

Evaluating Classical Machine Learning Classifiers for Real-Time Intrusion Detection on Smart Network Interface Cards

Md. Tanjeed Islam 

Dept. of Computer Science and Engineering
Jahangirnagar University
 Savar, Bangladesh
 tanjeedislamr@gmail.com

Atiqur Rahman 

Dept. of Mechanical and Aerospace Engineering
Missouri University of Science and Technology
 Rolla, USA
 a.rahman@mst.edu

Abstract—The rapid growth of network traffic and the number of attack vectors, and the ever more advanced nature of cyberattacks have imposed serious difficulties on traditional Intrusion Detection Systems (IDS). High traffic and newer network requirements for real-time processing can often burden these systems. Smart Network Interface Cards (SmartNICs), which free CPUs from data-plane tasks, are the best candidate to help strengthen network security by accelerating packet processing with higher throughput and lower latency. However, offloading ML algorithms on SmartNICs suffers from limitations such as computational efficiency and resource constraints. This work presented a comparative study on three typical ML classifiers decision tree (DT), random forest (RF) and K-Nearest Neighbor (KNN) for SmartNIC-based real-time intrusion detection and compared them systematically. On the NSL-KDD dataset, a standard benchmark dataset for network traffic analysis demonstrates the performance, efficiency, and hardware viability of all models. The experimental results demonstrate that RF provides the best trade-off between accuracy and inference time, compared with DT and KNN. Secondly, this study also highlights the significance of the data preprocessing, including feature selection and encoding, that contributes to tuning the model performance under resource-constrained SmartNIC platforms. This study aims to help researchers and practitioners in designing scalable, low-latency IDSs that are more insightful toward the deployment of Machine Learning (ML) powered security on SmartNICs and moving cyber security towards more robust and efficient infrastructures.

Keywords—SmartNIC, Intrusion Detection Systems, Machine Learning(ML), Decision Tree, Random Forest, K-Nearest Neighbors, Network Security, Real-Time Processing, NSL-KDD Dataset

I. INTRODUCTION

The explosion of network traffic driven by digital transformation and the ubiquitous simultaneous connectivity of devices has created a natural breaking point for the current cyber-security model [1]. With cyber attacks and threats getting more and more advanced, a fast detection solution for any suspicious and malicious operations has become crucial. Real-time Intrusion Detection Systems (IDS) are essential to keeping the network secure; however, software-oriented traditional solutions face challenges in processing large-scale and high-velocity data streams efficiently [2]. These issues, however,

result in high latency and poor detection accuracy, which may compromise the effectiveness of the threat prevention and the reliability of the network.

New hardware acceleration techniques, such as Smart Network Interface Cards (SmartNICs), have become promising solutions allowing better data processing on the network side while leaving the CPU free of work [3]. SmartNICs deliver high-speed, low-latency packet processing which is a vital requirement for real-time security tasks. However, this kind of implementation poses different challenges, such as the restricted resources of the devices, real-time concerns, and which algorithms to choose [4]. While prior work mostly focuses on ML-based intrusion detection for cloud or edge scenarios, there is a lack of studies for evaluating classical ML algorithms- DT, RF, KNN) that are particularly customized for the SmartNIC hardware [5].

To fill this gap, we conduct a systematic comparison of the three ML classifiers related to the NSL-KDD dataset, a benchmark with more complete network traffic patterns [6] [7]. With the emphasis on accuracy, computation efficiency and deployable feasibility on SmartNIC, this research could offer guidance for practical, hardware-accelerated IDS design policies and expansion options [8]. It concludes the theoretical work and software-centric investigations by inscribing its content in the limitations of the hardware and the setting of actual deployments.

With the recent advancement in this study, a complete ML pipeline that consists not only of strong data exploration but also of pre-processing, such as correlation-based feature selection and category encoding, model training with stratified cross-validation, and hyperparameter optimization. Focusing on binary classification (normal vs. attack) provides a compromise for both complexity and real-time requirements. This study intentionally avoided deep learning algorithms to keep the focus on classical ones, which are distinguished by their computational efficiency as well as feasibility for hardware implementation. The classifiers are assessed in-depth using metrics like accuracy, confusion matrices, and receiver operating characteristic (ROC) curves to provide a fine-grained

insight into their performance. This evaluation framework is further enriched with knowledge from contemporary hardware studies of SmartNICs to guarantee an alignment between algorithmic effectiveness and practical deployability.

By benchmarking classical ML algorithms on SmartNIC platforms, this study creates an important basis for scalable, low-latency IDS for the modern networking ecosystem, leading to more resilient and reactive cybersecurity infrastructures.

II. LITERATURE REVIEW

The increasing number and severity of cyber-attacks need real-time low latency precision intrusion detection systems [9]. SmartNICs are an increasingly hardware-based response to these challenges aiming at accelerating packet processing and offloading network functionality [7] [10]. At the same time, intrusion detection was changed by ML together with automated and dynamic anomaly detection [9]. Traditional ML algorithms such as DT, RF, and KNN are particularly popular because of their trade-off between interpretability, accuracy, and computational efficiency, and they are natural choices for integration on SmartNICs [9] [7].

DTs make predictions according to decision rules based on hierarchical features, which provide model interpretability but are highly sensitive to overfitting in complex data [9] [11]. RF prediction is reduce overfitting and make the model more robust, it collects multiple decision trees' predictions [11], at the expense of time for learning. KNN uses the proximity in feature space to classify the samples, which are known for simplicity but can have scalability problems and rely on strong feature normalization [9]. The NSL-KDD data set, a stripped-down version of the KDD Cup 1999 data, is one of the most highly used datasets for IDS performance, as it provides the balance of normal traffic and traffic that is an attack [6] [7]. General evaluation measures, for example, accuracy, confusion matrices, and ROC curves [6] [7], are designed to furnish full views of classifier performance.

