MALWARE DETECTION USING MACHINE LEARNING

Ramnath Kumar

27-03-2019

OBJECTIVE

In this document, I will propose my idea of malware detection using machine learning, the progress so far and the tasks left.

Our goal with this report is to summarize my attempts, both failures and successes in categorizing binary files into malware or benign categories.

DATASET

My dataset comprises of roughly 4000 malware binaries and 2000 benign binaries.

Malware was labelled as 1 and our benign as 0 and our goal was to classify any given binary into one of the above. Each binary was read and 8 bits were grouped to give a value between 0 and 255, depicting the intensity of a pixel.

However, each binary were not of the same size. They varied from a sequence of 1502 pixels to more than 900,000 pixels.

We needed a uniform method to classify each of them into two separate categories.

We went over mainly 3 attempts, two which failed and one which performed fairly well.They are as follows:

1. RNN with LSTM
2. Transfer Learning CNN (Deep)
3. Simple CNN (Shallow)

RNN WITH LSTM

This model was tried in a assumption that the sequence of the values in the binary matter more than the values themselves.

To run RNN on our dataset, we need to pad each binary string with zeros to make all of them equal. However, note that this makes our dataset huge with each data being a sequence of 900,000 values.

However, a network with input of 900,000 nodes will not work since we lack powerful GPUs and other resources. Hence, the first goal was to reduce the dimensionality of our dataset.

We tried various dimensionality reduction techniques such as Autoencodes, PCA, LDA, Step-PCA and Truncated SVD. Unfortunately, only truncated SVD was able to run smoothly. The others ended with memory error as the algorithm ran out of virtual memory.

We reduced the 900,000 values into a mere 300 values. This was done so that we could run our RNN on the above data.

The dataset was then split into training set and test set with a train_test split ratio of 0.2. Our model ran fairly fine but could only reach a maximum validation accuracy of 0.74 as shown by the two graphs below:

This could be because of two reasons:

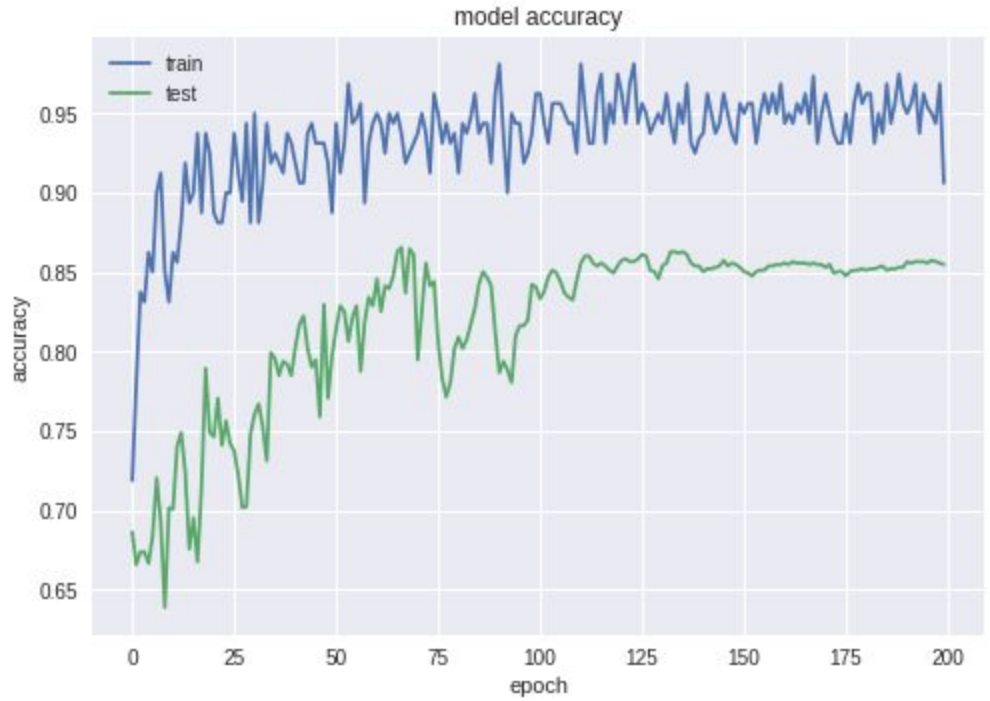
1. Our dimensionality reduction was not a lossless decomposition and we lost information in that process. However, this cannot be fixed given our resources.
2. The sequence of values in the binary do not hold much information themselves and do not play an important role in classifying binaries as malware or benign. Since, there is no paper suggesting so, we assume that this reason is more possible, and we choose to switch away from RNN and move to the CNN.

