data and the corresponding record label, while eliminating any duplicates.

2.2.2.2. Format 2: user meta data + user tweet

For models that require both user information and tweet text, the same 8 previous features are utilized, along with an added vector representation for each tweet. To vectorize the tweets, a pre-trained tool called Spacy[67] is utilized to make a 300-dimensional vector representation for each tweet.

Note that the dataset was prepared in batches, owing to constraints on memory and disk space.

Overall, the process of preparing the data in this format involves the following steps:

1- Import libraries such as NumPy and Pandas

2- loading the data into the Python notebook (**Figure 2.8**)

```
import pandas as pd
import numpy as np
TWT_users = pd.read_csv("users.csv")
TWT_followers = pd.read_csv("followers.csv")
TWT_friends = pd.read_csv("friends.csv")
TWT_tweets = pd.read_csv("tweets.csv", encoding = "ISO-8859-1")
```

Figure 2.8: load dataset

3- create an empty DataFrame, this will be the new dataset(**Figure 2.9**).

```
df = pd.DataFrame(index=range(numRows),columns=['id', 'statuses_count', 'followers_count',
       'friends_count', 'favourites_count', 'listed_count',
       'url', 'time_zone', 'dataset','tweet_text'])
```

Figure 2.9: create new DataFrame

By copying only, the required values to it from the users.csv file, including user ID, statuses count, followers count, friends count, favourites count, listed count, URL, time zone, and dataset. Then, using the user ID, a loop is executed to search for a match with the user ID in Tweets.csv. Each time a match is

found, the previous nine values of the record are duplicated into a new record, followed by the tweet.

4- Mapping URL and time zone to numeric values
5- normalizing all columns values between 0 and 1
by dividing each feature by its highest possible value in the whole dataset.
6- Adding a score column and calculating it.
7- Drop empty rows. And replacing the remaining Nan with 0
Check if there is a whole row empty and remove it(**Figure 2.10)**.

```
df = df.dropna(how='all')
```

Figure 2.10: remove empty records

8- Checking for duplicates in the records and removing them(**Figure 2.11**).

```
df = df.drop_duplicates()
```

Figure 2.11: remove duplicate records

9- Assign an output label to each DataFrame.
- legit accounts records have the label = 0
- fake account records have the label = 1

10- check for balanced classes in the dataset(**Figure 2.12**), by randomly sampling an equal number of samples from each label.

```
zeros = data.query("outputLabel == 0").sample(n = 187329)
```

Figure 2.12: balanced classes

11- splitting the dataset into train and test samples.

2.2.3. Data augmentation

To create a comprehensive model capable of detecting diverse types of spammers and fraudulent accounts, it is essential to train it on a wide range of data that reflects various real-world contexts and scenarios. This comes in contrast to E13 dataset for example, which was designed exclusively for political contexts.

To achieve this objective, a new dataset was created to emulate real-world fuzziness by utilizing not only the original dataset[59] but also randomly chosen tweets from the Social Honeypot Dataset[60].

	statuses_count	followers_count	user_score	friends_count	favourites_count	listed_count	url	time_zone
count	187329.000000	187329.000000	187329.000000	187329.000000	187329.000000	187329.000000	187329.000000	187329.000000
mean	0.063013	0.002844	-0.025771	0.028615	0.018721	0.022940	0.452973	0.836891
std	0.096688	0.026511	0.062525	0.061012	0.061069	0.087064	0.497785	0.369466
min	0.000027	0.000000	-0.935752	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.014713	0.000279	-0.030051	0.008315	0.000406	0.001316	0.000000	1.000000
50%	0.031403	0.000627	-0.014063	0.015118	0.002210	0.002632	0.000000	1.000000
75%	0.063580	0.001447	-0.007636	0.032251	0.010395	0.011842	1.000000	1.000000
max	1.000000	1.000000	0.991853	1.000000	1.000000	1.000000	1.000000	1.000000

Figure 2.13: statistics about real accounts class(0) before augmentation

	statuses_count	followers_count	user_score	friends_count	favourites_count	listed_count	url	time_zone
count	187329.000000	187329.000000	187329.000000	187329.000000	187329.000000	187329.000000	187329.000000	187329.000000
mean	0.014691	0.000265	-0.025033	0.025298	0.001674	0.001423	0.164219	0.438443
std	0.039392	0.000544	0.014620	0.014872	0.005607	0.005040	0.370475	0.496198
min	0.000000	0.000000	-0.083786	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000356	0.000032	-0.029708	0.014614	0.000000	0.000000	0.000000	0.000000
50%	0.002547	0.000056	-0.023043	0.023517	0.000000	0.000000	0.000000	0.000000
75%	0.014036	0.000240	-0.014565	0.029858	0.000271	0.001316	0.000000	1.000000
max	0.295040	0.003977	0.000000	0.084156	0.031613	0.042105	1.000000	1.000000

Figure 2.14: statistics about Fake accounts class(1) before augmentation

The process starts by analyzing each class of the created dataset. **Figure 2.13** and **Figure 2.14** demonstrate some basic statistics in both labels 0 and 1, respectively. The augmented dataset consists of 100K records (50k for each label)

The steps of the augmentation process for each label are:

1- Getting the minimum and maximum value for each feature we have
2- Generate a random value that lies between these 2 values (randomly select between 0 and 1 for features with either 0/1 value)
3- For the text vector representation, the same process of transforming the tweet to a 300 vector has been used on some tweets from dataset 2 [66].

2.3. Details Of The Proposed Method

As shown in section 2.1, different models and implementations were experimented with and adjusted. The approach was influenced by the methodology described in reference [65]. The objective was to determine which approach would yield superior results in the classification task: utilizing only user metadata or incorporating both metadata and the text of user tweets.

To accomplish this, the dataset was gathered and preprocessed in accordance with section 2.2. Next, models inspired by relevant prior research were developed and trained on the

dataset. Finally, the performance of the models was compared and evaluated based on accuracy and confusion matrix metrics. 2 main architectures were used:

2.3.1. One branched models

These models follow a single-branch architecture where all the data undergoes sequential layer processing. Such models can perform classification tasks by taking inputs such as user information or tweet text. In this study, only user data was utilized in the one-branch models.

2.3.2. Two branched models

This refers to models that employ two parallel branches of layers, with each branch receiving only a specific segment of data per record. One branch is responsible for processing user metadata, while the other processes the user's tweet text representation. Before the output layer, these two branches are merged together.

In Chapter 3, every implemented model architecture will be showcased in detail, along with various applied variations and the training process. Following that, chapter 4 will present the results obtained for each test case.

CHAPTER 3

3. EXPERIMENTAL PART

3.1. Test Environment

Data preparations and model implementation and training were conducted on different machines.

Data pre-processing machine :

For data preparation, the process was made on a virtual machine cluster provided by the university, equipped with the following specifications:

- Ubuntu Linux – 64-bit architecture
- 96 GB Ram
- 24 CPUs : Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz

Training & testing machine :

Instead of using a local workstation for the training and testing, Google Colab—a Google tool that functions as a hosted Jupiter notebook—was used. It is appropriate for ML, DL, and data analytic activities, and allows the writing and execution of Python scripts and codes through the browser. Depending on the resources required, Google Colab[68] offers a number of subscription levels that are both free and premium.

Colab provides access to various runtime options and resources, the most basic resources are as follows:

CPU, GPU, and TPU:

Depending on availability, Colab offers access to a virtual machine with either a CPU (central processing unit), GPU (graphics processing unit), or TPU (tensor processing unit). The GPU is an Nvidia Tesla K80 with 12GB of video RAM, while the CPU is an Intel Xeon 2-core.

Users of Colab's free edition may have to wait in queue to access the TPU, which is a specially created ASIC (application-specific integrated circuit) built for deep learning tasks.

RAM:

The type of virtual machine determines how much RAM is accessible. The GPU virtual computer has 15 GB of RAM, compared to 13GB for the CPU virtual machine.

In the case of the TPU, it offers 64 GB of high bandwidth memory Ram.

Storage:

Colab provides temporary storage space that is only available during a session. The amount of storage space provided is typically around 100GB, but it can vary. However colab allow to read and write from different other storage locations including google drive, Microsoft OneDrive and local storage and many other.

Software Libraries:

TensorFlow[69], PyTorch,[70], Keras[71], Pandas[72] and Numpy[73] are just a few of the well-known Python libraries that come pre-installed in Colab, making it simple to start working on projects involving deep learning and machine learning.

3.2 Testing Plan

In this study, a total of 7 different model architectures were implemented, comprising 4 One Branched architectures and 3 Two Branched architectures. 4 different optimizers were used on each architecture: Stochastic Gradient Descent (SGD)[77], RMSprop[78], Adam[79], and Adadelta optimizer[80]. In general, optimizers aim to adjust the weights and biases of the neural network during training in order to minimize the loss function and enhance the accuracy of the model's predictions. Although the specific update rules may vary, the ultimate goal remains the same - to optimize the model's performance[12][76].

The training process for all models was conducted over 200 epochs, utilizing a batch size of 64. The initial learning rate was set to 0.015, except when using a learning rate reducer where the initial value was 0.001 and 0.015.

In addition to That, Keras early stopping was implemented as a part of the training process[81], which stops the training when a monitored metric shows no improvement.

Figure 3.1 illustrates the application of early stopping in this research, where the validation loss was tracked during the training process with patience of 7 epochs. This way can help to prevent overfitting by ensuring that the model does not continue to specialize too much to the training data and perform poorly on unseen data. And encourages the model to learn more generalizable features that are likely to perform well on new, unseen data.

```
early_stopping = EarlyStopping(monitor='val_loss', patience=7)
```

Figure 3.1: Early stopping Callback

Similarly, to enhance the convergence of the models, a learning rate reducer was applied twice on each architecture using the SGD optimizer. This could be accomplished through the utilization of the ReduceLROnPlateau function in Keras[82], which automatically decreases the learning rate during training when a specific metric ceases to improve. **Figure 3.2** shows the applied reducer, which monitors the validation loss, and reduces the learning rate by a factor of 0.1 if no improvement is observed for 5 consecutive epochs.

```
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.1, patience=5,
                              verbose=1, min_delta=1e-4, mode='min', cooldown=2, min_lr=1e-7)
```

Figure 3.2: learning rate reducer

By employing ReduceLROnPlateau, the optimization process becomes more efficient as the learning rate is dynamically adjusted based on the model's performance. This approach facilitates improved convergence and aids in achieving better generalization of the models.

In order to assess these models, a range of metrics was utilized, including loss, accuracy, area under the curve(AUC), confusion matrix, precision, recall, specificity, F1 and F2 scores, and Matthews Correlation Coefficient (MCC)[83][84]. These metrics provide comprehensive insights into the model's performance and help assess its effectiveness in classification tasks.

The importance of each of these metrics lies in its definition and its role in evaluating the performance of binary classification models. Here is a simplified description of each metric.

1 - loss : measures the difference between the predicted output and the actual target output. In this work, binary cross entropy loss (**E 3.1**) is used for this binary classification task where y the true binary label (0 or 1), and p represents the predicted probability of the positive class for that instance.

$$\text{Binary Cross Entropy Loss} = -[y * \log(p) + (1 - y) * \log(1 - p)] \qquad (\mathbf{3.1})$$

2 – accuracy: is the proportion of correctly predicted instances out of the total number of instances in a dataset as shown in **E 3.2**.

$$\text{Accuracy} = \frac{\text{number of correctly predicted instance}}{\text{total number of instances}} \qquad (\mathbf{3.2})$$

3 – Area Under the Curve (AUC): measures the overall quality of the model's predictions across different classification thresholds, it is calculated based on the Receiver Operating Characteristic curve, which can be obtained by plotting the True Positive Rate (TPR) versus the False Positive Rate (FPR) at various classification thresholds.

$$AUC = \sum [(TPR[i] + TPR[i+1]) / 2] * (FPR[i+1] - FPR[i]) \qquad \mathbf{(3.3)}$$

E 3.3 illustrates how to calculate the AUC of a model, where TPR[i] represents the True Positive Rate (Sensitivity) at the i-th threshold value, and FPR[i] represents the False Positive Rate (1 - Specificity) at the i-th threshold value.

The AUC value would range from 0 to 1, where a higher value indicates better classification performance. 0.5 represents a random classifier, while an AUC of 1 suggests an ideal classifier.

4 – confusion matrix: is a measure that provides a concise representation of how well a classification model performs on a given test dataset. It shows the model's predictions compared to the actual true labels for each class. In a binary classification scenario, a confusion matrix(**Figure 3.3**) consists of four elements:

True Positive (TP): Instances that the model correctly identifies as positive.

True Negative (TN): Instances that the model correctly identifies as negative.

False Positive (FP): Negative instances that the model incorrectly identifies as positive.

False Negative (FN): Positive instances that the model incorrectly identifies as negative.

These elements are later used to measure various evaluations metrics.

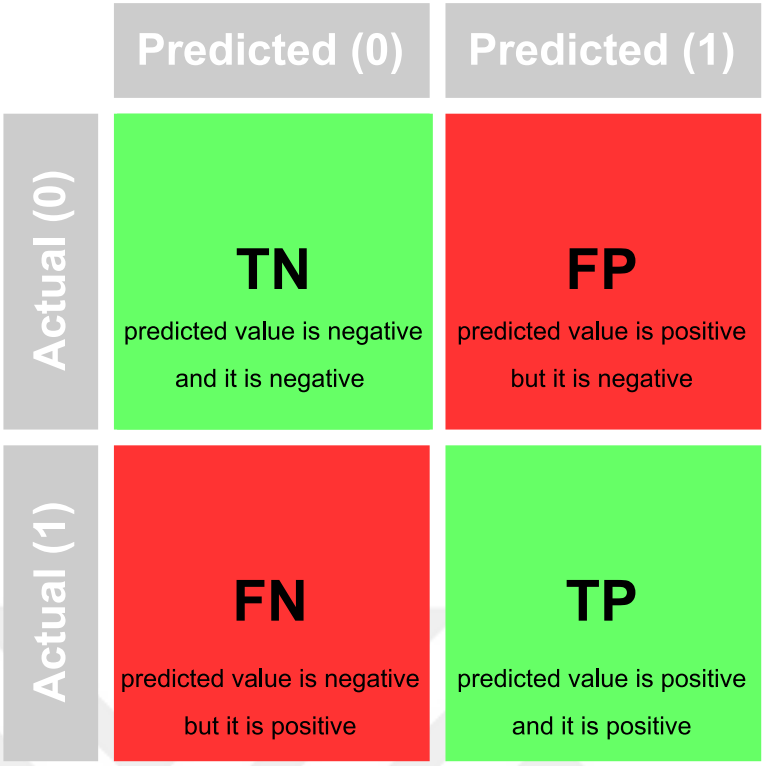


Figure 3.3: Confusion matrix

5 – precision: as described in **E 3.4** measures the proportion of correctly predicted positive instances out of all instances predicted as positive by the model.

$$Precision = \frac{TP}{TP+FP} \quad (\mathbf{3.4})$$

6 – Recall(sensitivity):**E 3.5**, measures the proportion of correctly predicted positive instances out of all actual positive instances in the dataset.

$$Recall = \frac{TP}{TP+FN} \quad (\mathbf{3.5})$$

7 – Specificity: measures the proportion of actual negative cases that are correctly identified by the model out of all the cases that the model predicted as negative as illustrated in **E 3.6**.

$$specificity = \frac{TN}{TN+FP} \quad (\mathbf{3.6})$$

It provides an indication of how well a model can avoid false positives and correctly identify true negative cases. The value of specificity ranges from 0 to 1, with a higher value indicating better performance in identifying negative cases.

8 - F1 Score: combines the precision and recall into a single value, while provides a balance between the two metrics. It's calculated as shown in **(E 3.7)**. Ranging between 0 and 1, where a higher value indicates better performance. An elevated F1 score shows that the model has reached a good balance between precision and recall, reliably predicting positive cases while minimizing false positives and false negatives.

$$F1\ score = \frac{2*(\text{precision} * \text{recall})}{\text{precision} + \text{recall}} \quad (\mathbf{3.7})$$

9 -F2 Score: also combines precision and recall, while giving more weight to recall. It can be calculated as mentioned in **(E 3.8)**, where β is a parameter that controls the weight given to recall. In the case of F2, β will be set to 2. Similarly, F1 score is obtained by setting β to 1.

$$F.\beta\ score = \frac{(1 + \beta^2)*(\text{Precision} * \text{Recall})}{\beta^2 * \text{Precision} + \text{Recall}} \quad (\mathbf{3.8})$$

F2 score favors models that achieve high recall while still maintaining a reasonable level of precision. It is especially useful in imbalanced datasets or scenarios where false negatives are more costly or harmful than false positives.

10 – Matthews Correlation Coefficient (MCC): also evaluates the performance of binary classification models. It considers all the 4 elements of confusion matrix as described in **(E 3.9).** MCC is especially useful when handling imbalanced datasets, as it considers both the false positives and false negatives. It is a reliable metric for evaluating model performance, especially in situations where there is a significant class imbalance or when false positives and false negatives have different impacts or costs.

MCC value varies between -1 to 1; where 1 refers to a perfect classifier, while 0 represent a random classifier, and -1 refers to a complete incorrect classifier.

$$MCC = \frac{\text{TP} * \text{TN} - \text{FP} * \text{FN}}{\sqrt{(TP + FP) * (TP + FN) * (TN + FP) * (TN + FN)}} \quad (\mathbf{3.9})$$

In this Chapter, a comprehensive overview of each implemented model is presented. The results of these models are subsequently referred to by their respective model names in Chapter 4. All models discussed in this thesis, along with additional ones that were not included, have been saved on WandB and Colab Notebook. The saved records encompass the model codes, training processes, logs and corresponding plots [74][75].

The one-branched architecture models follow the naming convention of (F-number-optimizer name), while the two-branched models are referred to as (2B-number-optimizer name). For the models that incorporate a learning rate reducer, they are denoted as (model name-optimizer-lr(lr_value)).

3.2.1 F-1 model

The implemented model described here is the simplest one in this work, comprising of only 2 Dense layers and Dropout. Dense layers, also known as fully connected layers, connect every neuron to all neurons in the previous and subsequent layers, allowing for the learning of non-linear relationships in data[12][89]. Dropout is a technique that randomly selects and deactivates nodes in the network during each training iteration. The selected nodes' outputs are set to zero, which helps prevent overfitting and promotes a more robust and generalized model [87].

F-1 model has a total of 321 trainable parameters. **Figure 3.4** and **Figure 3.5** demonstrate its architecture and provide a summary of the model.

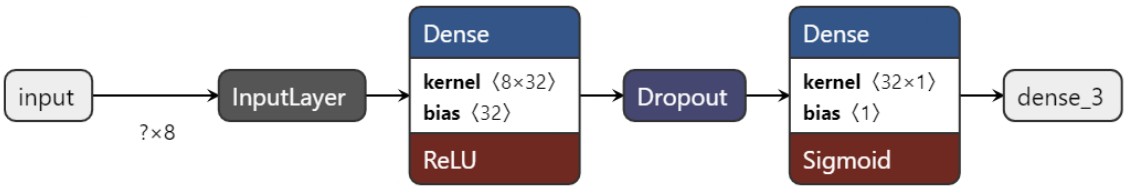


Figure 3.4: F-1 model architecture

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 32)	288
dropout (Dropout)	(None, 32)	0
dense_1 (Dense)	(None, 1)	33

Total params: 321
Trainable params: 321
Non-trainable params: 0

Figure 3.5: F-1 model summary

3.2.2 F-2 model

This model is more complex, featuring 7 Dense layers with a greater number of nodes. It consists of a total of 6,041 trainable parameters. **Figure 3.6** and **Figure 3.7** showcase the architecture of the model and provide a summary of its key components.

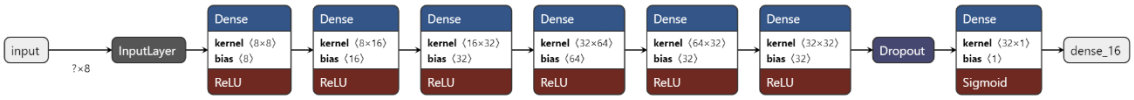


Figure 3.6: F-2 model architecture

Layer (type)	Output Shape	Param #
dense_10 (Dense)	(None, 8)	72
dense_11 (Dense)	(None, 16)	144
dense_12 (Dense)	(None, 32)	544
dense_13 (Dense)	(None, 64)	2112
dense_14 (Dense)	(None, 32)	2080
dense_15 (Dense)	(None, 32)	1056
dropout_5 (Dropout)	(None, 32)	0
dense_16 (Dense)	(None, 1)	33

Total params: 6,041
Trainable params: 6,041
Non-trainable params: 0

Figure 3.7: F-2 model summary

3.2.3 F-3 model

With approximately 3.5 times the number of parameters as F-2, this model boasts a total of 20,569 trainable parameters. Additional layers were incorporated and the dropout rate was slightly reduced. **Figure 3.8** and **Figure 3.9** visually represent the architecture of the model and its summary.

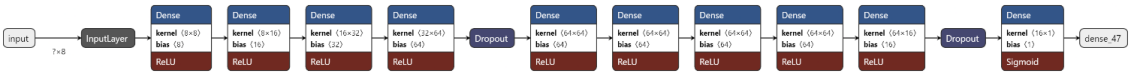


Figure 3.8: F-3 model architecture

Layer (type)	Output Shape	Param #
dense_38 (Dense)	(None, 8)	72
dense_39 (Dense)	(None, 16)	144
dense_40 (Dense)	(None, 32)	544
dense_41 (Dense)	(None, 64)	2112
dropout_6 (Dropout)	(None, 64)	0
dense_42 (Dense)	(None, 64)	4160
dense_43 (Dense)	(None, 64)	4160
dense_44 (Dense)	(None, 64)	4160
dense_45 (Dense)	(None, 64)	4160
dense_46 (Dense)	(None, 16)	1040
dropout_7 (Dropout)	(None, 16)	0
dense_47 (Dense)	(None, 1)	17

Total params: 20,569
Trainable params: 20,569
Non-trainable params: 0

Figure 3.9: F-3 model summary

3.2.4 F-4 model

In order to enhance the stability of loss and potentially enhance performance, this model introduced additional dropout layers. This approach aimed to assess whether a reduced number of parameters could still yield favorable outcomes. By incorporating more dropout layers, the model aimed to mitigate overfitting and promote generalization. **Figure 3.10** and **Figure 3.11** demonstrate the model architecture and characteristics.

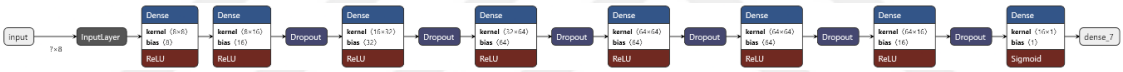


Figure 3.10: F-4 model architecture

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 8)	72
dense_1 (Dense)	(None, 16)	144
dropout (Dropout)	(None, 16)	0
dense_2 (Dense)	(None, 32)	544
dropout_1 (Dropout)	(None, 32)	0
dense_3 (Dense)	(None, 64)	2112
dropout_2 (Dropout)	(None, 64)	0
dense_4 (Dense)	(None, 64)	4160
dropout_3 (Dropout)	(None, 64)	0
dense_5 (Dense)	(None, 64)	4160
dropout_4 (Dropout)	(None, 64)	0
dense_6 (Dense)	(None, 16)	1040
dropout_5 (Dropout)	(None, 16)	0
dense_7 (Dense)	(None, 1)	17

Total params: 12,249
Trainable params: 12,249
Non-trainable params: 0

Figure 3.11: F-4 model summary

3.2.5 2B1 model

This model represents a significant advancement in this work as it incorporates both user metadata and tweet content in a two-branched architecture. The utilization of LSTM in

this model allows for the analysis of contextual relationships within the textual representation of tweets. LSTM has demonstrated its effectiveness in various applications, making it a suitable choice for this task.

During the experimentation process, other types of layers were tested on the tweet representation. However, their performance was either negligible or significantly poorer compared to LSTM. As a result, they were not deemed noteworthy for inclusion in this work. The focus remained on leveraging the capabilities of LSTM to extract meaningful information from the tweet text, thereby enhancing the overall performance of the model. The architecture and summary of the 2B1 model are presented in **Figure 3.12** and **Figure 3.13**, respectively. This model is the first two-branched model introduced in this work with a total of 449,537 trainable parameters. the left side branch takes the 300-dimensional tweet representation, which undergoes an LSTM layer[84] followed by a flatten layer to ensure a one-dimensional array as the next input. Subsequently, several dense layers and dropouts are applied.

On the right-side branch receives the 8 meta-data features as input and passes them through a sequence of 4 dense layers with dropouts.

The two branches are then merged using a concatenation layer, resulting in a one-dimensional output of size 512. Finally, the output layer produces the final prediction.

