Capsule Network for Cyberthreat Detection

Sahar Altalhi[1], Maysoon Abulkhair[2], Entisar Alkayal[3]
Information Technology Department
King Abdulaziz University

***Abstract*—In cybersecurity, analyzing social network data has become an essential research area due to its property of providing real-time updates about real-world events. Studies have shown that Twitter can contain information about security threats before some specialized sites. Thus, the classification of tweets into security-related and not security-related can help with early warnings for such attacks. In this study, the use of a capsule network (CapsNet), the new deep learning algorithm, is investigated for the first time in the field of security attack detection using Twitter. The aim was to increase the accuracy of tweet classification by using CapsNet rather than a convolutional neural network (CNN). To achieve the research objective, the original implementation of CapsNet with dynamic routing is adapted to be suitable for text analysis using tweet data set. A random search technique was used to tune the model's hyperparameters. The experimental results showed that CapsNet exceeded the baseline CNN on the same data set, with accuracy of 92.21% and a 92.2% F1 score; also, word2vec embedding performed better than a random initialization.**

***Keywords*—Capsule network; dynamic routing; deep learning; Twitter; text analysis; attack detection**

I. Introduction

Security monitoring and attack detection are essential parts of any organization's management for protection against cyber-attacks. These attacks can cause service disruption, asset damage, data breaches, or data loss. To avoid such dangerous effects, a number of official security data sources are available, including the National Vulnerability Database (NVD) [1], which contains a security analysis of discovered vulnerabilities, and the ExploitDB [2], which provides a user-friendly interface for all discovered exploits targeting known vulnerabilities. These traditional data sources provide trusted security information, but it comes at a cost, which is the delay of reporting the information [3]. Not all reported vulnerabilities will be exploited in the real world, and some have a higher probability of being exploited and thus need to be patched first [4].

For system administrators, the time between the detection of a cyberattack plan and the actual occurrence is critical. They need up-to-date information about current or imminent attacks to analyze them, study their impact, and be aware of new attack types and hacking tools in real time [5]. One of the new solutions for this problem is utilizing social network data to extract real-time notifications about the security situation of the organization or software and hardware used in its infrastructure.

As one of the most popular social networks, Twitter is considered a rich source of information about different security threats. This claim is supported by studies showing that Twitter contained information about security threats before some specialized sites [6]–[8]. This observation attracted researchers to analyze Twitter data and extract knowledge to be used in the detection and prediction of security attacks. The objectives when using Twitter data in the security field vary, from vulnerability and exploit detection [4], [9], [10] to attack detection by linking a sentiment score to a specific target with real security events [11]–[13], and trying to determine the threshold of tweet sentiment that predicts the probability of the attack occurring [14].

Text classification using different machine learning (ML), neural network (NN), and deep learning (DL) algorithms has been widely investigated for detecting cyber-attacks using Twitter data. One of the most advanced techniques for this purpose is the convolutional neural network (CNN) [15]. As one of the DL algorithms, a CNN overcomes the traditional ML technique limitation by providing automation for the learning process [16]. However, the CNN comes with limitations that are mainly related to the use of the pooling layer [17], which will be described in detail in Section II.

In 2017, the godfather of DL, Geoffrey Hinton, proposed the capsule network (CapsNet), which was first examined using the Modified National Institute of Standards and Technology's (MNIST) data set [18]. CapsNet outperforms its predecessor, the CNN, in many image classification tasks [19], but it is still in early stages for text classification [20]. This study aimed to use Twitter to examine CapsNet's capability for providing accurate classifications of security tweets with the goal of cyberthreat detection. The CapsNet is implemented by building an NN model to classify tweets as security-related or not security-related. Then, the CapsNet model was evaluated in terms of classification accuracy and F1 score, and using the CNN as a baseline model, compared the performance of CapsNet in tweet classification for the security field.

The rest of this paper is organized as follows: In Section II, an overview of CapsNet's improvements in comparison to CNNs will be given. Section III covers the main recent work done in the field using Twitter for cyberthreat detection. This is followed by Section IV, which describes the implementation of the proposed model. In Section V, the details of the experiments conducted is described, and Section VI includes the results and discussion. Finally, the paper's conclusion and future works are discussed in Section VII.

