Dynamic Inference from IoT Traffic Flows under Concept Drifts in Residential ISP Networks

Arman Pashamokhtari, Norihiro Okui, Masataka Nakahara, Ayumu Kubota,
Gustavo Batista and Hassan Habibi Gharakheili

***Abstract*—Millions of vulnerable consumer IoT devices in home networks are the enabler for cyber crimes putting user privacy and Internet security at risk. Internet service providers (ISPs) are best poised to mitigate risks by automatically inferring active IoT devices per household and notifying users of vulnerable ones. Developing a scalable inference method that can perform robustly across thousands of home networks is a non-trivial task. This paper focuses on the challenges of developing and applying data-driven inference models when labeled data of device behaviors is limited and the distribution of data changes across time and space domains (concept drifts). Our contributions are fourfold: (1) We collect and analyze more than six million network traffic flows of 24 types of consumer IoT devices from 12 real homes over six weeks to highlight the challenge of temporal and spatial concept drifts in network behaviors of IoT devices – we publicly release our training and testing instances data; (2) We analyze the performance of two inference strategies, namely "*global inference*" (a model trained on a combined set of all labeled data from training homes) and "*contextualized inference*" (several models each trained on the labeled data from a training home) in the presence of concept drifts; (3) To manage concept drifts, we develop a method that dynamically applies the "best" model (from a set) to network traffic of unseen homes during the testing phase, yielding better performance in a fifth of scenarios when the labels are available for the testing data (ideal but unrealistic settings); and (4) We develop a method to automatically select the best model without needing labels of unseen data (a realistic inference) and show that it can achieve 94% of the ideal model's accuracy.**

***Index Terms*—IoT, IPFIX data, traffic inference, concept drifts, machine learning**

I. Introduction

IOT devices are becoming popular in households. It is estimated that in 2021 there were about 258 million smart homes around the globe [1]. The smart home market is also anticipated to grow at least by the next two years [2]. This means more IoT devices will be deployed in home networks by the near future [3] supplied by the overwhelming market of IoT device manufacturing that includes more than 1200 companies delivering software/hardware platforms for IoT devices [4].

Despite their popularity, IoT devices bring several vulnerabilities that make them attractive to cyber criminals. Weak (or default) passwords and open insecure service ports are considered the top two exploited vulnerabilities of IoT devices [5]. Home users typically do not have adequate knowledge for securing their IoT devices which causes malware like Mirai and Mozi to compromise thousands of insecure devices and create a botnet for launching large-scale DDoS (Distribute Denial of Service) attacks [6]. Additionally, due to the diverse market of IoT manufacturing, which includes small/startup companies, the lack of security standards, and the urge to keep the manufacturing cost to a minimum, IoT devices often come without sufficient embedded security features [7].

Now with the massive number of vulnerable IoT devices in households, they are becoming a risk challenge for ISPs [8]. Compromised IoT devices participating in volumetric attacks can take the bandwidth of ISPs, downgrading the quality of service for other legitimate customers and making the ISPs accountable for carrying attack traffic across the Internet. The first step in every risk assessment and security analysis is obtaining visibility by discovering IoT devices connected to the network and determining their type (*e.g.,* camera, TV, speaker) [9] using characteristics known a priori.

Tools like Avira SafeThings [10] are developed to be installed on home routers for detecting and monitoring IoT devices; however, deploying such a service requires the involvement of every customer, which is a tremendous effort for an ISP that servers thousands of customers. A practical solution could be a passive subscription-based service provided by ISPs that analyzes the traffic of residential networks (with the users' consent) to detect the IoT devices in each home network. As done in this paper, the ISPs may only use metadata and flow-based statistical information without any deep packet inspection to reduce privacy concerns. Once an IoT device is detected in a home network, the ISP can check public vulnerability databases like CVE [11] and notify the users to take recommended actions (*e.g.,* changing credentials or upgrading firmware) to rectify the risk.

IoT devices often show identifiable patterns in their network behaviors, making them relatively distinguishable from each other (though behavioral overlaps are common too [12]). Many works have developed methods for detecting IoT devices using their network traffic [13]–[22]. From the existing works, only a few of them [14], [17], [18], [20], [23], [24] are capable of detecting IoT devices through limited and obfuscated traffic due to Network Address Translation (NAT) on home gateways which makes them feasible to be deployed by ISPs.

However, these patterns may change [25]–[27] by the time and context of their use across various networks. The behavioral change is more pronounced when IoT traffic inference is the objective of an ISP tasked to serve and manage tens of

A. Pashamokhtari and H. Habibi Gharakheili are with the School of Electrical Engineering and Telecommunications, and G. Batista is with the School of Computer Science and Engineering, University of New South Wales, Sydney, NSW 2052, Australia (e-mails: a.pashamokhtari@unsw.edu.au, h.habibi@unsw.edu.au, g.batista@unsw.edu.au).

N. Okui, M. Nakahara, and Ayumu Kubota are with KDDI Research, Inc., Saitama, Japan (e-mails: no-okui@kddi.com, ms-nakahara@kddi.com, ay-kubota@kddi.com).

thousands of home networks, each with a unique composition of assets and users, all distributed across sizable geography (city, state, or country). Several existing research works studied different methods that ISPs can leverage to detect IoT devices in residential networks [20], [23], [24], [28]–[30]. Prior works tend to train a global model with machine learning algorithms and fine-tune it by traffic data collected from a testbed (representing a single context). Due to the limitation of data or evaluation scenarios, they did not encounter context variations and could not highlight and/or address their impacts.

The challenge of data distribution change in data-driven modeling is known as concept drift which has been studied in various contexts like image recognition [31], insect species detection [32], air quality detection [33], electricity price [34], and network intrusion detection [35]. Some existing methods for addressing concept drift [35]–[40] require true labels during the testing phase to measure the error rate and re-train the model if the error rate increases significantly. This assumption is challenging for our context as obtaining labeled data for thousands of home networks is not trivial for an ISP. Therefore, a practical solution must be capable of addressing concept drift with a limited amount of labeled data from a few home networks for training purposes.

Context-aware or contextualized modeling [31]–[33] is an alternative to global modeling where (1) a set of training contexts are identified; (2) models are trained on a per-context basis; and (3) a given testing sample is predicted by the "closest" model selected from all available trained models. The definition of context is dependent on the application. In our case, each home network can be considered as a context. Selecting the closest model is the key process where we can leverage the classification scores of the models without requiring the true labels in the testing phase [31], [32], which makes this method more appealing for our problem. Note that each contextualized model is trained by data collected from a single context with narrow and relatively tight knowledge. In contrast, the global model captures data from multiple contexts, making its knowledge broader and loose. Limiting a model to learn a single context may increase the chance of overfitting, while exposing a model to a diversified data set may not necessarily result in better performance, especially when the data is noisy. Though contextualized modeling has been studied in other domains [31]–[33], to the best of our knowledge, no relevant study has been found in the area of IoT traffic inference.

This paper compares global and contextualized modeling for classifying IoT devices in home networks. Specifically, we aim to answer the following question: "Given a labeled dataset (training) from N homes and an unlabeled dataset (testing) from other M homes, which of the global versus contextualized modeling does yield better performance in classifying devices during the testing phase?" Our **first** contribution highlights the presence of concept drifts over time and space domains in IoT network traffic behavior by analyzing more than 6 million flow records (§III). Our data (labeled training and testing instances) is publicly released [41] to the research community. For our **second** contribution, we develop global and contextualized models (aiming to manage concept drifts in the space domain) and compare their performance (§IV). For the **third** contribution, we demonstrate that a dynamic inference can be applied to a combination of global and contextualized models to address concept drifts in the time domain (§V). In the **fourth** contribution, we develop a selection strategy to select the closest model by using score distributions of contextualized models without using the true labels of the testing dataset (§VI).

II. Related Work

Classifying IoT devices has received attention in the research community during the last recent years [12], [13], [15], [16], [28]–[30], [42], [43]. However, the majority of the existing works need full visibility over the network traffic like relying on local traffic (*e.g.,* SSDP, DHCP, and mDNS) [42], [44]–[46] or they take advantage of device identities like IP/MAC address [25], [26], [29], [43], [47] for grouping their network traffic features over a time window. These methods become practically challenging for an ISP aiming to deliver the device identification service because the ISP would need to deploy its traffic measurement engine *inside* thousands of home networks which requires a software/hardware change to the existing home gateways. Alternatively, if the measurement is done on the ISP network (outside individual home networks), the local traffic and device identities (due to NAT) will be lost.

As traffic measurement scalability is an important factor for ISPs, some research works developed their inference model to infer from partial information available when the measurement is done outside the home network [14], [17], [18], [23]. In [14], authors used domain names extracted from DNS query packets sent by IoT devices to map server IP addresses to their corresponding domain names. Then, they created a bag of domains that each IoT device types communicate with so if a certain number of domains from a given bag were contacted, they would infer the presence of the device type associated with that bag of domains. Work in [17], used a similar inference technique *i.e.,* using the IP address and domain names; but, instead of extracting them from raw network packets, they employed IPFIX [48], and NetFlow [49] which are well-known tools for generating flow-based information in networks. As neither DNS packets nor domain names are accessible through IPFIX and NetFlow, the authors used a third-party tool like DNSDB [50] and Censys [51] to look up domain names. Work in [18] transformed TCP payloads into gray-scale images and trained a machine learning model to detect IoT device types based on that.

