IEEE Open Journal of the Communications Society

Received XX Month, XXXX; revised XX Month, XXXX; accepted XX Month, XXXX; Date of publication XX Month, XXXX; date of current version 11 January, 2024.

Digital Object Identifier 10.1109/OJCOMS.2024.011100

ATELIER: service tailored and limited-trust network analytics using cooperative learning

Mattia Milani [1,3], Dario Bega[2], Marco Gramaglia[3], Pablo Serrano[3] *(Senior Member, IEEE)*, and Christian Mannweiler[1]

[1]Nokia S&T, Standards, Munich, Germany
[2]Nokia Bell-Labs, Espoo, Finland
[3]Universidad Carlos III de Madrid, Telematics Engineering, Leganes, Spain

CORRESPONDING AUTHOR: Marco Gramaglia (e-mail: mgramagl@it.uc3m.es).

ABSTRACT The current trends in mobile network architectural design are moving toward the adoption of open interfaces that allow data exchange among different stakeholders in the network. This open circulation of data, happening across all network domains, including Access and Core, aims to improve the network operation through the usage of Artificial Intelligence (AI)-based solutions. This paper focuses on the interaction among Service Providers delivering applications to their customers using the infrastructure of Mobile Network Operators. In this scenario, it is paramount that such interactions occur with *limited trust*, as network operators and service providers may be competitors in the market, therefore avoiding the exchange of raw data and labels. In this work, we propose ATELIER, a deep learning solution for the provisioning of tailored network analytics from the network operator to the service providers in a limited trust fashion. Our design, which leverages *similarity learning* and *reinforcement learning* solutions, demonstrates how it can improve the analytics for a multimedia streaming service under various service configuration parameters, such as the video type and the desired Quality of Experience (QoE) level for the end users, achieving a performance increase of up to 37.7% compared to other methods, while also doubling the precision.

INDEX TERMS Zero-trust Networks, Network Analytics, Cooperative Learning.

I. Introduction

THE current approach of Service Provider (SP) deploying Over The Top (OTT) applications over Mobile Network Operator (MNO) is suboptimal for both MNOs and SPs. For the MNOs, it circumvents their billing systems and reduces them to "dumb pipes"; for the SPs, they cannot support traffic engineering via network re-configurations that would improve the performance of the service. With the arrival of 5G, the few supported interactions are limited to the exchanges of service templates for the deployment of network slices, and the situation is unlikely to change despite recent initiatives of open network Application Programming Interfaces (APIs) such as O-RAN [1], 5G-ACIA [2] or CAMARA [3].

The root cause of this de-coupling of the operations of the SP and the MNO lies in the lack of trust among them, due to several reasons such as, e.g., the MNO and SP being market competitors (for instance, in the case of *triple play* services), or the MNO not trusting SP-driven re-configurations or even the installation of specific modules in its infrastructure. But this lack of trust results in capacity overprovisioning, a non-sustainable approach. In fact, recent initiatives call for a tighter Network-Application Integration (NAI), to enable an information-driven management consistent with the changing network circumstances and rapid development of new technologies towards 6G.

To support a tighter integration while circumventing the above trust issues, in this paper, we introduce the idea of **limited-trust** collaborations between stakeholders. This type of collaboration is characterized by a *restricted* exchange of information between the parties, to prevent the leakage or inference of sensitive information (confidential, strategic, etc.). For instance, in environments where Artificial Intelligence (AI) is used to drive the autonomous operation of systems, a limited-trust collaboration forbids the exchange of e.g., raw data, labels, or even gradients from the training

models, while allowing the exchange of aggregated or less critical information.

We exemplify the operation of a limited-trust collaboration by considering the case of a network analytics service, which serves to illustrate how the MNO and the SP can collaborate to align their interests without disclosing critical information: on the one hand, the MNO provides a qualitative classification of flows (good vs. bad performance) without revealing the sensitive metrics used to compute this classification, while on the other hand, the SP helps the MNO in making this classification without revealing service-specific information. We validate this approach by designing and implementing a novel exposure interface that extends the vision presented in [4].

Overall, the major contributions of this paper are:

- We introduce the limited trust collaboration, a novel approach to tackle non-aligned or conflicting interests between stakeholders in mobile networking.
- We exemplify the approach for the case of a network analytics service by designing a novel framework based on deep learning, ATELIER, where both the MNO and the SP collaborate for a common interest while not disclosing critical information.
- We present the novel components used in the framework, namely, an algorithm for anomaly detection in network flows, based on a similarity learning technique, a data augmentation algorithm that overcomes biases typically found in certain training datasets, and a Reinforcement Learning (RL) solution that drives the alignment of the MNO metrics to the SP interests.
- We release as Open Source all the elements of the ATELIER framework, including the data generation tools, which are available on GitHub[1]

The rest of the paper is organized as follows. We introduce the generic problem of limited trust network analytics in Section II. Then we detail ATELIER in Section III, its training steps in Section IV, and discuss the evaluation results in Section V. The related work is described in Section VI before concluding in Section VII.

II. Limited trust networking

Mobile networks are witnessing a growing trend of integrating multiple parties into their architecture, a trend that will likely grow in the future [5]. In such an administrative fragment domain, the optimization of network operation (which is mandatory to achieve sustainability) becomes a daunting task. The integration of AI-based solutions over the network architecture [6] is already a challenging *coordination* problem of AI instances. In the following, we present our general approach to tackle this challenge, which is exemplified in the use case: Limited Trust Network Analytics.

[1] https://github.com/nokia/SNNExperimentFramework

TABLE 1: Characteristics of different learning approaches

Approach	Architecture	Knowledge	Interactions among elements
Single Server	Centralized	Common data and objective function	None
Centralized	Hub and Spoke	Shared Data	Data exchange
Multi-agent or Federated learning	Distributed	Objective function	Model exchange
Limited trust / Cooperative learning	Multiple instances	None	Steering signals

A. Achieving Limited Trust

Achieving the autonomous operation of a mobile network has been widely studied, where solutions roughly fitting into one of these categories: centralized approaches, where the training is performed on a single server that either has all the data or dynamically fetches from other entities, or multi-agent [7]/federated learning [8], where data remains distributed across nodes that run the training, and the model is exchanged across these nodes. We summarize in Table 1 the characteristics of these two paradigms in terms of the architecture, the information shared across nodes, and the interactions among elements. The centralized paradigm is obviously not suitable when there are different competing stakeholders and neither a multi-agent nor federated learning approach, since they require a tight interaction based on model exchanges that could unveil sensitive information from the corresponding datasets.

