Cross-network Embeddings Transfer for Traffic Analysis

Cross-network Embeddings Transfer for Traffic Analysis

Luca Gioacchini, Marco Mellia, Luca Vassio, Idilio Drago, Giulia Milan, Zied Ben Houidi, Dario Rossi

***Abstract*—Artificial Intelligence (AI) approaches have emerged as powerful tools to improve traffic analysis for network monitoring and management. However, the lack of large labeled datasets and the ever-changing networking scenarios make a fundamental difference compared to other domains where AI is thriving. We believe the ability to transfer the specific knowledge acquired in one network (or dataset) to a different network (or dataset) would be fundamental to speed up the adoption of AI-based solutions for traffic analysis and other networking applications (e.g., cybersecurity). We here propose and evaluate different options to transfer the knowledge built from a provider network, owning data and labels, to a customer network that desires to label its traffic but lacks labels. We formulate this problem as a domain adaptation problem that we solve with embedding alignment techniques and canonical transfer learning approaches. We present a thorough experimental analysis to assess the performance considering both supervised (e.g., classification) and unsupervised (e.g., novelty detection) downstream tasks related to darknet and honeypot traffic. Our experiments show the proper transfer techniques to use the models obtained from a network in a different network. We believe our contribution opens new opportunities and business models where network providers can successfully share their knowledge and AI models with customers.**

***Index Terms*—Darknets, network monitoring, transfer learning, representation learning, domain adaptation.**

I. Introduction

Artificial Intelligence (AI) and Deep Learning (DL) based approaches are fundamental to process network traces and address traffic classification, anomaly detection, cybersecurity, and other networking problems [1]–[5]. These applications share a common task: characterize and understand the activity performed by hosts or other entities in the network, eventually highlighting relevant phenomena.

While the application of AI in network traffic analysis and cybersecurity is growing, works typically assume the availability of both data and labels to train models. However, the availability of labeled data (i.e., the ground truth) is often the major bottleneck in the building of AI solutions for traffic analysis. The extraction of such labels is still a cumbersome and costly process, generally done in a custom fashion for each network and task. In addition, the ever-evolving Internet scenarios require continuous updates of both models and labels. This scenario faced in networking problems is strongly opposed to what we observe in domains where AI-based approaches are thriving. For example, problems in computer vision (CV) and natural language processing (NLP) are more stable, e.g., with slow changes in the interpretation of images and the meaning of words.

In fact, the sharing of knowledge distilled in models is fundamental to scaling the deployment of AI algorithms in practical traffic analysis scenarios. In a nutshell, the broad success of AI-based techniques hinges on the ability to transfer the knowledge built in a system to another system. Two fundamental questions hence arise: i.e., *Do the models built on a network generalize to other networks?* If not, *how to transfer the knowledge acquired in one network to another network?*

We here set out to give pragmatic answers to the two questions, which we formulate as a *domain adaptation problem* that we solve by proposing *canonical transfer learning* and *explicit alignment* approaches [6], [7] in the networking scenario. Transfer learning is the process of transferring knowledge acquired from one source domain to another (often data-scarce) target domain. Domain adaptation is the case in which the tasks are similar, but the domains are different, such that the source models cannot be applied as-is. In this work, we consider the case in which a network operator (*provider network*) possesses (i) data, and (ii) (some) labels, and desires to keep both private. A second network (*customer network*) has data but no labels and looks for the support of the provider network to analyze the traffic it observes.

In CV and NLP, the process of learning generic and intermediate embeddings from complex data has been proven key to solving final tasks. This trend has been illustrated in NLP [8], [9] with the availability of Large Language Models like BERT [10] or GPT [11] empowering powerful automatic translation or interactive chat tasks. Similarly in CV, a linear classifier on top of learned embeddings outperforms state-of-the-art models in few-shot image classification [12]. More recently, the approach has gained popularity in the context of network traffic analysis [13]–[17].

We here follow the same principle. We use domain adaptation to solve network monitoring use cases: (i) the classification of hosts sending traffic to *darknets*, i.e., portion of the IP address space that passively observes packets scanners send,[1] and (ii) the transfer of the knowledge acquired from *honeypots* to classify hosts scanning *darknets*. Our results show that the labels learned from the honeypot successfully

Luca Gioacchini, Marco Mellia, and Luca Vassio are with Politecnico di Torino, Italy (email: first.last@polito.it). Idilio Drago is with Università di Torino, Italy (email: idilio.drago@unito.it). Giulia Milan is with the University of California, San Diego, USA (email: gimilan@ucsd.edu). Zied Ben Houidi and Dario Rossi are with Huawei Technologies Co.LTD, France (email: first.mid.last@huawei.com).

[1]Darknets (or network telescopes) shall not be confused with the Darkweb.

transfer to the darknet after alignment or transfer is in place. In fact, the customer can exploit the rich honeypot labels to discover the activities of bots and scanners that were previously unknown in the darknet trace. All in all, the knowledge transfer lets the customer increase its knowledge, labeling 38% more hosts.

We believe that the ability to transfer knowledge from provider to customer networks paves the road to the successful adoption of AI pipelines in networking use cases. Upon domain adaptation, this enables the possible creation of new business and collaboration models for network traffic analysis towards a model-as-a-service approach.

We next discuss related work (Section II), formalize the problem (Section III) and present our domain adaptation approaches (Section IV). We then discuss the case studies and results (Section V– Section VIII), before summarizing the results and discuss open challenges (Section IX).

II. Related work

Despite promising recent applications [13], [14], [16], [18], [19], the use of self-supervised AI to learn embeddings from network data is still in its infancy. The same applies to domain adaptation on network-related tasks. To the best of our knowledge, we are the first to investigate domain adaptation for cross-network embedding in network traffic analysis.

A. Use of word embeddings in communication networks

Recent efforts have noted the similarities between *sequences* of events observed in network data and of words in natural languages [13], [14], [16]–[19]. Following this intuition, they leverage a two-step pipeline in which embeddings are first learned using techniques like Word2Vec [20], then used in downstream machine learning tasks. Authors of [19] leverage Word2Vec to learn vector representations of bash commands observed in honeypot logs. Authors of [18] follow a similar idea to learn representations for router configuration commands for the purpose of synthesis, verification, and translation. Close to our work, IP2Vec [13], DANTE [14] and DarkVec [16] first learn host embeddings leveraging the sequence of packets observed in network logs. We build on our previous work DarkVec [16] and its incremental training extension i-DarkVec [17], with openly available code, data, and labels. We introduce novel ideas to the approach, allowing domain adaptation for downstream ML tasks across networks.

As done in NLP to build language models, we obtain the embedding representation by formulating a self-supervised problem where we train a neural network using a masked model [21], i.e., a model that predicts the hosts appearing in the sequence by masking part of them. Differently from NLP where one can build the word embeddings once and then use them multiple times [22], the evolving nature of the internet traffic calls for a continuous update of host embeddings.

B. Domain adaptation in communication networks

Domain adaptation has recently attracted the attention of the network community for various use cases, from wireless [23]–[27] to traffic classification [28], [29], from anomaly detection [30], [31] to network management [32]. Here we are the first to explore domain adaptation problems in network traffic analysis.

As stressed by the authors of [25], existing domain adaptation techniques, e.g., as developed for computer vision or natural language processing, often cannot be used as-is for other types of data. Therefore, several custom domain adaptation strategies have been devised per use case. Examples include WiFi-based human activity recognition [23], [27], localization [24], [25] and signal detection in ambient backscatter communication [26]. Different from the above, we define and evaluate a generic domain adaptation pipeline in the context of traffic analysis.

Some recent efforts focus on system aspects of transfer learning [33]–[35] that are orthogonal to our work. Chen *et al.* present an optimizing task allocation in distributed transfer learning systems [33]. Cartel [34] is an example of such distributed transfer learning systems at the edge. Subsequent work further focused on the operational costs of running a distributed transfer learning infrastructure [35].

C. Word embeddings adaptation in NLP

From a methodological viewpoint, the closest problem to ours is language translation, where embeddings learned in a source language are mapped to their counterparts in the target language. Mikolov et al. [36] noticed that an alignment of the source and target Word2Vec [20] embeddings using a few *anchors* could help translate across languages (domains) in a nearly unsupervised manner. They found that a linear mapping learned on a few source embeddings and their target counterparts can align the embeddings and perform word translation. Since Mikolov's work, several supervised [37]–[39] and unsupervised [40]–[43] embedding translation and alignment strategies have been devised.

Casting the above to the network traffic domain, a host observed in a network shares the behavior with other hosts observed in other networks. However, they have different IP addresses, and no common models are available to classify all hosts. It would therefore be desirable to align the host embeddings trained on a network so that they can also serve other (possibly label-scarce) networks. However, although gathered via NLP techniques, host embeddings learned from network traffic are different from those learned from natural language: due to the much more frequent changes in network traffic, we find that linear-based mapping, like the one successfully used in NLP [36], does not suffice for the translation of host embeddings across networks.

