electronics

Article

A Novel Hybrid Deep Learning Model for Detecting and Classifying Non-Functional Requirements of Mobile Apps Issues

Abdulsamad E. Yahya [1], Atef Gharbi [1], Wael M. S. Yafooz [2,*] and Arafat Al-Dhaqm [3]

1 Faculty of Computing and Information Technology, Northern Border University, Rafha 76413, Saudi Arabia
2 Department of Computer Science, College of Computer Science and Engineering, Taibah University, Medina 42353, Saudi Arabia
3 Faculty of Computing, Universiti Teknologi Malaysia, Johor Bahru 81310 UTM, Johor, Malaysia
* Correspondence: wyfooz@taibahu.edu.sa

Abstract: As a result of the speed and availability of the Internet, mobile devices and apps are in widespread usage throughout the world. Thus, they can be seen in the hands of nearly every person, helping us in our daily activities to accomplish many tasks with less effort and without wasting time. However, many issues occur while using mobile apps, which can be considered as issues of functional or non-functional requirements (NFRs). Users can add their comments as a review on the mobile app stores that provide for technical feedback, which can be used to improve the software quality and features of the mobile apps. Minimum attention has been given to such comments by scholars in addressing, detecting, and classifying issues related to NFRs, which are still considered challenging. The purpose of this paper is to propose a hybrid deep learning model to detect and classify NFRs (according to usability, reliability, performance, and supportability) of mobile apps using natural language processing methods. The hybrid model combines three deep learning (DL) architectures: a recurrent neural network (RNN) and two long short-term memory (LSTM) models. It starts with a dataset construction extracted from the user textual reviews that contain significant information in the Arabic language. Several experiments were conducted using machine learning classifiers (MCLs) and DL, such as ANN, LSTM, and bidirectional LSTM architecture to measure the performance of the proposed hybrid deep learning model. The experimental results show that the performance of the proposed hybrid deep learning model outperforms all other models in terms of the F1 score measure, which reached 96%. This model helps mobile developers improve the quality of their apps to meet user satisfaction and expectations by detecting and classifying issues relating to NFRs.

Keywords: mobile apps; machine learning; deep learning; non-functional requirements

check for updates

Citation: Yahya, A.E.; Gharbi, A.; Yafooz, W.M.S.; Al-Dhaqm, A. A Novel Hybrid Deep Learning Model for Detecting and Classifying Non-Functional Requirements of Mobile Apps Issues. *Electronics* **2023**, *12*, 1258. https://doi.org/10.3390/electronics12051258

Academic Editor: Arkaitz Zubiaga

Received: 30 January 2023
Revised: 25 February 2023
Accepted: 27 February 2023
Published: 6 March 2023

1. Introduction

The growth in mobile application development and usage in several aspects of life has increased the usage of apps in daily activities as a whole. This increase is due to the proliferation of mobile devices, their portability, and their accessibility [1]. Many people, whether old or young, use mobile applications for entertainment or other reasons. The huge demand for mobile devices has encouraged software companies to develop more mobile apps. Several apps have been developed to assist with teaching services, training services, education services, contracting services, food delivery services, market staff delivery, hospital assistance, health service, etc. The most popular platform for mobile apps is the Google apps store, the Apple app store, and many other specialized app stores. These platforms allow users to express their feedback through reviews and ratings that are considered indicators of the quality of apps [2,3].

User comments and reviews of mobile apps can help in improving the quality and features of software used in mobile apps. User reviews communicate valuable information to mobile apps developers, including information of technical value and complaints about the functionality of the system [4,5]. Such information can be used by developers as a source for improving the functionality of mobile apps and meeting user expectations. Therefore, via textual user reviews, apps developers can improve software usability and determine user satisfaction, as well gain an understanding user needs. Users often prefer to learn and read the other users' comments to understand their experiences with mobile apps before making a decision. In addition, users can form an initial idea about mobile apps and compare features and functions before buying, downloading or using an app [6].

Scholarly reviews of mobile apps and ratings from 0–5 are utilized to classify bug reports and user complaint issues regarding system functions. In addition, using existing data analytics, companies focus on sentiment analysis and study user comments in assessing products or provided services. Software-oriented issues remain an interesting area to study, particularly the non-functional requirements (NFRs) of mobile apps, which play a significant role in user experience and user satisfaction. NFRs are considered as measures that are used to assess the quality of mobile apps. NFRs are viewed from the users' perspectives in meeting user satisfaction with mobile apps.

NFRs include usability, reliability, performance, behavior issues, and security. The usability of a mobile app refers to how the functions and the graphical user interface of the mobile app operate. Users face difficulties in using mobile apps for many different reasons. Reliability measures the consistency of mobile app functions based on updated feedback from users, as well as function failure [7,8]. Performance refers to the issues related to mobile apps during installation, opening, and carrying out the tasks or functions of an app.

Using manual procedures, researchers have paid attention to text mining in mobile apps to reduce human efforts and time consumption in detecting and classifying issues related to NFRs [9]. The use of machine learning and deep learning models is increasing, due to their high performance in detecting and classifying issues in natural language processing or computer vision domains using existing methods in mining information about mobile apps' classifications [4,10,11], summarizations of apps' features [12,13], comparisons of apps' features [14], sentiment analysis [15–21], and wearable devices [22–24]. However, to achieve a high performance rate, a huge amount of data is required. In addition, mining, detecting, and classifying issues related to NFRs are given little attention. Detecting and classifying mobile issues in the Arabic language remains a challenge. Therefore, exploiting and combining multiple DL models can lead to significant improvements in performance, compared with that of other models. In addition, an automated technique is required to detect and classify mobile app issues.

Therefore, this research paper proposes a hybrid deep learning model that can detect and classify issues related to NFRs, using textual user reviews that are extracted from the Google Play Store. The proposed model combines three DL learning architectures: an RNN model and two LSTM models. This hybrid model was trained and evaluated based on an introduced Arabic dataset of NFRs. The dataset was collected from the most popular home-delivery apps. The dataset preparation used several natural language processing methods, such as data extraction, data cleaning, and data annotation. The dataset contained the five main classes of NFRs: usability, reliability, performance, behavior issues, and security. The dataset was annotated by three experts, including an expert with a Ph. D. in software engineering. Additionally, several experiments were carried out, based on machine learning classifiers and deep learning approaches—ANN, RNN, LSTM, and Bi-LSTM. The various models were examined on the basis of various hyper-parameters. The experimental results showed that, in terms of F1 scores, the proposed hybrid model outperformed the ML classifiers and the DL models.

The main contributions of this study are summarized as follows:

- Issues related to NFRs in mobile apps were explored;
- A hybrid model was proposed and implemented to detect and classify mobile apps issues related to NFRs, using RNN and LSTM models;
- An Arabic dataset was constructed, utilizing user reviews from the Google Play Store for five classes of NFRs;
- The effect of data augmentation on the NFRs' dataset was examined to improve the model performance and solve the overfitting issue. In addition, the Arabic auto corrector technique was used to enhance the model's performance in detecting and classifying issues of mobile apps;
- Finally, the performance of the proposed hybrid deep leaning model was evaluated, analyzed, and compared with those of many ML classifiers and DL models.

The remaining structure of this paper is organized as follows: Section 2 discusses related works. The problems in detecting and classifying mobile apps issues are explained in Section 3. The methodology is discussed in Section 4. A discussion of the results is provided in Section 4. Finally, the conclusions and suggestions for future work are provided in Section 5.

2. Related Studies

User satisfaction and meeting user needs are the key factors in the successful development of mobile apps. Therefore, attention to feedback from users regarding mobile issues can achieve the aforementioned objectives. Existing methods and related methods are used to detect and classify mobiles apps issues. Recently, the issue of mobile apps has been given attention by researchers to improve software quality and meet user satisfaction.

Aslam et al., in [25], proposed a deep learning model to classify mobile apps' user reviews. The proposed mode trained on the basis of meta-data—1,126,453 items of textual information extracted from Apple and 146,057 items of information extracted from Google Play Store. It is classified the information into five classes: rating, bug, enhancement, user, and experience. The model performance achieved at the average level of accuracy for the five classes was 94%. Similarly, Ciurumelea et al. [11] helped developers save time and effort in analyzing user feedback. The authors proposed a method to classify user comments based on the issues discussed by user requests. The gradient boosted regression trees (GBRT) model, trained on a dataset of 7,754 user comments from 39 mobile apps, categorized the user comments into high-level and low-level taxonomy. The average F1 score recorded was 87.7% for the high level, while for the low level, the average F1 score was 85.5%.

Rustam et al., in [26], developed a sentiment analysis on a public dataset of the Shopify app store dataset. They divided the dataset into happy and unhappy classes. Then, they carried out several experiments using machine learning classifiers with feature-engineering methods, such as TFIDF, BoW, and Chi2. The best accuracy was recorded using logistics registration, reaching 86% with a combination of TFIDF and Chi2 as input features. Similarly, Guzman et al., in [27], applied machine learning as single and ensemble classifiers on a dataset collected from user mobile reviews that contained 4450 user textual comments, with seven classes. The best F1 score was achieved using ensemble approaches, particularly the approaches of naive Bayes (NB) machines and support vector (SVMs) machines, which recorded 64%.

In addition, Isa et al., in [28], focused on a sentiment analysis with an optimization process using machine learning classifiers NB, and SVM and a decision tree (DT). The experiments were carried out on a dataset collected from Google Play Store, with 64,294 user comments. They used different feature engineering, such as TFIDF, word2Vec, and Word2Doc, with a grid search algorithm. The model's performance, in terms of accuracy, was recorded after the optimization process using the DT classifier. In a different way. Li et al., in [29], focused on user comments based on mobile apps before and after updates. They studied the positive and negative user comments. The dataset was collected from WhatsApp with 1,148,032 user comments. They applied sentiment analysis and topic modelling to identify the similarities and correlations between user comments before and after the mobile app's updates. In the same way, Li et al., in [14], studied comparative opinions among mobiles apps user comments, then summarized the opinions based on topic analysis. The experiment was based on dividing around 5 million reviews into six categories from Google Play Store. The precision on the comparative opinions was 93.7%, while the summarization was 60%.

Panichella et al., in [30], proposed a tool for extracting and analyzing user review comments to help developers identify issues and weaknesses of mobile applications. It was based on three main approaches: NLP, sentiment analysis, and machine learning using Weka API. They categorized user reviews into five classes: user seeking information, information giving, problem discovery, request features, and others. The tool archived 88.9% of the F1- score measure. Luiz and Maalej and Jha and Mahmoud, in [31,32], used the same three steps to identify the issues by extracting the topics using topic modelling. The sentiment analysis was detected in the user review and, finally, provided summarization for mobile developers. In the end, these methods provided useful information for developers from user comments, which could have a positive or a negative impact.

Jha and Mahmoud, in [8], proposed a method to classify the NFRs of mobile issues into four classes: performance, support, usability, and dependability. They trainde the model based on 6000 user comments. Both manual and automatic classifications were applied. Automatic classification used NB and SVM on a dataset of 1,100 user comments. The model's performances were an average of 70% for precision and 86% for recall. Malik and Shakshuki in [33] extracted features from 8,000 user reviews that were collected from Google Play Store. Sentiment analysis was applied to the user comments and the final stage provided a recommendation via a similarity among user comments.

