DISTILLER: Encrypted Traffic Classification via Multimodal Multitask Deep Learning

Giuseppe Aceto[a], Domenico Ciuonzo[a], Antonio Montieri[a], Antonio Pescapé[a]

[a]*University of Napoli "Federico II", Italy*

Abstract

Traffic classification, i.e. the inference of applications and/or services from their network traffic, represents the workhorse for service management and the enabler for valuable profiling information. The growing trend toward encrypted protocols and the fast-evolving nature of network traffic are obsoleting the traffic-classification design solutions based on payload-inspection or machine learning. Conversely, deep learning is currently foreseen as a viable means to design traffic classifiers based on automatically-extracted features. These reflect the complex patterns distilled from the multifaceted (encrypted) traffic, that implicitly carries information in "multimodal" fashion, and can be also used in application scenarios with diversified network visibility for (simultaneously) tackling multiple classification tasks. To this end, in this paper a novel multimodal multitask deep learning approach for traffic classification is proposed, leading to the DISTILLER classifier. The latter is able to capitalize traffic-data heterogeneity (by learning both intra- and inter-modality dependencies), overcome performance limitations of existing (myopic) single-modal deep learning-based traffic classification proposals, and simultaneously solve different traffic categorization problems associated to different providers' desiderata. Based on a public dataset of encrypted traffic, we evaluate DISTILLER in a fair comparison with state-of-the-art deep learning architectures proposed for encrypted traffic classification (and based on single-modality philosophy). Results show the gains of our proposal over both multitask extensions of single-task baselines and native multitask architectures.

Keywords: deep learning, encrypted traffic, traffic classification, multimodal learning, multitask learning.

1. Introduction

The clear understanding of the processes occurring in networks is of paramount importance for multiple stakeholders, including network operators, who aim at the full visibility required by both network management and security. Indeed, the effectiveness of security and quality-of-service enforcement devices, as well as network monitors, is limited (or even hampered) when there is no accurate knowledge of the application generating the observed traffic.

The process of inferring such knowledge, known as Traffic Classification (TC), has a long-established application in several fields [1]. However, TC is severely challenged by both the volume growth and the evolving nature of the traffic traversing today's networks, that in turn is impacted by the new ways users behave, interact, and access the network [2]. Also, the widespread adoption [3] of encrypted protocols (e.g., Transport Layer Security) as well as network address translation and dynamic ports, increases the difficulty of accurate TC, defeating established approaches such as *deep packet inspection* and *port-based* methods, and can be only bypassed in closed-world (e.g., enterprise) scenarios by man-in-the-middle proxies [4].

Interestingly, the assessment of TC performance provides valuable insight also to designers of privacy-preserving protocols, and obfuscation and circumvention techniques [5, 6]. Indeed, stronger privacy needs have arisen in recent years, especially for anonymity tools and censorship evasion [7].

In this complex scenario, Machine Learning (ML) classifiers have proved to be the most appropriate for modern-traffic classification, since they also suit Encrypted Traffic (ET) while not necessarily relying on port information [8]. However, their use relies on obtaining handcrafted (domain-expert driven) features, which in TC context usually correspond to packet-sequence statistics. Such *feature engineering* process is unable to cope with modern network-traffic evolution, and impairs the design of both *accurate* and *up-to-date* traffic classifiers [9] using "traditional" ML approaches [10, 11, 12].

Therefore, Deep Learning (DL) has emerged as the methodological set of algorithmic solutions toward the fulfillment of high performance in the dynamic and challenging (encrypted) TC contexts. Indeed, DL allows to train classifiers directly from input data by automatically distilling structured and complex feature representations [13]. Accordingly, in the last few years several works have started tackling TC via DL [14, 15, 16, 17]. However, these attempts revealed a low level of maturity in applying DL to this specifically hard problem: we found in [18] how misguided design choices in this field led to *biased* conclusions or inflated classification outcomes. Moreover, DL approaches provide new performance enhancement possibilities: by neglecting these, designers would fail to harness DL full potential.

To face these shortcomings in TC state-of-the-art, in [18] we laid a sound groundwork for the design of DL-based classifiers aimed at highly-diverse traffic. The objective was two-fold: (*i*)

Email addresses: giuseppe.aceto@unina.it (Giuseppe Aceto), domenico.ciuonzo@unina.it (Domenico Ciuonzo), antonio.montieri@unina.it (Antonio Montieri), pescape@unina.it (Antonio Pescapé)

avoiding the pitfalls of naïve adoption of DL to TC and (*ii*) allowing for best exploiting the potential of DL architectures. Then, in [19] we have leveraged such groundwork to explore multimodal approaches to enhance the performance of ET classification. Indeed, we have proved multimodal approaches to be able to leverage traffic data according to the different input information (e.g., payload bytes or header fields), i.e. to represent the same concept according to different "views" or "modalities". Hence, a multimodal DL traffic classifier can automatically learn a hierarchical representation exploiting jointly all the available modalities, without the need for handcrafting modality-specific and ad-hoc features for a certain ML approach.

However, the sophisticated management required by next-generation networks relies on the accomplishment of a number of diversified *network visibility* tasks [20, 21]. This motivates the need for solving several traffic-analysis tasks, there in included a number of different TC tasks, each associated to a given visibility viewpoint. In order to meet the above desiderata, the aforementioned tasks need to be performed with a satisfactory accuracy and reasonable complexity.

Accordingly, in this paper, we broaden our exploration with *multimodal* approaches applied to TC, additionally investigating *multitask learning*. Multitask learning is an approach to inductive transfer that improves learning for one task by using information in the training signals of other related tasks [22]. This is accomplished by learning tasks in parallel while using a shared representation. On the one hand, compared with multiple single-task models, multitask learning can reduce computational overhead, because only one—comprehensive—model is learned (resp. used) in the training (resp. testing) phase. This has the benefit of limiting redundancy by sharing part of the feature-learning architecture. On the other hand, this approach promises improved generalization and better overall classification performance.

More specifically, the *main contributions* provided by our work are summarized as follows:

- We capitalize the joint and effective adoption of *multitask* and *multimodal* deep learning to devise the DISTILLER (encrypteD multItaSk Traffic classIfication via muLtimodaL dEep leaRning) classifier. The proposed classifier relies on a *general DL architecture* which provides the simultaneous solution of multiple (related) ET classification tasks (*multitask*) via the effective exploitation of heterogeneous (*multimodal*) inputs. Our proposal thus provides support to application scenarios with diversified network visibility by accomplishing automatic feature learning from heterogeneous and structured traffic input data.
- We propose a learning procedure for DISTILLER classifier based on a *wise* two-step training phase, made of pre-training and fine-tuning. Such procedure avoids convergence of the learning process to suboptimal parameter values, as a result of the overfitting toward the strongest modality [13]. The reason for a pre-training phase is the need for correctly distilling discriminative information from each modality so as to later capitalize the advantage of the multimodal traffic representation during the fine-tuning.
- We experimentally compare a specific instance of the DISTILLER architecture on the *public dataset* `ISCX VPN-nonVPN` [23] to validate our proposal, collecting traffic from *activity of real human users*. Such well-known dataset naturally surfaces the *multitask* goals related to network monitoring, and provides a supporting example for the motivations of this work.
- We favorably compare our approach with state-of-the-art multitask DL traffic classifiers [15, 17, 20, 21, 24, 25, 26], showing DISTILLER *achieves gains over all* the TC tasks considered, while having appreciably-lower complexity than the best performing baseline. Additionally, the outcome of an in-depth analysis concerning the fine-grained behavior of such classifiers highlight that our proposal reports also improved per-class performance, a better soft-output behavior, and improved calibration in the associated classification confidence. Finally, an investigation of the tasks' relationship and the degree of knowledge transferred among them highlights the advantage in solving multitask problems by means of multimodal approaches.

The rest of the paper is organized as follows. Sec. 2 reviews the literature on ET classification, including most relevant DL works, and details the paper contribution; Sec. 3 describes our DISTILLER encrypted-traffic classifier; the experimental setup is described in Sec. 4, while discussion of the results is reported in Sec. 5; finally, Sec. 6 provides conclusions and future research avenues.

2. Related Work and Contribution Positioning

Several recent works have faced the problem of TC by means of ML-based [33] and DL-based approaches [14, 17, 27]. Some works have leveraged DL approaches specialized for time-series [17, 31]. Also, reflecting the modern nature of traffic, most approaches have focused on ET [15, 30, 33, 34, 35]. We report relevant works in these lines of research in Tab. 1, that *summarizes the key aspects of each paper*.

Notably, the vast majority of these works have adopted publicly-available datasets (**Open** column) for validating and assessing the performance of their proposals, save from a few cases [14, 17, 29, 21]. With few exceptions [16, 21], the traffic object (**TO** column) usually considered—both for extracting input data and assigning classification labels—is the bidirectional flow, or *biflow*. Additionally, the input data are mostly provided to the classifiers as **Raw** inputs, as opposed to the inefficient ad-hoc feature extraction [20, 25, 26, 28]. Indeed, this latter procedure partially defeats a major advantage of DL methods, namely the automatic extraction of features from input data and the consequent reduced need of human-expert intervention in this process.

A great deal of variation can be found regarding the specific algorithms proposed for the classification task (**DL Classifier** column), including Deep Neural Networks (DNNs), dif-

Table 1: Related and notable works adopting DL for TC. Papers are grouped based on the type of architecture adopted: the first group encompasses single-modal/single-task architectures, the second group multimodal/single-task (or partially multitask) architectures, the third group single-modal/multitask architectures. The last row summarizes the present paper. Within each group the works are reported in chronological order. Columns and acronyms meaning is reported hereafter.
Open: publicly available dataset.
Traffic object (TO): biflow (BF), flow (F), HTTP session (H), packet (P), Transport Block Size aggregation (T).
Input Data: Raw data of PCAP trace (PCAP), X^{th} layer of ISO/OSI model (LX).
DL Classifier: AutoEncoder (AE), Auxiliary Classifier Generative Adversarial Network (AC-GAN), Bidirectional Gated Recurrent Unit (BiGRU), Convolutional Neural Network (CNN), Deep Belief Network (DBN), Deep Neural Network (DNN), Hierarchical Attention Network (HAN), Long Short-Term Memory (LSTM), MultiLayer Perceptron (MLP), Stacked AutoEncoder (SAE), Variational AutoEncoder (VAE); + symbol indicates hybrid architectures; & symbol indicates intermediate fusion of input data.
Multimodal (MM). Multitask (MT), *Supervised Shared Representation (SSR), Training-Phase Specification (TPS).*
● present, ○ lacking, ◐ partial, — not applicable.

MM	MT	Open	TO	Input Data	Raw	DL Classifier	SSR	TPS	Research
○	○	○	BF	TCP payload [1000 B]	●	SAE	—	●	Z. Wang, 2015, *Briefing Black Hat USA* [14]
○	○	○	BF	6 fields [20 packets]	●	2D-CNN+LSTM	—	●	M. Lopez-Martin *et al.*, 2017, *IEEE Access* [17]
○	○	●	F/BF	PCAP [784 B] L4 payload [784 B]	●	1D-CNN	—	●	W. Wang *et al.*, 2017, *Proc. IEEE ISI* [15]
○	○	●	H	HTTP fields [28×36 B]	●	VAE	—	●	D. Li *et al.*, 2017, *Proc. IEEE CIS* [27]
○	○	●	BF	Flow-based statistics	○	AC-GAN	—	●	L. Vu *et al.*, 2018, *Proc. ACM SoICT* [28]
○	○	○	BF	IP packet lengths	●	BiGRU	—	●	C. Liu *et al.*, 2019, *Proc. IEEE INFOCOM* [29]
○	○	●	BF	PCAP [900 B]†	●	1D-CNN, LSTM, SAE	—	●	Y. Zeng *et al.*, 2019, *IEEE Access* [30]
○	○	●	BF	L4 payload [10×1500B] L2 payload◦ [10×1500B]	●	Attention-based LSTM, HAN	—	●	H. Yao *et al.*, 2019, *IEEE TBD* [31]
○	○	●	P	L2 payload [1500 B]	●	SAE, 1D-CNN	—	●	M. Lotfollahi *et al.*, 2020, *Soft Computing* [16]
●	○	◐	BF	L4 payload [516 B] 4 fields [12 packets]	●	1D-CNN & BiGRU	—	●	G. Aceto *et al.*, 2019, *Elsevier ComNet* [19]
●	◐	—	∗	L4 payload [N_b B] Packet fields [N_p packets]	●	*Generic Framework*	—	○	G. Aceto *et al.*, 2020, *Elsevier NEUCOM* [32]
○	●	●	BF	PCAP [1024 B]	●	2D-CNN	●	●	H. Huang *et al.*, 2018, *IAOE iJET* [24]
○	●	●	BF	Flow-based statistics	○	DNN	●	●	H. Sun *et al.*, 2019, *IEEE Access* [25]
○	●	●	BF	Flow-based statistics	○	DNN	●	●	Y. Zhao *et al.*, 2019, *Proc. ACM SoICT* [26]
○	●	●	BF	Packet length, direction Inter-arrival Time Flow-based statistics	○	1D-CNN	●	●	S. Rezaei and X. Liu, 2020, *Proc. IEEE ICCCN* [20]
○	●	○	T	Transport Block Size [#Bytes / 1 second]	●	AE, Seq2seq-AE w/ LSTM	○	●	A. Rago *et al.*, 2020, *IEEE TVT* [21]
●	●	●	BF	L4 payload [784 B] 4 fields [32 packets]	●	1D-CNN & BiGRU 1D-CNN⋆, 2D-CNN⋆ MLP⋆, 2D-CNN+LSTM⋆	●	●	***This Paper***

† TCP/UDP headers and MAC addresses are removed.
◦ IP addresses are removed.
⋆ Baselines for performance comparison.

ferent variants of AutoEncoders (AEs), both one- and two-dimensional Convolutional Neural Networks (1D- and 2D-CNNs), different types of Recurrent Neural Networks (RNNs), and Generative Adversarial Networks (GANs).

Finally, regarding the proposed architectures, only few works have considered *multimodal* inputs (**MM** column) [19, 32], with the lower layers being trained on specific subsets of the—heterogeneous—inputs. Relatively more works have devised *multitask* architectures (**MT** column) [24, 25, 20, 26, 21]: hereinafter we discuss them in detail.

