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

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

Junshen Xu
Amazon Web Services
Boston, Massachusetts, USA
jsxu@amazon.com

Bugra Can
Amazon Web Services
New York, New York, USA
bugracan@amazon.com

Yi Fan
Amazon Web Services
New York, New York, USA
fnyi@amazon.com

Abstract

Web and mobile systems show constant distribution shifts due to the evolvement of services, users, and threats, severely degrading the performance of threat detection models trained on prior distributions. Fast model adaptation with minimal new data is essential for maintaining reliable security measures. A key challenge in this context is the lack of ground truth, which undermines the ability of existing solutions to align classes across shifted distributions. Moreover, the limited new data often fails to represent the underlying distribution, providing sparse and potentially noisy information for adaptation. In this paper, we propose REACT, a novel framework that adapts the model using a few unlabeled data and contextual insights. We leverage the inherent data imbalance in threat detection and meta-train weights on diverse unlabeled subsets to generalize common patterns across distributions, eliminating the reliance on labels for alignment. REACT decomposes a neural network into two complementary components: meta weights as a shared foundation of general knowledge, and residual adaptive weights as adjustments for specific shifts. To compensate for the limited availability of new data, REACT trains a hypernetwork to predict adaptive weights based on data and contextual information, enabling knowledge sharing across distributions. The meta weights and the hypernetwork are updated alternately, maximizing both generalization and adaptability. Extensive experiments across multiple datasets and models demonstrate that REACT improves AUROC by 14.85% over models without adaptation, outperforming the state-of-the-art.

CCS Concepts

• **Computing methodologies** → **Learning under covariate shift**; **Anomaly detection**; **Neural networks**.

Keywords

Threat detection; Distribution shift; Meta learning; Hypernetwork

*Work done during internship at Amazon Web Services.

WWW '25, Sydney, NSW, Australia

ACM ISBN 979-8-4007-1274-6/25/04
https://doi.org/10.1145/3696410.3714577

ACM Reference Format:
Jiayun Zhang, Junshen Xu, Bugra Can, and Yi Fan. 2025. REACT: Residual-Adaptive Contextual Tuning for Fast Model Adaptation in Threat Detection. In *Proceedings of the ACM Web Conference 2025 (WWW '25), April 28-May 2, 2025, Sydney, NSW, Australia.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3696410.3714577

1 Introduction

Threat detection is an essential component in web and mobile systems that identifies malicious activities across networks, endpoints, and software, defending against security risks. Cyber environments undergo continual distribution shifts due to various factors, including users joining and leaving the network, user behavior changes, and software updates. These shifts severely degrade the performance of threat detection models trained on prior distributions. For example, 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. Threat detection models trained on typical traffic are less effective in these scenarios. Adapting model weights after minimal exposure to new data is crucial for timely and effectively identifying threats in dynamic and adversarial environments.

A critical challenge in managing distribution shifts in threat detection is the lack of ground truth, as this requires user reports or detailed inspections by domain experts. Traditional methods [5, 18, 48, 61, 62, 66, 74], which rely on labels from either source or target domains to align classes across shifted distributions, fail to address this scenario. Moreover, the limited new data often does not fully represent the underlying distribution, providing sparse and potentially noisy information for adaptation. Existing methods [16, 38, 40, 43] fine-tune models exclusively on these limited data. They may exhibit large variations in performance and are sensitive to the quality of the observed data, often prone to overfitting [29, 72, 79, 80]. The problem setting is illustrated in Figure 1.

Due to label scarcity, threat detection often exhibits extreme data imbalance, with a few suspicious activities (e.g., unauthorized access attempts, anomalous traffic, malware) hidden among a vast majority of benign behaviors. This imbalance presents an opportunity to address distribution shifts. Instead of learning the exact benign and suspicious behaviors and matching them across domains, models could learn to generalize majority patterns for various distributions.

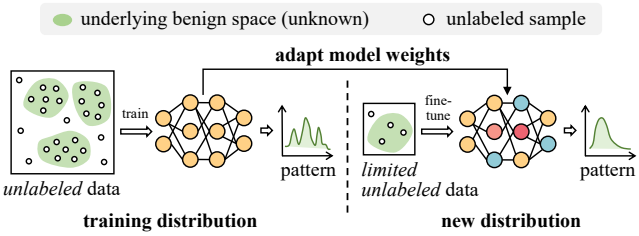

Figure 1: Illustration of our problem setting.

Samples that deviate from such patterns are regarded as potential threats [1, 28]. Therefore, we apply meta-learning on diverse unlabeled subsets dominated by benign samples. These subsets are sampled according to underlying shifts, e.g., time-based sampling for temporal shifts. After being meta-trained on various scenarios, the model can quickly adapt to new distributions by adjusting the learned pattern using a few unlabeled samples.

To compensate for the limited new data, we utilize contextual information to model correlations among distributions. For example, in network intrusion detection, a newly deployed service like a microservices-based API can find similarities with common web servers (e.g., Apache and Tomcat) based on their intrinsic characteristics of services and deployment environments (e.g., service configuration, user role). By modeling these contexts, we can recognize relationships across distributions and transfer knowledge from mature systems to newly deployed ones.

Building on these insights, we introduce REACT (Residual-Adaptive Contextual Tuning), a novel framework adapting models with a few unlabeled new data and contextual insights. Given a neural network, REACT decomposes its weights into the sum of two components: meta weights, which are meta-trained to form a solid foundation of general knowledge and shared globally, and adaptive weights, which are the residual components fine-tuned to specific distributions. We leverage a hypernetwork [26] to generate adaptive weights based on data and contextual information. Intuitively, the hypernetwork maps its inputs onto a low-dimensional manifold within the parameter space [13, 60]. This mapping positions adaptive weights for similar contexts and data patterns close to each other, enabling knowledge transfer across different distributions. During training, REACT optimizes the meta weights and the hypernetwork alternately through meta-learning on subsets sampled according to underlying shifts. At inference, the adaptive weights are fine-tuned from the prediction given by the trained hypernetwork, while the meta weights are fixed, preserving the generalizability of the model [14, 41, 50, 69]

We theoretically analyze the convergence of REACT on linear models, showing the parameters converge at a linear rate characterized by the eigenvalues of the sample matrices and other hyperparameters in REACT. Our framework is model-agnostic, broadly applicable to various neural networks and loss functions. We evaluate REACT on three datasets with different backbone models. Compared to models without adaptation, REACT improves the AUROC by 14.85% with few fine-tuning efforts (e.g., update 1 to 10 gradient steps on 10 to 100 samples). Sensitivity analyses show that REACT is robust to variations in the number of samples, the number of fine-tuning steps, and contamination in training data. We further showcase the capability of REACT for parameter-efficient fine-tuning, achieving 5.75% higher AUROC with 94.3% fewer parameters updated compared to full fine-tuning, highlighting its efficiency. Our contributions are as follows:

- We study the problem of fast model adaptation under distribution shift in threat detection, focusing on a practical yet challenging scenario where labels are unavailable and only limited data from new distributions are observed.
- We introduce REACT, a novel adaptation framework using a few unlabeled data and contextual insights. REACT decomposes model weights into meta and adaptive components and updates them through meta-learning alternately. It employs a hypernetwork to generate adaptive weights based on data and contexts, enabling knowledge transfer across distributions.
- We establish the convergence rate of REACT through theoretical analysis. Moreover, we conduct extensive evaluations on multiple model architectures and datasets, demonstrating that REACT consistently outperforms various state-of-the-art methods.

2 Related Work

Threat Detection. Threat detection [6, 47, 64, 65, 76] aims to identify security risks in systems and networks, such as insider threat [47, 76], intrusion attack [15], malware [42], spammer [64]. Typically, the ground truths for benign and malicious activities are not available, as they require user reports or inspections by domain experts. As a result, threat detection follows unsupervised or semi-supervised approaches in anomaly detection based on different assumptions of data distribution [1, 28]. These methods assume the majority of data belong to a "normal" class, while anomalies deviate from this norm, e.g., lying in low-density regions [81], being far from normal data clusters [57], or showing high reconstruction errors from the latent space of normal data [1]. These methods are designed for static environments and are not robust to distribution shifts. Changes in data distributions can significantly degrade model performance.

Distribution Shifts in General Machine Learning. Distribution shift means the distributions of the training and testing data are different, leading to poor model generalization to unseen data [23]. To address the challenge, adaptation methods have been proposed [18, 48, 74]. We focus on works designed for unsupervised or semi-supervised scenarios due to the data specificity in threat detection. Unsupervised domain adaptation [5] is a closely related topic, which adapts models to target domains that have no labeled data. Methods include invariant representation learning [62], prototype-oriented conditional transport [66], contrastive pre-training [61]. However, these methods assume the availability of labels from source data to guide adaptation, falling short in threat detection where labels in source domains are also unavailable.

