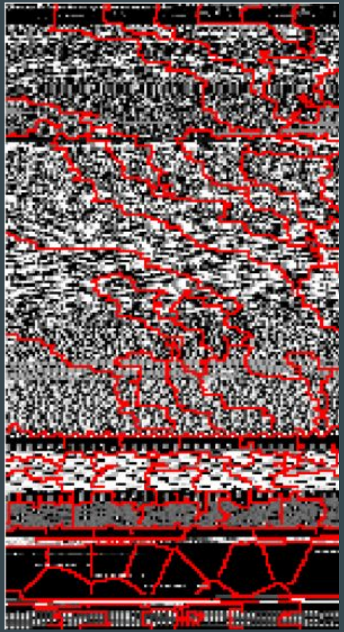
Malware Classification through Computer Vision

Prasann Viswanathan Iyer
190070047

Introduction

- Static Malware classification - without executing the programme
- Signature matching - Standard Method but with limitations
    - Obfuscations in code
    - Exponentially growing number of signatures
- Machine Learning Approaches are more robust and scalable
- Feature Construction issues
    - Disassembly step
    - Binning the types of function calls, counting loops
    - Large Feature space requires reduction with PCA or K-PCA
- Aim is to consider Malware Classification as a Computer Vision problem

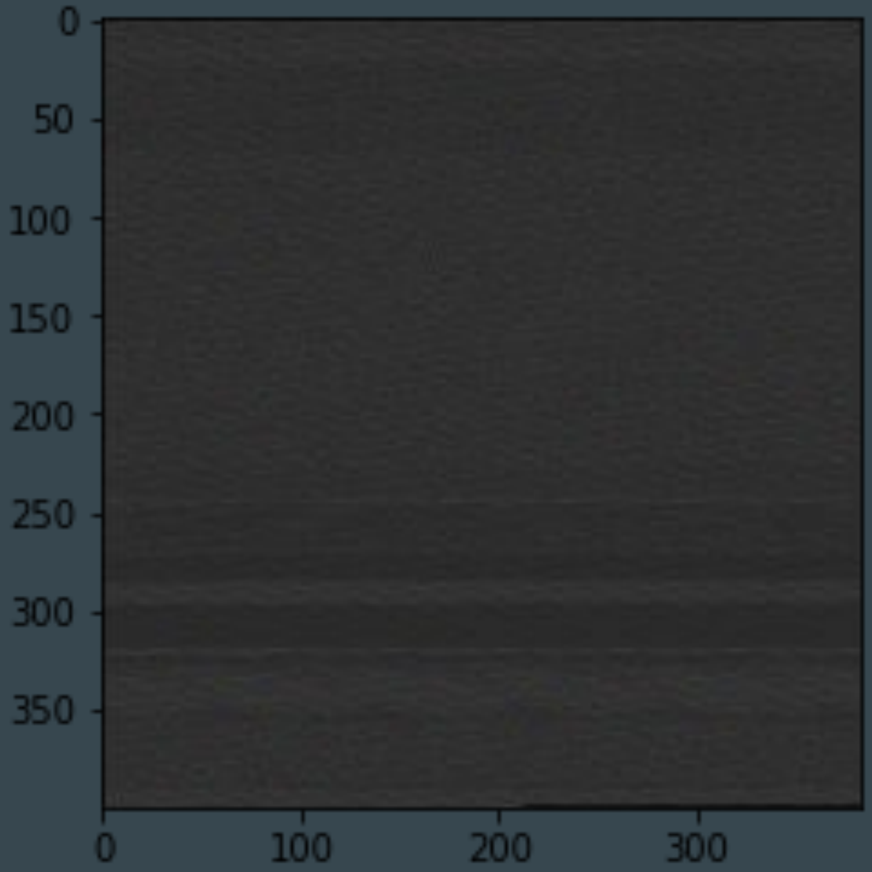
Methodology



- Convert Byte Code of Malware to images
  - Map Binary Code to integers in [0,255]
  - Convert to 2D image with dimensions based on size of Malware file
- Apply Transfer Learning via existing Deep Image Neural Networks
  - Inception
  - ResNet
  - VGG
  - DenseNet
- Classic feature construction method and image method to Decision Forest
- Compare the classification performance

