

LAMPIRAN



KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET DAN TEKNOLOGI
POLITEKNIK NEGERI SRIWIJAYA
Jalan Srijaya Negara, Palembang 30139 Telp. 0711-353414
Laman: <http://polsri.ac.id>, Pos El : info@polsri.ac.id



KESEPAKATAN BIMBINGAN TUGAS AKHIR (TA)

Kami yang bertanda tangan dibawah ini,

Pihak Pertama

Nama : Muhammad Fadli Ramadhan
NIM : 061940352343
Jurusan : Teknik Elektro
Program Studi : Sarjana Terapan Teknik Telekomunikasi

Pihak Kedua

Nama : Sopian Soim, S. T., M. T.
NIP : 197103142001121001
Jurusan : Teknik Elektro
Program Studi : Sarjana Terapan Teknik Telekomunikasi

Pada hari ini Jumat tanggal 05 Mei 2023 telah sepakat untuk melakukan konsultasi bimbingan Tugas Akhir (TA).

Konsultasi bimbingan sekurang- kurangnya 1 (satu) kali dalam satu minggu. Pelaksanaan bimbingan pada setiap hari Paku pukul 15.00 tempat di Politeknik Negeri Sriwijaya.

Demikianlah kesepakatan ini dibuat dengan penuh kesadaran guna kelancaran penyelesaian Tugas Akhir.

Pihak Pertama

(Muhammad Fadli Ramadhan)
NIM. 061940352343

Palembang, 05 Mei 2023
Pihak Kedua

(Sopian Soim, S. T., M. T.)
NIP. 197103142001121001

Mengetahui
Ketua Jurusan Teknik Elektro

(Ir. Iskandar Lutfi, M.T.)
NIP. 196501291991031002



KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET DAN TEKNOLOGI
POLITEKNIK NEGERI SRIWIJAYA

Jalan Srijaya Negara, Palembang 30139 Telp. 0711-353414
Laman: <http://polsri.ac.id>, Pos El : info@polsri.ac.id



KESEPAKATAN BIMBINGAN TUGAS AKHIR (TA)

Kami yang bertanda tangan dibawah ini,

Pihak Pertama

Nama : Muhammad Fadli Ramadhan
NIM : 061940352343
Jurusan : Teknik Elektro
Program Studi : Sarjana Terapan Teknik Telekomunikasi

Pihak Kedua

Nama : Lindawati, S. T., M. T. I.
NIP : 197105282006042001
Jurusan : Teknik Elektro
Program Studi : Sarjana Terapan Teknik Telekomunikasi

Pada hari ini Rabu tanggal 26 April 2023 telah sepakat untuk melakukan konsultasi bimbingan Tugas Akhir (TA).

Konsultasi bimbingan sekurang- kurangnya 1 (satu) kali dalam satu minggu. Pelaksanaan bimbingan pada setiap hari Selasa pukul 15.00 tempat di Politeknik Negeri Sriwijaya.

Demikianlah kesepakatan ini dibuat dengan penuh kesadaran guna kelancaran penyelesaian Tugas Akhir.

Pihak Pertama

(Muhammad Fadli Ramadhan)
NIM. 061940352343

Palembang, 26 April 2023
Pihak Kedua

(Lindawati, S. T., M. T. I.)
NIP. 197105282006042001

Mengetahui
Ketua Jurusan Teknik Elektro

(Ir. Iskandar Lutfi, M.T.)
NIP. 196507291991031002



**KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET DAN TEKNOLOGI**
POLITEKNIK NEGERI SRIWIJAYA
Jalan Srijaya Negara, Palembang 30139 Telp. 0711-353414
Laman: <http://polsri.ac.id>, Pos El : info@polsri.ac.id

















LEMBAR BIMBINGAN TUGAS AKHIR

Lembar : 1


Nadisma : Muhammad Fadli Ramadhan
 NIM : 061940352343
 Jurusan/Program Studi : Teknik Elektro/Sarjana Terapan Teknik Telekomunikasi
 Judul Tugas Akhir : "Peningkatan Performa Model *Neural Network* Menggunakan Optimasi *Hyperparameter Long Short Term Memory* untuk Deteksi Berita Palsu"
 Pembimbing I : Sopian Soim, S.T., M.T.

No.	Tanggal	Uraian Bimbingan	Tanda Tangan Pembimbing
1.	07/03 2023	Diskusikan konsep karya ilmiah dan teknik penulisan ilmiah	
2.	17/03 2023	Diskusikan bersama kerdosma untuk pendanaan TA	
3.	05/04 2023	Pengajuan Draft hasil percobaan	
4.	05/05 2023	Diskusikan model dan riset penelitian untuk paper	
5.	07/05 2023	Diskusikan Rekomendasi untuk submit jurnal	
6.	12/05 2023	Diskusikan metodologi penelitian untuk paper	

No.	Tanggal	Uraian Bimbingan	Tanda Tangan Pembimbing
7.	$\frac{24}{05}$ 2023	Pengajuan Draft Paper Jurnal JIKI (Jurnal Ilmu Komputer dan Informasi) Sinta 2.	
8.	$\frac{01}{06}$ 2023	Draft Bab I, II, <u>III</u> .	
9.	$\frac{05}{06}$ 2023	Diskusi hasil dan pembahasan penelitian	
10.	$\frac{06}{06}$ 2023	Pengajuan kembali draft Laporan TA, ACC BAB I, II, <u>III</u> .	
11.	$\frac{08}{06}$ 2023	Diskusi keputusan editor Jurnal JIKI (Ditolak); Revisi Paper	
12.	$\frac{12}{06}$ 2023	Pengajuan Progress Revisi Paper, Buat template Jurnal yang baru,	
13.	$\frac{13}{06}$ 2023	Diskusi Rekomendasi Jurnal Sinta 2	
14.	$\frac{19}{06}$ 2023	Diskusi informasi Penulis untuk Jurnal	
15.	$\frac{22}{06}$ 2023	Submit Jurnal SJI (Scientific Journal of Informatics) Sinta 2, dan Diskusi rencana Publikasi Sinta 1.	

No.	Tanggal	Uraian Bimbingan	Tanda Tangan Pembimbing
16.	$\frac{09}{07}$ 2023	Diskusikan keputusan editor Jurnal IJEECS Sinta 1, Revisi Paper dari Jurnal IJEECS	
17.	$\frac{24}{07}$ 2023	Diskusikan keputusan editor Jurnal SJI Sinta 2, Revisi Paper dari Jurnal IJEECS	
18.	$\frac{25}{07}$ 2023	Diskusikan Pencarian Kontributor Paper Jurnal SJI untuk Revisi editor Jurnal SJI, Pengajuan kembali draft Bab <u>IV</u> , <u>V</u>	
19.	$\frac{26}{07}$ 2023	ACC Bab <u>IV</u> , <u>V</u>	
20.	$\frac{27}{07}$ 2023	Acc Siap Ujian.	

Palembang, Agustus 2023
 Koordinator Program Studi
 Sarjana Terapan Teknik Telekomunikasi


Lindawati, S.T., M.T.I.
 NIP. 197105282006042001

Catatan:
 Lembar pembimbingan LA ini harus dilampirkan dalam Laporan Akhir.



**KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET DAN TEKNOLOGI**
POLITEKNIK NEGERI SRIWIJAYA
Jalan Srijaya Negara, Palembang 30139 Telp. 0711-353414
Laman: <http://polsri.ac.id>, Pos El : info@polsri.ac.id



LEMBAR BIMBINGAN TUGAS AKHIR

Lembar : 1


Nama : Muhammad Fadli Ramadhan
NIM : 061940352343
Jurusan/Program Studi : Teknik Elektro/Sarjana Terapan Teknik Telekomunikasi
Judul Tugas Akhir : "Peningkatan Performa Model *Neural Network* Menggunakan Optimasi *Hyperparameter Long Short Term Memory* untuk Deteksi Berita Palsu"
Pembimbing II : Lindawati, S.T., M.T.I.

No.	Tanggal	Uraian Bimbingan	Tanda Tangan Pembimbing
1.	26 April 2023	Konsultasi awal publikasi Jurnal	
2.	12 Mei 2023	Bimbingan rekomendasi Jurnal Sinta 2.	
3.	19 Mei 2023	Dis kusi dan submit Paper Jurnal Sinta 2 di Jurnal Ilmu Komputer dan InFormasi	
4.	06 Juni 2023	Konsultasi awal Laporan TA	
5.	08 Juni 2023	Dis kusi terkait Penolakan Paper di Jurnal Ilmu Komputer dan InFormasi Sinta 2.	
6.	13 Juni 2023	Dis kusi Revisi Paper Jurnal Ilmu Komputer dan InFormasi Sinta 2.	

No.	Tanggal	Uraian Bimbingan	Tanda Tangan Pembimbing
7.	20 Juni 2023	Diskusi hasil Revisi Paper dan Rekomendasi Jurnal Sinta 1 dan 2.	
8.	22 Juni 2023	Submit Paper Jurnal Sinta 2 di SCIENTIFIC JOURNAL OF INFORMATICS (SJI)	
9.	26 Juni 2023	Bimbingan BAB I, II, III Laporan TA	
10.	04 Juli 2023	Diskusi Persiapan Paper Jurnal Sinta 1.	
11.	07 Juli 2023	Submit Paper Jurnal Sinta 1 di Indonesian Journal of Electrical Engineering and Computer Science (IJECS)	
12.	09 Juli 2023	Revisi Paper Jurnal Sinta 1 IJECS	
13.	11 Juli 2023	ACC BAB I, II, III	
14.	24 Juli 2023	Revisi Paper Jurnal SJI Sinta 2	
15.	28 Juli 2023	Re Submit hasil Revisi Jurnal SJI Sinta 2	

16.	01 Agustus 2023	Bimbingan Bab <u>IV</u> dan <u>V</u>	<i>LD</i>
17.	02 Agustus 2023	Revisi Bab <u>IV</u> , <u>V</u> (Acc)	<i>LD</i>
18.	03 Agustus 2023	Acc Sidang TA	<i>LD</i>

Palembang, Agustus 2023
Koordinator Program Studi
Sarjana Terapan Teknik Telekomunikasi


Lindawati, S.T., M.T.I.
NIP. 197105282006042001

Catatan:
Lembar pembimbingan LA ini harus dilampirkan dalam Laporan Akhir.



**KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET DAN TEKNOLOGI**
POLITEKNIK NEGERI SRIWIJAYA
Jalan Srijaya Negara, Palembang 30139
Telp. 0711-353414 fax. 0711-355918
Website : www.polsri.ac.id E-mail : info@polsri.ac.id

**REKOMENDASI UJIAN TUGAS AKHIR**

Pembimbing Tugas Akhir memberikan rekomendasi kepada,

Nama : Muhammad Fadli Ramadhan
NIM : 0619 4035 2343
Jurusan/Program Studi : Teknik Elektro/Sarjana Terapan Teknik
Telekomunikasi
Judul Tugas Akhir : Peningkatan Performa Model *Neural Network*
Menggunakan Optimasi *Hyperparameter Long Short Term Memory* Untuk Deteksi Berita Palsu

Mahasiswa tersebut telah memenuhi persyaratan dan dapat mengikuti Ujian Tugas Akhir (TA) pada Tahun Akademik 2022/2023

Palembang, Agustus 2023

Pembimbing I,

Sopian Soim, S. T., M. T.
NIP. 196501291991031002

Pembimbing II,

Lindawati, S. T., M. T. I.
NIP. 197105282006042001



**KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET DAN TEKNOLOGI
POLITEKNIK NEGERI SRIWIJAYA
Jalan Srijaya Negara, Palembang 30139
Telp. 0711-353414 Fax. 0711-355918
Website : www.polsri.ac.id E-mail : info@polsri.ac.id**

PELAKSANAAN REVISI TUGAS AKHIR

Mahasiswa berikut,

Nama : Muhammad Fadli Ramadhan
 NIM : 0619 4035 2343
 Jurusan/Program Studi : Teknik Elektro/Sarjana Terapan Teknik Telekomunikasi
 Judul Tugas Akhir : Peningkatan Performa Model *Neural Network* Menggunakan Optimasi *Hyperparameter Long Short Term Memory* untuk Deteksi Berita Palsu

Telah melaksanakan revisi terhadap Proposal Tahapan Persiapan Tugas Akhir (TA) yang diseminarkan pada hari Kamis tanggal 10 bulan Agustus tahun 2023 Pelaksanaan revisi terhadap Proposal Tahapan Persiapan Tugas Akhir tersebut telah disetujui oleh Dosen Penguji yang memberikan revisi:

No.	Komentar	Nama Dosen Penguji *)	Tanggal	Tanda Tangan
1.	Sudah revisi	Ir. Jon Endri, M.T NIP 19620115 199303 1 001	28/08 2023	
2.	SUDAH DIREVISI	Hj. Adewasti, S.T., M.Kom NIP 19720114 200112 2 001	28/08 2023	
3.	ACC	Sopian Soim, S.T., M.T NIP 19710314 200112 1 001	24/08 2023	
4.	Ace	RA. Halimatussa'diyah, S.T., M.Kom NIP 19740602 200501 2 002	29/08 23	

Palembang, 28 Agustus 2023

Ketua Penguji,

(Ir. Jon Endri, M.T)

NIP 19620115 199303 1 001



[HOME](#) [ABOUT](#) [USER HOME](#) [SEARCH](#) [CURRENT](#) [ARCHIVES](#)

[Home](#) > [User](#) > [Author](#) > [Active Submissions](#)

Active Submissions

ACTIVE [ARCHIVE](#)

ID	MM-DD SUBMIT	SEC	AUTHORS	TITLE	STATUS
45420	06-22	ART	Lindawati, Ramadhan, Soim, Novianda	FAKE NEWS DETECTION MODELS PERFORMANCE IMPROVEMENT USING...	IN EDITING





ABOUT THE JOURNAL

[Focus and Scope](#)

[Author Guidelines](#)

Editor Decision

Decision	Accept Submission 2023-08-27
Notify Editor	 Editor/Author Email Record  2023-08-27
Editor Version	45420-117548-1-ED.DOCX 2023-07-04
Author Version	45420-119351-1-ED.DOCX 2023-07-28 DELETE
Upload Author Version	<input type="button" value="Choose File"/> <input type="text" value="No file chosen"/> <input type="button" value="Upload"/>

Scientific Journal of Informatics (SJI)

p-ISSN 2407-7658 | e-ISSN 2460-0040

Published By Department of Computer Science Universitas Negeri Semarang

Website: <https://journal.unnes.ac.id/nju/index.php/sji>

Email: sji@mail.unnes.ac.id



This work is licensed under a Creative Commons Attribution 4.0 International License.



M. Fadli Ramadhan <fadlinisasteam@gmail.com>

[SJI] Manuscript Acceptance Letter: 45420

5 messages

SJI Unnes <sjj@mail.unnes.ac.id>
To: fadlinisasteam@gmail.com

Thu, Aug 24, 2023 at 3:35 PM

Dear L Lindawati, Muhammad Fadli Ramadhan, Sopian Soim, Nabila Rizqi Noviana,

Thank you for submitting your manuscript to our journal, Scientific Journal of Informatics.

It's our pleasure to inform you that your paper entitled "Fake News Detection Models Performance Improvement Using Long Short-Term Memory Hyperparameter Optimization" has been **ACCEPTED (Conditional)** to be published in the upcoming issue of Scientific Journal of Informatics. In addition, we kindly request you to **Proofread** your final manuscript in order to prepare the submission of the Scientific Journal of Informatics to Scopus, and please send a certificate of proofreading via this email reply.

Based on the Author Fee Policy for the accepted paper, you may submit publication fee (IDR 1.705.000) to our official bank account:

Bank Account Number : 127001008677534
Bank Name : Bank Republik Indonesia (BRI)
Bank Account Name : Apri Dwi Lestari

Please complete the transaction before 28 August 2023 23:00:00.

Once you do full payment, please do the confirmation by sending your payment proof to this email. After the payment has been made and the final proofread manuscript has been uploaded, the status of the manuscript will be updated from "Accepted (Conditional)" to "Accepted".

For more information, please do not hesitate to contact us.

Thank you for your consideration.

DISCLAIMER

This electronic mail and/or any files transmitted with it may contain confidential or copyright information of Universitas Negeri Semarang and/or its Subsidiaries. If you are not an intended recipient, you must not keep, forward, copy, use, or rely on this electronic mail, and any such action is unauthorized and prohibited. If you have received this electronic mail in error, please reply to this electronic mail to notify the sender of its incorrect delivery, and then delete both it and your reply. Finally, you should check this electronic mail and any attachments for the presence of viruses. Universitas Negeri Semarang accepts no liability for any damages caused by any viruses transmitted by this electronic mail.

Editor Subject: [SJI] Editor Decision

DELETE

2023-

08-27

09:15

AM

Mr. Muhammad Fadli Ramadhan:

We have reached a decision regarding your submission to Scientific Journal of Informatics, "Fake News Detection Models Performance Improvement Using Long Short-Term Memory Hyperparameter Optimization".

Our decision is to: Accept Submission

Much Aziz Muslim
Universitas Negeri Semarang, Indonesia
a212muslim@yahoo.com

Scientific Journal of Informatics

<http://journal.unnes.ac.id/nju/index.php/sji>



Performance Improvement of Fake News Detection Models Using Long Short-Term Memory Hyperparameter Optimization

Lindawati^{1*}, Muhammad Fadli Ramadhan², Sopian Soim³, Nabila Rizqia Novianda⁴

^{1,2,3}Department of Electrical Engineering, Politeknik Negeri Sriwijaya, Indonesia

⁴Department of Electronics Engineering, National Chin-yi University of Technology, Taiwan

Abstract.

Purpose: The proposed model was developed based on prior research that distinguished between fake and real news using a deep learning-based methodology and an LSTM neural network, with a model accuracy of 99.88%. This study uses hyperparameter tuning techniques on a Long Short-Term Long Memory (LSTM) neural network architecture to improve the accuracy of a fake news detection model.