Model Adaptation in Threat Detection. Distribution shifts are observed in threat detection, such as malware detection [34], network intrusion detection [11, 75], and log anomaly detection [33]. Traditional supervised approaches [4, 34, 52] require extensive labeling, making them impractical for real-world deployment. Recent efforts have recognized this limitation and have been exploring

adaptation approaches without relying on labels. Unsupervised domain adaptation methods, like learning domain-invariant representations [10] have been extended. However, they typically require simultaneous training on source and target domains, making them less suitable for emerging domains. Test-time adaptation, such as batch normalization updates [40], energy-based models [73], and trend estimation [35], updates models during inference without gradient descent. Though efficient, they are limited to minor shifts [24] or sequential shifts that display continuous patterns [35]. To address more severe and random shift, meta-learning [19, 68] is a promising approach that trains a meta model on a variety of learning tasks, enabling adaptation to new distributions with a small amount of data. Prior works have applied meta-learning to graph neural networks [16] and autoencoders [38] for few-shot detection, and introduced prototype-oriented optimal transport for adapting models to new multivariate time-series [43]. However, these methods fine-tune models solely on limited data from new distributions, leading to variations in adaptation performance. In contrast, our method considers contextual information about shifts to understand correlations among distributions and transfer knowledge, improving adaptability.

3 Preliminaries

3.1 Problem Definition

Distribution shifts in threat detection involve changes in the probability distribution of data over time or across domains (e.g., users, services). These shifts can affect the marginal feature distribution $\mathcal{P}(x)$, the conditional distribution $\mathcal{P}(y|x)$, or both, where x is the data and y is the category. Consider a threat detection model $f(\cdot;\theta)$ trained on a dataset $\mathbf{D} = \{x_i\}_{i=1}^{N}$ drawn from a distribution $\mathcal{P}$. The dataset $\mathbf{D}$ is *unlabeled* and is dominated by samples from the benign class. The model parameters θ are optimized by minimizing a loss function $\mathcal{L}$. Common choices for $\mathcal{L}$ include reconstruction loss in autoencoders or contrastive loss in self-supervised learning. The objective is: $\theta^* = \arg\min_\theta \mathbb{E}_{x\sim\mathcal{P}}\mathcal{L}(f(x;\theta))$.

Our goal is to develop an adaptation method that updates model parameters to θ' using a few examples from the new distribution $\mathcal{P}'$. The observed dataset $\mathbf{D}'$ from distribution $\mathcal{P}'$ is unlabeled, and its size is small $|\mathbf{D}'| = k \ll |\mathbf{D}|$.

3.2 Meta-Learning

Meta-learning trains models that can quickly adapt to new tasks using only a few examples. A task $\mathcal{T}_i$ is defined as an independent learning problem with a dataset following a specific distribution $\mathcal{P}_i$ and a learning objective which is to find the optimal parameters θ_i^* that minimize the expected loss $\theta_i^* = \text{argmin}_\theta \mathbb{E}_{x\sim\mathcal{P}_i}\mathcal{L}(f(x;\theta))$. The dataset $\mathbf{D}_i$ for task $\mathcal{T}_i$ is divided into a support set $\mathbf{D}^i_{\text{support}}$ and a query set $\mathbf{D}^i_{\text{query}}$. $\mathbf{D}^i_{\text{support}}$ is used to fine-tune the model to learn task-specific parameters for $\mathcal{T}_i$, while $\mathbf{D}^i_{\text{query}}$ evaluates how well the model generalizes the learned task-specific knowledge.

One of the most representative algorithm is Model-Agnostic Meta-Learning (MAML) [19], which optimizes the initial model parameters θ so that after fine-tuning, the model performs well across various tasks, minimizing the average loss. For each task $\mathcal{T}_i$, the parameters are fine-tuned using the support set $\mathbf{D}^i_{\text{support}}$: $\theta_i^* = \text{argmin}_\theta \sum_{x\in\mathbf{D}^i_{\text{support}}} \mathcal{L}(f(x;\theta))$. The weight initialization θ is optimized using the query sets:

$$\theta^* = \text{argmin}_\theta \sum_{\mathcal{T}_i} \sum_{x\in\mathbf{D}^i_{\text{query}}} \mathcal{L}(f(x;\theta_i^*)).$$

During inference, the model is fine-tuned on a few samples from the new distribution, and then applied to all testing samples.

3.3 Hypernetwork

A hypernetwork [26] is a neural network that predicts the weights of another neural network (i.e., target network). By training a single hypernetwork to predict weights across multiple tasks rather than optimizing each one independently, hypernetworks offer a parameter-efficient solution for model adaptation. It has shown effective in improving learning efficiency through parameter sharing [2, 8, 49, 77, 78]. Let h represent the hypernetwork with parameters ϕ, and let f denote the target network. Given a representation $\mathbf{V}_i$ for describing task $\mathcal{T}_i$, the hypernetwork generates the model weights $\theta_i = h(\mathbf{V}_i;\phi)$, which are loaded into f for the downstream task. Given multiple tasks $\mathcal{T}_i$ and the corresponding task representations $\mathbf{V}_i$, the learning objective is to optimize the hypernetwork's parameters ϕ to minimize the loss $\mathcal{L}$ across these tasks:

$$\phi^* = \text{argmin}_\phi \sum_{\mathcal{T}_i} \sum_{x\in\mathbf{D}_i} \mathcal{L}[f(x;h(\mathbf{V}_i;\phi))].$$

4 The REACT Framework

We approach the problem from both meta-training and fine-tuning perspectives. Through meta-training, the model establishes a strong, generalizable foundation that can be applied to most scenarios. Then, through fine-tuning, the model weights are slightly adjusted for specific shifts. We propose a framework that decomposes model weights into two components and alternately optimizes them to address both perspectives. Appendix A provides the pseudo-code.

4.1 Weight Decomposition

Given a neural network, we decompose its weights into two complementary components: meta weights θ_{meta} and adaptive weights θ_{adapt}. Meta weights capture global patterns that are common across different distributions, representing the core knowledge acquired during meta-learning. Adaptive weights, on the other hand, serve as a small "residual" component that allows the model to be fine-tuned to the unique characteristics of specific data distributions, while still leveraging the global patterns encoded in the meta weights. The full model weights are then formed by adding the two components together: $\theta = \theta_{\text{meta}} + \theta_{\text{adapt}}$. By applying a small residual update to the pretrained meta weights, the model can adapt to new distributions without overwriting the essential pretrained knowledge.

4.2 Residual-Adaptive Weight Generation with Hypernetwork

We incorporate a hypernetwork to generate adaptive weights based on data characteristics and contextual information. The architecture of the hypernetwork is presented in Figure 2.

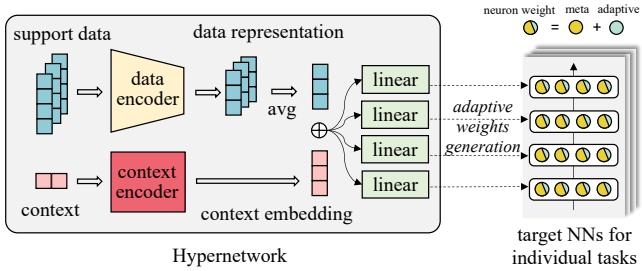

Figure 2: Architecture of the proposed hypernetwork.

Data Encoding. Our hypernetwork includes a data 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 producing the weights for a corresponding layer in the target network.

Context Encoding. 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 [70] generating embeddings. Details about context modeling for different tasks can be found in Section 6.1.

4.3 Alternating Optimization

We design an alternating optimization scheme to iteratively update the meta weights and the hypernetwork. This approach balances the learning dynamics and prevents mutual interference between the two components. Figure 3 illustrates the process.

Task Sampling for Meta Learning. To let the model learn how to adapt to new distributions, we first create a diverse set of tasks that reflect the expected variations in the application. We sample tasks from training set by simulating the underlying data shifts. For instance, if the goal of adaptation is to address distribution shifts over time, the data can be grouped according to temporal factors such as day or month, with each time period forming a separate task. If the focus is on handling shifts across different users, the data can be grouped by users, with each user forming a task.

Update of Meta Component. Let θ^i_{adapt} denote the adaptive weights of task $\mathcal{T}_i$. In each iteration, we begin by updating the meta weights. We sample a set of tasks $\{\mathcal{T}_i\}_{i=1}^M$ and contextual information $\{c_i\}_{i=1}^M$. The hypernetwork $h(\cdot;\phi)$ is fixed and used to generate adaptive weights, $\theta^i_{\text{adapt}} = h(\mathbf{D}^i_{\text{support}}, c_i; \phi)$. The generated adaptive weights are then fine-tuned to derive the optimal model weight $\theta^{i,*}_{\text{adapt}}$ for task $\mathcal{T}_i$ by minimizing the empirical loss over the support set $\mathbf{D}^i_{\text{support}}$:

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

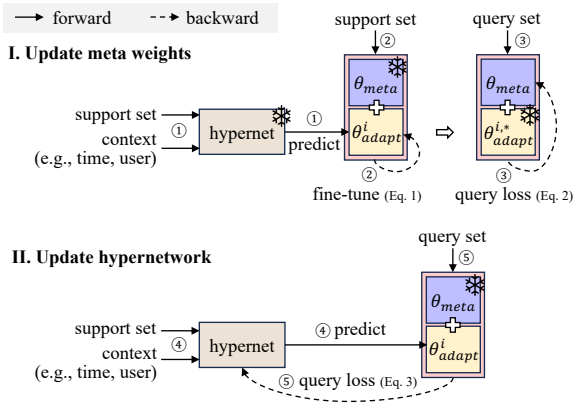

Figure 3: Alternating optimization in REACT. In each training iteration, we sample a set of tasks to update the meta weights, then sample another set to train the hypernetwork.

