Machine Learning Algorithms For Network Anomaly and Intrusion Detection

Authors: Henrique Sousa, 98324 and Vasco Regal, 97636 Computer Science Students from
Departamento de Eletrónica, Telecomunicações e Informática at University of Aveiro
Tópicos de Aprendizagem Automática
Professor: Pétia Georgieva

***Abstract*—Machine Learning models have proven to be effective in various business fields, such as recommendation systems or spam detection. In the context of network and cyber security, we can predict and prevent malicious activity by analyzing network traffic patterns. In this paper we attempt to create robust Machine Learning models able to classify packets as known exploits like injection, impersonation and flooding.**

***Index Terms*—Network Security, Machine Learning, Deep Learning**

I. Introduction

Cyber Attacks can wreak upon an organization and be extremely expensive. In this article we will start by looking at the basic of Machine Learning in cyber security using Python with the help of Hands on Machine Learning for Cyber Security book by Packt Publishing [4] and implement some Machine Learning models in order to detect malicious network activity. Although using Machine Learning in this context has a decent amount of attention in the academic field, after doing some research, we found that in the ”real world”, these algorithms are generally not implemented since they require careful planning before being deployed. [7] , [6].

II. State of the Art

Staffan Truvé, CTO and Co-Founder of Recorded Future states in his article ”4 Ways Machine Learning Is Powering Smarter Threat Intelligence” [9] that in recent years, humanity’s ability to make accurate predictions have improved in many fields thanks to a combination of augmented sensor capabilities and new prediction algorithms. The detection of Security Threats is one of those fields and in this paper Steffan Truvé talks about the ways Recorded Future uses machine learning to analyze data, structure this information at scale, and present threat intelligence for sharing with humans or security systems. One of them is Natural Language Processing which transforms unstructured text in multiple languages into a structured representation. The use of natural language processing allowed them to build a system capable of analyzing millions of documents per day, something that would be impossible for humans, in eight different languages, and to transform that data into a representation that gives analysts insights, independently of their language skills. Humans can read and understand text, but machines can do that in a massive scale and with huge amounts of data as seen in figure 1.

Figure 1: Human vs Machine Processing Power

According to Vitaly Ford and Ambareen Siraj in their article ”Applications of Machine Learning in Cyber Security” [10] Machine Learning techniques are well known for their adaptability, scalability, and potential to rapidly adjust to new and unknown challenges. A lot of machine learning methods have been successfully deployed to address such wide-ranging problems in computer security. Phishing detection, network intrusion detection, testing security properties of protocols, authentication with keystroke dynamics, cryptography, human interaction proofs, spam detection in social network, smart meter energy consumption profiling, and issues in security of machine learning techniques itself are the topics discussed in this paper. Starting with Phishing, it is a way of stealing personal sensitive information. There are three principal anti-phishing methods: detective, that works by monitoring the account life cycle, filtering content, anti-spam and anti-malware, preventive, based on authentication and change management, and, finally, corrective, that works by taking the phishing website down, and forensics and investigation. Network Intrusion Detection system are used to identify malicious network activity leading to confidentiality, integrity, or availability violation of the systems in a network. This systems are usually based on machine learning techniques due

to their adaptability to new and unknown attacks. Regarding CAPTCHAs some researchers have already discussed how the Human Interaction Proofs can be broken by utilizing machine learning. Several experiences were done in order to prove that these systems are not threat proof. As we can see from the countless examples stated in this article Machine Learning is an effective tool that can be employed in many areas of information security. There are some robust anti-threat systems developed out there and some models can be successfully used for developing authentication systems, evaluating the protocol implementation, assessing the security of human interaction proofs, smart meter data profiling, and many other cases. This paper concludes by explaining that although machine learning facilitates keeping various systems safe, the machine learning classifiers themselves are vulnerable to some malicious attacks.

On 802.11 network analysis, several publications present robust Machine Learning algorithms which can accurately identify threats and anomalies.

On [3], the authors present the creation of a network analysis dataset, now known as the AWID dataset [1]. For this dataset, several classic machine learning algorithms were implemented with accuracy scores reaching 96%. This particular paper details the features they considered crucial to analyze a network packet, consisting on collecting data from the network frame, the radiotap and finally the wlan. This paper spiked our interest due to the diversity of the features, which, as will be mentioned on further sections of our report, ended up being the base for the dataset we trained our models with.

