Transformer-Based Deep Learning Architecture for Android Malware Classification and TTP Prediction

Andrew Balch

Governor's School for Science and Technology

2020 - 2021

I have read this document and approved the content.

_____

Mentor Signature

Mr. Wes Jordan

903 Enterprise Pkwy Ste 200, Hampton, Virginia, 23666, United States

Abstract

Transformers promise groundbreaking advances in deep learning by allowing models to generalize and form attention between elements in a sequence better than ever before. While typically constrained to Natural Language Processing (NLP) tasks, this study applies Transformers to Android malware proposing a unique, multi-task approach to malware classification and technique prediction from time-series dynamic analysis and context from MITRE's ATT&CK. The model would use the encoder outputs for classification and the decoder element to predict technique chains with a diverging transfer learning element allowing the two to generalize, then separate and fine-tune. Compared to previous research into deep learning and cybersecurity in traditional deep, convolutional, or recurrent neural networks, expected results from the proposed architecture improve on traditional networks yielding 98% accuracy in malware classification and 99.9% accuracy in technique prediction (**on training data only**). The expected findings support the utility of Transformers in use cases beyond NLP and provide a method for proactive anti-malware to detect malicious activity in real-time via malware techniques. Unfortunately, the study could not be completed as planned due to unforeseen time and resource constraints.

Transformer-Based Deep Learning Architecture for Android Malware Classification and TTP Prediction

New Android malware samples totaled 10.5 million in 2019 (Clement, 2020). Additionally, Positive Technologies (2019) has found that 43% of all Android apps contain dangerous vulnerabilities and that such risks are typically made up of many smaller oversights, spread throughout the application. The always-changing nature of malware tactics, techniques, and procedures (TTPs) and the large distribution of the vulnerabilities they exploit makes vicious code progressively difficult, and expensive, to detect and foil. For this reason, deep learning is becoming increasingly prevalent in Android cybersecurity solutions. A subset of machine learning, deep learning takes in large amounts of "training" data and learns complex "mappings" from the inputs to a desired output. The layers of artificial neurons that deep learning models are made up of can effectively generalize and form relationships between the many different features in the training dataset en route to producing the output, making them effective for detecting malicious trends spread throughout the system. However, the architecture and size of neuron layers can lead to drastically different results in different use cases. The goal of the proposed study is to design and train a novel deep learning architecture utilizing encoder-decoder and attention elements and information from MITRE's ATT&CK to accurately and precisely classify malware in an Android device and predict TTP chains using collected time-series data of malware actions within the system, with a stretch goal being to reliably deploy it on real Android devices.

Alzaylee, Yerima, Sezer (2019) implemented a vanilla deep learning classifier with the goal of labeling apps as benign or malicious. They placed a focus on the effectiveness of different forms of input generation, a practice where user inputs are simulated to increase code

coverage of dynamic analysis. It studied the classification of 30,000 samples from Intel Security running on real devices as benign or malicious. The dataset was made up of dynamically collected, binary features each indicating its presence in the sample. The model boasted a 99.6% detection rate using stateful input generation and 97.8% using stateless (random) (Alzaylee, Yerima & Sezer, 2019). Another study proposed semi-supervised learning on features from over 17,000 app samples dynamically analyzed using CopperDroid (Tam et al., 2015) to label apps as Adware, Banking, SMS, Riskware, and Benign. The paper argued that traditional, supervised deep learning models are expensive and difficult to develop as a result of having to find and label data and proposed a Pseudo-Labeled Deep Neural Network (PLDNN) as an alternative (Mahdavifar et al., 2020). Feature vectors were made up of L2-normalized frequency values for detected behaviors. Their approach involved training an initial supervised model on entirely labeled data, using that model to generate pseudo-labels for the remaining unlabeled data, and then retraining the model on labeled and pseudo-labeled instances. Although boasting an F1-score of 97.84% on their final semi-supervised model, the study assumed the pseudo-labeled instances are ground truth data which introduces bias into training and evaluation (Mahdavifar et al., 2020). The authors proposed additional study of the system using state-of-the-art RNNs or CNNs. Bibi et al. (2020) proposed a recurrent neural network (RNN) leveraging gated recurrent unit (GRU) cells (Cho et al. 2014). This is compared to an RNN with long-short term memory (LSTM) cells (Gers, Schmidhuber & Cummins, 1999), a convolutional neural network (CNN) (LeCun et al., 1989), and a vanilla deep learning model all with the task of classifying app samples as Trojan, Backdoor, or Benign. The models were trained on 150 of the most common features of 1,900 extracted from the 38,842 samples and it was found that the GRU RNN performed the best with 98.99% detection accuracy (Bibi et al., 2020). Notably, no

hyperparameter tuning is described in the paper with the number and size of layers seemingly arbitrarily selected and others like the learning rate remaining constant, meaning that not all of the models may have had their best shot at success. To better defend against new and innovative malware samples, the authors endorsed the use of varied architectures to ensure the best results in different use cases (Bibi et al., 2020).

