Neurosymbolic AI Transfer Learning Improves Network Intrusion Detection

Huynh T. T. Tran*, Jacob Sander*, Achraf Cohen*, Brian Jalaian*, Nathaniel D. Bastian†
*University of West Florida, Pensacola, FL, USA
†United States Military Academy, West Point, NY, USA

***Abstract*—Transfer learning is commonly utilized in various fields such as computer vision, natural language processing, and medical imaging due to its impressive capability to address subtasks and work with different datasets. However, its application in cybersecurity has not been thoroughly explored. In this paper, we present an innovative neurosymbolic AI framework designed for network intrusion detection systems, which play a crucial role in combating malicious activities in cybersecurity. Our framework leverages transfer learning and uncertainty quantification. The findings indicate that transfer learning models, trained on large and well-structured datasets, outperform neural-based models that rely on smaller datasets, paving the way for a new era in cybersecurity solutions.**

***Index Terms*—Network Intrusion Detection Systems, Neurosymbolic AI, Uncertainty Quantification, Transfer Learning**

I. Introduction

Network Intrusion Detection Systems (NIDS) play a critical role in modern cybersecurity by monitoring network traffic to detect and classify malicious activities [1]. Traditional machine learning models, while effective in many scenarios, struggle to address the increasingly complex and diverse nature of cyber threats. To overcome these challenges, Neurosymbolic AI (NSAI) has emerged as a promising approach. NSAI combines the powerful data-processing capabilities of neural networks with the logical reasoning strengths of symbolic AI [2], [3]. The symbolic component, often implemented through models such as XGBoost, leverages rule-based decision-making, thereby enhancing the overall robustness of intrusion detection systems.

However, the limitation of current NSAI systems is their reliance on the availability of labeled data tailored to specific datasets or attack types. As new attacks and datasets emerge, building a new model for each scenario can be resource-intensive and inefficient. This challenge is particularly evident in NIDS, where different datasets and tasks may require tailored models, hindering scalability and adaptability.

Transfer learning can be considered a viable solution. Transfer learning leverages pre-trained models, adapting them to new tasks or domains with limited labeled data. While widely used in fields such as computer vision [4], natural language processing [5], [6], and medical imaging [7], transfer learning remains underexplored in the cybersecurity domain.

This work extends our Open Set Recognition with Deep Embedded Clustering for XGBoost and Uncertainty Quantification (ODXU) framework [2], by integrating transfer learning to enhance scalability and adaptability. Using the Canadian Institute for Cybersecurity Intrusion Detection Systems 2017 (CIC-IDS-2017) dataset [8], which is large and resembles real-world scenarios. This dataset contains approximately 7 million samples of both benign traffic and up-to-date common cyberattacks. Our contributions are:

- Developing a transfer learning framework to adapt pre-trained NSAI models to new cybersecurity datasets from the Army Cyber Institute, enabling improved adaptability and generalization in dynamic threat environments.
- Integrating and evaluating more uncertainty quantification techniques, such as SHAP value and information gain, to enhance the interpretability and reliability of model predictions in uncertain conditions.

The paper is organized as follows: Section II reviews related work, Section III outlines the ODXU model and transfer learning framework, Section IV details the experimental setup, Section V presents results and discussion, and Section VI concludes with future work.

II. Background and Related Work

A. Network Intrusion Detection Systems

Network Intrusion Detection Systems (NIDS) are central to modern cybersecurity frameworks, monitoring network traffic to detect and mitigate malicious activities. Traditional NIDS primarily rely on signature-based methods, which compare observed traffic against predefined patterns of known attacks [1], [2]. While effective for known threats, these systems often fail against novel or unknown attacks due to their static nature and reliance on comprehensive signature databases.

To address these limitations, data-driven anomaly detection has emerged as a more adaptive alternative. These methods utilize machine learning models, including neural networks, decision trees, and ensemble techniques such as XGBoost, to learn normal traffic patterns and identify deviations that may indicate intrusions [1]. Such models improve generalization and are better suited for detecting emerging threats in dynamic network environments.

