TEXCEPTION: A CHARACTER/WORD-LEVEL DEEP LEARNING MODEL FOR PHISHING URL DETECTION

Farid Tajaddodianfar *Jack W. Stokes* *Arun Gururajan*

Microsoft Corporation, One Microsoft Way, Redmond, WA 98052 USA

ABSTRACT

Phishing is the starting point for many cyberattacks that threaten the confidentiality, availability and integrity of enterprises' and consumers' data. The URL of a web page that hosts the attack provides a rich source of information to determine the maliciousness of the web server. In this work, we propose a novel deep learning architecture, Texception, that takes a URL as input and predicts whether it belongs to a phishing attack. Architecturally, Texception uses both character-level and word-level information from the incoming URL and does not depend on manually crafted features or feature engineering. This makes it different from classical approaches. In addition, Texception benefits from multiple parallel convolutional layers and can grow deeper or wider. We show that this flexibility enables Texception to generalize better for new URLs. Our results on production data show that Texception is able to significantly outperform a traditional text classification method by increasing the true positive rate by 126.7% at an extremely low false positive rate (0.01%) which is crucial for our model's healthy operation at internet scale.

***Index Terms*—** Phishing, Detection, Character-Level, Deep Learning, Word Embedding

1. INTRODUCTION

Many computer security-related services need to verify that a URL (Uniform Resource Locator) does not provide the location of a malicious web page on the internet [1]. In this paper, we specifically focus on leveraging URLs for detecting phishing attacks. Phishing can be broadly defined as a type of social engineering attack via electronic channels that tricks humans into performing certain actions for the attacker's benefit [2]. These actions could include harvesting passwords or bank account numbers which are then sold on the black market. To this end, we present a novel character-level deep learning model which learns to detect malicious URLs associated with these phishing web sites.

There are many signals that one could consider as features for models that detect phishing pages including the requestor's URL, static HTML content, DOM of the page, and screenshot of the page. Most of these raw signals need a significant level of transformation in order to be leveraged as useful features, which often requires a non-trivial amount of time to be spent on feature engineering.

In this work, we show that by using a novel character-level deep learning approach, we can eliminate the feature engineering step, while significantly improving the efficacy of the phishing detection classifier. To demonstrate this, we utilize the URL as the only input feature. Two main reasons for classifying URLs include:

1. In a real-time phishing detection scenario, where minimal latency is of paramount importance, URLs are often the primary signals that can be captured. This approach does not preclude the possibility of using other signals, but usually the minimum bar that gives the most information is almost always the requested URL.
2. The other scenario where the URL is useful is for detecting "potential" phishing domains. These are essentially newly registered domains that have not yet hosted malicious content, but have a high probability of doing so in the near future. In this scenario where the domain is not hosting any content, the only signal that we can leverage is the URL.

Character-Level Deep Learning. Character-level deep learning text classification has been previously studied in the literature [3, 4, 5]. Le, et al. [6] proposed URLNet as a deep learning model for malicious URL classification. In this work, we describe Texception as a character/word-level deep learning model for text classification that inputs a raw URL as text and performs a binary classification. In Section 5, we describe how Texception differs from previous character-level models.

Contextual Embedding. Contextual embedding approaches, such as Word2Vec [7] and FastText [8, 9], have widely been used in recent years in the field of Natural Language Processing (NLP). The basic idea in contextual embedding is to construct a vocabulary of words in a text corpus and assign a low-dimensional randomly initialized dense vector to each word. During training, which is unsupervised, each sentence from the corpus is tokenized to extract words, and the algorithm targets either predicting the surrounding words for a given word (i.e., skipgram architecture) or predicting a single word given its surrounding words or context (i.e., Continuous Bag of Words or CBOW architecture). Vectors assigned to each word are adjusted to optimize for this task, and the trained embedding vectors are expected to reflect contextual relationships. After training, words that have similar meaning are closely located in the embedding space. In many NLP applications, word embeddings trained on large text corpuses are used as inputs for the downstream model. In our phishing detection problem, however, such pre-trained vectors are not useful because words that appear in our corpus of URLs are not similar to any language corpus. Therefore, we train the word embeddings on our training dataset and use them as input to the Texception model. This is discussed in the Section 3.

In Section 4, we describe our experiments and results which illustrate the superiority of the proposed approach at extremely low false positive rates (FPRs). To be able to function at internet scale, we operate our models at an $FPR = 0.01\%$ at which the best Texception model provides an increase of 126.7% in the true positive rate (TPR) compared to a traditional baseline model.