Before doing that, for completeness, we also briefly discuss the works [15, 17], providing single-task architectures (**MT** = "○") belonging to two relevant DL approaches, whose multitask *extensions* are described (resp. evaluated) in later Sec. 4 (resp. Sec. 5) as baselines. In detail, Wang et al. [15] have applied DL to malware-traffic classification, proposing a method based on 1D-CNN tailored for ET. The experimental evaluation has been conducted on a selection of the public `ISCX VPN-nonVPN` dataset [23] and is divided in four different TC problems: (*i*) VPN/nonVPN, (*ii*) 6 encrypted traffic classes, (*iii*) 6 VPN-tunneled traffic classes, and (*iv*) 12 encrypted applications. Different inputs have been considered, including *biased* ones [18]. Conversely, Lopez-Martin et al. [17] have proposed different hybrid DL architectures originating from the combination of Long Short-Term Memory (LSTM) and 2D-convolutional layers. The proposals, evaluated on real traffic (from an academic backbone network, not publicly available) have shown high performance, also highlighting a penalty associated to the use of inter-arrival times as input. Unluckily, also in [17] biased inputs encompassing source and destination ports have been used.

Recently, Huang et al. [24] have applied the multitask learning paradigm to solve the tasks of (*i*) malware (binary) detection, (*ii*) recognition of VPN-encapsulation (binary), and (*iii*) Trojan classification (9 classes). The proposed DL algorithm is a 2D-CNN, tested on an assembled dataset obtained from the `CTU-13` (malware) and `ISCX VPN-nonVPN` traffic datasets. Also in this case, following the work in [15], *biased* input data have been employed.

Another multitask DL architecture has been proposed by Zhao et al. [26] in the context of *federated learning*. This scenario, mainly motivated by privacy concerns, performs model learning in a distributed fashion, preventing local data (e.g., traffic traces) to be shared, and communicating and merging only the partial models learned locally. The considered tasks are: anomaly detection (binary), VPN recognition (binary), and TC (6 classes). Notably, a set of statistical features is defined on the biflow, partially defying the *feature learning* capability of DL algorithms. The performance has been compared with centralized (i.e. non federated-learning based) methods showing slight improvement (maximum +1.5% on accuracy or recall), but a significant reduction in training time with respect to the baseline architecture (i.e. a single-task DNN).

We also report for completeness a recent work by Rezaei and Liu [20], where a 1D-CNN architecture has been proposed for classifying the traffic (aggregated in 5 traffic categories) and also "predicting" biflow bandwidth and duration (quantized over 5 and 4 intervals with saturation at 1 Mbps and 60 s, respectively). The traffic categories from the `ISCX VPN/non-VPN` dataset are defined according to QoS characteristics. The performance has been evaluated in a *transfer-learning* setup, with one of the tasks (i.e. traffic-category classification) limited by a *scarce ground-truth*.

A similar set of classification tasks (i.e. duration, flow rate, and application) has been considered by Sun et al. [25]. For both duration and flow rate, a two-level quantization (binary classification) has been applied. Conversely, for application-traffic classification, the authors have considered between $11 \sim 50$ classes according to the specific dataset used. Like several others works on DL applied to TC, inefficient ad-hoc feature engineering (e.g., feature selection and statistical preprocessing) has been performed referring to previous ML literature. In addition to this, transport-level ports have been used as input: a choice hardly in line with realistic traffic scenarios (e.g., mobile apps, encryption, tunneling, NAT). The above approach has been evaluated in *transfer-learning* and *one-shot learning* schemes, leveraging the multitask approach to improve performance in case of ground-truth scarcity for a single task.

Recently, Rago et al. [21] have devised a link-level multitask approach for application classification and prediction. The input data are extracted from the sequence of *Transport Block Size* values (a field of the LTE downlink control channel) aggregated on a 1-second interval. This traffic object can be equated with a byte-rate measurement over fixed time intervals. The dataset has been collected by the authors and it is not publicly available. The proposed method learns the shared features by either (*a*) standard or (*b*) Seq2seq AEs (implemented via LSTM layers). Specifically, this work has tailored a multitask model running directly at the edge of the network to foresee the traffic type to be served and the resource allocation pattern of each service during its execution. Furthermore, the impact of the sliding observation-window on classification and prediction performance, as well as complexity, has been also investigated.

Notably, by looking at the above literature tackling diversified TC tasks via multitask DL, it is apparent that almost all the works exploit the multitask concept with a *supervised* aim (column **SSR**), except for [21]. In detail, in the supervised case, the set of layers constituting the shared representation is obtained by enforcing the solution of the considered tasks, rather than in an unsupervised fashion. Hence, by doing so, better-performing shared representations can be attained.

Unlike the above literature, in our previous work [32], we have discussed a general framework to effectively apply DL to modern TC tasks, focusing *only* on the architectural view. While proposing a multimodal and multitask framework, in said work we have investigated its effectiveness in a *multimodal* (experimental) setup only, to tackle mobile-ET classification. Consequently, the aforementioned study (due to its more general and architectural focus) is the only one *not* providing a rigorous training-phase specification (**TPS** column) of the architecture.

Accordingly, based on the previous discussion, we *position our work against the state-of-the-art* as detailed hereinafter:

- This paper tackles classification of ET via *sophisticated*

DL architectures. Hence, our proposal avoids the redundant (and potentially-degrading) preliminary (manual) feature engineering phase (as opposed to e.g. [20, 26]) and feeds the architecture with *unbiased* inputs (as opposed to e.g. [16]), strengthening both the significance and the generalization of the reported results.

- This paper investigates the joint and effective adoption of *multitask* and *multimodal* learning to devise the DISTILLER classifier. Indeed, on the one hand, DISTILLER generalizes the adoption of multimodal architectures (previously conceived only for single-task TC [19]) to application scenarios with diversified network visibility. This is achieved by automatic feature learning from heterogeneous and structured traffic data given as input. On the other hand, DISTILLER is able to overcome the limitations of existing multitask architectures, based only on single-modal DL [20, 21, 24, 25, 26]. Also, the resulting architecture comes with the full specification of the (two-step) training phase (differently from [32]). As a result, we progress on both the nature of the classification goal(s), and the associated overall architecture.

- This paper employs the *public dataset* `ISCX VPN-nonVPN` [23] to validate our proposal, collecting traffic from *activity of real human users*. Such well-known dataset naturally surfaces the *multitask* goals related to network monitoring, and provides a supporting example for the motivations of our work.

- This paper experimentally compares a specific instance of the DISTILLER architecture with the best proposals from the state-of-the-art [15, 17, 20, 21, 24, 25, 26]—for a total of eight baselines—implementing all with the same technologies and feeding and evaluating all in fair conditions. Our comparison also includes an in-depth analysis of the fine-grained behavior of such classifiers, dissected along the considered tasks. Also, we investigate the tasks' relationship and the degree of knowledge transferred among them.

3. Multimodal Multitask Deep Learning–based Traffic Classification

Herein, we describe the DISTILLER classifier, starting from the overall architecture specification in Sec. 3.1 (depicted via Fig. 1a). We then focus, in Sec. 3.2, on the general procedure adopted for training it (see Figs. 1b and 1c). Finally, in Sec. 3.3, we describe the *specific instance* of DISTILLER (see Fig. 2) chosen for the considered multitask TC problem and evaluated in later Sec. 5. For reader's convenience, Tab. 2 summarizes the mathematical notations used in the following.

3.1. Overall Architecture

In this section, we describe the proposed methodology for multipurpose ET classification via multimodal multitask DL. In detail, we assume that we are required to solve $v = 1, \dots, V$ different TC problems (*tasks*). Formally, given each TC object observed [1], the v^{th} TC problem (T_v) consists in assigning a label among L_v classes within the set $\{1, \cdots, L_v\}$.

Table 2: Chart of the mathematical symbols and notations used in the definition the DISTILLER architecture. Symbols and notations are grouped based on their context: input data, architecture, and loss functions & training.

	Symbol	Definition
Input Data	V	Number of TC tasks
	M	Number of training samples
	L_v	Number of classes of v^{th} TC task
	$\boldsymbol{x}(m)$	m^{th} sample of the training set
	$\ell^v(m)$	Label (true class) of $\boldsymbol{x}(m)$ for v^{th} TC task
	$\boldsymbol{t}^v(m)$	One-hot representation of $\ell^v(m)$
Architecture	P	Number of different modalities
	J_p	Number of single-modality layers of p^{th} modality
	S	Number of shared-representation layers
	U_v	Number of task-specific layers of v^{th} TC task
	$\boldsymbol{\theta}_p$	Parameters of the p^{th} single-modality layer
	$\boldsymbol{\theta}_p^{\text{stub}}$	Parameters of the p^{th} "stub" layer
	$\boldsymbol{\theta}_0$	Parameters of shared-representation and task-specific layers
	$\boldsymbol{\theta}_p^{\downarrow}$	Parameters optimized only in pre-training
	$\boldsymbol{\theta}_p^{\uparrow}$	Parameters optimized in pre-training and fine-tuning
Losses & Training	$\mathcal{L}_p(\cdot)$	Loss function minimized in pre-training of p^{th} modality
	$\mathcal{L}(\cdot)$	Loss function minimized in fine-tuning
	$\text{CE}(\boldsymbol{t}, \boldsymbol{c})$	Categorical cross-entropy between $\boldsymbol{t}$ and $\boldsymbol{c}$
	$\boldsymbol{c}^v(m)$	Predicted class confidences of $\boldsymbol{x}(m)$ for v^{th} TC task
	λ_v	Preference level of v^{th} task in multitask objective function
	$\hat{\boldsymbol{\theta}}_p$	Pre-trained parameters of the p^{th} single-modality layers
	$\hat{\boldsymbol{\theta}}_p^{\text{stub}}$	Trained parameters of the p^{th} "stub" layer

As depicted in Fig. 1a, when performing TC, a necessary prerequisite consists in segmenting the raw network traffic into elementary entities constituting the samples of the learning task. Such a process is known as **traffic object segmentation** and it is in charge of "packing" the raw traffic into distinct TC objects [1], each constituting a subset of network packets sharing some common properties defined by the segmentation rule.

We remark that DL-based traffic classifiers learn the distinctive fingerprint of each traffic type/application via a *training set*. In a multitask TC context, we make the assumption that each traffic object of the above set is associated with as many labels as the TC tasks to be solved. Hence, for notational convenience, we define the m^{th} TC object of the training set (made of M samples) as $\boldsymbol{x}(m)$, while the corresponding label of the v^{th} classification task as $\ell^v(m)$. The above label may belong to one out of L_v different classes, namely $\ell^v(m) \in \{1, \dots, L_v\}$. The main advantage of DL (as opposed to ML) approaches for TC is to overcome the little-adaptable feature design process [18]. This is due to their ability to learn traffic fingerprints in an *end-to-end* fashion, that is *directly* from the type of input selected. Still, traffic data are intrinsically highly-structured, as each sample $\boldsymbol{x}(m)$ contains information from the whole protocol stack. As a result, *early fusion* by a monolithic DL architecture taking the whole input coming from a TC object in bulk, is likely to be suboptimal. Indeed, the parameter set would overfit to one input subset while underfitting the others. This often precludes reaping the true benefits of multi-modality. Conversely, *late*

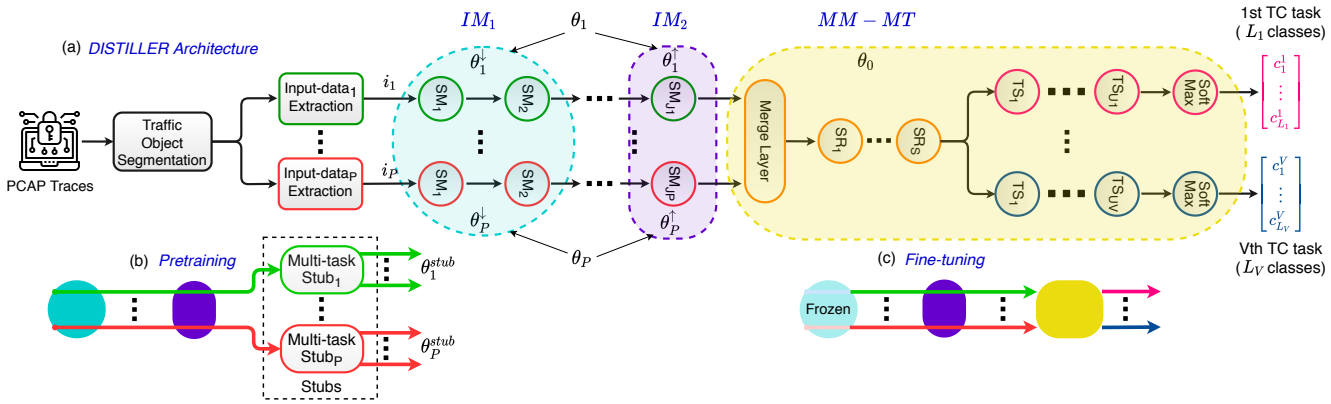


Figure 1: Architectural view of the DISTILLER framework. (*a*) depicts the architecture by highlighting single-modality representation layers, differentiated as those that are only pre-trained (IM_1) and those that are also fine-tuned (IM_2), and shared representation-layers (MM-MT), along with the corresponding parameter set. (*b*) and (*c*) depict the proposed training procedure based on pre-training and fine-tuning.

fusion via the capitalization of score-results (viz. confidence vectors) of DL-based traffic classifiers built on different modalities, does not fully exploit the benefits of multi-modality (see e.g. the results [19] pertaining to the single-task case). Accordingly, we foresee multimodal multitask DL as the appealing means toward a sophisticated form of information fusion, termed *intermediate fusion* [36], overcoming these limitations by offering a flexible tool for practical ET classification, able to solve multiple tasks by processing *effectively* the heterogeneous inputs available.

The **architecture** of our DISTILLER classifier is depicted in Fig. 1a at an abstract level and described hereinafter. DISTILLER is fed with P different inputs (*modalities*) for each TC object to be classified, with the p^{th} modality provided from `Input-data`$_p$ extraction block. The first part (IM_1+IM_2) of our deep network architecture consists of J_p *single-modality* (`SM`) layers, which are input-specific and allow to extract in an increasingly-abstract fashion the discriminative features pertaining to the p^{th} modality alone. The trainable parameters of the first part are collected in $\boldsymbol{\theta}_p$.

On top of these layers, the abstract features are joined via a *merge layer*, which represents the first layer channeling the modality-specific distilled information toward a joint multimodal multitask (shared) representation [37]. Although the most general choice is represented by a *concatenation operation*, other options may be pursued in case the features originating from different modalities have the *same size* (e.g., average and entry-wise maximum).