Also on this topic, On [8] a Neural Network model was developed with hyper parameter tuning and deep learning methods. In this paper, the creators demonstrate in various tables and graphics the resulting scores of their implementations with variations in number of hidden layers and different activation functions, where it was concluded that with the J48 classifier, 2 hidden layers and PReLU function worked the best. This work outputted an accuracy of 98.6688%, claiming the second best performance we could find. Figure 2 summarizes the tests done with the aforementioned classifier and hidden layers.

III. AWID Dataset

To train and test our models, the AWID2 dataset [1] was used. Each entry of this dataset contains more than 150 fields regarding a network packet captured on a 802.11 network and classifies it as a normal network packet or as a malicious packet. The type of attack is also identified, as the malicious packets are also subcategorized in injection, impersonation or flooding, which represent the type of exploit. A more in depth analysis of the generation and content of this dataset was published in [3], which we studied to better understand the data we were dealing with.

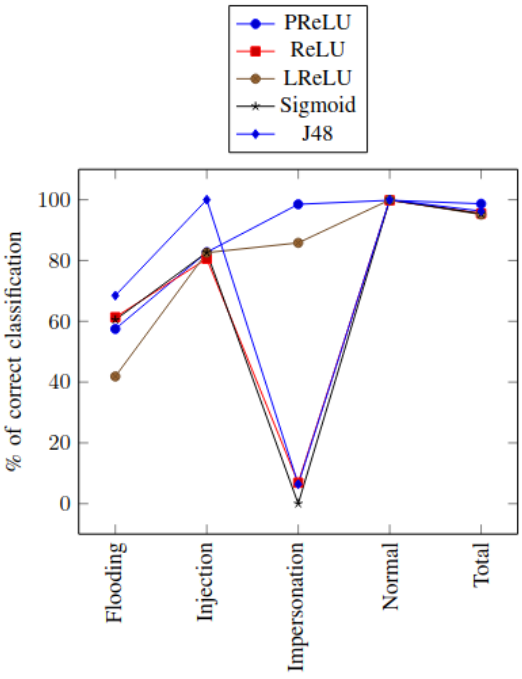

Figure 2: Vrizlynn L. L. Thing's paper test results

IV. Dataset Pre-Processing

The original dataset contains 1795575 entries. Since we don't have the computational power to effectively handle such a volume of data, we decided to trim the dataset, reducing it to 1000000 entries. Although this could impact our results when faced with solutions working with the original dataset, for this project's scope, we believe decent conclusions can still be achieved. Despite the fact that columns are unnamed in the dataset, we were able to identify them and transform them in meaningful feature names with the help of the example provided in the Hands On Github [5]. Features are categorized in **frame**, **radiotap** and **wlan**.

A. Removing Unnecessary Data

In this dataset, unknown values are marked with a "?". We start by replacing this mark with the **None** value, so python can interpret it as a missing field. Then, we noticed a lot of features had mostly None values, so we dropped the columns with more than 40% missing values, which resulted in discarding 72 features. We also removed any non-numerical feature, leaving us with 83 of the original 155 features. Rows with missing values were also removed, resulting in a dataset with 782934 entries.

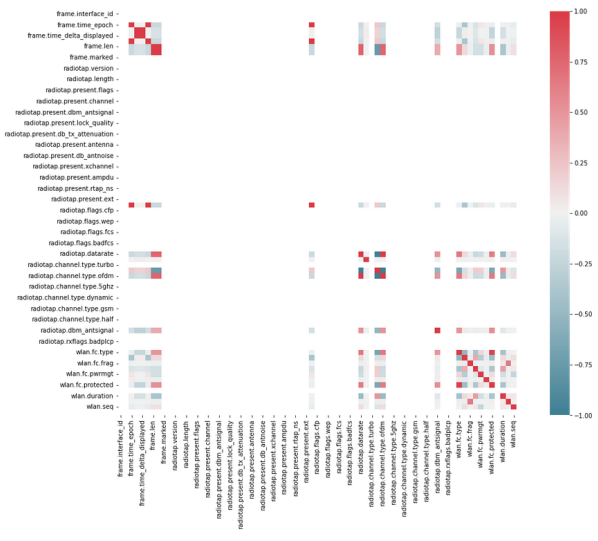

Figure 3: Correlation Matrix

Feature 1	Feature 2	Correlation
wlan.fc.type	wlan.fc.protected	0.9554845088605738
radiotap.channel.type.ofdm	radiotap.datarate	0.9817854008068971
frame.time_relative	radiotap.mactime	0.9999999999998339
radiotap.mactime	frame.time_epoch	0.9999999999998621
frame.time_epoch	frame.time_relative	0.9999999999999979
frame.time_delta	frame.time_delta_displayed	1.0
frame.len	frame.cap_len	1.0