The remainder of this paper is organized as follows. Section 2 describes the experimental data that was used as the reference to train our proposed model as well as the other baseline ML and deep learning models. Section 3 expounds on the proposed modeling approach. This is followed by a section on numerical evaluation,

where the proposed approach is compared against a set of baseline models. Finally, Sections 5 and 6 provide the related works in this area and concluding remarks, respectively.

2. EXPERIMENTAL DATA

We use Microsoft's anonymized browsing telemetry data which is primarily comprised of the URL and a binary phishing grade (phishing/benign). The grades serve as the labels for training and evaluating our model. We retrospectively collected six weeks of data, out of which we use the first two weeks as the training set. Due to severe class imbalance (only 0.01% positive class), we downsample the benign class to improve the training set's balance to 5% for the positive class in total. The resulting training set contains 1.7M samples from which we sample 20% of the instances based on stratified sampling to construct a validation set with the same class balance. We also use a test set containing 20M instances sampled directly from the production pipeline without any preprocessing to serve as a representative set of the real production data.

3. TEXCEPTION MODEL

Texception uses both character and word embeddings that are known to be more efficient than traditional Bag-Of-Words (BOW) techniques. In this section, we first explain training the FastText contextual word embeddings as well as building the character embeddings. Next, we describe the individual Texception block and the overall architecture.

3.1. FastText Word Embedding

We employ the FastText model to generate the contextual word embeddings. One advantage of FastText compared to Word2Vec is that the FastText model treats each word as a bag of its n-grams. This enables FastText to generate embedding vectors for previously unseen words. We use the GenSim [10] Python package for the FastText model implementation. To train the model, we ignore labels and tokenize each URL based on special characters (including ".?/-_\=%&@+;") to obtain words. We also include these special characters in the model vocabulary as suggested in [6]. The FastText model builds a vocabulary of words that are observed more than a user-defined minimum in the training set. During training, each URL is treated as a sentence after tokenization. We setup a skip-gram architecture with the parameters given in Table 1.

3.2. Character Embedding

For the character embeddings, we first build an alphabet of all characters which occur in the training set and then assign a lookup table that takes the integer index of each character in the alphabet and returns a low-dimensional dense vector which is trained during the network training. Two additional indexes are reserved for unknown characters and the empty space. We also need to define the maximum number of characters to be processed in each URL. Longer URLs are trimmed and shorter ones are padded with the empty character to meet this maximum value. With these hyperparameter settings, each input URL is mapped to a dense matrix of character embeddings.

3.3. Texception Block

By intuition, a 1D convolution of filter size N on the sequence of character embeddings is similar to extracting an N-gram from input text (Figure 1). Inspired by Inception [11], which received state-of-the-art performance in image processing, we propose using a Texception module as the main building block. Our intuition is that multiple convolutional layers in parallel each with different filter sizes should be able to better capture text patterns compared to multiple consecutive layers with a fixed filter size. First, we define a basic block as shown in Figure 2 where F and D are the list of filter sizes and dilation sizes for the parallel 1D convolutional layers. Each convolutional layer is followed by Batch Normalization, Max Pooling, and ReLu blocks.

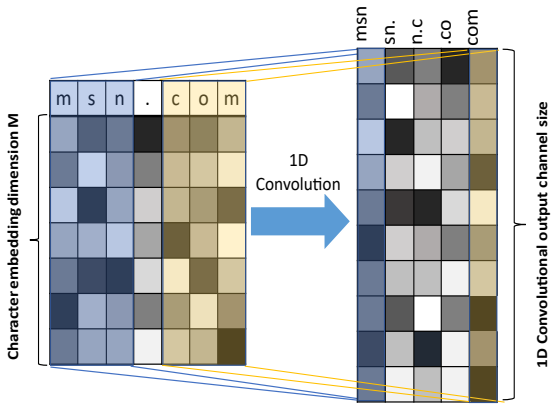

Fig. 1. 1D convolutional operator with filter size 3 and no zero-padding applied to a sequence of character embeddings. This operation maps every set of 3 characters to a new vector with size equal to the output channel size of the operator. Intuitively, this is similar to extracting 3-grams.

3.4. Full Texception Model

The full model is comprised of two parallel paths, one for extracting character-level information and the other for working with word-level information. The character-level path first converts a raw URL into a dense matrix of character embedding vectors as described in Section 3.2. Next, multiple layers of Texception blocks are applied followed by an Adaptive Max Pooling layer [12] that limits the dimension of the last Texception block's output to a user defined value.