In Fig. 1 we showcase such interactions and the privacy threats exposed by the previously mentioned paradigms. The figure describes both a normal SP scenario, where the SP controls a generic video service and the MNO provides the network service, and also a triple-play SP scenario, where both the SP and MNO are providing the same service (such as a video service, the dashed part for the MNO) and therefore are business competitors. With both the (*Centralized* and *Multi-Agent/Federated Learning*) approaches, the distribution of possible sensible information is required. In the first scenario, data aggregation from all stakeholders is required to execute the learning procedure on top of it.

In the second scenario, the SP needs to share some of its internal intelligence to correctly tune the algorithms running in the MNO premises and vice-versa, causing thus a possible trust problem. Also, the MNO has to redistribute model information to coordinate with the SP. On top of all that the triple play scenario, where the SP and the MNO are competitors for the same service, is enough to stop any SP

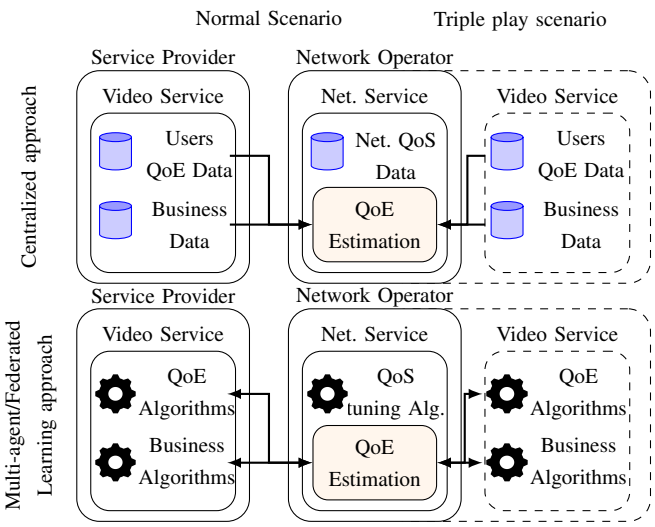


FIGURE 1: Illustration of possible privacy issues for the different scenarios

from sharing business-relevant information, as leakage or misuse of the information could end up in business damage.

Indeed, achieving limited trust scenarios in the application of Machine Learning solutions has been in the scope of the community in recent years. For instance works investigating privacy-oriented deployments of such technologies, like the one presented in [9], where the authors provide a procedure to enhance *federated learning* participants' privacy. A randomized mechanism is applied to hide the single clients' contributions to the overall model during the aggregation phase, enforcing *Differential-Privacy* [10]. This approach has also been followed in a more recent work, that builds in this direction, which has been presented in [11] where the usage of homomorphic encryption and multi-party computation ensures that the aggregator cannot access client model updates. Other security perspectives are also considered in [12] where the authors compare two differential privacy techniques considering the advantages and disadvantages of countering possible attacks. On the opposite, the work presented in [13] provides an overview of how differential-privacy techniques have been misused in Machine Learning (ML) evaluating then the trade-off between utility/privacy/efficiency achieving better results with standard techniques. Still, both solutions do not consider applications of ML solutions that involve limited trust between involved parties.

In general, the solutions aim to make secure and private the sender identity or the sender data transfer, but they do not solve the issue of trusting the centralized entity that has to compute the analytics, which still has to share a common view on the loss functions used to map the received feedback with the ground truth, that needs to be shared across the different parties, hence requiring full trust across them.

In this paper, we propose a novel *cooperative learning* paradigm, which is based on a *limited trust* operation between stakeholders. The key idea is that intelligent algorithms distributed across the network *cooperate* as much as possible to optimize performance, but without the exchange of data, functions or models that would require specific business agreements (i.e., trust) among the entities running them. In this way, by supporting partial interactions among elements from different stakeholders, these can signal each other towards the direction that could jointly optimize performance. In what follows, we describe how to implement this paradigm into the current mobile network architecture.

B. Study Item: Limited Trust Network Analytics

Here we exemplify a limited trust operation by enriching the interaction between an MNO and the SP to support a QoE-driven operation of networks. As mentioned above, both parties share only aggregated or limited information for confidentiality or privacy issues, with the common goal of optimizing the accuracy of the analytics. This is aligned with the current trends in architectural design, which envision more direct interactions between different players in the 5G ecosystem [4]. For instance, the 3GPP study items reported in TR 28.824 [14] define who, what, and how management services can be exposed to third parties, effectively enforcing three levels of access, which range from *baseline* (i.e., consumer access) to *hyperscalers* (more advanced control such as Quality of Service (QoS) Management).

One of the new interactions between the MNO and SP may revolve around Network Analytics. The Network Data Analytics Framework introduced by 3GPP [15] allows different network functions, including those from the SP, to access analytics that could be used to optimize performance. Since the default analytics provided by the MNO is oblivious of SP-specific QoE information, cooperation is required to learn the best approach to provide a tailored analytics service.

