Deployment of Deep Learning Models to Mobile Devices for Spam Classification

Ameema Zainab, Dabeeruddin Syed and Dena Al-Thani

January 18, 2020

Deployment of Deep Learning Models to Mobile Devices for Spam Classification

Ameema Zainab*, Dabeeruddin Syed*
Department of Electrical & Computer Engineering
*Texas A&M University
College Station, Texas, U.S.A.
{azain, dsyed}@tamu.edu

Dena Al-Thani†
Department of Information & Communications Technology
†Hamad bin Khalifa University
Doha, Qatar
dalthani@hbku.edu.qa

***Abstract*—The advent of deep learning brings the possibility of better and faster applications in real world. In this work, deep learning models are used for application of spam classification in mobile devices. A Binary Classification model is trained with deep learning and is transformed to a graph using tensorflow and then, is converted to a protobuf file to be deployed on mobile devices. Instead of looking into the spam messages in an algorithmic way i.e. just with keywords, binary model deals with experience of learning and predicts if a text message is spam. The training was performed multiple times on resource-deficient devices and hyper-parameter optimization was performed to enhance the training accuracy to 99.87%. The test accuracy of mobile application is 98.7% and testing happens in real-time without any internet access. Our simulation shows that a model with an embedding layer (size 128), an LSTM layer (size 64, dropout 0.2) and a dense layer (sigmoid) yields the highest performance. Also, the comparative evaluation with state-of-the-art methods displayed that our model achieves higher accuracy.**

***Keywords*—Deep learning; recurrent neural networks; transfer learning; spam classification; text analysis.**

I. Introduction

The number of spam messages received on mobiles is tremendously growing. There is a high probability that the next target of spammers will be the instant messaging apps. Such spam messages received over instant messengers are called SPIM (SPAM over Instant Messaging). Panda labs in Spain have announced in 2015 of the well-known deception hoax messages on a famous internet messaging platform called Whatsapp. The messages request the users to click on links as well as to forward the message to a chain of friends to acquire a reward. Also, in the U.K., there have been reports of many spam campaigns focusing on Whatsapp users who receive messages from supposedly U.S. numbers, but actually advancing phony extravagance products. To prevent such spam messages, a strong spam detection system must be in place. Spam classification initially started with simpler techniques like blacklist, whitelist and greylist [1]. Messages can be classified using automatic spam detection techniques using basic principles based on the content of text. Change in media type leads to challenges of using rule based techniques. Later, there has been significant improvement in the detection of spam messages using different machine learning (ML) models involving Support Vector Machines (SVM), Naïve Bayes, ensemble methods like Random Forest and classifiers based on Decision Trees. These algorithms worked very well, but involve feature engineering and hybrid classifiers, which are time consuming. This study focuses on building a spam detection model using Recurrent Neural Networks (RNN). The trained RNN model is transferred to a mobile device to execute as an application (app) to classify if a message is spam.

II. Research Objectives

The primary goal of this work is to deploy the deep learning (DL) model on a mobile device. Our research was successful deploying text classification model that exploits the context of the words in the sentences and takes advantage of DL models to enhance the classification accuracy. DL models are usually computationally hungry and deploying these models on a mobile device with less resources stands out as a challenge.

The essential objective of this research work is to enable the mobile carriers to provide the facility to classify an SMS as spam. This study focuses on using DL model built on RNN to detect spam messages, by extracting features automatically. The model used has shown exceptional performance on spam messages test data with ROC curve area of 0.9958 (Fig. 5).

In this setting, we present the following objectives:

- *Representation of words based on the context*: extracting features that uses the context of words in sentences.
- *Evaluate performance of text classification using DL models*: involves pre-processing the training data, training DL models with the spam dataset and performing hyper-parameter optimization to enhance the training accuracy.
- *Deployment of a DL model to mobile phone and transfer of knowledge to resource-deficient devices*: Here, the aim to generate final sequential model and do graph transformations. These allow for the transfer of a trained DL model to mobile device and the developed app uses the model for the classification of the incoming texts.

III. Background Work

DL is a division of ML inspired by the neurons in brains. The word deep refers to the large count of layers in neural networks (NN), however its meaning has changed with time from 10 layers considered as deep once to hundreds of layers being deep today [2]. The basic architecture of a DL model involves input, hidden and output layers. A few of the concepts used in the work are described below:

A. Recurrent Neural Networks (RNN)

RNN are a class of Artificial NN used in Natural Language Processing (NLP) applications with promising results. The idea behind RNN is to utilize sequential information. It uses the concept of 'memory' which arrests and accumulates information of the calculations.

