Machine Learning vs Deep Learning for Identifying DDoS Traffic

Alexander Wolosewicz, Alec Burnworth, Ranjitha Aswath, Andrew Cook

***Abstract*—Internet traffic classification and attack recognition is an active field of research. DDoS attacks are a growing problem, and using AI could enable organizations to defend themselves in a cost-effective and self-evolving manner. Our work uses a newly-released DDoS attack dataset to compare the effectiveness of Random Forest and Neural Network methodologies in approaching the problem of DDoS attack detection. We find that Neural Networks show greater potential for a final model; however, the Random Forest is not without merit. We use our Random Forest model to narrow the originally 315 features into a much smaller feature space to train the Neural Network on, and evaluate it against examples held out of the dataset. We find that it is able to confidently class some attacks, but confidently misclassifies others as benign.**

I. Introduction

DDoS attacks are a growing problem worldwide, for both public and private organizations. Cloudfare, a content delivery network service used by nearly 19% of the internet, stated in 2024 that 6.8% of all internet traffic is malicious with 37.1% of them being DDoS attacks [1]. Cybersecurity Ventures predicts that the economic cost of DDoS attacks worldwide will continue to grow, reaching $10.5 trillion by 2025 [2].

There are traditional difficulties when it comes to automatically analyzing DDoS attacks. DDoS Attacks are constantly evolving, and traditional methods struggle to classify traffic fast enough and accurately enough to be effective [3]. AI models could theoretically address these, being trained so that they can classify traffic and respond to threats in real-time. Further, they would be able to generalize to possibly identify and address novel attacks in a way traditional methods cannot.

Historically, research into using AI to address DDoS and other attacks has suffered due to a lack of modern datasets. In a 2018 survey on the use of machine learning in networking [4] it was found that the majority of current research used the KDD'99 [5] dataset, which as its name suggests dates to the year 1999. This predates many modern attack techniques, the prevalence of end-to-end encryption, the vastly higher bandwidth of benign traffic flows, and many other features of the modern internet. To address this, the BCC-cPacket-Cloud-DDoS-2024 dataset was published which incorporates modern attacks and modern benign traffic patterns.

Our goal is to use this modern dataset to evaluate whether the machine learning technique of Random Forest or deep learning Neural Network techniques are better suited to this classification problem. We classify in a binary fashion whether a flow is an attack or benign, using just the Attack and Benign labeled rows in the dataset.

For our work, we use a cleaned version of the dataset found on Kaggle [6] as our base dataset due to issues obtaining the original version on time for our work. This version cleans the original dataset by filling NaN values as 0, dropping duplicate rows, removing known contaminating features (flow ID, timestamp, and IP addresses), dropping the protocol column due to 0 variance, and some datatype and value cleanup. We do some further modifications of our own, such as dropping all rows labeled Suspicious (uncertain if Attack or Benign), as well as some additional per-model pruning which is described later.

This paper details our work training Random Forest and Neural Network models on the dataset, evaluating them, then building a synergistic regimen using Random Forest to reduce the feature space the final models are trained on. The final models are then evaluating, achieving excellent performance at accurately identifying benign flows while being less accurate at discerning certain attack flows.

II. Implementation

A. Random Forest

The scikit python library was used to generate the random forest model.

In the first iteration of the random forest model, data preprocessing involved removing rows where the packet duration is zero or labeled as 'Suspicious' to remove ambiguity in the training phase. Input features with only one unique value were removed as they provided no additional info helpful for classification. A secondary output label titled 'activity' was also removed because our model is not intended to classify different types of attack packets. Furthermore, all numeric input features were normalized with z-score standardization so all numeric values have the same variance, helping the model train faster. The resulting dataset contained 199468 rows and 310 columns.

Due to interesting results from the initial random forest model 2, we re-approached data cleaning and feature selection for version two. This time, rows with a duration of zero remained in the dataset, giving the model more inputs to work with. Moreover, source and destination ports were left out of normalization as they are not continuous numeric values. The other numeric values were normalized with the min-max normalization instead of z-score.

