IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 28, 2021, accepted May 11, 2021, date of publication May 17, 2021, date of current version June 8, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3081479

A Spam Transformer Model for SMS Spam Detection

XIAOXU LIU, HAOYE LU, (Member, IEEE), AND AMIYA NAYAK, (Senior Member, IEEE)
School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, ON K1N 6N5, Canada
Corresponding author: Haoye Lu (hlu044@uottawa.ca)

ABSTRACT In this paper, we aim to explore the possibility of the Transformer model in detecting the spam Short Message Service (SMS) messages by proposing a modified Transformer model that is designed for detecting SMS spam messages. The evaluation of our proposed spam Transformer is performed on SMS Spam Collection v.1 dataset and UtkMl's Twitter Spam Detection Competition dataset, with the benchmark of multiple established machine learning classifiers and state-of-the-art SMS spam detection approaches. In comparison to all other candidates, our experiments on SMS spam detection show that the proposed modified spam Transformer has the optimal results on the accuracy, recall, and F1-Score with the values of 98.92%, 0.9451, and 0.9613, respectively. Besides, the proposed model also achieves good performance on the UtkMl's Twitter dataset, which indicates a promising possibility of adapting the model to other similar problems.

INDEX TERMS SMS spam detection, transformer, attention, deep learning.

I. INTRODUCTION

A. MOTIVATION AND OBJECTIVE

THE Short Message Service (SMS) has been widely used as a communication tool over the past few decades as the popularity of mobile phone and mobile network grows. However, SMS users are also suffering from SMS spam. The SMS spam, also known as drunk message, refers to any irrelevant messages delivered using mobile networks [1]. There are several reasons that lead to the popularity of spam messages. Firstly, there is a large number of users who use mobile phones in the world, making the potential victims of the spam messages attack also high. Secondly, the cost of sending out spam messages is low, which could be good news to the spam attacker. Last but not least, the capability of the spam classifier on most mobile phones is relatively weak due to the shortage of computational resources, which limits them from identifying the spam message correctly and efficiently.

Machine learning is one of the most popular topics in the last few decades, and there are a great number of machine learning based classification applications in multiple research areas. Specifically, spam detection is a relatively mature research topic with several established methods. However, most of the machine learning based classifiers were dependent on the handcrafted features extracted from the training data [2].

As a class of machine learning techniques, deep learning has been developing rapidly recently thanks to the surprising growth of computational resources in the last few decades. Nowadays, deep learning based applications play a significant part in our society, making our lives much easier in many aspects. As one of the most effective and widely used deep learning architectures, Recurrent Neural Network (RNN), as well as its variants such as Long Short-Term Memory (LSTM), were applied to spam detection and proved to be extremely effective during the last few years.

The Transformer [3] is an attention-based sequence-to-sequence model that was originally designated for translation task, and it achieved great success in English-German and English-French translation. Moreover, there are multiple improved Transformer-based models such as GPT-3 [4] and BERT [5] proposed recently to address different Natural Language Process (NLP) problems. The accomplishments of the Transformer and its successors have proved how powerful and promising they are. In this paper, we aim to explore whether it is possible to adapt the Transformer model to the SMS spam detection problem. Therefore, we propose a modified model based on the vanilla Transformer to identify SMS spam messages. Additionally, we analyze and compare the performance of SMS spam detection between traditional

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Xiang.

machine learning classifiers, an LSTM deep learning solution, and our proposed spam Transformer model.

B. RELATED WORK

There are several different machine learning based classification applications proposed in the last few decades [6], [7] [8], [9]. In the field of SMS spam detection, a great number of these approaches are based on traditional machine learning techniques, such as Logistic Regression (LR), Random Forest (RF) [10], Support Vector Machine (SVM) [11], Naïve Bayes (NB), and Decision Trees (DT). Recently, with the prosperity of the deep learning techniques, an increasing number of methods have been introduced to address the SMS spam problem using deep learning based solutions such as Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), and Long Short-Term Memory (LSTM), which is a successful variant of RNN.

In [12], Gupta *et al.* compared the performance of 8 different classifiers including SVM, NB, DT, LR, RF, AdaBoost, Neural Network, and CNN. The experimental tests on the SMS Spam Collection v.1 [13] dataset that was conducted by the authors shows that the CNN and Neural Network are better compared to other machine learning classifiers, and the CNN and Neural Network achieved an accuracy of 98.25% and 98.00%, respectively.

In [14], Jain *et al.* proposed a method to apply rule-based models on the SMS spam detection problem. The authors extracted 9 rules and implemented Decision Tree (DT), RIPPER [15], and PRISM [16] to identify the spam messages. According to the experimental results from the authors, the RIPPER outperformed the PRISM and the DT, yielding a 99.01% True Negative Rate (TNR) and a 92.82% True Positive Rate (TPR).

In [1], Roy *et al.* aimed to adapt the CNN and LSTM to the SMS spam messages detection problem. The authors evaluated the performance of CNN and LSTM by comparing them with Naïve Bayes (NB), Random Forest (RF), Gradient Boosting (GB) [17], Logistic Regression (LR), and Stochastic Gradient Descent (SGD) [18]. The experiments that were conducted by the authors showed that the CNN and LSTM perform significantly better than the tested traditional machine learning approaches when it comes to SMS spam detection.

In [2], the authors proposed the Semantic Long Short-Term Memory (SLSTM), a variant of LSTM with an additional semantic layer. The authors employed the Word2vec [19], the WordNet [20], and the ConceptNet [21] as the semantic layer, and combined the semantic layer with the LSTM to train an SMS spam detection model. The experimental evaluation that was conducted by the authors claimed that the SLSTM achieved an accuracy of 99% on the SMS Spam Collection v.1 dataset.

In [22], Ghourabi *et al.* proposed the CNN-LSTM model that consists of a CNN layer and an LSTM layer in order to identify SMS spam messages in English and Arabic. The authors evaluated the CNN-LSTM by comparing it with the CNN, LSTM, and 9 traditional machine learning solutions. The experimental tests that were conducted by the authors showed that the CNN-LSTM solution performed better than other approaches and yield an accuracy of 98.3% and an F1-Score of 0.914.