Figure 4: Correlated Features

B. Feature Selection

With this filtered dataset, a correlation matrix, figure 3, was generated to identify which features could also be discarded due to high correlation. Looking at the plot, we can summarize the correlation in a table. We discarded features with values above 0.95, so looking at figure 4, we discarded the following columns:

- frame.cap_len
- frame.time_delta_displayed
- frame.time_epoch
- radiotap.mactime
- radiotap.datarate
- wlan.fc.protected

C. Balancing the Data

This dataset has an unbalanced distribution on classes, as seen on figure 5. To obtain decent results in all our models, an even class distribution was achieved by randomly undersampling to the number of samples of the lowest class. This resulted in an equal ratio of classification cases, shown in 6, thus creating our final dataset of 193936 samples, which will be used to develop the algorithms.

D. Splitting the Dataset

The dataset was split into training and testing subsets, with a 80:20 ratio, respectively. This testing subset will be used to test the performance of the model after training it with the training subset.

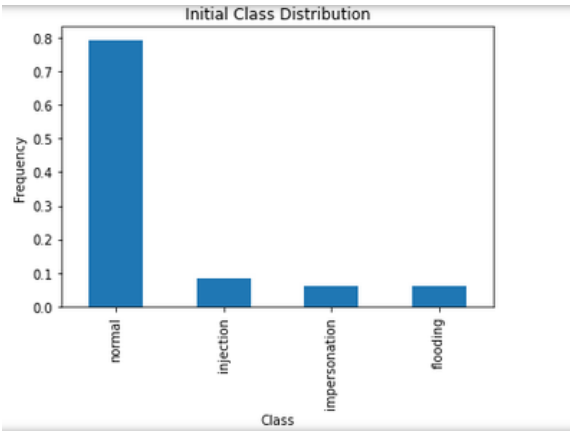

Figure 5: Unbalanced class distribution

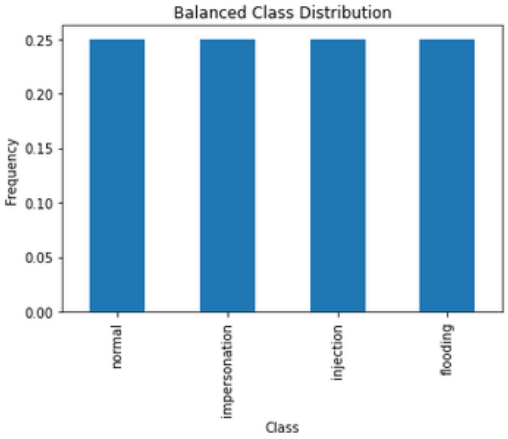

Figure 6: Balanced class distribution

E. Categorical Encoding

Encoding categorical data was one data preparation task which we considered to be crucial. Our data came with categorical string values and most of the machine learning models work with integer values. For that we did Categorical Encoding which is a process of converting categorical data into integer format so that the data with converted categorical values can be provided to the different models. In our case, we associated the values as the Table I shows.

String Value	Integer Value
flooding	0
impersonating	1
injection	2
normal	3

Table I: Categorical Encoding Results

V. Models

To get the the best machine learning model to identify malicious network activity we tested various different models and took the best ones in order to apply Hyper-parameter tuning and find the best parameters for that model. The results vary for each one of the types of models used. Lastly, we tried a deep learning approach with the help of TensorFlow[1] and Keras[2] python libraries. We are looking for a model with at least 87% accuracy.

