REACT: Residual-Adaptive Contextual Tuning for Fast Model Adaptation in Cybersecurity

Jiayun Zhang*
University of California, San Diego
jiz069@ucsd.edu

Junshen Xu
Amazon Web Services
jsxu@amazon.com

Yi Fan
Amazon Web Services
fnyi@amazon.com

Abstract

Cybersecurity applications are challenged by constant distribution shifts due to the evolvement of services, users, and threats, degrading pretrained model performance. Fast adaptation is crucial for maintaining reliable security measures. Existing works primarily focus on pretraining models that can quickly adapt to new distributions, yet their fine-tuning relies on a rudimentary strategy that treats each shift independently. In this paper, we introduce REACT, a novel adaptation framework via Residual-Adaptive Contextual Tuning, working for sparsely-labeled and imbalanced cybersecurity data. REACT decomposes the weights of a neural network into two complementary components: meta weights, a shared foundation of general knowledge, and adaptive weights, tailored to specific shifts. A hypernetwork is employed to learn distribution patterns from few-shot data and relevant contexts and prime adaptive weights close to the optimal configuration, reducing fine-tuning effort. The meta weights and the hypernetwork are updated alternatingly to maximize generalization and adaptability. Extensive experiments across multiple datasets and neural networks demonstrate that REACT improves AUC by 14.85% compared to models without adaptation, outperforming the state-of-the-art.

1 Introduction

Cybersecurity applications undergo continual distribution shifts for various factors, including users joining and leaving the network, changes in user behavior, and software updates. For instance, during special events on the Web, such as major sales promotions, there is often a surge in users visiting the site and subscribing to services, and many of them may cancel the subscription and reduce their activity after the event, causing abrupt shifts in network traffic. Models trained on typical traffic patterns are less effective in these scenarios, resulting in increased false positives and false negatives. Fast adaptation is essential for timely and effectively identifying threats in dynamic environments.

A critical challenge in managing distribution shifts in cybersecurity is the lack of ground truth, which limits the ability to align models with shifted distributions. Existing unsupervised adaptation methods (Stojanov et al. [2021], Tanwisuth et al. [2021], Shen et al. [2022]), which rely on labeled source domains to guide adaptation, fails to address this scenario.

Besides, cybersecurity data often exhibit extreme imbalance, with a few suspicious activities (e.g., unauthorized access attempts, anomalous traffic, malware) hidden among a vast majority of benign patterns. This imbalance, however, presents an opportunity to address distribution shift—rather than learning what the exact patterns of every benign and suspicious behaviors are, the model could learn to distinguish between the majority and minority for better generalization. Building on this insight, we approach the problem from both pretraining and fine-tuning perspectives. From pretraining perspective, the pretrained model should establish a broad, generalizable baseline to distinguish

*work done during internship at Amazon Web Services

38th Workshop on Fine-Tuning in Machine Learning (NeurIPS 2024).

diverse sets of majority and minority. From the fine-tuning perspective, the model should then be fine-tuned based on a summary of the shifted patterns, quickly adapting to specific distribution.

Recent advances in adaptation for cybersecurity (Li et al. [2023a], Wang et al. [2022], Li et al. [2023b]) have emphasized the pretraining, typically via meta-training a model initialization that can adapt with a few data. However, their fine-tuning is conducted independently for each testing scenario, which could be inefficient especially when numerous new tasks share similar distributions. We propose that by recognizing task similarities for fine-tuning, the adaptation can be further enhanced.

In this work, we introduce REACT, a model adaptation method via Residual-Adaptive Contextual Tuning, working effectively with unlabeled and highly imbalanced data. Given a neural network, REACT decomposes its weights into the sum of two complementary components: meta weights, which are shared globally and capture general knowledge, and adaptive weights, which are the residual component tailored to specific distributions. The framework meta-learns the model on a diverse set of tasks with different distributions. To capture task similarities, we integrates a hypernetwork (Ha et al. [2016]) that generates adaptive weights based on a small set of data and relevant contextual information. The hypernetwork positions the adaptive weights close to the optimal configuration, reducing fine-tuning effort. During training, REACT alternates between updating the meta weights and the hypernetwork via meta learning, iteratively refining them to maximize generalizability and adaptability. The framework is model-agnostic, applicable to various neural networks and objective functions within cybersecurity contexts. We evaluate REACT on three datasets for key applications in cybersecurity, network intrusion detection and malware detection, using different neural network models in both unsupervised and semi-supervised settings. Compared to models without adaptation, REACT improves the AUROC by 14.85% with few-shot fine-tuning (e.g., update 1 - 10 iterations on 10 - 100 samples). Ablation studies and sensitivity analyses demonstrate that REACT is robust to variations in the number of available samples and contamination in training data. We also showcase the capability of REACT for parameter-efficient fine-tuning, achieving 5.75% higher AUROC with 94.3% fewer parameters updated compared to conventional fine-tuning, highlighting its efficiency.