1) Objective and challenges

The objective is for the MNO to provide an analytics service that is tailored to the QoE of the SP. This service could both support a QoE-driven operation of the network and constitute a new revenue stream for the operator. Following the 5G architecture (although it could apply to other architectures), we envision that it can be implemented at the Network Data Analytics Function (NWDAF), with a catalogue of models tailored to each specific service and provider, as illustrated in Fig 2. There, we depict our reference scenario which is composed of a set of different services running on top of the same network. The network operator supports the different services through the Analytics Framework, initially with a *vanilla model*, which is then refined to a set of different *aligned models* that provide an optimized version

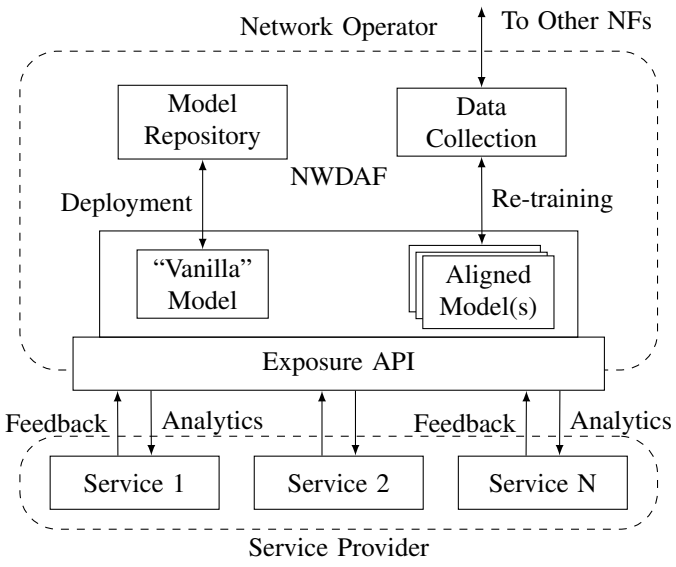
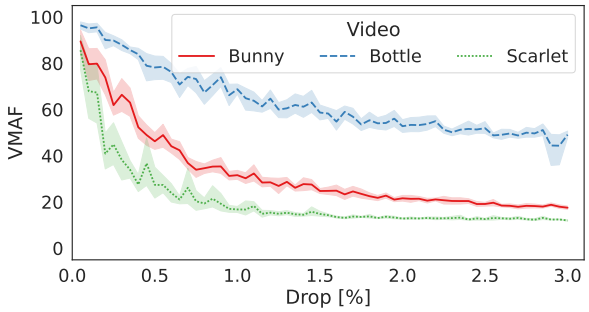
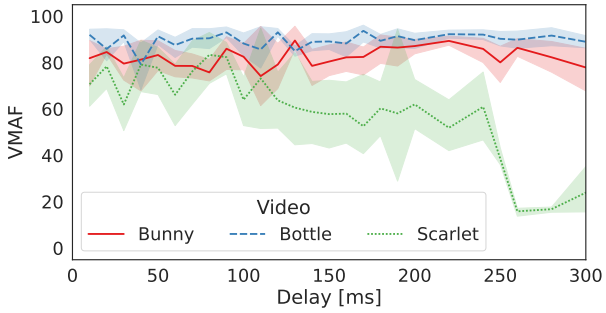


FIGURE 2: overview of the MNO-SP Loop. Please notice the (s) to identify possible multiple models, one for each service



(a) Impact of PDR on VMAF



(b) Impact of packet delay on VMAF

FIGURE 3: Impact of network perturbations on QoE. The line represents the average from 10 experiments, the shadowed area represents the 95 percentile

of the analytics for the specific service under consideration. To support this vision, we need to address two related but conflicting challenges:

1) The mapping between QoS and QoE follows non-linear and multivariable behaviour. We illustrate this in Fig. 3), where we plot the impact of losses (top) and delays (bottom) on the resulting video quality of three different videos, following the experimental setup described in Section A. This further confirms the need for tailored approaches (i.e., there is no *one-size-fits-all* solution) and calls for the use of data-driven and therefore time- and resource-consuming approaches, such as the one proposed in [16].
2) The transmitted QoE and QoS information is costly and sensitive. Neither the MNO nor the SP are likely to share such type of raw data, especially with possible competitors.

In what follows, we present an approach to overcome these challenges and support QoE-tailored analytics, which builds on top of the concepts introduced in Section 2:

Limited trust: there is no exchange of raw data in any of the two directions, i.e., the MNO shall not send raw QoS metrics or metadata that could leak sensitive information (e.g. about User Equipment (UE) location or network topology) and the SP shall not send QoE metrics (i.e., labels) that could leak internals related to users' performance or the service architecture.

Cooperative learning: the limited trust operation precludes the use of a centralized or multi-agent/federated approach. In contrast, we propose the use of two intelligent algorithms that interact with each other towards a common goal.

2) Limited trust

As we have illustrated in Fig. 3, the complexity of QoE forecasting from current network QoS statistics calls for the usage of machine learning. The most straightforward approach would be to train a model using QoS and QoE raw information, but this breaks the *limited trust* principle discussed above.[2] Following this, each party reveals only the strictly needed information: instead of reporting per-flow or per-user information, one side may convey only aggregated metrics (e.g., summary statistics) that obfuscate detailed information. For instance, the MNO does not report per-flow fine-grained QoS statistics, but only a coarser estimation (e.g., *good* vs. *bad*) of their performance (in this paper we focus on the specific case of tailored analytics, but our vision could also be applied to other scenarios with different privacy-preserving mechanisms).

Since no entity has access to the raw data to perform the training, we cannot expect a single intelligent component,

[2]For instance, if the training is performed by the MNO, *labels* shall be sent from the SP; if the training is performed by the SP, it needs raw QoS information.

either centralized or spread across the MNO and the SP. Hence, we require two separate intelligent entities that cooperatively learn to achieve a common goal, which we address next.

3) Cooperative learning

We next describe how we implement a cooperative learning approach between the MNO and the SP to support a tailored network analytics service. Building on top of the seminal work performed in [4], we illustrate how these two stakeholders and corresponding interact through the components depicted in Figure 2, where the MNO provides analytics to the SP (down arrows) and the SP provides feedback to iteratively improve these analytics (up arrows), which are specific to each service.

In the beginning, the MNO trains a *"vanilla" model* using only QoS information, starting from a pre-configured model best suited to the service (chosen from a repository). This trained model translates the detailed (and sensitive) QoS quantitative statistics into some aggregated or qualitative information, which is passed to the SP. In addition, the model exposes one or more parameters that influence the training for the SP to steer the results from the training (this can be implemented via e.g., an *Exposure API* layer)

On the SP side, the results from the analytics service are then processed along with other sensitive information which could include business-related metrics (e.g., running costs) or the perceived quality according to a model. This processing includes the training of a different model, to optimize performance according to the SP requirement. Based on the results from this model, the SP provides a new set of parameters for the training of the model at the MNO. In this way, both the MNO and SP are unaware of the internals of the algorithms run by each other, including the detailed and sensitive information used, thus achieving a limited trust operation. Furthermore, this approach enables a dynamic operation where changes in the computation of network conditions by the MNO or in the policies to optimize the service by the SP can be seamlessly adopted.