MITRE's ATT&CK is a well-known knowledge base of techniques used by malicious actors in systems like Android, Windows, or iOS with a focus on archetyping how such actors interact with the system itself (Storm, 2018). Tactics (the "why" or end goal of the action) are made up of techniques (the "how" of achieving a malicious goal) that together give cybersecurity professionals valuable context into a malicious action (Storm, 2018). Noor et al. (2019) noted that current systems diagnose a threat with low-level Indicators of Compromise (IoCs) such as hashes, IP addresses, and domain names whereas the use of higher-level IoCs such as TTPs can yield accuracy improvements in threat detection. They proposed a machine learning system to semantically correlate low-level threat artifacts to ATT&CK TTPs using latent semantic indexing (Deerwester et al., 1990) and used them to suggest detection mechanisms and predict threats via a belief network (Noor et al., 2019). It was found that the framework had a 92% accuracy in threat identification. Another study applying ATT&CK in Windows systems, Kok et al. (2020) aimed to predict ATT&CK TTPs in Windows event logs from known and unknown cyberattacks, applying supervised deep learning to the former and unsupervised deep learning (autoencoder) and clustering to the latter. Features were extracted and placed into 13-minute time windows with a 2-minute time step and over-normalized to between -0.07 and 1.02 (Kok et al., 2020). The deep learning model was 99.7% accurate on the training set and the authors include the following recommendations for future work: 1) A model that can predict threats as well as technique

sequences 2) Determine the most important logs and entry types 3) Repeat with data from real attacks 4) Explore options for real-time inference 5) Explore other options for feature selection and dimensionality reduction (Kok et al., 2020).

Most studies found applying deep learning in the field of Android cybersecurity focused on simply classifying Android malware. Although they are becoming increasingly effective and are critical works in their own right, such studies offer reactive solutions that identify malicious activity only after a full analysis of the application and its interactions with the user and the system. None, as of yet, have been found detailing proactive solutions that focus on not only classifying the presence of malware but also accurately forecasting its likely next steps in a system in pursuit of a predicted threat. When it comes to studies involving MITRE's ATT&CK, they emphasize the relationships between TTPs and threats, unlike the studies involving Android. However, the ones presented focus on only either predicting threats based on TTPs or simply detecting TTPs present, all in non-mobile systems. No studies have been found that pursue the prediction of not only threats but also chains of malware TTPs in time-series in exclusively Android systems. Finally, studies in both contexts do not implement both encoder-decoder models and attention mechanisms (Bahdanau et al., 2015). Encoder-decoder models and attention mechanisms are state-of-the-art deep learning architectural elements that have shown promising results in related use cases, such as the Transformer model in machine translation (Vaswani et al., 2017).

Any increase in the accuracy and precision of techniques used to counter malware directly benefits and protects the millions of people and businesses around the world that rely on Android. Whether that comes in the form of new or improved emulators, dynamic analysis techniques, or deep learning models that all build on top of each other to identify malicious

intent, progress in the field of Android cybersecurity is always worthwhile as it has such far-reaching impacts. Outside of this, however, the proposed solution could spur the development of proactive anti-malware solutions. Rather than hogging vast amounts of hardware resources constantly scanning and profiling the system to catch vicious code like current solutions, proactive solutions could utilize effective predictions of threats and TTP chains, as proposed here, to dynamically focus resources and adjust permissions to foil malware and protect compromised apps, potentially leading to increased efficiency and effectiveness compared to current solutions.

The hypothesis is that the encoder-decoder model coupled with attention mechanisms and added context from MITRE's ATT&CK repository will enable the system to perform at or better than the models from Mahdavifar et al. (2020) and Bibi et al. (2020) at threat classification and Kok et al. (2020) at detecting threat techniques according to metrics outlined in Tharwat (2017) on classification assessment.

Methods and Materials

This state-of-the-art Deep Learning model is predominantly a proof of concept for the application of similarly advanced architectures to be implemented in the field of Android cybersecurity. It has the potential to improve upon malware identification and classification and introduce prediction of malware Tactics, Techniques, and Procedures (TTPs) in this use case. Similar technology could be implemented in the real world via proactive anti-malware solutions on Android. This would allow for efficient utilization of system logs and resources to actively foil steps in a malicious app's TTP chain, effectively blocking malicious activities before they occur. A model of the proposed architecture could be ideal for such a use case where attention to a few key behaviors across large system logs and effective generalization of how TTPs are implemented across malware types, families, and samples are critical to handle ever-evolving malware. Vaswani et al.'s Transformer model has shown great promise in the use case of language to language translation where long-term attention and effective generalization is necessary, making it a promising candidate for this use case (2017).