Our previous works [23], [24] analyzed data collected from a testbed in our lab environment (training and testing environments were identical) and studied the feasibility of inferring IoT devices from their IPFIX records. We trained an inference model on IPFIX data of IoT devices collected from the testbed and evaluated its performance on the unseen data of the same testbed. This paper leverages the findings of and builds upon our previous works. It specifically extends the scope of network traffic inference to a distributed set of environments (home networks). With the new scope, this paper

primarily aims to study the practical challenges of concept drifts in time and space domains and their impacts on the quality of inference, where data can be collected from, and inference is made across multiple home networks.

Concept drift has been studied by researchers [36]–[39], [52]–[56] in contexts like image recognition, electricity market, and network intrusion detection; however, to the best of our knowledge, no existing work has explored this challenge in the context IoT traffic inference.

Some of the existing works [35]–[40] assume that the true labels are available during the testing phase so they can evaluate the model performance and if they observe an increase of the model's error rate, they detect the concept drift. Another body of works like [57]–[62] use the underlying data distribution to measure the distance between the training and testing data, so if the distance is greater than a threshold, they consider it as concept drift. Although this method is more realistic than relying on the testing labels, it becomes computationally expensive as the number of features (*i.e.,* dimensions) increases [52]. Works in [57], [63] developed decision tree models with specialized leaves capable of measuring the error rate; their methods can detect regions in the feature space that are drifting.

Authors in [31], [32] argue that obtaining true labels could sometimes be delayed or infeasible. Therefore, they used statistical tests to measure the similarity between classification score distributions without requiring labels. A classification score is a number a classification model returns for each prediction. Higher values indicate the model is more confident in its predicted label. They showed that matching the distribution of scores yields acceptable results in detecting concept drifts while it is computationally more attractive than computing the distance in the distribution of multi-dimensional feature space.

Once the concept drift is detected, there are different ways to address it. Works like [36], [53], [64], [65] used the benefits of testing labels to train a new model which replaces the old one. Authors in [39], [55], [66] used an ensemble model that includes several models trained via past data, and once the drift occurs, they train a new model purely using the new data which is added to the ensemble. Works like [67], [68] developed decision trees-based models that can be updated in real-time by replacing low-performance sub-trees with newly trained ones based on new data. Contextualized modeling is another method for handling concept drift [31]–[33], [35], [40] which identifies some recurring contexts and trains a separate model for each context. Then, based on the testing data and a selection strategy, the best model is selected from the previously trained models in the testing phase.

Works in [35], [40] assumed that true labels for testing instances are available to help them select the model that has the best performance for the given test data. In [33], authors used temporal information to select the model that is trained specifically for a certain month of the year or a specific time of the day. Works in [31], [32] select the best model by comparing the classification score distributions of the models on their training dataset and the given test dataset. The model with the closest score distribution between its training and test datasets is selected as the best model. To our knowledge, this paper is the first work highlighting the efficacy of global and contextualized strategies for IoT traffic inference. Our dynamic combination of the two strategies improves the quality of traffic inference by 20%.

Our Novelty: Existing works (ours [23], [24] included) developed inference models to identify/classify connected IoT devices from network traffic. To our best knowledge, no prior work studied the impact of concept drifts on IoT traffic inference. Additionally, no work attempted to manage practical challenges of concept drifts in the context of traffic inference at the sale of ISPs where gaining labeled data is nontrivial, particularly at scale. In this paper, we first quantify the impact of concept drifts in traffic data collected from 12 real homes. We next develop, compare, and combine ideal and practical methods to manage concept drifts for IoT traffic inference.

III. Concept Drifts in Network Behaviors of IoT Devices

A prerequisite step for an ISP that aims at automatically detecting IoT devices in home networks is to collect labeled data of network traffic for an intended set of devices. Obtaining labeled datasets is a common challenge of machine learning applications in different domains. This is even more challenging for entities like ISPs due to the privacy concerns around the Internet traffic data of users. An ISP may choose to obtain labeled datasets in two broad ways:

(a) With offers like discounted monthly bills, the ISP may incentivize subscribers who own and use IoT devices on their home network to contribute certain and well-specified data, allowing the ISP to collect non/less private meta-data from within and/or outside their home network. Obtaining data from subscribers' networks would require the users' explicit consent with a clear explanation of what part of network traffic is being collected and used by the ISP. We note that contributing certain forms of data can preserve user privacy to a great extent, particularly when data is stored as IPFIX records which merely carry 5-tuple information (source/destination IP addresses, transport protocol, and source/destination port numbers) and flow-based statistical measures with no application contents (payloads). In addition, the ISP may employ middleboxes and/or preprocessing techniques to obfuscate packet contents [44] before exporting network traffic to generate IPFIX records to avoid further privacy concerns. In this work, to ensure preserving user privacy, we obfuscated packet contents by zeroing all bytes in the content of recorded packets; or (b) The ISP can set up a limited number of testbeds that each emulates a household with several IoT devices. The ISP must consider a wide range of users' behavior (*e.g.,* single versus family, less active versus highly active) to diversify the data it collects. Although this approach may sound more expensive for the ISP than the user-supplied one, it has no privacy concerns.

Alternatively, suppose collecting raw data with appropriate user consent becomes infeasible, the ISP can employ federated learning, where they ship the model training engines to subscribers' premises where the data belongs and where users have complete control. That way, no private raw data will be sent out of home networks. Instead, trained models will

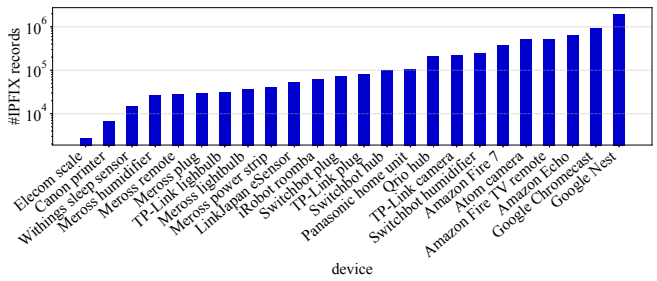


Fig. 1. Number of IPFIX records per device type across 12 homes.

be shared by users with the ISP. Federated learning has been used for IoT traffic classification [47], [69], [70] due to its privacy-preserving advantage, but it is beyond the scope of this paper.

Focusing on labeled data, it is important to note that the number of contexts (real households and/or testbeds) from which the data is collected will be limited and hence can affect the learning process. This paper highlights this practical challenge, quantifies its impacts, and develops various modeling techniques to maximize the inference quality.

Concept drift is a known challenge of inference models that occurs when the distribution of data changes in the testing set from what it was in the training set. This challenge can be perceived differently in the time domain (when the model was trained on data collected some time ago) versus the space domain (when the model was trained on data from different homes with slight variations in context).

In the scope of this paper (inferring IoT devices in home networks), we encounter both sources of concept drifts. The probability and intensity of concept drifts vary by how the ISP trains its inference models (composition of homes) and how often they get re-trained. Frequent re-training can make the models more tolerant to concept drifts in the time domain; however, the re-training process has its practical challenges, such as requiring recent labeled data. Another point to consider is that the ISP may need to carefully select the training context (composition of homes) to cater to diversity, becoming relatively resilient to concept drifts in the space domain.

A. Data Collection

For this paper, we obtained network traffic collected[1] from 12 real homes. The households were selected from volunteer employees (who provided us with written consent) of the specialist contractor in Japan. The selected households have diverse residential settings, including single-person and family[2]. We have a set of 24 IoT device types (makers and models), including two cameras, four power switches, two humidifiers, an air sensor, two speakers, two media streamers, three hubs, two lightbulbs, a weighing scale, a tablet, a printer, a sleep sensor, a smart remote, and a vacuum cleaner. For each of these types, we procured 12 units, meaning a total of 288 IoT units. Individual homes are given a unit of each device type. In other words, each of the 12 homes has its own set of 24 devices.

[1]Traffic collection was outsourced to a third-party specialist contractor.

[2]We have no information about linking households to their corresponding dataset.

TABLE I
ACTIVITY FEATURES OF IPFIX RECORDS.

Feature	Description
`packetTotalCount`	# packet
`octetTotalCount`	# byte
`smallPacketCount`	# packet with $<$ 60 bytes payload
`largePacketCount`	# packet with $\geq$ 220 bytes payload
`nonEmptyPacketCount`	# packet with payload
`dataByteCount`	payload size in total
`averageInterarrivalTime`	packets inter-arrival time μ
`firstNonEmptyPacketSize`	first non-empty payload size
`maxPacketSize`	maximum payload size
`standardDeviationPayloadLength`	payload size σ
`standardDeviationInterarrivalTime`	packets inter-arrival time σ

We collected the data from these homes for 47 days, using the first 30 days for training and the remaining 17 days for testing. One may construct (semi) randomized training/testing datasets by shuffling instances. However, we split our data chronologically for two reasons:

(1) In real-world settings, an ISP would start by collecting data for a certain period (say, a month) to train models before applying them in production. Therefore, learning from past behaviors to predict future behaviors provides results closer to realistic scenarios. In contrast, shuffling data instances can break the chronological order, mixing past and future behaviors that may not necessarily reveal intended insights.

