Encrypted Multitask Traffic Classification via Multimodal Deep Learning

Giuseppe Aceto, Domenico Ciuonzo, Antonio Montieri, Alfredo Nascita, Antonio Pescapé
University of Napoli "Federico II" (Italy)
{giuseppe.aceto, domenico.ciuonzo, antonio.montieri, pescape}@unina.it, a.nascita@studenti.unina.it

***Abstract*—Traffic Classification (TC), i.e. the collection of procedures for inferring applications and/or services generating network traffic, represents the workhorse for service management and the enabler for valuable profiling information. Sadly, the growing trend toward encrypted protocols (e.g. TLS) and the evolving nature of network traffic make TC design solutions based on payload-inspection and machine learning, respectively, unsuitable. Conversely, Deep Learning (DL) is currently foreseen as a viable means to design traffic classifiers based on automatically-extracted features, reflecting the complex patterns distilled from the multifaceted (encrypted) traffic nature, implicitly carrying information in "multimodal" fashion. To this end, in this paper a novel multimodal DL approach for multitask TC is explored. The latter is able to capitalize traffic data heterogeneity (by learning both intra- and inter-modality dependencies), overcome performance limitations of existing (myopic) single-modality DL-based TC proposals, and solve different traffic categorization problems associated with different providers' desiderata. Based on a real dataset of encrypted traffic, we report performance gains of our proposal over (a) state-of-art multitask DL architectures and (b) multitask extensions of single-task DL baselines (both based on single-modality philosophy).**

***Index Terms*—traffic classification; encrypted traffic; deep learning; multitask learning; multimodal learning.**

I. Introduction

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 traffic. Such process, known as Traffic Classification (TC), has a long-established application in several fields [1]. Sadly, the widespread adoption [2] of encrypted protocols (TLS) as well as NAT and dynamic ports, increases the difficulty to 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 [3]. On the other hand, stronger privacy needs arose, against authoritarian governments enforcing surveillance and censorship on the Internet, and service providers negating network neutrality [4]: the assessment of TC performance provides high value insight for designers of (privacy-preserving) protocols, and obfuscation and circumvention techniques.

In this complex scenario, Machine Learning (ML) classifiers have proved to be the most appropriate for modern-traffic classification, as they suit Encrypted Traffic (ET) while not necessarily relying on port information [5]. However, their use relies on handcrafted (domain-expert driven) features: such process is unable to cope with modern network traffic evolution, and impairs the design of both *accurate* and *up-to-date* traffic classifiers [6] using "traditional" ML approaches.

Based on these considerations, in last years several works started tackling TC via DL since the work [7], including convolutional [8, 9] and recurrent [10] architectures. These attempts revealed a low level of maturity in applying DL to this specifically hard problem: we found in [11] how misguided design choices led to *biased* conclusions. Also, DL approaches provide new performance gain possibilities: by neglecting these, designers would fail to harness the full potential offered by DL. To face these shortcomings in TC state-of-art, in [11] we laid a sound groundwork for the design of DL-based classifiers aimed at highly-diverse traffic, both avoiding the pitfalls of naïve adoption of DL to TC, and allowing for best exploiting the potential of DL architectures. In [12] we have leveraged on such groundwork to explore multimodal approaches to enhance the performance of ET classification. Herein, we further our exploration with *multimodal multitask* approaches applied to the same goal, leading to the implementation of the DISTILLER (encrypteD multItaSk Traffic classIfication via muLtimodaL dEep leaRning) proposal.

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. Besides reducing some redundancy (sharing part of the feature-learning architecture), this approach promises improved generalization and better overall classification performance. To experimentally investigate these properties, our work adopts a general architecture for the simultaneous solution of multiple (related) encrypted TC tasks via DL techniques (*multitask*), also able to best exploit heterogeneous (*multimodal*) inputs. Based on the `ISCX VPN-nonVPN` [13] dataset, we investigate the performance of the proposed approach comparing it with the best *seven* state-of-art multitask approaches, in fair conditions. To our knowledge, no similar design and experimental investigation have been presented in the encrypted TC scenario to date.[1]

The paper is organized as follows. Sec. II reviews literature on (encrypted) TC (including most relevant DL works), whereas Sec. III describes our DISTILLER encrypted traffic classifier; experimental evaluation is provided in Sec. IV; Sec. V provides conclusions and future research avenues.