III. Problem Definition

A. Processing pipeline

Figure 1 presents the overall processing pipeline. The ultimate goal is to characterize the behavior of hosts based on the traffic they generate. Consider first the top part of the figure, which depicts the pipeline the provider, in yellow, adopts to solve the AI-based traffic analysis problem. It consists of three main steps, from left to right:

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2023.3329442

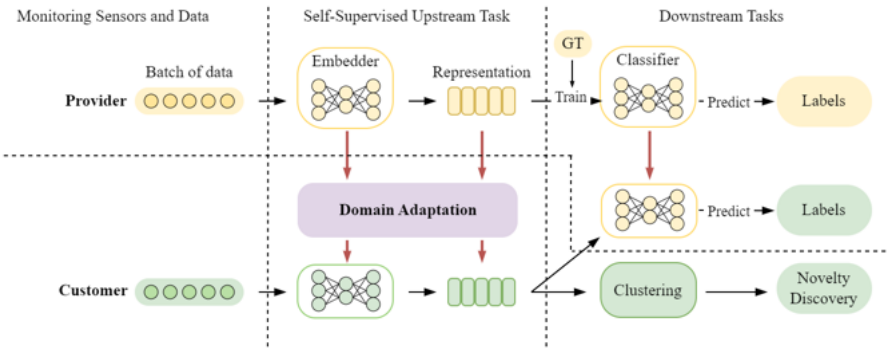


Fig. 1: Processing pipeline. The provider network (on the top) has data and some ground truth (GT) to execute the whole pipeline. The customer network has only data and needs to perform domain adaptation before facing the downstream tasks.

- A monitoring infrastructure allows the operator to collect raw data carried by a network. Such data consists of traces continuously collected over time, which can be at packet level [13], [14], [16], flow level [44] or any other sequences of events hosts generate over time. We do not consider specific solutions for this step, and we only assume data can be processed in batches every time interval ΔT, e.g., every day.
- The provider then builds a representative model of the hosts sending traffic by building an intermediate compact representation from the traffic each host sends. This is achieved by creating an embedding function that projects hosts with similar behavior into the same region(s). Here, we assume the host embeddings are updated every ΔT. The definition of the intermediate embeddings is obtained by the so-called *Self-Supervised Upstream Task* and requires no domain knowledge or labels.
- The provider uses the host embeddings for the final downstream task which can be a supervised or an unsupervised task (e.g., clustering). Supervised tasks require the availability of ground-truth data (GT), e.g., to train a classifier.

Our main question is whether and how the knowledge extracted from the provider network (e.g., the classifier) can be used in a different network. The block *Domain Adaptation* in Figure 1 is responsible for this step.

We consider a customer network (in green) that collects data but has no labels to perform downstream tasks – i.e., it has data and can build the intermediate embeddings. It then uses the provider classifier to label hosts. A second option for the customer is to use the provider embeddings to bootstrap the construction of local embeddings. In this case, the customer can use the resulting embeddings to cluster hosts and face an unsupervised downstream task.

B. Learning host embeddings from traffic

We build the intermediate representation through our previous works DarkVec and i-DarkVec, which rely on Word2Vec to produce the host embeddings. In [16], [17] we showed that DarkVec and i-DarkVec outperform other embedding creation mechanisms specifically designed for network traffic analysis such as IP2Vec [13] and DANTE [14].

In a nutshell, DarkVec (and i-DarkVec, which introduces the incremental training) takes the packets as they arrive and groups them based on the destination TCP or UDP ports, maintaining the sequence of hosts contacting such ports. By making an analogy with document processing, hosts are "words" that appear in a "document", i.e., a packet trace. By exploiting the relative position of the packets sent by hosts, DarkVec uses Word2Vec [9], [20] to learn from a trace (document) a vector representation for each host (word) in a self-supervised fashion. It uses a neural network model that is trained to predict which host (word) will appear in the context of a trace (document). In our case, given the i-th host in a sequence of packets, we train the neural network to predict the probability of occurrence of the $(i - c_w)$ and the $(i + c_w)$ hosts, being c_w the context window.

Formally, given a sequence $\{s_1, s_2, s_3, \ldots\}$ of host observations, we map each entity $s_j \in W \rightarrow u_j \in \mathbb{R}^e$ where W is the set of hosts and u_j is the embedding of s_j in the e dimension space. The function $g : W \rightarrow \mathbb{R}^e$ is the embedding function, which is a neural network. Given g, let $\mathbf{Y} = [g(w)]_{w \in W}$ be the matrix of embeddings for all hosts in W, i.e., $\mathbf{Y} \in \mathbb{R}^{|W| \times e}$.

Similarly to the natural language case, Word2Vec learns host embeddings that encode the co-occurrences of hosts (words), mapping those hosts performing similar (different) activities in the same (different) region of the embedding space. The intuition is that hosts performing similar or coordinated activities – i.e., contacting the same set of ports nearby in time – would co-occur in a similar context in the input sequences.

In [16], [17] we demonstrated how the intermediate representation extracted by DarkVec allows us to solve both supervised and unsupervised downstream tasks very efficiently, uncovering novel coordinated attacks in darknets, or extending the set of hosts being part of a known attack.

C. Downstream tasks

We use the host embeddings as input for supervised or unsupervised downstream machine-learning tasks.

In the supervised case, some ground truth allows us to train a classifier to extend the labels to previously unlabelled hosts or to detect changes or anomalies when a host modifies its behavior (and embeddings) over time. Let $l \in L$ be a

class label, being L the set of classes. We train a classifier function $z : \mathbb{R}^e \rightarrow L$ to predict the class label of a host given its embedding. Note that we use l only for this downstream classification task.

Now consider two networks, provider p and customer c. We can build the embeddings of the two networks $\mathbf{Y}_p = [g_p(w)]_{w \in W_p}$ and $\mathbf{Y}_c = [g_c(w)]_{w \in W_c}$. In the scenario of domain adaptation, the provider owns the labels corresponding to some hosts in W_p. Thus, it can build the supervised model z_p that maps the embeddings $\mathbf{Y}_p$ into labels $\mathbf{l}_p = z_p(\mathbf{Y}_p)$. The same classifier can be used to label host embeddings of the customer network using their aligned images.

In the unsupervised case, the identification of clusters of hosts that have similar embeddings (and thus behavior) allows the network administrators to gain insights and simplify the discovery of patterns. The labels of a few hosts can help in characterizing clusters, identifying new classes, or observing mutation in the host behavior. Intuitively, we directly leverage the embeddings to extract clusters of hosts. In fact, two points shall be close in the embeddings if they co-occur frequently in the same context. That is, if they send packets to the same ports, at the same time. Any distance-based clustering algorithms could thus identify hosts with similar patterns.

IV. Domain Adaptation: Definition and Approaches

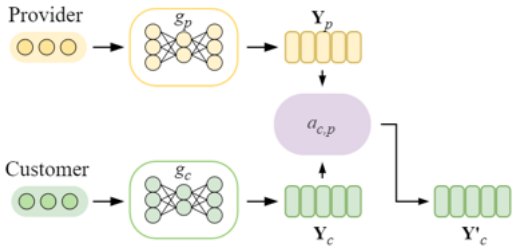
Given the representation and models obtained from data and labels at the disposal of a provider, we find the best approach to leverage this information for solving a downstream task using data at the disposal of a customer network.

The closest problem to ours is the domain adaptation of embeddings in natural language translation (see Section II-C). In its simplest form, a function transforms the embeddings of a source domain (provider) into embeddings compatible with the destination domain (customer). We rely on two different techniques: (i) an alignment function to align the embeddings of the customer network to those in the provider network; and (ii) a canonical transfer learning problem where the customer fine-tunes the provider embeddings. As shown in Figure 1, the customer then sends the aligned or fine-tuned embeddings to the provider, which uses them as input to the downstream task, returning results to the customer in a *models-as-a-service* approach. Each strategy implies a different collaboration model for sharing information among partners.

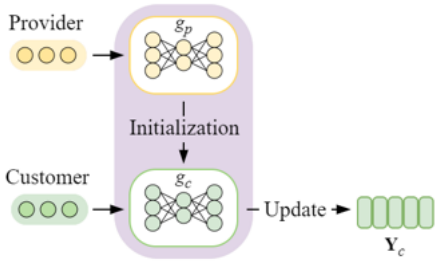
Without domain adaptation, $\mathbf{Y}_p$ and $\mathbf{Y}_c$ are not compatible, being elements sampled from the two embedding functions (g_p and g_c) trained on different data with different random initial weight assignment. We thus need strategies to transfer the knowledge from the provider to the customer network, i.e., execute the same downstream tasks on both host embeddings.

A. Anchors

We assume the existence of a common subset of senders active in both networks $W^* = W_p \cap W_c$, where $w \in W^*$ are called *anchors*. In fact, it is quite common to observe large scanners or botnets whose senders target the entire IP address space [45] and thus multiple darknets in the same observation period. Next, we introduce our strategies to adapt the knowledge using such anchors.



(a) Explicit alignment



(b) Canonical transfer

Fig. 2: Domain adaptation strategies.

