Improving Performance, Reliability, and Feasibility in Multimodal Multitask Traffic Classification with XAI

Alfredo Nascita, Antonio Montieri, *Member, IEEE*, Giuseppe Aceto, Domenico Ciuonzo, *Senior Member, IEEE*, Valerio Persico, and Antonio Pescapé, *Senior Member, IEEE*

***Abstract*—The promise of Deep Learning (DL) in solving hard problems such as network Traffic Classification (TC) is being held back by the severe lack of transparency and explainability of this kind of approaches. To cope with this strongly felt issue, the field of eXplainable Artificial Intelligence (XAI) has been recently founded, and is providing effective techniques and approaches. Accordingly, in this work we investigate interpretability via XAI-based techniques to understand and improve the behavior of state-of-the-art multimodal and multitask DL traffic classifiers. Using a publicly available security-related dataset (ISCX VPN-nonVPN), we explore and exploit XAI techniques to characterize the considered classifiers providing *global interpretations* (rather than sample-based ones), and define a novel classifier, Distiller-Evolved, optimized along three objectives: performance, reliability, feasibility. The proposed methodology proves as highly appealing, allowing to much simplify the architecture to get faster training time and shorter classification time, as fewer packets must be collected. This is at the expenses of negligible (or even positive) impact on classification performance, while understanding and controlling the interplay between inputs, model complexity, performance, and reliability.**

***Index Terms*—Deep learning, encrypted traffic, explainable artificial intelligence, multimodal learning, multitask learning, traffic classification.**

I. Introduction

Traffic Classification (TC) is a key activity instrumental to traffic management, resource planning, and security enforcement in today's networks, which are characterized by high heterogeneity and dynamicity of transmitted traffic. To meet these challenging demands, recent research is strongly focusing on Deep Learning (DL) methods for designing effective tools for accurate TC [1]. While promising unparalleled performance, and the capability to adapt (without an expert human in the loop) to the change of observed traffic, these methods constitute a black-box whose behavior is extremely hard to explain, and therefore to improve, or secure against attacks: as such, they can not be trusted. As a result, all the main stakeholders agree there is a **strong need to research the explainability of Artificial Intelligence (AI) solutions applied to TC** and more generally to networking, with the aim of improving performance, reliability, and feasibility of such solutions.

Indeed, having explanations for AI systems is no longer just an attractive and desirable feature, but has become the fundamental basis of any AI design solution that users and network operators can consider safe, reliable, and fair. In fact, in critical contexts such as network management, it is no longer sufficient to have accurate systems: network administrators and users must trust the results and policies suggested by AI algorithms to decide whether to act and how.[1] This is more pressing as AI is being increasingly proposed to govern the complexity of real-time network resources management and network security, whose landscape has been subject to continuous evolution, last originated by the pandemic events.

This need for transparency motivated the Defense Advanced Research Projects Agency (DARPA) to launch its program for *eXplainable AI* (XAI) already in 2017 [2], with the aim of shaping new learning processes that (a) produce better-explainable models, (b) design effective explanation interfaces, and (c) understand the psychological requirements for effective explanations. Notably, while these needs apply generally to AI, they are specifically felt in the communications and networking field, with several telecommunications companies investigating how to produce AI-based solutions that can be the (safe) engine for their core business activities. For instance, Telefonica has focused on "Responsible Use of AI" [3] to address potential discrimination, lack of interpretability of the results provided by the algorithms, and transparency of the personal data they use, and has published its AI Principles in 2018, covering *fair AI, transparent and explainable AI, human-centric AI, privacy and security by design*. Besides, many other institutions and companies have defined rules

Manuscript received 30 April 2022; revised 3 November 2022 and 13 February 2023; accepted 15 February 2023. Date of publication 20 February 2023; date of current version 6 July 2023. This work is partially supported by the Italian Research Program "PON *Ricerca e Innovazione* 2014–2020 (PON R&I) – Asse IV *REACT-EU* – Azione IV.4", the "Centro Nazionale HPC, Big Data e Quantum Computing – ICSC" and the "RESTART" Project funded by MUR. The associate editor coordinating the review of this article and approving it for publication was N. Stakhanova. *(Corresponding author: Domenico Ciuonzo.)*

The authors are with the Department of Electrical Engineering and Information Technologies, University of Naples Federico II, 80125 Naples, Italy (e-mail: alfredo.nascita@unina.it; antonio.montieri@unina.it; giuseppe.aceto@unina.it; domenico.ciuonzo@unina.it; valerio.persico@unina.it; pescape@unina.it).

Digital Object Identifier 10.1109/TNSM.2023.3246794

[1]We underline that when referring to "trust" in results/decisions, we do not imply network security aspects.

and principles to guide their research. In line with the EU Commission guidelines for *Trustworthy AI*, Ericsson defined this concept [4] by focusing mainly on the explainability, safety, and verifiability of AI solutions. They also included traceability and accountability, always considering the human being at the center of the whole process. The lack of explainability is, according to Huawei, the main reason for the security vulnerabilities of AI systems. Indeed, unlike traditional systems, this lack of explainability poses specific and serious security threats to DL-based applications that can be exploited by adversarial machine learning methods such as evasion, poisoning, and backdoor attacks [5]. Equally important, the relevance and timeliness of XAI adoption are also motivated by the rising adoption of encryption (e.g., TLS encapsulation) in modern Internet traffic [6], [7].

Therefore, in response to the needs arising from DL adoption for effective TC, the field of XAI is providing tools to link the outcome of the classification to the structure of the DL model and the input, making the model *explainable*, in a number of ways [8]. In addition to explainability, in this work we consider also *reliability* [9], measuring to which extent the confidence associated with a given decision by a (possibly opaque) DL algorithm can be deemed reliable—i.e., low (resp. high) confidence in labeling a certain traffic object actually leads to low (resp. high) accuracy in classifying it. This property is crucial as it informs decisions with an impact on user experience and economic efficiency of network management. Finally, we also assess how XAI can be instrumentally used to ensure the prototyping of *feasible* (viz. ready-to-be-deployed) TC models [10]. Indeed, the understanding of the behavior of the learned model enables focused performance enhancements, much more efficient than a less-informed search over the huge hyper-parameters space. As the adoption of DL is relatively new, even more so in the field of network traffic analysis, it is no wonder that XAI has not yet found mature application to TC as well, despite its acute need: with this work we move an important step in the direction of tackling this open challenge.

In detail, the **main contributions** of this paper can be summarized as follows:

- Based on our recently proposed general framework (DISTILLER [11]), we design a novel architecture operating at *biflow level* which (*i*) effectively exploits the heterogeneous nature of the different views of a traffic object by distilling both intra- and inter-modalities dependence via *multimodal* learning and (*ii*) is able to solve multiple (related) TC tasks simultaneously via *multitask* learning. To support this improvement, we devise a *general methodology* for *interpreting* and *assessing the reliability* of multimodal multitask DL-based traffic classifiers in practical experimental scenarios. The above analysis includes *different levels of granularity or viewpoints*, including: (*a*) relative importance of different input modalities for each task and down to specific parts of each modality; (*b*) how each task affects the others' behavior. Specifically, our study goes through a number of *stages*, each associated with a different realization of the general DISTILLER framework, as detailed hereinafter.
- We propose and systematically evaluate a first variation of the original realization based on the DISTILLER framework—named DISTILLER-EMBEDDINGS—and compare its performance against state-of-the-art multitask TC baselines [12], [13], [14], including our original multimodal multitask proposal [11] referred to as DISTILLER-ORIGINAL herein. We investigate the intrinsic working behavior of DISTILLER-EMBEDDINGS applying state-of-the-art XAI tools (i.e., Deep SHAP [15] and Integrated Gradients [16]) to understand input importance associated with each modality, and within each single modality. In this context, a qualitative interpretability comparison with DISTILLER-ORIGINAL is also put forward.
- We leverage the results of interpretability analysis to improve our proposal and obtain a faster version of it—named DISTILLER-EARLIER—with reduced training times and allowing earlier classification. This is obtained by using a limited number of inputs and discarding irrelevant ones for both modalities without losing performance. We also compare the aforementioned method with (*a*) a classic feature selection method that relies on estimating mutual information, and we assess the results with (*b*) a complete sensitivity analysis based on grid search.
- We evaluate the reliability of DISTILLER-EARLIER via a calibration analysis, in order to assess how reliable the confidence value reported with the prediction is. This analysis supports further improvements of the proposed architecture by leveraging the *label smoothing* technique to improve the generalization capability of the model associated with each TC task, leading to DISTILLER-CALIBRATED. Indeed, the adopted calibration technique reduces the excessive confidence associated with predictions and consequently reduces overfitting.
- Aiming at improving the feasibility in terms of model size (and related memory occupation), in order to make the attained architecture deployable even on resource-constrained devices, we investigate (*i*) pruning, (*ii*) quantization, and (*iii*) knowledge distillation techniques to compress DISTILLER-CALIBRATED and obtain our final proposal, named DISTILLER-EVOLVED. In this context, we assess in detail if (and, in positive case, how) performance, interpretability, and reliability are *affected* by the consequent model simplification. Overall, DISTILLER-EVOLVED enhances the previously-proposed DISTILLER-ORIGINAL from different viewpoints: architecture and training procedure, input dimension, reliability (via calibration analysis), and feasibility (in terms of model size).
- The experimental campaign is conducted on ISCX VPN-NONVPN [17], a publicly-released human-generated encrypted-traffic dataset, to foster reproducibility.

The rest of the paper is organized as follows. Section II surveys the current state-of-the-art on TC via multimodal multitask DL and the recent application of XAI to networking and TC, positioning our contributions against related literature. Section III describes the considered XAI-based multimodal multitask TC methodology; the experimental setup considered and the experimental results discussed are reported in Sections IV and V, respectively; Section VI ends the paper with conclusions and future directions of research.

II. Background and Related Work

In this section, we first review the current state-of-the-art on TC via multimodal multitask DL (Section II-A). Then, we discuss the current application of XAI methods to the networking field (Section II-B). Finally, we position our contribution against existing methods adopting XAI to DL-based TC (Section II-C).

A. Traffic Classification via Multimodal Multitask DL

Several works perform encrypted TC by means of a variety of DL algorithms, including Deep Neural Networks (DNNs) [12], [13], different types of AutoEncoders (AEs) [1], [33], [34], [35], [36], one- and two-dimensional Convolutional Neural Networks (1D- and 2D-CNNs) [1], [11], [14], [26], [29], [30], [32], [35], [37], [38], [39], [40], [41], [42], [43], variants of Recurrent Neural Networks (RNNs) such as Long Short-Term Memory (LSTM) [1], [29], [34], [39], [44], [45] and Gated Recurrent Unit (GRU) [11], [30], [38], [45], [46], possibly exploiting the composition capabilities of hybrid DL architectures [1], [30], [44]. The way input data are fed to such architectures is paramount for taking full advantage of the DL paradigm. Unfortunately, some works [12], [13], [14], [41] counter-productively conduct a preliminary ad-hoc feature extraction and *do not exploit the key advantage of DL* of automatic extracting knowledge from raw traffic data without the need of human-expert intervention. Conversely, other works [37], [40], [44], [45] feed DL models with input data containing traffic-trace metadata (e.g., timestamps or ad-hoc IDs) or biased fields (e.g., local IP addresses or source/destination ports) which are likely to *introduce bias and misleadingly inflate TC performance*. Differently, in the present work, we exploit *unbiased raw input data* providing different views on traffic (i.e., payload bytes and packet sequence information) to take advantage of its intrinsic heterogeneity via multimodal approaches.

Indeed, even when using unbiased raw traffic data as input, most previous studies still *partially capitalize* the capabilities of DL by leveraging myopic single-modal architectures, while only few works exploit the intrinsic heterogeneity of network traffic data by means of *multimodal proposals* [11], [29], [30], [38], [39], [42], where the lower layers of the deep architecture are trained on heterogeneous subsets of input data with the aim of learning intra- and inter-modality dependencies.

On the other hand, a larger number of works propose *multitask DL models* which make use of a shared representation to improve their learning ability. Specifically, they train multiple tasks in parallel and exploit for each task the information in the training signals of related ones. Multitask approaches have been used to jointly tackle various problems associated with as many operators' needs: (i) malware detection/classification and TC [12], [40], [41], (ii) traffic classification and prediction [14], [34], (iii) different TC tasks, such as classification of duration, flow rate, and application (with the former two preprocessed via a two-level quantization) [13], classification of VPN-encapsulated traffic at different granularity [11], or joint classification of application, application category, user operation, operating system, and browser [45]. Moreover, the multitask paradigm has been applied in the context of federated learning [12], transfer learning [13], [14], and one-shot learning [13]. We underline that all the works proposing multitask architectures—with only one exception [34]—leverage such a learning paradigm in a supervised manner, namely the solution of the considered tasks (regardless of the specific task dealt with) is taken into account when training the layers of the shared representation. We point the interested reader to our recent work [11] for a more in-depth analysis of the state-of-the-art concerning TC via multitask DL.

B. XAI for Networking

As **AI techniques** have been increasingly adopted **to tackle networking tasks** in recent years, networking researchers are starting to explore XAI techniques to make AI models interpretable, trustworthy, and manageable [10]. Table I highlights the main aspects of recent works facing such networking tasks in the light of interpretability (e.g., anomaly detection [18], resource allocation [19], global and local network control [22], and various networking-related prediction tasks [20], [21], [23], [24], [25]), with a specific focus on TC whose related literature is clustered in the bottom half of the table. It is worth noting that Tab. I does not report the other works dealing with multitask TC described in Section II-A *because no XAI technique is applied in them* (with the sole exception of [11], containing a calibration analysis of the proposed multitask TC classifier). Still, these works are later reported in Tab. III and considered as baselines in the experimental phase.

AI techniques applied belong to both supervised and unsupervised Machine Learning (ML) and DL approaches, or they are reinforcement learning solutions. Consequently, the considered **traffic object** (i.e., the relevant elementary samples of analysis) strongly depends on the specific networking task—with biflows dominating the studies performing TC—while multimodal (**MM**) or multitask (**MT**) architectures are exploited only to face traffic prediction and classification.

Focusing on the specific **XAI methodology** adopted, we can notice that the related works mostly apply *interpretability techniques* to provide *post-hoc explanations* [30] in various forms. These include (i) different types of *perturbation analyses* (e.g., occlusion analysis [26], [29] or universal perturbation attacks [31]), (ii) attribution based on *Shapley values* [23], [30], [39] (iii) *Layer-wise relevance propagation* [18], (iv) *Interpretable local surrogates* [20], [21], [23], and (v) *Integrated or smoothed gradients*. Other approaches (e.g., saliency/feature maps, t-SNE) are based on *visual representations* [19], [27] that help to highlight the most important "features"—and ultimately the portion of input data—that led to the classification outcome (e.g., by inspecting the activation of intermediate neurons in hidden layers). *Easier-to-interpret models* than DL ones have been also exploited, namely performing distillation toward (naturally interpretable) decision

TABLE I
RELATED WORKS IN THE NETWORKING DOMAIN APPLYING XAI METHODOLOGIES FOR DIFFERENT GOALS (IMPROVING MODEL PERFORMANCE, TRUSTWORTHINESS, AND FEASIBILITY). THE SECOND PART OF THE TABLE GROUPS THE PAPERS TACKLING TRAFFIC CLASSIFICATION. IN EACH GROUP, THE PAPERS ARE REPORTED IN CHRONOLOGICAL ORDER. THE MEANING OF ACRONYMS AND SYMBOLS IS SHOWN AT THE BOTTOM OF THE TABLE

Paper	**Networking Task**	**TO**	**MM**	**MT**	**AI Technique**	**XAI Methodology**	**P**	**T**	**F**
Amarasinghe et al. [18], 2018	Anomaly detection	Biflow			DNN	LRP	✓	✓	
Zheng et al. [19], 2018	Resource allocation	Job			DNN	AM, Saliency Maps	✓	✓	✓
Dethise et al. [20], 2019	Video quality prediction	Video chunk			A3C (Pensieve)	LIME	✓	✓	
Morichetta et al. [21], 2019	Video quality prediction	Video session			HAC-Ward, HAC-Single, K-means, BIRCH	LIME		✓	
Meng et al. [22], 2020	Global and local control	Video chunk Flow Src-dst path			A3C (Pensieve), DNN (AuTO), GNN (RouteNet)	DT-Distillation, Hypergraph-Distillation, LIME*, LEMNA*	✓	✓	✓
Terra et al. [23], 2020	SLA violation prediction	5G e2e flow			XGBoost	XGBoost, SHAP, PI, CD, LIME, Eli5		✓	
Aceto et al. [24], 2021	Traffic prediction	Biflow	✓		HMM, MC	Markovian-Distillation		✓	
Montieri et al. [25], 2021	Traffic prediction	Biflow	✓	✓	CNN, LSTM, GRU, SeriesNet, DSANet	Markovian-Distillation		✓	
Rezaei et al. [26], 2019	Traffic classification	Biflow			1D-CNN, 1D-CNN+LSTM	Occlusion Analysis	✓		
Beliard et al. [27], 2020	Traffic classification	Biflow			1D-CNN	Feature Maps, t-SNE		✓	
Wang et al. [28], 2020	Traffic classification	Biflow			SDAE, 1D-CNN, LSTM	Deep SHAP		✓	
Aceto et al. [11], 2021	Traffic classification	Biflow	✓	✓	DISTILLER-ORIGINAL	Calibration Analysis		✓	
Akbari et al. [29], 2021	Traffic classification	Biflow	✓		1D-CNN & LSTM & DNN	Occlusion Analysis	✓		
Nascita et al. [30], 2021	Traffic classification	Biflow	✓		MIMETIC-ENHANCED	Deep SHAP, Calibration Analysis (improved via FL & LS)	✓	✓	
Sadeghzadeh et al. [31], 2021	Traffic classification	Packet Biflow			1D-CNN	UAP	✓		
Fauvel et al. [32], 2022	Traffic classification	Biflow			CNN+LERes+LProto	Explainable-by-design, SHAP*, Grad-CAM*	✓	✓	✓
This Paper	Traffic classification	Biflow	✓	✓	DISTILLER-EVOLVED† DISTILLER-ORIGINAL* 1D-CNN*, 2D-CNN*, LSTM*, HYBRID*	Deep SHAP, Integrated Gradients, Calibration Analysis (improved via FL & LS)	✓	✓	✓

TO: Traffic Object; **MM**: MultiModal; **MT**: MultiTask; **P**: XAI for improving Performance; **T**: XAI for improving Trustworthiness; **F**: XAI for improving Feasibility.
AI Technique: Asynchronous Advantage Actor-Critic (A3C), Bidirectional Gated Recurrent Unit (BiGRU), Convolutional Neural Network (CNN), Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH), Deep Neural Network (DNN), Extreme Gradient Boosting (XGBoost), Gated Recurrent Unit (GRU), Graph Neural Network (GNN), Hidden Markov Model (HMM), Hierarchical Agglomerative Clustering (HAC), Lightweight & Efficient Residual Block (LERes), Lightweight Prototype layer (LProto), Long Short-Term Memory (LSTM), Markov Chain (MC), Stacked Denoising AutoEncoder (SDAE); DISTILLER-ORIGINAL: MM-MT 1D-CNN & BiGRU; MIMETIC-ENHANCED: MM Embeddings & 1D-CNN & BiGRU.
+ symbol indicates hybrid architectures; & symbol indicates intermediate fusion of input data.
XAI Methodology: Activation Maximization (AM), Causal Dataframe (CD), Decision Tree (DT), Focal Loss (FL), Integrated Gradients (IG), Local Explanation Method using Nonlinear Approximation (LEMNA), Label Smoothing (LS), Local Interpretable Model-Agnostic Explanations (LIME), Layer-wise Relevance Propagation (LRP), Permutation Importance (PI), SHapley Additive exPlanations (SHAP), t-Distributed Stochastic Neighbor Embedding (t-SNE), Universal Adversarial Perturbation (UAP).
✓ present; * baselines for performance comparison; † all proposed variants not expressly indicated (see Sections V for details).

trees [22], [23], hypergraphs [22] or Markov models [24], [25]. These gray-box models despite being more simple and interpretable than DL, usually show poorer performance and need carefully hand-crafted features. Going further, solutions aiming at *explainability-by-design* based on the detection of class prototypes and communication to the end-user are also investigated [32].