Methods: To improve the accuracy of the fake news detection model and optimize the model from previous research, this study uses the hyperparameter tuning technique on models with Long Short-Term Memory (LSTM) neural network architecture. For this technique, three different types of experiments, hyperparameter tuning on the LSTM layer, Dense layer, and Optimizer, were conducted to obtain the best hyperparameters in each layer of the model architecture and the model parameters proposed. The fake and real news dataset, which has also been used in earlier studies, was used in this study.

Results: The proposed model could detect fake news with a high accuracy of 99.97%, surpassing the previous research models with an accuracy of 99.88%.

Novelty: The novelty of this study was hyperparameter tuning technique on different layers of the LSTM neural network to optimize the fake news detection model. The research aims to improve upon previous approaches and increase the accuracy of the model.

Keywords: Fake news detection, Long short-term memory, Hyperparameter optimization, Performance improvement, Machine learning models.

Received June 2023 / Revised July 2023 / Accepted August 2023

This work is licensed under a [Creative Commons Attribution 4.0 International License](https://creativecommons.org/licenses/by/4.0/).



INTRODUCTION

Many emerging technologies help understand human behavior in this technological world. Emerging technologies can assist governments in developing safety policies. One of the unique technologies developed by John McCarthy in 1955 is artificial intelligence. Afterward, neural networks, machine learning, deep learning, natural language processing, and predictive analytics were developed [1]. The emergence of new technologies has significantly advanced in every field of life [2].

One of the emerging technologies that is altering how we handle business issues is artificial intelligence [3], [4]. Machine learning and advanced data analytics are being used by more and more companies to solve problems. Natural Language Processing (NLP), which has improved during the artificial intelligence era, provides much potential for companies wishing to interpret human behavior using existing data [5].

All forms of communication in social and natural settings, including audio, video, and text, can be used with NLP. Text mining has been helping identify numerous relevant patterns and trends in the textual collection. By strategically utilizing NLP in today's market environments, organizations can gain an edge over rivals. The vast amounts of unstructured data in various fields, as well as public opinion in the government sectors, in areas including healthcare, education, fake news, economic sectors, security, and trust, can be fought with artificial intelligence and natural language processing. The use of natural language processing improves communication between humans and robots, which improves decision-making and increases overall company productivity [6].

*Corresponding author.

Email addresses: lindawati@polsri.ac.id (Lindawati)

DOI: [10.15294/sji.v10i3.45420](https://doi.org/10.15294/sji.v10i3.45420)

Unrecognized fake news tends to spread more quickly. Fake news is currently more popular on social media platforms, such as Facebook News Feed, than in the past when it was prevalent in print. The rise of fake news has been connected to post-truth politics and political extremism. To better comprehend how fake news spreads, the authors [7] examined a dataset of Twitter rumor chains spanning from 2006 to 2017. The study aimed to understand and evaluate the influence of fake news on society. The authors observed the wide-scale distribution of inaccurate information, with around 3 million people spreading approximately 126,000 instances of fake news. Therefore, for the benefit of society and government backing, bogus news must be removed. Recognizing fake news before being spread is meaningful and valuable because of the fast-growing social media database and technological advances.

The importance of identifying fake news has been highlighted. A method has been devised to detect fake news, such as neural network method used in research [2], [8]–[12], which is very commonly used for classification cases. Apart from deep learning techniques, several traditional learning methods are also used for classification, such as Support vector machines (SVM) [13] and decision trees [14]. However, using deep learning methods such as neural networks is superior to these two methods because they are more scalable, and higher accuracy can be achieved by increasing the network size or training data set [15]. In addition, [16] shows that traditional decision trees and Support Vector Machine learning methods are inefficient for many modern applications. This means that traditional learning methods require a large number of observations to achieve generalization and impose significant manpower to determine prior knowledge in models. The neural network method can solve the problem of binary text classification to identify fake and real news. However, the neural network method has weaknesses, as it is difficult to select the optimal hyperparameter layers of the neural network model and requires much experimentation to determine a parameter. In addition, it is optimally used for a particular dataset.

Previous studies regarding the detection of fake news, [2], [8]–[12], used various optimization techniques and methods to enhance the performance of the neural network model. Studies [8]–[12] were chosen since they were compared in [2], so it is related to the comparison of the performance of the detection model in this research. The datasets used in these six studies differ, but in essence, the accuracy of the models proposed was assessed in the six studies, comparing the techniques used to create models for the fake news identification [17]. Kaliyar in [9] used the DeepFake multi-layer deep neural network method with a model accuracy of 88.64%. In addition to the DeepFake method, Goswami in [10] used the Echo FakeD method to optimize the coupled matrix-tensor factorization approach with the obtained model accuracy of 92.30%. Raj [12] used optimization of the neural network model using the coupled ConvNet method or the CNN framework for detecting fake news with a model accuracy of 93.56%. The BERT (Bidirectional Encoder Representations from Transformers) method can also be used to optimize deep learning techniques like CNN and LSTM to deal with the ambiguity of natural language understanding by models [18]. The method used by Narang in [8] achieved a model accuracy of 98.90%. Ozbay in [11] also used the neural network model optimization method using the improved Salp Swarm Optimization (SSO) method to detect fake news on social media, with a model accuracy of up to 99.50%. The effectiveness of the neural network model in the five previous studies was surpassed with the LSTM method by Chauhan [2], where the parameters used can achieve a model accuracy of 99.88%.

Referring to the background, a better neural network model to use for performing NLP tasks, especially for detecting fake news, is the LSTM neural network model. However, higher accuracy can be achieved by increasing the network size or training data set. This indicates that by choosing efficient and optimal hyperparameters for the fake news detection model and further research, the accuracy of neural network model can be increased [1], [19]. Therefore, we use the hyperparameter tuning method on the LSTM neural network model to improve the performance of the model for fake news detection. In this paper, the authors would optimize the hyperparameters in the LSTM neural network architectural model using the Hyperparameter tuning method to get better model accuracy performance achieved by [2]. Thus, the model proposed could help researchers and machine learning developers in the development of neural network models to prevent the spread of fake news on social media, such as Twitter.

METHODS

A system block diagram of the entire system is created as the first step in the research design process. In the research design process, flowchart is essential to give a general overview of how the research suite functions. This flowchart helps understand the steps in the research methodology. Consequently, a thorough research block diagram aids in developing a system that can work effectively. The steps taken in conducting the research method are shown in Figure 1.

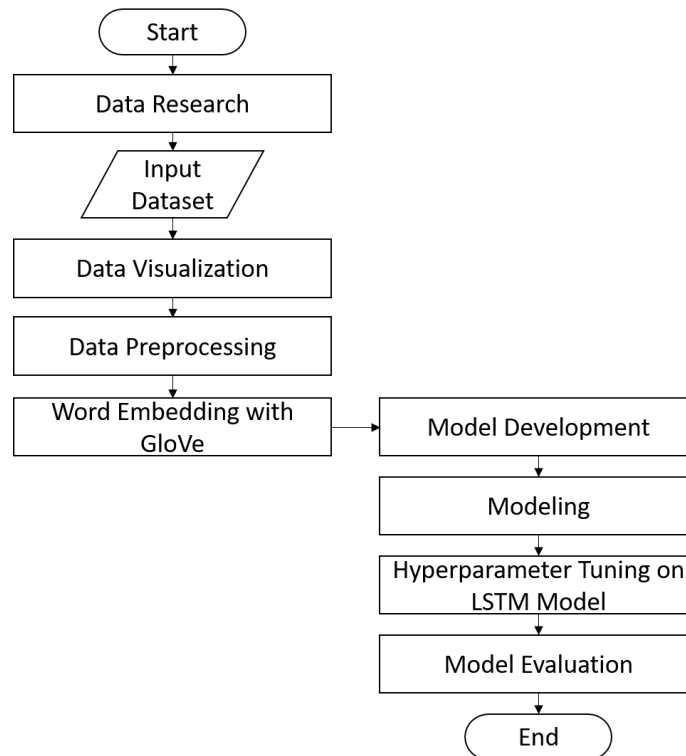


Figure 1. Flowchart of research methodology

Data Research

Data research examines the model used in previous research to get a better model evaluation value. Data research uses the Code feature on Kaggle in Jupyter Notebook and Python. The stages and detailed explanations of the data research are presented in the following sections.

Datasets

The public dataset² used is obtained from the Kaggle website, which has been tested in [20] and [2]. The dataset contains around 40,000 articles covering fake and real news. The fake and real news data are separated into two sets of around 20,000 articles. The dataset contains four labels, namely title, text, subject, and date. Glove Twitter³ is also used with pre-trained word embedding data available on the Kaggle website, which has been identified in [21]. The word embedding data has specifications of 2B tweets, 27B tokens, 1.2M vocabs, and 100-dimensional tweet vector word embedding. This information is useful for the input layer that embeds the model.

Data Visualization

Data visualization simplifies the understanding of comparative data by visually representing it using graphs or maps. This enhances the ability to identify trends and insights in large datasets, leveraging the human mind's natural analytical capabilities. The dataset is divided into two categories: real news (class '1') and fake news (class '0').

² <https://www.kaggle.com/datasets/clmentbisailon/fake-and-real-news-dataset>

³ Glove Twitter Source: <https://nlp.stanford.edu/projects/glove/>

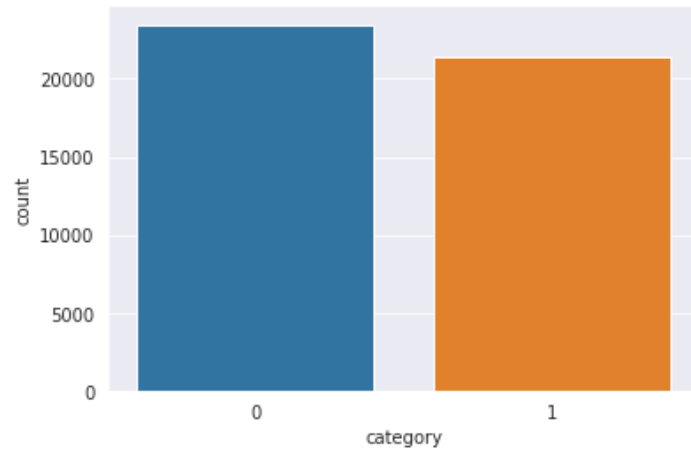


Figure 2. Dataset visualization

Figure 2 shows an article on the corresponding fake and real class labels. There are not many differences between the two datasets, so they look balanced. In addition, class '0' (blue) represents the wrong news category while '1' (orange) represents the real news category. After observing the graph, the number of datasets representing the original news category is 21,417 data, and the fake news category is 23,481, making it 44,898 total data. The corresponding subject, title, and date of each story can be omitted, leaving only the main text for further processing because the content in the subject section differs in the two categories.

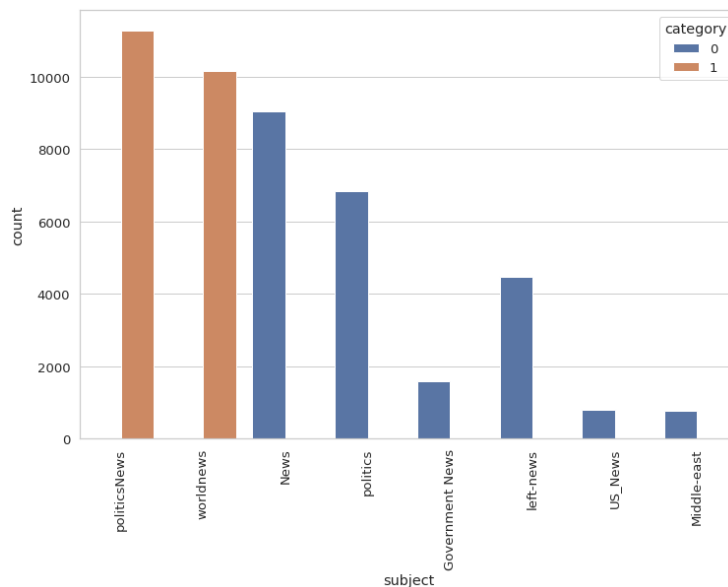


Figure 3. Topics on fake and real news categories

Figure 3 describes the various subjects that make up the dataset. The quantity of pertinent topics reflects how widely news is spread throughout the populace. The blue bar displays unreliable or false information, while the orange bar displays trustworthy and accurate news. Real news includes, among its topics, coverage of politics and world news. On the other hand, fake news discusses politics, the government, the left news, the United States, and Middle Eastern news.

Figures 4 and 5 display the keywords found in the real and fake news datasets, respectively. A word cloud is generated for each category, including up to 2000 words. Word clouds visualize groups of words by representing and emphasizing them in various sizes and lengths.

provider site for data scientists and machine learning developers, namely Kaggle⁴. Glove Twitter has a Massive data scale and Relevance to natural language processing. Twitter is a very popular social media platform, with millions of users sending millions of tweets every day. The data available from Twitter is huge, covers a wide variety of topics, and reflects the diversity of languages and writing styles that exist on the internet. This data is used for data input at the embedding layer of the neural network architecture. The model demonstrates how co-occurrence probability can be applied to the corpus to extract particular underlying meanings [24]. Consider the words a and b from the corpus to clarify this idea. The ratio of these two words' co-occurrence probabilities with probe words, c , can be used to confirm the relationship between them. For c related to a but not b , the ratio $\frac{Pa|c}{Pb|c} > 1$. Similarly, for c related to b but not a , the ratio $\frac{Pa|c}{Pb|c} < 1$. If c is either related to both a and b or not related to both a and b , the ratio $\frac{Pa|c}{Pb|c}$ is close to 1. Therefore, it implies that determining the ratio of the probabilities of word co-occurrence is the first step in learning word vectors. Eq.1 shows the model's general form.

$$F(w_a, w_b, w^{\#}_c) = \frac{Pa|c}{Pb|c} \tag{1}$$

where $w \in \mathbb{R}^d$ are word vectors and $w^{\#} \in \mathbb{R}^d$ are probe word vectors in the corpus.

Model Development

The research model is developed to create a fake news detection model with greater model accuracy than what earlier studies proposed. After conducting research on the processed dataset, this stage is attained. The process of creating a model, or modeling, and developing the model using Hyperparameter Tuning optimization are both included in the model development stage.

Modeling

After transforming the data at the data preprocessing stage, the next step is building a neural network model. The LSTM model has the advantage of solving sequential data problems. The challenge lies in dealing with sequences of different lengths, diverse vocabulary of input symbols, and the need for models to grasp long-term context and relationships within the input sequence of symbols. The Long Short-Term Memory (LSTM) model is employed for fake news detection.

The following is the LSTM and Hyperparameter model layer architecture in [2] which was used as an initial model for the hyperparameter tuning technique.

Table 1. LSTM layer architecture in previous research [2]

Layer (type)	Output size	Number of Paramet
Embedding_1 (embedding)	300 x 100	1,000,000
Lstm_1 (LSTM)	300 x 128	117,248
Lstm_2 (LSTM)	128	49,408
Dense_1 (Dense)	64	2080
Dense_2	1	33

Table 2. Hyperparameters in previous research [2]

Parameter	Value
Embedding layer	1
LSTM layer	2
Dense layer	2
Loss Function	Binary cross entropy
Activation function	ReLu
Optimizer	Adam
Learning_rate	0.01
Number of epochs	10
Embedding size	100
Batch size	256

⁴ <https://www.kaggle.com/datasets/bertcarremans/glovetwitter27b100dtx>

Hyperparameter Tuning on the LSTM Model

In this research, we used the Hyperparameter Tuning method on the LSTM model for the fake news detection model. This method contributes to improving the accuracy of the detection model in [2].

In the context of the LSTM (Long Short-Term Memory) model, hyperparameter tuning optimizes the hyperparameter values used to construct the LSTM model. Hyperparameters are parameters that the model does not directly learn, but they have an impact on the model's functionality, rate of convergence, and capacity. Finding the right mix of hyperparameters to produce the model with the best or optimal performance is the goal of this hyperparameter tuning. Model performance can be evaluated using metrics like accuracy, precision, recall, F1-score, and mean squared error (MSE), depending on the type of problem being solved or other pertinent evaluation metrics. Model accuracy is the evaluation metric used in this research.

To find the ideal combination that results in the best model performance, hyperparameter tuning involves experimenting with different combinations of these hyperparameter values. Finding the ideal hyperparameter combination can be done manually by repeatedly trying different values, or it can be done using optimization techniques like grid search, random search, or other heuristic methods [25]. The Hyperparameter Tuning process in this study is the LSTM model hyperparameter tuning. It can be done manually, and the hyperparameter combinations are used in the LSTM model architecture, as in Table 1 and Table 2. In this research, the Hyperparameters in the LSTM model optimized in this study are as follows:

1. The LSTM Layer includes the number of layers, the number of LSTM units, and dropouts. For hyperparameter tuning experiments on the LSTM layer, 4 hyperparameter combination experiments are conducted.
2. Layer Dense includes the number of layers, number of dense units, and dropouts. For hyperparameter tuning experiments on the LSTM layer, 8 hyperparameter combination experiments are conducted.
3. Optimizer includes the learning rate. For hyperparameter tuning experiments on the LSTM layer, 3 hyperparameter combination experiments are conducted.

The hyperparameters in this LSTM model are selected based on the results of the best model accuracy of each combination in the Hyperparameter Tuning experiment at each layer. Moreover, they are used as the proposed LSTM model architecture for comparison with the models in [2], which used the same fake news datasets and detection models.

The LSTM model architecture that has been optimized using the hyperparameter tuning technique is expected to be able to build a fake news detection model with better model accuracy performance than the detection model in [1].

Model Evaluation

Testing the system's capacity to recognize fake news is the main goal of system evaluation. Measuring the degree to which the LSTM model can identify and differentiate between real and fake news is a key component of evaluating the accuracy of the LSTM model on the fake news detection model. The model evaluation metric used in this research is model accuracy.