[1]Extended results from this study have been published as journal publication [14].

II. Related Work

Several works face the problem of TC by means of ML- and (more recently) DL-based approaches. After exploratory attempts, some works started considering *multitask* architectures as well [15]: we hereafter describe the most relevant.

Different DL architectures for encrypted TC are proposed by Lopez-Martin et al. [10], combining Long Short-Term Memory (LSTM) and 2D-convolutional layers. The proposals are evaluated on a real traffic (from academic backbone network). The results show high performance, also highlighting a penalty associated to the use of inter-arrival times as input.

Wang et al. [8] apply DL to malware TC, proposing a method based on 1D-Convolutional Neural Network (CNN) tailored for ET. The experimental evaluation is conducted on a selection of the `ISCX VPN-nonVPN` dataset [13] and is divided in four different classification problems: (i) VPN/nonVPN, (ii) 6 encrypted traffic classes, (iii) 6 VPN-tunneled traffic classes, and (iv) 12 encrypted applications. Different inputs are considered, including *biased* ones [11].

More recently, Huang et al. [16] have applied the multitask learning paradigm to solve (i) malware detection (binary), (ii) VPN-encapsulation recognition (binary), and (iii) Trojan classification (9 classes) tasks. The proposed DL algorithm is a 2D-CNN, tested on an assembled dataset obtained from `CTU-13` (malware) and `ISCX VPN-nonVPN` [13] datasets. Following the work in [8], *biased* input data are employed.

A multitask DL architecture is proposed by Zhao et al. [17] 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 are 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 (single-task Deep Neural Network).

Rezaei and Liu [18] propose an architecture to simultaneously classify the traffic (aggregated in 5 application categories) and *predict* biflow bandwidth and duration (quantized over 5 and 4 intervals, respectively). Performance is evaluated in a *transfer learning* setup, with one of the classification tasks limited by a *scarce ground-truth*.

The same set of classification tasks is considered by Sun et al. [19], but for both duration and flow rate a quantization at two levels is investigated; the application type spans between 11 and 50 classes, according to the specific dataset (4 datasets are employed, collected between 2003 and 2010). In this case as well, a preliminary feature selection is performed referring to previous ML literature, including statistical preprocessing and transport-layer ports. Again, we highlight how this is a misguided approach when leveraging DL architectures. Moreover the inclusion of transport-level ports does not abide by modern traffic nature (NAT, mobile apps, tunneling, encryption). The authors evaluate performance in terms of *perplexity*, and also training time, not always obtaining gains compared with single-task baselines. Performance is evaluated also in *transfer learning* and *one-shot learning* scenarios, highlighting the effectiveness of the multitask approach in facing ground-truth scarcity for a single task.

Compared with the above state-of-the-art, our contributions are manifold: (i) we substitute the preliminary (manual) feature selection with a more apt Deep Neural Network; (ii) we adopt a multimodal architecture to process input data—the combination of (i) and (ii) allows for automatic feature learning from heterogeneous data input; (iii) we feed the architecture with simple *unbiased* inputs; (iv) we experimentally compare our proposed DISTILLER classifier with the best proposals from the state-of-the-art, implementing all with the same technologies and feeding and evaluating all in fair conditions.

III. Multimodal Multitask Deep Learning–based Traffic Classification

A. Overall Architecture

In this section, we describe the proposed methodology for multipurpose encrypted TC via multimodal multitask DL. Herein, we assume that we are required to solve $v = 1, \ldots, V$ 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 $\{1, \cdots, L_v\}$.