Extra steps were taken in version two to reduce the complexity of the model due to concerns of over-fitting in version one. First, a Pearson correlation matrix with all input features was generated to identify which features had a high linear relationship with each other. This allowed us to remove redundant features that don't provide any information that

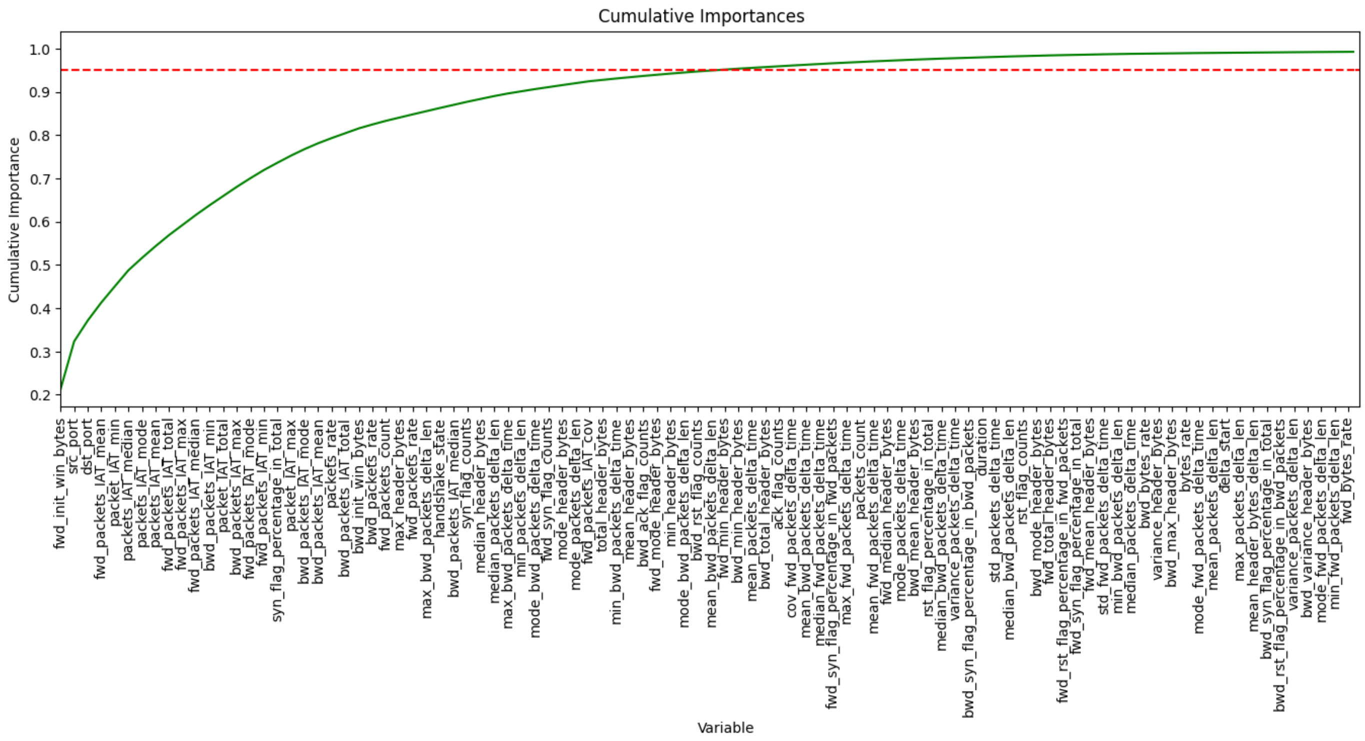

Fig. 1. Cumulative importance of features in the original dataset, the red line denotes 95% of total importance reached

wasn't already supplied by another feature. This reduced the 314 input features down to 83. Then, we ran a recursive feature eliminator using a random forest model as its estimator. The recursive feature eliminator was trained five times with 5-fold validation in case any folds contain a disproportionate amount of Benign or Attack packets. The compilation took nearly two hours running on an M2 Macbook Pro laptop and returned six features, 'src port', 'dst port', 'fwd init win bytes', 'packet IAT max', 'fwd packets IAT max', and 'bwd packets IAT max'. The resulting shape of the clean dataset was 504701 rows and 7 columns.