(2) In this work, we highlight the existence of concept drifts in the time domain, meaning the network behavior of devices can change (to some extent) over time. Note that constructing training and testing datasets in ways other than chronologically would not have demonstrated this phenomenon.

To protect user privacy, precautionary actions, particularly for cameras (*e.g.,* ensuring the use of TLS-based encryption for their data transmission and placing them in less private locations of homes), have been taken. Also, the collected data excludes network traffic of other household devices (*e.g.,* personal computers, phones) by connecting a new WiFi access point (implemented with a Raspberry Pi) to the LAN interface of the existing home gateway that only serves our 24 experimental IoT devices (where other household devices are served by their existing gateway) and is the vantage point of data collection. Additionally, the process of traffic capture was configured by the MAC addresses (safelist) of our experimental IoT devices to avoid recording data of unintended devices.

For an ISP serving thousands of households, the choice of traffic data and the measurement location are nontrivial tasks mainly due to operational and scalability challenges. Works in [17], [20], [23] showed that these challenges could be managed by collecting data at the edge of the ISP network (outside homes). This paper also considers that traffic is measured at the same vantage point and that flow records are employed to manage computing costs. That said, it is important to note that to evaluate the efficacy of our inference methods, we collect raw packets (in PCAP format) from inside home networks to obtain true device labels. We transform them into post-NAT IPFIX records [48] (emulating a real scenario) and use them for training and testing models.

Our entire dataset consists of 6,305,626 IPFIX records,

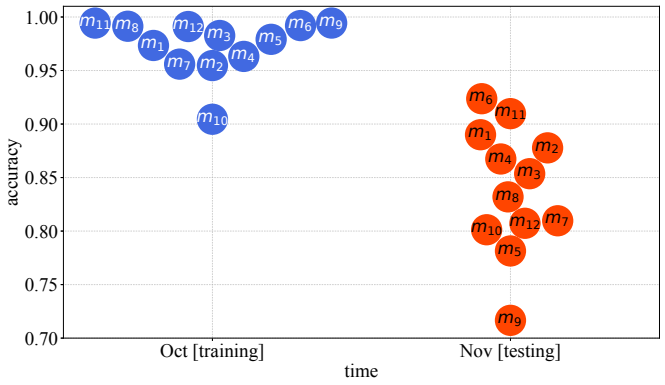


Fig. 2. Temporal drifts in IoT network behaviors lead to performance decay across models (one per each home) from training to testing.

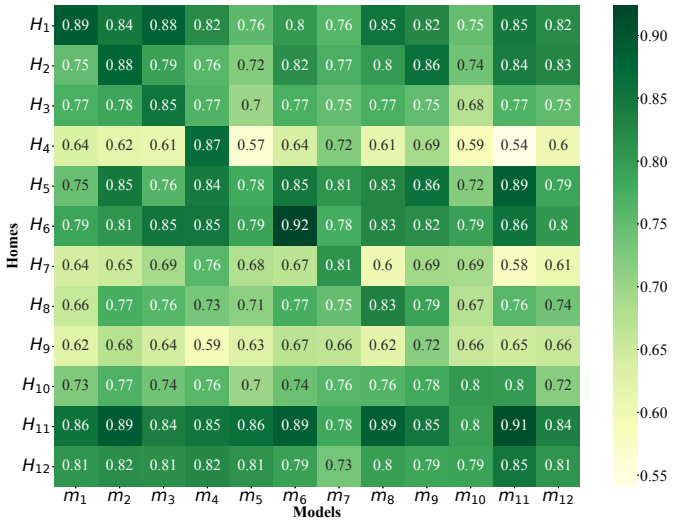


Fig. 3. Spatial drifts in IoT network behaviors deteriorate model performance when tested against data of other contexts.

shown in Fig. 1 for each device type. Each IPFIX record corresponds to a five-tuple flow distinguished by source/destination IP address, source/destination port number, and IP protocol number. IPFIX records are bidirectional, which means they include activity features corresponding to both directions of a flow *e.g.,* incoming packet count and outgoing packet count. Inspired by prior work, we extract 22 activity features from each IPFIX record that indicate packet count, byte count, packet size, and inter-arrival time. Table I shows the IPFIX features we extracted from IPFIX records. As IPFIX records are bi-directional, a reverse equivalent for each of these features captures the activity of the other flow direction; hence, there is a total of 22 activity features. In addition to the activity features, we extract 6 binary features for indicating the protocol of flows into one of the following protocols: `HTTP`, `TLS`, `DNS`, `NTP`, `TCP` (other than `HTTP` and `TLS`), and `UDP` (other than `DNS` and `NTP`). Therefore, we extract a total of 28 features per IPFIX record. Our dataset (labeled instances for each of the 12 home networks) can be found in [41] in the CSV (Comma Separated Values) format.

In this paper, we use multi-class Random Forest to develop our inference models as it has proven effective in network traffic inferencing [71]. Model inputs are IPFIX features; outputs are a device type (from 24 classes) and a classification score. We note that the inference is not bound to any specific algorithm, so one may use other methods like neural networks for this purpose. As IPFIX records are coarse-grained, it is not unlikely for the model to be less confident in its prediction. To increase the prediction quality, we apply class-specific thresholds to the score given by the model, thereby accepting predictions accompanied by relatively high scores. We obtain class-specific thresholds during the training phase by taking the average score for correctly predicted training instances per class.

B. Temporal Drifts in IoT Behaviors

Let us begin with how the behavior of IoT devices changes over time. We train a Random Forest classifier for each home using its corresponding training dataset, obtaining 12 inference models *i.e.,* one model per home ($m_i : i \in [1, 12]$). We use a 10-fold cross-validation technique to obtain average accuracy.

We apply these 12 models to their corresponding training and testing data to track their performance (average prediction accuracy across 24 classes). Fig. 2 shows the accuracy (a value between 0 and 1) of models on average drops from training (shown in blue circles) to testing (red circles). All models realize a lower accuracy in the testing phase than in the training phase. Temporal drifts deteriorated the performance of models by about 13% on average. The smallest and largest gaps are seen for m_6 and m_9, where their testing accuracy drops by more than 6% and 27%, respectively.

C. Spatial Drifts in IoT Behaviors

The second form of drift is when models learn and infer across the spatial domain. To analyze this scenario, we apply the models ($m_i : i \in [1, 12]$) trained in §III-B to the testing set of each home ($H_i : i \in [1, 12]$). Fig. 3 shows the results of this experiment. Each cell is the average prediction accuracy of models (listed across columns) when tested against data of homes (listed across rows).

Overall, models perform better when they apply to their corresponding context. We observe that diagonal cells are often among the highest in each row, with some exceptions. For example, in H_{12}, its own model m_{12} is beaten by m_{11}, giving an accuracy of 0.81 versus 0.85. Such an unexpected pattern is more pronounced in $H5$, where the performance of m_5 is lower than that of eight other models.

Certain homes like H_{11} and H_{12} receive relatively consistent predictions (fairly green cells) from all models. Conversely, in homes like H_4 and H_7, inconsistent performance across models is evident (yellow versus green cells) – spatial drifts deteriorate the performance more. Considering H_4, for example, the accuracy of its own model (m_4) is 0.87, whereas all other models at best give an accuracy of 0.72 – a non-negligible gap of 0.15.

Summary: Concept drifts are unavoidable, particularly at the scale of an ISP with only a limited amount of labeled data available from a diverse set of homes. In the following sections, we will develop strategies and methods to manage the impact of concept drifts on our IoT traffic inference.

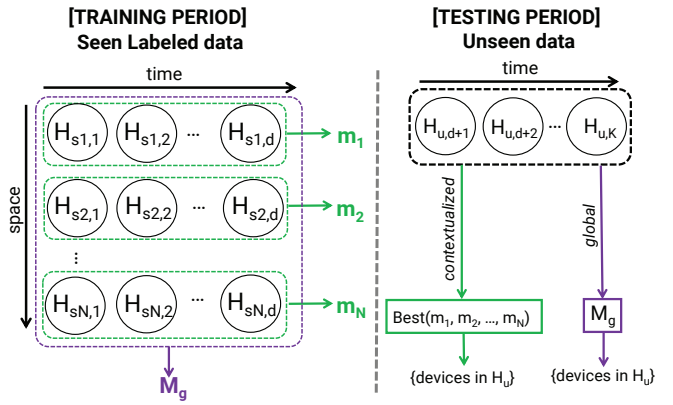


Fig. 4. Global versus contextualized modeling for detecting IoT devices in home networks.

IV. IoT Traffic Inference Strategies

With high-level observations in the previous section, let us assume labeled data is available from some seen contexts (say, N homes). One may employ different strategies to train models and apply them to unseen contexts in practice. We illustrate in Fig. 4 two representative strategies, highlighted by purple (global) and green (contextualized) colors. Suppose we have d days worth of labeled data from N homes, constructing our training set (shown in the left section of Fig. 4). Therefore, $H_{sN,d}$ denotes the unit of dataset collected from the seen home N on day d.