We then 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:

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

Update of Hypernetwork. 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_h be the learning rate for updating the hypernetwork. The weight update of hypernetwork after one gradient step is as follows:

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

Regularization. We apply L2 regularization to the adaptive weights generated by the hypernetwork, encouraging them to act as residuals to the globally shared meta weights. The query loss for optimizing the hypernetwork, denoted as $\mathcal{L}^i_{\text{query}}$, is combined with the regularization as $\mathcal{L} = \mathcal{L}^i_{\text{query}} + \lambda \|\theta^i_{\text{adapt}}\|_2^2$, where λ is the hyperparameter to control the regularization strength.

4.4 Adapting to New Distributions

When doing inference on a new distribution $\mathcal{P}_j$, the meta weights and the hypernetwork are fixed. This ensures the pre-trained knowledge is not "forgotten" during fine-tuning [14, 41, 50, 69], preserving generalizability of the model. A small number of support data $\mathbf{D}^j_{\text{support}}$ from $\mathcal{P}_j$ along with its contextual information c_j are fed into the hypernetwork to predict the adaptive weights $\theta^j_{\text{adapt}} = h(\mathbf{D}^j_{\text{support}}, c_j; \phi)$. With this initialization, the adaptive weights are then fine-tuned on $\mathbf{D}^j_{\text{support}}$ following Equation 1. Finally, the two parts of the weights are merged by summing them as if there is only one target network. The merged weights are used for inference on data from the new distribution $\mathcal{P}_j$.

Table 1: Experiment configurations and dataset statistics after preprocessing.

Dataset	# Train/Test	# Train/Test tasks	k	Shift by	Model
AnoShift	1.3M / 1.8M	50 / 110	100	Time	AutoEncoder
NSL-KDD	28K / 6K	8 / 6	10	Service	GOAD
Malware	15K / 5K	36 / 12	10	Time	DeepSVDD

5 Analysis

We provide convergence analysis of REACT on linear models. Let X^i be the matrix whose rows are the samples from the dataset of task $i \in \{1, ..., M\}$, i.e., $\mathbf{D}_i$. The data matrix X^i can be split into support set X_s^i and query set X_q^i. We assume that the relevant datasets are sampled at the beginning of the algorithm. Given linear model[1]

$$h(X;\phi) = X\phi, \quad f(X;\theta_{\text{meta}},\theta_{\text{adapt}}) = X(\theta_{\text{meta}} + \theta_{\text{adapt}}), \quad (4)$$

Theorem 1 provides convergence guarantees for REACT.

Theorem 1. *Consider REACT on the linear model in* (4) *with Eq.* (1) *being solved exactly. Let X_s^i and X_q^i satisfy $(X_s^i)^\top X_s^i = (X_q^i)^\top X_q^i = \sigma_i I$ for each task $i \in \{1, .., M\}$, where σ_i are the variances and I is the identity matrix. Learning rates are chosen as $\eta_{meta} < 1/\sum_{i=1}^M \sigma_i\lambda/(\sigma_i+\lambda)$ and $\eta_h < 1/\max\left(\sum_{j=1}^{n_h}\sigma_j(\sigma_j+\lambda), \|\mathbf{X}_s\|\right)$ where $\mathbf{X}_s = \sum_{j=1}^M \sigma_j (X_s^j)^\top$. Then, for any $\varepsilon > 0$, there exists*

$$K = O\left(\log_{1/\rho_{meta}}(1/\varepsilon) + \log_{1/\rho_h}(1/\varepsilon)\right)$$

for $\rho_{meta} = 1-\eta_{meta}\sum_{i=1}^M \sigma_i\lambda/(\sigma_i+\lambda)$ and $\rho_h = 1-\eta_h\sum_{j=1}^M \sigma_j(\sigma_j+\lambda)$ such that the K-iteration of Algorithm 1 satisfies

$$\|\theta^K - \theta^*\| \le \varepsilon, \qquad \text{and} \qquad \|\phi^K - \phi^*\| \le \varepsilon,$$

where θ^* and ϕ^* are stationary points of the algorithm.

The proof is provided in Appendix E. Our results suggest that θ_{meta} and ϕ converge to stationary points at a linear rate which can be characterized based on the eigenvalues of the sample matrices in each task and hyperparameters considered in REACT.

6 Experiments

6.1 Experiment Setup

Datasets and Backbone Models. Our evaluation focuses on two key applications in cybersecurity, network intrusion detection and malware detection, and targets both temporal and domain shifts. REACT is compatible with various neural network architectures. To assess its performance across different models, we employ three representative architectures, AutoEncoder (AE) [1], DeepSVDD (DSVDD) [57], and GOAD [7], paired with the following datasets:

- **AnoShift** [17] is a benchmark for network intrusion detection under distribution shifts. It collects traffic logs from a university network between 2006 and 2015. Data shifts occur over time due to reasons such as user behavior changes and software updates. Each sample has 15 features (9 numerical and 6 categorical) and a label of whether it is an attack. We use the train-test split provided by the dataset, including training subsets from 2006 to 2010 and test subsets from 2006 to 2015. Each month is regarded as a separate task. AutoEncoder is used as the backbone model.
- **Malware** [30] contains executables collected between 2010 and 2014 from VirusShare[2], an online malware analysis platform. Data shifts happen over time. Each executable has 482 counting features and a risk score (ranging from 0 to 1) indicating the probability of it being malware. These risk scores are converted to binary labels using thresholding, with executables labeled as malicious if the score is greater than 0.6 and benign if the score is less than 0.4 [30]. Following previous work [40], the dataset is split into training data from 2011 to 2013, validation data from 2010, and testing data from 2014. Each month is treated as a separate task. DeepSVDD is used as the backbone model.
- **NSL-KDD** [67] is another dataset for evaluating network intrusion detection. Each sample has 40 attributes describing the network traffic, with 6 categorical and 34 numerical features. We simulate domain shifts by grouping data according to services (e.g., HTTP, Telnet) and randomly assigning half of the services as training tasks and the other half as test tasks. We use the official train-test split provided by the dataset and remove services not selected for the respective splits. Besides, services with fewer than 20 benign samples are excluded to ensure sufficient unseen data for testing. GOAD is used as the backbone model.

We sample the datasets to form a 10% ratio of threat samples for both training and testing. In Section 6.4, we vary this ratio to test the robustness of REACT to the contamination of training data. For the NSL-KDD dataset, since GOAD is a semi-supervised method that trains only on benign data, we remove attack samples from the training set. Table 1 summarizes the statistics and configurations of the datasets and backbone models in the experiments. Further details on the backbone models are provided in Appendix B.

Baselines. We compare REACT with unsupervised methods from the anomaly detection benchmark [28], including linear and statistical models: **ECOD** [45], **COPOD** [44], **OCSVM** [58]; distance- and proximity-based methods: **LOF** [9], **KNN** [55]; ensemble methods: **LODA** [53], **IForest** [46]; and neural networks: **AE** [1], **DSVDD** [57], **LUNAR** [22]. These methods assume static environments and do not account for distribution shifts. In addition, we compare REACT with training from scratch, fine-tuning strategies, and state-of-the-art model adaptation methods. Brief descriptions are as follows:

- **Static model**: The model is trained on the training data and directly tested on each test task, i.e., the pretrained model.
- **Train-from-scratch**: For each task, a model is trained from scratch using k samples and is used for evaluation.
- **Fine-tuning**: For each task, the model is fine-tuned using k samples from the task based on the pretrained model.
- **Continual learning**: Starting from pretrained model, we sequentially fine-tune the model using k samples from each task.
- **Experience Replay (ER)** [12] is a method to mitigate catastrophic forgetting in continual learning. We maintain a memory buffer to store historical data. In each fine-tuning iteration, we sample a batch from this buffer and compute its loss. This loss is then weighted and combined with the loss from the new batch.
- **ACR** [40] is a zero-shot adaptation which adopts meta-learning to train the model and update the batch normalization (BN) layers

[1]We note that we abuse the notation and set $h(X, c_i;\phi) = h(X;\phi)$, i.e., the context information is part of the input data

[2]https://virusshare.com/

Table 2: Main experiment results (averaged over 5 runs). The left sub-table reports the performance of static methods, while the right focuses on fine-tuning and adaptation across three backbone models. REACT consistently outperforms all other methods. $|\mathbf{D}'_i|$ denotes the number of samples observed from each test task for gradient updates[3].