Mobile app models have been proposed in the literature for several purposes. For example, Sankar et al., in [20], investigated the usability of mobile applications within virtual environments, considering several criteria, including the effectiveness and efficiency of task completion, user satisfaction, and the number of errors users made. Empirical research was carried out by [21] with the use of a set of measures to assess the usability of mobile applications that operate on various mobile operating systems. Previously, Mujahid et al., in [22], investigated the most important characteristics that define mobile applications to facilitate the delivery of valuable, exceptional, and user-friendly mobile apps to meet user requirements. Mardonova and Choi, in [23], performed a systematic literature review for the identification and collection of required evidence regarding the automated testing of mobile applications. The measurement of usability was carried out by considering three factors, i.e., effectiveness, efficiency, and satisfaction. Mujahid, in [24], designed a novel usability model in which people were the center of mobile application development. In addition, they reviewed mobile applications for specific fields. They reported that a variety of mobile applications have been designed for use in health services, e.g., heart failure symptoms [25], cardiology [26], and diabetes [27]. Another study, [28], suggested seven different strategies to assess and select health-related applications. Table 1 illustrates the comparative study of the existing methods of detecting and classifying mobile apps issues.

Table 1. Comparative analysis between mobile apps user mining methods.

Author(s)	Platform	Dataset Size	Classes	Data Type	Approach	Accuracy/F1 Score
[11]	Google Play Store	7754	Six	User comments	GBRT	Average 87.7%
[25]	Apple and Google Play Store	1,126,453 146,057	Four	User comments and meta-data	CNN	94%
[26]	Shopify App Store dataset	287,467	Two	User comments and rating	Logistics regression Random forest ADABOOST	86% using LR
[27]	Multiple sources	4550	Seven	User comment	ML classifiers Ensemble (NB, SVM)	64%
[28]	Google Play	64,294	Three	User comments	SVM DT NB	89% using DT
[29]	Android platform	1,148,032	Two	User comments	Sentiment analysis and topic modelling	Opinion mining
[14]	Google Play Store	5,000,000	Six	User comments	Opinion mining summarization	Precision 93.7%
[30]	N/A	500	Five	User comments	ML classifiers	88.8%
[31,32]	Google and Apple Play Stores	32,210	N/A	User comments and rating	Sentiment analysis and summarization	Opinion mining
[33]	iOS mobile apps	6000	Four	User comments	NB SVM	Precision 70% Recall 86%
[34]	Google Play Store	8000	Three	User comments	Opinion mining summarization	report
[35]	Google Play Store and Apple App Store	7456	Fourteen	User comments	SVM	59%

3. Problem Formulation

Mobile app issues can be considered a problem in the software engineering area. In this paper, the problem can be treated as a natural language processing problem, as we worked with textual data extracted from user-generated comments, as reviews, on mobile apps.

Detecting and classifying mobile app issues can be categorized into five classes: usability, reliability, performance, behavior issues, and security. User textual comment, denoted by (UR), is a subset of the set of collections of user textual comments (CTs) that are extracted from mobile apps; and Y is denoted in the classes.

$$\text{Let CT} = \{UR_1, UR_2, UR_3, \ldots\ \ldots\ UR_n\}, \text{where n is the total number of URs}$$

Based on the predefined classes of the NRFs, let the NRFs = {$Class_1$, $Class_2$, $Class_3$, $Class_4$, and $Class_5$}.

After the annotation process, the dataset (D) consists of URs and NFRs. Each UR is assigned to one of the classes of NRFs, as follows:

$$UR \in NRFs$$

Let D be divided into D_1 and D_2, where D_1 represents the training data for the model and D_2 represents the testing for the model's performance:

$$D = D_1 \text{ U } D_2$$

The extracted features from D are converted to numerical data using the TFIDF technique. which produces a set of vectors denoted by V. Each V is a representation of UR and Y is a representation of e NRFs.

Let $V = \{V_1, V_2, V_3, \ldots .. V_n\}$, where n is the total number of extracted words from D.

The problem is formulated as follows:

Each UR $\rightarrow$ TFIDF $\rightarrow$ X(V) to Y; Y is the label of a multiclass; the mobile apps' issues are categorized into five classes of NRFs. Let X be the function that categorizes V to the appropriate class in Y:

$$\text{UR} \rightarrow \text{TFIDF} \rightarrow X(V) \in Y$$

4. Methods and Materials

The research methodology used to apply the hybrid deep learning model in detecting the non-functional requirements of mobile apps is explained and discussed in the following section. The methodology consisted of several phases to perform the objectives of this study: data collection, data cleaning, data annotation, data pre-processing, data representation, building the model, and the model's performance evaluation, as presented in Figure 1. All the phases are described in detail in the following subsections.

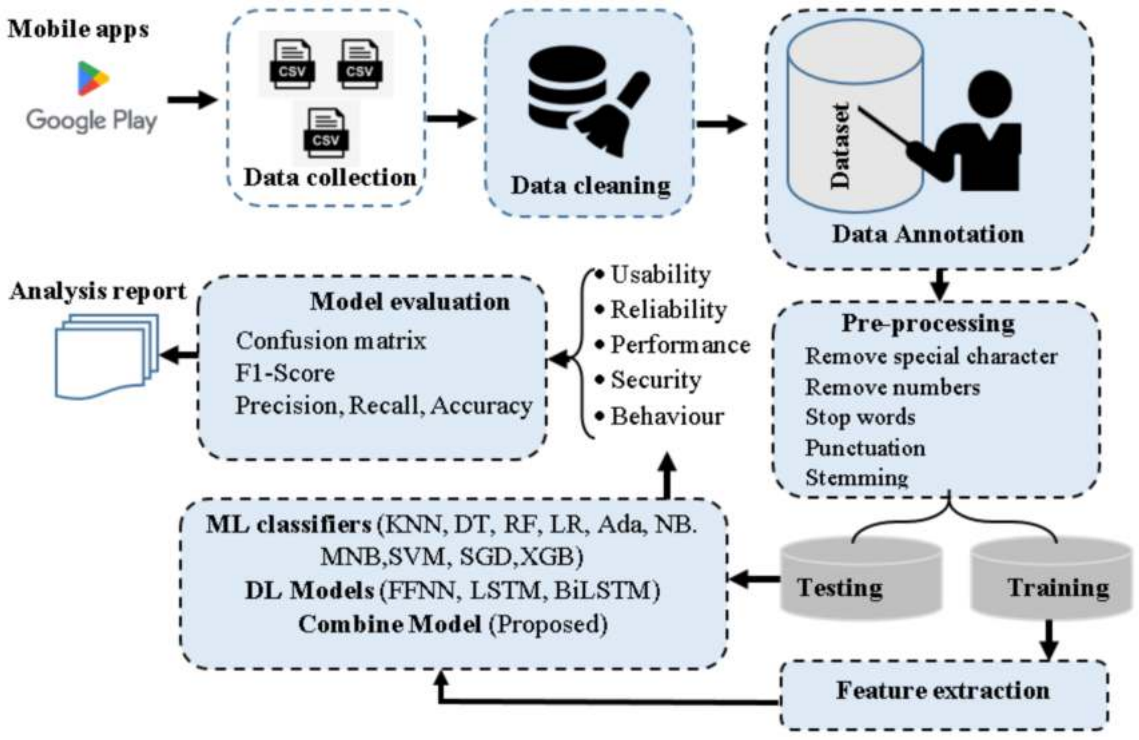


Figure 1. Research methods.

4.1. Data Collection

The data in this phase were collected from user-generated textual comments extracted from the Google Play Store. For this step, the Google API was utilized through Python programming language. A total of 10 home delivery applications were selected based on the most popular apps that are commonly used in the Arabic area and the high-rated apps that contained many user comments. Comments and meta-data were extracted from each mobile app into one comma-separated value (CSV) format from Google Play Store. The created file contained the following meta-data description, as presented in Table 2.

Table 2. Mobile apps meta-data on Google.

Item	Description
reviewId	The review identification number of that comment
userName	The user's name
userImage	The user's image in his/her profile
content	The user's comments
score	The rating score of mobile apps
thumbsUpCount	The number of users who give thumbs up
reviewCreatedVersion	The version of the review tool
at	The time of comment created by the user
replyContent	If there is a reply for that comment from other users
repliedAt	The reply time

In this step, all the items were removed except the content item, which included the textual data generated from a user comment. The total number of comments in all mobile delivery apps was 77, 000. These comments were considered the input for the next phase.

4.2. Data Cleaning

Several steps were carried out to prepare data for the next phases. To complete this task, some columns, duplicate data, data noise, and Arabic sentences written in English (while their meaning was in Arabic languages) were removed. In addition, the five classes were identified based on non-functional software issues. These classes are shown in Table 3 and the class description is presented in Table 4.

Table 3. Mobile apps issues—classes.

Class Name	Class Number
Usability	1
Reliability	2
Performance	3
Behavior issues	4
Security	5

Table 4. Mobile apps issues—categories description.

Class Name	Description
Usability	How easily do the functions and the graphical user interface of the mobile app operate with the user?
Reliability	Measures the consistency of the mobile app functions and function failures.
Performance	Relates to issues for slow applications or function hanging.
Behavior issues	Issues that are related to users, such as respect or lateness in service delivery.
Security	Measures how the system is secured.

The total number of user comments for all applications was reduced to 9200 rows. All files were combined in one single file to use for the data annotation phases.

4.3. Data Annotation

Three Arabic native speakers and experts with Ph. D.s in computer science and software engineering assisted in the annotation process, assigning the user comments to non-functional requirement classes as previously defined. After the annotation process was completed, the verification task for classes was performed using Cohen's Kappa measure. It showed strong agreement between the three annotators. The final dataset is shown in Table 5. In addition, examples of each class of NFRs of mobile apps issues are presented in Table 6.

Table 5. Non-functional classes.

Classes	Class	Count
Usability	1	2490
Reliability	2	1762
Performance	3	810
Behavior issues	4	1611
Security	5	1341
	Total	8014

Table 6. Examples of user comments and classes of NFRs.

Classes	Comment	In English
Usability	البرنامج يحتاج يتطور و الواجهات معقدة و صعبة على المستخدم ، و يرجى تسهيل الواجهة	The program needs to be developed and the interfaces are complex and difficult for the user. Please make the interface easier.
Reliability	احاول اشغل البرنامج لي نص ساعه ما يشتغل	I'm trying to run the program for half an hour, but it won't work.
Performance	التطبيق يتأخر دائمًا ويعلقوبطى فى الاستجابة لتنقيذ اى امر	The application is always lagging, hanging and slow to respond to the command.
Behavior Issues	برنامج سيء جدا جدا مافي احترام للناس اطلب طلب اكثر من ساعة ونص مايوصل والبرنامج مافية خدمة عملاء ترد عليك معاملة سيئة جدا	A very, very bad program. There is no respect for people. Ask for an order for more than an hour and a half, and it does not arrive. The program does not have customer service that responds to you with very bad treatment.
Security	احتيال و نصابين و سرقة من البطاقة	Fraud and theft of the card when purchasing.