Table 3. Confusion matrix

	FALSE POSITIVE (FP)	FALSE NEGATIVE (FN)
TRUE POSITIVE	TP	FN
TRUE NEGATIVE	FP	TN

The effectiveness of the classification model is assessed using a confusion matrix. It provides a visual summary of the model's predicted results by comparing the actual predictions with those predicted by the model.

In the context of fake news detection, the confusion matrix can be used to measure how well the model can distinguish between fake news and true news. For example, the confusion matrix can show how much fake news is successfully detected correctly (TP), how much correct news is detected correctly (TN), how much correct news is incorrectly detected as fake (FP), and how many false news is detected as true (FN).

Accuracy in the confusion matrix is one of the evaluation metrics calculated based on its entries [26]. Accuracy measures the proportion of accurate predictions among all predictions made:

$$\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{TN} + \text{FP} + \text{FN}) \quad (2)$$

Based on entries in the confusion matrix, accuracy shows a general idea of how well the model can distinguish between false and real news. It should be understood, though, that accuracy can provide inaccurate information if the data is class unbalanced. The accuracy of the model under consideration in this research is expressed as accuracy metrics for the LSTM model's performance in identifying fake news from the dataset under consideration by the detection model.

RESULTS AND DISCUSSIONS

The LSTM model employed in the previous studies obtained the accuracy value of the detection model for predicting fake news datasets at 99.88%. The hyperparameter tuning optimization method, on the other hand, could enhance model accuracy by carrying out experiments to obtain the best LSTM model hyperparameter selection.

The results of this study are expected to improve the fake news detection model in [2] and increase the accuracy of the model in detecting fake and real news on social media by optimizing the hyperparameter setting of the LSTM Model.

The initial plan for the proposed LSTM optimization model architecture shows the hyperparameters for each layer in the LSTM neural network (Table 1). This model architecture was the experimental basis for better LSTM model optimization by modifying the hyperparameters for each layer in the LSTM model.

Real and fake news are distinguished by the LSTM model. The modeling processes were conducted in 15 experiments to determine the best hyperparameter to be used as the architecture of the LSTM model.

The following sections describe the experiments conducted in optimizing the LSTM model architecture Hyperparameters on the LSTM input layer, Dense layer, and Optimizer to get the Hyperparameter optimization results for each layer in the Proposed LSTM model.

Results of the Hyperparameter Optimization for Each Layer in the Proposed LSTM Model

The hyperparameters with the highest accuracy in each LSTM model layer as the LSTM model proposed are as follows:

1. Results of the Hyperparameter Tuning Experiments on the LSTM Layer

The LSTM layer, which serves to remember and process input sequences, is the fundamental layer in the LSTM model architecture. The results of hyperparameter optimization experiments on the LSTM layer are shown in the following table.

Table 4. Results of the hyperparameter tuning experiments on the LSTM layer

Test	Hyperparameter Tuning				Model Accuracy
	Number of Layers	Layer 1 Units	Layer 2 units	Dropout	
1	2	64	32	0.2	0.9932
2	2	128	64	0.2	0.9980
3	2	256	128	0.2	0.9962
4	2	128	64	0.5	0.9962

Table 4 shows four experiments conducted to obtain the LSTM layer hyperparameters with the best model accuracy. The values for the hyperparameter in the LSTM layer were obtained through two stages: determining the LSTM memory unit and the Dropout value. The experiment results show that the second experiment obtained the best model accuracy among the other experiments, with a model accuracy of 0.9980.

2. Results of the Hyperparameter Tuning Experiments on the Dense Layer

The Dense layer in the LSTM model is used as an output layer to classify whether a text is fake news. The results of hyperparameter optimization experiments on the dense layer are shown in the following table.

Table 5. Results of the hyperparameter tuning experiment on the dense layer

Test	Hyperparameter Tuning				Model Accuracy
	Number of Layers	Number of Neurons 1	Number of Neurons 2	Dropout	
1	1	16	-	0.2	0.9969
2	1	32	-	0.2	0.9922
3	1	64	-	0.2	0.9953
4	1	128	-	0.2	0.9979
5	2	32	16	0.2	0.9936
6	2	64	32	0.2	0.9918
7	2	128	64	0.2	0.9981
8	2	128	64	0.5	0.9970

Table 5 shows eight experiments with three different selections of hyperparameters for the dense layer, including the number of layers, the number of neuron units, and the dropout value. The experiment results show that the 7th experiment obtained the highest dense layer hyperparameter, with a model accuracy of 0.9981.

3. Results of the Hyperparameter Tuning Experiment on the Optimizer

In this process, a model was arranged to be ready for training. The number of weighted inputs and biases in neural networks was determined using the activation function [27]. The results of the hyperparameter optimization experiment on the optimizer output layer are shown in the following table.

Table 6. Results of the hyperparameter tuning experiment on the optimizer

Test	Hyperparameter Tuning	
	Learning Rate	Model Accuracy
1	0.001	0.9798
2	0.01	0.9965
3	0.1	0.5234

Table 6 shows that the hyperparameters on the Optimizer are the existing parameters for the model training process, meaning that the Optimizer is configured before the modeling process. In the hyperparameter optimization on the optimizer, the function of optimizer and activation of loss are the same, namely Optimizer Adam and loss Binary Cross Entropy. At the same time, the Learning Rate is only configured to be three tries. As a result, the second experiment with a 0.01 learning rate obtained a model accuracy of 0.9965.

The Proposed Model

The LSTM neural network model is proposed in this study. Tables 4, 5, and 6 show that the architecture of this model has been optimized using a hyperparameter selection technique or hyperparameter tuning for each layer in the LSTM model. This proposed model was trained and tested to show the performance of the model optimized using this technique.

Table 7. The proposed model of LSTM model architecture

Layer (type)	Output size	Number of Paramet
Embedding_25 (embedding)	300 x 100	1.000.000
Lstm_50 (LSTM)	300 x 128	117,248
Lstm_51 (LSTM)	64	49,408
Dense_57 (Dense)	128	8320
Dense_58 (Dense)	64	8256
Dropout_11 (Dropout)	64	0
Dense_59 (Dense)	1	65

The proposed model of LSTM network architecture is shown in Table 7. The proposed model includes details on the model architecture's layer type, output size, and parameter count. The output size in LSTM depends on how many units or output dimensions the LSTM layer generates. At every step, the LSTM layer's individual units generate output. The output size of the LSTM layer depends on how many units are present there. The weights and biases in the LSTM layer are included in the number of parameters in the LSTM model architecture. The input size, output size, and configuration of each LSTM layer affect the number of parameters in that layer.

Table 8. Hyperparameters in the proposed model

<i>Parameter</i>	<i>Value</i>
Embedding layer	1
LSTM layer	2
Dense layer	3
Dropuout	0.2
Loss Function	Binary cross entropy
Activation function	ReLu
Optimizer	Adam
Learning_rate	0.01
Number of epochs	10
Embedding size	100
Batch size	256

Table 8 shows the hyperparameters in the model proposed in this study. Compared with the hyperparameters in [1] as in Table 2, the difference is only in the number of dense layers, with 3 layers, and dropouts. The LSTM model performe better due to the larger number of memory units giving the model more ability to capture complex patterns in the input data. However, using too many memory units may result in overfitting or raise model complexity. There is an additional hyperparameter (retaining probability) introduced by the dropout function [28]. Dropout hyperparameters were employed to counteract this overfitting and stop it from occurring in the model. In a single LSTM layer, dropout during training disabled several units at random. Thus, the risk of overfitting was reduced, and the units did not overly reliant on one another. Dropouts improved model generalizability and lowered reliance on particular features in the training data.

The Performance of the Proposed Model

Prior to the modeling process, the dataset in this study was divided into train and test subsets with a ratio of 75:25. The proposed model was trained using a train subset and evaluated using a test subset. Model performance can be measured using model evaluation metrics of accuracy. The accuracy evaluation results for the proposed model are shown in Figure 6. Meanwhile, the Confusion Matrix Model is shown in Figure 7.

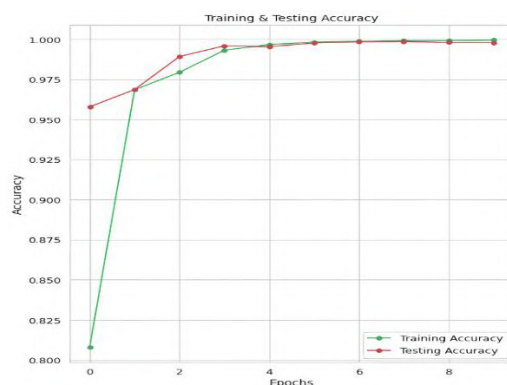


Figure 6. Accuracy of the proposed LSTM model

Figure 6 shows two indicators in the graph, namely Training accuracy (green line) and Testing accuracy (red line). Model performance on training and test datasets was evaluated using two metrics, training and test accuracy.

The term “training accuracy” describes how well the model can predict the training dataset. Calculated is the ratio of the number of accurate predictions to the total sample size of the training dataset. The model’s training accuracy describes how well it recognizes patterns in the training data and how well it can feature the data. Calculated accuracy measures the model’s capacity for accurate prediction on a test dataset unused for training. The test dataset’s sample size is divided by the percentage of correct predictions, which is calculated.

The accuracy test shows how well the model is able to generalize to new data that has never been seen before. The test accuracy is used to objectively measure model performance on independent data. If the test accuracy is not much different from the training accuracy, the model can generalize well and not overfit the training data.

In the model accuracy graph above, the training accuracy value in 10 epochs is 99.9703049659729%, while the test accuracy value is 99.7861921787262%. The accuracy value of this model shows that the test and training accuracy values are not significantly different, allowing for proper generalization and preventing overfitting of the training data. This is also influenced by the dropout layer used in the model architecture so the model accuracy is not too high or overfitting.

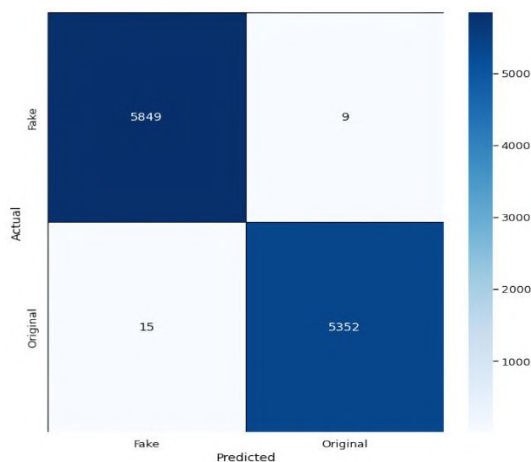


Figure 7. Confusion matrix on the proposed model

Figure 7 shows the confusion matrix table in the proposed model taken from the training data subset. The Confusion Matrix shows that the data tested/evaluated was 11225 data because the dataset division for the test data subset in this study was 25% of the total dataset. In this test data, it appears that the model can recognize fake news data and real news well. From 11225 test data, the model can recognize 5849 fake news data and 5352 real news data. The real news detected as fake news are 15 data, while the fake news detected as the real news are 9 data. Accuracy is determined by calculating the percentage of correct positive and negative predictions compared to the total data [29]. The result is an accuracy rate of 99.78619 percent.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \times 100\%$$

$$\text{Accuracy} = \frac{5352 + 5849}{5352 + 5849 + 9 + 15} \times 100\%$$

$$\text{Accuracy} = 99.78619\% \text{ of test accuracy.}$$

Comparison of the Model Evaluation Results

After completing the model evaluation and obtaining information about the accuracy metrics of the proposed model, the next step is to compare the performance of the fake news detection model between the model proposed in this study and in [2]. This study compares the effectiveness of fake news detection models using model training accuracy metrics describing how well the model learns patterns in the training data and measures the extent to which the model can predict accurately on the training dataset, rather than giving a general overview of how well the model is able to generalize to new data that has never been seen before. This is done because this study uses the same dataset as in [2], To improve the performance of the fake news detection model compared with [2], certain factors must be taken into account when developing the proposed model[2]. Table 9 and Figure 8 show the conclusions of this comparison.

Table 9. Comparison of detection model accuracy

Author	Method	Model Accuracy	Year
Lindawati, Muhammad Fadli Ramadhan, Sopian Soim, Nabila Rizqia Novianda	The LSTM neural network model with Hyperparameter Tuning	99.97%	2023
Tavishee Chauhan, ME, Hemant Palivela, PhD	The LSTM neural network model	99.88%	2021

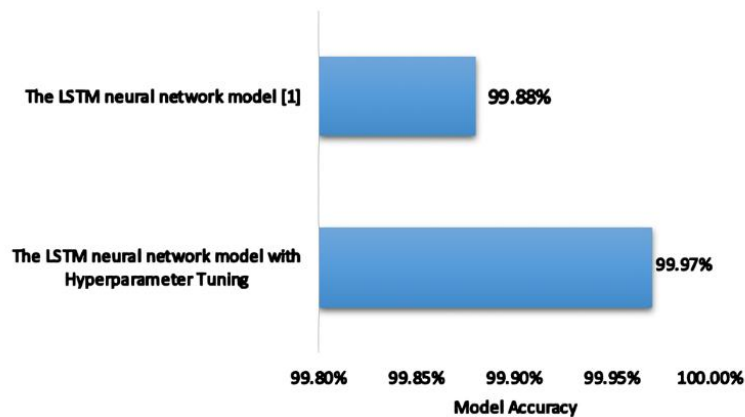


Figure 8. Comparative analysis of the proposed model with previous models

Table 9 shows a comparison of model accuracy performance of the Hyperparameter tuning method in this study with the models in [2]. To improve the performance of the fake news detection model, this study uses the hyperparameter tuning technique to determine the most effective hyperparameters, whereas [2] did not use optimization techniques in its model. This comparison allows us to assess how well each detection model performed when it came to identifying fake news in the dataset. A comparison of the proposed model's accuracy performance with models from earlier studies is shown in Figure 8. The comparison chart help better understand the discrepancies between the proposed model's accuracy and that of earlier research methods. In particular, the neural network-based model uses the hyperparameter tuning method for each layer of the LSTM model to determine how accurate the model is in differentiating between fake and real news data based on the training dataset. In addition, the comparison helps better understand how to increase the accuracy performance of the model.

It can be concluded that the hyperparameter tuning technique on the LSTM model to detect fake news training datasets was more accurate than the previous research models with an accuracy value of 99.97% and outperformed [2] with an accuracy of 99.88%. These results indicate that the LSTM model optimized using the Hyperparameter tuning technique on the model is able to improve the performance of the detection model. It was more accurate in detecting fake news datasets compared to the performance of models in previous studies even though the difference is only slightly, namely 0.09%, considering that the accuracy of the model in the previous study was very high and close to 100% accuracy. However, the results of this study contribute to enhancing the fake news detection model's performance, which is more accurate than earlier models. It is expected that these findings are able to assist other authors in enhancing the LSTM model's performance for identifying fake news with the methodology used in this study and be applied as fake news detection systems to assist people in dealing with the spread of fake news.

CONCLUSION

This study uses the Hyperparameter Tuning method to optimize the LSTM (Long Short-Term Memory) model to detect fake news more accurately than previous model. Despite using the same deep learning algorithm as earlier model, choosing an accurate hyperparameter can have an impact on how well the fake news detection model performs. The number of layers, memory units, and cells in each layer, the learning rate value on the pre-trained parameters, the pre-trained word embedding that serves as the input layer embedding, i.e., Glove, the number of epochs in the model training process, the type of activation function utilized, and the Dropout function to overcome overfitting are just a few of the variables that have an impact on the accuracy of this model. The results of this research can be a reference and consideration for researchers when optimizing the performance of the LSTM model in NLP (Natural Language Processing) cases using the Hyperparameter Tuning method. In the future, it is expected that there will be other neural network methods that can be used to improve the model's accuracy in detecting fake news and a project to implement a fake news detection model to assist the public in dealing with the spread of fake news.

REFERENCES

- [1] J. I.-Z. Chen, H. Wang, and K.-L. Du, *Machine Learning and Autonomous Systems: Proceedings of ICM LAS 2021*. Springer Science and Business Media LLC, 2022.

