Android Malware Categorization Using a Semi-Supervised Deep Learning Architecture Based on Ladder Networks

Samaneh Mahdavifar, Mohammad Rasool Fatemi, Dima Alhadidi, Ali A. Ghorbani

Canadian Institute for Cybersecurity (CIC), University of New Brunswick (UNB)

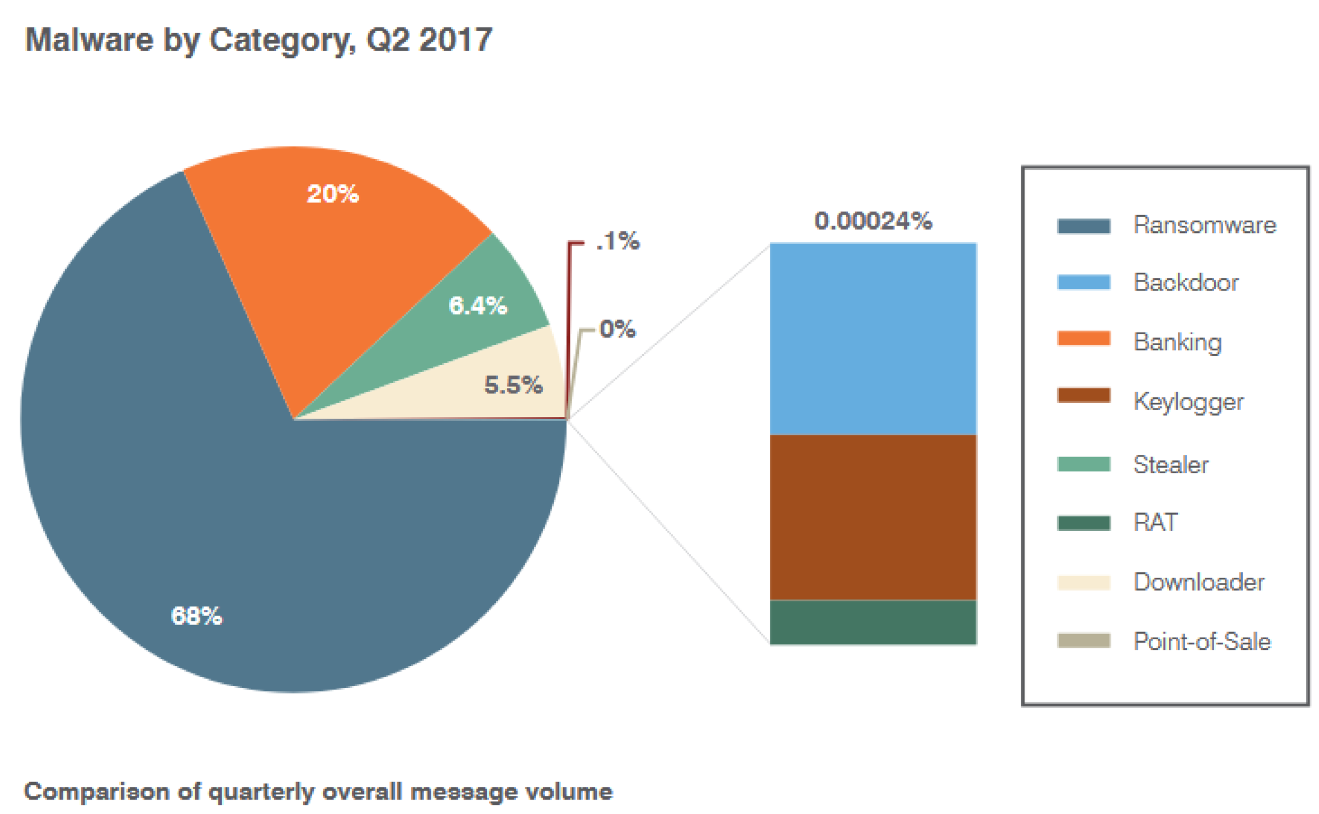
Abstract

Owning the largest share of Smartphone platform tendency, Android OS has changed to an undeniable target for many malware authors. According to real world scenario, most of Android malware applications don’t benefit from having labels and applying supervised learning wouldn’t be the right solution to pick. That’s because of the major drawback of supervised learning, i. e., requiring lots of labeled data that is so expensive to collect. Addressing this issue, in this work, we employ a semi-supervised deep learning architecture, called, ladder networks that minimizes the sum of supervised and unsupervised cost functions by backpropagation at the same time. We craft two different types of feature vectors, namely, API Call frequency and API Call sequences to feed into ladder networks. The experimental results show a reasonable gap of 7-8% in total accuracy for 100 used labels and all used labels. At last, we fine-tune the learning parameters of ladder networks to improve its generalization ability and present corresponding graphs and diagrams