B. Neural Network

Two libraries were used to construct our Neural Network models - Tensorflow [7] and PyTorch [8]. We designed a sequential model in Keras, a linear stack of layers where each feeds into the next. The input layer comprised 128 neurons, leveraging a ReLU (Rectified Linear Unit) activation function to introduce non-linearity. The number of input features defined the input shape. The architecture included two hidden layers with 64 and 32 neurons, both using ReLU. The output layer contained a single neuron with a Sigmoid activation function to output probabilities for binary classification. The model was compiled using the Adam optimizer, binary cross-entropy loss, and accuracy as the evaluation metric.

The model was trained over 20 epochs with a batch size of 32 and validated on the test set after each epoch. Initial training runs achieved an accuracy of 0.9687 with a loss of 0.0844. Subsequent tuning improved the accuracy to 0.9731 with a slightly higher loss of 0.0871. Despite these results, challenges such as long training times and occasional spikes in loss/accuracy during epochs highlighted the need for further refinements.

The initial PyTorch models were implemented as Sequential models with repeating successive Linear layers with SeLU activation functions, with hidden layer widths equal to the size of the full dataset (314). Models evaluated had three, four, and eight hidden layers, and we also evaluated doubling the width of the hidden layers. These results are described more in evaluation. The Linear layer was selected as a default option [9] and SeLU was selected as the activation function due to its general strength. The model was built to use the Adam optimizer with binary cross-entropy loss. It was loaded onto an NVIDIA RTX 4090 for training and evaluation. As our task was a binary classification problem, the final layer used a sigmoid activation function.

The final neural network model was done in Tensorflow, since we found it significantly easier to work with while still achieving good results. We used 10 hidden layers since we found that greater depth led to better performance, and we could afford to train to that depth by loading this model onto the RTX 4090. The layers were Dense, with width equal to the input size which was the feature-reduced dataset of size 50. This model used SeLU activation functions for the same reason as the PyTorch models, but also added BatchNormalization and Dropout layers to improve generalization and reduce the risk of overfitting. Again, the final layer used a sigmoid activation function and the model used an Adam optimizer with binary cross-entropy loss function.

III. Experimental Evaluation

A. Random Forest

The first model used the gini information gain loss function with balanced class weights. It was training on a simple 80-20 train test split. A confusion matrix and ROC curve were generated to measure the results. However, six variations of the second model were trained to identify the best combination of class weights and loss functions. Class weight variations include balanced, 5 to 1 in favor of attack, and 10 to 1 in favor of attack. Loss functions include entropy and gini information gain. Cross-validation with 5 folds was done on a reshuffled dataset to maintain relatively consistent class distribution and prevent over-fitting. Each fold generated its own confusion matrix as well as accuracy and AUC score. TPR and FPR rates were calculated using the results from the confusion matrix.

B. Neural Network

For the PyTorch models, the dataset was split into 90% training, 5% validation, 5% testing. All six variants were evaluated using 5 split stratified KFolds and their average accuracies compared. The models were evaluated not just for final accuracy but for improvement in validation accuracy and loss to identify the metrics to use for the final neural network model.

The final model was first evaluated using an 80% training, 10% validation, 10% testing split on the feature-reduced dataset. Additionally, source port and destination port were dropped even though Random Forest ranked them as important because we found it did not significantly alter accuracy, and we believe it is better for generality not to rely on port numbers. This model was evaluated for its final accuracy on the test set as well as on 10 specific examples (5 benign, 5 attack) held out of the overall dataset. Additionally, its misclassification rate for both attacks and benign traffic was explored. Due to a noticeably high misclassification of attacks yet with good overall accuracy, as shown in results, it was trained again but with a random validation set consisting of 50% attacks and 50% benign flows. This was to force the model to not compensate for poor attack classification by correctly classifying the far more prevalent benign flows. This was then also evaluated on the same metrics.

IV. Results

A. Random Forest

For the first Random Forest model, we achieved an accuracy score of 88% and an AUC of 0.85. The confusion matrix shows that out of the 1571 misclassified packets, 1549 of them were false negatives (Figure 2). This means the model has a natural bias toward benign packets and is more likely to let a malicious network packet through than stop a benign one. However, cybersecurity agencies would much rather classify benign packets as attacks than the alternative to play it safe.