Finally, the *reliability* of DL outcomes is also investigated to check whether such probabilistic outputs are *calibrated*, namely to verify whether the confidence actually reflects the reliability of the final decision. Techniques for both evaluating calibration and improving it—to make the DL models more reliable—are adopted in conjunction with TC tasks [11], [30].

According to our review, the considered body of literature applies XAI methodologies to different extents, aiming at improving the proposed solutions in one or more aspects (see the three rightmost columns in Tab. I). Specifically, XAI is used to improve (i) model performance (**P**): it provides means to scrutinize the model for revealing bias/variances and discerning whether the decisions are derived from the intended portions of input data, thus allowing model performance improvement as well as robustness and vulnerability assessment (e.g., susceptibility to *adversarial attacks*); (ii) trustworthiness (**T**): as humans are reluctant to trust decisions made by AI-based solutions without proper insights into their internal mechanisms, XAI techniques can expedite the validation of functional coherence, constraints violation, and policy obligations, and can make decisions and recommendations more trustworthy; (iii) feasibility (**F**): it can assist model refinement in order to allow these models to be accommodated by resource-constrained network devices.

Hereinafter, we focus on XAI techniques applied to interpret, refine, and improve solutions that tackle the problem of **network-traffic classification**. Beliard et al. [27] propose a platform to graphically visualize the inference process of a TC engine based on CNNs. Wang et al. [28] use Deep SHAP [15] to explain a few representative outcomes obtained through a 1D-CNN for mobile-app TC. Rezaei et al. [26] perform an occlusion analysis that allows to inspect how the CNN model proposed for the classification of mobile-app traffic can classify SSL/TLS flows, revealing that certain handshake fields can leak the information exploited in the TC process. Sadeghzadeh et al. [31] tackle the robustness of DL-based TC models against adversarial samples, and demonstrate that they are vulnerable to universal adversarial perturbation. Fauvel et al. [32] propose an explainable-by-design CNN for TC which also fulfills lightweight and efficiency purposes. Akbari et al. [29] perform a feature engineering study that considers exclusively encrypted Web traffic in which they occlude parts of the input that allow the DL model to learn a lazy and unsophisticated logic.

Concerning reliability of TC results, in a recent work [11] we perform a calibration analysis for a proposed multimodal multitask DL architecture (but with no attempt in improving it). Differently, in another work [30] we investigate interpretability and reliability to improve the behavior of state-of-the-art multimodal DL traffic classifiers by applying global-interpretation XAI techniques based on Deep SHAP and methods to both assess and improve the reliability of classifiers.

C. Positioning of This Work

In this work, we exploit XAI methodologies to enhance state-of-the-art multimodal multitask TC solutions along the three previously mentioned dimensions: model performance, trustworthiness (via improved reliability), and feasibility. Such solutions are clearly appealing because they can tackle several TC problems simultaneously (multitask) by inspecting the traffic from complementary viewpoints (multimodal). Particularly, we leverage XAI methodologies to directly provide the explanation (i.e., evaluate the importance) of raw traffic data and not of manually-extracted features, a task not only more challenging but also more useful for shedding light on the black-box nature of complex DL-based traffic classifiers.

Concerning *interpretability* techniques, differently than our previous work [30] and most of the surveyed literature tackling TC—and similarly to Terra et al. [23]—we apply multiple XAI approaches (Deep SHAP and Integrated Gradients) to compare and cross-validate the outcomes of these techniques. Furthermore, unlike Fauvel et al. [32]—which also tackle the problem of enhancing performance, trustworthiness, and feasibility in the context of TC—we do not subvert the nature of the original DL architecture for explainability purposes (along explainable-by-design principles) but rely on *post-hoc* analyses that can be applied to any black-box ML/DL-based solution previously proposed.

Moreover, we exploit methods to assess (*reliability diagrams* and related metrics [47]) and improve (*focal loss* [48] and *label smoothing* [49]) the *reliability* of traffic classifiers. The latter methods enhance the generalization capability of classifiers through the reduction of excessive confidence associated with predictions and consequently the possibility of overfitting. Also, by means of the reliable confidence score coupled with the classification outcome, actionable context is provided to the (human or automated) user.

Concerning *feasibility*, our investigation on model compression covers and compares a number of approaches (i.e., knowledge distillation, pruning, and quantization) providing a more systematic study than previous works.

Finally, similarly to Aceto et al. [11], we employ a *multimodal multitask DL* architecture. In detail, we capitalize on the general DISTILLER framework proposed in the latter work and extend it by considering additional or (partially-)unexplored aspects to overcome different limitations of its very first realization leveraged in [11], here named DISTILLER-ORIGINAL. In doing so, through a process of sequential enhancements, each associated with a different realization (see Section V for details on every novel realization), we come to the final definition of DISTILLER-EVOLVED. Enhancements are here meant from different viewpoints, namely classification performance, reliability, and feasibility. Indeed, these aspects have been either improved w.r.t. [11] (i.e., performance and reliability) or addressed for the first time here (i.e., feasibility). Overall, to the best of our knowledge, the present paper provides **the first attempt in interpreting and enhancing this kind of multimodal multitask DL architectures** with the support of XAI methodologies.

III. MULTIMODAL MULTITASK DEEP LEARNING–BASED EXPLAINABLE TRAFFIC CLASSIFICATION

In this section, we describe our general methodology for interpreting and designing improved multimodal multitask DL-based traffic classifiers. Specifically, in Section III-A, we refresh the general DISTILLER framework recalling its peculiar characteristics. Then, in Section III-B, we introduce the concept of interpretability in DL architectures and describe our approach based on Deep SHAP and Integrated Gradients techniques. In Section III-C, we motivate the role of reliability and introduce metrics to assess—and techniques to improve—the calibration of DL-based TC approaches. Finally, Section III-D describes model compression techniques capitalized in this work.

A. Multimodal Multitask DL-Based Traffic Classification

Herein, we recall the DISTILLER framework [11] we exploit for encrypted TC via *multimodal multitask DL* and the corresponding training procedure adopted for each classifier based on it. Details of each instance are reported in later Section IV-B.

Generally speaking, *multimodal* DL is able to automatically learn a hierarchical representation of traffic data by jointly exploiting multiple “views” (viz. modalities) of the same traffic object, for instance: raw bytes of payload data and informative protocol fields of packet sequences. Additionally, *multitask* learning improves the ability to tackle a given network visibility task (e.g., classifying the mobile app generating each flow) by exploiting the information distilled from other related tasks (e.g., predicting the average packet length of each flow and detecting mice/elephant flows), particularly by learning them in parallel via a shared representation. It allows a DL model to reduce redundancy and computational overhead by leveraging a one-comprehensive model while providing better generalization and classification performance.

1) DISTILLER *Overview:* Our goal is solving $v = 1, \dots, V$ different related TC tasks, for example inferring the traffic-type and the particular application generating a traffic object. Formally, given a traffic object (i.e., a subset of network packets sharing some common properties and constituting our TC sample), the v^{th} TC task (T_v) consists in assigning a label among L_v classes (e.g., apps or services) within the set $\{1, \dots, L_v\}$. When tackling multitask TC, each traffic object is labeled with as many labels as the TC tasks to be solved. We define the m^{th} traffic object of the training set (encompassing M samples) as $\boldsymbol{x}(m)$, while the corresponding label of the v^{th} classification task as $\ell^v(m)$. Such a label may belong to one out of the L_v different classes, namely $\ell^v(m) \in \{1, \dots, L_v\}$.

To fully exploit the highly-structured information contained in each sample $\boldsymbol{x}(m)$, we distill such information via a multimodal DL architecture. A multimodal DL architecture leverages different data types (e.g., header fields or payload bytes) to capitalize complementary views or modalities of the same traffic object exploiting an advanced form of information fusion—named *intermediate fusion*—for capitalizing the heterogeneity of network traffic data when solving multiple (related) tasks in parallel.

2) DISTILLER *Architectural Definition:* The generic DISTILLER architecture is made of P different *modalities* (each corresponding to a different input type). The first part of such a framework consists of a certain number of input-specific *single-modality* layers, which extract the discriminative features distilling the *intra-modality* dependencies of the p^{th} modality. On top of these layers, such features are fused via a *merge layer*, which is in charge of channeling the modality-specific features in a joint multimodal multitask (shared) representation. The second part of the DISTILLER framework is made of some *shared-representation* layers followed by *task-specific* layers. The former layers extract the features distilling *inter-modality* dependencies; the latter layers synthesize the task-oriented features (of the v^{th} task) from the shared ones. The DISTILLER architecture is terminated with one `softmax` layer for each TC task to be solved.

3) DISTILLER *Training Procedure:* We train DISTILLER via a *two-stage procedure*: a preliminary *pre-training* to distill the features of each single-modality branch and a successive *fine-tuning* of the whole DISTILLER architecture. In more detail, when performing pre-training, each single-modality branch is topped with V softmax "stubs".[2] In this case, a *weighted sum* of the *categorical cross-entropy* of each TC task is minimized for promoting the capability of the p^{th} modality to solve V different TC tasks *alone*. Since DISTILLER solves multiple learning tasks in parallel, each weight represents the preference level of the v^{th} task in the multitask categorical cross-entropy to be minimized. For the *fine-tuning*, we remove the softmax stubs and train the whole DISTILLER architecture, i.e., by introducing the shared-representation and task-specific layers. Nevertheless, during fine-tuning, the "lowest" single-modality layers (i.e., those aimed at intra-modality feature extraction) are frozen, namely their weights keep the value learned during the pre-training. Both the categorical cross-entropy functions concerning pre-training and fine-tuning are minimized via standard first-order local optimizers.

[2]In the next Section III-B, we will show how the auxiliary outputs of the stubs are exploited to perform per-modality interpretation analysis.

B. Interpreting Multimodal Multitask DL Traffic Classifiers

The starting point for interpreting complex DL architectures is to consider a simpler explanation model $g(\cdot)$, which is designed to closely-approximate the original model $f(\cdot)$. In the present work, we focus on *local explanation methods*, which explain the original model $f(\boldsymbol{x})$ in the neighborhood of a particular *per-biflow* instance $\boldsymbol{x}$ using the so-called *simplified inputs* $\boldsymbol{x}'$ that map to the original ones through a mapping function $\boldsymbol{x} = \boldsymbol{h}_{\boldsymbol{x}}(\boldsymbol{x}')$. Per-sample explanation outcomes based on local methods are then aggregated to obtain global explanations, as illustrated at the end of this subsection.

The majority of interpretability techniques (e.g., LIME, LEMNA, DeepLIFT) assumes a peculiar functional form for the explanation model $g(\cdot)$ leading to the definition of *Additive Feature Attribution (AFA)*. Formally, AFA methods are linear functions of binary variables:

$$g(\boldsymbol{z}') = \phi_0 + \sum_{m=1}^{M} \phi_m \, z'_m \tag{1}$$

where $\boldsymbol{z}' \in \{0,1\}^M$, M denotes the number of simplified inputs, and $\phi_m \in \mathbb{R}$. Hence, they provide an explanation model associating an "effect" ϕ_m to each input: the original model output $f(\boldsymbol{x})$ can be approximated by summing the effects of all input attributions.

In the present work, we leverage *Deep SHAP* [15] and *Integrated Gradients*, both detailed hereinafter.

1) Deep Shap: The first way to compute AFA solutions is by means of the well-known *Shapley values*. The initial concept of Shapley values originates from cooperative game theory and specifies the contribution of player m to the payoff $v(\mathcal{P})$ achieved by the whole coalition $\mathcal{P}$ [50]. To this aim, the method assesses the payoff of *every subset* of cooperating players $\mathcal{S} \subset \mathcal{P}$ and evaluates the effect of removing or adding the player m to $\mathcal{S}$ on the total payoff $v(\mathcal{S})$ obtained by $\mathcal{S}$ if the players cooperate. When leveraging this method for the interpretation of a DL-based model, the input data correspond to the players of the cooperative game, and the output of the DL architecture $f(\boldsymbol{x})$ to the payoff function.

Since the exact computation of Shapley values grows exponentially with the input size M, we approximate them (in a lightweight form) via *SHapley Additive exPlanation (SHAP)* by eliminating the need to re-train the models. Specifically, SHAP approximates these values via the conditional expectation [15] formally defined as:

$$f\big(\boldsymbol{h}_{\boldsymbol{x}}(\boldsymbol{z}')\big) \approx \mathbb{E}\{f(\boldsymbol{z}) | \boldsymbol{z}_{\mathcal{S}}\} \tag{2}$$

where $\mathcal{S}$ denotes the set of non-zero indices within $\boldsymbol{z}'$.

We can further simplify the formulation in Eq. (2) by assuming the statistical independence of the inputs and the linearity of the model [15], formally: $f(\boldsymbol{x}) = \sum_{m=1}^{M} w_m x_m + b$. When both these hypotheses hold, the ϕ_m's are in *closed-form* and equal to $\phi_m(f, \boldsymbol{x}) = w_m\,(x_m - \mathbb{E}\{x_m\})$. We exploit the latter assumption to actually calculate the Shapley values.

In more detail, we leverage DeepLIFT [51] for the explicit computation of the SHAP values. DeepLIFT is an AFA recursive explanation method for the decisions of DL architectures, which attributes to each input x_m a value $C_{\Delta x_m \Delta o}$ representing the effect of that input being set to a reference value as opposed to its original value. Specifically, DeepLIFT capitalizes a linear composition rule for the calculation of the $C_{\Delta x_m \Delta o}$'s, which is based on the linearization of the non-linear components of a DNN, such as (soft)max, products, or divisions. The reference value $f(\boldsymbol{r})$ is a user-defined parameter typically chosen to be an uninformative background value for the m^{th} input. When setting $\phi_0 = f(\boldsymbol{r})$ in Eq. (1) the explanation model of DeepLIFT is compliant with the functional form of AFA methods and Shapley values represent the unique solution. Consequently, DeepLIFT can be used to obtain a compositional and fast approximation algorithm of Shapley values, named *Deep SHAP* [15]. We underline that since Deep SHAP relies on an approximate computation of Shapley values, the local accuracy property $f(\boldsymbol{x}) = g(\boldsymbol{x})$ may be not satisfied with perfect equality.[3]

2) Integrated Gradients (IG): The second interpretability technique we used in the present work is the *Integrated Gradients (IG)*. IG is based on the founding concept of input *baseline*. The underlying idea springs from the human inclination to make attributions based on counterfactual intuition, namely by implicitly comparing a certain effect (e.g., the decision of a classifier) against the absence of the effect. IG models such an "absence" using a single baseline input. In other words, we interpret the IG value ϕ_m as the importance value of the m^{th} input in moving the confidence relative to the same class with the baseline as input (along the i^{th} dimension for an input). Accordingly, IG is obtained by (*a*) considering the straightline path from the baseline ($\boldsymbol{x}'$) to the input ($\boldsymbol{x}$), (*b*) computing the gradients at all points along the path and (*c*) cumulating them. Hence, given an input $\boldsymbol{x} = \{x_1, \ldots, x_M\} \in \mathbb{R}^M$, the "effect" ϕ_m of each input x_m is given by:

$$\phi_m(\boldsymbol{x}) = \left(x_m - x'_m\right) \times \int_{\alpha=0}^{1} \frac{\partial p_i\left(\boldsymbol{x}' + \alpha\left(\boldsymbol{x} - \boldsymbol{x}'\right)\right)}{\partial x_m} d\alpha \quad (3)$$

According to the *completeness* axiom, all the effects add up to the difference between the output of the model relative to the i^{th} class, $p_i(\boldsymbol{x})$, at the input $\boldsymbol{x}$ and the baseline $\boldsymbol{x}'$:

$$\sum_{m=1}^{M} \phi_m(\boldsymbol{x}) = p_i(\boldsymbol{x}) - p_i\left(\boldsymbol{x}'\right) \quad (4)$$

Other axioms satisfied by this technique are: (*i*) *sensitivity*–for every input and baseline that differ in one feature but have different predictions, the differing feature should be given a non-zero effect and (*ii*) *implementation invariance*–the effects for two functionally-equivalent networks[4] are always identical.