C. PAPER ORGANIZATION

The rest of the paper is organized as follows. Section II provides the backgrounds and details of the LSTM and our spam Transformer approaches. Concretely, Section II-A introduces the architecture of RNN, followed by one of its most successful variant LSTM in Section II-B. We then introduce Sequence-to-Sequence in Section II-C, attention mechanism in Section II-D, and the original version of Transformer for translation tasks in Section II-E. Furthermore, Section III discusses the modified spam Transformer that we proposed in detail. Afterward, Section IV demonstrates the experiment designs, results and analysis. Finally, we conclude in Section VI and describes the future work in Section VII.

II. DEEP LEARNING APPROACHES

While the traditional machine learning techniques do perform well in many fields, they are still much interference or guidance from human specialists required when people try to apply these technologies to address problems. For instance, extracting and representing the features from data is always a challenging but indispensable work for machine learning scientists. In another word, the inadequate capacity of many traditional machine learning classifiers is a major limitation to a more effective and massive application. However, many deep learning techniques are able to not only learn much more amount of features but also extract more higher-level features that are formed by the composition of lower-level features. With an effective training process, the deep learning techniques are more capable to consume and make good use of a large amount of data and thus perform better especially in coping with difficult jobs compared to the traditional machine learning approaches.

A. RECURRENT NEURAL NETWORK

As is known to all, shuffling the order of words in a sentence can severely influence the meaning of the entire sentence, which could potentially turn a legitimate message into spam messages, and vice versa. Therefore, in many Natural Language Process (NLP) problems, the order of words is no less important than the words themselves. To address this problem, we need a new kind of model that is capable to effectively learn from prior knowledge to improve the understanding of the data. Although the classical feed-forward neural network is a powerful deep learning technique that generally works well in many areas, it cannot utilize the information from the past. Derived from the feed-forward neural network, recurrent neural network (RNN) [23] has the ability to reuse the saving information at the time of processing input values. Additionally, unlike the traditional feed-forward neural network

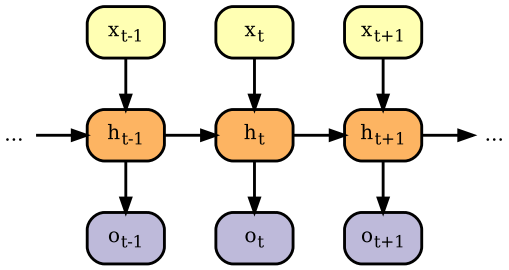

FIGURE 1. Structure of a typical Recurrent Neural Network.

supports only the input sequence with a fixed length, RNN is capable to handle the input sequence with different length.

The Fig. 1 shows the typical structure of RNN models, with the input sequence, output sequence, and the hidden layers at time t are represented by x_t, o_t, and h_t respectively. At time t, the current hidden layers state h_t is calculated based on the current input sequence x_t and the last hidden layers h_{t-1}. After the calculation of h_t is finished, the output at the current time step o_t is generated and the hidden layers state h_t will get involved in the calculation at the next time step t + 1. Unlike the normal neural network, where the neurons in the same layer of the hidden layers are independent of each other, RNN models usually allow the data flows within the same layer. In another word, connections between neurons in the same layers or even self-connections are allowed generally allowed in RNN based models.

A major advantage of RNN models is that they are able to utilize the information from previous input and apply it at the current time, which is significantly useful in NLP problems since the context can help us understand the sentence better. However, a major drawback of the vanilla RNN is the vanishing and exploding gradients [24]. In back-propagation training process, the vanishing gradients refers to gradients go exponentially close to 0, while the exploding gradients refers to the gradients go exponentially increase. The vanishing and exploding gradients are usually caused by the multiplication of multiple derivatives in training process. Although there are several approaches [25] existing to address the vanishing and exploding gradients problem, in practice, it is still difficult for vanilla RNN to memorize and learn the features from long distance, which is described as long-term dependencies problem. In order to deal with the long-term dependencies problem, many researchers have proposed multiple variants of RNN, such as the Long Short-Term Memory (LSTM) [26], the Gated Recurrent Unit (GRU) [27], and the Clockwork RNN (CW-RNN) [28].

B. LONG SHORT-TERM MEMORY

The Long Short-Term Memory (LSTM) is a famous variant of RNN. The main idea of the LSTM is the introduction of gate units, which are the structures that can determine to keep or discard the current information. A typical LSTM network consists of multiple memory cells, and each memory cell is formed by an input gate, a forget gate, and an output gate. In LSTM, at time t, the state of a memory cell c_t is calculated based on the input x_t and the last hidden state h_{t-1}. The state of input gate, output gate, and forget gate at time t are represented as i_t, o_t, f_t, respectively. Therefore, the LSTM transition functions are defined as follows [29]:

$$\begin{aligned} i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\ q_t &= tanh(W_q \cdot [h_{t-1}, x_t] + b_q) \\ o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\ c_t &= f_t \odot c_{t-1} + i_t \odot q_t \\ h_t &= o_t \odot tanh(c_t) \end{aligned} \tag{1}$$

The σ denotes the sigmoid function, and the operator $\odot$ denotes the element-wise multiplication. The sigmoid function is a logistic function with the returning value between 0 and 1. The sigmoid function is defined as follow:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2}$$

When the output of a gate unit is close to 1, the information is more likely to be memorized. On the contrary, a returning value close to 0 from a gate unit means that the information should not be kept. The input gate i_t is the gate unit that controls how much information should be stored at this time. The forget gate f_t is responsible to determine to what extent the memory from the last time c_{t-1} should be kept at time t. The output gate o_t at time t is designed to be used in the computation of the output (hidden state) based on the memory cell state.

In our LSTM approach for SMS spam detection, the input message embedding is fed into an LSTM network as an input sequence. Meanwhile, the LSTM network saves the important features and outputs a sequence with the same length as the input sequence. The output sequence is then fed into a feed-forward fully connected layer with a single neuron since SMS spam detection is a binary classification problem. Finally, a sigmoid function is applied to the output of the single neuron to produce a final prediction.