The remaining Random Forest models performed much better in comparison, although using a gini or entropy loss function did not affect its scores in any significant way. With balanced class weights, the average accuracy score was 97%

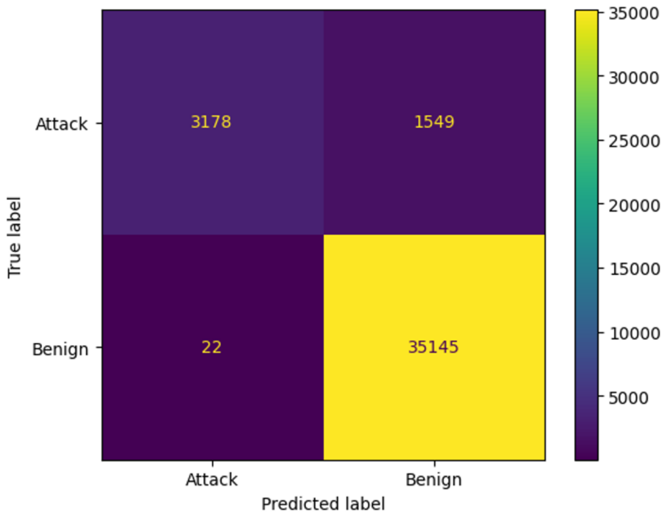

Fig. 2. Confusion matrix for the 1st Random Forest model

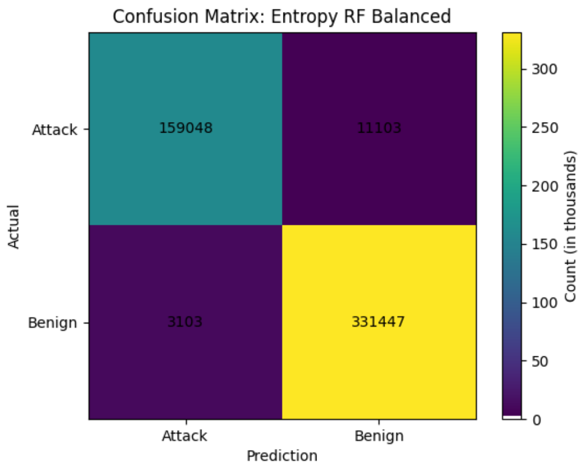

Fig. 3. Confusion matrix for the 2nd Random Forest model with an entropy loss function and balanced class weights

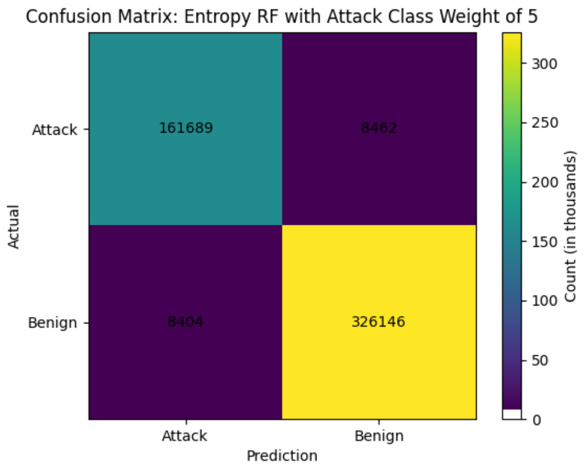

Fig. 4. Confusion matrix for the 2nd Random Forest model with an entropy loss function and Attack class weight of 5