Contributions: (1) We address distribution shift in cybersecurity. We approach the problem from both pretraining and fine-tuning perspectives to maximize generalizability and adaptability. **(2)** We introduce REACT, a novel adaptation framework that decomposes a model into the sum of a meta component for generalization and an adaptive component for personalization. REACT alternately meta-learns the meta component and a hypernetwork for adaptive weight generation, enabling fast adaptation via fine-tuning on limited new data with few gradient steps. **(3)** Extensive evaluations on key cybersecurity applications demonstrate that REACT consistently outperforms various state-of-the-art methods across diverse models and datasets.

2 Related Work

Distribution Shifts in General Machine Learning. Distribution shift occurs when the training and testing data distributions differ, leading to poor generalization (Gretton et al. [2008]). To address the challenge, adaptation methods have been proposed (Wilson and Cook [2020], Farahani et al. [2021], Liu et al. [2022]). We focus on works designed for unsupervised or semi-supervised cases due to the data specificity in cybersecurity. Unsupervised domain adaptation (Ben-David et al. [2010]) is a closely-related topic, which adapts models to target domains that have no labeled data. Methods include invariant representation learning (Stojanov et al. [2021]), prototype-oriented conditional transport Tanwisuth et al. [2021], contrastive pre-training (Shen et al. [2022]). These methods rely on labels from source domains, which are often not available in cybersecurity scenarios.

Distribution Shifts in Cybersecurity. In cybersecurity, distribution shifts have been observed in various applications, such as malware detection (Jordaney et al. [2017], Pendlebury et al. [2019], Barbero et al. [2022]), network intrusion detection (Channappayya et al. [2024]), and log anomaly detection (Jia et al. [2023]). Traditional supervised approaches (Jordaney et al. [2017], Pendlebury et al. [2019], Barbero et al. [2022]) require extensive labeling, limiting their practicality in real-world deployment. Recent efforts have recognized this limitation and have been exploring adaptation approaches for scenarios with scarce labels. Unsupervised domain adaptation methods, like learning domain-invariant representations (Carvalho et al. [2023]) have been extended to cybersecurity. However, they typically requires simultaneous training on source and target domains, making it less suitable for emerging domains. Test-time adaptation methods, including batch normalization

updates (Li et al. [2023a]) energy-based models (Wang et al. [2022]), and trend estimation (Kim et al. [2024]), update models during inference without gradient descent. Though efficient, they are limited to minor shifts Gui et al. [2024] or sequential shifts that display continuous patterns (Kim et al. [2024]). To address more severe and random shift, meta-learning (Finn et al. [2017]) is a promising approach. Methods train a meta model on a variety of learning tasks, enabling quick adaptation with minimal data, as seen in few-shot anomaly detection (Ding et al. [2021]) and prototype-oriented transport for time-series anomaly detection (Li et al. [2023b]). However, these existing methods overlook the potential of learning to adapt during the meta-learning process. Our work focuses on both pretraining and fine-tuning aspects to enhance adaptability and scalability of the systems.

3 Problem Definition

Distribution shift in cybersecurity involves changes in the probability distribution of data over time or across domains (e.g., users), affecting feature distribution $\mathcal{P}(x)$, posterior distribution $\mathcal{P}(y|x)$, or both. Consider a model $f(\cdot;\theta)$ trained on a dataset $\mathbf{D}$ from distribution $\mathcal{P}$. Its parameters θ are optimized by the objective: $\theta^* = \text{argmin}_\theta \mathbb{E}_{x\sim\mathcal{P}}\mathcal{L}(f(x;\theta)$, where $\mathcal{L}$ is the loss function. Dataset $\mathbf{D} = \{x_i\}_{i=1}^N$ is either unlabeled or contains limited labels and is dominated by benign samples. Our goal is to develop an adaptation method that updates model parameters to θ' using a small unlabeled or sparsely-labeled set $\mathbf{D}'$ ($|\mathbf{D}'| = k \ll |\mathbf{D}|$) from the new distribution $\mathcal{P}'$.

4 Residual-Adaptive Contextual Tuning for Fast Model Adaptation

We propose a meta-learning framework for adaptation that alternates between optimizing a meta model and a hypernetwork. The pseudo-code is provided in Algorithm 1.

Algorithm 1: Training Procedure of REACT

Input: Task distribution $p(\mathcal{T})$, target network f, hypernetwork h, training iterations T
Output: Meta weights θ_{meta}, hypernetwork weights ϕ

Initialize model weights θ_{meta} and ϕ;
while *not converged* **do**
  `// Update meta weights.`
  Sample a set of tasks $\{\mathcal{T}_i\}_{i=1}^M \sim p(\mathcal{T})$;
  for *each task* $\mathcal{T}_i$ **do**
    Form support set $\mathbf{D}^i_{\text{support}}$ and query set $\mathbf{D}^i_{\text{query}}$ and extract contextual information c_i;
    Generate adaptive weights: $\theta^i_{\text{adapt}} = h(\mathbf{D}^i_{\text{support}}, c_i; \phi)$;
    Fine-tune θ^i_{adapt} on $\mathbf{D}^i_{\text{support}}$ following Eq. 1;
  Update θ_{meta} following Eq. 2;
  `// Update hypernetwork.`
  Sample a set of tasks $\{\mathcal{T}_j\}_{j=1}^M \sim p(\mathcal{T})$;
  for *each task* $\mathcal{T}_j$ **do**
    Form support set $\mathbf{D}^j_{\text{support}}$ and query set $\mathbf{D}^j_{\text{query}}$, and extract contextual information c_j;
    Generate adaptive weights: $\theta^j_{\text{adapt}} = h(\mathbf{D}^j_{\text{support}}, c_j; \phi)$;
  Update ϕ following Eq. 3;

