# Assessing Modality Selection Heuristics to Improve Multimodal Deep Learning for Malware Detection

**Farzana Ahamed Bhuiyan, Katherine E. Brown,**
**Md Bulbul Sharif, Quentin D Johnson, Douglas A. Talbert**
{fbhuiyan42, kebrown42, msharif42, qdjohnson42 }@students.tntech.edu, {dtalbert}@tntech.edu
Department of Computer Science, Tennessee Technological University, Cookeville, TN 38501

## Abstract

With the growing use of Android devices, security threats are also increasing. While there are some existing malware detection methods, cybercriminals continue to develop ways to evade these security mechanisms. Thus, malware detection systems also need to evolve to meet this challenge. This work is a step towards achieving that goal. Malware detection methods need as much information as possible about the potential malware, and a multimodal approach can help in this regard by combining different aspects of an Android application. Multiple modalities can improve classification by providing complementary information, however, the use of all available modalities does not necessarily maximize algorithm performance. Thus, multimodal machine learning could benefit from a mechanism to guide the selection of modalities to include in a multimodal model. This work uses a malware detection problem to compare multiple heuristics for this selection process and the assumptions behind them. Our experiments show that selecting modalities with low predictive correlation works better than the other examined heuristics.

## Introduction

*Modality* refers to the way something occurs or is experienced, and when it involves various such modalities, a research problem is described as *multimodal* (Ramachandram and Taylor 2017). Observing multiple modalities together could enable better identification of the true intent and capabilities of an application and can provide a malware detector with capabilities beyond what is achievable by any single modality alone. For our purpose, we will let our model automatically learn patterns behind various malware apps. For these types of tasks, deep learning has proven to be useful (Liu et al. 2018).

Though the main goal is to enable all the modalities to interact and inform each other, the incorporation of all these modalities could result in a detrimental performance of the algorithm (Ramachandram and Taylor 2017). We might find a less complex model with fewer modalities that performs at least as well or better than the model with all the modalities. Here, the challenge lies in deciding which modalities to fuse. In this case, some techniques for modality selection may come in handy. To find the set of modalities that gives us the best performing models, we can exhaustively evaluate all the different combinations of modalities and compare them. If, however, we have many modalities, this type of exhaustive approach will be inefficient and, perhaps, infeasible. In such a case, some kind of heuristic to guide an approximate search for modality selection could be beneficial. We have come up with three different heuristic approaches for modality selection and compare them to find which among them most effectively guides us to the best set of modalities that, when fused, leads to best the performance.

The specific goal of this work is to assess several heuristic approaches for modality selection, with Android malware detection providing the experimental domain. We have collected our dataset following a similar fashion as in (Kim et al. 2018). The following are the main contributions of this paper - 1) We collected data from various modalities for Android devices which allowed us to develop a multimodal malware detection approach, which was not widely explored by previous works. 2) To our knowledge, none of the prior works did any type of modality selection for malware analysis, which is explored in this paper.

## Related Work

There has been some work on malware classification using deep neural networks (Linda, Vollmer, and Manic 2009; Mohammadpour et al. 2018). However, most of the existing approaches are unimodal, focusing on feature selection (Javaid et al. 2016; Salama et al. 2011), not modality selection, whereas we are more interested in a multimodal approach. The most related work to us is (Kim et al. 2018). They have developed a multimodal deep learning method for malware classification. In their work, they used all the available modalities but they have not done any rigorous experiment to show why they chose all the modalities. None of the prior work tried to do any type of modality selection, whereas we have proposed three heuristic methods for modality selection to reduce our model costs while improving performance.

Table 1: Configuration of our Multimodal Neural Network with just two modalities

| Layers | Input Shape | Output Shape | Number of Units | Activation Function |
|---|---|---|---|---|
| Input | (None, 39512) | (None, 39512) | 39512 | ReLU |
| Hidden | (None, 39512) | (None, 2000) | 2000 | ReLU |
| Hidden | (None, 2000) | (None, 1000) | 1000 | ReLU |
| Merge | [(None, 1000),(None, 1000)] | (None, 2000) | 2000 | ReLU |
| Hidden | (None, 2000) | (None, 100) | 100 | ReLU |
| Hidden | (None, 100) | (None, 10) | 10 | ReLU |
| Output | (None, 10) | (None, 1) | 1 | Sigmoid |

## Data Preparation

The use of multimodal data is an active area of research. However, to our knowledge, there are still not enough multimodal datasets for malware detection available (Kim et al. 2018). We believe a much better malware classifier can be built using a good multimodal dataset. Therefore, we built a labeled multimodal dataset for malware analysis for Android devices.