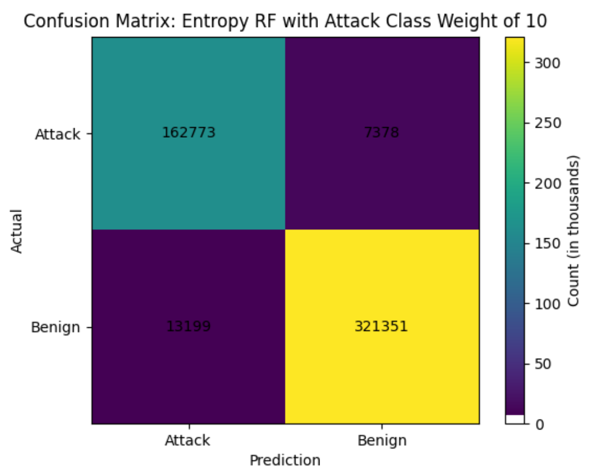

Fig. 5. Confusion matrix for the 2nd Random Forest model with an entropy loss function and Attack class weight of 10

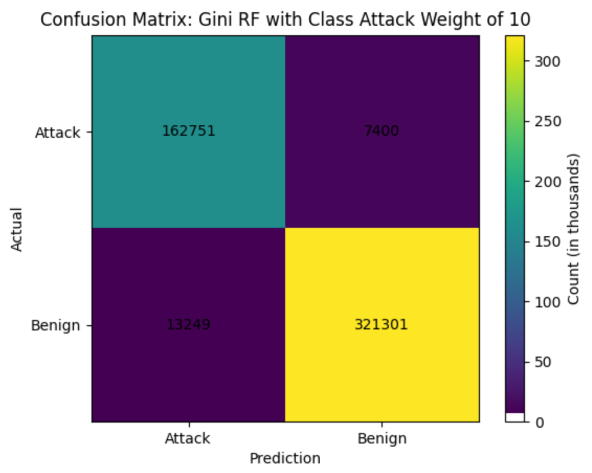

Fig. 8. Confusion matrix for the 2nd Random Forest model with a gini loss function and Attack class weight of 10

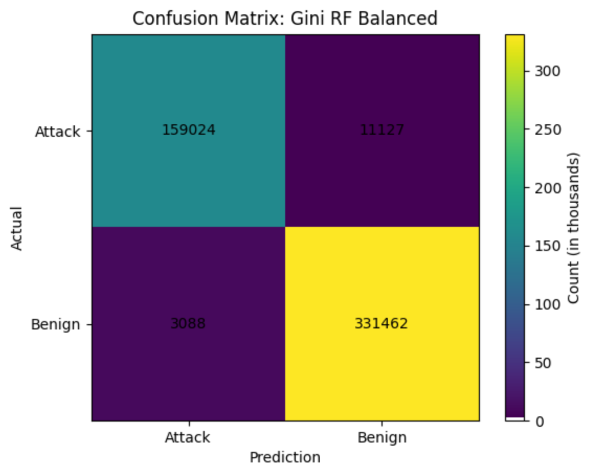

Fig. 6. Confusion matrix for the 2nd Random Forest model with a gini loss function and balanced class weights

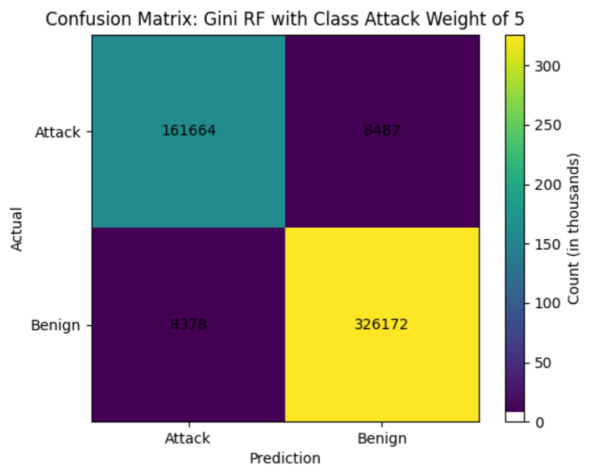

Fig. 7. Confusion matrix for the 2nd Random Forest model with a gini loss function and Attack class weight of 5

and the average AUC was 0.96, a significant improvement to the first model. However, similar to the first model, the confusion matrices show a high percentage of misclassified packets as false negatives (Figures 3 6). Class weights of 5 and 10 to the attack class were added to the model in an attempt to reduce bias with some success, reducing the number of false negatives by 30% while accuracy fell only 1% (Figures 4 5 7 8).

