University of Nevada, Reno

Department of Computer Science and Engineering

Final Project

Brandon Pastorius

` May 1, 2024.

Project Overview:

This project is aimed to create an anomaly detection program for PCAP files utilizing neural network autoencoders. The dataset that I used was from the MACCDC 2010 dataset that is included in the zip file for this project (or GitHub Repository).

Why you decided to work on this project

I felt as if the implementation of some form of machine learning on a PCAP dataset would be fairly simple and straightforward, while also being able to be efficient in showing the importance of not only the need for more sophisticated methods of effective anomaly detection such as neural networks, but also efficiently showing results when given multiple forms of datasets.

Anything else you would like to incorporate in this section (How to run my code/what it does):

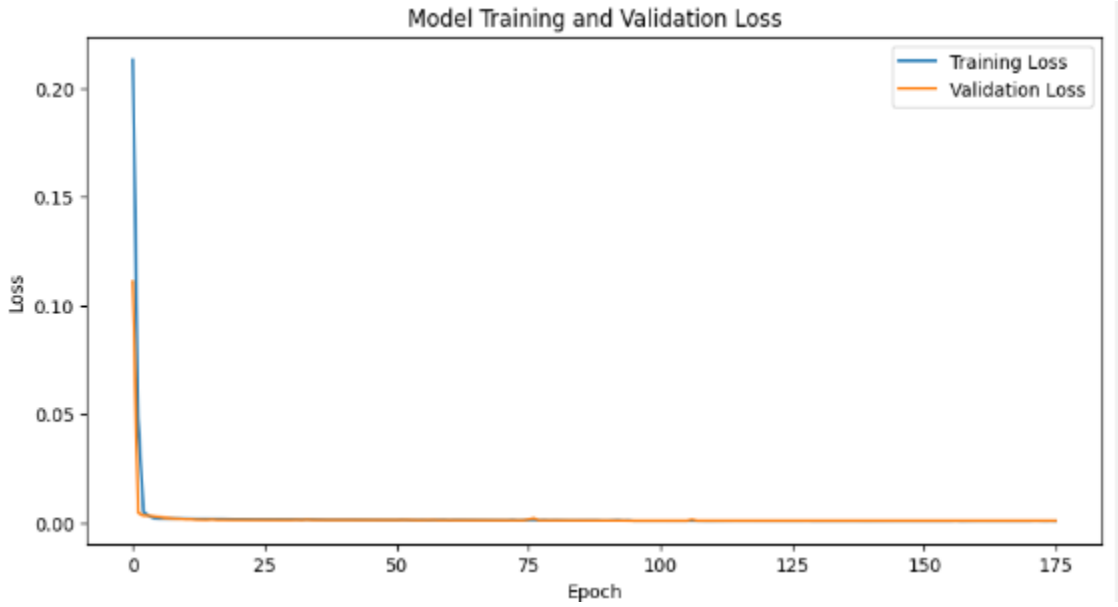
1.) My code is currently formatted to run with the Dataset provided in the zip file/repository titled “maccdc2010_00000_20100310205651.pcap,” in the code this is referenced by “networkData = loadData('/content/maccdc2010_00000_20100310205651.pcap', packet_limit=5000)” under main in case you wish to change this to an alternative Dataset.
2.) The project is created in Google Colab which utilizes .ipynb files (as is attached with this document),
3.) First make sure all libraries are installed utilzing either “!pip install pandas seaborn pyshark scikit-learn tensorflow nest_asyncio asyncio scapy” if you are running this through Google Collab or “pip install pandas seaborn pyshark scikit-learn tensorflow nest_asyncio asyncio scapy” if running it through normal Python in an IDE such as VSCode.
4.) Then run the code attached to this file titled “FinalProj.ipynb“
5.) 3 graphs will be generated,
    a.) showTrainingProgress graph shows the validation and training loss over epochs. This graph is useful as it shows the overall learning progress of the model. Steady decrease in training loss with stable validation loss shows that the model is learning generaralizable patterns (As outlined in screenshot 1,)
    b.) The evaluateModel graph shows the distribution of MSE values on the dataset, this visualizes the spread and error resulting frmo the encoder attempting to reconstruct the dataset. The majority of the data points should cluster around lower values and spikes indicate anomalies in the network data.
    c.) This is the most important graph as the plotErrors graph shows each data point’s MSE with a threshold line, the points that have values higher than this line are anomalies in the network data.