C. SEQUENCE-TO-SEQUENCE MODELS

Sequence-to-sequence (Seq2Seq) [30] was introduced in 2014 by Sutskever *et al.* aiming to find a mapping between two sequences for translation tasks. Seq2Seq models employed the Encoder-Decoder architecture, which consists of an encoder stack, a hidden state, and a decoder stack. Fig. 2 presents a typical Encoder-Decoder architecture. The encoders take the input sequence and produce a hidden state with critical information, which is consumed by the decodes to generate the output sequences. One of the crucial advantages of the Encoder-Decoder architecture is that the input sequence and output sequence can be different in terms of size or format, which provides much more flexibility and possibility. In reality, the Seq2Seq models have been proved themselves in language translation [30], Speech Recognition [31], and Video to Text [32]. Undoubtedly, Seq2Seq

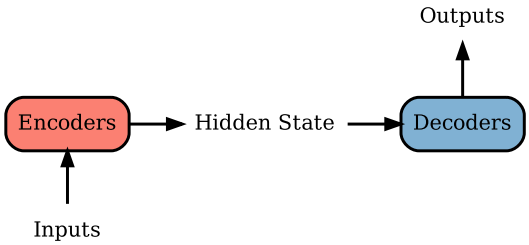

FIGURE 2. Structure of Encoder-Decoder architecture.

architecture is designed to fit translation tasks exceptionally well, since it can extract the relationship between the sequences in one language and the sequences in a different language. The vanilla version of the Seq2Seq model proposed in 2014 choose LSTM as both encoder and decoder, because LSTM has the ability to successfully learn on data with long-term dependencies [30].

D. ATTENTION MECHANISM

The main purpose of the attention mechanism is to find out the most important part from the input sequence. Concretely, the attention mechanism produces weights that represent the importance of the elements based on their correlation with the context. The attention mechanism makes it possible to focus on the key elements.

In [33], the attention mechanism was introduced as an improvement of the RNN Encoder-Decoder model hidden state in Neural Machine Translation (NMT). The most important contribution of the attention mechanism in NMT is that it computes the weights based on all the hidden states generated by the encoder, and the decoder consumes the weighted combination of all the hidden states instead of focusing only on the latest one. The introduction of the attention mechanism greatly boosts the performance of NMT.

There are also other forms of attention mechanism proposed. In [34], the attention mechanism is applied to the field of computer vision by Xu *et al.*, and they also proposed two different approaches of attention named ''soft attention'' and ''hard attention''. In [35], Luong *et al.* proposed global attention and local attention. The global attention is similar to the model of Bahdanau *et al.* in [33] with a simpler architecture, while the local attention is a combination of soft and hard attention from Xu *et al.* in [34].

E. THE TRANSFORMER MODEL

The Transformer [3] model is a sequence-to-sequence (Seq2Seq) model that was proposed in 2017 by Vaswani *et al.*, as an approach to English-German and English-French translation tasks. Compared to those previous Seq2Seq models, the main innovation of Transformer is that it completely relies on the attention mechanism to efficiently learn from the most informative elements [36].

Though LSTM and some other RNN variants were proved to perform well as encoders and decoders in Seq2Seq based models, the high training consumption of recurrent models becomes a significant limitation. At time t, the computation of hidden state h_t relies on the previous hidden state h_{t-1}, which is the sequential computation nature of recurrent models. This sequential computation nature prevents the computing of RNN variants from parallelization, leading to the limitation on computational efficiency during the training process.

In order to address the computational efficiency limitation of RNN variants, the Transformer uses only multi-head attention mechanism instead of RNN variants as encoders and decoders. This not only greatly reduces the cost of training through parallelization, but also surprisingly improves the performance in translation tasks as is mentioned in [3].

In Transformer, the attention function takes a query Q and a set of key-value pairs (K, V) as input, and computes the weighted sum of values as output, where the weights are calculated based on the queries and keys. Particularly, Scaled Dot-Product Attention is used in Transformer as the attention function. The Scaled Dot-Product Attention is the dot-product attention [35] with a scaling factor of $\frac{1}{\sqrt{d_k}}$, which aims to counteract the massive growth of dot-product when dimensions of queries and keys d_k is large.

Another important innovation of Transformer is the Multi-Head Attention. In the previous practice, the attention is directly performed on the queries, keys, and values, where their dimension is d_{model}. In this way, there is only a single attention function calculated at one turn. However, Transformer finds an effective way to apply multiple attention functions at once. Specifically, the queries, keys, and values are sent to some different learned linear layers to be projected h times to the dimension of d_k, d_k, and d_v, respectively. In another word, the projection linear layers are individually learned, and output projections have dimensions of d_k, d_k, and d_v, where $d_k = d_v = d_{model}/h$. After that, a number of h attention functions are performed in parallel on these projected queries, keys, and values, resulting in h different output values. Finally, all these h values are concatenated together and then projected back to a dimension of d_{model}. The entire process of the attention mechanism in the Transformer is defined as follows [3]:

$$\begin{aligned} Attention(Q, K, V) &= softmax(\frac{QK^T}{\sqrt{d_k}})V \\ MultiHead(Q, K, V) &= Concat(head_1, \ldots, head_h)W^O \\ head_i &= Attention(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \quad (3)$$

The W_i^Q, W_i^K, and W_i^V are parameters matrices in linear projection layers, where they are used to project d_{model}-dimension queries, keys, and values to d_k, d_k, and d_v dimension, respectively. In both vanilla Transformer and our modified Transformer for SMS spam detection, $d_k = d_v = d_{model}/h$.

In RNN, the computation of the hidden states is based on the previous states, making it available to learn from the order of words naturally. However, there is no recurrent or convolutional structure in Transformer. Therefore, Transformer

introduces a positional encoding function based on *sine* and *cosine* functions of different frequencies.

In vanilla Transformer model designed for language translation tasks, source language texts and shifted right target language texts are first sent to embedding layers as input sequence and output sequence. Secondly, positional information is injected into the input and output sequence in the positional encoding layer. After that, the input and output sequence is fed into encoders and decoders, respectively. Then, the Multi-Head Attention layers and fully-connected feed-forward layers, combined as a single encoder or decoder, produce the output of dimension of d_{model}. The results of decoders are passed to a linear layer. Finally, the softmax function is performed on the output of the linear layer, producing the translation in the target language.

III. PROPOSED MODIFIED TRANSFORMER MODEL FOR SMS SPAM DETECTION

In Fig. 3, the main architecture of the modified Transformer model for SMS spam detection is described. In order to apply the Transformer model to the SMS spam detection task, two major modifications are done to the vanilla Transformer model, which is described in Section III-A and Section III-B, respectively. After that, several implementation details are discussed.