II. Background

Until recently, CNNs achieved state-of-the-art results for many natural language processing (NLP) tasks [21]. However, CNNs have limitations and drawbacks, such as with pooling. Pooling, one of the building blocks of CNNs, is used to reduce the complexity and the number of parameters in the CNN while preserving the main features [20]. This makes CNNs

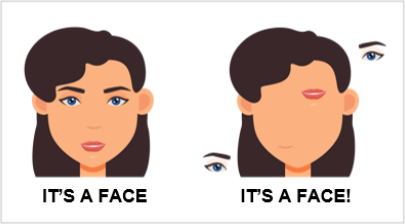

Fig. 1. Face recognition in CNN.

particularly efficient with classification tasks, but it causes a loss of valuable information such as the precise location of an object or the relationships between the object's parts [18]. Fig. **??** shows the way that the CNN works. Even when parts of the face are not arranged correctly, the CNN will classify it as a face regardless of the location and relationships between the parts [19]. For better modeling of spatial relationships among parts, CapsNet is proposed [22].

The architecture of the CapsNet overcomes CNN's drawbacks by different properties [18]. First, the basic unit in the CapsNet is the capsule (vector), where each one is a set of neurons representing an object or an object part. CapsNet transforms vector inputs into vector outputs; thus, it can learn more complex transformations than CNNs, which operate on scalars. The output of a capsule is an activity vector, where its length represents the probability of the existence of the object, and its coordinates (dimensions) encode the object's attributes (pose information), which preserves the spatial relationship between features. Second, CapsNet uses a routing-by-agreement technique to replace the routing by max pooling used in the CNN. In simple terms, instead of extracting the most important features by using max pooling and ignoring the less important ones, propagation between the layers will be based on routing-by-agreement. This means that the output of each capsule will be forwarded to the next layers' capsules with different weights that are based on the agreements between the capsules.

In NLP, CapsNet has a greater ability to efficiently learn the spatial relationships between words, such as the local order of words and their semantic representations [22]. Many researchers have investigated the use of CapsNet for NLP tasks like sentiment analysis [23], [24], fake news detection [25], stock performance prediction using Twitter [26], implicit emotion detection [27], and offensive posts on social media [28]. The results of these studies showed that CapsNet outperformed the CNN in text classification, which was part of the motivation for the present study.

III. Related Work

In this section, some studies that used Twitter data for the detection of security attacks are reviewed. For each work, the specific problem that was solved by each of these studies is summarized, the analysis techniques used, and the results obtained to give an overview of the research already conducted in the field of interest.

In order to discover Twitter discussions about emerging attacks against a specific target, the authors of [29] proposed an approach to security event detection that learned with positive and unlabeled data based on user-provided expectations. Expectation regularization (ER) was used to find the ratio between positive and negative examples in the training process. The study's security events included denial of service (DoS) attacks, data breaches, and account hijacking represented in the form of (Entity, Date) as training examples. Two sets of manually extracted features were used to find new events. Using the logistic regression (LR) classifier, the proposed solution was able to detect new events automatically in real time for each predefined category.

The use of simple discrete features may suffer limitations in representing subtle semantic differences between true event mentions and false cases with similar word patterns. To overcome this limitation, the researchers in [16], based on [29], modified the method to be more semantically based by using a long short-term memory (LSTM) based neural embedding model that learns tweet-level features automatically. This change improved the detection accuracy as compared to the previous method because of the NN's ability to represent deep semantic information, which is more difficult to capture through discrete features.

As an end-to-end solution for cyberthreat detection, SYNAPSE [30] provided a real-time extraction of security events from Twitter with high-level abstraction. A data set of more than 195,000 tweets was collected from security-related accounts and filtered by keywords related to the monitored infrastructure. The statistical method called term frequency-inverse document frequency (TF-IDF) was used to extract the tweets' features. Support vector machine (SVM) algorithm was used for feature learning, and it achieved a minimum true positive rate (TPR) and a true negative rate (TNR) of 90% in classifying tweets. For more informative extractions, the model proposed in [30] included stream tweet clustering using a dynamic clustering algorithm and summarization of each cluster with the exemplar tweet. The model was able to detect important actionable threats by verifying them with threats reported in the Common Vulnerability Scoring System (CVSS).

With the same objectives as the previous work [30], the authors in [31] proposed an event detection model with joint phases that performed the filtering, clustering, and summarization with shared tweet representation. Features were extracted using skip-gram and LSTM to obtain vector representations, and a multi-layer perceptron (MLP) classifier was used for tweet classification identifying security-related tweets. The tweets were clustered in groups, and each cluster was summarized with the most informative tweet provided. All these phases were conducted jointly based on features extracted at the beginning. The collaborative event detection and summarization model was more effective than solutions that used discrete or neural models for new event detection, clustering, and summarization.