B. Neural Network

For the PyTorch models, we found that increasing depth led to a slight decrease in average accuracy but a larger decrease in standard deviation. Increasing the width had a similar pattern but a far more noticeable decrease in accuracy. We thus opted to use the narrow yet deep model architecture, since with the reduced standard deviation the model was more likely to perform well across the full dataset and generalize better. Further, an analysis of accuracy (Figure 9 and loss per epoch showed the narrow models performing better over time than the wide ones, cementing our choice.

For the final model, the initial assessment showed a strong accuracy of 96.3% on the test dataset. However, this masked a significant misclassification of attack flows as benign. Since the dataset after cleaning was heavily skewed benign, it could compensate for this misclassification by correctly classifying benign flows, and it was highly accurate in this regard 10. Of the 10 held-out cases, it correctly identified the 5 benign flows, confidently with the worst as 95% benign, yet only accurately identified the 2 of the 5 attacks, with 2 classified as 100% benign.

When forcing the model to use a 50-50 attack-benign validation set, even though it was smaller, it became significantly more balanced in its misclassification when evaluated on a 50-50 test set 11 but took a significant accuracy hit, falling to around 90%. It now correctly identified 4 of the 5 held-out attacks with high confidence. This was done as further training to the same trained model from before. It is possible that in

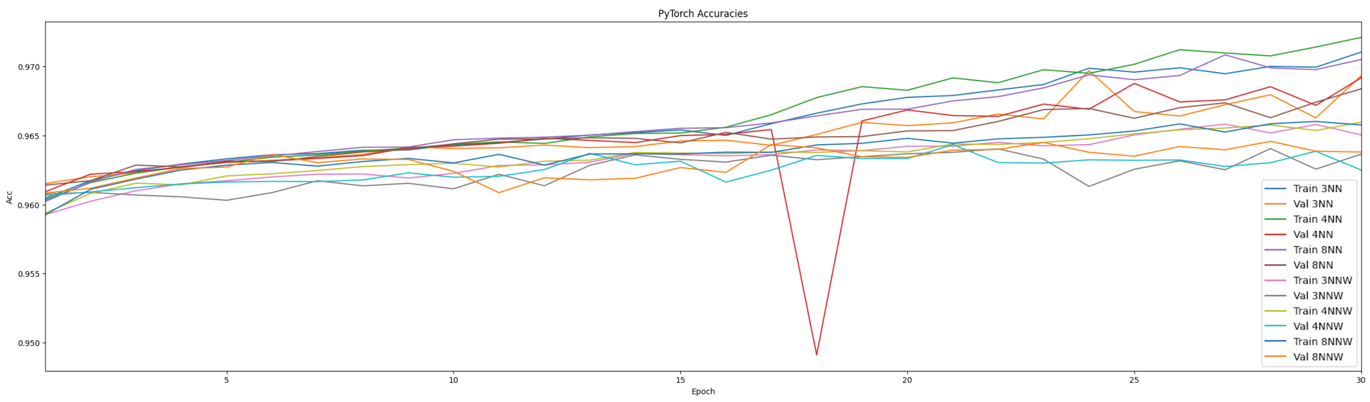

Fig. 9. Training and validation accuracy of the various PyTorch models per epoch

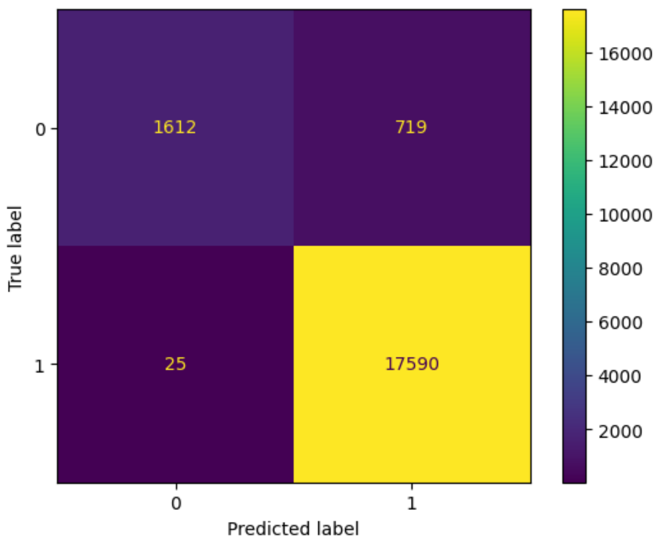

Fig. 10. Confusion matrix of the final Neural Network model applied to a 10% test sample of the cleaned dataset.