Materials

Datasets

The three datasets used were AndMal2020 (Rahali et al., 2020), MalDroid2020 (Mahdavifar et al., 2020), and MITRE's ATT&CK. AndMal and MalDroid are the most comprehensive and recent catalogs of Android malware samples. AndMal is a dataset of 400,000 Android app samples classified into malware families and benign. MalDroid is a set of static features and dynamically-collected behaviors for over 17,000 Android app samples separated into the categories of Adware, Banking malware, SMS malware, Riskware, and Benign.

ATT&CK has been described in the introduction and is the only readily available dataset containing Android malware TTPs by family.

Software

The model was developed using the Python 3.7 programming language through a Jupyter Notebook for its ease-of-use and readability. Four major Python software libraries were used: TensorFlow is one of the most popular options for developing and training Deep and Machine Learning models. Keras is a library that is built into TensorFlow but will be used on its own for hyperparameter tuning. Pandas is a library that provides a useful set of tools for working with structured data in Python. Scikit-learn is a Machine Learning library like TensorFlow and is intended to be used for miscellaneous data preprocessing and train-test splits of the datasets. NumPy was also used as it extends Python's mathematical functionality. The model was developed and intended to be trained on a Windows 10 Pro 64-bit system.

Hardware

The hardware used to develop and intended train the model is a laptop equipped with 16 GB LPDDR4 2666 MHz memory, NVIDIA GTX 1660Ti 6GB Graphical Processing Unit (GPU), Intel Core i7-9750H processor, and 2 x 512 GB NVMe Solid State Drives (SSDs). Although better performance could be achieved on a machine with increased processing power, the one described above is on-hand and theoretically capable of the task. If storage or training time becomes an issue, however, a cloud storage solution and/or a virtual machine running an improved GPU or Tensor Processing Unit (TPU) can be configured on the Google Cloud Platform.

Procedure

Data Science

Sample file hashes in MalDroid were translated from SHA-256 to MD5 using the VirusTotal application programming interface (API) (Academic API key provided by VirusTotal). These MalDroid MD5 hashes were then intended to be cross-referenced with AndMal's MD5 hashes, therefore labeling MalDroid samples with AndMal families. Families present in the labeled MalDroid dataset will next be cross-referenced with those represented in MITRE's ATT&CK dataset. The process will provide a trimmed dataset of MalDroid samples that are matched with their malware family and, subsequently, techniques recorded in ATT&CK. This is a necessity as no single dataset has both dynamically-collected behaviors and TTP chains, with the only thing in common being the malware family. Next, the behavioral analysis of samples from the final dataset were pulled from MalDroid's remote repository, dropping any that raise an exception on opening/extracting or exceed GitHub's per-file limit of 100 Megabytes (MB). All remaining samples were cleaned of formatting abnormalities and tokenized into vectors of indices representing a string of data in a reference vocabulary. Any imbalance in malware class will also be noted and compensated for later.

Model and Infrastructure Building

The Deep Learning model was built using TensorFlow. The model will have a Transformer element for TTP/technique prediction from time-series, dynamically-collected behaviors following the structure outlined in Vaswani et al. (2017). Branching off of the Encoder element of the Transformer is planned to be a Deep Neural Network (DNN) Classifier for malware class inference. The model will supplement the static malware features with the encoded dynamic features from the Transformer to aid in classification, allowing the model to learn both tasks in parallel, aiding in the Encoder's ability to generalize its input. A 'diverging' transfer learning element will also be implemented to guide the learning of the Transformer and

the Classifier, described as follows: When the accuracy or loss of the model reaches a point of diminishing returns between training epochs, model weights will be frozen and duplicated, producing two independent models, one for classification and the other for TTP/technique prediction. In theory allowing the model to effectively generalize but not form an overreliance on one use case at the expense of the other. Then, an input pipeline was developed with TensorFlow. A best practice in Deep Learning, an input pipeline allows for efficient and parallel staging, preprocessing, shuffling, windowing, and serving data in batches and epochs to the model for training. An input pipeline will drastically improve training time, as it allows the model to train on a sample while the system is preparing the next one. This will also be where necessary adjustment for class imbalance was done through oversampling in the pipeline (TensorFlow, 2021). If storage space or ingestion bandwidth is an issue, Google Cloud Storage can be used instead of the local filesystem as it allows for much larger storage space and parallel ingestion from multiple buckets.