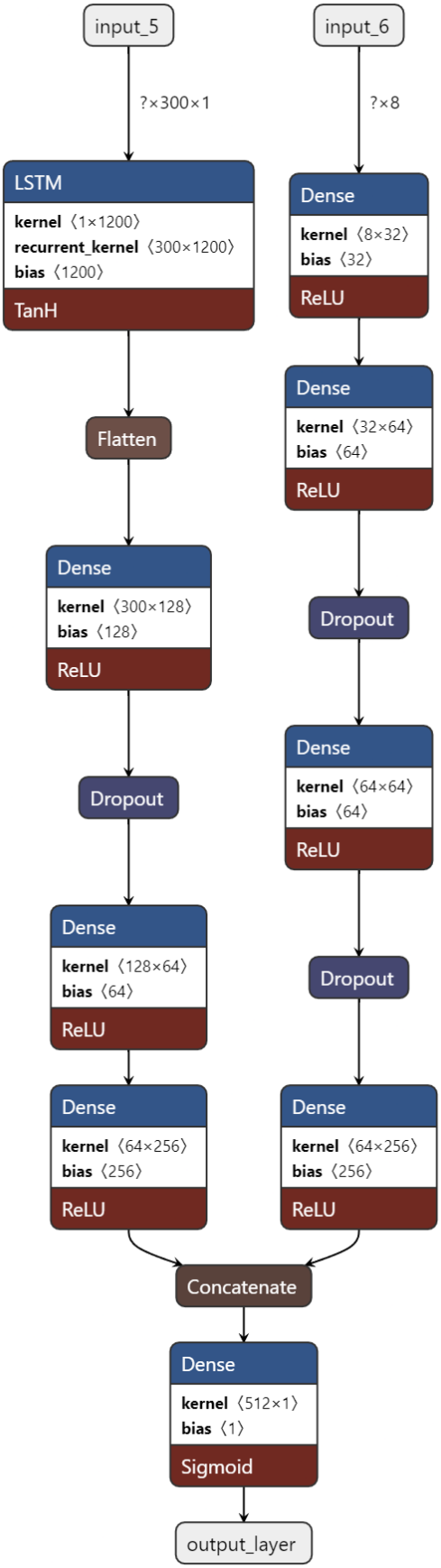
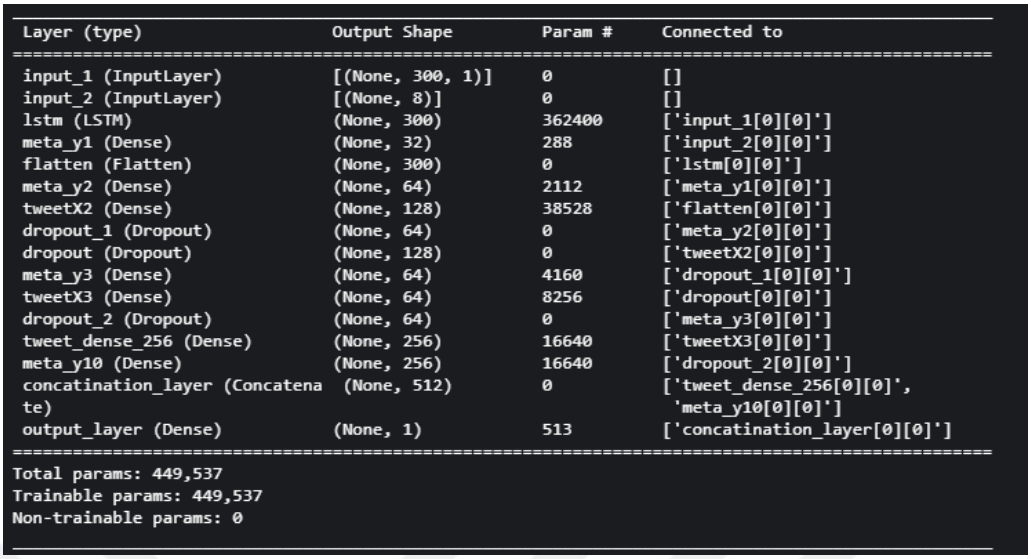


Figure 3.12: 2B1 model architecture



Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 300, 1)]	0	[]
input_2 (InputLayer)	[(None, 8)]	0	[]
lstm (LSTM)	(None, 300)	362400	['input_1[0][0]']
meta_y1 (Dense)	(None, 32)	288	['input_2[0][0]']
flatten (Flatten)	(None, 300)	0	['lstm[0][0]']
meta_y2 (Dense)	(None, 64)	2112	['meta_y1[0][0]']
tweetX2 (Dense)	(None, 128)	38528	['flatten[0][0]']
dropout_1 (Dropout)	(None, 64)	0	['meta_y2[0][0]']
dropout (Dropout)	(None, 128)	0	['tweetX2[0][0]']
meta_y3 (Dense)	(None, 64)	4160	['dropout_1[0][0]']
tweetX3 (Dense)	(None, 64)	8256	['dropout[0][0]']
dropout_2 (Dropout)	(None, 64)	0	['meta_y3[0][0]']
tweet_dense_256 (Dense)	(None, 256)	16640	['tweetX3[0][0]']
meta_y10 (Dense)	(None, 256)	16640	['dropout_2[0][0]']
concatination_layer (Concatena te)	(None, 512)	0	['tweet_dense_256[0][0]', 'meta_y10[0][0]']
output_layer (Dense)	(None, 1)	513	['concatination_layer[0][0]']

Total params: 449,537
Trainable params: 449,537
Non-trainable params: 0

Figure 3.13: 2B1 model summary

3.2.6 2B2 model

In this model, Dense layers are utilized in both branches to explore the possibility of replacing LSTM with a simpler layer. The objective is to determine whether the model can still capture the main features and relationships within the text representation or if its performance will deteriorate and becomes similar to random guessing.

By substituting LSTM with Dense layers, the total number of trainable parameters reduces to 267,809. This modification also leads to a decrease in the training time and memory requirements. LSTM is known for being computational-intensive and is often recommended to be used in conjunction with a GPU and high ram for efficient computation.

The model architecture and summary are demonstreated in **Figure 3.14** and **Figure 3.15**, respectively. These visualizations highlight the changes made to the model. In the tweet text branch, instead of using an LSTM layer, 5 dense layers are employed. Additionally, an extra dense layer is added to the right branch.

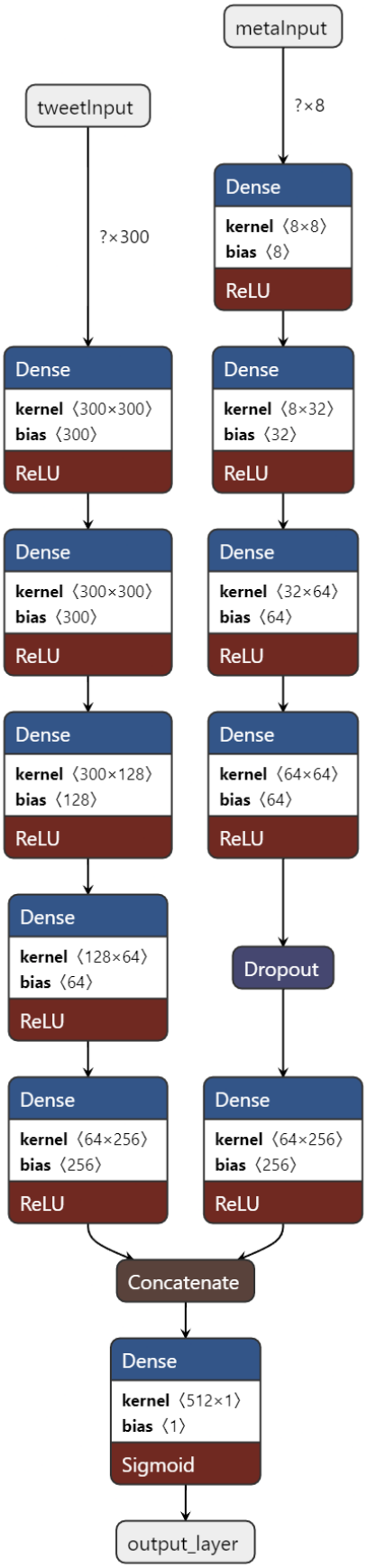
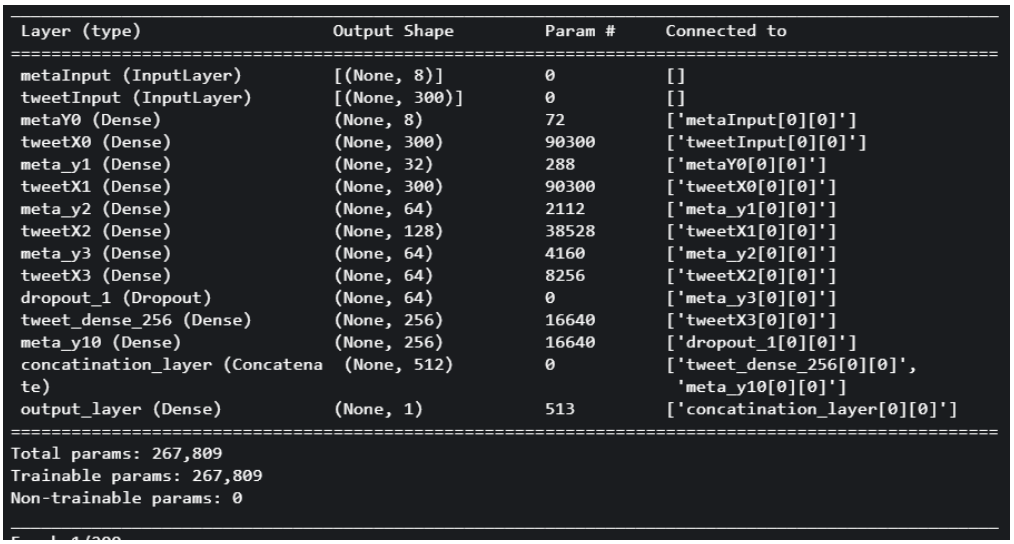


Figure 3.14: 2B2 model architecture



Layer (type)	Output Shape	Param #	Connected to
metaInput (InputLayer)	[(None, 8)]	0	[]
tweetInput (InputLayer)	[(None, 300)]	0	[]
metaY0 (Dense)	(None, 8)	72	['metaInput[0][0]']
tweetX0 (Dense)	(None, 300)	90300	['tweetInput[0][0]']
meta_y1 (Dense)	(None, 32)	288	['metaY0[0][0]']
tweetX1 (Dense)	(None, 300)	90300	['tweetX0[0][0]']
meta_y2 (Dense)	(None, 64)	2112	['meta_y1[0][0]']
tweetX2 (Dense)	(None, 128)	38528	['tweetX1[0][0]']
meta_y3 (Dense)	(None, 64)	4160	['meta_y2[0][0]']
tweetX3 (Dense)	(None, 64)	8256	['tweetX2[0][0]']
dropout_1 (Dropout)	(None, 64)	0	['meta_y3[0][0]']
tweet_dense_256 (Dense)	(None, 256)	16640	['tweetX3[0][0]']
meta_y10 (Dense)	(None, 256)	16640	['dropout_1[0][0]']
concatination_layer (Concatena te)	(None, 512)	0	['tweet_dense_256[0][0]', 'meta_y10[0][0]']
output_layer (Dense)	(None, 1)	513	['concatination_layer[0][0]']

Total params: 267,809
Trainable params: 267,809
Non-trainable params: 0

Figure 3.15: 2B2 model summary

3.2.7 2B3 model

This model shares the same architecture and specifications as the 2B1 model(**Figure 3.12** and **Figure 3.13**), with the exception that all dropout layers have been removed. In computer vision tasks, it has been suggested that certain regularization techniques like weight decay and dropout may not be necessary when sufficient data augmentation is applied, challenging the conventional belief of their significance[86].

In this model, the objective is to explore the impact of removing all dropout layers in this classification context. By eliminating dropouts, the aim is to understand how it affects the model's performance and generalization capabilities.

CHAPTER 4

4. RESULTS AND DISCUSSION

As mentioned earlier, the models were monitored and saved on WandB and Colab Notebook [74][75]. Every model underwent training using four different optimizers and in addition, two with a learning rate reducer.

Each presents the main results in a table, followed by plots and figures, including accuracy, validation accuracy, loss, and validation loss, and the confusion matrix for all five utilized optimizers. This approach enhances visualization and facilitates effective comparison between the different optimizers. Subsequently, each optimizer is individually examined, featuring its own dedicated plots alongside comprehensive evaluation metrics and corresponding discussions.

The previous chapter extensively covered the validity and calculation procedures for each metric utilized in this study. The primary objectives were as follows: to minimize the loss, false positives, and false negatives as much as possible, while maximizing all other relevant metrics.

The training and accuracy plots serve as valuable tools for monitoring metric changes throughout the training process. These plots provide insights into various aspects of the model's performance, such as determining if the model converges on the data and identifying potential instances of underfitting or overfitting.

In a well-fitted model, the training loss typically exhibits a consistent decrease throughout the training process, indicating effective learning and improved predictive performance. As training progresses, the loss should reach a stable level near the end. The validation loss, following a similar curve, is expected to be lower than the training loss.

Regarding accuracy, the training accuracy generally increases at the beginning of training and stabilizes towards the end. The validation accuracy, although slightly lower, should also exhibit a similar trend.

Typically, since the model is optimized and learns from the training data, the training loss is expected to be lower than the validation loss, and the training accuracy is expected to be higher than the validation accuracy. However, it's important to note that this general expectation may not hold true in all cases, as the behavior of these metrics can vary depending on the specific dataset and model complexity. The split of the dataset also affects the whole process because the classes considered by the model may not be well represented in the training and validation sets, and hence a skewed data in the training or validation may lead to diverse result. This may be overcome by applying stratified cross-validation which guarantees fair distribution of the data for effective testing and reasonable outcome.

4.1. F-1 Model

The results of Different models and optimizers of F1 architectures are summarized in **Figure 4.1** and **Figure 4.2**. **Table 4.1** shows the evaluation metrics of each model after testing it on the unseen data, **Figure 4.1** illustrates how the loss and validation loss evolve during the training process. **Figure 4.2** also shows how the accuracy and validation accuracy changes for these 6 models, each F1 model plots are covered separately from **Figure 4.1** to **Figure 4.14**.

Despite having the fewest number of parameters(321) and being the simplest model implemented in this study. This model provides some satisfactory results that are even better than other more complicated and complex models.

F1-SGD model halted training after 72 epochs due to early stopping, achieving an accuracy of 79% and an AUC of 0.866. According to the confusion matrix in **Figure 4.9**, the majority of the predicted instances fall within the range of 34953 true positives and 47756 true negatives. However, there were 17476 instances that were classified incorrectly as false negatives and 4747 instances that were classified incorrectly as false positives. Figure **4.3** demonstrates the evolution of loss, validation loss and accuracy, validation accuracy for this model, which illustrates that the model effectively adapts to the data and converges.

In **F1-RMSprop**, the training process was terminated after 38 epochs due to early stopping. The CM in **Figure 4.4** revealed that the F1-SGD model exhibited higher values of 35018 true positives and 47761 true negatives. Additionally, 4742 false positives were in close proximity, while the false negatives decreased to 17411.

According to **Figure 4.4**, there is an increased gap of approximately 0.06 between the loss and validation loss compared to the F1-SGD model. However, despite this larger gap, the model exhibits faster convergence, and the loss begins to stabilize even before reaching the tenth epoch of training.

As these results show, most of the metrics are quite similar to F1-SGD model. The loss is slightly lower, which explain the slight improvement in the CM and some other metrics like the AUC, F1 score, F2 score and MCC.

The results of **F1-Adam** model are quite similar to F1-RMSprop, with a 0.001 increase in the loss and some very small changes in the TN and FP as shown in the CM in **Figure 4.11**. **Figure 4.5** reveals a noticeable pattern when using the Adam optimizer. The loss fails to minimize or stabilize, and the validation loss displays fluctuating behavior. As a result, the training process was halted after only 26 epochs. Interestingly, the validation accuracy remains constant throughout all training epochs. These observations strongly suggest that the issue is primarily related to the optimizer itself or initialized weights, as these were the only variables altered in this scenario.

F1-Adadelta underwent the entire 200 pre-defined training epochs without any intervention by early stopping. **Figure 4.12** and **Figure 4.6** depict the confusion matrix, loss, and accuracy plots. During the training process, the loss consistently decreased; however, it did not reach a stable level by the end. This suggests that further training and tuning may potentially improve the model's performance.

In **F1-SGD-LR(0.015)**, applying the learning rate reducer with 0.015 as initial learning rate, the value decreased down to 0.00001500. the training halted after 72 epochs, achieving an accuracy of 79% and AUC of 0.867and the loss value was 0.423.

Despite the application of learning rate reducer in **F1-SGD-LR(0.0010)**, the learning rate value remained constant at 0.0010 throughout all the epochs, indicating that the reducer did not modify the learning rate. the model's accuracy experienced a decline of 1 percent, reaching a final accuracy of 78%. Interestingly, the loss with a value of 0.445 was the second highest observed in this architecture. Just like the F1-Adadelta model, the training

process successfully completed all 200 epochs without any early stopping. According to the confusion matrix shown in **Figure 4.14**, this model performed exceptionally well in identifying true positives, representing the fake accounts. It achieved the highest number of true positives (TP) among all the F1 models. Additionally, the number of true negatives (TN) slightly decreased to 46336. In terms of false negatives (FN), this model had the lowest count compared to the other four optimizers. However, the number of false positives (FP) increased to 6167.

Based solely on the confusion matrix results, it can be concluded that this model outperforms the other F1 models. It successfully identified the highest number of fake accounts correctly (TP) and had the lowest number of false positives (FP) among the models.

All the results and plots related to this model are demonstrated in **Table 4.1** and from **Figure 4.1** to **Figure 4.14**.

Table 4.1: F1 Results

Name	epochs	Acc	Loss	AUC	Confusion Matrix				Precision	Recall	Specificity	F1	F2	MCC
					TP	TN	FP	FN						
F1-SGD	72	0.79	0.428	0.866	34953	47756	4747	17476	0.881	0.668	0.91	0.759	0.7	0.5941
F1-RMSprop	38	0.79	0.416	0.868	35018	47761	4742	17411	0.881	0.668	0.91	0.76	0.702	0.5953
F1-Adam	26	0.79	0.417	0.868	35018	47759	4744	17411	0.881	0.668	0.91	0.76	0.702	0.5953
F1-Adadelta	199	0.75	0.496	0.825	32033	46236	6267	20396	0.836	0.611	0.881	0.706	0.646	0.5106
F1_SGD-lr(0.015)	72	0.79	0.423	0.867	35000	47755	4748	17429	0.881	0.668	0.91	0.759	0.702	0.5949
F1- SGD-lr(0.001)	199	0.78	0.445	0.858	35525	46336	6167	16904	0.852	0.678	0.883	0.755	0.707	0.5723

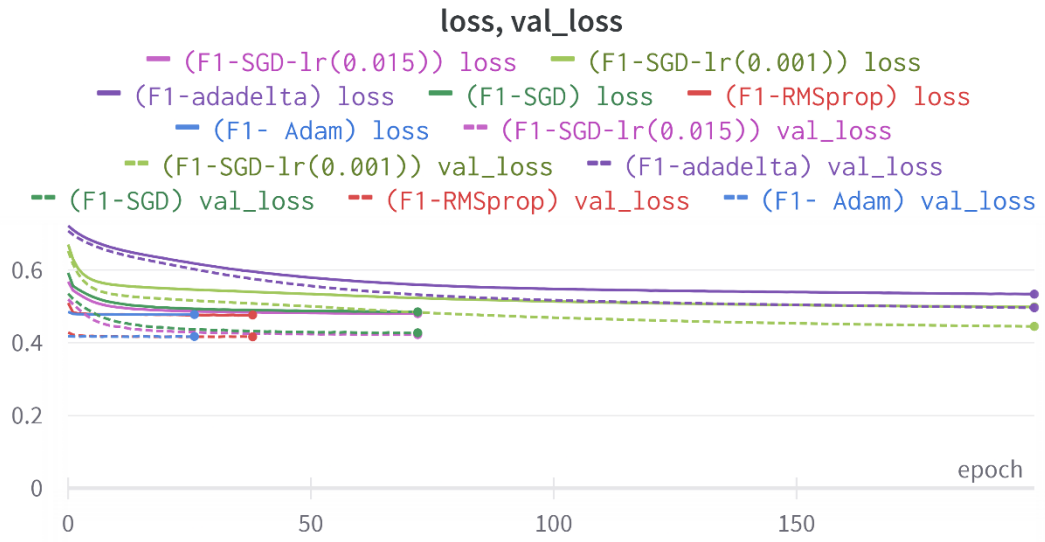


Figure 4.1: F1 models loss, validation loss

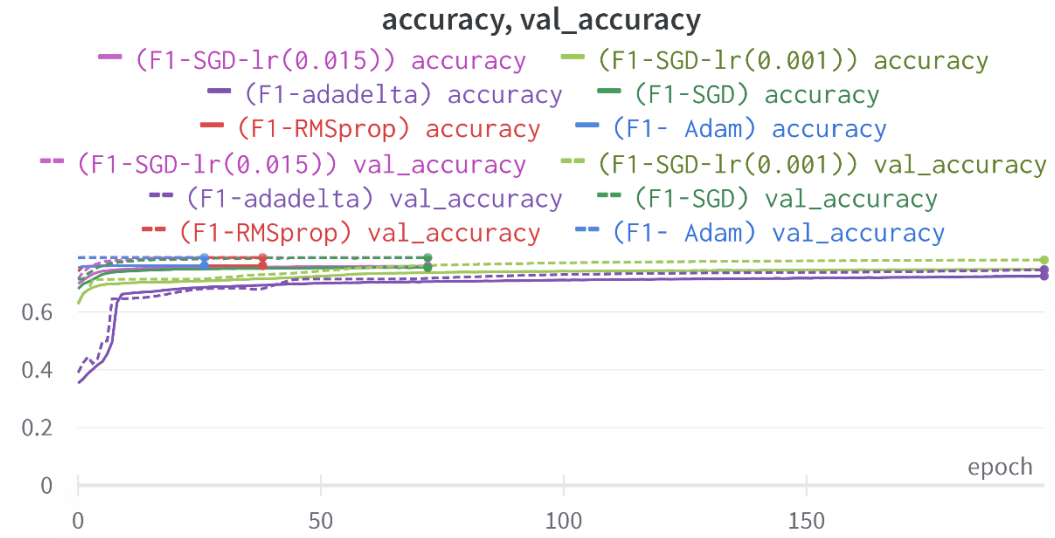


Figure 4.2: F1models accuraccy, validation accuraccy

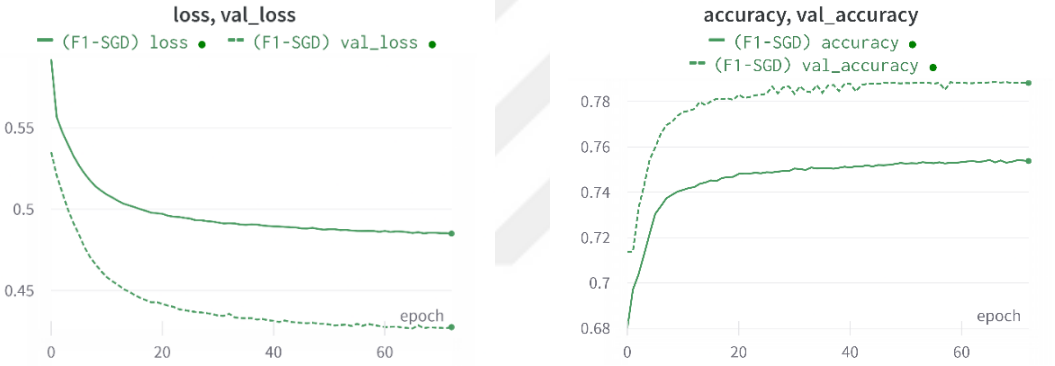


Figure 4.3: the loss in F1-SGD and the validation loss, as well as the accuracy and the validation accuracy