III. Use case: video streaming analytics

Here we instantiate the framework introduced above for the case of tailored video streaming analytics. The motivation is the "Abnormal UE behaviour" analytics service provided by the NWDAF [17], which classifies traffic flows as *bad* or *good* depending on whether they experience impairments or not, respectively. The outcome of this categorization can be used for many purposes, both from the MNO side (e.g., re-configuration of radio bearers, QoS enforcement, re-orchestration of resources) or the SP side (e.g., change of video encoder). But this classification cannot be performed by the MNO alone, since only the SP knows the ground truth about QoE (using e.g., Mean Opinion Score (MOS) models [18], or Video Multi-method Assessment Fusion (VMAF) [19], [20]), and the traffic could be end-to-end encrypted. Furthermore, the SP is also interested in an accurate classification, since this would result in a better handling of the flows and eventually higher user satisfaction.

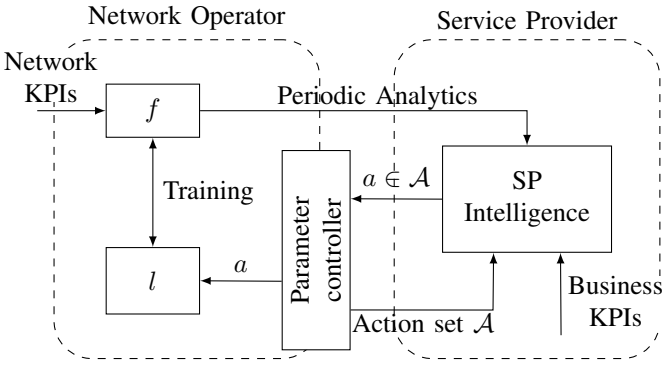


FIGURE 4: Overview of the MNO-SP cooperation. The Action set represents the configurable parameters offered by the MNO.

Given the complex relationship between QoE and QoS for the case of video already illustrated in Fig. 3, both the MNO and the SP need to cooperate to optimize performance.

A. Solution overview

To solve the above, using the framework discussed in Sec. II, we need the SP and the MNO to collaborate by learning from each other. We depict the overview of our solution in Fig. 4, which is based on the following components:

On the MNO side, a classification *model* f discriminates between good and bad flows based on network Key Performance Indicators (KPIs). Assuming that the vast majority of the flows have *good* QoS statistics, this discrimination is implemented with an *anomaly detection* algorithm, which periodically provides results to the SP.[3] The MNO also exposes a configurable parameter that the SP uses to *steer* the anomaly detection algorithm by altering its sensitivity.

On the SP side, its intelligent module collects the analytics produced by the anomaly detector and compares them against the (confidential) QoE business metrics. Based on this comparison, and taking into account the possible action set over the configurable parameter of the MNO, a *steering algorithm* determines a new value of this parameter to improve the anomaly detection algorithm.

In the following sections, we describe the detailed design of the different components described above and illustrated in Figure 4, while the description of the system operation is presented in Sec. IV.

[3]Our solution can be easily coupled with a orchestration algorithms to improve QoS, but we leave this out of the scope of the paper.

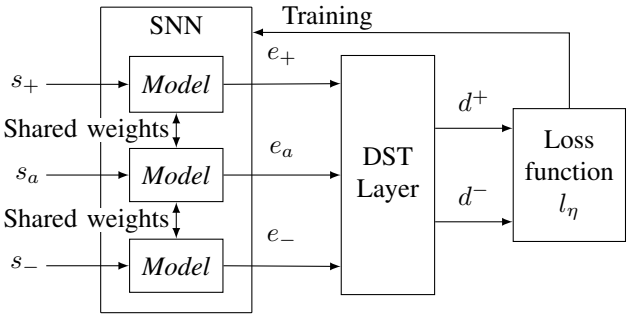


FIGURE 5: SNN f with three inputs: positive and negative references, and the anchor.

B. MNO: steerable *anomaly detection*

An anomaly detection method assumes that the vast majority of flows do not suffer any impairments, i.e., they are *good* flows, and only a tiny fraction of flows receive a service below their requirements and therefore are classified as anomalous or *bad*. This results in a dataset very unbalanced across the two classes, which makes it very challenging to classify using classic techniques [21].

To overcome this challenge, we design the anomaly detection using *similarity learning*, inspired by the recent advances in image recognition [22], [23]. With similarity learning, data features are projected into *embeddings* in a latent space, which are then used to classify data as similar or dissimilar, depending on relative distances. A major benefit of similarity learning in contrast to classifiers, is that it only requires examples of objects that are similar and dissimilar, instead of class labels which are often very costly to obtain.

We depict the anomaly detection *model* f in Fig. 5, which is composed of two main blocks (highlighted in a dashed box): (1) a Siamese Neural Network (SNN) to project the features of a given set of flows $\{s\}$ $\{s_0, \ldots, s_n\}$ into embeddings in the latent space $\{e\}$ $\{e_0, \ldots, e_n\}$, and (2) a distance (DST) block that computes the distance between the embeddings. The anomaly detection *model* f is then trained with a tailored loss function to detect anomalies while providing the SP with the ability to steer the learning process. We next describe the design of these two blocks.

1) Similarity Learning

A Siamese Neural Network is a type of neural network architecture very popular in similarity learning because of its ability to learn from few data and its robustness to class imbalance. An SNN contains $n \geq 2$ neural networks with the same parameters and working weights, takes as input n feature vectors and produces as output n embeddings, one per input, to learn their semantic similarity. In our case, the SNN has three networks that receive as input three samples corresponding to three different flows: a positive reference s_+, an *anchor* s_a, and a negative reference s_-. When performing the training phase, flows that are considered similar are used as s_+ and s_a, while flows that have different characteristics are used as s_-. The key objective of the training is to generate the embeddings such that the distance between the embedding generated by the positive reference (e_+) and the anchor (e_a) is minimized, while the distance between the negative reference (e_-) and the anchor is maximized.[4] We next describe how to leverage on these distances to generate a loss function l to train the SNN.

