DeepSentiment: Finding Malicious Sentiment in Online Social Network based on Dynamic Deep Learning

Putra Wanda[1,2], Huang Jin Jie[1], *Member, IAENG*

***Abstract*—Current Online Social Network (OSN) needs real-time and adaptive security model. The tremendous success of deep learning algorithms at computer vision tasks in recent years inspires us to adopt the method. It is becoming increasingly popular for various applications include in OSN security and privacy-preserving. In this paper, we propose DeepSentiment, a dynamic deep learning model to detect and classify malicious sentiment in OSN. Different from conventional CNN, we introduce RunPool, a dynamic pooling function to train the sentiment features. By using the function, we find a significant increase in the graph's performance with the DeepSentiment CNN model. Demonstrated by the experiment, we harvest a higher accuracy and small loss in malicious sentiment classification with the benchmark dataset.**

***Index Terms*—Neural Networks, Malicious Sentiment, Dynamic Deep Learning, Online Social Network**

I. Introduction

CURRENTLY, almost each of the devices has a security option to unlock and access the device, such as using a PIN, a password, keyboard patterns. The current security techniques put user data at risk because there are no additional security checks performed after the device unlocked or login in the application environment. Unauthorized people may able to crack the simple passwords or PIN of mobile phone or wearable devices because of security weaknesses [1]. However, common security technique as cryptography is no longer suitable for the dynamic environment.

The rapid growth of Social Network has a potential problem in security and privacy problem. The OSN environment remains risky and becomes a target of attacking the public network such as the internet. Based on a survey, users are more preferred in social existence, flow, and self-disclosure than security. It is serious issues for data privacy in the popular application [2]. As a primary research area in OSN, Some explorations have been proposed to solve OSN security and privacy issues [39][41][44].

Many users utilize public OSN for daily activities, but the primary concern in these applications is privacy and security. In the interacting process, the problem appears when the user can see the received messages of some online users without any registration [3]. Unluckily, Telegram, one of the famous OSN, the most encrypted messenger still remains a weakness. A study reveals a technique that may able to reconstruct the data log that send or received by the user [4].

Traditionally, authentication is one of the most critical security services in computer application The conventional model like onetime authentication remains an issue. There are password and public/private key authentication as the two most popular approaches. The method including public-key cryptography for authentication [6] requires a large computations memory and time. Computational overhead is still become the main concerns for public key security. Therefore, it needs an efficient technique for authenticating OSN users after logging in to the system. Several studies propose a security model to address the issue by conducting continuous authentication [7].

However, the conventional model puts private data at risk when user attacked after logged in to a system and attacker pretends as the real user. Another weakness is some public OSN transmit user data in plaintext over the public network [5]. Most of OSN does not implement continuous privacy preserving in the communication process. Luckily, continuous authentication becomes a hot topic for several years. Many studies explore the security model in the mobile context. For example, a paper proposes continuous authentication by collecting and using biometric information to classify a genuine or fake user [8].

In recent years, deep learning, specifically Convolutional Neural Networks [9] are becoming increasingly popular in solving various applications., The neural networks have achieved state-of-the-art results on a variety of challenging problems in computer vision [10] [11]. The CNN runs the computation by using the convolutional operation in the hidden layer. By a set of filters, it computes the output feature maps by convolving the feature maps of the previous layer.

In this paper, we introduce DeepSentiment, a dynamic deep learning model to detect and classify malicious sentiment on the OSN. We assume that better detection of malicious sentiment can act as active authentication for the OSN. To convince the model, we provide experiments result. We will summarize the main innovations and contributions of our work, particularly in the OSN malicious sentiment problem as follows:

Manuscript received November 1, 2018; revised March 27, 2018.

Huang Jin Jie (Corresponding Author) / [1] Harbin University of Science & Technology. He is Professor and Ph.D. Supervisor in School of Computer Science and Technology; His current research interests include artificial intelligence and control system (huangjinjie163@163.com).

Putra Wanda/ ([2] University of Respati Yogyakarta). He is Ph.D. student in School of Computer Science and Technology, Harbin University of Science and Technology, Harbin. China.

1. We demonstrate that DeepSentiment is more efficient than several learning methods. Instead of using conventional CNN, we develop a new architecture of the neural network by adopting a dynamic model of deep learning architecture.
2. We propose a dynamic pooling method for training the neural network. It is to calculate and detect the malicious sentiment on the OSN. Hence, it can classify malicious content based on the message sentiment analysis. The pooling manipulation model achieves state-of-the-art performance in malicious sentiment problem.
3. We show analysis of the dynamic pooling function can improve state-of-the-art on the benchmark an OSN dataset. To convince model classification performance, we undergo metric evaluation at the end of this experiment.

Organization: The paper will discuss the introduction of sentiment analyses in Chapter I, related works in Chapter II. Chapter III describes the proposed model of CNN and Chapter IV and Chapter V discusses the experiment result and analysis of this experiment with detail explanation. In Chapter VII, we describe the conclusion and future research directions.

II. Related Works

OSN applications have grown quickly and significantly. Unfortunately, it remains anomalous nodes issue. The anomalous user can spread malicious software or threat, such as viruses, malware, and so forth over the network. Infected users will spread malicious software automatically by sending fake requests to other users. Some studies present a method to detect OSN activities by constructing a community detection algorithm [15] or building message classifier with Naive Bayes [42].

Various research has proposed a diverse OSN protection strategy. A study proposes a model by adding a secure module and applying a hash algorithm to maintain the path in transceiver and routing modules [12]. The paper utilizes the hash algorithm to secure network conversation and to produce a private environment. Not just using the hash protocol, a study constructs security strategy by using the group authentication model. It is to authenticate all users simultaneously within the group. Instead of using one-to-one authentication, it adopts many-to-many authentication [13]. Another model presents an organizing scheme in the OSN based on trust chain model [14]

In OSN group environment, there are critical elements including privacy, authenticity, integrity, and non-repudiation as the important requirements in the system. Thus, a study implements common cryptography like Elliptic Curve Cryptosystem (ECC) to build a security scheme. In the model, an agent creates connectivity anytime, anywhere, any device [16]. To increase the OSN security level, a paper employs multiprotocol for OSN. It combines various protocol including end-to-end encryption and off the record messaging protocol [17].

In other applications of OSN, a study proposes Mobile Healthcare Social Networks (MHSNs) security to solve conflicting privacy concerns on protecting individual symptoms from strangers. In the process, a similar symptom matching process to achieve personal health information (PHI) sharing [18]. Besides, a paper introduces a game-theoretic framework to model interactions among user in an OSN. In this framework, the interaction process may influence decisions to conduct privacy protection [19].

A common technique to address the threat in OSN is utilizing cryptography. However, the conventional technique is no longer suitable for the OSN dynamic environment. Cryptography is a conventional technique to construct information security [20], It converts the text (plaintext), random text (ciphertext) or vice-versa [21]. Authentication is a process when people and the application are authorized in a system. Diverse papers introduce various methods to get an efficient authentication process. They present key agreement scheme to provide secure roaming services information [22], to address the problem of quickly detecting intrusions with lower false detection rates [23].

Nowadays, multimodal continuous authentication is one of the more promising authentication methods. As systems begin to support this model of security, users do not need to memorize their login passwords or tokens, and system administrators feel more confident when using their accounts. For several years, continuous authentication or active authentication has been a hot topic of studies, but research in the mobile context has only recently still grown. Current digital interaction is necessary for real-time authentication in many research areas [24].

For example, the Internet of Things as one of the hottest topics in computer science. It needs an efficient model for the authentication process. The future small smart devices lack the conventional interfaces used for authentication (such as keyboards, mice, and touchscreens). So, a deep study needs to ensure how can users be authenticated and authorized continuously. These issue deal with Continuous authentication mechanisms [25]. In a mobile environment, the studies introduce continuous authentication by face recognition, gait, profiling behavior, and other approaches such as device movement and the ambient noise [26] [27].

Modern authentication method utilizes learning algorithms as the classifier. Deep learning is becoming increasingly popular in solving various applications, one of them is the authentication process. It is a part of machine learning algorithms which imitate the structure and function of the brain. To construct a protection model, a study presents the Deep Belief Networks for authorship verification model (CA). It also implements Gaussian-Bernoulli (DBN) units to model real-valued data [28]

In sentiment analyses problem, Zhang Shan et al [35] propose a technique to construct Bayes classifier and to employ the microblog’s emoticons to build the Chinese sentiment corpus. To improve the performance, it calculates the particular entropy. A paper explores SVM to compute sentiment analysis. It calculates three features including the emoticon, the sentiment lexicon and the hybrid approach over the hierarchical structure. The experiments show the result can achieve a good performance [36].