Differently, the second part (MM-MT) of the architecture consists of S *shared-representation* (`SR`) layers and U_v *task-specific* (`TS`) layers. The former set of layers distills the features capturing inter-modality dependencies. Conversely, the latter set of layers synthesizes the task-oriented features (of the v^{th} task) from the shared ones. Accordingly, DISTILLER is based on a *hard parameter sharing* approach, based on the categorization in [22]. Finally, the architecture is completed with a `softmax` layer for each ET classification-task to solve, namely returning the corresponding soft-output (prediction) vector $\boldsymbol{c}^v \triangleq \begin{bmatrix} c_1^v & \cdots & c_{L_v}^v \end{bmatrix}$ (with $\boldsymbol{c}^v \in [0,1]^{L_v}$) for task v. The trainable parameters of the second part are collected in $\boldsymbol{\theta}_0$. We recall that common choices for elementary DL layers are *dense*, *convolutional*, *pooling*, and *recurrent layers*, which can be jointly employed within a hybrid DL architecture[1] capitalizing the "connectionist" philosophy [13].

3.2. Loss Function Definition and Training Procedure

The **training procedure** proposed for our DISTILLER classifier consists of a *two-stage phase*: (*i*) *pre-training* and (*ii*) *fine-tuning* (Figs. 1b and 1c, respectively). The reason for a preliminary pre-training procedure is to correctly distill discriminative information from each modality so as to capitalize the advantage of the multimodal traffic representation. In other words, pre-training allows the DL branch of each modality to be able to acceptably solve—by itself—all the considered tasks [13].

Precisely, each single-modality stack is first (pre-)trained independently (see Fig. 1b), that is *without* MM-MT (corresponding to `SR` and `TS` layers) and by topping each modality chain with V different softmax-layer "stubs". The trainable parameters of the V stubs (for p^{th} modality) are collected within $\boldsymbol{\theta}_p^{\text{stub}}$. Specifically, the p^{th} "stubbed" chain is trained to minimize the classification loss function $\mathcal{L}_p(\cdot)$ to promote p^{th} modality capability to solve the V different TC tasks *alone*. Accordingly, we aim to minimize a *weighted sum* of the *categorical Cross-Entropy (CE)* of each TC task, namely:

$$\mathcal{L}_p\left(\boldsymbol{\theta}_p, \boldsymbol{\theta}_p^{\text{stub}}\right) \triangleq \sum_{v=1}^{V} \lambda_v \left\{ \sum_{m=1}^{M} \text{CE}(\boldsymbol{t}^v(m),\ \boldsymbol{c}^v(m)\,[\boldsymbol{\theta}_p, \boldsymbol{\theta}_p^{\text{stub}}]) \right\} \quad (1)$$

Such a distance is measured via the functional $\text{CE}(\boldsymbol{t}, \boldsymbol{c}) \triangleq -\{\sum_{\ell=1} t_\ell \log c_\ell\}$, denoting the CE distance for the generic training sample. In Eq. (1), the vector $\boldsymbol{c}^v(m) \triangleq \begin{bmatrix} c_1^v(m) & \cdots & c_{L_v}^v(m) \end{bmatrix}^T$ collects the predicted class confidences of the DL classifier (which depend on DL network parameters)

[1] Also, to promote *regularization* (to avoid overfitting), dropout between successive layers and early-stopping techniques are commonly adopted [13].

for the label of the m^{th} training sample on the v^{th} task. Differently, $\boldsymbol{t}^v(m) \triangleq \left[t_1^v(m) \quad \cdots \quad t_{L_v}^v(m)\right]^T$ denotes the corresponding *one-hot* representation of the label for the v^{th} learning task $\ell^v(m)$. The aim of a high-performing traffic classifier on the v^{th} task is to have the confidence vector $\boldsymbol{c}^v(m)$ as close as possible to the (ground-truth originated) one-hot vector $\boldsymbol{t}^v(m)$. Finally, since our architecture is in charge of solving multiple learning tasks at once, the weight λ_v represents the preference level of the v^{th} task in the multitask objective function to be optimized. The learned parameters from the above optimization are indicated with $(\hat{\boldsymbol{\theta}}_p, \hat{\boldsymbol{\theta}}_p^{\text{stub}})$.

Then, during the *fine-tuning* (see Fig. 1c), the softmax stubs are removed (i.e. $\hat{\boldsymbol{\theta}}_1^{\text{stub}}, \cdots, \hat{\boldsymbol{\theta}}_P^{\text{stub}}$ are discarded from the optimization) and the *whole* DISTILLER classifier is trained (i.e. including both the parameter sets $\boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_P$ and $\boldsymbol{\theta}_0$, associated to the IM_1+IM_2 and *MM-MT* blocks, respectively). However, as a result of the pre-training, a share of `SM` layers (the "low" layers in DL hierarchy, named IM_1) are typically *frozen* when fine-tuning is performed: low-level layers refer indeed to *intra-modality automatic* feature extraction. In other terms, within $\boldsymbol{\theta}_p \triangleq \left[\boldsymbol{\theta}_p^{\downarrow} \quad \boldsymbol{\theta}_p^{\uparrow}\right]$ only the subset $\boldsymbol{\theta}_p^{\uparrow}$ is (further) optimized during fine-tuning (i.e. the parameters corresponding to IM_2), while $\boldsymbol{\theta}_p^{\downarrow}$ is kept *fixed* to the value learned during pre-training, i.e. $\boldsymbol{\theta}_p^{\downarrow} = \hat{\boldsymbol{\theta}}_p^{\downarrow}$. As a result, the following *weighted* form of the categorical CE is minimized:

$$\mathcal{L}\left(\boldsymbol{\theta}_{1:P}^{\uparrow}, \boldsymbol{\theta}_0\right) \triangleq \sum_{v=1}^{V} \lambda_v \sum_{m=1}^{M} \text{CE}(\boldsymbol{t}^v(m),\ \boldsymbol{c}^v(m)[\boldsymbol{\theta}_{1:P}^{\uparrow},\ \boldsymbol{\theta}_0]) \qquad (2)$$

In the above equation, the weights $\lambda_1, \ldots, \lambda_V$ retain the same interpretation as in the pre-training objective function reported in Eq. (1). The loss functions concerning pre-training (single-modal multitask, $\mathcal{L}_p(\cdot)$) and fine-tuning (multimodal multitask, $\mathcal{L}(\cdot)$) phases are minimized via standard first-order local optimizers (e.g., Adam, AdaGrad, etc.), resorting to the usual back-propagation for gradient evaluation [13]. Hereinafter, we detail the specific instance obtained from the general architecture of DISTILLER and used for the experimental evaluation in Sec. 5.

3.3. Description of Proposed Instance

Aiming at a consistent comparison with earlier works [15, 17, 18, 19], this particular implementation of the devised DISTILLER architecture operates with **biflow TC objects** (see Sec. 4.1) and is made of $\boldsymbol{P = 2}$ **modalities**. Furthermore, the considered multitask TC scenario is composed of $\boldsymbol{V = 3}$ **TC tasks**. These correspond to (T_1) encapsulation identification ($L_1 = 2$ classes), (T_2) traffic type recognition ($L_2 = 6$ classes), and (T_3) application classification ($L_3 = 15$ classes).

Fig. 2 depicts the detailed composition of the proposed DISTILLER instance. We observe that the present network is only one of the possible instances which may be (*a*) designed according to the general architecture described in Sec. 3.1 (represented graphically via Fig. 1) and (*b*) trained via the corresponding procedure reported in Sec. 3.2.

The input data fed to the DISTILLER classifier belong to *two* types: (a) the first N_b bytes of transport-layer payload (`PAY`) of the TC object [14, 15]; (b) informative protocol header fields (`HDR`) of the first N_p packets [17]. In the *first* case, the input is represented in *binary format*, arranged in a *byte-wise* fashion and normalized within [0, 1]. The *second* type of input data is constituted by: (i) number of bytes in transport-layer payload, (ii) TCP window size (set to *zero* for UDP packets), (iii) inter-arrival time, and (iv) packet direction $\in \{0, 1\}$—of the first N_p packets [17]. We underline that, in *both* cases, *longer* (resp. *shorter*) instances are truncated (resp. padded with zeros) to the designed length of bytes (N_b) or packets (N_p). These specific input data derive from the necessity of avoiding *biased* inputs—a common pitfall in related works [15, 17]—included for instance in PCAP metadata, data-link layer, and some transport-layer header fields (e.g., source and destination ports), as they may lead to inflated performance and lack of generalization [18].

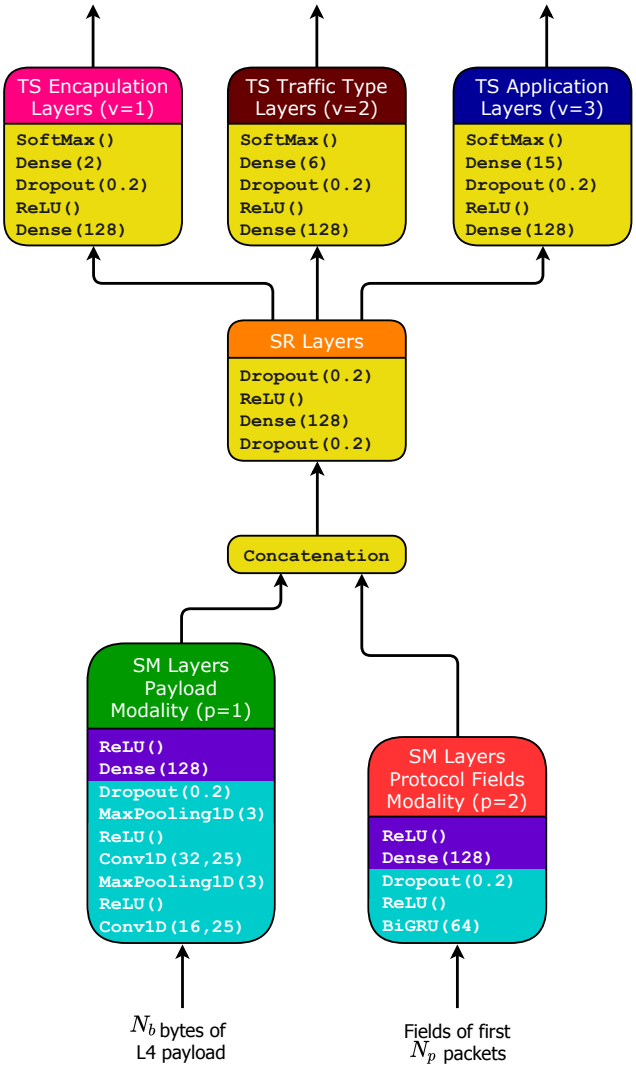


Figure 2: Proposed instance of the DISTILLER classifier. The layer parameters represent: `Conv1D(#filters, kernel_size)`, `BiGRU(#units)`, `Dense(#nodes)`, `MaxPooling1D(pool_size)`, `Dropout(rate)`. Background colors highlight the sets of layers that are pre-trained or fine-tuned: IM_1 in cyan, IM_2 in purple, and *MM-MT* in yellow.

We notice that the two considered input types (viz. "traffic-originated" modalities) refer to different levels of abstraction (biflow vs. packet) and standpoints (encryption-dependent vs. encryption-independent) and thus are naturally conducive to the *multimodal* approach. Also, they suit well "early" TC [38], namely taking a classification decision based only on the first segments of the considered TC object.

The architecture implements[2] the `SM` layers of the "payload" modality ($p = 1$, grouped in green in Fig. 2) with two 1D convolutional layers (16 and 32 filters, respectively, with kernel size of 25 and unit stride), each followed by a 1D max-pooling layer (with unit stride and spatial extent equal to 3) and, finally, by one dense layer (128 nodes). On the other hand, the `SM` layers of the "protocol fields" modality ($p = 2$, grouped in red in Fig. 2) are, in order, a bidirectional GRU (BiGRU with 64 units and return-sequences behavior) and one dense layer (128 nodes).[3] The intermediate features of the two branches are then *merged* via a `concatenation` layer and fed to a single ($S = 1$) `SR` dense layer (128 nodes). The latter is connected to $V = 3$ layers, each constituting one `TS` dense layer (128 nodes) for the v^{th} task ($U_v = 1$), before the corresponding `softmax` layer (i.e. a dense layer with L_v nodes and softmax activation). In all the layers, the outputs are obtained via Rectifier Linear Unit (ReLU) activations. Finally, 20% dropout is applied after (*a*) each dense layer (including the concatenation layer) and (*b*) after flattening the 2D representation of both the stack of convolutional/pooling layers and BiGRU.

The considered DISTILLER instance is *trained* via the *two-stage phase* previously described: during *pre-training* phase, each single-modality stack is first (pre-)trained independently for 30 epochs each by topping V softmax layer stubs and by minimizing the loss $\mathcal{L}_p(\cdot)$ in Eq. (1). Then, *fine-tuning* of the whole architecture is performed for 40 epochs by minimizing the loss $\mathcal{L}(\cdot)$ in Eq. (2), after freezing IM_1 (corresponding to the two 1D convolutional and BiGRU layers, depicted with cyan background in Fig. 2).

For both phases, we employ the Adam optimizer [39] (with a batch size of 50) and early-stopping technique (to prevent overfitting) with a patience of 10 epochs and a minimum delta of 0.01 measured on the training accuracy of the hardest TC task (i.e. with the highest number of classes L_v). The Adam optimizer is set with a learning rate of $2 \cdot 10^{-3}$ and 10^{-3} during pre-training and fine-tuning, respectively. Also, the exponential decay rates for the estimates of the first-order and second-order moments are set to 0.9 and 0.999 (Keras default values), respectively. Finally, when not otherwise specified, we reasonably set the preference weights $\lambda_v = 1/V = 1/3$, i.e. a uniform allocation.

An example of the corresponding evolution of the loss function vs. the number of epochs associated to DISTILLER is reported in Fig. 3. Specifically, the two-phases (i.e. pre-training and fine-tuning) are highlighted therein with different background colors. First, during the *pre-training* phase the loss $\mathcal{L}_p(\cdot)$ is measured for the p^{th} modality stack at the output of the corresponding softmax stub. The two modalities are trained in parallel for 30 epochs each. Then, during the *fine-tuning* phase, the overall $\mathcal{L}(\cdot)$ is measured up to the remaining 40 epochs (the training may terminate earlier because of early stopping). In the above plot, for completeness, the corresponding (three) per-task contributions of $\mathcal{L}_p(\cdot)$ ($p = 1, 2$) and $\mathcal{L}(\cdot)$ are also reported (cf. Eqs. (1) and (2), respectively). It is evident that the exploitation of multi-modality by the proposed two-stage training process is able to achieve lower (multitask) loss functions as opposed to the single-task cases.