What are your motivators:

Machine learning is a very fast growing aspect of computer science and I believe the current trend of increasing results rising drastically in parallel with interest in the field [1]. I believe with the rise of popularity of Artifical Intelligence as a whole it will become the dominant field of study within computer science. Due to not many classes I have taken allowing me the freedom to incorporate these emerging concepts I decided to take advantage of this opportunity to incorporate something I have been interested in for awhile with aspects learned from the slides for class in order to create this final project

Results and Details:

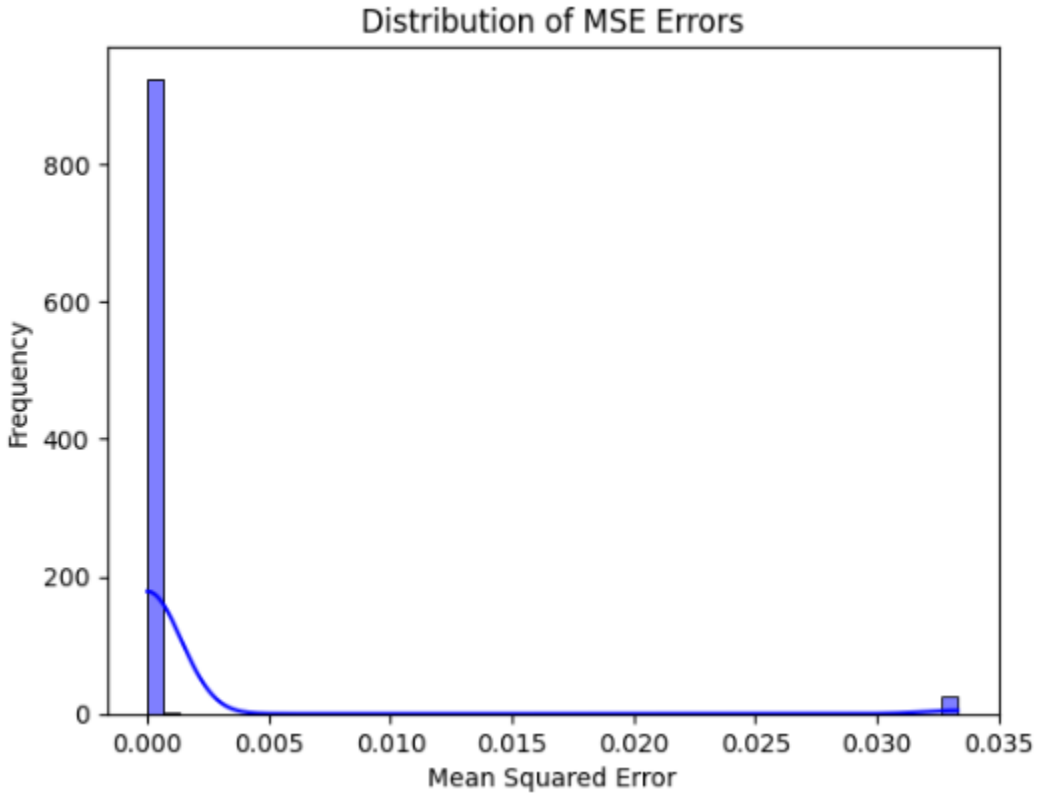
.As outlined before the dataset I started with was a pcap file from the MACCDC 2010 dataset. The 'scapy' library allowed me to get features from the packets in this file like the length of the packets, transport layer protocols, and destination/source internet protocol addresses. The loadData function in my script reads and parses pcap files, then it is processed in preProcessData to prepare for the neural network encoding. The autoencoder is created utilizing TensorFlow's Keras modules, which the documentation that comes with this libraries modules was a huge help in how I created this project. The architecture of the model is created to learn efficient representation of what normal traffic patterns should look like then reconstruct the dataset from this representation. The model then minimizes the MSE (mean squared error) of the original dataset and the reconstruction of the dataset, primarily utilzing the normal traffic it learned upon earlier to detect anomalies in data. The autoencoder utilizes an input layer to match the features in a given set of data, following by 3 encoding layers with 64, 32, and 16 neurons for the network. The decoding layer then attempts to mirror the encoding inversely.