B. Long Short Term Memory (LSTM)

A well-known architecture of RNN is the LSTM network. It uses gradient based method to store information for extended duration of time and combats vanishing gradients. It consists of a memory block with one cell, three gates and multiplicative units. The gating functions are learned together with weights, and determine how much information is preserved from last state and current computation, etc.

The memory in LSTM can be taken as input of the o_{t-1} and y_t. For output, m_t is the current memory unit and o_t is the current result. Different sections of LSTM are given below: New Memory: New memory and a new memory gate(valve).

$$i_t = \sigma(\theta_{xi}y_t + \theta_{oi}o_{t-1} + \theta_{mi}m_{t-1} + b_i) \tag{1}$$

Forget Valve: Valve that shuts down the old memory.

$$f_t = \sigma(\theta_{xf}y_t + \theta_{of}o_{t-1} + \theta_{mf}m_{t-1} + b_f) \tag{2}$$

Memory: Sum to combine new and old memory to form m_t.

$$m_t = f_t x_{t-1} + i_t \tanh \theta_{xc}y_t + \theta_{oc}o_{t-1} + b_c \tag{3}$$

Output: Output of LSTM unit and output valve.

$$\begin{aligned} o_t &= \sigma(\theta_{xo}y_t + \theta_{oo}o_{t-1} + \theta_{mo}m_t + b_o) \\ o_t &= o_t \tanh m_t \end{aligned} \tag{4}$$

C. TensorFlow

Tensorflow uses a multi-layered nodes' system, that allows to design and train artificial NN with massive data at high speed. It provides various dependencies and functionalities. In this work, TensorFlow has been utilized for training RNN.

D. Keras

It is an open source NN API developed in Python and has the capability of running on top of tensorflow. Keras makes it easier to work with tensorflow which otherwise is not user-friendly and is low-level. It runs seamlessly with GPU, CPU and supports RNN and CNN. Keras is used in our model.

The sequential modelling in Keras provides a model with a linear stack of layers [2]. The sequential modelling expects the input's shape to be an array or a matrix. The input_shape and batch_size are given as inputs for stateful RNN. The process of Keras sequential modeling involves following stages:

1) *model.fit* - Trains the model for a fixed number of iterations (epochs) and returns a history object attribute as a record of training loss values, metrics for each epoch and validation loss metrics.
2) *model.compile* - sets the model configuration for training.
3) *model.evaluate* - evaluates the model based on evaluation metrics and returns the test dataset loss value.

E. Regularization

Deep Neural Networks (DNN) are vulnerable to overfitting. Some of the most used regularization techniques are L1 or Lasso (penalizes non-zero weights), L2 or Ridge (penalizes large weights) and soft weight sharing. In this work, Dropout regularization [3] has been utilized. Unlike L1 or L2 regularizer, instead of modifying the cost function, the system network is modified using Dropout. Units (hidden or visible) are randomly (temporarily) dropped thinning the networks, accompanied by removing the connections. Dropout provides the impact of averaging the results over a number of networks. At test time, it makes easier to average the output of the thinned networks by utilizing a solitary network with little weights.

F. One Hot Encoding

In ML applications, data is required to be in binary or number format so computers can interpret. Hence, huge amount of time is spent to pre-process and clean the data. Categorical or string features need to be encoded to a number to be fed to the NN. The method of transformation of categorical features to integer representations is One Hot Encoding. This method modifies each categorical feature with n +ve values into n binary features, with just a single active [4]. In cases of missing values in the train data, one has to use any imputation method to fill in the missing fields [5].

IV. Related Work

Research is actively in progress in the area of spam classification. Although the field of implementing deep inference networks on mobile devices is new, researchers have active profound interest to convert the heavy and trained NN models into miniature models which can run in mobile devices. Mostly, research in implementation of the DL models and in the area of text classification have been done mostly independently. In this section, we will first present the DL models and literature review in text classification and then, present our work that combines two different areas.