Previous Research

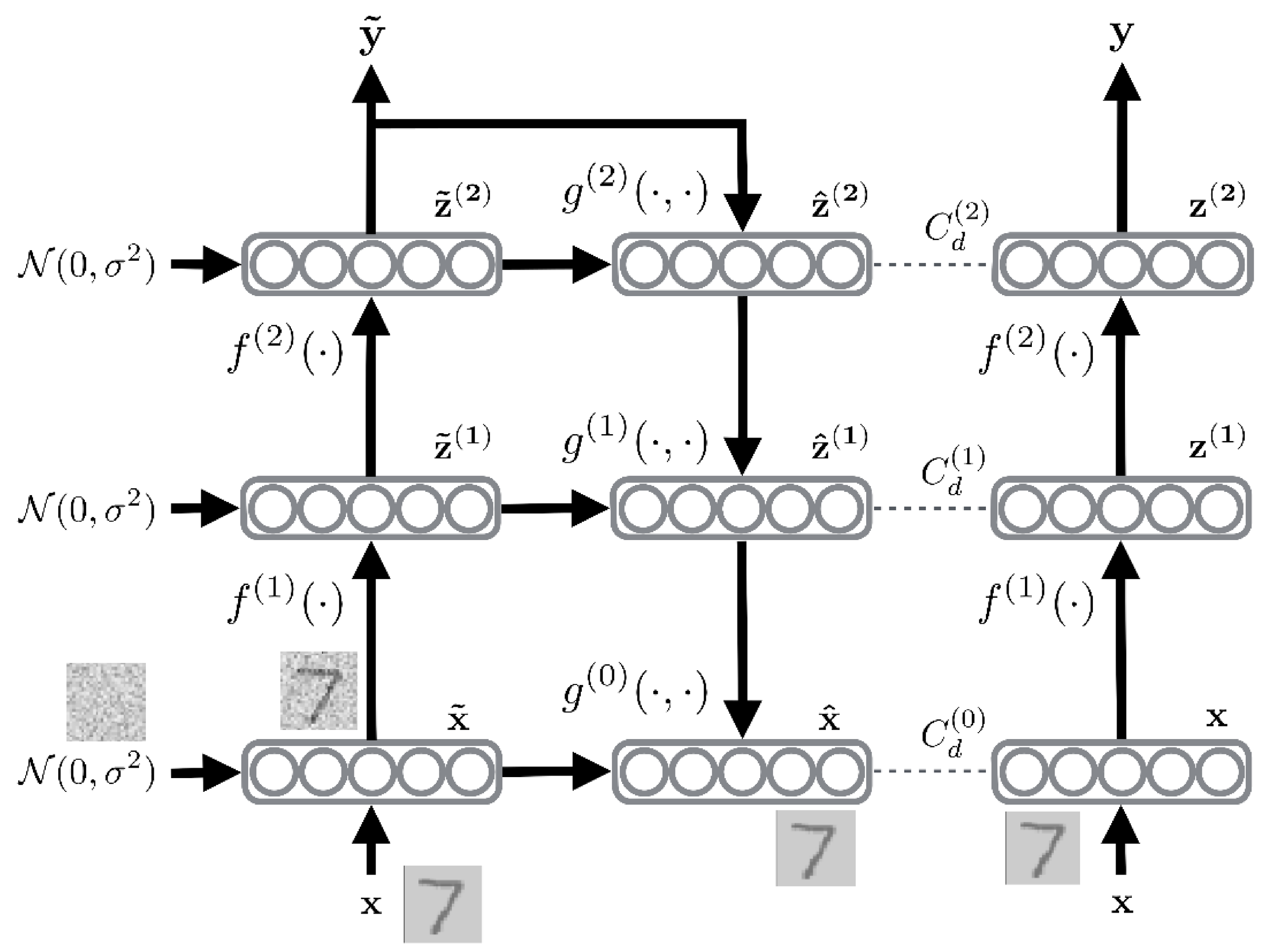
Year	Authors	Publication Title	Focus Area	Analysis Technique
2013	Aafer et. Al.	Droidapiminer: Mining api-level features for robust malware detection in android [SecureComm]	Android Malware Detection	Lightweight approach based on semantic information inside bytecode of the applications such as, invoked critical API calls, package level information as well as some important parameters
2014	Zhang et. al.	Semantics-aware android malware classification using weighted contextual api dependency graphs [ACM SIGSAC]	Android Malware Detection	Semantic-based approach that classifies Android malware via a weighted contextual API dependency graph
2017	Mariconti et. al.	Mamadroid: Detecting android malware by building markov chains of behavioral models [NDSS]	Android Malware Detection	Static Android malware detection system based on the sequences of abstracted API Calls which are used to build a Markov chain
2016	Hou et. al.	Deep4maldroid: A deep learning framework for android malware detection based on linux kernel system call graphs [WIW]	Android Malware Detection using Deep Learning	Dynamic Android malware detection based on a weighted graph of Linux kernel system calls using Stacked Auto-Encoders