- [2] T. Chauhan and H. Palivela, "Optimization and improvement of fake news detection using deep learning approaches for societal benefit," *Int. J. Inf. Manag. Data Insights*, vol. 1, no. 2, p. 100051, 2021, doi: 10.1016/j.jjime.2021.100051.
- [3] V. R. Palanivelu and B. Vasanthi, "Role of artificial intelligence in business transformation," *Int. J. Adv. Sci. Technol.*, vol. 29, no. 4 Special Issue, pp. 392–400, 2020.
- [4] J. L. Ruiz-Real, J. Uribe-Toril, J. A. Torres, and J. D. E. Pablo, "Artificial intelligence in business and economics research: Trends and future," *J. Bus. Econ. Manag.*, vol. 22, no. 1, pp. 98–117, 2021, doi: 10.3846/jbem.2020.13641.
- [5] Y. Kang, Z. Cai, C. W. Tan, Q. Huang, and H. Liu, "Natural language processing (NLP) in management research: A literature review," *J. Manag. Anal.*, vol. 7, no. 2, pp. 139–172, 2020, doi: 10.1080/23270012.2020.1756939.
- [6] M. Bahja, "Natural Language Processing Applications in Business," *E-bus. - High. Educ. Intell. Appl.*, 2021, doi: 10.5772/intechopen.92203.
- [7] S. Vosoughi, D. Roy, and S. Aral, "The spread of true and false news online," *Science (80-.)*, vol. 359, no. 6380, pp. 1146–1151, 2018, doi: 10.1126/science.aap9559.
- [8] R. K. Kaliyar, A. Goswami, and P. Narang, "FakeBERT: Fake news detection in social media with a BERT-based deep learning approach," *Multimed. Tools Appl.*, vol. 80, no. 8, pp. 11765–11788, 2021, doi: 10.1007/s11042-020-10183-2.
- [9] R. K. Kaliyar, A. Goswami, and P. Narang, "DeepFakeE: improving fake news detection using tensor decomposition-based deep neural network," *J. Supercomput.*, vol. 77, no. 2, pp. 1015–1037, 2021, doi: 10.1007/s11227-020-03294-y.
- [10] R. K. Kaliyar, A. Goswami, and P. Narang, "EchoFakeD: improving fake news detection in social media with an efficient deep neural network," *Neural Comput. Appl.*, vol. 33, no. 14, pp. 8597–8613, 2021, doi: 10.1007/s00521-020-05611-1.
- [11] F. A. Ozbay and B. Alatas, "Adaptive Salp swarm optimization algorithms with inertia weights for novel fake news detection model in online social media," *Multimed. Tools Appl.*, vol. 80, no. 26–27, pp. 34333–34357, 2021, doi: 10.1007/s11042-021-11006-8.
- [12] C. Raj and P. Meel, "ConvNet frameworks for multi-modal fake news detection," *Appl. Intell.*, vol. 51, no. 11, pp. 8132–8148, 2021, doi: 10.1007/s10489-021-02345-y.
- [13] W. F. Abror and M. Aziz, "Journal of Information System Bankruptcy Prediction Using Genetic Algorithm-Support Vector Machine (GA-SVM) Feature Selection and Stacking," vol. 1, no. 2, pp. 103–108, 2023.
- [14] L. Alfaris, R. C. Siagian, A. C. Muhammad, and U. I. Nyuswantoro, "Classification of Spiral and Non-Spiral Galaxies using Decision Tree Analysis and Random Forest Model : An Investigation of the Zoo Galaxy Dataset," vol. 10, no. 2, pp. 139–150, 2023, doi: 10.15294/sji.v10i2.44027.
- [15] H. Mostafa and X. Wang, "Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization," *36th Int. Conf. Mach. Learn. ICML 2019*, vol. 2019-June, pp. 8163–8172, 2019.
- [16] M. Loni, S. Sinaei, A. Zoljodi, M. Daneshtalab, and M. Sjödin, "DeepMaker: A multi-objective optimization framework for deep neural networks in embedded systems," *Microprocess. Microsyst.*, vol. 73, Mar. 2020, doi: 10.1016/j.micpro.2020.102989.
- [17] N. Guimarães, Á. Figueira, and L. Torgo, "Can fake news detection models maintain the performance through time? A longitudinal evaluation of twitter publications," *Mathematics*, vol. 9, no. 22, 2021, doi: 10.3390/math9222988.
- [18] M. Madani, H. Motameni, and R. Roshani, "Fake News Detection Using Feature Extraction, Natural Language Processing, Curriculum Learning, and Deep Learning," *Int. J. Inf. Technol. Decis. Mak.*, pp. 1–36, Apr. 2023, doi: 10.1142/S0219622023500347.
- [19] Sulistiana and M. A. Muslim, "Support Vector Machine (SVM) Optimization Using Grid Search and Unigram to Improve E-Commerce Review Accuracy," *J. Soft Comput. Explor.*, vol. 1, no. 1, pp. 8–15, 2020.
- [20] H. Ahmed, I. Traore, and S. Saad, "Detecting opinion spams and fake news using text classification," *Secur. Priv.*, vol. 1, no. 1, p. e9, 2018, doi: 10.1002/spy2.9.
- [21] J. Pennington, R. Socher, and C. Manning, "Glove: Global Vectors for Word Representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, vol. 19, no. 5, pp. 1532–1543. doi: 10.3115/v1/D14-1162.
- [22] Y. Nezu and M. Takao, *Extracting stopwords on social network service*. Information Modelling and Knowledge Bases XXXI 321, 2020.
- [23] S. Choo and W. Kim, "A study on the evaluation of tokenizer performance in natural language

- processing,” *Appl. Artif. Intell.*, vol. 37, no. 1, 2023, doi: 10.1080/08839514.2023.2175112.
- [24] J. Ravi and S. Kulkarni, “Text embedding techniques for efficient clustering of twitter data,” *Evol. Intell.*, no. 0123456789, 2023, doi: 10.1007/s12065-023-00825-3.
- [25] L. Yang and A. Shami, “On hyperparameter optimization of machine learning algorithms: Theory and practice,” *Neurocomputing*, vol. 415, pp. 295–316, 2020, doi: 10.1016/j.neucom.2020.07.061.
- [26] M. A. Mahjoubi, S. Hamida, O. El Gannour, B. Cherradi, A. El Abbassi, and A. Raihani, “Improved Multiclass Brain Tumor Detection using Convolutional Neural Networks and Magnetic Resonance Imaging,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 14, no. 3, pp. 406–414, 2023, doi: 10.14569/IJACSA.2023.0140346.
- [27] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, “Activation Functions: Comparison of trends in Practice and Research for Deep Learning,” *Math. Comput. Sci.*, pp. 1–20, 2018.
- [28] B. Ait Skourt, A. El Hassani, and A. Majda, “Mixed-pooling-dropout for convolutional neural network regularization,” *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 34, no. 8, pp. 4756–4762, 2022, doi: 10.1016/j.jksuci.2021.05.001.
- [29] W. Budiawan Zulfikar, A. Rialdy Atmadja, and S. F. Pratama, “Sentiment Analysis on Social Media Against Public Policy Using Multinomial Naive Bayes,” *Sci. J. Informatics*, vol. 10, no. 1, pp. 25–34, 2023, doi: 10.15294/sji.v10i1.39952.

```

# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load in

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# Any results you write to the current directory are saved as output.

/kaggle/input/fake-and-real-news-dataset/True.csv
/kaggle/input/fake-and-real-news-dataset/Fake.csv
/kaggle/input/glovetwitter27b100dtx/glove.twitter.27B.100d.txt

```

PENINGKATAN PERFORMA MODEL NEURAL NETWORK MENGGUNAKAN OPTIMASI HYPERPARAMETER LONG SHORT TERM MEMORY UNTUK DETEKSI BERITA PALSU

Dibuat oleh : Muhammad Fadli Ramadhan (061940352343) dari Politeknik Negeri Sriwijaya, Palembang, Sumatera Selatan, Indonesia - Google Certified TensorFlow Developer

TENTANG PENELITIAN INI

Tugas akhir ini akan melakukan penelitian mengenai peningkatan performa model neural network menggunakan optimasi hyperparameter model Long Short Term Memory (LSTM) untuk deteksi berita palsu. Tujuan dari penelitian ini yaitu mendapatkan peningkatan akurasi model neural network apabila optimasi hyperparameter Long Short Term Memory diterapkan pada teknik pemilihan parameter model yang optimal sehingga untuk deteksi berita palsu menjadi lebih akurat dan lebih baik dari penelitian termutakhir sebelumnya dalam deteksi berita palsu menggunakan model jaringan LSTM dan teknik penyematan kata GloVe untuk memperbaiki model neural network dan meningkatkan akurasi model sebesar 99.88%[1]. Metode yang diusulkan adalah metode optimasi hyperparameter model jaringan saraf LSTM pada masing-masing lapisan jaringan saraf LSTM melalui berbagai percobaan untuk memodifikasi hyperparameter lapisan jaringan saraf model LSTM agar optimal dan meningkatkan performa model. Metrik evaluasi model menggunakan metrik akurasi model dan hasil percobaan ini akan mendapatkan model pembelajaran mesin baru berbasis neural network dan LSTM dengan akurasi model yang lebih tinggi dari model penelitian termutakhir sebelumnya.

Kata Kunci: neural network, long short term memory, optimasi hyperparameter, deteksi berita palsu

[1] T. Chauhan and H. Palivela, "Optimization and improvement of fake news detection using deep learning approaches for societal benefit," Int. J. Inf. Manag. Data Insights, vol. 1, no. 2, p. 100051, 2021, doi: 10.1016/j.jjime.2021.100051.

PERFORMANCE IMPROVEMENT OF NEURAL NETWORK MODELS USING LONG SHORT-TERM MEMORY HYPERPARAMETER OPTIMIZATION FOR FAKE NEWS DETECTION

Created by: Muhammad Fadli Ramadhan (061940352343) from Sriwijaya State Polytechnic, Palembang, South Sumatra, Indonesia - Google Certified TensorFlow Developer

ABOUT THIS RESEARCH

This final project will conduct research on improving the performance of neural network models using hyperparameter optimization of the Long Short Term Memory (LSTM) model for fake news detection. The purpose of this study is to obtain an increase in the accuracy of the neural network model when optimization of the Long Short Term Memory hyperparameter is applied to the optimal model parameter selection technique so that the detection of fake news becomes more accurate and better than previous recent research in detecting fake news using the LSTM network model and the GloVe word embedding technique to improve the neural network model and increase the model accuracy by 99.88% [1]. The proposed method is a method of optimizing the hyperparameters of the LSTM neural network model in each layer of the LSTM neural network through various experiments to modify the hyperparameters of the LSTM neural network layer so that it is optimal and improves model performance. Model evaluation metrics use model accuracy metrics and the results of this experiment will get a new machine learning model based on neural networks and LSTM with higher model accuracy than the previous latest research models.

Keywords: neural network, long short-term memory, hyperparameter optimization, fake news detection

[1] T. Chauhan and H. Palivela, "Optimization and improvement of fake news detection using deep learning approaches for societal benefit," Int. J. Inf. Manag. Data Insights, vol. 1, no. 2, p. 100051, 2021, doi: 10.1016/j.jjime.2021.100051.

HYPERPARAMETER TUNING OF LSTM MODEL

Hyperparameter model LSTM yang akan dioptimasi:

1. Hyperparameter Layer LSTM

- Jumlah Unit Memori pada Layer LSTM
- Nilai Regularisasi Dropout pada Layer LSTM

2. Hyperparameter Layer Dense

- Jumlah Layer Dense
- Jumlah Unit Neuron pada Layer Dense
- Nilai Regularisasi Dropout pada Layer Dense

3. Hyperparameter Optimizer

- Nilai Learning Rate pada Optimizer Adam

HYPERPARAMETER TUNING OF LSTM MODEL

LSTM model hyperparameters to be optimized:

1. LSTM Layer Hyperparameters

- Number of Memory Units in the LSTM Layer
- Dropout Regularization Value on the LSTM Layer

2. Layer Dense Hyperparameters

- Number of Dense Layers
- Number of Neuron Units in the Dense Layer
- Dropout Regularization Value on Layer Dense

3. Hyperparameter Optimizer

- Learning Rate value on Optimizer Adam

IMPORT LIBRARIES

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import nltk
from sklearn.preprocessing import LabelBinarizer
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from wordcloud import WordCloud, STOPWORDS
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize, sent_tokenize
from bs4 import BeautifulSoup
import re, string, unicodedata
from keras.preprocessing import text, sequence
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.model_selection import train_test_split
from string import punctuation
from nltk import pos_tag
from nltk.corpus import wordnet
import keras
from keras.models import Sequential
from keras.layers import Dense, Embedding, LSTM, Dropout
from keras.callbacks import ReduceLROnPlateau
import tensorflow as tf
```

IMPORT DATASET

```
true = pd.read_csv("../input/fake-and-real-news-dataset/True.csv")
false = pd.read_csv("../input/fake-and-real-news-dataset/Fake.csv")
```

DATA VISUALIZATION AND PREPROCESSING

```
true
```

	title	text	subject	date
0	As U.S. budget fight looms, Republicans flip t...	WASHINGTON (Reuters) - The head of a conservat...	politicsNews	December 31, 2017
1	U.S. military to accept transgender recruits o...	WASHINGTON (Reuters) - Transgender people will...	politicsNews	December 29, 2017
2	Senior U.S. Republican senator: 'Let Mr. Muell...	WASHINGTON (Reuters) - The special counsel inv...	politicsNews	December 31, 2017
3	FBI Russia probe helped by Australian diplomat...	WASHINGTON (Reuters) - Trump campaign adviser ...	politicsNews	December 30, 2017
4	Trump wants Postal Service to charge 'much mor...	SEATTLE/WASHINGTON (Reuters) - President Donal...	politicsNews	December 29, 2017

```
true.count()
```

```
title      21417
text       21417
subject    21417
date       21417
dtype: int64
```

```
false
```

	title	text	subject	date
0	Donald Trump Sends Out Embarrassing New Year'...	Donald Trump just couldn t wish all Americans ...	News	December 31, 2017
1	Drunk Bragging Trump Staffer Started Russian ...	House Intelligence Committee Chairman Devin Nu...	News	December 31, 2017
2	Sheriff David Clarke Becomes An Internet Joke...	On Friday, it was revealed that former Milwauk...	News	December 30, 2017
3	Trump Is So Obsessed He Even Has Obama's Name...	On Christmas day, Donald Trump announced that ...	News	December 29, 2017
4	Pope Francis Just Called Out Donald Trump Dur...	Pope Francis used his annual Christmas Day mes...	News	December 25, 2017
...
23476	McPain: John McCain Furious That Iran Treated ...	21st Century Wire says As 21WIRE reported earl...	Middle-east	January 16, 2016
23477	JUSTICE? Yahoo Settles E-mail Privacy Class-ac...	21st Century Wire says It s a familiar theme. ...	Middle-east	January 16, 2016
23478	Sunnistan: US and Allied 'Safe Zone' Plan to T...	Patrick Henningsen 21st Century WireRemember ...	Middle-east	January 15, 2016
23479	How to Blow \$700 Million: Al Jazeera America F...	21st Century Wire says Al Jazeera America will...	Middle-east	January 14, 2016
23480	10 U.S. Navy Sailors Held by Iranian Military ...	21st Century Wire says As 21WIRE predicted in ...	Middle-east	January 12, 2016

```
23481 rows x 4 columns
```

```
false.count()
```

```
title      23481
text       23481
subject    23481
date       23481
dtype: int64
```

```
true['category'] = 1
false['category'] = 0
```

```
df = pd.concat([true,false]) #Merging the 2 datasets
```

```
sns.set_style("darkgrid")
sns.countplot(df.category)
```

```
/opt/conda/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. F
FutureWarning
<AxesSubplot:xlabel='category', ylabel='count'>
```

BALANCED DATASET

```
df
```

	title	text	subject	date	category
0	As U.S. budget fight looms, Republicans flip t...	WASHINGTON (Reuters) - The head of a conservat...	politicsNews	December 31, 2017	1
1	U.S. military to accept transgender recruits o...	WASHINGTON (Reuters) - Transgender people will...	politicsNews	December 29, 2017	1
2	Senior U.S. Republican senator: 'Let Mr. Muell...	WASHINGTON (Reuters) - The special counsel inv...	politicsNews	December 31, 2017	1
3	FBI Russia probe helped by Australian diplomat...	WASHINGTON (Reuters) - Trump campaign adviser ...	politicsNews	December 30, 2017	1
4	Trump wants Postal Service to charge 'much mor...	SEATTLE/WASHINGTON (Reuters) - President Donal...	politicsNews	December 29, 2017	1
...
23476	McPain: John McCain Furious That Iran Treated ...	21st Century Wire says As 21WIRE reported earl...	Middle-east	January 16, 2016	0
23477	JUSTICE? Yahoo Settles E-mail Privacy Class-ac...	21st Century Wire says It s a familiar theme. ...	Middle-east	January 16, 2016	0

```
df.isna().sum() # Memeriksa Nilai nan
```

```
title      0
text       0
subject    0
date       0
category   0
dtype: int64
```

```
df.title.count()
```

```
44898
```

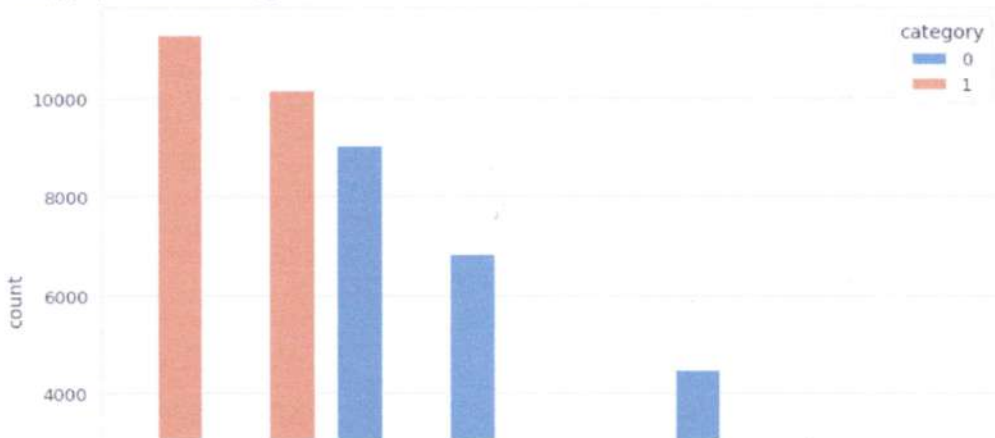
```
df.subject.value_counts()
```

```
politicsNews    11272
worldnews       10145
News             9050
politics        6841
left-news       4459
Government News  1570
US_News         783
Middle-east     778
Name: subject, dtype: int64
```

COMBINING ALL TEXT DATA INTO 1 COLUMN namely 'text'

```
plt.figure(figsize = (12,8))
sns.set(style = "whitegrid",font_scale = 1.2)
chart = sns.countplot(x = "subject", hue = "category" , data = df)
chart.set_xticklabels(chart.get_xticklabels(),rotation=90)
```

```
[Text(0, 0, 'politicsNews'),
Text(1, 0, 'worldnews'),
Text(2, 0, 'News'),
Text(3, 0, 'politics'),
Text(4, 0, 'Government News'),
Text(5, 0, 'left-news'),
Text(6, 0, 'US_News'),
Text(7, 0, 'Middle-east')]
```



BECAUSE THE TOPICS IN THE SUBJECT COLUMN ARE DIFFERENT FOR THE BOTH CATEGORIES, WE SHOULD EXCLUDE THEM FROM THE FINAL TEXT COLUMN