In [6], Nicolas et al. present the possibility of using DL models for speech and object detection on mobile devices in the form of sensor app. Their implementation prototypes a low-paper DNN engine for activity recognition. The findings prove the robustness and acceptable levels of resource usage of DNNs on the mobile devices In [7], Syed et al. present the use of DL models in the integration of multi-sensor data including video feed for real-time occupancy detection for environment control strategies. They have worked on resource-deficient devices of raspberry pi and micro-arduino controller. The results show high accuracy after transfer learning to the devices. In [8], Mohammad et al. present a survey on the use of DL in mobile big data analytics and propose a framework using apache spark. The spark-based framework has been validated for speed effectiveness of DL on mobile devices using context-aware application with real-world dataset. Adoption of DL models on mobiles requires considerable resources and this issue has been lowered by design and implementation of a software accelerator called DeepX by Nicholas et al. [9]. The accelerator uses resource control algorithms which work during the inference stage of DL and performs the resource scaling that adjusts the architecture of DL models to be used efficiently on mobile devices and the results show better efficiency over cloud-based offloading. In [10], Nicholas et al. developed a cloud-free DSP prototype for the smart phones for speech recognition applications. The application called DeepEar has been developed after training 5-layer 1024 unit deep NN with large datasets and the resultant trained model containing 2.3 million parameters is still feasible to be executed on mobile devices in terms of accuracy and battery consumption.

With the increasing usage of the internet, the chances of spam messages have been increasing. Spam detection is an adversarial classification problem, where the predictive models are trying to adapt continuously to spam with evolving spamming techniques to evade filtering [11]. The traditional method of using bag of words always had limitations in text categorization problems . In [12], Zongda et al. presented an effective approach of text classification based on semantic matching using huge wikipedia semantic space. In [13], Zenun et al. worked on the classification of financial documents based on semantics in the document. In the first stage of classification, the documents are represented semantically using ontology and this serves as input to the next stage. The results were found using different simulations with DL configuration of 3 layers each with 1024 neurons resulting in an accuracy of 78% with INFUSE dataset.

Similarly, we have used word2Vec because it stresses on the context of the words. Word-vectors position themselves in the vector space so that the words having common context are positioned at close proximity. Basically it uses the words close to each other so as to estimate the target words within a NN. Also, there have been many methods that tried to complement developed techniques with Bag of Words features. Almeida et al. offered real, open and non-encoded dataset (SMS Spam Collection) and used it to analyze spam detection using ML strategies [14]. Their study demonstrated that the SVM outperformed other classifiers, concluding that for further correlation, it can be classified as a decent baseline. Uysal et al. (2012) [15] used two distinct feature selection methods to discover distinctive features that constitute SMS messages, based out of chi-square metrics and information gain. To label the SMS as spam or not spam, the discriminated feature subsets were employed into two different Bayesian-based classifiers with highly accurate classification results.

Ahmed et al. employed the Bayesian filtering approaches utilized in blocking email spam and extended it to the issue of detecting and curbing spam in mobiles [16]. They eventually built two SMS corpus test collections in two languages (English and Spanish) to conduct their study. The results display that Bayesian filter techniques used for email spam can be successfully used on spam SMS. Almeida et al. have evaluated a procedure to normalize and interpret short messages with an aim of acquiring better attributes resulting in an enhanced classification performance [17]. The proposed approach is based on semantic and lexicographic dictionaries for text processing, context identification and semantic analysis. Boujnouni proposed a filter based on three components for text message classification: N-grams method (for extracting features from short messages), the information gain ratio (for selecting the most relevant features), and an enhanced version of Support Vector Domain Description (SVDD) to SSPV-SVDD [18].

V. Methodology

Training of a network needs high computational resources. Most complex models take weeks to train on machines even with GPU resources. Owing to the model complexities and size, training of a model is usually hosted in the cloud. For a mobile-device to use this model, connection with the network is required and it increases response time. There has been considerable success to equip mobile devices with GPUs to be able to train the models on the device. This work focuses on the device-based model where the trained model weights are moved to a mobile device. With the help of protobuf(.pb) file(android) and core ML(iOS), the trained ML models can be integrated into an application. Tensorflow, a DL library, supports this feature in Android. The design in Fig. 1 illustrates the process flow of implementation.