B. Explicit alignment

We define an aligner $a_{c,p} : \mathbb{R}^e \rightarrow \mathbb{R}^e$ as a mapping function[2] from the embedding space in $\mathbb{R}^e$ of the customer network c to the embedding space in $\mathbb{R}^e$ of the provider network p. Let $\mathbf{Y}^*_p = [g_p(w)]_{w \in W^*}$ be the embeddings of the anchors in the provider network and $\mathbf{Y}^*_c = [g_c(w)]_{w \in W^*}$ the embeddings of the same anchors in the customer network.

An *explicit* aligner is a function $a_{c,p}$ that when applied to the embeddings of the customer network, maps them into embeddings that are compatible with those of the embedding space of the provider, i.e., $\mathbf{Y}^*_p \approx [a_{c,p}(\mathbf{y})]_{\mathbf{y} \in \mathbf{Y}^*_c}$. We consider both linear and non-linear functions to approximate the aligner function $a_{c,p}$ using as a metric to minimize the mean square error between anchors. Once the aligner $a_{c,p}$ is obtained, we apply it to all host embeddings $\mathbf{Y}_c$ of the customer, obtaining the new host embeddings $\mathbf{Y}'_c = [a_{c,p}(\mathbf{y})]_{\mathbf{y} \in \mathbf{Y}_c}$ that are aligned to the embeddings in provider's network.

Figure 2a summarizes the *explicit alignment* strategy (hereafter called *align* for short). In this scenario, the provider and customer share only the anchor embeddings to build the aligner function $a_{c,p}$. The aligned customer host embedding $\mathbf{Y}'_c$ can then be given as input to the downstream classification task, in a model-as-a-service approach.

Here, both p and c observe the activity of hosts during a time interval $t_0, \ldots, t_n$. Both networks autonomously build their embeddings applying the embedding functions g_p or g_c which is updated at every t_i.[3]

To demonstrate the approach, we build the aligner function through a DNN with 3 hidden layers with 1 024 neurons each.

[2]Embeddings have the same dimension e. Generalization is however trivial.
[3]To simplify notation, we refer to the last embedding function at time t_n.

Each layer is activated through either a linear function (linear aligner)[4] or a non-linear Sigmoid function (non-linear aligner). The input and output layers contain e neurons each and we add a normalization layer [46] between the input and the first hidden layer. Given any customer anchor $y_c \in \mathbf{Y_c^*}$ and the corresponding provider anchor $y_p \in \mathbf{Y_p^*}$, we train the DNN to minimize the mean squared error between $a_{c,p}(y_c)$ and y_p. We use the standard Adam optimizer [47] training the DNN for 50 epochs and a batch size of 32 samples.

C. Canonical transfer

A second option to perform the domain adaptation consists of transferring the provider embedding and letting the customer update and fine-tune it as shown in Figure 2b.

In this case, the customer incrementally builds the host embeddings starting from the embeddings of the provider – i.e., using $\mathbf{Y}_p$ as initial weights to start the building of g_c. To this end, network p sends its trained embedding function g_p and the host embeddings $\mathbf{Y}_p$ to network c. Notice that anchors also play a key role in this step, as their host embeddings are already initialized in g_p. The anchors thus provide context for Word2Vec, which implicitly creates g_c embeddings that are likely to be aligned with network p too. With this step, the embedding function g_c of the customer network extends g_p – i.e., g_c generates an *updated* embedding space that will also reflect the data observed by network c.

D. Towards a model-as-a-service approach

Our approaches open the way for new scenarios in which parties do not exchange raw data, limiting both the amount of data to exchange and the privacy implications. In both cases, the amount of information scales with the number of hosts and not with the number of packets observed in the networks. Considering privacy, the only shared information is whether a given host is sending packets (i.e., it is possible to build its embedding). There is no disclosure of ports, protocols, payload, transfer rates, and packet timing. This approach opens new collaboration and business opportunities for network traffic analysis. Setting up such collaborations will require (i) new business agreements between operators, and (ii) systems to support the distribution of embeddings. We leave these aspects for future work.

V. Darknet Use case: Scenario

A. Darknet sensors and traffic

Darknets are networks whose contiguous IP addresses are announced on the Internet but without hosting any services. All traffic reaching them is thus unsolicited and hence suspicious. Darknets represent a valuable source of information for cybersecurity and threat intelligence [48], [49], allowing analysts to uncover large-scale Internet scans and attacks.

Darknets observe hundreds of thousands of sources, making the identification of events very difficult. A large share of packets arriving at darknets comes from sources that perform coordinated activities [4], [5], [16], e.g., botnets or distributed scanners. Multiple groups of such coordinated sources contact the darknet following similar patterns that word embedding techniques applied on sequences of observed hosts can successfully extract to ease the analyst's job [14], [16]. Here, we design a use case in which two network operators host one darknet each. The first acts as a provider and has labels. The second has no labels and thus would like to transfer the information from the provider network to gain insights about its network traffic.

[4]Note that a linear activation function will make such network equivalent to a vanilla network, linearly combining the e inputs.

B. Darknet datasets and ground truth

We rely on two datasets to explore domain adaptation in darknets. Each dataset has been captured in a /24 darknet for 43 days. We use each as the provider and customer networks, respectively, and they are deployed in different geographical locations and ASes. Both darknets observe millions of packets on a daily basis, coming from tens of thousands of remote hosts. Here we rely on the packet level trace collected from the 1st of Dec 2022 to the 11th of Jan 2023.

We follow the process as done in [16]: we consider batches of data collected every $\Delta T = 1$ day so that at the end of each day, the processing pipeline sketched in Figure 1 can be executed. For each day, we consider only hosts that sent at least 5 packets, i.e., the most active hosts.

Here we formulate a supervised learning use case. Our goal is to use the knowledge (ground truth labels) obtained in the provider network to classify the hosts contacting the customer darknet. We need such ground truth both to train models in the provider network as well as to assess the performance of the classifier when applied to customers' traffic. Naturally, the full power of transferring knowledge across networks emerges when uncovering labels that otherwise would not be available in the customer network – a scenario that we will target in our second use case (see Section VII).

To build such ground truth, we use public information and manual analysis to extend as much as possible the set of hosts for which we can associate a label. We start from the labels shared by authors of [50]. These lists include IP addresses of known Internet scanners run by security companies, research projects, etc. We next use the per-packet fingerprint of well-known botnets [51] to identify some hosts and label them as *zombies*. These are typically bots that continuously perform large-scale scans of the IP address space in search of possible victims.

We use the 31 days of Dec 2022 to build the host embeddings [16]. Then we use each day of Jan 2023 to train and test the downstream classifier performance. Specifically, we train it on the provider data and test it on the customer data with different domain adaptation options. For comparison, we consider the (unrealistic) case where the customer also owns the GT data, and thus can train the classifier independently.

We focus our manual effort on building the GT on hosts active on Jan 1st, 2023. For completeness, Table I details the number of active hosts for each ground truth class. The last column reports the number of hosts in common, i.e., possible anchors. In total, we have 10 ground truth classes that sum

TABLE I: Overview of the datasets for the Darknet knowledge transfer use case. The table reports the number of active hosts observed on the January 1st, 2023 datasets.

		Provider	Customer	Intersection
1	**Zombie [52]**	9 127	8 991	7 108
2	**Shadowserver [53]**	289	289	289
3	**Cyber.casa [54]**	252	252	252
4	**Internetcensus [55]**	208	208	208
5	**Spammer**	25	106	25
6	**Onyphe [56]**	96	96	96
7	**Rapid7 [57]**	29	33	5
8	**Censys [58]**	24	25	24
9	**Shodan [59]**	30	24	23
10	**Securitytrails [60]**	18	18	18
–	*Unknown*	4 809	5 124	3 004
	Total	14 907	15 166	11 052

up about 10 thousand hosts in both darknets. Classes are severely unbalanced, with some cases (e.g., *Zombie*) including thousands of hosts and others (e.g., Shodan, a security search engine) accounting for a few dozen hosts. We refer readers to [50] and to the references in the table for more information about the classes. We use the *unknown* class for those hosts for which we were not able to assign a class. Here we assume that any unknown host do not belong to any ground truth class (even if this case is possible).

C. Downstream task for the darknet use case

The authors of [16] use a simple k-nearest neighborhood classifier which is ill-suited for transfer learning since it requires the samples to be labeled, i.e., the customer shall own (or get) the GT. Here instead we consider a DNN classifier that we train using the provider data and labels. We consider a simple feed-forward neural network composed of a first layer that receives the host embeddings in $\mathbb{R}^e$ as input, and two hidden layers composed of 512 and 256 neurons, respectively. At each hidden layer we apply a 30% of dropout and all the activation functions are ReLU. The output layer has $|C|$ neurons activated using a Softmax function. Then, we minimize the categorical cross-entropy function using the Adam optimizer [47]. We use batches of 128 samples for training and we select the model with minimum validation loss over 50 epochs. We perform a coarse grid search. The DNN is robust to choices in the proposed range.

We run the experiments on a Tesla V100-PCIE-16GB. The 1 epoch training and update of i-DarkVec requires 1.7s for each day on average. The average training time per fold of the classifier and the aligners is 24s and 60s respectively.