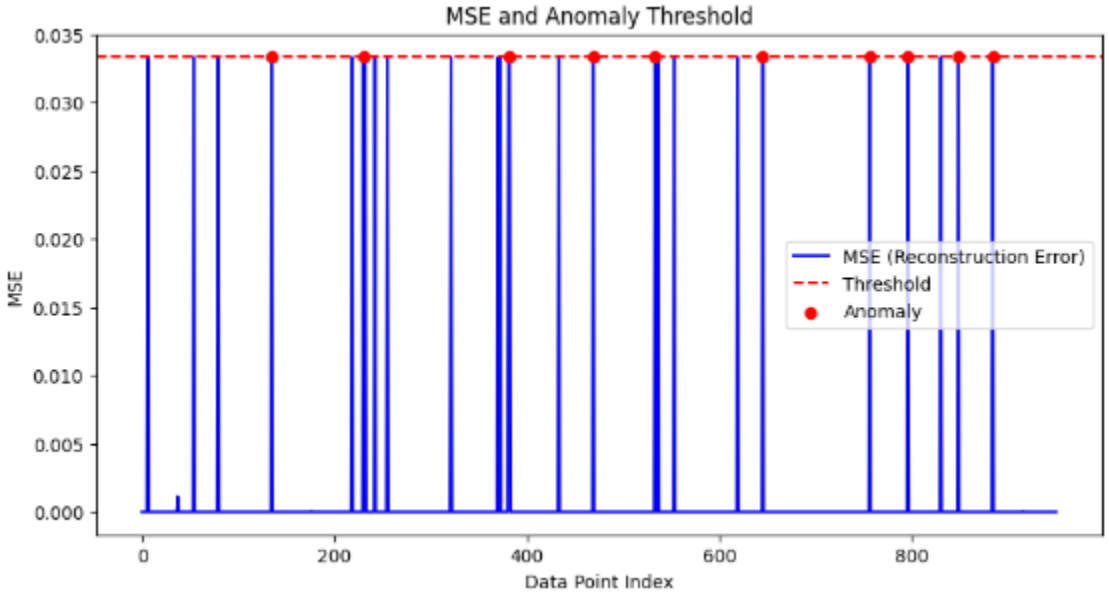
This first graph outlines the training loss and validation loss over each Epoch. This is useful as it shows the learning progress of the encoding model, indicating how quickly the

[1] https://www.forbes.com/advisor/business/ai-statistics/

model learns generalizable patterns. The sharp decrease in the beginning shows that the model trains from this dataset relatively quickly.



This graph shows the distribution of Mean Squared Error values within the data. Lower values are good as they show normal data while higher values could represent anomalies.



This graph is the most important as it actually outlines anomalistic data within the dataset that gets detected, plotting each data packet's Mean Squared Error values with a threshold that is the 99th percentile of errors to catch massive outliers in given datasets to catch the most anomalistic data. The anomalies caught are unusually large packet sizes during

normal traffic situations. Then flagging each anomaly that actually passes that threshold (not just getting close to it) due to the large deviation from the norm. This suggests towards there potentially being a security threat in the packets processed.

Tools and technologies utilized:

Python was the main language used, the project was done in a Google Collab Notebook. The libraries utilized were pandas, numpy, matplotlib, seaborn, scarpy, and Tensorflow. Scapy handled the parsing of pcap files to generalize network data. Tensorflow and the Keras modules provided the autoencorder neural network model.

Conclusion

Were the results consistent with your initial expectations?:

Yes they were consistent with initial expectations as initial expectations were just to make the autoencoder learn the patterns of normal network behavior and detect extreme anomalies in network traffic which I was confident in what the results would look like.

Do you plan to continue doing personal research and expand in this focus area?

Yes machine learning interests me greatly as I believe it will be a dominant aspect of computer science within the next few years and I am grateful for the opportunity to combine exploring this technology with pairing it for a final project in one of my courses.

What have you ultimately learned?

I have ultimately learned how to use Google Collab Notebooks and Python to create an anomaly detection program. This project also provided insights to me about tying machine learning to cyber security.

Work Cited:

https://www.youtube.com/watch?v=ceu7CESRsI0
This video started my interest in particularly using Autoencoders and starting to understand how they work.
https://towardsdatascience.com/build-the-right-autoencoder-tune-and-optimize-using-pca-principles-part-i-1f01f821999b
Was the main help for setting up the autoencoder and using keras.
https://anno-ai.medium.com/scalable-machine-learning-for-packet-capture-data-with-kubeflow-b485a64c870a
Where the idea for the threshold to find anomalous data came from.
https://www.geeksforgeeks.org/python-mean-squared-error/
Mean Squared Error tutorial
https://codewithgolu.com/python/network-packet-analysis-with-scapy-a-beginner-s-guide/
Taught me how to start utilizing Scapy.
https://scikit-learn.org/stable/user_guide.html
Scikit learn user guide
https://www.tensorflow.org/guide/keras
Tensorflow guide for Keras