The authors in [32] proposed a model that consists of three steps: data collection and pre-processing, feature extraction, and class prediction. They collected two balanced sets of tweets. The first set, which represented the positive class, contained tweets retrieved from cybersecurity accounts, while the negative set was tweets retrieved from non-security specialized accounts, such as health, news, and magazines. Next, for feature extraction, they used TF-IDF. In the class prediction step, the binary naïve Bayes (NB) classifier was

trained using a 10-fold validation approach, which resulted in an average accuracy of 77.90% for tweet classification into security-related or not.

The authors in [33] proposed a DL classification model based on domain-specific and contextual embeddings to extract features from raw tweets. These features are convolved using a meta-encoder and then combined to be sent to the CNN, LSTM, and contextual encoder for feature learning in parallel. The resultant feature maps were concatenated with contextual embeddings in a fusion layer. A softmax classifier was used for the final prediction for each tweet as security-related or not. Compared to a set of ML and NN baseline models, the proposed model performed better with accuracy of 82%, precision of 79%, 72% recall, and an F1 score equal to 76%.

The authors in [34] used a CNN to classify tweets containing security keywords as security-related or not. All tweets were retrieved from security-specialized Twitter accounts that mentioned three predefined organizations or their assets. The results obtained were using GloVe and word2vec embedding and also used random initialization trained on the classification task. The embedded tweets were fed into three CNN layers in parallel to be convolved with a different number of filters and filter sizes. The researchers suggested a named entity recognition (NER) step to extract the main entities in the tweet using bidirectional LSTM (BiLSTM). The results confirmed that the CNN model performed better than traditional ML techniques. The classification performance achieved 94% recall and 91% TNR, while the NER achieved a 92% F1 score with specifying appropriate entities.

Recent studies that reviewed in this section used a CNN, which opened the door for more investigation and encouraged more studies to improve the accuracy of detecting potential security attacks. The present study aimed to implement the new CapsNet algorithm for the first time in the field of attack detection using Twitter.

IV. Implementation

Before describing the model implementation, a general representation of the tweet classification model that has the purpose of cyberthreat detection is illustrated. As shown in Fig. 2, it contains three layers: an input layer, a classification algorithm, and an output layer. The input layer holds the tweets to be classified, which pass to the classification algorithm in the second layer, the CapsNet in this work. The final goal of this architecture is the label's prediction of each tweet, which is the task of the output layer in labeling each tweet as security-related or not security-related.

The CapsNet model that is proposed for classification purpose is the main contribution of this work. The input of the model is a tokenized tweet of n words, and the output is the predicted class of this tweet. The same principles used in [18] for MNIST handwritten digit data set classification will be followed and adapted to be compatible with the tweet data set. The architecture of the model is shown in Fig. 3 and described in the following sections.

A. Embedding Layer

This layer acts as a link between the input layer and the NN because the NN does not understand the textual input. If

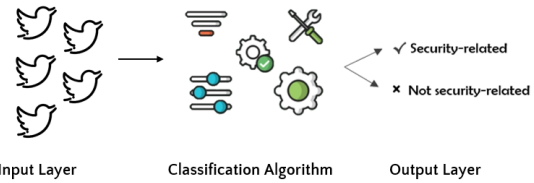

Fig. 2. The general architecture of the cyberthreat detection model using Twitter.

n indicates the number of words in a tweet, then each tweet is represented by an array of length n. Thus, there is a need to convert each word into a numeric representation using a word embedding model. Each word will be mapped to its corresponding numeric representation in the embedding model. According to this description, the embedding layer converts the tokenized tweet from an n-dimensional vector to an $n \times d$ dimensional tensor of a floating points matrix to be sent to the next layer.

B. Convolutional Layer

The first convolutional layer is a regular convolutional layer that the embedded tweet is fed to before passing to the primary capsule layer. It convolves the embedding matrix with a set of filters f and a kernel size $k \times d$. This means it processes k words at a time, which results in a tensor with size $f(n-k+1)$.

C. Primary Capsule Layer

The primary capsule layer is fully connected to the next layer and consists of three transformations performed sequentially:

- Second convolutional layer: Similar to the previous convolutional layer. It performs a convolution operation on its input with a kernel size k, which reduces the input by $k + 1$.
- Reshape: As mentioned previously, one of the significant contributions of CapsNet is that it deals with vectors. Scalar is a quantity with magnitude only, while a vector is a quantity with the magnitude as well as direction. This layer is added to reshape the input feature maps, scalar values, to an output vector map of the desired dimensions to get a set of vectors (capsules) instead of scalars.
- Squashing function: To ensure that all vectors' lengths, which represent a probability, are between 0 and 1 while preserving the orientations of the vectors (features detected) as the following equation [18]:

$$v_j = \frac{\|s_j\|^2}{1 + \|s_j\|^2} \frac{s_j}{\|s_j\|} \quad (1)$$

where v_j is the vector output of capsule j and s_j is its total input.