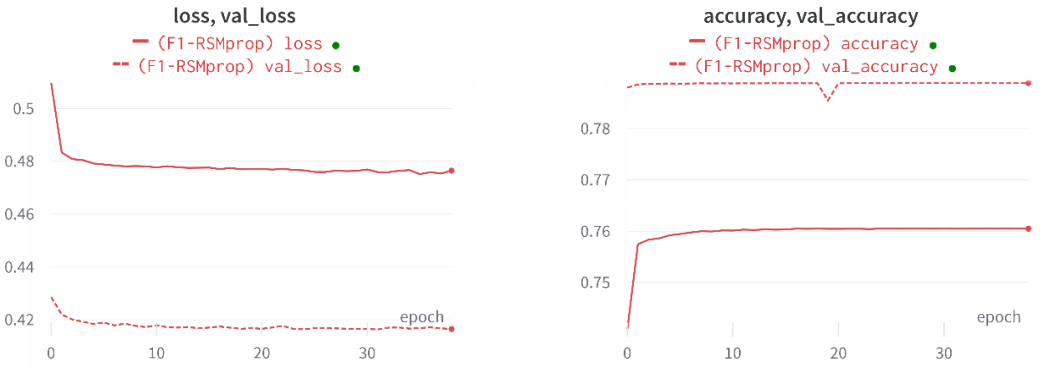


Figure 4.4: the loss in F1-RMSprop and the validation loss, as well as the accuracy and the validation accuracy

Figure 4.5: the loss in F1-AdaM and the validation loss, as well as the accuracy and the validation accuracy

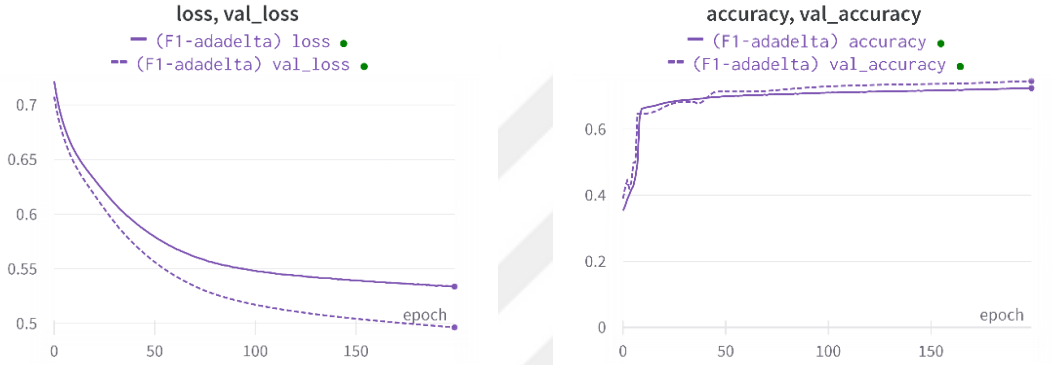


Figure 4.6: the loss in F1-Adadelta and the validation loss, as well as the accuracy and the validation accuracy

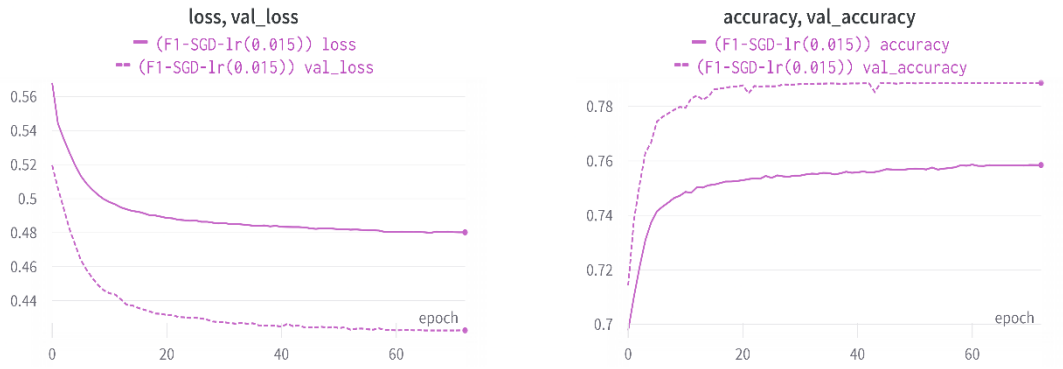


Figure 4.7: the loss in F1-SGD-LR(0.015) and the validation loss, as well as the accuracy and the validation accuracy

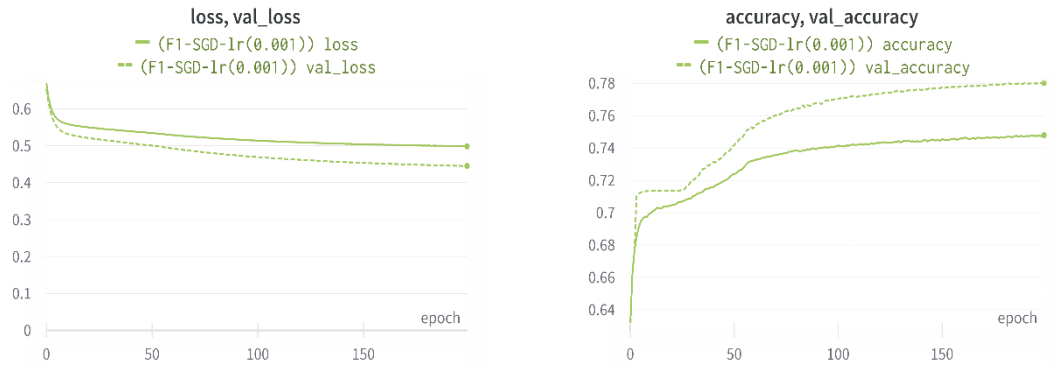


Figure 4.8: the loss in F1-SGD-LR(0.001) and the validation loss, as well as the accuracy and the validation accuracy

Following are the CM for this model Variations:

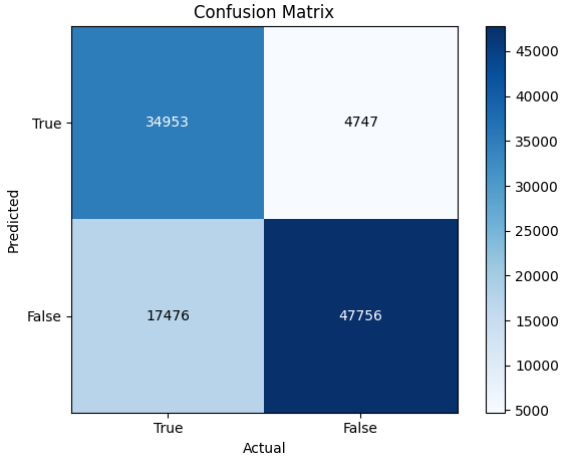


Figure 4.9: F1-SGD Confusion matrix

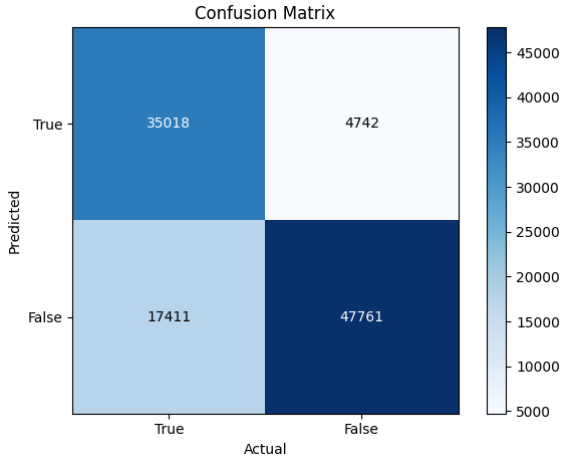


Figure 4.10: F1-RMSprop Confusion matrix

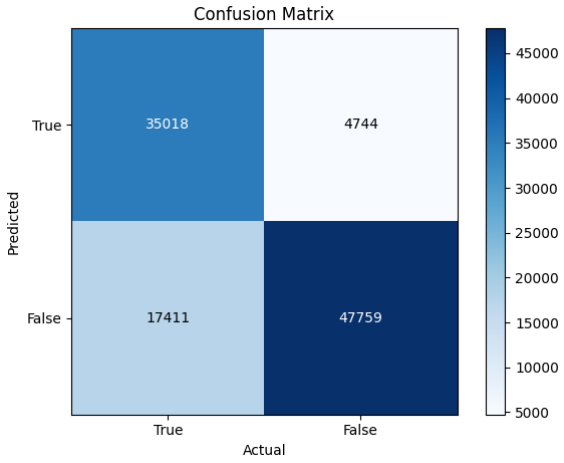


Figure 4.11: F1-Adam Confusion matrix

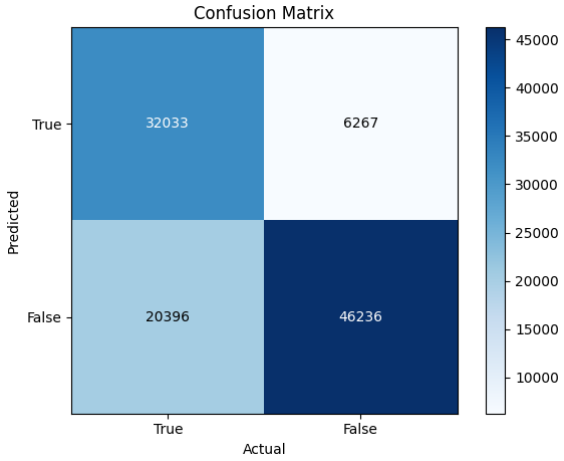


Figure 4.12: F1-Adadelta Confusion matrix

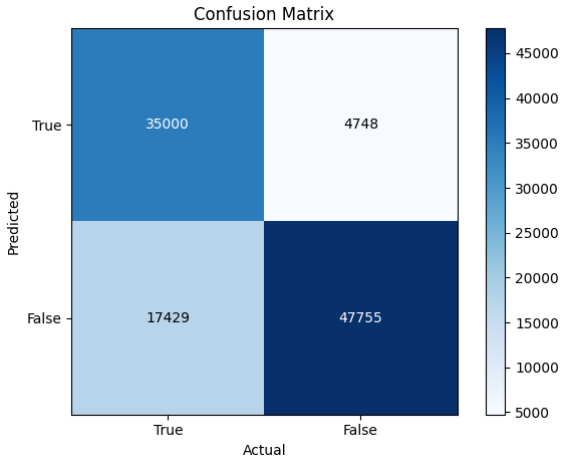


Figure 4.13: F1-SGD-LR(0.015) Confusion Matrix

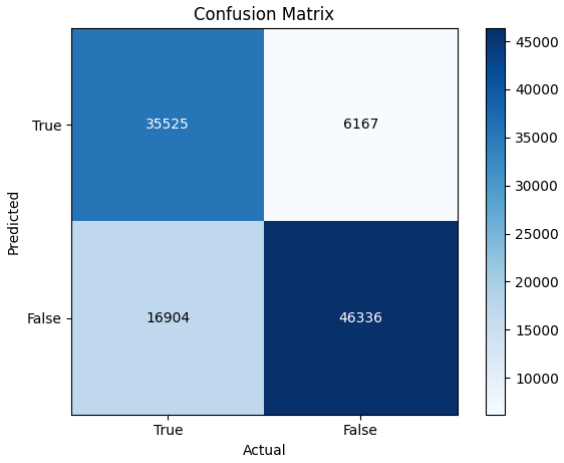


Figure 4.14: F1-SGD-LR(0.001) Confusion matrix

4.2. F-2 Model

Despite the inclusion of additional densely connected layers in this model, expected to enhance the capture of interconnected relations among various features, the results contradicted this intention. Surprisingly, the performance was mostly equal to that of the simpler F-1 model. **Figure 4.15**, **Figure 4.16**, and **Table 4.1** summarizes the results.

Early stopping halted the training for Most of the F2 models; **Figure 4.16** shows that **F2-SGD** was the first one to halt training after 16 epochs(**Figure 4.17**). While **F2-Adadelta** was the only F2 model to complete the 200 training epochs.

In the **F2-RMSprop** model, the loss remains relatively stagnant after the initial 2 epochs, which can be seen in the loss and accuracy plots(**Figure 4.18**). This observation may indicate a potential issue of overfitting, suggesting that the model has reached a point where it is unable to effectively learn from the data and generate generalized rules. **F2-Adam** shows a similar behavior, judging only using the confusion matrix. The false positives (FP) decreased significantly, down to 1. However, the false negatives (FN) increased to 37484. The true positives (TP) also decreased, from the average of other models to 14945. However, the most noticeable value was the large increase in the number of true negatives (TN), which correctly identified fake accounts. There were 52502 TN in the F2-Adam model.

However, when looking at the unstable increasing loss and decreasing accuracy in **Figure 4.19**, in addition to the early stopping of the training after only 7 epochs, this implies the same issue that the model is not able to generalize and learn from the data.

between all F2s, **F2-SGD-LR(0.015)** and **F2-SGD-LR(0.001)** have the lowest loss. With the use of learning rate reducer in **F2-SGD-LR(0.015),** the learning rate went down to 0.0015, and in **F2-SGD-LR(0.001)**.

All the results and plots related to this model are demonstrated in **Table 4.2** and from **Figure 4.15** to **Figure 4.28** .

Table 4.2: F2 Results

Name	epochs	Acc	Loss	AUC	Confusion Matrix				Precision	Recall	Specificity	F1	F2	MCC
					TP	TN	FP	FN						
F2-SGD	16	0.79	0.417	0.866	35018	47753	4750	17411	0.881	0.668	0.91	0.76	0.702	0.5951
F2-RMSprop	17	0.79	0.417	0.868	35018	47759	4744	17411	0.881	0.668	0.91	0.76	0.701	0.5953
F2-Adam	7	0.64	0.584	0.643	14945	52502	1	37484	0.1	0.285	0.1	0.444	0.332	0.4078
F2-Adadelta	199	0.78	0.432	0.86	34065	47733	4770	18364	0.877	0.645	0.909	0.747	0.685	0.5788
F2-SGD-lr(0.015)	18	0.79	0.416	0.868	35018	47758	4745	17411	0.881	0.668	0.91	0.76	0.702	0.5952
F2-SGD-lr(0.001)	33	0.79	0.416	0.868	35018	47760	4743	17411	0.881	0.668	0.91	0.76	0.701	0.5953

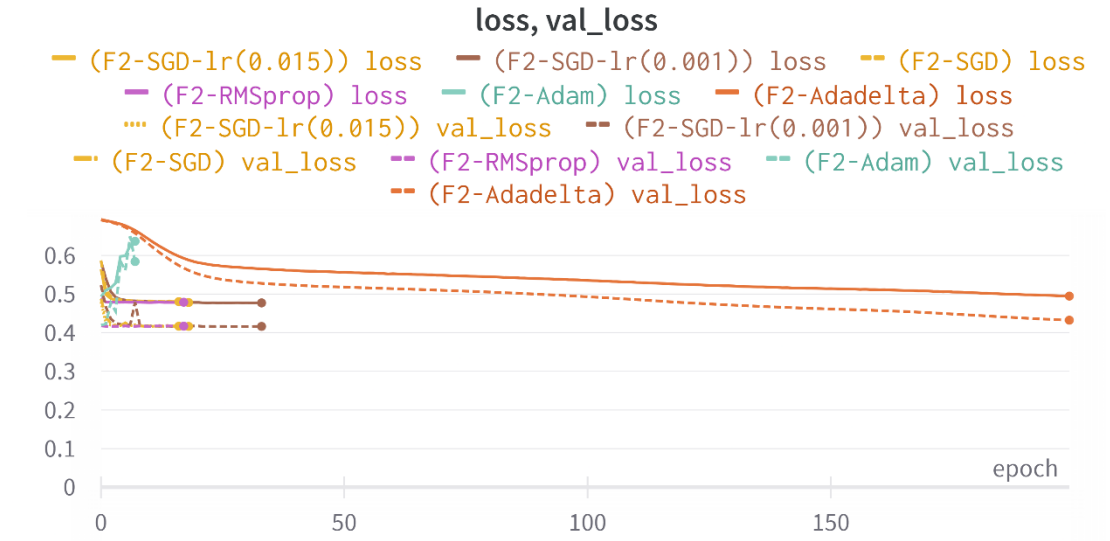


Figure 4.15: F2 models loss, validation loss

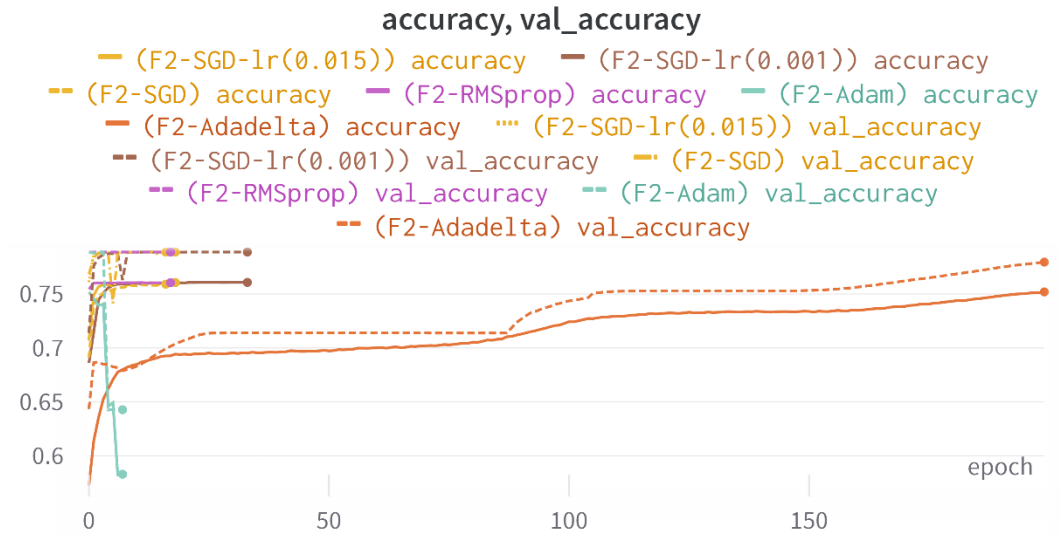


Figure 4.16: F2 models accuracy, validation accuracy

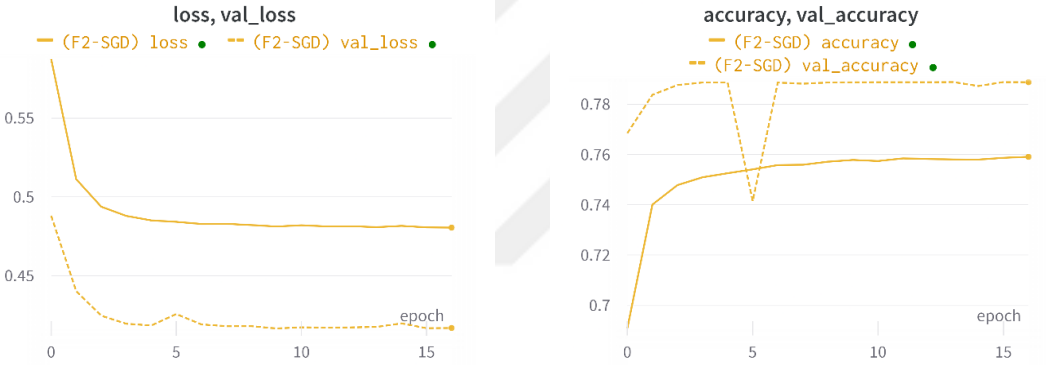


Figure 4.17: the loss in F2-SGD and the validation loss, as well as the accuracy and the validation accuracy

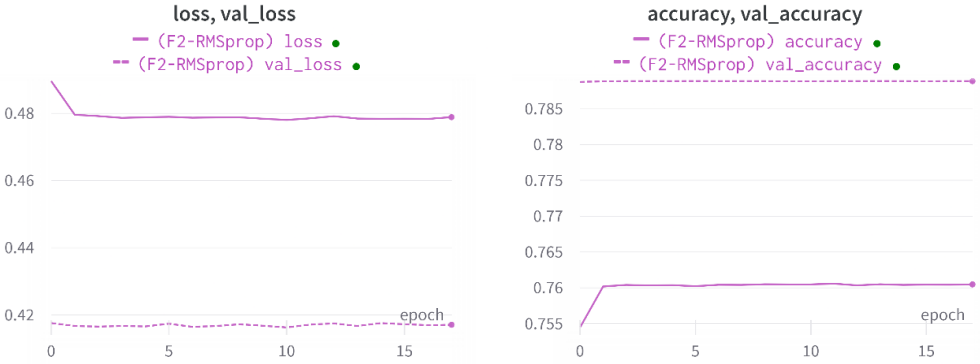


Figure 4.18: the loss in F2-RMSprop and the validation loss, as well as the accuracy and the validation accuracy

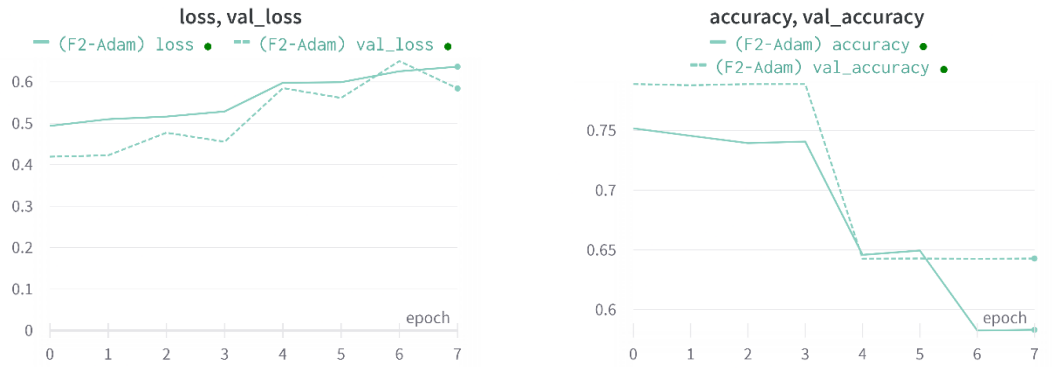


Figure 4.19: the loss in F2-Adam and the validation loss, as well as the accuracy and the validation accuracy

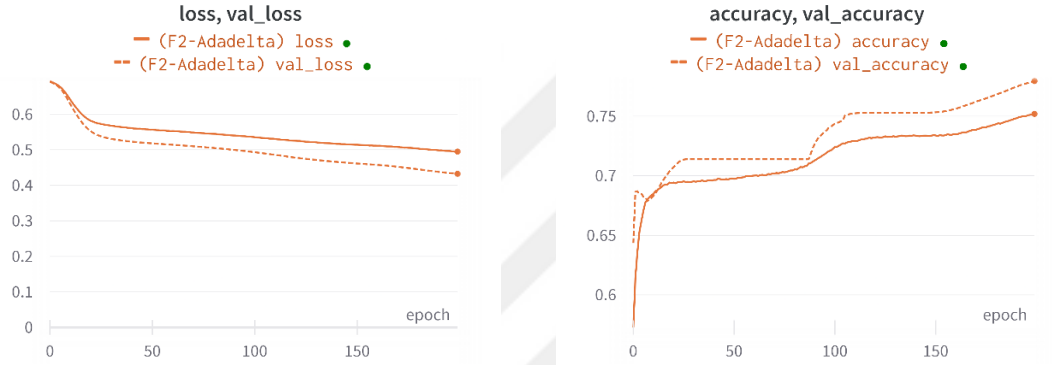


Figure 4.20: the loss in F2-Adadelta and the validation loss, as well as the accuracy and the validation accuracy

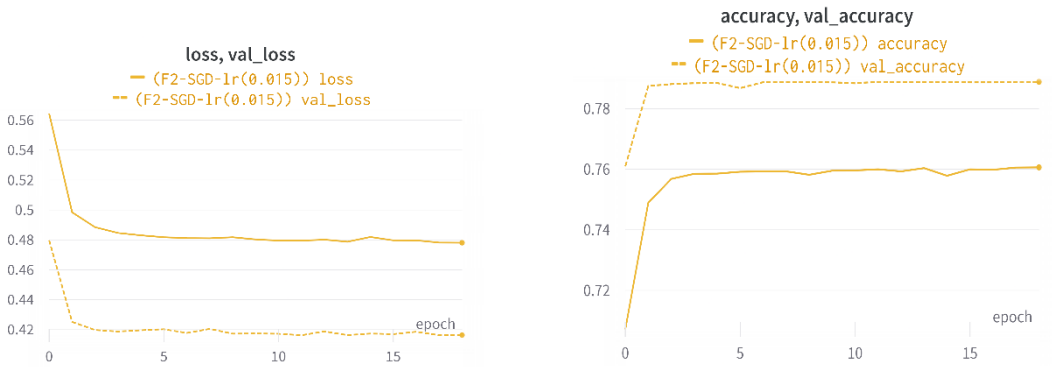


Figure 4.21: the loss in F2-SGD-LR(0.015) and the validation loss, as well as the accuracy and the validation accuracy