In a large OSN, millions of users posting millions of messages every day. It requires to know which is normal or malicious sentiment accurately. Hence, current papers present diverse sentiment analysis model based on a deep learning algorithm. A study proposes sentiment measurement with CNN's and SVM. In the paper, they feed the word vector as the input and calculate the CNN model to conduct automatic feature learner. At the final step, they implement the SVM as the text classifier [37]. Another study calculates CNN with multi-Channel Distributed Representation for classifying the Tweets [43].

III. Model of CNN

A. Common CNN

Commonly, CNN has many identical copies of the same neuron. In a CNN, a computational process runs large models' computation with a number of hyper-parameters. CNN has many interleaved convolutional and pooling layers over the network. The layer receives the feature maps and computers feature maps as its output by running convolutional operation. The parameters of the convolution layer called filters. For testing the loss, it needs back-propagation to learn during training the model [8].

In the CNN, there are a forward pass and backward pass. The forward pass computes from the inputs data until the output layers. To obtain the loss function, it traverses through all neurons from first to the last layer. In the Forward Pass process, the learning is usually run in groups of N samples. In this study denoted by x_i^n, the i-th input feature map of sample *n* and y_j^n the j-th output of *n.* In this process of the convolutional layer. The y_j^n computed using the convolutional operator (*). Equation (1) shows forward pass formulation in Conventional CNN networks.

$$y_j^n = \sum_i k_{ij} * x_i^n \qquad (1)$$

In the Forward Pass process, the output feature maps are computed using the summation process. It will calculate the filter k_{ij} convolute with input feature map x_i^n.

A backward pass is computed from the last layer, move back to the first layer. The process employs a gradient descent algorithm or a similar technique. This process will calculate the gradient descent function denoted by k_{ij} in these following formulas. Equation (2), (3), (4) describe how to calculate Backward Pass in Conventional CNN Networks.

$$\frac{\partial l}{\partial x_i^n} = \sum_j \left(\frac{\partial l}{\partial y_j^n}\right) * (k_{ij}) \qquad (2)$$

$$\frac{\partial l}{\partial k_{ij}} = \frac{1}{N}\sum_j \left(\frac{\partial l}{\partial y_j^n}\right) * (x_i^n) \qquad (3)$$

$$k_{ij} = k_{ij} - \alpha \cdot \left(\frac{\partial l}{\partial k_{ij}}\right) \qquad (4)$$

In the above formulas, loss function *l* for the gradient of the network concerning x_i^n calculated with equation (2). Besides, calculating of loss function with k_{ij} computed by the next equation (3) and after calculating $\frac{\partial l}{\partial k_{ij}}$, gradient descent will update parameters k_{ij} by calculating α learning rate shown with equation (4).

B. Dynamic CNN

In this study, we propose a dynamic CNN to measure message sentiment of OSN users. Instead of using a linear parameter, a dynamic CNN computes the CNN layer according to the length of the input matrix in each layer. Dynamic CNN alternates between wide convolution layers with dynamic pooling layers. The dynamic CNN layer receives two inputs within the operation. The first input is the previous layer of the features maps and the second is the filters

We construct a dynamic CNN by formulating a pooling function to get an appropriate k-max pooling value. By using dynamic CNN, it can compute the graph by harnessing the current parameters of each layer. The approach employs dynamic k-max pooling describe features that correspond to suitable features in the neural network layer. We implement the pooling function within the hidden layers. IN the process, the pooling operator is activated after the topmost convolutional layer. It makes the input to the fully connected layers is the independent length of the input message.

We construct an algorithm to calculate the value of K-Max pooling. The model employs the *k* value at intermediate layers (hidden layer). The parameter *k* is a dynamic value to enable the extraction of higher order and longer-range features. The dynamic CNN calculates forward pass with function 5.

$$y_j^n = \sum_i k_{ij}^n * x_i^n \qquad (5)$$

In this process, the first network computes the features maps as the input layer to the dynamic CNN. In Equation (5), $x_i^n$is the i-th input feature map of the sample *n* and k_{ij}^n is the *ij* input kernel of the sample *n*. then, j-th becomes output feature map of sample *n* networks. Operation of backward and forward pass run simultaneously in one iteration**.** The dynamic CNN calculates backward pass with the function (6), (7).

$$\frac{\partial l}{\partial x_i^n} = \sum_j \left(\frac{\partial l}{\partial y_j^n}\right) * (k_{ij}^n) \qquad (6)$$

$$\frac{\partial l}{\partial k_{ij}^n} = \left(\frac{\partial l}{\partial k_j^n}\right) * x_i^n \qquad (7)$$

In the Backward Pass process with Equation (6), the layer will compute the gradient of the *l* (loss function) with respect x_i^n. The values of the gradient calculated by the partial derivative function $\frac{\partial l}{\partial x_i^n}$ and passed to the first layer of the network. Then Equation (7) is used to calculate the gradient of the loss function with respect to k_{ij}^n . Fig. 1 depicts Network topology of Convolutional Neural Network with dynamic k-max pooling to obtain the output.

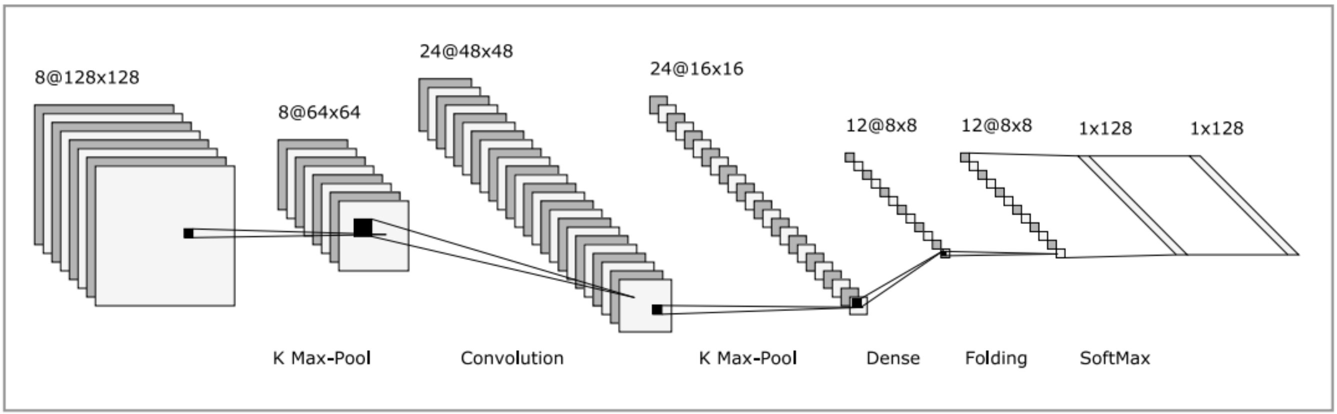


Fig. 1. The DeepSentiment topology of Neural Network with dynamic k-max pooling. In the model, a computational process runs large models' computation with a number of parameters. The network has many interleaved convolutional and pooling layers over the network

Fig 1 depicts the computation process in the hidden layer. The hidden layer runs the convolution by using filters and input matrix. In the model, a set of filters form F ∈ $R^{d\times m}$ will be convolved the input matrix S ∈ $R^{d\times|s|}$. It is to compute feature extraction (word sequences) throughout the training process. In the input layer, network layer projects text extraction as matrices input.

The main contribution of this model is using RunPool pooling function. The RunPool computes the matrices based on the feature extraction process. In RunPool, assume *k* be a function of the length of the input and the depth of the network as the computing model. *l* is some current convolution in the hidden layer, *L* is the total number of convolution layer, k*top* is the fixed max pooling parameter for the topmost convolutional layer.

Table I: Mathematical Notation of RunPool Pooling

Notation	Description
f	filter size
s	stride
l	index of the current layer
k_l	k value of current layer
k_{top}	k value of top layer
k_{map}	k value after calculation of input features
L	total number of layers
C	length of input matrices
p	padding
S_p	Number of padding pixels
	k_{top} and *L* are constants

The model calculates the RunPool in text input data with pooling manipulation function: Hence, we get the "dynamic winning out" by calculating the simple pooling function to obtain *k* value of layer *l* k_l. To get the k_l value, we calculate k_{map} with the following function:

$$k_{map} = \left(\frac{L-l}{L}(C+S_p)\right) \qquad k_l = max(k_{top},\, k_{map}) \qquad (8)$$

The function 8 has task to choose a suitable pooling value, the function calculates network elements to determine the value of *k*. It is to find appropriate k-max value in the hidden layer. We optimize the pooling by re-computing the graph with the k-max pooling manipulation.