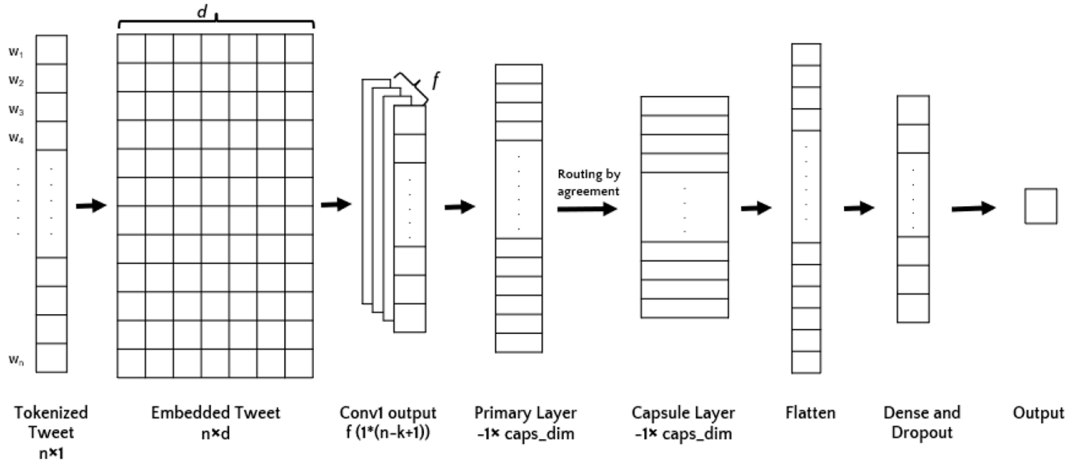

Fig. 3. CapsNet architecture for cyberthreat detection.

D. Capsule Layer

At the point, between the primary and the capsule layer, the novel routing algorithm sits. The goal of routing-by-agreement is to send the output of the lower-level capsule (output of the primary capsule) with high weights to the capsule in the next layer (capsule layer) that it agrees with. To do that, it calculates the predicted output of the next layer by learning routing coefficients in multiple routing rounds. In other words, it strengthens routing weights where predictions made by primary capsules match secondary capsule outputs based on the routing algorithm proposed in [18]. Between the CNNs, which usually implement routing by max pooling that results in loss of some information and the fully connected layers, routing-by-agreement reduces the noise forwarded to the next layer while keeping all the desired information for accurate classification.

E. Flatten, Dense, and Dropout Layers

The output of the previous layer (capsule layer) is a two-dimensional array/matrix, and the next layer is a dense layer that expects a one-dimensional array. The flatten layer is responsible for transforming the two-dimensional matrix of features into a vector by stacking the rows next to each other in a way that can be fed into a fully connected layer for prediction. Then, instead of using the decoder proposed in the work by [18], dropout is used as a regularization method against overfitting [35], which will drop a percentage of the neurons in the flatten layer randomly [36].

F. Output Layer

Since the problem of this work was a binary classification, the final layer of this architecture is a dense layer that predicts the class of each input tweet. Many activation functions can be used to accomplish the aim of this layer, such as softmax or sigmoid, which labels the tweet to be security-related (positive) or not security-related (negative).

TABLE I. DATA SET STATISTICS.

	Training	%	Validation	%	Testing	%
Positive	2,134	50.14	300	50	712	50
Negative	2,122	49.86	300	50	712	50
Total	4,256	100	600	100	1424	100

V. EXPERIMENTS

In this section, the hardware and software configurations used in the experiments is reviewed. In addition, the data set that were used, the embedding layer specifications, the baseline CNN model that was proposed for comparison purposes, and the optimization process that was conducted to tune the models' hyperparameters will be described.

A. Tools

The experiments were run on Google Colab [37], a free cloud-based service, with a Tesla P4 GPU and 25 GB of RAM. The code was implemented in Python 3.6.9 with Keras 2.2.4 [38], using TensorFlow 1.15.2 [39] as a backend.

