Supplementary Material: Learning a Neural-network-based Representation for Open Set Recognition

Mehadi Hassen
School of Computing
Florida Institute of Technology
Melbourne, FL 32901
mhassen2005@my.fit.edu

Philip K. Chan
School of Computing
Florida Institute of Technology
Melbourne, FL 32901
pkc@cs.fit.edu

February 24, 2020

A Evaluation Network Architectures

We evaluate 5 networks: ii, ce, ii+ce, Openmax and G-Openmax. The first four networks and the final classifier of G-Openmax have the same architecture up to the fully connected z-layer. In case of the MNIST dataset, the input images are of size (28,28) and are padded to get an input layer size (32,32) with 1 channel. Following the input, layer are 2 non-linear convolutional layers with 32 and 64 units (filters) which have a kernel size of (4,4) with a (1,1) strides and SAME padding. The network also has max polling layers with a kernel size of (3,3), strides of (2,2), and SAME padding after each convolutional layer. Two fully connected non-linear layers with 256 and 128 units follow the second max pooling layer. Then a linear z-layer with a dimension of 6 follows the fully connected layers. In the case of ii+ce and ce networks, the output of the z-layer is fed to an additional linear layer of dimension 6 which is then given to a softmax function. We use Relu activation function for all the non-linear layers. Batch normalization is used throughout all the layers. We also use Dropout with keep probability of 0.2 for the fully connected layers. Adam optimizer with a learning rate of 0.001, beta1 of 0.5, and beta2 of 0.999 is used to train our networks for 5000 iterations. In case of the Openmax network, the output of the z-layer is directly fed to a softmax layer. Similar to the Openmax paper we use a distance that is a weighted combination of normalized Euclidean and cosine distances. For the ce, ii, and ii+ce we use contamination ratio of 0.01 for the threshold selection.

The open set experiments for MS Challenge dataset also used similar architectures as the four networks used for MNIST dataset with the following differences. The input layer size MS Challenge dataset is (67,67) with 1 channel after padding the original input of (63,63). Instead of the two fully connected non-linear layers, we use one fully connected layer with 256 units. We use dropout in the fully connected layer with keep probability of 0.9. Finally, the network was trained using Adam optimizer with 0.001 learning rate, 0.9 beta1, and 0.999 beta2.

We do not use a convolutional network for the Android dataset open set experiments. We use a network with one fully connected layer of 64 units. This is followed by a z-layer with a dimension of 6. For ii+ce and ce networks we further add a linear layer with a dimension of 6 and the output of this layer is fed to a softmax layer. In case of Openmax, the output of the z-layer is directly fed to the softmax layer. For Openmax we use a distance that is a weighted combination of normalized Euclidean and cosine distances. We use Relu activation function for all the nonlinear layers. We used batch normalization for all layers. We also used Dropout with keep probability of 0.9 for the fully connected layers. We used Adam optimizer with a learning rate of 0.1 and first momentum of 0.9 to train our networks for 10000 iterations. For the ce, ii, and ii+ce we use contamination ratio of 0.01 for the threshold selection.

The closed set experiments use the same set up as the open set experiments with the only difference coming from the dimension of the z-layer. For the MNIST dataset, we used z dimension of 10. For the MS and Android datasets, we use z dimension of 9.

B Closed Set Classification

In this section, we would like to show that on a closed dataset, a network trained using ii-loss performs comparably to the same network trained using cross entropy loss. For closed set classification, all the classes in the dataset are used for both training and test. For MS and Android datasets, we randomly divide the

Copyright © 2020 by SIAM
Unauthorized reproduction of this article is prohibited

datasets into training, validation, and test and report the results on the test set. The MNIST dataset is already divided into training, validation, and test.

On closed MNIST dataset, a network trained with cross-entropy achieved a 10-run average classification accuracy of 99.42%. The same network trained using ii-loss achieved an average accuracy of 99.31%. The network trained only on cross-entropy gives better performance than the network trained on ii-loss. The results from a network trained both ii-loss cross entropy loss to achieve an average classification accuracy of 99.40%. This result makes it comparable to the performance of the same network trained using cross-entropy only (with a p-value of 0.22). We acknowledge that both results are not state-of-art as we are using simple network architectures. The primary goal of these experiments is to show that the ii-loss trained network can give comparable results to a cross entropy trained network. On the Android dataset, the network trained on a cross entropy gets an average classification accuracy of 93.10% while ii-loss records 92.68%, but the difference is not significant (with a p-value at 0.43).

C Discussion

We mentioned earlier that we used function call graph (FCG) feature for the malware dataset. We also mentioned that in case of the MS Challenge dataset we reformatted the FCG features proposed in [1] to form a $(63, 63)$ adjacency matrix representation of the graph. We feed this matrix as an input to the convolutional network with a (4,4) kernel. Such kernel shape makes sense when it comes to image input because in images proximity of pixels hold essential information. However, it is not apparent to us how nearby cells in a graph adjacency matrix hold meaning full information. We tried different kernel shapes, for example taking an entire row of the matrix at once (because a row of the matrix represents single nodes outgoing edge weights). However the simple (4,4) gives a better close set performance.

References

[1] M. Hassen and P. K. Chan. Scalable function call graph-based malware classification. In *7th Conference on Data and Application Security and Privacy*, pages 239–248. ACM, 2017.

Copyright © 2020 by SIAM
Unauthorized reproduction of this article is prohibited