4.4. Data Pre-Processing

The dataset consisted of unstructured textual data that contained noise that could affect the model's performance. Therefore, data pre-processing was used to enhance the accuracy of the model's performance, to classify the classes correctly. The most common NLP methods utilized in this process were the removal of stop words, the repetition of the same character in a sentence, the removal of special characters, the removal of numbers, and the removal of punctuation. These items were considered unnecessary data that could reduce the model's performance.

4.5. Data Representation

In this phase, feature-extraction methods were used to convert items into numerical data using TFIDF word representation from the Sklearn package's TfidfVectorizer Library. TFIDF is a statistical method used to show the significance of words in textual data; scholarly research has proved that it is significantly important in many text-mining areas. The weight of TFIDF was used as input for the ML classifiers or the DL learning models. The mathematical formula of TFIDF is as follows):

$$W_{a,b} \ = \ tf_{a,b}\ X\ log(\frac{N}{df_a}) \tag{1}$$

where $W_{a,b}$ is the weight of TFIDF for word (a) in user comment (b), $tf_{a,b}$ is the occurrence of word (a) in user comment (b), and df_a is the total number of user comments containing word (a).

4.6. Machine and Deep Learning Models

This phase utilizes several machine learning and deep learning classifiers. These models were used to obtain the highest accuracy in detecting and classifying non-functional issues in mobile apps. Three types of models were implemented through several experiments: ML classifiers, the DL model, and the proposed model.

The machine learning classifiers were logistic regression (LR), decision tree (DTs), naive Bayes (NB), multinomial naive Bayes (MNB), support vector machines (SVMs), AdaBoost, random forest (RF), K-nearest neighbors (KNNs), stochastic gradient descent (SGD), and ensemble classifiers such as AdaBoost, random forest (RF), and XGBoost (XGB).

Recently, deep learning models have been widely used, due to their accuracy in the classification task. Therefore, in this study, three DL architectures were used: feed-forward neural network (FFNN) (known as fully connected layers), long short-term memory (LSTM), and bidirectional long short-term memory (Bi-LSTM). FFNN consists of three layers: input, hidden, and output layers, as illustrated in Figure 2.

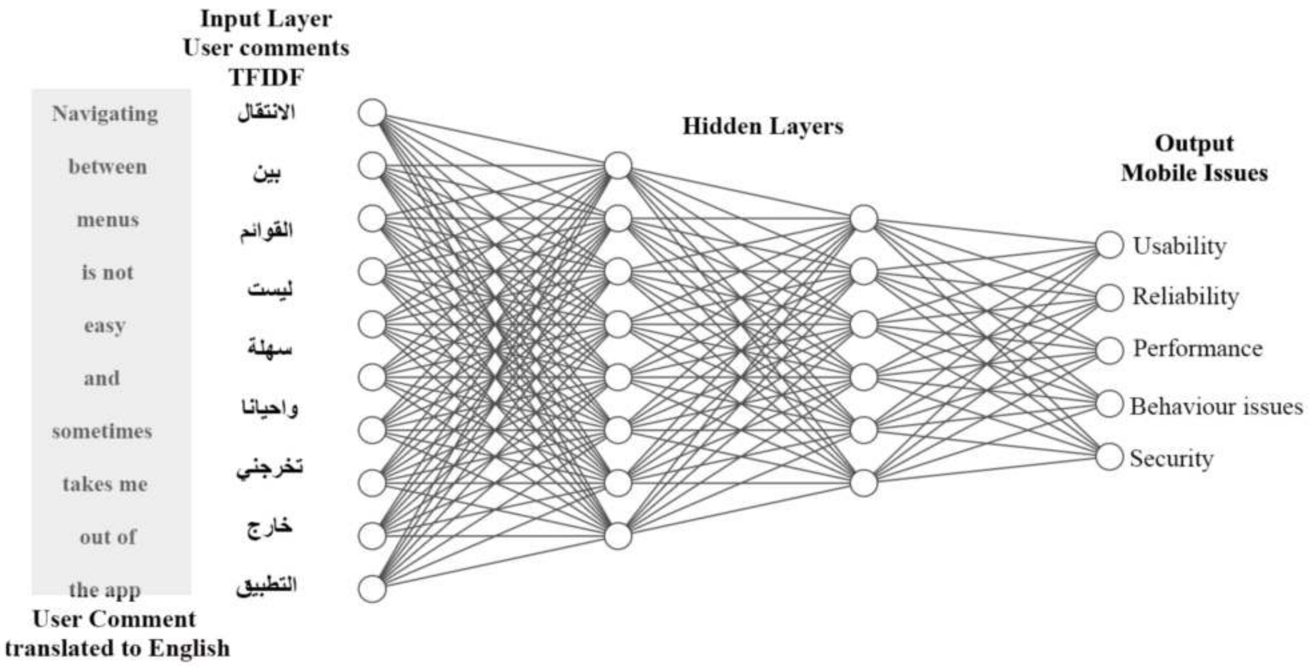


Figure 2. Feed-forward neural network.

The LSTM model was developed as a type of RNN architecture that deals with the time-sequence feature model. The RNN architecture learns automatically, based on the feature's dependencies, unlike the ANN architecture. The main issue with the RNN architecture is the vanishing gradient issue, which makes it difficult to train. The LSTM architecture is used to overcome this issue. Generally, the RNN architecture and the LSTM architecture are commonly used successfully in scholarly natural-language-processing applications. As a result, it was used to detect the issues of mobile apps based on user-generated comments that were considered as a sequence of words. The LSTM model architecture is shown in Figure 3. In addition, the Bi-LSTM is an advanced model of LSTM used for additional training in the NLP area, which allows for training in both backward and forward directions.. Therefore, this model was trained and tested in several experiments.

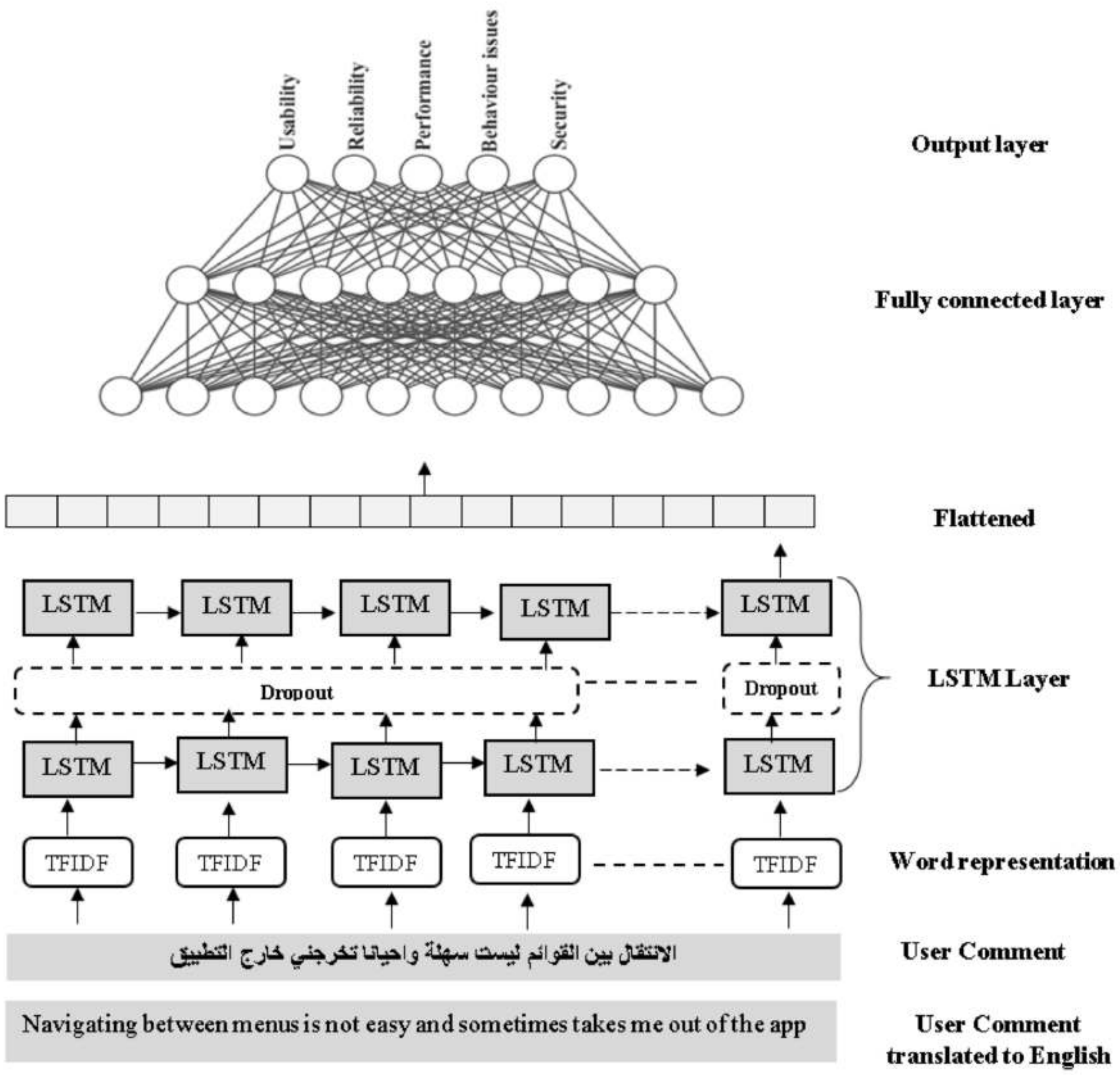

Usability
Reliability
Performance
Behaviour issues
Security
Output layer
Fully connected layer
Flattened
LSTM
Dropout
LSTM Layer
TFIDF
Word representation
الانتقال بين القوائم ليست سهلة واحيانا تخرجني خارج التطبيق
User Comment
Navigating between menus is not easy and sometimes takes me out of the app
User Comment translated to English

Figure 3. LSTM model architecture.

A hybrid deep learning model was proposed in this study to improve the precision and accuracy in detecting and classifying non-functional software requirements issues. In the experimenalt observations, the best accuracy level was recorded using the LSTM model. The proposed model consisted of three models: an RNN model and two LSTM models. Each of these three models received input in the form of TFIDF, which was calculated based on the word/term frequency. In the end, in these three models, the features were concatenated together; then, the output was fed into the fully connected dense layer using a fattening layer to transform the features into single-dimension input, while the output layer used the softmax function that provided the prediction to one of the five mobile apps issue categorizations. The softmax function was calculated as shown in mathematical Formula (2), while all the hidden layers of the LSTM and RNN models used a rectified linear unit (ReLU) to ensure that the models trained effectively from the features and avoided the overfitting issue by adding a regularization and dropout. The general architecture of the proposed hybrid deep learning model is illustrated in Figure 4.

$$Softmax\ \sigma\ (Z)_j = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_i}} \tag{2}$$

where Z is the input vector/list of the user comments in numerical values, Z_j represents the values of the input vector, e^{z_i} is the exponential function to provide a positive value, and K is the number of classes.