Hyperparameter Tuning

Deep Learning models have many variables outside of the overall architecture and training data that affect their accuracy. These are called hyperparameters and some examples of them are the number of layers in a network, the number of neurons in a layer, the batch size to train the model on, the optimizer function that handles backpropagation (how the model learns), etc. Hyperparameter tuning via Keras allows for A/B testing of such variables by repeatedly training the model on a few batches, evaluating it, and then training it again with different hyperparameters to find the most optimal configuration. The process will also be used to A/B test the proposed transfer learning element and shuffled and unshuffled TTPs in training. The latter

will be done to decrease the model's reliance on the order of TTPs, as they are not in time-series but the Transformer will treat them as such.

Finally, full training of the model will take place with the optimized hyperparameters. If necessary, this is where a Google Cloud Platform virtual machine with increased processing power can be configured and utilized if training time becomes an issue.

Evaluation

Evaluation of the model will be done on a set of holdout samples not introduced during training to mitigate bias. The main metrics collected are as follows, derived from Tharwat (2017):

- Accuracy - the number of samples correctly predicted over the total number of samples.
- Precision - the number of True Positives over Total Positives.
- Recall, True Positive Rate, and Sensitivity - the number of True Positives over the sum of True Positives and False Negatives.
- Inverse Recall, Specificity, and True Negative Rate- the number of True Negatives over the sum of True Negatives and False Positives.
- False Positive Rate and Fallout - the number of False Positives over the sum of False Positives and True Negatives.
- False Negative Rate and Miss Rate - the number of False Negatives over the sum of False Negatives and True Positives.
- F1-Score and F-Measure - the harmonic mean of precision and recall. This is another measure of accuracy.

These metrics will be compared against the averages of those from Mahdavifar et al. (2020) and Bibi et al. (2020) for malware classification and Kok et al. (2020) for TTP/technique inference.

Expected Results

The proposed state-of-the-art Deep Learning model will be able to effectively classify malware samples from the MalDroid dataset. Mahdavifar et al. (2020) is the main benchmark for comparison in this use case as it uses the same ground truth dataset. The proposed model will have an overall accuracy that exceeds 97.4% on a validation dataset when trained on 5000 samples (Figure 1). This result is comparable to that of Mahdavifar et al. (2020) on the same size training dataset. It will also have per-class true positive rates (TPRs) that exceed those of Mahdavifar et al. (2020) with a smaller standard deviation on a validation dataset when trained on 1000 samples (Table 1). Overall model evaluation metrics (Accuracy, Precision, Recall, F1-Score, False Positive/Negative Rate) will be an improvement over models from both Mahdavifar et al. (2020) and Bibi et al. (2020) (Figure 2).

When used to detect the presence of MITRE ATT&CK techniques, the described model will have an overall accuracy that exceeds 99.7% on training data only, making it comparable to the Deep Neural Network from Kok et al. (2020) (Figure 3). Per-technique performance can not be effectively evaluated, as Kok et al. (2020) involved ATT&CK techniques from exclusively Windows machines.

Discussion

Implications of Expected Results

The purpose of this study was to develop and train a transformer-based deep learning architecture using behavioral analysis of Android malware and added context from MITRE's ATT&CK to classify malware and predict TTPs. It was hypothesized that the proposed architecture would perform at or better than models from Mahdavifar et al. (2020) and Bibi et al. (2020) at threat classification and Kok et al. (2020) at detecting threat techniques according to metrics outlined in Tharwat (2017) on classification assessment. Formal results could not be reached, so expected results are being used. Expected results suggest that the proposed model will outperform these baseline models, supporting the original hypothesis and showing the effectiveness of encoder-decoder architectures and attention mechanisms in diagnosing malware.

These expected results would be supported by existing studies detailing the usefulness of Transformer architectures in language to language translation (Vaswani et al., 2017) and sentence classification (Devlin et al., 2018), showing marked benefits over alternatives such as RNNs and CNNs. MalDroid training data, while structured as JSON, has many similarities to Natural Language Processing (NLP) tasks. This data was treated as plain text and a dictionary for tokenization was created, breaking the hundreds of thousands of lines of data per sample into comprehensible building blocks for the model to understand. This approach is similar to how these studies worked with corpora. TTP prediction would have these vectors of tokenized data run through a full Transformer, thereby translating them to a sequence of techniques, consistent with Vaswani et al. (2017) who were able to effectively take vectors of one language and translate them into another. Similarly, classification utilized the encoder element only with a

dense output layer for the output, similar to how Devlin et al. (2018) were able to classify sentences.