.bytes File to .png Conversion

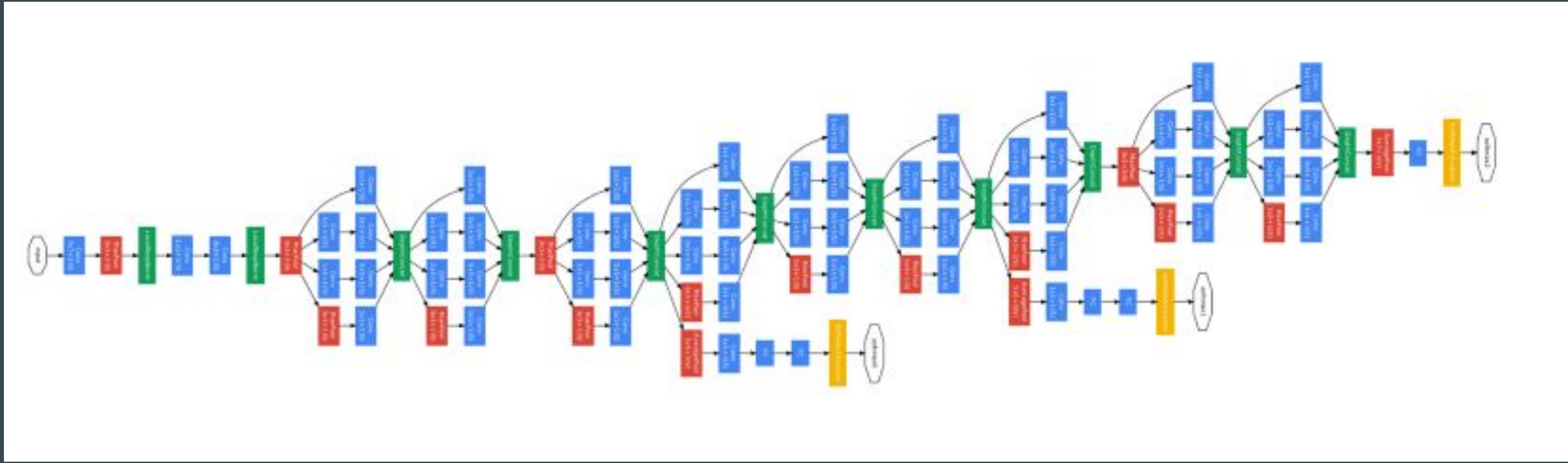
- .bytes file contains byte-wise representation of the malware .asm code
- Each byte represents a number from 0 to 255
  - To construct the R channel of the image we take bytes indexed 0 mod 3
  - To construct the G channel of the image we take bytes indexed 1 mod 3
  - To construct the B channel of the image we take bytes indexed 2 mod 3
- Next we resize the 1D R,G,B arrays to 2D with height and width $\propto$ file size
- My dataset consists of 4307 .bytes files with sizes from 1Kb to 10Kb