A. Naive Bayes

Naive Bayes Classifier is a probabilistic machine learning model that's used for classification task. It is based on the Bayes theorem. First the Naive Bayes

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Figure 7: Naive Bayesian equation

algorithm converts the data set into a frequency table, then it creates a likelihood table by finding the probabilities, next it uses 7 to calculate the posterior probability for each class where the one with the highest probability is the outcome of the prediction. Similar to the other models we used *sklearn* [3] library, specifically *sklearn.naive_bayes.GaussianNB*. This model returns an accuracy of 74% and F1 scores of 0.49, 0.96, 0.74, 0.71 for flooding, impersonation, injection, and normal, respectively. The confusion matrix based in this results can be seen in Figure 8

Although the initial accuracy for this base model was not so bad, the F1 scores were too unbalanced and we decided to discard this model and not hypertune it, since there were other models with much better initial results.

B. Logistic Regression

Logistic regression is a process of modeling the probability of a discrete outcome given some input variables. To use the Logistic Regression model on our dataset we used the *sklearn.linear_model.LogisticRegression* library. This model is usually used in binary classification problems so as expected the results were not so good either. With only 68,4% of accuracy and F1 Scores of 0.61, 0.83, 0.63 and 0.63 for flooding, impersonation, injection and normal cases, respectively, and the confusion matrix we can see at figure 9.

After analysing these results we end up discarding this model and did not try to hypertune it since the base

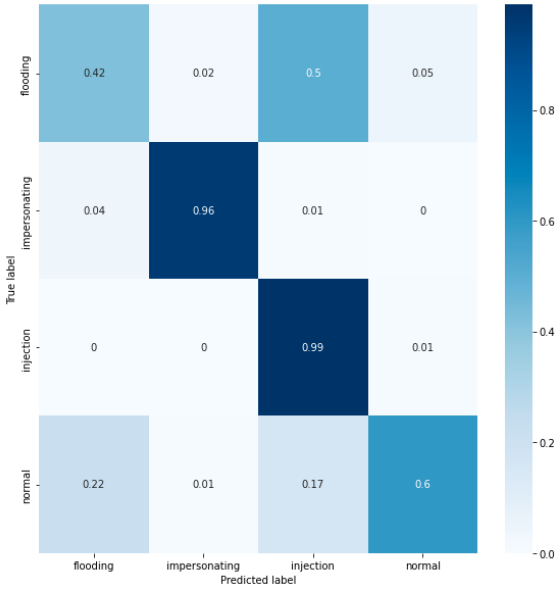

Figure 8: Naive Bayes Confusion Matrix

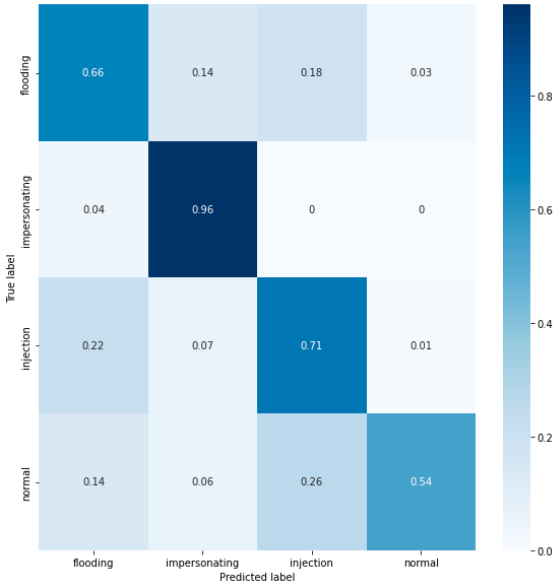

Figure 9: Logistic Regression Confusion Matrix

model results were so low when comparing to the next models we tried.

C. Random Forest

Random forest is an ensemble learning method for classification, regression, and other tasks that operates by constructing a multitude of decision trees at training time. We used *sklearn.emsemble.RandomForestClassifier* to train this model and got an accuracy of 92.1% and F1 Scores of 0.88, 0.95, 0.97 and 0.86 for flooding, impersonation, injection and normal cases, respectively

[1] **TensorFlow** Resources: https://www.tensorflow.org/resources/libraries-extensions

[2] **Keras** Documentation: https://keras.io/

[3] **scikit-learn** documentation: https://scikit-learn.org/

on the hyper tuned model. To get these values we had to tweak the model parameters, namely the max_depth of the tree, presented in table II. We also used k-fold cross validation during training, with k=5.

max_depth	model accuracy
1	89.1%
2	90.43%
3	92.1%

Table II: Accuracy variation with max_depth hypertuning

This model got us a better confusion matrix Figure 10 and we were able to achieve our goal of 87%+ accuracy.