Just as studies from Vaswani et al. (2017) and Devlin et al. (2018) support the proposed architecture, the expected results would reinforce the usefulness of Transformer models in non-NLP use cases such as image classification shown by Dosovitskiy et al. (2020) or multimodal information understanding in Google's new Multitask Unified Model (MUM) (Nayak, 2021). As no existing studies were found implementing Transformer architectures in Android malware classification and TTP prediction, the expected results do not differ from any other studies.

Issues and Limitations

The expected results for the proposed study were not reached for the following reasons:

Training Overhead

The main barrier to training the created architecture was the immense resource (particularly system/accelerator memory) cost of Transformer models, which was underestimated during planning. The model was intended to be trained on a local machine (16 GB system RAM, 6 GB Video RAM), but training was moved to a Google Colab environment where ~12 GB system RAM and Video RAM are freely available. This also proved insufficient. For additional context into resource costs of Transformer models, BERT-Large, a variant of the Bidirectional Encoder Representations from Transformers (BERT) from Devlin et al. (2018), has 330 million trainable parameters (Hui, 2020). As such, BERT-Large was trained on 16 TPUs for a total of 4 days, an amount of compute resources not easily available. These limitations are typically overcome by "fine-tuning" the model which involves taking a pre-trained Transformer model and fitting it to a smaller, edge case sample. This approach allows the model to perform well

with minimal end-resource costs because the majority of training was done beforehand. Unfortunately, this solution is not viable for this use case as available pre-trained models are trained for NLP tasks.

These base resource issues are exacerbated by the extreme maximum sequence length of the vectorized MalDroid samples at over 2.7 million. This is due to the nature of Transformer models as Devlin et al. (2018, as cited in Hui, 2020) explain:

> Longer sequences are disproportionately expensive because attention is quadratic to the sequence length. In other words, a batch of 64 sequences of length 512 is much more expensive than a batch of 256 sequences of length 128. The fully connected/convolutional cost is the same, but the attention cost is far greater for the 512-length sequences.

The original BERT model began training with a sequence length of 128 and finished with 512, significantly less than this study (Devlin et al., 2018, as cited in Hui, 2020). Even when limiting maximum sequence length (and as a result, sample size) and model hyperparameters to below that of BERT-Mini as described in Devlin et al. (2018), the model would throw an out-of-memory error when training.

Sample Cross-Referencing

Another issue that limited the scope of the research was the failure of sample cross-referencing. This process was meant first to match malware families in the AndMal dataset with those present in MITRE's ATT&CK and then AndMal samples with MalDroid samples, creating a dataset with matching behavioral analysis, malware family, and TTP chain for each MalDroid sample. Unfortunately, ATT&CK lacked appropriate representation of malware families in AndMal, therefore samples could not be matched with any ground-truth TTP chains

to train the TTP prediction segment of the model. This resulted in the removal of the use case from the final model, making the proposed transfer learning element unnecessary. The final product involved a BERT-like (Transformer encoder with dense output) architecture for malware classification only.

Time Constraints

Time was a large limiting factor in this study and led to a large number of compromises in the design and scope of the research. The tokenization of the behavioral analysis for each sample could be much more refined, allowing the model to draw better inferences from the vectorized data and save resources on vocabulary size and/or sequence length. Hyperparameter tuning had to be bypassed, which limited the model's chances at success and placed it at a relative disadvantage. However, this step is not entirely crucial as the ideal hyperparameters for BERT and other Transformer architectures are well documented. Performance profiling of the model using TensorBoard was also unable to be completed but would be recommended to maximise training efficiency. Finally, the sample size of the training dataset was limited to 5,139 out of the 9,167 samples under 100 MB able to be extracted from the MalDroid repository. This was due to the time required to vectorize each sample. A larger sample size would enable the model to more effectively generalize Android malware and its behavior.

Future Work

Future research should properly train the model on a scale that is not extremely detrimental to its performance. This would ideally be with a maximum sequence length of ~200,000 (thereby limiting the sample size to ~4,000), a batch size of 16, and BERT-Base hyperparameters as described in Devlin et al. (2018). It is estimated that training would require a virtual machine (VM) with approximately:

- 2 or more virtual CPUs
- 128 GB or more system RAM
- TPU or GPU accelerator with 128 GB memory or more (*Note:* Running on a TPU would require modification of existing code for the model and training loop)

Implementation of the proposed TTP prediction and transfer learning elements would be recommended as well. Researchers could achieve this by collecting their own ground-truth data from samples known to be represented in ATT&CK.

Additionally, further studies should address the above compromises made due to time constraints. This would include: 1) refining tokenization to be more statistically significant and optimized for a shorter sequence length, 2) properly tuning hyperparameters, 3) profiling and optimizing model performance with TensorBoard and mixed-precision, and 4) increasing the sample size to fully represent MalDroid.