B. Transfer Learning with Neurosymbolic AI

Transfer learning is widely adopted across domains to improve model performance when labeled data for the target domain is limited or distribution shifts occur. In computer vision tasks, surgical fine-tuning—selective adjustment of

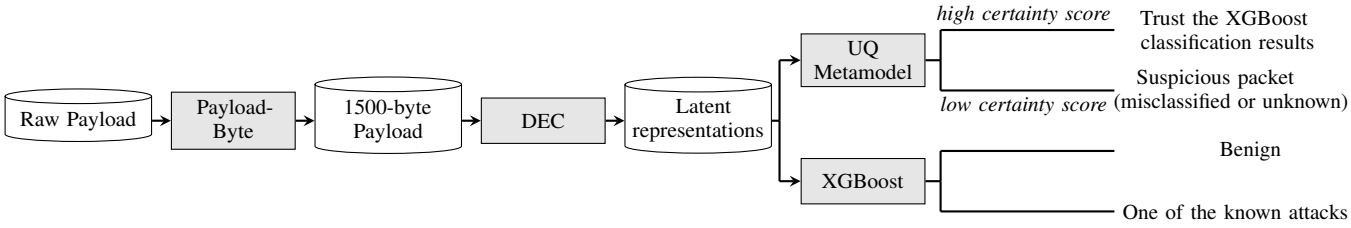
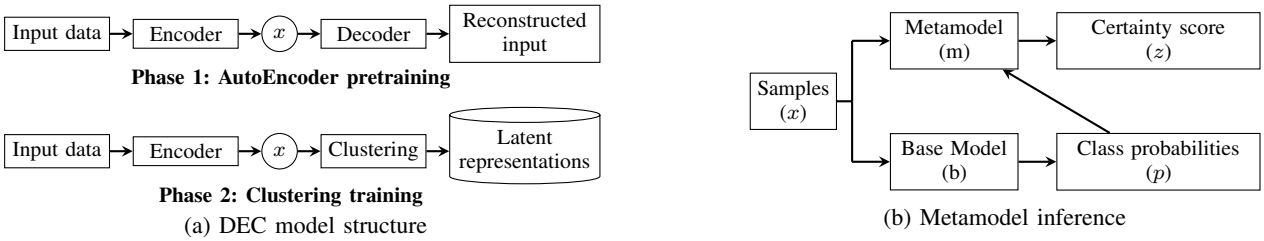


Fig. 1: The architecture of UQ NSAI (ODXU) model [2].



(a) DEC model structure

(b) Metamodel inference

Fig. 2: General structure of (a) DEC and (b) metamodel.

specific neural network layers—has shown significant improvement in adapting models to input-level, feature-level, and output-level shifts [4], [9]. Techniques such as TransTailor have optimized model architectures through targeted pruning to better align the model structure with task-specific data distributions [6], [10]. Adaptive fine-tuning approaches, like SpotTune, dynamically decide which layers to fine-tune per input instance, enhancing flexibility and performance [6], [11].

Although effective in various fields, transfer learning techniques are still underutilized in cybersecurity applications. Meanwhile, NSAI has emerged as a promising approach that combines the predictive capabilities of neural networks with the interpretability of symbolic reasoning [2].

In this paper, we propose integrating transfer learning within the NSAI framework. By allowing the reuse and adaptation of pre-trained NSAI models for new datasets or related tasks, this approach improves the scalability and robustness of intrusion detection systems in dynamic threat environments.

C. Uncertainty Quantification

Uncertainty Quantification (UQ) is crucial in security-critical systems, where reliable decision-making relies on confidence in model predictions. UQ techniques estimate prediction uncertainty, helping identify unknowns or out-of-distribution inputs. Standard methods for assessing the reliability of predictions from classifiers include confidence scores, Shannon Entropy [12], and BlackBox metamodeling [13]. The latter involves training a secondary model to evaluate the trustworthiness of the outputs from the primary classifier [2]. These techniques improve the robustness of NIDS by identifying uncertain predictions that may indicate new types of threats.

Recent advancements, such as SHAP [14] (Shapley Additive Explanations) and Information Gain [15] (IG) metrics, further support interpretable and trustworthy decision-making by highlighting the contributions of different features and the confidence in decisions. In this study, we integrate these techniques into our NSAI framework to assess and enhance predictive reliability in uncertain conditions.