2) Steerable loss function

The loss function drives the generation of the embeddings. To this aim, it takes the two inputs from the *DST Layer*: the Euclidean distance[5] between the embeddings generated by the positive reference (e_+) and the anchor (e_a), denoted as d^+, and the distance between the negative reference (e_-) and the anchor, denoted as d^-. In order to maximize d^- while minimizing d^+, we use the following loss function:

$$l = \max(0, \|d^+\|_2 - \|d^-\|_2 + \eta) \tag{1}$$

which includes, in addition to d^+ and d^-, the additional term η that enforces a minimum distance. This loss function simultaneously moves positive embeddings close to each other and the negative embeddings at (at least) the margin η. Furthermore, although Equation (1) must be fixed during the training process, changes to η affect how the training process is driven and hence on the produced analytics: larger η values further separate e_a from e_- and move it closer to e_+, while low or negative η values have the opposite effect. Following this, although it would be possible to interact with a model in several ways (e.g., layer dropout levels, batch dimensions), in our case we decided to support tailoring the analytics via the adjustment of η. More specifically, the MNO exposes η to the SP after each training cycle, i.e., the loss function is represented as

$$l(\eta_t) = \max(0, \|d^+\|_2 - \|d^-\|_2 + \eta_t) \tag{2}$$

which emphasizes that η_t is a dynamic parameter that changes over time (t) as specified by the SP. This approach also keeps the MNO in control of the exposed parameters (so it prevents malicious behaviour). It supports service differentiation between multiple SPs by, e.g., having multiple models for different SPs, different geographical locations, and/or different times of the day/week. Different models can be easily differentiated and adopted through APIs including a SP id. In what follows, we design the algorithm at the SP to compute this parameter.

C. SP side: business alignment

The above specifies how the MNO uses similarity learning to detect abnormal flows and report them to the SP (it could

[4] First SNN techniques employed n=2 and "contrastive loss" [24], [25] to instruct the network to separate *genuine* and *impostors* pairs, but nowadays it is more common to use the so-called *triplet loss* [22], [26].

[5] Other distance measures could be used.

also perform corrective actions, but we have left these outside the scope of the paper). We next detail how the SP is able to *steer* the detection algorithm at the MNO, to align it with the real QoE metrics instead of network QoS.

1) Service oriented metric

The SP requires a metric M to assess the accuracy of the anomaly detection performed by the MNO and react accordingly. Such analytics requests, according to [15] section 6.4, can be done individually per UE or group of UEs, slice, application/s, specific User Plane (UP) paths etc. This gives the capability to the SP to correlate its own local information with what it receives from the MNO and also to control the size of the analytics traffic. M is computed by comparing the analytics received with the real labels, which are only known to the SP. To obtain these labels, in this paper, we rely on the established VMAF estimator [20] that we have used in the past [27], and define a simple policy based on a threshold v_{min} to distinguish between *satisfactory* flows (VMAF $> v_{min}$) and *unsatisfactory* flows (VMAF $< v_{min}$). Note that this threshold v_{min} may depend on the specific provider and service (e.g., sports video streaming may have lower requirements than a movie on-demand video).

Following the above, for each iteration, the SP computes the *confusion matrix* by comparing the analytics from the MNO (good vs. bad) with the computed labels (satisfactory vs. unsatisfactory):

- True positives (TP): good and satisfactory flows.
- True negatives (TN): bad and unsatisfactory flows.
- False positives (FP): good but unsatisfactory flows.
- False negatives (FN): bad but satisfactory flows.

Based on these numerical values, the SP computes M as follows:

$$M = \frac{TP + TN}{TP + TN + 2FP + FN} \tag{3}$$

where we double the impact of false positives ($2FP$ term in the denominator), since we assume a higher relative cost for classifying as *good* those flows that experienced a poor quality.[6] All variables in Eq. (3) are counts, e.g., TP represents the number of TP flows.

The SP objective is then to maximize M through the *steering signal* η_t exposed by the MNO (see Eq. 2). This challenge matches a RL scenario, where the SP takes actions (i.e., changes η_t) to improve performance while the MNO represents the environment that changes after the actions. We next describe the design of the RL algorithm that the SP runs to maximize M.

2) Reinforcement Learning algorithm

As described in Section B, the MNO uses a triplet loss function $l(\eta_t)$ that exposes the η_t parameter so the SP can tailor it to its needs. To compute the optimal value for this parameter, we assume that at each step the SP takes an action from the following action set $\mathcal{A} = \{a_+, a_=, a_-\}$ to update the current value of η_t:

- a_+ increases η_t by Δ_a;
- a_- decreases η_t by Δ_a;
- $a_=$ leaves η_t unchanged.

To select the best action, we use the Advantage Actor-Critic (A2C) algorithm [28]: an actor-network computes a probability distribution for the next action to take, while a critic network defines the quality of the decision taken. For simplicity, we focus on a single service and therefore a single actor, note that with this approach it is straightforward to train multiple actors (e.g., one per service) using the same critic. In our implementation, both the actor and critic share the same neural network π_θ.

The A2C is trained through episodes, where each episode consists of T steps and uses the same QoS dataset. Each step is made up of the following sub-steps, illustrated in Fig. 6: (1) the MNO cooperatively trains its model f and produces a new set of analytics; (2) the SP uses these analytics to update:

- The business metric M.
- The system state σ_t, defined by: the current value of η_t, the confusion matrix at time t, and the last $K < T$ values of the confusion matrix.