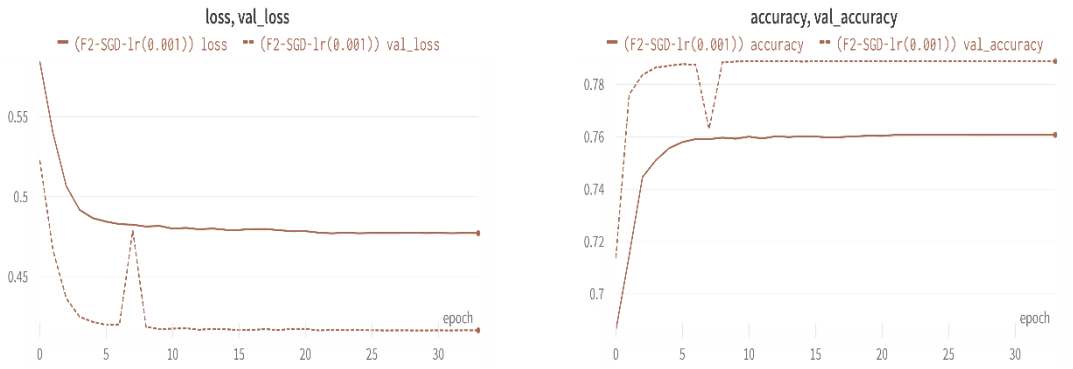


Figure 4.22: the loss in F2-SGD-LR(0.001) and the validation loss, as well as the accuracy and the validation accuracy

Following are the CM for this model Variations:

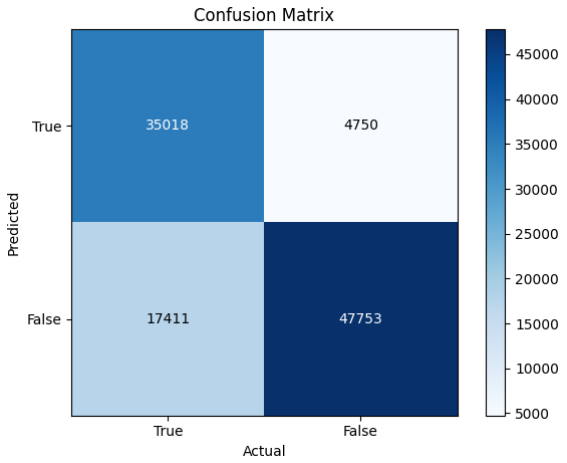


Figure 4.23: F2-SGD Confusion Matrix

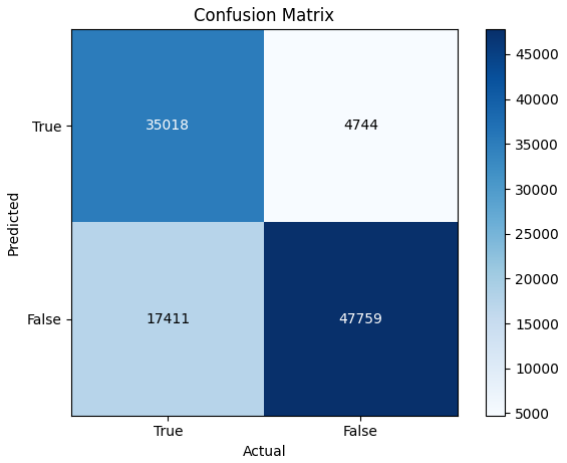


Figure 4.24: F2-RMSprop Confusion Matrix

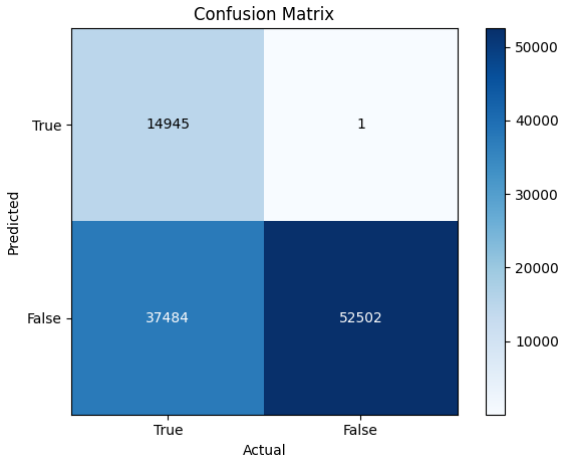


Figure 4.25: F2-Adam Confusion Matrix

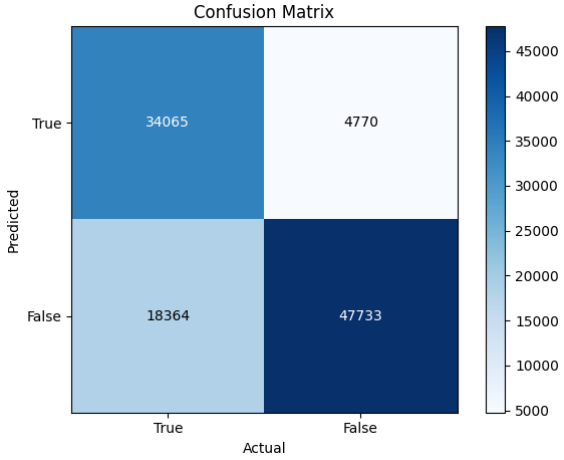


Figure 4.26: F2-Adadelta Confusion Matrix

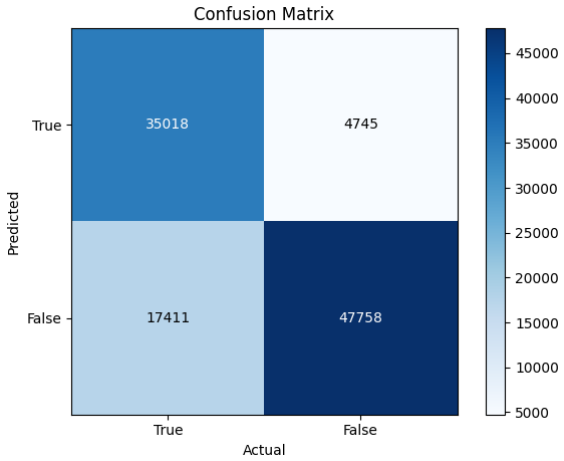


Figure 4.27: F2-SGD-LR(0.015) Confusion Matrix

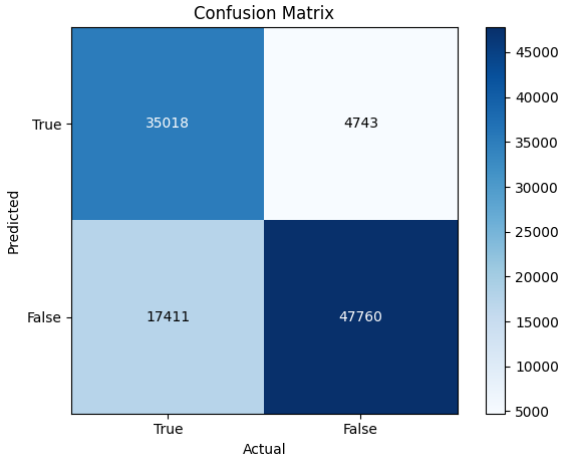


Figure 4.28: F2-SGD-LR(0.001) Confusion Matrix

4.3. F-3 Model

Out of the various optimizers tested in this model, only two have demonstrated promising performance: **F3-Adadelta** and **F3-SGD-LR(0.001)**. The implementation of early stopping adversely affected the other models. For instance, **F3-SGD** stopped after 15 epochs due to signs of overfitting towards the end(**Figure 4.31**). **F3-Adam** failed to effectively learn from the data, resulting in increasing loss and decreasing accuracy over the course of 7 training epochs(**Figure 4.33**). Similarly, **F3-RMSprop** exhibited similar behavior(**Figure 4.32**). Additionally, **F3-SGD-LR(0.015)**,**Figure 4.35** struggled to converge to a stable loss value and continuously fluctuated until early stopping intervened to halt the process.

Despite early stopping being applied after 58 epochs, the **F3-SGD_LR(0.001)** model in demonstrated the ability to learn from the data and generalize, resulting in a loss of 0.416 and an accuracy of 79%. Similarly, the **F3-Adatelta** model continued training for 196 epochs, achieving a comparable loss of 0.418 and the same accuracy(**Figure 4.34**). **Table 4.3** displays other favorable metrics for these two models. Both models exhibited similar values, with precision around 0.8, recall around 0.6, specificity of 0.9, and F1 and F2 scores approximately at 0.7. The Matthews correlation coefficient (MCC) was approximately 0.59 for both models.

All the results and plots related to this model are demonstrated in **Table 4.3** and from **Figure 4.29** to **Figure 4.42** .

Table 4.3: F3 Results

Name	epochs	Acc	Loss	AUC	Confusion Matrix				Precision	Recall	Specificity	F1	F2	MCC
					TP	TN	FP	FN						
F3-SGD	15	0.79	0.422	0.867	35000	47759	4744	17429	0.881	0.668	0.91	0.759	0.702	0.595
F3-RMSprop	8	0.79	0.418	0.867	35017	47761	4742	17412	0.881	0.668	0.91	0.76	0.902	0.5953
F3-Adam	7	0.5	0.693	0.5	52429	0	52503	0	0.5	1	0	0.666	0.833	N/A
F3-Adadelta	196	0.79	0.418	0.867	35016	47742	4761	17413	0.88	0.668	0.909	0.76	0.702	0.5949
F3-SGD-lr(0.015)	14	0.79	0.419	0.868	34983	47760	4743	17446	0.881	0.667	0.91	0.76	0.701	0.595
F3-SGD-lr(0.001)	58	0.79	0.419	0.868	35017	47760	4743	17412	0.881	0.668	0.91	0.76	0.702	0.5953

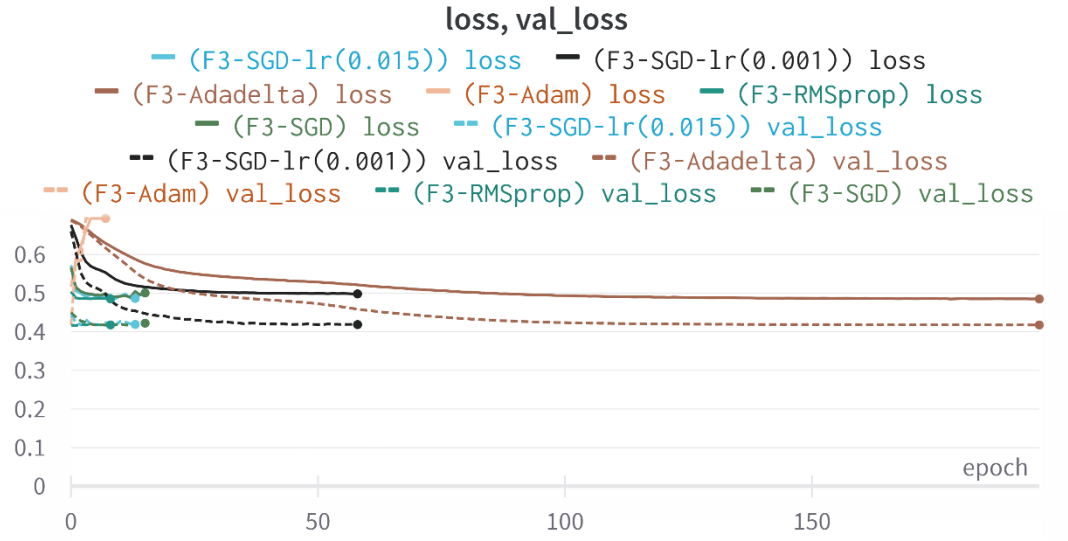


Figure 4.29: F3 models loss, validation loss

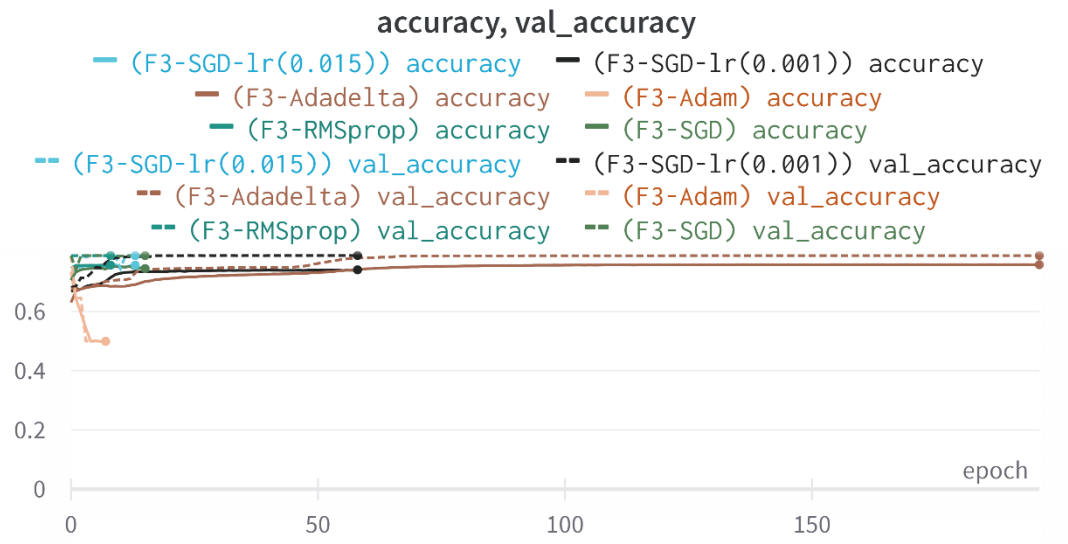


Figure 4.30: F3 models accuracy, validation accuracy

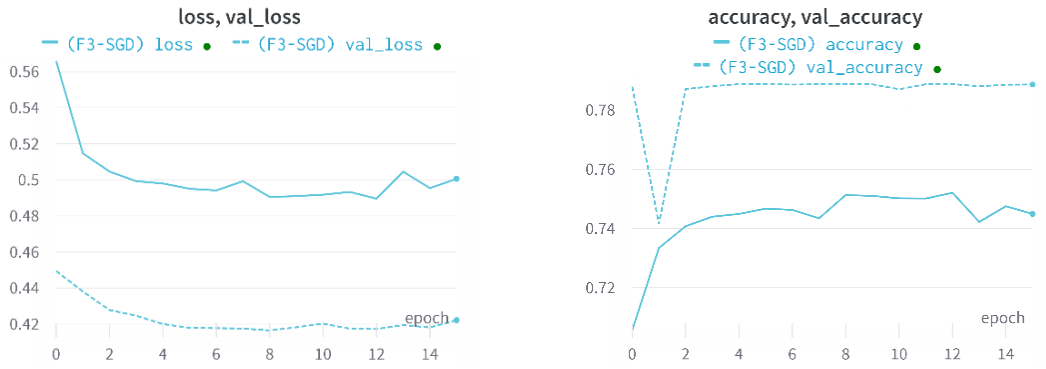


Figure 4.31: the loss in F3-SGD and the validation loss, as well as the accuracy and the validation accuracy

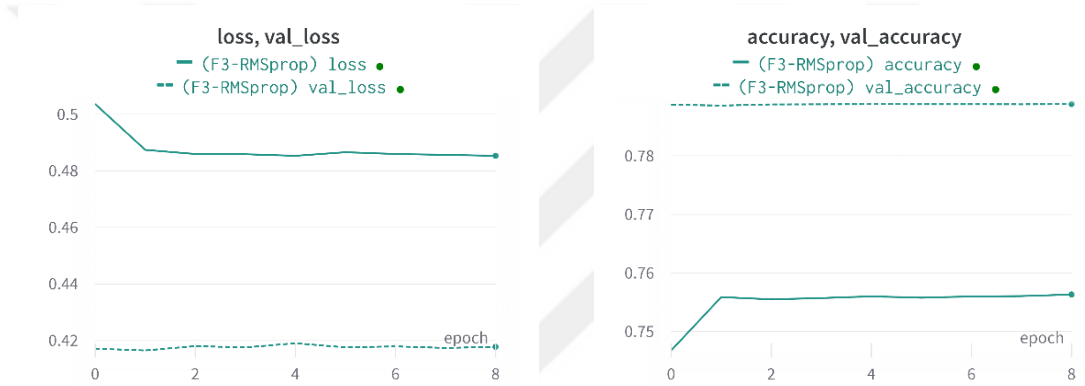


Figure 4.32: the loss in F3-RMSprop and the validation loss, as well as the accuracy and the validation accuracy

Figure 4.33: the loss in F3-Adam and the validation loss, as well as the accuracy and the validation accuracy

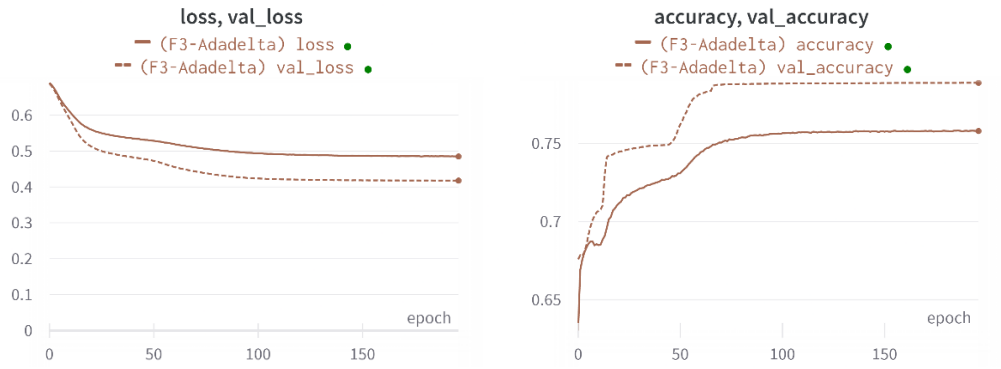


Figure 4.34: the loss in F3-Adadelta and the validation loss, as well as the accuracy and the validation accuracy

Figure 4.35: the loss in F3-SGD-LR(0.015) and the validation loss, as well as the accuracy and the validation accuracy

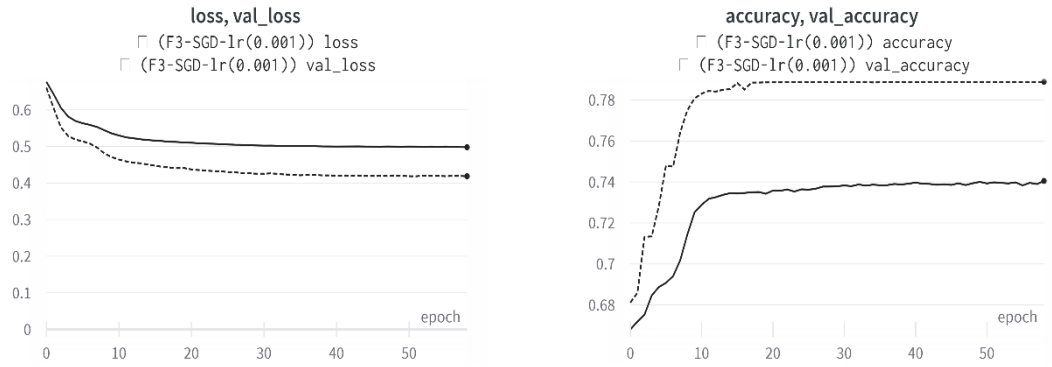


Figure 4.36: the loss in F3-SGD-LR(0.001) and the validation loss, as well as the accuracy and the validation accuracy

Following are the CM for this model Variations.:

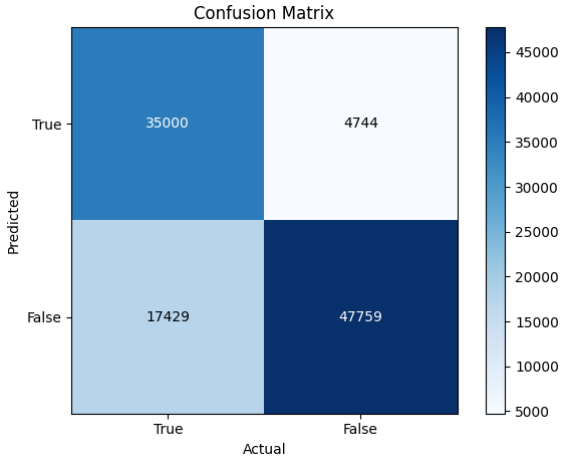


Figure 4.37: F3-SGD Confusion Matrix

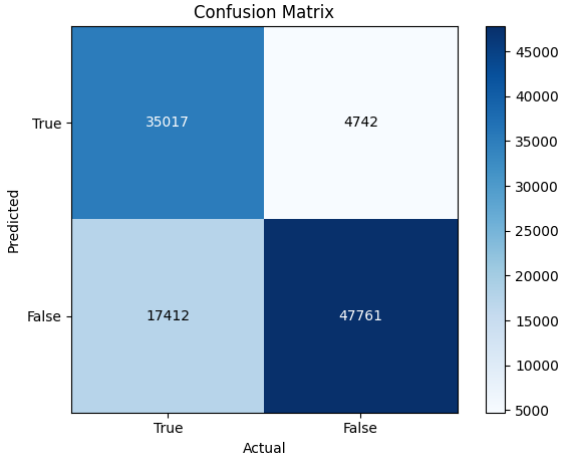


Figure 4.38: F3-RMSprop Confusion Matrix

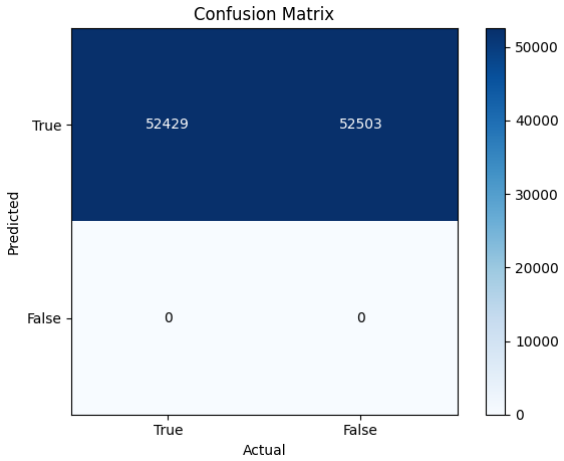


Figure 4.39: F3- Adam Confusion Matrix

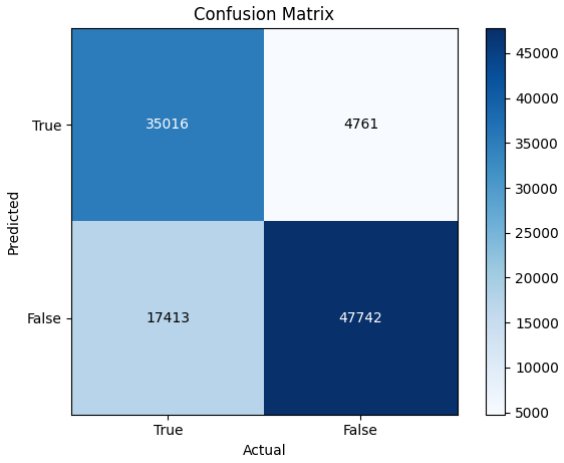


Figure 4.40: F3-Adadelta Confusion Matrix

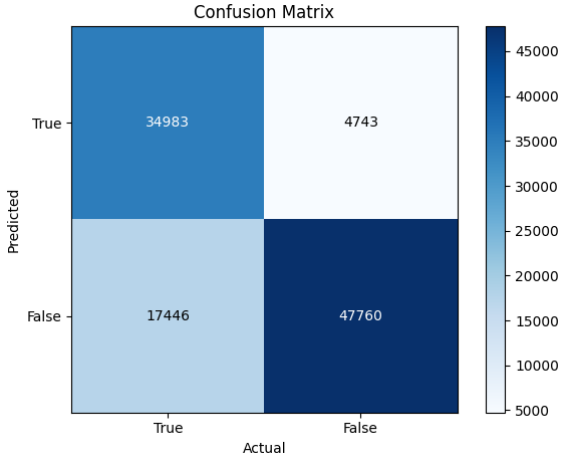


Figure 4.41: F3-SGD-LR(0.015) Confusion Matrix

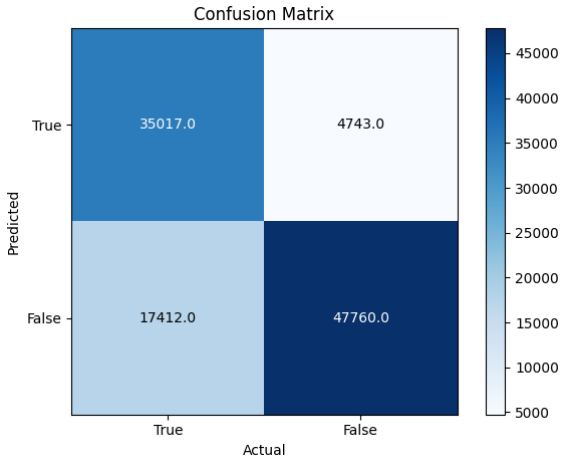


Figure 4.42: F3-SGD-LR(0.001) Confusion Matrix

4.4. F-4 Model

Out of the different optimizers of this architecture, only **F4-Adadelta** completed the 200 training epochs. Both **F4-Adam** and **F4-RMSprop** exhibited underfitting and failed to effectively learn from the data, as evidenced by their unstable curves depicted in **Figure 4.47** and **Figure 4.46** respectively. **F4-SGD** training was stopped after 25 epochs due to fluctuating loss and validation loss towards the end, as illustrated in **Figure 4.49**. On the other hand, **F4-Adadelta** completed the full 200 epochs, but did not converge to a stable loss value. This indicates that further training and fine tuning might be necessary, as both accuracy and loss did not stabilize, as shown in **Figure 4.48**.