We have collected Android APK files for both benign and malware samples. We then extracted the raw data to make the Android APK files interpretable. We first unzipped the APK files and extracted the manifest, dex, and the shared library files. Then these files were disassembled. APKTool (Ryszard Wiśniewski 2019) was used for disassembling the manifest and dex files, respectively and the shared library files were disassembled using Ghidra (Agency 2019). After disassembling these files, we obtained the five different modalities: (1) the *Manifest modality* (items from the *AndroidManifest.xml* file such as permissions, configurations, components, and environment features); (2-4) the *String modality*, the *Method Opcode modality*, and the *Method API modality* (all obtained from the dex files); and (5) the *Shared Library Function Opcode modality* (obtained from the shared library files). The details of each of these modalities can be found in (Kim et al. 2018).

After collecting the multimodal data, these datasets were used to generate feature vectors corresponding to similarities and existences of features in the multimodal dataset described above to help distinguish between benign and malicious applications. Since we have used neural networks, we converted all the modalities to fixed-sized vectors so that they can be fed as input to our neural network. To generate these vectors we followed the algorithms from prior work (Kim et al. 2018). For our experiment, we collected 2063 Android APK (Android Package Kit) samples. Among these 1257 are malware samples and 806 are benign samples. We have used 1382 samples as our training dataset and 681 samples (33% of all the data) as our test data. From the training dataset, 1243 samples are used for training and 139 samples (10% of the training data) are used as our validation dataset.

## Experimental Setup

Ideally, we want to fuse the optimal set of modalities to maximize performance measure. Instead of exhaustively trying all subset of modalities, we have experimented with various greedy methods that used forward step-wise selection. We have used three different heuristics approaches for selecting the modalities at each step - the *maxDifference* heuristic, the *maxSimilarity* heuristic, and the *maxAccuracy* heuristic.

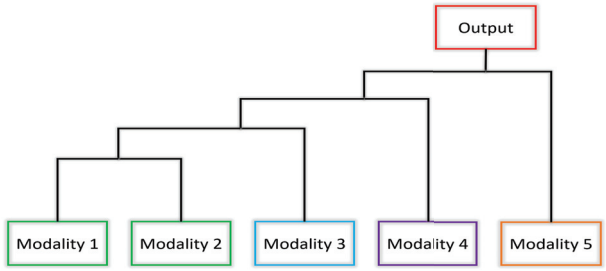


Figure 1: Greedy Forward Step-wise Selection

For each of these approaches, at the first stage, we have designed separate unimodal models for each of the modalities. For the five modalities, we used five different unimodal models. The architecture of each of these models is identical (except for the size of the input layer). Every model gives us a prediction result. Finally, we got five different classification results from the five models. We then took the model with the highest accuracy as our initial model. After selecting the most accurate unimodal model, new multimodal models are designed including the best model and then additional models, added one at a time.

In our first approach, the maxDifference heuristic approach, we selected the additional model as the model with the highest number of disagreements with the initial model. Note that, though this second model has the highest number of disagreements, it may not be the model with the lowest accuracy. Then we merged these two models to get a multimodal model with just two modalities. All the detailed parameter settings of each of the layers of our multimodal model are summarized in Table 1. Similarly, we then continued to add modalities one by one with the already build multimodal model. We continued until we merged all the modalities. In the end, we have a multimodal model with all five modalities that gives us the final classification result (Fig: 1).

The maxSimilarity heuristic approach is essentially the opposite of the first one. Instead of selecting the model with the highest disagreement, we selected the one with the highest agreement at each step. So, after selecting the most accurate unimodal model, another unimodal model is selected

Table 2: Performance Measure for Forward Selection using maxDifference Heuristic

| Modality | Accuracy | Precision | Recall | F-score |
|---|---|---|---|---|
| String + Function opcode | 0.973 | 0.98 | 0.97 | 0.97 |
| String + Function opcode + Manifest | 0.982 | 0.98 | 0.98 | 0.98 |
| String + Function opcode + Manifest + Method opcode | 0.980 | 0.97 | 0.97 | 0.97 |
| String + Function opcode + Manifest + Method opcode + Method API | 0.959 | 0.96 | 0.96 | 0.96 |