Task Sampling for Meta Learning. To let the model learn how to adapt to new distributions, we create a diverse set of tasks that reflect the expected variations in the target application. We sample tasks from training set $\mathbf{D}$ by simulating relevant data shifts. For instance, if the goal of adaptation is to address temporal shifts, the data can be grouped by periods such as days or months, with each period forming a separate task. The dataset $\mathbf{D}_i$ for each task $\mathcal{T}_i$ is further divided into a support set $\mathbf{D}^i_{\text{support}}$ and a query set $\mathbf{D}^i_{\text{query}}$. The support set is used to fine-tune the model to obtain task-specific parameters, while the query set evaluates the generalization of the fine-tuned model.

Weight Decomposition Given a neural network, we decompose its weights into the sum of two complementary components: meta weights θ_{meta}, which capture global patterns, and adaptive weights

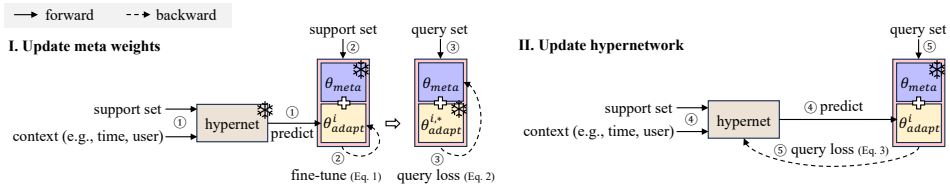


Figure 1: Alternating training. REACT alternates between updating the meta weights and the hypernetwork to iteratively maximize generalization and adaptability. In each iteration, we first sample a set of tasks to update the meta weights, then sample another set to train the hypernetwork.

θ_{adapt}, which fine-tune the model for specific data distributions. The meta weights provide a broad understanding of the global distribution, while the adaptive weights act as a residual component that tailors the model to new distributions. The two components are added to form the full model weights, i.e., $\theta = \theta_{\text{meta}} + \theta_{\text{adapt}}$.

Adaptive Weight Generation with Hypernetwork. We integrate a hypernetwork to predict adaptive weights based on the aggregated specificity of data from new distributions. A hypernetwork (Ha et al. [2016]) is a neural network that predicts the weights for another neural network. It has shown effective in improving learning efficiency through parameter sharing (Mahabadi et al. [2021], Zhang et al. [2024a], Bonet et al. [2024], Zhang et al. [2024b]). Our hypernetwork includes an encoder that processes data from the support set to produce feature representations. These representations are averaged and passed through a series of linear layers, with each layer generating the weights for a corresponding layer in the target network. This weight generation reduces fine-tuning efforts by preparing the adaptive weights close to its optimal values. Let h represent the hypernetwork with parameters ϕ. Given the support set $\mathbf{D}^i_{\text{support}}$ for task $\mathcal{T}_i$, the hypernetwork generates model weights $\theta^i_{\text{adapt}} = h(\mathbf{D}^i_{\text{support}}; \phi)$, which are then loaded into the target network f. Given multiple tasks $\{\mathcal{T}_i\}_{i=1}^M$ with datasets $\{\mathbf{D}_i\}_{i=1}^M$, the objective is to minimize the loss over the query sets of these tasks.

To enhance the hypernetwork's ability to handle varying distributions, we integrate the contextual information c_i about distribution $\mathcal{P}_i$ as an additional input. This context offers semantic insights into the shifts and helps capture similarities across distributions. We incorporate a context encoder in the hypernetwork to transform contexts into embeddings, which are then added to the data representations for weight generation. The choice of context depends on the type of shift. For example, we use time information for temporal shifts, with positional encoding (Vaswani [2017]) generating embeddings. Details about context modeling for different tasks can be found in Section 5.1.

Alternating Training. We design an alternating training scheme for these two components to maximize generalization and adaptability. Figure 1 illustrates the process. Each iteration begins with updating the meta weights. We sample a set of tasks $\{\mathcal{T}_i\}_{i=1}^M$, fix the hypernetwork h and use it to generate the initialization of adaptive weights $\theta^i_{\text{adapt}} = h(\mathbf{D}^i_{\text{support}}, c_i; \phi)$. The generated weights are then fine-tuned to derive the optimal weights ${\theta^i_{\text{adapt}}}^*$ for distribution $\mathcal{P}_i$ using the support set $\mathbf{D}^i_{\text{support}}$:

$$ {\theta^i_{\text{adapt}}}^* = \underset{\theta_{\text{adapt}}}{\operatorname{argmin}} \sum_{x \in \mathbf{D}^i_{\text{support}}} \mathcal{L}(f(x; \theta_{\text{meta}}, \theta_{\text{adapt}})). \tag{1} $$