Method	AnoShift		Malware		NSL-KDD	
	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR
KNN [55]	0.6714	0.4062	0.2732	0.1040	0.5323	0.2956
LOF [9]	0.6107	0.2873	0.2781	0.1101	0.4150	0.1827
OCSVM [58]	0.6903	0.3157	0.3880	0.1190	0.6649	0.3492
IForest [46]	0.6658	0.2830	0.2660	0.0722	0.7809	0.4798
LODA [53]	0.5723	0.2111	0.5190	0.1368	0.5207	0.2532
AE [1]	0.7110	0.3204	0.3789	0.1156	0.6057	0.2678
DSVDD [57]	0.7716	0.3895	0.5165	0.1644	0.6006	0.2848
COPOD [44]	0.7664	0.3831	0.4450	0.1102	0.7849	0.4471
ECOD [45]	0.7461	0.3727	0.5403	0.1390	0.8100	0.4706
LUNAR [22]	0.4449	0.2450	0.2719	0.0880	0.5243	0.2350

Method	$\lvert\mathbf{D}'_i\rvert$	AnoShift		Malware		NSL-KDD	
		AUROC	AUPR	AUROC	AUPR	AUROC	AUPR
train-from-scratch	all	0.7681	0.3689	0.6010	0.1822	0.9308	0.7107
static model	-	0.7110	0.3204	0.5165	0.1644	0.7420	0.5110
train-from-scratch	k	0.7398	0.3398	0.3659	0.1337	0.7382	0.4219
fine-tuning	k	0.7039	0.3333	0.5678	0.1797	0.8175	0.5098
continual learning	k	0.6087	0.3063	0.5879	0.1873	0.8285	0.5188
ER [12]	k	0.6144	0.2996	0.6022	0.1932	0.8356	0.5114
ACR [40]	-	0.7634	0.3753	0.5798	0.1794	0.7513	0.4658
OC-MAML [20]	k	0.7770	0.3811	0.6779	0.2334	0.8547	0.5504
REACT (ours)	k	**0.8226**	**0.4376**	**0.7252**	**0.2750**	**0.8673**	**0.5559**

with the batch statistics during inference. We add a BN layer after each linear or convolutional layer in the model.

- **OC-MAML** [20] is a few-shot one-class classification method. It extends MAML by modifying the episodic data sampling strategy. It forms one-class support sets to optimize the meta model.

Training and Adaptation Configurations. The size of support data k is set based on the data quantity, with $k = 100$ for AnoShift and $k = 10$ for Malware and NSL-KDD. The size of query data varies proportionally, with 1000 for AnoShift and 100 for Malware and NSL-KDD. In each meta-training iteration, we sample $M = 5$ tasks for Malware and NSL-KDD, and $M = 1$ for AnoShift. The number of fine-tuning epochs E is determined by the convergence rate of the learning task, with $E = 10$ epochs for AnoShift and Malware, and $E = 1$ for NSL-KDD due to its faster convergence. Regularization parameter $\lambda = 10$.

Choices of Contexts. For AnoShift and Malware whose shifts occur over time, we use time index as the context, which is modeled by positional encoding [70] to generate contextual embedding for each task. For NSL-KDD dataset whose shifts occur across services, we first feed these services names to GPT-4 [51] with the prompt "please briefly describe each of these web services, including the normal and anomalous patterns". The use of a large language model (LLM) is to reduce the reliance on domain experts. The LLM choice is generic, and other advanced models may be used. Then, we use Sentence Transformer[4] to generate embeddings for the descriptions.

Evaluation Metrics. For each test task, we adapt the model and evaluate its performance using AUROC and AUPR scores. All experiments are repeated for five times with the same set of random seeds, and the results are averaged across all test tasks and runs.

6.2 Main Results and Analysis

The results are presented in Table 2, where the left sub-table shows the performance of static methods and the right one focuses on fine-tuning and adaptation methods across three backbone models. The static methods (left table) generally show lower performance than models with adaptation (right table), highlighting the negative impact of distribution shifts on model performance. The right table also includes the performance of train-from-scratch using all data from each individual test task (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 can be a poor initialization for shifted distributions, e.g., AnoShift. We also observe that 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 the model trained from scratch using all data from individual tasks on two datasets. This is because REACT adapts from a model meta-trained on a larger and more diverse training set than each individual task, providing a stronger foundation for adaptation. Furthermore, the training data in AnoShift and Malware contains noise (10% threat ratio). Training a model on all data from an individual task increases the likelihood of exposing the model to many threat samples within that distribution, which, due to the specific training objectives of AutoEncoder and DSVDD, may cause the models to mistakenly learn malicious patterns as benign ones. In contrast, REACT is less prone to such overfitting as it utilizes the meta model and only updates the adaptive weights on a small set of new data. 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 the real world. When compared to other baselines using the same k samples from new tasks, REACT achieves the highest scores. Among state-of-the-art methods, ACR, which performs test-time adaptation, shows relatively lower performance as it does not apply gradient updates during inference, limiting its adaptation ability. OC-MAML achieves the second-best performance, demonstrating the strength of meta-learning. However, REACT outperforms OC-MAML by weight decomposition and incorporating a hypernetwork for contextual tuning. These designs help maintain generalizability and enhance adaptability beyond meta-learning alone.

[3] ACR updates BN statistics through forward passes without backward propagation.
[4] https://huggingface.co/sentence-transformers

Table 3: Ablation study. The results demonstrate that every component in our framework contributes to the overall performance improvement.

Method	AnoShift		Malware		NSL-KDD	
	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR
static model	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
OC-MAML	0.7770	0.3811	0.6779	0.2334	0.8547	0.5504
REACT (ours)	**0.8226**	**0.4376**	**0.7252**	**0.2750**	**0.8673**	**0.5559**
w/o fine-tuning	0.7873	0.3977	0.7159	0.2696	0.7282	0.4838
w/o context	0.7865	0.3838	0.6772	0.2325	0.8503	0.5323
w/ random context	0.7922	0.3888	0.6892	0.2426	0.8597	0.5522
w/o regularization	0.6541	0.3379	0.6326	0.2001	0.8045	0.4705

6.3 Ablation Studies

We crafted four ablated versions of REACT by systematically removing each key component: (1) We disable fine-tuning during inference and use the merged weights from meta weights and the hypernetwork's prediction to do the inference directly, denoted as **w/o fine-tuning**. (2) We remove the use of context and only provide the support data for hypernetwork, denoted as **w/o context**. (3) We replace the context with fixed random vectors of the same shape as context embeddings sampled from a normal distribution, denoted as **w/ random context**. (4) We remove the regularization term on the hypernetwork's prediction, denoted as **w/o regularization**.

The results in Table 3 show that every component in REACT contributes to performance improvement. Fine-tuning and regularization have notable impacts. REACT w/o fine-tuning shows competitive performance on AnoShift and Malware compared to the baselines, indicating its potential for zero-shot adaptation. However, with just a few gradient updates, performance can be largely improved. Besides, adding regularization ensures the adaptive weights predicted by the hypernetwork do not overpower the full model, maintaining model generalizability. We present the results with different values of the regularization parameter in Appendix C. REACT w/o context learns distribution patterns solely from the support data, which is less effective than incorporating contexts since the support data is limited and might not provide sufficient insights. REACT w/ random context introduces randomness into the encoded representations. This exposes the hypernetwork to variations during training and reduces reliance on specific patterns, enhancing robustness to noise. Therefore, it performs slightly better than w/o context. However, these random contexts do not provide task-specific knowledge to capture meaningful patterns. With additional information about tasks, REACT can model similarity among distributions more effectively. We anticipate that selecting indicative contexts for downstream applications and using advanced techniques like graph-based prompting [63] could further enhance context encoding. These explorations are left for future work.

6.4 Sensitivity Analyses

Number of Support Samples. We vary the number of support samples k for each task from 5 to 100 and compare REACT with the fine-tuning baseline. Figure 4 shows that REACT consistently outperforms the fine-tuning baseline across all datasets. This demonstrates REACT's robustness in data-scarce scenarios and highlights its ability to efficiently leverage available data for fast adaptation.

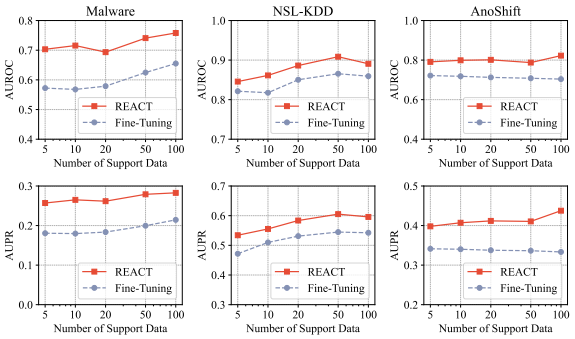

Figure 4: Sensitivity analysis: number of support samples (k).

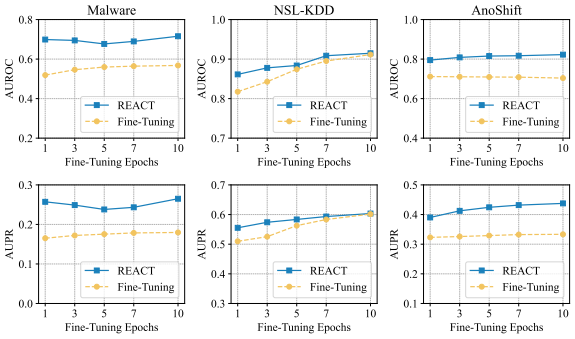

Figure 5: Sensitivity analysis: number of fine-tuning epochs.

Table 4: Sensitivity analysis: different contamination levels.