Android Malware Categories



Ladder Network

- Auto-encoder with lateral shortcut connections from the encoder to the decoder at each level of the hierarchy
- Allows the higher levels of the network to discard details and focus on representing more abstract invariant features



Experiments


```
{
  "0": {
    "inputs": [],
    "package": "com.android.music",
    "outputs": {
      "ReturnType": "java.lang.Boolean",
      "ReturnValue": "true"
    },
    "class": "java.io.File",
    "timestamp": 1477336771863,
    "method": "exists"
  },
  "1": {
    "inputs": [],
    "package": "com.android.music",
    "outputs": {
      "ReturnType": "java.lang.Boolean"
      "ReturnValue": "true"
    },
    "class": "java.io.File",
    "timestamp": 1477336771863,
    "method": "exists"
  },
  "2": {
    "inputs": [],
    "package": "com.android.music",
    "outputs": {
      "ReturnType": "java.lang.Boolean"
      "ReturnValue": "true"
    },
    "class": "java.io.File",
    "timestamp": 1477336831841,
    "method": "exists"
  },
```


TABLE I
PARAMETER FINE-TUNING

Parameter	Values
layers	[4761, 1000, 500, 250, 250, 250, 5]
learning rate	0.02
Decay-after	5
Batch-size	20
Epoch	20

Algorithm 1 Calculation of the output $\mathbf{y}$ and cost function C of the Ladder network