As a necessary prerequisite, raw traffic is first segmented via **traffic object segmentation** into distinct TC objects [1], each constituting a subset of network packets sharing some common properties (defined by the segmentation rule). Most works tackling DL-based TC (see Sec. II) considered either *flows* or *biflows*, with the latter outperforming the former. A *flow* is defined as a stream of packets sharing the 5-tuple: source (IP, port), destination (IP, port) and transport-level protocol. Differently, a *biflow* includes also packets with source and destination pairs exchanged.

We remark that DL-based traffic classifiers learn the distinctive fingerprint of each traffic type/application via a *training set*. 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 v^{th} classification task $\ell^v(m)$, belonging to one out of L_v different classes (i.e. $\ell^v(m) \in \{1, \ldots, L_v\}$). The main advantage of DL (as opposed to ML) approaches for TC is to overcome the little-adaptable feature design process [11]. This is due to their ability to learn traffic fingerprints in a *end-to-end* fashion, i.e. *directly* from the type of input selected. Still, traffic data is intrinsically highly-structured, as each sample 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. Conversely, *late fusion* via the capitalization of score-results of DL-based traffic classifiers built on different modalities, does not fully exploit the benefits of

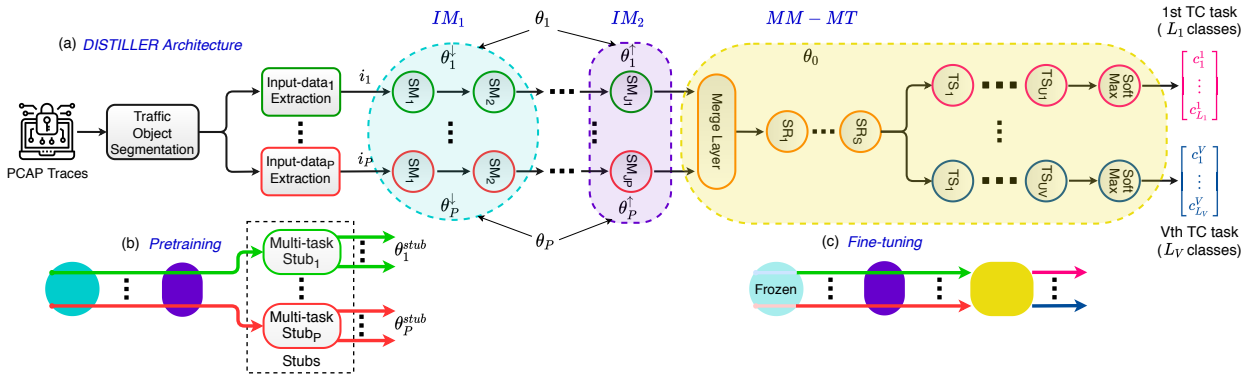


Figure 1: Architectural view of DISTILLER classifier. (a) depicts the architecture by highlighting single-modality representation layers, differentiated as those that are only pre-trained (IM_1) and those that are also fine-tuned (IM_2), and shared representation-layers ($MM-MT$), along with the corresponding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

multi-modality. Accordingly, we foresee multimodal multitask DL as the appealing means toward a sophisticated *intermediate fusion* [20], overcoming these limitations by offering a flexible tool for practical encrypted TC, able to solve multiple tasks by processing heterogeneous inputs.

The **architecture** of our DISTILLER classifier is depicted in Fig. 1(a) 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 representation.[2] Differently, the second part (`MM-MT`) of the architecture consists of S *shared-representation* (`SR`) layers, distilling features capturing inter-modality dependencies and U_v *task-specific* (`TS`) layers, synthesizing the task-oriented features (of the v^{th} task) from the shared ones. Finally, the architecture is completed with a `softmax` layer for each encrypted TC task to solve, returning the corresponding soft-output (prediction) vector. 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[3] capitalizing the "connectionist" philosophy [21].

B. 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. 1(b) and 1(c), respectively). Indeed, pre-training allows the DL branch of each modality to be able to acceptably solve—by itself—all the considered tasks [21].