III. Methodology

The architecture of the ODXU model is shown in Figure 1. The raw network payload is first processed using the Payload-Byte tool, which extracts the 1500 bytes of each packet to form fixed-length feature vectors. These 1500-byte feature vectors are then passed into a DEC model, which encodes them into 12-dimensional latent representations.

The latent representations produced by the DEC serve as input to two downstream models: (1) an XGBoost classifier for attack recognition and (2) a metamodel for UQ, which evaluates the uncertainty of the XGBoost predictions.

The DEC model is trained in two phases, as shown in Figure 2a: *I) AutoEncoder (AE) pretraining:* In this phase, an AE is trained to map the input data in lower-dimensional features by using an encoder and reconstruct the input data by using a decoder [16]. *II) Clustering training:* In this phase, the decoder is replaced by a clustering model, and the parameters of the clustering model are optimized while the encoder remains fixed. This phase adjusts the cluster centroids and the soft assignments of data points to clusters, refining the latent space to improve the separability of the clusters [16].

A. Transfer Learning Framework for ODXU

To assess the transferability of the ODXU model across different datasets or tasks, we asked two research questions:

1) Which components of the ODXU architecture (e.g., AE, clustering, XGBoost) should be fine-tuned or trained for effective transfer learning?
2) How many labeled samples are required to outperform baseline machine learning models such as Fully Connected Neural Networks (FcNN) or 1D Convolutional Neural Networks (1D-CNN)?

To answer these questions, we designed an experiment using the six transfer learning scenarios as shown in Table I. The AE has two options: "As is," where the pre-trained AE from

CIC-IDS-2017 is loaded and used without any further training, and "FT" (fine-tune), where the pre-trained AE is loaded and then further trained on a target dataset. The clustering component also considers two options. In the "FT" option, it loads a pre-trained clustering model from CIC-IDS-2017 and fine-tunes it on the target dataset. In the "Train" option, it loads the parameters of the pre-trained AE instead of the pre-trained clustering model and trains these parameters on the target dataset[1]. The classifier has two options: "FT," where a pre-trained classifier is loaded and fine-tuned on the target dataset, and "Train," where the classifier is trained entirely from scratch.

TABLE I: Transfer Learning Scenarios; FT: fine-tune

ODXU Components	Case					
	1	**2**	**3**	**4**	**5**	**6**
AE	FT	As is	As is	FT	As is	As is
Clustering	Train	FT	Train	Train	FT	Train
Classifier (XGBoost)	Train	Train	Train	FT	FT	FT

We adopt six metrics, including multiclass classification accuracy, binary classification accuracy, misclassified positive rate, false omission rate, F1 score, and competence. These metrics are consistent with some prior work [2], which comprehensively evaluates the effectiveness of transfer learning in each case.

B. Uncertainty Quantification Methods

We assess five uncertainty quantification (UQ) methods for our models. This includes two score-based methods: Confidence Scoring and Shannon Entropy [12], as well as the metamodel-based methods.

The UQ, through metamodeling, utilizes a base model $b(.)$ and a metamodel binary classifier $m(.)$, trained to estimate the correctness of $b(.)$'s predictions. The metamodel input, $\boldsymbol{X}_{\text{MetaUQ}}$, includes the original features $\boldsymbol{X_b}$ and augmented features derived from the base model. The training target $\boldsymbol{y_m}$, which denotes whether the base model predicts the sample's labels correctly (0) or not (1), is defined as:

$$\boldsymbol{y_m} = \begin{cases} 0 & \text{if } b(x_b) = y_b \\ 1 & \text{if } b(x_b) \neq y_b, \end{cases} \tag{1}$$

where y_b is the true label of x_b and $b(x_b)$ is the predicted output from the base model.

The metamodel, $m(.)$, will be trained using the target variable in Eq. (1) to minimize the loss between the training target $\boldsymbol{y_m}$ and the output $m(x_m) = z$. As a result, the output of the metamodel, z, provides an estimate of the probability that the base model's prediction $b(x_b) = y_b$ is correct, giving us a measure of the model's confidence in its predictions. A general interface of the metamodel is shown in Figure 2b.