We fix these fine-tuned adaptive weights and update the meta model by minimizing the loss on the query set $\mathbf{D}^i_{\text{query}}$. Let η_{meta} be the learning rate for updating meta weights. The update of meta weight after one gradient step is as follows:

$$ \theta_{\text{meta}} \leftarrow \theta_{\text{meta}} - \eta_{\text{meta}} \nabla_{\theta_{\text{meta}}} \sum_{\mathcal{T}_i} \sum\nolimits_{x \in \mathbf{D}^i_{\text{query}}} \mathcal{L}(f(x; \theta_{\text{meta}}, {\theta^i_{\text{adapt}}}^*)). \tag{2} $$

Next, we sample another set of tasks $\{\mathcal{T}_j\}_{i=1}^M$, fix the meta weights learned in the previous step, and update the hypernetwork using the query sets. Let η_{HN} be the learning rate for updating the hypernetwork. The weight update of hypernetwork after one gradient step is as follows:

$$ \phi \leftarrow \phi - \eta_{\text{HN}} \nabla_{\phi} \sum\nolimits_{\mathcal{T}_j} \sum\nolimits_{x \in \mathbf{D}^j_{\text{query}}} \mathcal{L}(f(x; \theta_{\text{meta}}, h(\mathbf{D}^j_{\text{support}}, c_j; \phi))). \tag{3} $$

Regularization. We apply L2 regularization to the hypernetwork-generated adaptive weights, encouraging them to act as residuals to the globally shared meta weights. The query loss for optimizing

Table 1: Experiment configurations and dataset statistics after preprocessing.

Dataset	Application	# Train	# Test	# Train Tasks	# Test Tasks	k	Shift by	Model
AnoShift	Network Intrusion Detection	1,388,884	1,833,329	50	110	100	Time	AutoEncoder
NSL-KDD	Network Intrusion Detection	28,920	6,033	8	6	10	Service	GOAD
Malware	Malware Detection	15,301	5,320	36	12	10	Time	DeepSVDD

the hypernetwork, denoted as $\mathcal{L}^i_{\text{query}}$, is combined with regularization as $\mathcal{L} = \mathcal{L}^i_{\text{query}} + \lambda|\theta^i_{\text{adapt}}|^2_2$, where λ controls the regularization strength.

Inference During inference, the meta weights and the hypernetwork are fixed. A small set of support data $\mathbf{D}^j_{\text{support}}$ from the new distribution $\mathcal{P}_j$ along with its contextual information c_j are fed into the hypernetwork for adaptive weight generation $\theta^j_{\text{adapt}} = h(\mathbf{D}^j_{\text{support}}, c_j; \phi)$. The adaptive weights are then fine-tuned on $\mathbf{D}^j_{\text{support}}$. Finally, the two parts of the weights are summed and used for inference.

5 Experiments

5.1 Experiment Setups

Datasets and Backbone Models. We consider two key applications in cybersecurity: network intrusion detection and malware detection. For network intrusion detection, we use AnoShift (Drăgoi et al. [2022]) and NSL-KDD (Tavallaee et al. [2009]), and for malware detection, we use the Malware dataset (Huynh et al. [2017]). We sample the datasets to form a 10% anomaly rate for both training and testing. REACT is model-agnostic and applicable with various neural networks. We employ three representative model architectures: AutoEncoder (Aggarwal and Aggarwal [2017]) for AnoShift, DeepSVDD (Ruff et al. [2018]) for Malware, and GOAD (Bergman and Hoshen [2020]) for NSL-KDD. Both AutoEncoder and DeepSVDD are unsupervised methods. Since GOAD is a semi-supervised methods which assumes training on benign data only, we filter out intrusion attacks in the training data for the NSL-KDD dataset. Table 1 summarize the configuration.

Compared Methods. We compare our approach with the following baselines: **(1) w/o adaptation** ($\mathcal{A}$): The model is trained on the training data and directly tested on each test task. This serves as the pretrained model, denoted as $\mathcal{A}$. **(2) Training from scratch**: For each task, a model is trained from scratch using k samples and is used to perform evaluation on the task. **(3) Fine-tuning**: For each task, the pretrained model $\mathcal{A}$ is fine-tuned on k samples from the task. **(4) Continual learning**: Starting from the pretrained model $\mathcal{A}$, we sequentially fine-tune the latest updated model using k samples from each task. **(5) CL w/ experience replay** (Chaudhry et al. [2019]): To mitigate catastrophic forgetting, we include a variant of continual learning with experience replay. We maintain a memory buffer that stores previously used data. During each fine-tuning iteration, we sample a batch from the buffer and combine it with the loss from the new batch, applying a weighted sum.**(6) OC-MAML** (Frikha et al. [2021]): It extends MAML to one-class classification by forming one-class support sets during episodic data sampling to optimize the meta model specifically for one-class classification tasks.

Choices of Contexts For AnoShift and Malware dataset where shifts occur along time, we use time information as the context, which is modeled with positional encoding (Vaswani [2017]). For NSL-KDD dataset where shifts occur across services, we first feed these services names to a Large Language Model with the prompt "please briefly describe each of these web services, including the normal and anomalous patterns". Then, we use a pretrained Sentence Transformer[2] to generate embeddings for the descriptions.