The most straightforward strategy (baseline) is that we combine all labeled data from N homes and train a classifier (shown as purple M_g) based on the combined dataset. We call this global modeling throughout the paper. Alternatively, one may step back and, instead of combing all data, trains a separate model per home (shown as green $m_1, ..., m_N$). We call this method contextualized modeling. When it comes to the testing phase (detecting a set of IoT devices in an unseen home H_u), shown in the right section of Fig. 4, M_g is readily applied to the daily data of $H_{u,K}$, giving prediction. For contextualized modeling, instead, an additional computation is required. It needs to "select the best" model from N available models before applying it to data of an unseen home. In §VI, we develop a strategy for selecting the best model in the absence of labeled data in the testing period; but before that, our primary objective is to compare the efficacy of global versus contextualized modeling, assuming the best model can be selected (either automatically or "given"). For now, we leverage ground truth labels (provided by an oracle[3]) available for all contexts (seen and unseen) during the training and testing phases of our experiments. In other words, we choose the model with the highest accuracy (given ground truth) for unseen homes.

These two strategies have key differences: (1) Suppose a labeled dataset becomes available from an unseen (or even a seen) home. Global modeling requires re-training M_g on past data combined with new data. Contextualized modeling, instead, trains an isolated model specific to a newly added/updated home, which is relatively faster and less expensive computationally; and (2) Although this paper selects the best model by leveraging ground truth labels, a practical approach like what we will explain in §V may not always be able to select the best model from a set of available models (hence, affecting the inference performance).

[3]Note that this would be infeasible in real practice. We will develop a practical method in §VI.

TABLE II
Performance of Global versus Contextualized Modeling.

Run	M_g	$Best(\{m_i\})$	$Best(M_g, \{m_i\})$	$Best(M_g, \{m_i\})^d$
①	0.770	0.748	0.769	0.782
②	0.827	0.805	0.834	0.843
③	0.823	0.803	0.816	0.834
④	0.825	0.791	0.826	0.840
⑤	0.771	0.728	0.768	0.783
⑥	0.822	0.795	0.820	0.819
⑦	0.844	0.782	0.836	0.837
⑧	0.840	0.787	0.838	0.836
⑨	0.841	0.788	0.840	0.831
⑩	0.826	0.806	0.820	0.851
Avg.	**0.818**	**0.783**	**0.816**	**0.826**

Evaluating Inference Strategies: For our evaluation, we assume labeled data of five[4] homes is available for training (seen), and data from the remaining seven homes is used for testing (unseen). To avoid creating bias, we ran our experiments ten times and randomly selected five training homes for each run. Let us call the first month (of the entire 47 days) training period. Note that we train models on data of seen homes during training. We train a global model (M_g) and five contextualized models (m_i) for the chosen seen homes, as shown in Fig. 4.

It is important to note that contextualized modeling requires the best model assigned to an unseen home before the testing phase. Given an unseen home, the best model (one of five m_i's in our evaluation) is selected and assigned based on the highest accuracy obtained by applying m_i's to the labeled data of unseen homes (which is assumed to be available in this paper). It is possible that the best model selected for an unseen home may not necessarily perform the best during the testing period. In fact, in nine runs (out of ten), we found at least one unseen home where the ideal model for its training period differs from the testing period due to temporal concept drifts discussed in §III-B.

Table II summarizes the prediction accuracy (averaged across testing homes) for 10 runs. The second and third columns show the performance of global modeling (*i.e.,* M_g) versus that of contextualized modeling (*i.e.,* $Best(\{m_i\})$), respectively. Global modeling consistently outperforms contextualized modeling "on average" across all runs. However, this pattern is not necessarily present at individual home levels. Let us closely look at a representative run to draw detailed insights. Considering run ②, homes H_1, H_4, H_5, H_7, and H_9 were randomly chosen as seen contexts (resulting in m_i's) and therefore remaining homes H_2, H_3, H_6, H_8, H_{11}, H_{10}, and H_{12} were considered unseen. For unseen home H_2, the

[4]Less than five seen homes result in insufficient data.

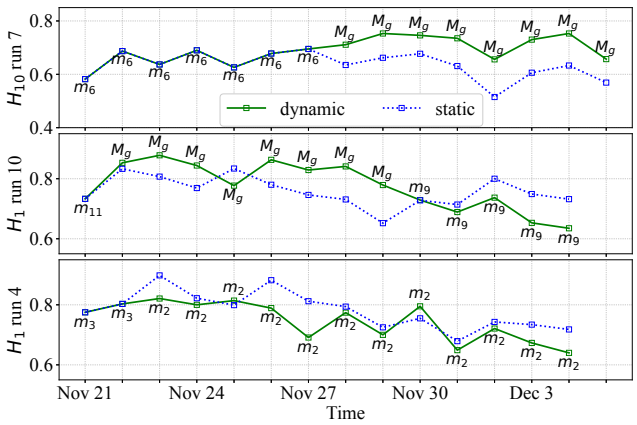


Fig. 5. Accuracy of static versus dynamic models during testing period for three representative homes.

global model gives an accuracy of 0.822; however, the best selected contextualized model m_9 gives an accuracy of 0.860. Similarly, in the same run, for unseen home H_8, the accuracy of the global model is 0.774, but m_9 gives better accuracy of 0.790.

Overall, in half of the ten runs, we found cases of unseen homes whereby the contextualized model outperforms the global model. Scaling our small-size experiment to the size of the operation of an ISP with thousands of homes, one would appreciate the whole argument that either global or contextualized modeling can be sub-optimal; hence, a better strategy is required.

V. Combined and Dynamic Inference for Managing Concept Drifts

In the previous section, we saw that global modeling might sound superior at a macro level but contextualized modeling is not always worse (if not better) at a micro level. Therefore, we choose a strategy whereby a combined capability of global and contextualized modeling is leveraged. In other words, we bring M_g as an additional context to the mix of contextualized models $\{m_i : i \in [1..N]\}$ to select the best model, $Best(Mg, \{m_i\})$, for an unseen home.

Under the fourth column of Table II we report the accuracy of the two approaches combined. Unsurprisingly, the augmented contextualized modeling consistently outperforms its original form (third column of Table II) at a macro level (average accuracy across unseen homes). Also, it can be seen that in some runs like (2) and (4), the performance of the new strategy is slightly better than that of the global M_g. However, M_g marginally wins the competition with the new strategy by the metric of aggregate accuracy. At a micro level (individual homes), we found that the two strategies yield almost the same prediction accuracy for about 60% testing homes across different runs. For the remaining 40% testing homes, the superiority of M_g or the combined strategy is insignificant ($< 1\%$). It is important to note that the best model (of the combined strategy) is selected based on observations during the training period. In other words, our selection of the best model, though it has improved by incorporating M_g into the mix, is still "static", which makes it vulnerable to temporal concept drifts.

TABLE III
Average number of days across 10 runs that dynamic (d) approach gives higher ($>$), lower ($<$), or equal ($=$) accuracy compared to static (s) approach.

	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8	H_9	H_{10}	H_{11}	H_{12}
$d > s$	5	1	0	0	0	0	4	4	0	2	0	1
$d < s$	3	2	1	0	0	1	1	1	0	1	0	1
$d = s$	6	12	14	14	9	14	10	10	15	12	16	14

To overcome this challenge, we slightly refine our inference strategy and make it "dynamic". This means that the process of best model selection continues in the time domain (occurs, say, once a day). Such a change in our inference will certainly introduce additional computing costs (for periodic selection). Instead, the inference process becomes relatively more resilient to temporal concept drifts.

Another practical challenge is that certain homes may not have sufficient instances (only a few IPFIX records due to the limited activity of the corresponding IoT devices) to make a meaningful selection for some days during the testing period. Therefore, we maintain a sliding window equal to the length of our testing period (a month). The sliding window data is considered for the dynamic selection of the best model. Note that we still have M_g in our mix.

The fifth column in Table II shows the performance (average daily accuracy of all unseen homes) of our model selected dynamically. Unsurprisingly, our dynamic combined approach consistently outperforms the baseline of contextualized modeling (third column). In six (out of ten) runs, the performance of the dynamic combined approach is better than that of both M_g (second column) and static combined (third column) approaches. Although M_g performs slightly better in the remaining four runs, its superiority is marginal. Averaging across all runs (the last row in Table II), the dynamic approach gives an aggregate accuracy of 0.826, greater than all other approaches we evaluated.

Note that results in Table II are at an aggregate level. Fig. 5 helps us look closer at the performance of static versus dynamic approaches for three representative homes unseen to our models, each from a certain run, across the testing days. Blue dotted lines correspond to the prediction accuracy of the static combined approach, and solid green lines correspond to the accuracy of the dynamic combined approach. For the dynamic approach, the model selected each day is annotated accordingly. Note that home H_1 was entirely inactive on 5th Dec, and hence no data point for that day.

Let us start from the top plot, corresponding to home H_{10} in run (7). For the first six days (between 21st and 27th Nov), M_6 is the selected model by both static and dynamic approaches. However, after that, the dynamic approach selects M_g, which gives higher accuracy than M_6. Moving to the second plot (H_1 in run (10)), we observe that the dynamic approach outperforms the static one on days between 21st Nov and 30th Nov (except for 25th Nov). However, its superiority fades out from 1st Dec onward. Lastly, for home H_1 in run (4), the static approach defeats the dynamic approach almost every day, except for 25th Nov and 30th. The key takeaway is that a definite winner cannot be easily concluded from these observations.