VI. Darknet use case: Results

We mimic the scenario in which the provider uses the data of December 2022 to build reliable host embeddings, then it trains the supervised downstream classifier on Jan 1st, 2023. Then, it runs its model on the customer host embeddings after different domain adaptation options. We focus on the analysis of this downstream task using the same day (Jan 1st, 2023) as a test set in the customer network.

To avoid overfitting in the domain adaptation phase, we split both the provider and customer hosts into 5 folds (stratified cross-validation [61]). We take care of having consistent folds so that hosts that are present in both the provider and customer data belong to the same fold. For each of the five folds, we train the aligner and classifier on the 4 folds and validate on the 5th fold of the provider data. Then we test its performance on the 5th fold of customer data, after the domain adaptation step. Finally, we average the performance on the 5 test folds.

TABLE II: Supervised task macro-average performance on darknet use case – 100% of the anchors. The provider uses 32 days; the customer only Jan. 1st. The best results are in bold.

	Collaborative				Independent
	No Adaptation	Transfer	Linear Align	Non-linear Align	Reference
Precision	0.00	**0.94**	0.27	0.72	*0.78*
Recall	0.00	**0.92**	0.23	0.63	*0.86*
F1-Score	0.00	**0.92**	0.19	0.66	*0.81*

The test sets are always the same in the following cases, hence the results are comparable. We compute the classic performance metrics for supervised learning, including per-class precision, recall, and F1-Score. The results are reported as macro-averages. Considering the unbalanced classes, the macro average is a suitable conservative metric.

A. Constrained case: The customer has only one day of traffic

We consider the scenario in which the provider executes the pipeline on a daily basis updating its host embeddings and training a new classifier using the GT at its disposal. The customer asks the provider to analyze the data it collected for one day only (Jan 1st). We compare the following cases:

- *Independent (reference)*: The customer owns the GT and works independently – it builds host embeddings, trains the classifier, and tests it on the same day.
- *Collaborative*: The provider creates its embedding on 32 days (Dec and Jan 1st) then it trains the classifier on the last day (Jan 1st). In all cases, the provider classifies the customer host embeddings using the provider classifier.
  - *No adaptation*: customer creates embeddings using 1 day of data (Jan 1st) and sends them to the provider.
  - *Canonical transfer*: provider sends the host embeddings to the customer. The customer fine-tunes the embeddings using the last day of data (Jan 1st) and returns it to the provider.
  - *Linear and non-linear explicit alignment*: customer creates host embeddings using 1 day of data (Jan 1st) and sends them to the provider for alignment.

Table II summarizes results, with per-class details in Table VII in the Appendix. A few considerations hold: (i) The usage of the classifier model without domain adaptation is unfeasible, suggesting that the provider and customer converged to incompatible embedding spaces; (ii) Linear alignment does not suffice and the complexity of the embedding space calls for non-linear functions (here obtained using Sigmoid activation function in the DNN aligner); (iii) Canonical transfer guarantees excellent performance (F1-Score = 0.92), outperforming the independent reference.

TABLE III: Supervised task macro-average performance on darknet use case – 100% of the anchors. Provider and customer use 32 days (except for *Transfer*). The best results are in bold.

	Collaborative				Independent
	No Adaptation	Transfer*	Linear Align	Non-linear Align	Reference
Precision	0.03	0.94	0.75	**0.96**	*0.97*
Recall	0.01	**0.92**	0.68	0.90	*0.95*
F1-Score	0.00	**0.92**	0.68	**0.92**	*0.95*

*This case is the same of Table II.

This last result shows the benefit of doing a fine-tuning step on an already good representation captured by the provider host embeddings. In a nutshell, the knowledge captured from the 32-day-long provider data is successfully transferred to the customer embeddings. The scarce 1-day long customer data suffices to fine-tune such a model but fails to provide enough information to execute the whole pipeline (independent reference – F1-Score limited to 0.81).

B. Unconstrained case: The customer has more days of traffic

In this case, the customer has been running its darknet for a long time, 32 days in our case. Hence it can create embeddings in these 32 days. As in the case of 1-day embeddings, we compare the following alternatives:

- *Independent (reference)*: The customer owns the GT and works independently. It builds host embeddings on 32 days, trains the classifier, and tests it on the last day.
- Collaborative: provider creates its embedding on 32 days, then it trains the classifier on the last day (Jan 1st). In all cases, the provider classifies the customer host embeddings using the provider classifier.
  - *No adaptation*: customer creates host embeddings using 32 days of data and sends them to the provider.
  - *Canonical transfer*: same as before.
  - *Linear and non-linear explicit alignment*: customer creates host embeddings using 32 days of data and sends them to the provider for alignment.

Table III summarizes results. Again, domain adaptation is strongly needed, and simple linear alignment functions do not suffice. Comparing Table II and Table III, we observe that the availability of 32-days of data at the customer side improves the performance of the non-linear alignment case, which now reaches the performance of the transfer case (0.92 F1-Score for both).[5] This benefit is again due to the ability to extract more expressive host embeddings from the 32-day data than from 1-day only. The much-improved performance of the independent reference testifies to this intuition.

C. Impact of the number of anchors

For our domain adaptation strategies, the number of anchors $|W^*|$ plays a crucial role (i) to provide the necessary context to *fine-tune* the provider host embeddings with data from the customer network in *canonical transfer* strategies; (ii) to explicitly train the aligner in the *explicit alignment* strategies. Without such a few anchors, none of our methodologies would work. In practice, different darknets have variable numbers of hosts in common, potentially impacting the domain adaptation performance. To assess such impact, we perform experiments reducing the number of anchors.

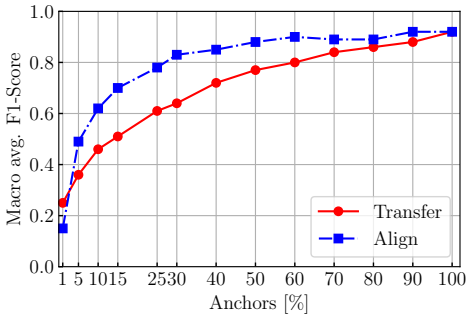


Fig. 3: Impact of the number of anchors in common; the full set corresponds to ≈ 73% of the anchors.

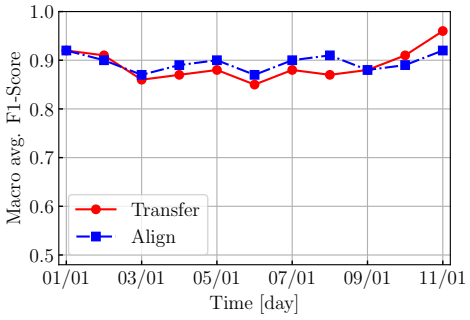


Fig. 4: Classification performance when repeating the whole processing pipeline over several days (using all anchors).

Here, we consider the same scenario as in Section VI-B. Both the provider and customer can leverage 32 days of data to build the host embeddings. We train the classifier on the provider side and test it on the Jan. 1st customer data as usual. Given the set of all possible anchors W^*, we artificially consider only a percentage of them to align the embeddings. We report results in Figure 3. We select such a percentage by performing a stratified sampling to obtain the set $\hat{W}^*$. We then either train the non-linear aligner function $a_{c,p}$ considering only embeddings of anchors in $\hat{W}^*$; or transfer only the portion of the host embeddings $\hat{\mathbf{Y}}_p^* = [g_p(w)]_{w \in \hat{W}^*}$ to perform the fine-tuning of g_p at the customer side.[6]

After this step, we repeat the 5-fold cross-validation experiments already described. We repeat the stratified sampling of anchors five times to increase our confidence in the results. We then present the average among the 5 experiments and 5 folds.[7]

The *transfer* strategy (red line in the figure) increases its performance when we increase the number of anchors. It

[5]Here, we consider that the customer runs a fine-tuning step using only the current day of data. We leave for future work the optimization of the number of past days of data, for canonical transfer learning.

[6]We randomly initialize embeddings for anchors discarded in the transfer.

[7]Except for the right-most point in which the full set of anchors is used.

achieves the maximum 0.92 average F1-Score only when we use 100% of available anchors (11 052 in our case, about 73% of the total number of active hosts – see Table I).

With a low number of anchors, the fine-tuning of g_p at the customer network is less effective, as Word2Vec has few known host embeddings to guide the mapping of the new customer host into the g_p embedding space. In other words, without anchors, this scenario converges to the *no alignment* case of Table III – host embeddings seen only on the customer network are not properly mapped into the g_p embedding space.

The *explicit align* strategy (blue line) is slightly less sensitive to the number of anchors. Here, the aligner function $a_{c,p}$ suffers less when reducing the number of samples since the DNN training is less affected by the amount of data. 30% of the available anchors (3 300, corresponding to about 22% of total samples) already suffice to reach 0.8 or higher F1-Score.

D. Robustness over time

Here, we repeat the experiments on different days, mimicking a live system that receives and processes a new batch of data each day. For each new day, we update the embeddings (provider and customer), train a new classifier (provider), perform the domain adaptation, and finally test the classifier (customer).