Adaptation Configurations and Evaluation Metrics For AnoShift and Malware, we adapt the model to each test month. For NSL-KDD, we adapt the model to each service (e.g., HTTP, Telnet, etc.) by randomly sampling half of the services for training and the other half for testing. After dividing the training and testing tasks, we exclude services with fewer than 20 benign data points to ensure enough data for testing after allowing the model to be updated on k test samples for adaptation. For fine-tuning, we set the number of support samples k and the steps of gradient update according to the dataset size and the convergence rate of the learning task. Specifically, we use $k = 100$ samples for AnoShift and $k = 10$ for Malware and NSL-KDD. Fine-tuning is conducted for 10 epochs for

[2] `https://huggingface.co/sentence-transformers`

Table 2: Experiment results on three datasets, each sampled with a 10% positive (i.e., intrusion attack, malware) rate. REACT consistently achieves the highest performance across all datasets.

Method	AnoShift		Malware		NSL-KDD	
	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR
training from scratch (on all data)	0.7681	0.3689	0.6010	0.1822	0.9308	0.7107
w/o adaptation ($\mathcal{A}$)	0.7110	0.3204	0.5165	0.1644	0.7420	0.5110
training from scratch	0.7398	0.3398	0.3659	0.1337	0.7382	0.4219
fine-tuning	0.7039	0.3333	0.5678	0.1797	0.8175	0.5098
continual learning	0.6087	0.3063	0.5879	0.1873	0.8285	0.5188
CL w/ experience replay (Chaudhry et al. [2019])	0.6144	0.2996	0.6022	0.1932	0.8356	0.5114
OC-MAML (Frikha et al. [2021])	0.7770	0.3811	0.6779	0.2334	0.8547	0.5504
REACT	**0.8226**	**0.4376**	**0.7252**	**0.2750**	**0.8673**	**0.5559**

AnoShift and Malware and 1 epoch for NSL-KDD since its tasks converge faster. In Section 5.4, we conduct a sensitivity analysis on k to assess the robustness of our model to varying degrees of data scarcity. We report both the AUROC and AUPR scores. All experiments are repeated for five times with the same set of random seeds for all methods. Results are averaged across all test tasks and runs.

5.2 Main Results and Analysis

The results are presented in Table 2. To show the upper bound of training from scratch, we also evaluate the performance of training a model from scratch using all data from each test task (highlighted in grey). When sufficient data is available from the new tasks, training a model from scratch yields better performance than using a pretrained model without adaptation. When comparing the models trained from scratch, fine-tuning, and continual learning, it is shown that the pretrained model $\mathcal{A}$ could be a poor initialization for shifted distributions (e.g., on AnoShift). We also observe that incorporating experience replay slightly improves performance compared to continual learning without any strategy to prevent catastrophic forgetting. However, this improvement is limited. REACT consistently outperforms the baselines and even surpasses models trained from scratch using all data on two datasets. This is because REACT adapts from a model meta-trained on a larger and more diverse training set than each independent test task, which provides a stronger foundation. We note that on NSL-KDD, GOAD achieves high scores when trained from scratch using all data since it is trained on benign data only, but such training is impractical in real world. When compared to other fine-tuning methods, REACT achieves the best performance. Furthermore, by incorporating residual-adaptive contextual tuning with meta learning, REACT outperforms the state-of-the-art OC-MAML, which focuses solely on meta-learning.

5.3 Ablation Studies

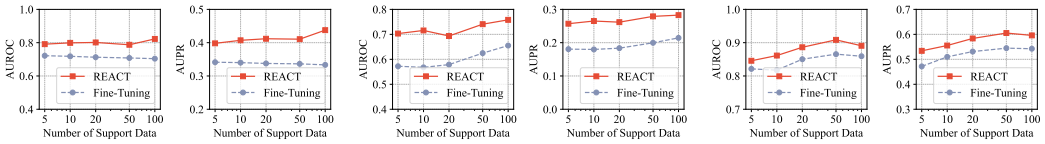
We crafted three ablated versions of REACT by removing each key component and evaluating the impact on performance. **(1) REACT w/o fine-tuning**: We disable the fine-tuning during inference and use the merged weights from meta weights and the hypernetwork's prediction to do the inference directly. **(2) REACT w/o context**: We replace the context with randomly-generated embeddings. **(3) REACT w/o regularization**: We remove the regularization on the hypernetwork's prediction. Table 3 presents the experiment results. The first two rows serve as baselines for comparison with our ablated methods. The results demonstrate that every component in our framework contribute to the overall performance improvement. Among all, enabling fine-tuning and applying regularization have the most significant impact. The contribution of context encoding can be attributed to its selection of contexts, which uses time and service descriptions represented by pretrained text embeddings for the datasets respectively. This effectively helps the hypernetwork recognize new tasks through modeling task similarity.

5.4 Sensitivity Analyses

(1) Number of Support Samples: We vary the number of support samples for each new task (i.e., k) from 5 to 100 and compare the performance of REACT with the fine-tuning baseline. As shown in Figure 2, REACT consistently outperforms the fine-tuning baseline across all datasets. This

Table 3: Ablation study. We crafted three ablated version by removing each key designs from our framework. The results demonstrate that every component in our framework contribute to the overall performance improvement.