When using the learning rate reducer on the SGD optimizer with two different initial learning rate values of 0.001 and 0.015, it was observed that **F4-SGD-LR(0.015)** achieved faster convergence and reached a stable loss value earlier compared to F4-**SGD-LR(0.001)**. The former model completed training after 22 epochs, while the latter model halted after 79 training epochs.

In terms of loss, **F4-RMSprop** achieved the lowest value of 0.416. Apart from **F4-Adam**, all models had a similar accuracy of approximately 79%. The learning rate was reduced to 0.00015 in **F4-SGD-LR(0.015)** and to 0.00001 in **F4-SGD-LR(0.001)**.

When considering the F1 and F2 scores, **F4-Adadelta** had the highest values, with 0.77 for the F1 score and 0.84 for the F2 score. However, when considering the Matthews Correlation Coefficient (MCC) and examining the confusion matrix, it is evident that **F4-Adadelta** may not be the best performing model.

All results and plots related to this model are demonsrated in the following figures and table.

Table 4.4: F4 Results

Name	epochs	Acc	Loss	AUC	Confusion Matrix				Precision	Recall	Specificity	F1	F2	MCC
					TP	TN	FP	FN						
F4-SGD	25	0.79	0.425	0.868	34981	47760	4743	17448	0.881	0.667	0.91	0.76	0.701	0.5947
F4-RMSprop	26	0.79	0.416	0.868	35018	47761	4742	17411	0.881	0.668	0.91	0.76	0.702	0.5953
F4-Adam	9	0.64	0.582	0.645	52429	15197	37306	0	0.586	1	0.289	0.738	0.875	0.4112
F4-Adadelta	199	0.74	0.476	0.861	46886	30581	21922	5543	0.681	0.894	0.582	0.773	0.842	0.5017
F4-SGD-lr(0.015)	22	0.79	0.446	0.855	34964	47761	4742	17465	0.881	0.667	0.91	0.759	0.7	0.5944
F4-SGD-lr(0.001)	79	0.79	0.444	0.854	34845	47761	4742	17584	0.88	0.665	0.91	0.757	0.699	0.5924

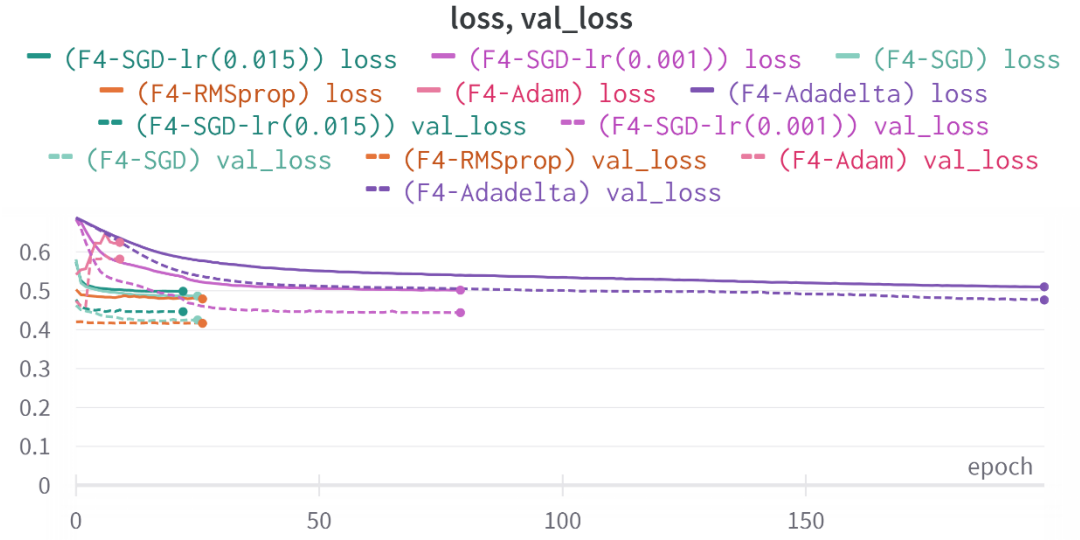


Figure 4.43: F4 models loss, validation loss

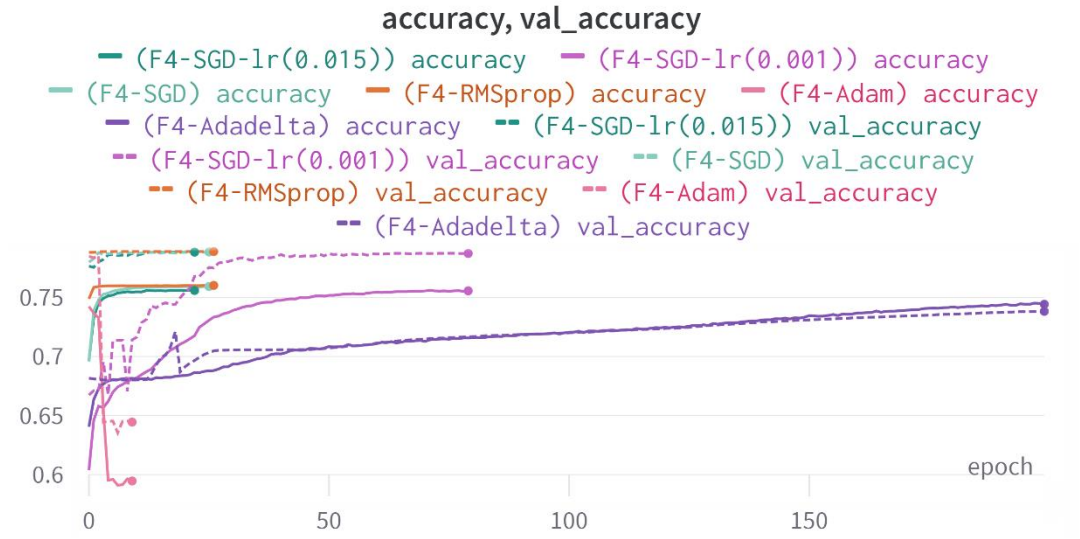


Figure 4.44: F4 models accuracy, validation accuracy

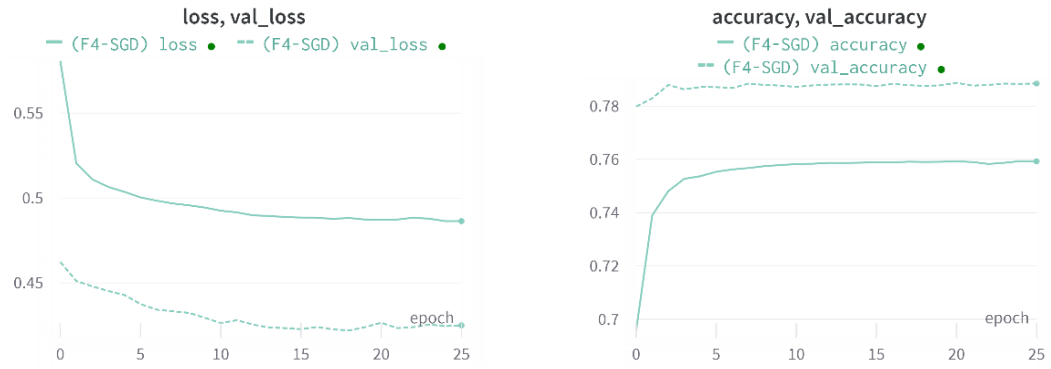


Figure 4.45: the loss in F4-SGD and the validation loss, as well as the accuracy and the validation accuracy

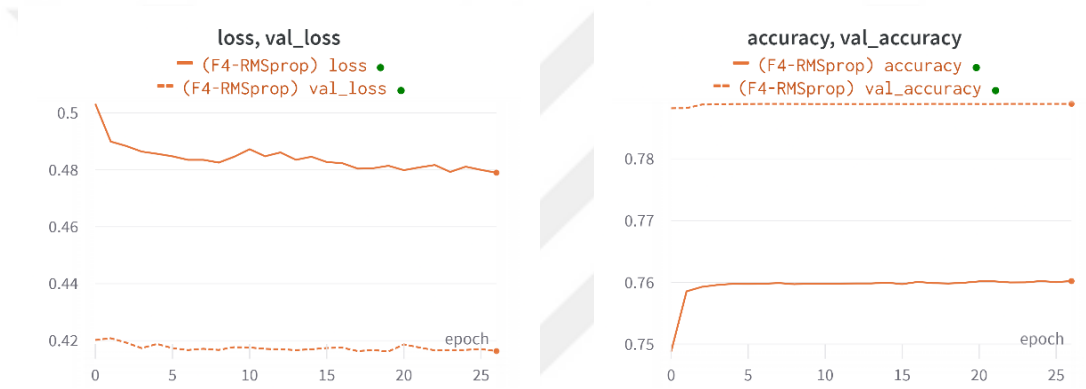


Figure 4.46: the loss in F4-RMSprop and the validation loss, as well as the accuracy and the validation accuracy

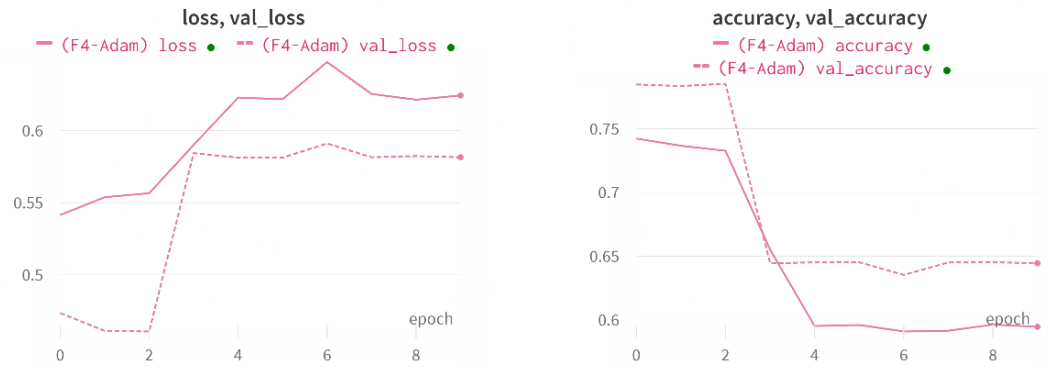


Figure 4.47: the loss in F4-Adam and the validation loss, as well as the accuracy and the validation accuracy

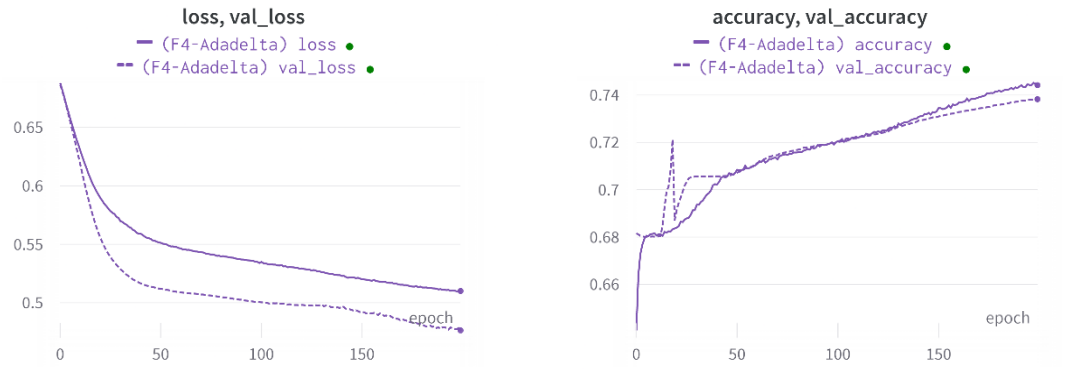


Figure 4.48: the loss in F4-Adadelta and the validation loss, as well as the accuracy and the validation accuracy

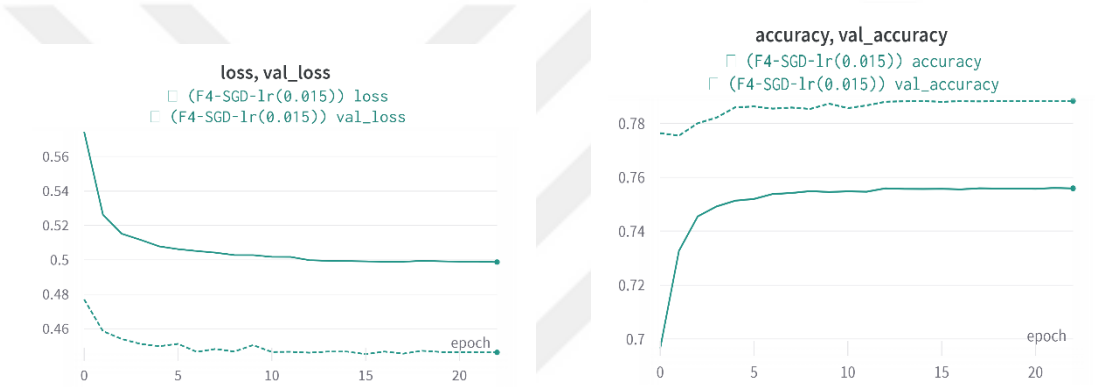


Figure 4.49: the loss in F4-SGD-LR(0.015)and the validation loss, as well as the accuracy and the validation accuracy

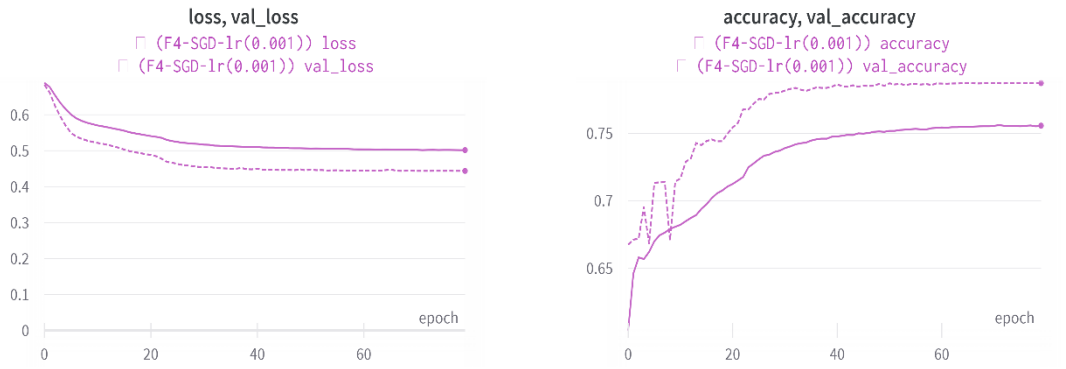


Figure 4.50: the loss in F4-SGD-LR(0.001)and the validation loss, as well as the accuracy and the validation accuracy

Following are the CM for this model Variations:

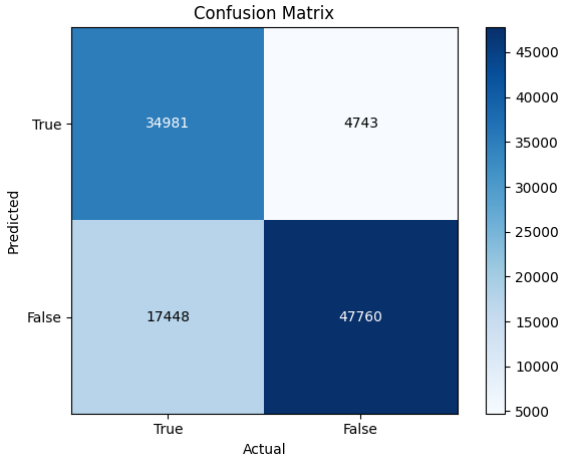


Figure 4.51: F4-SGD Confusion Matrix

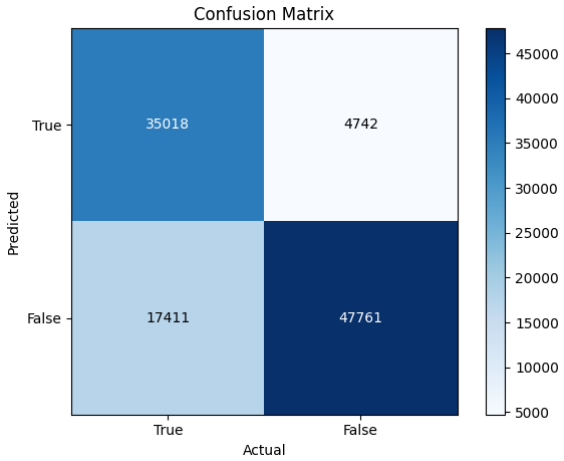


Figure 4.52: F4-RMSprop Confusion Matrix

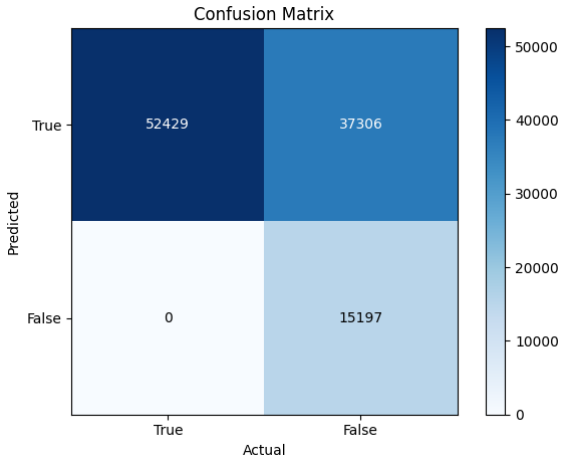


Figure 4.53: F4-Adam Confusion Matrix

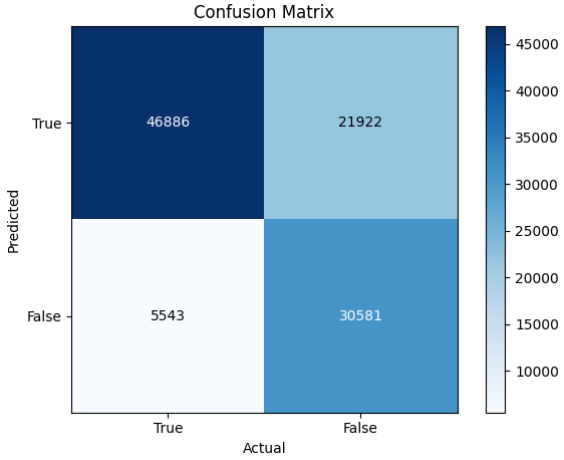


Figure 4.54: F4-Adadelta Confusion Matrix

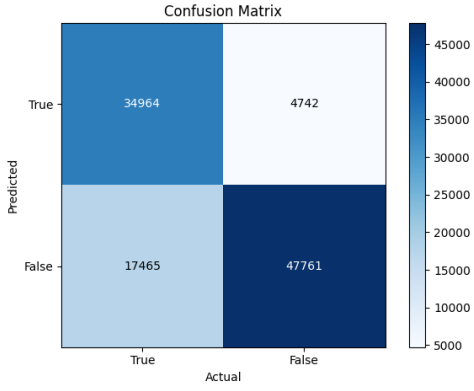


Figure 4.55: F4-SGD-LR(0.015) Confusion Matrix

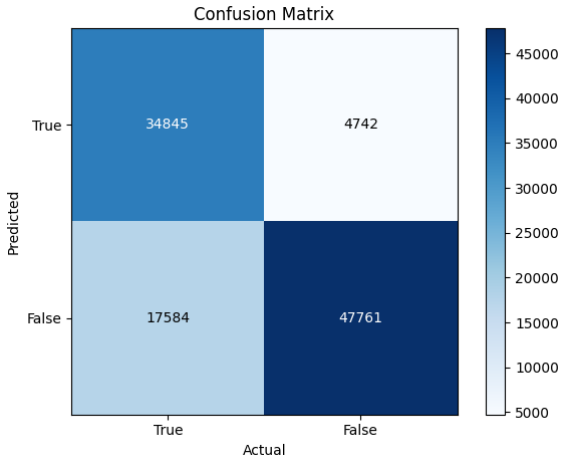


Figure 4.56: F4-SGD-LR(0.001) Confusion Matrix

4.5. 2B1 Model

Training of **2B1-SGD** was stopped after 42 epochs due to the presence of fluctuating validation loss and unstable accuracy(**Figure 4.59**). Despite these issues, other metrics indicate high values for this model. On the other hand, both **2B1-RMSprop(Figure 4.60)** and **2B1-Adam**(**Figure 4.61**) exhibit clear signs of underfitting, suggesting that they were unable to adequately learn from the data.

Even though **2B1-Adadelta** obtained lower values, its performance, as depicted in **Figure 4.62**, demonstrates better fit and learning from the data. However, it is worth noting that both accuracy and loss did not reach a stable level towards the end of training. Additional training and further fine-tuning might improve the model's performance.

When applying the learning rate reducer, the learning rate in **2B1-SGD-LR(0.015)** reached around 0.0015 in an attempt to fit the data. However, as depicted in **Figure 4.63**, these attempts were unsuccessful. Conversely, in **2B1-SGD-LR(0.001)**, the learning rate remained the same and resulted in a better fit to the data. Nevertheless, **Figure 4.64** illustrates a similar fluctuating behavior in the accuracy plot for this model.