Precisely, each single-modality stack is first (pre-)trained independently (Fig. 1(b)), i.e. *without* `MM-MT` (no `SR` and `TS` layers) and by topping each modality chain with V different softmax layer "stubs" (whose parameters 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 (within the curly brackets), namely:

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

Such distance is measured via $\text{CE}(\boldsymbol{t}, \boldsymbol{c}) \triangleq -\{\sum_{\ell=1} t_\ell \log c_\ell\}$, denoting the CE distance for the generic training sample. In Eq. (1), the vector $\boldsymbol{c}^v(m) \triangleq \left[c_1^v(m) \;\; \cdots \;\; c_{L_v}^v(m)\right]^T$ collects the predicted class confidences of DL classifier (which depend on DL network parameters) for the label of the m^{th} training sample on the v^{th} task. Differently, $\boldsymbol{t}^v(m) \triangleq \left[t_1^v(m) \;\; \cdots \;\; t_{L_v}^v(m)\right]^T$ denotes the corresponding *one-hot* representation of the label for v^{th} classification task $\ell^v(m)$. The aim of a high-performing traffic classifier on v^{th} task is to have the confidence $\boldsymbol{c}^v(m)$ as close as possible to the (ground-truth originated) one-hot vector $\boldsymbol{t}^v(m)$. 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* (Fig. 1(c)), 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 parameters $\boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_P$ and $\boldsymbol{\theta}_0$, associated to $MM-MT$ block). 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 $\boldsymbol{\theta}_p \triangleq \left[\boldsymbol{\theta}_p^{\downarrow} \;\; \boldsymbol{\theta}_p^{\uparrow}\right]$ only the subset $\boldsymbol{\theta}_p^{\uparrow}$ is (further) optimized during fine-tuning (i.e. those

[2]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, entry-wise maximum).

[3]Also, to promote *regularization* (to avoid overfitting), dropout between successive layers and early-stopping techniques are adopted [21].

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:

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

The loss functions concerning pre-training (single-modality multitask, $\mathcal{L}_p(\cdot)$) and fine-tuning (multi-modality multitask, $\mathcal{L}(\cdot)$) phases are minimized via standard first-order local optimizers (e.g., SGD, ADAM, etc.), resorting to the usual back-propagation for gradient evaluation [21]. Hereinafter, we detail the specific instance obtained from the general architecture of DISTILLER and used for experimental evaluation in Sec. IV.

C. Description of Proposed Instance

For a fair comparison with earlier works [8, 10, 11, 12], this particular implementation of the proposed DISTILLER classifier operates with **biflow TC objects** and is made of $\boldsymbol{P = 2}$ **modalities**.

The input data fed to the DISTILLER classifier belong to *two* types[4]: (a) the first N bytes of transport-layer payload (`PAY`) of the TC object [7, 8]; (b) informative protocol header fields (`HDR`) of first N_p packets [10]. 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: number of bytes in transport-layer payload, TCP window size (set to *zero* for UDP packets), inter-arrival time, and packet direction $\in \{0, 1\}$—of first N_p packets [10]. These specific input data derive from the necessity of avoiding *biased* inputs (a common pitfall in related works) included e.g. in PCAP metadata, data-link layer, and some transport-layer header fields, as they may lead to inflated performance, and lack of generalization [11]. 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 are naturally conductive to the *multimodal* approach. Also, they suit well "early" TC [22].

The architecture[5] implements the `SM` layers of the "payload" modality ($p = 1$) 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 ($p = 2$) are, in order, a bidirectional GRU (64 nodes and return-sequences behavior) and one dense layer (128 nodes).[6] 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 layers, each constituting one `TS` dense layer (128 nodes) for the v^{th} task ($U_v = 1$), before the corresponding `softmax` layer. In all the layers, the outputs are obtained via 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 GRU. The considered architectural 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 GRU layers). For both phases, we employ ADAM optimizer (batch size of 50) and early-stopping technique (to prevent overfitting) measured on the training accuracy of the hardest TC task (i.e. with the highest number of classes L_v).

Table I: `ISCX VPN-nonVPN` Dataset Tasks and Classes.