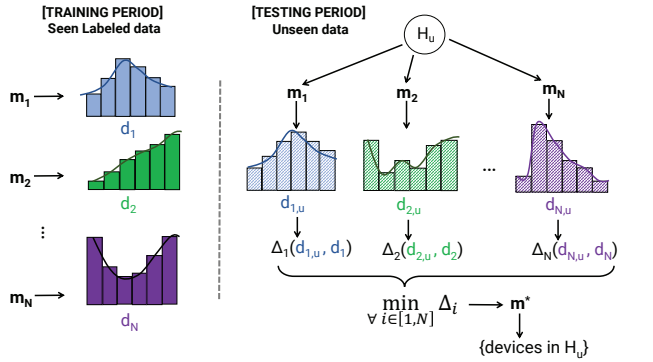


Fig. 6. Selecting the best among contextualized models by measuring distances of score distributions.

Inspired by observations in Fig. 5, we analyze the performance using a different lens. In each run, we count the number of days that the dynamic approach yields lower, higher, and equal accuracy compared to the static approach. We then compute the average count across ten runs. Table III shows the results of this evaluation. While both dynamic and static techniques often select the same model ($D = S$), the dynamic approach outperforms when the two differ. It can be seen that in 17 home days, the dynamic approach wins ($D > S$) versus 11 home days with the static approach winning ($D < S$), meaning more than 50% superiority for the dynamic approach. We also quantified the accuracy delta between these two techniques. When the dynamic approach wins, it outperforms by 0.004, on average. This metric is about 0.002 for the static approach, meaning half of the dynamic approach. With the dynamic approach, we found that for 20% of home days, one of $\{m_i\}$ (instead of M_g) is selected for inference. This means our dynamic combined method improves the baseline inference (by M_g) in a fifth of our experimented scenarios.

VI. Practical Method for Selecting the Best Contextualized Model

So far, we have assumed that labels of unseen data are available, assisting us in selecting the best model (by measuring prediction accuracy). However, this assumption is unrealistic in practical settings, particularly at scale. Therefore, in this section, we develop and evaluate the efficacy of a method independent of unseen labels that selects the closest (best) model based on the labeled training data and unlabeled test data.

For this purpose, one approach is to compare the data distribution of a given unseen home against the seen (labeled) home datasets and select the model trained for the closest seen dataset. This method can be complex and computationally expensive as our data is multi-dimensional, while statistical tests often work best for univariate distributions [72]. Inspired by work in [32], we use the distribution of classification scores (*e.g.,* prediction confidence), a one-dimensional signal, obtained from the models as a proxy for selecting the best model. It is important to note that the model selection only applies to contextualized models where we have a set of models to choose from – the single global model is free from a selection method. We will demonstrate in §VI-B that our method may not pick the ideal model (from §V) in all cases, but the performance of the selected model is close to ideal (upper bound). We will also experiment with a baseline method that randomly selects a model and show that our method (matching distributions of classification scores) often performs better than the baseline.

Fig. 6 illustrates the method of selecting the best model among contextualized models by measuring the distances of score distributions from individual m_i's. For each contextualized model m_i, we obtain its score distribution by applying it to the seen labeled data of H_i during the training period. To get unbiased score distributions, we use 10-fold cross-validation to obtain 10 sets of score distributions (one set in each run). The final score distribution is a superset of all score values obtained from ten runs. We denote by d_i the score distribution of model m_i, shown on the left side of Fig. 6.

Moving to the testing period with an unseen home H_u. We present its data to each of contextualized models ($m_i : 1 \leq i \leq N$) trained on labeled seen data and compute a corresponding score distribution $d_{i,u}$, as shown at the top right of Fig. 6. We next measure the distance between each pair of $d_{i,u}$ and d_i, denoted by $\Delta(d_{i,u}, d_i)$, an approximate measure of distance between the underlying data distributions. Eventually, we select the best model m^* giving the shortest distance $\min \Delta(d_{i,u}, d_i)$ and use it to determine the class of devices connected to home network H_u from its IPFIX flow records. There are several statistical tests for measuring the distance (*i.e.,* the function Δ) of two given distributions. We experiment with four well-known tests described in what follows.

A. Distance Metrics

Let us assume $F(x)$ and $G(x)$ are empirical Cumulative Distribution Functions (eCDF) computed from prediction scores when a classifier is applied to training (d_i) and testing instances ($d_{i,u}$), respectively.

Kolmogorov-Smirnov (KS): KS is a non-parametric fitness test for continuous distributions that measures the greatest distance (supremum) between the two distributions [73].

$$KS(F, G) = \sup |F(x) - G(x)| \tag{1}$$

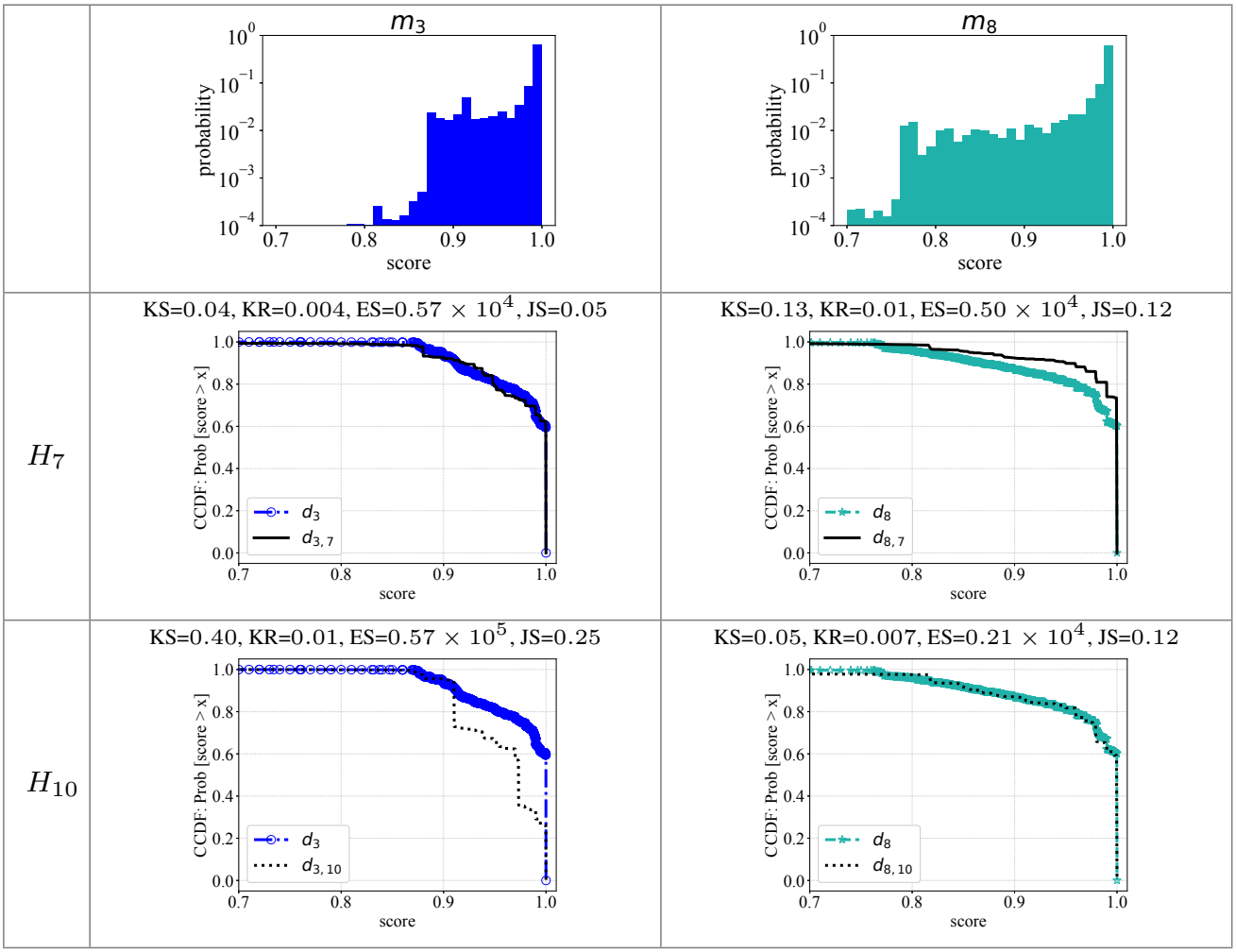
Kantorovich–Rubinstein (KR): KR (also known as Wasserstein distance) is a non-parametric fitness test for two given distributions that measures the amount of change needed to transform a distribution to another [74].

$$KR(F, G) = \int_{-\infty}^{+\infty} |F(x) - G(x)| \tag{2}$$

Epps–Singleton (ES): ES is a parametric test, proved to be more accurate than KS in some cases [75]. ES performs Fourier transform on the given distributions before measuring their distance which can be tuned using certain parameters. The inventors of ES suggested a set of parameters that performed well on different distributions, and we use them in this paper.