Basically, to conduct sentiment analysis, a model should analyze, process, summarize and conclude the subjective texts [33]. Current studies propose a analyses strategy by using machine learning and the rule-based method. As far as we know, machine learning is a better model to classify feature with emotion words input [34]. However, conventional machine learning algorithm requires manual feature selection and training for the dataset, so it is not appropriate with the dynamic environment like Online Social Network.

Inspired by deep learning success, we propose a deep learning algorithm to construct a supervised learning model. This study proposes a CNN with dynamic pooling to detect anomalies sentiment by analyzing OSN message. The model uses wide convolution in each window, to ensures that all weights get the entire parameter action. This study adopts k-max pooling concept to optimize the network performance. Choosing a useful value of dynamic k-max pooling is crucial to achieving the benchmark result.

The key idea of the experiment is to modify the original CNN layer into dynamic pooling computation with RunPool implementation. To construct efficient graph computation, the model presents dynamic pooling computation. We also conduct dynamic graph computation. The graph is not fixed but rather is dynamically updated after each layer of the network. We hypothesize the advantage of the dynamic graph which computed with the concept of "Define by Run" is very efficient for the varying input.

D. RunPool Pooling Manipulation

In this research, we construct RunPool to train the features of message sentiment. In the DeepSentiment, we set the tweet of a message as the input feature. Before training and testing the features, the dataset element such as the character of the word will be converted into binary value as matrix input. The model optimizes the number of parameters in the computation process. We need to tweak the hyper-parameters in the input layer, hidden layer, and an output layer. A good network topology is able to efficiently handle the high dimensionality of raw data. Fig. 2 illustrates the DeepSentiment network to calculate the message sentiment.

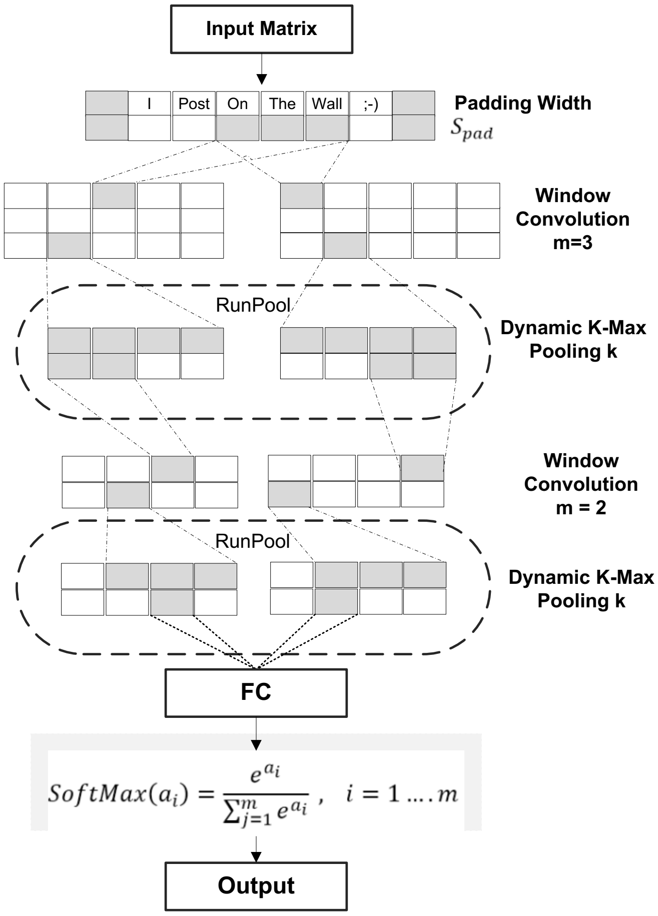


Fig. 2 The dynamic CNN network while calculating the input matrix. It is to produce a classification category and to analyze user's messages

Fig 2 depicts the computation process in the visible and the hidden layer. The hidden layer runs the convolution process by using filters and input matrix (sentence). In the model, a set of filters form F ∈ $R^{d\times m}$ will be convolved the input matrix S ∈ $R^{d\times|s|}$. The aim of the convolutional layer is to compute feature extraction (word sequences) throughout the training process. Consists of three-layer, the input layer, hidden layer, and an output layer. In the input layer, network layer projects text extraction as matrices input. In this phase, the model converts the posting message assigned as word structure become matrix values before the feature extraction process.

In the hidden layer, the network runs pooling operation. The Pooling layer is an operation between the Convolution and ReLU. It is to subtract the number of parameters, such as the size of the image (width and height). In this process, the common technique is Max-pooling operation. Max-pooling obtains the biggest value within a filter and deletes the other values. It takes the strongest activations over a neighborhood. The pooling run in the relative location of a strongly feature. However, instead of using common max pooling, we construct RunPool, a dynamic pooling manipulation to train the sentiment corpus.

The model presents the dynamic k-max pooling concept to compute the feature extractions. The algorithm utilizes a dynamic k-max pooling operation to obtain a suitable feature map in the neural networks. Operation of dynamic k-max pooling run among hidden layer before fully connected layer. Dynamic k-max pooling operator determines the effectiveness of CNN operation. The pooling parameter k can be dynamically chosen by making k a function of other aspects of the network. It retrieves k maximum values from the pooling window.

E. Activation Function

In the hidden layer process, it consists of two primary operations; convolution and pooling. In the convolutional process, the network employs wide convolution filtering to the best local feature in each layer. In pooling layer, it utilizes an activation function to choose the most informative feature. To optimize the network, we test some activation functions to achieve efficient loss and accuracy. There are Sigmoid, Tanh, and ReLU activation.

Activation Function can limit the output signal to a specific value based on the input. Recently the application of neural networks used the non-linear activation functions. In a neural network like a convolutional neural network, the activation value of the unit based on input values. It is used to decide based on the classification or predict the value of several variables. Activation function has the primary purpose of a multilayer neural network. It is to separate many successive linear transformations by nonlinearity function; if not operate, it would collapse to a single linear transformation.

The function maps negative values to zero and maintaining positive values in the features. In the process, it applies each layer learning to detect different features of CNN networks. Pooling simplifies the output features by performing nonlinear computation and reducing the number of parameters. The function can fasten and achieve effective training result. In the common CNN, it repeats the operation over tens of layers. The ReLU activation function is defined as $f(x) = max(0, x)$ where x is the input to the neuron. A ReLU map has output 0 when the input is less than 0 or minus values; if the input values are higher than 0, the outputs are equal to the input. The ReLU process like a switch for linearity. We also test the other activation functions include the sigmoid, Tanh, and leaky ReLU,

By using the ReLU activation to capture the valuable properties of the input values. It can avoid the vanishing gradient issue, simplifies and accelerates calculations and training. The function performs efficiently and accurately on multi-label datasets. ReLU has a significant advantage in large dataset computation. It has high computational efficiency. Various research demonstrated that ReLU outperforms the conventional sigmoid or hyperbolic tangent function [32].

F. Fully Connected Layer

The Fully Connected Layer is the final layer in a Convolutional Neural network. In the layer, every neuron in the preceding layer is connected to every neuron. In the operation, there can be 1 or more fully connected layers, it is depending on the level of feature abstraction. This layer gets the output from the convolutional, ReLU or pooling layer as its input, and calculates the accuracy and loss score.

Fully Connected (FC) layer computes that outputs a vector of K (the number of classes) dimensions in the classification process. The vector owns the probabilities for each class. In the final phase of the model, uses a designated SoftMax function to reduce noise signal in the fully connected layer before producing the classification result.

The network acts as a classifier for a problem with classes $c_1 \ldots\ldots c_n$, the output layer contains one neuron per class and building a vector $a = a_1 \ldots\ldots a_n$. The SoftMax will be used to convert the values into probabilities, where SoftMax (a_j) is the probability of the input to belong to class c . We need all the output neurons to produce values close to zero. Equation (9) show the SoftMax formulation.

$$S(a_i) = \frac{e^{a_i}}{\sum_{j=1}^{m} e^{a_i}}, \quad i = 1 \ldots. m \qquad (9)$$

We provide the Softmax Layer to normalize the output of the fully connected layer. The CNN network has an output that consists of positive numbers and sum to the one. To calculate probabilities, the classification layer utilizes the output result. The model creates the SoftMax layer after the last fully connected layer in the hidden layer.

G. Avoid Overfitting

To constrains network adaptation to the data, we implement a regularization. It helps to avoid overfitting while training the neural network. The role of the hidden unit is to approximate a function efficiently from the dataset which can be generalized to unseen data.