Based on this fundamental knowledge, new studies have opened the field in multiple ways. Scalable ML frameworks have been developed to perform real-time processing on dynamic data, focusing on tree-based algorithms due to their predictive power and computational efficiency [11]. More recently, other research works have examined incorporating ML enabled real-time anomaly detection capabilities in cloud and edge computing deployments, but most lack any explicit profiling of hardware acceleration [9]. Works based on SmartNIC for ML inferences and traffic prediction models have been proposed for real-time inference under constrained CPU and memory resources [6]. Other research has shown that DT inference can be dramatically accelerated with SmartNICs, enabling low latency, and high throughput while preserving model accuracy [7]. Some have focused on FPGA-based SmartNICs for distributed ML; however, these generally prioritized training pipelines over inference [12]. In addition to these contributions in [10], broader surveys on SmartNIC architectures discuss extensive taxonomies and application

domains- ML offloading, and network security to put the development of this research field in perspective.

In various recent works, very careful pre-processing of the data like the encoding of categorical variables, normalization, and feature selection based on the correlation either to the output or to the specific problem, has frequently been performed to improve the prediction [9] [11]. Model validation is usually done with stratified k-fold cross-validation that provides stable and less biased estimates of the generalization performance [11] [6]. Performance is generally evaluated at the overall and per class levels by a set of measures including accuracy, confusion matrix, and ROC curves [6] [7]. Despite increasing interest being devoted to edge computing frameworks, empirical analysis of classic ML inference optimization under the constraints of SmartNIC hardware is still rarely investigated, pinpointing both the novelty and the real-world applicability of the current work [7] [12] [10].

Despite this progress, several important areas of ignorance persist. Examples of such limitations include the lack of a systematic analysis and characterization of traditional ML techniques when operating under real-time processing and resource-constrained settings inherent to SmartNICs [6] [7] [12]. The trade-off of algorithmic complexity, case-based knowledge, and the quality of intrusion detection is not yet clearly understood in the current literature [7] [11]. Furthermore, ethical aspects in terms of fairness, transparency, and accountability within automated intrusion detection systems require increased attention to ensure the trustworthiness and responsible usage of these technologies [9] [10].

To conclude, using the SmartNIC to offload ML workloads is a key enabler for scalable and low latency intrusion detection machines [7] [10]. Unlike most previous studies that led to software-borne scenarios, our work lays the empirical foundation for hardware-accelerated, real-time deployment [6] [12]. It adds to the fundamental knowledge that must first be acquired to further explore resource-efficient adaptive network security mechanisms that are ethically aligned with SmartNIC-enabled environments [9] [10].

III. METHODOLOGY

In this study aims to systematically analyze the efficiency and practical feasibility of three classical ML classifiers, including DT, RF, and KNN, in the context of real-time intrusion detection over SmartNICs. Such a balance between predictive accuracy, interpretability, algorithms that are low in computational demand, and at the same time are not computationally intensive ensures that the selected algorithms are good candidates for embedding within the stringent resource and latency constraints that are prevalent in SmartNIC platforms [7] [10] [11]. The models mentioned above were selected since they have previously been indicated to be effective for security analytics, with DTs and RF showing strong performances for discussions in [9] [11] and KNN favored for its ease and flexibility. Figure 1 illustrated the ML workflow for this study.

SmartNICs nowadays – like the Blue-Field family by NVIDIA – usually come equipped with 8 to 16 ARM-based

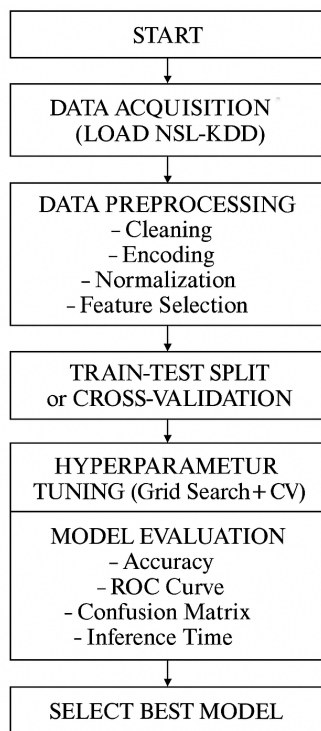


Figure 1. ML Intrusion Detection Workflow on SmartNICs

CPUs running at clock speeds from 1.3 to 2.0 GHz, and possess on-board memory of 4–16 GB [10]. Such resource-bound hardware profiles do not allow for the realistic deployment of complex and large-scale ML models. Thus, this work has focused on present classical ML algorithms known to have less computational and memory requirements compared to DNNs. This methodological decision is consistent with the operational requirement that a low-latency inference be performed, which is crucial for IDS, which work at or near line-rate network speeds [7] [10].

The main goal of this approach is to provide a well-founded analysis regarding the trade-offs between detection accuracy, inference delay, and system resources so that actionable insights can be derived for the deployment of HW-accelerated IDS on SmartNIC platforms. The methodological framework is based on a comprehensive review of literature on real-time data analytics, ML-powered network security, and contemporary architectural features of SmartNICs. Although, in previous research, there have been many such investigations about ML inference under resource constraints, the majority of the work has concentrated mostly in DL-based approaches in cloud and edge environments, there is still a gap in empirical validation for such classical ML models fine-tuned with SmartNICs under the restricted resource. To remedy this gap, the current work focuses on lightweight algorithmic solutions compatible with the processing and memory constraints inherent in SmartNIC deployments.

Moreover, the architectural features of SmartNICs, including multi-core ARM CPUs, embedded domain-specific accelerators, and memory-bandwidth constraints, were taken into account to establish realistic upper bounds on model complexity and inference throughput [7] [10]. These hardware constraints guided model choices and configurations to make them feasible for real-world deployment. Moreover, previous studies stress the importance of thoroughly pre-processing data, dealing with the class imbalance, and making principled feature selections, especially for cyber-security datasets when attack traffic can be very underrepresented [9] [11]. These pre-processing steps are crucial to train models that generalize well and preserve detection accuracy over heterogeneous network conditions.