Jensen–Shannon (JS): This measure is a symmetric version [76] of Kullback–Leibler divergence that measures the average number of bits lost by approximating $F(x)$ using $G(x)$ [77]. Kullback–Leibler divergence can be measured for both continuous and discrete distributions. However,

TABLE IV
SCORE DISTRIBUTIONS AND DISTANCE MEASURES FOR TWO REPRESENTATIVE UNSEEN HOMES H_7 AND H_{10}, GIVEN TWO REPRESENTATIVE MODELS m_3 AND m_8.



since the measurement for continuous distributions leads to a numerical approximation rather than an exact value [78], we use the discrete version of it. For this purpose, we transform the score distributions to 10 discrete bins (X) with the size of 0.1. Discrete Kullback–Leibler divergence of $F(x)$ and $G(x)$ is defined by:

$$D_{KL}(F,G) = \sum_{x \in X} F(x) \log \frac{F(x)}{G(x)} \tag{3}$$

We compute the JS distance as follows.

$$JS(F,G) = \sqrt{\frac{D_{KL}(F,M) + D_{KL}(G,M)}{2}} \tag{4}$$

where, $M(x) = \frac{F(x)+G(x)}{2}$.

To better understand how the score distributions work, let us consider a scenario where we apply two models m_3 and m_8 (trained on seen data of H_3 and H_8, respectively) to traffic data of two unseen homes H_7 and H_{10}. Table IV summarizes and illustrates score distributions and distance measures for this representative scenario. The top row in this table shows the score distribution (histogram) of the two models from their respective training data. For a better illustration, we limit the scores (x-axis of the plots) to values greater than 0.7, as most scores fall in this range.

In this table's second and third rows, our testing homes are H_7 and H_{10}. Complementary Cumulative Distribution Function (CCDF) plots visually highlight how closely the distribution of H_7 data (solid black lines) matches that of m_3 (dashed blue lines with circle markers). In other words, $d_{3,7}$ is closer to d_3 than d_8. We also see that $d_{8,10}$ (dotted black lines) matches d_8 (dashed green lines with star markers) better than d_3, meaning m_8 is the best model to infer from network traffic of home H_{10}.

In addition, CCDF plots are accompanied by the four quantitative distance metrics (KS, KR, ES, and JS) we consider in this paper. The smaller each metric value is, the closer the two distributions are. For example, the KS metric measures the distance between $d_{3,7}$ and d_3 as 0.04 where this distance is 0.13 between $d_{8,7}$ and d_8, corroborating our visual observation from the CCDF plots in the second row. We also note that the KS distance shows that the data of home H_7 is ten times closer than that of H_{10} to model $m3$ – $KS = 0.04$ versus $KS = 0.40$. Although KS and KR distance metrics consistently support visual similarities in CCDF plots, ES and, to some extent, JS behave differently, particularly for

TABLE V
ACCURACY RATIO OF A DISTANCE-/RANDOM-BASED SELECTED MODEL TO ITS CORRESPONDING IDEAL MODEL.

	Runs	①	②	③	④	⑤	⑥	⑦	⑧	⑨	⑩	**Avg.**
$Best(M_g, \{m_i\})^d$	**KS**	0.955	0.915	0.961	0.911	0.868	0.976	0.943	0.901	0.906	0.991	**0.933**
	KR	0.934	0.907	0.955	0.889	0.885	0.915	0.912	0.876	0.924	0.926	**0.912**
	ES	0.982	0.921	0.971	0.918	0.914	0.974	0.980	0.925	0.904	0.916	**0.940**
	JS	0.980	0.916	0.990	0.876	0.874	0.949	0.967	0.865	0.921	0.890	**0.923**
	RND	0.917	0.939	0.937	0.947	0.911	0.927	0.926	0.908	0.914	0.926	**0.925**
$Best(M_g, \{m_i\})$	**KS**	0.963	0.904	0.974	0.923	0.899	0.976	0.952	0.930	0.917	1.012	**0.945**
	KR	0.957	0.903	0.982	0.891	0.896	0.917	0.920	0.886	0.931	0.964	**0.925**
	ES	0.969	0.925	0.987	0.928	0.918	0.970	0.964	0.930	0.896	0.940	**0.943**
	JS	0.970	0.904	1.005	0.854	0.899	0.936	0.958	0.865	0.923	0.926	**0.924**
	RND	0.908	0.912	0.978	0.941	0.964	0.931	0.973	0.922	0.889	0.912	**0.933**

the scenario of d_8 versus $d_{8,7}$. This is perhaps because minor similarities at a macroscopic level between d_8 and $d_{8,7}$ are more signified by ES and JS. In contrast, ES and JS better conform to KS and KR in the case of d_3 versus $d_{3,10}$, where the two distributions are more distinct (dissimilar), highlighting relatively larger distances.

B. Evaluating Efficacy of Practical Model Selection

We now evaluate the performance of the distance metrics in selecting the best model for a given unseen home. We experiment with our selection strategy over the same ten runs, similar to what was discussed in §IV. To quantify how our selected model (by distance metrics) is close to the ideal model (the upper bound), we measure a ratio as $\frac{A_S}{A_I}$, where A_S and A_I are the average accuracy of the selected and the ideal model, respectively. A higher ratio indicates how our method performs close to ideal. Note that this ratio is expected to be less than 1. However, it may go beyond 1 when the static inference strategy (§IV) is employed. It is because the static inference selects the best model based on training period data. When the selected model is applied to testing data, the upper bound A_I is no longer applicable (A_S can be greater than A_I). We observed and explained similar patterns in Table II.

Baseline: To compare with our method, let us establish a baseline for selecting a model. We choose a random selection method by randomly picking one of the six available models (five contextualized models $\{m_i\}$ plus a global model M_g) at the time of inference for a given unseen test home.

Table V summarizes our evaluation by reporting the accuracy ratio obtained from the selected model based on the four distance metrics plus the baseline method (random selection denoted by "RND") across ten runs. Let us start with the dynamic inference, rows corresponding to $Best(M_g, \{m_i\})^d$ in Table V. It can be seen that all metrics perform relatively well as their selected models are close to ideal (giving ratios greater than 0.9). Given a run, the cell of the best-performing method is shaded in light green. ES seems to outperform the others by winning in 40% of the runs (①, ⑤, ⑦, and ⑧), as well as by the average ratio of 0.940 across the ten runs. The baseline RND wins in two runs ② and ④, but on average (0.925) ranks the third after ES and KS. Regarding the standard deviation of accuracy ratios, the four metrics plus the baseline measure around 0.04, indicating relatively consistent performance across ten runs. In addition to the accuracy ratio, we measured the absolute delta of the overall accuracy $|A_S - A_I|$, which turns out to be about 0.05, 0.07, 0.04, 0.06, and 0.06 for KS, KR, ES, JS, and RND, respectively, on average across ten runs. Again, by this measure, ES slightly outperforms its rivals, yielding a relatively minor loss of prediction accuracy compared to the upper bound (the ideal model) in practical settings.

Moving to the static inference, rows corresponding to $Best(M_g, \{m_i\})$ in Table V, it can be seen that the selected model sometimes outperforms the ideal model, giving a ratio of more than 1. We now observe that KS is the winner metric (marginally) by giving the highest ratio in three runs and yielding an average ratio of 0.945. Similar to what we saw for the dynamic inference, the baseline RND comes third after KS and ES metrics. The standard deviation of accuracy ratios is about 0.05, not much different from what was seen in the dynamic inference. Finally, the absolute accuracy delta in the static inference is about 0.04 for KS and ES, 0.06 for KR and JS metrics, and 0.05 for RND.

F1-score vs. Accuracy: Throughout this paper, we have been using accuracy as the performance metric of our inference models – the best model was the one that yields the highest overall accuracy across IoT classes. Let us now evaluate the efficacy of inference models using two other well-known metrics, precision and recall. Precision is computed by the number of correctly predicted IoT classes (which indeed connect to the network) divided by the total number of predicted classes (correct and incorrect) in a given home. In other words, the precision metric indicates how precise the prediction of an inference model is, which decreases by false positives (those predicted IoT classes that are not present on the target home network). On the other hand, recall is the number of correctly predicted IoT classes divided by the number of IoT classes connected to a home network. Recall decreases by false negatives (missing classes that indeed exist in the home network. Precision and recall are incomplete and cannot be reliable performance indicators individually. However, combining these two metrics gives a better picture

TABLE VI
F_1-SCORE RATIO OF A DISTANCE-/RANDOM-BASED SELECTED MODEL TO ITS CORRESPONDING IDEAL MODEL.