Method	AnoShift		Malware		NSL-KDD	
	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR
w/o adaptation ($\mathcal{A}$)	0.7110	0.3204	0.5165	0.1644	0.7420	0.5110
fine-tuning	0.7039	0.3333	0.5678	0.1797	0.8175	0.5098
REACT w/o fine-tuning	0.7873	0.3977	0.7159	0.2696	0.7282	0.4838
REACT w/o context	0.7922	0.3888	0.6892	0.2426	0.8597	0.5522
REACT w/o regularization	0.6541	0.3379	0.6326	0.2001	0.8045	0.4705
REACT	**0.8226**	**0.4376**	**0.7252**	**0.2750**	**0.8673**	**0.5559**



(a) AnoShift (b) Malware (c) NSL-KDD

Figure 2: Sensitivity analysis: number of support samples (k). REACT consistently outperforms the fine-tuning baseline across different k, showing its robustness in data-scarce scenario.

demonstrates its robustness in data-scarce scenarios and highlights its ability to efficiently leverage available data for fast adaptation. **(2) Contamination on Training Data:** We evaluate our system against contamination in the training data when applying to AutoEncoder and DeepSVDD models on AnoShift and Malware respectively, which are both unsupervised methods. We note that GOAD is a semi-supervised method trained solely on benign data (as applied to the NSL-KDD dataset) so this evaluation is trivial to it. We fix the number of benign samples while varying the ratio of positive samples (e.g., intrusion attacks, malware). Table 4 shows the AUROC scores with contamination levels ranging from 1% to 20%. REACT consistently achieves better performance across different contamination rate than all compared methods, showing that it is robust to noise in training data.

5.5 Parameter-Efficient Fine-Tuning

Our framework supports parameter-efficient fine-tuning, which is especially useful when working with large models. By incorporating adaptive weights into only a subset of the model's parameters and having the hypernetwork predict this subset of weights (i.e., partial fine-tuning Xu et al. [2023]), we can reduce the number of parameters to be fine-tuned. We conduct experiments using an AutoEncoder on the AnoShift dataset to showcase REACT's ability in parameter-efficient fine-tuning. Specifically, we predict adaptive weights for either the two symmetric linear layers closest to the input (denoted as **REACT-Inner**) or those closest to the latent representations (denoted as **REACT-Outer**). Full fine-tuning of REACT is denoted as **REACT-Full**. The results are shown in Figure 3. Both methods achieve better performance compared to the baselines, although they slightly underperform compared to REACT-Full which fine-tunes all layers. Notably, REACT-Inner achieves a 5.75% higher AUROC while updating 94.3% fewer parameters compared to conventional full fine-tuning, highlighting its efficiency.

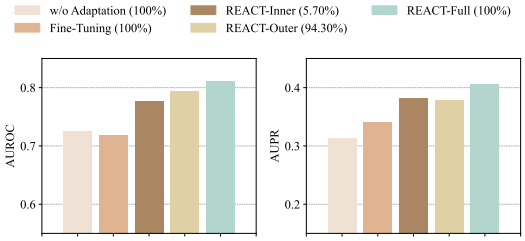


Figure 3: Parameter-Efficient Fine-Tuning.

Table 4: Sensitivity analysis: contamination on training data. REACT is robust against noise in training data, achieving the highest AUROC scores across different contamination levels.

Method	Malware				AnoShift			
	1%	5%	10%	20%	1%	5%	10%	20%
w/o adaptation ($\mathcal{A}$)	0.5055	0.5126	0.5165	0.5671	0.7635	0.7525	0.7110	0.6335
training from scratch	0.3659	0.3679	0.3659	0.3766	0.8177	0.7907	0.7398	0.7397
fine-tuning	0.5500	0.5448	0.5678	0.5799	0.8122	0.7651	0.7039	0.6050
continual learning	0.5618	0.5590	0.5879	0.5903	0.6832	0.5724	0.6087	0.4527
CL w/ experience replay	0.5816	0.5849	0.6022	0.6021	0.7340	0.6693	0.6144	0.5770
OC-MAML	0.6834	0.6879	0.6779	0.6868	0.8273	0.8034	0.7770	0.7550
REACT	**0.7249**	**0.7384**	**0.7252**	**0.7186**	**0.8319**	**0.8126**	**0.8226**	**0.7753**

5.6 Case Study

To understand how well the hypernetwork leverages the contextual information, we do a case study by calculating the cosine similarity of the weights generated by the trained hypernetwork in our framework. We use the AnoShift dataset as an example, where shifts in the data distribution occur over time. We specifically visualize the adaptive biases of the last layer in the AutoEncoder. Figure 4 shows how these adaptive weights vary across different months from 2006 to 2015, with warmer colors indicating higher similarity. The diagonal line of high cosine similarity values, shown in red, indicates that the generated weights for each month are similar to those of neighboring months. This suggests that the hypernetwork effectively captures the temporal dynamics of the data, smoothly adapting the model weights as the data distribution shifts over time.

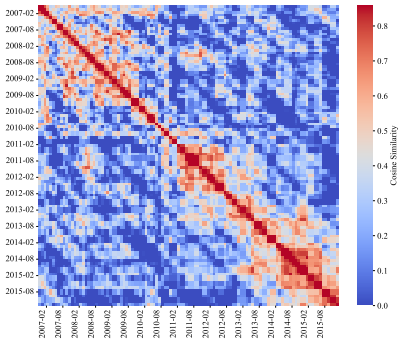


Figure 4: Cosine similarity of generated weights by the hypernetwork.