Conclusions

Unfortunately, the proposed study was not able to be completed as planned due to time and resource constraints. Expected results accept the hypothesis that the proposed encoder-decoder model coupled with attention mechanisms and added context from MITRE's ATT&CK repository will enable the model to perform at or better than models from Mahdavifar et al. (2020) and Bibi et al. (2020) at threat classification and Kok et al. (2020) at detecting threat techniques according to metrics outlined in Tharwat (2017) on classification assessment.

Works Cited

Alzaylaee, M. K., Yerima, S. Y., & Sezer, S. (2020). DL-Droid: Deep learning based android malware detection using real devices. Computers & Security, 89, 101663.

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

Bibi, I., Akhunzada, A., Malik, J., Iqbal, J., Mussaddiq, A., & Kim, S. (2020). A Dynamic DL-Driven Architecture to Combat Sophisticated Android Malware. IEEE Access, 8, 129600-129612.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.

Classification on imbalanced data: TensorFlow Core. (2021, January 6). Retrieved January 16, 2021, from https://www.tensorflow.org/tutorials/structured_data/imbalanced_data

Clement, J. (2020, September 09). Global Android malware volume 2020. Retrieved December 8, 2020, from https://www.statista.com/statistics/680705/global-android-malware-volume/

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. Journal of the American society for information science, 41(6), 391-407.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929.

Gers, F. A., Schmidhuber, J., & Cummins, F. (1999). Learning to forget: Continual prediction with LSTM.

Kok, A., Mestric, I. I., Valiyev, G., & Street, M. (2020). Cyber Threat Prediction with Machine Learning. Information & Security, 47(2), 203-220.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. Neural computation, 1(4), 541-551.

Mahdavifar, S., Kadir, A. F. A., Fatemi, R., Alhadidi, D., & Ghorbani, A. A. (2020, August). Dynamic Android Malware Category Classification using Semi-Supervised Deep Learning. In 2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech) (pp. 515-522). IEEE.

Nayak, P. (2021, May 18). MUM: A new AI milestone for understanding information. Google. https://blog.google/products/search/introducing-mum/

Noor, U., Anwar, Z., Malik, A. W., Khan, S., & Saleem, S. (2019). A machine learning framework for investigating data breaches based on semantic analysis of adversary's attack patterns in threat intelligence repositories. Future Generation Computer Systems, 95, 467-487.

Positive Technologies. (2019, September 11). Vulnerabilities and threats in mobile applications, 2019. Retrieved December 8, 2020, from https://www.ptsecurity.com/ww-en/analytics/mobile-application-security-threats-and-vulnerabilities-2019/

Rahali, A., Lashkari, A. H., Kaur, G., Taheri, L., Gagnon, F., & Massicotte, F. (n.d.). DIDroid: Android Malware Classification and Characterization Using Deep Image Learning. In ICCNS 2020: 10th International Conference on Communication and Network Security: Tokyo, Japan, November 27-29, 2020. ACM.

Strom, B. (2020, June 24). ATT&CK 101. Retrieved December 09, 2020, from https://medium.com/mitre-attack/att-ck-101-17074d3bc62

Tam, K., Khan, S. J., Fattori, A., & Cavallaro, L. (2015, February). Copperdroid: Automatic reconstruction of android malware behaviors. In Ndss.

Tharwat, A. (2017). Classification assessment methods. Applied Computing and Informatics.