	Runs	①	②	③	④	⑤	⑥	⑦	⑧	⑨	⑩	**Avg.**
$Best(M_g, \{m_i\})^d$	**KS**	0.986	0.988	0.989	0.992	0.986	0.995	0.990	0.994	0.992	0.9974	**0.991**
	KR	0.991	0.989	0.989	0.992	0.986	0.993	0.989	0.991	0.994	0.984	**0.990**
	ES	0.992	0.994	0.993	0.993	0.984	0.995	0.996	0.995	0.987	0.992	**0.992**
	JS	0.993	0.991	0.993	0.989	0.985	0.994	0.991	0.987	0.991	0.981	**0.990**
	RND	0.988	0.991	0.988	0.995	0.985	0.988	0.986	0.989	0.987	0.983	**0.988**
$Best(M_g, \{m_i\})$	**KS**	1.000	0.984	1.003	1.005	0.990	1.007	1.008	0.997	0.997	0.994	**0.999**
	KR	1.007	0.981	1.008	1.002	0.999	1.008	1.008	1.000	1.003	0.997	**1.001**
	ES	0.997	0.991	1.015	1.002	0.988	1.004	1.008	0.997	0.997	0.987	**0.998**
	JS	1.007	0.984	1.015	0.992	0.990	1.008	1.004	0.991	0.994	0.984	**0.997**
	RND	1.001	0.994	1.007	1.004	0.990	0.994	1.008	0.981	0.997	0.981	**0.996**

of the model performance. F_1-score, the harmonic mean of precision and recall, is widely used as a performance measure in machine learning applications. F_1-score is computed by:

$$F_1\text{-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5}$$

Note that our objective is to select a model with the highest F1-score. Table VI reports the ratio of F_1-score ratio of a distance-/random-based selected model to that of the ideal model across ten runs with dynamic and static inference strategies. Given a strategy, the best-performing model of each run is highlighted. Though various metrics perform very closely, ES is the winner of the dynamic inference by giving the highest F1-score ratio in half of the runs and the average ratio of 0.992. At the same time, KR outperforms other metrics with the static inference by giving the highest F1-score ratio in 70% of the runs and the average ratio of 1.001. Compared with the metric-selected models, RND has the lowest F_1-score ratio in dynamic and static strategies. It is important to note that distance-/random-based selected models yield a better F1-score ratio in Table VI compared to the accuracy ratio in Table V.

VII. CONCLUSION

Network operators of various sizes increasingly employ machine learning-based models to infer passively from their network traffic, identifying and characterizing connected assets. Vulnerable consumer IoT devices coming online in home networks are particularly interesting to residential ISPs who are tasked to manage large-scale networks. This paper focused on the practical challenges of concept drifts in modeling the behavior of IoT devices in traffic flows from residential networks. We collected and analyzed over 6 million IPFIX flow records from 12 homes, each serving 24 IoT device types – we publicly released our training and testing instances data. We quantified the impact of concept drifts in traffic data across time and space domains. Given concept drifts, we next quantitatively compared the performance of two broad inference strategies: global (one model trained on aggregate data from all seen homes) versus contextualized (one model per seen home) when predicting traffic data of unseen homes. We dynamically combined the capabilities of the two strategies, improving the baseline inference by 20%. Finally, we developed a practical method to automatically and dynamically select the best model with overall accuracy and F1-score very close (94% and 99%, respectively) to those of the upper bound from the ideal model without needing labels of unseen data for a realistic inference.

ACKNOWLEDGMENT

Funding for this project was partially provided by the National Institute of Information and Communications Technology (NICT) in Japan, Project No. 05201.

REFERENCES

[1] Earthweb, "Smart Home Statistics," 2022. [Online]. Available: https://earthweb.com/smart-home-statistics/
[2] Security Sales & Integration, "Global Smart Home Market Projected to Reach $158B by 2024," 2020. [Online]. Available: https://www.securitysales.com/research/global-smart-home-158b-2024/
[3] World Economic Forum, "The Market for Smart Home Devices is Expected to Boom Over the Next 5 Years," 2022. [Online]. Available: https://www.weforum.org/agenda/2022/04/homes-smart-tech-market/
[4] IoT Business News, "The 1,200 IoT Companies that are Creating the Connected World of the Future," 2021. [Online]. Available: https://bit.ly/3CG4fJt
[5] D. Aronoff, "Top 5 IoT Vulnerability Exploits in Smart Homes," 2020. [Online]. Available: https://bit.ly/3f5ElXo
[6] SAM Seamless Network, "Security IoT Landscape," 2021. [Online]. Available: https://securingsam.com/wp-content/uploads/2022/04/SAM_IOT-Security-Report.pdf
[7] D. Harkin *et al.*, "Consumer IoT and its under-regulation: Findings from an Australian Study," *Policy & Internet*, vol. 14, no. 1, pp. 96–113, 2022.
[8] Bitdefender, "Common IoT Devices Become the ISPs' Worst Enemy," 2020. [Online]. Available: https://businessinsights.bitdefender.com/common-iot-devices-become-the-isps-worst-enemy
[9] PaloAlto, "Unit 42 IoT Threat Report," 2020. [Online]. Available: https://unit42.paloaltonetworks.com/iot-threat-report-2020/
[10] Avira, "Avira SafeThings," 2022. [Online]. Available: https://oem.avira.com/en/solutions/safethings-for-router-manufacturers
[11] MITRE Corporation, "Common Vulnerabilities and Exposures." [Online]. Available: https://cve.mitre.org/
[12] A. Sivanathan *et al.*, "Classifying IoT Devices in Smart Environments Using Network Traffic Characteristics," *IEEE TMC*, vol. 18, no. 8, pp. 1745–1759, Aug 2019.
[13] B. Bezawada *et al.*, "Behavioral Fingerprinting of IoT Devices," in *Proc. ACM ASHES*, Toronto, Canada, Oct 2018.
[14] H. Guo *et al.*, "IP-Based IoT Device Detection," in *Proc. ACM IoT S&P*, Budapest, Hungary, Aug 2018.
[15] N. Msadek *et al.*, "IoT Device Fingerprinting: Machine Learning based Encrypted Traffic Analysis," in *Proc. IEEE WCNC*, Marrakesh, Morocco, Apr 2019.

[16] Y. Meidan *et al.*, "ProfilIoT: A Machine Learning Approach for IoT Device Identification Based on Network Traffic Analysis," in *Proc. SAC*, Marrakesh, Morocco, Apr 2017.

[17] S. J. Saidi *et al.*, "A Haystack Full of Needles: Scalable Detection of IoT Devices in the Wild," in *Proc. ACM IMC*, New York, USA, Oct 2020.

[18] J. Kotak *et al.*, "IoT Device Identification Using Deep Learning," in *Proc. CISIS*, Seville, Spain, May 2021.

[19] X. Ma *et al.*, "Inferring Hidden IoT Devices and User Interactions via Spatial-Temporal Traffic Fingerprinting," *IEEE/ACM Transactions on Networking*, vol. 30, no. 1, pp. 394–408, Feb 2022.

[20] Y. Meidan *et al.*, "A Novel Approach For Detecting Vulnerable IoT Devices Connected Behind a Home NAT," *Computers & Security*, vol. 97, pp. 1–23, Oct 2020.

[21] A. Hamza *et al.*, "Clear as MUD: Generating, Validating and Applying IoT Behavioral Profiles," in *Proc. ACM S&P*, Aug 2018.

[22] R. Trimananda *et al.*, "PingPong: Packet-Level Signatures for Smart Home Device Events," in *Proc. NDSS*, San Diego, California, Feb 2019.

[23] A. Pashamokhtari *et al.*, "Inferring Connected IoT Devices from IPFIX Records in Residential ISP Networks," in *Proc. IEEE LCN*, Virtual Event, Canada, Oct 2021.

[24] ——, "Combining Stochastic and Deterministic Modeling of IPFIX Records to Infer Connected IoT Devices in Residential ISP Networks," *to appear in IEEE IoTJ*, 2022.

[25] A. Sivanathan *et al.*, "Detecting Behavioral Change of IoT Devices Using Clustering-Based Network Traffic Modeling," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7295–7309, 2020.

[26] ——, "Managing IoT Cyber-Security Using Programmable Telemetry and Machine Learning," *IEEE TNSM*, vol. 17, no. 1, pp. 60–74, Mar 2020.

[27] R. Kolcun *et al.*, "Revisiting IoT Device Identification," in *Proc. IFIP TMA*, Virtual, Sep 2021.

[28] S. Marchal *et al.*, "AuDI: Toward Autonomous IoT Device-Type Identification Using Periodic Communication," *IEEE JSAC*, vol. 37, no. 6, pp. 1402–1412, Jun 2019.

[29] M. Miettinen *et al.*, "IoT SENTINEL: Automated Device-Type Identification for Security Enforcement in IoT," in *Proc. IEEE ICDCS*, Atlanta, USA, Jun 2017.

[30] V. Thangavelu *et al.*, "DEFT: A Distributed IoT Fingerprinting Technique," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 940–952, Feb 2019.

[31] D. M. Reis *et al.*, "Unsupervised Context Switch for Classification Tasks on Data Streams with Recurrent Concepts," in *Proc. ACM SAC*, Pau, France, Apr 2018.

[32] ——, "Classifying and Counting with Recurrent Contexts," in *Proc. ACM SIGKDD*, London, United Kingdom, Jul 2018.

[33] N. Nascimento *et al.*, "A Context-Aware Machine Learning-Based Approach," in *Proc. CASCON*, Markham, Canada, Oct 2018.

[34] J. B. Gomes *et al.*, "Tracking Recurrent Concepts Using Context," *Intelligent Data Analysis*, vol. 16, pp. 803–825, Sep 2012.

[35] Y. Yang *et al.*, "Combining Proactive and Reactive Predictions for Data Streams," in *Pro. ACM SIGKDD*, Chicago, USA, Aug 2005.

[36] A. Bifet *et al.*, "Learning from Time-Changing Data with Adaptive Windowing," in *Proc. SDM*, Minneapolis, USA, Apr 2007.