All the results and plots related to this model are demonstrated in **Table 4.5** and from **Figure 4.57** to **Figure 4.70**

Table 4.5: 2B1 Results

Name	epochs	Acc	Loss	AUC	Confusion Matrix				Precision	Recall	Specificity	F1	F2	MCC
					TP	TN	FP	FN						
2B1-SGD	42	0.79	0.444	0.866	34992	47755	4748	17437	0.881	0.667	0.91	0.759	0.701	0.5947
2B1-RMSprop	19	0.79	0.417	0.868	35018	47761	4742	17411	0.881	0.668	0.91	0.76	0.702	0.5953
2B1-Adam	10	0.79	0.461	0.831	36057	46368	6135	16372	0.855	0.688	0.883	0.762	0.716	0.5822
2B1-Adadelta	199	0.74	0.446	0.867	46198	31969	20534	6231	0.692	0.881	0.609	0.775	0.836	0.5092
2B1-SGD-lr(0.015)	10	0.74	0.454	0.868	47235	30580	21923	5194	0.683	0.901	0.582	0.777	0.847	0.5099
2B1-SGD-lr(0.001)	61	0.79	0.444	0.866	36042	46366	6137	16387	0.855	0.687	0.883	0.762	0.715	0.5818

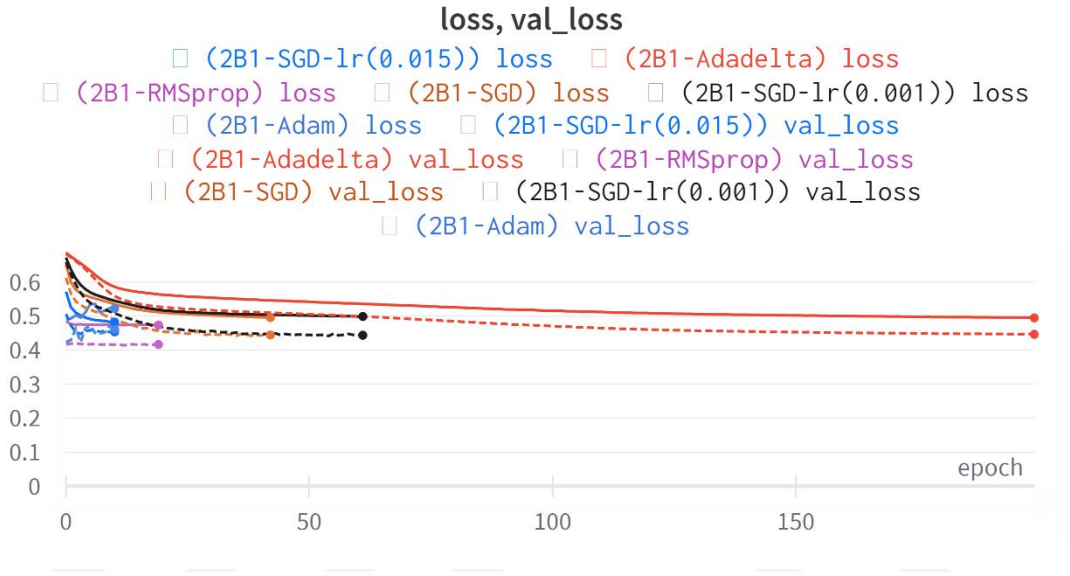


Figure 4.57: 2B1 models loss, validation loss

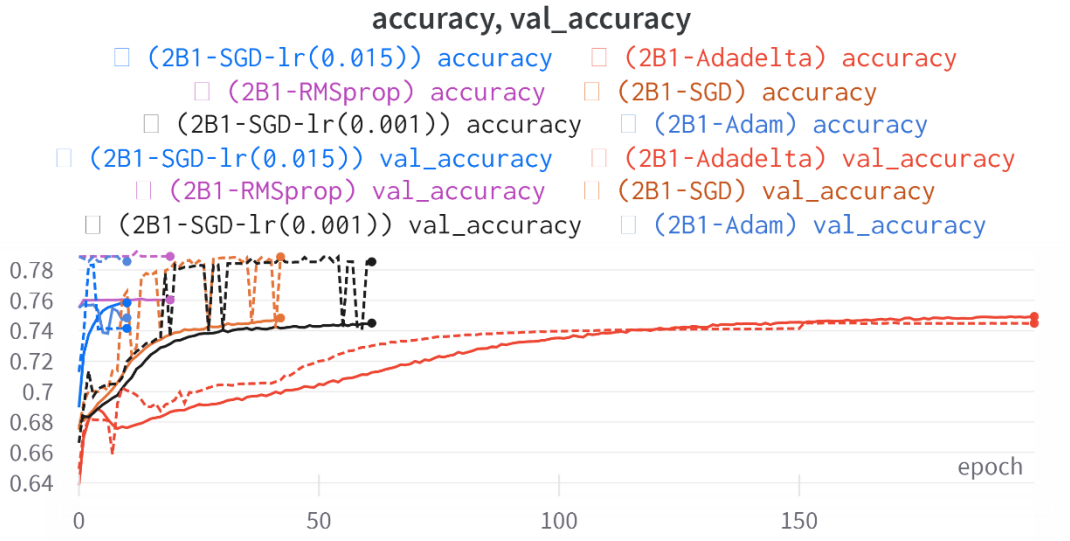


Figure 4.58: 2B1 models accuracy, validation accuracy

Figure 4.59: the loss in 2B1-SGD and the validation loss, as well as the accuracy and the validation accuracy

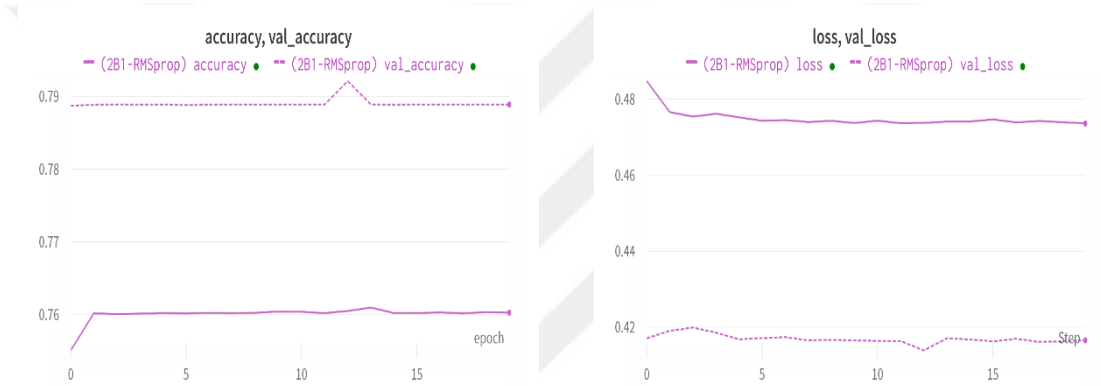


Figure 4.60: the loss in 2B1-RMSprop and the validation loss, as well as the accuracy and the validation accuracy

Figure 4.61: the loss in 2B1-Adam and the validation loss, as well as the accuracy and the validation accuracy

Figure 4.62: the loss in 2B1-Adadelta and the validation loss, as well as the accuracy and the validation accuracy

Figure 4.63: the loss in 2B1-SGD-LR(0.015) and the validation loss, as well as the accuracy and the validation accuracy

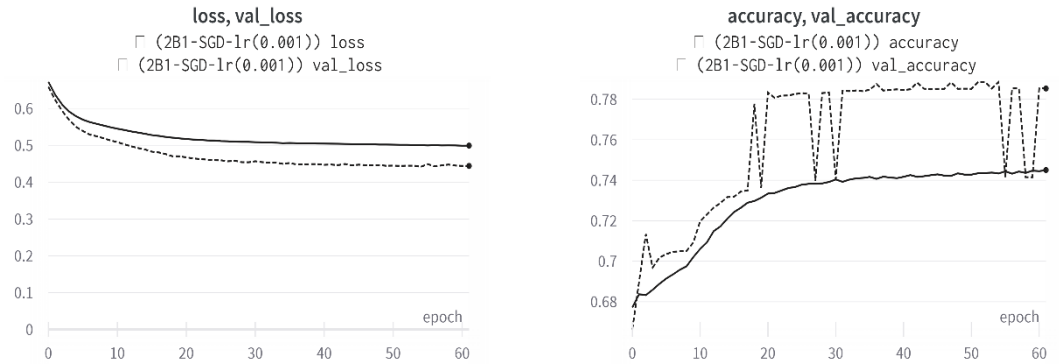


Figure 4.64: the loss in 2B1-SGD-LR(0.001) and the validation loss, as well as the accuracy and the validation accuracy

Following are the CM for this model Variations:

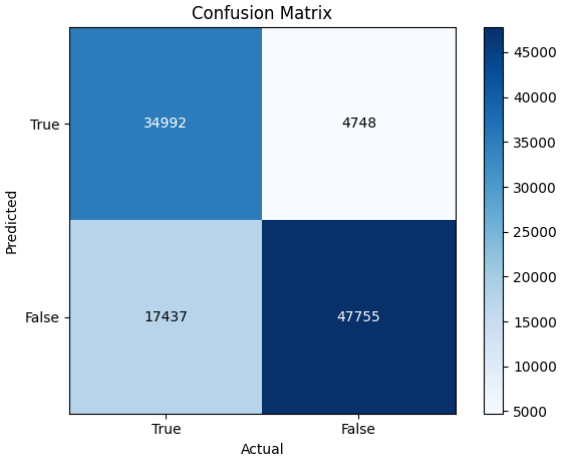


Figure 4.65: 2B1-SGD Confusion Matrix

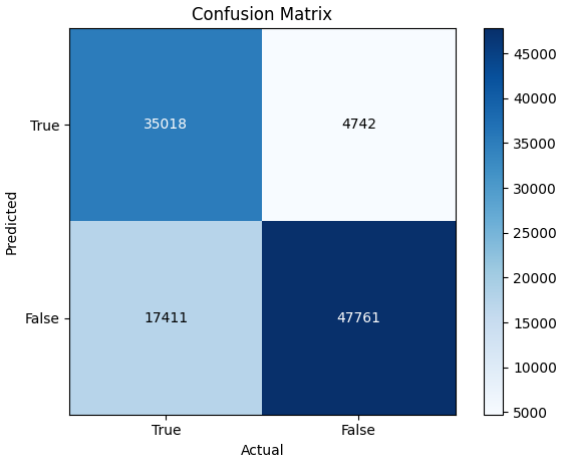


Figure 4.66: 2B1-RMSprop Confusion Matrix

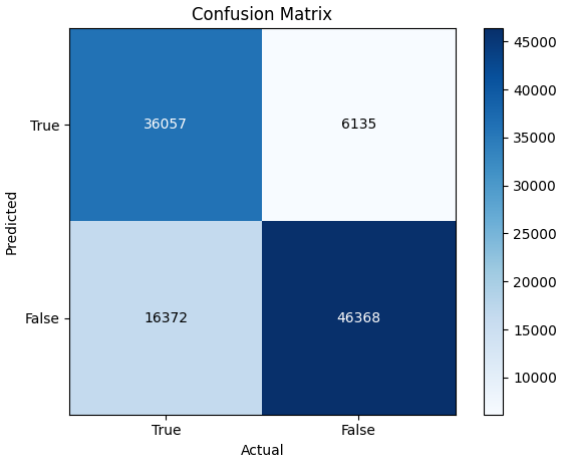


Figure 4.67: 2B1-Adam Confusion Matrix

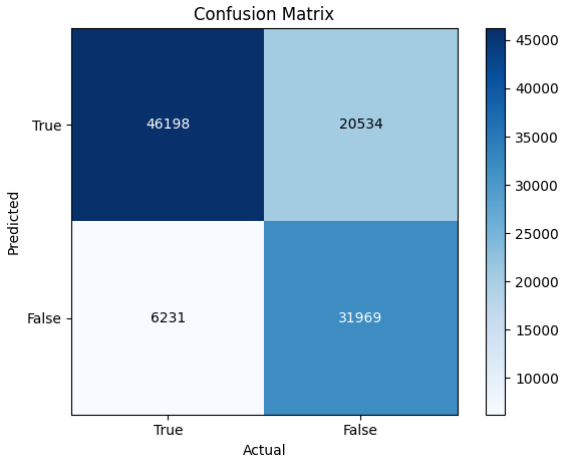


Figure 4.68: 2B1-Adadelta Confusion Matrix

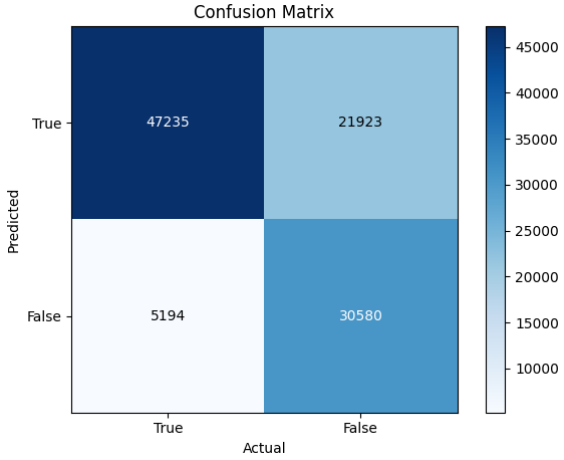


Figure 4.69: 2B1-SGD-LR(0.015) Confusion Matrix

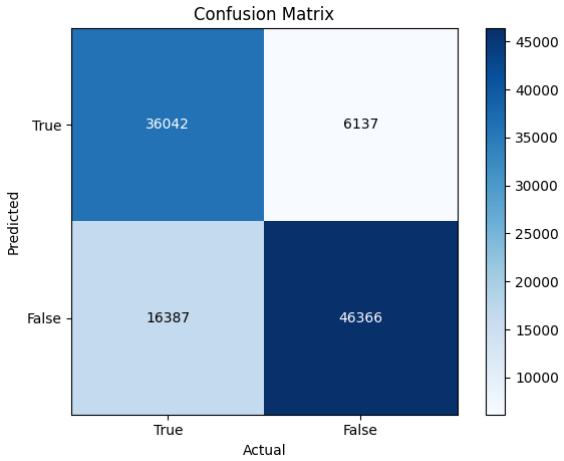


Figure 4.70: 2B1-SGD-LR(0.001) Confusion Matrix

4.6. 2B2 Model

Among the models examined, only 3 of them were able to effectively learn and generalize: **2B2-SGD**, **2B2-Adadelta**, and **2B2-SGD-LR(0.001)**. In the case of **2B2-SGD-LR(0.001)**, the learning rate reducer reduced the learning rate value to 0.000015.

2B2-Adadelta took the longest training time, halting after 106 epochs.

When comparing these three models using various evaluation metrics listed **Table 4.6**, it can be observed that they exhibit very similar values. However, **2B2-Adadelta** slightly outperformed the others in terms of F1 and F2 scores, achieving values of 0.782 and 0.759 respectively. On the other hand, **2B2-SGD** had a slightly higher Matthews Correlation Coefficient (MCC) of 0.5915. These results suggest that a lower initial learning rate may be necessary, and 2B2-Adadelta might benefit from hyperparameter tuning to improve its performance.

All the results and plots related to this model are demonstrated in **Table 4.6** and from **Figure 4.71** to **Figure 4.84**, **Figure 4.70**.

Table 4.6: 2B2 Results

Name	epochs	Acc	Loss	AUC	Confusion Matrix				Precision	Recall	Specificity	F1	F2	MCC
					TP	TN	FP	FN						
2B2-SGD	22	0.79	0.392	0.888	38787	44532	7971	13642	0.83	0.74	0.848	0.782	0.756	0.5915
2B2-RMSprop	8	0.8	0.398	0.886	38908	44558	7945	13521	0.83	0.742	0.849	0.784	0.758	0.5942
2B2-Adam	7	0.79	0.402	0.888	39028	44359	8144	13401	0.827	0.744	0.845	0.784	0.76	0.5923
2B2-Adadelta	106	0.79	0.394	0.887	39020	44281	8222	13409	0.826	0.744	0.843	0.783	0.759	0.5906
2B2-SGD-lr(0.015)	10	0.74	0.424	0.884	48272	29567	22936	4157	0.678	0.921	0.563	0.781	0.86	0.518
2B2-SGD-lr(0.001)	29	0.79	0.394	0.887	38762	44508	7995	13667	0.829	0.739	0.848	0.782	0.756	0.5906

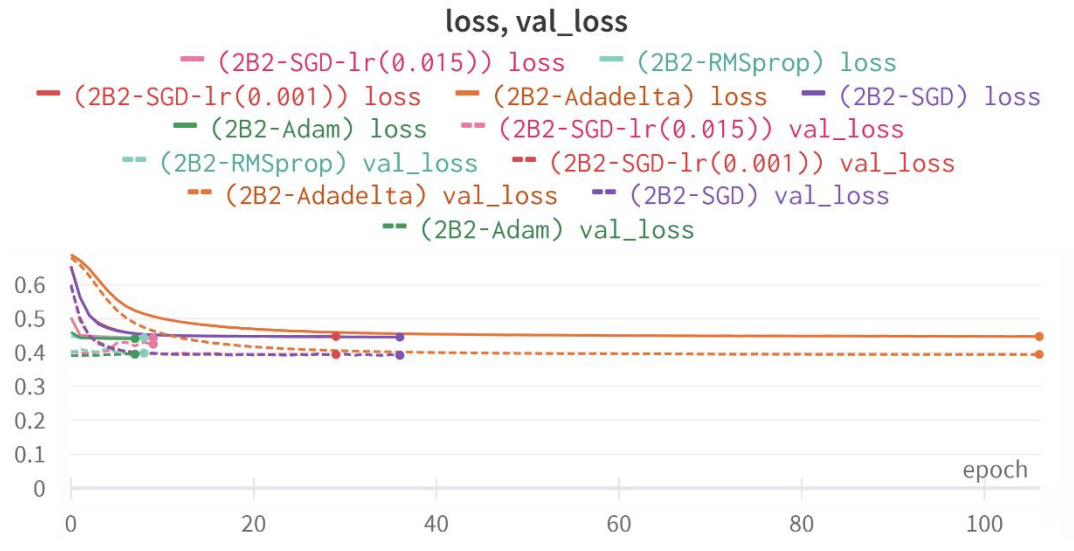


Figure 4.71: 2B2 loss, validation loss

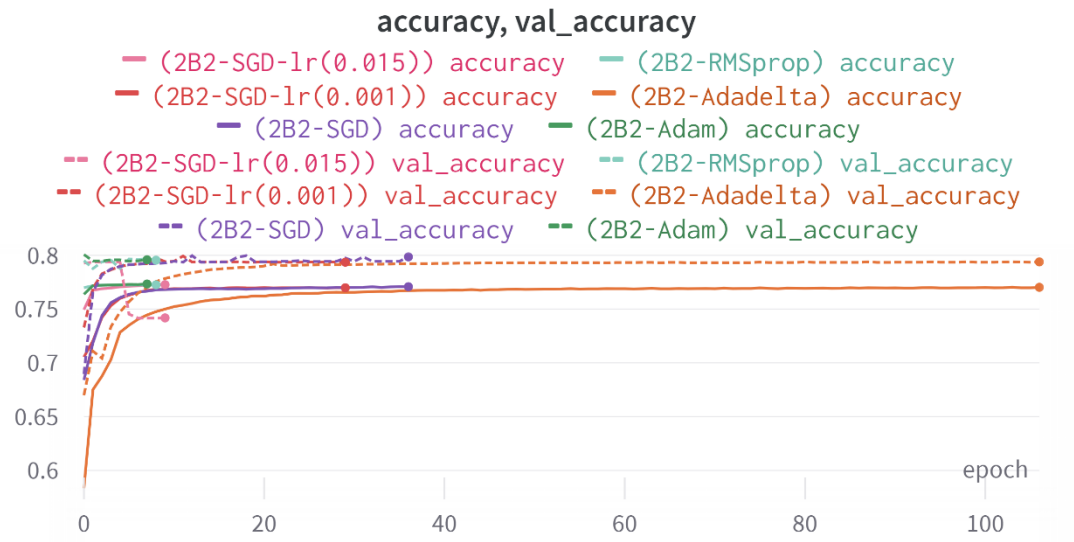


Figure 4.72: 2B2 accuracy, validation accuracy

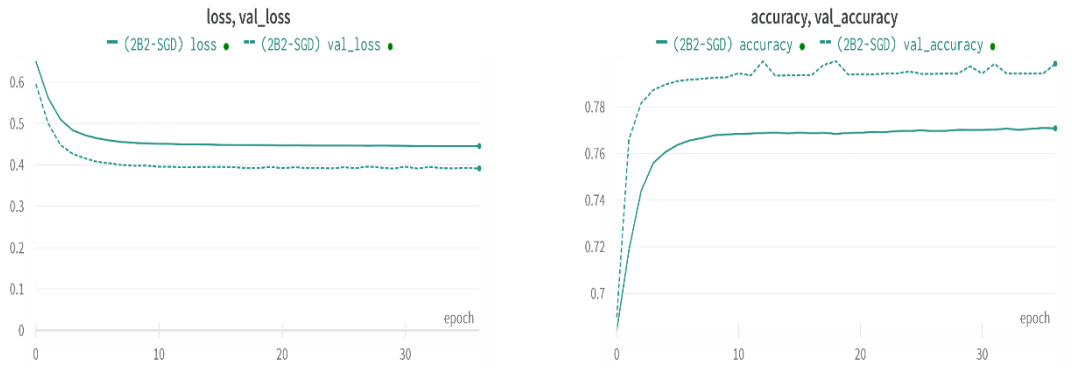


Figure 4.73: the loss in 2B2-SGD and the validation loss, as well as the accuracy and the validation accuracy

Figure 4.74: the loss in 2B2-RMSprop and the validation loss, as well as the accuracy and the validation accuracy

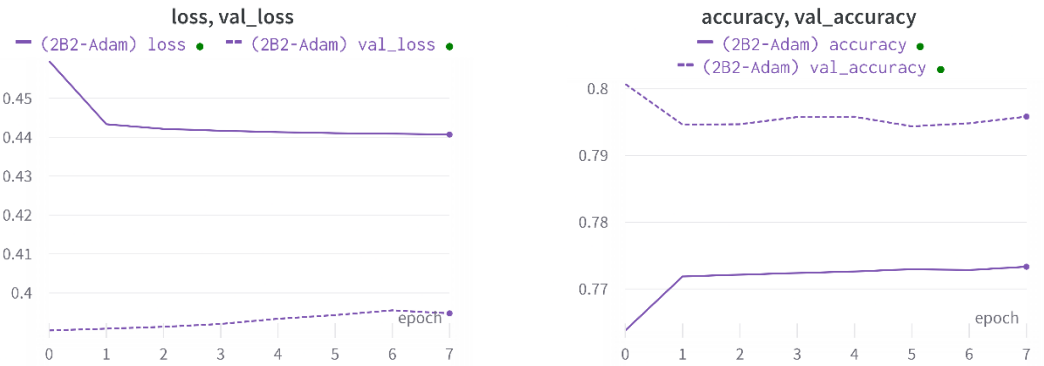


Figure 4.75: the loss in 2B2-Adam and the validation loss, as well as the accuracy and the validation accuracy

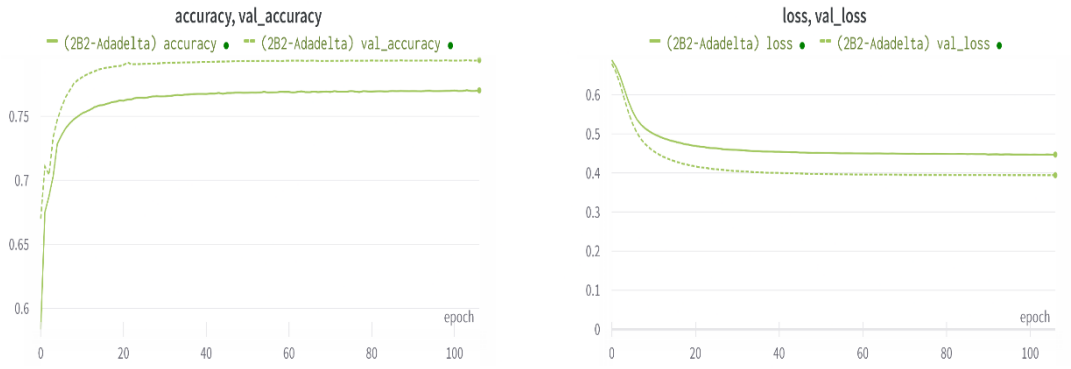


Figure 4.76: the loss in 2B2-Adadelta and the validation loss, as well as the accuracy and the validation accuracy

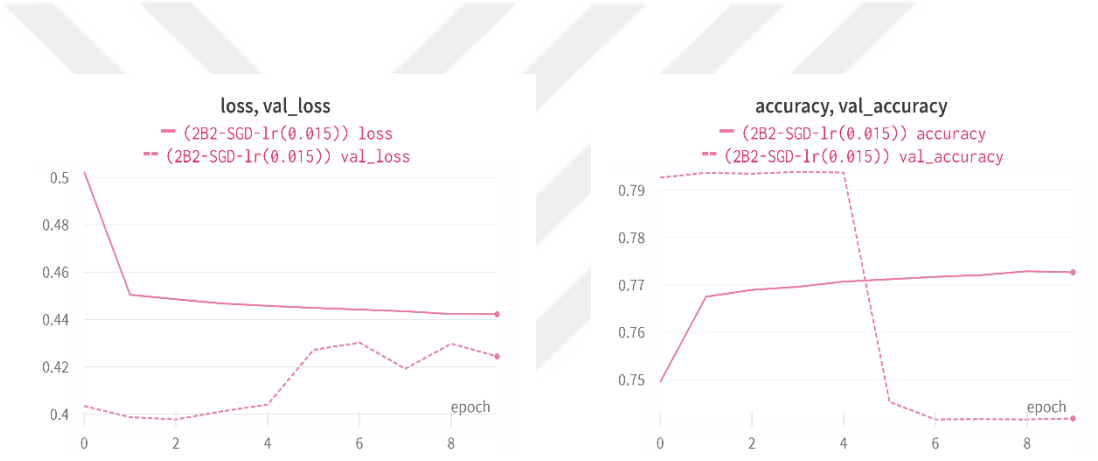


Figure 4.77: the loss in 2B2-SGD-LR(0.015) and the validation loss, as well as the accuracy and the validation accuracy

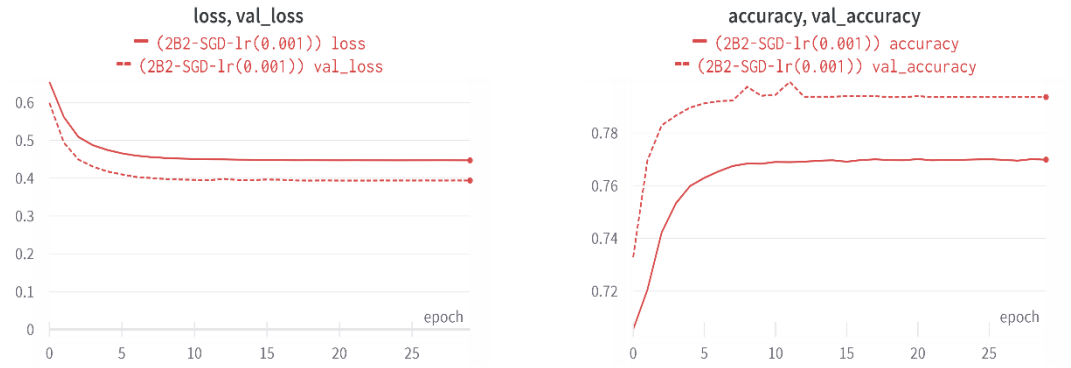


Figure 4.78: the loss in 2B2-SGD-LR(0.001) and the validation loss, as well as the accuracy and the validation accuracy