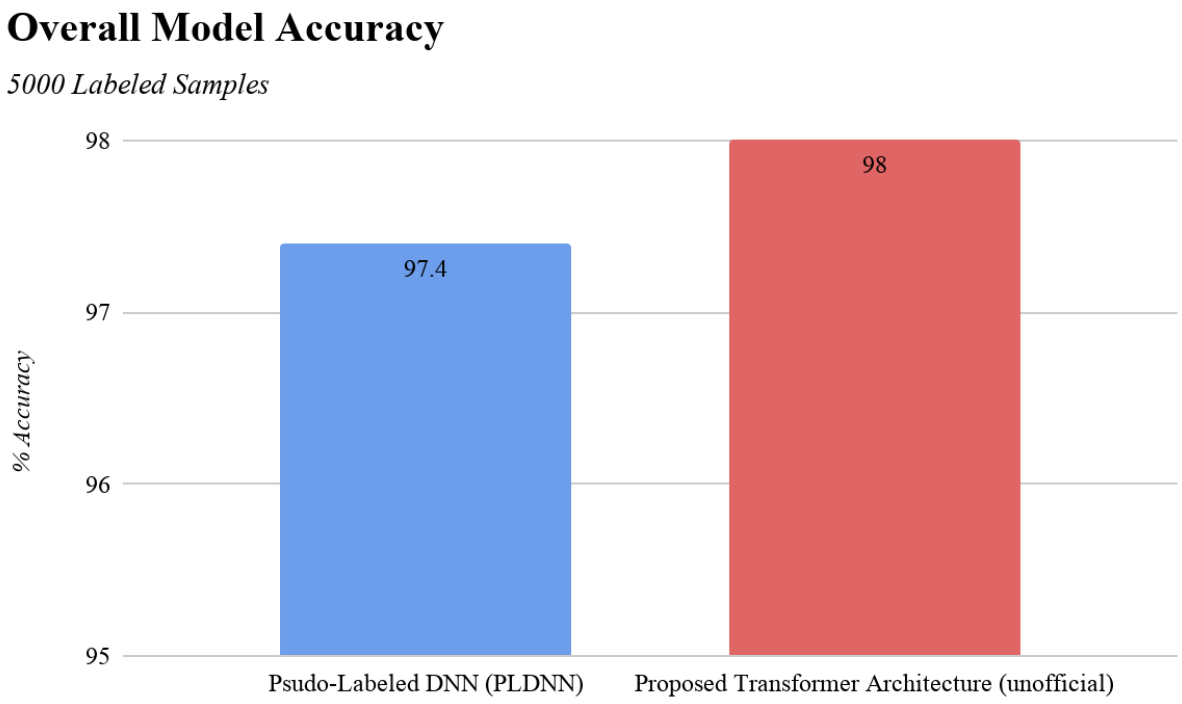
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30, 5998-6008.

Weng, L. (2018, June 24). Attention? Attention! Retrieved December 09, 2020, from https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html

Appendix

Figure 1

Overall Classification Accuracy of PLDNN and Proposed Transformer Architecture



Note: The data for PLDNN is from Mahdavifar et al. (2020).

Table 1

Per-Class Results of True/False Negative/Positive Rates for Malware Classification

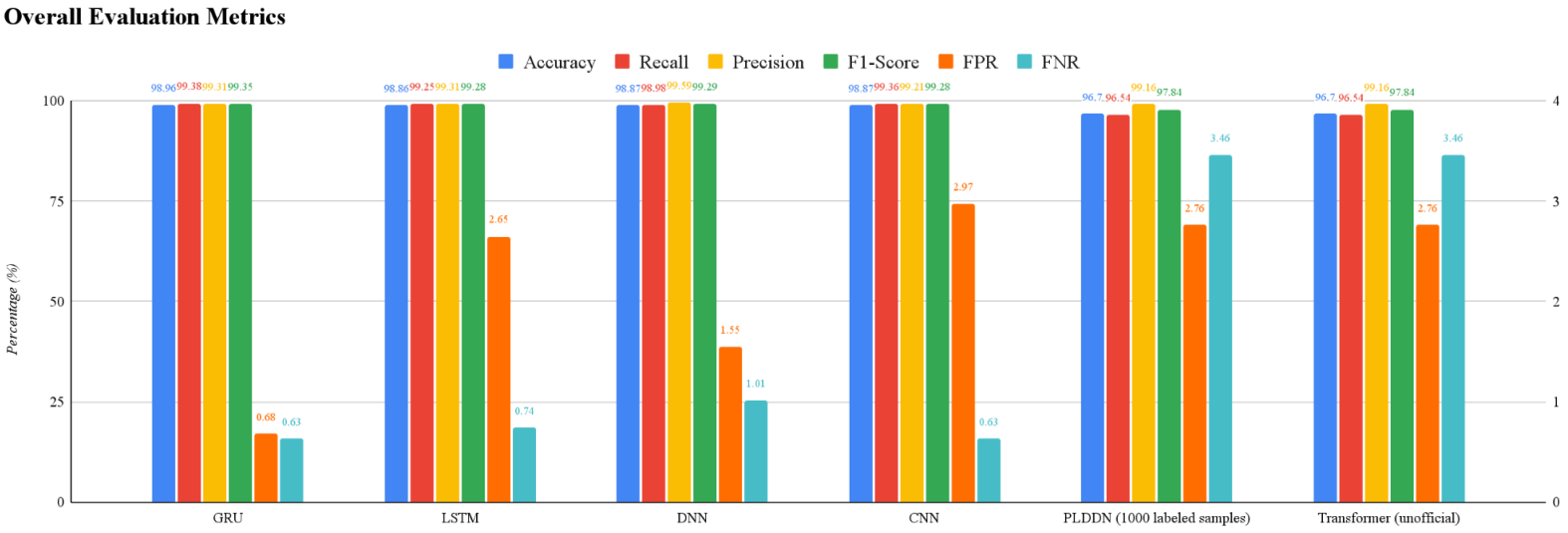
	Prediction Category					
Real Category		*Adware*	*Banking*	*SMS*	*Riskware*	*Benign*
	Adware	0.85	0.02	0.02	0.05	0.07
	Banking	0.0	0.96	0.03	0.01	0.0
	SMS	0.0	0.0	1.0	0.0	0.0
	Riskware	0.01	0.0	0.0	0.98	0.01
	Benign	0.01	0.0	0.0	0.01	0.98