[37] ——, "Adaptive Learning from Evolving Data Streams," in *Proc. IDA*, Lyon, France, Aug 2009.

[38] J. Gama *et al.*, "Learning with drift detection," in *Proc. SBIA*, Sao Luis, Brazil, Sep 2004.

[39] N. Oza, "Online Bagging and Boosting," in *Proc.*, Waikoloa, USA, Oct 2005.

[40] J. Gama *et al.*, "Tracking Recurring Concepts with Meta-learners," in *Proc. Progress in Artificial Intelligence*, Berlin, Germany, Oct 2009.

[41] A. Pashamokhtari, N. Okui, M. Nakahara, A. Kubota, G. Batista, and H. Habibi Gharakheili, "IoT IPFIX Home Dataset," 2023. [Online]. Available: https://iotanalytics.unsw.edu.au/homedataset.html

[42] R. A. Sharma *et al.*, "Lumos: Identifying and Localizing Diverse Hidden IoT Devices in an Unfamiliar Environment," in *Proc. USENIX Security*, Boston, USA, Aug 2022.

[43] A. Bremler-Barr *et al.*, "IoT or NoT: Identifying IoT Devices in a Short Time Scale," in *Proc. IEEE/IFIP NOMS*, Budapest, Hungary, Apr 2020.

[44] D. Huang *et al.*, "IoT Inspector: Crowdsourcing Labeled Network Traffic from Smart Home Devices at Scale," *ACM IMWUT*, vol. 4, no. 2, pp. 1–12, Jun 2020.

[45] D. Kumar *et al.*, "All Things Considered: An Analysis of IoT Devices on Home Networks," in *Proc. USENIX Security*, Santa Clara, USA, Aug 2019.

[46] M. Mazhar *et al.*, "Characterizing Smart Home IoT Traffic in the Wild," in *Proc. IEEE/ACM IoTDI*, Los Alamitos, USA, Apr 2020.

[47] T. D. Nguyen *et al.*, "DÏoT: A Federated Self-learning Anomaly Detection System for IoT," in *IEEE ICDCS*, Dallas, USA, Jul. 2019.

[48] B. Trammell *et al.*, "Bidirectional Flow Export Using IP Flow Information Export (IPFIX)," RFC 5103, Jan 2008. [Online]. Available: https://rfc-editor.org/rfc/rfc5103.txt

[49] Cisco, "Cisco NetFlow." [Online]. Available: https://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-netflow/index.html

[50] F. Security, "DNSDB," 2017. [Online]. Available: https://www.dnsdb.info

[51] Censys, "Censys." [Online]. Available: https://search.censys.io/

[52] J. Lu *et al.*, "Learning under Concept Drift: A Review," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 12, pp. 2346–2363, 2019.

[53] S. Xu *et al.*, "Dynamic Extreme Learning Machine for Data Stream Classification," *Neurocomputing*, vol. 238, pp. 433–449, 2017.

[54] D. Liu *et al.*, "FP-ELM: An Online Sequential Learning Algorithm for Dealing with Concept Drift," *Neurocomputing*, vol. 207, pp. 322–334, 2016.

[55] N. C. Oza *et al.*, "Experimental Comparisons of Online and Batch Versions of Bagging and Boosting," in *Proc. ACM KDD*, San Francisco, USA, Aug 2001.

[56] A. Bifet *et al.*, "Leveraging Bagging for Evolving Data Streams," in *Proc. PKDD*, Barcelona, Spain, Sep 2010.

[57] T. Dasu *et al.*, "An Information-Theoretic Approach to Detecting Changes in Multi-Dimensional Data Streams," in *Proc. ISCSA*, Citeseer, 2006.

[58] A. A. Qahtan *et al.*, "A PCA-Based Change Detection Framework for Multidimensional Data Streams: Change Detection in Multidimensional Data Streams," in *Proc. ACM KDD*, Sydney, Australia, Aug 2015.

[59] F. Gu *et al.*, "Concept Drift Detection Based on Equal Density Estimation," in *Proc. IJCNN*, Vancouver, Canada, Jul 2016.

[60] J. Shao *et al.*, "Prototype-Based Learning on Concept-Drifting Data Streams," in *Proc. ACM KDD*, New York, USA, Aug 2014.

[61] M. van Leeuwen *et al.*, "StreamKrimp: Detecting Change in Data Streams," in *Proc. PKDD*, Antwerp, Belgium, Sep 2008.

[62] X. Song *et al.*, "Statistical Change Detection for Multi-Dimensional Data," in *Proc. KDD*, San Jose, USA, Aug 2007.

[63] J. a. Gama *et al.*, "Learning with Local Drift Detection," in *Proc. ADMA*, Xi'an, China, Aug 2006.

[64] S. H. Bach *et al.*, "Paired Learners for Concept Drift," in *Proc IEEE ICDM*, Pisa, Italy, Dec 2008.

[65] D. Han, C. Giraud-Carrier, and S. Li, "Efficient Mining of High-Speed Uncertain Data Streams," *Applied Intelligence*, vol. 43, no. 4, p. 773–785, 2015.

[66] H. M. Gomes *et al.*, "Adaptive Random Forests for Evolving Data Stream Classification," *Machine Learning*, vol. 106, no. 9, pp. 1469–1495, 2017.

[67] G. Hulten *et al.*, "Mining Time-Changing Data Streams," in *Proc. ACM KDD*, San Francisco, USA, Aug 2001.

[68] J. a. Gama *et al.*, "Accurate Decision Trees for Mining High-Speed Data Streams," in *Proc. ACM KDD*, Washington, USA, Aug 2003.

[69] A. Feraudo *et al.*, "CoLearn: Enabling Federated Learning in MUD-Compliant IoT Edge Networks," in *Proc. ACM EdgeSys*, Heraklion, Greece, Apr. 2020.

[70] M. Abbasi *et al.*, "FLITC: A Novel Federated Learning-Based Method for IoT Traffic Classification," in *Proc. IEEE SMARTCOMP*, Helsinki, Finland, Jun. 2022.

[71] J. Zhao *et al.*, "Network Traffic Classification for Data Fusion: A Survey," *Information Fusion*, vol. 72, pp. 22–47, 2021.

[72] D. W. Scott, *The Curse of Dimensionality and Dimension Reduction*. John Wiley, Ltd, 1992, ch. 7, pp. 195–217.

[73] J. L. Hodges, "The Significance Probability of the Smirnov Two-sample Test," *Arkiv för Matematik*, vol. 3, pp. 469–486, 1958.

[74] L. V. Kantorovich, "Mathematical Methods of Organizing and Planning Production," *Management Science*, vol. 6, no. 4, pp. 366–422, 1960.

[75] T. Epps and K. J. Singleton, "An Omnibus Test for the Two-sample Problem Using the Empirical Characteristic Function," *Journal of Statistical Computation and Simulation*, vol. 26, no. 3-4, pp. 177–203, 1986.

[76] J. Lin, "Divergence Measures Based on the Shannon Entropy," *IEEE Transactions on Information theory*, vol. 37, no. 1, pp. 145–151, 1991.

[77] S. Kullback and R. A. Leibler, "On Information and Sufficiency," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79 – 86, 1951.

[78] I. Goldenberg and G. I. Webb, "Survey of distance measures for quantifying concept drift and shift in numeric data," *Knowledge and Information Systems*, vol. 60, no. 2, pp. 591–615, 2019.

Arman Pashamokhtari received his B.Sc. degree in Computer Engineering from the Amirkabir University of Technology in Tehran, Iran, in 2019. He is currently pursuing Ph.D. degree in the area of computer networks from the University of New South Wales (UNSW) in Sydney, Australia. His research interests include programmable networks, IoT network traffic analytics and applied machine learning.

Norihiro Okui received the B.E. and M.E. degrees in Computer Science and Engineering from Waseda University, Japan, in 2010 and 2012, respectively. He joined KDDI in 2012. He is a research engineer at the Cyber Security Lab. in KDDI Research, Inc. His research interest includes cyber security for IoT.

Masataka Nakahara received the B.Eng. degree in Electrical Engineering and the M. Informatics degree from the Graduate School of Informatics from Kyoto University, Japan, in 2014 and 2016, respectively. He joined KDDI in 2016 and joined KDDI Research, Inc. in 2019. His current research interest includes cyber security for IoT.

Ayumu Kubota received the B.E. and M.E degrees from Kyoto University, Japan, in 1993 and 1995, respectively. He joined KDD (now KDDI) in 1995 and has been researching computer networks and cyber security.

Gustavo Batista received his MSc and PhD degrees in Computer Science from the University of Sao Paulo, Brazil, in 1997 and 2003, respectively. He was an associate professor at the University of Sao Paulo (2007-2018) and an associate professor at the University of New South Wales (UNSW). His research interests fall in Machine Learning, including time series and data stream quantification and classification.

Hassan Habibi Gharakheili received his B.Sc. and M.Sc. degrees in Electrical Engineering from the Sharif University of Technology in Tehran, Iran, in 2001 and 2004, respectively, and his Ph.D. in Electrical Engineering and Telecommunications from the University of New South Wales (UNSW) in Sydney, Australia in 2015. He is currently a Senior Lecturer at UNSW Sydney. His research interests include programmable networks, learning-based networked systems, and data analytics in computer systems.