Method	Malware				AnoShift			
	1%	5%	10%	20%	1%	5%	10%	20%
static model	0.506	0.513	0.517	0.567	0.764	0.753	0.711	0.634
train-from-scratch	0.366	0.368	0.366	0.377	0.818	0.791	0.740	0.740
fine-tuning	0.550	0.545	0.568	0.580	0.812	0.765	0.704	0.605
continual learning	0.562	0.559	0.588	0.590	0.683	0.572	0.609	0.453
ER	0.582	0.585	0.602	0.602	0.734	0.669	0.614	0.577
ACR	0.544	0.570	0.580	0.570	0.785	0.774	0.763	0.773
OC-MAML	0.683	0.688	0.678	0.687	0.827	0.803	0.777	0.755
REACT (ours)	**0.725**	**0.738**	**0.725**	**0.719**	**0.832**	**0.813**	**0.823**	**0.775**

Number of Fine-Tuning Epochs. We vary the number of fine-tuning epochs for each new task from 1 to 10 and compare the performance of REACT with the fine-tuning baseline. Figure 5 shows that REACT outperforms the baseline in all settings. On NSL-KDD, the improvement of REACT over the baseline becomes less significant with more fine-tuning epochs, as its tasks are simpler and the model is able to adapt to them with fewer epochs.

Contamination on Training Data. We evaluate the robustness of our system against contamination in the training data when

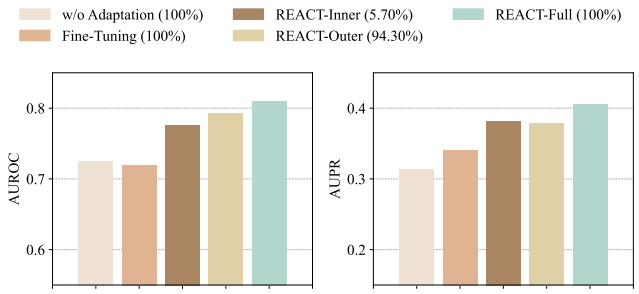

Figure 6: Parameter-efficient fine-tuning. Numbers in the legend indicate the percentage of fine-tuned parameters. Both REACT-Inner and REACT-Outer outperform the baselines.

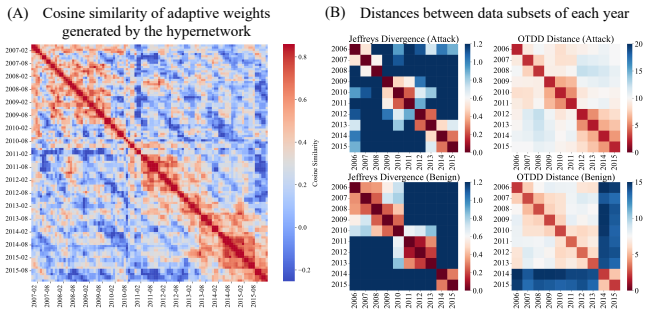

Figure 7: Case Study. The adaptive weights generated for each month are similar to those of nearby months, reflecting the data shift pattern.

applying to AutoEcoder and DeepSVDD models on AnoShift and Malware respectively—both unsupervised methods. We note that GOAD is a semi-supervised method trained solely on benign data (as applied to the NSL-KDD dataset) thus the evaluation is trivial to it. We fix the number of benign samples while varying the ratio of threat samples from 1% to 20%. Table 4 shows the AUROC scores. REACT consistently achieves higher AUROC scores across different contamination rates than the fine-tuning baseline, showing that it is robust to noise in training data.

6.5 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, 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 predicted 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 6. Both methods achieve better performance compared to the baselines, although they slightly underperformed compared to REACT-Full which fine-tunes all layers. Notably, REACT-Inner achieved a 5.75% higher AUC while updating 94.3% fewer parameters compared to conventional full fine-tuning, highlighting its efficiency.

6.6 Case Study

To understand how well REACT leverages contextual information, we analyze the weights generated by the trained hypernetworks. Specifically, we compare the adaptive biases of the last layer in the AutoEncoder for AnoShift across different months and calculate their cosine similarities. The results are presented in Figure 7 A, with warmer colors indicating higher similarity. The high similarities around the diagonal indicate the weights generated for each month are similar to those of nearby months. This suggests that REACT effectively captures the temporal dynamics and smoothly adapts model weights over time. As a reference for how data shifts, we follow the analyses in [17] to calculate distances between data subsets of each year. Specifically, we measure the Jeffrey's Divergence [31] averaged over categorical features and the Optimal Transport Dataset Distance (OTDD) [3] across all features. As shown in Figure 7 B, data from adjacent years exhibit smaller distances (in red). Besides, it presents block patterns where data from 2006–2010, 2011–2013, and 2014–2015 are more similar within their respective groups than with other years. This temporal shift corresponds with trends in weight similarity over time. The observations also hint at the potential for detecting shifts, a research question actively discussed in the literature [27, 37, 54]—by monitoring deviations in the hypernetwork's predictions compared to prior tasks, we may identify moments where shifts occur.

7 Conclusions

Our work sheds light on how to approach the distribution shift problem—from both meta-learning and fine-tuning perspectives. We propose a novel framework, REACT, that decomposes the weights of a neural network into the sum of meta and adaptive components, following a meta-learning paradigm to train the components. By integrating a hypernetwork to generate adaptive weights, REACT enables knowledge sharing and adjusts weights for new distributions with minimal fine-tuning effort. 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 threat detection models and objectives.

While focused on cybersecurity, the principles and methods developed in our research can be adapted to other fields facing similar distribution shift challenges, such as finance [21, 25, 71] and healthcare [32, 36, 59]. Our study provides insights for studies in the general machine learning community, fostering a more comprehensive understanding of adaptation and fine-tuning by showcasing applications in cybersecurity. One of the future directions is to incorporate 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. This approach could enhance the framework's ability to continuously adapt to evolving distributions, especially for scenarios with significant distribution shifts over a long period of time.

References

[1] Charu C Aggarwal and Charu C Aggarwal. 2017. *An introduction to outlier analysis*. Springer.

[2] Yuval Alaluf, Omer Tov, Ron Mokady, Rinon Gal, and Amit Bermano. 2022. Hyperstyle: Stylegan inversion with hypernetworks for real image editing. In *Proceedings of the IEEE/CVF conference on computer Vision and pattern recognition*. 18511–18521.

[3] David Alvarez-Melis and Nicolo Fusi. 2020. Geometric dataset distances via optimal transport. *Advances in Neural Information Processing Systems* 33 (2020), 21428–21439.

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

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

[6] Sidahmed Benabderrahmane, Ghita Berrada, James Cheney, and Petko Valtchev. 2021. A rule mining-based advanced persistent threats detection system. *arXiv preprint arXiv:2105.10053* (2021).

[7] Liron Bergman and Yedid Hoshen. 2020. Classification-based anomaly detection for general data. *arXiv preprint arXiv:2005.02359* (2020).

[8] David Bonet, Daniel Mas Montserrat, Xavier Giró-i Nieto, and Alexander G Ioannidis. 2024. HyperFast: Instant Classification for Tabular Data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 11114–11123.

[9] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. 2000. LOF: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*. 93–104.

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

[11] Sumohana Channappayya, Bheemarjuna Reddy Tamma, et al. 2024. Augmented Memory Replay-based Continual Learning Approaches for Network Intrusion Detection. *Advances in Neural Information Processing Systems* 36 (2024).

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

[13] Vinod Kumar Chauhan, Jiandong Zhou, Ping Lu, Soheila Molaei, and David A Clifton. 2023. A brief review of hypernetworks in deep learning. *arXiv preprint arXiv:2306.06955* (2023).

[14] Sanyuan Chen, Yutai Hou, Yiming Cui, Wanxiang Che, Ting Liu, and Xiangzhan Yu. 2020. Recall and learn: Fine-tuning deep pretrained language models with less forgetting. *arXiv preprint arXiv:2004.12651* (2020).

[15] Hua Ding, Lixing Chen, Shenghong Li, Yang Bai, Pan Zhou, and Zhe Qu. 2024. Divide, Conquer, and Coalesce: Meta Parallel Graph Neural Network for IoT Intrusion Detection at Scale. In *Proceedings of the ACM on Web Conference 2024*. 1656–1667.

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

[17] Marius Drăgoi, Elena Burceanu, Emanuela Haller, Andrei Manolache, and Florin Brad. 2022. AnoShift: A Distribution Shift Benchmark for Unsupervised Anomaly Detection. *Neural Information Processing Systems NeurIPS, Datasets and Benchmarks Track* (2022).

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

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

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

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

[22] Adam Goodge, Bryan Hooi, See-Kiong Ng, and Wee Siong Ng. 2022. Lunar: Unifying local outlier detection methods via graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 6737–6745.

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

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

[25] Yue Guo, Chenxi Hu, and Yi Yang. 2023. 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*. 1029–1038.

[26] David Ha, Andrew Dai, and Quoc V Le. 2016. Hypernetworks. *arXiv preprint arXiv:1609.09106* (2016).

[27] Dongqi Han, Zhiliang Wang, Wenqi Chen, Kai Wang, Rui Yu, Su Wang, Han Zhang, Zhihua Wang, Minghui Jin, Jiahai Yang, et al. 2023. Anomaly Detection in the Open World: Normality Shift Detection, Explanation, and Adaptation.. In *NDSS*.

[28] Songqiao Han, Xiyang Hu, Hailiang Huang, Minqi Jiang, and Yue Zhao. 2022. Adbench: Anomaly detection benchmark. *Advances in Neural Information Processing Systems* 35 (2022), 32142–32159.