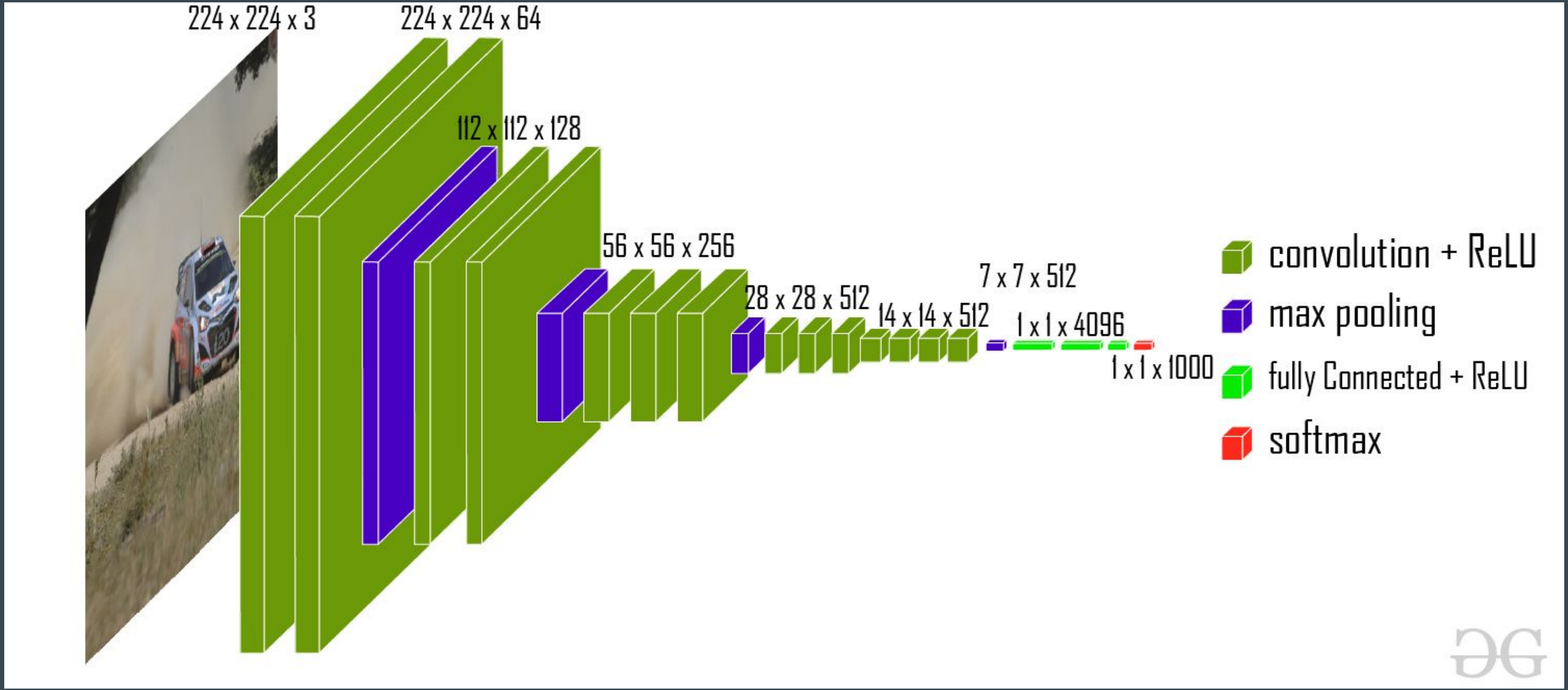

0
50
100
150
200
250
300
350
0
100
200
300

Transfer Learning Models Used

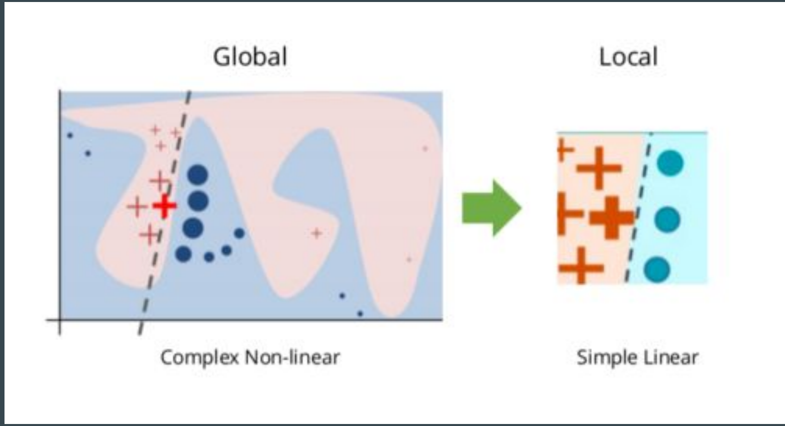
- Inception Network
  - global average pooling layers
  - dense layer with 512 nodes ReLu activation
  - output dense layer with 9 nodes with softmax activation

- VGG16 Network
  - flattened VGG16 model with 14 million fixed weights
  - 2 dense layer with 512 nodes ReLu activation
  - output dense layer with 9 nodes with softmax activation




224 x 224 x 3
224 x 224 x 64
112 x 112 x 128
56 x 56 x 256
28 x 28 x 512
14 x 14 x 512
7 x 7 x 512
1 x 1 x 4096
1 x 1 x 1000
convolution + ReLU
max pooling
fully Connected + ReLU
softmax