We define a word-embedding layer that returns a dense vector for each word in the FastText model vocabulary, reserving one index for unknown words and one for empty words. We can choose to initialize the word embeddings by the weights from the FastText model and freeze them so that those weights do not change during the training. Other options are to initialize words with FastText weights and allow them to adjust during training, or simply not use FastText weights and randomly initialize weights.

The word-level path first tokenizes the incoming URL using the same regular expression (regex) key that has been used in building the FastText model. Then, the list of words in the URL is matched with the maximum word length, so that longer word lists are trimmed and shorter ones are zero padded. This list then passes through the word embedding layer to produce a sequence of word vectors which is followed by multiple Texception blocks. The output of the word path is also limited using an Adaptive Max Pooling operator.

Outputs from the character-level and word-level paths are concatenated and input to multiple fully connected layers that

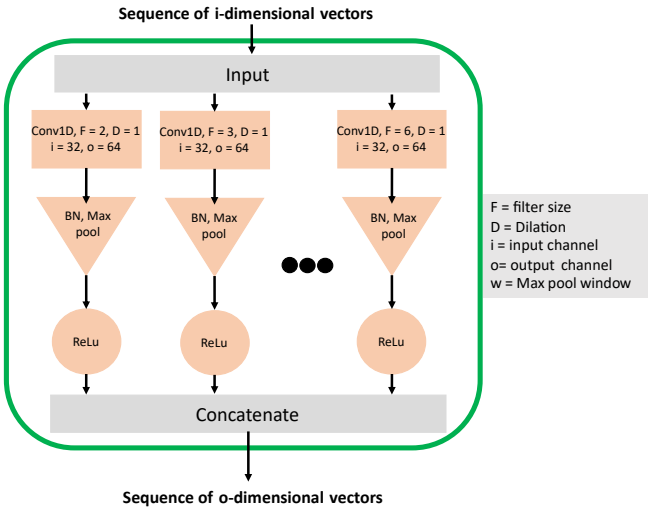

Fig. 2. Texeption Block. A list of filter sizes and dilation parameters are provided to the block. The number of parallel layers equals the number of filters times the number of dilations. Input and output channel sizes are user defined. Numbers are given as example.

eventually produce a class probability. Layers with more nodes have a larger dropout probability. Figure 3 displays a schematic of the full Texception model. Note that users can choose multiple structures: deeper networks with more consecutive Texception blocks, wider networks with more parallel layers within the block, using words with pre-trained FastText weights versus random weights or even not using words. This framework provides users with sufficient flexibility to experiment and find the best architecture.

4. NUMERICAL EVALUATION

In this section, we next evaluate the proposed Texception models on the task of detecting malicious phishing URLs from the dataset described in Section 2.

Setup. All models are trained using the PyTorch [13] deep learning framework. A Binary Cross Entropy loss function along with a SGD optimizer with momentum equal to 0.9 and learning rate initialized at 0.01 are used. Training is performed for 30 epochs with the minibatch size set to 128, and at the end of each epoch, validation scores are obtained. A model with the best validation loss is returned as the final model of each experiment. The learning rate is also halved every 5 epochs.

Baselines. We also consider two baseline models for evaluating the performance of Texception. The URLNet model [6] is also a character-level CNN which was proposed for detecting malicious URLs. We obtained the code for URLNet from the online repository published by the authors and executed it on our data using its default parameters. The other baseline is a Logistic Regression (LR) model trained and cross-validated on the aforementioned training set and evaluated on the same test set. The LR model is based on manually crafted features, which is essentially a bag of character 4-grams (including trigrams, bigrams and unigrams) of the URL. We also restrict the vocabulary to 100K n-grams to avoid an extremely sparse feature space to prevent overfitting. In addition, the training process for the LR model includes a weighted combination of L1 and L2-norm-based regularizers (Lasso and Shrinkage operators). With