In this study, we apply Dropout approaches to obtain efficient training in the social network dataset. With dropout, the weights of the nodes in hidden layers become somewhat more insensitive to the weights of the other nodes and learn to decide the outcome independent of the other neurons. Drop-out turns-off some of the hidden units randomly, therefore the hidden units do not need to learn every unnecessary detail of instances in the training set. The dropout operation uses deletion function to each hidden element. The process used to generate a thinned network structure. An important purpose of the model is to find the optimal dropout probability for each hidden element in the network.

Dropout is a modern and excellent regularizer that is easy to implement and compatible with many training algorithms and model. In the experiment, instead of doing it randomly, we tested some types of dropout with different Dropout value. Based on the testing of the value, it gives the contribution of the neuron to the output.

Table II: Mathematical notation of regulizer model

Notation	Description
$l \in \{1, \cdots, L\}$	index the hidden layers of the network
z^l	the inputs vector into layer l
y^l	the outputs vector from layer l $y^l = x$ is the input
r^l	an independent vector of Bernoulli random varia bles with probability p of being 1
$\check{y}^l$	thinned outputs which calculated by $\check{y}^l = r^l * y^l$
W^l	the weights at layer l
b^l	the biases at layer l
f	any activation function for example: $\mathrm{f}(x) = 1/(1 + exp(-x))$

On the function, consider a network has L hidden layers. We use Dropout function to calculate the y_{train} training process for parameters like input x and Bernoulli probability p. In the standard feedforward as for $l \in \{0, \cdots, L-0\}$ we calculate:

$$z_i^{(l+1)} = z_i^{(l+1)} y^l \theta + z_i^{(l+1)}$$
$$y_i^{(l+1)} = f\left(z_i^{(l+1)}\right)$$

In feed forward with Dropout operation, we calculate the Dropout with the following formula:

$$r_j^l \sim \text{Bernoulli probability } p$$
$$\check{y}^l = r^l * y^l$$
$$z_i^{(l+1)} = z_i^{(l+1)} \check{y}^l \theta + z_i^{(l+1)}$$
$$y_i^{(l+1)} = f\left(z_i^{(l+1)}\right)$$

To calculate the next layer, the regulizer utilizes the thinned outputs $y_i^{(l+1)}$ as new input. The process is applied to each layer in the hidden layer. This amount is to construct to a sub-network sampling from a bigger network. In the training time, it back-propagates the derivatives of the loss function via the sub-network. At test time, the weights are multiplied by p and scale the weight as $W_{test}^{(l)} = pW^{(l)}$, so the unit is always present.

We adopt the regulizer to deal with overfitting issues in the training process. Provides a way of approximately combining exponentially many different network architectures. Applying dropout is to obtaining a \thinned" network in the training process. The study utilizes the Dropout regularization during training sample (backward pass), not in predictions process (forward pass).

We also test a dense layer, a type of hidden layer to construct a densely connected network. It is a common layer of neurons where the neuron receives input from all the neurons in the previous layer, So, the layer connects every node to every other node in the next layer. The layer consists of a weight matrix w, a bias vector b, and the previous layer activations a.

IV. Experimental Setup

A. Dataset

This study utilizes the dataset D consisting of message data based on OSN posting activities. To achieve a good classification result, we provide a large corpus OSN for D_{train} and D_{test}. It consists of thousands of sentiments as the corpus in with a set of pairs $\left(x^{(i)}, y^{(i)}\right)$. The sample contains emoticons, usernames, and message elements in the English language. We choose and extract many posting messages from the sample data to build a benchmark dataset. Before computing, we separate the dataset into D_{train}, D_{valid} and D_{test}. By using the extracted features, we undergo sentiment measurement on the posting messages using diverse learning algorithms.

We provide the sample data which consists of different tweets including positive tweets and anomalous tweets. Then, we classify the polarity to mark either positive or negative. If the tweet has both positive and negative parts, the model chooses the more dominant sentiment as the final label. To obtain more features for this experiment, we provide several corpora datasets, original posts, and comments. We construct the training input $\left(x^{(i)} \in \mathrm{R}\right)$ as the i-th training sample of D and $y^i \in (0, \cdots, L)$ is the label assigned to $x^{(i)}$. Fig. 3 illustrates the distribution of the sentiment in the corpus.

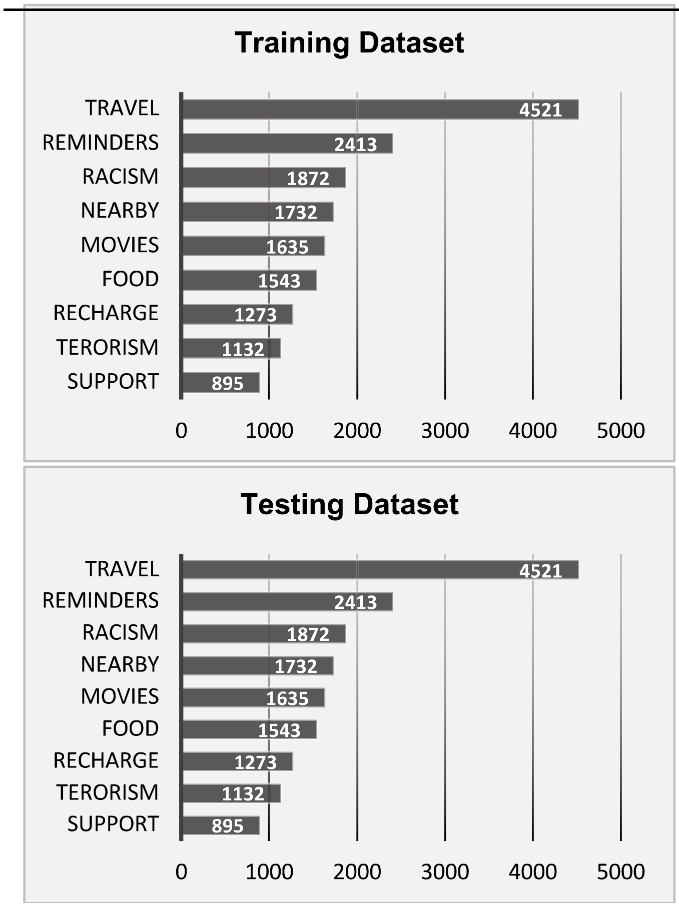


Fig. 3 The histogram for depicting the training and testing dataset. It is to feed into the classifier for sentiment classification.

To gather comprehensive of feature maps, this study explores broad sentiment in OSN as the input and employs the model to analyze sentiment polarity. The result of categories computation determines the type of user. To achieve a better decision for classifying the user, we test diverse OSN message and measuring the sentiment for determining the sentiment categories.

Finally, we plot the histogram to represents the distribution of numerical data accurately. It is an estimate of the probability distribution of a quantitative variable in the training sample. The histogram plots the numeric data that group the data into bins. It relates only one variable and differs from a bar graph, in the sense that a bar graph relates two variables.

B. Data Pre-Processing

Firstly, we undergo pre-processing dataset before feeding the data into the model. The experiment provides the features from all post of OSN. To get clean features and clear data, we undergo preprocessing process. In the process, we eliminate dates, thread titles, hashtags, usernames, and other metadata. Hence, the classifier model just trains the dataset features with the discussion text. In a real implementation, the metadata may not disappear.

To perform predictions in Natural Language Processing (NLP) including sentiment analyses, one of the key ideas is how to convert the words (string type) into numeric vectors efficiently and feed the values into the learning models. In this paper, we implement a current technique which is called "Word2Vec" to undergo the operation [29]. The library takes input and attempting to estimate the other words or phrases probability and feed into the DeepSentiment model.

In this study, before feeding the words or phrases into the models, we convert those into a set of numeric vectors. To address the issue, the study adopts a Continuous Bag of Words (CBOW) method of converting the word into a vector representation. In the Word2Vec, there are two main operations in converting words to a usable vector format. Word embedding is the mapping of a high dimensional style representation of words to a lower dimensional vector. The next step is to maintaining word context to some extent meaning. Word2Vec includes two learning models, Continuous Bag of Words (CBOW) and Skip-gram [30][31].

The paper implements Word2Vec based on CBOW model with user message as the inputs. Thousands of messages have been converted into usable vector format in order to enable training and testing process with deep learning model. In CBOW function with R is the set of real numbers. Input layer includes 2n word vectors in *context(w)* for example $\mathrm{e}(w_{-n}), \ldots, \mathrm{e}(w_n) \in R^m$. To accumulate the 2n, it uses projection layer $x_w = \sum_{i=-n,i\neq 0}^{n}\left(e(w_i)\right)$. In the output layer, it applies a Huffman Tree with the words appears in the corpus data C that corresponds to a word in dictionary D. In the pre-processing phase, we feed the numerical vector array into the DeepSentiment model to compute the training and testing. Data pre-processing convert all of the input samples into numerical arrays. Then, the model feeds the arrays into the neural network.