3) From Local to Global Explanations: Hereinafter, Deep SHAP and IG methods are used to evaluate the (relative) importance of input data extracted from raw traffic (see Section IV for details) of a given traffic object when performing a certain TC task.[5] Therefore, to explain the predictive behavior of DL-based traffic classifiers, the prediction model $f(\boldsymbol{x})$ is chosen as the soft-output associated with the i^{th} class (regardless of the specific task considered), i.e., $p_i(\boldsymbol{x})$. Hence, we interpret the Deep SHAP or IG ϕ_m as the *importance value* of the m^{th} input in forming the confidence p_i associated with labeling the traffic sample (whose overall input is $\boldsymbol{x}$) with the i^{th} class.

It is worth noticing that ϕ_m can be also negative. Consequently, an importance value can be interpreted as follows: positive (negative) values increase (decrease) the confidence $p_i(\boldsymbol{x})$ in the classification of the i^{th} class w.r.t. its average $\mathbb{E}\{p_i\}$ or baseline $p_i(\boldsymbol{x}')$ value for Deep SHAP or IG, respectively. In more detail, for Deep SHAP, the sum of the importance values equals the considered soft-output value $(p_i(\boldsymbol{x}))$ minus the so-called *base output*. The latter represents the average of the same soft-output obtained in correspondence of the samples associated with the background set. Herein, for each traffic object, we focus on explaining the soft-output associated with the predicted class $\hat{p}(\boldsymbol{x})$, as this represents the most relevant (and highest) output for a given TC task.

The additive form of employed methods enables the evaluation of importance attributed to non-overlapping input subsets. This investigation fits evaluating to which degree the *different modalities* of a multimodal DL traffic classifier contribute to the interpretation of the decisions made (on a given TC task). Formally, we denote the input subsets associated with the P modalities as $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_P$, where $\boldsymbol{x} = \bigcup_{p=1}^{P} \boldsymbol{x}_p$. Then, to quantify the importance of the p^{th} modality to multimodal TC effectiveness (we omit in what follows the index associated with the v^{th} task), we resort to a *pooled* importance value $\phi_{\mathcal{M}_p}$. The latter represents the *importance value* of the input subset $\boldsymbol{x}_p$ (corresponding to the p^{th} modality) in classifying the traffic object associated with the overall input $\boldsymbol{x}$ with the label $\hat{\ell}$. The pooled importance value is obtained as $\phi_{\mathcal{M}_p} \triangleq \sum_{m \in \mathcal{M}_p} \phi_m$, where $\mathcal{M}_p$ denotes the index set associated with the p^{th} input subset within $\boldsymbol{x}$, and its interpretation with is analogous to that of the unpooled value ϕ_m.

On the other hand, to focus on a *given modality* and assess the related importance contribution of each *individual input*, we consider the *stub output* associated with the p^{th} modality as our $f(\cdot)$ only depending on $\boldsymbol{x}_p$. In such a way, we can exclusively focus on the behavior of the p^{th} single-modality branch, namely before the combined effect of intermediate fusion achieved by the shared-representation layers. This procedure isolates the interacting effect of other modalities on

[3]In our previous work [30], we have experimentally proven that only a *negligible discrepancy* (always lower than 1%) exists.

[4]Two functionally-equivalent networks have the same output for all inputs, despite different implementations.

[5]To simplify the notation, in the following, we avoid explicitly indicating the v^{th} task considered. Indeed, all the considerations are valid for all the TC tasks constituting our multitask formulation.

the p^{th} modality and permits *per-modality interpretation*. The (isolated) importance values associated with the input subset $\boldsymbol{x}_p$ (feeding the p^{th} modality) are represented by the importance values $\phi_m^{(p)}$, where $m = 1, \dots, |\mathcal{M}_p|$. In the latter case, before proceeding to the calculation of the importance values, we perform an *additional fine-tuning* of both single-modality branches topped with the stubs—which follows the conventional pre-training and fine-tuning phases performed to train the classifiers based on the DISTILLER framework. This further step does not affect the performance of classifiers but it is required to update the stub weights to reflect on them the changes made to the rest of the DL architecture during the (conventional) fine-tuning phase (see Section III-A).

Notably, a soft-output can assume a range of different values. Accordingly, the absolute importance of the m^{th} input may differ from sample to sample. Therefore, once we have obtained a local explanation for a single instance, our proposed *global explanation* approach relies on aggregating explanations over different samples $\boldsymbol{x}_1, \dots, \boldsymbol{x}_N$. The aggregation step is carried out on *normalized* importance values, obtained by dividing each value by their overall sum, namely:

$$\widetilde{\phi}_m \triangleq \phi_m / \sum_{m=1}^{M} \phi_m. \tag{5}$$

Considering $\widetilde{\phi}_m$ allows focusing on the *relative importance* of each input (indeed, for each sample, the sum of the importance values equals one). Additionally, as in [28], [30], we aggregate only on *correctly-classified samples* to focus on the correct behavior of considered classifiers and to allow to interpret their counter-intuitive (while right) decisions a posteriori.

The above methodology for interpreting the behavior of multimodal multitask DL traffic classifiers will be applied to the models based on DISTILLER framework described in Section III-A. Specifically, to obtain the global explanations pertaining to different granularities, we will consider the following views of aggregation: (*i*) over the whole ISCX VPN-NONVPN dataset and (*ii*) related to certain classes of a given task (e.g., P2P, FTPS). Nevertheless, we underline that the proposed interpretability approach is general and can be potentially applied to any multitask and/or multimodal DL-based architecture exploited for TC.

C. Calibration in Multitask Deep Learning

As discussed in Section II, it is of paramount importance to assess the *reliability* of (traffic) classifiers, which is one of the considered pillars of XAI. Specifically, we evaluate the reliability of a classifier in providing TC decisions with high confidence (or not), namely if they are calibrated (or not). Additionally, given the multitask nature of traffic classifiers considered, we assess how reliability on a given task (i.e., on a specific network visibility problem) is affected by correct/wrong decisions on other tasks.

Formally, given the generic input sample $\boldsymbol{x}$ fed to the multitask DL traffic classifier, we investigate the reliability of the whole confidence vector (i.e., $\boldsymbol{p}^v(\boldsymbol{x}) = p_1^v(\boldsymbol{x}), \dots, p_{L_v}^v(\boldsymbol{x})$) and of the confidence associated to the predicted class (i.e., $\hat{p}^v(\boldsymbol{x}) = \max_{i=1,\dots,L_v} p_i^v(\boldsymbol{x}))$ when solving the v^{th} task. In detail, a *confidence-calibrated* multitask classifier is such that for each sample, the confidence of a prediction $\hat{p}^v$ related to the v^{th} task equals $\Pr\{\hat{\ell^v} = \ell^v | \hat{p^v}\}$, where ℓ^v is the actual class and $\hat{\ell^v}$ is the predicted one. On the other hand, a *miscalibrated* classifier returns excessively optimistic (or pessimistic) confidence outputs associated with its decisions.

To illustrate this property when varying $\hat{p}^v$, we exploit the *reliability diagrams* [47], which depict the accuracy as a function of the confidence (i.e., $\Pr\{\hat{\ell^v} = \ell^v | \hat{p}^v\}$ vs. $\hat{p}^v$) for each of the V different tasks. The so-obtained diagram is commonly compared with the ideal $\Pr\{\hat{\ell^v} = \ell^v | \hat{p^v}\} = \hat{p^v}$ identity line: a perfectly-calibrated classifier has a reliability diagram corresponding to the identity function. Going into detail, a reliability diagram is obtained by partitioning the predictions into M equally-spaced bins (with width $1/M$) and calculating the accuracy of each bin. Let B_m be the set of evaluated samples such that the confidence associated with the predicted class falls into the range $I_m \triangleq (\frac{m-1}{M}; \frac{m}{M}]$, the corresponding bin-accuracy equals:

$$\mathrm{acc}(B_m) = |B_m|^{-1} \sum_{n \in B_m} 1\left(\hat{\ell^v}(n) = \ell^v(n)\right) \tag{6}$$

where $\ell^v(n)$ and $\hat{\ell^v}(n) \triangleq \arg\max_{i=1,\dots,L_v} p_i^v(n)$ are the true and predicted labels of the v^{th} task for the n^{th} sample, respectively. Confidence values range in the interval $[1/L_v,\, 1]$, where L_v is the number of classes of the v^{th} TC task. Hence, the starting point of the confidence interval is $1/L_v$.

We complement the reliability diagrams with the *Expected Calibration Error (ECE)*, a concise metric measuring the deviation from a perfect calibration. The ECE for the v^{th} task is defined as $\mathbb{E}_{\hat{p}^v}\{|\Pr\{\hat{\ell^v} = \ell^v | \hat{p}^v\} - \hat{p}^v|\}$ and expresses the expected absolute deviation between confidence and confidence-conditional accuracy. We approximately calculate it using the formula:

$$\mathrm{ECE} \approx \sum_{m=1}^{M} (|B_m| / N) \, |\mathrm{acc}(B_m) - \mathrm{conf}(B_m)| \tag{7}$$

which depends on the total number of tested samples N and the confidence averaged within the bin B_m, obtained as $\mathrm{conf}(B_m) = |B_m|^{-1} \sum_{n \in B_m} \hat{p}^v(n)$. In the last term $\hat{p}^v(n) \triangleq \max_{i=1,\dots,L_v} p_i^v(n)$ denotes the predicted confidence of the v^{th} task for the n^{th} sample.

In what follows, other than analyzing the calibration of each TC task separately, we also investigate *how the reliability of a task is affected by the other tasks*. To do so, we generalize the above concepts to the *task-conditional reliability diagram* ($\Pr\{\hat{\ell^v} = \ell^v | \hat{p}^v, \mathrm{tsk}\}$ vs. $\hat{p}^v$) and *task-conditional ECE* ($\mathbb{E}_{\hat{p}^v | \mathrm{tsk}}\{|\Pr\{\hat{\ell^v} = \ell^v | \hat{p}^v, \mathrm{tsk}\} - \hat{p}^v|\}$), where tsk denotes the generic task-conditional intersection of classification events on the other tasks, defined as:

$$\mathrm{tsk} \triangleq \bigcap_{v^\star = 1, \dots, V;\; v^\star \neq v} \left(\hat{\ell}^{v^\star} \underset{\neq}{=} \ell^{v^\star} \right) \tag{8}$$

The above notation takes into account compactly all the combinations of correct $(=)$ and wrong $(\neq)$ classification events on all the tasks except v. Clearly, the task-conditional reliability diagram and ECE can be both approximated similarly as the unconditional case. Task-conditional calibration analysis is useful in multitask DL traffic classifiers as it allows understanding if (and, in affirmative case how much) soft-outputs may become over-optimistic (or over-pessimistic) due to a correct/wrong decision on a related network visibility task.

To improve the calibration (and consequently the reliability) of a DL traffic classifier we resort to the *label smoothing*[6] technique which is a type of loss regularization aiming at improving the generalization ability of DL models and reducing an overly high prediction confidence [49]. More in detail, during the training phase, the label smoothing dictates that the cross-entropy loss minimizes the prediction w.r.t. a *smoothed* one-hot representation of the ground truth for the v^{th} task $\ell^v(n)$, computed as

$$\boldsymbol{t^v}_{\mathrm{ls}}(n) = (1-\alpha)\,\boldsymbol{t^v}(n) + \frac{\alpha}{L_v}\,\mathbf{1}_{L_v} \tag{9}$$

where $\boldsymbol{t^v}(n) \triangleq \left[t_1^v(n) \quad \cdots \quad t_{L_v}^v(n)\right]^T$ is the one-hot representation of the label $\ell^v(n)$. The smoothing parameter α defines the amount of uncertainty enforced on the ground truth, with $\alpha \to 0$ collapsing to the usual non-smoothed cross-entropy-based training procedure.

D. Deep Learning Model Compression

As mentioned earlier, another key objective of our analysis is to assist model refinement to enable the deployment of DL architectures in resource-constrained environments such as on network devices. This is also one of the main reasons underlying the design of a (single) multitask DL architecture to solve multiple related TC tasks [11]. Specifically, we aim at compressing (multitask) DL models while limiting any potential loss in model "quality" (in terms of TC performance and calibration on all the tasks). The techniques we have individually considered for performing such model compression are *pruning*, *quantization* and *knowledge distillation* [52]. Moreover, we investigate the results obtained when applying quantization to a pruned model. Compression techniques considered are briefly described in the following.

1) Knowledge Distillation: it has been initially proposed for compressing the knowledge of an ensemble of models into a single one [53]. By extension, the same methodology can be applied to train a smaller model ("the student") to imitate the (soft) predictions of a larger (and more accurate) pre-trained model ("the teacher").

The underlying idea is to exploit the soft outputs of the teacher model to train the student for capitalizing on the high informative value of soft outputs in combination with the common hard decisions. Indeed, to attain the best performance, a balance—expressed by a λ factor—between two loss functions should be reached, taking into account both soft and hard outcomes. In detail, such loss functions are: (i) the cross-entropy between one-hot encoded labels and student hard decisions and (ii) the cross-entropy[7] between teacher and student soft outputs. To effectively convey more and even better information to the student, the soft outputs and hard decisions are obtained from the logits scaled by a factor *T*, called *temperature*. The data used to train the student model is the so-called *transfer set*: in this work, we make the common choice of using the entire training set. Moreover, we train the student model for as many epochs as the teacher model since longer training should be beneficial for performance [55].

Since both the teacher and student architectures are based on the DISTILLER framework, we apply knowledge distillation to both pre-training and fine-tuning phases. Hence, to distill knowledge from the single-modality branches during their pre-training, we consider the stub outputs of the teacher model to obtain its hard and soft outcomes.[8]

2) Pruning: it involves purging connections between neurons or some neurons altogether, that contribute less to the performance of the model. This procedure potentially improves the inference time and energy efficiency of models having sparse connection matrices: hence it is expected to work well with DL-based traffic classifiers. In this work, we exploit the *gradual pruning* approach [56], in which the sparsity increases from an initial value S_i (usually zero) to a final value S_f, over a span of *n* pruning steps. In detail, this method introduces a binary mask variable in each layer to be pruned, having the same size and shape of the layer weights, and specifying the weights that contribute to the forward execution of the optimization procedure. The aim is to mask to zero the smallest magnitude weights until a specified sparsity level S_f is reached: zeroed weights do not get updated in the back-propagation step. In our analyses, pruning is applied to either (*i*) an already-trained model or (*ii*) to a model to be trained (i.e., within the training phase).

3) Quantization: it approximates a DL model that employs floating-point values with a model using lower precision data-types for storing model weights and performing computation. As a result, the memory requirements and computing costs of the quantized model are drastically reduced by limiting the number of bits (viz. resolution) to represent each trainable parameter. More specifically, we apply *post-training quantization* [57] which is a desirable (and popular) compression strategy not requiring re-training of the DL model (or access to the whole training set) and thus circumventing the usual difficulties in performing such an activity (e.g., lack of computing-resource or data). The simplest form of post-training quantization statically approximates the weights of an already-trained model, e.g., from 32-bit to 16-bit floating point numbers (or to 8-bit precision integers).

[6] We have also tried to employ the *focal loss* (in place of the cross-entropy loss) function which is commonly used to deal with class imbalance, but that can be leveraged also to improve calibration by capitalizing its implicit regularization properties [30]. Unfortunately, using the focal loss (even in an optimized fashion) we have attained a notable deterioration of performance, which majorly impacts the hardest TC task. Moreover, the results (whose details are omitted for brevity) suggest its weaker impact on calibration when compared with label smoothing.

[7] Other functions (e.g., Kullback–Leibler divergence or focal loss [54]) can be also used for computing the loss between student and teacher soft outputs, however, these functions lead to worse results than the common cross-entropy loss function.

[8] We have also tested other scenarios performing distillation either only during the pre-training or the sole fine-tuning, without substantial differences in performance.

IV. Experimental Setup

This section describes the experimental setup considered in this work. Specifically, in Section IV-A a brief description of the ISCX VPN-nonVPN dataset is provided. Then, we introduce in Section IV-B the specific DL-based traffic classifiers originating from the general Distiller framework and the multitask baselines used for comparative performance. Finally, in Section IV-C, we report implementation details for reproducibility.

A. Dataset Description

For our experiments, we used the ISCX VPN-nonVPN dataset [17] collected at the Canadian Institute for Cybersecurity. It contains human-generated traffic related to different *traffic types* and *applications* and collected through sessions both *regular* and *encapsulated over VPN*. The dataset is provided in raw (PCAP) format, and the ground-truth is generated at trace-level: each PCAP trace corresponds to an encapsulation technique (VPN or nonVPN), a traffic type (6 classes), and a specific application (15 classes). For this reason, it is possible to associate a *three-view* label (i.e., *encapsulation*, *traffic type*, and *application*) to any traffic object and for each of them define *three TC tasks* to be tackled. As the vast majority of papers tackling (multitask) TC works with either unidirectional or bidirectional flows (viz. biflows), we segment the raw traffic collected in the ISCX VPN-nonVPN dataset in biflows. A biflow is defined as the set of packets sharing the same quintuple (`Src_IP, Src_Port, Dst_IP, Dst_Port, L4 Protocol`) where source and destination IP addresses and ports of the quintuple can be swapped [1].