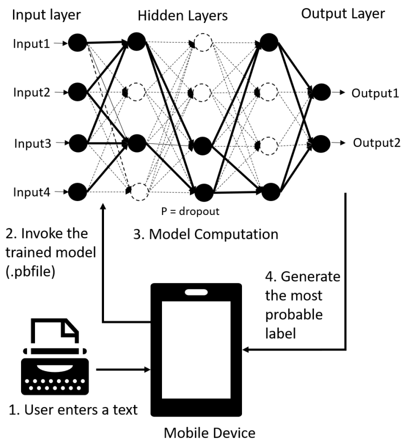

Fig. 1: Device-based Inference - Process Flow

A. Tools

a) Android Studio: It is the official Integrated Development Environment for Android operating system. In this work, the android studio version of 3.0, SDK version of 26.0.2 and NDK version of 12.1.2977051 were used on the mobile device (Lenovo P2 2017).

b) Tensorboard: Tensorboard was used to visualize the trained ML model. Since the graph visualizations are complicated and humanely impossible to interpret, tensorboard provides visual capabilities and aid in debugging the graph.

The tool makes it easy to understand the in-depth nodes and the flow of the graph. Visualization was comprehensive enough to locate all the node labels, hidden nodes, grouping of nodes, etc.

1) Data Acquisition: The SMS Spam Collection database has been used for training our model [14]. This collection is considered a benchmark dataset in SMS Spam research. The data set contains 747 spam messages (13.40%) and 4827 'not spam' messages (86.60%).

2) Data Statistics: The longest word length of a sentence (MAX_LENGTH) in train data was 215 and words count in the dictionary (WORD_FREQS) was 9774.

3) Word to Vector (word2vec): To deal with the strings data, the data has been converted to arrays of vectors with the help of python word2vec feature. The word2vec feature uses the concept of creating dictionaries using all the data fed, and assigns a rank to each word. The dictionary in our analysis consisted of 9774 words. Once the dictionary is created, each word is replaced with a rank from the dictionary. Each word is tracked with the occurrence of a space in the sentence. Since all the sentences are not of unique length, zero padding has been used to append all the arrays with 0's (to match the length of the longest sentence). Hence, all the sentences are converted to arrays of fixed length and fed to the Keras DL model.

B. Keras model used

Keras Sequential model, that is used to train the data, contains Embedding, LSTM and Dense layers. The embedding layer is used to process the words in NLP and is a vital advancement in the process.. The Dense layer (or fully connected layer) has been added at the end. The data has been divided into train (80%) and validation set (20%). The dataset has also been evaluated on various division metrics such as (60% train - 40% validation, 50% train - 50% test) and found to have achieved similar accuracy results (Table II). All the layers can be summarized as below:

(4459, 220) (1115, 220) (4459, 1) (1115, 1)

Layer (type)	Output Shape	Param #
I (Embedding)	(None, 220, 128)	1251328
lstm 9 (LSTM)	(None, 64)	49408
output activation node (Dens)	(None, 1)	65

Total parameters: 1,300,801,
Dropout % in LSTM layer is 20%

TABLE I: Keras sequential model

The model has been fit with training data with batch size 32 and 10 epochs. The trained model is moved to a weights' file of format .h5. using graph feature compatible with tensorflow in Python. The trained weights have been converted to a protobuf file. The workflow of the model is depicted in Fig. 2.

C. Graphs

The base of the computations in tensorflow is a graph object which contains a network of nodes linked to each other as input and output. After creation, a graph object can be saved as a GraphDef object. A protocol buffer library creates an object using GraphDef class. The graph converted to protobuf is in text (graph.pbtxt) or binary (graph.pb) format. The text file is readable but is huge in size whereas the binary file is a smaller in size and not readable. Once a file is loaded into a graphdef object, the data inside it can be accessed i.e., list of nodes, name of the layers, etc.

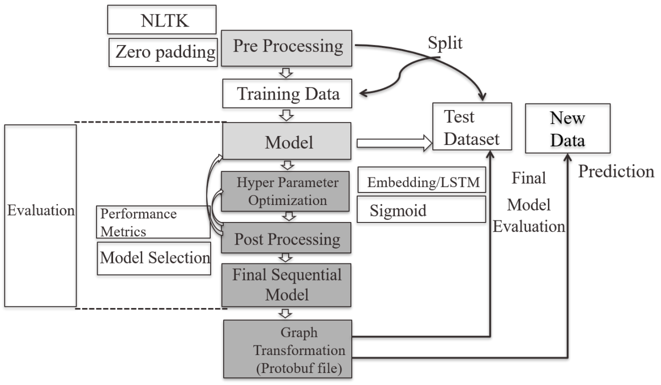

Fig. 2: Steps in the process of application

Each node in a NodeDef is the building block of Tensorflow graphs. The weights file is usually stored in variable ops called checkpoint files, which saves the latest values when initialized. These files are freezed into a single file to overcome the hassle of having separate files. This replaces each variable operation (add, mul, etc.) with a constant operation in all the variables from latest checkpoint file. The constant operation removes all the extra nodes which are not utilized for forward inference and stores only the numerical data of the stored weights in its attributes. The flow process is described briefly in the Fig. 3.