```
df['text'] = df['text'] + " " + df['title']
del df['title']
del df['subject']
del df['date']
```

REMOVING THE STOPWORDS

```
stop = set(stopwords.words('english'))
punctuation = list(string.punctuation)
stop.update(punctuation)
```

DATA CLEANING

```
def strip_html(text):
    soup = BeautifulSoup(text, "html.parser")
    return soup.get_text()
```

```
#Removing the square brackets
def remove_between_square_brackets(text):
    return re.sub('\[[^\]]*\]', '', text)
```

```
# Removing URL's
def remove_between_square_brackets(text):
    return re.sub(r'http\S+', '', text)
```

```
#Removing the stopwords from text
def remove_stopwords(text):
    final_text = []
    for i in text.split():
        if i.strip().lower() not in stop:
            final_text.append(i.strip())
    return " ".join(final_text)
```

```
#Removing the noisy text
def denoise_text(text):
    text = strip_html(text)
    text = remove_between_square_brackets(text)
    text = remove_stopwords(text)
    return text
```

```
#Apply function on review column
df['text']=df['text'].apply(denoise_text)
```

```
/opt/conda/lib/python3.7/site-packages/bs4/__init__.py:439: MarkupResemblesLocatorWarning: The input looks more like a filename than a document.
MarkupResemblesLocatorWarning
```

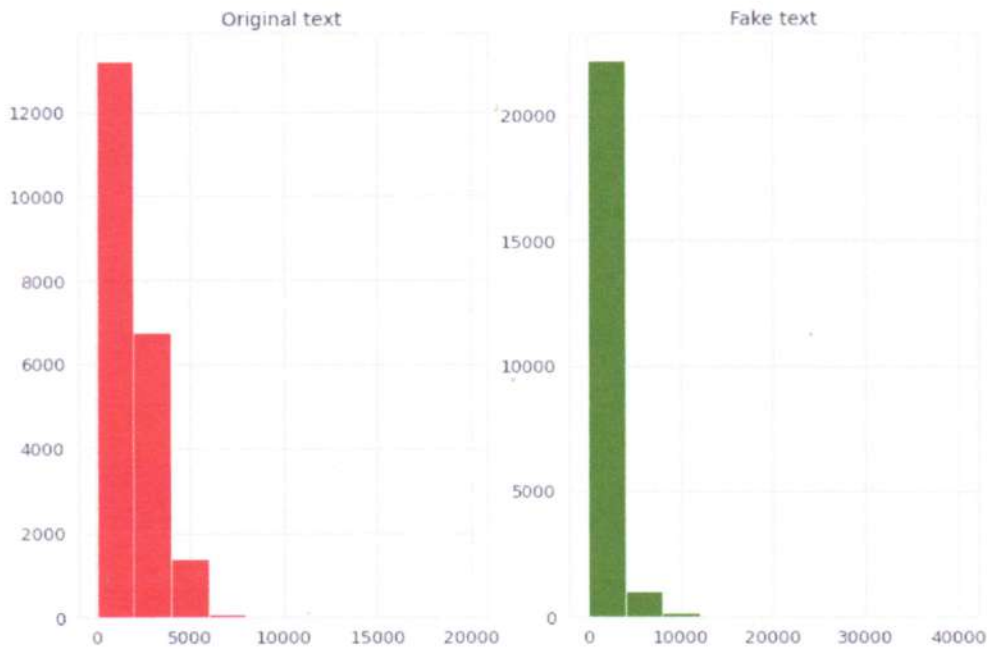
WORDCLOUD FOR REAL TEXT (LABEL - 1)


```

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 8))
text_len = df[df['category'] == 1]['text'].str.len()
ax1.hist(text_len, color='red')
ax1.set_title('Original text')
text_len = df[df['category'] == 0]['text'].str.len()
ax2.hist(text_len, color='green')
ax2.set_title('Fake text')
fig.suptitle('Characters in texts')
plt.show()

```

Characters in texts



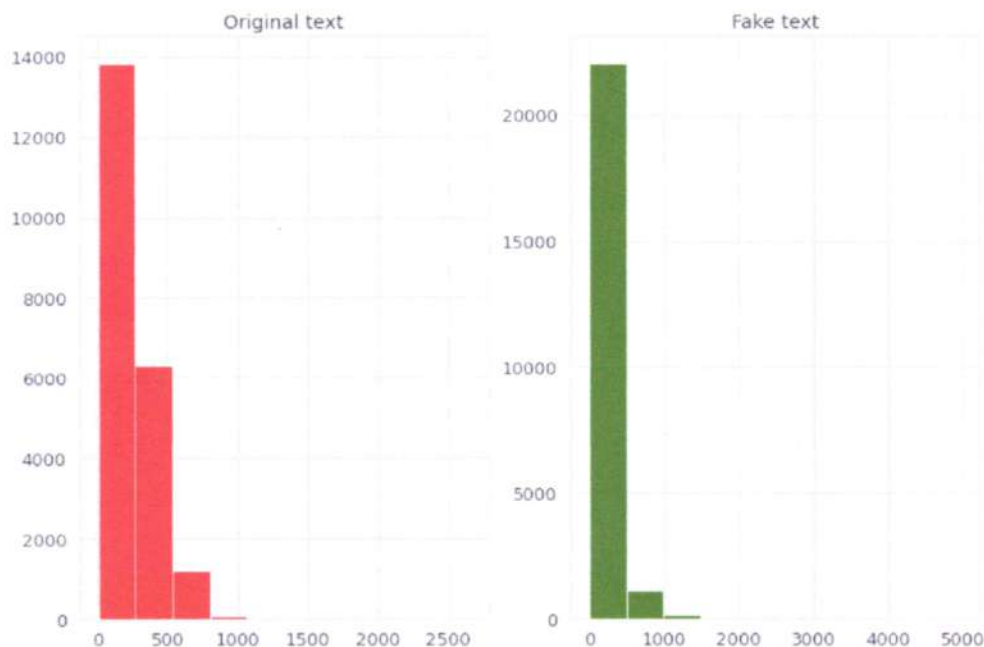
Number of words in each text

```

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 8))
text_len = df[df['category'] == 1]['text'].str.split().map(lambda x: len(x))
ax1.hist(text_len, color='red')
ax1.set_title('Original text')
text_len = df[df['category'] == 0]['text'].str.split().map(lambda x: len(x))
ax2.hist(text_len, color='green')
ax2.set_title('Fake text')
fig.suptitle('Words in texts')
plt.show()

```

Words in texts

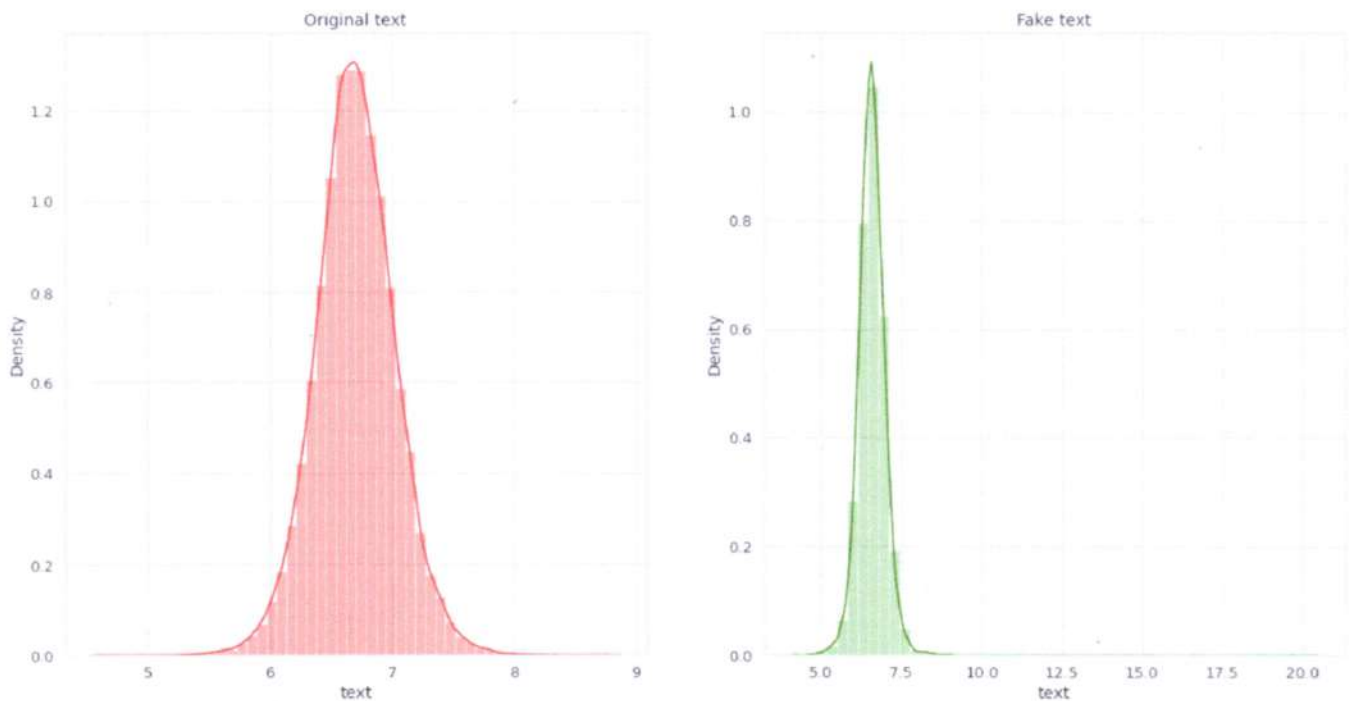


Average length of words in a text

```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(20, 10))
word = df[df['category'] == 1]['text'].str.split().apply(lambda x: [len(i) for i in x])
sns.distplot(word.map(lambda x: np.mean(x)), ax=ax1, color='red')
ax1.set_title('Original text')
word = df[df['category'] == 0]['text'].str.split().apply(lambda x: [len(i) for i in x])
sns.distplot(word.map(lambda x: np.mean(x)), ax=ax2, color='green')
ax2.set_title('Fake text')
fig.suptitle('Average word length in each text')
```

```
/opt/conda/lib/python3.7/site-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Use `sns.displot` instead.
warnings.warn(msg, FutureWarning)
/opt/conda/lib/python3.7/site-packages/numpy/core/fromnumeric.py:3441: RuntimeWarning: Mean of empty slice.
out=out, **kwargs)
/opt/conda/lib/python3.7/site-packages/numpy/core/_methods.py:189: RuntimeWarning: invalid value encountered in double_scalars
ret = ret.dtype.type(ret / rcount)
/opt/conda/lib/python3.7/site-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Use `sns.displot` instead.
warnings.warn(msg, FutureWarning)
Text(0.5, 0.98, 'Average word length in each text')
```

Average word length in each text



```
def get_corpus(text):
    words = []
    for i in text:
        for j in i.split():
            words.append(j.strip())
    return words
corpus = get_corpus(df.text)
corpus[:5]

['WASHINGTON', '(Reuters)', 'head', 'conservative', 'Republican']
```

```
from collections import Counter
counter = Counter(corpus)
most_common = counter.most_common(10)
most_common = dict(most_common)
most_common
```

```
{'Trump': 111503,
'said': 93162,
'would': 54613,
'U.S.': 50441,
'President': 33180,
'people': 33115,
'also': 30325,
```

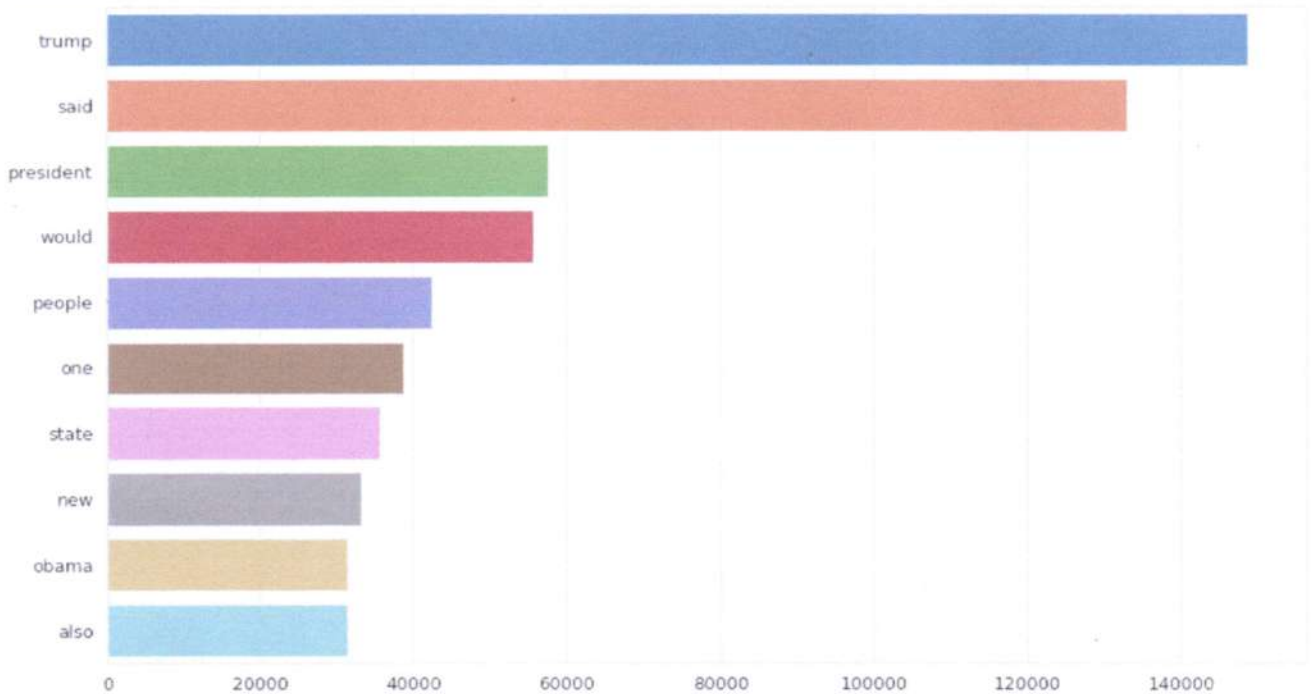
```
'one': 29370,
'Donald': 27795,
'said.': 26194}
```

```
from sklearn.feature_extraction.text import CountVectorizer
def get_top_text_ngrams(corpus, n, g):
    vec = CountVectorizer(ngram_range=(g, g)).fit(corpus)
    bag_of_words = vec.transform(corpus)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.items()]
    words_freq = sorted(words_freq, key = lambda x: x[1], reverse=True)
    return words_freq[:n]
```

Unigram Analysis

```
plt.figure(figsize = (16,9))
most_common_uni = get_top_text_ngrams(df.text,10,1)
most_common_uni = dict(most_common_uni)
sns.barplot(x=list(most_common_uni.values()),y=list(most_common_uni.keys()))
```

<AxesSubplot:>



Bigram Analysis

```
plt.figure(figsize = (16,9))
most_common_bi = get_top_text_ngrams(df.text,10,2)
most_common_bi = dict(most_common_bi)
sns.barplot(x=list(most_common_bi.values()),y=list(most_common_bi.keys()))
```

<AxesSubplot:>

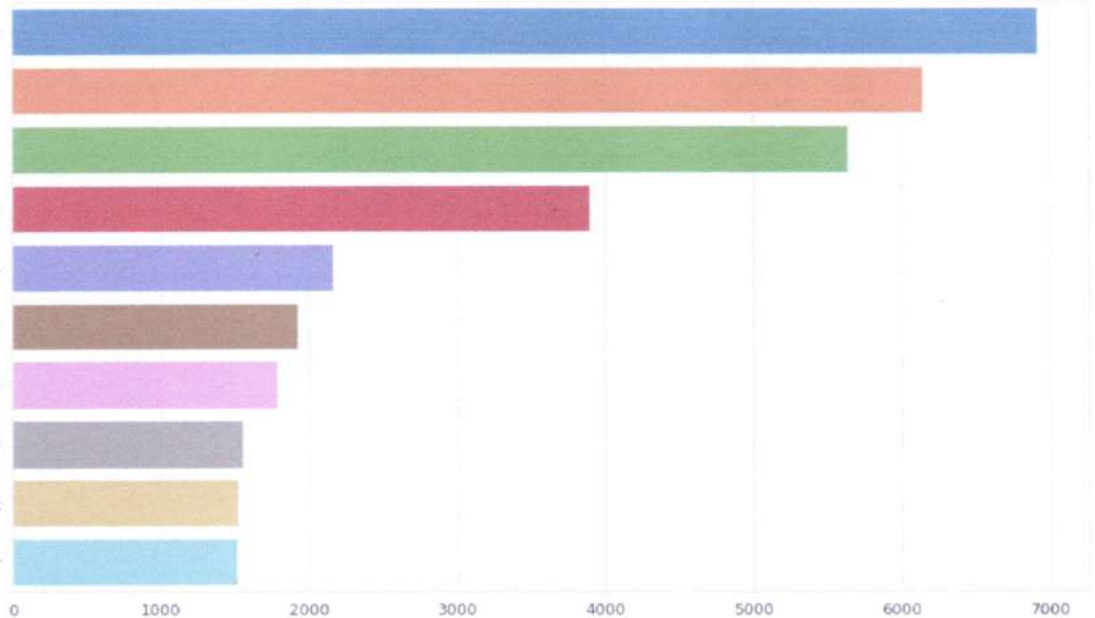
donald trump
united states

**Trigram Analysis**

```
plt.figure(figsize = (16,9))
most_common_tri = get_top_text_ngrams(df.text,10,3)
most_common_tri = dict(most_common_tri)
sns.barplot(x=list(most_common_tri.values()),y=list(most_common_tri.keys()))
```

<AxesSubplot:>

president donald trump
pic twitter com
featured image via
president barack obama
new york times
21st century wire
donald trump realdonaldtrump
reuters president donald
washington reuters president
black lives matter

**Dividing data into 2 parts - data train and test**