Table 3: Performance Measure for Forward Selection using maxSimilar Heuristic

| Modality | Accuracy | Precision | Recall | F-score |
|---|---|---|---|---|
| String + Method API | 0.972 | 0.97 | 0.97 | 0.97 |
| String + Method API + Function opcode | 0.957 | 0.96 | 0.96 | 0.96 |
| String + Method API + Function opcode + Method opcode | 0.847 | 0.75 | 0.81 | 0.77 |
| String + Method API + Function opcode + Method opcode + Manifest | 0.965 | 0.97 | 0.96 | 0.96 |

Table 4: Performance Measure for Forward Selection using maxAccurate Heuristic

| Modality | Accuracy | Precision | Recall | F-score |
|---|---|---|---|---|
| String + Manifest | 0.980 | 0.98 | 0.98 | 0.98 |
| String + Method opcode | 0.967 | 0.97 | 0.96 | 0.97 |
| String + Method API | 0.975 | 0.98 | 0.97 | 0.97 |
| String + Function opcode | 0.965 | 0.97 | 0.96 | 0.97 |
| String + Manifest + Method opcode | 0.964 | 0.97 | 0.96 | 0.97 |
| String + Manifest + Method API | 0.976 | 0.97 | 0.97 | 0.97 |
| String + Manifest + Function opcode | 0.971 | 0.97 | 0.97 | 0.97 |
| String + Manifest + Method API + Method opcode | 0.974 | 0.98 | 0.97 | 0.97 |
| String + Manifest + Method API + Function opcode | 0.978 | 0.98 | 0.96 | 0.97 |
| String + Manifest + Method API + Function opcode + Method opcode | 0.837 | 0.96 | 0.96 | 0.96 |

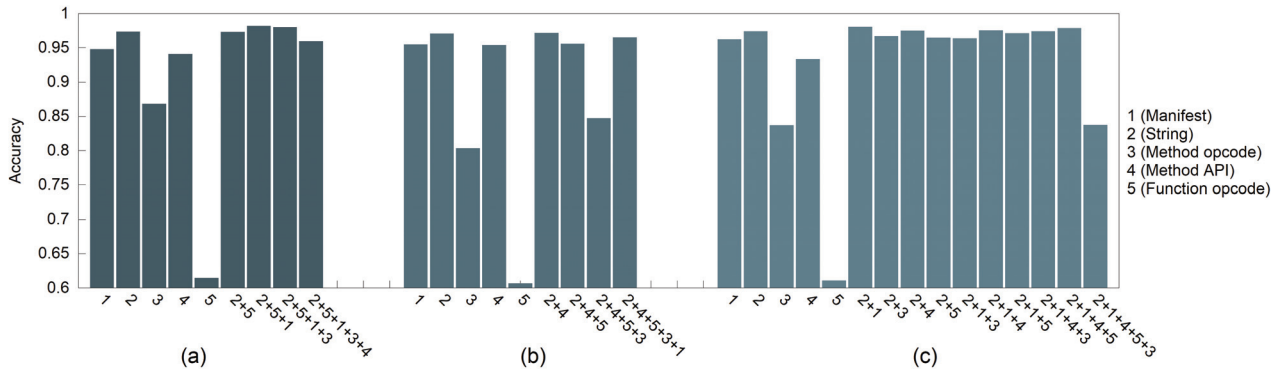


Figure 2: Accuracy comparison of multimodal neural networks using different combinations of modalities using three different heuristics - (a) maxDifferernt Heuristic, (b) maxSimilar Heuristic and (c) maxAccurate Heuristic

that agrees the most with the classification result of the first model. We continue selecting unimodal models in this way until we have a multimodal model with all the modalities.

For our maxAccuracy heuristic approach, the additional model, which together with the best model improves the performance the most (classification accuracy), is selected. To find out the additional model, we fused all the other modalities with the initial one, one by one, and selected the multimodal model with the highest accuracy. Thus we continued adding modalities until all modalities have been fused.

The ultimate goal of these procedures is a multimodal model that represents the best combination of modalities to accurately classify malware samples. The parameter settings of each of the layers of our multimodal model are similar to the one described in Table 1. However, it should be emphasized that the combination found may not be the absolute best one since the first selected model, even though it is the best from a single unimodal model point of view, may not be the ideal one when multiple models are combined in a more sophisticated way. This implies that, potentially, other

combinations of models could be as good as or even better than the selected one. Since our goal is to not perform an exhaustive search, this is an unavoidable reality.