B. Data Set

The data set that satisfied the model requirements was the data set created in the work [34]. It contains tweets that were retrieved from predefined Twitter security-related accounts that mention the infrastructures being monitored or its assets and denoted as A, B, and C. The use of specialized Twitter accounts eliminated the retrieval of tweets containing the desired keyword without security context, such as the words "apple, windows, network, virus, worm, root." The data set was already divided into three sets: training, validation, and testing. Two sets of security specialist accounts, denoted as S1 and S2, were used. The training and validation sets contained the tweets that were retrieved from the S1 accounts, while the testing set was compiled from the S1 and S2 Twitter accounts. The goal of using different sets of Twitter accounts was to give us insights about the models' performances on not only unseen tweets but also tweets retrieved from a different set of Twitter accounts. Another property for the data set was the

TABLE II. CONVOLUTIONAL NEURAL NETWORK BASELINE RANDOM SEARCH SPECIFICATION AND RESULTS.

Layer	Parameters	Search Values	Best Value	
			Random	Word2vec
Convolutional Layer 1	Number of filters	[128, 192, 256, 320, 384, 448, 512]	256	384
	Filter size	[3,5,7,9]	7	3
Convolutional Layer 2	Number of filters	[128, 192, 256, 320, 384, 448, 512]	128	128
	Filter size	[3,5,7,9]	7	7
Convolutional Layer 3	Number of filters	[128, 192, 256, 320, 384, 448, 512]	384	256
	Filter size	[3,5,7,9]	9	3
Dense Layer 1	Number of neurons	[64]	64	64
Dropout	Dropout rate	[0.1, 0.2, 0.3, 0.4, 0.5]	0.1	0.3

TABLE III. CAPSNET RANDOM SEARCH SPECIFICATIONS AND RESULTS.

Layer	Parameters	Search Values	Best Value	
			Random	Word2vec
Convolutional Layer	Number of filters	[128, 256, 384, 512	384	384
	Filter size	[3, 5, 7, 9]	7	5
Primary Capsule Layer	Number of filters	[16, 24, 32, 40, 48, 56, 64]	48	24
	Filter size	[3, 5, 7, 9]	3	9
	Primary capsule dimensions	[8, 12, 16, 20, 24]	24	12
Capsule Layer	Secondary capsule dimensions	[8, 12, 16, 20, 24]	16	12
	Number of capsules	[64]	64	64
	Number of routing iterations	[3, 4, 5, 6]	6	5
Dense Layer	Number of neurons	[64, 128, 192, 256]	64	256
Dropout	Dropout rate	[0.1, 0.2, 0.3, 0.4, 0.5]	0.3	0.3

time interval of the tweets, where the validation and testing sets were retrieved from time intervals following the training set. This means that the obtained results would simulate the real deployment of the model. Then, the collected tweets were filtered based on a set of keywords describing the selected organizations and labeled as security or not.

C. Data Set Retrieval and Statistics

Because of Twitter's policy, which prevents publishing tweets in plain text, the data set was only available in the form of (tweet_ID, label). Thus, a Twitter developer account was created to retrieve the text of the tweets knowing the IDs using Python and Tweepy library. At the time of retrieving the tweets, some were missed due to deletion by the user or the user being suspended. The data set was manipulated to serve the work objectives as follows: the tweets from the three infrastructures were merged and duplicates were deleted since division by the organization was out of the scope of this study. In addition, to work with a balanced data set, the validation and testing tweets were merged, and 300 tweets from each class were specified as a validation set and the remaining as testing tweets, while keeping the classes balanced. The resultant data set statistics are shown in Table I.

D. Data Set Pre-Processing

The raw tweets were cleaned in a pre-processing step, the approach used by [34] for the same data set was followed. In detail, each tweet was converted into lowercase, special characters other than "." and "-" were removed and replaced with a dot and hyphen, respectively. These symbols were needed because they could exist in the software versions and common vulnerabilities and exposures (CVE) numbers. Then, all numbers were converted into its textual representation to be analyzed as text, which resulted in tokenized tweets that were the input for the embedding layer, as described in Section IV-A.

E. Word Embedding

The embedding layer received the tokenized tweet results from the data pre-processing and converted each word into a high-dimensional vector. A widely used NLP technique for feature extraction that represents the semantic meaning of words is word embedding [40]. In this work, two ways of initializing the embedding matrix were examined: using Keras embedding [38] and word2vec pre-trained word embeddings [41], both with 300 dimensions.

F. Baseline Convolutional Neural Network (CNN) Model

In order to choose the appropriate architecture for the baseline model that was used for purposes of comparison, the CNN model used as a baseline in the CapsNet-MNIST proposal [18] was manipulated to be suitable for the text data set. In [18], the CNN and CapsNet models were not architecturally similar, but the authors designed them with similar computational efforts that served the work's objectives. Similarly, the baseline CNN model of this work was built with three convolutional layers, flatten layer, dense layer, and dropout layer, and then added a last dense layer for final prediction.