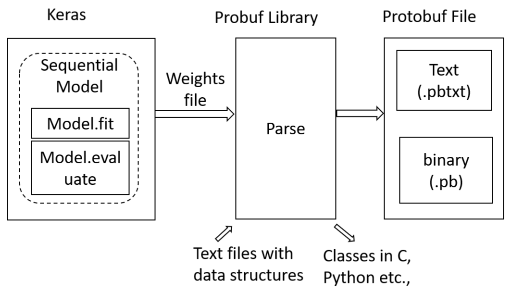

Fig. 3: Graph Model structural flow

D. Evaluation Measures

1) Area under the Receiver Operating Characteristic (ROC): The region under the ROC curve (AUC) is an established method of evaluation in classification analysis. It is a graph of operating points that focus on the plausible trade off of a classifiers' True Positive (TP) rate against the False Positive (FP) rate. The ROC value between 0.9 and 1.0 indicates highest performance of the classifier.

2) Matthews correlation coefficient (MCC): MCC score is utilized for performance evaluation of binary classification ML models. It is a balanced measure, even if the classes are of unequal sizes. It takes into account sensitivity, specificity, false positives and false negatives. It typically lies between -1 and +1.

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(FP+TP)(FN+TP)(FP+TN)(FN+TN)}} \quad (5)$$

Value of +1 for MCC indicates impeccable prediction, value of 0 shows average random prediction and value of -1 illustrates an inverse prediction.

3) K-Fold Cross Validation: It has been utilized for the evaluation of our training model. It splits the data set into k sections considering k-1 bins as part of the training data set and the rest as the validation set. For a K-Fold cross validation, the algorithm is run k times and the k test set performances are averaged. This method helps avoid over fitting. The accuracy results on considered dataset are shown in Table II.

Training Data(%)	**Test data** (%)	**Accuracy** (%)
50	50	99.89
80	20	99.87
60	40	99.85
70	30	99.95

TABLE II: Accuracy results for different data partitioning

E. Android Implementation

The model expects a text input and results in a numerical output (0 or 1) classifying if the message is a spam or not. The steps applied to achieve this are:

The Protobuf (.pb) file is the output of the trained model and is added to the assets folder. Two different classes built for android implementation are discussed below:

1) MainAcitivity Class: The input to the android app is the text. This text input is divided into an array of words as shown below:

input - 'The essence of deep networks will ever be interesting'
input to the ArrayList - ['The' 'essence' 'of' 'deep' 'networks' 'will' 'ever' 'be' 'interesting']
This input array is passed through a HashMap dictionary.
Output of HashMap Array - [51, 15, 426, 80, 41, 25, 551, 34, 208]
This input sequence length is one-hot encoded to the length of the longest sentence and sent to the textClassifier class using intents in Android studio.

2) TextClasssifier Class: The textClassifier is a main class file that performs classification. The .pb file is collected from the Assets folder and input, output layer names are provided as inputs to TensorflowInferenceInterface. Tensorflow provides Android support and the necessary packages for compatibility of Tensorflow and Android. This library supports Tensorflow and understands the graph file (.pb). The Tensorflow Android follows three steps namely (1) feed: which feeds the input name, type and size, (2) run: which runs the feed forward network, and finally, (3) fetch: which gets the result from the network.

The result is displayed in the application using textView feature. A sample result is shown in Fig. 6.