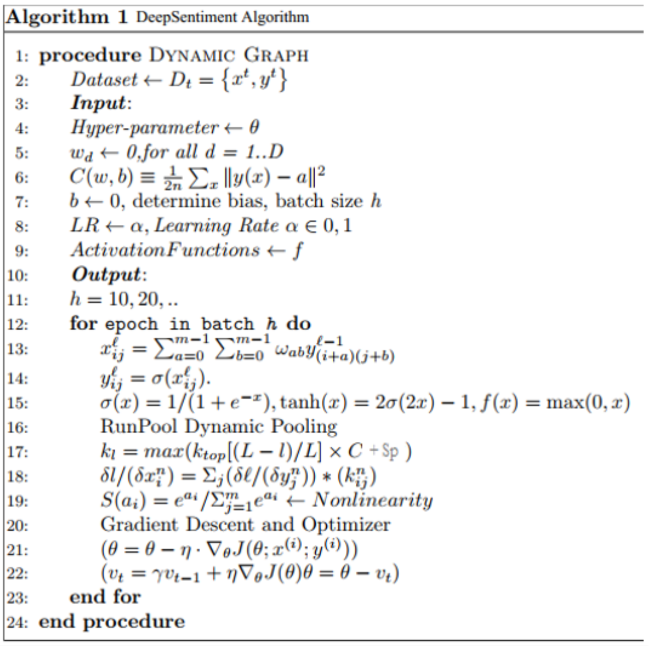
V. Result & Analysis

A. DeepSentiment Algorithm

To implement the Neural Network model, we employ Keras Framework with TensorFlow backend. To train the DeepSentiment network, the study also utilizes the dense vector representation of the tweets. The DeepSentiment constructs model operation by taking a set of input data and crossing it on to the hidden layers. To produce the output, the network undergoes computation in convolution, pooling, and fully connected layers.

The model needs an efficient convolutional to avoid underfitting and overfitting network. In a neural network, the convolutional layer computes a large number of arrays. So, to simplify the array, it needs pooling operation. The pooling layer is able to reduce the feature maps and retain the most informative element. The goal of the convolutional is to harvest numeric features by sliding a filter (smaller matrix) over the input data. The layers compute a rectangular grid of neurons to obtain amounts of features maps. The size of feature maps depends on the of filters.

To obtain the best result of sentiment classification, we provide a benchmark sample in training processes, validation, and the testing process. The effectiveness of the training operation determines the accuracy level of classification. Algorithm 1 describes the step of classification in the DeepSentiment operation. It also figured out how to calculate the forward and backward pass.



Algorithm 1 DeepSentiment Algorithm

```
procedure DYNAMIC GRAPH
    $Dataset \leftarrow D_t = \{x^t, y^t\}$
    Input:
    $Hyper\text{-}parameter \leftarrow \theta$
    $w_d \leftarrow 0, for\ all\ d = 1..D$
    $C(w,b) \equiv \frac{1}{2n} \sum_x \|y(x) - a\|^2$
    $b \leftarrow 0$, determine bias, batch size $h$
    $LR \leftarrow \alpha, Learning\ Rate\ \alpha \in 0, 1$
    $ActivationFunctions \leftarrow f$
    Output:
    $h = 10, 20, ..$
    for epoch in batch $h$ do
        $x^{\ell}_{ij} = \sum_{a=0}^{m-1} \sum_{b=0}^{m-1} \omega_{ab} y^{\ell-1}_{(i+a)(j+b)}$
        $y^{\ell}_{ij} = \sigma(x^{\ell}_{ij})$.
        $\sigma(x) = 1/(1+e^{-x}), \tanh(x) = 2\sigma(2x) - 1, f(x) = \max(0, x)$
        RunPool Dynamic Pooling
        $k_l = max(k_{top}[(L-l)/L] \times C + \mathrm{Sp}\ )$
        $\delta l/(\delta x^n_i) = \Sigma_j(\delta\ell/(\delta y^n_j)) * (k^n_{ij})$
        $S(a_i) = e^{a_i}/\Sigma^m_{j=1} e^{a_i} \leftarrow Nonlinearity$
        Gradient Descent and Optimizer
        $(\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i)}; y^{(i)}))$
        $(v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta)\theta = \theta - v_t)$
    end for
end procedure
```

Based on the algorithm 1, we construct a DeepSentiment model with diverse parameters. We test various hyper-parameters to obtain the optimal result. In the training process, we test the model different hyper-parameters includes a hidden layer, filter, kernel size, pooling, activation, and optimization function.

Principally, to build the DeepSentiment model, we adopt the dynamic graph in the hidden layer. The DeepSentiment model undergoes "Downsampling" process by using the pooling algorithm. The layer also to avoid overfitting in the neural network. The technique allows us to create static graphs that emulate dynamic computation graphs of arbitrary shape and size. In the dynamic graph, the model adopts the dynamic pooling of k. It is to compute each layer in the neural network. In the algorithm, parameter k is scalar value to compute the pooling layer. The value of k-max pooling can be dynamically chosen by making a function based on the network parameters. The dynamic value of the k determines the neural network performance.

It is a mutable directed graph that represents operations on data and the edges (arrows) represent the system output. The advantage of the graphs appears to include the ability to adapt to varying quantities in input data. The characterization of the dynamic graph is an automatic selection of the number of layers, the number of neurons in each layer, the activation function, and other neural network parameters, depending on each input set instance during the training. The framework consists of a system of libraries, interfaces, and components that provide a flexible, programmatic, run time interface.

With the dynamic graph, we test several values of max pooling. As the neural network computation, we also map the multi-dimensional tensor distribution of k-max pooling. Tensor is an exchange type for homogenous multi-dimensional data for 1 to N. We separate the tensor into the dimension and a type. The dimension refers to the rows and columns of the tensor. The study utilizes the two-dimensional tensors in the computing processes.

The simulation calculates average neuron to train the dataset. It produces relatively high accuracy with k-max pooling. In our experiment, adding more layers and neuron numbers in the hidden layer computation cannot improve the predictive capability. However, it enlarges resource in the computing process. Therefore a limitation of neuron number and k-max pooling is an effective method to achieve an efficient result for the neural network.

B. Initial Test with LSTM

At the initial experiment, we test the LSTM algorithm as a classifier. We provide a large number of words from the sample. We add the dense vector representation for training the models. In further tweaking, we set the dense vector representation to make it equal to the Max Length parameter of the network. After computing the LSTM layer, the model calculates a fully-connected layer and activation function (ReLU). To reduce the noise in the fully connected layer, we employ Softmax activation. We also implement dropouts layer to regularize the network and avoid the overfitting. Table III depicts the result of different LSTM models

TABLE III
COMPARISON OF LSTM MODEL WITH DIFFERENT LOSS FUNCTION

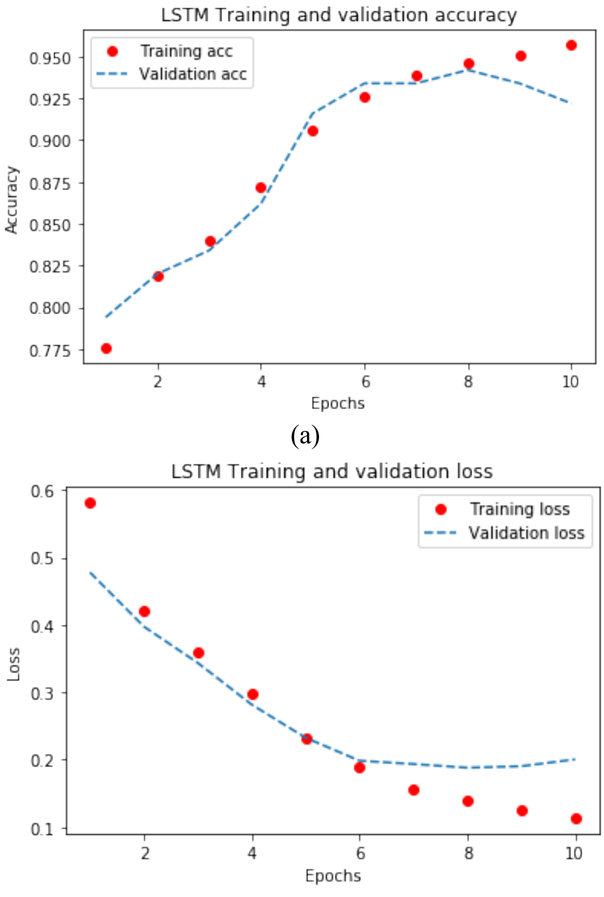
LSTM Unit	Optimizer	Loss	Positive Acc.	Negative Acc.
128	Adam	**BCE**	**81.13 %**	**96.70 %**
64	Adam	BCE	83.01 %	95.93 %
128	Adagrad	MSE	70.75 %	95.43 %
64	Adagrad	MSE	78.30 %	93.40 %
128	RMSProp	BCE	83.96	95.93 %