Based on this information, (3) the RL algorithm computes a new reward r_t, which is based on a discounted sum of M over the last K steps (we use a value low enough to allow

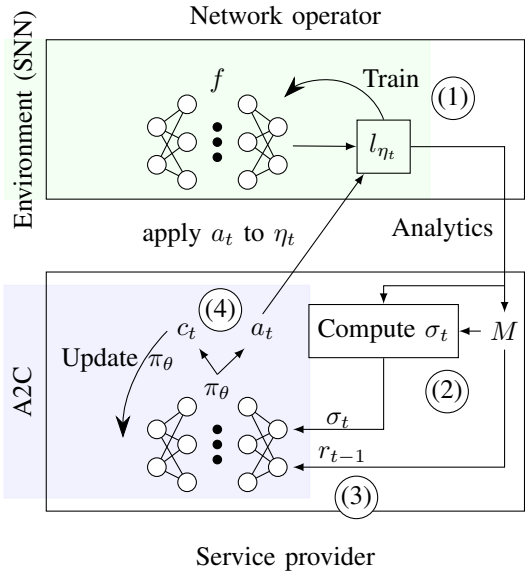


FIGURE 6: Visual representation loss alignment process through RL

[6] In our paper, M is inspired by the well-known accuracy metric, but others (sensitivity, precision, F1 score, etc) could be used.

multiple iterations inside a single episode), i.e.,

$$M_{t+1} = \sum_{j=1}^{K} \gamma^{K-j} M_t^{(j)} \tag{4}$$

$$r_t = \text{sign}(M_{t+1} - M_t) \tag{5}$$

where $M_t^{(j)}$ represents the j sub-step values of M and γ the discount factor. Based on this, (4) the actor outputs an action $a_t \in \mathcal{A}$, and the process is repeated. At the end of the episode, the overall reward R is computed as

$$R_t = r_t + \gamma R_{t-1} \tag{6}$$

The overall objective of the RL is to maximize the discounted reward R, obtaining thus a version of f that can be used in inference to provide the service-tailored analytics.

IV. ATELIER Prototype and System Operation

As discussed in Section III, the interaction between network operators and SPs involves several steps that together drive the system towards optimal operation. We detail them next in the context of the dataset-gathering process.

A. Implementation

We implement our solution with an emulated mobile network through state-of-the-art tools such as `Docker` and `srsRAN`. The software modules are orchestrated through a Python controller, which automatically generates the experimental configuration, and runs the experiment. The purpose of this prototype is two-fold: first, to generate data from an experimental testbed, where we collect both QoS and QoE KPIs, and second, to prove the feasibility of the system as it implements a state-of-the-art 3GPP mobile network architecture. The details of the experimental setup are given in Appendix A.

B. Dataset generation

Throughout our experiments, we use three videos `Bunny`, `Bottle`, and `Scarlet`, that have different characteristics in terms of bandwidth as they are encoded with `h264` which produces different bitrates according to the complexity. Each video is split into shorter samples prior to their stream, and each stream is artificially impaired by introducing packet and byte errors, and changes to its delay. The impact of some of these impairments on the resulting VMAF was already illustrated in Figure 3. The figure illustrates that the same impairment impacts different videos in different ways, but there are two common areas where the relation between the impairment and the VMAF is clear: when there is no impairment (left part of the figure) or when the impairment is large (right part of the figure). These two areas serve as the starting point for the anomaly detection, as discussed next.

C. Data labelling

The similarity learning framework (Section 1) requires two reference sets, one positive and one negative. Hence, the MNO has to perform an initial discrimination of the flows, inferring the expected QoE based on the QoS. To this aim, and following the results discussed above regarding Figure 3, we assume that it is accurate to infer very good QoE when the QoS metrics are very good, and to infer very bad QoE when the QoS metrics are very bad. In this way, we assume that the operator specifies a set of "acceptable thresholds" and "unacceptable thresholds" for each networkingKPI, and proceed to classify flows into three sets:

- $\hat{G}$ (Very likely good): flows where all network KPIs are within the acceptable thresholds.
- $\hat{B}$ (Very likely bad): flows where at least one networkKPI is larger than an unacceptable threshold.
- $\hat{U}$ (Undecided): flows that are neither in $\hat{G}$ or $\hat{B}$.

1) Baseline

The most immediate approach that the MNO can use to train the model f is to use a dataset D_t that only includes samples from $\hat{G}$ and $\hat{B}$, randomly picked according to the relative cardinality of the sets. In this way, if s_a is picked from $\hat{G}$, then s_+ is also chosen from $\hat{G}$ and s_- is chosen from $\hat{B}$; correspondingly, if s_a is picked from $\hat{B}$, then s_+ is also chosen from $\hat{B}$ and s_- is chosen from $\hat{G}$. We refer to this approach as *Baseline* since it is the most direct strategy to use that does not make any further assumptions on the relation between QoE and QoS. It has the main drawback of not taking advantage of the samples in the undecided set $\hat{U}$, which is particularly critical since, in general, the operation conditions will be such that most of the flows are provided with good quality, and therefore the cardinality of $\hat{B}$ will be very small, which challenges the training (e.g., [21]). We next describe how to take advantage of $\hat{U}$ and enlarge $\hat{B}$.

2) Self-Labelling

The objective is to exploit the information in the undecided set $\hat{U}$ to augment the training data with more cases, in particular for the set $\hat{B}$. To this aim, we follow a technique similar to the one presented in [29] for a different challenge (adapting to changing conditions), which we denote as an unsupervised Self-Labelling (SL) approach. It consists of the following steps:

1) f is trained using D_t (i.e., $\hat{G}$ and $\hat{B}$).
2) For each sample $u \in \hat{U}$ generate a tuple (s_+, u, s_-) picking s_+ and s_- randomly from $\hat{G}$ and $\hat{B}$ respectively.
3) Use the generated tuple as input for the model f, just trained over D_t, to compute the distance between u and s_+ denoted as $d^+(u)$ and from the sample s_-, denoted as $d^-(u)$
4) If the sample u is very close to s_+ (i.e., $d^+(u) < d^-(u) - \eta$), it will be marked for inclusion into $\hat{G}$; if the sample u is very close to s_- (i.e., $d^-(u) < d^+(u) - \eta$), it will be marked for inclusion into $\hat{B}$.