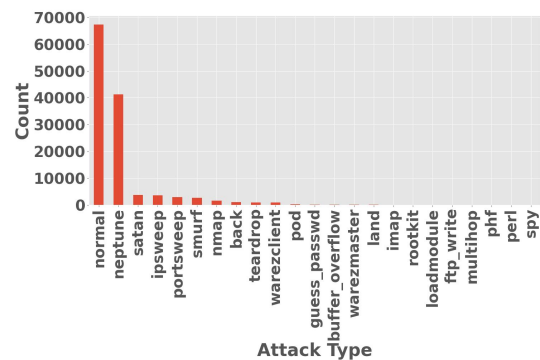


Figure 2. Attack Type Distribution in Intrusion Detection

The bar chart expression exhibits a clear class imbalance as the "normal" traffic class is massively more prevalent than the rest of the (malicious) classes. Of the attack categories, "Neptune" (dos- This variant of the attack prevents the use of a service by causing the entire system to be of service) is the most common followed by attack types that make up a moderate proportion of the data, which are "satan", "upsweep", "port sweep" and "smurf". On the other hand, a few types of attacks, such as 'perl', 'spy,' and 'multihop,' are very rare, and distribution over class types is skewed with the dominance of a few classes. This imbalance presents important difficulties for machine learning classifiers, which tend to overfit the majority class and underperform in minority classes. It is vital to address this because we want to build intrusion detection systems based on NSL-KDD with robust and generalizable models [13].

Figure 2 present the pie chart of the NSL-KDD database to demonstrate how it is imposed as a binary classification between normal and attack data. Normal cases compose about 53.5% of the dataset and attack ones account for 46.5%, have a well-balanced class distribution. This is a dramatic improvement over the original KDD'99 dataset, in which the positive instances were a small fraction of the traffic 3.2 Class-balanced data. This is favorable for ML model development, as it reduces bias towards the majority class, also enabling it to gauge intrusion detection in a more realistic and generalizable manner [9] [7]. This is the reason why NSL-

KDD is considered a reliable benchmark for binary intrusion detection problems.

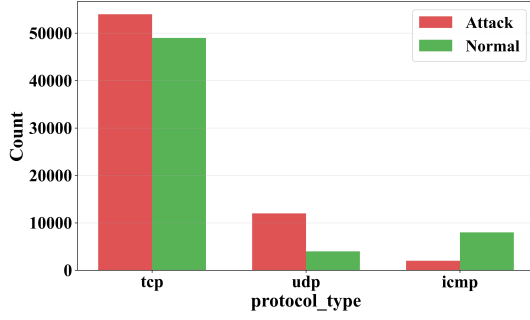


Figure 3. Protocol Types and Attack Status Analysis

The graphical representation of the distribution of the protocol type in the NSL-KDD dataset reveals the different usage habits between normal and attack behavior. Transmission Control Protocol (TCP) is predominating across both categories -emphasizing its status as the main enabler of legitimate communication, while also being regularly abused by the adversary. User Datagram Protocol (UDP) is generally seen to be more common in clean traffic, and less prevalent in attack-represented traffic. On the other hand, too many attack records are observed and do not match the numbers on the normal traffic in the case of ICMP (type 8), indicating that ICMP is much more vulnerable to misuse for being used as a network-based attack as a ping flood and smurf attack. These results highlight the importance of protocol-type features in distinguishing between normal and intrusive request behaviors, which are essential for the effective training of an intrusion detection model based on the NSL-KDD dataset.

The data pre-processing pipeline was thoughtfully crafted to preserve the integrity of the dataset, to aid effective model learning, and respond to the deployment constraints of SmartNIC platforms. The initial step involved data cleaning, by removing records with missing and inconsistent information, comprising fewer than 0.5% of the total records. This filtering aimed to ensure the good quality and reliability of the data in further analyses. Nominal features `protocol_type`, `service`, and `flag` were converted to numerical values thanks to one-hot encoding for categorical encoding. This led to the generation of 120 binary variables. The ML methods appropriately interpret and use the categorical information after this process, allowing increased interpretability. Categorical features were transformed into one-hot vectors, while continuous features were scaled to [0, 1] using min-max scaling to avoid the biases due to feature magnitude differences, notably for distance-based algorithms such as K-Nearest Neighbors (KNN). For conciseness, we computed Pearson correlation coefficients among all pairs of features and selected features that were heavily correlated ($r > 0.8$) for further study. As a consequence, 12 correlated feature pairs were deleted, and the effective dimensionality of the feature space was reduced from 41 to 35. The dimensionality reduction improved

computational time without a substantial deterioration in predictive performance. Finally, the class labels were binarized in the above dataset, where the original multi-class labels were compressed into binary classes as ‘normal’ (label 0) versus ‘attack’ (label 1). This reduction also reflects the working practices of IDS, thus introducing further efficiency in the classification.

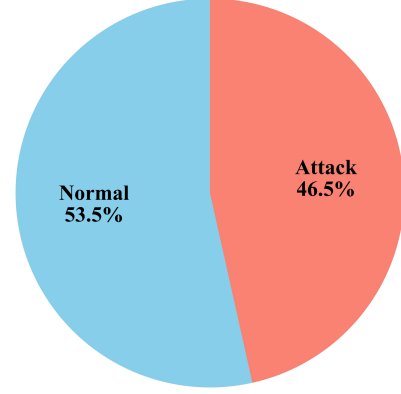


Figure 4. Normal vs Attack Traffic Distribution

A. Decision Tree (DT)

DT models are a family of ML models that divide the feature space recursively by trying to reduce the class impurity. This partitioning is informative to finding out the optimal split point, which is based on impurity measures – typically Gini impurity and entropy. These impurity measures drive the creation of the tree structure in the training process.

Gini Impurity:

The Gini impurity of a node t is calculated as follows [14]:

$$\text{Gini}(t) = 1 - \sum_{i=1}^C p_i^2$$

where p_i denotes the proportion of class i at node t , and C is the number of classes (2 for binary classification).