```
x_train,x_test,y_train,y_test = train_test_split(df.text,df.category,random_state = 0)
```

```
max_features = 10000
maxlen = 300
```

Tokenization Text -> Represent each word with a number

The original word-to-number mapping is preserved in the word_index property of the tokenizer

Tokenizer implements basic processing such as converting it to lowercase, explicitly setting it as False

Keep all stories to 300, add padding to stories less than 300 words, and trim long ones

```
tokenizer = text.Tokenizer(num_words=max_features)
tokenizer.fit_on_texts(x_train)
tokenized_train = tokenizer.texts_to_sequences(x_train)
x_train = sequence.pad_sequences(tokenized_train, maxlen=maxlen)
```

```
tokenized_test = tokenizer.texts_to_sequences(x_test)
X_test = sequence.pad_sequences(tokenized_test, maxlen=maxlen)
```

```
EMBEDDING_FILE = '/kaggle/input/glovetwitter27b100dtx/glove.twitter.27B.100d.txt'
```

```
def get_coefs(word, *arr):
    return word, np.asarray(arr, dtype='float32')
embeddings_index = dict(get_coefs(*o.rstrip().rsplit(' ')) for o in open(EMBEDDING_FILE))
```

```
[]
```

```

all_embs = np.stack(embeddings_index.values())
emb_mean,emb_std = all_embs.mean(), all_embs.std()
embed_size = all_embs.shape[1]

word_index = tokenizer.word_index
nb_words = min(max_features, len(word_index))
#change below line if computing normal stats is too slow
embedding_matrix = embedding_matrix = np.random.normal(emb_mean, emb_std, (nb_words, embed_size))
for word, i in word_index.items():
    if i >= max_features: continue
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None: embedding_matrix[i] = embedding_vector

/opt/conda/lib/python3.7/site-packages/IPython/core/interactiveshell.py:3472: FutureWarning: arrays to stack must be passed as a "s
    if (await self.run_code(code, result,  async_=asy)):

learning_rate_reduction = ReduceLROnPlateau(monitor='val_accuracy', patience = 2, verbose=1,factor=0.5, min_lr=0.00001)

```

TRAINING MODEL

Experiment 1 LSTM Layer

```

# Hyperparameter LSTM Layer 1

batch_size = 1024
epochs = 5

embed_size = 100 # Output_dim
max_features = 10000 # Input_dim

#Defining Neural Network
model = Sequential()
#Non-trainable embedding layer
model.add(Embedding(max_features, output_dim=embed_size, weights=[embedding_matrix], input_length=maxlen, trainable=False))
#LSTM
model.add(LSTM(units=64 , return_sequences = True , recurrent_dropout = 0.2 , dropout = 0.2))
model.add(LSTM(units=32 , recurrent_dropout = 0.2 , dropout = 0.2))
model.add(Dense(units = 32 , activation = 'relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer=tf.keras.optimizers.Adam(lr = 0.01), loss='binary_crossentropy', metrics=['accuracy'])

model.summary()

history = model.fit(x_train, y_train, batch_size = batch_size , validation_data = (X_test,y_test) , epochs = epochs , callbacks = [learni

```

Model: "sequential_10"

Layer (type)	Output Shape	Param #
embedding_10 (Embedding)	(None, 300, 100)	1000000
lstm_20 (LSTM)	(None, 300, 64)	42240
lstm_21 (LSTM)	(None, 32)	12416
dense_20 (Dense)	(None, 32)	1056
dense_21 (Dense)	(None, 1)	33
Total params: 1,055,745		
Trainable params: 55,745		
Non-trainable params: 1,000,000		

```

Epoch 1/5
33/33 [=====] - 105s 3s/step - loss: 0.3166 - accuracy: 0.8646 - val_loss: 0.1294 - val_accuracy: 0.9561
Epoch 2/5
33/33 [=====] - 99s 3s/step - loss: 0.0959 - accuracy: 0.9673 - val_loss: 0.0602 - val_accuracy: 0.9787
Epoch 3/5
33/33 [=====] - 103s 3s/step - loss: 0.0784 - accuracy: 0.9720 - val_loss: 0.1021 - val_accuracy: 0.9702
Epoch 4/5
33/33 [=====] - 104s 3s/step - loss: 0.0614 - accuracy: 0.9804 - val_loss: 0.0275 - val_accuracy: 0.9910
Epoch 5/5
33/33 [=====] - 99s 3s/step - loss: 0.0215 - accuracy: 0.9932 - val_loss: 0.0182 - val_accuracy: 0.9948

```

Hyperparameter LSTM Layer 2

```

batch_size = 1024
epochs = 5

```

```

embed_size = 100 # Output_dim
max_features = 10000 # Input_dim

#Defining Neural Network
model = Sequential()
#Non-trainable embedding layer
model.add(Embedding(max_features, output_dim=embed_size, weights=[embedding_matrix], input_length=maxlen, trainable=False))
#LSTM
model.add(LSTM(units=128 , return_sequences = True , recurrent_dropout = 0.2 , dropout = 0.2))
model.add(LSTM(units=64 , recurrent_dropout = 0.2 , dropout = 0.2))
model.add(Dense(units = 32 , activation = 'relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer=tf.keras.optimizers.Adam(lr = 0.01), loss='binary_crossentropy', metrics=['accuracy'])

model.summary()

history = model.fit(x_train, y_train, batch_size = batch_size , validation_data = (X_test,y_test) , epochs = epochs , callbacks = [learni

```

Model: "sequential_11"

Layer (type)	Output Shape	Param #
embedding_11 (Embedding)	(None, 300, 100)	1000000
lstm_22 (LSTM)	(None, 300, 128)	117248
lstm_23 (LSTM)	(None, 64)	49408
dense_22 (Dense)	(None, 32)	2080
dense_23 (Dense)	(None, 1)	33

Total params: 1,168,769
 Trainable params: 168,769
 Non-trainable params: 1,000,000

```

Epoch 1/5
33/33 [=====] - 105s 3s/step - loss: 0.3097 - accuracy: 0.8601 - val_loss: 0.0954 - val_accuracy: 0.9668
Epoch 2/5
33/33 [=====] - 100s 3s/step - loss: 0.0656 - accuracy: 0.9779 - val_loss: 0.0334 - val_accuracy: 0.9908
Epoch 3/5
33/33 [=====] - 100s 3s/step - loss: 0.0182 - accuracy: 0.9943 - val_loss: 0.0129 - val_accuracy: 0.9967
Epoch 4/5
33/33 [=====] - 100s 3s/step - loss: 0.0141 - accuracy: 0.9956 - val_loss: 0.0089 - val_accuracy: 0.9972
Epoch 5/5
33/33 [=====] - 100s 3s/step - loss: 0.0063 - accuracy: 0.9980 - val_loss: 0.0063 - val_accuracy: 0.9985

```

Hyperparameter LSTM Layer 3

```

batch_size = 1024
epochs = 5
embed_size = 100 # Output_dim
max_features = 10000 # Input_dim

#Defining Neural Network
model = Sequential()
#Non-trainable embedding layer
model.add(Embedding(max_features, output_dim=embed_size, weights=[embedding_matrix], input_length=maxlen, trainable=False))
#LSTM
model.add(LSTM(units=256 , return_sequences = True , recurrent_dropout = 0.2 , dropout = 0.2))
model.add(LSTM(units=128 , recurrent_dropout = 0.2 , dropout = 0.2))
model.add(Dense(units = 32 , activation = 'relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer=tf.keras.optimizers.Adam(lr = 0.01), loss='binary_crossentropy', metrics=['accuracy'])

model.summary()

history = model.fit(x_train, y_train, batch_size = batch_size , validation_data = (X_test,y_test) , epochs = epochs , callbacks = [learni

```

Model: "sequential_12"

Layer (type)	Output Shape	Param #
embedding_12 (Embedding)	(None, 300, 100)	1000000
lstm_24 (LSTM)	(None, 300, 256)	365568
lstm_25 (LSTM)	(None, 128)	197120
dense_24 (Dense)	(None, 32)	4128
dense_25 (Dense)	(None, 1)	33

Total params: 1,566,849

Trainable params: 566,849
 Non-trainable params: 1,000,000

```
Epoch 1/5
33/33 [=====] - 106s 3s/step - loss: 0.3593 - accuracy: 0.8407 - val_loss: 0.1112 - val_accuracy: 0.9596
Epoch 2/5
33/33 [=====] - 101s 3s/step - loss: 0.0838 - accuracy: 0.9703 - val_loss: 0.0472 - val_accuracy: 0.9838
Epoch 3/5
33/33 [=====] - 101s 3s/step - loss: 0.0365 - accuracy: 0.9874 - val_loss: 0.0246 - val_accuracy: 0.9908
Epoch 4/5
33/33 [=====] - 101s 3s/step - loss: 0.0198 - accuracy: 0.9935 - val_loss: 0.0128 - val_accuracy: 0.9960
Epoch 5/5
33/33 [=====] - 101s 3s/step - loss: 0.0120 - accuracy: 0.9962 - val_loss: 0.0093 - val_accuracy: 0.9971
```

Hyperparameter LSTM Layer 4

batch_size = 1024
 epochs = 5
 embed_size = 100 # Output_dim
 max_features = 10000 # Input_dim

#Defining Neural Network

```
model = Sequential()
#Non-trainable embedding layer
model.add(Embedding(max_features, output_dim=embed_size, weights=[embedding_matrix], input_length=maxlen, trainable=False))
#LSTM
model.add(LSTM(units=128, return_sequences = True, recurrent_dropout = 0.5, dropout = 0.5))
model.add(LSTM(units=64, recurrent_dropout = 0.5, dropout = 0.5))
model.add(Dense(units = 32, activation = 'relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer=tf.keras.optimizers.Adam(lr = 0.01), loss='binary_crossentropy', metrics=['accuracy'])
```

model.summary()

history = model.fit(x_train, y_train, batch_size = batch_size, validation_data = (X_test,y_test), epochs = epochs, callbacks = [learni

Model: "sequential_13"

Layer (type)	Output Shape	Param #
embedding_13 (Embedding)	(None, 300, 100)	1000000
lstm_26 (LSTM)	(None, 300, 128)	117248
lstm_27 (LSTM)	(None, 64)	49408
dense_26 (Dense)	(None, 32)	2080
dense_27 (Dense)	(None, 1)	33
Total params: 1,168,769		
Trainable params: 168,769		
Non-trainable params: 1,000,000		

```
Epoch 1/5
33/33 [=====] - 106s 3s/step - loss: 0.4673 - accuracy: 0.7924 - val_loss: 0.1531 - val_accuracy: 0.9433
Epoch 2/5
33/33 [=====] - 101s 3s/step - loss: 0.1114 - accuracy: 0.9583 - val_loss: 0.0636 - val_accuracy: 0.9802
Epoch 3/5
33/33 [=====] - 100s 3s/step - loss: 0.0459 - accuracy: 0.9850 - val_loss: 0.0210 - val_accuracy: 0.9935
Epoch 4/5
33/33 [=====] - 100s 3s/step - loss: 0.0221 - accuracy: 0.9931 - val_loss: 0.0118 - val_accuracy: 0.9957
Epoch 5/5
33/33 [=====] - 101s 3s/step - loss: 0.0311 - accuracy: 0.9902 - val_loss: 0.0331 - val_accuracy: 0.9871
```

Experiment 2 Dense Layer

Hyperparameter Dense Layer 1

embed_size = 100 # Output_dim
 max_features = 10000 # Input_dim

#Defining Neural Network

```
model = Sequential()
#Non-trainable embedding layer
model.add(Embedding(max_features, output_dim=embed_size, weights=[embedding_matrix], input_length=maxlen, trainable=False))
#LSTM
model.add(LSTM(units=128, return_sequences = True, recurrent_dropout = 0.2, dropout = 0.2))
model.add(LSTM(units=64, recurrent_dropout = 0.2, dropout = 0.2))
model.add(Dense(units = 16, activation = 'relu'))
model.add(Dropout(0.2))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer=tf.keras.optimizers.Adam(lr = 0.01), loss='binary_crossentropy', metrics=['accuracy'])
```



```
model.summary()
```

```
history = model.fit(x_train, y_train, batch_size = batch_size , validation_data = (X_test,y_test) , epochs = epochs , callbacks = [learni
```

```
Model: "sequential_14"
```

Layer (type)	Output Shape	Param #
embedding_14 (Embedding)	(None, 300, 100)	1000000
lstm_28 (LSTM)	(None, 300, 128)	117248
lstm_29 (LSTM)	(None, 64)	49408
dense_28 (Dense)	(None, 16)	1040
dropout (Dropout)	(None, 16)	0
dense_29 (Dense)	(None, 1)	17

```

Total params: 1,167,713
Trainable params: 167,713
Non-trainable params: 1,000,000

```

```

Epoch 1/5
33/33 [=====] - 104s 3s/step - loss: 0.3012 - accuracy: 0.8634 - val_loss: 0.0969 - val_accuracy: 0.9686
Epoch 2/5
33/33 [=====] - 100s 3s/step - loss: 0.0671 - accuracy: 0.9792 - val_loss: 0.0429 - val_accuracy: 0.9864
Epoch 3/5
33/33 [=====] - 99s 3s/step - loss: 0.0378 - accuracy: 0.9888 - val_loss: 0.0174 - val_accuracy: 0.9952
Epoch 4/5
33/33 [=====] - 98s 3s/step - loss: 0.0153 - accuracy: 0.9958 - val_loss: 0.0129 - val_accuracy: 0.9963
Epoch 5/5
33/33 [=====] - 98s 3s/step - loss: 0.0137 - accuracy: 0.9969 - val_loss: 0.0078 - val_accuracy: 0.9976

```

```

# Hyperparameter Dense Layer 2
batch_size = 1024
epochs = 5
embed_size = 100 # Output_dim
max_features = 10000 # Input_dim

```

```

#Defining Neural Network
model = Sequential()
#Non-trainable embedding layer
model.add(Embedding(max_features, output_dim=embed_size, weights=[embedding_matrix], input_length=maxlen, trainable=False))
#LSTM
model.add(LSTM(units=128 , return_sequences = True , recurrent_dropout = 0.2 , dropout = 0.2))
model.add(LSTM(units=64 , recurrent_dropout = 0.2 , dropout = 0.2))
model.add(Dense(units = 32 , activation = 'relu'))
model.add(Dropout(0.2))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer=tf.keras.optimizers.Adam(lr = 0.01), loss='binary_crossentropy', metrics=['accuracy'])

```

```
model.summary()
```

```
history = model.fit(x_train, y_train, batch_size = batch_size , validation_data = (X_test,y_test) , epochs = epochs , callbacks = [learni
```

```
Model: "sequential_15"
```

Layer (type)	Output Shape	Param #
embedding_15 (Embedding)	(None, 300, 100)	1000000
lstm_30 (LSTM)	(None, 300, 128)	117248
lstm_31 (LSTM)	(None, 64)	49408
dense_30 (Dense)	(None, 32)	2080
dropout_1 (Dropout)	(None, 32)	0
dense_31 (Dense)	(None, 1)	33

```

Total params: 1,168,769
Trainable params: 168,769
Non-trainable params: 1,000,000

```

```

Epoch 1/5
33/33 [=====] - 104s 3s/step - loss: 0.2711 - accuracy: 0.8832 - val_loss: 0.0749 - val_accuracy: 0.9737
Epoch 2/5
33/33 [=====] - 98s 3s/step - loss: 0.0591 - accuracy: 0.9797 - val_loss: 0.0757 - val_accuracy: 0.9684
Epoch 3/5
33/33 [=====] - 99s 3s/step - loss: 0.0410 - accuracy: 0.9873 - val_loss: 0.0539 - val_accuracy: 0.9837
Epoch 4/5
33/33 [=====] - 98s 3s/step - loss: 0.0366 - accuracy: 0.9883 - val_loss: 0.0183 - val_accuracy: 0.9945

```

Epoch 5/5
33/33 [=====] - 97s 3s/step - loss: 0.0237 - accuracy: 0.9922 - val_loss: 0.0164 - val_accuracy: 0.9954

```
# Hyperparameter Dense Layer 3
batch_size = 1024
epochs = 5
embed_size = 100 # Output_dim
max_features = 10000 # Input_dim
```

```
#Defining Neural Network
model = Sequential()
#Non-trainable embedding layer
model.add(Embedding(max_features, output_dim=embed_size, weights=[embedding_matrix], input_length=maxlen, trainable=False))
#LSTM
model.add(LSTM(units=128, return_sequences = True, recurrent_dropout = 0.2, dropout = 0.2))
model.add(LSTM(units=64, recurrent_dropout = 0.2, dropout = 0.2))
model.add(Dense(units = 64, activation = 'relu'))
model.add(Dropout(0.2))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer=tf.keras.optimizers.Adam(lr = 0.01), loss='binary_crossentropy', metrics=['accuracy'])
```

```
model.summary()
```

```
history = model.fit(x_train, y_train, batch_size = batch_size, validation_data = (X_test,y_test), epochs = epochs, callbacks = [learni
```

Model: "sequential_16"

Layer (type)	Output Shape	Param #
embedding_16 (Embedding)	(None, 300, 100)	1000000
lstm_32 (LSTM)	(None, 300, 128)	117248
lstm_33 (LSTM)	(None, 64)	49408
dense_32 (Dense)	(None, 64)	4160
dropout_2 (Dropout)	(None, 64)	0
dense_33 (Dense)	(None, 1)	65
Total params: 1,170,881		
Trainable params: 170,881		
Non-trainable params: 1,000,000		