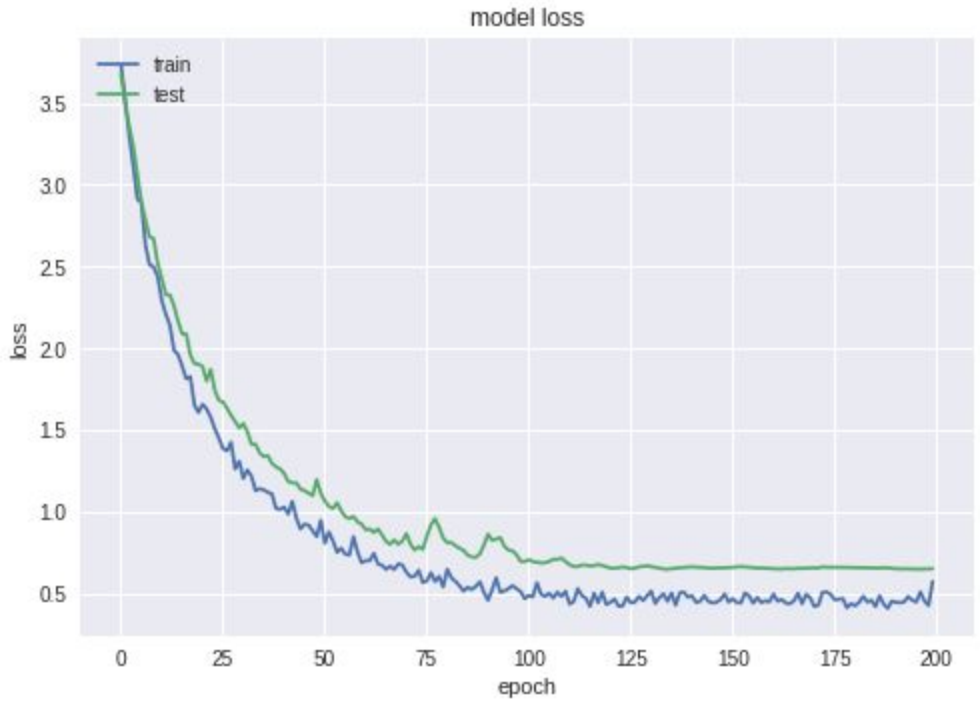
TRANSFER LEARNING CNN(DEEP)

This model was to test out our idea that the spatial arrangement of the binaries play an important role in classification of the binaries into the two subclasses.
CNN allows us to remove both limitations which were raised in our previous approach. We need not use dimensionality reduction and can rather resize the image into a 64*64 pixel image.
Our transfer learning approach involved the use of the Xception model and the fully connected layer. The weights of the Xception net were loaded from the imagenet competition and only the first 15 layers were set as trainable. This was done since, only the beginning layers of the Xception model focuses on basic shape features in the image.

We also included Reduce_Learning_Rate_On_Plateau and Checkpoint to make sure our model converged nicely.

The dataset was again split into training set and test set with a train_test split ratio of 0.2. Our model ran fairly fine but could only reach a maximum validation accuracy of 0.86 as shown by the two graphs below:





Although this model seems to perform better than the RNN approach, the results have not been substantially high which I expected in the case of transfer learning models.