[29] Zhongzhan Huang, Mingfu Liang, Shanshan Zhong, and Liang Lin. [n. d.]. AttNS: Attention-Inspired Numerical Solving For Limited Data Scenarios. In *Forty-first International Conference on Machine Learning*.

[30] Ngoc Anh Huynh, Wee Keong Ng, and Kanishka Ariyapala. 2017. 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*. Springer, 18–32.

[31] Harold Jeffreys. 1946. An invariant form for the prior probability in estimation problems. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences* 186, 1007 (1946), 453–461.

[32] Xiayan Ji, Hyonyoung Choi, Oleg Sokolsky, and Insup Lee. 2023. 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*. 327–339.

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

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

[35] Dongmin Kim, Sunghyun Park, and Jaegul Choo. 2024. When Model Meets New Normals: Test-Time Adaptation for Unsupervised Time-Series Anomaly Detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 13113–13121.

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

[37] Sean Kulinski, Saurabh Bagchi, and David I. Inouye. 2021. Feature Shift Detection: Localizing Which Features Have Shifted via Conditional Distribution Tests. *ArXiv* abs/2107.06929 (2021). https://api.semanticscholar.org/CorpusID:224344377

[38] Atsutoshi Kumagai, Tomoharu Iwata, Hiroshi Takahashi, and Yasuhiro Fujiwara. 2023. Meta-learning for Robust Anomaly Detection. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 675–691.

[39] Yann LeCun, Koray Kavukcuoglu, and Clement Farabet. 2010. Convolutional networks and applications in vision. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*. 253–256. doi:10.1109/ISCAS.2010.5537907

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

[41] Hongyu Li, Liang Ding, Meng Fang, and Dacheng Tao. 2024. Revisiting Catastrophic Forgetting in Large Language Model Tuning. *arXiv preprint arXiv:2406.04836* (2024).

[42] Heng Li, Shiyao Zhou, Wei Yuan, Xiapu Luo, Cuiying Gao, and Shuiyan Chen. 2021. Robust android malware detection against adversarial example attacks. In *Proceedings of the Web Conference 2021*. 3603–3612.

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

[44] Zheng Li, Yue Zhao, Nicola Botta, Cezar Ionescu, and Xiyang Hu. 2020. COPOD: copula-based outlier detection. In *2020 IEEE international conference on data mining (ICDM)*. IEEE, 1118–1123.

[45] Zheng Li, Yue Zhao, Xiyang Hu, Nicola Botta, Cezar Ionescu, and George H Chen. 2022. Ecod: Unsupervised outlier detection using empirical cumulative distribution functions. *IEEE Transactions on Knowledge and Data Engineering* 35, 12 (2022), 12181–12193.

[46] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation forest. In *2008 eighth ieee international conference on data mining*. IEEE, 413–422.

[47] Liu Liu, Olivier De Vel, Chao Chen, Jun Zhang, and Yang Xiang. 2018. Anomaly-based insider threat detection using deep autoencoders. In *2018 IEEE international conference on data mining workshops (ICDMW)*. IEEE, 39–48.

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

[49] Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. 2021. Parameter-efficient multi-task fine-tuning for transformers via

shared hypernetworks. *arXiv preprint arXiv:2106.04489* (2021).
[50] Yuren Mao, Yaobo Liang, Nan Duan, Haobo Wang, Kai Wang, Lu Chen, and Yunjun Gao. 2022. Less-forgetting Multi-lingual Fine-tuning. *Advances in Neural Information Processing Systems* 35 (2022), 14917–14928.
[51] OpenAI. 2024. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL] https://arxiv.org/abs/2303.08774
[52] Feargus Pendlebury, Fabio Pierazzi, Roberto Jordaney, Johannes Kinder, and Lorenzo Cavallaro. 2019. {TESSERACT}: Eliminating experimental bias in malware classification across space and time. In *28th USENIX security symposium (USENIX Security 19)*. 729–746.
[53] Tomáš Pevnỳ. 2016. Loda: Lightweight on-line detector of anomalies. *Machine Learning* 102 (2016), 275–304.
[54] Stephan Rabanser, Stephan Günnemann, and Zachary Lipton. 2019. Failing loudly: An empirical study of methods for detecting dataset shift. *Advances in Neural Information Processing Systems* 32 (2019).
[55] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. 2000. Efficient algorithms for mining outliers from large data sets. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*. 427–438.
[56] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster R-CNN: towards real-time object detection with region proposal networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1* (Montreal, Canada) *(NIPS'15)*. MIT Press, Cambridge, MA, USA, 91–99.
[57] Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. 2018. Deep one-class classification. In *International conference on machine learning*. PMLR, 4393–4402.
[58] Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. 2001. Estimating the support of a high-dimensional distribution. *Neural computation* 13, 7 (2001), 1443–1471.
[59] Jessica Schrouff, Natalie Harris, Sanmi Koyejo, Ibrahim M Alabdulmohsin, Eva Schnider, Krista Opsahl-Ong, Alexander Brown, Subhrajit Roy, Diana Mincu, Christina Chen, et al. 2022. Diagnosing failures of fairness transfer across distribution shift in real-world medical settings. *Advances in Neural Information Processing Systems* 35 (2022), 19304–19318.
[60] Aviv Shamsian, Aviv Navon, Ethan Fetaya, and Gal Chechik. 2021. Personalized federated learning using hypernetworks. In *International Conference on Machine Learning*. PMLR, 9489–9502.
[61] Kendrick Shen, Robbie M Jones, Ananya Kumar, Sang Michael Xie, Jeff Z HaoChen, Tengyu Ma, and Percy Liang. 2022. Connect, not collapse: Explaining contrastive learning for unsupervised domain adaptation. In *International conference on machine learning*. PMLR, 19847–19878.
[62] Petar Stojanov, Zijian Li, Mingming Gong, Ruichu Cai, Jaime Carbonell, and Kun Zhang. 2021. Domain adaptation with invariant representation learning: What transformations to learn? *Advances in Neural Information Processing Systems* 34 (2021), 24791–24803.
[63] Xiangguo Sun, Hong Cheng, Jia Li, Bo Liu, and Jihong Guan. 2023. All in One: Multi-Task Prompting for Graph Neural Networks. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Long Beach, CA, USA) *(KDD '23)*. Association for Computing Machinery, New York, NY, USA, 2120–2131. doi:10.1145/3580305.3599256
[64] Xiangguo Sun, Hongzhi Yin, Bo Liu, Hongxu Chen, Jiuxin Cao, Yingxia Shao, and Nguyen Quoc Viet Hung. 2021. Heterogeneous hypergraph embedding for graph classification. In *Proceedings of the 14th ACM international conference on web search and data mining*. 725–733.
[65] Zachariah Sutton, Peter Willett, and Yaakov Bar-Shalom. 2018. Modeling and detection of evolving threats using random finite set statistics. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 4319–4323.
[66] Korawat Tanwisuth, Xinjie Fan, Huangjie Zheng, Shujian Zhang, Hao Zhang, Bo Chen, and Mingyuan Zhou. 2021. A prototype-oriented framework for unsupervised domain adaptation. *Advances in Neural Information Processing Systems* 34 (2021), 17194–17208.
[67] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A Ghorbani. 2009. A detailed analysis of the KDD CUP 99 data set. In *2009 IEEE symposium on computational intelligence for security and defense applications*. Ieee, 1–6.
[68] Sebastian Thrun and Lorien Pratt. 1998. Learning to learn: Introduction and overview. In *Learning to learn*. Springer, 3–17.
[69] Cheng-Hao Tu, Hong-You Chen, Zheda Mai, Jike Zhong, Vardaan Pahuja, Tanya Berger-Wolf, Song Gao, Charles Stewart, Yu Su, and Wei-Lun Harry Chao. 2024. Holistic transfer: towards non-disruptive fine-tuning with partial target data. *Advances in Neural Information Processing Systems* 36 (2024).
[70] A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems* (2017).
[71] Chen Wang, Ziwei Fan, Liangwei Yang, Mingdai Yang, Xiaolong Liu, Zhiwei Liu, and Philip Yu. 2024. 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*. 2970–2979.
[72] Hao Wang. 2024. Improving Neural Network Generalization on Data-limited Regression with Doubly-Robust Boosting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 20821–20829.
[73] Ze Wang, Yipin Zhou, Rui Wang, Tsung-Yu Lin, Ashish Shah, and Ser Nam Lim. 2022. Few-shot fast-adaptive anomaly detection. *Advances in Neural Information Processing Systems* 35 (2022), 4957–4970.
[74] Garrett Wilson and Diane J Cook. 2020. A survey of unsupervised deep domain adaptation. *ACM Transactions on Intelligent Systems and Technology (TIST)* 11, 5 (2020), 1–46.
[75] Shuo Yang, Xinran Zheng, Jinze Li, Jinfeng Xu, Xingjun Wang, and Edith CH Ngai. 2024. ReCDA: Concept Drift Adaptation with Representation Enhancement for Network Intrusion Detection. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3818–3828.
[76] Shuhan Yuan, Panpan Zheng, Xintao Wu, and Hanghang Tong. 2020. Few-shot insider threat detection. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2289–2292.
[77] Hai Zhang, Chunwei Wu, Guitao Cao, Hailing Wang, and Wenming Cao. 2024. HyperEditor: Achieving Both Authenticity and Cross-Domain Capability in Image Editing via Hypernetworks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 7051–7059.
[78] Jiayun Zhang, Shuheng Li, Haiyu Huang, Zihan Wang, Xiaohan Fu, Dezhi Hong, Rajesh K Gupta, and Jingbo Shang. 2024. How Few Davids Improve One Goliath: Federated Learning in Resource-Skewed Edge Computing Environments. In *Proceedings of the ACM on Web Conference 2024*. 2976–2985.
[79] Jiayun Zhang, Xinyang Zhang, Dezhi Hong, Rajesh K Gupta, and Jingbo Shang. 2023. Minimally Supervised Contextual Inference from Human Mobility: An Iterative Collaborative Distillation Framework.. In *IJCAI*. 2450–2458.
[80] Yong Zhong, Hongtao Liu, Xiaodong Liu, Fan Bao, Weiran Shen, and Chongxuan Li. 2022. Deep generative modeling on limited data with regularization by nontransferable pre-trained models. *arXiv preprint arXiv:2208.14133* (2022).
[81] Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. 2018. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *International conference on learning representations*.

