Performance Report of DDAN, Dual-DDAN, LSTM and Random Forest

After finishing the Dual-DDAN method last sprint, we conducted an experiment to compare the performance of DDAN, Dual-DDAN, LSTM and Random Forest. The datasets we used were: FFmpeg, LibPNG and LibTIFF. We will introduce the performance of each method from different aspects: performance metrics, execution time, implementation difficulty and resource occupation rate.

In order to emphasize the comparability, we used the same datasets for each method. According to the paper description, all the methods were used their introduced data pre-processing procedure. We operated two different dataset combinations: 1. FFmpeg & LibPNG. 2. FFmpeg & LibTIFF.

We used five performances metrics: FNR, FPR, Recall, Precision and F1-score.

FNR (False Negative Rate): It represents that in the positive class, how many samples are predicted to be the negative class. In the case here, it means that the rate of non-vulnerable function has been identified as vulnerable.

FPR (False Positive Rate): It represents that in the negative class, how many samples are predicted to be a positive class. In the case here, it means that the rate of the Vulnerable function has been identified as non-vulnerable. In another word, it shows the capability of the model to correctly find out the vulnerable functions.

Precision: Model precision score represents the model's ability to correctly predict the positives out of all the positive predictions is made.

Recall: Model recall score represents the model's ability to correctly predict the positives out of actual positives. The higher the recall score, the better the machine learning model is at identifying both positive and negative examples.

F1-Score: Model F1 score represents the model score as a function of precision and recall score. F1-score is a machine learning model performance metric that gives equal weight to both the Precision and Recall for measuring its performance in terms of accuracy. It's often used as a single value that provides high-level information about the model's output quality.

In summary, The ideal model should have FNR and FPR close to 0 , Recall, Precision and F1-score should be close to 1.Here are our results：

Dataset	Methods	FNR	FPR	Recall	Precision	F1-Score	Time(Mins)
peg->png	LSTM	0.01	0.97	0.99	0.97	0.98	115.60
	RF	0.00	1.00	1.00	0.93	0.96	9.66
	DDAN	0.00	0.04	1.00	0.67	0.80	2.23
	Dual-DDAN	0.13	0.00	0.88	1.00	0.93	5.45
peg->tiff	LSTM	0.03	0.81	0.97	0.96	0.96	248.56
	RF	0.00	0.98	1.00	0.89	0.94	8.23
	DDAN	0.45	0.08	0.55	0.46	0.50	2.53
	Dual-DDAN	0.23	0.09	0.76	0.52	0.62	5.99

(The best performances are highlighted)

From the performance indicators, we found that:

All the four methods we replicated are having abilities to correctly predict the Non-vulnerable function due to Low FNR and High Recall scores. But we also discovered that LSTM and Random Forest have significate high FPR scores which means those two methods do not have the capability to find any vulnerable functions. LSTM and Random Forest have great FNR, Recall, Precision and F1-Score but it is because the dataset contains mostly non-vulnerable functions and the methods can also predict the non-vulnerable function. Therefore, those performance indicators are high. However, our project is aimed to discover the vulnerable functions by using those methods so LSTM and Random Forest methods we will not recommend for software vulnerability detection. As for DDAN and Dual-DDAN methods, both have Low FPR scores which means the models can successfully predict the vulnerabilities. For the other performance metric value, Dual-DDAN shows a better performance than DDAN. The disadvantage of the DDAN & Dual-DDAN is that the FNR score is high, which means the models will recognize some non-vuln as vulnerable. This will influence the further discovery of the vulnerable functions.

As for execution time, DDAN and Dual-DDAN we record the time to train one model for one combination of the parameters instead of all available combinations. The result shows that DDAN is the most time-efficient, then the Dual-DDAN, the next is RF and LSTM is the least time-efficient.

As for implementation difficulty. LSTM and Random Forest methods have complete datasets, executable codes and included runnable data pre-processing functions. So implement LSTM and Random Forest methods are easy and friendly to new users. DDAN and Dual-DDAN have complete datasets and executable codes as well. But the data pre-processing methods and functions are not included. The installation of tools for data pre-processing is very hard and costs time. Importantly, the author didn't mention the progress for data transforms and numberize the vectors. Therefore, for new users, DDAN and Dual-DDAN are hard to implement.

As for resource occupation rate, LSTM and RF will consume full CPU resources. So during the training, the other computer process will on hold or slow response. However, DDAN and Dual-DDAN methods are optimized for CPU usage, those methods will be utilized approx. 70% of the CPU resources. In terms of usability concerns, DDAN and Dual-DDAN are better than LSTM and RF.