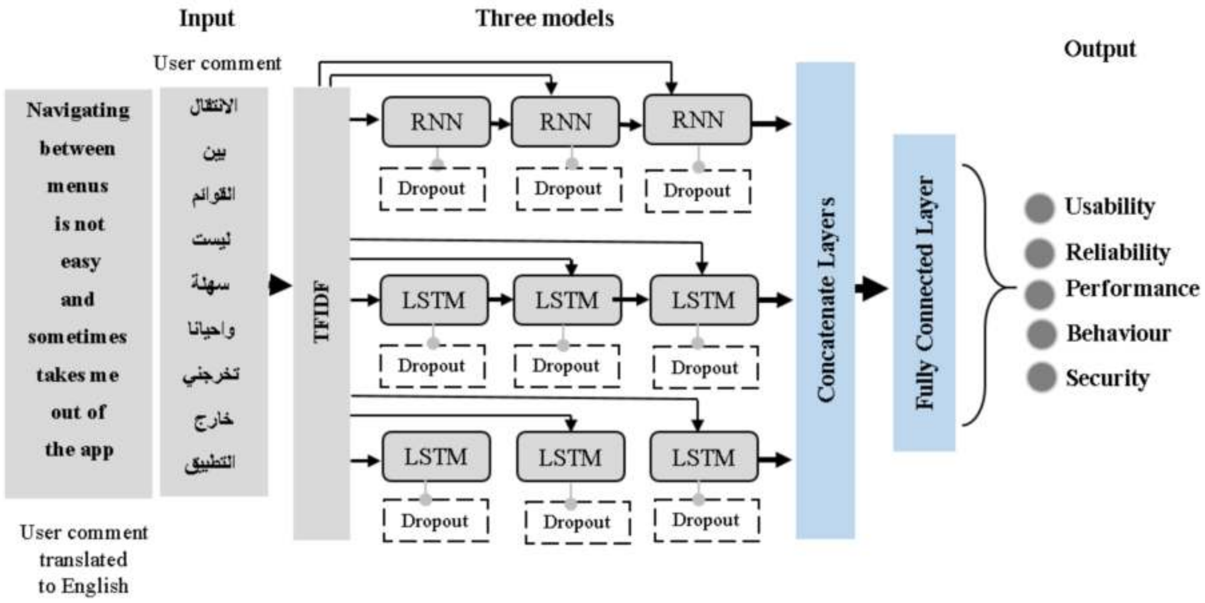

Input
User comment
Three models
Output
Navigating between menus is not easy and sometimes takes me out of the app
الانتقال بين القوائم ليست سهلة واحيانا تخرجني خارج التطبيق
User comment translated to English
TFIDF
RNN
LSTM
Dropout
Concatenate Layers
Fully Connected Layer
Usability
Reliability
Performance
Behaviour
Security

Figure 4. Proposed hybrid deep learning model.

Moreover, the ar-corrector technique was used to correct the user review comments. This way, the model performance was enhanced to detect mobile apps issues and classify them into the appropriate predefined class.

4.7. Model Evaluation

This section provides details about how the model performance was evaluated. All the results of the ML and DL experiments were analyzed and evaluated based on the most common measures. These measures were precision (3), recall (4), accuracy (5), and F1 score (6), through the confusion matrix. The following tables show the mathematical formula for the mentioned measures.

$$Precision = \frac{True\ positive}{True\ Positive + False\ Positive} \tag{3}$$

$$Recall = \frac{True\ Positive}{True\ Positive + Flase\ Negative} \tag{4}$$

$$Accuracy = \frac{True\ positive + True\ Negative}{True\ Positive + True\ Negative + False\ Positive + False\ Negative} \tag{5}$$

$$F1\text{-}score = \frac{2 * (Precision * Recall)}{Precision + Recall} \tag{6}$$

where true positive is when the model predicts the correct user comments in the correct class; false positive is when the model predicts that the user comment is correct while it is negative; true negative is when the model predicts that the user comment is negative and it is negative; false negative is when the model predicts that the user comment is incorrect while it is correct.

5. Results and Discussion

This section presents the details of the experimental environment and the results of the three types of experiments. To evaluate the proposed model's ability to detect and classify the issues of mobile apps based on the introduced two datasets, several experiments were carried out. In these experiments, the traditional machine learning classifiers and the deep learning methods, as well as the proposed combined model, were utilized.

5.1. Experimental Environment

All the experiments in both machine and deep learning were executed using Python Keras, which is a high-level deep learning programming language, with the backend TensorFlow framework using Google COLAB. For all experiments, the experimental GPU was utilized to train the models. During the process of the experiments, the models were trained in both datasets by using 70% and 30% for training and testing data, respectively. Two datasets were used in all experiments: the first dataset contained five classes, as shown in Table 3; the second dataset contained only four classes, as shown in Table 7.

Table 7. Second dataset.

Classes	Class	Count
Usability	1	4252
Performance	3	810
Behavior issues	4	1611
Security	5	1341
	Total	8014

Both datasets were considered imbalanced datasets. The data augmentation techniques were exploited to increase the dataset, which led to improving the model's performance. In order to use the data augmentation techniques, the library called "textattack" was utilized, particularly "EasyDataAugmenter". In this way, the number of user comments were increased by using a swap-per-augmented technique for the words by 10%. As result, the model's performance was assessed using F1 scores of a confusion matrix, in addition to precision, recall, and accuracy.

5.2. Classical Machine Learning Classifiers

This section explains the results obtained through the execution of several experiments to examine the best model performance in terms of accuracy, using the ML classifiers. These classifiers used were LR, DT, NB, MNB, SVM, RF, KNN, and SGD, as well as ensemble classifiers such as Ada, RF, and XGB. The experimental results showed that the highest F1 score was recorded using RF with dataset 2, which reached 94.37%. This followed the data augmentation (WDA) technique, which outperformed all the other classifiers. The same classifier recorded 94.26% on dataset 1, with average precision and recall of 95% for all four classes, followed by the SVC classifier, which recorded 94.26% in the F1 score.

The results of the experiments on dataset 2 using the WDA technique showed that the best F1 score recorded was 74.93%, when using LR, followed by the SGD classifier, which slightly decreased in F1 score, reaching 74.55% Using dataset 1 with WDA, the best model performance recorded was 73.85% using LR, followed by SVC, which recorded 73.60%. Overall, the lowest F1 score, 38.84%, was recorded using KNN with the data augmentation technique.

Several experiments were conducted using dataset 2. In these experiments, the results showed that the highest F1 score was measured using RF and SVC, reaching 94% using the data augmentation. Repeatedly, the lowest F1 score achieved using the three classifiers, DT, KNN, and NB, were 38% and 49% without augmentation and with augmentation, respectively. In addition, the average precision and recall for the three classifiers were recorded between 42% to 71%. Table 8 provides the results of all ML classifiers using both datasets.

Table 8. F1-Score using ML classifiers.

Model	Dataset 1		Dataset 2	
	WODA	**WDA**	**WODA**	**WDA**
Ada	62.41%	64.43%	68.98%	69.64%
DT	49.77%	49.08%	49.56%	49.54%
GB	68.90%	74.07%	70.44%	71.85%
KNN	38.84%	70.68%	44.99%	68.28%
LR	73.85%	81.99%	74.93%	82.91%
MNB	72.22%	76.29%	74.35%	65.84%
NB	45.90%	57.22%	54.18%	65.84%
RF	71.14%	94.26%	72.18%	94.37%
SGD	73.22%	82.13%	74.55%	83.02%
SVC	73.60%	93.50%	74.72%	94.26%
SVM	73.22%	83.68%	74.05%	83.83%
XGB	71.85%	85.71%	73.14%	86.22%

WODA: without data augmentation; WDA: with data augmentation.

In the data augmentation technique, the model's performance in terms of the F1 score measure was improved, as shown in Figure 5, using dataset 1 and dataset 2. Therefore, the highest improvement in the model's performance in terms of the F1 score was recorded using KNN in both datasets, which improved in percentage by 45% and 34% using datasets 1 and 2, respectively, followed by RF, with improvement between 24% and 25% when using both datasets, respectively. The confusion matrix demonstrates the highest and lowest values of the F1 score with/without data augmentation, as shown in Figures 6 and 7 for dataset 1 and Figures 8 and 9 for dataset 2.

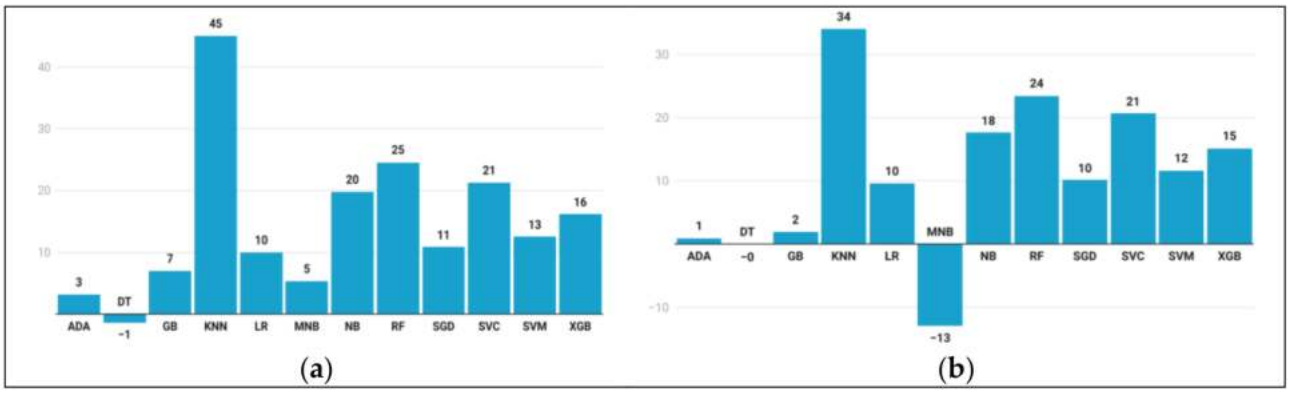
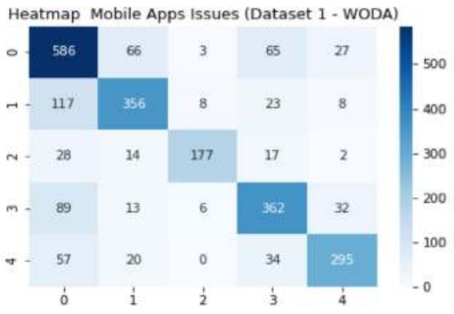


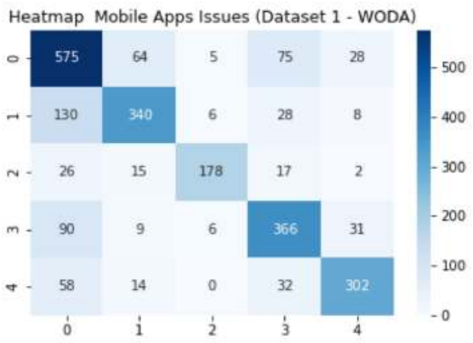
Figure 5. F1 score improvement: (**a**) dataset 1; (**b**) dataset 2.