Algorithm 1 SP metric alignment

$f \leftarrow model\ initialization$
$\pi_\theta \leftarrow A2C\ model\ initialization$
$\eta_0 \leftarrow \eta$ ▷ η_0 default initialized
$D_t \leftarrow Split(\hat{G}, \hat{B})$ ▷ Properly split $\hat{G}$, $\hat{B}$ into D_t
repeat ▷ Execute the A2C training cycle
  $f \leftarrow Train(f, D_t)$ ▷ Train f over D_t
  while $i < T_{SL}$ **do**
    $\tilde{G}, \tilde{B} \leftarrow SL(f, D_t, \hat{U})$ ▷ Execute 2 separation
    $D_t \leftarrow Split(\tilde{G}, \tilde{B})$
    $f, \pi_\theta \leftarrow Train(f, D_t, \pi_\theta)$ ▷ Train both π_θ and f
  end while
  $f, \eta_0 \leftarrow reset$
until π_θ Convergence
$Freeze\ \pi_\theta$ ▷ π_θ converged

Algorithm 2 Steerable anomaly detection

$f, \eta_0 \leftarrow reset$
$D_t \leftarrow Split(\hat{G}, \hat{B})$ ▷ Properly split $\hat{G}$, $\hat{B}$ into D_t
$f \leftarrow Train(f, D_t, \pi_\theta)$ ▷ Train f on D_t using π_θ
while $f\ not\ trained$ **do**
  $\tilde{G}, \tilde{B} \leftarrow SL(f, D_t, \hat{U})$
  $D_t \leftarrow Split(\tilde{G}, \tilde{B})$
  $f \leftarrow reset$
  $f \leftarrow Train(f, D_t, \pi_\theta)$
end while

5) After all samples from $\hat{U}$ are processed, include the new samples into $\hat{G}$ and $\hat{B}$.
6) Restart from step 1 until there are no more samples in $\hat{U}$ or the maximum number of repetitions T_{SL} is reached.

We will assess the impact of this approach in Section 2.

D. Cooperative algorithm

The last step in the learning process is the one needed to align f with the quality metric M from the SP through the algorithm depicted in Section C with the parametrized loss function defined in Eq. (2). These steps are summarized in Algorithms 1 and 2, which describe the two major blocks of the algorithm.

The training happens in two consequent phases which are used to train both the networks used by the A2C and the Siamese, π_θ and f respectively, freezing one of the two and training the other network.

1) Algorithm overview

The first part of the algorithm is described by Algorithm 1. This part is responsible for the cooperative training with a focus on π_θ. So, the final expected outcome is a network π_θ capable of making decisions to achieve the optimal η_t^* using as input the MNO's provided analytics and the SP's business metrics. After this part, the intelligent part of the SP is considered completed and therefore *frozen* to be used in the second algorithm only in inference.

In the second phase, presented in Algorithm 2, the MNO is actively training its anomaly detection *model* f with the help of the SP intelligent system. The outcome of the algorithm would be a *model* tuned by the SP following its needs, maximized through π_θ. Both algorithms are discussed in detail in the following section.

2) Steering anomaly detection with SP Metric

The two models, π_θ and f, cooperate in both Algorithms 1 and 2, but not only do the algorithms differ in the overall goal the inner procedures are also different. We are going now to analyse the first algorithm, Algorithm 1, in the preliminary phase the main components are initialized: both models f and π_θ with random weights, η_0 set to the default value, and the training dataset D_t with the *Baseline* approach separation. D_t is balanced according to the requirements selecting elements randomly from $\hat{G}$ and $\hat{B}$[7]. The more external cycle in Algorithm 1 runs until the π_θ model is considered *converged*, i.e., fully trained. There are two possible situations in which π_θ is considered trained:

1) If, on average, the reward over the last $RL_{R_{SGD}}$ training iteration is kept above a certain threshold RL_c;
2) The maximum number of iterations T_{RL} is reached and the model is considered trained anyway.

This is because, during the first exploration phase, the RL process may use non-optimal action probability distributions, leading to sub-optimal η_t values. Inside the RL training loop the episode evolves as follows:

1) The network operator initially trains f using D_t following the *Baseline* approach described in Section 1;
2) For T_{SL} the procedure described in Section 2 is repeated, augmenting D_t using $\tilde{G}$ and $\tilde{B}$ and executing the *cooperative* training loop with both π_θ and f.

During the *cooperative* training function, the analytics are provided to the SP, in batches of size β_{SP}. For each item (that for the considered use case represents a video flow) the label, *bad* or *good*, selected by the MNO is provided. The SP computes M following Eq. (3), and performs the training of π_θ, determining the output action that is applied to η_t.

Once the training of π_θ is concluded, the second algorithm, i.e., Algorithm 2, takes place. The outcome of this

[7]As the working assumption is to have more *good* samples than *bad* ones, the training dataset is computed accordingly.

process is the actual tailored model f with the feedback provided by the SP, as at the beginning of the algorithm the *model* f is reset and the dataset D_t is regenerated. At this point, π_θ is given as trained and therefore used only in exploitation mode. The MNO can now execute the *Baseline* and SL (repeated at most T'_{SL} times) processes with the help of the π_θ network from the SP that dynamically updates the η during the training.

Further details on the design and operation of the learning systems are available in Appendix B.

V. Experimental Evaluation

In this section, we evaluate ATELIER performance along three different axes: first, we evaluate the impact of the labelling process, then, we analyze the performance of the proposal, and finally, we explain the internals of the algorithm. By evaluating different videos with different QoE requirements, we empirically showcase the adaptability and generalizability properties of the solutions in a broad setting range.

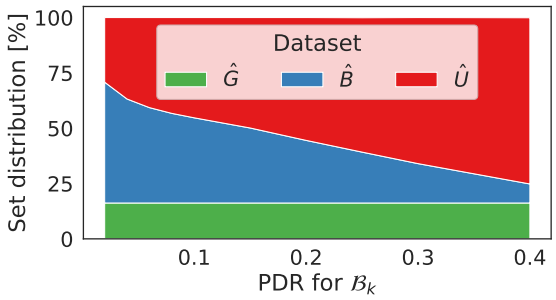
A. Labelling strategies

As discussed in Section C, the MNO first performs a baseline labelling of samples based on a set of QoS thresholds, which is then refined using a Self-labelling approach. Here we illustrate the impact of these two mechanisms on the resulting datasets.