[1]Note: As depicted in Figure 2a, the clustering module is initialized either from its pre-trained parameters of CIC-IDS-2017 or from the encoder of the AE. Therefore, if the AE is fine-tuned, the clustering cannot be initialized from its pre-trained checkpoint, as the encoder has changed. As a result, combinations such as FT-FT-Train or FT-FT-FT are invalid and excluded from our experiments.

1) Confidence scoring: is a straightforward and efficient method for estimating a model's certainty. The certainty scores can be computed using the order statistic of the prediction probabilities following the formulation:

$$z_{\text{conf}}(x) = p_{(k)} - p_{(k-1)}, \tag{2}$$

where $z_{\text{conf}}(x)$ is the certainty score of a sample x, $p_{(k)}$ is the largest probability from the list of probabilities $\boldsymbol{p}$ of k possible outcomes. A higher score indicates greater certainty, while a smaller value suggests more uncertainty.

2) Shannon entropy: is a well-known concept from information theory and is commonly used to measure uncertainty. This approach calculates the entropy for each sample based on the predicted probabilities for all classes generated by the base model. The entropy for a given sample x is computed using the following formula:

$$z_{\text{entropy}}(x) = -\sum p_i \times log(p_i), \tag{3}$$

where p_i represents the probability of class i, and the sum is taken over all possible classes. Unlike the confidence score, a higher entropy value indicates more uncertainty in the model's prediction, while a lower entropy value suggests higher confidence.

3) MetaUQ: is a metamodel approach to UQ. Three metamodels were considered, each utilizing a distinct type of uncertainty scores to augment the base model data.

a) MetaUQ$_{prob}$: The metamodel is augmented with the sorted predicted probabilities, $\boldsymbol{p'}$, and the confidence score (Eq. (2)). Including $\boldsymbol{p'}$ provides a more comprehensive view of the model's predictive distribution [17]. While the confidence score captures only the two top class probabilities, the full sorted vector reflects the distribution of belief across all classes. This enables the metamodel to identify uncertainty patterns, such as class ambiguity or flat distributions, that are not captured by the input $\boldsymbol{X_b}$ or confidence score alone. The augmented input of the metamodel (MetaUQ$_{prob}$) can be written as:

$$\boldsymbol{X}_{\text{MetaUQ}_{prob}} = [\boldsymbol{X_b}, \boldsymbol{p'}, \boldsymbol{z}_{\text{conf}}]. \tag{4}$$

b) MetaUQ$_{SHAP}$: Originally, Shapley Additive Explanations (SHAP) [14] values were a game-theoretic measure to attribute value to members of a coalition. Still, recently they have been widely adopted in machine learning for explainability purposes [18]. In this context, we consider the individual features in x as members of a coalition and aim to attribute to each feature a value that represents its contribution to the final prediction made by the model.

For tree-based models, such as XGBoost, SHAP values can be computed efficiently using a polynomial-time algorithm that leverages the structure of decision trees [19]. This allows us to get local explanations (i.e., the contribution of each feature to an individual prediction) without relying on approximations or sampling. In our implementation, for each input instance $\boldsymbol{x}$, we compute SHAP values as follows [14]:

$$\phi_i(b,x) = \sum_{S \subseteq I \setminus \{i\}} \frac{|S|!(|I|-|S|-1)!}{|I|!} \left[b(S \cup \{i\}) - b(S) \right], \tag{5}$$

where $\phi_i(b,x)$ is the SHAP value for feature i with respect to the base model $b(.)$ and input instance $\boldsymbol{x}$. ! is the factorial operator, $|\cdot|$ denotes the number samples of set or subsets. I is the set of input features. $S \subseteq I \setminus \{i\}$ is a subset of features from I excluding feature i. $b(S \cup \{i\}) - b(S)$ represents the change in the model's output when feature i is added to subset S, compared to the output when only S is used. We then extract SHAP values for the predicted class only, following the practices in class-specific interpretability [18].