5.3. Deep Learning Approach

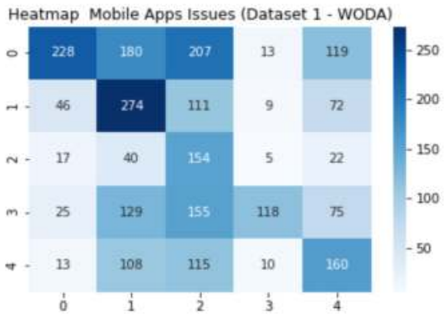
The DL models achieved the highest performance level according to the existing studies. In the second approach, several experiments were carried out to obtain the best F1 score using DL models with various hyper-parameters. The most common models used in NLP applications were ANN, LSTM, and Bi-LSTM. The three models were applied using both datasets 1 and 2. In these experiments, the TFIDF was utilized as an input for the DL models. Sequential architecture was implemented in all experiments using Keras. Table 9 presents the optimal hyper-parameters utilized for the three models after several attempts in the experiment to obtain the best model performance.



(a)

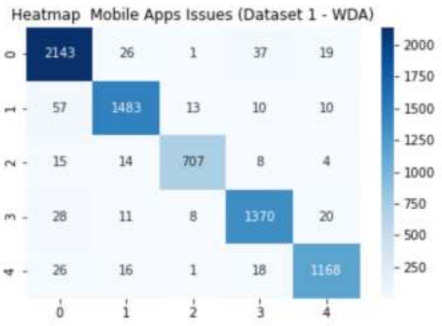


(b)

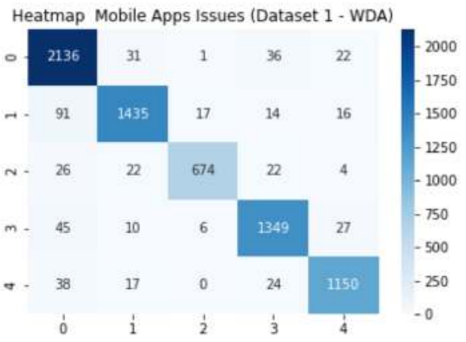


(c)

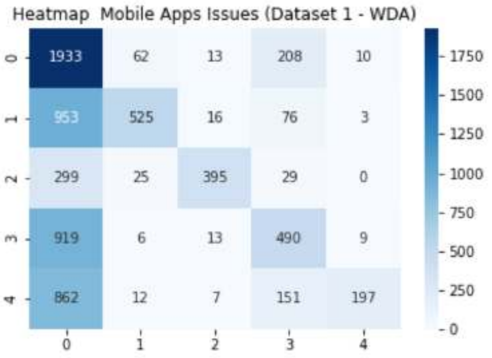
Figure 6. Confusion matrix for ML classifiers using dataset 1 without data augmentation: (**a**) LR, (**b**) SVC, (**c**) KNN.



(a)



(b)



(c)

Figure 7. Confusion matrix for ML classifiers suing dataset 1 with data augmentation: (**a**) RF, (**b**) SVC, (**c**) DT.

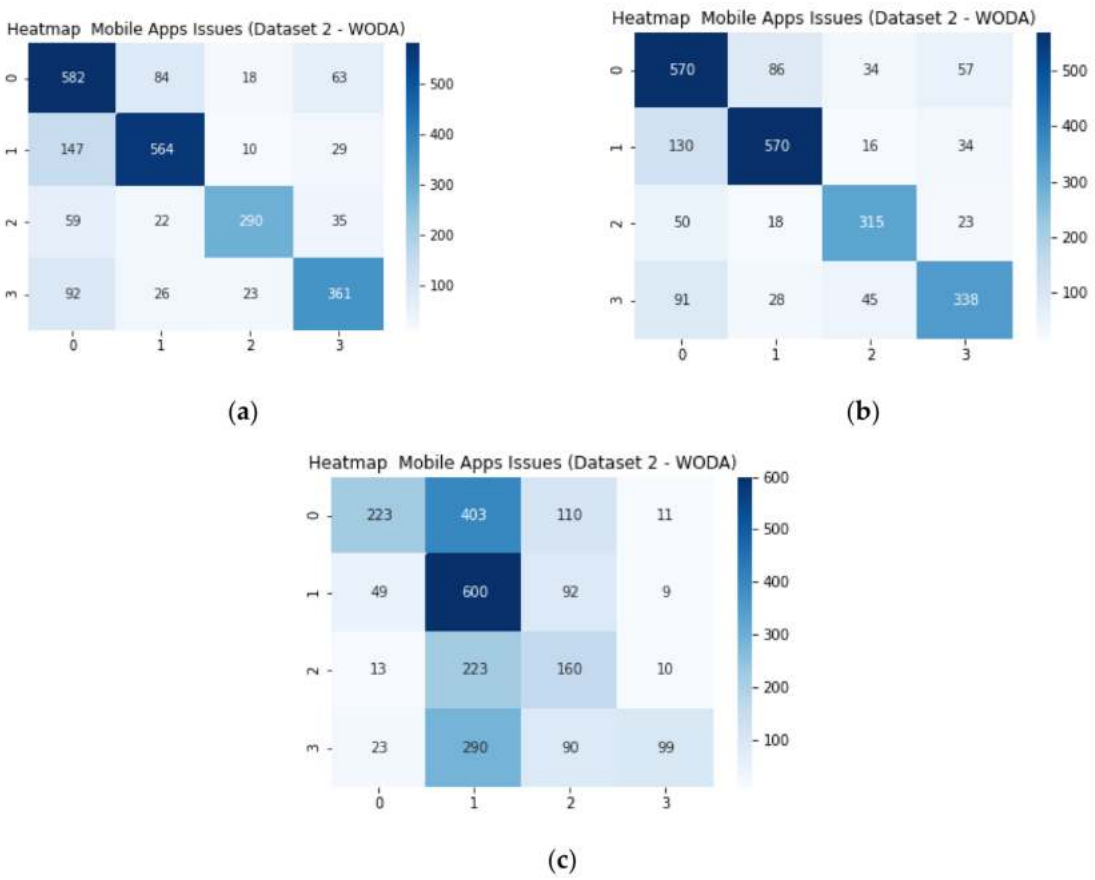


Figure 8. Confusion matrix for ML classifiers suing dataset 2 without data augmentation: (**a**) SVC, (**b**) SGD, (**c**) KNN.

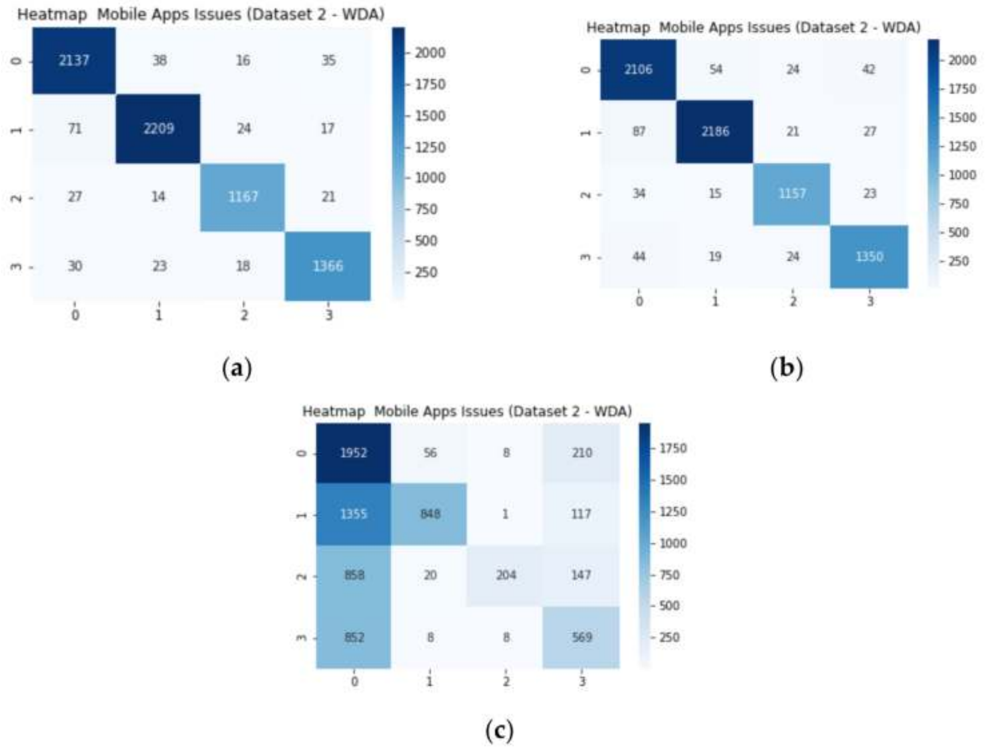


Figure 9. Confusion matrix for ML classifiers using dataset 2 with data augmentation: (**a**) RF, (**b**) SVC, (**c**) DT.

Table 9. Parameter used in ANN approach using both datasets.

Parameters	ANN	LSTM	BiLSTM
Batch-size	128	32	64
Hidden layer-activation function	Relu	Relu	Relu
Output layer-activation function	Softmax	Softmax	Softmax
Dropout	0.5	0.5	0.5
Number of epochs	100	100	100
Loss-function	sparse_categorical_crossentropy		
Optimizer	Adam	Adam	Adam
Regularization	L2 (0.001)	L2 (0.001)	L2 (0.001)

Table 9 demonstrates the optimal hyper-parameters that were used in the experiment to obtain the best model performance. In the ANN experiment, the F1 score reached 80% and 88% using dataset 1 WDA and WODA, respectively, while in dataset 2, the F1 score reached 78% and 89% using WDA and WODA, respectively, in detecting mobile non-functional issues and classifying according to relatives' class. Figure 6 indicates the model performance according to the F1 score using both datasets. The overfitting problem was raised using datasets 1 and 2, as shown in Figures 10a and 11a; when the data augmentation technique was applied, the overfitting issues were solved, as shown in Figures 10b and 11b.

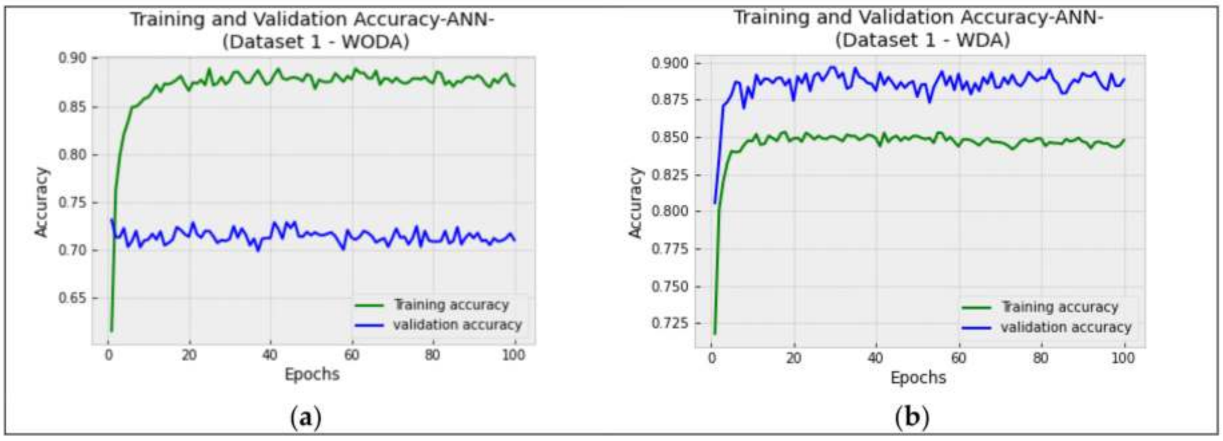


Figure 10. ANN model validation and testing accuracy and loss without data augmentation (dataset 1): (**a**) model accuracy (WODA); (**b**) model accuracy (WDA).