Notably, analyzing the dataset, we have found that $\approx 65\%$ of biflows extracted from ISCX VPN-nonVPN raw traffic data have *only one UDP packet* with (`Dst_IP, Dst_Port`) equal to (`255.255.255.255, 10505`). After further inspection, we have found that these packets are network broadcasts periodically sent by BlueStacks, an Android emulator for PCs.[9] Moreover, we also noted some biflows pertaining to certain protocols (e.g., SNMP, Dropbox LanSync Protocol, BOOTP) used in LANs for different purposes not strictly related to the traffic types or applications constituting the dataset. Therefore, as opposed to the other works leveraging ISCX VPN-nonVPN as is, we carried out careful preprocessing cleaning operations to remove this noisy traffic and thus obtain more meaningful results.

As a consequence, the final dataset contains $\approx$ 10.5k biflows whose distribution among the different classes for each task is shown in Fig. 1. In the same figure we also highlight (with different hatches) the portion of `VPN` and `nonVPN` traffic for each class. For the applications, in almost all the cases (12 out of 15), the traffic type is *unique*: `Aim` and `ICQ` (100% `Chat`), `Email` (100% `Email`), `Spotify`, `Netflix`, `Vimeo`, `VoipBuster` and `YouTube` (100% `Streaming`), `Torrent` (100% `P2P`), `FTPS`, `SFTP` and `SCP` (100% `File Transfer`). However, there are also apps for which the situation is different as they represent a

[9]https://www.bluestacks.com/

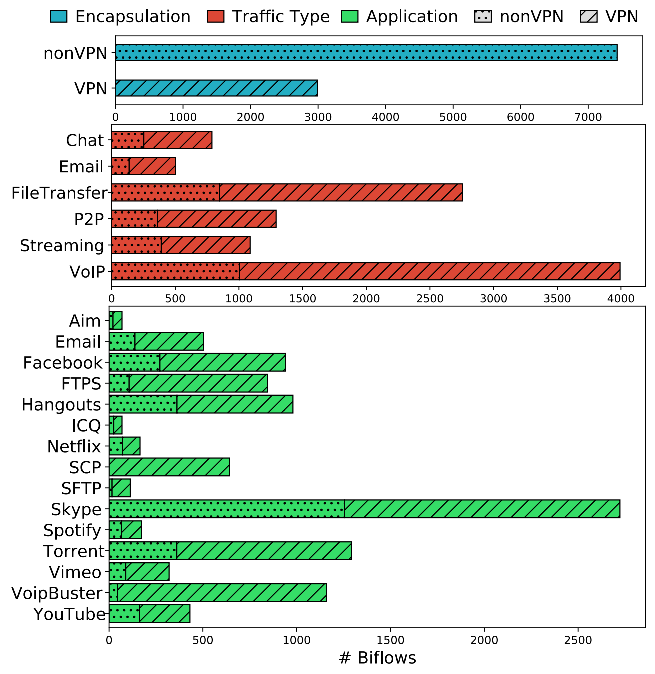


Fig. 1. Number of per-class biflows for each ISCX VPN-nonVPN dataset task. Hatches underline the number of nonVPN and VPN biflows for each class.

mix of traffic types: `Facebook` (16% `Chat`, 84% `VoIP`), `Hangouts` (28% `Chat`, 72% `VoIP`), `Skype` (8.4% `Chat`, 42.5% `FileTransfer`, 49.1% `VoIP`).

Finally, we also report (as a complementary viewpoint) that the distribution of ISCX VPN-nonVPN in terms of protocols[10] is as follows: 33.76% TLS, 37.79% UDP:DATA, 15.23% HTTP, and the remaining 13.22% associated with gQUIC, STUN, SSH, TCP:DATA, and other undetected protocols.[11]

B. Multitask Traffic Classifiers

Hereinafter, we describe the traffic classifiers we have investigated in this work, based on the general multimodal multitask Distiller framework introduced in Section III-A, along with the other baselines considered for performance comparison. Given the ISCX VPN-nonVPN dataset, we tackle a multitask TC problem with $V = 3$ TC tasks: (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).

As mentioned in Section IV-A, aiming at a consistent comparison with the state-of-the-art, all considered models operate with *biflow traffic objects* and use the same inputs. Specifically, the latter are chosen among those proposed in the most related (recent) literature based on a preliminary investigation (not shown for brevity): we consider the first N_b bytes of transport-layer payload arranged in byte-wise fashion (`PAY` input type)

[10]In detail, the Python wrappers `PyShark` (https://kiminewt.github.io/pyshark) and `Scapy` (https://scapy.net/) were employed.

[11]TCP:DATA and UDP:DATA indicate that the payload of the corresponding TCP or UDP packets, respectively, can not be identified more exactly (e.g., unavailable protocol dissector, non-standard ports used, missing beginning of the communication).

or informative unbiased fields extracted from the sequence of the first N_p packets (`PSQ` input type), namely (i) the number of bytes in the transport-layer payload (`PL`), (ii) the direction (`DIR`) $\in \{0,1\}$, (iii) the TCP windows size (`TCP_WS`) equal to 0 for UDP packets, and (iv) the time elapsed since the arrival of the previous packet, i.e., the inter-arrival time (`IAT`). More in detail, we set $N_b = 784$ bytes and $N_p = 32$ packets—by truncating longer samples and zero-padding shorter ones—and normalize both input data within [0, 1].[12]

We underline that we leverage neither biased inputs (e.g., raw PCAP metadata encompassing timestamps or *ad-hoc* IDs and other biased fields as local IP addresses or source/destination ports) which could wrongly inflate performance leading to misleading outcomes [1], nor manually-extracted features (e.g., statistics extracted on the sets of packet/payload lengths or inter-arrival times) to fully exploit the benefits of DL, namely the possibility of working directly with raw traffic significantly limiting human-expert intervention. It is worth noting that `PAY` input could be possibly affected by the ratio of cleartext data (e.g., encryption based on TLS 1.2 vs. 1.3), while `PSQ` input by network-specific conditions, application-specific behaviors, and OS-specific patterns. Nevertheless, in this regard, XAI allows us to trace back and evaluate the importance of each input (and even part of it) and how importance can mutate as a result of environmental fluctuations. On the other hand, both `PAY` and `PSQ` input types are suited for "early" TC—as opposed to classification decisions that need to wait for the whole traffic object to be taken—and they refer to different levels of abstraction (biflow vs. packet) and standpoints (encryption-dependent vs. encryption-independent).

1) Distiller-Based Traffic Classifiers: All the classifiers described hereinafter are designed according to the DISTILLER general framework and trained based on the two-phase procedure illustrated in Section III-A. Our starting point is the DISTILLER-ORIGINAL classifier we have firstly exploited in [11]. DISTILLER-ORIGINAL is made of $P = 2$ *single-modality* branches: the `PAY`-modality branch is fed with the `PAY` input, while the `PSQ`-modality branch with the `PSQ` one. The *single-modality* layers of the `PAY`-modality branch are two 1D convolutional layers (with 16 and 32 filters, respectively, 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 (with 128 neurons). The *single-modality* layers of the `PSQ`-modality branch are instead a bidirectional GRU (BiGRU with 64 units and return-sequences behavior) and a dense layer (128 neurons). To capture the inter-modality dependencies, the abstract features extracted by such branches are fused using a concatenation layer and fed to a *shared-representation* dense layer (with 128 neurons). The latter is then connected to $V = 3$ layers, each constituting one *task-specific* dense layer (with 128 neurons), before performing the v^{th} TC task via the corresponding softmax (i.e., a dense layer with L_v neurons and softmax activation). Except for the last softmax, all the layers are equipped with Rectifier Linear Unit (ReLU) activations. To provide regularization and avoid overfitting, a 20% dropout is applied after each dense layer (including the concatenation one) and after flattening the 2D representation of both the stack of convolutional/pooling layers and BiGRU.

TABLE II
VARIANTS OF DISTILLER-ORIGINAL INVESTIGATED IN THIS WORK AND RELATED ENHANCEMENTS

Variant	**EL+ALR**	**IR**	**Ca**	**Co**
DISTILLER-ORIGINAL	—	—	—	—
DISTILLER-EMBEDDINGS	✓	—	—	—
DISTILLER-EARLIER	✓	✓	—	—
DISTILLER-CALIBRATED	✓	✓	✓	—
DISTILLER-EVOLVED	✓	✓	✓	✓

EL+ALR: Embedding Layers and Adaptive Learning Rate;
IR: Input Refinement (driven by Deep SHAP);
Ca: Enhanced Calibration (via label smoothing);
Co: Compression (via pruning).

Starting with this traffic classifier, in our analyses, we follow a process of *sequential improvement* of the basic DISTILLER-ORIGINAL architecture via various optimizations. All the DISTILLER-based variants, along with their improvements, are introduced throughout Section V and are summarized in Tab. II. They are all trained via the two-phase procedure involving the independent pre-training of single-modality branches for 30 epochs each, and the successive fine-tuning of the whole architecture for 40 epochs during which the lowest two 1D convolutional (of `PAY`-modality) and BiGRU (of `PSQ`-modality) layers are frozen. Specifically, pre-training and fine-tuning minimize the respective multitask categorical cross-entropy loss functions (set with a uniform allocation of the preference weights, that is each per-task weight equals 1/3) via the standard ADAM optimizer (set with a batch size of 50 samples). Finally, to further reduce the chance of overfitting, we apply the early-stopping technique by monitoring the variation of the training accuracy.[13]

2) Multitask Traffic Classification Baselines: In the following, we provide some details on the baselines against which we compare DISTILLER-based classifiers. We report in parentheses the input with which we feed each baseline.

We have implemented a modified version—henceforth named `2D-CNN (PAY)`—of the multitask (single-modal) architecture made of two 2D-CNN branches originally presented in [40]. In its original formulation, this baseline is fed with biased input data (i.e., raw PCAP formatted as images) and characterized by an excessively *ad-hoc* structure. We adapted it to our scenario by (i) feeding it with the unbiased `PAY` input and (ii) considering a different task mapping

[12]To properly exploit the Integrated Gradients technique (see Section III-B), we need to distinguish actual from padded zeros. To this end, we add 1 to all bytes of the `PAY` input before normalization (by dividing each byte by 256) and zero-padding. Similarly, for the `PSQ` input, we constrain the quantile transformer to map padding values to zeros.

[13]The most common approach monitors early-stopping by means of a validation set. Nevertheless, some classes in the training set (e.g., `Aim`, `ICQ`, `SFTP`) have a reduced number of samples due to the class-imbalance inherent in the ISCX VPN-NONVPN dataset (cf. Section IV-A) but also typical of a real TC scenario. Therefore, using part of the training set for validation could impair the performance associated with these minority classes. For this reason, we use early-stopping on training data by evaluating the "knee" of the training accuracy, and exiting when this condition is satisfied.

to the two CNN-based branches, namely one binary (T_1 - Encapsulation) and two multi-class learning tasks (T_2 - Traffic Type and T_3 - Application).

We have also extended three state-of-the-art DL-based single-task traffic classifiers to the multitask setup, namely the `1D-CNN (PAY)` architecture proposed in [37] and the `LSTM (PSQ)` and `HYBRID (PSQ)` architectures proposed in [44]. The former encompasses two 1D convolutional layers and a dense layer. The latter two are an LSTM and a hybrid cascade of two 2D convolutional layers and an LSTM layer, respectively. We have extended these single-task architectures by replacing the last softmax with three separate softmax layers, one for each task.

Finally, we compare DISTILLER-based classifiers with five native multitask DL architectures proposed for TC: two different (deep) `MLP (PAY/PSQ)` architectures adopted in [12], [13] and the `1D-CNN (PSQ)` proposed in [14]. For the first two MLP baselines, we evaluate the performance with `PAY` and `PSQ` input types, as in the reference works, they used handcrafted PL/IAT stats as input. For the `1D-CNN (PSQ)` baseline, we exploit the best-performing `PSQ` input type [11], as opposed to the original work in which the authors used a subset of `PSQ` (signed PL and IAT) as input.

To be consistent with DISTILLER-based classifiers, all the baselines are trained to minimize a multitask categorical cross-entropy loss function set with a uniform allocation of preference weights and using the ADAM optimizer and the early-stopping technique to prevent overfitting. The maximum number of training epochs is set to 100, corresponding to the number of epochs obtained summing up those of pre-training and fine-tuning stages of DISTILLER-based classifiers. Other training parameters are set in accordance with the recommendation provided in the respective original studies.

C. Implementation Details

To allow reproducibility, we provide specific implementation details on the whole experimental workbench. All the APIs refer to `Python (3.7)` programming language. Specifically, we exploit the DL models provided by `Keras` (https://keras.io) and `TensorFlow 2` (https://www.tensorflow.org/), to *implement*, *test*, and *calibrate* the traffic classifiers described above. For *pruning*, we leverage the `TensorFlow Model Optimization Toolkit`, a suite of tools for optimizing ML and DL models for deployment and execution. For *quantization*, we exploit the functionalities provided by `TensorFlow Lite` (https://www.tensorflow.org/lite) which enables to convert TensorFlow models for the deployment on lightweight devices. Also, we use the `shap` library (https://github.com/slundberg/shap) to leverage its Deep SHAP implementation and the open source Python library `Alibi` (https://www.seldon.io/tech/products/alibi/) to calculate the IG. *Data pre-* and *post-processing operations* 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.

All the experiments refer to the *same* hardware architecture: 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 $\times$ Intel Xeon E5-4610v2 CPUs @ 8 $\times$ 2.30 GHz and 64 GB of RAM. For the evaluation of (per-epoch) training complexity of DL approaches, we computed the execution times via `time.process_time()` to consider only the actual runtime on the CPUs (i.e., in a sequential fashion).

V. EXPERIMENTAL EVALUATION

This section reports the experimental evaluation performed in this work. First, we introduce a new version of the classifier (namely DISTILLER-EMBEDDINGS) by introducing trainable *embedding layers*[14] for both the modalities and a *learning rate scheduler*.[15] We compare its performance against some of the state-of-the-art multitask traffic classifiers and analyze it in depth w.r.t. the three TC tasks (Section V-A).

Then, in Section V-B we analyze the importance of the traffic modalities used by the classifier, considering them both together and separately. To this aim, we employ two well-known techniques, namely *Deep SHAP* and *IG*. By leveraging the median importance obtained with Deep SHAP for each modality input, we perform an analysis to choose the most appropriate input dimensions, discarding non-influential inputs that do not help in classifying the instances (Section V-C). In this way, we obtain another variant for the classifier, fed with a subset of the original inputs, that is more suitable for early classification (named for the above reason DISTILLER-EARLIER).

In Section V-D, we assess and improve the calibration of this by using *label smoothing* during the training phase, thus achieving DISTILLER-CALIBRATED. We also analyze whether and how calibration changes when considering some combinations of instances that have been correctly or incorrectly classified according to the three TC tasks.

Finally, in Section V-E, we compare different techniques (e.g., *knowledge distillation*, *pruning* and *quantization*) to define a lightweight architecture with similar performance to the classifier optimized so far. With all these improvements in place, we define DISTILLER-EVOLVED, which is enhanced from different points of view: input size, calibration, performance, and dimension.

To conclude our study, in Section V-F we provide an interpretability analysis of DISTILLER-EVOLVED. Table II summarizes the variants of DISTILLER-ORIGINAL that we obtain in our study, with the characterizing enhancements. We compare the different versions of DISTILLER-ORIGINAL in Section V-G.

In all the following analyses, the performance evaluation is based on a *stratified ten-fold cross-validation*. Indeed, the latter represents a solid assessment setup since it keeps the sample ratio among classes for each fold: in this case, since

[14]Each input element is embedded into a vector of dimension $e = 10$. To reduce the training complexity, in the `PSQ`-modality we embed only the number of bytes in transport-layer payload.

[15]The newly introduced learning rate scheduler implements an adaptive learning rate which is halved every five epochs during both the pre-training and the fine-tuning.

TABLE III
COMPARISON OF DISTILLER-EMBEDDINGS (HIGHLIGHTED IN ORANGE) ACCURACY AND F-MEASURE WITH STATE-OF-THE-ART BASELINES. RESULTS ARE IN THE FORMAT *avg.* ($\pm$ *std.*) OBTAINED OVER 10-FOLDS. RANK IS BASED ON THE AVERAGE OF ALL PERFORMANCE METRICS ON ALL THE TASKS. THE LAST ROW SHOWS DISTILLER-EMBEDDINGS GAIN [%] ON THE OVERALL-BEST-BASELINE: DISTILLER-ORIGINAL (HIGHLIGHTED IN BLUE)

Multitask DL Traffic Classifier	**T_1 - Encapsulation**		**T_2 - Traffic Type**		**T_3 - Application**	
	Accuracy [%]	**F-measure [%]**	**Accuracy [%]**	**F-measure [%]**	**Accuracy [%]**	**F-measure [%]**
DISTILLER-EMBEDDINGS	93.01 (± 0.60)	91.46 (± 0.84)	81.71 (± 1.26)	80.16 (± 1.79)	79.92 (± 1.31)	66.38 (± 2.99)
DISTILLER-ORIGINAL (Aceto et al. [11])	91.69 (± 0.80)	89.89 (± 1.00)	79.02 (± 1.40)	77.65 (± 1.79)	78.01 (± 1.24)	64.93 (± 1.79)
`1D-CNN (PAY)` (Wang et al. [37])	86.22 (± 0.79)	83.09 (± 0.80)	74.93 (± 1.51)	73.58 (± 1.76)	76.23 (± 1.22)	64.04 (± 1.74)
`2D-CNN (PAY)` (Huang et al. [40])	86.25 (± 1.07)	82.86 (± 1.27)	73.93 (± 1.14)	72.54 (± 1.63)	74.94 (± 1.07)	62.56 (± 1.84)
`MLP (PAY)` (Zhao et al. [12])	85.65 (± 1.01)	82.07 (± 1.35)	72.38 (± 1.21)	70.47 (± 1.93)	72.48 (± 1.12)	58.57 (± 2.01)
`MLP (PAY)` (Sun et al. [13])	84.34 (± 0.71)	80.67 (± 0.69)	69.70 (± 1.42)	67.42 (± 1.66)	69.53 (± 1.60)	55.01 (± 2.65)
`1D-CNN (PSQ)` (Rezaei and Liu [14])	85.45 (± 1.03)	82.02 (± 1.16)	65.56 (± 2.72)	65.07 (± 3.02)	63.92 (± 1.59)	52.30 (± 2.45)
`HYBRID (PSQ)` (Lopez-Martin et al. [44])	85.03 (± 2.82)	81.16 (± 4.53)	66.46 (± 6.67)	64.30 (± 7.61)	62.97 (± 7.00)	51.92 (± 7.12)
`MLP (PSQ)` (Zhao et al. [12])	84.96 (± 1.78)	80.91 (± 3.19)	66.23 (± 1.70)	65.05 (± 2.86)	63.51 (± 1.18)	50.21 (± 1.92)
`MLP (PSQ)` (Sun et al. [13])	83.83 (± 0.63)	79.51 (± 1.04)	63.41 (± 1.55)	61.44 (± 2.47)	60.09 (± 1.88)	44.81 (± 2.53)
`LSTM (PSQ)` (Lopez-Martin et al. [44])	82.94 (± 0.76)	77.78 (± 1.07)	60.79 (± 1.93)	59.37 (± 2.70)	57.83 (± 2.29)	44.93 (± 2.56)
DISTILLER-EMBEDDINGS GAIN	+ 1.32 (± 0.82)	+ 2.69 (± 0.71)	+ 1.91 (± 0.73)	+ 1.57 (± 0.98)	+ 2.51 (± 1.30)	+ 1.45 (± 2.39)

we are facing multiple TC tasks, the stratification is performed herein on T_3, representing the hardest task. Hence, we report both the *mean* and the *standard deviation* of each performance measure as a result of the evaluation on the ten different folds.