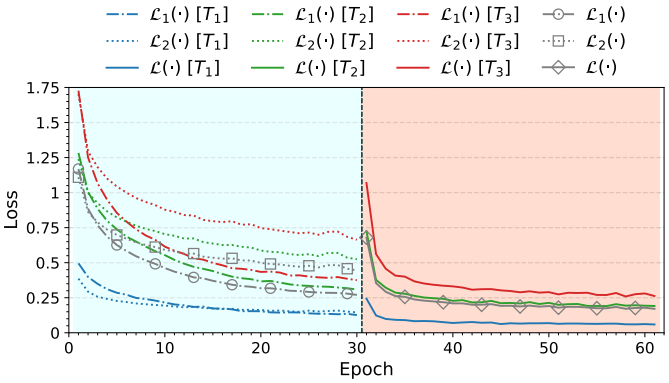


Figure 3: Loss function $\mathcal{L}_p(\cdot)$ of p^{th} single-modality stack during pre-training (in light blue) and loss function $\mathcal{L}(\cdot)$ during fine-tuning (in salmon) of the DISTILLER classifier. Results pertain to a single fold. Per-task components of both $\mathcal{L}_p(\cdot)$ and $\mathcal{L}(\cdot)$ are also shown. The pre-training of single-modality stacks is performed in parallel. The number of epochs of fine-tuning is less than that prescribed (i.e. 40 epochs) because of early-stopping.

4. Experimental Setup

The classifier instance taken from the general architecture of DISTILLER is carefully evaluated and compared against several multitask TC baselines on a real dataset. Accordingly, the present section provides a complete description of the experimental setup considered. Specifically, we first provide the description of the human-generated (public) dataset we have leveraged for this purpose and of the traffic segmentation we have performed to extract our classification samples from raw PCAP data (Sec. 4.1). Thereafter, we give details on the state-of-the-art multitask baselines taken into account (Sec. 4.2). Finally, we end the section (Sec. 4.3) with a description of the implementation details required for reproducibility of the considered setup.

4.1. Dataset and TC Object Description

Our experimental setup employs the `ISCX VPN-nonVPN` dataset [23] collected at the Canadian Institute for Cybersecurity and provided in raw PCAP format with trace-level labels,

[2] The choice of hyperparameters builds on our previous experience with multimodal DL architectures for TC [19], extended with a set of experiments varying the parameters of the dense layers.

[3] 1D convolutional layers aim at extracting spatially-invariant patterns from the payload, while (Bi)GRUs aim at capturing long-term dependencies pertaining to the initial segments of the TC object.

Table 3: ISCX VPN-nonVPN dataset tasks and classes. Classes are ordered by decreasing number of biflows. Categorical labels are reported in round brackets.

Task	Classes
T_1 - *Encapsulation*	nonVPN (1), VPN (2)
T_2 - *Traffic Type*	VoIP (1), FileTransfer (2), P2P (3), Streaming (4), Chat (5), Email (6)
T_3 - *Application*	Skype (1), Torrent (2), Hangouts (3), VoipBuster (4), Facebook (5), FTPS (6), SCP (7), Email (8), YouTube (9), Vimeo (10), Spotify (11), Netflix (12), SFTP (13), Aim (14), ICQ (15)

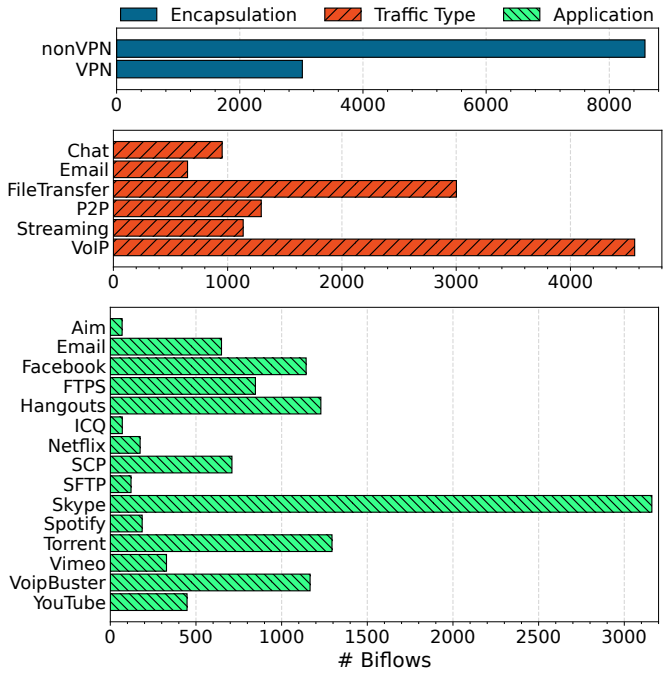


Figure 4: Number of per-class samples (i.e. biflows) for each ISCX VPN-nonVPN dataset task.

that is the ground-truth is associated to the whole PCAP trace and not to each TC object.

It includes *human-generated* traffic encompassing different traffic types—with information also on the related applications—collected through both regular sessions and sessions encapsulated over VPN. In view of this structure, we can associate a three-view label (i.e. encapsulation, traffic type, and application) to whatever segmentation of raw network traffic (i.e. to a generic TC object). Such three-view label corresponds to just as many TC *tasks* to be tackled. Table 3 lists the ISCX VPN-nonVPN classes associated to each task.

TC Object. While the finest atomic TC object is the single packet/datagram [16], the vast majority of papers tackling (multitask) ET classification reasonably work with either unidirectional or bidirectional flows (viz. biflows), attaining better performance with the latter TC object (see Sec. 2). A flow is defined as a stream of packets having the following 5-tuple in common: transport-layer protocol, source and destination IP addresses, source and destination ports [1]. The difference between bidirectional and unidirectional flows is whether they take the direction of packets or not into consideration in their definition, respectively. For the sake of completeness, we touch upon other meaningful TC objects, however, adopted in the context of *non-multitask encrypted TC* such as the IP packet [16], HTTP session [27], and service burst [8]. Hence, based on these considerations, we segment the raw traffic collected in the ISCX VPN-nonVPN dataset in *biflows*.

Pre-processing operations. We have found that $\approx$ 65% of biflows extracted from the raw traffic of the ISCX VPN-nonVPN dataset have *only one UDP packet* and destination (IP address, port) equal to (255.255.255.255, 10505). After further inspection, we have found that these packets are network broadcasts periodically sent—every two seconds by default—by BlueStacks, an Android emulator for PCs. Therefore, as opposed to the other works leveraging the ISCX VPN-nonVPN dataset [24, 20, 26], we have performed a pre-processing cleaning operation to remove this noisy traffic and thus make our results more meaningful. As a result, the final dataset contains 11.6k biflows whose distribution among the different classes for each task is shown in Fig. 4. We emphasize that, given the imbalanced per-class share of biflows, the dataset thus obtained constitutes a realistic and challenging evaluation benchmark for our DISTILLER classifier.

4.2. Description of Baselines Considered

We compare DISTILLER with several baselines proposed in most-related (recent) literature [15, 17, 20, 24, 25, 26] fed with the same input types (reported in brackets) whenever possible, and considering $N = 784$ bytes and $N_p = 32$ packets for PAY and HDR, respectively.[4] Also, each baseline is trained for 100 epochs corresponding to the total number of epochs summing up pre-training and fine-tuning of DISTILLER.

First of all, we highlight that we do not employ (*i*) biased input types (e.g., raw PCAP data and inputs comprising source/destination ports [18]) and (*ii*) manually-extracted features (e.g., PL/IAT stats). On the one hand, the former choice would prevent learning general (non-overfitted) patterns for discrimination. On the other hand, the latter choice would nullify a key benefit of DL: no need of human-expert intervention for extracting informative features. Accordingly, we have considered the following *baselines*, which have been carefully implemented and whose details are reported hereinafter:

- *Two multitask extensions* of state-of-the-art DL-based single-task traffic classifiers, namely the 1D-CNN (PAY) and the HYBRID 2D-CNN + LSTM (HDR) proposed in [15] and [17], respectively. Specifically, the original

[4] These choices are based on common values suggested in state-of-the-art works [15] and our past experience with (single-task) DL traffic classifiers [18], and constitute a satisfactory trade-off between complexity and performance.

architecture in [15] is made of two 1D convolutional layers (with 32 and 64 filters, respectively)—each followed by a 1D max-pooling—one dense layer (with 1024 nodes and ReLU activation), and terminated with one softmax layer. Conversely, the original architecture in [17] is a combination of two 2D convolutional layers (with 32 and 64 filters, respectively, and batch normalization), where the output tensor of the second convolutional layer is reshaped into a matrix fed as input to an LSTM (with 100 units). It is likewise terminated with a single softmax layer. We extend both these single-task architectures by replacing the last softmax with three separate softmax layers, one for each task.

- *Five native multitask DL architectures* proposed for TC, that is the `1D-CNN (HDR)`[5] proposed in [20] and two different (deep) `MLP (PAY/HDR)` architectures adopted in [25, 26]. For the last two baselines, we have considered two variants each, fed with either input types, since the original proposals had *handcrafted* PL/IAT stats as input.

- A *modified* version—we named `2D-CNN (PAY)`—of the *multitask* architecture originally presented in [24]. Its structure is made of two branches—the first shared by two binary learning tasks and the second related to a multi-class task—of 2D convolutional layers (with a number of filters ranging from 32 to 128), with the first two layers followed by a 2D max-pooling. Each branch is terminated with a dense layer (1024 nodes). Since the architecture is fed with biased inputs (i.e. raw PCAP data formatted as images) and characterized by an excessively ad-hoc structure, we adapted it to our problem by (*i*) feeding it with the unbiased `PAY` input and (*ii*) considering a different task mapping to the two CNN-based branches, namely one binary (T_1) and two multi-class learning tasks (T_2 and T_3) as opposed to two binary and one multi-class, respectively.

4.3. Implementation details

We leveraged the DL models provided by `Keras` (`https://keras.io`) Python API running on top of `TensorFlow 2` (`https://www.tensorflow.org/`) to implement and test the approaches described in this work. Also, data pre- and post-processing have been performed mainly by means of `numpy` (`https://numpy.org/`) and `pandas` (`https://pandas.pydata.org/`) libraries. Finally, the graphical data representation has been obtained using `matplotlib` (`https://matplotlib.org/`) and `seaborn` (`https://seaborn.pydata.org/`) libraries, along with a customized version of the `pySankey` (`https://pypi.org/project/pySankey/`) module.

All the experiments refer to the same hardware architecture, namely an OpenStack virtual machine with 16 vCPUs and 32 GB of RAM, and Ubuntu 16.04 (64 bit) operating system, running on a physical server with 2 × Intel(R) Xeon(R) E5-4610v2 CPUs @ 8 × 2.30 GHz and 64 GB of RAM. We highlight that all the execution times have been logged in the same load conditions (i.e. the DL classifier was the sole CPU-intensive running process).

[5] For this baseline, to be as pertinent as possible to the original proposal, we have used a subset of `HDR` encompassing signed PL (positive for upstream and negative for downstream) and IAT as input.

5. Experimental Evaluation

In the following analyses, the performance evaluation is based on a *stratified five-fold cross-validation*, representing a solid assessment setup as it keeps the sample ratio among classes for each fold. Since we are facing multiple TC tasks, the stratification is performed herein on the hardest task, i.e. T_3. Therefore, we report both the *mean* and the *standard deviation* of each performance measure as a result of the evaluation on the five different folds.

Our experimental evaluation is organized as follows. First, in Sec. 5.1, we perform an overall comparison of Distiller with the considered baselines. Then, in Sec. 5.2, we explore the performance dependence of our approach with respect to a varying weight-configuration $\{\lambda_1, \ldots, \lambda_V\}$ choice, namely a Pareto optimization. Further, Sec. 5.3 contains a fine-grained analysis of the soft-outputs of Distiller and the best baselines identified. Finally, in Sec. 5.4, knowledge transfer among the considered tasks is investigated with reference to (*a*) reject option adoption and (*b*) incoherence analysis.

5.1. Overall Comparison

Hereinafter, we first provide an overall performance comparison of Distiller with the baselines introduced in Sec. 2 and detailed in Sec. 4.2. To this end, Tab. 4 compares their *accuracy* (the percentage of samples properly classified) and macro (i.e. arithmetically-averaged over classes) *F-measure* on the three tasks. In detail, the latter is the harmonic mean of precision (the per-class fraction of decisions being correct) and recall (the class-conditional accuracy) accounting for both metrics at once.

Additionally, for the sake of a fair comparison, we have employed for all the classifiers a *uniform* weight configuration (i.e. $\lambda_1 = \lambda_2 = \lambda_3 = 0.33$ in Eqs. (1) and (2)). On the basis of the results attained with this configuration, we sort them (including Distiller) according to an *average performance ranking*, that is by decreasing average between accuracy and F-measure, over all the three tasks. Accordingly, we notice that Distiller *ranks first* (i.e. it is the overall-best-performing classifier) and shows *significant performance gains for all the metrics, for all the tasks*—ranging from +4.90% to +8.45%—with respect to the *overall-best-performing baseline* (ranking II). The latter corresponds to the `1D-CNN (PAY)` [15] classifier and is highlighted in blue in Tab. 4. The last row of Tab. 4 (highlighted in green) summarizes the performance gain of Distiller with respect to this overall-best-performing baseline. Finally, even considering the classification performance for each single task, Distiller *always outperforms* also the best-per-metric baseline (marked with ↑ in Tab. 4).

Table 4: Comparison of DISTILLER Accuracy, F-measure, and Run-Time Per-Epoch (RTPE) with state-of-the-art baselines. Results are in the format *avg. (± std.)* obtained over 5-folds. Rank is based on the average of all performance metrics on all the tasks. Last row shows DISTILLER Gain [%] on the overall-best-baseline (ranking II, highlighted in blue). Marked values are: (🏆) *overall* best classifier and (🡅) best *baseline*, for each metric.