```
Epoch 1/5
33/33 [=====] - 102s 3s/step - loss: 0.3166 - accuracy: 0.8580 - val_loss: 0.1065 - val_accuracy: 0.9628
Epoch 2/5
33/33 [=====] - 97s 3s/step - loss: 0.0759 - accuracy: 0.9744 - val_loss: 0.0492 - val_accuracy: 0.9827
Epoch 3/5
33/33 [=====] - 97s 3s/step - loss: 0.0475 - accuracy: 0.9841 - val_loss: 0.0480 - val_accuracy: 0.9834
Epoch 4/5
33/33 [=====] - 98s 3s/step - loss: 0.0412 - accuracy: 0.9860 - val_loss: 0.0187 - val_accuracy: 0.9937
Epoch 5/5
33/33 [=====] - 98s 3s/step - loss: 0.0141 - accuracy: 0.9953 - val_loss: 0.0128 - val_accuracy: 0.9955
```

```
# Hyperparameter Dense Layer 4
batch_size = 1024
epochs = 5
embed_size = 100 # Output_dim
max_features = 10000 # Input_dim
```

```
#Defining Neural Network
model = Sequential()
#Non-trainable embedding layer
model.add(Embedding(max_features, output_dim=embed_size, weights=[embedding_matrix], input_length=maxlen, trainable=False))
#LSTM
model.add(LSTM(units=128, return_sequences = True, recurrent_dropout = 0.2, dropout = 0.2))
model.add(LSTM(units=64, recurrent_dropout = 0.2, dropout = 0.2))
model.add(Dense(units = 128, activation = 'relu'))
model.add(Dropout(0.2))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer=tf.keras.optimizers.Adam(lr = 0.01), loss='binary_crossentropy', metrics=['accuracy'])
```

```
model.summary()
```

```
history = model.fit(x_train, y_train, batch_size = batch_size, validation_data = (X_test,y_test), epochs = epochs, callbacks = [learni
```

Model: "sequential_17"

Layer (type)	Output Shape	Param #
embedding_17 (Embedding)	(None, 300, 100)	1000000

lstm_34 (LSTM)	(None, 300, 128)	117248
lstm_35 (LSTM)	(None, 64)	49408
dense_34 (Dense)	(None, 128)	8320
dropout_3 (Dropout)	(None, 128)	0
dense_35 (Dense)	(None, 1)	129

=====
 Total params: 1,175,105
 Trainable params: 175,105
 Non-trainable params: 1,000,000

```

Epoch 1/5
33/33 [=====] - 102s 3s/step - loss: 0.3640 - accuracy: 0.8215 - val_loss: 0.0931 - val_accuracy: 0.9664
Epoch 2/5
33/33 [=====] - 98s 3s/step - loss: 0.0781 - accuracy: 0.9719 - val_loss: 0.0375 - val_accuracy: 0.9874
Epoch 3/5
33/33 [=====] - 98s 3s/step - loss: 0.0247 - accuracy: 0.9915 - val_loss: 0.0115 - val_accuracy: 0.9962
Epoch 4/5
33/33 [=====] - 97s 3s/step - loss: 0.0107 - accuracy: 0.9967 - val_loss: 0.0070 - val_accuracy: 0.9978
Epoch 5/5
33/33 [=====] - 96s 3s/step - loss: 0.0064 - accuracy: 0.9979 - val_loss: 0.0075 - val_accuracy: 0.9980

```

Hyperparameter Dense Layer 5

```

batch_size = 1024
epochs = 5
embed_size = 100 # Output_dim
max_features = 10000 # Input_dim

```

#Defining Neural Network

```

model = Sequential()
#Non-trainable embedding layer
model.add(Embedding(max_features, output_dim=embed_size, weights=[embedding_matrix], input_length=maxlen, trainable=False))
#LSTM
model.add(LSTM(units=128, return_sequences = True, recurrent_dropout = 0.2, dropout = 0.2))
model.add(LSTM(units=64, recurrent_dropout = 0.2, dropout = 0.2))
model.add(Dense(units = 32, activation = 'relu'))
model.add(Dense(units = 16, activation = 'relu'))
model.add(Dropout(0.2))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer=tf.keras.optimizers.Adam(lr = 0.01), loss='binary_crossentropy', metrics=['accuracy'])

```

model.summary()

history = model.fit(x_train, y_train, batch_size = batch_size, validation_data = (X_test,y_test), epochs = epochs, callbacks = [learni

Model: "sequential_18"

Layer (type)	Output Shape	Param #
embedding_18 (Embedding)	(None, 300, 100)	1000000
lstm_36 (LSTM)	(None, 300, 128)	117248
lstm_37 (LSTM)	(None, 64)	49408
dense_36 (Dense)	(None, 32)	2080
dense_37 (Dense)	(None, 16)	528
dropout_4 (Dropout)	(None, 16)	0
dense_38 (Dense)	(None, 1)	17

=====
 Total params: 1,169,281
 Trainable params: 169,281
 Non-trainable params: 1,000,000

```

Epoch 1/5
33/33 [=====] - 102s 3s/step - loss: 0.3293 - accuracy: 0.8539 - val_loss: 0.0795 - val_accuracy: 0.9727
Epoch 2/5
33/33 [=====] - 97s 3s/step - loss: 0.0616 - accuracy: 0.9799 - val_loss: 0.0326 - val_accuracy: 0.9898
Epoch 3/5
33/33 [=====] - 95s 3s/step - loss: 0.0420 - accuracy: 0.9869 - val_loss: 0.0324 - val_accuracy: 0.9887
Epoch 4/5
33/33 [=====] - 96s 3s/step - loss: 0.0314 - accuracy: 0.9898 - val_loss: 0.0133 - val_accuracy: 0.9954
Epoch 5/5
33/33 [=====] - 97s 3s/step - loss: 0.0187 - accuracy: 0.9936 - val_loss: 0.0190 - val_accuracy: 0.9941

```

Hyperparameter Dense Layer 6

```

batch_size = 1024
epochs = 5
embed_size = 100 # Output_dim
max_features = 10000 # Input_dim

```

```
#Defining Neural Network
model = Sequential()
#Non-trainable embedding layer
model.add(Embedding(max_features, output_dim=embed_size, weights=[embedding_matrix], input_length=maxlen, trainable=False))
#LSTM
model.add(LSTM(units=128, return_sequences = True, recurrent_dropout = 0.2, dropout = 0.2))
model.add(LSTM(units=64, recurrent_dropout = 0.2, dropout = 0.2))
model.add(Dense(units = 64, activation = 'relu'))
model.add(Dense(units = 32, activation = 'relu'))
model.add(Dropout(0.2))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer=tf.keras.optimizers.Adam(lr = 0.01), loss='binary_crossentropy', metrics=['accuracy'])

model.summary()

history = model.fit(x_train, y_train, batch_size = batch_size, validation_data = (X_test,y_test), epochs = epochs, callbacks = [learn
```

Model: "sequential_19"

Layer (type)	Output Shape	Param #
embedding_19 (Embedding)	(None, 300, 100)	1000000
lstm_38 (LSTM)	(None, 300, 128)	117248
lstm_39 (LSTM)	(None, 64)	49408
dense_39 (Dense)	(None, 64)	4160
dense_40 (Dense)	(None, 32)	2080
dropout_5 (Dropout)	(None, 32)	0
dense_41 (Dense)	(None, 1)	33

Total params: 1,172,929
 Trainable params: 172,929
 Non-trainable params: 1,000,000

```
Epoch 1/5
33/33 [=====] - 102s 3s/step - loss: 0.3473 - accuracy: 0.8558 - val_loss: 0.1019 - val_accuracy: 0.9660
Epoch 2/5
33/33 [=====] - 96s 3s/step - loss: 0.0744 - accuracy: 0.9743 - val_loss: 0.0352 - val_accuracy: 0.9873
Epoch 3/5
33/33 [=====] - 97s 3s/step - loss: 0.1066 - accuracy: 0.9697 - val_loss: 0.0605 - val_accuracy: 0.9792
Epoch 4/5
33/33 [=====] - 97s 3s/step - loss: 0.0481 - accuracy: 0.9841 - val_loss: 0.0298 - val_accuracy: 0.9914
Epoch 5/5
33/33 [=====] - 98s 3s/step - loss: 0.0262 - accuracy: 0.9918 - val_loss: 0.0162 - val_accuracy: 0.9947
```

```
# Hyperparameter Dense Layer 7
batch_size = 1024
epochs = 5
embed_size = 100 # Output_dim
max_features = 10000 # Input_dim
```

```
#Defining Neural Network
model = Sequential()
#Non-trainable embedding layer
model.add(Embedding(max_features, output_dim=embed_size, weights=[embedding_matrix], input_length=maxlen, trainable=False))
#LSTM
model.add(LSTM(units=128, return_sequences = True, recurrent_dropout = 0.2, dropout = 0.2))
model.add(LSTM(units=64, recurrent_dropout = 0.2, dropout = 0.2))
model.add(Dense(units = 128, activation = 'relu'))
model.add(Dense(units = 64, activation = 'relu'))
model.add(Dropout(0.2))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer=tf.keras.optimizers.Adam(lr = 0.01), loss='binary_crossentropy', metrics=['accuracy'])

model.summary()

history = model.fit(x_train, y_train, batch_size = batch_size, validation_data = (X_test,y_test), epochs = epochs, callbacks = [learn
```

Model: "sequential_20"

Layer (type)	Output Shape	Param #
embedding_20 (Embedding)	(None, 300, 100)	1000000
lstm_40 (LSTM)	(None, 300, 128)	117248
lstm_41 (LSTM)	(None, 64)	49408
dense_42 (Dense)	(None, 128)	8320

dense_43 (Dense)	(None, 64)	8256
dropout_6 (Dropout)	(None, 64)	0
dense_44 (Dense)	(None, 1)	65

=====
Total params: 1,183,297
Trainable params: 183,297
Non-trainable params: 1,000,000

```
Epoch 1/5
33/33 [=====] - 101s 3s/step - loss: 0.3031 - accuracy: 0.8561 - val_loss: 0.0969 - val_accuracy: 0.9678
Epoch 2/5
33/33 [=====] - 97s 3s/step - loss: 0.0667 - accuracy: 0.9767 - val_loss: 0.0297 - val_accuracy: 0.9910
Epoch 3/5
33/33 [=====] - 97s 3s/step - loss: 0.0385 - accuracy: 0.9878 - val_loss: 0.0186 - val_accuracy: 0.9941
Epoch 4/5
33/33 [=====] - 96s 3s/step - loss: 0.0177 - accuracy: 0.9950 - val_loss: 0.0111 - val_accuracy: 0.9966
Epoch 5/5
33/33 [=====] - 97s 3s/step - loss: 0.0071 - accuracy: 0.9981 - val_loss: 0.0090 - val_accuracy: 0.9976
```

```
# Hyperparameter Dense Layer 8
batch_size = 1024
epochs = 5
embed_size = 100 # Output_dim
max_features = 10000 # Input_dim
```

```
#Defining Neural Network
model = Sequential()
#Non-trainable embedding layer
model.add(Embedding(max_features, output_dim=embed_size, weights=[embedding_matrix], input_length=maxlen, trainable=False))
#LSTM
model.add(LSTM(units=128, return_sequences = True, recurrent_dropout = 0.2, dropout = 0.2))
model.add(LSTM(units=64, recurrent_dropout = 0.2, dropout = 0.2))
model.add(Dense(units = 128, activation = 'relu'))
model.add(Dense(units = 64, activation = 'relu'))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer=tf.keras.optimizers.Adam(lr = 0.01), loss='binary_crossentropy', metrics=['accuracy'])
```

```
model.summary()
```

```
history = model.fit(x_train, y_train, batch_size = batch_size, validation_data = (X_test,y_test), epochs = epochs, callbacks = [learni
```

```
Model: "sequential_21"
```

Layer (type)	Output Shape	Param #
embedding_21 (Embedding)	(None, 300, 100)	1000000
lstm_42 (LSTM)	(None, 300, 128)	117248
lstm_43 (LSTM)	(None, 64)	49408
dense_45 (Dense)	(None, 128)	8320
dense_46 (Dense)	(None, 64)	8256
dropout_7 (Dropout)	(None, 64)	0
dense_47 (Dense)	(None, 1)	65

=====
Total params: 1,183,297
Trainable params: 183,297
Non-trainable params: 1,000,000

```
Epoch 1/5
33/33 [=====] - 100s 3s/step - loss: 0.3954 - accuracy: 0.8079 - val_loss: 0.1418 - val_accuracy: 0.9527
Epoch 2/5
33/33 [=====] - 97s 3s/step - loss: 0.0998 - accuracy: 0.9665 - val_loss: 0.0452 - val_accuracy: 0.9854
Epoch 3/5
33/33 [=====] - 97s 3s/step - loss: 0.0478 - accuracy: 0.9864 - val_loss: 0.0207 - val_accuracy: 0.9935
Epoch 4/5
33/33 [=====] - 96s 3s/step - loss: 0.0208 - accuracy: 0.9941 - val_loss: 0.0140 - val_accuracy: 0.9965
Epoch 5/5
33/33 [=====] - 97s 3s/step - loss: 0.0109 - accuracy: 0.9970 - val_loss: 0.0086 - val_accuracy: 0.9972
```

Experiment 3 Optimizer Output Layer

```
# Hyperparameter Optimizer Output Layer 1
batch_size = 1024
epochs = 5
embed_size = 100 # Output_dim
```

```

max_features = 10000 # Input_dim

#Defining Neural Network
model = Sequential()
#Non-trainable embedding layer
model.add(Embedding(max_features, output_dim=embed_size, weights=[embedding_matrix], input_length=maxlen, trainable=False))
#LSTM
model.add(LSTM(units=128 , return_sequences = True , recurrent_dropout = 0.2 , dropout = 0.2))
model.add(LSTM(units=64 , recurrent_dropout = 0.2 , dropout = 0.2))
model.add(Dense(units = 128 , activation = 'relu'))
model.add(Dense(units = 64 , activation = 'relu'))
model.add(Dropout(0.2))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer=tf.keras.optimizers.Adam(lr = 0.001), loss='binary_crossentropy', metrics=['accuracy'])

model.summary()

history = model.fit(x_train, y_train, batch_size = batch_size , validation_data = (X_test,y_test) , epochs = epochs , callbacks = [learni

```

Model: "sequential_22"

Layer (type)	Output Shape	Param #
embedding_22 (Embedding)	(None, 300, 100)	1000000
lstm_44 (LSTM)	(None, 300, 128)	117248
lstm_45 (LSTM)	(None, 64)	49408
dense_48 (Dense)	(None, 128)	8320
dense_49 (Dense)	(None, 64)	8256
dropout_8 (Dropout)	(None, 64)	0
dense_50 (Dense)	(None, 1)	65

Total params: 1,183,297

Trainable params: 183,297

Non-trainable params: 1,000,000

```

Epoch 1/5
33/33 [=====] - 105s 3s/step - loss: 0.3533 - accuracy: 0.8520 - val_loss: 0.1589 - val_accuracy: 0.9390
Epoch 2/5
33/33 [=====] - 98s 3s/step - loss: 0.1442 - accuracy: 0.9449 - val_loss: 0.0992 - val_accuracy: 0.9629
Epoch 3/5
33/33 [=====] - 99s 3s/step - loss: 0.1069 - accuracy: 0.9603 - val_loss: 0.0872 - val_accuracy: 0.9667
Epoch 4/5
33/33 [=====] - 98s 3s/step - loss: 0.0867 - accuracy: 0.9680 - val_loss: 0.0600 - val_accuracy: 0.9806
Epoch 5/5
33/33 [=====] - 98s 3s/step - loss: 0.0567 - accuracy: 0.9798 - val_loss: 0.0478 - val_accuracy: 0.9854

```

Hyperparameter Optimizer Output Layer 2

```

batch_size = 1024
epochs = 5
embed_size = 100 # Output_dim
max_features = 10000 # Input_dim

```

```

#Defining Neural Network
model = Sequential()
#Non-trainable embedding layer
model.add(Embedding(max_features, output_dim=embed_size, weights=[embedding_matrix], input_length=maxlen, trainable=False))
#LSTM
model.add(LSTM(units=128 , return_sequences = True , recurrent_dropout = 0.2 , dropout = 0.2))
model.add(LSTM(units=64 , recurrent_dropout = 0.2 , dropout = 0.2))
model.add(Dense(units = 128 , activation = 'relu'))
model.add(Dense(units = 64 , activation = 'relu'))
model.add(Dropout(0.2))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer=tf.keras.optimizers.Adam(lr = 0.01), loss='binary_crossentropy', metrics=['accuracy'])

model.summary()

```

```

history = model.fit(x_train, y_train, batch_size = batch_size , validation_data = (X_test,y_test) , epochs = epochs , callbacks = [learni

```

Model: "sequential_23"

Layer (type)	Output Shape	Param #
embedding_23 (Embedding)	(None, 300, 100)	1000000
lstm_46 (LSTM)	(None, 300, 128)	117248
lstm_47 (LSTM)	(None, 64)	49408

```
dense_51 (Dense)      (None, 128)      8320
-----
dense_52 (Dense)      (None, 64)       8256
-----
dropout_9 (Dropout)   (None, 64)       0
-----
dense_53 (Dense)      (None, 1)        65
=====
Total params: 1,183,297
Trainable params: 183,297
Non-trainable params: 1,000,000
```

```
Epoch 1/5
33/33 [=====] - 102s 3s/step - loss: 0.2747 - accuracy: 0.8701 - val_loss: 0.0740 - val_accuracy: 0.9741
Epoch 2/5
33/33 [=====] - 98s 3s/step - loss: 0.0630 - accuracy: 0.9786 - val_loss: 0.0415 - val_accuracy: 0.9872
Epoch 3/5
33/33 [=====] - 96s 3s/step - loss: 0.0262 - accuracy: 0.9913 - val_loss: 0.0115 - val_accuracy: 0.9960
Epoch 4/5
33/33 [=====] - 98s 3s/step - loss: 0.0123 - accuracy: 0.9965 - val_loss: 0.0129 - val_accuracy: 0.9952
Epoch 5/5
33/33 [=====] - 97s 3s/step - loss: 0.0112 - accuracy: 0.9965 - val_loss: 0.0099 - val_accuracy: 0.9977
```

Hyperparameter Optimizer Output Layer 3