Task	Classes
T_1 - *Encapsulation*	VPN, nonVPN
T_2 - *Traffic Type*	Chat, Email, FileTransfer, P2P, Streaming, VoIP
T_3 - *Application*	Aim, Email, Facebook, FTPS, Hangouts, ICQ, Netflix, SCP, SFTP, Skype, Spotify, Torrent, Vimeo, VoipBuster, YouTube

IV. EXPERIMENTAL EVALUATION

Herein we investigate and compare performance of one classifier instance taken from the general architecture of DISTILLER. The description of the human-generated dataset and of the existing multitask baselines is given in Sec. IV-A; corresponding results are discussed in Sec. IV-B.

A. Dataset and Baseline Description

The dataset employed herein is `ISCX VPN-nonVPN` [13] collected at the Canadian Institute for Cybersecurity and provided in raw PCAP format with trace-level labels (i.e. the ground-truth is associated to a whole PCAP trace). It includes human-generated traffic encompassing different traffic types, captured via both regular sessions and sessions over VPN. Given this structure, we can associate a three-view label (encapsulation, traffic type, and application) to each biflow, corresponding to likewise TC *tasks* to be tackled. We list the classes associated to each task T_v in Tab. I.

Pre-processing operations: we have found that $\approx 65\%$ of biflows have only one UDP packet and destination (IP address, port) equal to (255.255.255.255, 10505)[7]. Thus, as opposed to

[4]We underline that, in *both* cases, *longer* (resp. *shorter*) instances are truncated (resp. padded with zeros) to the designed length of bytes or packets.

[5]We leveraged DL models provided by *Keras* (https://keras.io) Python API running on top of *TensorFlow* (https://www.tensorflow.org/) to implement and test the approaches described in this work. Hyperparameters are chosen based on our experience with DL architectures for TC [11, 12], complemented with a set of experiments tuning the parameters of the dense layers.

[6]1D convolutional layers extract spatially-invariant patterns from the payload, while GRUs capture long-term dependencies pertaining to the initial segments of the biflow.

[7]We have found that these packets are network broadcasts periodically sent (every 2 seconds by default) by BlueStacks, an Android emulator for PCs.

Table II: 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 10-folds. Rank is based on average of all performance metrics on all tasks. Last row shows DISTILLER Gain [%] on the overall-best baseline (ranking II). Highlighted 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	94.09 (± 0.84) ⋆	92.32 (± 0.97) ⋆	81.22 (± 0.73) ⋆	79.21 (± 1.50) ⋆	78.09 (± 1.00) ⋆	66.28 (± 1.61) ⋆	9.08 (± 0.15)
II	1D-CNN (PAY) [8]	88.06 (± 0.88)	84.41 (± 0.92)	73.59 (± 1.51) †	71.67 (± 1.58) †	73.32 (± 1.10) †	60.93 (± 2.00) †	13.25 (± 0.41)
III	MLP (PAY) [17]	87.27 (± 0.91)	83.01 (± 1.05)	71.66 (± 0.86)	69.10 (± 0.82)	70.02 (± 1.55)	57.10 (± 2.45)	2.55 (± 0.03)
IV	MLP (HDR) [17]	88.67 (± 1.11) †	84.90 (± 1.31) †	68.10 (± 1.30)	65.84 (± 1.05)	63.69 (± 0.85)	51.35 (± 2.15)	2.36 (± 0.02)
V	MLP (PAY) [19]	85.81 (± 0.91)	81.52 (± 1.17)	68.63 (± 1.23)	65.97 (± 1.50)	66.70 (± 1.20)	53.06 (± 2.02)	0.91 (± 0.03)
VI	HYBRID (HDR) [10]	87.69 (± 3.32)	83.98 (± 3.27)	67.23 (± 5.25)	63.65 (± 5.40)	63.09 (± 5.89)	51.42 (± 7.04)	2.91 (± 0.03)
VII	MLP (HDR) [19]	86.76 (± 0.93)	82.15 (± 0.92)	63.55 (± 2.06)	59.95 (± 2.87)	59.78 (± 0.84)	45.37 (± 1.73)	0.83 (± 0.02)
VIII	1D-CNN (HDR) [18]	83.83 (± 1.47)	78.29 (± 2.02)	61.55 (± 2.35)	57.95 (± 2.93)	58.68 (± 2.32)	42.93 (± 2.34)	1.81 (± 0.02)
	DISTILLER Gain	**+ 6.03 (± 0.55)**	**+ 7.91 (± 0.85)**	**+ 7.63 (± 1.14)**	**+ 7.54 (± 1.21)**	**+ 4.76 (± 0.77)**	**+ 5.35 (± 1.22)**	**- 4.16 (± 0.45)**