	Prediction Category					
Real Category		*Adware*	*Banking*	*SMS*	*Riskware*	*Benign*
	Adware	TBD	TBD	TBD	TBD	TBD
	Banking	TBD	TBD	TBD	TBD	TBD
	SMS	TBD	TBD	TBD	TBD	TBD
	Riskware	TBD	TBD	TBD	TBD	TBD
	Benign	TBD	TBD	TBD	TBD	TBD

Note: Compares results from PLDNN (top) to the proposed Transformer Architecture (bottom). Specific points of interest are how often one class is mislabeled compared to others and which classes it is frequently mislabeled as. The data for PLDNN are from Mahdavifar et al. (2020).

Figure 2

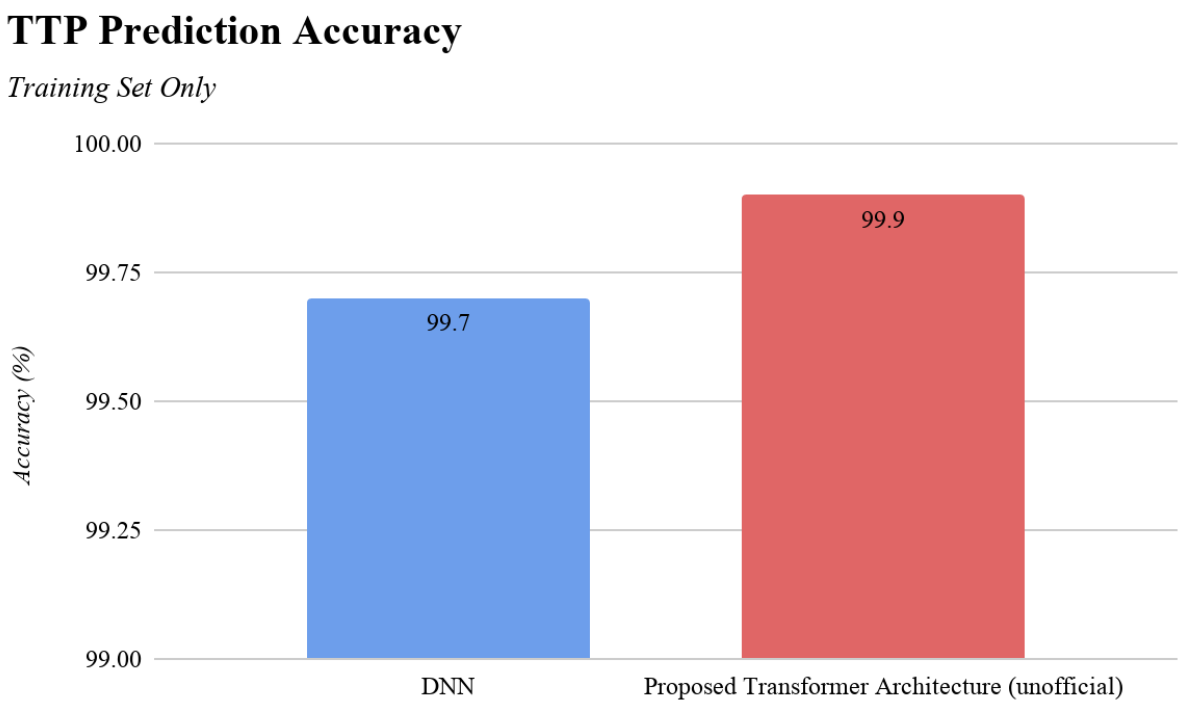
Comparison of Classification Evaluation Metrics for Multiple Model Architectures



Note: Accuracy, recall, precision, F1-score, false positive/negative rates metrics for each model architecture. To be compared against the proposed Transformer in malware classification. The data for GRU, LSTM, DNN, and CNN are from Bibi et al. (2020). The data for PLDNN are from Mahdavifar et al. (2020).

Figure 3

Overall Prediction Accuracy of TTPs



Note: Compares hypothetical results of the proposed transformer to a DNN on **training data only**. The data for DNN is from Kok et al. (2020).