The final augmented input to the SHAP metamodel is:

$$\boldsymbol{X}_{\text{MetaUQ}_{SHAP}} = [\boldsymbol{X_b}, \boldsymbol{\phi}(b,x)]. \tag{6}$$

c) MetaUQ$_{IG}$: Information Gain (IG) is a key concept in decision trees and gradient boosting, and it measures how much a split on a feature reduces uncertainty in the prediction. For classical classification tasks, the IG of a feature is calculated as follows: [15]:

$$\text{IG}(I,f) = H(I) - H(I|f), \tag{7}$$

where $H(I)$ represents the entropy of the entire dataset I, and $H(I|f)$ represents the conditional entropy, measuring the remaining uncertainty in the dataset I after splitting I based on the values of feature f.

In XGBoost, the concept of IG is extended to general loss functions. In this context, IG scores are defined as the expected reduction in loss due to a split; more details are here [20].

These scores represent the average usefulness of each feature across all decision splits. We then replicate the gain scores across all data samples to form an IG matrix, denoted as $\text{IG}_{\text{matrix}}$. We also add the sorted class probabilities from the base model's output. As a result, the final augmented input to the metamodel becomes:

$$\boldsymbol{X}_{\text{MetaUQ}_{IG}} = [\boldsymbol{X_b}, \boldsymbol{p'}, \text{IG}_{\text{matrix}}] \tag{8}$$

4) UQ Evaluation Metrics: To evaluate the effectiveness of our UQ methods on the transfer learning model, we focus on two tasks: misclassification detection and Open Set Recognition (OSR) detection. We compute the Area Under the Receiver Operating Characteristic Curve (AUROC) for each task to detect the misclassified and OSR samples. These two AUROC scores are the core features of evaluating the UQ methods.

IV. Experimental Setup

All experiments in this study were conducted using the Army Cyber Institute (ACI) Internet of Things (IoT) Network Traffic Dataset 2023 (ACI-IoT-2023) [21], which provides recent and comprehensive intrusion scenarios in Internet of Things (IoT) environments. The descriptive statistics of the dataset are shown in Table II.

The experiments were performed on a system with dual Intel® Xeon® Gold 5218R CPUs (2.10 GHz), providing a total of 80 logical threads and 754 GiB of system memory. For GPU acceleration, each experiment was conducted on a single NVIDIA A40 GPU, one of eight available, which is based on the Ampere architecture and features 48 GB of dedicated memory. Additionally, all experiments were timed and monitored to evaluate their computational efficiency.

TABLE II: Descriptive statistics of class distribution in the ACI-IoT-2023 dataset.

Class	Samples	Percent (%)
Benign	601,868	95.31
DNS Flood	18,577	2.94
Dictionary Attack	4,645	0.74
Slowloris	2,974	0.47
SYN Flood	2,113	0.33
Port Scan	582	0.09
Vulnerability Scan	445	0.07
OS Scan	156	0.02
UDP Flood	68	0.01
ICMP Flood	58	0.01
Total	**631,486**	**100.00**

A. Attack Recognition

To address the class imbalance in the ACI-IoT 2023 dataset, the benign class was downsampled by 95%, while the ICMP Flood and UDP Flood attack classes were upsampled by 200% [2]. The adjusted dataset was split into two subsets: *DEC-Train* and *DEC-Test*, with a 50/50 split. The *DEC-Train* set was further subsampled into portions of 10%, 25%, 50%, and 75%, which were then used to train and evaluate the models. Each portion of *DEC-Train* was divided into training and validation sets (75/25 split). Meanwhile, *DEC-Test* was used for XGBoost training, which is output from DEC and consists of 12 features. The *DEC-Test* set was further split 50/50 into *XGBoost-Train* and *XGBoost-Test*.

For the evaluation of transfer learning models, we trained two neural-based models: a FcNN and a 1D-CNN. The FcNN model consisted of three hidden layers with sizes [1024, 512, 100], totaling 2,115,180 trainable parameters. The 1D-CNN model consisted of three convolutional layers with channels [32, 64, 128] and a kernel size of [3], followed by a hidden layer with 50 neurons. The 1D-CNN model had a total of 181,904 trainable parameters.