To compare loss functions in the neural network, the model utilizes with Mean Squared Error (MSE) and Binary Cross-Entropy (CE) loss. Based on the loss accuracy, we achieve that Binary Cross Entropy works better than the MSE. The cross-entropy logarithm enables the network to assess the small errors and try to diminish them. The MSE evaluation is suitable for a regression problem while for a classification task, it is compatible to calculate the loss function the cross-entropy with the function:

$$\mathrm{J}(\Theta) = -\frac{1}{T_{mb}} \sum_{T=0}^{T_{mb}-1} \sum_{f=0}^{F_N-1} \partial^{f}_{y^{(t)}} \ln h^{(t)}_{f} \qquad (11)$$

The function (11) consists of T_{train} as the number of training examples, T_{mb} represents the number of training examples in a mini-batch, $h^{(t)}_{f}$ represents the hidden neuron of each layer and $\partial^{(f)}_{y^t}$ represents the derivative.

In text processing, LSTM is a new powerful algorithm which can classify, cluster and make predictions about text data. Thus, we can feed the engineered features into LSTM classifier in order to train the network. LSTM mitigates the vanishing gradient problem. In the operation process, the algorithm feeds the long-term memory into RNN. It runs by utilizing a couple of gates and has memory blocks which are connected through LSTM layers, Fig. 4 displays the validation accuracy and loss with several popular optimizers by using LSTM algorithm.



(a)

(b)

Fig. 4: Validation Accuracy and Loss with LSTM network (128 embedded dimension).

Fig. 4 depicts that the LSTM suffers from overfitting after training in 6 epochs. Because the algorithm stops learning when the updates to the various weights within a given neural network become smaller and smaller. By computing the features with LSTM, the model also tries to implement weight sharing in the hidden layer. It can reduce the number of parameters in the network, and it can increase the ability of the network's generalization. It can substantially lower the degrees of freedom of network complexity. Thus, the benefit of applying weight sharing is using fewer parameters to optimize, it able to faster convergence to some minimal and avoiding overfitting as the weights are shared with some other neurons.

C. *DeepSentiment CNN Test*

By utilizing DeepSentiment CNN with RunPool function, we test diverse hyper-parameters to the model include learning rate, batch size, number of hidden layer and epoch. Selection of hyper-parameter is becoming an important aspect which can determine the performance of the graph. Firstly, the model converts the input data into features map. The model feeds the features into the graph for classification. Then, we test gradient descent with different hyperparameter. It is to obtain the highest value of the accuracy and get the minimum loss function.

DeepSentiment is also testing the optimization algorithm to measure accuracy and loss in the neural network. The DeepSentiment model tests several gradient descent optimization algorithms such as Stochastic Gradient Descent (SGD) with Momentum, Adam (Adaptive Moment Estimation), Adagrad, and RMSprop. It is to calculate the gradient of loss in the training and testing phase.

We consider broad strategies to optimize gradient descent. We employ the gradient descent to minimize the objective function $j(\theta)$ with model's parameters $\theta \in R^d$. The model updates the parameters in the opposite direction of the gradient of the objective function $\nabla_\theta j(\theta)$. The study sets the learning rate η to determine the stride size to reach a (local) minimum.

The optimizers are to calculate the gradient of the loss concerning the layer weight. The adaptive optimizer divides the learning rate for weight by running an average of the magnitudes. The learning rate algorithm changes iteration according to some cyclic function. To measure validation level between training and test dataset, the study calculates the gradient of loss. The gradient influence efficiency of the result between training and test phase The scheme can enhance the accuracy in DeepSentiment model. We set a local variable to compute the loss function.

In order to measure the accuracy of the model, we test several loss gradients in the benchmark datasets with different amount of hidden layer and gradient descent. Choosing a starting value for a learning rate is one of the most important parts of the neural network study. The result showed that learning rate with dynamic optimizer like Adam produces a more efficient result than the others. Fig. 5 depicts the validation accuracy and loss with Adam Optimizer (128 embedded dimension) when training and testing process with DeepSentiment network.

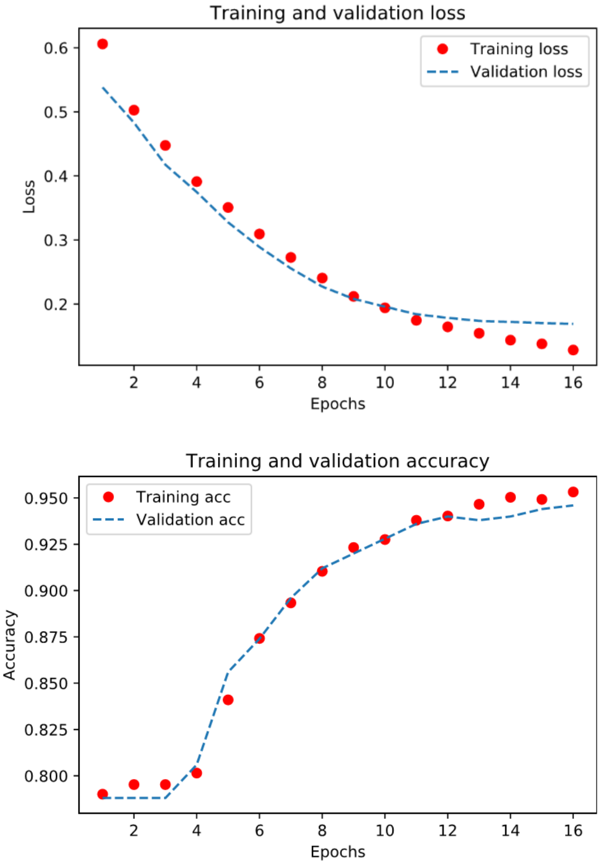


Fig. 5: Validation Accuracy and Loss with Adam Optimizer (128 embedded dimension). The graph shows that the adaptive optimizer is more efficient than the others for this study.

Fig 5 explains the result with the tweaking parameter of Adam gradient descent algorithms. We test both Adam optimizer and SGD with momentum. In the initial process, we feed Adam optimizer into the graph with different amount of hidden layers. Adam is an improvement of the RMSProp. The study finds the Adam optimizer harvests is better and faster in the network. The result shows a dynamic optimizer is able to achieve a small loss in a large number of hidden layers. However, using a large number of hidden layer cause overhead computation in a particular device.

We also test the SGD optimizer; we get the best value in the SGD optimization (*momentum = 0.8*) for the training process. As incremental gradient descent, it is a repetitive technique for optimizing a differentiable objective function. The SGD with momentum produces an efficient result by adopting weight decay in the hidden layers. The more layers in SGD cannot achieve efficient loss and accuracy. TABLE 2 describes the comparison of CNN models with different optimizer. Table IV displays the MSE and BCE calculation result.

TABLE IV

COMPARISON OF CNN MODEL WITH DIFFERENT OPTIMIZERS

CNN Embed-dim	Optimizer	Loss	Positive Acc.	Negative Acc.
128	Adam	**BCE**	**85.84 %**	**97.96 %**
64	Adam	BCE	80.18 %	97.71 %
128	Adagrad	MSE	84.90 %	97.46 %
64	Adagrad	MSE	74.52 %	97.20 %
128	SGD + Momentum	BCE	63.20 %	95.43 %
64	SGD + Momentum	BCE	62.26 %	95.13 %

The result shows the adaptive learning rate such as Adam is more efficient than the others. Adams' runs by changing the learning rate every iteration according to some cyclic function in the network. Dynamic value for calculating gradient loss more rapid traversal of saddle point plateaus. If the momentum optimizer runs like a ball running down a slope, Adam acts like a heavy ball with friction. In the computation process, m_t represents the mean value and v_t reflects the un-centered variance of the gradients. It calculates the decaying averages of past and past squared gradients m_t and v_t as follows:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

Toward the testing of the gradient descent optimization, we obtain the Adam optimizer is more efficient for training and testing in this case. By using dynamic CNN and tweaking appropriate hyper-parameter, we obtain the DeepSentiment CNN can produce better accuracy than LSTM. We find a significant increase in CNN graph's performance within dynamic k-max pooling operation for training data. The DeepSentiment graph can better deal with the problem of data noise, alignment, and other data variations.