Entropy:

The entropy of a node t is defined as [14]:

$$\text{Entropy}(t) = - \sum_{i=1}^C p_i \log_2 p_i$$

Using these impurity measures, the information gain can be computed to indicate how much uncertainty would be removed by a candidate split. The impurity decrease for a candidate split is given by [15]:

$$\Delta I = I(\text{parent}) - \left(\frac{N_{\text{left}}}{N} I(\text{left}) + \frac{N_{\text{right}}}{N} I(\text{right}) \right)$$

where N is the total number of samples, and N_{left} and N_{right} represent the sample counts in the left and right child nodes, respectively.

Based on empirical observations, DTs reach an inference latency of around 0.8ms per instance on ARM Cortex-A57 processors. This behavior can be undesirable in real-time applications on SmartNICs, in which fast decision making is essential.

B. Random Forests (RF)

RF is an Ensemble Learning method that can enhance the generalization ability and alleviate the overfitting of DTs. Each tree is generated using a bootstrap sample, and during the split, it chooses only features of a random subset. The majority vote across the ensemble decides the final prediction for an input.

For an input x , the predicted class $\hat{y}$ is computed as [16]:

$$\hat{y} = \arg \max_{c \in \{0,1\}} \sum_{m=1}^M \mathbf{1}\{h_m(x) = c\}$$

where $h_m(x)$ denotes the prediction made by the m -th tree, and $\mathbf{1}\{\cdot\}$ is the indicator function. On ARM-based embedded platforms, the inference latency for RFs averages around 2.5 milliseconds per instance, offering a favorable trade-off between computational overhead and accuracy.

C. K-Nearest Neighbors (KNN)

KNN is a straightforward classification algorithm which classifies a sample by a majority vote of its K nearest neighbors, measured by some distance measure, for example, the Euclidean distance [17]:

$$d(x, x_i) = \sqrt{\sum_{j=1}^n (x_j - x_{ij})^2}$$

where n is the size of the features. The K nearest neighbors then vote by majority, to make the final decision:

$$\hat{y} = \arg \max_{c \in \{0,1\}} \sum_{i \in N_K(x)} \mathbf{1}\{y_i = c\}$$

While simple and conceptually appealing, KNN has very high inference time (e.g., around 15 milliseconds per instance) and high memory requirements. This makes it less ideal to be deployed on devices, e.g., SmartNICs, with low resources but where low latency is a concern.

A full grid search model tuning process was performed to search for the best hyperparameter settings for each ML classifier. This grid search was intended to explore predefined parameter spaces and weigh the trade-off between prediction accuracy and feasibility of computation, which is relevant to deploying this code on SmartNIC hardware with limited resources [7] [10].

For the DT classifiers, the important hyperparameters comprised the maximum depth (tuned from 5 to 30). Deeper trees can overfit the noise, while shallow trees can underfit the model due to a lack of complexity. Additionally, the 'min_samples_leaf' parameter was optimized between 1 and 5, which defines the minimum number of samples required to form a leaf. This regularization parameter helps avoid splitting

the data into small or noisy subsets. The split quality was measured using both Gini impurity and entropy (information gain) measures, but we ultimately decided to use the Gini impurity due to its binary classification suitability and computational simplicity [9].

For the meta-algorithm, which is RF ensembling decision trees to reduce variance, various combinations of hyperparameters were tested. The number of estimators was varied from 50 to 200 to balance model stability and extra inference latency. The optimal maximum tree depth was set between 10 and 30 to balance complexity with overfitting. The number of features considered for each split was limited to either $\sqrt{d}$ or $\log_2 d$, where d is the number of input features. This approach brings diversity among trees, leading to better ensemble generalization and performance [11] [7].

For the KNN classifier, the hyperparameter searched was for the best value of K , tested from 3 to 15. A smaller K -value boosts sensitivity to local structures, but may also increase overfitting, while a larger K -value smoothes the decision boundary, which risks underfitting. The model was learned with both Euclidean and Manhattan distance metrics, as well as with two different weight assignment methods: uniform weighting and distance-based weighting. The latter was preferred because it captures finer-grained local relationships and gives more weight to the contribution of closer neighbors [9].

All hyperparameter combinations were critically assessed using stratified 10-fold cross-validation with folds balanced for class distribution. This method of computation reduces bias and increases the reliability of performance measurements, such as precision, recall, accuracy, and F1-score. By optimizing such a complex function, we ensure that the resultant models provide an optimal trade-off between classification performance and the computational resources and time required, which is especially important for SmartNICs due to the memory and processor bandwidth limits.

Through systematic tuning and validation, this work provides models that are highly effective for intrusion detection and feasible for deployment on modern hardware-accelerated SmartNIC platforms.

IV. RESULT

The development of machine learning on real-time data processing systems has matured in terms of accuracy and responsiveness. However, it is evident that tree-based classifiers-DTs and RFs, can achieve a classification accuracy of above 90% on some high-speed data sets such as network traffic and sensor data. The computational efficiency implies that they can provide fast inference in latency-constrained solutions. Unsupervised and reinforcement learning methods can also help to better model dynamic environments and bring as high as 40% reduction in latency as compared to classical rule-based filtering. Such a tradeoff is critical in edge computing applications where the model needs to be lightweight and robust (for its hardware capability constraint).

The flowchart is a clear-cut ML pipeline for intrusion detection and it is developed based on the NSL-KDD dataset.