other works leveraging the ISCX VPN-nonVPN dataset [16, 17, 18], we have performed a cleaning operation to remove this noisy traffic and make our results more meaningful. As a result the dataset contains 11.6k biflows.

We compare DISTILLER with several baselines proposed in most-related (recent) literature [8, 10, 16, 17, 18, 19] 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. We do not employ biased input types (e.g., raw PCAP data and ports [11]) and manually-extracted features (e.g., PL/IAT stats). Indeed, such choices would nullify a key benefit of DL: no need of human-expert intervention for extracting informative features. Notably, we have implemented multitask variants 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 [8] and [10], respectively. More important, we have also considered *native multitask DL architectures* proposed for TC, that is the 1D-CNN (HDR) [8] proposed in [18] and two variants of the (deep) MLP (PAY/HDR) devised in [17, 19]. For the last two baselines, we have considered two variants each, fed with either input type, since the original proposals had handcrafted PL/IAT stats as input. Last, we have discarded the proposal [16], due to biased inputs, unjustified use of 2D convolutional layers, and excessively ad-hoc architecture.

B. Classification Results and Discussion

For each considered analysis, our evaluation is based on a (stratified on T_3) 10-fold cross-validation, representing a stable performance evaluation setup. As a consequence, we report both the mean and the variance of each performance measure as a result of the evaluation on the ten different folds.

Our first analysis assesses the performance of DISTILLER in Tab. II, comparing it with most-related baselines (see Sec. II). We compare them in terms of accuracy (the share of samples properly classified) and macro (i.e. arithmetically-averaged over classes) F-measure[9] on the three tasks. We also investigate the computational complexity of the considered (multitask) DL-architectures via their training phase runtime. This is a key aspect in modern TC, where frequent re-training is required, due to aging of training data as a result of fast-paced evolution of network usage. Since training is performed on multiple epochs, for conciseness we report the Run-Time Per-Epoch (RTPE) in last column. All the classifiers (including DISTILLER) are listed according to an average performance ranking, i.e. by decreasing average of both performance metrics over all three tasks. This led to an *overall-best performing baseline* classifier (1D-CNN (PAY) [8]). At the bottom of the table, the performance gain of DISTILLER w.r.t. this overall-best-performing baseline is summarized as well.

It can be noted that **DISTILLER ranks first** as overall-best-performing classifier, showing **significant performance gains for all the metrics, for all the tasks**, ranging from $+4.8\%$ to $+7.9\%$, w.r.t. the overall-best-performing baseline. DISTILLER also obtains a **reduction in the training time** of -4.2 seconds per-epoch, thus exhibiting lower empirical computational complexity, when compared with the overall-best-performing baseline.[10] Even considering the classification performance for each single task, DISTILLER always outperforms also the best-per-metric baseline (marked with † in Tab. II).