The DeepSentiment model computes the unstructured text as the dataset. We find that the dynamic CNN architecture is extremely useful to deal with the case. It is a method to minimize data noise in the text dataset and other data variations. The training and testing results show that the DeepSentiment algorithm can achieve a promising result in detecting and classifying OSN sentiment. By conducting an experiment in the CNN, we can reach better and timelier results when compared with the common approaches such as machine learning algorithms.

By using the algorithm, we find the CE is more efficient than MSE function in loss computation. The CE loss function evaluates the neural network error when it tries to make a prediction to the data. In this experiment, we conduct a classification task, hence, the cross-entropy optimization is more efficient than MSE. The little error can be produced, the better the neural network model.

At the final step, we conduct sentiment classification based on the classifier result. It is to classify sentiment category whether the normal or malicious content. In real OSN application, malicious content like sentiment can reflect malicious activity within the environment. Good monitoring of sentiment measurement is able to provide an advance warning about attacker capabilities and intent for system administrators. The technique can act as one of the active authentication methods for the OSN environment. The learning model allows automating extracting and calculating process for the sentiment. Therefore, in the OSN sentiment classification issue, the originality of this article is measuring OSN sentiment with DeepSentiment CNN to solve the malicious sentiment problem accurately.

D. Evaluation Metric

On the above section, we present how to measure the user sentiment categories. By feeding the extracted features into the classifier, we may able to predict the malicious user categories based on the sentiment analysis. The DeepSentiment experiment also provides an evaluation of the performance model by using cross-validation. It is to measures sensitivity, specificity and accuracy rate. The evaluation of the model calculates False Positive Rate (FPR), True Positive Rate (TPR), accuracy, Receiver Operating Characteristic (ROC) and Area Under Curve (AUC).

The study conducts a performance assessment to evaluate the performance of classification techniques. It is to help performance measurement of the model. We construct the classification model with a supervised learning algorithm and measure the model performance by using evaluation metrics. Commonly, an evaluation technique utilizes the confusion matrix table. A positive value represents the malicious and normal sentiment assigned as negative. Table V displays the structure of the confusion matrix.

TABLE V

CONFUSION MATRIX

Predictive class	Sentiment Categories	
	Malicious	Normal
Malicious	True Positive (TP)	False Positive (FP)
Normal	False Negative (FN)	True Negative (TN)

The statistical approach is to evaluate the performance of the malicious account classification. We divide the confusion matrix into 4 kinds, those are True Positive (TP), False Negative (TN), True Negative (TN) and False Positive (FP). Table VI portrays the classification term of the malicious and normal sentiment.

TABLE VI

CLASSIFICATION TERMS FOR MEASUREMENT

TP	Indicates the number of malicious sentiments that is identified as a malicious sentiment.
FN	Indicates the number of malicious sentiments that is identified as a normal sentiment.
TN	Indicates the number of normal sentiments that is identified as normal sentiment,
FP	Indicates the number of normal sentiments that is identified as a malicious sentiment.

Sensitivity or True Positive Rate (TPR) defines the percentage of the datasets. TPR shows the samples belong to the group and identifies it as such. Equation 12 shows the TPR computation.

$$\mathrm{TPR} = \frac{TP}{TP + FN} \qquad (12)$$

Specificity or True Negative Rate (TNR) defines the rate of the datasets which actually do not include the group and identifies it as such. Equation 13 shows the TPR computation.

$$\mathrm{TNR} = \frac{TN}{TN + FP} \qquad (13)$$

In this model, accuracy (ACC) defines the ratio of all datasets including to the group and the total amount of datasets. Equation 14 shows the TPR computation.

$$\mathrm{ACC} = \frac{TP + TN}{TP + TN + FP + FN} \qquad (14)$$

The study presents classification problems in detecting abnormal tweet. We test the evaluation metric to measure DeepSentiment performance. This experiment adopts the binary classification task. The method predicts and classifies the elements of the dataset into two groups (which group belongs to positive or negative). It needs to calculate the classification accuracy of the model. We plot Receiver Operating Characteristic (ROC) Curves and Area Under Curve (AUC) [38]. It is able to estimate sensitivity and specificity in diverse thresholds without truly changing the threshold.

The ROC in DeepSentiment plots the performance of a binary classifier. To learn the output, DeepSentiment adopts binary classification in the classifier. It needs to convert the output into binaries values to expand the ROC curve. The ROC enables the model to select a threshold that balances sensitivity and specificity. It is useful for classification context. With different threshold tuning, the ROC curve plots the true positive rate (TPR) against the false positive rate (FPR). It is a relative operating characteristic curve with (TPR and FPR) as the criterion changes [40]. The paper calculates cross-validated AUC to estimate the percentage of the ROC plot that is underneath the curve. The result shows some classifiers are more accurate and produce a higher AUC.

In this part, we display several AUC values from different classifier's class. The AUC illustrates the percentage of the area under the ROC curve with ranging between 0~1. The AUC run by measuring the ranking based on the separation of the two classes. The AUC reflects the DeepSentiment' ability to discriminate between positive and negative classes. The model makes good level predictions because it produces the area at 0.8 or above. In this experiment, we obtain a value with an area is larger than 0.5 (AUC= 0.83) for the ROC curve. It reflects the model to make sense for the sentiment case. Fig. 6 depicts the ROC to display the evaluation of the classification model.

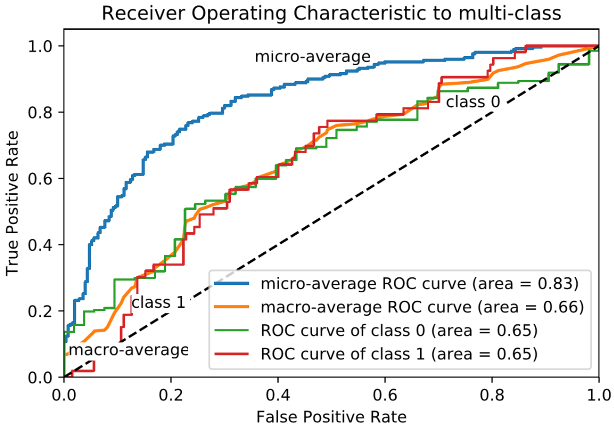


Fig. 6: DeepSentiment adopts binary classification in the classifier. It needs to convert the output into binaries values to expand the ROC curve

VI. Conclusion & Future Direction

OSN has dynamic features with the various posting messages. Inspired by the deep learning success, we present the DeepSentiment, a classification technique with dynamic deep learning for detecting malicious sentiment in the OSN. Instead of using the conventional learning model, we present dynamic deep learning in the training and testing process.

In the DeepSentiment, we test the RunPool, a novel CNN network with dynamic pooling layer in the neural network. We find a significant increase in CNN graph's performance with the model. In this study, the model computes the OSN sentiment as matrix inputs and employs the model to classify benign or malicious over the posting message. The user posting message categorizes differ the sentiment as a normal or suspicious activity.

To test the malicious detection model, we calculate sentiment analysis by using different gradient descent. We harvest Adam optimizer is more efficient than others. Adams' runs by changing the learning rate every iteration according to some cyclic function. Dynamic value for calculating gradient loss more rapid traversal of saddle point plateaus. By using dynamic CNN and tweaking appropriate hyper-parameter, we obtain the DeepSentiment CNN can produce a better accuracy and small loss than LSTM. We find a significant increase in CNN graph's performance by using RunPool pooling function for training data. To evaluate the model performance, we conduct evaluation metrics. The ROC curve depicts that the model produces the AUC score is larger than 0.83. Hence, it makes sense as a detection method for sentiment classification.

As the future directions, to achieve better analysis for OSN tweet, it needs an exploration for handling emotion ranges. The emotion reflects polarity of sentiments. Because the tweets are not only positive or negative sentiment but also it has no sentiment (neutral) and gradations sentiment. Then, neural network computation is necessary to calculate with a new technique like adaptive loss function rather than a conventional loss function.

Acknowledgment

The paper is conducted in the Institute of Research in Information Processing Laboratory, Harbin University of Science and Technology under CSC Scholarship.

REFERENCES

[1] Takahashi T, Panta B, Kadobayashi Y, Nakao K. and Web of cybersecurity. "Linking, locating, and discovering structured cybersecurity information." Int. J. Commun. Syst. (2018): 31:3470.