G. Hyperparameter Tuning

Hyperparameters are the model's parameters that were not included in the training. These parameters should be set carefully because they affect how the model will learn from the data. The manual selection of these values could be less than optimal, and the solution for that problem is hyperparameter tuning. This step was performed because one of the work objectives was finding the optimal values that would lead to the best model performance and generate acceptable results. The random search is used for optimization [42]. For a fair comparison, 100 combinations of each model were tested, the CNN and CapsNet in 200 epochs and early stopping after five epochs. Table II lists all the layers that were included in the

TABLE IV. ACCURACY AND F1 SCORE RESULTS.

Model	Embedding	Accuracy	F1 Score
CNN baseline pooling	Random	91.01	90.84
	Word2vec	91.15	90.92
CNN baseline	Random	91.43	91.44
	Word2vec	91.64	91.46
CapsNet	Random	91.85	91.78
	Word2vec	**92.21**	**92.20**

search process, the values to be examined for each hyperparameter, and in the final column, the optimal values based on the validation set results. Similarly, Table III shows the hyperparameter tuning specification for the proposed CapsNet model. To reduce the total number of combinations, hyperparameters such as the optimizers or the activation functions were not included because we fixed them in both models, and for batch size and learning rate, we kept the default values.

VI. RESULTS AND DISCUSSION

This study was conducted to investigate the use of CapsNet for cyberthreat detection by classifying tweets into security-related or not security-related. The model was built based on hypothesizing and aiming at proving that CapsNet could classify security tweets more accurately than a CNN and that routing-by-agreement is more efficient than pooling. In order to verify the correctness of the work hypothesis, the final architectures generated from the hyperparameter tuning process was trained to minimize the validation loss using binary cross-entropy loss function, and evaluated them using the classification accuracy and F1 score. Classification accuracy is the ratio of correct predictions (positive and negative) to the total number of samples and is computed as in [43]:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

while the F1 score is calculated as:

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{3}$$

For the first hypothesis, mentioned above, the CapsNet model was compared with a CNN model that did not include a pooling layer. However, the second hypothesis was tested by comparing the CapsNet with a CNN model that had a pooling layer to prove the efficiency of the routing over the pooling. As can be seen in Table IV, the proposed model achieved competitive results over the strong baseline models. In addition, the accuracy and F1-score comparisons are presented in Fig. 4 and Fig. 5 respectively. In the three models, using the pre-trained word embedding word2vec gave better results than the randomly initialized ones. In general, CapsNet models' results were better than the CNN, followed by the CNN with a pooling layer. The CapsNet model with word2vec achieved the best results, with 92.21% accuracy and 92.20% F1 score, while the worst result was related to the CNN model with a pooling layer at 91.01% accuracy and an F1 score of 90.84%. By comparing the CNN models to each other, it became clear that the use of a pooling layer in the text classification tasks would not be a wise choice, at least in the context of this work.

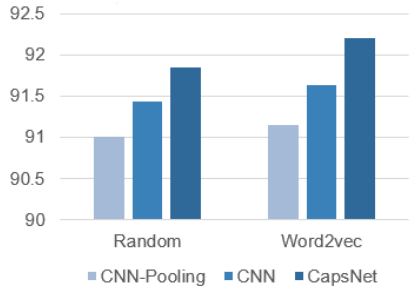

Fig. 4. Accuracy Comparison.

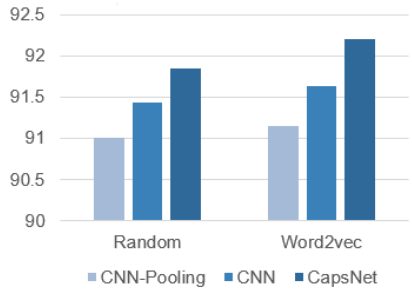

Fig. 5. F1-score Comparison.

VII. CONCLUSION

Securing software, hardware, data, and services has become a crucial part of any organization's management due to the increasing numbers of emerging attacks that threaten its security. In this study, the novel DL algorithm CapsNet was utilized along with Twitter data to provide accurate classification of tweets for the purposes of cyberthreat detection. A random search was used for hyperparameter tuning for random and word2vec embeddings. CapsNet model was built based on the hyperparameters was found after the random search, then the model was compared with two CNN architectures: a CNN baseline model without a pooling layer and a CNN baseline pooling model that included a pooling layer. The model was evaluated using accuracy and F1 score, and from the results, multiple remarks were gleaned. First, it was proved the better efficiency of routing-by-agreement compared to pooling. Second, in the three models, the pre-trained word embedding word2vec achieved better results than random embedding. Third, the proposed CapsNet model outperformed the strong competitor of CNN, with 92.21% accuracy and 92.2% F1 score.