A Pseudo-code of REACT

Algorithm 1: Training Procedure of REACT

Input: Task distribution $\mathcal{P}(\mathcal{T})$, target network f, hypernetwork h
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 $\{\mathcal{T}_i\}_{i=1}^{M} \sim \mathcal{P}(\mathcal{T})$;
  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 $\{\mathcal{T}_j\}_{j=1}^{M} \sim \mathcal{P}(\mathcal{T})$;
  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;
  Update ϕ following Eq. 3;

B Backbone Models and Implementation Details

- **AutoEncoder (AE) [1]:** is an unsupervised model trained to reconstruct the input through an encoder-decoder structure. The key idea is that, threat samples, appearing less frequently, tend to have larger reconstruction losses, making them distinguishable by observing the loss. We implement the AutoEncoder with four linear layers followed by ReLU activation. These layers project the data into [64, 32, 64]-dimensional features and finally map the features to the original data dimension. Cross-entropy loss is applied to categorical features, and mean-square error is used for numerical features.
- **DeepSVDD (DSVDD) [57]:** is an unsupervised model which encodes data into feature representations and measures their distances from a learnable center. The encoder is a multi-layer perception (MLP) consisting of two linear layers with ReLU activation, mapping data to representations of dimension [64, 32]. Similar to the AutoEncoder, threat samples tend to have larger distances from the center. The smooth L1 loss [56] is used to measure the distances for its robustness against outliers.
- **GOAD [7]:** is a semi-supervised model that applies multiple transformations to the input data and uses a convolutional neural network (CNN) [39] to extract feature representations. We implement a 5-layer CNN with kernel size of 1. The loss function has two components: a center triplet loss, which measures the distance between the learned representations and their mean, and a cross-entropy loss for predicting which transformation was applied to the data.

C Effect of Regularization

We experiment with different values of the regularization parameter λ as presented in Table 5. Adjusting λ controls the trade-off between adaptability and generalization. A larger λ reduces the norm of adaptive weights and emphasizes generalization, while a smaller λ encourages adaptability. Selecting an appropriate λ is essential for achieving optimal performance.

Table 5: Results with different regularization parameter λ.

λ	AnoShift		Malware		NSL-KDD	
	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR
0	0.6541	0.3379	0.6326	0.2001	0.8045	0.4705
1	0.7276	0.3860	0.6878	0.2531	0.8260	0.5331
10	0.8226	0.4376	0.7252	0.2750	0.8673	0.5559
100	0.7889	0.3947	0.7048	0.2561	0.8192	0.5134

D Computational Efficiency in Adaptation

The hypernetwork is fixed after training. During adaptation, it conducts a single forward pass for a new task which incurs minimal overhead. As adaptation typically occurs less frequently (e.g., once daily) than inference, this overhead is negligible. Table 6 shows the run time of hypernet forward pass (t_1) and gradient descent fine-tuning (t_2) during the adaptation of a new task. Experiments are performed with a NVIDIA Tesla T4 GPU.

Table 6: Run time of REACT for adaptation.

Dataset	t_1 (ms per task)	t_2 (ms per task)	t_1/t_2
AnoShift	1.52	255.39	0.59%
NSL-KDD	1.58	59.42	2.66%
Malware	1.61	18.69	8.62%

E Convergence Analysis on REACT

We provide convergence analysis of REACT on linear models under the premise of Theorem 1, that the models h and f admit the form (4), the adaptive weights are updated by exactly solving Eq. 1 and relevant datasets are sampled at the beginning of the algorithm and fixed throughout the iterations.

Based on Eq. (4), we consider the following objective function.

$$\mathcal{L}(f(X;\theta_{\text{meta}},\theta_{\text{adapt}})) = \frac{1}{2}\|f(X;\theta_{\text{meta}},\theta_{\text{adapt}}) - Y\|^2 + \frac{\lambda}{2}\|\theta_{\text{adapt}}\|^2,$$

which consists of a mean squared error and an L2 regularization for the adaptive weights (see Section 4.2). Y is the target associated with the loss function. It can have different forms according to the underlying target model. For example, it can be the input data for reconstruction loss, center of samples for methods like DeepSVDD, or labels in cases of supervised or semi-supervised learning.

Without loss of generality we set $\theta \in \mathbb{R}^{d_1}$ and $\phi \in \mathbb{R}^{d_2}$ for some $d_1, d_2 > 0$. Notice that this assumption can be generalized by considering vectorization of the matrix product and hence our results can easily be extended to more generic output spaces. We also note that the assumption that the datasets have uncorrelated constant variance, i.e. $(X^i)^\top(X^i) = \sigma_i I$ is to make the computations in the proof easier. The proof can be relaxed to bounded norm, i.e. $\|X^i\|_2 \le \sigma_i$ where $\|.\|_2$ is L-2 norm on the matrix space.

We restate the theorem here.

Theorem 1. *Consider REACT on the linear model in* (4) *with Eq.* (1) *being solved exactly. Let X_s^i and X_q^i satisfy $(X_s^i)^\top X_s^i = (X_q^i)^\top X_q^i = \sigma_i I$ for each task $i \in \{1,..,M\}$, where σ_i are the variances and I is the identity matrix. Learning rates are chosen as $\eta_{meta} < 1/\sum_{i=1}^M \sigma_i\lambda/(\sigma_i+\lambda)$ and $\eta_h < 1/\max\left(\sum_{j=1}^{n_h}\sigma_j(\sigma_j+\lambda), \|\mathbf{X}_s\|\right)$ where $\mathbf{X}_s = \sum_{j=1}^M \sigma_j (X_s^j)^\top$. Then, for any $\varepsilon > 0$, there exists*

$$K = O\left(\log_{1/\rho_{meta}}(1/\varepsilon) + \log_{1/\rho_h}(1/\varepsilon)\right)$$

for $\rho_{meta} = 1-\eta_{meta}\sum_{i=1}^M \sigma_i\lambda/(\sigma_i+\lambda)$ and $\rho_h = 1-\eta_h\sum_{j=1}^M \sigma_j(\sigma_j+\lambda)$ such that the K-iteration of Algorithm 1 satisfies

$$\|\theta^K - \theta^*\| \le \varepsilon, \qquad \text{and} \qquad \|\phi^K - \phi^*\| \le \varepsilon,$$

where θ^* and ϕ^* are stationary points of the algorithm.

Proof. We prove the result following the steps in Algorithm 1. Let $\theta_{\text{adapt}}^{i,k}$ be the fine-tuned adaptive weights of task i at k-th iteration of REACT and similarly, ϕ^k denote the hypernetwork parameters, and θ_{meta}^k be the meta weights at iteration k.

Task fine-tuning. The exact intermediate updates defined in (1) can be rewritten as follows.

$$\theta_{\text{adapt}}^{i,k+1} = \underset{\theta}{\text{argmin}}\ \frac{1}{2}\|X_s^i(\theta_{\text{meta}}^k + \theta) - Y_s^i\|^2 + \frac{\lambda}{2}\|\theta\|^2$$

Setting gradient to zero, we have

$$\mathbf{0} = (X_s^i)^\top X_s^i(\theta_{\text{adapt}}^{i,k+1} + \theta_{\text{meta}}^k) - (X_s^i)^\top Y_s^i + \lambda\theta_{\text{adapt}}^{i,k+1},$$

where $\mathbf{0}$ is the vector of all zeros. This implies

$$\theta_{\text{adapt}}^{i,k+1} = \frac{1}{\sigma_i+\lambda}(X_s^i)^\top Y_s^i - \frac{\sigma_i}{\sigma_i+\lambda}\theta_{\text{meta}}^k, \tag{5}$$

where we used the fact that $(X_s^i)^\top X_s^i = \sigma_i I$.