In detail, we consider ten new days (from Jan 2nd, 2023 to Jan 11th, 2023). The provider keeps updating its embeddings for each new day (up to 42 days), and each time trains a new classifier on the new data, using its GT. Then, the customer receives the provider embedding and classifier. In the canonical transfer case, the customer fine-tunes the embeddings on its new data and then tests the provider classifier (5-fold, over the given day). For the explicit alignment case, the customer keeps updating their embeddings for each new day (up to 42 days), aligns the received embedding to its own, and then tests the received classifier (5-fold, over the given day). Results (Figure 4) show little changes over time, for both the transfer explicit align approaches.

Takeaway: we conclude that transferring knowledge from a provider to a customer network is possible with domain adaptation and anchors play a key role. In the case of alignment, non-linear aligners are required.

VII. Honeypot Use case: Scenario

We now consider a more challenging scenario for the application of domain adaptation for traffic analysis: we transfer knowledge from a provider running *honeypots* to a customer running *darknets*. Honeypots offer a richer view than darkness: they mimic the functioning of a real system and engage with the attackers producing rich application logs that support the security analyst in understanding the attacks. It is possible then to detect attacks that would be impossible to classify using darknet traffic alone.

Using the hosts in common between the darknet and honeypot datasets, we first compare our domain adaption options by facing a supervised downstream task (Section VII-C). We then present an unsupervised downstream task use case (Section VIII), in which the customer leverages the knowledge from the honeypot to shed light on the activity of hosts observed in its darknets.

A. Honeypot sensors and traffic

Honeypots are *active sensors* that collect information about unsolicited traffic. They engage with attackers by running software that replicates the basic functions of real systems. A honeypot mimics a vulnerable system to allow attackers to progress in their attacks while saving as much information as possible. Honeypots are thus complementary to darknets. While a darknet provides a broad but shallow view of scanning activity only, honeypots allow analysts to obtain deeper insights into specific attack patterns. On the downside, honeypots are harder to deploy and maintain. They represent a risk to the network hosting them, as they attract large volumes of malicious traffic [62], [63]. Having a few honeypot providers that share and transfer their knowledge represents an asset to the whole Internet ecosystem.

B. Honeypot dataset and extended ground truth

We rely on a dataset obtained from a honeypot infrastructure deployed in a /24 network of our university network. The infrastructure relies on the TPOT honeypots [64], which offers a collection of third-party *low-interaction honeypots*, i.e., simple scripts built to simulate a vulnerable service communicating over a given protocol.

Most of the honeypots we deploy simulate services such as SSH, RDP, POP3, IMAP, and MySQL. We include Cowrie [65], a sophisticated honeypot that simulates a vulnerable server accessible via SSH/Telnet. It allows attackers to go further in their attempts, from discovery, SSH channel negotiation, brute-force login attempts, and initial shell access, up to the download and execution of malware binaries and scripts. This (and other honeypot logs) allow us to observe the attackers' intentions and accordingly define new GT classes, far beyond the simple scanners seen in darknets. In particular, we define two new classes: (i) *Brute-forcers* – hosts performing more than 10 login attempts, regardless of the target service; (ii) *Exploiters* – hosts that login and download files in any honeypots. Note that they would be just classified as unknown scanners in darknet sensors.

In total, the new GT contains the 11 classes already known, plus the Brute-forcer and Exploiter classes. Notice that only a small amount (about 20 and 30 senders, respectively) contact the darknet too. This is expected since previous works showed that different hosts are engaged in the different phases of attacks [51], [66]. We use these two additional classes to demonstrate how knowledge from the honeypots can be transferred to tag hosts observed in the darknets – hosts that otherwise would remain classified as a specific scanner at best.

C. Supervised downstream task: from honeypots to darknets

As a preliminary assessment, we validate that the domain adaptation techniques work in our cross-domain scenario with a supervised downstream task. The rationale of this experiment is to verify whether the embeddings learned in the provider

TABLE IV: Supervised task on the honeypot use case – 100% of the anchors. We report only the case in which both provider and customer have 32 days of traffic to learn embeddings (except the customer in the *Transfer* column), as in Table III.

	Collaborative				Independent
	No Adaptation	Transfer	Linear Align	Non-linear Align	Reference
Precision	0.02	**0.88**	0.64	0.83	*0.81*
Recall	0.05	**0.88**	0.62	0.81	*0.84*
F1-Score	0.01	**0.86**	0.60	0.80	*0.81*

honeypots can be effectively transferred to the customer darknet. For that, we repeat the validation pipeline used in the collaborative experiments with darknets in Section VI-B, however using the honeypot as the provider network.

We collect the honeypot traffic for 42 days in the same period we collect the darknet traces (see previous sections). As input, we consider the sequence of TCP-SYN packets sent by hosts contacting any honeypots. In total, we observe more than 18 000 active hosts for which we can build an embedding.

The provider creates its embedding on 32 days of honeypot traffic and trains the classifier on the last day (Jan. 1st) with the extended GT. Then, the provider and the customer perform the domain adaptation (either with canonical transfer and Jan. 1st data or with explicit alignment) between the customer and provider host embeddings. Finally, the provider classifies the adapted customer embeddings and we extract the performance as previously (5-fold average over Jan. 1st).

According to the larger amount of packets collected by the honeypot with respect to the darknet case, the training and update of i-DarkVec requires on average 23s for each day. The average training time per fold of the classifier and the aligners is 24s and 69s, respectively.

Table IV reports results. Trends similar to those observed in previous experiments emerge, even if numbers cannot be directly compared since we now consider a different GT. In the independent customer scenario, average precision, recall, and F1-score are lower at 0.81–0.84 in this case. Recall that this is our reference experiment, which assumes that the customer has the extended GT labels (thus including the two new classes). However, the host embeddings built independently using only darknet traffic cannot correctly separate the exploiters and brute-forcers. For these two classes, the F1-Score drops to 0.29 and 0.04, respectively.

Considering the collaborative cases, domain adaptation is necessary. Interestingly, the canonical transfer allows the customer to achieve better performance than in our hypothetical independent reference – e.g., precision increases to 0.88. This is due to the ability to transfer the knowledge about the two new classes of brute-forcers and exploiters: the host embedding trained on the provider honeypots enables the transfer of the new knowledge to the customer darknet so that the classifier can correctly discover hosts of these new classes.

Takeaway: Adapting embeddings from the honeypot to the darknet allows to transfer labels only present at the provider. Canonical transfer learning performs slightly better than non-linear alignment thanks to the ability to transfer the knowledge about new classes present in providers' data.

TABLE V: Clusters obtained when applying HDBSCAN on the customer embeddings (i.e., reference) and on embeddings adapted through the proposed approaches.

		Transfer	Align	Reference
Clusters		60	154	76
Noise		51%	45%	61%
Homogeneity†		0.53	0.89	0.82
Silhouette‡	μ	-0.37	0.18	-0.01
	σ	0.51	0.6	0.54

† Without 'unknown' samples; ‡ Without noisy samples

TABLE VI: Knowledge gained from the transfer.

	Reference		Transfer		Align	
	Hosts	Clusters	Hosts	Clusters	Hosts	Clusters
GT_{11}	3 662	24	–	–	–	–
GT_{13}	–	–	5 445	16	4 692	84
Extended GT_{11}	461	6	–	–	–	–
Extended GT_{13}	–	–	369	7	498	11
New GT	532	9	327	10	606	13
Suspicious	112	10	35	5	803	16
Total	4 767	49	6 176	38	**6 599**	**124**
Total [%]	31.43	64.47	40.72	63.33	**43.51**	**80.52**
Unknown	10 399	27	8 990	22	8 567	30

VIII. Honeypots use case: Results

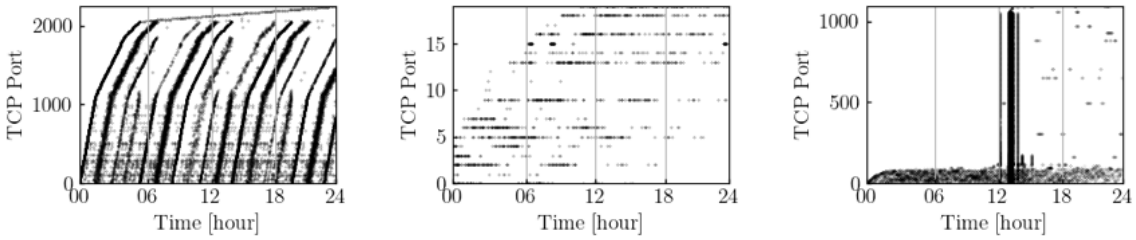
A. Methodology

We take as provider the same honeypot infrastructure used in Section VII-B and as the customer, the darknet. The provider learns embeddings using the complete 32-day dataset, and the customer uses them to either transfer or align its host embeddings. It then performs clustering on such host embeddings. As a baseline, we consider the case in which the customer performs clustering directly with its independently build host embeddings. This allows us to assess the knowledge gains when using domain adaptation.