B. Misclassification and Open Set Recognition Detection

For the misclassification and OSR detection tasks, the Slowloris attack was held out as the "unknown" attack. The remaining samples were subsampled to balance the benign and attack classes, with the dataset split following the same procedure as in the attack recognition task. In the case of our UQ metamodel, a more refined training set was required. Using a simple split of *XGBoost-Test* resulted in poor performance. We found that a metamodel trained on a highly accurate base model would tend to classify all samples as high certainty, which minimized the model's training objective. Through experimentation, we determined that holding five times as

Many correctly classified samples as misclassified samples resulted in improved MetaUQ performance. Consequently, we created new labels for the *XGBoost-Test* set: 0 for correct predictions and 1 for incorrect predictions. Class 0 was then subsampled to five times the size of class 1. This adjusted dataset was split 80/20 into *Metamodel-Train* and *Metamodel-Test*. For testing, we generated a new test set for the OSR detection task by combining equal numbers of unknown samples and *Metamodel-Test* samples. This combined set was used for the final evaluation of our models.

V. Results and Discussion

A. Attacks Recognition

Table III presents the multiclass accuracy of transfer learning models across different training set portions, based on the configurations in Table I. When considering the composition of the DEC components, performance increases from models using a pre-trained AE (As is) with fine-tuned clustering (e.g., Case 2), to those fine-tuning the AE and training the clustering module (e.g., Case 1), and is highest when using the pre-trained AE and training the clustering (e.g., Case 3). For example, with 50% of the training set, Case 2 achieves .9799 accuracy, Case 1 improves to .9805, and Case 3 reaches the highest at .9827.

TABLE III: The multiclass accuracy of transfer learning across varying portions of the training dataset.

%	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6
10	.9666	.9616	.9764	.9680	.9659	.9784
25	.9690	.9679	.9789	.9736	.9706	.9802
50	.9805	.9799	**.9827**	**.9813**	**.9808**	**.9841**
75	**.9824**	**.9809**	**.9836**	**.9833**	**.9816**	**.9845**

Furthermore, when comparing models with the same AE and clustering settings, fine-tuning the classifier (Cases 4 to 6) consistently yields higher performance than training it from scratch (Cases 1 to 3). For instance, at 75% of the data, Case 3 achieves .9836, while Case 6 achieves a higher accuracy of .9845. These results suggest that starting from a well-initialized classifier leads to better generalization and performance, emphasizing the advantage of transfer learning. When compared with neural-based models (.9808 for FcNN and .9679 for 1D-CNN), the transfer learning models can outperform them when trained with at least 50% (≈16,000 samples) of the dataset in Cases 3–6, and at least 75% (≈ 23,000 samples) in Cases 1 and 2[2]. As Case 6 (AE: As is, clustering: Train, and classifier: FT) provides the highest accuracy, we will use Case 6 for further experimentation in subsequent experiments.

a) Early Stopping Improvement: During the experiments, all models were trained in the same setting (i.e., number of epochs). The models improve quickly at first, then slow down. To prevent slow progress, we introduced "early stopping rounds" (η) and two thresholds in loss (δ) as hyperparameters. We assume that if the change in the loss of AE pre-training or clustering training within the δ for a specified number of epochs, the model stops. We tested the Case 6 model using early stopping rounds of [10, 15, 20], and δ values for AE pre-training of [0.0005, 0.001] and clustering training of [0.005, 0.01], with 50% and 75% dataset portions.

[2]These results are based on an ablation study and examining several cases.

TABLE IV: The multiclass accuracy across hyperparameters.

Exp	η	δ		Accuracy		Training time	
		AE	Cluster	50%	75%	50%	75%
1	10	0.001	0.01	.9807	.9791	0:23:07	0:20:09
2	10	0.0005	0.005	**.9817**	**.9820**	0:25:00	0:28:48
3	15	0.001	0.01	**.9815**	.9806	0:27:32	0:21:38
4	15	0.0005	0.005	**.9819**	**.9831**	0:29:10	0:37:01
5	20	0.001	0.01	**.9820**	**.9829**	0:39:02	0:42:46
6	20	0.0005	0.005	**.9824**	**.9834**	**0:49:39**	**1:02:50**

Table IV presents the multiclass accuracy across various combinations of hyperparameters. The "**Exp**" column denotes the experiment index. For the 50% training data, all configurations except Experiment 1 outperform the FcNN baseline (.9808). The best accuracy for 50% data is obtained in Experiment 6 with an accuracy of .9824. For the 75% data portion, Experiments 2, 4, 5, and 6 outperform the FcNN baseline, with Experiment 6 again achieving the highest accuracy of .9834. These results confirm that Experiment 6 offers the most robust configuration overall.