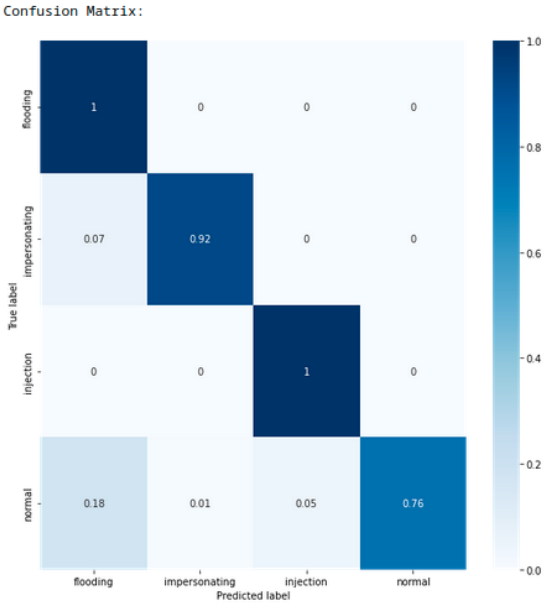

Figure 10: Random Forest Confusion Matrix

D. Neural Networks

Deep learning is a form of machine learning and artificial intelligence that mimics how humans learn specific subjects. Data science, which also encompasses statistics and predictive modeling, contains deep learning as a key component. A neural network is an example of a deep learning method used in machine learning. It works by simulating a large number of interconnected processing units that resemble abstract version of neurons. To help us implement deep learning in python we used TensorFlow and Keras to find the best model and set of parameters (HyperTuning) that gave us the best results.

Parameters that were varied:

- **learning_rate** - learning rate value
- **num_layers** - number of hidden layers
- **units_1** - number of neurons in hidden layer 1
- **dropout_1** - dropout rate for hidden layer 1
- **units_2** - number of neurons in hidden layer 2
- **dropout_2** - dropout rate for hidden layer 2
- **units_3** - number of neurons in hidden layer 3
- **dropout_3** - dropout rate for hidden layer 3
- **units_4** - number of neurons in hidden layer 4
- **dropout_4** - dropout rate for hidden layer 4
- **units_5** - number of neurons in hidden layer 4
- **dropout_5** - dropout rate for hidden layer 5

We varied the number of hidden layers from 1 to 5 with units from 32 to 512 with a 32 step-size for each of the layers and dropout rate of 0.1, 0.2, and 0.3, the learning rate was tuned for the following values: 1e-2, 1e-3. and 1e-4 with *adam* optimizer. We choose to use *adam* optimizer because after testing other optimizers like *sgd*, the one that gave us the best results was *adam* and according to our research generally *adam* is a better optimizer. For the hidden layers we used *relu* activation and *softmax* activation for the output layer with 4 units. We used *relu* activation function on the hidden layers because after some manual testing it gave us the best results when comparing with *sigmoid* and *tanh*. The *softmax* function was chosen because it is used as the activation function in the output layer of neural network models that predict a multinomial probability distribution. [8] also influenced our parameter selection.

After 300 trials the best model ended up getting a 96.52% accuracy, a lot more than the 87% accuracy value that was our goal with this work. F1 Scores were 0.96, 0.95, 0.99 and 0.95 for flooding, impersonation, injection and normal cases, respectively. The parameters that were found to be the best ones were the ones in Table III.

Parameter	Best Value
num_layers	4
units_1	160
dropout_1	0.1
learning_rate	0.0001
units_2	352
dropout_2	0.2
units_3	128
dropout_3	0
units_4	192
dropout_4	0.2
units_5	320
dropout_5	0.2

Table III: Best Parameters found with Hypertuning

Since the accuracy value was very high for this model the confusion matrix that was generated had a lot fewer wrongly predicted instances as we can see in Figure 11

We can better visualize the model accuracy along the epochs by looking at Figure 12 and the model loss in Figure 13.