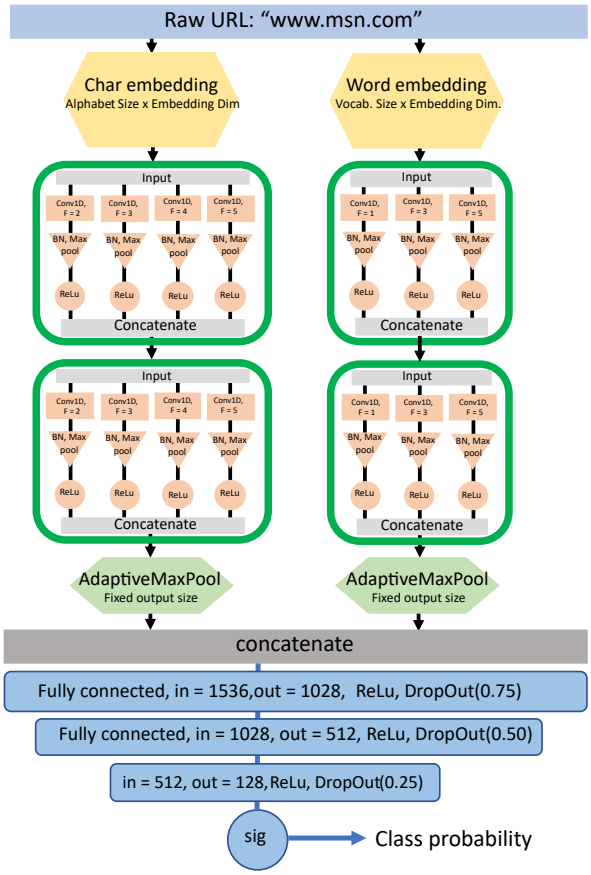

Fig. 3. Texception model comprised of two layers of consecutive Texception blocks. The character-level path uses filters of size 2,3,4, and 5. In word path, filters of size 1,3, and 5 are used. All dilation parameters are set to one.

Parameter		value
Characters Branch	embedding dimension	32
	number of blocks	1
	block filters	[2,3,4,5]
	Adaptive MaxPool output	32,32
	maximum characters	1000
Words Branch	embedding dimension	32
	number of blocks	1
	block filters	[1,3,5]
	Adaptive MaxPool output	32,16
	maximum words	50
FastText Model	minimum words to include	50
	vocabulary size	120000
	window size	7
	n-grams	2-6
	embedding dimension	32
	epochs trained	30

Table 1. Hyperparameters used for experimentation.

Model	TP Rate (%)	Error Rate (%)
Texception_character_only	31.2	0.37
Texception_words_random	**47.95**	**0.28**
Texception_FastText_Frozen	42.12	0.32
Texception_FastText_nonFrozen	45.83	0.30
Baseline_LR	21.15	0.42
Baseline_URLNet	12.3	0.48

Table 2. True Positive (TP) and Error rates on the test set measured at fixed 0.01% False Positive (FP) rate.

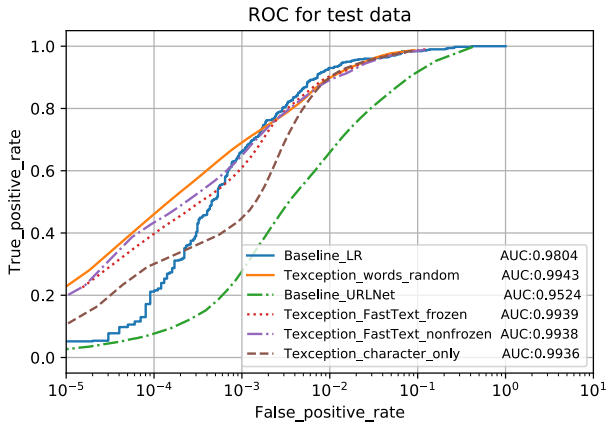

Fig. 4. ROC curve obtained by scoring the trained models on the test data set.

this setup, we use the regularizer weights as the hyperparameters, and use the validation set to select the best hyperparameters. We generate the results for LR on the test set using the best combination of hyperparameters.

Performance Results. Figure 4 shows the ROC (Receiver Operating Characteristic) curve obtained by scoring the trained models on the test set. In practice, we have an extremely tight tolerance for the false positive rate, and generally we choose our models to operate at an $FPR \leq 0.01\%$. Figure 4 shows that all versions of the Texception model are able to outperform the baselines at these very low FP rates. Table 2 shows the true positive and error rates for all evaluated models measured at a fixed $FPR = 0.01\%$. The Texception model that uses randomly initialized word weights yields a 47.95% TPR, which is an increase of 126.7% over the TP rate of the baseline LR model at the same FP rate.