Rank	Multitask Classifier	T_1 - Encapsulation		T_2 - Traffic Type		T_3 - Application		RTPE [s]
		Accuracy [%]	F-measure [%]	Accuracy [%]	F-measure [%]	Accuracy [%]	F-measure [%]	
I	DISTILLER	93.75 (± 0.73) 🏆	91.95 (± 0.67) 🏆	80.78 (± 0.95) 🏆	78.72 (± 1.05) 🏆	77.63 (± 0.66) 🏆	66.44 (± 1.76) 🏆	5.99 (± 0.13)
II	1D-CNN (PAY) [15]	87.47 (± 0.29)	83.50 (± 0.75)	73.14 (± 0.79) 🡅	71.14 (± 0.87) 🡅	72.73 (± 0.77) 🡅	61.35 (± 1.60) 🡅	13.83 (± 1.67)
III	2D-CNN (PAY) [24]	87.43 (± 0.66)	83.51 (± 0.46)	71.86 (± 0.95)	69.77 (± 0.96)	71.45 (± 1.13)	59.29 (± 2.06)	40.94 (± 3.57)
IV	MLP (PAY) [26]	86.95 (± 0.65)	82.38 (± 1.12)	70.67 (± 0.64)	68.14 (± 0.72)	69.50 (± 0.97)	56.44 (± 2.45)	2.58 (± 0.36)
V	MLP (HDR) [26]	88.71 (± 0.37) 🡅	84.94 (± 0.48) 🡅	68.57 (± 0.51)	65.87 (± 0.55)	63.97 (± 1.02)	51.14 (± 1.28)	2.24 (± 0.17)
VI	MLP (PAY) [25]	85.28 (± 0.66)	81.16 (± 0.55)	67.60 (± 1.10)	64.68 (± 1.36)	65.39 (± 1.06)	51.78 (± 1.31)	0.75 (± 0.10) 🡅🏆
VII	HYBRID (HDR) [17]	87.11 (± 1.88)	82.82 (± 1.28)	66.00 (± 2.61)	62.40 (± 4.34)	60.17 (± 3.70)	50.49 (± 2.40)	3.34 (± 0.38)
VIII	MLP (HDR) [25]	86.53 (± 0.65)	81.55 (± 1.03)	62.86 (± 0.92)	59.43 (± 1.40)	59.34 (± 0.88)	44.20 (± 1.22)	0.79 (± 0.02)
IX	1D-CNN (HDR) [20]	82.95 (± 1.33)	76.24 (± 2.55)	59.09 (± 3.34)	54.75 (± 2.24)	56.54 (± 2.65)	40.87 (± 2.13)	1.70 (± 0.02)
	DISTILLER Gain	**+ 6.28 (± 0.80)**	**+ 8.45 (± 1.13)**	**+ 7.65 (± 0.20)**	**+ 7.58 (± 0.95)**	**+ 4.90 (± 0.60)**	**+ 5.09 (± 1.17)**	**- 7.84 (± 1.67)**

Table 5: Run-Time Per-Epoch (RTPE in seconds) and Number of Trainable Parameters (TP in millions) of DISTILLER and state-of-the-art baselines. The RTPE is in the format *avg. (± std.)* obtained over 5-folds. The classifiers are sorted based on increasing RTPE. The columns "C" and "R" denote the usage of convolutional and recurrent layers, respectively. The last column "Rank" refers to the ranking based on the average of all TC performance metrics defined in Tab. 4.

Multitask Classifier	RTPE [s]	TP [M]	C	R	Rank
MLP (PAY) [25]	0.75 (± 0.10)	0.03	○	○	VI
MLP (HDR) [25]	0.79 (± 0.02)	0.01	○	○	VIII
1D-CNN (HDR) [20]	1.70 (± 0.02)	0.20	●	○	IX
MLP (HDR) [26]	2.24 (± 0.17)	1.55	○	○	V
MLP (PAY) [26]	2.58 (± 0.36)	1.71	○	○	IV
HYBRID (HDR) [17]	3.34 (± 0.38)	0.74	●	●	VII
DISTILLER	5.99 (± 0.13)	0.97	●	●	I
1D-CNN (PAY) [15]	13.83 (± 1.67)	5.84	●	○	II
2D-CNN (PAY) [24]	40.94 (± 3.57)	22.65	●	○	III

Moreover, we also investigate the computational complexity of the considered multitask DL-architectures evaluating their *training phase runtime*. This is a fundamental aspect in modern network TC, where frequent re-training is required, due to the aging of training data as a result of the fast-paced evolution in network usage. Since training is performed on multiple epochs, we report this outcome in a terse way, showing the Run-Time Per-Epoch (RTPE) in the last column of Tab. 4.

From the above viewpoint, DISTILLER attains a *reduction in the training time* of −7.84 seconds per-epoch, thus exhibiting lower empirical computational complexity when compared with the overall-best-performing baseline. Accordingly, the whole training time is *more than halved*, as a result of the RTPE cut on all the considered training epochs. Indeed, the training time equals the RTPE multiplied by the total number of epochs, which is set to the same value for all the considered architectures ($N_{\text{epochs}} = 100$, cf. Sec. 3.3 and 4.2). It is worth noting that the baseline with the shortest RTPE ranks VI in terms of the overall TC performance metrics, with losses of up to 14.66% of F-measure compared to DISTILLER on task T_3.

The complexity comparison among the considered multitask DL traffic classifiers is deepened in Tab. 5, where the RTPE of each architecture is paired with the corresponding *number of trainable parameters* TP, related to the theoretical complexity of the training phase. Indeed, we highlight that an exact $O(\cdot)$ expression cannot be easily drawn for general DL architectures [40]. However, the complexity associated to the whole training process can be generically expressed as $O(N_{\text{epochs}} \times M \times C_{\text{backprop}}(\text{TP}))$, where N_{epochs} denotes the number of epochs, M the number of training samples, and the term $C_{\text{backprop}}(\text{TP})$ refers to evaluating the contribution of a single sample to the gradient via the well-known backpropagation algorithm. The latter term is clearly a function of the specific DL architecture (e.g., presence of convolutional/recurrent layers) and leads to cumbersome computations, even in case of (simpler) MLPs [41]. Still, in general, such function retains a monotonic behavior with the number of trainable parameters TP. Accordingly, TP represents the main relevant factor for comparing the considered DL traffic classifiers, since they are trained on the same dataset (same M) and for the same number of iterations (same N_{epochs}). For completeness, in the aforementioned table we also highlight whether the training parameters are associated with more involved layers than dense ones, such as convolutional (column "C") and recurrent (column "R") layers.

Specifically, a direct comparison between the number of trainable parameters of DISTILLER (i.e. 0.97M) and those of 1D-CNN (PAY) [15] (i.e. 5.84M) reveals that the RTPE strongly depends on the TP value, with DISTILLER having *six times fewer parameters* than the overall-best-performing baseline. Such a general trend can be observed for almost all the considered multitask DL traffic classifiers. Still, the corresponding ranking between the RTPE and the TP value is influenced by the presence of convolutional layers, recurrent layers, or both. Accordingly, multitask DL architectures with a lower TP value (e.g., our DISTILLER) may incur in higher RTPE with respect to baselines based on a higher TP, because of the presence of convolutional and recurrent operations.

5.2. Pareto Optimization

Multitask learning is a peculiar application of *multi-objective optimization* (a.k.a. Pareto optimization) which consists in the mathematical optimization (i.e. maximization or minimization) of a given variable vector with respect to more than one objective function to be optimized simultaneously. Accordingly, the loss functions employed in Eqs. (1) and (2)—for pre-training and fine-tuning, respectively—are the result of the so-called *scalarization* of the original multi-objective problem, with the weights λ_i assigning a given (a-priori) preference to the i^{th} objective (viz. task) in the tackled optimization problem.

In this regard, we perform a *sensitivity performance analysis*, focusing on the DISTILLER classifier and the overall-best-performing baseline (i.e. `1D-CNN (PAY)` [15], cf. Tab. 4) whose outcomes are shown at the top and bottom rows of Fig. 5, respectively. Such sensitivity analysis is obtained by varying the weights $\{\lambda_1, \lambda_2, \lambda_3\}$ within both losses reported in Eqs. (1) and (2) during the (two-step) training phase. The same analysis is performed for the (one-step) training phase of the baseline. Specifically, Fig. 5 illustrates the TC performance in terms of *F-measure* with respect to $\{\lambda_1, \lambda_2\}$ only, as $\lambda_3 = 1 - (\lambda_1 + \lambda_2)$ is a dependent variable.

Results highlight quite smooth performance trends on the three tasks with respect to the weights, with the only exception of cases associated to a zero-preference level for a given task (i.e. $\lambda_1 = 0$, $\lambda_2 = 0$ and $\lambda_1 + \lambda_2 = 1$ for T_1, T_2, and T_3, respectively). Indeed, a zero preference implies that the architecture is not explicitly trained to solve the corresponding task.

Comparing DISTILLER with `1D-CNN (PAY)` [15], Fig. 5 highlights similar trends for both traffic classifiers. More specifically, `1D-CNN (PAY)` [15] has slightly higher performance only in correspondence of some zero-preference configurations for the specific task (also presenting the worst performance, i.e. orange and red cells with dashed contour in Fig. 5). As an example, considering the T_2-zero-preference weight configuration $\{\lambda_1, \lambda_2, \lambda_3\} = \{0.6, 0, 0.4\}$, `1D-CNN (PAY)` [15] attains 16.29% F-measure (see Fig. 5e) compared to DISTILLER reaching 11.81% with the same weights (see Fig. 5b). Similar outcomes can be highlighted also for the other two tasks. Summarizing, DISTILLER can seldom attain lower performance for a task, but only if the corresponding task contribution is ignored during training (zero-preference level). Differently, DISTILLER always outperforms `1D-CNN (PAY)` [15] for all other (more sensible) configurations of weights.

Focusing on DISTILLER performance, the uniform weight allocation (i.e. $\lambda_1 = \lambda_2 = \lambda_3 = 0.33$) considered in Sec. 5.1 does not result in the best solution for each of the three tasks: ad-hoc weight configurations should be employed to maximize per-task F-measure. However, a weight configuration optimal for a given task, in general implies a loss in performance for the remaining two, and for the overall performance as well. For instance, the optimal weight configuration for the hardest task T_3 is $\{\lambda_1, \lambda_2, \lambda_3\} = \{0.4, 0, 0.6\}$, obtaining an F-measure for T_3 of 66.17%, but resulting in a zero-preference level for the task T_2 (dropping to 10.58% F-measure). Even when this is not the case—for instance, the optimal configuration for T_2 implies $\{\lambda_1, \lambda_2, \lambda_3\} = \{0.2, 0.6, 0.2\}$—a uniform weight allocation is still able to globally outperform such per-task optimum. Specifically, with the aforementioned T_2-optimized weight configuration, DISTILLER can obtain 91.97% and 63.30% F-measure on T_1 and T_3, respectively, against 91.95% and 66.44% attained with the uniform weight allocation.

Summarizing, our DISTILLER *outperforms* the overall-best baseline according to all the three tasks. Additionally, the evaluation over the weight surface $\{\lambda_1, \lambda_2, \lambda_3\}$ generally highlights that learning multiple tasks *simultaneously* provides *improved TC performance* with respect to single-task learning. Equally important, uniform task-weight allocation represents a *reasonable* trade-off in terms of TC performance and simplicity in the choice. In view of these considerations, hereinafter we will employ a uniform weight allocation, unless explicitly stated otherwise.

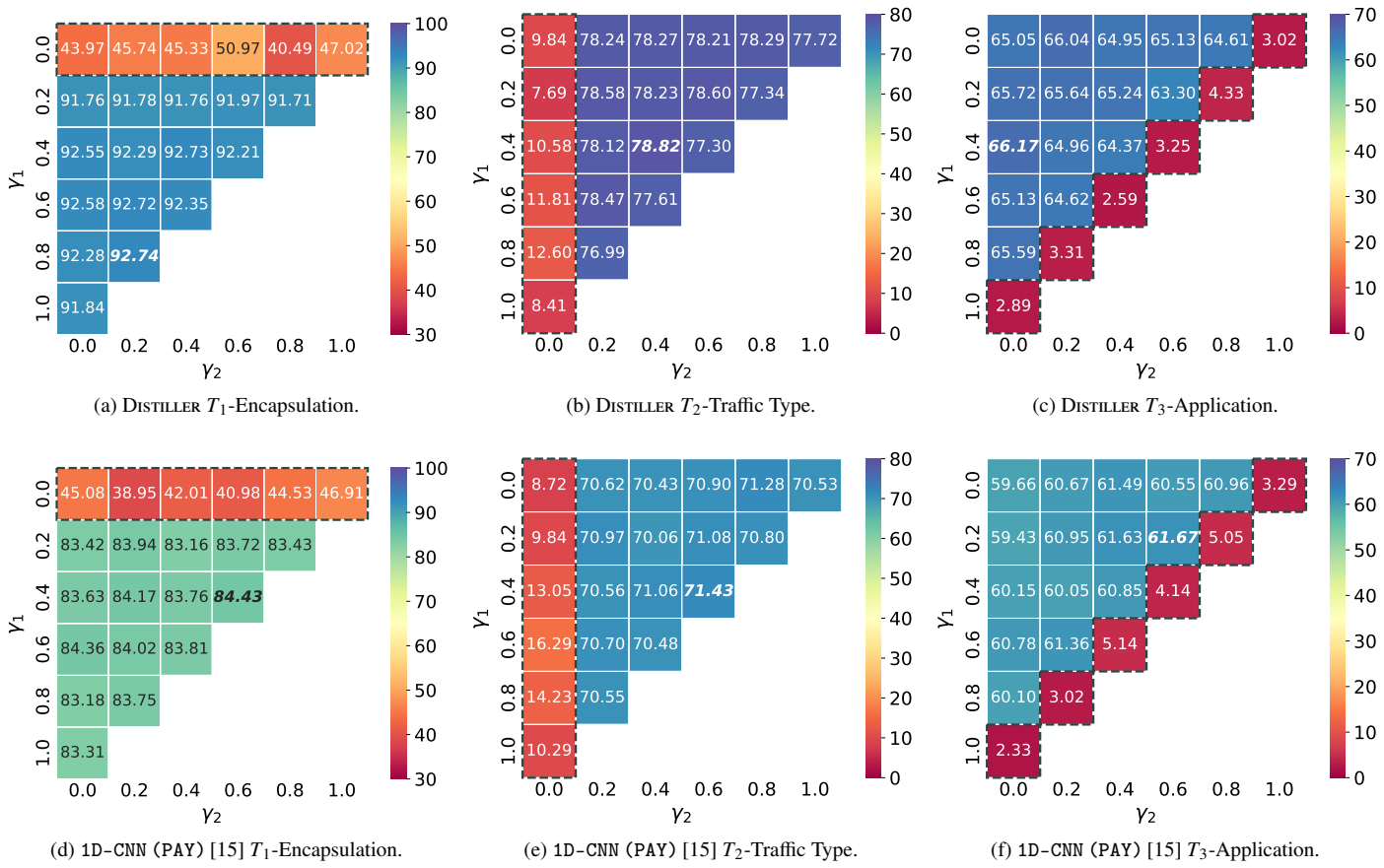
5.3. Fine Grained Analysis of (Soft-)Outputs

Delving into the performance of DL-based multitask TC, in Fig. 6, we report the *Top-K accuracy* of DISTILLER and all the considered baseline traffic classifiers. In detail, the Top-K accuracy defines a correct classification event if the true class is within the top K predicted labels ($K < L_v$ is a free parameter). Of course, $K = 1$ coincides with the standard accuracy; hence, such metric has an informative interpretation only for tasks corresponding to *multi-class problems* ($L_v > 2$). Accordingly, in what follows we focus only on T_2 ($L_2 = 6$ classes) and T_3 ($L_3 = 15$ classes), since $L_1 = 2$ classes for T_1 (i.e. it is a binary TC task).