Following are the CM for this model Variations:

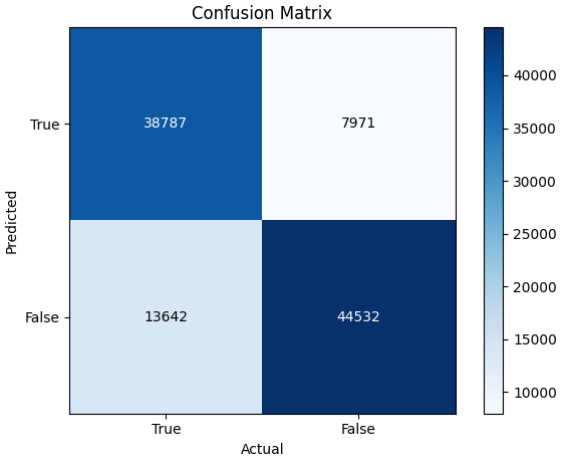


Figure 4.79: 2B2-SGD Confusion Matrix

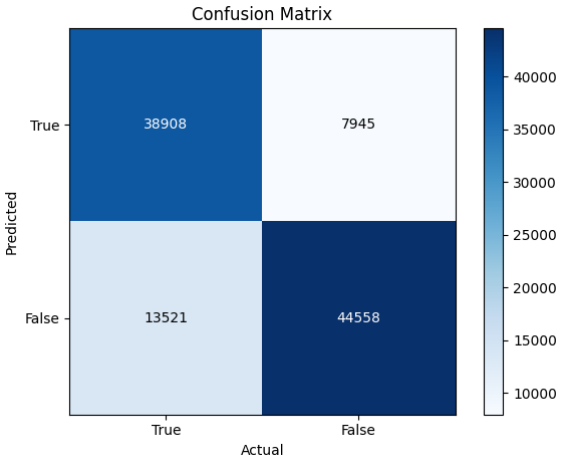


Figure 4.80: 2B2-RMSprop Confusion Matrix

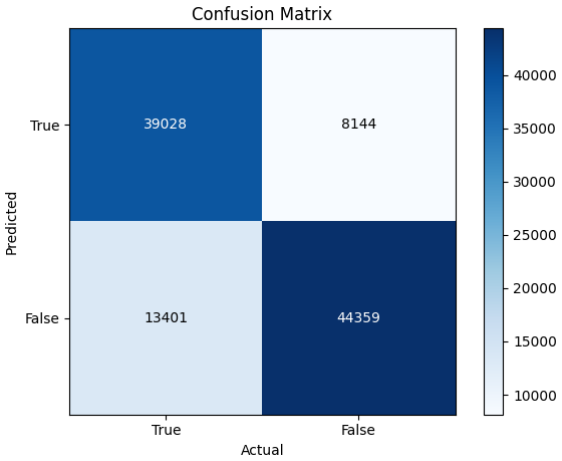


Figure 4.81: 2B2-Adam Confusion Matrix

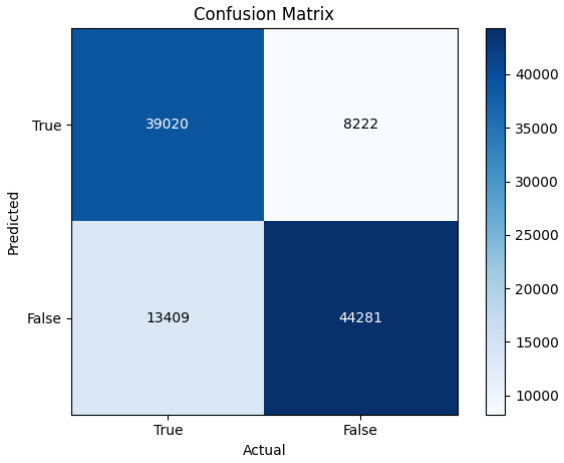


Figure 4.82: 2B2-Adadelta Confusion Matrix

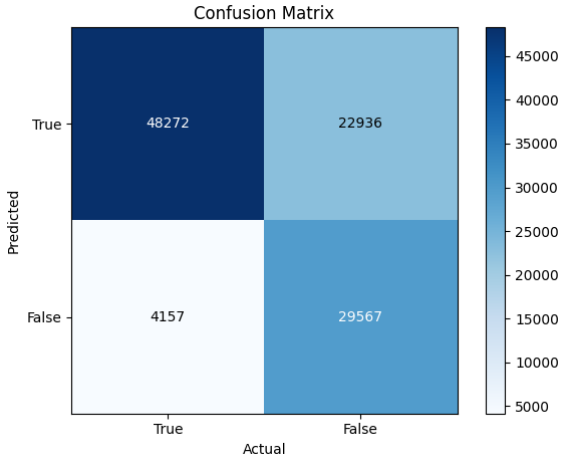


Figure 4.83: 2B2-SGD-LR(0.015) Confusion Matrix

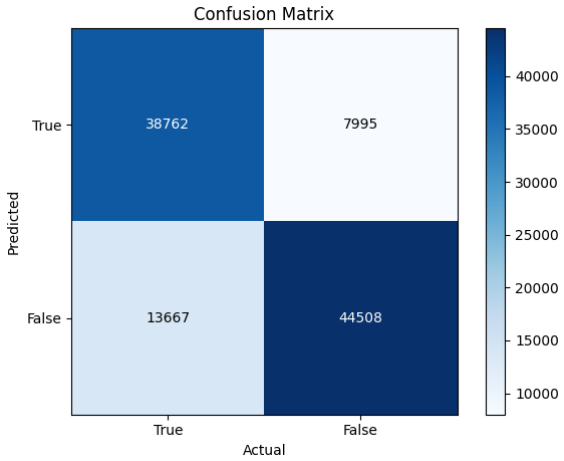


Figure 4.84: 2B2-SGD-LR(0.001) Confusion Matrix

4.7. 2B3 Model

In different runs on this architecture, the models **2B3-RMSprop** (**Figure 4.88**, **Figure 4.94**) and **2B3-Adam** (**Figure 4.89**, **Figure 4.95**) failed to converge and did not learn anything from the data.

On the other hand, the 2B3 model with the SGD optimizer, both with and without the learning rate reducer, showed similar results and demonstrated a good fit with relatively high metrics. Among all the models proposed in this work and using different optimizers, the **2B3-SGD** and **2B3-SGD-LR(0.015)** achieved the highest MCC value(**Table 4.7**). In the **2B3-SGD-LR(0.015)** model, while attempting to optimize the learning rate, the reducer decreased its value to (1.500000053056283e-07). This change had an impact on the loss, reducing it to 0.413 instead of 0.414. There was a slight improvement in specificity. It's worth mentioning that these values were obtained by training for only 60 epochs instead of 69 without the reducer.

The Adadelta optimizer also demonstrated high performance, **2B3-Adadelta** Model performed similarly to **2B3-SGD-LR(0.001)** in most metrics, achieving a Precision of 0.88, Recall of 0.668, Specificity of 0.91, F1 score of 0.759, an F2 score of 0.702, and a MCC of 0.5947.

All the results and plots related to this model are demonstrated in **Table 4.7** and from **Figure 4.85** to **Figure 4.98**.

Table 4.7: 2B3 Results

Name	epochs	Acc	Loss	AUC	Confusion Matrix				Precision	Recall	Specificity	F1	F2	MCC
					TP	TN	FP	FN						
2B3-SGD	69	0.79	0.414	0.87	35386	47727	4776	17043	0.881	0.675	0.909	0.764	0.708	0.6007
2B3-RMSprop	12	0.79	0.417	0.868	35018	47760	4743	17411	0.881	0.668	0.91	0.76	0.702	0.5953
2B3-Adam	25	0.645	0.494	0.747	52429	15287	37216	0	0.585	1	0.291	0.738	0.876	0.4127
2B3-Adadelta	199	0.79	0.419	0.868	35013	47739	4764	17416	0.88	0.668	0.91	0.759	0.702	0.5947
2B3-SGD-lr(0.015)	60	0.79	0.413	0.87	35387	47726	4777	17042	0.881	0.675	0.91	0.764	0.708	0.6007
2B3-SGD-lr(0.001)	59	0.79	0.419	0.867	35011	47748	4755	17418	0.88	0.668	0.91	0.76	0.702	0.595

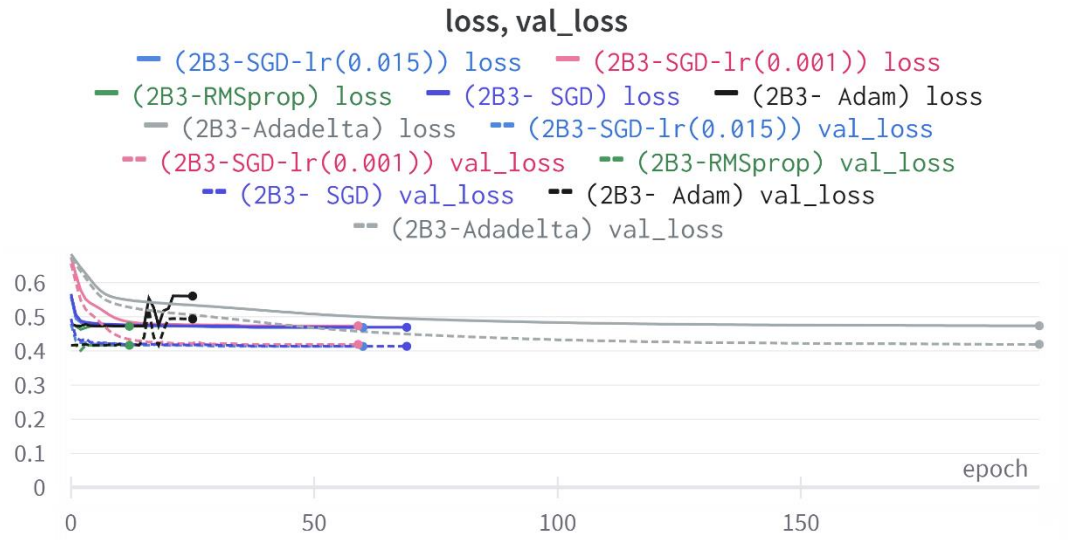


Figure 4.85: 2B3 loss, validation loss

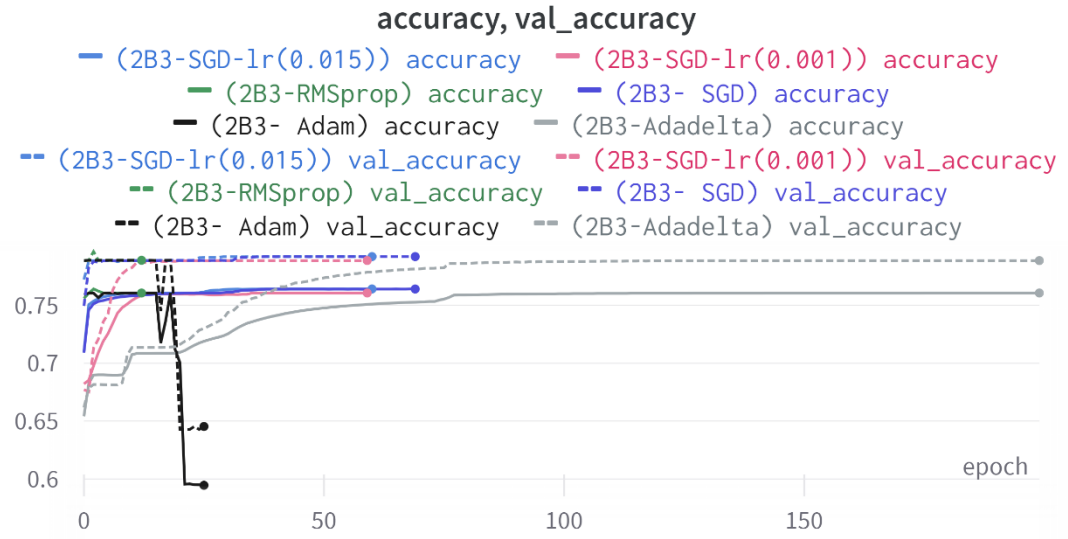


Figure 4.86: 2B3 accuracy, validation accuracy

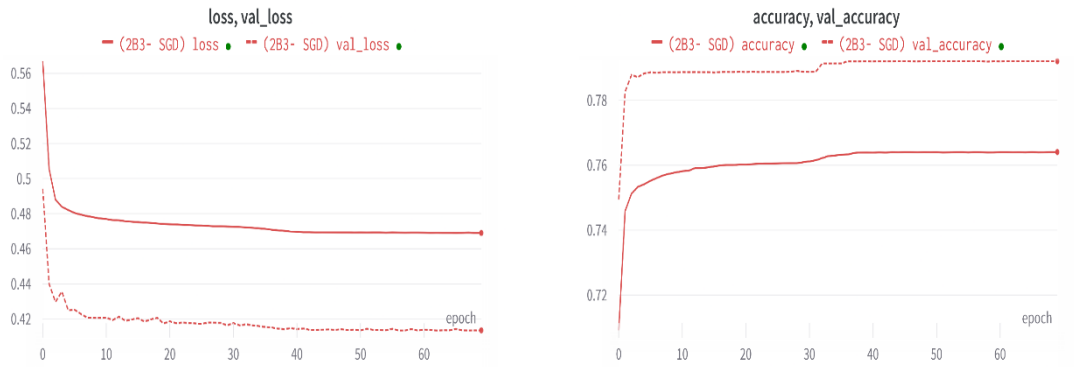


Figure 4.87: the loss in 2B3-SGD and the validation loss, as well as the accuracy and the validation accuracy

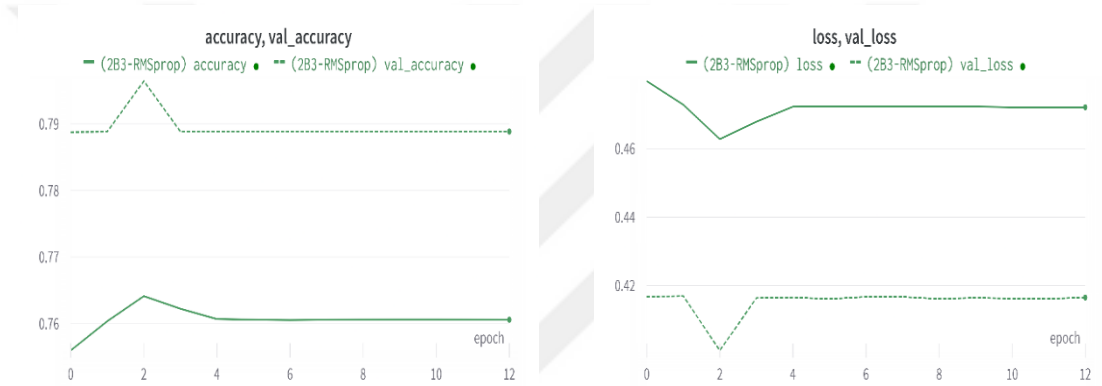


Figure 4.88: the loss in 2B3-RMSprop and the validation loss, as well as the accuracy and the validation accuracy

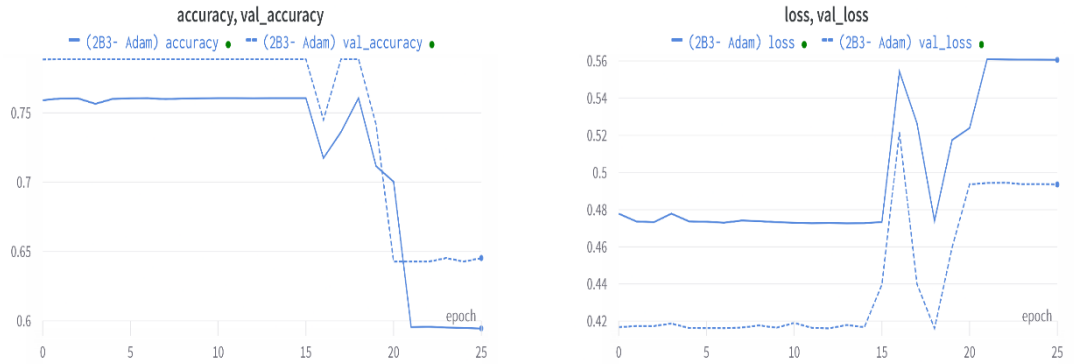


Figure 4.89: the loss in 2B3-Adam and the validation loss, as well as the accuracy and the validation accuracy

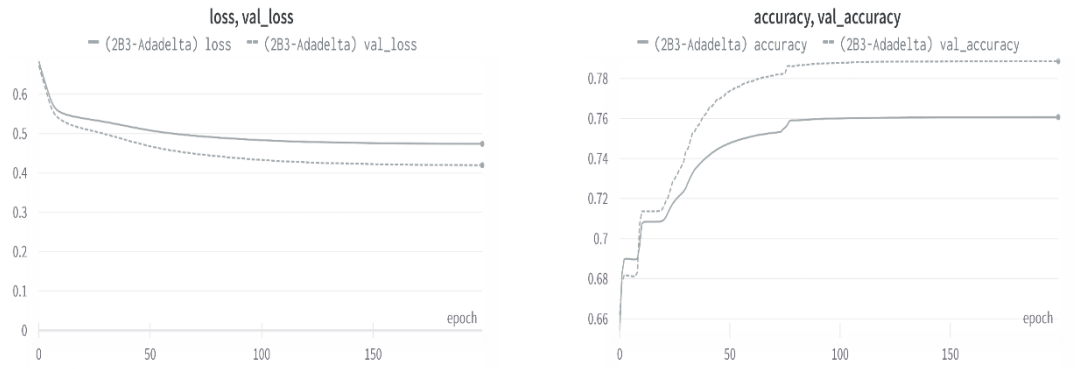


Figure 4.90: the loss in 2B3-Adadelta and the validation loss, as well as accuracy and the validation accuracy

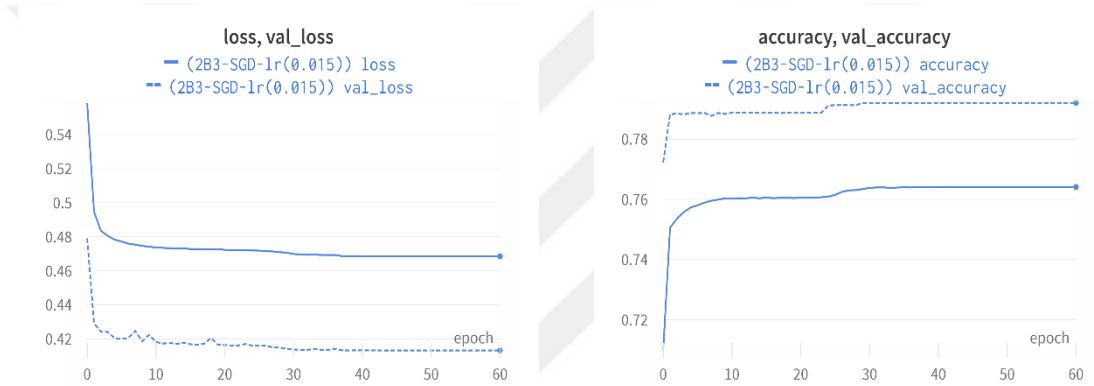


Figure 4.91: The loss in 2B3-SGD-LR(0.015) and the validation loss, as well as the accuracy and validation accuracy

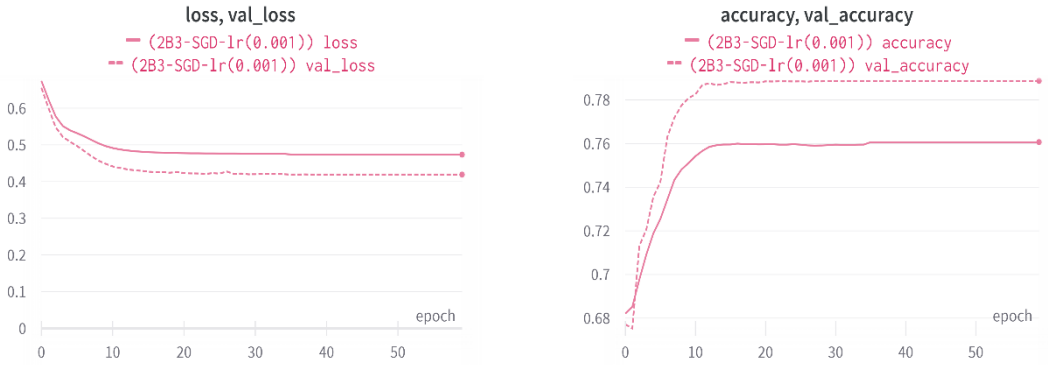


Figure 4.92: The loss in 2B3-SGD-LR(0.001) and the validation loss, as well as the accuracy and the validation accuracy

Following are the CM for this model Variations:

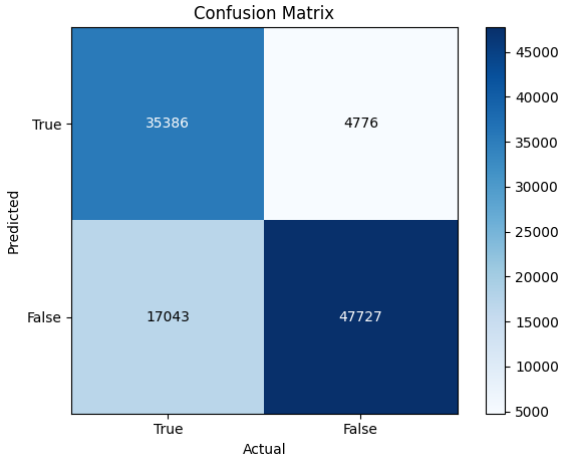


Figure 4.93: 2B3-SGD Confusion Matrix

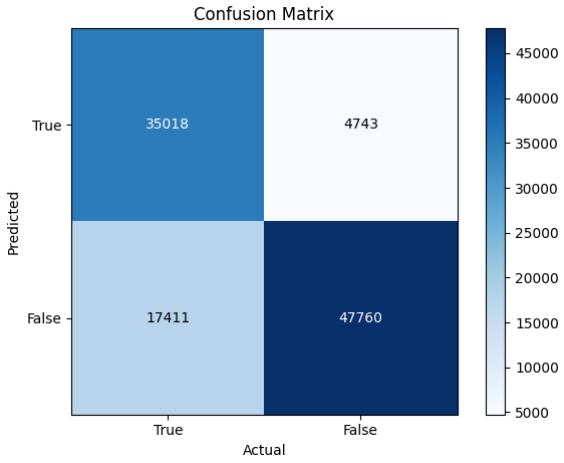


Figure 4.94: 2B3-RMSprop Confusion Matrix

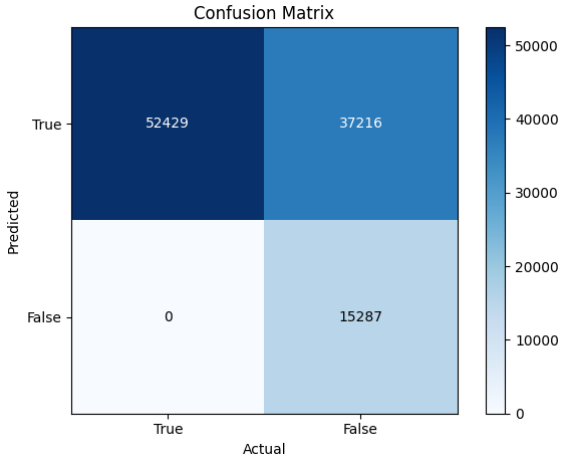


Figure 4.95: 2B3-Adam Confusion Matrix

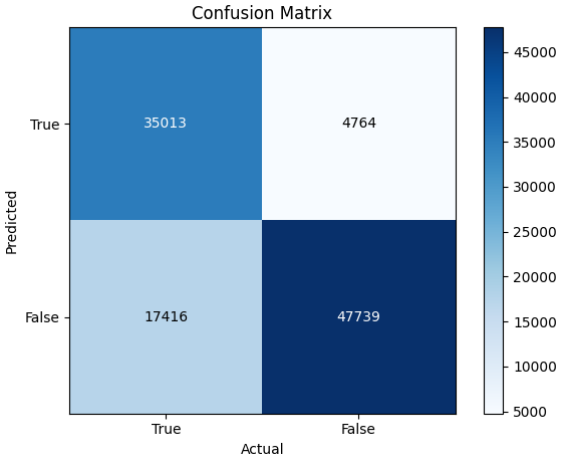


Figure 4.96: 2B3-Adadelta Confusion Matrix

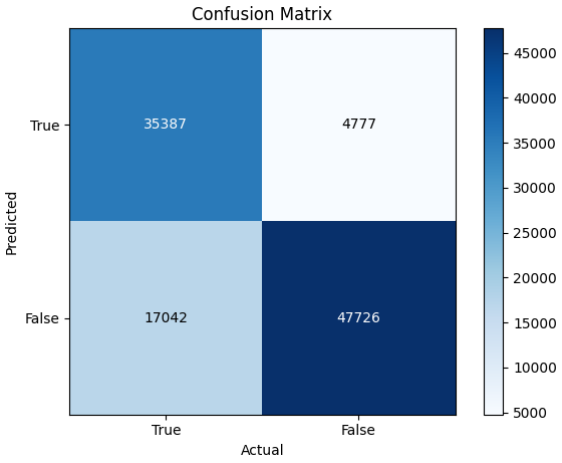


Figure 4.97: 2B3-SGD-LR(0.015) Confusion Matrix

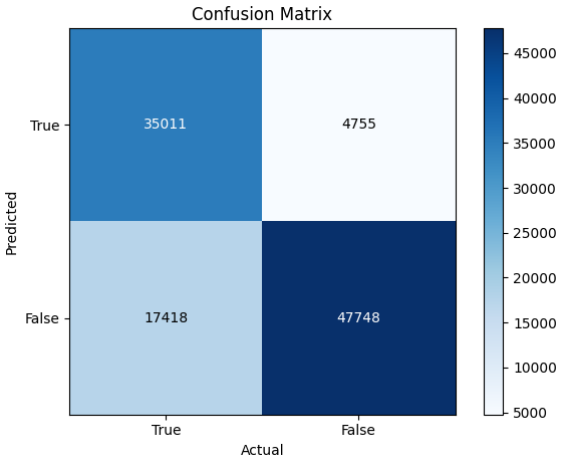


Figure 4.98: 2B3-SGD-LR(0.001) Confusion Matrix

In this study, multiple models were trained using different optimizers. The results show that the performance of the optimizers varied across the models. The SGD optimizer consistently showed good performance in terms of accuracy and convergence. RMSprop and Adadelta also performed well in some models. However, the Adam optimizer exhibited unstable behavior and struggled to converge. Overall, SGD was the most efficient optimizer in the majority of cases, while Adam showed less effectiveness. Focus should be placed on optimizing the learning rate and monitoring metrics such as loss, accuracy, and convergence during training.