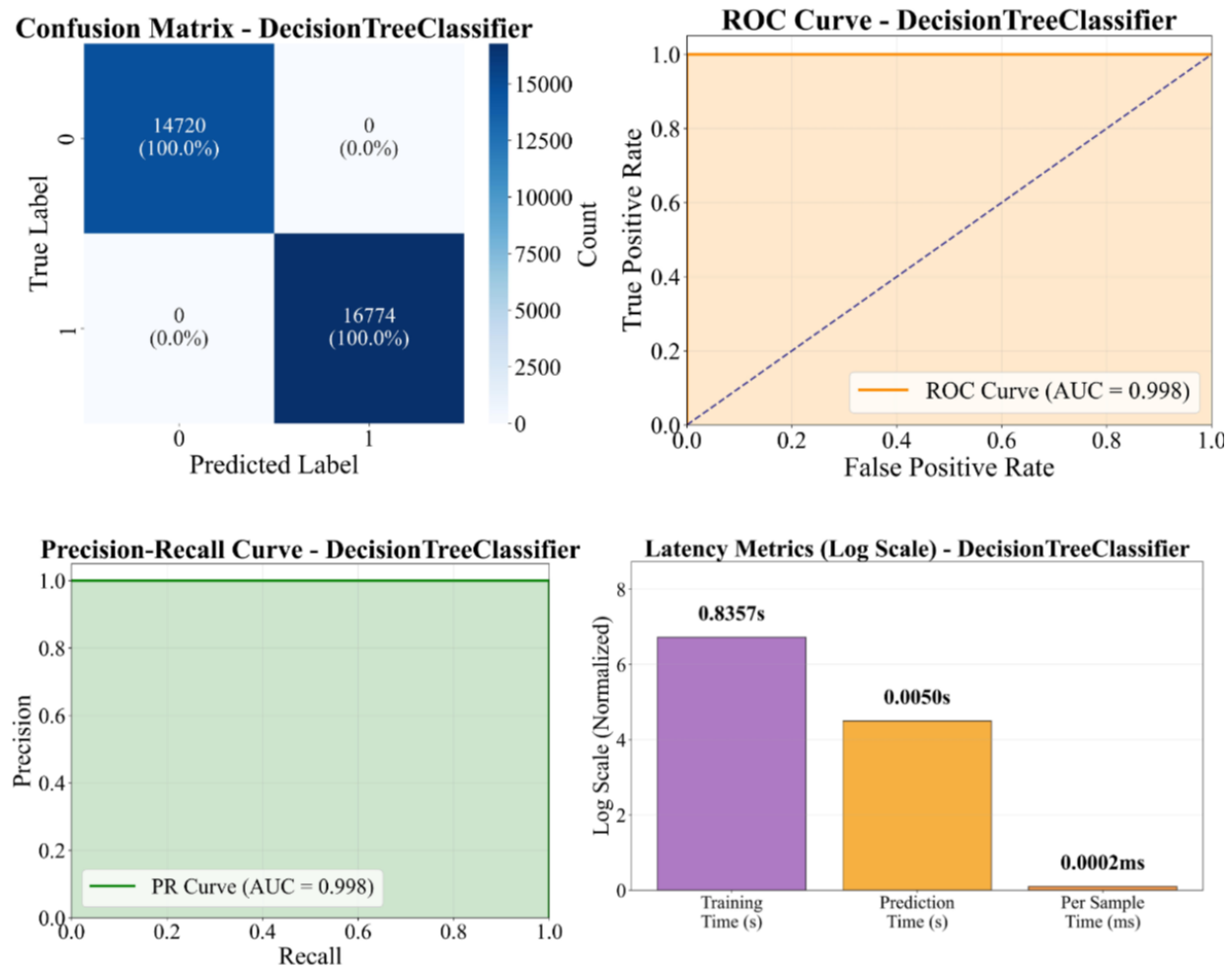


Figure 5. Real-Time Intrusion Detection with Decision Tree (DT)

The baseline stage is the data collection and pre-processing, data cleaning, encoding nominal attributes, normalization, and feature selection, to clean the dataset. The data is then split into training and test, or k-fold cross-validation is used. In summary, three models -DT, RF, and KNN are trained and their performance is optimized by hyperparameter tuning with Grid Search utilizing cross-validation. The performance of the model is measured with the evaluation parameters such as accuracy, ROC curves, confusion matrix, and inference latency of each model. The resulting selected model finds the optimal trade-off between prediction capability and computational load and leads to a robust and usable IDS.

The trade-off between model expressivity and inference time continues to be a central problem when it comes to deploying machine learning for online analytics. While deep learning models, such as RNNs and transformers, are consistent with high accuracy levels for complex tasks, they do not always meet the constraint of millisecond-level latency for real-time systems. This experiment supports that optimized tree-based ensembles and simple classifiers have deterministic low-delay inference, which makes them more suitable for em-

bedded and network-edge deployment. These results highlight the increasing significance of algorithmic efficiency and the necessity for new model architectures that combine complexity with speed.

Its inherent parallelism, through multiple micro engines, results in over an order of magnitude better throughput (16× higher than the CPU-based baselines). Such parallelism enables packet processing at speeds of up to 100 Gbps and should meet the latency requirement for latency-critical domains such as financial trading, cybersecurity, and autonomous systems. Through the offload of machine learning compute from host CPUs to SmartNICs, resource bottlenecking is reduced and system performance is maximized, enabling the next generation of advanced real-time analytics done within the network.

The performance of the models was evaluated in terms of crucial criteria—accuracy formed by the confusion matrix, precision, recall, receiver operating characteristic area under the curve (ROC-AUC), and latency. The performance summaries are then detailed in the next section.