[2] S. Park, K. Cho, and B. G. Lee, "What makes smartphone users satisfied with the mobile instant messenger, Social presence, flow, and self-disclosure." Int. J. Multimed. Ubiquitous Eng., (2014): 9:11:315–324

[3] Mehdi Dadkhah, Tole Sutikno, Shahaboddin Shamshirband. "Social Network Applications and Free Online Mobile Numbers: Real Risk." International Journal of Electrical and Computer Engineering (2015): 5:2:175-176.

[4] M Al-Qurishi, M Al-Rakhami, A Alamri, M Alrubaian, S. M. M. Rahman and M. S. Hossain. "Sybil Defense Techniques in Online Social Networks: A Survey." IEEE Access; (2017): 5:1200-1219.

[5] C. Anglano, M. Canonico, M. Guazzone. "analysis of Telegram Messenger on Android smartphones." Digital Investigation, Elsevier (2017): 23, 31-49.

[6] Said, N. B. Al Barghuthi and H. "Social networks IM forensics: Encryption analysis." J. Commun (2013): 8:11:708–715..

[7] Ren. J, L. Harn, "Generalized Digital Certificate for User Authentication and Key Establishment for Secure Communications." IEEE Trans. Wireless Comm. (2011): 10: 7:2372-2379.

[8] M. Frank, R. Biedert, E. Ma, I. Martinovic and D. Song. "Touchalytics: On the Applicability of Touchscreen Input as a Behavioral Biometric for Continuous Authentication." Transactions on Information Forensics and Security (2013): 8:1:136-148.

[9] Kumar M, Verma HK, Sikka G. "A secure lightweight signature-based authentication for Cloud-IoT crowdsensing environment." Trans Emerging Tel Tech (2018).

[10] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradientbased learning applied to document recognition" Proceedings of the IEEE. 1998. 86:11:2278–2324.

[11] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. "Going Deeper with convolutions." arXiv:1409.4842, (2014).

[12] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. "Deepface: Closing the gap to human-level performance in face verification." In Computer Vision and Pattern Recognition (CVPR), IEEE Conference on Pattern Recognition, (2014): 1701–1708.

[13] Abidin, M. Yusof and A. "A secure private instant messenger." Proc. 17th Ascia-Pacific Conference on Communications (2011): 821-825.

[14] L. Ham, "Group Authentication," IEEE Trans. Vehicular Technology, (2013): 62: 9.

[15] Li M, Xiang Y, Zhang B, Wei F, Song Q. "A novel organizing scheme of single topic user group based on trust chain model in social network." Int. J. Commun. Syst. (2018): 31:3387

[16] Liu, B.-H., Hsu Y.-P., and Ke W.-C. "Virus infection control in online social networks based on probabilistic communities." Int. J. Commun. Syst. (2014): 27:4481–4491.

[17] Ham, L. "Agent Based Secured e-Shopping Using Elliptic Curve Cryptography." International Journal of Advanced Science and Technology (2012): 38.

[18] K. H. Yeh, C. Su, W. Chiu and L. Zhou. "I Walk, Therefore I Am: Continuous User Authentication with Plantar Biometrics." IEEE Communications Magazine (2018): 56:2:150-157

[19] S. Jiang, M. Duan and L. Wang. "Toward Privacy-Preserving Symptoms Matching in SDN-based Mobile Healthcare Social Networks." IEEE Internet of Things Journal (2018): 99:1-1.

[20] J. Du, C. Jiang, K. C. Chen, Y. Ren and H. V. Poor. "Community-Structured Evolutionary Game for Privacy Protection in Social Networks." IEEE Transactions on Information Forensics and Security (2018): 574-589.

[21] B, Scheiner. Applied Cryptography Protocols, Algorithms and Source Code in C. Second Edition. New York: John Wiley Sons, inc, 1996.

[22] Stallings, W. Cryptography and Network Security. Prentice Hall, 2006.

[23] Arshad H, Rasoolzadegan A. "A secure authentication and key agreement scheme for roaming service with user anonymity." Int. J. Commun. Sys (2017): 30:3361.

[24] Patel. W. M. P. Perera. "Efficient and Low Latency Detection of Intruders in Mobile Active Authentication." IEEE Transactions on Information Forensics and Security (2018): 1392-1405.

[25] K. B. Schaffer, "Expanding Continuous Authentication with Mobile Devices," Computer (2015):48:11: 92-95.

[26] Singh, M. Shahzad and M. P. "Continuous Authentication and Authorization for the Internet of Things." IEEE Internet Computing (2017): 2:1:86-90.

[27] V. M. Patel, R. Chellappa, D. Chandra and B. Barbello. "Continuous User Authentication on Mobile Devices: Recent progress and remaining challenges." IEEE Signal Processing Magazine (2016): 33:4:49-61

[28] G. Peng, G. Zhou, D. T. Nguyen, X. Qi, Q. Yang and S. Wang. "Continuous Authentication With Touch Behavioral Biometrics and Voice on Wearable Glasses." IEEE Transactions on Human-Machine System (2017): 47:3, 404-416.

[29] T. Mikolov, K. Chen, G. Corrado, J. Dean, "Efficient Estimation of Word Representations in Vector Space", *Proc. Workshop at ICLR*, 2013.

[30] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean, "Distributed Representations of Words and Phrases and their Compositionality", *Proc. NIPS*, 2013.

[31] T. Mikolov, W. Yih, G. Zweig, "Linguistic Regularities in Continuous Space Word Representations", *Proc. NAACL HLT*, 2013.

[32] LeCun Y, Bengio Y, Hinton G. "Deep learning." Nature (2015): 436–444.

[33] Y Y Zhao, B Qin, T. Liu, "Sentiment Analysis[J]", *Journal of Software*, vol. 21, no. 8, pp. 1834-1848, 2010.

[34] A Yang, J Lin, Y. Zhou, "Method on Building Chinese Text Sentiment Lexicon[J]", Journal of Frontiers of Computer Science & Technology, 2013.

[35] Zhang Shan, Yu Liubao, Hu Changjun, "Sentiment analysis of Chinese Mircro-blog based on emotions and emotional words [J]", Computer Science, vol. 39, no. 11A, pp. 146-148, 2012.

[36] L Xie, M Zhou, M. Sun, "Hierarchical Structure Based Hybrid Approach to Sentiment Analysis of Chinese Micro Blog and Its Feature Extraction [J]", Journal of Chinese Information Processing, vol. 26, no. 1, pp. 73-83, 2012.

[37] Y. Chen and Z. Zhang, "Research on text sentiment analysis based on CNNs and SVM," 2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA), Wuhan, 2018, pp. 2731-2734. doi: 10.1109/ICIEA.2018.8398173

[38] Hanley JA, & McNeil BJ. "The meaning and use of the area under a receiver operating characteristic (ROC) curve". Radiology. 143, 29–36, 1982.

[39] P. Wanda, Selo and B. S. Hantono, "Efficient message security based Hyper Elliptic Curve Cryptosystem (HECC) for Mobile Instant Messenger," *2014 The 1st International Conference on Information Technology, Computer, and Electrical Engineering*, Semarang, 2014, pp. 245-249.

[40] Swets, John A.."Signal detection theory and ROC analysis in psychology and diagnostics : collected papers", Lawrence Erlbaum Associates, Mahwah, NJ, 1996.

[41] P Wanda, H J Jie, "Efficient Data Security for Mobile Instant Messenger." Telkomnika Journal, Vol. 16 (3), 2018.

[42] Bin Ning, Wu Junwei, Hu Feng "Spam Message Classification Based on the Naive Bayes Classification Algorithm". IAENG International Journal of Computer Science, Vol. 46, No. 1, pp. 46-53, 2019

[43] Hashida S, Tamura K, Sakai T. "Classifying Tweets using Convolutional Neural Networks with Multi-Channel Distributed Representation" IAENG International Journal of Computer Science, Vol. 46, No. 1, pp. 68-75, 2019.

[44] W. Putra, H J Jie, "URLDeep: Continuous Prediction of Malicious URL with Dynamic Deep Learning in Social Networks." FEMTO International Journal of Network Security, Vol. 21 (5), 2019.

Putra Wanda

He received B.Eng in Informatic Engineering in 2011. and M.Eng. degrees in Information Technology from Gadjah Mada University 2015. Since August 2016, he is with the School of Computer Science and Technology from Harbin University of Science and Technology as a Ph.D. candidate. https://orcid.org/0000-0003-0130-3196.

Huang Jin Jie

He is Professor and Ph.D. Supervisor in School of Science and Technology, Harbin University of Science and Technology, China. His current research interests include Deep Learning, Pattern Recognition and Automation system (https://orcid.org/0000-0002-2107-2690)