Given the above definition of Top-K accuracy, the higher the value of K, the better the accuracy achieved (by construction). Specifically, this metric allows to investigate the soft-output of a (multi-class) DL classifier and its fine-grained behavior. From the inspection of the histograms depicted in Fig. 6, we can notice that DISTILLER *outperforms* all the baselines taken into account for the considered (reasonable) range of K (in our analysis, $K \in \{1, \ldots, 5\}$). In detail, considering the first two ($K = 2$) most confident soft-outputs, our DISTILLER classifier is able to hit $> 90\%$ accuracy for task T_2 and almost 90% for task T_3, with an improvement of +3.63% (resp. +2.85%) over the best baseline `MLP (HDR) [26]` (resp. `1D-CNN (PAY)` [15]) for the specific task T_2 (resp. T_3). Moreover, Figs. 6a and 6b reveal that, even for higher values of K, there is a performance difference between DISTILLER and all the baselines for both the task T_2 and T_3. Especially for the latter (more complex) task, DISTILLER is able to attain the most significant performance enhancement, reaching up to $\approx 97\%$ accuracy with $K = 5$.

To deepen the analysis of fine-grained behavior of multitask traffic classifiers, Fig. 7 depicts the *Sankey diagrams* corresponding to tasks T_1 and T_2.[6] Specifically, these diagrams show the per-class ratio of samples correctly and incorrectly

[6] Given the higher number of classes ($L_3 = 15$) for the task T_3, the readability of the related Sankey diagram is severely impaired and consequently it is replaced by an equivalent *confusion matrix* (Fig. 8).



(a) DISTILLER T_1-Encapsulation. (b) DISTILLER T_2-Traffic Type. (c) DISTILLER T_3-Application.

(d) 1D-CNN (PAY) [15] T_1-Encapsulation. (e) 1D-CNN (PAY) [15] T_2-Traffic Type. (f) 1D-CNN (PAY) [15] T_3-Application.

Figure 5: F-measure of DISTILLER classifier (a-c) and best performing baseline (d-f) for different combinations of the weights associated to the three tasks. The **best per-task weight-configuration** is highlighted in boldface. The zero-preference levels are highlighted in dashed boxes; the performance is minimum for the task T_v when the respective λ_v is set to 0 (note that $\lambda_3 = 0$ on the anti-diagonal: $\lambda_1 + \lambda_2 = 1$).

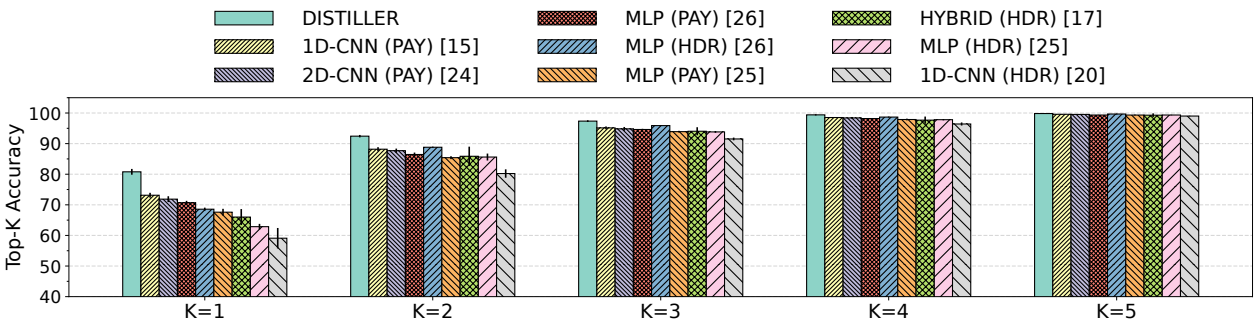
classified, with a clear highlight on error patterns (quantitatively summarized by the recall values reported in square brackets for each class). By comparing the DISTILLER diagrams (Figs. 7a and 7b) with those of the overall-best-performing baseline 1D-CNN (PAY) [15] (Figs. 7c and 7d), we can observe a *substantial reduction of misclassified biflows* for all the classes and for both tasks. Particularly, even when the error patterns are less severe (e.g., VoIP class), the recall improvement for T_2 is always higher than 5%. This results in a beneficial *reduction of systematic error patterns* (e.g., VoIP traffic misclassified as FileTransfer). Besides general improvements of recall, high impact gains are obtained for specific classes, the highest ones for VPN (task T_1) and Email (task T_2), with +15.02% and +12.95% recall, respectively.

To complement the investigation of fine-grained behavior, Fig. 8 shows the *confusion matrices* of DISTILLER and 1D-CNN (PAY) [15] for the task T_3. Indeed, confusion matrices represent a complementary means to highlight misclassification patterns, with a higher concentration toward the diagonal—where predicted classes equal the actual ones—implying better performance of the generic classifier. Results highlight a general improvement of the confusion matrix of DISTILLER against the (best) baseline classifier also for the *hardest* learning task. Indeed, despite both classifiers show qualitatively-similar error patterns, DISTILLER presents less imbalance toward the Skype class (+3.41% in terms of recall). Overall, there is a recall improvement *for all the T_3 classes* and the highest gain is attained for FTPS (class "6") and YouTube (class "9"), reaching +14.40% and +10.02%, respectively.

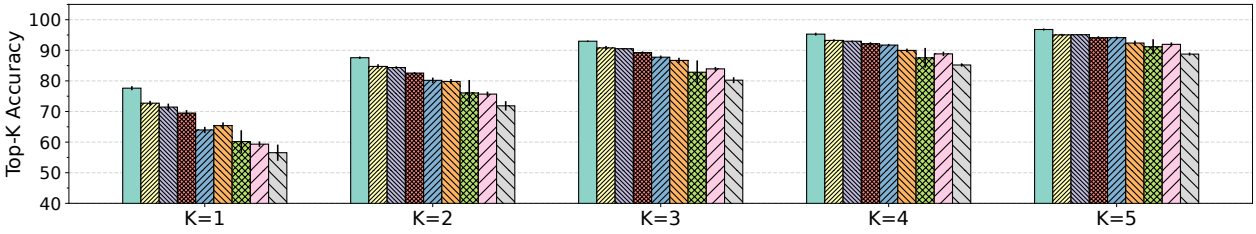
Finally, we complete the fine-grained investigation with a *calibration analysis*, as reported in Fig. 9. In fact, the latter aims to assess whether the class-probability estimates (the soft-output values) match the actual-class posterior probabilities (and thus their value can be exploited for more than just a ranking, as in *Top-K accuracy*).

In detail, we depict the *reliability diagrams*, that show the accuracy as a function of the confidence and are obtained by partitioning the predictions into equally-spaced bins and computing the accuracy of each bin. Confidence values range in $[1/L_v, 1]$, where L_v is the number of classes of TC task v. Accordingly, the starting-point $1/L_v$ of the confidence interval is pointed out with a dashed blue line in Fig. 9.

Indeed, a *miscalibrated* classifier (on a given task) returns excessively optimistic ("Over" in Fig. 9) or pessimistic ("Under" in Fig. 9) confidence outputs associated to its decisions. Conversely, a *perfectly-calibrated* classifier entails a reliability



(a) T_2-Traffic Type.



(b) T_3-Application.

Figure 6: Top-K accuracy of DISTILLER and baseline traffic classifiers. Results refer to T_2 (a) and T_3 (b). Error bars report average ± standard deviation.

diagram corresponding to the identity function, that is, a classifier having 80% confidence leads to 80% accuracy. To obtain a concise metric of the deviation from perfect calibration, we also integrate the above diagrams with the *Expected Calibration Error* (ECE). The latter metric is defined as the weighted mean—based on the number of samples and evaluated over all the bins—of the difference between accuracy and confidence, and provides a synthetic comparison metric for classifiers.

Once more, we contrast our DISTILLER classifier (Figs. 9a, 9b, and 9c) with the overall-best baseline (Figs. 9d, 9e, and 9f) over all the three TC tasks. We recall that, since T_1 is a binary TC problem, the range of the corresponding reliability diagrams is shorter since the class prediction probability is always > 0.5. Conversely, the lower limit of the same range equals ≈ 0.167 and ≈ 0.067 for T_2 and T_3, respectively.

Interestingly, both classifiers exhibit a miscalibration that tends to be *over-confident* (viz. optimistic) in predictions for all but one bin. Indeed, for the predictions on T_2 whose confidence falls within [0.2, 0.3[, DISTILLER exhibits a more under-confident (pessimistic) behavior than `1D-CNN (PAY)` [15]—also having an under-confident outcome in the same range. This effect, however, is associated with a very limited number of classified samples characterized by poor confidence values (i.e. corresponding to the lowest-admissible confidence-bin for T_2) and, consequently, with the least discriminative power.

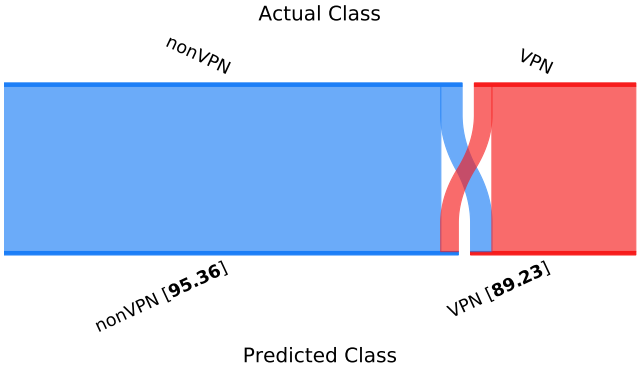
In general, in each bin—except for predictions in [0.2, 0.3[for T_2—the confidence is higher than the accuracy for both classifiers. This effect can be attributed to a slight overfitting phenomenon and it is a distinctive characteristic of DL architectures, although the calibration analysis shows that our multimodal-multitask DISTILLER *significantly mitigates* it. Indeed, overall DISTILLER *is considerably better calibrated* than `1D-CNN (PAY)` [15], showing an ECE *reduction* of 1.75 times on T_3, 2.03 times on T_2, and 3.64 times on T_1.

Summarizing, the multimodal nature of DISTILLER is able to provide *a more consistent* soft-output TC behavior on the two hardest tasks (i.e. T_2 and T_3) with respect to all the considered (single-modal) baselines, namely the true class belongs to the set of most probable ones even when it is not the highest. Moreover, our proposal is able to achieve a systematic error pattern reduction with respect to the overall-best baseline, as shown by Sankey diagrams (on T_1 and T_2) and confusion matrices (on T_3). Finally, the calibration analysis highlights that our DISTILLER (due to the successful capitalization of multimodality) significantly mitigates the overfitting phenomenon—leading to the usual over-confident (viz. optimistic) behavior of DL architectures—as shown by the ECE halving on each task compared to the overall-best baseline.

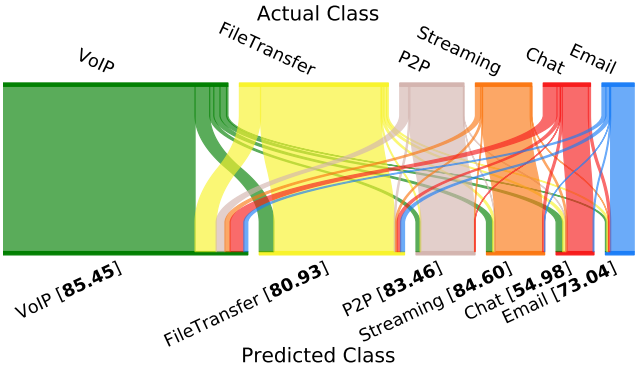
5.4. Transfer of Knowledge among Tasks

Hereinafter, we investigate the knowledge transfer among the tasks firstly testing the multitask classifiers when a *reject option* (viz. a censoring policy of "unsure" outcomes) is adopted for *all the tasks*, that is the classification is performed only if the highest class prediction probability associated to the v^{th} TC task exceeds its related threshold γ_v, thus emitting a confident verdict on the corresponding task.

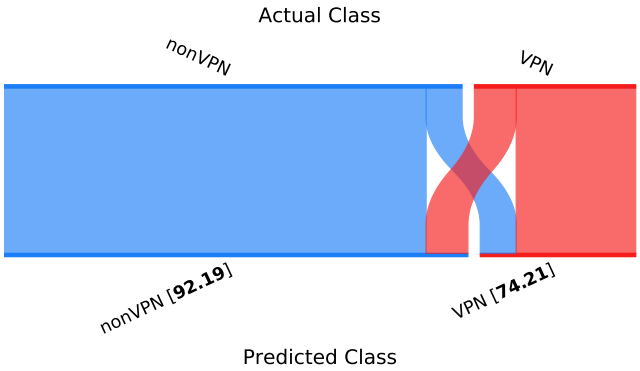
Its adoption has been justified in the context of encrypted traffic, e.g. for the TC of mobile apps [8] and anonymity tools [42]. Specifically, given the high number of (bi)flows commonly generated by network applications and services, there is an excellent chance of identifying these latter only considering the more characteristic biflows, namely those corresponding to a classification confidence above γ_v. Hence, a



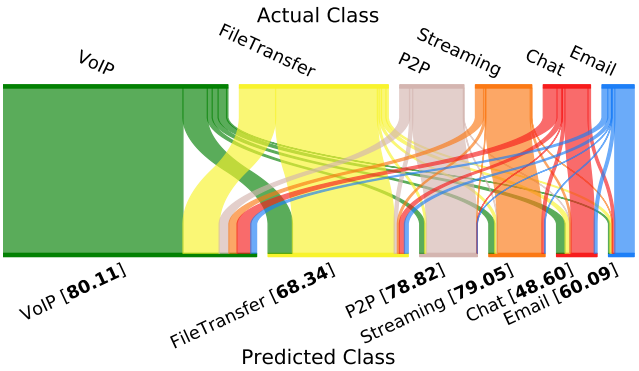
(a) DISTILLER T_1-Encapsulation.