```
batch_size = 1024
epochs = 5
embed_size = 100 # Output_dim
max_features = 10000 # Input_dim
```

#Defining Neural Network

```
model = Sequential()
#Non-trainable embedding layer
model.add(Embedding(max_features, output_dim=embed_size, weights=[embedding_matrix], input_length=maxlen, trainable=False))
#LSTM
model.add(LSTM(units=128, return_sequences = True, recurrent_dropout = 0.2, dropout = 0.2))
model.add(LSTM(units=64, recurrent_dropout = 0.2, dropout = 0.2))
model.add(Dense(units = 128, activation = 'relu'))
model.add(Dense(units = 64, activation = 'relu'))
model.add(Dropout(0.2))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer=tf.keras.optimizers.Adam(lr = 0.1), loss='binary_crossentropy', metrics=['accuracy'])
```

```
model.summary()
```

```
history = model.fit(x_train, y_train, batch_size = batch_size, validation_data = (X_test,y_test), epochs = epochs, callbacks = [learni
```

Model: "sequential_24"

Layer (type)	Output Shape	Param #
embedding_24 (Embedding)	(None, 300, 100)	1000000
lstm_48 (LSTM)	(None, 300, 128)	117248
lstm_49 (LSTM)	(None, 64)	49408
dense_54 (Dense)	(None, 128)	8320
dense_55 (Dense)	(None, 64)	8256
dropout_10 (Dropout)	(None, 64)	0
dense_56 (Dense)	(None, 1)	65

```
=====
Total params: 1,183,297
Trainable params: 183,297
Non-trainable params: 1,000,000
```

```
Epoch 1/5
33/33 [=====] - 101s 3s/step - loss: 2.8456 - accuracy: 0.4999 - val_loss: 0.6933 - val_accuracy: 0.4781
Epoch 2/5
33/33 [=====] - 96s 3s/step - loss: 0.6926 - accuracy: 0.5214 - val_loss: 0.6922 - val_accuracy: 0.5219
Epoch 3/5
33/33 [=====] - 97s 3s/step - loss: 0.6921 - accuracy: 0.5234 - val_loss: 0.6922 - val_accuracy: 0.5219
Epoch 4/5
33/33 [=====] - 98s 3s/step - loss: 0.6921 - accuracy: 0.5234 - val_loss: 0.6923 - val_accuracy: 0.5219
Epoch 00004: ReduceLRonPlateau reducing learning rate to 0.05000000074505806.
Epoch 5/5
33/33 [=====] - 97s 3s/step - loss: 0.6921 - accuracy: 0.5234 - val_loss: 0.6922 - val_accuracy: 0.5219
```

OPTIMIZED LSTM MODEL

```
batch_size = 1024
epochs = 10
```

```
# Hyperparameter Optimizer Output Layer 3
embed_size = 100 # Output_dim
max_features = 10000 # Input_dim

#Defining Neural Network
model = Sequential()
#Non-trainable embedding layer
model.add(Embedding(max_features, output_dim=embed_size, weights=[embedding_matrix], input_length=maxlen, trainable=False))
#LSTM
model.add(LSTM(units=128, return_sequences = True, recurrent_dropout = 0.2, dropout = 0.2))
model.add(LSTM(units=64, recurrent_dropout = 0.2, dropout = 0.2))
model.add(Dense(units = 128, activation = 'relu'))
model.add(Dense(units = 64, activation = 'relu'))
model.add(Dropout(0.2))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer=tf.keras.optimizers.Adam(lr = 0.01), loss='binary_crossentropy', metrics=['accuracy'])

model.summary()

history = model.fit(x_train, y_train, batch_size = batch_size, validation_data = (X_test,y_test), epochs = epochs, callbacks = [learnr
```

Model: "sequential_25"

Layer (type)	Output Shape	Param #
embedding_25 (Embedding)	(None, 300, 100)	1000000
lstm_50 (LSTM)	(None, 300, 128)	117248
lstm_51 (LSTM)	(None, 64)	49408
dense_57 (Dense)	(None, 128)	8320
dense_58 (Dense)	(None, 64)	8256
dropout_11 (Dropout)	(None, 64)	0
dense_59 (Dense)	(None, 1)	65

Total params: 1,183,297

Trainable params: 183,297

Non-trainable params: 1,000,000

```
Epoch 1/10
33/33 [=====] - 106s 3s/step - loss: 0.4439 - accuracy: 0.8081 - val_loss: 0.1195 - val_accuracy: 0.9579
Epoch 2/10
33/33 [=====] - 101s 3s/step - loss: 0.0932 - accuracy: 0.9686 - val_loss: 0.0835 - val_accuracy: 0.9686
Epoch 3/10
33/33 [=====] - 100s 3s/step - loss: 0.0604 - accuracy: 0.9794 - val_loss: 0.0332 - val_accuracy: 0.9892
Epoch 4/10
33/33 [=====] - 101s 3s/step - loss: 0.0203 - accuracy: 0.9931 - val_loss: 0.0124 - val_accuracy: 0.9959
Epoch 5/10
33/33 [=====] - 101s 3s/step - loss: 0.0106 - accuracy: 0.9967 - val_loss: 0.0145 - val_accuracy: 0.9954
Epoch 6/10
33/33 [=====] - 100s 3s/step - loss: 0.0060 - accuracy: 0.9982 - val_loss: 0.0086 - val_accuracy: 0.9977
Epoch 7/10
33/33 [=====] - 100s 3s/step - loss: 0.0037 - accuracy: 0.9988 - val_loss: 0.0066 - val_accuracy: 0.9984
Epoch 8/10
33/33 [=====] - 100s 3s/step - loss: 0.0025 - accuracy: 0.9991 - val_loss: 0.0079 - val_accuracy: 0.9987
Epoch 9/10
33/33 [=====] - 100s 3s/step - loss: 0.0031 - accuracy: 0.9992 - val_loss: 0.0058 - val_accuracy: 0.9988
Epoch 10/10
33/33 [=====] - 100s 3s/step - loss: 0.0021 - accuracy: 0.9995 - val_loss: 0.0077 - val_accuracy: 0.9979
```

Epoch 00010: ReduceLRonPlateau reducing learning rate to 0.004999999888241291.

▼ ANALYSIS AFTER MODEL TRAINING

```
print("Accuracy of the model on Training Data is - ", model.evaluate(x_train,y_train)[1]*100, "%")
print("Accuracy of the model on Testing Data is - ", model.evaluate(X_test,y_test)[1]*100, "%")

1053/1053 [=====] - 155s 148ms/step - loss: 9.5871e-04 - accuracy: 0.9997
Accuracy of the model on Training Data is - 99.9703049659729 %
351/351 [=====] - 52s 148ms/step - loss: 0.0077 - accuracy: 0.9979
Accuracy of the model on Testing Data is - 99.7861921787262 %
```

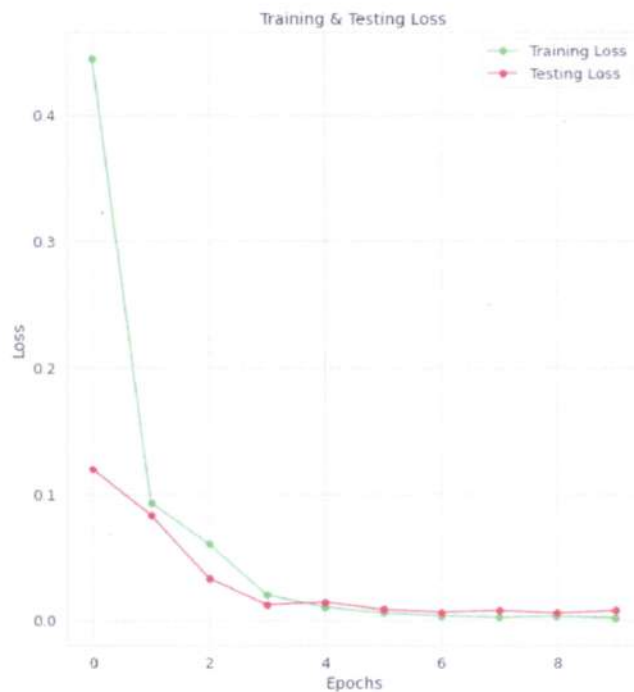
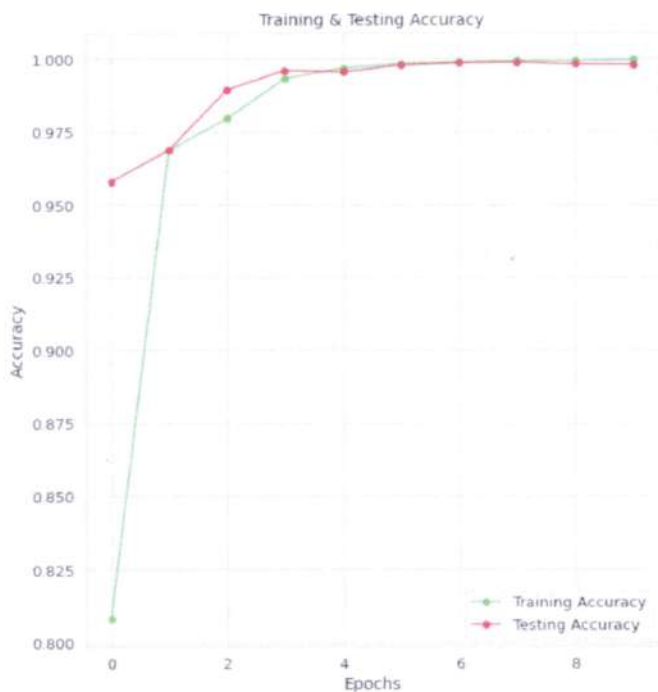
```
epochs = [i for i in range(10)]
fig, ax = plt.subplots(1,2)
train_acc = history.history['accuracy']
train_loss = history.history['loss']
val_acc = history.history['val_accuracy']
val_loss = history.history['val_loss']
```



```
fig.set_size_inches(20,10)
```

```
ax[0].plot(epochs , train_acc , 'go-' , label = 'Training Accuracy')
ax[0].plot(epochs , val_acc , 'ro-' , label = 'Testing Accuracy')
ax[0].set_title('Training & Testing Accuracy')
ax[0].legend()
ax[0].set_xlabel("Epochs")
ax[0].set_ylabel("Accuracy")
```

```
ax[1].plot(epochs , train_loss , 'go-' , label = 'Training Loss')
ax[1].plot(epochs , val_loss , 'ro-' , label = 'Testing Loss')
ax[1].set_title('Training & Testing Loss')
ax[1].legend()
ax[1].set_xlabel("Epochs")
ax[1].set_ylabel("Loss")
plt.show()
```



```
pred = (model.predict(X_test) > 0.5).astype("int32")
pred[:5]
```

```
array([[0],
       [0],
       [0],
       [0],
       [1]], dtype=int32)
```

```
print(classification_report(y_test, pred, target_names = ['Fake', 'Not Fake']))
```

	precision	recall	f1-score	support
Fake	1.00	1.00	1.00	5858
Not Fake	1.00	1.00	1.00	5367
accuracy			1.00	11225
macro avg	1.00	1.00	1.00	11225
weighted avg	1.00	1.00	1.00	11225

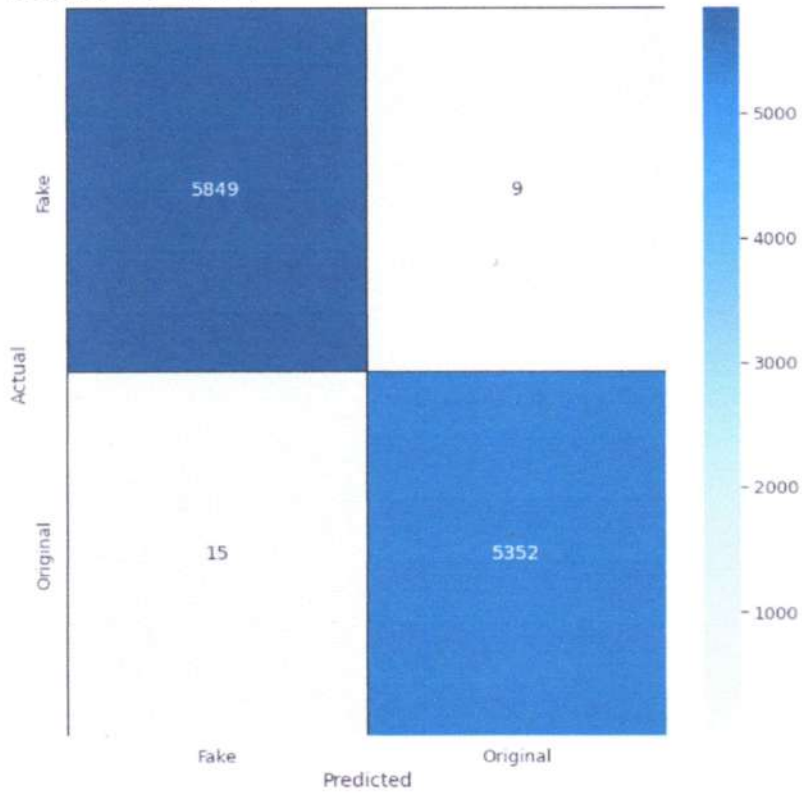
```
cm = confusion_matrix(y_test,pred)
cm
```

```
array([[5849,  9],
       [ 15, 5352]])
```

```
cm = pd.DataFrame(cm , index = ['Fake','Original'] , columns = ['Fake','Original'])
```

```
plt.figure(figsize = (10,10))
sns.heatmap(cm,cmap= "Blues", linecolor = 'black' , linewidth = 1 , annot = True, fmt='', xticklabels = ['Fake','Original'] , yticklabel
plt.xlabel("Predicted")
plt.ylabel("Actual")
```

Text(63.5, 0.5, 'Actual')



Program Pengujian Model Deteksi Berita Palsu

Model Testing dengan entri manual

```
### News
"""

def output_label(n):
    if n == 0:
        return "Fake News"
    elif n == 1:
        return "Not A Fake News"

def manual_testing(news):
    testing_news = {"text":[news]}
    new_def_test = pd.DataFrame(testing_news)
    new_def_test["text"] = new_def_test["text"].apply(wordopt)
    new_x_test = new_def_test["text"]
    new_xv_test = vectorization.transform(new_x_test)
    pred = model.predict(new_xv_test)

    return print("This news is {}".format(output_label(pred[0])) )
```

Hasil Deteksi Model untuk Kategori Berita Palsu

```
▶ news = str(input())
  manual_testing(news)

↳ In the wake of a bombshell Washington
   This news is Fake News
```

Hasil Deteksi Model untuk Kategori Bukan Berita Palsu

```
▶ news = str(input())
  manual_testing(news)

↳ JAKARTA (Reuters) - Indonesia will buy 11 Sukhoi fighter jets
   This news is Not A Fake News
```

DAFTAR RIWAYAT HIDUP

NAMA LENGKAP : MUHAMMAD FADLI RAMADHAN
NIM : 0619 4035 2343
TEMPAT TANGGAL LAHIR : PALEMBANG, 06 DESEMBER 2001
ALAMAT : PERUM. BUKIT SEJAHTERA, BLOK EK-21
TELEPON : 081274278668

RIWAYAT PENDIDIKAN FORMAL

PENDIDIKAN	NAMA SEKOLAH	TAMAT TAHUN
SD	SD ISLAM AZZAHRAH PALEMBANG	2013
SMP	SMP ISLAM AZ-ZAHRAH 2 PALEMBANG	2016
SMA	SMA NEGERI 10 PALEMBANG	2019

RIWAYAT PENDIDIKAN NON-FORMAL

JENIS PENDIDIKAN NON-FORMAL	INSTANSI	TAHUN
MACHINE LEARNING DEVELOPER	DICODING ACADEMY	2021-2022

PENGALAMAN MAGANG DAN STUDI INDEPENDEN BERSERTIFIKAT

NO.	PENGALAMAN	INSTANSI	PERIODE
1	PENGEMBANG MACHINE LEARNING DAN FRONT-END WEB	PT. PRESENTOLOGICS (DICODING)	23 Agu 2021 - 19 Jan 2022
2	BANGKIT ACADEMY 2022 BY GOOGLE, GOTO, TRAVELOKA – MACHINE LEARNING LEARNING PATH	PT. PRESENTOLOGICS (DICODING)	14 Feb 2022 – 29 Jul 2022
3	MAGANG	BALAI MONITOR SPEKTRUM FREKUENSI RADIO KELAS I PALEMBANG	15 Agu 2022 – 15 Feb 2023

PENGALAMAN PROYEK

NO.	NAMA PROYEK	URAIAN	TAHUN
1	GREENSCENE	CAPSTONE PROJECT AKHIR MSIB BATCH 1 DICODING, WEBSITE PENDETEKESI KEBERSIHAN LINGKUNGAN	2021
2	SKINCAN – SKIN CANCER DETECTION APP	CAPSTONE PROJECT AKHIR MSIB BATCH 2 BANGKIT ACADEMY, APLIKASI ANDROID YANG DAPAT MENDETEKSI GEJALA DINI KANKER KULIT SECARA REAL-TIME MELALUI KAMERA	2022

PENGALAMAN SERTIFIKASI

NO.	JENIS SERTIFIKASI	TAHUN
1	JUNIOR NETWORK ADMINISTRATOR SKKNI	28 MEI 2021
2	GOOGLE CERTIFIED TENSORFLOW DEVELOPER	01 AGUSTUS 2022

Semua data yang saya tulis dan isi pada daftar riwayat hidup ini adalah benar adalah dapat dipertanggungjawabkan.

Palembang, Agustus 2023

(MUHAMMAD FADLI RAMADHAN)