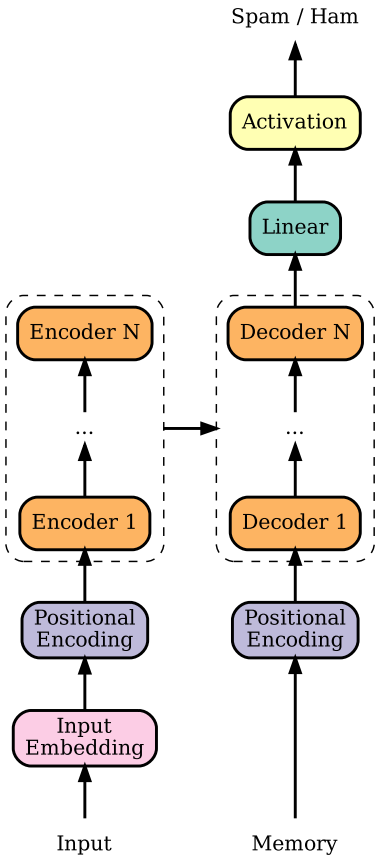

FIGURE 3. Structure of proposed modified Transformer model for SMS spam detection. The input messages embeddings and memory (trainable parameters) are positional encoded, respectively. Then, the processed message vectors are passed to encoder layers, where the self-attention is performed. The results of encoder layers are passed to decoder layers. In decoder layers, the Multi-Head Attention is executed based on the results of encoder layers and the processed memory. Then, the decoded vectors are sent to some fully-connected linear layers, followed by a final activation function for classification.

A. MEMORY

The first modification for the SMS spam detection task is the introduction of memory. Since there is no output sequence (target sequence) in the SMS spam detection task, we used a list of trainable parameters named "memory" to be the substitute for output sequence embedding. The length of the memory is a configurable hyper-parameter. Each element of the memory is a vector of dimension d_{model} so that it can be adapted to the Transformer model without any extra projection. In other words, the memory is a matrix of dimension $len_{memory} \times d_{model}$. The output embedding layer in the original Transformer model is also removed since there are no target sequence texts anymore to be mapped to numeric vectors. Similar to the output sequence in the vanilla Transformer model, the positional information is injected into the memory at the positional encoding layer before being fed into decoders.

During the training process, the parameters of memory are trained, and the memory matrix is expected to contain the important information that can help to predict whether or not a message is a spam. Therefore, in the decoders of the modified spam Transformer model, with the help of the attention mechanism, the memory can contribute to locate the significant part of the output sequence of the encoder stack that summarized the message, and eventually help to classify the spam SMS messages.

B. LINEAR LAYERS AND FINAL ACTIVATION FUNCTION

The second modification is the final activation function. In the vanilla Transformer, the dimension of outputs of decoder layers is $T \times d_{model}$, where T is the target sequence length and d_{model} is the model size (number of features). Therefore, intuitively, it is a promising approach to use the linear layers to map the output to a vector that has the same dimension as the number of words in the dictionary and apply a softmax function on the vector to find the closest candidate word from the dictionary.

However, the SMS spam detection task is a binary classification problem. Therefore, to convert the output from the decoder stacks with dimension d_{model} into a single probability of the message being spam, the linear layers after the decoders are also modified. Instead of mapping the output of the decoder stack to a vector, the linear layer in the modified Transformer model for SMS spam detection has only one single neuron in the last layer. Thus, the outputs of the decoder stack are converted into a single numeric probability value.

Additionally, the final activation function needs to be replaced with a function that can map the result to a binary outcome. Thus, in the modified Transformer for SMS spam detection, a sigmoid function, which is defined in

Equation (2), as the final activation function, is applied to the output of the linear layers after decoders, generating a binary result that predicts whether or not the message is spam.

C. DROPOUT

Dropout [37] is a powerful technique published by Hinton *et al.* in 2012 in order to prevent over-fitting in a large feed-forward neural network. Concretely, the Dropout refers to randomly omit some nodes in those large feed-forward layers on each specific training case. The modified spam Transformer model that we proposed employs multiple feed-forward layers. Thus, the Dropout technique is also implemented in the feed-forward layers of our spam Transformer model. Besides, the Dropout technique is also used in positional encoding and calculation of attention function.

D. BATCHES AND PADDING

During each epoch of training on our proposed models, the whole training set is divided into multiple batches. As the length of the message with the same batch should be the same, some padding words (empty words) should be added into the shorter message vectors, interfering with the detection to some extent. Therefore, the algorithm of dividing the training set into batches is designed to minimize the padding words. Specifically, the training data is sorted by the message length first, and the batches are created to minimize the padding words based on the sorted messages.

Admittedly, adding padding words may pose a negative influence on the model. However, using batch has been proved to be a good idea for model training as it increases the training speed extraordinarily. In fact, a larger batch size accelerates a ton for the training speed. Additionally, the negative influence of padding words is addressed by minimizing the use of padding words. Besides, the padding masks are also passed into the model along with the training batches so that the Transformer model can ignore the padding words during training.

E. OPTIMIZATION AND LEARNING RATE

The gradient descent is employed to optimize our modified spam Transformer model. The main idea of the gradient descent algorithm is to minimize the loss function of the model by updating the parameters along the opposite way of the gradient to the loss function, where the gradient is the partial derivatives of the loss function of the parameter. There are plenty of variant optimizers of gradient descent. We use the AdamW [38] optimizer for our proposed modified spam Transformer with $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\epsilon = 10^{-9}$. Learning rate is a critical hyper-parameter in machine learning. It is defined as the step size of updating parameters, which basically represents the speed of learning of the model. Having the learning rate set too high will lead to the situation that the model fails to locate the best parameters (weights and biases), while a learning rate that is too small sticks the model around the local optimal point rather than finding a better parameter solution. For the modified spam Transformer model, the same way of determining the learning as mentioned in [3] is utilized. The learning rate lr first increases linearly until reaching the $warmup\_steps$ steps and then decreases proportionally to the square root of the step numbers. Concretely, we used $warmup\_steps = 8000$.