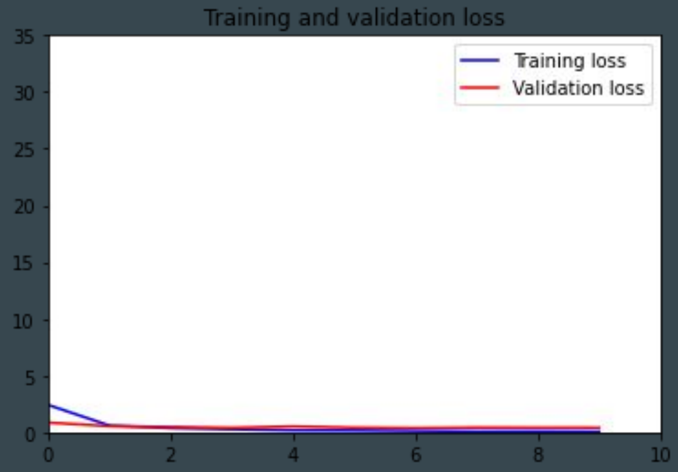
Evaluation and Interpretation

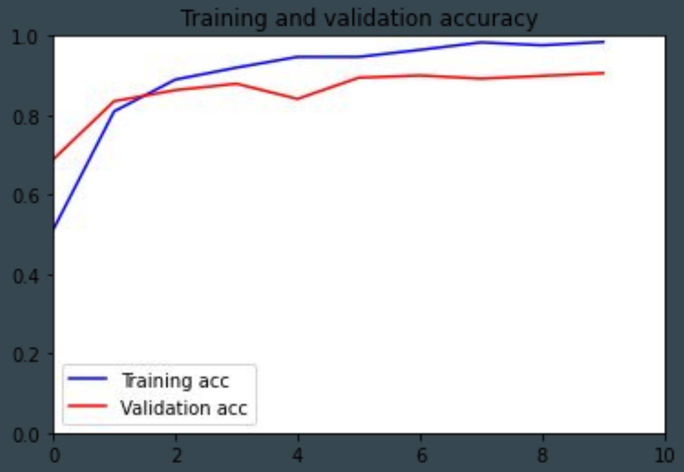


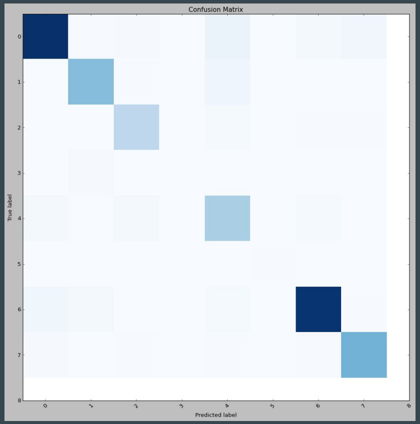
- Classification Metrics
  - Accuracy
  - True Positive Rate
  - False Positive Rate
  - F1 score
- Local Interpretable Model-Agnostic explanation algorithm
  - To understand the reason for our models prediction
- Conclude whether conversion to images assists in Malware Classification
  - With respect to both Classification Metrics and Model Performance Speed

Model Performance

- Accuracy - (98.46% on Training and 90.6% on Test)
- Loss - (0.0555 on Training and 0.3987 on Test)
- Precision - 0.91
- Recall - 0.91
- F1-Score - 0.91







	precision	recall	f1-score	N Obs
0	0.94	0.89	0.91	251
1	0.92	0.90	0.91	107
2	0.87	0.92	0.90	66
3	0.00	0.00	0.00	2
4	0.71	0.85	0.77	88
5	1.00	1.00	1.00	1
6	0.96	0.93	0.94	235
7	0.92	0.96	0.94	112
accuracy			0.91	862
macro avg	0.79	0.81	0.80	862
weighted avg	0.91	0.91	0.91	862

Future Work Directions and Conclusion

- Additional topics not done primarily due to lack of computing abilities
- Training a model from scratch instead of transfer learning (article)
- benign file or a harmful file rather than according to their family
    - Unavailability of a suitable dataset for this
- I could improve performance if I knew which layers to include in the VGG16 model to prevent overfitting to some degree
- Super Pixel Validation technique

Being able to perform so well on RGB photos of malware byte code with only a little fine tweaking is amazing considering that the majority of these models were trained on photographs of dogs, cats, and other pets.

Thank You