A. Performance Comparison With State-of-the-Art Baselines

In this section, we compare DISTILLER-EMBEDDINGS against the considered baseline architectures w.r.t. the three TC tasks considered. Table III summarizes the results in terms of (i) *Accuracy* (i.e., the fraction of correctly classified biflows over their total number) and (ii) *F-measure*, (which takes into account recall and precision in a more concise way). In detail, as we consider multi-class traffic classifiers, we employ their arithmetically-averaged (viz. macro) versions.

Experimental results highlight that **DISTILLER-EMBEDDINGS performs better than all the considered baselines according to both the considered performance metrics**. Indeed, it provides remarkable improvements w.r.t. the best-performing baseline that is DISTILLER-ORIGINAL in all the cases. More in detail, DISTILLER-EMBEDDINGS reports performance improvements of 1.32% and 1.57% for task T_1, of 2.69% and 2.51% for task T_2 and of 1.91% and 1.45% for task T_3 in terms of Accuracy and F-measure, respectively. It is worth noting that the performance rank of the architectures witnesses that multimodal approaches (i.e., DISTILLER-EMBEDDINGS and DISTILLER-ORIGINAL) are able to achieve better performance as they capitalize on two different views of traffic data. Generally speaking, results highlight that classifiers based on the `PAY` input outperform those exploiting `PSQ` input.

To understand the performance of DISTILLER-EMBEDDINGS in more detail, we also investigate the relationship among classification results on different tasks. Hence, the bar plot in Fig. 2 reports the *joint probability* of classification outcomes for the three tasks, with each outcome being either correct (✓) or wrong ($\times$). For each probability, we report the mean and standard deviation averaged over

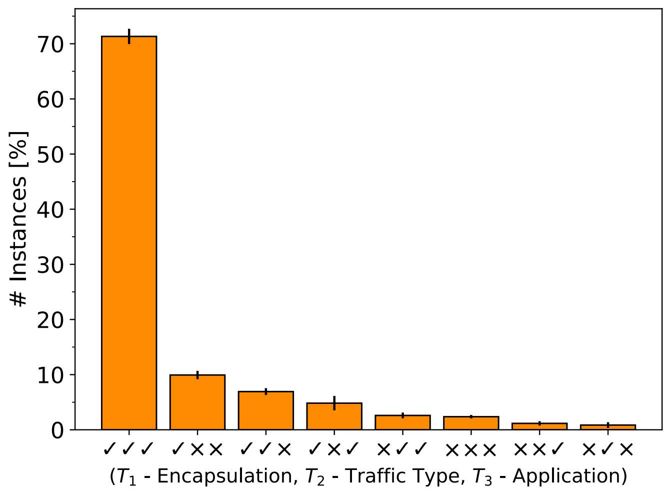
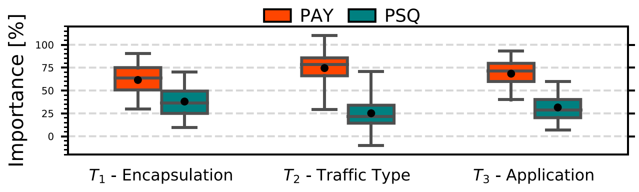


Fig. 2. Detailed performance of DISTILLER-EMBEDDINGS. Error bars report average standard deviation of the joint probability of classification outcomes for $V = 3$ TC tasks: *Encapsulation*, *Traffic Type*, and *Application* (T_1, T_2, and T_3, respectively). Each outcome can be either correct (✓) or wrong ($\times$), thus leading to $2^V = 8$ configurations.

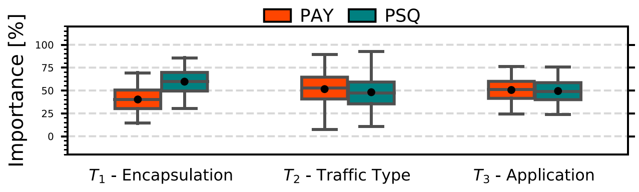
the 10 folds. Results show that for most of the instances ($\approx$70%) DISTILLER-EMBEDDINGS succeeds in classifying correctly on all the tasks simultaneously, i.e., (✓,✓,✓), thus allowing full network visibility. Moreover, it is evident that the cases where the classifier is able to recognize the two most difficult tasks (T_2–Traffic Type and T_3–Application) while failing at the simplest one (T_1–Encapsulation), i.e., (×,✓,✓), are very infrequent. In addition, the probability of wrongly predicting all three tasks $(\times, \times, \times)$ is about twice as high as the probability of identifying one of the two most difficult tasks when the other two have not been correctly classified, namely $(\times, \times, ✓)$ or $(\times, ✓, \times)$.

B. Interpretability Analysis

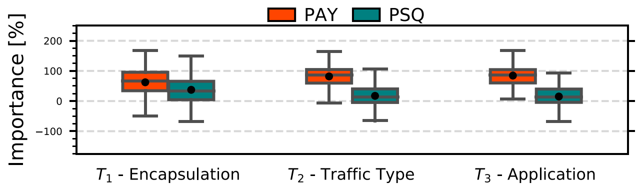
This section presents the interpretability analysis for DISTILLER-ORIGINAL and DISTILLER-EMBEDDINGS to



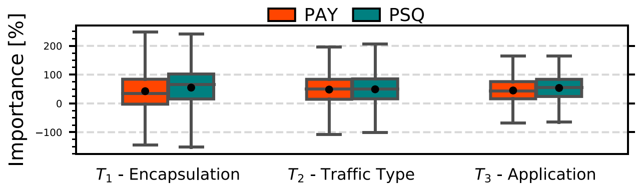
(a) Modality Contributions for DISTILLER-ORIGINAL with Deep SHAP.



(b) Modality Contributions for DISTILLER-EMBEDDINGS with Deep SHAP.



(c) Modality Contributions for DISTILLER-ORIGINAL with IG.



(d) Modality Contributions for DISTILLER-EMBEDDINGS with IG.

Fig. 3. Modality contributions of DISTILLER-ORIGINAL and DISTILLER-EMBEDDINGS with Deep SHAP (a, b) and IG (c, d) techniques. Normalized importance $\widetilde{\phi}_{\mathcal{M}_p}$ for both PAY-modality and PSQ-modality is reported for the three considered TC tasks. Whiskers show 5th and 95th percentiles.

highlight their differences in terms of performance explainability when investigated with *Deep SHAP* or *IG*. In detail, in Section V-B1 we assess the contribution of each modality to compare their relative importance. Then, in Sections V-B2 and V-B3 we analyze in more detail PAY- and PSQ-modalities.[16]

1) Contribution of the Modalities to Correct TC Decisions: In Fig. 3, we investigate the (relative) contribution that PAY and PSQ give to TC, focusing on DISTILLER-ORIGINAL and DISTILLER-EMBEDDINGS and using both Deep SHAP (Figs. 3(a) and 3(b)) and IG (Figs. 3(c) and 3(d)). We consider the pooled Deep SHAP values and the pooled IG $\widetilde{\phi}_{\mathcal{M}}$ for each modality and for each of the three TC tasks. The corresponding distributions (shown via boxplots) are then obtained by selecting the correctly classified test samples of the whole dataset and they refer to PAY-modality (red) and PSQ-modality (green). Focusing on DISTILLER-ORIGINAL and Deep SHAP (Fig. 3(a)), PAY-modality contributes with higher importance values: the median value is higher than $\approx 65\%$ for all the considered TC tasks. The situation is very similar when considering the outcome of the analysis with IG (Figs. 3(c)), but reporting lower values in absolute terms.

On the other hand, the results of the analysis are remarkably different for DISTILLER-EMBEDDINGS (Figs. 3(b) and 3(d)). Indeed, the differences among median values of pooled Deep SHAP are less significant, with the two modalities contributing almost equally to the model predictions. Specifically, for the task T_1 with both the techniques, we can observe that the PSQ-modality median value exceeds that of the other modality.

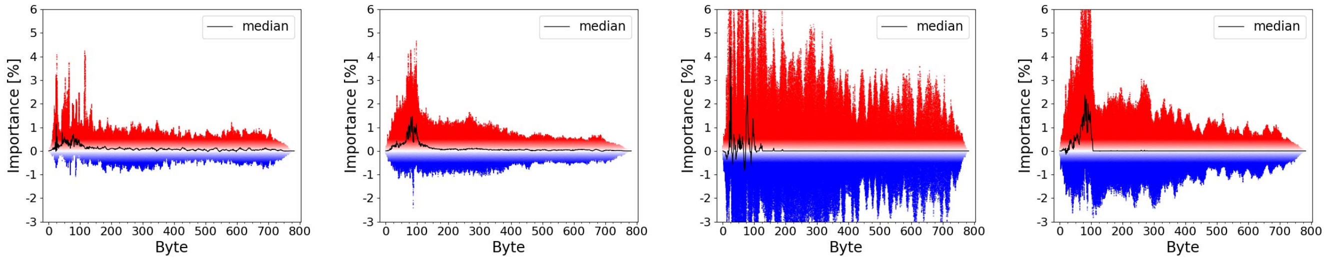
[16]**Implementation Details for Interpretability Methods:** the baseline for IG is set to all zero values for both modalities. Moreover, for DISTILLER-ORIGINAL, we calculate IG w.r.t. the input layers. Conversely, for DISTILLER-EMBEDDINGS, we calculate IG w.r.t. the embedding layer and then we sum the attribution of each input's vector representation [58]. Specifically, when we investigate its PSQ-modality, we refer to the layer concatenating the output of the embedding layer used for PL with the other fields (and then we isolate its representations). Similarly, when we investigate the PAY-modality of the same architecture, we refer to the output of the embedding layer. Conversely, for Deep SHAP, not being based on a gradient definition, the effect is always evaluated w.r.t. the original inputs.

In summary, **regardless of the specific interpretability technique adopted, DISTILLER-EMBEDDINGS results in a more balanced importance between the two input modalities w.r.t. DISTILLER-ORIGINAL.** This can be attributed to the addition of embedding layers to the basic architecture of DISTILLER-ORIGINAL.

2) Interpretability of PAY-Modality: In this section, we focus on providing global explanations for the PAY-modality of DISTILLER-ORIGINAL and DISTILLER-EMBEDDINGS, which rely on the first $N_b = 784$ transport-layer payload bytes of each biflow. In the following, sample-wise positive and negative Deep SHAP values are highlighted with red and blue colors, respectively. Also, for completeness, the median importance value of each byte (over different samples) is reported as a solid black line. This allows for highlighting regions that are more consistently influential (if any) for predictions. We conducted this analysis for each of the three TC tasks.

When considering the traffic pertaining to the two classes of task T_1 (i.e., VPN and non-VPN), clear influential regions can be hardly highlighted, regardless of the classifier and the technique. This behavior results from the fact that a mix of traffic generated by different applications is found at this level, with biflows exposing dramatically different characteristics.

Moving to task T_2 (Traffic Type), for some classes the situation appears to be clearer. In Fig. 4, we compare the explanations obtained for the P2P traffic type as an explanatory example. We show the outcomes provided by Deep SHAP and IG for both models. Looking at Figs. 4(a) and 4(b), relating to the explanations obtained with Deep SHAP, the first 100 bytes of the biflows result to be essential. The same conclusion can be drawn according to IG (Figs. 4(c) and 4(d)), although the importance reports more extreme values and the explanation is less clear overall. We can notice that the median assumes exactly a zero value after 100 bytes (IG assigns zero importance to the bytes with the same value of the baseline). Comparing the two models, a more precise explanation is obtained for DISTILLER-EMBEDDINGS (Figs. 4(b)



(a) P2P Deep SHAP DISTILLER-ORIGINAL. (b) P2P Deep SHAP DISTILLER-EMBEDDINGS. (c) P2P IG DISTILLER-ORIGINAL. (d) P2P IG DISTILLER-EMBEDDINGS.

Fig. 4. Importance for the inputs (transport-layer payload bytes) of PAY-modality. Exemplifying service type (T_2): P2P.

and 4(d)), probably due to the "focusing effect" brought by the embedding layer.

By inspecting the content of these biflows, the bytes with higher importance values often correspond to strings get_peers1 and info_hash20 which are typical of DHT Protocol used by BitTorrent. These packets start with the expression d1:ad2:id20 and have padding values bytes in the final part. In the analyzed cases this padding receives very low values differently from the initial $\approx$ 100 bytes. Among Torrent biflows, there are also HTTP biflows (GET) and for them, it is difficult to recognize consistently influential regions.

For the task T_2, the importance peak is always found in the first 100–200 bytes (even if it is less evident than in Fig. 4 in some cases). As a notable exception, for the Email class, there are a few groups of important bytes both in the central and the last part of the considered portion of the biflow.

The same phenomenon occurs for the classes associated with the task T_3, with the exception of some apps (including Email, Hangouts, SCP, Vimeo and YouTube) which have groups of bytes of non-negligible importance in addition to the initial ones. There are also other apps for which we cannot generalize and identify the most influential regions. For such apps, this observation does not imply that the first 100–200 bytes are not important, but that areas with higher importance are more hardly identifiable. More specifically, this occurs for some apps either having very few correctly classified samples (such as ICQ and Aim) or whose traffic is composed of different traffic types (such as Skype, Facebook, and Hangouts). For instance, in the latter case (e.g., Skype), the traffic mix corresponds to 8.4% Chat, 42.5% FileTransfer, and 49.1% VoIP (see Section IV-A).

3) Interpretability of PSQ-Modality: Similarly to the previous section, here we use the above-mentioned interpretability techniques and investigate the importance of the inputs associated with the PSQ-modality of DISTILLER-ORIGINAL and DISTILLER-EMBEDDINGS, thus discussing the branch fed with the 4 header fields extracted from the first 32 packets of each biflow. Hence, considering the importance associated with these fields, the figures in the following show (for some representative examples) the median importance values for each element of the 4 $\times$ 32 matrix used as input for this modality.

First, **when considering T_1 also the interpretation of PSQ-modality is hard to understand**, likely due to the mix of traffic belonging to the related classes.

Moving to T_2 and T_3, often, the role of a field is not clearly stated (with sequences having both elements assuming positive and negative importance). However, **the two techniques agree for most classes in identifying positive importance for the PL field or the role (positive or negative) of specific fields**. IG often highlights as important a (limited) number of packets that assume more extreme values in absolute terms when compared with Deep SHAP.

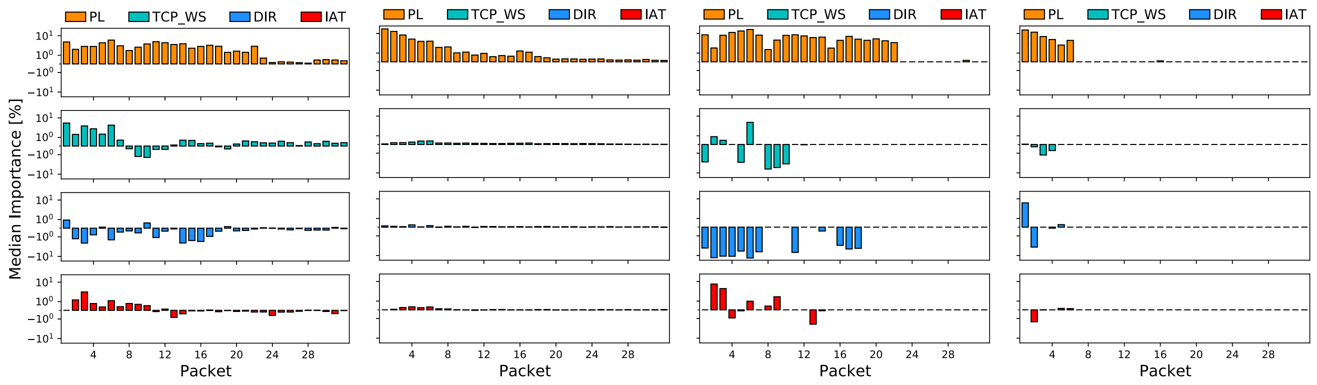
When referring to the explanations obtained for DISTILLER-EMBEDDINGS with Deep SHAP, **the PL field always assumes higher importance values for all the classes of both T_2 and T_3 tasks**. Although the other fields assume lower importance values than PL, they are worthy of consideration as they assume positive values.