F. DATAFLOW OF MODIFIED TRANSFORMER

As is shown in Figure 3, the input messages are first converted into word embeddings using the Glove model. Following this, the memory (trainable parameters) and the embeddings of the input sequence are positionally encoded, respectively. Then, the processed message vectors are passed to encoder layers, where the multi-head self-attention is performed and the important parts of the input sequence are given larger weights. The results of encoder layers are passed to decoder layers. In decoder layers, the multi-head self-attention is computed on the memory. After that, the multi-head attention is executed based on the results of encoder layers and the processed memory. Finally, the decoded vectors are sent to some fully-connected linear layers, followed by a final activation function for classification.

IV. EXPERIMENT

A. DATASETS

In the experiments, two different datasets are utilized. The first dataset is SMS Spam Collection v.1 [13] dataset, which is labeled SMS messages dataset collected for mobile phone message research. The second one is UtkMl's Twitter Spam Detection Competition (UtkMl's Twitter) [39] from Kaggle. Table 1 shows the overview statistics of the two datasets.

TABLE 1. The statistics of two datasets.

	SMS Spam Collection v.1	UtkMl's Twitter
Spam	747	5815
Ham	4827	6153
Total	5574	11968

Although the Twitter posts are not precisely the same as the SMS messages, they are still in some ways common. For instance, they both have approximately less than 100 words. People tend to use more casual language and abbreviations in both Twitter posts and SMS messages. Therefore, UtkMl's Twitter dataset can also be used to test our model. Besides, we can also analyze the extensibility of our model by comparing the performance of our model on these two datasets.

In comparison with SMS Spam Collection v.1 [13] dataset, UtkMl's Twitter dataset contains more data in both spam and ham classes. Besides, UtkMl's Twitter dataset is balanced since the number of spam messages and ham messages are approximately equal. In terms of the language, although they are a lot of casual language and abbreviation used in both datasets, casual language and abbreviation appear more frequently in UtkMl's Twitter dataset. The reason for this observation may be the feature of the Twitter posts. Alternatively, it could also because of the date that the dataset was collected, as SMS Spam Collection v.1 was published in 2011.

TABLE 2. The confusion matrix.

	Predicted Spam	Predicted Ham
Actual Spam	True Positive (TP)	False Negative (FN)
Actual Ham	False Positive (FP)	True Negative (TN)

B. EVALUATION MEASURES

In order to evaluate the performance of the proposed modified spam Transformer model, some metrics such as accuracy, precision, recall, and F1-Score are used in the experiments. All these metrics are calculated based on the confusion matrix. As is mentioned in the previous section, the spam messages in the SMS Spam Collection v.1 dataset are significantly less than the ham messages, which means that the dataset is unbalanced. Therefore, the accuracy is not sufficient as a measurement to evaluate the performance of the proposed model, and the F1-Score is employed in the experiments. The accuracy, precision, recall, and F1-Score is defined as follows:

$$Accuracy = \frac{TP + TN}{TP + FP + FP + FN} \tag{4}$$

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{7}$$

The precision, also known as the positive predictive value, represents the percentage of the predicted positive cases that are actually positive, meaning the possibility that the classifier is correct given that it predicts positive. The recall, also known as sensitivity, denotes the number of true positives instances divided by the number of actual positive instances, which can also be described as the percentage of the positive cases that are identified successfully. The F1-Score is the harmonic mean of precision and recall, which measures the performance of a classifier in terms of precision and recall in a balanced way.

C. DATA SPLITTING

For the traditional machine learning approaches, the data is divided into training set (70%), and test set (30%). For the LSTM and our proposed modified spam Transformer model, the data is split into training set (50%), validation set (20%), and test set (30%), where the validation set is used after each epoch of training to help us select the best model and perform early stopping to avoid over-fitting.

D. DATA PRE-PROCESSING

The textual messages in the dataset are first tokenized. Tokenization refers to the task of splitting textual into meaningful words. Specifically, the SpaCy [40] library is employed for data pre-processing in order to tokenize the data.

After that, the numeric representation vectors (word embeddings) are calculated based on the textual messages. There are two major methods of calculating representation vectors are employed in our experiments.

- **TF-IDF Representation**: The TF-IDF (Term Frequency–Inverse Document Frequency) is a widely-used numerical statistic in NLP. It is designed to reflect the importance of a word to a document in the given text corpus. The Term Frequency (TF) is defined as the number of times that a term occurs in a document. A larger TF means the term is referred for more times in the given document, showing that the term is more relevant to the document. There are multiple different means to weigh the TF in order to adapt it in different applications. In our experiment, we use the raw count of the term in the document as the TF. The Inverse Document Frequency (IDF) is a value to qualify the specificity of a term, which is normally defined as the logarithmically scaled inverse fraction of the number of documents that contain the term. In another word, when a term occurs in a great number of documents, the IDF is numerically low, leading to a low TF-IDF. For instance, the term "the" occurs in almost every English document, leading to a document frequency of almost 1 (100% of the documents in the corpus contain the term "the"). Thus, the IDF of "the" is close to 0, which means that its importance to any documents in the corpus is low.
- **GloVe Representation**: GloVe [41] is an unsupervised learning algorithm for obtaining vector representations for words. The main idea is to map words into a meaningful space where the distance between words is related to semantic similarity. GloVe produces a vector space with a meaningful substructure, and it can also find the relations like synonyms between words.

In our experiments, for the deep learning approaches such as LSTM and our proposed spam Transformer model, the GloVe model is employed to create representation vectors for them. Specifically, in our experiments, we used the "glove.840B.300d", a pre-trained model with 2.2 million words in the dictionary that converts textual data into 300-dimensional vectors. For benchmark machine learning algorithms, although the vectors generated by GloVe model have more dimensions and theoretically contain more information, presumably due to the limitation of traditional machine learning classifiers, the TF-IDF representation performs better in practice. Therefore, TF-IDF representation is used for calculating representation vectors in benchmark machine learning algorithms.

E. LOSS FUNCTION

The loss function we used for deep learning approaches including LSTM and modified spam Transformer is Binary Cross Entropy function, which is defined as follow:

$$l(x_i, y_i) = -w_i[y_i \cdot logx_i + (1 - y_i) \cdot log(1 - x_i)] \tag{8}$$

The weight w_i is the rescaling factor for loss. Since the SMS Spam Collection v.1 is unbalanced, where spam

messages are severely less than ham (legitimate) messages, a larger weight is given to the actual spam messages to counteract the negative effect of the unbalanced dataset. The rescaling weight is calculated based on the ratio between the number of ham messages and spam messages.