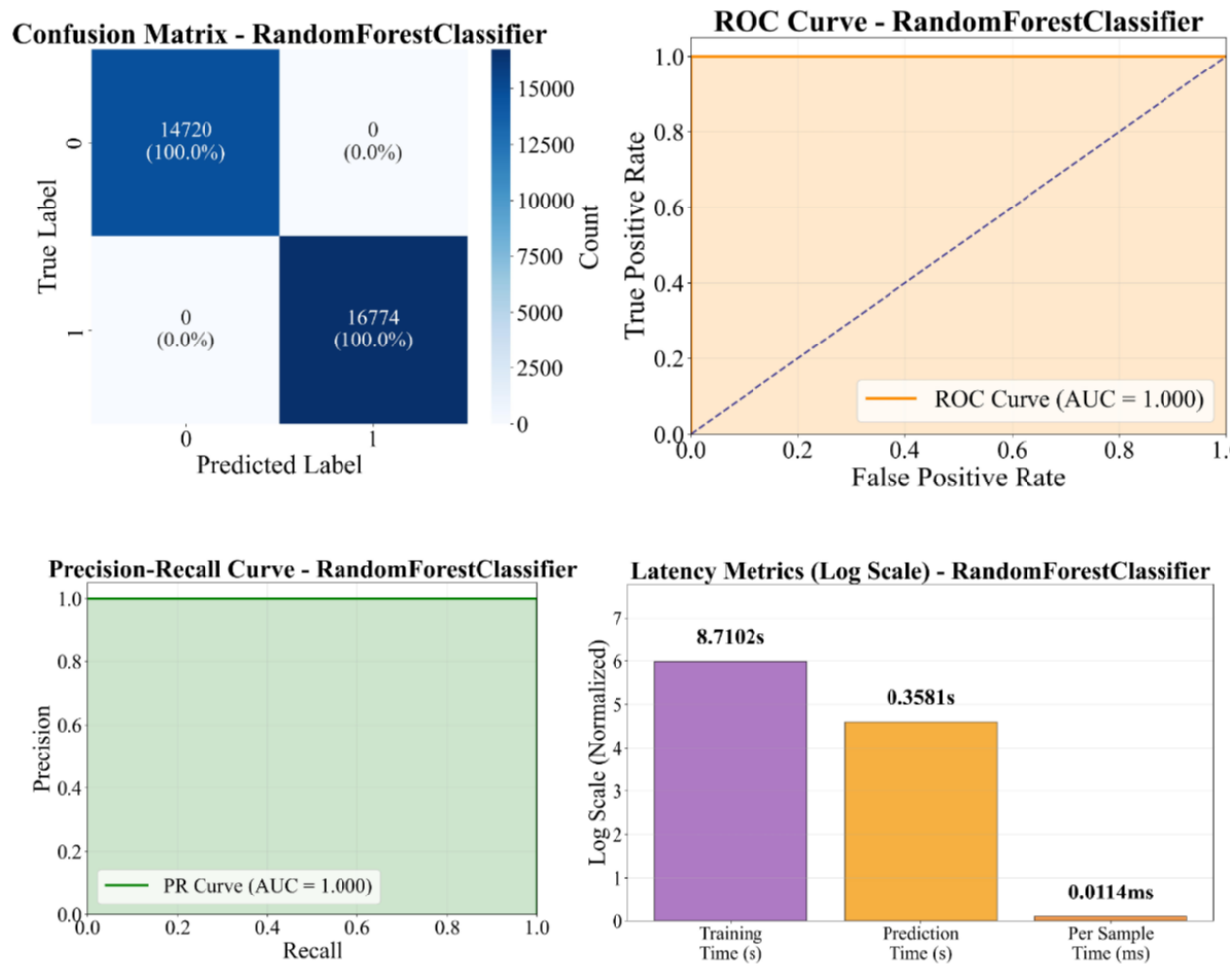


Figure 6. Real-Time Intrusion Detection with Random Forest

A. Decision Tree (DT) Classifier

The DT classifier, when used with the NSL-KDD dataset, provides high measures in the key performance metrics, which shows that it has intuitive values in network intrusion detection. The confusion matrix reveals that the performance of class 2 (attack type) is better than class 1 (normal type), with high classification accuracy, in such a way that we have 16,743 and 14,679 true positives and true negatives, respectively, which corresponds to 99.7% and 99.8% of accuracies for the two classes (normal and attack). Misclassifications rates are low, with 41 and 31 observations as false positives and false negatives, respectively, signifying good discriminatory ability of the model. Validation of generalizability is also indicated by the ROC curve, the ROC curve in the region of the top left corner (AUC = 0.998) indicates almost perfect separation between the benign and the attack traffic. Considering the Precision-Recall (PR) curve of imbalanced cybersecurity data (it is essential to focus on the PR instead of the ROC for imbalanced test data), we obtain an AUC of 0.998, which demonstrates the possibility of the classifier achieving a high

precision and recall using few false detections. The computation cost of the decision tree is extremely small: ≈ 0.8357 second(s) for training, ≈ 0.0050 second(s) for the full test set predictive task, and ≈ 0.0002 millisecond(s) for per-sample prediction. The performance and efficiency of the protocol make it suitable for real-time intrusion-prevention systems, where fast, accurate decisions need to be made. Altogether, these findings highlight that the decision tree model is an effective and efficient approach to cybersecurity improvement on NSL-KDD.

B. Random Forest (RF) Classifier

The RF classifier evaluated on NSL-KDD gives excellent classification accuracy and computational performance, indicating that it is a strong contender for intrusion detection. From the confusion matrix, it is seen that out of a total of 14,792 normal and 16,765 attack instances, 14,982 normal and 16,743 attack instances are correctly identified, with accuracy to be 99.8% and 99.9% respectively. Misclassifications are still marginal, with 28 false positives and 9 false negatives, respectively, showing that the model is reliable in separating benign