Figure 5 reports interesting evidences for FTPS. Looking at Figs. 5(a) and 5(b), it is evident that moving from DISTILLER-ORIGINAL to DISTILLER-EMBEDDINGS, the importance is much more concentrated on the PL field, which becomes the most important field playing a fundamental role in the classification of the biflows of this application. Indeed, while the various fields assume comparable values in Fig. 5(a), the PL field assumes much higher median values for the DISTILLER-EMBEDDINGS model (cf Fig. 5(b)). This result is justifiable if we consider that the modification made to DISTILLER-ORIGINAL for the PSQ-modality consists in introducing an embedding level just for this field. This confirms that the change made to the architecture has increased the expressive power of the PL.

Although less clear, the same phenomenon stems out from the analysis with IG (Figs. 5(c) and 5(d)). In fact, the importance of the DIR, IAT and TCP_WS fields is reduced in the transition from DISTILLER-ORIGINAL to DISTILLER-EMBEDDINGS and the very first elements of PL sequence assume higher median values.

In general terms, **the initial elements of the sequences show higher values than the others** (even if there are cases where the trend is not always strictly decreasing).

From Fig. 5(b) we can notice that after the introduction of embedding layers, the first packets have much higher importance than the following ones since the explanation provided by Deep SHAP clearly shows a decreasing trend for the PL



(a) FTPS Deep Shap DISTILLER-ORIGINAL. (b) FTPS Deep Shap DISTILLER-EMBEDDINGS. (c) FTPS IG DISTILLER-ORIGINAL. (d) FTPS IG DISTILLER-EMBEDDINGS.

Fig. 5. Importance for fields and packets for `FTPS` app. Comparison of Deep SHAP and IG explanations for DISTILLER-ORIGINAL and DISTILLER-EMBEDDINGS.

field. We can find this trend also for `PL` of the first packets in Fig. 5(d). Furthermore, in IG explanations we have different packets with a zero median value. This phenomenon can be explained both by the fact that packets have decreasing importance and by the IG characteristic of assigning a value of zero importance to packets with a padding value.

Notably, inspecting Fig. 5 both interpretability techniques suggest that for the two architectures (above all for DISTILLER-ORIGINAL) the `DIR` field assumes negative importance values, potentially confusing the classifier. In contrast, the `IAT` field has positive values, although the median values are not very high. Generally speaking, there can be no total agreement for the other fields different from `PL`.

In conclusion, although the two techniques are based on two different assumptions, they mostly agree in suggesting the most important groups of bytes or packets for predictions, albeit with some differences. Deep SHAP provides more stable and limited (less extreme) values that are easier to visualize and understand (and are in line with expectations, e.g., about the impact of the adoption of embedding). In fact, IG explanations often result in a noisier and less clear importance representation. While IG has the advantage of clearly reporting input regions that are not influential to prediction (e.g., padding) which are assigned a zero importance value *by construction*, results show that Deep SHAP also clearly identifies these regions (assigning minimal importance values). For these reasons, **we will refer to the Deep SHAP technique for the considerations in the following sections**.

C. Capitalizing on XAI to Choose Input Dimensionality

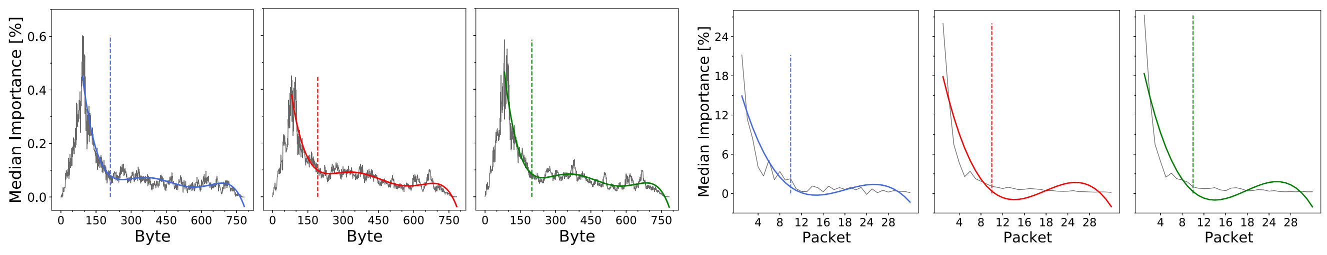
The analysis of the importance of each modality suggests that the initial bytes and packets of each biflow are the most important, despite some differences among traffic types and applications. Starting from the observations above, we can exploit this valuable information to improve the model by selecting a subset of the original inputs, disregarding the less influential ones. This analysis allows for optimizing the architecture in a targeted way, avoiding the more onerous *sensitivity analysis* that, considering the multimodal nature of DISTILLER-EMBEDDINGS, requires the training of a DL model for each combination of `PAY` and `PSQ` inputs.

Note that concerning the `PSQ`-modality, our aim is to reduce the number of packets rather than the number of fields considered: while the `PL` field plays a crucial role in classification, the importance of the other fields depends upon the specific classes. Hence, considering them could still be beneficial for predictions. Moreover, reducing the number of packets (instead of their fields) is beneficial to reduce both the network training time and the time needed to gather the relevant input to the DL architecture (i.e., allowing earlier TC).

In order to identify where the inputs are to be trimmed, we propose the XAI-driven optimization procedure that follows. Since the importance of the considered inputs for both modalities shows a decreasing trend, we identify as more promising those inputs before the point of maximum curvature of the median importance curve obtained for each modality. In more detail, a single importance curve for `PSQ`-modality is obtained by summing the median importance of the fields considered for each packet. We interpolate these curves (to mitigate their discontinuous trend) and find the point of maximum curvature with the *kneedle algorithm* [59].

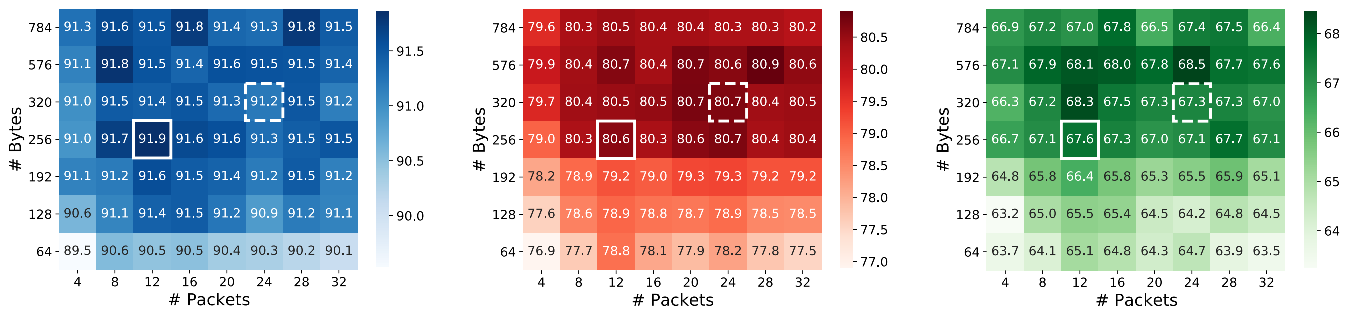
Figure 6 reports the outcome of applying the above XAI-driven optimization procedure for the three tasks to both modalities. Specifically, the procedure highlights that the initial $\approx$ 200 bytes and the first $\approx$ 10 packets are the most influential for `PAY`-modality and `PSQ`-modality, respectively. To exploit this outcome, we can conservatively define a configuration that considers the first 256 bytes and the first 12 packets of each biflow as input for DISTILLER-EMBEDDINGS to obtain input dimensions commonly exploited in recent literature [29], [30], [38].

In order to evaluate these results, we compare them with the indications provided via the *mutual information*, which is a classic alternative method. Specifically, we measure the mutual dependence between independent variables (the inputs of the two modalities) w.r.t. the dependent variable (the actual label) and assign a score accordingly. This method is used to obtain a curve analogous to that obtained with the XAI-driven analysis, thus to compute the point of maximal curvature accordingly. The optimization based on mutual information (whose results are omitted for brevity) turns out to be even more conservative because recommends utilizing at least 300 bytes and



(a) T_1 PAY. (b) T_2 PAY. (c) T_3 PAY. (d) T_1 PSQ. (e) T_2 PSQ. (f) T_3 PSQ.

Fig. 6. Median Importance of DISTILLER-EMBEDDINGS inputs for each TC task according to Deep SHAP. Figures (a–c) refer to PAY-modality, Figures (d–f) to PSQ-modality. The colored line represents the interpolating polynomial (5th and 3rd degree, respectively) starting from the point of maximum median importance. The dashed line shows the location of interpolating polynomial knee (point of maximum curvature).



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

Fig. 7. Sensitivity analysis results for DISTILLER-EMBEDDINGS: Figures (a-c) depict the F-measure for the three tasks versus (no. of bytes, no. packets). The square with the solid line highlights the point identified with the XAI-driven method, whereas the one with the dotted line represents the configuration found with the analysis based on mutual information. Similar results are observed with the accuracy (not shown, for brevity)

24 packets for all tasks, leading to a bigger architecture with longer training times.

Both methods are validated against the resource-consuming *sensitivity analysis* which explores all the combinations of values within the range of interest (i.e., a grid search). For this analysis, we selected for the PAY-modality some values among the most used in state-of-the-art classifiers (e.g., 128, 256, 576). In addition, from 64 to 320 bytes, we increase by 64 bytes the considered dimension both to take into account the dimensions selected with mutual information and, above all, to understand if the performance reflects the importance profile obtained through Deep SHAP. For the PSQ-modality, we start with 4 packets and increase the size by 4 until 32 (the initial number of packets).

The heatmaps in Fig. 7 detail the performance (in terms of F-measure) of the DISTILLER-EMBEDDINGS classifier with the different combinations for the input size. For task T_1, the point identified by the XAI-driven procedure corresponds with the best performance. For task T_2, however, the configuration suggested by the two methods provides similar performance, comparable with the ones obtained with more bytes and packets. For task T_3, the dimensions identified with our proposal provide higher performance than those identified with the other method. Actually, the highest performance corresponds with 576 bytes and 24 packets, but, considering that the gain in performance is not so high and the mean RTPE (Run Time Per-Epoch) is much higher ($\approx$ 37 seconds), the most convenient configuration remains the one suggested by our proposed methods.

Indeed, reducing the inputs leads to a significant reduction in terms of mean RTPE: DISTILLER-EMBEDDINGS has a mean RTPE of $\approx$ 50 seconds whereas the configuration suggested by the proposed XAI-driven approach requires $\approx$ 21 seconds, i.e., a 58% decrease. Notably, considering fewer bytes and packets also provides slight improvements for all three tasks. In detail, concerning F-measure, we have improvements of 0.39, 0.46, and 1.26 for the three TC tasks, respectively.

At the end of this analysis, we obtain an instance of DISTILLER, which performs better and is more suitable for early classification since it requires fewer packets to provide the classification outcome. For this reason, we refer to this new configuration as DISTILLER-EARLIER.

D. Calibration Analysis

In this section, we investigate the *reliability* of the TC models. Specifically, we focus on assessing and enhancing the *calibration* of DISTILLER-EARLIER, as defined in Section III-C. The assessment is performed based on the calibration analysis (reliability diagrams and ECE). On the other hand, the calibration enhancement is pursued by investigating the benefits provided by the adoption of *label smoothing* during the training phase of our classifier.

For the latter technique, we let the value of α (cf. Eq. (9)) vary in $\{0.0125, 0.025, 0.05, 0.075, 0.1\}$. In the configuration with $\alpha = 0$, the usual cross-entropy is employed (viz. without label smoothing). Figure 8 shows the calibration performance obtained when using label smoothing in terms of ECE. Although a unimodal trend can be spotted for each task, the illustration highlights that it is not possible to select a unique value for the parameter α such as to provide the best

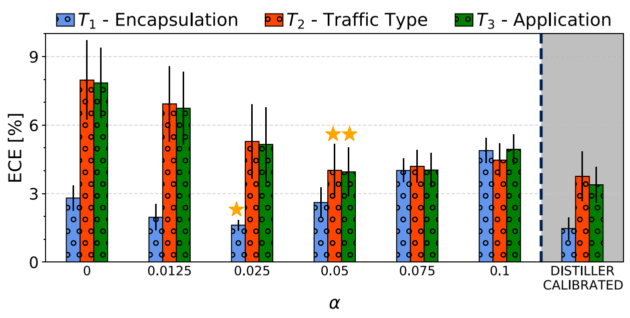


Fig. 8. Calibration sensitivity analysis of DISTILLER-EARLIER in terms of ECE. At the right of the dashed line, the best configuration is shown (i.e., with $\alpha = 0.025$ for the task T_1 and $\alpha = 0.05$ for tasks T_2 and T_3). The value of α providing the best calibration for each single task is highlighted via a $\star$.

TABLE IV
F-MEASURE EVALUATION OF DISTILLER-EARLIER VERSUS THE (LABEL) SMOOTHING PARAMETER α. THE LAST ROW QUANTIFIES THE F-MEASURE [%] DIFFERENCE BETWEEN DISTILLER-EARLIER ($\alpha = 0$) AND ITS BEST-CALIBRATED VERSION: DISTILLER-CALIBRATED

α	$\mathbf{T_1}$ - Encapsulation	$\mathbf{T_2}$ - Traffic Type	$\mathbf{T_3}$ - Application
0	91.86 (± 0.80)	80.61 (± 1.74)	67.64 (± 2.38)
0.0125	91.76 (± 0.67)	80.34 (± 1.84)	67.55 (± 2.53)
0.025	91.73 (± 0.80)	80.47 (± 1.43)	67.99 (± 2.55)
0.05	91.84 (± 0.94)	80.27 (± 1.48)	67.44 (± 2.75)
0.075	91.62 (± 0.79)	80.58 (± 1.68)	67.33 (± 2.10)
0.1	91.88 (± 0.75)	80.60 (± 1.74)	67.83 (± 2.27)
$\alpha_1 : 0.025$ - $\alpha_{2,3} : 0.05$	91.80 (± 1.05)	80.67 (± 1.55)	67.17 (± 3.17)
Difference	− 0.06 (± 0.68)	+ 0.05 (± 0.97)	− 0.47 (± 2.43)

calibration for all the three TC tasks. Hence, in order to define a calibrated version of the architecture (namely, DISTILLER-CALIBRATED) we set α to 0.025 for the binary task T_1, whereas this parameter is chosen equal to 0.05 for the other two tasks (T_2 and T_3). The calibration figures obtained with this combination of values are shown at the right of the dashed line with a gray background.

By comparing the reliability diagrams of DISTILLER-EARLIER and DISTILLER-CALIBRATED (not directly shown for brevity) to understand the differences in terms of calibration, we can spot that DISTILLER-EARLIER is over-confident in its predictions, especially for the (harder) tasks T_2 and T_3. This phenomenon appears in a way lighter fashion when focusing on DISTILLER-CALIBRATED which takes advantage of label smoothing.

Conversely, in Tab. IV we focus on the classification performance and report the assessment of the impact on F-measure when employing label smoothing (with varying α). The last row shows the difference between DISTILLER-EARLIER (where label smoothing is not employed) and DISTILLER-CALIBRATED. Notably, both versions of the classifier achieve comparable results. Hence, **leveraging label smoothing during the training can improve the calibration of the investigated multimodal multitask architecture, without significant losses in performance (we even have small improvements in some cases)**.

Finally, by focusing on DISTILLER-CALIBRATED, in Fig. 9 we report a task-conditional calibration analysis as described in Section III-C. To this end, in Fig. 9(a) we depict the *task-conditional ECE* of each task w.r.t. the four possible combinations of classification outcomes—namely, correct (✓) or wrong (×) prediction—on the other two tasks. For completeness, the weighted average of the task-conditional ECEs for each case (the first bar of each group) is also reported. By looking at the results, for each task we can observe worse calibration outcomes (i.e., a higher conditional ECE) for the combinations involving at least one wrong decision on another task, and especially when considering the instances misclassified for both the other two tasks (corresponding to a "?" and two "×"). Indeed, for tasks T_2 and T_3, we have a similar trend, with the worst calibration observed in correspondence of the instances erroneously classified by all the other tasks. Still, when there is only one task misclassification, T_2 and T_3 seem to be highly affected each other. Conversely, for task T_1 we have a slightly different situation with calibration approximately the same when we consider instances misclassified on both tasks T_2 and T_3 ($? \times \times$) or only on task T_2 ($? \times ✓$).

To further deepen such an analysis, we inspect the *task-conditional reliability diagrams* for task T_2, referring to $(T_1, T_3) = (✓, ✓)$, $(T_1, T_3) = (\times, ✓)$ and $(T_1, T_3) = (✓, \times)$ in Fig. 9(b), 9(c), and 9(d) respectively. We omit the case $(T_1, T_3) = (\times, \times)$ as this is less frequent (this generally applies to all the tasks considered) but also the case where it is more expected to have a calibration degradation. Overall, we can observe that we have an under-confident classifier when focusing on the instances correctly classified for both the other two tasks (Fig. 9(b)). Conversely, a particularly over-confident model is observed when the instances are misclassified only on task T_3 (Fig. 9(c)). Finally, the shift to overconfidence is less apparent when misclassifications are on task T_1 (Fig. 9(d)). This again confirms the higher coupling effect between $T_2 - T_3$, rather than $T_1 - T_2$.