F. MODEL TRAINING

We trained our experiment models on NVIDIA GeForce RTX 3090 GPU. For the machine learning classifiers, the experiments are performed on the Scikit-learn 0.24.0 [42] environment. For deep learning approaches like LSTM and spam Transformer model, the experiments are conducted on the Ubuntu 20.04 LTS, CUDA 11.1, and PyTorch 1.7.1 [43] environment. The early stopping technique is implemented to fight against the over-fitting. Besides, we also trained and tested the CNN-LSTM SMS spam detection model proposed in [22] on both datasets as a benchmark to evaluate our modified spam Transformer model.

G. HYPER-PARAMETERS TUNING

In order to tune the models and find the best hyper-parameters set, the Ray Tune [44] library is employed. The Ray Tune is a hyper-parameter tuning extension tool that supports multiple machine learning frameworks. Given a candidate hyper-parameters set, the Ray Tune can find the optimized hyper-parameters set by training multiple models with different settings and comparing the results automatically. In our experiments, with the help of the Ray Tune, we first explored optimal settings for the overall architectural hyper-parameters such as Encoder layers, Decoder layers, and Model size. After that, other hyper-parameters such as the rate of dropout and Feed-forward layer size are tuned under the candidate optimal model settings.

For the LSTM model, the optimized parameters on both datasets are shown in Table 3. For our modified spam Transformer model on SMS Spam Collection v.1, Table 4 presents the initial hyper-parameters that we started from and the optimized values when the better result was achieved after tuning. Table 5 demonstrates the initial as well as the optimized hyper-parameters of modified spam Transformer on UtkMl's Twitter dataset.

TABLE 3. Optimized hyper-parameters for LSTM.

Hyper-parameter	SMS Spam Collection v.1	UtkMl's Twitter
LSTM units per layer	100	100
LSTM layers	1	2
LSTM Dropout	0.1	0.1
Linear Dropout	0.4	0.1

V. RESULTS AND ANALYSIS

A. EVALUATION

We demonstrate the performance of the modified spam Transformer model by comparing it on two datasets with some other typical spam detection classifiers, including Logistic Regression, Naïve Bayes, Random Forests, Support Vector Machine (classifier), and Long Short-Term Memory. Besides, for the SMS Spam Collection v.1 dataset, we also compare our models with the CNN-LSTM approaches in [22], since they aim to solve the same problem on the same dataset with us.

Table 6 summarizes the results on SMS Spam Collection v.1 dataset. For accuracy, our modified spam Transformer model achieved the best value of 98.92%. Concerning precision, the best score was from the Random Forests classifier with a value of 1.0, and our proposed spam Transformer got a value of 0.9781. When it comes to recall, the optimal result came from the spam Transformer model with a value of 0.9451, and the same value came from the Naïve Bayes classifier as well. Finally, in terms of F1-Score, our spam Transformer also achieved the best value of 0.9613. The experiment of CNN-LSTM [22] that was conducted

TABLE 4. Initial and optimized hyper-parameters for modified spam Transformer on SMS Spam Collection v.1.

Hyper-parameter	Initial	Optimized
Encoder layers	6	6
Decoder layers	6	6
Model size	512	600
Feed-forward size	2048	1200
Attention head size	8	8
Transformer Dropout	0.1	0.01
Linear Dropout	0.1	0.05

TABLE 5. Initial and optimized hyper-parameters for modified spam Transformer on UtkMl's Twitter.

Hyper-parameter	Initial	Optimized
Encoder layers	6	2
Decoder layers	6	4
Model size	600	600
Feed-forward size	1200	1200
Attention head size	8	8
Transformer Dropout	0.01	0.01
Linear Dropout	0.05	0.1

TABLE 6. Results obtained on SMS Spam Collection v.1.

Classifiers	Accuracy	Precision	Recall	F1-Score
Logistic Regression	98.56%	0.9863	0.9113	0.9473
Naïve Bayes	98.38%	0.9411	**0.9451**	0.9431
Random Forests	97.90%	**1.0**	0.8535	0.9209
SVM	98.62%	0.9908	0.9113	0.9494
LSTM	98.56%	0.9570	0.9409	0.9489
CNN-LSTM [22]	97.66%	0.9159	0.9198	0.9178
Spam Transformer	**98.92%**	0.9781	**0.9451**	**0.9613**

TABLE 7. Results obtained on UtkMl's Twitter.

Classifiers	Accuracy	Precision	Recall	F1-Score
Logistic Regression	81.51%	0.8441	0.7615	0.8007
Naïve Bayes	83.21%	0.8316	0.8221	0.8269
Random Forests	79.28%	0.8449	0.7044	0.7683
SVM	82.68%	0.8681	0.7604	0.8107
LSTM	81.04%	0.8594	0.7307	0.7898
CNN-LSTM [22]	79.45%	0.8182	0.7438	0.7792
Spam Transformer	**87.06%**	**0.8746**	**0.8576**	**0.8660**

TABLE 8. The confusion matrices on SMS Spam Collection v.1.

	Logistic Regression		Naïve Bayes		Random Forests		SVM		LSTM		CNN-LSTM		Spam Transformer	
Act. \ Pred.	**Spam**	**Ham**	**Spam**	**Ham**	**Spam**	**Ham**	**Spam**	**Ham**	**Spam**	**Ham**	**Spam**	**Ham**	**Spam**	**Ham**
Spam	216	21	224	13	204	35	216	21	211	26	218	19	224	10
Ham	3	1433	14	1422	0	1434	2	1434	17	1419	20	1416	5	1431

TABLE 9. The confusion matrices on UtkMl's Twitter.

	Logistic Regression		Naïve Bayes		Random Forests		SVM		LSTM		CNN-LSTM		Spam Transformer	
Act. \ Pred.	**Spam**	**Ham**	**Spam**	**Ham**	**Spam**	**Ham**	**Spam**	**Ham**	**Spam**	**Ham**	**Spam**	**Ham**	**Spam**	**Ham**
Spam	1332	417	1438	311	1232	517	1330	419	1278	471	1301	448	1500	249
Ham	246	1592	291	1547	226	1612	202	1636	209	1629	289	1549	215	1623

by Ghourabi *et al.* on the same dataset, are also included in Table 6. In Table 8, we demonstrate the confusion matrix of all the approaches that we tested in the experiments on SMS Spam Collection v.1 dataset.