We test multiple algorithms, such as k-means, DBSCAN [67], HDBSCAN [68], and the Louvain-based approach used in [16], noticing minor differences. We present results using the HDBSCAN algorithm because it is an almost parameter-free algorithm and has proven to deliver good qualitative results. We first characterize the obtained clustering using generic quality metrics (e.g., the number of clusters, the silhouette, and the homogeneity). Here we do not consider the *noise cluster* – i.e., points for which HDBSCAN assigns no cluster.

Next, we manually evaluate some clusters trying to *explain* the activity of hosts belonging to them. For this explanation, we take into account (i) the hosts for which the darknet could autonomously determine a ground truth as before in Section V – here called GT_{11},; (ii) the additional ground truth labels obtained from the provider honeypots as in Section VII – here called GT_{13}; (iii) external on-line sources of Cyber Threat Intelligence (CTI) [59], [69], [70]. We consider a cluster to be *explained* when at least 50% of the hosts in the cluster have the same label. We thus ignore cases of clusters where there is a mix of hosts from multiple classes.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2023.3329442



(a) Cluster of *cyber.casa* hosts only. (b) Cluster of likely web crawlers only. (c) Cluster of horizontal scanners.

Fig. 5: Examples of activity patterns of hosts in new clusters. Activity period: January 1st 2023.

B. Knowledge gain from domain adaptation

Table V provides an overview of the clusters obtained in the reference baseline scenario, as well as when adapting the embeddings with canonical transfer and non-linear alignment. The number of clusters increases substantially from the reference (76) to the alignment case (154). Observe also how the number of hosts remaining in the *noisy* group is reduced from 61% in the reference scenario, to 51% with the canonical transfer, and further to 45% with the alignment. The homogeneity and silhouette metrics also suggest somehow better clustering in the alignment case.

To detail the analysis we provide qualitative results in Table VI. The table compares the number of hosts that the customer could explain (provide a label) (i) for clusters built with its own embeddings (*reference* columns), (ii) after adapting the embeddings from the provider through canonical transfer (*transfer* columns) and (iii) after aligning the embeddings from the provider (*align* columns).

First, focus on the GT_{11} and Extended GT_{11}. Using only the GT information readily available to a darknet, the customer could explain the activity of around 3 600 hosts without the help of the provider (GT_{11}). By applying clustering algorithms on the autonomously derived host embeddings it could explain another 6 clusters, which include 461 hosts (Extended GT_{11}). Manual checks with reverse DNS lookups, ownership of IP addresses, AS numbers and CTI sources confirm the goodness of clusters. These groups can be uncovered only thanks to the clustering that aggregates unknown hosts together with some hosts present in the ground truth. As such, this experiment simply reinforces the attractiveness of the NLP-based approach in the networking scenario.

GT_{13} and Extended GT_{13} lines instead support the knowledge transfer approaches more directly. These lines report the number of hosts that the darknet can explain thanks to the labels that only the honeypot allows the provider to get, i.e., the brute-forcers and exploiters. Overall, the customer can explain more than 5 000 hosts in total with both the adaptation approaches and can achieve a finer-grained clustering (95 clusters compared to the 30 of the reference) with the embeddings alignment. These hosts represent a knowledge gain thanks to the adaptation approach.

Our manual inspection of clusters also let us uncover new classes of scanners (*New GT* in the table) and multiple clusters containing hosts that are consistently reported in conjunction by multiple CTI sources (*Suspicious* in the table) – but for which we have no confirmation of their class. Here again, the customer could check these sources autonomously and explain the activity of some hosts without any help from the provider. This leads to 532 (new GT) and 112 (suspicious) hosts using the embeddings in the reference case. Yet, the alignment of embeddings from the provider allows the customer to expand the number of explained clusters, increasing the number of explained hosts to 606 (new GT) and 803 (suspicious).

Takeaway: The transfer of knowledge from the provider increases the hosts for which the customer can explain the unsolicited traffic by about 30% (from 4 767 to 6 176 hosts) with canonical transfer and by about 38% (from 4 767 to 6 599 hosts) with domain adaptation. While the number of unknown hosts remains high (e.g., 8 567 with alignment), we argue that the adaptation approaches allow transferring knowledge across multiple networks, easing the analysts' tasks.

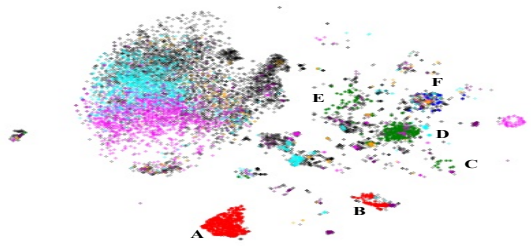
C. Qualitative analysis of discovered clusters

We investigate some clusters that emerge thanks to the adaptation of embeddings from the provider. Here we do not aim to be exhaustive but only exemplify the usefulness of the information uncovered with the approach. Figure 5 illustrates the activity of hosts in 3 clusters. The x-axes show the time, and each point in the figures marks the moment in which a new flow is observed from a host in the cluster. Figures 5a to 5c mark in the y-axis (the rank of) contacted TCP ports.

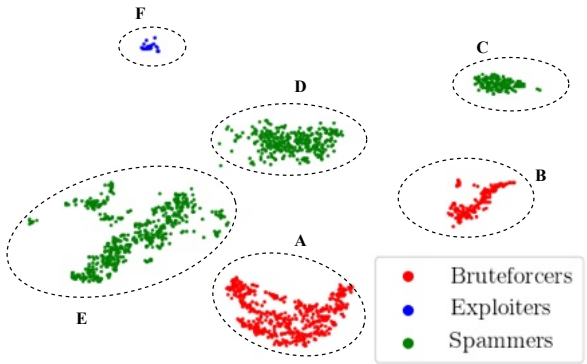
We observe clear temporal patterns, which strongly support the aggregation of the respective hosts. Manual inspection confirms that flows depicted in Figure 5a come from 252 hosts from the *cyber.casa* scanner, which targets 2249 TCP ports. These hosts synchronously rotate in some predetermined ports over time. Figure 5b instead shows the activity of 22 hosts that mostly target a few ports associated with web hosting. CTI suggests them to be crawlers. Finally, Figure 5c shows the activity of a cluster composed of 27 hosts that very aggressively contacted a range of ports of the darknet in a short time interval – here CTI information points to bots that often are reported for sending spam.

Finally, Figure 6 illustrates how points marked as noise when clustering the *reference* customer embeddings are reassigned to valid clusters after aligning the embeddings. To visualize the high-dimensional embeddings we rely on t-SNE [71], which non-linearly projects such data in a

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2023.3329442



(a) Noisy points in customer independently built host embeddings, which are assigned to some cluster in the adapted host embeddings. Samples are colored according to the clusters after alignment. Note multiple clusters may share some colors.



(b) Some new clusters discovered from the aligned host embeddings thanks to the transfer of honeypot knowledge.

Fig. 6: t-SNE of some clusters explained after the alignment.

bi-dimensional space while preserving the local and global structure of the original embedding. In Figure 6a we show a t-SNE plot of the noisy points of the clustering with the customer embeddings. As is usually the case when exploring the noisy set of HDBSCAN, we observe a cloud of points without clear structures. After repeating the clustering with aligned embeddings almost 4 000 hosts out of the total 9 000 in the noisy group are assigned to valid clusters. We mark these points with colors in Figure 6a.

In Figure 6b we illustrate examples of clusters emerging from the noisy points. Here a new t-SNE plot is built using only hosts belonging to some of the new clusters of aligned embeddings. These examples can be fully explained with these clusters. In red we show clusters identified as brute-forcers (764 hosts), in green spammers (≈1 000 hosts), and in blue exploiters (22 hosts). Letters identify the clusters and are also reported in Figure 6a.

Takeaway: The alignment allows us to characterize harmful hosts that would otherwise remain uncovered in the noise.

IX. Conclusion

This paper studied the problem of transferring knowledge from a provider network to other potentially label-scarce networks. We started from a knowledge extraction pipeline that builds intermediate representations (host embeddings) from raw packet traces and then uses them in supervised and unsupervised downstream tasks.

The knowledge transfer from a provider to a customer calls for domain adaptation, such that models learned on the former work on the latter. We formulated such a problem and proposed both canonical transfer and explicit alignment solutions. We tested and compared both approaches showing that, without the need to transfer labels, they perform on par or better than the independent reference. We observed that (i) adaptation requires non-linear alignment functions (unlike in classic NLP settings), (ii) adaptation works even with few anchors, i.e. when networks observe only a few hosts in common. Then, we successfully showcased the transfer of knowledge between two heterogeneous domains, namely, from a honeypot to a darknet. Here, applying the domain adaptation techniques on the intermediate host embeddings allowed us to extend the understanding of darknet traffic by identifying classes of hosts that could only be detected thanks to richer honeypot information.

With the increasing adoption of word embeddings and language models to learn features from sequences of network entities, adaptation between networks will only become more important. Our work paves the road towards efficiently sharing learned information between different networks.