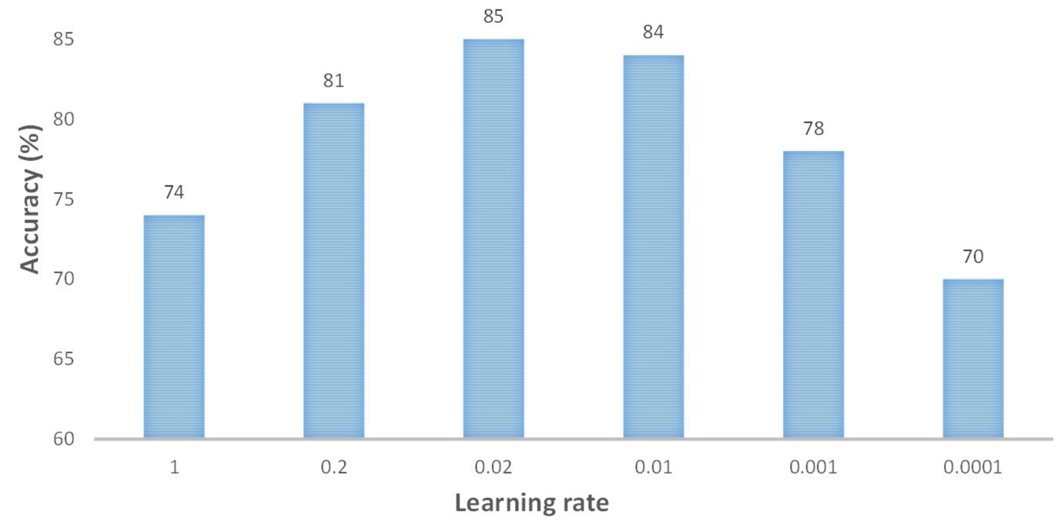
Require: $\mathbf{x}(n)$
Corrupted encoder and classifier
$\tilde{\mathbf{h}}^{(0)} \leftarrow \tilde{\mathbf{z}}^{(0)} \leftarrow \mathbf{x}(n) + \texttt{noise}$
for l = 1 **to** L **do**
$\tilde{\mathbf{z}}^{(l)} \leftarrow \texttt{batchnorm}(\mathbf{W}^{(l)}\tilde{\mathbf{h}}^{(l-1)}) + \texttt{noise}$
$\tilde{\mathbf{h}}^{(l)} \leftarrow \texttt{activation}(\boldsymbol{\gamma}^{(l)} \odot (\tilde{\mathbf{z}}^{(l)} + \boldsymbol{\beta}^{(l)}))$
end for
$P(\tilde{\mathbf{y}} \mid \mathbf{x}) \leftarrow \tilde{\mathbf{h}}^{(L)}$
Clean encoder (for denoising targets)
$\mathbf{h}^{(0)} \leftarrow \mathbf{z}^{(0)} \leftarrow \mathbf{x}(n)$
for l = 1 **to** L **do**
$\mathbf{z}^{(l)}_{\text{pre}} \leftarrow \mathbf{W}^{(l)}\mathbf{h}^{(l-1)}$
$\boldsymbol{\mu}^{(l)} \leftarrow \texttt{batchmean}(\mathbf{z}^{(l)}_{\text{pre}})$
$\boldsymbol{\sigma}^{(l)} \leftarrow \texttt{batchstd}(\mathbf{z}^{(l)}_{\text{pre}})$
$\mathbf{z}^{(l)} \leftarrow \texttt{batchnorm}(\mathbf{z}^{(l)}_{\text{pre}})$
$\mathbf{h}^{(l)} \leftarrow \texttt{activation}(\boldsymbol{\gamma}^{(l)} \odot (\mathbf{z}^{(l)} + \boldsymbol{\beta}^{(l)}))$
end for
# Final classification:
$P(\mathbf{y} \mid \mathbf{x}) \leftarrow \mathbf{h}^{(L)}$
# Decoder and denoising
for l = L **to** 0 **do**
if l = L **then**
$\mathbf{u}^{(L)} \leftarrow \texttt{batchnorm}(\tilde{\mathbf{h}}^{(L)})$
else
$\mathbf{u}^{(l)} \leftarrow \texttt{batchnorm}(\mathbf{V}^{(l+1)}\hat{\mathbf{z}}^{(l+1)})$
end if
$\forall i : \hat{z}^{(l)}_i \leftarrow g(\tilde{z}^{(l)}_i, u^{(l)}_i)$ # Eq. (2)
$\forall i : \hat{z}^{(l)}_{i,\text{BN}} \leftarrow \frac{\hat{z}^{(l)}_i - \mu^{(l)}_i}{\sigma^{(l)}_i}$
end for
# Cost function C for training:
$C \leftarrow 0$
if $t(n)$ **then**
$C \leftarrow -\log P(\tilde{\mathbf{y}} = t(n) \mid \mathbf{x}(n))$
end if
$C \leftarrow C + \sum_{l=0}^{L} \lambda_l \left\| \mathbf{z}^{(l)} - \hat{\mathbf{z}}^{(l)}_{\text{BN}} \right\|^2$ # Eq. (3)

Dataset and Results

Type	Number
Banking	250
Ransomware	250
Adware	250
Botnet	250
Benign	250

TABLE II
ACCURACY BASED ON NUMBER OF USED LABELS

Accuracy % with # of used labels	100	All
API Call Frequency	59%	66%
API Call Sequence	77%	85%



Future Work

Practical

- We have to add more data to our dataset since doing so will improve the results and considering that we can get our hands on more data, this is definitely going to make a noticeable improvement in accuracy
- In the near future, there is going to be more works on semi-supervised android malware detection and we need to compare our method to the other algorithms
- There are more datasets that we can trust and as a result, perform our model on them. This makes our method more comparable to others and also may open some new aspects to the problem.

Theorical

- The analysis of feature is rather incomplete, as we can extract more and better features from API call sequences and frequencies. Much remains to be done in this regard