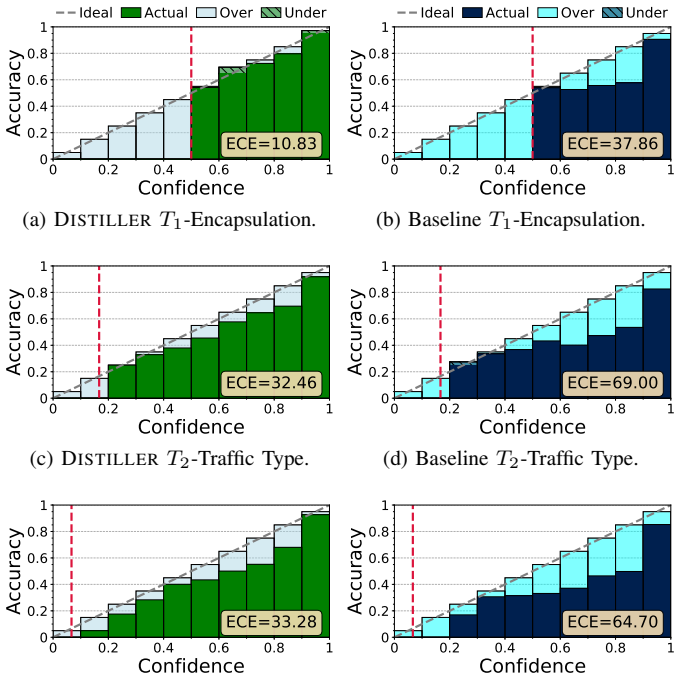
To deepen the performance evaluation, Fig. 2 reports a *calibration analysis*, assessing whether the class-probability estimates are representative of the true-class (posterior) probabilities, where a miscalibrated classifier (on a given task) returns excessively optimistic or pessimistic decisions. Specifically, we consider *reliability diagrams*, that show accuracy as a function of confidence and are obtained by partitioning the predictions into equally-spaced bins and calculating the accuracy of each bin. Confidence ranges in $[1/L_v\,,\,1]$, where L_v is the number of classes of task v. A perfectly-calibrated classifier implies a diagram coinciding with the identity function, e.g. operating with 70% confidence leads to 70% accuracy.

The above diagrams are complemented with the *Expected Calibration Error* (ECE), i.e. the weighted (based on the number of samples) sum of the difference between accu-

[8]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.

[9]The latter is the harmonic mean of precision (the per-class fraction of decisions being correct) and recall (the class-conditional accuracy).

[10]Overall, DISTILLER has 0.99M trainable parameters against 1D-CNN (PAY) [8] having 5.84M parameters.



(a) DISTILLER T_1-Encapsulation. (b) Baseline T_1-Encapsulation.

(c) DISTILLER T_2-Traffic Type. (d) Baseline T_2-Traffic Type.

(e) DISTILLER T_3-Application. (f) Baseline T_3-Application.

Figure 2: Reliability diagrams of DISTILLER (a,c,e), and best overall baseline (b,d,f) for the three 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 task v. Under and over gap represent an under-confident (pessimistic) and over-confident (optimistic) miscalibration pattern, respectively. Synthetic *ECE* is reported for each case.

racy and confidence bin values. We compare our proposed DISTILLER instance with the best overall baseline over the three TC tasks considered. It can be seen that **DISTILLER is better calibrated** with respect to the best baseline, showing an ECE *smaller than a third* on T_1 (resp. *approximately a half* on T_2, T_3). Interestingly, both classifiers exhibit always a miscalibration that tends to be over-confident (optimistic) in predictions. This effect can be attributed to a slight overfitting phenomenon and is one of the distinctive characteristics of DL architectures (although our MM-MT architecture mitigates it).

All these results support our claim that using a principled approach in structuring a DL architecture for TC tasks allows to better exploit DL potential, while simultaneously avoiding domain-specific pitfalls, and that a properly-structured multimodal multitask architecture can be effective in both raising TC performance and lowering computational complexity.

V. CONCLUSIONS AND FUTURE DIRECTIONS

We tackled multipurpose encrypted TC via a general multimodal multitask DL architecture, based on a suitably-defined training procedure which enforces information distillation from each modality. The evaluation has been performed on a dataset of human-generated traffic labelled according to three different TC tasks. Results have shown performance gains by DISTILLER over state-of-the-art multitask architectures up to $+7.9\%$ (T_1), $+7.5\%$ (T_2), and $+5.3\%$ (T_3) in terms of F-measure, with very manageable training complexity. Also the calibration analysis has underlined general improved behavior on the three different tasks, due to the benefits of multimodality to solve multiple tasks. *Future directions* include: (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; (iv) open-set TC, i.e. ability to handle classes not present in the training set; (v) Big Data-enabled traffic classifiers drawn from DISTILLER.