Acknowledgement

The research leading to these results has been partly funded by the Huawei R&D Center (France), by the project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU and the PRIN 2022 Projects xInternet (eXplainable Internet - 20225CETN9) and ACRE (AI-Based Causality and Reasoning for Deceptive Assets - 2022EP2L7H).

References

[1] S. Rezaei and X. Liu, "Deep learning for encrypted traffic classification: An overview," *IEEE communications magazine*, 2019.

[2] F. Soro, T. Favale, D. Giordano, L. Vassio, Z. Ben Houidi, and I. Drago, "The New Abnormal: Network Anomalies in the AI Era," *Communication Networks and Service Management in the Era of Artificial Intelligence and Machine Learning*, 2021.

[3] S. A. Salloum, M. Alshurideh, A. Elnagar, and K. Shaalan, "Machine learning and deep learning techniques for cybersecurity: a review," in *The International Conference on Artificial Intelligence and Computer Vision*, 2020.

[4] R. Prajapati, V. Honavar, D. Wu, J. Yen, and M. Kallitsis, "Shedding Light into the Darknet: Scanning Characterization and Detection of Temporal Changes," in *Proc. of the 17th CoNEXT*, 2021.

[5] M. Kallitsis, V. G. Honavar, R. Prajapati, D. Wu, and J. Yen, "Zooming Into the Darknet: Characterizing Internet Background Radiation and its Structural Changes," *arXiv:2108.00079*, 2021.

[6] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *27th International Conference on Artificial Neural Networks*, 2018.

[7] G. Wilson and D. J. Cook, "A Survey of Unsupervised Deep Domain Adaptation," *ACM Transactions on Intelligent Systems and Technology*, 2020.

[8] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, 2020.

[9] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," 2013.

[10] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv:1810.04805*, 2018.

[11] "ChatGPT: Optimizing Language Models for Dialogue," 2022, https://openai.com/blog/chatgpt.

[12] Y. Tian, Y. Wang, D. Krishnan, J. B. Tenenbaum, and P. Isola, "Rethinking few-shot image classification: a good embedding is all you need?" in *European Conference on Computer Vision*, 2020.

[13] M. Ring, A. Dallmann, D. Landes, and A. Hotho, "IP2vec: Learning similarities between IP addresses," in *2017 IEEE ICDMW*, 2017.

[14] D. Cohen, Y. Mirsky, M. Kamp, T. Martin, Y. Elovici, R. Puzis, and A. Shabtai, "DANTE: A framework for mining and monitoring darknet traffic," in *ESORICS 2020*, 2020.

[15] Z. B. Houidi, R. Azorin, M. Gallo, A. Finamore, and D. Rossi, "Towards a systematic multi-modal representation learning for network data," in *Proc. of the 21st ACM HOTNETS*, 2022.

[16] L. Gioacchini, L. Vassio, M. Mellia, I. Drago, Z. B. Houidi, and D. Rossi, "DarkVec: automatic analysis of darknet traffic with word embeddings," in *Proc. of the 17th CoNEXT*, 2021.

[17] L. Gioacchini, L. Vassio, M. Mellia, I. Drago, and Z. B. Houidi, "i-DarkVec: Incremental Embeddings for Darknet Traffic Analysis," *ACM Transactions on Internet Technology*, 2023.

[18] Z. B. Houidi and D. Rossi, "Neural language models for network configuration: Opportunities and reality check," *Computer Communications*, 2022.

[19] M. Boffa, G. Milan, L. Vassio, I. Drago, M. Mellia, and Z. B. Houidi, "Towards NLP-based Processing of Honeypot Logs," in *2022 IEEE EuroS&PW*, 2022.

[20] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013.

[21] J. K. Tripathy, S. C. Sethuraman, M. V. Cruz, A. Namburu, M. P., N. K. R., S. I. S, and V. Vijayakumar, "Comprehensive analysis of embeddings and pre-training in NLP," *Computer Science Review*, 2021.

[22] F. A. Acheampong, H. Nunoo-Mensah, and W. Chen, "Transformer Models for Text-Based Emotion Detection: A Review of BERT-Based Approaches," *Artificial Intelligence Review*, 2021.

[23] Z. Zhou, F. Wang, J. Yu, J. Ren, Z. Wang, and W. Gong, "Target-oriented Semi-supervised Domain Adaptation for WiFi-based HAR," in *IEEE INFOCOM*, 2022.

[24] X. Chen, H. Li, C. Zhou, X. Liu, D. Wu, and G. Dudek, "Fido: Ubiquitous fine-grained wifi-based localization for unlabelled users via domain adaptation," in *Proc. of The Web Conference*, 2020.

[25] H. Li, X. Chen, J. Wang, D. Wu, and X. Liu, "DAFI: WiFi-Based Device-Free Indoor Localization via Domain Adaptation," *Proc. of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2022.

[26] C. Liu, Z. Wei, D. W. K. Ng, J. Yuan, and Y.-C. Liang, "Deep transfer learning for signal detection in ambient backscatter communications," *IEEE Transactions on Wireless Communications*, 2020.

[27] S. Arshad, C. Feng, R. Yu, and Y. Liu, "Leveraging transfer learning in multiple human activity recognition using wifi signal," in *IEEE 20th WoWMoM*, 2019.

[28] G. Sun, L. Liang, T. Chen, F. Xiao, and F. Lang, "Network traffic classification based on transfer learning," *Computers & electrical engineering*, 2018.

[29] J. Guan, J. Cai, H. Bai, and I. You, "Deep transfer learning-based network traffic classification for scarce dataset in 5G IoT systems," *International Journal of Machine Learning and Cybernetics*, 2021.

[30] S. Zhang, Z. Zhong, D. Li, Q. Fan, Y. Sun, M. Zhu, Y. Zhang, D. Pei, J. Sun, Y. Liu *et al.*, "Efficient KPI Anomaly Detection Through Transfer Learning for Large-Scale Web Services," *IEEE Journal on Selected Areas in Communications*, 2022.

[31] P. Xiong, B. Cui, and Z. Cheng, "Anomaly network traffic detection based on deep transfer learning," in *International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, 2020.

[32] Z. Xu, D. Yang, J. Tang, Y. Tang, T. Yuan, Y. Wang, and G. Xue, "An Actor-Critic-Based Transfer Learning Framework for Experience-Driven Networking," *IEEE/ACM Transactions on Networking*, 2021.

[33] Q. Chen, Z. Zheng, C. Hu, D. Wang, and F. Liu, "Data-driven task allocation for multi-task transfer learning on the edge," in *IEEE 39th ICDCS*, 2019.

[34] H. Daga, P. K. Nicholson, A. Gavrilovska, and D. Lugones, "Cartel: A System for Collaborative Transfer Learning at the Edge," in *Proc. of the ACM Symposium on Cloud Computing*, 2019.

[35] Y. Yuan, L. Jiao, K. Zhu, X. Lin, and L. Zhang, "AI in 5G: The Case of Online Distributed Transfer Learning over Edge Networks," in *IEEE INFOCOM*, 2022.

[36] T. Mikolov, Q. V. Le, and I. Sutskever, "Exploiting similarities among languages for machine translation," *arXiv:1309.4168*, 2013.

[37] M. Artetxe, G. Labaka, and E. Agirre, "Generalizing and improving bilingual word embedding mappings with a multi-step framework of linear transformations," in *Proc. of the AAAI Conference on Artificial Intelligence*, 2018.

[38] A. Joulin, P. Bojanowski, T. Mikolov, H. Jégou, and E. Grave, "Loss in translation: Learning bilingual word mapping with a retrieval criterion," *arXiv:1804.07745*, 2018.

[39] P. Jawanpuria, A. Balgovind, A. Kunchukuttan, and B. Mishra, "Learning multilingual word embeddings in latent metric space: a geometric approach," *Transactions of the Association for Computational Linguistics*, 2019.

[40] G. Lample, A. Conneau, L. Denoyer, and M. Ranzato, "Unsupervised machine translation using monolingual corpora only," *arXiv:1711.00043*, 2017.

[41] M. Zhang, Y. Liu, H. Luan, and M. Sun, "Adversarial training for unsupervised bilingual lexicon induction," in *Proc. of the 55th Annual Meeting of the Association for Computational Linguistics*, 2017.

[42] D. Alvarez-Melis and T. S. Jaakkola, "Gromov-Wasserstein alignment of word embedding spaces," *arXiv:1809.00013*, 2018.

[43] P. Jawanpuria, M. Meghwanshi, and B. Mishra, "Geometry-aware domain adaptation for unsupervised alignment of word embeddings," *arXiv:2004.08243*, 2020.

[44] R. Gonzalez, C. Soriente, J. M. Carrascosa, A. Garcia-Duran, C. Iordanou, and M. Niepert, "User Profiling by Network Observers," in *Proc. of the 17th CoNEXT*, 2021.

[45] F. Soro, I. Drago, M. Trevisan, M. Mellia, J. Ceron, and J. J. Santanna, "Are Darknets All The Same? On Darknet Visibility for Security Monitoring," in *Proc. of the IEEE International Symposium on Local and Metropolitan Area Networks*, 2019.

[46] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv:1607.06450*, 2016.