(b) DISTILLER T_2-Traffic Type.

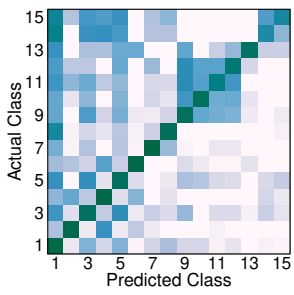


(c) `1D-CNN (PAY)` [15] T_1-Encapsulation.

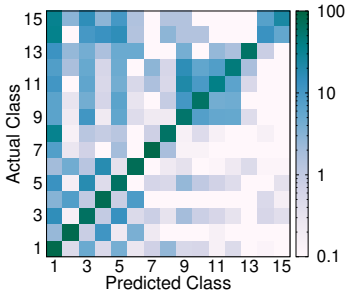


(d) `1D-CNN (PAY)` [15] T_2-Traffic Type.

Figure 7: Sankey diagrams of the DISTILLER classifier (a, b) and the overall-best-baseline (c, d) for the T_1-Encapsulation (a, c) and T_2-Traffic Type (b, d) TC tasks. The classes are ordered by decreasing number of biflows. In square brackets **per-class recalls [%]** are reported.



(a) DISTILLER.



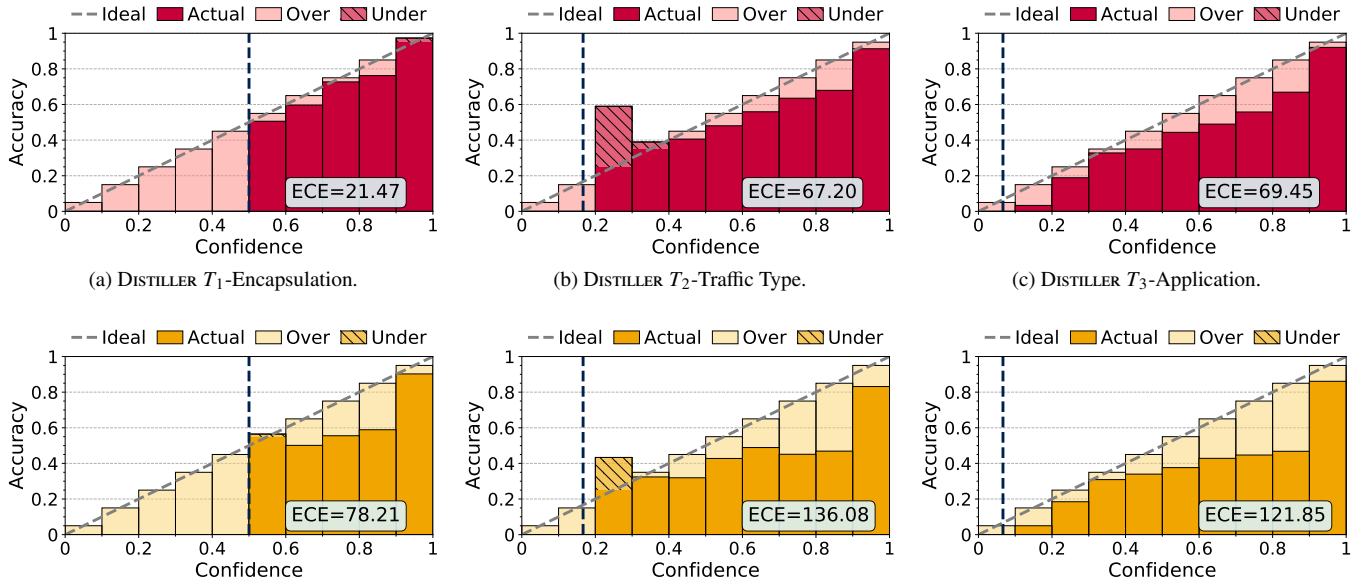
(b) `1D-CNN (PAY)` [15].

Figure 8: Confusion matrices of the DISTILLER classifier (a) and the overall-best baseline (b) for the T_3-Application TC task. Note that the log scale is used to evidence small errors. The classes are ordered by decreasing number of biflows. Categorical class-labels are reported in Tab. 3.

generic multitask classifier can improve its TC performance on the v^{th} TC task with γ_v at the price of a reduced percentage of classified biflows, namely the classified ratio (CR). Indeed, the decisions on the other biflows that are under the γ_v are *censored*. Thus, tuning γ_v enables a fine-grained control of the classifier and further (useful) flexibility to encrypted TC [8] with also the ability of a differentiated tuning for each v^{th} task to tackle (i.e. using different γ_v).

To this end, Fig. 10 shows the F-measure and the CR when varying the censoring threshold γ_v for each of the three tasks. The present analysis is carried out for both DISTILLER and the `1D-CNN (PAY)` [15] baseline. Hence, by looking at the plots row-wise (resp. column-wise) the performance depicts the effects of reject option on the two approaches (resp. the three tasks). Within each plot, the F-measure on the v^{th} task vs. the censoring threshold γ_v is depicted via bigger markers. For completeness, to also truly exploit the related nature of the multiple TC tasks considered (viz. the knowledge transfer among them), in each figure we also report the F-measure values corresponding to the other two tasks and resulting from censoring the samples based on the considered γ_v (e.g., in Figs. 10a and 10d the F-measure values for the tasks T_2 and T_3 are obtained by censoring the biflows based on the predicted probability of T_1 against γ_1).

Results highlight the gain of DISTILLER over the baseline even when a reject option is applied. For instance, referring to the task T_2, $\approx$ 85% F-measure can be obtained by rejecting only 15% of the overall biflows with our DISTILLER classifier, whereas `1D-CNN (PAY)` [15] achieves the same F-measure value by rejecting more than the double of samples (i.e. $>$ 30%). Similarly, when considering T_3 (the hardest task), $\approx$ 75% F-measure can be obtained by rejecting 25% of the overall bi-



(a) DISTILLER T_1-Encapsulation. (b) DISTILLER T_2-Traffic Type. (c) DISTILLER T_3-Application.

(d) 1D-CNN (PAY) [15] T_1-Encapsulation. (e) 1D-CNN (PAY) [15] T_2-Traffic Type. (f) 1D-CNN (PAY) [15] T_3-Application.

Figure 9: Reliability diagrams of DISTILLER (a,c,e) and overall-best baseline (b,d,f) for the three TC tasks. Confidence is divided in 10 bins, and is $\geq 1/L_v$ (vertical dashed line), with L_v being the number of classes for the v^{th} task. Over and under gap represent an over-confident (optimistic) and under-confident (pessimistic) miscalibration pattern, respectively. To ease the comparison, the non-percentage accuracy (i.e. within [0, 1]) is shown. Concise *ECE* metric is reported for each case.

flows with DISTILLER, whereas the baseline can achieve a comparable F-measure score by rejecting 35% of biflows. Finally, by looking at the F-measure obtained by censoring on different tasks, it is evident that the tasks T_2 and T_3 *seem more related*. Indeed, a variation of γ_2 (resp. γ_3) significantly reflects on the F-measure performance of the task T_3 (resp. T_2). Such effect is quite general as it applies to both our DISTILLER and the baseline: hence, we can infer that these tasks share common features for their inference process.

Finally, to complement our investigation on transfer of knowledge among tasks, we perform an *incoherence analysis* of our DISTILLER and the best-performing baseline. As described in Sun et al. [25], the *incoherence measure*[7] is defined as follows:

$$\text{incoherence}(v, v_s) \triangleq 100 \cdot \frac{\sum_{\ell_v=1}^{L_v} \sum_{n \in \mathcal{N}_{\ell_v}} d(\boldsymbol{c}^{v_s}(n), \bar{\boldsymbol{c}}_{\ell_v}^{v_s})}{\sum_{\ell_v=1}^{L} \sum_{n \in \mathcal{N}_{\ell_v}} \left\{ \sum_{\ell'_v \neq \ell_v} d(\boldsymbol{c}^{v_s}(n), \bar{\boldsymbol{c}}_{\ell'_v}^{v_s}) \right\}} \tag{3}$$

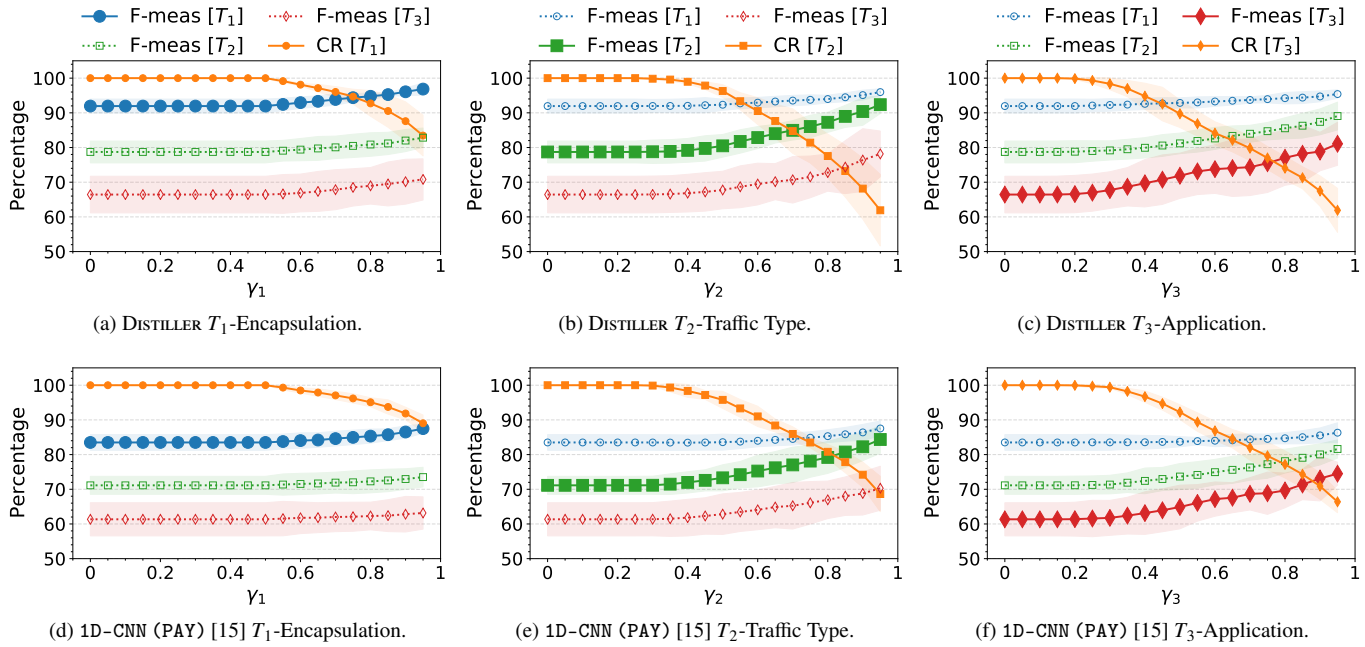
where $d(\cdot, \cdot)$ denotes the Euclidean distance and $\mathcal{N}_{\ell_v}$ represents the set of test samples whose true label for the v^{th} task is ℓ_v. Furthermore, $\boldsymbol{c}^{v_s}(n)$ is the soft-output vector associated to the v_s^{th} task for the n^{th} test sample, while $\bar{\boldsymbol{c}}_{\ell_v}^{v_s}$ is the average of all the soft-output vectors associated with the v_s^{th} task whose true label on the v^{th} task is ℓ_v. The above measure can be interpreted as a metric of *incoherence* among the soft-output vectors associated to the same label: the closer the vector is to the centroid of its respective true label, the lower the incoherence. Hence, *the lower the incoherence, the better* the representation learned (with implications on explainability [25]).

Table 6: incoherence analysis of DISTILLER and overall-best-performing baseline. incoherence for the multitask implementation is reported in *MT* column, while Single-Task evaluation is reported in columns ST_v. incoherence evaluated in the same space the task is defined on (i.e. $v = v_s$) is highlighted with **green**. Marked values are: overall best incoherence (🏆) and per-task best incoherence (🡅).

	incoherence(v, v)	**incoherence**(v, v_s)		
	MT	**ST$_1$ [v$_s$ = 1]**	**ST$_2$ [v$_s$ = 2]**	**ST$_3$ [v$_s$ = 3]**
	DISTILLER 🏆			
v = 1	**13.82 (± 1.22)** 🡅	**13.99 (± 1.80)**	97.85 (± 0.47)	96.58 (± 0.67)
v = 2	**6.82 (± 0.27)** 🡅	18.93 (± 0.05)	**7.16 (± 0.14)**	13.84 (± 0.16)
v = 3	**2.52 (± 0.06)** 🡅	6.44 (± 0.05)	3.18 (± 0.06)	**2.56 (± 0.06)**
	1D-CNN (PAY) [15]			
v = 1	**28.41 (± 0.75)**	**27.91 (± 2.32)** 🡅	98.75 (± 0.47)	97.13 (± 0.69)
v = 2	**9.14 (± 0.21)**	18.95 (± 0.19)	**9.05 (± 0.28)** 🡅	14.65 (± 0.24)
v = 3	**3.08 (± 0.07)** 🡅	6.40 (± 0.16)	3.67 (± 0.09)	**3.13 (± 0.05)**

Based on the above definition, it is apparent that incoherence(v, v_s) can be evaluated on the same space the task is defined (i.e. $v = v_s$) or on a different vector space (i.e. $v \neq v_s$). In the latter case, the classification output distance is taken from the centroids of a different set of labels (with respect to another task v_s): a small incoherence means that the two tasks share a significant common representation. Indeed, for the multitask architectures there is a soft-output vector corresponding to each task and then $v = v_s$ for every task (all tasks are considered,

[7] In Sun et al. [25] the metric is named *perplexity*, but we avoid this term as it is widely adopted for a different concept in ML. Compared with that definition, we normalize it to 100 instead of 1 for the sake of results readability.



(a) DISTILLER T_1-Encapsulation. (b) DISTILLER T_2-Traffic Type. (c) DISTILLER T_3-Application.