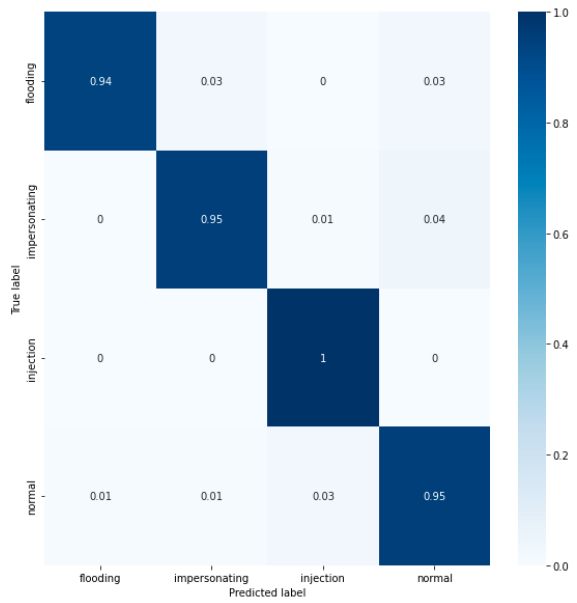

Figure 11: Deep Learning Confusion Matrix

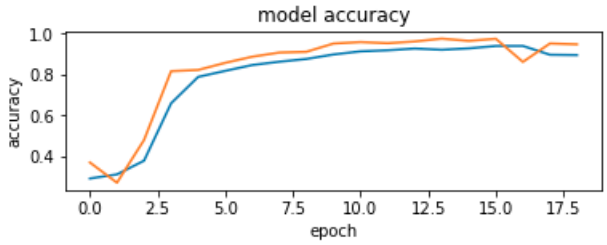

Figure 12: Neural Network Model Accuracy

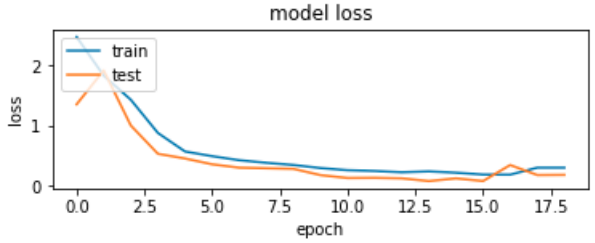

Figure 13: Neural Network Model Loss

VI. Results

A. Discussion and Interpretation

Comparing our developed models, we can safely conclude that the Neural Network has the best performance, as expected since, unlike the others, a deep learning approach was taken in its development. The summarized performances can be analyzed in figure 14

Another interesting result is the fact that Injection attacks are the most consistently identifiable with this set

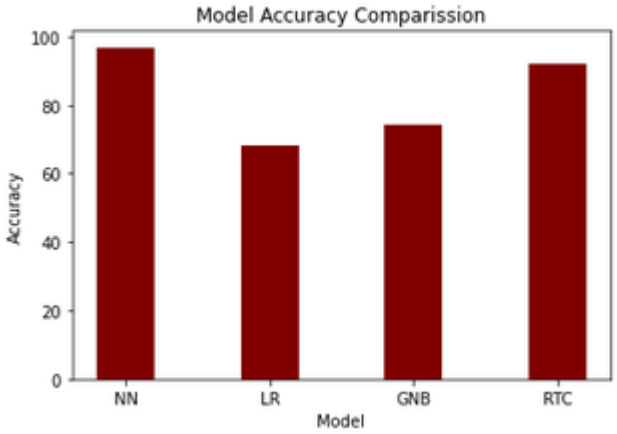

Figure 14: Accuracy of each Model

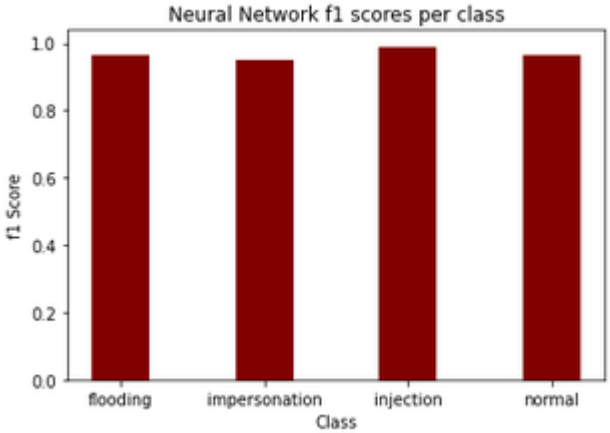

Figure 15: F1 Score Comparison

of features. In fact, our best model successfully predicted all cases of actual injection occurrences. We can confirm this behaviour by looking at the f1 score distribution for each label on figure 15.