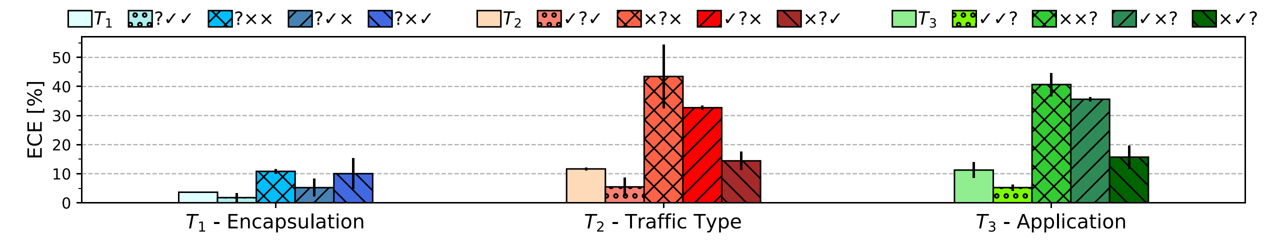
E. Model Compression

This analysis aims at evaluating whether we can obtain a lighter version of the architecture optimized so far using model compression techniques (namely, *knowledge distillation*, *pruning*, and *quantization*) with no significant loss in classification performance and calibration. We highlight that this aspect is particularly important for deployment in resource-constrained environments. To fairly quantify the benefits achieved with the techniques investigated, we refer to the memory occupation of the model.[17]

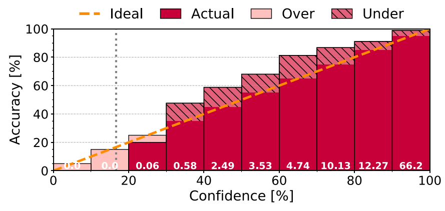
To apply **knowledge distillation**, we consider DISTILLER-CALIBRATED as the teacher model, and we define a lightweight version of it as the student model, by halving the number of filters in convolutional layers, the number of units of dense and GRU layers, and the output dimension of embedding layers.

We evaluate different combinations for T and λ parameters (cf. Section III-D). The configuration which returns the best

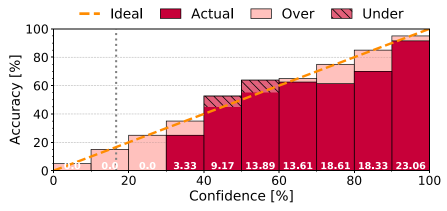
[17]To appreciate the differences, we consider the size of the compressed file (gzip) used to save each model. We cannot use the number of parameters for comparison because it remains unchanged after compression (pruning sets parameters to zero, while quantization uses different representations for parameters).



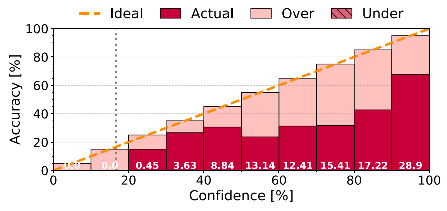
(a) Task-conditional ECE of each task w.r.t. combination of TC outcomes.



(b) DISTILLER-CALIBRATED Task T_2. (given $T_1 = \checkmark$ and $T_3 = \checkmark$).



(c) DISTILLER-CALIBRATED Task T_2 (given $T_1 = \times$ and $T_3 = \checkmark$).



(d) DISTILLER-CALIBRATED Task T_2 (given $T_1 = \checkmark$ and $T_3 = \times$).

Fig. 9. Calibration analysis of DISTILLER-CALIBRATED in terms of task-conditional ECE (as affected by other two tasks) for all three TC tasks (a). Task-conditional Reliability Diagrams for DISTILLER-CALIBRATED, referring to task T_2, are reported in (b-d). In detail, figure (b) depicts the diagram when instances are correctly classified on both tasks T_1 and T_3. Conversely, in figure (c) (resp. (d)) the diagram refers to instances correctly classified by T_3 (resp. T_1) but misclassified by T_1 (resp. T_3).

results is with $T = 1$ and $\lambda = 0$ (notably, in this configuration the student model can rely only on the logits provided by the teacher model and we use them as they are without scaling them up). However, knowledge distillation does not achieve satisfactory performance figures in the case investigated: the performance of the student is not comparable to that of the DISTILLER-CALIBRATED (especially for task T_3, where a 3.73% loss is experienced). Likely, the cause of this result can be found in the fact that (*i*) either distillation usually assumed to be initialized with a model trained on a larger balanced dataset [60] or (*ii*) the student model suffers from the halving of parameters and has irretrievably lost the ability to distinguish certain classes (especially those with few samples). In light of this last consideration, in future work, we plan to more accurately design the student model.

Concerning **pruning**, a number of parameters have to be tuned, including: (i) the number of epochs to perform the additional training phase required to identify the weights to be pruned, (ii) the value of the final sparsity; (iii) whether initializing the model to prune with random weights or with the weights of the model optimized so far (DISTILLER-CALIBRATED)—with the latter approach being recommended in the official documentation of the `tfmot` module. Hence we performed an exhaustive experimental campaign[18] (whose results are not shown, for brevity) aimed at identifying the setup leading to a pruned model whose performance is as much as possible similar to the one of the not pruned model. Results show that 20 epochs (6 epochs for each pretraining phase and 8 for the fine-tuning one) are enough to prune our already-trained model while keeping high classification performance, while with final sparsity values higher than 0.6 classification performance starts decreasing. In such a configuration, the pruned model is $\approx 50\%$ lighter than DISTILLER-CALIBRATED, and exhibits performance comparable with the starting model in terms of both F-measure and expected calibration error.

Finally, we have also evaluated **quantization** in two configurations, transforming the architecture weights from `float32` to `float16` and `int8` representations, respectively. Notably, this compression technique has the remarkable advantage of being a *post-training technique*, i.e., does not require any additional training phase. Quantization leads up to a more than 70% occupation reduction (achieved when using *int8* representation). In our experimental campaigns, we have also explored the compression figures achievable when **combining quantization and pruning** (i.e., applying quantization to an already-pruned model). In this case, the reduction is even more significant (more than 85%).

Table V details the results of the experiments, reporting the memory occupancy (averaged over the 10 folds) achieved via each compression technique evaluated, together with classification performance in terms of F-measure as well as the related calibration figures.

Quantization produces compressed models with higher reductions in size. In addition, as far as F-measure is concerned, the performance of quantized models (whether we prune or not the model) is very close to the original. Although these observations might suggest quantization as the best choice, when also considering calibration the results in the Tab. V highlight that using quantization (either with or without pruning) produces highly decalibrated models. In more detail, reliability diagrams (not shown, for brevity) witness that applying quantization results in strongly under-calibrated classifiers for all three tasks, with the confidence being lower than the accuracy for every bin (similar observations can be made when considering `int8` representation).

[18]Number of epochs and final sparsity took values in $\{10, 20, 50, 100\}$ and $\{0.6, 0.7, 0.8, 0.9\}$, respectively.

TABLE V
OVERALL COMPARISON OF COMPRESSED MODELS. SIZE REFERS TO THE MEMORY NEEDED TO STORE EACH MODEL. RESULTS ARE REPORTED IN TERMS OF CLASSIFICATION (F-MEASURE) AND CALIBRATION (ECE) PERFORMANCE, AND ARE IN THE FORMAT *avg. (± std.)* OBTAINED OVER 10-FOLDS

Multitask DL Traffic Classifier	**Size [MB]**	**T_1 - Encapsulation**		**T_2 - Traffic Type**		**T_3 - Application**	
		F-measure [%]	**ECE [%]**	**F-measure [%]**	**ECE [%]**	**F-measure [%]**	**ECE [%]**
DISTILLER-CALIBRATED	3.67	91.80 (± 1.05)	1.47 (± 0.49)	80.67 (± 1.55)	3.76 (± 1.09)	67.17 (± 3.18)	3.39 (± 0.79)
DISTILLER-CALIBRATED Student	1.49 (−59.4%)	91.12 (± 1.04)	1.75 (± 0.50)	79.05 (± 1.63)	3.48 (± 0.53)	62.46 (± 2.12)	3.46 (± 0.69)
DISTILLER-CALIBRATED Student KD	1.49 (−59.4%)	91.12 (± 0.82)	1.80 (± 0.65)	79.58 (± 1.58)	3.54 (± 1.46)	63.44 (± 2.82)	3.34 (± 1.30)
DISTILLER-CALIBRATED P	1.85 (−49.6%)	91.41 (± 0.95)	1.49 (± 0.46)	80.35 (± 1.48)	3.73 (± 1.05)	66.11 (± 3.42)	3.54 (± 1.10)
DISTILLER-CALIBRATED Q (`float16`)	1.80 (−50.9%)	91.80 (± 1.05)	5.23 (± 1.22)	80.67 (± 1.55)	14.35 (± 2.92)	67.15 (± 3.16)	26.76 (± 3.12)
DISTILLER-CALIBRATED Q (`int8`)	0.84 (−77.1%)	91.79 (± 0.99)	5.00 (± 1.29)	80.62 (± 1.51)	14.33 (± 2.89)	67.42 (± 3.17)	26.76 (± 3.16)
DISTILLER-CALIBRATED P&Q (`float16`)	1.00 (−72.7%)	91.41 (± 0.95)	4.61 (± 0.46)	80.36 (± 1.48)	11.26 (± 1.84)	66.11 (± 3.42)	20.20 (± 1.97)
DISTILLER-CALIBRATED P&Q (`int8`)	0.52 (−85.8%)	91.40 (± 0.89)	4.52 (± 0.62)	80.38 (± 1.48)	11.21 (± 1.81)	66.13 (± 3.45)	20.18 (± 1.97)

The values in parentheses for Size report Size reduction w.r.t. DISTILLER-CALIBRATED
P: Pruning, Q: Quantization, KD: Knowledge Distillation.
DISTILLER-CALIBRATED P (highlighted in green), resulting in the best trade-off between Calibration and Size, is the model that we define as DISTILLER-EVOLVED.

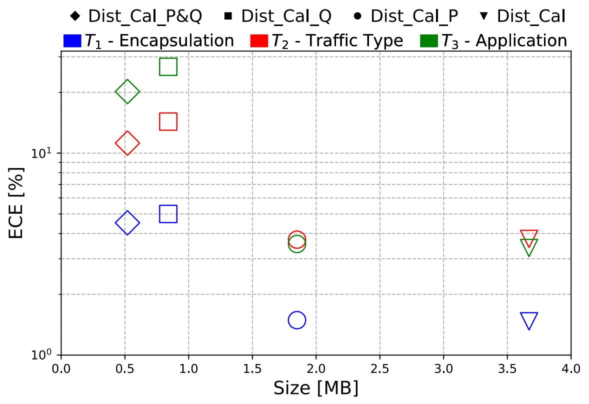


Fig. 10. Comparison of Expected Calibration Error (ECE) against the size of compressed models (only `int8` quantization is shown).

Figure 10 compares the ECE against the size for some of the compressed models discussed above. The inspection is limited to DISTILLER-CALIBRATED and those models with an F-measure drop $\leq$ 1.5%. Moreover, only the `int8` quantization is considered since it achieves a higher compression (with similar calibration performance). Specifically, the three models compared with DISTILLER-CALIBRATED are those obtained after the application of (*i*) pruning, (*ii*) quantization, or (*iii*) both techniques. It is evident that quantization allows a drastic size reduction but, unfortunately, it also results in a substantial increase in the ECE. As a result, both the architectures where quantization is applied have similar representations and are positioned in the top-left part of the graph where the size is reduced, but the ECE becomes unacceptable. On the other hand, with pruning we obtain a model whose size is about $\approx$ 50% less than the original model, keeping the calibration performance almost unchanged. Indeed, its representation is positioned in the same horizontal band identified by the calibration error of DISTILLER-CALIBRATED but in the central part of the graph w.r.t. the model size axis.

Hence, in our sequential process, we select pruning as the best compression technique, thus defining DISTILLER-EVOLVED as the result of applying pruning to DISTILLER-CALIBRATED and resulting in the model optimized along the three objectives.

F. DISTILLER-EVOLVED *Interpretability Analysis*

In this section, we focus on the interpretability of DISTILLER-EVOLVED via Deep SHAP.

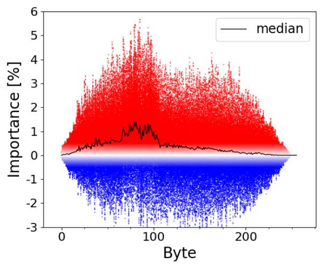
Focusing on `PAY`-modality, we can say that the most crucial region is not always the same for the various classes, confirming the importance of considering a value for N_b no less than 256 bytes. Indeed, when looking at task T_2, the most significant bytes are those in the range (50–150). On the other hand, for `P2P` (Fig. 11(a)), the first 100 bytes are those with the highest importance (same applies to DISTILLER-ORIGINAL, as shown in Section V-B2), while for `Email` (Fig. 11(b)), the significant bytes are those up to position 200. For task T_3, the bytes around position 100 are almost always the most important, with the exception of isolated cases with few samples. In some cases, such as `Facebook` and `Vimeo`, the region of highest importance extends to bytes around position 175; for other classes (e.g., `Email` and `SCP`), one has to go even further to consider all bytes with the highest median importance.

Regarding `PSQ`-modality, the `PL` field remains the most important field, while the others highlight lower values (positive in most of the cases). It may happen that they also assume negative values, but in isolated cases and often for single values of the sequence. Accordingly, they may help in classification even if they contribute less to the output. Figures 11(c) and 11(d) report the median importance for `PSQ`-modality for two exemplary classes (`FTPS` and `Chat`).

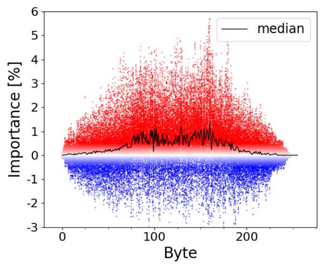
Notably, Fig. 11(e) shows the median importance for either modality and highlights how the `PSQ`-modality becomes the most important for all three tasks after all the improvements.

G. *Architecture Comparison*

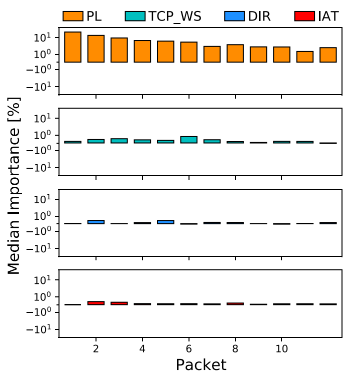
This section provides an overview of the performance for all the architectures we have defined in our process of sequential enhancement leading from DISTILLER-ORIGINAL to DISTILLER-EVOLVED (cf. Tab. II). The radar plot in Fig. 12 compares the five realizations based on DISTILLER framework along the three dimensions of interest, reporting the related memory occupation as well as the calibration performance and the classification performance. Notably, the



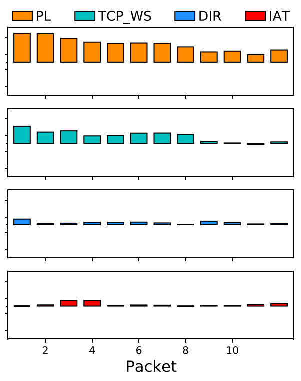
(a) P2P.



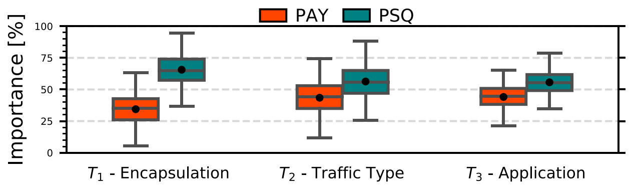
(b) Email.



(c) FTPS.



(d) Chat.



(e) Modality Contribution.

Fig. 11. Interpretability of DISTILLER-EVOLVED based on Deep SHAP.

latter is reported as its complement to 1 (1–F-measure) in order to obtain a plot where smaller areas are associated with major improvements.

First, we have obtained DISTILLER-EMBEDDINGS, which capitalizes embedding layers and adaptive learning rate and achieves better classification performance than DISTILLER-ORIGINAL, but resulting in huge memory occupation. Then, we have performed XAI-driven input optimization, thus defining DISTILLER-EARLIER, which reports better performance (for all the dimensions of interest), in spite of being able to provide classification verdicts after only 256 bytes and 12 packets. To enhance the reliability of DISTILLER-EARLIER, we have leveraged label smoothing and obtained DISTILLER-CALIBRATED, which severely limits its calibration error trading this improvement for limited loss in classification performance. Finally, we focus on enhancing also memory occupation by taking advantage of pruning. **The resulting DISTILLER-EVOLVED outperforms DISTILLER-ORIGINAL in all the aspects of interest while paying limited classification degradation ($\approx$ 1% F-measure) w.r.t. the other variants, but remarkably improving calibration and memory occupation as it achieves the best performance figures for both.**

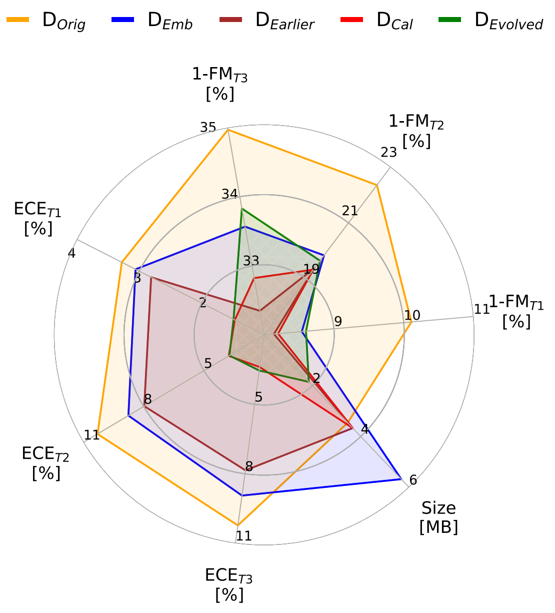


Fig. 12. Comparison of the different DISTILLER versions obtained in the sequential optimization process. Axes report Size, Calibration (ECE_{T1}, ECE_{T2}, and ECE_{T3}), and Classification Performance ($1 - FM_{T1}$, $1 - FM_{T2}$, and $1 - FM_{T3}$). Note that classification performance is reported as the complement of F-measure to 1 so as to obtain **smaller areas for better models**.