Table 7 summarizes the results on UtkMl's Twitter dataset. The modified spam Transformer model outperformed all other candidates in all four aspects that we tested with the values of 87.06%, 0.8746, 0.8576, and 0.8660 on the accuracy, precision, recall, and F1-Score, respectively. The confusion matrix of the modified spam Transformer model on UtkMl's Twitter is presented in Table 9.

B. ANALYSIS

Although the experimental results show an improved performance of the proposed spam Transformer model compared to other candidates, the false predictions also indicate the drawback of the proposed model. We analyzed the content of the false prediction samples including false positive and false negative samples and found that there were a great number of the *UNK* marks in the data passed to the model, which is produced because the words are never seen in the training data. In other words, the unknown words obstruct the model from understanding the messages. Besides, the SMS messages are usually short, which increases the influence of every single word and makes the unknown words more influential. Actually, due to the unknown words, the model did not have enough information to detect spams in many false prediction cases.

Though our proposed model performs better than other candidate algorithms on UtkMl's Twitter dataset, the results are still not as good as that in case of SMS Spam Collection v.1 dataset. From our observation, the major cause is also the unknown words. Compared to SMS Spam Collection v.1 dataset, there are more casual language and abbreviations in UtkMl's Twitter dataset, which may be caused by the feature of Twitter posts or the date of collection of the dataset, as is discussed in Section IV-A. Therefore, the negative influence from casual language and abbreviation is more severe on UtkMl's Twitter dataset, and that is the major cause of more unknown words and eventually worse performance from our perspective.

In addition, Table 8 and Table 9 show the excellent robustness of our model to classify both the spams and hams effectively on no matter balanced (UtkMl's Twitter) or unbalanced (SMS Spam Collection v.1) datasets.

VI. CONCLUSION

In this paper, we proposed a modified Transformer model that aims to identify SMS spam. We evaluated our spam Transformer model by comparing it with several other SMS spam detection approaches on the SMS Spam Collection v.1 dataset and UtkMl's Twitter dataset. The experimental results show that, compared to Logistic Regression, Naïve Bayes, Random Forests, Support Vector Machine, Long Short-Term Memory, and CNN-LSTM [22], our proposed spam Transformer model performs better on both datasets.

On the SMS Spam Collection v.1 dataset, our spam Transformer has a better performance in terms of accuracy, recall, and F1-Score compared to other classifiers. Specifically, our modified spam Transformer approach accomplished an exceeding result on F1-Score.

Additionally, on the UtkMl's Twitter dataset, the results from our modified spam Transformer model demonstrate its improved performance on all four aspects in comparison to other alternative approaches mentioned in this paper. Concretely, our spam Transformer does exceptionally well on recall, which contributes to a distinct F1-Score.

VII. FUTURE WORK

Although the experimental results in this paper have shown an improvement of our proposed spam Transformer model in comparison with some previous approaches on SMS spam detection, we still believe that there is great potential in the model we proposed.

Firstly, since our current two datasets contain only thousands of messages, in the future, we plan to extend our spam Transformer model to a larger dataset with more messages or even other types of content, for the purpose of better performance.

Besides, in our proposed model, we flattened the outputs from decoders and applied linear fully-connected layers before applying the final activation function and getting the prediction. We believe that some dedicated designs or

implementations instead of simple flattening and linear layers could absolutely boost the performance, which would be one of the most important future works.

Additionally, although the experimental results show that our modified model based on the vanilla Transformer performs well on SMS spam detection and confirms the availability of the Transformer on this problem, the model is still far from optimal. There are some improved models based on the Transformer with more complex architecture such as GPT-3 [4] and BERT [5] that could be explored in the future. Specifically, the BERT seems to be a promising starting point of future work as it has fewer features and is easier to be fine-tuned.

Finally, as is discussed in Section V-B, the proposed model is severely influenced by the unknown words in many cases of false prediction. To address this problem, more data pre-processing techniques could be applied. For instance, a larger vocabulary with more words could be a good option, and some semantic operations such as replacing unknown words with their synonyms could also be explored. Besides, there are some other data-preprocessing and feature extraction techniques that could be done, such as the extraction and analysis of the abbreviation, URLs, tags, or emoji in data.