In addition to performance, training time is an important consideration. As η increases and δ decreases, training time, showing in hh:mm:ss format, tends to rise. For example, Experiment 6, which combines a higher η (20) and lower δ values (0.0005, 0.005), requires the longest training time for both data portions (up to **1:02:50** for 75%). The choice of specific hyperparameters and portions of the dataset will depend on specific tasks or the dataset. In our study, the next experiment will proceed with Case 6 using the hyperparameters from Experiment 6. The extended evaluation of this configuration on the attack recognition task is reported in Table V.

TABLE V: Results of six metrics across models.

Measurement	FcNN	1D-CNN	Case 6
Multiclass Accuracy	.981	.968	**.983**
Binary Accuracy	.985	.974	**.987**
Misclassified Positive Rate	.022	.035	**.019**
False Omission Rate	.016	.029	**.014**
F1 Score	.985	.974	**.988**
Competence	.948	.935	**.969**

B. Misclassification and Open Set Detection

The results for misclassification and OSR detection are shown in Table VI. In misclassification detection, Confidence and Shannon Entropy perform similarly, both achieving an accuracy of .911, while the metamodels obtain higher accuracy. The MetaUQ$_{prob}$ and MetaUQ$_{SHAP}$ have accuracy values close to .924, and the MetaUQ$_{IG}$ achieved the highest performance at .926. A similar trend is observed with TP@(TN=.95). These results suggest that the MetaUQ$_{IG}$ is effective for detecting misclassification.

TABLE VI: Performance comparison across UQ methods.

Metrics	Score-based		MetaUQ		
	Conf	Entropy	Prob	SHAP	IG
Misclassification AUROC	.911	.911	.924	.924	**.926**
Misclassification TP@(TN=.95)	.529	.529	.559	.559	**.588**
Unknown attack AUROC	.916	.919	.921	**.938**	.921
Unknown attack TP@(TN=.95)	.435	.462	.489	**.590**	.469

For OSR detection, there is little difference between Confidence, Shannon Entropy, MetaUQ$_{prob}$, and MetaUQ$_{IG}$. However, the MetaUQ$_{SHAP}$ shows a big gap in AUROC for unknown attack detection. When fixing the true negative rate at 95%, the MetaUQ$_{SHAP}$ exceeds the second-highest method (MetaUQ$_{prob}$) by more than 10%. This indicates that the MetaUQ$_{SHAP}$ metamodel provides the best performance for OSR detection.

VI. Conclusion

This paper presents a transfer learning framework for the ODXU Neurosymbolic AI model applied in network intrusion detection systems. The results indicate that utilizing a pre-trained AE, followed by retraining the clustering algorithm and fine-tuning the classifier model (XGBoost), yields the highest accuracy. Transfer learning models began to surpass neural-based models when trained on at least 50% of the data, or 16,000 samples (these findings are based on the ablation study when examining several cases). To prevent prolonged training times, we implemented an early stopping condition that halts training if the AE and clustering losses drop below 0.0005 and 0.005, respectively, for 20 consecutive epochs.

Our findings also suggest that metamodel-based methods are more effective than score-based methods for uncertainty quantification (UQ), with each metamodel being tailored for specific tasks. However, we recognize that these results may differ depending on the datasets or tasks used. Therefore, future work will focus on applying our transfer learning model to additional cybersecurity datasets, including the Canadian Institute for Cybersecurity (CIC) IoT 2023 dataset and the Unified Multimodal Network Intrusion Detection Systems (UM-NIDS) dataset, to further assess its performance.

Acknowledgment

This work was supported by the U.S. Military Academy (USMA) under Cooperative Agreement No. W911NF-23-2-0108 and the Defense Advanced Research Projects Agency (DARPA) under Support Agreement No. USMA 23004. The views and conclusions expressed in this paper are those of the authors and do not reflect the official policy or position of the U.S. Military Academy, U.S. Army, U.S. Department of Defense, or U.S. Government.