Because there is always room for improvement, we plan to compare the obtained results with a recurrent neural network (RNN) model given that CapsNet introduces improvements on its architecture. In addition, we aim to examine replacing the CNN layers in the CapsNet with RNN to take advantage of dealing with tweets word-by-word rather than the whole tweet at once. In addition, we aim to test more embeddings, such as GloVe, Fasttext, BERT, and Elmo.

ACKNOWLEDGMENT

This work was supported by the Deanship of Scientific Research (DSR), King Abdulaziz University, Jeddah, under

grant No. (DG-1441-60-612). The authors, therefore, gratefully acknowledge the DSR technical and financial support.

REFERENCES

[1] H. Booth, D. Rike, and G. Witte, "The national vulnerability database (nvd): Overview," National Institute of Standards and Technology, Tech. Rep., 2013.

[2] O. Security, *Offensive Security's Exploit Database Archive*, (accessed May 31, 2020). [Online]. Available: https://www.exploit-db.com/

[3] S. Trabelsi, H. Plate, A. Abida, M. M. B. Aoun, A. Zouaoui, C. Missaoui, S. Gharbi, and A. Ayari, "Mining social networks for software vulnerabilities monitoring," in *2015 7th International Conference on New Technologies, Mobility and Security (NTMS)*. IEEE, 2015, pp. 1–7.

[4] C. Sabottke, O. Suciu, and T. Dumitraș, "Vulnerability disclosure in the age of social media: exploiting twitter for predicting real-world exploits," in *24th {USENIX} Security Symposium ({USENIX} Security 15)*, 2015, pp. 1041–1056.

[5] S. Samtani, R. Chinn, H. Chen, and J. F. Nunamaker Jr, "Exploring emerging hacker assets and key hackers for proactive cyber threat intelligence," *Journal of Management Information Systems*, vol. 34, no. 4, pp. 1023–1053, 2017.

[6] L. G. A. Rodriguez, J. S. Trazzi, V. Fossaluza, R. Campiolo, and D. M. Batista, "Analysis of vulnerability disclosure delays from the national vulnerability database," in *Anais do I Workshop de Segurança Cibernética em Dispositivos Conectados*. SBC, 2018.

[7] C. Sauerwein, C. Sillaber, M. M. Huber, A. Mussmann, and R. Breu, "The tweet advantage: An empirical analysis of 0-day vulnerability information shared on twitter," in *IFIP International Conference on ICT Systems Security and Privacy Protection*. Springer, 2018, pp. 201–215.

[8] R. Campiolo, L. A. F. Santos, D. M. Batista, and M. A. Gerosa, "Evaluating the utilization of twitter messages as a source of security alerts," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, 2013, pp. 942–943.

[9] D. Kergl, R. Roedler, and G. D. Rodosek, "Detection of zero day exploits using real-time social media streams," in *Advances in Nature and Biologically Inspired Computing*. Springer, 2016, pp. 405–416.

[10] M. Almukaynizi, E. Nunes, K. Dharaiya, M. Senguttuvan, J. Shakarian, and P. Shakarian, "Proactive identification of exploits in the wild through vulnerability mentions online," in *2017 International Conference on Cyber Conflict (CyCon US)*. IEEE, 2017, pp. 82–88.

[11] A. Hernández, V. Sanchez, G. Sánchez, H. Pérez, J. Olivares, K. Toscano, M. Nakano, and V. Martinez, "Security attack prediction based on user sentiment analysis of twitter data," in *2016 IEEE international conference on industrial technology (ICIT)*. IEEE, 2016, pp. 610–617.

[12] A. Sliva, K. Shu, and H. Liu, "Using social media to understand cyber attack behavior," in *International Conference on Applied Human Factors and Ergonomics*. Springer, 2018, pp. 636–645.

[13] A. Rodriguez and K. Okamura, "Generating real time cyber situational awareness information through social media data mining," in *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, vol. 2. IEEE, 2019, pp. 502–507.

[14] A. Hernandez-Suarez, G. Sanchez-Perez, K. Toscano-Medina, V. Martinez-Hernandez, H. Perez-Meana, J. Olivares-Mercado, and V. Sanchez, "Social sentiment sensor in twitter for predicting cyber-attacks using l1 regularization," *Sensors*, vol. 18, no. 5, p. 1380, 2018.

[15] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[16] C. Y. Chang, Z. Teng, and Y. Zhang, "Expectation-regulated neural model for event mention extraction," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016, pp. 400–410.