B. Comparing with Other Works

On [4]'s apporach, a few classic models were developed. In table IV, their results are presented compared to ours. To note that in their work, the whole dataset was used. Also, as mentioned before, we did some extra pre-processing, like removing some correlated features and undersampling the data. No deep learning techniques were used either on their algorithms.

A similar Neural Network with deep learning approach was implemented using the AWID dataset on [8]. The final result obtained is slightly better than ours, with a 98.6688% accuracy. Their score can be justified on the larger volume of data, since the full dataset was used.

Eight traditional supervised algorithms were developed with the AWID dataset on [3], with accuracies

Model	Accuracy (Ours)	Accuracy (Theirs)
Naive Bayes	74%	26.5%
Logistic Regression	68.4%	15.7%
Random Tree	92.1%	93.4%
Neural Network	96.52%	-
Best Result	96.52%	93.4%

Table IV: Model Comparission with Packt Publishing's book work

ranging from 89.43% to 96.2%. This particular implementation uses hand picked feature selection, for a total of 20 fabricated features.

Finally, on [2], the attack classes were separated in different subsets and different algorithms were implemented for each class. An accuracy of 99.86% was obtained for impersonation attacks, which used Neural Networks and SVM with deep learning techniques. Using different algorithms for different classes is something we did not do and could improve our results

VII. Conclusion

With the results obtained, we can conclude that, although lacking the computational power to effectively use the whole dataset, we were able to obtain decent results in our models, comparing to other works we could find.

Moreover, we were able to understand how powerful deep learning techniques can be in the Machine Learning field. While we were struggling to get past the 92% accuracy with manual model tweaking, by using keras and tensorflow we quickly reached our best performance.

VIII. Novelty and contributions

While this dataset has very limited community contributions on online platforms like Kaggle and Github, we were still able to gain some insight on the topic on several publications.

The pre-processing of the dataset was inspired by the work presented on PacktPublishing's book [4] (pages 194-206), which we obtained a copy. Their description of some of the features also helped our feature selection.

We used the information on [8] to guide us through which neural network parameters are supposed to be the most optimal.

[3] inspired us to make a more in-depth feature analysis and selection which significantly improved our models.

References

[1] University of the Aegean. *AWID - Aegean Wi-Fi Intrusion Dataset*. URL: https://icsdweb.aegean.gr/awid.

[2] M. E. Aminanto. *Weighted Feature Selection Techniques for Detecting Impersonation Attack in Wi-Fi Networks*.

[3] Constantinos Kolias et al. *Intrusion Detection in 802.11 Networks: Empirical Evaluation of Threats and a Public Dataset*. URL: https://ieeexplore.ieee.org/document/7041170.

[4] Packt Publishing. *Hands On Machine Learning for Cyber Security - Book*. URL: https://www.packtpub.com/product/hands-on-machine-learning-for-cybersecurity/9781788992282.

[5] Packt Publishing. *Hands On Machine Learning for Cyber Security - Github Repository*. URL: https://github.com/PacktPublishing/Hands-on-Machine-Learning-for-Cyber-Security.

[6] Robin Sommer and Vern Paxon. *Outside the Closed World: On Using Machine Learning For Network Intrusion Detection*. URL: https://personal.utdallas.edu/~muratk/courses/dmsec_files/oakland10-ml.pdf.

[7] Jonathan Spring et al. *Machine Learning in Cybersecurity: A Guide*. URL: https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=633583.

[8] Vrizlynn L. L. Thing. *IEEE 802.11 Network Anomaly Detection and Attack Classification: A Deep Learning Approach*. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7925567&tag=1.

[9] Stafan Truffé. *4 Ways Machine Learning Is Powering Smarter Threat Intelligence*. URL: https://go.recordedfuture.com/hubfs/white-papers/machine-learning.pdf.

[10] Ambareen Siraj Vitaly Ford. *Applications of Machine Learning in Cyber Security*. URL: https://www.researchgate.net/publication/283083699_Applications_of_Machine_Learning_in_Cyber_Security.