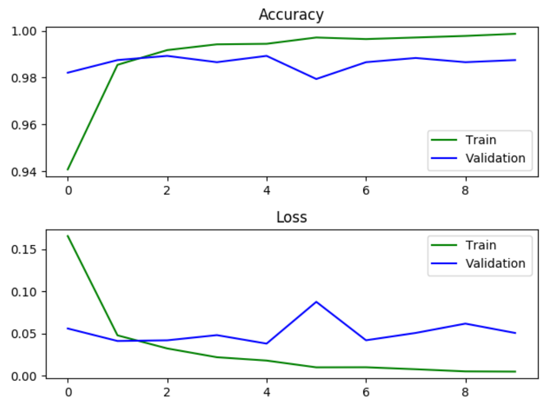

Fig. 4: Performance metrics of the Keras trained model

VI. Results

The results of proposed methodology have been compared to other detection mechanisms and the proposed model has achieved the best accuracy to date, especially compared to previously proposed models [17]. We have evaluated using MCC score which provide balanced evaluation even for unbalanced datasets.. Our model provides the best accuracy with an AUC of 0.9958 (Fig 5) and 0.96 MCC score (Table III).

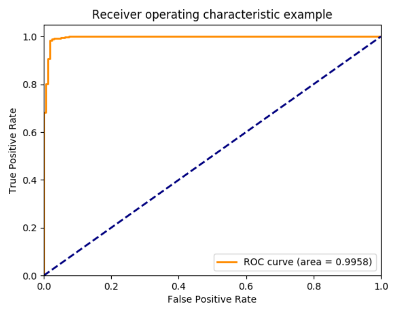

Fig. 5: Area under the Curve for test Data

Method used	**MCC Score**	**Accuracy %** (Train, Test)
Proposed	0.956	99.87, 98.7
Improved LZms Algorithm [17]	0.921	Not reported
SGD [19]	0.907	Not reported
SVM + tok1 [14]	0.893	Not reported
SSPV-SVDD[1] [18]	Not reported	95.13, 89.32
CHI2 and IG[2] [15]	Not reported	90.17 (overall)

TABLE III: Performance evaluation as against the benchmark works on SMS spam collection dataset

A. Spam Detection on Android

Fig. 4 shows the accuracy of experiment when the train-test split is 80% and 20%. The training accuracy is at 0.9987 and the test accuracy is 0.987. The ROC curve for the test data had Area under the curve of 0.9958 (Fig. 5). The implemented model on Android has outperformed the state-of-art results and runs successfully on the mobile device (Fig. 6).

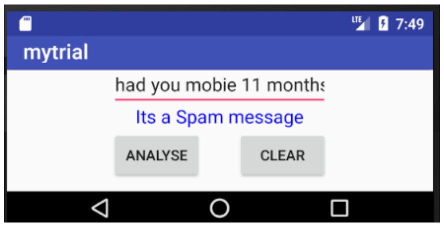

Fig. 6: Spam message: ”Had your mobile 11 months or more? You are eligible to win a latest camera phone and for free. For more information, call 07204996860”

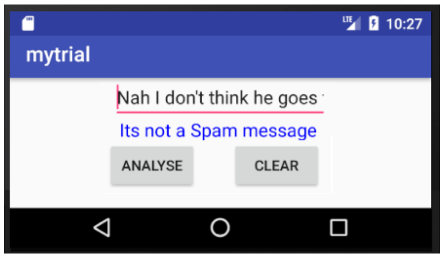

Fig. 7: Ham message: ”Nah I don’t think he goes there. He is very efficient and humble individual.”

VII. Challenges

The challenges during the deployment of DL model for spam detection on a mobile are summarized below:

- Making trained NN compatible for deployment
- DL is resource hungry and the NN models are quite complex. It is hard for a device to interpret the number of layers, the connections and the flow structure.
- Model compression techniques to convert huge model to a tiny constant model calls for multiple constraints.
- Tensorflow has mostly been used to train NN on PCs but the proposed work makes use of tensorflow on a phone.

VIII. Conclusion

The work focused on deploying deep NN to mobile devices and the testing has been performed successfully for spam detection. A mobile app has successfully been built, and it uses the Keras sequential model trained on a spam dataset. Fig. 6 shows the successful execution of installed app on a mobile device. The ML model has the same accuracy even after transfer. When a message is input, the app classifies if the message is a spam or not. This application is applicable to any messaging service and performs without any internet connection and with accuracy as high as 98%. Also, it is fast on a mobile with RAM as low as 1 GB. With transfer learning possible, it opens the gates for on-device intelligence leading to many possibilities for smart devices. TensorFlow Lite, a light weight cross platform for mobile and embedded devices, if used, would improve speed. In future, this work should be adapted to other languages like Spanish, Chinese, etc.