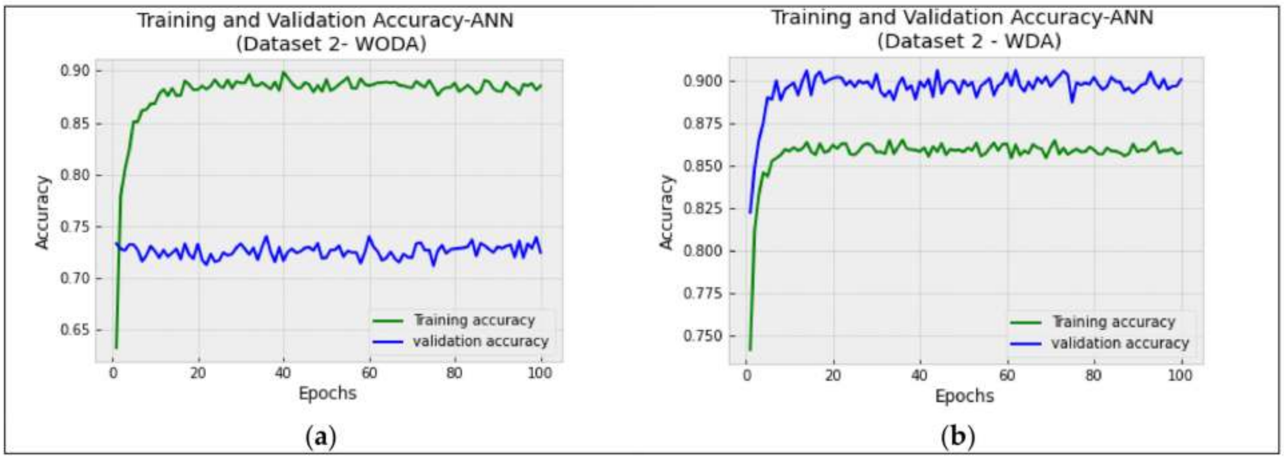


Figure 11. ANN model validation and testing accuracy and loss without data augmentation (dataset 2): (**a**) model accuracy (WODA); (**b**) model accuracy (WDA).

Figure 10b shows us that the model had much better accuracy with data augmentation, which reached a peak of 89% in validation, while in the model-training process it reached 85%. Similarly, it was significantly noticeable, as shown in Figure 11b, that when using dataset 2, the model improved when the applied data augmentation F score reached 86% and 90% in model training and validation, respectively.

In the LSTM experiment, the best accuracy level was recorded at 93% using both datasets with DA technique 1, while the lowest accuracy rate was recorded using LSTM, decreasing to 64% and 66% using dataset 1 and dataset 2, respectively. The figure shows the testing and validation accuracy and the loss for ANN experiments using dataset 1 and dataset 2. In the model overfitting, the difference between the validation and training was around 20%. These were WODA techniques in both datasets, as presented in Figures 12a and 13a, using dataset 1 and dataset 2, respectively. When using WDA, the model's performance in training and validation improved, and no overfitting issue was identified with the model, as shown in Figures 12b and 13b for dataset 1 and dataset 2, respectively.

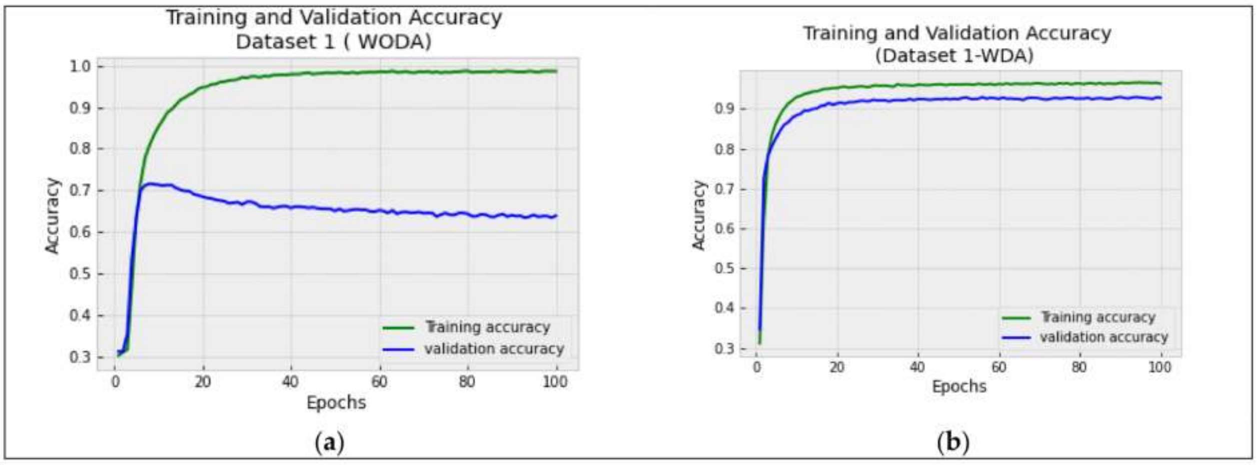


Figure 12. LSTM model validation and testing accuracy and loss dataset 1: (**a**) model accuracy (WODA); (**b**) model accuracy (WDA).

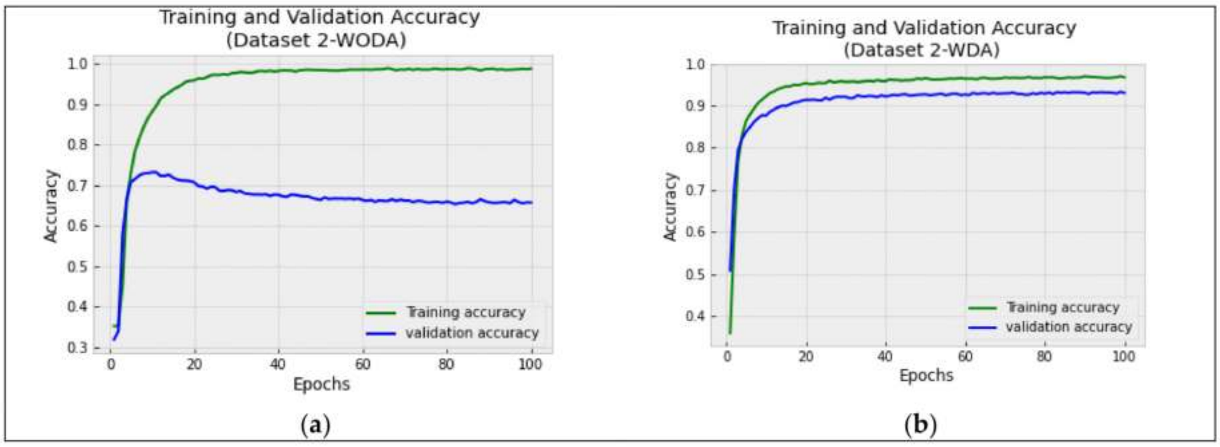


Figure 13. LSTM model validation and testing accuracy and lose (dataset 2): (**a**) model accuracy (WODA); (**b**) model accuracy (WDA).

Figures 12b and 13b show correlation with the statement made above that LSTM scored the highest in terms of accuracy with data augmentation in both datasets 1 and 2. The above graph verifies this information. The Bi-LSTM experiments scored the lowest rate out of all the experiments tested, at 66% to 68%, as presented in Figures 14b and 15b, with a overfitting issue. However, a drastic change was seen when data augmentation was applied; this number skyrocketed to 91% in both datasets, as shown in Figures 14b and 15b. In these figures the green line indicates the training accuracy while the blue line indicates the validation accuracy.

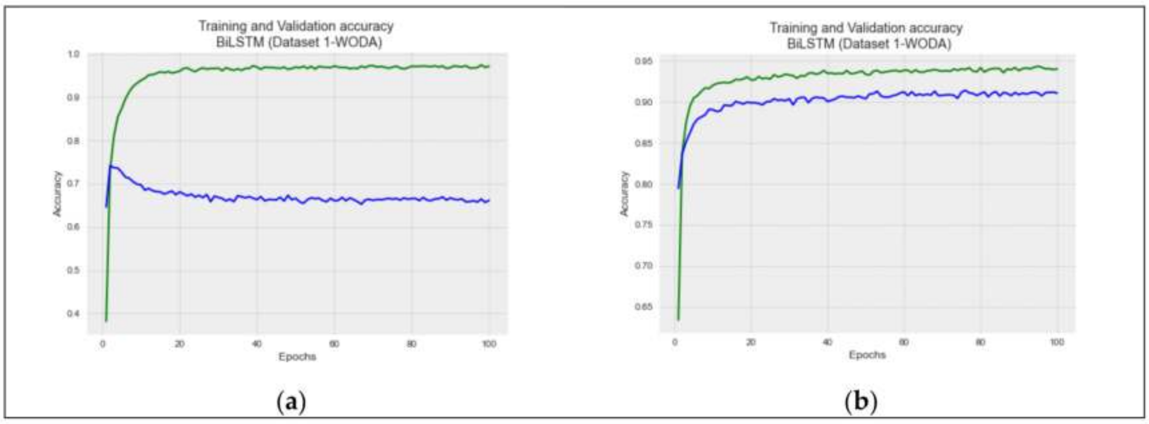


Figure 14. Bi-LSTM model validation and testing accuracy and loss (dataset 1): (**a**) model accuracy (WODA); (**b**) model accuracy (WDA).

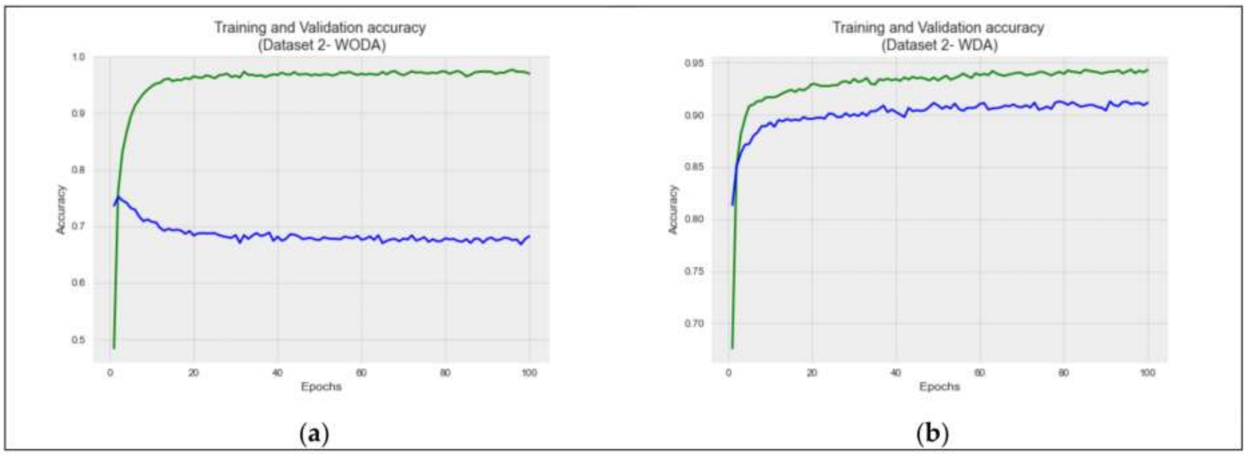


Figure 15. Bi-LSTM model validation and testing accuracy and loss using dataset 2: (**a**) model accuracy (WODA); (**b**) model accuracy (WDA).