[47] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv:1412.6980*, 2014.

[48] C. Fachkha, E. Bou-Harb, and M. Debbabi, "Inferring distributed reflection denial of service attacks from darknet," *Computer Communications*, 2015.

[49] M. Jonker, A. King, J. Krupp, C. Rossow, A. Sperotto, and A. Dainotti, "Millions of Targets Under Attack: A Macroscopic Characterization of the DoS Ecosystem," in *Proc. of the ACM IMC*, 2017.

[50] "Acknowledged Scanners," 2023, https://gitlab.com/mcollins_at_isi/acknowledged_scanners.

[51] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis *et al.*, "Understanding the mirai botnet," in *USENIX Security 17*, 2017.

[52] J. Fruhlinger, "The Mirai botnet explained: How teen scammers and CCTVcameras almost brought down the internet," 2018, https://www.csoonline.com/article/3258748/\\the-mirai-botnet-explained-\\how-teen-scammers-and-cctv-cameras-almost-brought-down-the\\-internet.html.

[53] "The Shadowserver Foundation," 2023, https://www.shadowserver.org/.

[54] "CyberCasa," 2023, https://www.cyber.casa/.

[55] "Internet Census Group," 2023, https://www.internet-census.org/home.html.

[56] "Onyphe," 2023, https://www.onyphe.io/.

[57] "Project Sonar," 2023, https://www.rapid7.com/research/project-sonar/.

[58] "Censys," 2023, https://censys.io/.

[59] "Shodan, the search engine," 2023, https://www.shodan.io/.

[60] "Securitytrails," 2023, https://securitytrails.com/.

[61] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[62] L. Metongnon and R. Sadre, "Beyond Telnet: Prevalence of IoT Protocols in Telescope and Honeypot Measurements," in *Proc. of the Workshop on Traffic Measurements for Cybersecurity*, 2018.

[63] J. Uitto, S. Rauti, S. Laurén, and V. Leppänen, "A survey on anti-honeypot and anti-introspection methods," 2017.

[64] "TPot – The All In One Honeypot Platform," 2023. [Online]. Available: https://github.com/telekom-security/tpotce

[65] "SSH/Telnet Honeypot," 2023. [Online]. Available: https://github.com/cowrie/cowrie

[66] R. Hiesgen, M. Nawrocki, A. King, A. Dainotti, T. C. Schmidt, and M. Wählisch, "Spoki: Unveiling a New Wave of Scanners through a Reactive Network Telescope," in *Usenix Security 22*, 2022.

[67] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *KDD*, 1996.

[68] L. McInnes, J. Healy, and S. Astels, "hdbscan: Hierarchical density based clustering." *The Journal of Open Source Software*, 2017.
[69] "GreyNoise," 2023, https://greynoise.io/.
[70] "AbuseIPDB - IP address abuse reports - Making the Internet safer, one IP at a time," 2023, https://www.abuseipdb.com/.
[71] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE." *Journal of machine learning research*, 2008.

APPENDIX

We now report the full per-class classification F1-Score complementing Sections VI and VIII.

TABLE VII: F1-Scores (darknet use case – see Table II).

	Collaborative				Independent
	No Adaptation	**Transfer**	**Linear Align**	**Non-linear Align**	**Reference**
Zombie	0.00	**0.99**	0.98	**0.99**	*0.98*
Shadowserver	0.00	**1.00**	0.04	0.74	*0.99*
Cyber.casa	0.00	**0.99**	0.32	0.95	*0.97*
Internetcensus	0.00	**1.00**	0.04	0.80	*0.83*
Spammer	0.02	**0.57**	0.07	0.26	*0.64*
Onyphe	0.00	**1.00**	0.15	0.50	*0.71*
Rapid7	0.00	**0.92**	0.03	0.12	*0.72*
Censys	0.00	**0.92**	0.20	0.63	*0.54*
Shodan	0.00	**0.86**	0.02	0.66	*0.78*
Securitytrails	0.00	0.96	0.08	**0.97**	*0.92*
Average	0.00	**0.92**	0.19	0.66	*0.81*

TABLE VIII: F1-Scores (darknet use case – see Table III).

	Collaborative				Independent
	No Adaptation	**Transfer***	**Linear Align**	**Non-linear Align**	**Reference**
Zombie	0.00	**0.99**	**0.99**	**0.99**	*0.99*
Shadowserver	0.00	**1.00**	0.99	**1.00**	*1.00*
Cyber.casa	0.00	**0.99**	0.97	**0.99**	*1.00*
Internetcensus	0.00	**1.00**	0.90	0.99	*0.99*
Spammer	0.02	**0.57**	0.27	0.46	*0.85*
Onyphe	0.00	**1.00**	0.75	**1.00**	*0.98*
Rapid7	0.00	**0.92**	0.55	**1.00**	*1.00*
Censys	0.00	**0.92**	0.69	0.90	*0.86*
Shodan	0.00	**0.86**	0.21	0.83	*0.81*
Securitytrails	0.00	0.96	0.49	**1.00**	*1.00*
Average	0.00	**0.92**	0.68	**0.92**	*0.95*

*This case is the same of Table VII.

TABLE IX: F1-Scores (honeypot use case – see Table IV).

	Collaborative				Independent
	No Adaptation	**Transfer**	**Linear Align**	**Non-linear Align**	**Reference**
Zombie	0.06	**0.99**	**0.99**	**0.99**	*0.99*
Shadowserver	0.00	**1.00**	**1.00**	**0.99**	*1.00*
Cyber.casa	0.00	**1.00**	0.97	**0.99**	*0.99*
Internetcensus	0.00	**0.99**	0.95	0.97	*0.98*
Spammer	0.01	**0.87**	0.64	0.86	*0.70*
Onyphe	0.00	**0.97**	0.89	**0.97**	*0.98*
Rapid7	0.00	**1.00**	0.16	**1.00**	*1.00*
Shodan	0.00	**0.89**	0.26	0.66	*0.80*
Censys	0.08	**0.89**	0.59	0.76	*0.89*
Securitytrails	0.00	**1.00**	0.52	**1.00**	*1.00*
Brute-forcer	0.01	**0.32**	0.28	**0.30**	*0.04*
Exploiter	0.00	**0.47**	0.00	0.13	*0.29*
Average	0.01	**0.86**	0.60	0.80	*0.81*

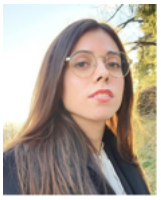
BIOGRAPHIES

Luca Gioacchini is a Ph.D. candidate at PoliTO. He received the B.Sc. in Electronic Engineering at Università Politecnica delle Marche in 2017 and the M.Sc. in ICT for Smart Societies at PoliTO in 2021. His research interests are in the field of machine learning and data science techniques applied to networking and cybersecurity. Luca is focusing on applying deep learning and unsupervised graph mining techniques to darknet traffic.

Marco Mellia (F'21) is a full professor at PoliTO, Italy. He coordinates the SmartData@PoliTO centre, an interdisciplinary lab focusing on Machine Learning, Data Science and applications to network management and cybersecurity. He has co-authored over 250 papers published in international journals and leading conferences. He won the IRTF ANR Prize at IETF-88, and many best paper awards. He is the EiC of the Proceedings of ACM on Networking.

Luca Vassio is an Assistant Professor at PoliTO, Italy. He received 'cum laude' a Ph.D. in telecommunication engineering and an M.Sc. in mathematical modeling. His research interests span from big data analytics to machine learning and optimization approaches, including GNNs. He applies them to internet measurements, social networks, and mobility. He collaborated, among others, with MIT, Bell Labs, and GE Aviation.

Idilio Drago is an Associate Professor at the University of Turin, Italy. His research interests include network security, machine learning, and Internet measurements. He is particularly interested in how machine learning can help extract knowledge from network data, and secure the network. Drago has a Ph.D. from the University of Twente, the Netherlands. He was awarded the IETF/IRTF Applied Networking Research Prize.



Giulia Milan received her master's degree in Computer Engineering from PoliTO. She is currently pursuing a Ph.D. degree at University of California, San Diego. She worked as a research engineer at Huawei Technologies in Paris, France and did an internship at Scripps Research in San Diego. Her research interests are in the field of data analysis and machine learning.

Zied Ben Houidi is a Principal AI Researcher in the Huawei Paris Research Center working on the intersection of NLP and networks with applications to network control, data analysis and security. He received his PhD from Université Pierre et Marie Curie in France while working at Orange Labs. He then joined Bell Labs where he led various research projects on network data valorization (e.g. human-level behavior analytics) as well as automated reasoning for standards specification.

Dario Rossi is the Director of Huawei AI4NET Lab and the DataCom Lab at the Paris Research Center, France. He held Full Professor positions at Telecom Paris and Ecole Polytechnique, was the holder of Cisco's Chair NewNet@Paris, and has coauthored 20+ patents and 200+ papers with 7500+ citations. He is a Senior Member of IEEE and ACM, and the recipient of 9 best paper awards, the Google Faculty Research Award (2015), and the IRTF Applied Network Research Prize (2016).