REFERENCES

[1] P. K. Roy, J. P. Singh, and S. Banerjee, "Deep learning to filter SMS spam," *Future Gener. Comput. Syst.*, vol. 102, pp. 524–533, Jan. 2020.
[2] G. Jain, M. Sharma, and B. Agarwal, "Optimizing semantic LSTM for spam detection," *Int. J. Inf. Technol.*, vol. 11, no. 2, pp. 239–250, Jun. 2019.
[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5999–6009.
[4] T. B. Brown *et al.*, "Language models are few-shot learners," 2020, *arXiv:2005.14165*. [Online]. Available: http://arxiv.org/abs/2005.14165
[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, vol. 1, Jun. 2019, pp. 4171–4186.
[6] G. Sonowal and K. S. Kuppusamy, "SmiDCA: An anti-Smishing model with machine learning approach," *Comput. J.*, vol. 61, no. 8, pp. 1143–1157, Aug. 2018.
[7] J. W. Joo, S. Y. Moon, S. Singh, and J. H. Park, "S-detector: An enhanced security model for detecting Smishing attack for mobile computing," *Telecommun. Syst.*, vol. 66, no. 1, pp. 29–38, Sep. 2017.
[8] S. Mishra and D. Soni, "Smishing detector: A security model to detect Smishing through SMS content analysis and URL behavior analysis," *Future Gener. Comput. Syst.*, vol. 108, pp. 803–815, Jul. 2020.
[9] C. Li, L. Hou, B. Y. Sharma, H. Li, C. Chen, Y. Li, X. Zhao, H. Huang, Z. Cai, and H. Chen, "Developing a new intelligent system for the diagnosis of tuberculous pleural effusion," *Comput. Methods Programs Biomed.*, vol. 153, pp. 211–225, Jan. 2018.
[10] T. K. Ho, "Random decision forests," in *Proc. Int. Conf. Document Anal. Recognit. (ICDAR)*, vol. 1, 1995, pp. 278–282.
[11] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.
[12] M. Gupta, A. Bakliwal, S. Agarwal, and P. Mehndiratta, "A comparative study of spam SMS detection using machine learning classifiers," in *Proc. 11th Int. Conf. Contemp. Comput. (IC3)*, Aug. 2018, pp. 1–7.
[13] T. A. Almeida, J. M. G. Hidalgo, and A. Yamakami, "Contributions to the study of SMS spam filtering: New collection and results," in *Proc. 11th ACM Symp. Document Eng.*, Sep. 2011, pp. 259–262.
[14] A. K. Jain and B. B. Gupta, "Rule-based framework for detection of Smishing messages in mobile environment," *Procedia Comput. Sci.*, vol. 125, pp. 617–623, 2018.
[15] W. W. Cohen, "Fast effective rule induction," in *Machine Learning Proceedings*, 1995, pp. 115–123.
[16] J. Cendrowska, "PRISM: An algorithm for inducing modular rules," *Int. J. Man-Machine Stud.*, vol. 27, no. 4, pp. 349–370, Oct. 1987.
[17] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Ann. Statist.*, vol. 29, no. 5, pp. 1189–1232, Oct. 2001.
[18] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proc. COMPSTAT*. Physica-Verlag, 2010, pp. 177–186.
[19] T. Mikolov, K. Chen, G. S. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proc. Int. Conf. Learn. Represent.*, 2013.
[20] G. A. Miller, "WordNet: A lexical database for English," *Commun. ACM*, vol. 38, no. 11, pp. 39–41, 1995.
[21] H. Liu and P. Singh, "ConceptNet — A practical commonsense reasoning tool-kit," *BT Technol. J.*, vol. 22, no. 4, pp. 211–226, Oct. 2004.
[22] A. Ghourabi, M. A. Mahmood, and Q. M. Alzubi, "A hybrid CNN-LSTM model for SMS spam detection in arabic and English messages," *Future Internet*, vol. 12, no. 9, p. 156, Sep. 2020.
[23] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986.
[24] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994.
[25] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proc. 30th Int. Conf. Mach. Learn. (ICML)*, 2013, pp. 2347–2355.
[26] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[27] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN Encoder–Decoder for statistical machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 1724–1734.
[28] J. Koutník, K. Greff, F. Gomez, and J. Schmidhuber, "A clockwork RNN," in *Proc. 31st Int. Conf. Mach. Learn. (ICML)*, vol. 5, 2014, pp. 3881–3889.
[29] C. Zhou, C. Sun, Z. Liu, and F. C. M. Lau, "A C-LSTM neural network for text classification," 2015, *arXiv:1511.08630*. [Online]. Available: http://arxiv.org/abs/1511.08630
[30] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 4, Sep. 2014, pp. 3104–3112.
[31] R. Prabhavalkar, K. Rao, T. N. Sainath, B. Li, L. Johnson, and N. Jaitly, "A comparison of sequence-to-sequence models for speech recognition," in *Proc. Interspeech*, Aug. 2017, pp. 939–943.
[32] S. Venugopalan, M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, and K. Saenko, "Sequence to sequence–video to text," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 4534–4542.
[33] D. Bahdanau, K. H. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, 2015.
[34] K. Xu, J. L. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *Proc. 32nd Int. Conf. Mach. Learn.*, vol. 3, 2015, pp. 2048–2057.
[35] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2015, pp. 1412–1421.
[36] E. S. D. Reis, C. A. D. Costa, D. E. D. Silveira, R. S. Bavaresco, R. D. R. Righi, J. L. V. Barbosa, R. S. Antunes, M. M. Gomes, and G. Federizzi, "Transformers aftermath," *Commun. ACM*, vol. 64, no. 4, pp. 154–163, Apr. 2021.
[37] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," 2012, *arXiv:1207.0580*. [Online]. Available: http://arxiv.org/abs/1207.0580
[38] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2017, *arXiv:1711.05101*. [Online]. Available: http://arxiv.org/abs/1711.05101
[39] *UtkMl's Twitter Spam Detection Competition | Kaggle*, UtkMl.
[40] M. Honnibal, I. Montani, S. Van Landeghem, and A. Boyd, "spaCy: Industrial-strength natural language processing in python," 2020, doi: 10.5281/zenodo.1212303.
[41] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global vectors for word representation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 1532–1543.

[42] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Nov. 2011.
[43] A. Paszke *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. 2019, pp. 8024–8035.
[44] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez, and I. Stoica, "Tune: A research platform for distributed model selection and training," 2018, *arXiv:1807.05118*. [Online]. Available: http://arxiv.org/abs/1807.05118

XIAOXU LIU received the bachelor's degree in computer science and technology from the Nanjing University of Posts and Telecommunications, China, and the master's degree in computer science program from the University of Ottawa, Canada, in 2019. His research interests include machine learning, deep learning, and natural language processing.

HAOYE LU (Member, IEEE) received the joint B.Sc. degree in computer science and mathematics, in 2017, and the master's degree in computer science, in 2019. In 2013, he joined the University of Ottawa, Canada. He is currently working as a Research Associate. His research interests include artificial intelligence and network structures.

AMIYA NAYAK (Senior Member, IEEE) received the B.Math. degree in computer science and combinatorics and optimization from the University of Waterloo, Canada, in 1981, and the Ph.D. degree in systems and computer engineering from Carleton University, Canada, in 1991. He is currently a Full Professor with the School of Electrical Engineering and Computer Science, University of Ottawa. He has over 17 years of industrial experience in software engineering, avionics and navigation systems, and simulation and system level performance analysis. His research interests include software-defined networking, mobile computing, wireless sensor networks, and vehicular ad hoc networks.

He has served on the Editorial Board of several journals, including IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, *International Journal of Parallel, Emergent and Distributed Systems*, *Journal of Sensor and Actuator Networks*, and *EURASIP Journal on Wireless Communications and Networking*. He is currently serving on the Editorial Board of the IEEE INTERNET OF THINGS JOURNAL, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE OPEN JOURNAL OF THE COMPUTER SOCIETY, *Future Internet*, and *International Journal of Distributed Sensor Networks*.