(d) 1D-CNN (PAY) [15] T_1-Encapsulation. (e) 1D-CNN (PAY) [15] T_2-Traffic Type. (f) 1D-CNN (PAY) [15] T_3-Application.

Figure 10: F-measure and ratio of classified samples (CR) [%] vs. censoring threshold $\boldsymbol{\gamma}$ of the DISTILLER classifier (a-c) and the overall-best-performing baseline (d-f). The F-measure on the v^{th} task vs. the censoring threshold γ_v is depicted via filled and bigger markers. Differently, the corresponding F-measure score on tasks different from v^{th}, as a function of γ_v, is highlighted with void and smaller markers.

there is no *other* task). Differently, this *does not apply to single-task architectures*. In this case, the generality of the above definition (Eq. (3)) allows also to analyze the incoherence properties of single-task classifiers on the other tasks, so as to consider them as baselines to assess the incoherence gain (the lower the better) provided by the multitask version. To this aim, both a multitask instance of DISTILLER and 1D-CNN (PAY) [15] is considered, as well as their respective single-task versions. These are obtained by setting a zero-preference value[8] for all but the considered task (e.g., $\{\lambda_1, \lambda_2, \lambda_3\} = \{1, 0, 0\}$ to obtain the single-task ST_1 instance). Accordingly, Tab. 6 reports the results of the incoherence analysis for both multitask and single-task variants of DISTILLER and 1D-CNN (PAY) [15]. From the results, it is apparent that the incoherence is lower when the metric is evaluated on the soft-output space that matches to the task considered (i.e. $v = v_s$ highlighted in green). While this is always the case for multitask architectures (MT in Tab. 6), for single-task architectures (ST in Tab. 6) this is only achieved when the incoherence can be evaluated s.t. $v_s = v$ for a given task v, since *the other tasks are not considered in the training phase*. Secondly, while DISTILLER *always* ensures lower incoherence with respect to the corresponding value of its single task counterpart, such claim is not always true for the 1D-CNN (PAY) [15] baseline. For instance, on both T_1 and T_2 the baseline is not able to improve (viz. reduce) the incoherence with respect to the single-task counterpart solely trained for T_1 and T_2, respectively. This confirms the effective capitalization by DISTILLER of the regularization (overfitting-avoidance) properties granted by multitask learning, as opposed to (separate) single-task counterparts. Finally, by comparing the incoherence values of the DISTILLER and baseline classifier, it is apparent that our proposal achieves *lower incoherence* on *all the three tasks*. Notably, for T_1 the incoherence reduction is more than two-fold (13.82 of DISTILLER vs. 28.41 of 1D-CNN (PAY) [15]): this can be explained by the presence of only two groups (i.e. two classes, since it is a binary task), emphasizing the improvement brought by the DISTILLER architecture.

In summary, the above analysis highlights the further improvement of TC effectiveness (in terms of F-measure) of DISTILLER on all the three tasks, achievable by means of a per-task reject option. Furthermore, by looking at the incoherence measure, we observe that our DISTILLER approach (as opposed to the overall-best baseline) ensures a better separation among different classes with respect to its single-task counterparts. This confirms the effective capitalization of multitask learning by our proposal, obtained via the fruitful transfer of knowledge among the different tasks.

6. Conclusions and Future Directions

In this paper, we tackled multipurpose ET classification via a general multimodal multitask DL architecture (termed DISTILLER). Our aim was to provide an effective design basis for sophisticated network management requiring the solution of different network visibility tasks. The proposed architecture was based on a suitably-defined (two-step) training procedure which enforces information distillation from each modality and reaps the regularization gains of multitask learning.

[8] See Sec. 5.2 on Pareto analysis.

Our evaluation was performed on a dataset of human-generated traffic (ISCX VPN-nonVPN) labeled according to three different TC tasks (i.e. encapsulation, traffic type, and application recognition). For the above TC scenario, we defined a peculiar *instance* of DISTILLER, characterized by two complementary input modalities (header vs. payload information) and providing the simultaneous solution to the aforementioned three TC tasks.

First, results from the *overall comparison* showed performance gains by DISTILLER over state-of-the-art multitask architectures up to +8.45% (on T_1), +7.58% (on T_2), and +5.09% (on T_3) in terms of F-measure with respect to the overall-best baseline. Equally important, DISTILLER exhibited very manageable training complexity and lower computational burden than the overall-best-performing multitask baseline, namely −7.84 s in terms of RTPE, whose total impact is magnified by two orders of magnitude according to the number of training epochs. Therefore, a properly-structured multimodal-multitask architecture was shown to be effective in both raising TC performance and lowering computational complexity.

Second, the *fine-grained analysis* of the DISTILLER outputs highlighted also the structural gain of our proposal in terms of (*a*) more consistent soft-output behavior (e.g., > 90% Top-2 accuracy, with a +3.63% gain over the best baseline), (*b*) systematic error pattern reduction, and (*c*) improved calibration (e.g., the ECE is at least halved on each task).

Third, DISTILLER advantages were confirmed by the measure of relatedness among the considered tasks (showing effective *knowledge transfer*), in terms of dependence among censored outcomes, and by the better separation ensured, in terms of lower incoherence, with positive impact on explainability as well. Accordingly, the whole experimental evaluation shown in Sec. 5 substantiates our claim that employing a principled approach for designing a DL architecture for TC tasks allows to better exploit DL potential, while simultaneously avoiding domain-specific pitfalls.

Our newly-proposed DISTILLER approach suggests the following *future directions* of research: (*i*) use of advanced DL layers (e.g., inception, residual, attention); (*ii*) semi-supervised multitask learning; (*iii*) gray-box analysis of DL traffic classifiers with explainable AI tools [43]; (*iv*) open-set TC, i.e. ability to handle classes not present in the training set; (*v*) sensible adoption of the Big Data paradigm to (encrypted) traffic classifiers drawn from DISTILLER.

References

[1] A. Dainotti, A. Pescapè, K. C. Claffy, Issues and future directions in traffic classification, IEEE Network 26 (2012) 35–40.

[2] X. Wang, K. Xu, W. Chen, Q. Li, M. Shen, B. Wu, ID-Based SDN for the Internet of Things, IEEE Network 34 (2020) 76–83.

[3] F. Jejdling, et al., Ericsson mobility report, Ericsson AB, Business Area Networks, Stockholm (SE), Tech. Rep. EAB-19 7381 (2019).

[4] H. Yao, G. Ranjan, A. Tongaonkar, Y. Liao, Z. M. Mao, SAMPLES: Self adaptive mining of persistent lexical snippets for classifying mobile application traffic, in: ACM 21st International Conference on Mobile Computing and Networking (MobiCom), 2015, pp. 439–451.

[5] R. Houser, Z. Li, C. Cotton, H. Wang, An investigation on information leakage of DNS over TLS, in: 15th International Conference on Emerging Networking Experiments And Technologies (CoNEXT), 2019, pp. 123–137.

[6] Z. Chai, A. Ghafari, A. Houmansadr, On the importance of encrypted-SNI (ESNI) to censorship circumvention, in: 9th USENIX Workshop on Free and Open Communications on the Internet (FOCI), 2019.

[7] A. Jonas, J. Burrell, Friction, snake oil, and weird countries: Cybersecurity systems could deepen global inequality through regional blocking, Big Data & Society 6 (2019) 2053951719835238.

[8] V. F. Taylor, R. Spolaor, M. Conti, I. Martinovic, Robust smartphone app identification via encrypted network traffic analysis, IEEE Transactions on Information Forensics and Security 13 (2018) 63–78.

[9] G. Aceto, D. Ciuonzo, A. Montieri, A. Pescapè, Multi-classification approaches for classifying mobile app traffic, Journal of Network and Computer Applications 103 (2018) 131–145.

[10] V. Carela-Español, P. Barlet-Ros, M. Solé-Simó, A. Dainotti, W. de Donato, A. Pescapè, K-dimensional trees for continuous traffic classification, in: 2nd International Workshop on Traffic Monitoring and Analysis (TMA), volume 6003, 2010, pp. 141–154.

[11] A. Dainotti, F. Gargiulo, L. I. Kuncheva, A. Pescapè, C. Sansone, Identification of traffic flows hiding behind TCP port 80, in: IEEE International Conference on Communications (ICC), 2010, pp. 1–6.

[12] M. Shen, Y. Liu, L. Zhu, K. Xu, X. Du, N. Guizani, Optimizing feature selection for efficient encrypted traffic classification: A systematic approach, IEEE Network 34 (2020) 20–27.

[13] I. Goodfellow, Y. Bengio, A. Courville, Deep learning, MIT press, 2016.

[14] Z. Wang, The Applications of Deep Learning on Traffic Identification., Black Hat USA, Las Vegas, 2015.

[15] W. Wang, M. Zhu, J. Wang, X. Zeng, Z. Yang, End-to-end encrypted traffic classification with one-dimensional convolution neural networks, in: IEEE International Conference on Intelligence and Security Informatics (ISI), 2017, pp. 43–48.

[16] M. Lotfollahi, M. J. Siavoshani, R. S. H. Zade, M. Saberian, Deep packet: A novel approach for encrypted traffic classification using deep learning, Soft Computing 24 (2020) 1999–2012.

[17] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, J. Lloret, Network traffic classifier with convolutional and recurrent neural networks for Internet of Things, IEEE Access 5 (2017) 18042–18050.

[18] G. Aceto, D. Ciuonzo, A. Montieri, A. Pescapé, Mobile encrypted traffic classification using deep learning: experimental evaluation, lessons learned, and challenges, IEEE Transactions on Network and Service Management 16 (2019) 445–458.

[19] G. Aceto, D. Ciuonzo, A. Montieri, A. Pescapè, MIMETIC: mobile encrypted traffic classification using multimodal deep learning, Elsevier Computer Networks 165 (2019) 106944.

[20] S. Rezaei, X. Liu, Multitask learning for network traffic classification, in: 29th IEEE International Conference on Computer Communications and Networks (ICCCN), 2020, pp. 1–9.

[21] A. Rago, G. Piro, G. Boggia, P. Dini, Multi-task learning at the mobile edge: an effective way to combine traffic classification and prediction, IEEE Transactions on Vehicular Technologies 69 (2020) 10362–10374.

[22] S. Ruder, An overview of multi-task learning in deep neural networks, http://arxiv.org/abs/1706.05098, 2017. arXiv:1706.05098.

[23] G. Draper-Gil, A. H. Lashkari, M. Mamun, A. A. Ghorbani, Characterization of encrypted and VPN traffic using time-related features, in: 2nd International Conference on Information Systems Security and Privacy (ICISSP), 2016, pp. 407–414.

[24] H. Huang, H. Deng, J. Chen, L. Han, W. Wang, Automatic multi-task learning system for abnormal network traffic detection, International Journal of Emerging Technologies in Learning 13 (2018) 4–20.

[25] H. Sun, Y. Xiao, J. Wang, J. Wang, Q. Qi, J. Liao, X. Liu, Common knowledge based and one-shot learning enabled multi-task traffic classification, IEEE Access 7 (2019) 39485–39495.

[26] Y. Zhao, J. Chen, D. Wu, J. Teng, S. Yu, Multi-task network anomaly detection using federated learning, in: ACM 10th International Symposium on Information and Communication Technology (SoICT), 2019, pp. 273–279.

[27] D. Li, Y. Zhu, W. Lin, Traffic identification of mobile apps based on variational autoencoder network, in: 13th IEEE International Conference on Computational Intelligence and Security (CIS), 2017, pp. 287–291.

[28] L. Vu, C. T. Bui, Q. U. Nguyen, A deep learning based method for handling imbalanced problem in network traffic classification, in: ACM 8th

International Symposium on Information and Communication Technology (SoICT), 2017, pp. 333–339.

[29] C. Liu, L. He, G. Xiong, Z. Cao, Z. Li, FS-Net: A flow sequence network for encrypted traffic classification, in: IEEE Conference on Computer Communications (INFOCOM), 2019, pp. 1171–1179.

[30] Y. Zeng, H. Gu, W. Wei, Y. Guo, *Deep − Full − Range*: a deep learning based network encrypted traffic classification and intrusion detection framework, IEEE Access 7 (2019) 45182–45190.

[31] H. Yao, C. Liu, P. Zhang, S. Wu, C. Jiang, S. Yu, Identification of encrypted traffic through attention mechanism based long short term memory, IEEE Transactions on Big Data, in press (2019).

[32] G. Aceto, D. Ciuonzo, A. Montieri, A. Pescapé, Toward effective mobile encrypted traffic classification through deep learning, Neurocomputing 409 (2020) 306–315.

[33] Z. Yao, J. Ge, Y. Wu, X. Lin, R. He, Y. Ma, Encrypted traffic classification based on Gaussian mixture models and hidden Markov models, Journal of Network and Computer Applications 166 (2020) 102711.

[34] D. Li, W. Li, X. Wang, C.-T. Nguyen, S. Lu, App trajectory recognition over encrypted internet traffic based on deep neural network, Elsevier Computer Networks 179 (2020) 107372.

[35] C. Dong, C. Zhang, Z. Lu, B. Liu, B. Jiang, CETAnalytics: comprehensive effective traffic information analytics for encrypted traffic classification, Elsevier Computer Networks 176 (2020) 107258.

[36] D. Ramachandram, G. W. Taylor, Deep multimodal learning: A survey on recent advances and trends, IEEE Signal Processing Magazine 34 (2017) 96–108.

[37] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, A. Y. Ng, Multimodal deep learning, in: 28th International Conference on Machine Learning (ICML), 2011, pp. 689–696.

[38] L. Bernaille, R. Teixeira, K. Salamatian, Early application identification, in: ACM CoNEXT conference, 2006, p. 6.

[39] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: 3rd International Conference for Learning Representations (ICLR), 2015.

[40] H. Qi, E. R. Sparks, A. Talwalkar, PALEO: A performance model for deep neural networks, in: International Conference on Learning Representations (ICLR), 2017.

[41] E. Mizutani, S. E. Dreyfus, On complexity analysis of supervised MLP-learning for algorithmic comparisons, in: IEEE International Joint Conference on Neural Networks (IJCNN), volume 1, 2001, pp. 347–352.

[42] A. Montieri, D. Ciuonzo, G. Aceto, A. Pescape, Anonymity services Tor, I2P, JonDonym: classifying in the dark (web), IEEE Transactions on Dependable and Secure Computing 17 (2020) 662–675.

[43] H. Hagras, Toward human-understandable, explainable AI, IEEE Computer 51 (2018) 28–36.