Meta weight update. Next, we consider the gradient update of meta weight in (2). The gradient with respect to θ_{meta} is

$$\begin{aligned}\nabla_{\theta_{\text{meta}}} \sum_{x\in\mathbf{D}_{\text{query}}^i} \mathcal{L}\left(f(x;\theta_{\text{meta}},\theta_{\text{adapt}}^{i,k+1})\right)\Big|_{\theta_{\text{meta}}^k}&\\ = (X_q^i)^\top X_q^i\left(\theta_{\text{meta}}^k + \theta_{\text{adapt}}^{i,k+1}\right) - (X_q^i)^\top Y_q^i&\\ = \sigma_i\left(\frac{\lambda}{\sigma_i+\lambda}\theta_{\text{meta}}^k + \frac{1}{\sigma_i+\lambda}(X_s^i)^\top Y_s^i\right) - (X_q^i)^\top Y_q^i,&\end{aligned}$$

where the last equality is given by (5) and the assumption in data covariance matrix. Therefore, the gradient update step is

$$\begin{aligned}\theta_{\text{meta}}^{k+1} &= \theta_{\text{meta}}^k - \eta_{\text{meta}}\sum_{i=1}^M\sum_{x\in\mathbf{D}_{\text{query}}^i}\nabla_{\theta_{\text{meta}}}\mathcal{L}\left(f(x;\theta_{\text{meta}}^k,\theta_{\text{adapt}}^{i,k+1})\right)\\ &= \theta_{\text{meta}}^k - \eta_{\text{meta}}\sum_{i=1}^M\sigma_i\left(\frac{\lambda}{\sigma_i+\lambda}\theta_{\text{meta}}^k + \frac{1}{\sigma_i+\lambda}(X_s^i)^\top Y_s^i\right) - (X_q^i)^\top Y_q^i\\ &= \left(1-\eta_{\text{meta}}\sum_{i=1}^M\frac{\sigma_i\lambda}{\sigma_i+\lambda}\right)\theta_{\text{meta}}^k\\ &\quad - \eta_{\text{meta}}\sum_{i=1}^M\frac{\sigma_i}{\sigma_i+\lambda}(X_s^i)^\top Y_s^i - (X_q^i)^\top Y_q^i\end{aligned}$$

Let us introduce $\rho_{\text{meta}} = 1-\eta_{\text{meta}}\sum_{i=1}^M\sigma_i\lambda/(\sigma_i+\lambda)$, and choose learning rate $0<\eta_{\text{meta}}<1/\sum_{i=1}^M\sigma_i\lambda/(\sigma_i+\lambda)$ so that $0<\rho_{\text{meta}}<1$.

The stationary point θ_{meta}^* should satisfy

$$\begin{aligned}\theta_{\text{meta}}^* &= \left(1-\eta_{\text{meta}}\sum_{i=1}^M\frac{\sigma_i\lambda}{\sigma_i+\lambda}\right)\theta_{\text{meta}}^*\\ &\quad - \eta_{\text{meta}}\sum_{i=1}^M\frac{\sigma_i}{\sigma_i+\lambda}(X_s^i)^\top Y_s^i - (X_q^i)^\top Y_q^i.\end{aligned}$$

Thus, we obtain

$$(\theta_{\text{meta}}^{k+1} - \theta_{\text{meta}}^*) = \rho_{\text{meta}}(\theta_{\text{meta}}^k - \theta_{\text{meta}}^*)$$

yielding,

$$\|\theta_{\text{meta}}^{k+1} - \theta_{\text{meta}}^*\| \le \rho_{\text{meta}}\|\theta_{\text{meta}}^k - \theta_{\text{meta}}^*\| \le \cdots \le \rho_{\text{meta}}^{k+1}\|\theta_{\text{meta}}^0 - \theta_{\text{meta}}^*\|. \tag{6}$$

Hypernetwork update. Lastly, the gradient of the objective function update with respect to ϕ^k is

$$\begin{aligned}\nabla_\phi \sum_{x\in\mathbf{D}_{\text{query}}^j}\mathcal{L}\left(f(x;\theta_{\text{meta}}^{k+1},h(X_s^j;\phi))\right)\Big|_{\phi^k}&\\ = \left(X_q^jX_s^j\right)^\top\left(X_q^jX_s^j\phi^k + X_q^j\theta_{\text{meta}}^{k+1}\right) - \left(X_q^jX_s^j\right)^\top Y_q^j + \lambda(X_s^j)^\top X_s^j\phi^k&\\ = \sigma_j(\sigma_j+\lambda)\phi^k + \sigma_j(X_s^j)^\top\theta_{\text{meta}}^{k+1} - \left(X_q^jX_s^j\right)^\top Y_q^j&\end{aligned}$$

Thus, the update (3) can be written as

$$\begin{aligned}\phi^{k+1} &= \phi^k - \eta_h\sum_{j=1}^{n_h}\sum_{x\in\mathbf{D}_{\text{query}}^j}\nabla_\phi\mathcal{L}\left(f(x;\theta_{\text{meta}}^{k+1},h(X_s^j;\phi))\right)\\ &= (1-\eta_h\sum_{j=1}^M\sigma_j(\sigma_j+\lambda))\phi^k\\ &\quad - \eta_h\left(\sum_{i=1}^M\sigma_j(X_s^j)^\top\right)\theta_{\text{meta}}^{k+1} + \eta_h\sum_{i=1}^M\left(X_q^jX_s^j\right)^\top Y_q^j\end{aligned}$$

where the last equality follows from $(X_q^j)^\top X_q^j = (X_s^j)^\top X_s^j = \sigma_j I$.

Notice that the choice of learning rate η_h implies $0<\eta_h<1/\max\left(\sum_{j=1}^{n_h}\sigma_j(\sigma_j+\lambda), \|\mathbf{X}_s\|\right)$ so that the rate ρ_h and $\mathbf{X}_s$ satisfy $0<\rho_h<1$ and $0<\eta_h\|\mathbf{X}_s\|<1$. On the other hand, the stationary points ϕ^* and θ^* satisfy

$$\phi^* = \rho_h\phi^* - \eta_h\mathbf{X}_s\theta_{\text{meta}}^* + \eta_h\sum_{i=1}^M\left(X_q^jX_s^j\right)^\top Y_q^j$$

yielding

$$\phi^{k+1} - \phi^* = \rho_h(\phi^k - \phi^*) - \eta_h\mathbf{X}_s(\theta_{\text{meta}}^{k+1} - \theta_{\text{meta}}^*)$$

and

$$\|\phi^{k+1} - \phi^*\| \le \rho_h\|\phi^k - \phi^*\| + \eta_h\|\mathbf{X}_s\|\|\theta_{\text{meta}}^{k+1} - \theta_{\text{meta}}^*\| \tag{7}$$

Convergence. With (6), we can show that for $k \ge K_{\text{meta}} = \log_{1/\rho_m}(1/\varepsilon) + \log_{1/\rho_m}(\|\theta_{\text{meta}}^0 - \theta_{\text{meta}}^*\|)$, we have

$$\|\theta_{\text{meta}}^k - \theta_{\text{meta}}^*\| \le \varepsilon.$$

Similarly, from (7), we get

$$\begin{aligned}\|\phi^k - \phi^*\| &\le \rho_h\|\phi^{k-1} - \phi^*\| + \eta_h\|\mathbf{X}_s\|\|\theta_{\text{meta}}^k - \theta_{\text{meta}}^*\|\\ &\le \rho_h\|\phi^{k-1} - \phi^*\| + \eta_h\|\mathbf{X}_s\|\rho_{\text{meta}}^k\|\theta_{\text{meta}}^0 - \theta_{\text{meta}}^*\|\\ &\le \rho_h^2\|\phi^{k-2} - \phi^*\| + \eta_h\|\mathbf{X}_s\|\|\theta_{\text{meta}}^0 - \theta_{\text{meta}}^*\|\left[\rho_{\text{meta}}^k + \rho_h\rho_{\text{meta}}^{k-1}\right]\\ &\cdots\\ &\le \rho_h^k\|\phi^0 - \phi^*\| + \frac{\rho_{\text{meta}}\eta_h\|\mathbf{X}_s\|\|\theta_{\text{meta}}^0 - \theta_{\text{meta}}^*\|}{\rho_{\text{meta}} - \rho_h}(\rho_{\text{meta}}^k - \rho_h^k),\end{aligned}$$

where we used (6) in the second inequality. Therefore, for any k satisfies

$$\begin{aligned}k &\ge K_h\\ &= \log_{1/\rho_h}(2/\varepsilon) + \log_{1/\rho_h}(\|\phi^0 - \phi^*\|)\\ &\quad + \log_{1/\rho_h}(4/\varepsilon) + \log_{1/\rho_h}\left(\frac{\rho_{\text{meta}}\eta_h\|\mathbf{X}_s\|\|\theta_{\text{meta}}^0 - \theta_{\text{meta}}^*\|}{|\rho_{\text{meta}} - \rho_h|}\right)\\ &\quad + \log_{1/\rho_{\text{meta}}}(4/\varepsilon) + \log_{1/\rho_{\text{meta}}}\left(\frac{\rho_{\text{meta}}\eta_h\|\mathbf{X}_s\|\|\theta_{\text{meta}}^0 - \theta_{\text{meta}}^*\|}{|\rho_{\text{meta}} - \rho_h|}\right),\end{aligned}$$

we have

$$\|\phi^k - \phi^*\| \le \frac{\varepsilon}{2} + \frac{\varepsilon}{4} + \frac{\varepsilon}{4} = \varepsilon.$$

Therefore, we can choose

$$K = \max(K_{\text{meta}}, K_h) = O\left(\log_{1/\rho_{\text{meta}}}(1/\varepsilon) + \log_{1/\rho_h}(1/\varepsilon)\right)$$

This completes the proof. □