VI. CONCLUSION AND FUTURE DIRECTIONS

In this paper, we exploited XAI as a Swiss Army Knife aiming for (*i*) interpreting, (*ii*) improving, and (*iii*) making feasible implementations of multimodal DL approaches that solve multiple (visibility) TC tasks via multitask learning, focusing on encrypted traffic. Pursuing these goals, we design, implement, and evaluate an evolved *multimodal multitask DL traffic classifier*, attained in multiple refinement steps driven by XAI, finally producing the DISTILLER-EVOLVED model, passing through a number of stages, each associated with different realizations of the DISTILLER framework.

Our evaluation was performed on the public ISCX VPN-NONVPN dataset of human-generated traffic labeled according to three different TC tasks, namely encapsulation (VPN-encapsulated or not), traffic type (6 classes), and application recognition (15 classes). We first showed that in all considered tasks our initial proposal (i.e., DISTILLER-EMBEDDINGS) outperforms several multitask DL-based traffic classifiers chosen as baselines from the most relevant literature [12], [13], [14], [37], [40], [44], including the previous state-of-the-art version (i.e., DISTILLER-ORIGINAL) sketched from the same general multimodal multitask framework [11]. Moreover, we exploited two interpretability methods (i.e., *Deep SHAP* and *Integrated Gradients*) to provide a *global explanation* of the underlying rationale for each considered modality, fed with transport-layer payload or fields extracted from packet-sequences, respectively. We quantified the contribution of each modality in solving each task and underlined how the payload still keeps high importance—despite the significant amount of encrypted traffic—as well as the fields extracted from the very first packets. Overall, DISTILLER-EMBEDDINGS resulted in more balanced importance between the two input modalities w.r.t. DISTILLER-ORIGINAL. Also, generalized evidence depends on the combination of input type, task to be solved, and employed DL architecture. Driven by interpretability outcomes, we designed a better performing and "earlier" version of our traffic classifier, namely DISTILLER-EARLIER.

Such an optimized proposal was then investigated in terms of *reliability*, namely how much we can trust its prediction confidence, via calibration. Leveraging *label smoothing* we designed the DISTILLER-CALIBRATED classifier that halves the calibration error on all the three tasks w.r.t. DISTILLER-EARLIER, obtaining a significant gain in reliability, without a significant loss in performance.

Finally, we investigated three techniques to reduce the model size aiming at improving its *feasibility*: *knowledge distillation*, *pruning*, and *quantization*. Pruning turned out to be the best compression technique and led us to the definition of DISTILLER-EVOLVED which outperforms the DISTILLER-ORIGINAL starting point in terms of all aspects of interest: performance, interpretability, reliability, and memory occupation.

Future prospects of research will include (a) employing other datasets to assess the generalizability of the proposed approach by taking into account different network conditions, applications, and operating systems; (b) taking advantage of XAI approaches toward the capitalization of unlabeled data via semi-supervised multitask learning; (c) the use of reliability and interpretability techniques for the analysis and improved design of hierarchically-arranged DL-based traffic classifiers; (d) a robustness assessment of multitask DL-based traffic classifiers to (possibly-multimodal) adversarial attacks; (e) investigating the effect of high classifier reliability on open-set TC (f) the design of (natively) self-explainable [32] and lightweight DL traffic classifiers.

REFERENCES

[1] 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, Jun. 2019.

[2] D. Gunning and D. Aha, "DARPA's explainable artificial intelligence (XAI) program," *AI Mag.*, vol. 40, no. 2, pp. 44–58, 2019.

[3] Telefónica. "Telefonica's approach to the responsible use of AI." 2020. [Online]. Available: https://www.telefonica.com/en/commitment/how-we-work/business-principles/

[4] A. Mujumdar, K. Čyras, S. Singh, and A. Vulgarakis. "Trustworthy AI: Explainability, safety and verifiability." Ericsson, Dec. 2020. [Online]. Available: https://www.ericsson.com/en/blog/2020/12/trustworthy-ai

[5] "AI security white paper," Huawei Technol., Shenzhen, China, White Paper, Oct. 2018. [Online]. Available: https://www.huawei.com/en/trust-center/resources/ai-security-white-paper

[6] A. Razaghpanah, A. A. Niaki, N. Vallina-Rodriguez, S. Sundaresan, J. Amann, and P. Gill, "Studying TLS usage in Android apps," in *Proc. 13th ACM Int. Conf. Emerg. Netw. Exp. Technol. (CoNEXT)*, 2017, pp. 350–362.

[7] D. Madariaga, L. Torrealba, J. Madariaga, J. Bermúdez, and J. Bustos-Jiménez, "Analyzing the adoption of QUIC from a mobile development perspective," in *Proc. ACM Workshop Evol. Perform. Interoperability QUIC (EPIQ)*, 2020, pp. 35–41.

[8] W. Samek, G. Montavon, S. Lapuschkin, C. J. Anders, and K. R. Müller, "Explaining deep neural networks and beyond: A review of methods and applications," *Proc. IEEE*, vol. 109, no. 3, pp. 247–278, Mar. 2021.

[9] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *Proc. 34th Int. Conf. Mach. Learn. (ICML)*, 2017, pp. 1321–1330.

[10] T. Zhang, H. Qiu, M. Mellia, Y. Li, H. Li, and K. Xu, "Interpreting AI for networking: Where we are and where we are going," *IEEE Commun. Mag.*, vol. 60, no. 2, pp. 25–31, Feb. 2022.

[11] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "DISTILLER: Encrypted traffic classification via multimodal multitask deep learning," *J. Netw. Comput. Appl.*, vols. 183–184, Jun. 2021, Art. no. 102985.

[12] Y. Zhao, J. Chen, D. Wu, J. Teng, and S. Yu, "Multi-task network anomaly detection using federated learning," in *Proc. ACM 10th Int. Symp. Inf. Commun. Technol. (SoICT)*, 2019, pp. 273–279.

[13] H. Sun et al., "Common knowledge based and one-shot learning enabled multi-task traffic classification," *IEEE Access*, vol. 7, pp. 39485–39495, 2019.

[14] S. Rezaei and X. Liu, "Multitask learning for network traffic classification," in *Proc. 29th IEEE Int. Conf. Comput. Commun. Netw. (ICCCN)*, 2020, pp. 1–9.

[15] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proc. 30th Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2017, pp. 4765–4774.

[16] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 3319–3328.

[17] G. Draper-Gil, A. H. Lashkari, M. Mamun, and A. A. Ghorbani, "Characterization of encrypted and VPN traffic using time-related features," in *Proc. 2nd Int. Conf. Inf. Syst. Security Privacy (ICISSP)*, 2016, pp. 407–414.

[18] K. Amarasinghe, K. Kenney, and M. Manic, "Toward explainable deep neural network based anomaly detection," in *Proc. 11th Int. Conf. Human Syst. Interaction (HSI)*, 2018, pp. 311–317.

[19] Y. Zheng, Z. Liu, X. You, Y. Xu, and J. Jiang, "Demystifying deep learning in networking," in *Proc. 2nd ACM Asia–Pacific Workshop Netw. (APNet)*, 2018, pp. 1–7.

[20] A. Dethise, M. Canini, and S. Kandula, "Cracking open the black box: What observations can tell us about reinforcement learning agents," in *Proc. ACM Workshop Netw. Meets AI ML (NetAI)*, 2019, pp. 29–36.

[21] A. Morichetta, P. Casas, and M. Mellia, "EXPLAIN-IT: Towards explainable AI for unsupervised network traffic analysis," in *Proc. 3rd ACM CoNEXT Workshop Big DAta, Mach. Learn. Artif. Intell. Data Commun. Netw/ (Big-DAMA)*, 2019, pp. 22–28.

[22] Z. Meng, M. Wang, J. Bai, M. Xu, H. Mao, and H. Hu, "Interpreting deep learning-based networking systems," in *Proc. Annu. Conf. ACM Special Interest Group Data Commun. Appl., Technol., Archit., Protocols Comput. Commun. (SIGCOMM)*, 2020, pp. 154–171.

[23] A. Terra, R. Inam, S. Baskaran, P. Batista, I. Burdick, and E. Fersman, "Explainability methods for identifying root-cause of SLA violation prediction in 5G network," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2020, pp. 1–7.

[24] G. Aceto, G. Bovenzi, D. Ciuonzo, A. Montieri, V. Persico, and A. Pescapé, "Characterization and prediction of mobile-app traffic using Markov modeling," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 1, pp. 907–925, Mar. 2021.

[25] A. Montieri, G. Bovenzi, G. Aceto, D. Ciuonzo, V. Persico, and A. Pescapè, "Packet-level prediction of mobile-app traffic using multitask deep learning," *Comput. Netw.*, vol. 200, Dec. 2021, Art. no. 108529.

[26] S. Rezaei, B. Kroencke, and X. Liu, "Large-scale mobile app identification using deep learning," *IEEE Access*, vol. 8, pp. 348–362, 2019.

[27] C. Beliard, A. Finamore, and D. Rossi, "Opening the deep pandora box: Explainable traffic classification," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, 2020, pp. 1292–1293.

[28] X. Wang, S. Chen, and J. Su, "Real network traffic collection and deep learning for mobile app identification," *Wireless Commun. Mobile Comput.*, vol. 2020, Feb. 2020, Art. no. 4707909.

[29] I. Akbari et al., "A look behind the curtain: Traffic classification in an increasingly encrypted Web," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 5, no. 1, pp. 1–26, 2021.

[30] A. Nascita, A. Montieri, G. Aceto, D. Ciuonzo, V. Persico, and A. Pescapé, "XAI meets mobile traffic classification: Understanding and improving multimodal deep learning architectures," *IEEE Trans. Netw. Service. Manag.*, vol. 18, no. 4, pp. 4225–4246, Dec. 2021.

[31] A. M. Sadeghzadeh, S. Shiravi, and R. Jalili, "Adversarial network traffic: Towards evaluating the robustness of deep-learning-based network traffic classification," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 2, pp. 1962–1976, Jun. 2021.

[32] K. Fauvel, F. Chen, and D. Rossi, "A lightweight, efficient and explainable-by-design convolutional neural network for Internet traffic classification," 2022, *arXiv:2202.05535*.

[33] D. Li, Y. Zhu, and W. Lin, "Traffic identification of mobile apps based on variational autoencoder network," in *Proc. 13th IEEE Int. Conf. Comput. Intell. Security (CIS)*, 2017, pp. 287–291.

[34] A. Rago, G. Piro, G. Boggia, and P. Dini, "Multi-task learning at the mobile edge: An effective way to combine traffic classification and prediction," *IEEE Trans. Veh. Technol.*, vol. 69, no. 9, pp. 10362–10374, Sep. 2020.

[35] 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 Comput.*, vol. 24, no. 3, pp. 1999–2012, 2020.

[36] G. Bovenzi, A. Foggia, S. Santella, A. Testa, V. Persico, and A. Pescapé, "Data poisoning attacks against autoencoder-based anomaly detection models: A robustness analysis," in *Proc. IEEE Int. Conf. Commun.*, 2022, pp. 5427–5432.

[37] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, "End-to-end encrypted Traffic Classification with one-dimensional convolution neural networks," in *Proc. IEEE Int. Conf. Intell. Security Inform. (ISI)*, 2017, pp. 43–48.

[38] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapè, "MIMETIC: Mobile encrypted traffic classification using multimodal deep learning," *Elsevier Comput. Netw.*, vol. 165, Dec. 2019, Art. no. 106944.

[39] X. Wang, S. Chen, and J. Su, "Automatic mobile app identification from encrypted traffic with hybrid neural networks," *IEEE Access*, vol. 8, pp. 182065–182077, 2020.

[40] H. Huang, H. Deng, J. Chen, L. Han, and W. Wang, "Automatic multi-task learning system for abnormal network traffic detection," *Int. J. Emerg. Technol. Learn.*, vol. 13, no. 4, pp. 4–20, 2018.

[41] O. Barut, Y. Luo, T. Zhang, W. Li, and P. Li, "Multi-task hierarchical learning based network traffic analytics," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2021, pp. 1–6.

[42] A. Nascita, F. Cerasuolo, D. D. Monda, J. T. A. Garcia, A. Montieri, and A. Pescape, "Machine and deep learning approaches for IoT attack classification," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, 2022, pp. 1–6.

[43] G. Bovenzi, F. Cerasuolo, A. Montieri, A. Nascita, V. Persico, and A. Pescapé, "A comparison of machine and deep learning models for detection and classification of android malware traffic," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, 2022, pp. 1–6.

[44] 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. 18042–18050, 2017.

[45] W. Li, Z. D. Hugues, R. Valentin, and M. Liewig, "Multi-task attention network for digital context classification from Internet Traffic," in *Proc. 7th Int. Conf. Mach. Learn. Technol. (ICMLT)*, 2022, pp. 1–12.

[46] C. Liu, L. He, G. Xiong, Z. Cao, and Z. Li, "FS-Net: A flow sequence network for encrypted traffic classification," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2019, pp. 1171–1179.

[47] A. Niculescu-Mizil and R. Caruana, "Predicting good probabilities with supervised learning," in *Proc. 22nd Int. Conf. Mach. Learn. (ICML)*, 2005, pp. 625–632.

[48] J. Mukhoti, V. Kulharia, A. Sanyal, S. Golodetz, P. H. Torr, and P. K. Dokania, "Calibrating deep neural networks using focal loss," in *Proc. 34th Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2020, p. 1282.

[49] R. Müller, S. Kornblith, and G. Hinton, "When does label smoothing help?" in *Proc. 32th Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2019, pp. 1–13.

[50] L. S. Shapley, "A value for n-person games," *Contribut. Theory Games*, vol. 2, no. 28, pp. 307–317, 1953.

[51] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," in *Proc. 34th Int. Conf. Mach. Learn. (ICML)*, 2017, pp. 3145–3153.

[52] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "A survey of model compression and acceleration for deep neural networks," 2017, *arXiv:1710.09282*.

[53] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *Proc. NIPS Deep Learn. Represent. Learn. Workshop*, 2015, pp. 1–9.

[54] Y. Wu and M. Zhang, "Lightweight network traffic classification model based on knowledge distillation," in *Proc. Int. Conf. Web Inf. Syst. Eng. (WISE)*, 2021, pp. 107–121.

[55] L. Beyer, X. Zhai, A. Royer, L. Markeeva, R. Anil, and A. Kolesnikov, "Knowledge distillation: A good teacher is patient and consistent," 2021, *arXiv:2106.05237*.

[56] M. Zhu and S. Gupta, "To prune, or not to prune: Exploring the efficacy of pruning for model compression," 2017, *arXiv:1710.01878*.

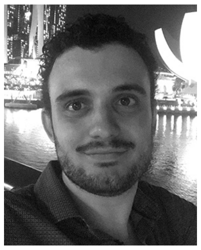
[57] S. Migacz, "8-bit inference with TensorRT," in *Proc. GPU Technol. Conf.*, vol. 2, 2017, p. 7.

[58] M. Ancona, E. Ceolini, C. Öztireli, and M. Gross, "Towards better understanding of gradient-based attribution methods for deep neural networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2018, pp. 1–16.

[59] V. Satopaa, J. Albrecht, D. Irwin, and B. Raghavan, "Finding a 'Kneedle' in a haystack: Detecting knee points in system behavior," in *Proc. 31st IEEE Int. Conf. Distrib. Comput. Syst. Workshops (ICDCSW)*, 2011, pp. 166–171.

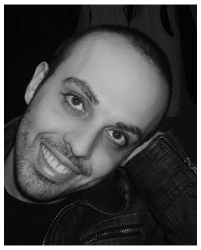
[60] E. Belouadah and A. Popescu, "IL2M: Class incremental learning with dual memory," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, 2019, pp. 583–592.

Alfredo Nascita received the M.S. Laurea degree in computer engineering from the University of Napoli Federico II in March 2021, where he is currently pursuing the Ph.D. degree in information technology and electrical engineering with DIETI. His research interests include traffic classification, machine and deep learning, and explainable artificial intelligence.

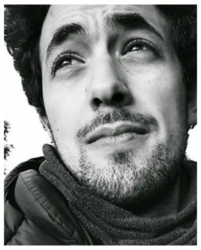


Antonio Montieri (Member, IEEE) received the Ph.D. degree in information technology and electrical engineering from the University of Napoli Federico II in April 2020, where he is an Assistant Professor with DIETI. He has coauthored 39 papers in international journals and conference proceedings. His work concerns network measurements, (encrypted and mobile) traffic classification, traffic modeling and prediction, and monitoring of cloud network performance.

Giuseppe Aceto received the Ph.D. degree in telecommunication engineering from the University of Napoli Federico II, where he is an Assistant Professor. His research concerns network performance, traffic analysis, and censorship, both in traditional networks and SDN, and ICTs applied to health. He received the best paper award at IEEE ISCC 2010 and the 2018 Best Journal Journal Paper Award by IEEE CSIM.



Domenico Ciuonzo (Senior Member, IEEE) received the Ph.D. degree from the University of Campania L. Vanvitelli. He is a Tenure-Track Professor with the University of Napoli Federico II. His research interests are data fusion, network analytics, IoT, and AI. He is the recipient of two Paper awards (IEEE ICCCS 2019 and Elsevier ComNet 2020), the 2019 IEEE AESS Exceptional Service Award, the 2020 IEEE Sensors Council Early-Career Technical Achievement Award, and the 2021 IEEE AESS Early-Career Award.



Valerio Persico received the Ph.D. degree in computer and automation engineering from the University of Napoli Federico II in 2016, where he is an Assistant Professor with DIETI. He has coauthored more than 30 papers within international journals and conference proceedings. His work concerns network measurements, cloud-network monitoring, and Internet path tracing and topology discovery.

Antonio Pescapé (Senior Member, IEEE) is a Full Professor of Computer Engineering with the University of Napoli Federico II. His work focuses on measurement, monitoring, and analysis of the Internet. He has coauthored more than 200 conference and journal papers. He is the recipient of a number of research awards. Also, he has served as an independent reviewer/evaluator of research projects/project proposals co-funded by a number of governments and agencies.

Open Access funding provided by 'Università degli Studi di Napoli "Federico II"' within the CRUI CARE Agreement