From the given information, we concluded that the Bi-LSTM model struggled immensely with data augmentation, scoring only 66% to 68% in the given graph; however, it performed much better with data augmentation, reaching higher than expected accuracy of 91% in both datasets.

The second dataset further proved the initial point and had a light boost in its accuracy without data augmentation, from 66% to 68% in dataset 2.

5.4. The Proposed Hybrid Deep Learning Model

This section explains the results of the experiment based on the proposed model. The proposed model was based on three RNN architectures, which consisted of three combined models: a simple RNN model and two LSTM models. The input for the three models was the same. These three models were concatenated to provide the outputs to the dense layer. The experiment settings are shown in Table 10 for the three models. Several experiments carried out were categorized into two types of experiments using dataset 1 and 2. All these experiments were based on function API architecture. The "Adam optimizer" and the loss function "categorical_crossentropy" were utilized. The input shape was 2, a 500 feature extracted using TDIDF.

Table 10. Hyper-parameter used to evaluate the proposed hybrid deep learning model.

Parameters	Simple RNN	LSTM 1	LSTM 2
Hidden layer-activation function	Relu	relu	relu
Number of neurons	128	64	32
Output layer-activation function	Softmax	softmax	softmax
Dropout	0.5	0.5	0.5
Regularization	l2 (0.0001)	l2 (0.0001)	l2 (0.0001)

The proposed hybrid deep learning model was applied to both datasets, and the model's performance achieved the highest F score of 96% using dataset 1 WDA; it also achieved 95% using dataset 2 with WDA, as shown in Figures 16b and 17b. The loss function for training and validation decreased, as shown in Figures 16d and 17d. The confusion matrix for the experimental results in both datasets is illustrated in Figures 16c and 17c.

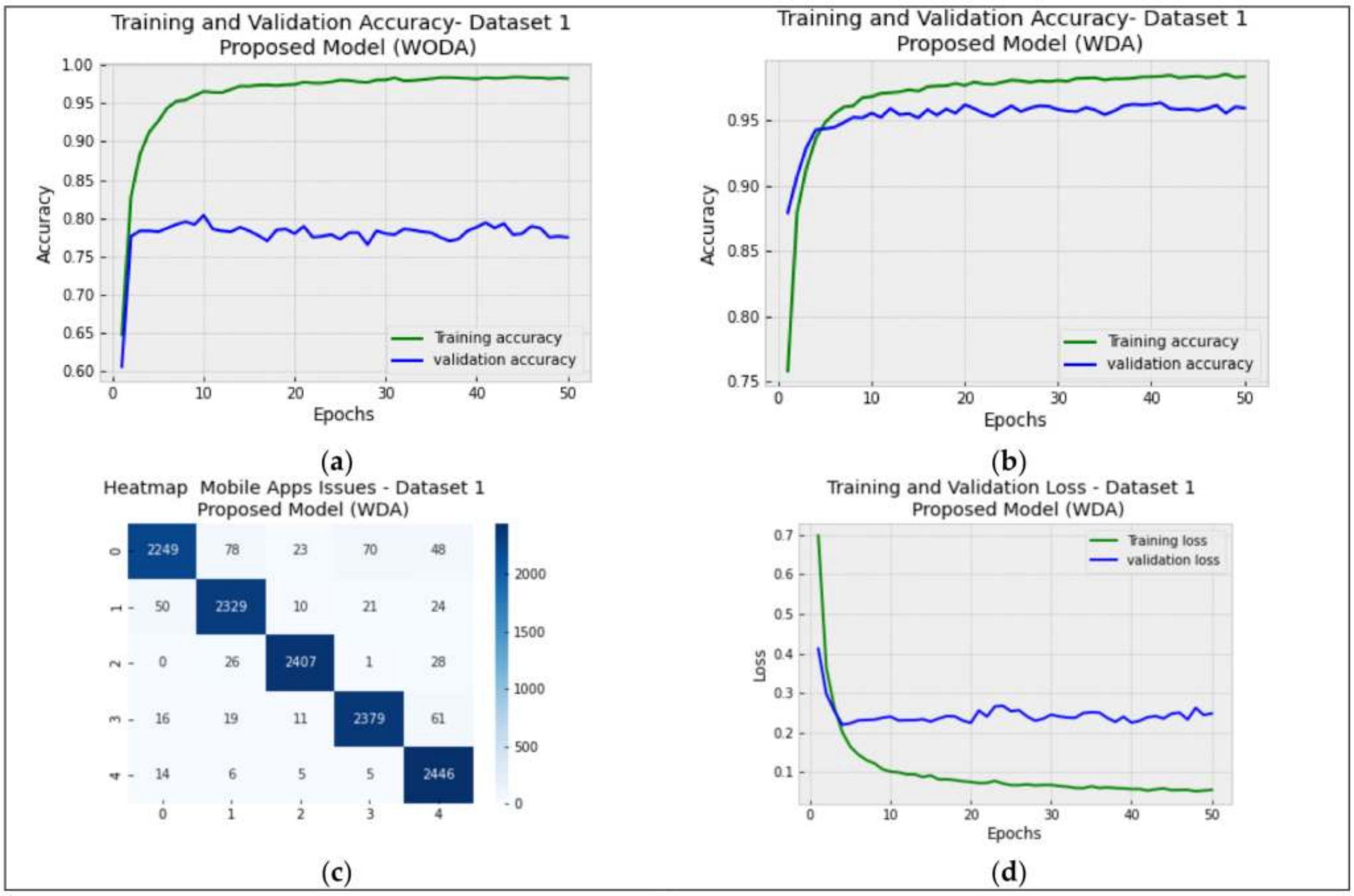


Figure 16. Proposed model validation and testing accuracy and loss using dataset 1: (**a**) model accuracy using dataset 1—WODA; (**b**) model accuracy using dataset 1—WDA; (**c**) confusion matrix (WDA); (**d**) training and validation loss.

All the aforementioned tests showed that the best scores were reached in both datasets; however, slightly worse scores resulted in the second dataset. We can confidently say that this study proved that the proposed model was the best and scored the highest percentages in all aspects. It can detect and classify the mobile app issues properly into the aforementioned five classes. To elaborate further, the comparison between the experiment results of all the models—the MLCs, the DLs, and the proposed model—are presented in Table 11.

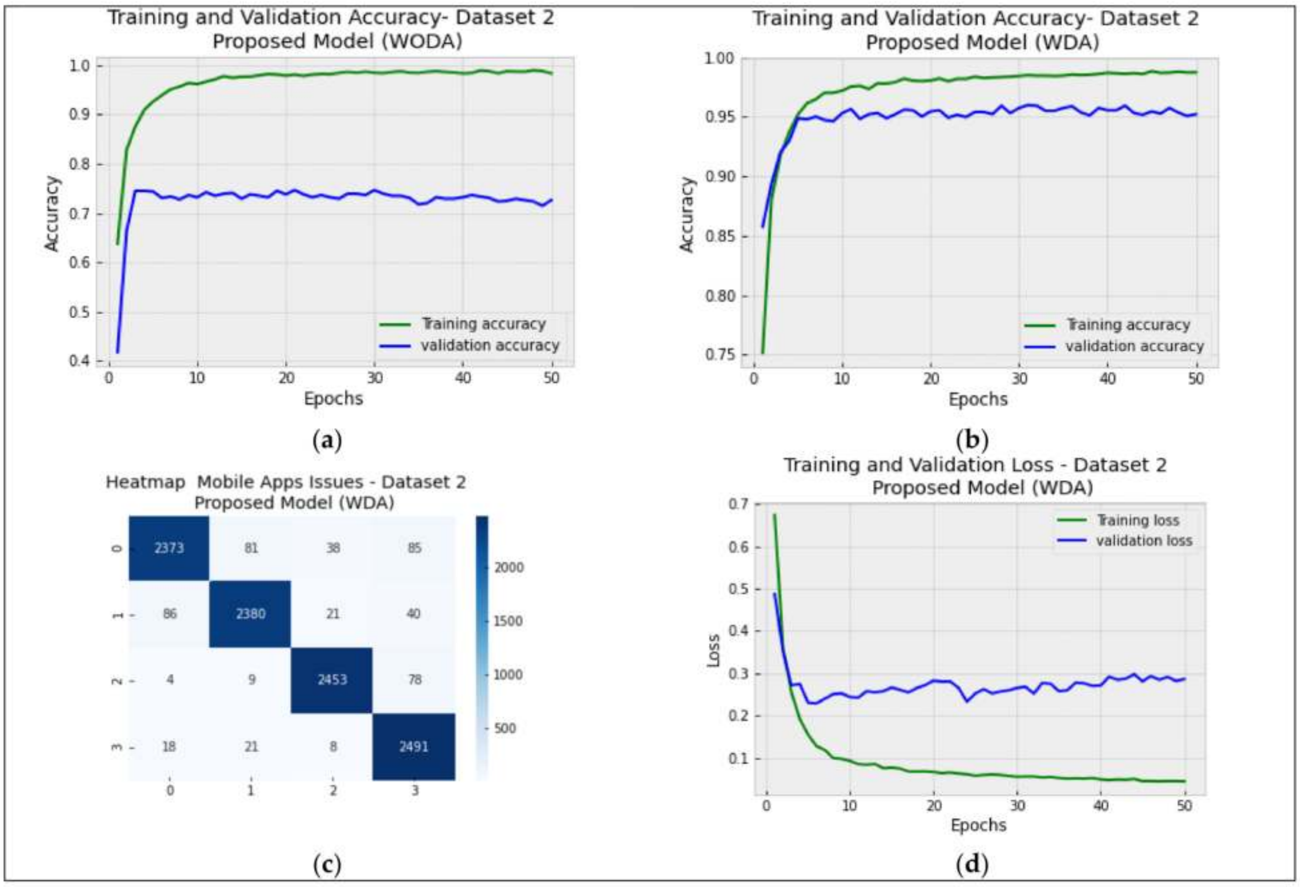


Figure 17. Proposed model validation and testing accuracy and loss using dataset 2: (**a**) model accuracy using dataset 2—WODA; (**b**) model accuracy using dataset 2—WDA; (**c**) confusion matrix (WDA); (**d**) training and validation loss.

Table 11. Comparison between model performance for MLCs, DLs, and the proposed model.