REFERENCES

[1] A. Dainotti, A. Pescapè, and K. C. Claffy, "Issues and future directions in traffic classification," *IEEE Network*, vol. 26, no. 1, pp. 35–40, 2012.
[2] F. Jejdling *et al.*, "Ericsson mobility report," *Ericsson AB, Business Area Networks, Stockholm (SE), Tech. Rep. EAB-19*, vol. 7381, Nov. 2019.
[3] H. Yao, G. Ranjan, A. Tongaonkar, Y. Liao, and Z. M. Mao, "SAMPLES: Self adaptive mining of persistent lexical snippets for classifying mobile application traffic," in *ACM MobiCom'15*.
[4] G. Aceto and A. Pescapé, "Internet censorship detection: A survey," *Elsevier Computer Networks*, vol. 83, pp. 381–421, 2015.
[5] V. F. Taylor, R. Spolaor, M. Conti, and I. Martinovic, "Robust smartphone app identification via encrypted network traffic analysis," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 1, pp. 63–78, 2018.
[6] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapè, "Multi-classification approaches for classifying mobile app traffic," *Journal of Network and Computer Applications*, vol. 103, pp. 131–145, 2018.
[7] Z. Wang, "The Applications of Deep Learning on Traffic Identification." Black Hat USA, Las Vegas, 2015.
[8] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," in *IEEE ISI'17*.
[9] M. Lotfollahi, M. J. Siavoshani, R. S. H. Zade, and M. Saberian, "Deep packet: A novel approach for encrypted traffic classification using deep learning," *Soft Computing*, vol. 24, no. 3, pp. 1999–2012, 2020.
[10] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Network traffic classifier with convolutional and recurrent neural networks for Internet of Things," *IEEE Access*, vol. 5, pp. 18 042–18 050, 2017.
[11] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Mobile encrypted traffic classification using Deep Learning: Experimental evaluation, lessons learned, and challenges," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 2, pp. 445–458, 2019.
[12] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapè, "MIMETIC: mobile encrypted traffic classification using multimodal deep learning," *Elsevier Computer Networks*, vol. 165, p. 106944, 2019.
[13] G. Draper-Gil, A. H. Lashkari, M. Mamun, and A. A. Ghorbani, "Characterization of encrypted and VPN traffic using time-related features," in *SciTePress ICISSP'16*, pp. 407–414.
[14] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Distiller: Encrypted traffic classification via multimodal multitask deep learning," *Journal of Network and Computer Applications*, p. 102985, 2021.
[15] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2224–2287, 2019.
[16] H. Huang, H. Deng, J. Chen, L. Han, and W. Wang, "Automatic multitask learning system for abnormal network traffic detection," *Int. Journal of Emerging Technologies in Learning*, vol. 13, no. 4, pp. 4–20, 2018.
[17] Y. Zhao, J. Chen, D. Wu, J. Teng, and S. Yu, "Multi-task network anomaly detection using federated learning," in *ACM SoICT'19*.
[18] S. Rezaei and X. Liu, "Multitask Learning for Network Traffic Classification," http://arxiv.org/abs/1906.05248, 2019.
[19] H. Sun, Y. Xiao, J. Wang, J. Wang, Q. Qi, J. Liao, and X. Liu, "Common knowledge based and one-shot learning enabled multi-task traffic classification," *IEEE Access*, vol. 7, pp. 39 485–39 495, 2019.
[20] D. Ramachandram and G. W. Taylor, "Deep multimodal learning: A survey on recent advances and trends," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 96–108, 2017.
[21] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.
[22] L. Bernaille, R. Teixeira, and K. Salamatian, "Early application identification," in *ACM CoNEXT'06*.