More specifically, it is clear from Table 2 that the TP rate of the Texception model with randomly initialized word weights (47.95%) is a 53.7% increase compared to the version which uses only character embeddings (31.2%). In addition, adjusting the pre-trained word embeddings during training results in better performance. Therefore, a Texception model with randomly initialized word embeddings performs better than those with pre-trained FastText embeddings. One explanation for this observation is that the FastText model is trained to capture contextual similarity which is different from the ultimate task of predicting the class probabilities. In the presence of abundant labeled data, learning word embeddings through the general training task better optimizes the embedding weights for the classification task at hand.

5. RELATED WORK

Detecting phishing pages using URLs is an understudied problem, but some previous work does exist. Our Texception model is most closely related to the URLNet model proposed in [6]. However, Texception is architecturally different from URLNet in two ways. One main difference is that the Texception model benefits from pretrained contextual word embeddings. The other difference is that Texception uses multiple layers of the Texception block and Adaptive Max Pooling to select the top features to retain at the end of each layer.

Texception is one type of character-level CNN, and there are a number of important works in this area [3, 4, 5]. The character-level CNN was initially proposed by Zhang et al. [4]. Other important character-level CNN models include [3, 5]. The most interesting property of these models is that they are not dependent on manually extracted features and are able to use raw text as input to predict the class. Previous character-level CNN models are mainly comprised of one or multiple layers of convolutional layers with fixed filter size followed by several fully connected layers. The use of word embeddings to improve the classification task has also been reported [14]. Generally, it is expected that deeper networks can capture more higher-level patterns resulting in better learning capability [3]. Our attempts to replicate that approach for URL-based phishing detection with character-level DNNs resulted in over-parameterized deep models that were not able to beat our baseline models. Texception, presented in this paper, differs from the previous CNNs because it allows for parallel CNN layers. Also Texception offers the ability to grow deeper or wider with the ability to use contextual word embeddings as input.

6. CONCLUSIONS

In this work, we present a novel character-level neural network for detecting phishing web pages. The Texception model learns sequential patterns associated with URLs.

The results are quite promising on a real dataset collected by the Microsoft SmartScreen service. The model outperforms previously proposed deep learning models for detecting phishing web site URLs. Most importantly, these results are obtained with severe class imbalance which is often encountered by users in practice. Therefore, we believe that Texception can provide significantly improved protection against phishing attacks in the wild.

7. ACKNOWLEDGEMENT

The authors thank Christian Seifert and Geoff McDonald of Microsoft for fruitful discussions.

8. REFERENCES

[1] D. Sahoo, C. Liu, and S. C. H. Hoi, "Malicious URL Detection using Machine Learning: A Survey," vol. 1, no. 1, pp. 1–37, 2017. [Online]. Available: http://arxiv.org/abs/1701.07179

[2] M. Khonji, Y. Iraqi, and A. Jones, "Phishing detection: A literature survey," *IEEE Communications Surveys and Tutorials*, vol. 15, no. 4, pp. 2091–2121, 2013.

[3] A. Conneau, H. Schwenk, Y. Le Cun, and L. Barrault, "Very Deep Convolutional Networks for Text Classification," Tech. Rep., 2017. [Online]. Available: https://arxiv.org/pdf/1606.01781.pdf

[4] X. Zhang, J. Zhao, and Y. Lecun, "Character-level Convolutional Networks for Text," pp. 1–9, 2015.

[5] X. Zhang and Y. Lecun, "Text Understanding from Scratch," Tech. Rep., 2016. [Online]. Available: http://www.libreoffice.org/

[6] H. Le, Q. Pham, D. Sahoo, and S. C. H. Hoi, "URLNet: Learning a URL Representation with Deep Learning for Malicious URL Detection," Tech. Rep., 2018. [Online]. Available: https://doi.org/10.475/123_4

[7] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Distributed-Representations-of-Words-and-Phrases-and-Their-Compositionality," in *Advances in neural information processing systems, NIPS*, 2013, pp. 3111–3119.

[8] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching Word Vectors with Subword Information," vol. 5, pp. 135–146, 2017.

[9] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," *15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017 - Proceedings of Conference*, vol. 2, pp. 427–431, 2017.

[10] R. Rehurek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," *Proceedings of LREC 2010 workshop New Challenges*, 2004.

[11] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 07-12-June, pp. 1–9, 2015.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[13] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *31st Conference on Neural Information Processing Systems (NIPS 2017)*, 2017.

[14] D. Hendler, S. Kels, and A. Rubin, "Detecting Malicious PowerShell Scripts Using Contextual Embeddings," Tech. Rep., 2019. [Online]. Available: https://www.powershellgallery.com/