The observed unusual phenomena in loss curve, where the validation loss consistently remains lower than the training loss throughout the training process, can be attributed to several factors that need to be addressed for a more robust analysis.

One potential factor that could contribute to the unusual phenomena is the possibility of overfitting or underfitting due to skewed data. To mitigate this, it is recommended to employ stratified cross-validation. Stratified cross-validation ensures that each fold in the training process contains a proportional representation of the different classes or data characteristics, reducing the risk of biased model evaluation. Furthermore, the random selection of data used for training might not be fully representative of the entire collected dataset, particularly for the real users' data. To address this concern, it is advisable to perform a more thorough analysis of the data distribution and consider stratified sampling techniques or other approaches to ensure a fair and representative selection of instances.

To address these factors, it is recommended to conduct another training iteration while considering the following steps: Implement stratified cross-validation during the training process to account for any imbalances or biases in the dataset and improve the reliability of model evaluation. Perform a detailed analysis of the data distribution and consider stratified sampling or other sampling techniques to ensure a fair and representative selection of instances. By incorporating these steps, we can enhance the model training and evaluation process, address potential issues related to data augmentation and data selection, and improve the overall reliability of results.

CHAPTER 5

5. CONCLUSIONS AND FUTURE WORK

This thesis manifests the significance and widespread influence of social media, along with its implications and potential threats. It specifically emphasizes the importance of identifying Abnormal actors on social media platforms. The study's main focus is on the detection of fake accounts on Twitter using a binary classification approach. Through an extensive review of related works and surveys, various approaches and techniques employed in previous studies are examined, while also highlighting the significant limitations and drawbacks present in the current literature that this study aims to overcome.

Overall, the shortcomings of the related work include limited scope, dependency on historical data, potential evasion by fraudulent actors, costly implementation, lack of comprehensive features, limited generalizability, lack of comparative analysis, and incomplete coverage of spam detection.

By recognizing and addressing these limitations, this study seeks to contribute to the advancement of fake account detection on social media platforms, particularly on Twitter.

To tackle the problem, a dataset was collected and carefully prepared from diverse sources and perspectives. To cover a wide range of real-life scenarios, data augmentation techniques were applied. The dataset was structured into two formats, aligning with the requirements of two types of models. The first format consisted of user metadata alone, while the second format incorporated both user metadata and one of their tweets. This novel approach, which combines numerical and textual data, has been found to be more efficient in this particular task according to some recent studies.

Drawing upon deep learning techniques, seven different architectures were proposed and implemented. These models were tested using four different optimizers: SGD, RMSprop, Adam, and Adadelta. Furthermore, Additionally, to explore the impact of the learning

rate, models utilizing the SGD optimizer were trained twice, incorporating a learning rate reducer.

The implemented models comprised four single-branched models that utilized only user metadata, and three two-branched models that incorporated both user metadata and tweets.

To assess the performance of each model, a range of metrics were employed, including loss, accuracy, area under the curve (AUC), confusion matrix, precision, recall, specificity, F1 and F2 scores, and the Matthews Correlation Coefficient (MCC). These metrics offered comprehensive insights into the models' performance and effectiveness in classification tasks.

Contrary to the claims made previously, the incorporation of tweets alongside user metadata did not significantly impact the models' ability to classify fake accounts. The SGD optimizer consistently demonstrated good performance, while RMSprop and Adadelta also yielded favorable results in certain models. However, the Adam optimizer exhibited unstable behavior and struggled to converge.

Overall, SGD emerged as the most efficient optimizer in the majority of cases, whereas Adam showed comparatively less effectiveness. It is crucial to focus on optimizing the learning rate and closely monitor metrics such as loss, accuracy, and convergence during training.

In general, the results indicated that SGD optimizers and learning rate reducers showed promising outcomes in most cases, while RMSprop and Adam optimizers exhibited some issues. Adadelta performed well in certain models but had its limitations.

Considering the similarities observed in the results of this study and the scope for improvement, several avenues for future work can be explored, including:

1. exploring additional datasets and features: incorporating diverse datasets and additional features may enhance the accuracy and robustness of classification.
2. Generalization to different platforms: Extend the study to encompass multiple social media platforms, as each platform may have unique characteristics and challenges regarding fake account detection. Investigate the transferability of the models and their performance on different platforms.
3. Investigate the use of more advanced deep learning architectures, such as transformers and how such architectures can improve classification performance.

4. use of transfer learning: by using some pretrained models on some related tasks in the domain of fake account detection. This may leverage existing models and improve their performance.
5. Additional training and fine tuning: : Focus on further refining the proposed models that exhibit a good fit and classification ability. Emphasize additional training and fine-tuning strategies, particularly for models that have successfully converged and completed all training epochs without reaching a fixed and stable loss.

By pursuing these future directions, the field of fake account detection can continue to advance, leading to more effective and reliable approaches for identifying non-real human actors on social media platforms.

BIBLIOGRAPHY

[1] Keles, B., McCrae, N., & Grealish, A. (2020). A systematic review: the influence of social media on depression, anxiety and psychological distress in adolescents. International Journal of Adolescence and Youth, 25(1), 79-93.

[2] Akram, W., & Kumar, R. (2017). A study on positive and negative effects of social media on society. International Journal of Computer Sciences and Engineering, 5(10), 351-354.

[3] Panjwani, A. (2023, February 14). False social media posts are hindering earthquake relief efforts in Turkey. you can help stop that | Abbas Panjwani. The Guardian. Retrieved March 10, 2023, from https://www.theguardian.com/commentisfree/2023/feb/14/turkey-syria-earthquake-misinformation-relief-efforts-turkey

[4] Romanov, A., Semenov, A., Mazhelis, O., & Veijalainen, J. (2017). Detection of fake profiles in social media. In Proceedings of the 13th International Conference on Web Information Systems and Technologies (WEBIST 2017) (pp. 363-369).

[5] Russell, S. J., & Norvig, P. (2010). Artificial Intelligence: A Modern Approach. Pearson Education.

[6] Nilsson, N. J. (1998). Artificial Intelligence: A New Synthesis. Morgan Kaufmann

[7] Poole, D. L., Mackworth, A. K., & Goebel, R. (1998). Computational Intelligence: A Logical Approach. Oxford University Press.]

[8] Alpaydin, E. (2010). Introduction to Machine Learning. MIT Press .

[9] "Data Science," Google Books, 2018. [Online]. Available: https://books.google.com.tr/books?id=UlpVDwAAQBAJ&printsec=frontcover&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false. [Accessed: Jun. 12, 2023]

[10] Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer.

[11] Resource: Murphy, K. P. (2012). Machine Learning: A Probabilistic Perspective. MIT Press.

[12] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

[13] Goldberg, Y. (2017). Neural Network Methods for Natural Language Processing. Morgan & Claypool Publishers.

[14] Jurafsky, D., & Martin, J. H. (2019). Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Pearson

[15] D. Khurana, A. C. Koli, Kiran Khatter, and S. Singh, "Natural language processing: state of the art, current trends and challenges," vol. 82, no. 3, pp. 3713–3744, Jul. 2022, doi: https://doi.org/10.1007/s11042-022-13428-4. [Online]. Available: https://link.springer.com/article/10.1007/s11042-022-13428-4. [Accessed: Jun. 12, 2023]

[16] "Exploratory Data Analysis with R," *Google Books*, 2016. [Online]. Available: https://books.google.com.tr/books?hl=en&lr=&id=XcskDAAAQBAJ&oi=fnd&pg=PP6&dq=Peng,+R.+D.+(2016).+Exploratory+Data+Analysis+with+R.+Springer&ots=rE1QmGa4f4&sig=Lbx8XFYvPC1L5T3ZgCke2dZlsCw&redir_esc=y#v=onepage&q&f=false. [Accessed: Jun. 12, 2023].

[17] C. M. Bishop, "Pattern Recognition and Machine Learning," SpringerLink, 2016, doi: https://doi.org/10.1007-978-0-387-45528-0. [Online]. Available: https://link.springer.com/book/9780387310732. [Accessed: Jun. 12, 2023]

[18] Data Mining: Concepts and Techniques by Jiawei Han, Micheline Kamber, and Jian Pei (Chapter 3)]

[19] Rubin, D. B. (2004).Introduction. In: Multiple imputation for nonresponse in surveys (Vol. 81). John Wiley & Sons.

[20] Aggarwal, C.C. (2017). Probabilistic and Statistical Models for Outlier Detection. In: Outlier Analysis. Springer, Cham. https://doi.org/10.1007/978-3-319-47578-3_2

[21] García, S., Luengo, J., & Herrera, F. (2015). Data preprocessing in data mining (pp. 63-94). Springer.

[22] Wu, X., & Kumar, V. (2009). The Top Ten Algorithms in Data Mining. CRC Press.

[23] Wei, J., Liang, Y., & Dong, Y. (2020). Data augmentation for natural language processing: A survey. Frontiers in Big Data, 3, 26.

[24] Wei, J., & Zou, K. (2019). EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In Proceedings of the 2019 Conference

on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP) (pp. 6382-6387).

[25] Zhang, Y., Sun, Y., & Jin, X. (2019). Augmenting data with paraphrases for neural machine translation. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP) (pp. 3379-3384).

[26] Fawaz, H. I., Forestier, G., Weber, J., Idoumghar, L., & Muller, P. A. (2019). Deep learning for time series classification: a review. Data Mining and Knowledge Discovery, 33(4), 917-963.

[27] Wang, Z., Yan, W., Oates, T., & Li, L. (2019). Time series augmentation for deep learning: a survey. arXiv preprint arXiv:1902.03932.

[28] Kwon, Y. H., Hong, S., & Lee, J. (2021). A Comprehensive Review of Audio Data Augmentation for Deep Learning. Sensors, 21(8), 2655.

[29] Jiang, R., Lu, Z., & Wen, Y. (2021). A Survey on Data Augmentation Techniques for Tabular Data. arXiv preprint arXiv:2104.02798.

[30] Khan, N. A., Khalid, S., & Khan, S. U. (2021). A survey on tabular data augmentation techniques for deep learning-based classification. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 11(4), e1402.

[31] Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. Journal of Big Data, 6(1), 60

[32] Singh, N., & Tiwari, S. (2021). A systematic survey of image data augmentation techniques for deep learning. Multimedia Tools and Applications, 80(11), 16675-16712.

[33] Perez-Rua, J. M., Ribeiro, E., & Nakamura, E. F. (2021). A comprehensive survey on data augmentation techniques for natural language processing and computer vision. Applied Sciences, 11(3), 1267.

[34] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Word embeddings: A survey. arXiv preprint arXiv:1301.3781.

[35] Jurafsky, D., & Russell, B. (2016). A survey of word embeddings. arXiv preprint arXiv:1611.07897.

[36] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444

[37] Lipton, Z. C., Kale, D. C., Elkan, C., & Wetzell, R. (2015). Learning to diagnose with LSTM recurrent neural networks. arXiv preprint arXiv:1511.03677.

[38] Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., & Schmidhuber, J. (2017). LSTM: A search space odyssey. IEEE transactions on neural networks and learning systems, 28(10), 2222-2232

[39] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780

[40] Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., & Schmidhuber, J. (2017). LSTM: A search space odyssey. IEEE transactions on neural networks and learning systems, 28(10), 2222-2232.

[41] Ratkiewicz, J., Conover, M. D., Meiss, M. R., Gonçalves, B., Flammini, A., & Menczer, F. (2013). Detecting and tracking political abuse in social media. ICWSM, 1(2), 297-306.

[42] Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine learning, 20(3), 273-297

[43] Schölkopf, B., & Smola, A. J. (2002). Learning with kernels: support vector machines, regularization, optimization, and beyond. MIT press.

[45] Song, J., Lee, S., Kim, J., 2015. CrowdTarget: Target-based Detection of Crowdturfing in Online Social Networks, in: Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security, CCS '15. ACM, New York, NY, USA, pp.793–804. doi:10.1145/2810103.2813661.

[46] Benevenuto, F., Magno, G., Rodrigues, T., & Almeida, V. (2010, July). Detecting spammers on twitter. In Collaboration, electronic messaging, anti-abuse and spam conference (CEAS) (Vol. 6, No. 2010, p. 12).

[47] Nazir, A., Raza, S., Chuah, C.-N., Schipper, B., 2010. Ghostbusting Facebook: Detecting and Characterizing Phantom Profiles in Online Social Gaming

Applications, in: Proceedings of the 3rd Wonference on Online Social Networks, WOSN'10. USENIX Association, Berkeley, CA, USA, pp. 1–1.

[48] Andriopoulou, F., Mavromatis, G., & Katsaros, D. (2018). Fake account detection in online social networks using graph-based features and ensemble learning. Journal of Network and Computer Applications, 121, 1-12.

[49] Idir, M. A., Hassouni, M. E., & Ouanan, M. (2016). Fake accounts detection in online social networks using graph-based features. International Journal of Advanced Computer Science and Applications, 7(2), 215-222.

[50] Cao, Y., Li, W., Zhang, J., 2011. Real-time traffic information collecting and monitoring system based on the internet of things, in: 2011 6th International Conference on Pervasive Computing and Applications. Presented at the 2011 6th International Conference on Pervasive Computing and Applications, pp. 45–49. doi:10.1109/ICPCA.2011.6106477

[51] Boshmaf, Y., Logothetis, D., Siganos, G., Lería, J., Lorenzo, J., Ripeanu, M., Beznosov, K., Halawa, H., 2016. Íntegro: Leveraging victim prediction for robust fake account detection in large scale OSNs. Comput. Secur. 61, 142–168. doi:10.1016/j.cose.2016.05.005.

[52] Egele, M., Stringhini, G., Kruegel, C., Vigna, G., 2015. Towards Detecting Compromised Accounts on Social Networks. IEEE Trans. Dependable Secure Comput. PP, 1–1. doi:10.1109/TDSC.2015.2479616.]

[53] Stringhini, G., Wang, G., Egele, M., Kruegel, C., Vigna, G., Zheng, H., Zhao, B.Y., 2013. Follow the Green: Growth and Dynamics in Twitter Follower Markets, in: Proceedings of the 2013 Conference on Internet Measurement Conference, IMC '13. ACM, New York, NY, USA, pp. 163–176. doi:10.1145/2504730. 2504731.

[54] Wang, G., Wilson, C., Zhao, X., Zhu, Y., Mohanlal, M., Zheng, H., Zhao, B.Y., 2012. Serf and Turf: Crowdturfing for Fun and Profit, in: Proceedings of the 21st International Conference on World Wide Web, WWW '12. ACM, New York, NY, USA, pp. 679–688. doi:10.1145/2187836.2187928.

[55] Wang, G., Wang, T., Zhang, H., Zhao, B.Y., 2014. Man vs. Machine: Practical Adversarial Detection of Malicious Crowdsourcing Workers, in: Proceedings of

the 23rd USENIX Conference on Security Symposium, SEC'14. USENIX Association, Berkeley, CA, USA, pp. 239–254

[56] Gurajala, S., White, J. S., Hudson, B., & Matthews, J. N. (2015, July). Fake Twitter accounts: profile characteristics obtained using an activity-based pattern detection approach. In Proceedings of the 2015 international conference on social media & society (pp. 1-7).

[57] Yuan, D., Miao, Y., Gong, N. Z., Yang, Z., Li, Q., Song, D., ... & Liang, X. (2019, November). Detecting fake accounts in online social networks at the time of registrations. In Proceedings of the 2019 ACM SIGSAC conference on computer and communications security (pp. 1423-1438).

[58] Lee, Kyumin & Caverlee, James & Webb, Steve. (2010). Uncovering social spammers: social honeypots + machine learning. 435-442. 10.1145/1835449.1835522.

[59] Crescia, S., Di Pietro, R., Petrocchi, M., Spognardi, A., & Tesconi, M. (2015). Fame for sale: Efficient detection of fake Twitter followers. Decision Support Systems, 80, 56-71.

[60] C. Chen, J. Zhang, X. Chen, Y. Xiang and W. Zhou, "6 million spam tweets: A large ground truth for timely Twitter spam detection," 2015 IEEE International Conference on Communications (ICC), London, UK, 2015, pp. 7065-7070, doi: 10.1109/ICC.2015.7249453.

[61] Gowda, S.R.S., Archana, B.R., Shettigar, P., Satyarthi, K.K. (2022). Sentiment Analysis of Twitter Data Using Naïve Bayes Classifier. In: Kumar, A., Senatore, S., Gunjan, V.K. (eds) ICDSMLA 2020. Lecture Notes in Electrical Engineering, vol 783. Springer, Singapore. https://doi.org/10.1007/978-981-16-3690-5_117

[62] M. Fazil and M. Abulaish, "A Hybrid Approach for Detecting Automated Spammers in Twitter," in IEEE Transactions on Information Forensics and Security, vol. 13, no. 11, pp. 2707-2719, Nov. 2018, doi: 10.1109/TIFS.2018.2825958.

[63] C. Yang, R. Harkreader, G. Gu, Empirical evaluation and new design for fighting evolving twitter spammers, IEEE Trans. Inf. Forensics Secur. 8 (8) (2013)1280–1293

[64] Wu, Tingmin & Liu, Shigang & Zhang, Jun & Xiang, Yang. (2017). Twitter spam detection based on deep learning. 1-8. 10.1145/3014812.3014815.

[65] Alom, Zulfikar & Carminati, Barbara & Ferrari, Elena. (2020). A deep learning model for Twitter spam detection. Online Social Networks and Media. 18. 100079. 10.1016/j.osnem.2020.100079.

[66] K. Lee, B. Eoff, and J. Caverlee. Seven Months with the Devils: A Long-Term Study of Content Polluters on Twitter. In Proceeding of the 5th International AAAI Conference on Weblogs and Social Media (ICWSM), Barcelona, July 2011.

[67] Honnibal, M., & Montani, I. (2017). spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. " https://spacy.io/ " accessed on 31.03.2023

[68] Google Research. (n.d.). Frequently asked questions (FAQ) - Colaboratory. Retrieved April 25, 2023, from https://research.google.com/colaboratory/faq.html#gpu-availability

[69] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[70] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in Advances in Neural Information Processing Systems 32 (NeurIPS 2019), 2019.

[71] Chollet, F. et al. Keras. (2015). GitHub repository. Retrieved from https://github.com/keras-team/keras

[72] Pandas. [Online]. Available: https://pandas.pydata.org/

[73] NumPy. [Online]. Available: https://numpy.org/

[74] Weights and Biases project. (2023). WandB. [website] Available at: https://wandb.ai [Accessed 01/05/2023].

[75] Google Colab notebook[online]. Availabe: https://colab.research.google.com/drive/1RX1UV4C6GR-0bVHD-6x2upCfKZghi3BW?usp=sharing

[76] Keras, "Optimizers," Keras Documentation, Available: https://keras.io/api/optimizers/. [Accessed: May 24, 2023].

[77] Keras. "Keras: SGD Optimizer." [Online]. Available: https://keras.io/api/optimizers/sgd/. [Accessed: May 24, 2023].

[78] Keras. "RMSprop Optimizer." Keras API Documentation. [Online]. Available: https://keras.io/api/optimizers/rmsprop/. [Accessed: May 24, 2023].

[79] "Adam Optimizer - Keras Documentation." Keras API. [Online]. Available: https://keras.io/api/optimizers/adam/. [Accessed: May 24, 2023].

[80] "AdaDelta Optimizer - Keras Documentation," Keras.io. [Online]. Available: https://keras.io/api/optimizers/adadelta/. [Accessed: May 24, 2023].

[81] Keras, "EarlyStopping," Keras.io, Available: https://keras.io/api/callbacks/early_stopping/. [Accessed: May 24, 2023].

[82] Keras. "ReduceLROnPlateau Callback." Keras API Documentation. Available online: https://keras.io/api/callbacks/reduce_lr_on_plateau/. [Accessed: May 24, 2023].

[83] Grandini, M., Bagli, E. and Visani, G., 2020. Metrics for multi-class classification: an overview. arXiv preprint arXiv:2008.05756.

[84] H. M and S. M.N, “A Review on Evaluation Metrics for Data Classification Evaluations,” International Journal of Data Mining & Knowledge Management Process, vol. 5, no. 2, pp. 01-11, Mar. 2015, doi: https://doi.org/10.5121/ijdkp.2015.5201. [Online]. Available: https://pdfs.semanticscholar.org/6174/3124c2a4b4e550731ac39508c7d18e520979.pdf

[84] OpenAI. "LSTM Layer - Keras API." Keras.io. [Accessed: May 26, 2023]. Available: https://keras.io/api/layers/recurrent_layers/lstm/

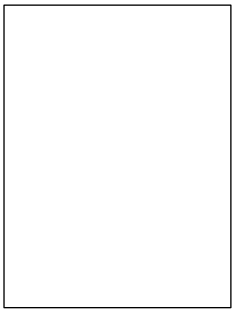
[85] OpenAI. "Flatten Layer - Keras API." Keras.io. [Accessed: May 26, 2023]. Available: https://keras.io/api/layers/reshaping_layers/flatten/

[86] Hernández-García, A. and König, P., 2018. Do deep nets really need weight decay and dropout?. arXiv preprint arXiv:1802.07042.

[87] Baldi, P. and Sadowski, P.J., 2013. Understanding dropout. Advances in neural information processing systems, 26.

[89] OpenAI. "Dense Layer - Keras API." Keras.io. [Accessed: May 26, 2023]. Available: https://keras.io/api/layers/core_layers/dense/

CURRICULUM VITAE



Name Surname :Ziad Walid Mohamed Abdelfattah ELGAMMAL

Place and Date of Birth :

E-Mail :

EDUCATION:

B.Sc. : 2021, Istanbul Medipol university, engineering and natural science, computer engineering

DETECTING FAKE TWITTER ACCOUNTS USING DEEP NEURAL NETWORK

ORIGINALITY REPORT

14%	8%	8%	5%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Zhaojing Huang, Luis Fernando Herbozo Contrera, Leping Yu, Nhan Duy Truong, Armin Nikpour, Omid Kavehei. "S4D-ECG: A shallow state-of-the-art model for cardiac arrhythmia classification", Cold Spring Harbor Laboratory, 2023 Publication	2%
2	Submitted to Universiti Teknologi Petronas Student Paper	1%
3	nebula.wsimg.com Internet Source	<1%
4	Submitted to Liverpool John Moores University Student Paper	<1%
5	Submitted to RMIT University Student Paper	<1%
6	www.eurecom.fr Internet Source	<1%
7	Submitted to Bournemouth University	