6 Conclusions

Our work sheds lights on how to approach the distribution shift problem—from both pretraining and fine-tuning perspective. We propose a novel framework, REACT, that decomposes the weights of a neural network model into the sum of a meta component and an adaptive component, following a meta-learning paradigm to train these two components. By integrating a hypernetwork, REACT primes adaptive weights close to its optimal values, enabling adaptation to new distributions with a small number of data and few gradient steps. The framework is model-agnostic, generally applicable to arbitrary neural networks. It works effectively with unlabeled and imbalanced data, making it broadly applicable to various neural network models and objectives in cybersecurity.

Broader Impact. While focused on cybersecurity, the methods developed in our research can be adapted to other fields facing similar challenges, such as finance (Gibbs and Candes [2021], Guo et al. [2023], Wang et al. [2024]) and healthcare (Schrouff et al. [2022], Ji et al. [2023], Klarner et al. [2023]). Our study provides insights for studies in the general machine learning community, fostering a more comprehensive understanding of adaptation by showcasing the investigation in cybersecurity domain.

Limitations and Future Work. While our framework shows promising results, it has limitations in scenarios with significant distribution shifts over long period of time. These limitations may stem from the model's reliance on an outdated meta model, which may not effectively generalize well to new distributions that are vastly different from the data for pretraining. As a future direction, we plan to explore incorporating a lightweight mechanism for updating the meta model within our framework. A potential solution could involve applying aggregation of the predicted adaptive weights into the meta model. In this way, we could enhance the framework's ability to continuously adapt to evolving distributions.

References

Petar Stojanov, Zijian Li, Mingming Gong, Ruichu Cai, Jaime Carbonell, and Kun Zhang. Domain adaptation with invariant representation learning: What transformations to learn? *Advances in Neural Information Processing Systems*, 34:24791–24803, 2021.

Korawat Tanwisuth, Xinjie Fan, Huangjie Zheng, Shujian Zhang, Hao Zhang, Bo Chen, and Mingyuan Zhou. A prototype-oriented framework for unsupervised domain adaptation. *Advances in Neural Information Processing Systems*, 34:17194–17208, 2021.

Kendrick Shen, Robbie M Jones, Ananya Kumar, Sang Michael Xie, Jeff Z HaoChen, Tengyu Ma, and Percy Liang. Connect, not collapse: Explaining contrastive learning for unsupervised domain adaptation. In *International conference on machine learning*, pages 19847–19878. PMLR, 2022.

Aodong Li, Chen Qiu, Marius Kloft, Padhraic Smyth, Maja Rudolph, and Stephan Mandt. Zero-shot anomaly detection via batch normalization. *Advances in Neural Information Processing Systems*, 36, 2023a.

Ze Wang, Yipin Zhou, Rui Wang, Tsung-Yu Lin, Ashish Shah, and Ser Nam Lim. Few-shot fast-adaptive anomaly detection. *Advances in Neural Information Processing Systems*, 35:4957–4970, 2022.

Yuxin Li, Wenchao Chen, Bo Chen, Dongsheng Wang, Long Tian, and Mingyuan Zhou. Prototype-oriented unsupervised anomaly detection for multivariate time series. In *International Conference on Machine Learning*, pages 19407–19424. PMLR, 2023b.

David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

Arthur Gretton, Alex Smola, Jiayuan Huang, Marcel Schmittfull, Karsten Borgwardt, and Bernhard Schölkopf. Covariate shift by kernel mean matching. 2008.

Garrett Wilson and Diane J Cook. A survey of unsupervised deep domain adaptation. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(5):1–46, 2020.

Abolfazl Farahani, Sahar Voghoei, Khaled Rasheed, and Hamid R Arabnia. A brief review of domain adaptation. *Advances in data science and information engineering: proceedings from ICDATA 2020 and IKE 2020*, pages 877–894, 2021.

Xiaofeng Liu, Chaehwa Yoo, Fangxu Xing, Hyejin Oh, Georges El Fakhri, Je-Won Kang, Jonghye Woo, et al. Deep unsupervised domain adaptation: A review of recent advances and perspectives. *APSIPA Transactions on Signal and Information Processing*, 11(1), 2022.

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79:151–175, 2010.

Roberto Jordaney, Kumar Sharad, Santanu K Dash, Zhi Wang, Davide Papini, Ilia Nouretdinov, and Lorenzo Cavallaro. Transcend: Detecting concept drift in malware classification models. In *26th USENIX security symposium (USENIX security 17)*, pages 625–642, 2017.

Feargus Pendlebury, Fabio Pierazzi, Roberto Jordaney, Johannes Kinder, and Lorenzo Cavallaro. {TESSERACT}: Eliminating experimental bias in malware classification across space and time. In *28th USENIX security symposium (USENIX Security 19)*, pages 729–746, 2019.

Federico Barbero, Feargus Pendlebury, Fabio Pierazzi, and Lorenzo Cavallaro. Transcending transcend: Revisiting malware classification in the presence of concept drift. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 805–823. IEEE, 2022.