References

[1] M. Al-Omari, M. Rawashdeh, F. Qutaishat, M. Alshira'H, and N. Ababneh, "An intelligent tree-based intrusion detection model for cyber security," *Journal of Network and Systems Management*, vol. 29, no. 2, p. 20, 2021.

[2] J. Sander, C.-E. J. Yu, B. Jalaian, and N. D. Bastian, "Uncertainty-quantified neurosymbolic ai for open set recognition in network intrusion detection," in *MILCOM 2024-2024 IEEE Military Communications Conference (MILCOM)*. IEEE, 2024, pp. 13–18.

[3] B. Jalaian and N. D. Bastian, "Neurosymbolic ai in cybersecurity: Bridging pattern recognition and symbolic reasoning," in *MILCOM 2023-2023 IEEE Military Communications Conference (MILCOM)*. IEEE, 2023, pp. 268–273.

[4] F. Shahoveisi, H. Taheri Gorji, S. Shahabi, S. Hosseinirad, S. Markell, and F. Vasefi, "Application of image processing and transfer learning for the detection of rust disease," *Scientific Reports*, vol. 13, no. 1, p. 5133, 2023.

[5] S. Amiriparian, T. Hübner, V. Karas, M. Gerczuk, S. Ottl, and B. W. Schuller, "Deepspectrumlite: A power-efficient transfer learning framework for embedded speech and audio processing from decentralized data," *Frontiers in Artificial Intelligence*, vol. 5, p. 856232, 2022.

[6] S. Moon, S. Kim, and H. Wang, "Multimodal transfer deep learning with applications in audio-visual recognition," *arXiv preprint arXiv:1412.3121*, 2014.

[7] H. E. Kim, A. Cosa-Linan, N. Santhanam, M. Jannesari, M. E. Maros, and T. Ganslandt, "Transfer learning for medical image classification: a literature review," *BMC medical imaging*, vol. 22, no. 1, p. 69, 2022.

[8] I. Sharafaldin, A. H. Lashkari, A. A. Ghorbani *et al.*, "Toward generating a new intrusion detection dataset and intrusion traffic characterization." *ICISSp*, vol. 1, pp. 108–116, 2018.

[9] Y. Lee, A. S. Chen, F. Tajwar, A. Kumar, H. Yao, P. Liang, and C. Finn, "Surgical fine-tuning improves adaptation to distribution shifts," *arXiv preprint arXiv:2210.11466*, 2022.

[10] B. Liu, Y. Cai, Y. Guo, and X. Chen, "Transtailor: Pruning the pre-trained model for improved transfer learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 10, 2021, pp. 8627–8634.

[11] Y. Guo, H. Shi, A. Kumar, K. Grauman, T. Rosing, and R. Feris, "Spottune: transfer learning through adaptive fine-tuning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4805–4814.

[12] G. Smith, "Quantifying information flow using min-entropy," in *2011 Eighth International Conference on Quantitative Evaluation of SysTems*. IEEE, 2011, pp. 159–167.

[13] T. Chen, J. Navrátil, V. Iyengar, and K. Shanmugam, "Confidence scoring using whitebox meta-models with linear classifier probes," in *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 2019, pp. 1467–1475.

[14] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems*, vol. 30, 2017.

[15] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, pp. 81–106, 1986.

[16] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ser. ICML'16. JMLR.org, 2016, p. 478–487.

[17] N. Tagasovska and D. Lopez-Paz, "Single-model uncertainties for deep learning," *Advances in neural information processing systems*, vol. 32, 2019.

[18] S. M. Lundberg, G. G. Erion, and S.-I. Lee, "Consistent individualized feature attribution for tree ensembles," 2019. [Online]. Available: https://arxiv.org/abs/1802.03888

[19] S. M. Lundberg, G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, and S.-I. Lee, "From local explanations to global understanding with explainable ai for trees," *Nature machine intelligence*, vol. 2, no. 1, pp. 56–67, 2020.

[20] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[21] N. Bastian, D. Bierbrauer, M. McKenzie, and E. Nack, "Aci iot network traffic dataset 2023," 2023. [Online]. Available: https://dx.doi.org/10.21227/qacj-3x32