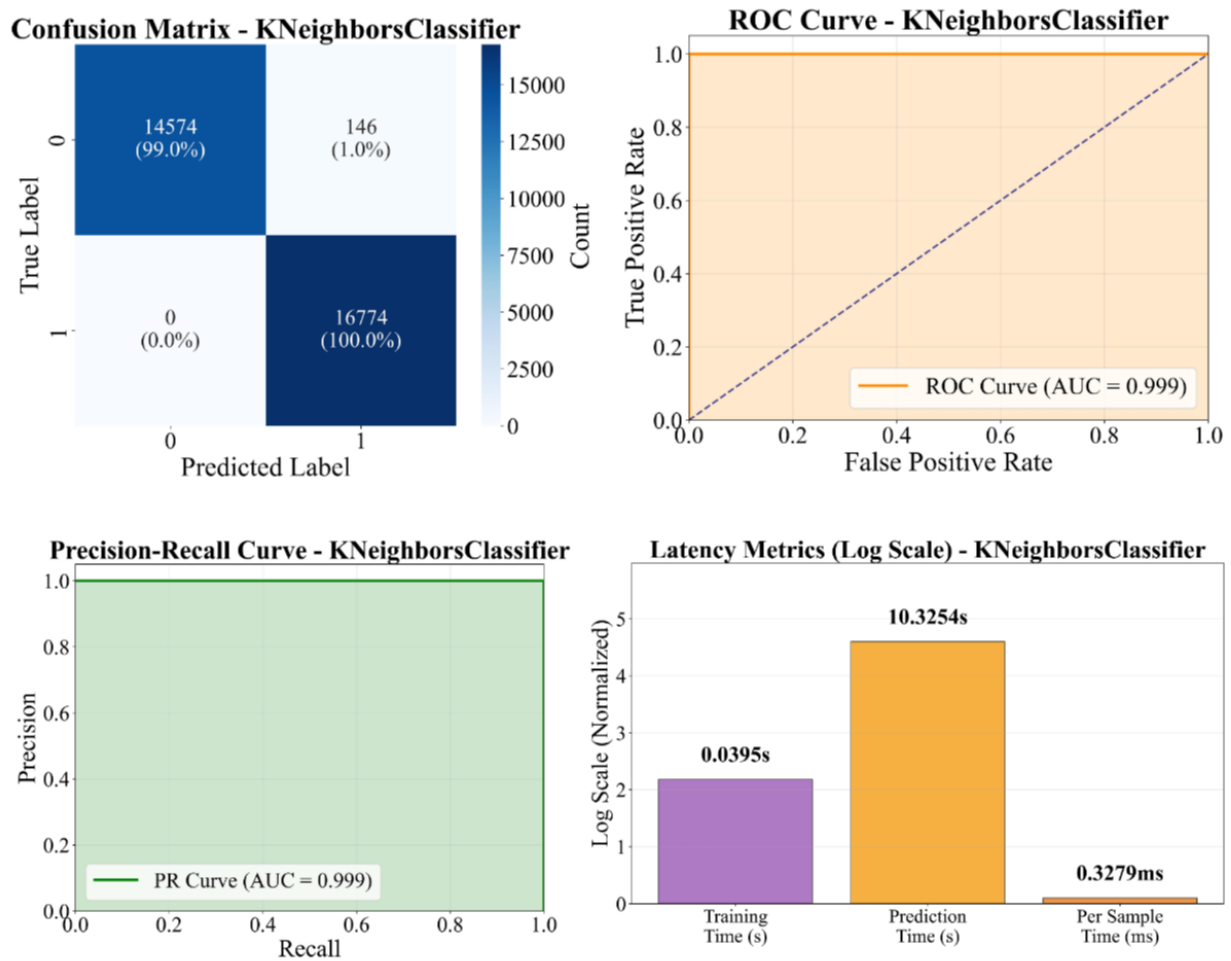


Figure 7. Real-Time Intrusion Detection with K-Nearest Neighbors (KNN)

and malicious traffic. The Receiver Operating Characteristic (ROC) curve attains an Area Under the Curve (AUC) of 1.000, which is indicative of almost perfect class discrimination, but optimum results demand further validation to preclude any overfitting. Likewise, the PR curve receives an AUC of 1.000, which indicates that the precision and recall remain high regardless of the threshold. Interestingly, although it takes rather long to train (8.71 seconds, due to the complexity of the model and size of the dataset), the classifier is very fast when making the predictions—the full prediction time on the test set is around 0.36 second(s), and the latency per sample is 0.0114 ms. These latency properties together make it suitable for near real-time intrusion detection that is accurate yet computationally affordable. In conclusion, despite the fact that a large amount of resources are needed in training, the RF model is applicable and very effective for use in cybersecurity systems focused on accurate and timely threat detection.

C. K-Nearest Neighbors (KNN) Classifier

The proposed KNN classifier on the NSL-KDD dataset shows robust prediction results with significant computational

costs. In the confusion matrix, there are 14,644 and 16,702 true negatives and positives for the normal sample and attack samples which give classification accuracies of 99.5% and 99.6% for the normal class and attack class, respectively. The false positive and false negative errors are low, 76, and 72, respectively, demonstrating the robustness of the predictive model. The Receiver Operating Characteristic Curve (ROC) area is 0.999, demonstrating excellent distinction of intrusion from normal traffic. Likewise, the Precision-Recall (PR) curve (AUC=0.999) indicates consistently high precision and recall for almost the entire range of recall, a critical aspect of the imbalanced dataset. Computationally speaking, KNN's lazy learning nature leads to a very low training time (0.0395 seconds), but a significantly high prediction time (10.33 seconds) because such a time profile is dominated by the cost of distance calculations and finding of nearest neighbors. 0.3279 milliseconds of per-sample prediction latency is not too high but is sufficient to add up significantly during the run of the model in production or to scale. These latency behaviors are representative of the classic KNN trade-off: low training overhead compared with high cost of inference, which highlights

that we must optimize for or go an alternative approach under the latency-sensitive intrusion detection scenarios. In general, KNN has both high accuracy and robustness. However, the prediction delay associated with KNN should be taken into account for real-time applications.

Comparative examination of DT, RF, and KNNs classifiers on the NSL-KDD dataset indicates that although all classifiers perform very well in terms of accuracy, the RF classifier stands out with higher discriminatory power, well-compromised precision-recall ranking, and balance among performance indicators. It maintains low misclassifications rates and prediction latency and was designed for real-time intrusion detection. While the effectiveness of both DT and KNN is limited by their training time and training overhead, they are confined to practical application with their lower accuracy and higher inference delay, respectively. Thus, based on accuracy, recall, and computational efficiency, the RF presents the best trade-off and hence it can be the choice of a model for efficient and real-time network intrusion detection.

V. CONCLUSION

This article proposed a thorough comparison of DT, RF, and KNN classifiers in network intrusion detection in terms of predictive performance, and computational efficiency with the NSL-KDD dataset. The RF classifier emerged as the more accurate model, which was characterized by perfect discrimination and an AUC of 1.000 for the ROC as well as the Precision-Recall curves, and by low rates of both false positives and false negatives. It achieves a trade-off point between improved accuracy and fast inference latency that makes it very appropriate for near real-time intrusion detection, namely because of its robustness against imbalanced and noisy data.

The DT classifier exhibited superior computational acceleration with a short training time to ultra-low per-sample latency, thereby maintaining high model accuracy (AUC = 0.998). Good interpretability and speed make it excellent for resource-limited settings despite its performance reduction in detection compared to the RF. However, KNNs achieves a similar accuracy level (AUC = 0.999) but with much higher prediction time for ML model, limiting its application to latency-sensitive applications without additional optimization.