Approaches	**Dataset 1**		**Dataset 2**	
	WODA	**WDA**	**WODA**	**WDA**
RF	71.14%	94.26%	72.18%	94.37%
SVC	73.60%	93.50%	74.72%	94.26%
ANN	71%	89%	73%	90%
LSTM	64%	93%	66%	93%
Bi-LSTM	66%	91%	68%	91%
Proposed model	80%	96%	77%	95%

Table 11 presents a comparison of the results of the best MLCs, DLs, and the proposed model in terms of accuracy. Based on the experimental results, the proposed model provided the best results in terms of percentage in all aspects; for example, it scored the highest in dataset 1 without data augmentation, at 80%, as well the highest score with data augmentation, at 96%. We can also conclude that the proposed model scored highest in terms of percentage in dataset 2 without data augmentation, at 77%, which was lower by 3% compared with dataset 1; however, it still reached the highest score after the SVC model, which scored 74.72%. We also believe that the proposed model obtained the highest score in dataset 2 with data, reaching 95%, which was only 1% less than its score with data augmentation from dataset 1.

6. Conclusions

This paper proposed a novel model based on combining three deep learning architectures for detecting and classifying mobile apps' non-functional requirement issues. In the proposed model, data augmentation and Arabic sentence correction techniques were applied. The model was trained on five identified main mobile issues, based on the introduced dataset. The model's performance and the experimental results were as evaluated and compared with those of ML classifiers, FFNN architecture, and LSTM models. It outperformed all the aforementioned models. In addition, the data augmentation technique helped in improving accuracy. The proposed model produced a score of 96% with data augmentation, in terms of the F-score. In the future, the proposed model may be applied to multilingual data and the dataset may be increased. Additionally, different methods, such as averaging and merging, may be applied to the model's features to improve its accuracy.

Author Contributions: Conceptualization, W.M.S.Y., A.E.Y. and A.A.-D.; methodology, W.M.S.Y. and A.E.Y.; software, W.M.S.Y., A.G. and A.A.-D.; validation, W.M.S.Y. and A.E.Y.; formal analysis, A.G.; investigation, W.M.S.Y.; resources, W.M.S.Y.; data curation, W.M.S.Y.; writing—original draft preparation, W.M.S.Y.; writing—review and editing, W.M.S.Y.; visualization, A.G.; supervision, W.M.S.Y. and A.E.Y.; project administration, W.M.S.Y.; funding acquisition, A.E.Y. and A.G. All authors have read and agreed to the published version of the manuscript.

Funding: This work is funded by the Deanship of Scientific Research at Northern Border University, Arar, Kingdom of Saudi Arabia under grant number CSCR-2022-11-1786.

Acknowledgments: The authors gratefully acknowledge the approval and the support of this research study by the grant no. CSCR-2022-11-1786 from the Deanship of Scientific Research at Northern Border University, Arar, K.S.A.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Villarroel, L.; Bavota, G.; Russo, B.; Oliveto, R.; Di Penta, M. Release planning of mobile apps based on user reviews. In Proceedings of the 2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE), Austin, TX, USA, 14–22 May 2016; pp. 14–24.
2. Iacob, C.; Harrison, R. Retrieving and analyzing mobile apps feature requests from online reviews. In Proceedings of the 2013 10th Working Conference on Mining Software Repositories (MSR), San Francisco, CA, USA, 18–19 May 2013; pp. 41–44.
3. Caldeira, C.; Chen, Y.; Chan, L.; Pham, V.; Chen, Y.; Zheng, K. Mobile apps for mood tracking: An analysis of features and user reviews. In Proceedings of the AMIA Annual Symposium Proceedings, Washington, DC, USA, 4–8 November 2017; Volume 2017, p. 495.
4. Palomba, F.; Salza, P.; Ciurumelea, A.; Panichella, S.; Gall, H.; Ferrucci, F.; De Lucia, A. Recommending and localizing change requests for mobile apps based on user reviews. In Proceedings of the 2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE), Buenos Aires, Argentina, 20–28 May 2017; pp. 106–117.
5. Mcilroy, S.; Shang, W.; Ali, N.; Hassan, A.E. User reviews of top mobile apps in Apple and Google app stores. *Commun. ACM* **2017**, *60*, 62–67. [CrossRef]
6. Khalid, H.; Shihab, E.; Nagappan, M.; Hassan, A.E. What do mobile app users complain about? *IEEE Softw.* **2014**, *32*, 70–77. [CrossRef]
7. Corbalán, L.; Thomas, P.; Delía, L.; Cáseres, G.; Sosa, J.F.; Tesone, F.; Pesado, P. A study of non-functional requirements in apps for mobile devices. In Proceedings of the Conference on Cloud Computing and Big Data, Honolulu, HI, USA, 29–31 May 2019; pp. 125–136.
8. Jha, N.; Mahmoud, A. Mining non-functional requirements from App store reviews. *Empir. Softw. Eng.* **2019**, *24*, 3659–3695. [CrossRef]
9. Yao, Y.; Jiang, W.; Wang, Y.; Song, P.; Wang, B. Non-Functional Requirements Analysis Based on Application Reviews in the Android App Market. *Inf. Resour. Manag. J.* **2022**, *35*, 1–17. [CrossRef]
10. Lu, M.; Liang, P. Automatic classification of non-functional requirements from augmented app user reviews. In Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering, Karlskrona, Sweden, 15–16 June 2017; pp. 344–353.
11. Ciurumelea, A.; Schaufelbühl, A.; Panichella, S.; Gall, H.C. Analyzing reviews and code of mobile apps for better release planning. In Proceedings of the 2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER), Klagenfurt, Austria, 20–24 February 2017; pp. 91–102.

12. Tao, C.; Guo, H.; Huang, Z. Identifying security issues for mobile applications based on user review summarization. *Inf. Softw. Technol.* **2020**, *122*, 106290. [CrossRef]
13. Wang, W.; Li, Z.; Tian, Z.; Wang, J.; Cheng, M. Extracting and summarizing affective features and responses from online product descriptions and reviews: A Kansei text mining approach. *Eng. Appl. Artif. Intell.* **2018**, *73*, 149–162. [CrossRef]
14. Li, Y.; Jia, B.; Guo, Y.; Chen, X. Mining user reviews for mobile app comparisons. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* **2017**, *1*, 1–15. [CrossRef]
15. Jeong, B.; Yoon, J.; Lee, J.-M. Social media mining for product planning: A product opportunity mining approach based on topic modeling and sentiment analysis. *Int. J. Inf. Manag.* **2019**, *48*, 280–290. [CrossRef]
16. Camacho-Rivera, M.; Vo, H.; Huang, X.; Lau, J.; Lawal, A.; Kawaguchi, A. Evaluating asthma mobile apps to improve asthma self-management: User ratings and sentiment analysis of publicly available apps. *JMIR mHealth and uHealth* **2020**, *8*, e15076. [CrossRef]
17. Valdivia, A.; Luzon, M.V.; Herrera, F. Sentiment Analysis in TripAdvisor. *IEEE Intell. Syst.* **2017**, *32*, 72–77. [CrossRef]
18. Lin, B.; Zampetti, F.; Bavota, G.; Di Penta, M.; Lanza, M.; Oliveto, R. Sentiment analysis for software engineering: How far can we go? In Proceedings of the 40th International Conference on Software Engineering, Gothenburg, Sweden, 27 May–3 June 2018; pp. 94–104.
19. Tang, F.; Fu, L.; Yao, B.; Xu, W. Aspect based fine-grained sentiment analysis for online reviews. *Inf. Sci.* **2019**, *488*, 190–204. [CrossRef]
20. Sankar, H.; Subramaniyaswamy, V.; Vijayakumar, V.; Kumar, S.A.; Logesh, R.; Umamakeswari, A. Intelligent sentiment analysis approach using edge computing-based deep learning technique. *Softw. Pract. Exp.* **2020**, *50*, 645–657. [CrossRef]
21. Nayebi, M.; Cho, H.; Ruhe, G. App store mining is not enough for app improvement. *Empir. Softw. Eng.* **2018**, *23*, 2764–2794. [CrossRef]
22. Mujahid, S.; Sierra, G.; Abdalkareem, R.; Shihab, E.; Shang, W. Examining user complaints of wearable apps: A case study on android wear. In Proceedings of the 2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems (MOBILESoft), Buenos Aires, Argentina, 22–23 May 2017; pp. 96–99.
23. Mardonova, M.; Choi, Y. Review of wearable device technology and its applications to the mining industry. *Energies* **2018**, *11*, 547. [CrossRef]
24. Mujahid, S. *Determining and Detecting Permission Iassues of Wearable Apps*; Concordia University: Montréal, QC, Canada, 2018.
25. Aslam, N.; Ramay, W.Y.; Xia, K.; Sarwar, N. Convolutional neural network based classification of app reviews. *IEEE Access* **2020**, *8*, 185619–185628. [CrossRef]
26. Rustam, F.; Mehmood, A.; Ahmad, M.; Ullah, S.; Khan, D.M.; Choi, G.S. Classification of shopify app user reviews using novel multi text features. *IEEE Access* **2020**, *8*, 30234–30244. [CrossRef]
27. Guzman, E.; El-Haliby, M.; Bruegge, B. Ensemble methods for app review classification: An approach for software evolution (n). In Proceedings of the 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE), Lincoln, NE, USA, 9–11 November 2015; pp. 771–776.
28. Isa, S.M.; Suwandi, R.; Andrean, Y.P. Optimizing the hyperparameter of feature extraction and machine learning classification algorithms. *Int. J. Adv. Comput. Sci. Appl.* **2019**, *10*, 69–76. [CrossRef]
29. Li, X.; Zhang, Z.; Stefanidis, K. Sentiment-Aware analysis of mobile apps user reviews regarding particular updates. *ICSEA* **2018**, *2018*, 109.
30. Panichella, S.; Di Sorbo, A.; Guzman, E.; Visaggio, C.A.; Canfora, G.; Gall, H.C. Ardoc: App reviews development oriented classifier. In Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering, Seattle, WA, USA, 13–18 November 2016; pp. 1023–1027.
31. Luiz, W.; Viegas, F.; Alencar, R.; Mourão, F.; Salles, T.; Carvalho, D.; Gonçalves, M.A.; Rocha, L. A feature-oriented sentiment rating for mobile app reviews. In Proceedings of the 2018 World Wide Web Conference, Lyon, France, 23–27 April 2018; pp. 1909–1918.
32. Guzman, E.; Maalej, W. How do users like this feature? A fine grained sentiment analysis of app reviews. In Proceedings of the 2014 IEEE 22nd International Requirements Engineering Conference (RE), Karlskrona, Sweden, 25–29 August 2014; pp. 153–162.
33. Malik, H.; Shakshuki, E.M. Mining collective opinions for comparison of mobile apps. *Procedia Comput. Sci.* **2016**, *94*, 168–175. [CrossRef]
34. McIlroy, S.; Ali, N.; Khalid, H.; EHassan, A. Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews. *Empir. Softw. Eng.* **2016**, *21*, 1067–1106. [CrossRef]
35. Rasool, M.; Ismail, N.A.; Boulila, W.; Ammar, A.; Samma, H.; Yafooz, W.M.; Emara, A.H.M. A Hybrid Deep Learning Model for Brain Tumour Classification. *Entropy* **2022**, *24*, 799. [CrossRef] [PubMed]