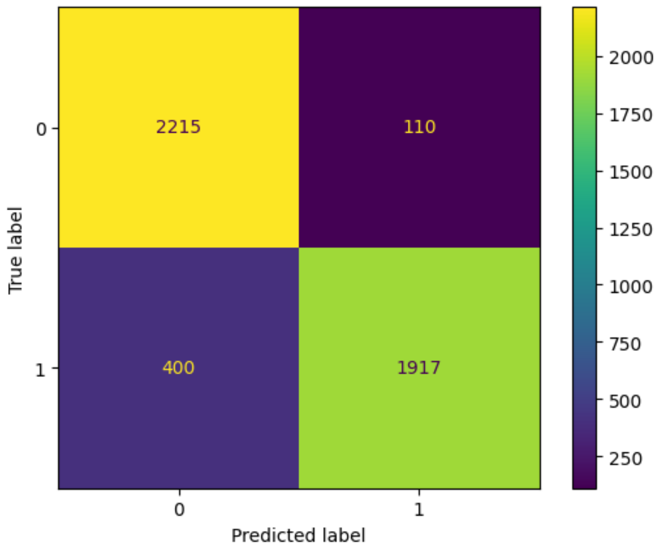

Fig. 11. Confusion matrix of the final Neural Network model applied to a 50-50 Attack-Benign test sample.

the future, training initially with a more equal dataset could produce better results.

V. Related Work

Intrusion detection and prevention are active and well-established fields of research and industries. Intrusion Detection (IDS) and Intrusion Prevention Systems (IPS) are available on the market which attempt to secure networks against attacks. As described in I however, this is a constantly evolving game. Traditional defenses must play catch-up to new attacks, a weakness that has driven research into AI and ML methods.

Many published methods rely upon the outdated KDD’99 dataset [10]–[12]. As described in [13] previous models also have significant issues of computation infeasibility or false detection rates. Their model BizSCOP represents the current state-of-the-art, being trained on recent datasets such as CIC-IDS-2017 [14]. It achieves a significant accuracy of 99.8% with similarly impressive other metrics, alongside low computational load.

However, the CIC-IDS-2017 dataset, as well as other recent datasets designed to supplant KDD’99, have been found to possess critical flaws [15] which brings into question the true accuracy of the scores. However, the techniques to produce a feasible model are still important, including combining models synergistically.

Wei et. al [16] utilize federated learning techniques and evaluate their model on the same BCC-cPacket-Cloud-DDoS2024 dataset we use. Their technique also achieves appreciable accuracy, but federated learning techniques may be a barrier for smaller users in terms of feasibility and scale.

Ultimately, no other published work uses this new dataset. Our work is not as accurate as related work, but still achieves appreciable results for the relatively few resources used, namely a single GPU.

VI. Future Work

More work remains to be done in the field of dataset creation. We agree with others in the field [3], [4] that a key factor holding the field back is a lack of high-quality datasets, both labeled and unlabeled. When designing labeled sets, we believe that authors should ensure that attacks are highly represented. Further, when models are trained on these sets, researchers should ensure that the models cannot compensate for poor attack classification by correctly classifying benign flows.

In terms of model design, future work can involve further investigation into Neural Network hyper parameters and architecture. We believe the neural network can improve significantly given significantly more data, and using other models such as Random Forest to prune the feature space helps make this approach feasible.

Finally, we believe it is an open research question on the best way to integrate these models into network architectures. With the growth of Software-Defined Networking, we believe an approach using a model that can take in network data and output commands to an SDN controller to handle attacks in real-time is viable, and should be explored.