IX. Acknowledgment

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

References

[1] Kurt Thomas, Chris Grier, Justin Ma, Vern Paxson, and Dawn Song. Design and evaluation of a real-time url spam filtering service. In *Security and Privacy (SP), 2011 IEEE Symposium on*, pages 447–462. IEEE, 2011.

[2] Antonio Gulli and Sujit Pal. *Deep Learning with Keras*. Packt Publishing Ltd, 2017.

[3] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[4] Geoff Hulten. Machine learning intelligence. In *Building Intelligent Systems*, pages 245–261. Springer, 2018.

[5] Jadran Sessa and Dabeeruddin Syed. Techniques to deal with missing data. In *2016 5th international conference on electronic devices, systems and applications (ICEDSA)*, pages 1–4. IEEE, 2016.

[6] Nicholas D Lane and Petko Georgiev. Can deep learning revolutionize mobile sensing? In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*, pages 117–122. ACM, 2015.

[7] Dabeeruddin Syed and Amine Bermak. Integration of multisensor data and deep learning for realtime occupancy detection for building environment control strategies. In *Qatar Foundation Annual Research Conference Proceedings Volume 2018 Issue 3*, volume 2018, page ICTPD671. Hamad bin Khalifa University Press (HBKU Press), 2018.

[8] Mohammad Abu Alsheikh, Dusit Niyato, Shaowei Lin, Hwee-Pink Tan, and Zhu Han. Mobile big data analytics using deep learning and apache spark. *IEEE network*, 30(3):22–29, 2016.

[9] Nicholas D Lane, Sourav Bhattacharya, Petko Georgiev, Claudio Forlivesi, Lei Jiao, Lorena Qendro, and Fahim Kawsar. Deepx: A software accelerator for low-power deep learning inference on mobile devices. In *Proceedings of the 15th International Conference on Information Processing in Sensor Networks*, page 23. IEEE Press, 2016.

[10] Nicholas D Lane, Petko Georgiev, and Lorena Qendro. Deepear: robust smartphone audio sensing in unconstrained acoustic environments using deep learning. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 283–294. ACM, 2015.

[11] Andrej Bratko, Gordon V Cormack, Bogdan Filipič, Thomas R Lynam, and Blaž Zupan. Spam filtering using statistical data compression models. *Journal of machine learning research*, 7(Dec):2673–2698, 2006.

[12] Zongda Wu, Hui Zhu, Guiling Li, Zongmin Cui, Hui Huang, Jun Li, Enhong Chen, and Guandong Xu. An efficient wikipedia semantic matching approach to text document classification. *Information Sciences*, 393:15–28, 2017.

[13] Zenun Kastrati, Ali Shariq Imran, and Sule Yildirim Yayilgan. The impact of deep learning on document classification using semantically rich representations. *Information Processing & Management*, 56(5):1618–1632, 2019.

[14] Tiago A Almeida, José María G Hidalgo, and Akebo Yamakami. Contributions to the study of sms spam filtering: new collection and results. In *Proceedings of the 11th ACM symposium on Document engineering*, pages 259–262. ACM, 2011.

[15] Alper Kursat Uysal, Serkan Gunal, Semih Ergin, and Efnan Sora Gunal. A novel framework for sms spam filtering. In *Innovations in Intelligent Systems and Applications (INISTA), 2012 International Symposium on*, pages 1–4. IEEE, 2012.

[16] José María Gómez Hidalgo, Guillermo Cajigas Bringas, Enrique Puertas Sánz, and Francisco Carrero García. Content based sms spam filtering. In *Proceedings of the 2006 ACM symposium on Document engineering*, pages 107–114. ACM, 2006.

[17] Tiago A Almeida, Tiago P Silva, Igor Santos, and José M Gómez Hidalgo. Text normalization and semantic indexing to enhance instant messaging and sms spam filtering. *Knowledge-Based Systems*, 108:25–32, 2016.

[18] Mohamed El Boujnouni. Sms spam filtering using n-gram method, information gain metric and an improved version of svdd classifier. *Journal of Engineering Science & Technology Review*, 10(1), 2017.

[19] Renato M Silva, Tulio C Alberto, Tiago A Almeida, and Akebo Yamakami. Towards filtering undesired short text messages using an online learning approach with semantic indexing. volume 83, pages 314–325. Elsevier, 2017.