Malware Classification

Nihal, Vamsi, Raunak

Problem Definition

Classification of Malwares into one of 9 classes.

Malware Representation - Bytes file containing the Hexadecimal pair

ASM file containing the Raw Text Files

Changing Modalities

Since each malware class has a distinct signature it is intuitive that a convolution network working on the image representation of the problem would give reasonable result.

In order to convert the bytes file into a image, we encode the Hex-pair into their decimal representation, and represent that as a pixel intensity.

The advantage of this method is that the convolution network automatically implements a n-gram model using the moving windows.

Another and more useful advantage of changing modalities is the size of the files. Even without any rescaling the size of the file reduces by a tenth of its original size.

After that we do scaling and downsample the images of size X,16 (where X depends on the number of lines in the bytes file), to 32x32 resolution. After this preprocessing step, the total size of the training and testing dataset falls under 30 megabytes.

Changing Modalities and Conv Net

We tried experiments on images varying in size from 16x16 to 512x512, with everything in between eg-32x16,256x256,512x16 etc. We observe that 32x32 gives similar performance as image sizes above it.

Before downsampling we first duplicated the original Xx16 shape into XxY and then resized it to YxY. It helped in increasing accuracy.

A simple conv net with 3 convolution layers and image size 32x32 gives 96.7% accuracy.

Converting Regular Network to Distributed

- Used the BigDL API from Intel
- Tested Network first on mnist data
- The files were converted to tensors and were stored in an RDD
- The training input for the CNN was the set of images together with its labels
- optimizer.optimize function to start training
- Almost no user community had to refer completely to API handbook

Deploying

Submitting a job on a cluster requires same environment setup all over the cluster.

Instead of installing the required packages on each and every worker, a isolated python environment can be created and can be attached to the archives when submitting the job. (*We just need python-dev and setuptools installed on workers*.)

The scripts to create and adding the environment while submitting the job can be found in our repository page:

https://github.com/dsp-uga/Margaret/tree/master/scripts