VII. Conclusion

We have explored the potential of machine learning and deep learning techniques for DDoS attack detection using the BCC-cPacket-Cloud-DDoS-2024 dataset. By employing Random Forest and Neural Network models, we found that both approaches have strengths and limitations in handling the complexities of modern internet traffic. The Random Forest model excelled at feature selection, effectively narrowing the dataset from 315 features to a concise subset that informed the subsequent Neural Network training. Despite achieving high accuracy and AUC scores, the Random Forest models exhibited a bias towards classifying traffic as benign, a limitation in cybersecurity where false negatives (missed attacks) are particularly costly. The Neural Network models demonstrated greater potential for classification accuracy, especially when trained on the reduced feature set. However, they initially struggled with misclassifying attack traffic due to the imbalanced nature of the dataset. By introducing balanced training and validation strategies, the model's attack detection improved significantly, at the expense of overall accuracy. Overall, our results show that combining these methodologies—using Random Forest for feature reduction and Neural Networks for classification—can offer a promising approach to identifying a DDoS attack.

References

[1] 2024. [Online]. Available: https://assets.ctfassets.net/slt3lc6tev37/5naLIMtcpQ1QuuFNKFDyp9/7ba5f021de7118016ffd766ef0b2388d/BDES-5907_State-of-App-Security-2024.pdf

[2] S. Morgan, "Cybercrime to cost the world $9.5 trillion usd annually in 2024," 2023. [Online]. Available: https://cybersecurityventures.com/cybercrime-damage-costs-10-trillion-by-2025/

[3] M. Shafi, A. H. Lashkari, V. Rodriguez, and R. Nevo, "Toward generating a new cloud-based distributed denial of service (ddos) dataset and cloud intrusion traffic characterization," *Information*, vol. 15, no. 4, 2024. [Online]. Available: https://www.mdpi.com/2078-2489/15/4/195

[4] R. Boutaba, M. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, and M. Caicedo, "A comprehensive survey on machine learning for networking: Evolution, applications and research opportunities," *Journal of Internet Services and Applications*, vol. 9, 05 2018.

[5] L. ML, "Kdd cup 1999 data," 1999. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[6] L. D'hooge, "Bccc-cpacket-cloud-ddos-2024," 2024. [Online]. Available: https://www.kaggle.com/datasets/dhoogla/bccc-cpacket-cloud-ddos-2024

[7] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[8] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," 2019. [Online]. Available: https://arxiv.org/abs/1912.01703

[9] A. Tam, "Building a binary classification model in pytorch," 2023. [Online]. Available: https://machinelearningmastery.com/building-a-binary-classification-model-in-pytorch/

[10] W. Wang, X. Du, D. Shan, R. Qin, and N. Wang, "Cloud intrusion detection method based on stacked contractive auto-encoder and support vector machine," *IEEE Transactions on Cloud Computing*, vol. 10, no. 3, pp. 1634–1646, 2022.

[11] A. Aldallal and F. Alisa, "Effective intrusion detection system to secure data in cloud using machine learning," *Symmetry*, vol. 13, no. 12, 2021. [Online]. Available: https://www.mdpi.com/2073-8994/13/12/2306

[12] W. Elmasry, A. Akbulut, and A. H. Zaim, "A design of an integrated cloud-based intrusion detection system with third party cloud service," *Open Computer Science*, vol. 11, no. 1, pp. 365–379, 2021. [Online]. Available: https://doi.org/10.1515/comp-2020-0214

[13] R. Menezes, P. Jayarin, and A. Sekar, "A bizarre synthesized cascaded optimized predictor (bizscop) model for enhancing security in cloud systems," *Journal of Cloud Computing*, vol. 13, 05 2024.

[14] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *International Conference on Information Systems Security and Privacy*, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:4707749

[15] L. Liu, G. Engelen, T. Lynar, D. Essam, and W. Joosen, "Error prevalence in nids datasets: A case study on cic-ids-2017 and cse-cic-ids-2018," in *2022 IEEE Conference on Communications and Network Security (CNS)*, 2022, pp. 254–262.

[16] Z. Wei, J. Wang, Z. Zhao, and K. Shi, "Toward data efficient anomaly detection in heterogeneous edge–cloud environments using clustered federated learning," *Future Generation Computer Systems*, vol. 164, p. 107559, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X24005235