The usual cause for such a condition is:

1. The low amount of data available.
2. Vanishing Gradient problem

There is no point in training a deep network such as Xception only on 6000 images. Such deep models require lakhs and millions of images, which we definitely do not have.

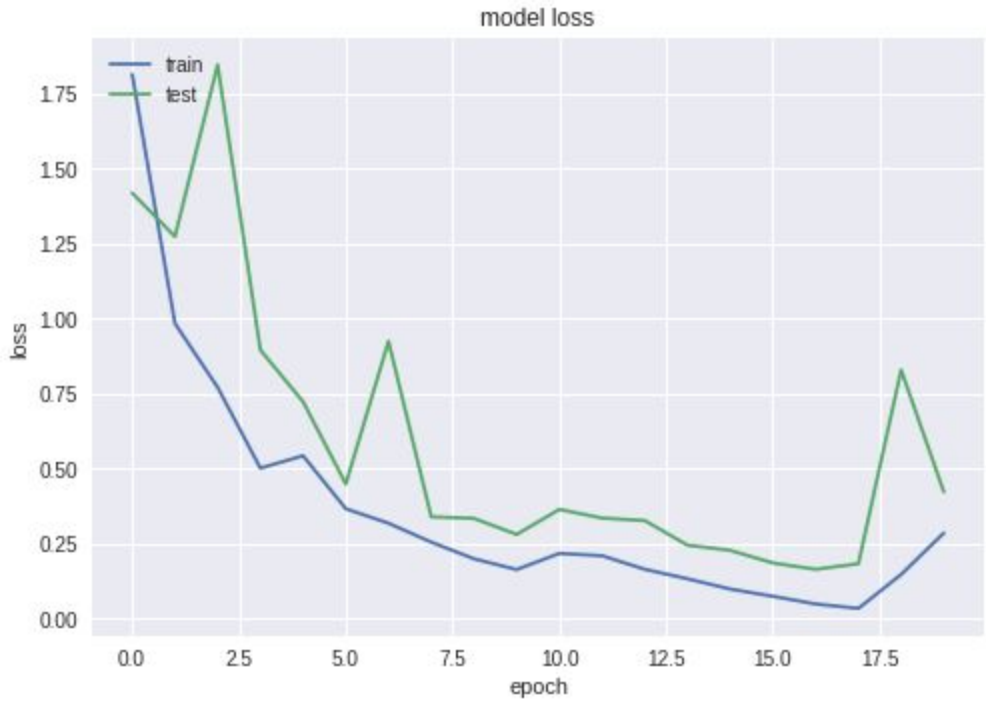
Such deep networks also give us the problem of vanishing gradient. This topic is a very interesting topic and can be covered in the next presentation. The topic is fairly large and would be easier to explain using slides.

Since, we observed that the deep networks do not seem to perform well for our dataset, we switch to a fairly simple and shallow network in our next approach.

SIMPLE CNN(SHALLOW)

This approach seemed to give the best approach. The model was fairly simple. It consisted of only 4 convolution layers and a fully connected layer. Since the model was learning very fast, I included batch normalization, dropout and regularizers. The model shot up to a 96% accuracy within 20 epochs.

The dataset was again split into training set and test set with a train_test split ratio of 0.2. Our model ran fairly fine but could only reach a maximum validation accuracy of 0.96 as shown by the two graphs below:



Therefore, we can conclude by comparing all three approaches and comment that the most simple and lightweight model seems to work best for the given domain and dataset.

Unfortunately, I have not collected other metric scores of all my models such as precision,recall, support, area under ROC curve,etc. which can help in comparing the three models in a better way.

CONCLUSION

This concludes my report. This report can give us a nudge towards the fact that the binaries hold more useful information in their spatial arrangement and not in their sequences. However, we cannot completely support or deny the above statement. No previous paper proves or disproves the above. However, we can see that our lightweight model is doing very well and can easily be deployed to help detect malwares in the IoT domain.