If we have many more modalities, we could just try adding modalities one by one until we can't find any better performing model. If a better performing multimodal model can be obtained, the same process is repeated once again adding an additional modalities, one at a time, until all remaining modes have been used. The process thus can be iteratively repeated until no better model can be obtained.

Since we have many features for each modality, we used GPUs to accelerate our algorithms. We used a high performance computing cluster where each node contains two Intel Xeon E5-2680v4 CPUs, 128 GB RAM and an Nvidia Tesla K40, 12 GB graphics memory. We have run each of our models three times and took the average results.

## Results

To get a detailed idea of the performance measures of each our model, we have reported the accuracy, precision, recall, and F-score values. From the accuracy graph in Fig-2, we see that the accuracy does not always increase with an increased number of modalities. Using the maxDifference heuristic we get the highest accuracy when we used only three modalities - String, Manifest and Function Opcodes (Table 2), using the maxSimilar heuristic we get the highest accuracy when we used only two modalities - String and Method APIs (Table 3), and finally for the maxAccuracy heuristic we get the highest accuracy when we used four modalities - String, Manifest, Method APIs and method opcodes (Table 4). By comparing all three heuristics, we see the forward selection search using the maxDifference heuristic worked the best for this task using the given data. Using this heuristic we can select the best set of modalities to maximize the performance measures. We also got the highest precision, recall, and F-score for this case. Adding the other modalities does not give us any additional information to improve performance. We conclude that selecting modalities with low predictive correlation works better than the other examined heuristics. It is worth noting, however, that maxAccuracy achieves almost as high an accuracy with fewer modalities. If model simplicity matters, then this one might be better. Addition experiments on more datasets are needed to better address this issue.

Our results suggest that we do not need to combine highly accurate unimodal models, but rather we need models that make different kinds of errors. The idea is similar to ensemble methods. High accuracy can be accomplished if different models misclassify different training examples, even if the unimodal classifier accuracy is low. Using this heuristic, we reduced our modalities from five to three. This work becomes even more important when the number of modalities is larger. We need not use every modality at our disposal for creating the best model. We can assist our algorithm by feeding in only those modalities that are most important. Thus we can reduce the training time and the evaluation time as well as the complexity of the model. In fields such as security, even the smallest amount of improvement of machine learning algorithms can be something truly valuable.

## Conclusion

Performing multimodal analysis helps identify the true intent and capabilities of advanced malware and can provide a more accurate technical indicator which may not be achievable by any single modalities alone. However, the inclusion of every modality blindly can have a negative effect. We proposed a heuristic method to select the most informative modalities. We got an improved overall performance and have a simpler model with fewer connections than just using every modality together. This method is designed to improve the stability and accuracy of our malware detection algorithms while reducing the overall cost.

## References


Agency, N. S. 2019. Ghidra - software reverse engineering framework.

Javaid, A.; Niyaz, Q.; Sun, W.; and Alam, M. 2016. A deep learning approach for network intrusion detection system. In *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*, 21–26. ICST (Institute for Computer Sciences, Social-Informatics and ....

Kim, T.; Kang, B.; Rho, M.; Sezer, S.; and Im, E. G. 2018. A multimodal deep learning method for android malware detection using various features. *IEEE Transactions on Information Forensics and Security* 14(3):773–788.

Linda, O.; Vollmer, T.; and Manic, M. 2009. Neural network based intrusion detection system for critical infrastructures. In *2009 international joint conference on neural networks*, 1827–1834. IEEE.

Liu, K.; Li, Y.; Xu, N.; and Natarajan, P. 2018. Learn to combine modalities in multimodal deep learning. *arXiv preprint arXiv:1805.11730*.

Mohammadpour, L.; Ling, T. C.; Liew, C. S.; and Chong, C. Y. 2018. A convolutional neural network for network intrusion detection system. *Proceedings of the Asia-Pacific Advanced Network* 46:50–55.

Ramachandram, D., and Taylor, G. W. 2017. Deep multimodal learning: A survey on recent advances and trends. *IEEE Signal Processing Magazine* 34(6):96–108.

Ryszard Wiśniewski, C. T. 2019. Apktool, a tool for reverse engineering android apk files.

Salama, M. A.; Eid, H. F.; Ramadan, R. A.; Darwish, A.; and Hassanien, A. E. 2011. Hybrid intelligent intrusion detection scheme. In *Soft computing in industrial applications*. Springer. 293–303.