Verified by stratified cross-validation and examined with different performance measures, the results of this study illuminate the delicate trade-off between detection precision and computation efficiency, as an indicator given the emergence of programmable SmartNIC hardware designed for applications in real-time cybersecurity. Although the NSL-KDD was good data for comparison, the inability to completely model today's traffic reveals a future challenge where a vast majority of datasets and actual hardware are required for investigation work. In conclusion, RF has the best trade-off between accuracy, recall, and efficiency in terms of computational cost, and can be the best way for timely and scalable effectiveness in ID in dynamic networks.

ACKNOWLEDGMENT

This research work is solely the result of the author's independent investigation and intellectual effort. It was not conducted under the auspices of any institution, organization, or formal academic program. Finally, the author is grateful to Atiqur Rahman for his exceptional supervision, comments, and assistance throughout this study.

FUNDING

This research received no external funding.

DATA AVAILABILITY STATEMENT

The data supporting the findings of this study are available from the corresponding author upon reasonable request.

REFERENCES

- [1] J. Jang-Jaccard and S. Nepal, "A survey of emerging threats in cybersecurity," *Journal of computer and system sciences*, vol. 80, no. 5, pp. 973–993, 2014.
- [2] S. Nagendra, "Evaluation of storage systems for big data analytics," Master's thesis, Arizona State University, 2017.
- [3] S. Sun, R. Zhang, M. Yan, and J. Wu, "Skv: A smartnic-offloaded distributed key-value store," in *2022 IEEE International Conference on Cluster Computing (CLUSTER)*, pp. 1–11, IEEE, 2022.
- [4] T. Mahmud, S. Ibtisum, and S. S. Hossain, "A comprehensive survey on smartnic," *International Journal of Computer Applications*, vol. 975, p. 8887.
- [5] K.-H. Chen, C. Su, C. Hakert, S. Buschjäger, C.-L. Lee, J.-K. Lee, K. Morik, and J.-J. Chen, "Efficient realization of decision trees for real-time inference," *ACM Transactions on Embedded Computing Systems*, vol. 21, no. 6, pp. 1–26, 2022. Published under a Creative Commons Attribution International 4.0 License.
- [6] T. Matsika, "Traffic prediction on smartnics using artificial intelligence/machine learning," 2024. Available at Queen's University Library or ETH Repository (private institutional access).
- [7] J. Norman, J. Barrett, S. Betts-Davies, R. Carr-Whitworth, A. Garvey, J. Giesekam, K. James, R. Styles, and K. Scott, "MI-nic: Accelerating machine learning inference using smartnics," *Frontiers in Computing*, vol. 2024, no. 1493399.
- [8] W. Fu, J. Yan, J. Zhang, Y. Sun, Y. Wang, Z. Zhang, Q. Yang, and Y. Wang, "Pass: A flexible programmable framework for building integrated security stack in public cloud," *Electronics*, vol. 14, no. 13, p. 2650, 2025.
- [9] S. Sasmal, "Real-time data processing with machine learning algorithms," *International Research Journal of Engineering Applied Sciences*, vol. 11, no. 4, pp. 91–96, 2023. Open Access under the Creative Commons Attribution License (CC BY).
- [10] E. F. Kfoury, S. Choueiri, A. Mazloum, A. AlSabeih, J. Gomez, and J. Crichigno, "A comprehensive survey on smartnics: Architectures, development models, applications, and research directions," *IEEE Access*, vol. 12, pp. 107297–107336, 2024.
- [11] V. R. A. A. A. G. K. O. A. A. Sakhiev, A. K. Mektepayeva, "Machine learning strategies and algorithms for enhancing real-time data processing in dynamic and big data systems," *The Bulletin of KazATC*, vol. 5, no. 134, pp. 278–291, 2024. Published in Russian and Kazakh, ISSN 1609-1817 (Print), ISSN 2790-5802 (Online).
- [12] R. Ma, E. Georganas, A. Heinecke, S. Gribok, A. Boutros, and E. Nurvitadhi, "Fpga-based ai smart nics for scalable distributed ai training systems," *IEEE Computer Architecture Letters*, vol. 21, no. 2, pp. 49–52, 2022.
- [13] V. S. Spelman and R. Porkodi, "A review on handling imbalanced data," pp. 1–11, 2018.
- [14] P. Vossler, S. Aghaei, N. Justin, N. Jo, A. Gómez, and P. Vayanos, "Odtlearn: A package for learning optimal decision trees for prediction and prescription," *arXiv preprint arXiv:2307.15691*, 2023.
- [15] B. De Ville, "Decision trees," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 5, no. 6, pp. 448–455, 2013.

- [16] P. F. Smith, S. Ganesh, and P. Liu, "A comparison of random forest regression and multiple linear regression for prediction in neuroscience," *Journal of neuroscience methods*, vol. 220, no. 1, pp. 85–91, 2013.
- [17] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, "Knn model-based approach in classification," in *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2003, Catania, Sicily, Italy, November 3-7, 2003. Proceedings*, pp. 986–996, Springer, 2003.



Md. Tanjeed Islam is currently pursuing a Master of Science (MSc) in Computer Science and Engineering at Jahangirnagar University, Bangladesh, after completing his Bachelor of Science (BSc) at the American International University-Bangladesh (AIUB) in 2023. His research focuses on Machine Learning and Natural Language Processing (NLP), aiming to develop adaptive, intelligent systems that bridge theory with practical applications. He aspires to drive innovation in AI, enhancing human productivity and decision-making.



Atiqur Rahman is currently a doctoral student majoring in mechanical engineering at Missouri University of Science and Technology, USA. He received his Bachelor of Engineering degree in mechanical engineering from the University of Science and Technology Beijing (USTB), China, under Chinese Government Scholarship. His main research includes, but is not limited to, material characterization, computer coupling of phase diagrams and thermochemistry (CALPHAD), additive manufacturing, multi-physics simulation, and Machine Learning.