Sumohana Channappayya, Bheemarjuna Reddy Tamma, et al. Augmented memory replay-based continual learning approaches for network intrusion detection. *Advances in Neural Information Processing Systems*, 36, 2024.

Peng Jia, Shaofeng Cai, Beng Chin Ooi, Pinghui Wang, and Yiyuan Xiong. Robust and transferable log-based anomaly detection. *Proceedings of the ACM on Management of Data*, 1(1):1–26, 2023.

João Carvalho, Mengtao Zhang, Robin Geyer, Carlos Cotrini, and Joachim M Buhmann. Invariant anomaly detection under distribution shifts: a causal perspective. *Advances in Neural Information Processing Systems*, 36, 2023.

Dongmin Kim, Sunghyun Park, and Jaegul Choo. When model meets new normals: Test-time adaptation for unsupervised time-series anomaly detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 13113–13121, 2024.

Shurui Gui, Xiner Li, and Shuiwang Ji. Active test-time adaptation: Theoretical analyses and an algorithm. *arXiv preprint arXiv:2404.05094*, 2024.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

Kaize Ding, Qinghai Zhou, Hanghang Tong, and Huan Liu. Few-shot network anomaly detection via cross-network meta-learning. In *Proceedings of the Web Conference 2021*, pages 2448–2456, 2021.

Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. *arXiv preprint arXiv:2106.04489*, 2021.

Jiayun Zhang, Shuheng Li, Haiyu Huang, Zihan Wang, Xiaohan Fu, Dezhi Hong, Rajesh K Gupta, and Jingbo Shang. How few davids improve one goliath: Federated learning in resource-skewed edge computing environments. In *Proceedings of the ACM on Web Conference 2024*, pages 2976–2985, 2024a.

David Bonet, Daniel Mas Montserrat, Xavier Giró-i Nieto, and Alexander G Ioannidis. Hyperfast: Instant classification for tabular data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 11114–11123, 2024.

Hai Zhang, Chunwei Wu, Guitao Cao, Hailing Wang, and Wenming Cao. Hypereditor: Achieving both authenticity and cross-domain capability in image editing via hypernetworks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 7051–7059, 2024b.

A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Marius Drăgoi, Elena Burceanu, Emanuela Haller, Andrei Manolache, and Florin Brad. Anoshift: A distribution shift benchmark for unsupervised anomaly detection. *Neural Information Processing Systems NeurIPS, Datasets and Benchmarks Track*, 2022.

Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A Ghorbani. A detailed analysis of the kdd cup 99 data set. In *2009 IEEE symposium on computational intelligence for security and defense applications*, pages 1–6. Ieee, 2009.

Ngoc Anh Huynh, Wee Keong Ng, and Kanishka Ariyapala. A new adaptive learning algorithm and its application to online malware detection. In *Discovery Science: 20th International Conference, DS 2017, Kyoto, Japan, October 15–17, 2017, Proceedings 20*, pages 18–32. Springer, 2017.

Charu C Aggarwal and Charu C Aggarwal. *An introduction to outlier analysis*. Springer, 2017.

Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep one-class classification. In *International conference on machine learning*, pages 4393–4402. PMLR, 2018.

Liron Bergman and Yedid Hoshen. Classification-based anomaly detection for general data. *arXiv preprint arXiv:2005.02359*, 2020.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019.

Ahmed Frikha, Denis Krompaß, Hans-Georg Köpken, and Volker Tresp. Few-shot one-class classification via meta-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 7448–7456, 2021.

Lingling Xu, Haoran Xie, Si-Zhao Joe Qin, Xiaohui Tao, and Fu Lee Wang. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment. *arXiv preprint arXiv:2312.12148*, 2023.

Isaac Gibbs and Emmanuel Candes. Adaptive conformal inference under distribution shift. *Advances in Neural Information Processing Systems*, 34:1660–1672, 2021.

Yue Guo, Chenxi Hu, and Yi Yang. Predict the future from the past? on the temporal data distribution shift in financial sentiment classifications. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1029–1038, 2023.

Chen Wang, Ziwei Fan, Liangwei Yang, Mingdai Yang, Xiaolong Liu, Zhiwei Liu, and Philip Yu. Pre-training with transferable attention for addressing market shifts in cross-market sequential recommendation. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2970–2979, 2024.

Jessica Schrouff, Natalie Harris, Sanmi Koyejo, Ibrahim M Alabdulmohsin, Eva Schnider, Krista Opsahl-Ong, Alexander Brown, Subhrajit Roy, Diana Mincu, Christina Chen, et al. Diagnosing failures of fairness transfer across distribution shift in real-world medical settings. *Advances in Neural Information Processing Systems*, 35:19304–19318, 2022.

Xiayan Ji, Hyonyoung Choi, Oleg Sokolsky, and Insup Lee. Incremental anomaly detection with guarantee in the internet of medical things. In *Proceedings of the 8th ACM/IEEE Conference on Internet of Things Design and Implementation*, pages 327–339, 2023.

Leo Klarner, Tim GJ Rudner, Michael Reutlinger, Torsten Schindler, Garrett M Morris, Charlotte Deane, and Yee Whye Teh. Drug discovery under covariate shift with domain-informed prior distributions over functions. In *International Conference on Machine Learning*, pages 17176–17197. PMLR, 2023.