[17] G. Hinton., *Machine Learning, reddit*, (accessed May 31, 2020). [Online]. Available: https://www.reddit.com/r/MachineLearning/comments/2lmo0l/ama_geoffrey_hinton/clyj4jv/

[18] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Advances in neural information processing systems*, 2017, pp. 3856–3866.

[19] M. K. Patrick, A. F. Adekoya, A. A. Mighty, and B. Y. Edward, "Capsule networks–a survey," *Journal of King Saud University-Computer and Information Sciences*, 2019.

[20] J. Kim, S. Jang, E. Park, and S. Choi, "Text classification using capsules," *Neurocomputing*, vol. 376, pp. 214–221, 2020.

[21] W. Zhao, J. Ye, M. Yang, Z. Lei, S. Zhang, and Z. Zhao, "Investigating capsule networks with dynamic routing for text classification," *arXiv preprint arXiv:1804.00538*, 2018.

[22] H. W. Fentaw and T.-H. Kim, "Design and investigation of capsule networks for sentence classification," *Applied Sciences*, vol. 9, no. 11, p. 2200, 2019.

[23] Y. Du, X. Zhao, M. He, and W. Guo, "A novel capsule based hybrid neural network for sentiment classification," *IEEE Access*, vol. 7, pp. 39 321–39 328, 2019.

[24] Y. Wang, A. Sun, J. Han, Y. Liu, and X. Zhu, "Sentiment analysis by capsules," in *Proceedings of the 2018 world wide web conference*, 2018, pp. 1165–1174.

[25] M. H. Goldani, S. Momtazi, and R. Safabakhsh, "Detecting fake news with capsule neural networks," *arXiv preprint arXiv:2002.01030*, 2020.

[26] J. Liu, H. Lin, X. Liu, B. Xu, Y. Ren, Y. Diao, and L. Yang, "Transformer-based capsule network for stock movement prediction," in *Proceedings of the First Workshop on Financial Technology and Natural Language Processing*, 2019, pp. 66–73.

[27] P. Rathnayaka, S. Abeysinghe, C. Samarajeewa, I. Manchanayake, and M. Walpola, "Sentylic at iest 2018: Gated recurrent neural network and capsule network based approach for implicit emotion detection," *arXiv preprint arXiv:1809.01452*, 2018.

[28] H. Hettiarachchi and T. Ranasinghe, "Emoji powered capsule network to detect type and target of offensive posts in social media," in *Proceedings of RANLP*, 2019.

[29] A. Ritter, E. Wright, W. Casey, and T. Mitchell, "Weakly supervised extraction of computer security events from twitter," in *Proceedings of the 24th International Conference on World Wide Web*, 2015, pp. 896–905.

[30] F. Alves, A. Bettini, P. M. Ferreira, and A. Bessani, "Processing tweets for cybersecurity threat awareness," *arXiv preprint arXiv:1904.02072*, 2019.

[31] Z. Wang and Y. Zhang, "A neural model for joint event detection and summarization." in *IJCAI*, 2017, pp. 4158–4164.

[32] Y. Erkal, M. Sezgin, and S. Gunduz, "A new cyber security alert system for twitter," in *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2015, pp. 766–770.

[33] S. Yagcioglu, M. S. Seyfioglu, B. Citamak, B. Bardak, S. Guldamlasioglu, A. Yuksel, and E. I. Tatli, "Detecting cybersecurity events from noisy short text," *arXiv preprint arXiv:1904.05054*, 2019.

[34] N. Dionísio, F. Alves, P. M. Ferreira, and A. Bessani, "Cyberthreat detection from twitter using deep neural networks," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–8.

[35] A. Khikmatullaev, J. Lehmann, and K. Singh, "Capsule neural networks for text classification," Ph.D. dissertation, 04 2019.

[36] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[37] Google, "Colaboratory: frequently asked questions." (accessed May 31, 2020). [Online]. Available: https://research.google.com/colaboratory/faq.html

[38] F. Chollet *et al.*, "Keras documentation," *keras. io*, 2015.

[39] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[40] S. Wang, W. Zhou, and C. Jiang, "A survey of word embeddings based on deep learning," *Computing*, vol. 102, no. 3, pp. 717–740, 2020.

[41] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of

word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[42] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of machine learning research*, vol. 13, no. Feb, pp. 281–305, 2012.

[43] J. M. Torres, C. I. Comesaña, and P. J. García-Nieto, "Machine learning techniques applied to cybersecurity," *International Journal of Machine Learning and Cybernetics*, vol. 10, no. 10, pp. 2823–2836, 2019.