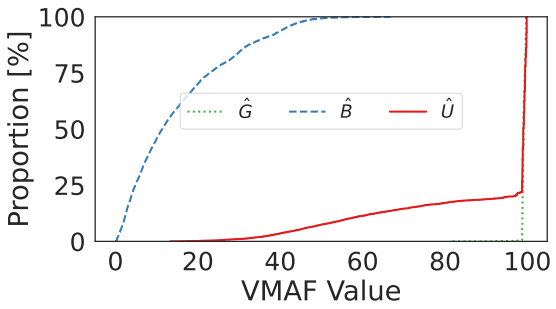
1) Baseline labelling

We first illustrate the operation of the baseline classification. To this aim, we follow the methodology described in Appendix A to emulate the transmission of videos under a variety of circumstances. Next, we analyze the impact of the choice of the PDR threshold for bad performance on the resulting cardinality of the $\hat{G}$, $\hat{B}$, and $\hat{U}$. We illustrate the relative distribution across these sets in Figure 7a for the `Bunny` video, which shows that, as the Packet Drop Rate (PDR) threshold becomes larger, fewer flows exceed this threshold and therefore can be clearly classified as bad flows, thus being assigned to $\hat{U}$. For the QoS values that we assume in this paper (Table 4), the relative distribution for the `Bunny` video is as follows: 8% for $\hat{G}$, 40% for $\hat{B}$, and 52% for $\hat{U}$. Since the values in Table 4 include more parameters than just the PDR, there are more flows in $\hat{B}$ than the ones in Figure 7a.

We next validate the initial assumption that flows with very good QoS metrics (i.e., those in the $\hat{G}$ as generated above) have very good QoE performance as quantified by the VMAF metric. To this aim, we compute the VMAF values of the `Bunny` video and depict their Cumulative Distribution Function (CDF) per each set in Figure 7b. The figure confirms that practically all flows that are classified as very good ($\hat{G}$) have a VMAF well above 90, while more than 90% of those close classified as bad ($\hat{B}$) have a VMAF



(a) Effect of the Packet Drop Rate (PDR) on datasets distribution



(b) ECDF of the QoE in the 3 sub-Datasets

FIGURE 7: Dataset distribution ECDF over the sample VMAF value, video `Bunny`

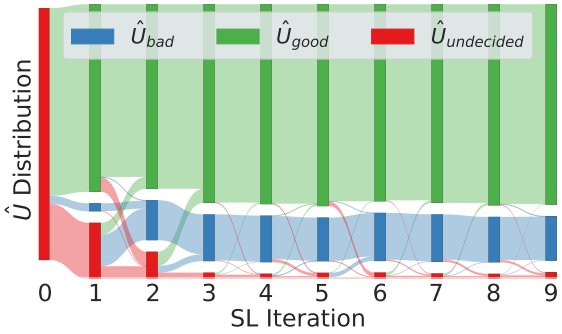
below 40 and no flow reaches a VMAF above 80. Flows from the undecided set $\hat{U}$ exhibit variability between these two cases: approx. $3/4$ of them have a VMAF close to 100, while 20% have a VMAF below 60. These results confirm that the baseline approach effectively discriminates between very good and very bad flows, but leaves a relatively large set with a mixture of them. We next illustrate how the self-labelling approach tackles this challenge.

2) Self-labelling

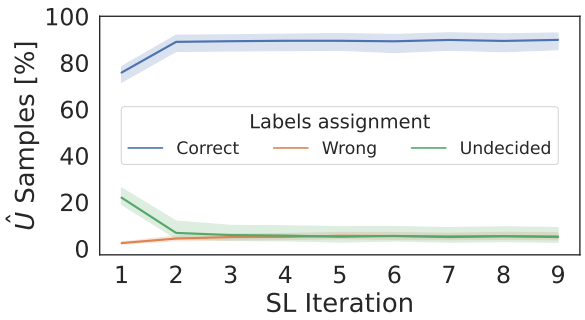
As discussed in Section 2, the objective of self-labelling is to enlarge the training set by assigning samples from $\hat{U}$ to either $\hat{G}$ or $\hat{B}$ (if possible). To illustrate its operation, we analyze the evolution of the cardinality of the different datasets over 9 iterations for the case of the `Bottle` video in Fig. 8a. As explained in Section 2, at every SL iteration f self-assesses the dataset to gain more insights on the internal sample distribution.

Already after the first SL iteration, the initial bulk of undecided samples is left to a very residual percentage, and after the third iteration, the amount of residuals in the $\hat{U}$ is a small fraction compared to the initial set, less than 3% of the initial $\hat{U}$.

To gain insight into the benefits of self-labelling, we analyze how many samples have been labelled correctly/incorrectly (or simply undecided) during the SL pro-



(a) Single experiment, labels distribution after each SL-iterations.



(b) 50 Experiments, labels correctness after each SL-iteration. The shadow represents the 95% confidence interval.

FIGURE 8: The time trends of the SL algorithm. Video `Bottle`, $v_{min} = 80$

cess independently of the assigned set $\hat{G}$ or $\hat{B}$. The results are presented in Fig. 8b where we show the evolution of 50 experiments, where the number of samples for each category has been normalized over $|\hat{U}|$. We illustrate that the classification becomes more accurate over time with an incremental trend for the correctly libelled samples and an opposite one for the undecided ones. While the wrongly labelled ones are a minor part of the set.

B. ATELIER performance evaluation

In this section, we analyze the performance obtained by ATELIER for heterogeneous scenarios comprising three different videos and three different VMAF thresholds for QoE, namely, $v_{min} = \{80, 90, 99\}$. In this way, we test ATELIER across a variety of scenarios and service profiles. We first provide a performance evaluation of ATELIER, and then deep into its internals.

1) Accuracy

We evaluate the performance of our system by computing the accuracy obtained by ATELIER, comparing it with two benchmarks: the *Baseline* approach discussed in Section 1, and the SL procedure discussed in Section 2. This allows us to show that ATELIER can reliably adapt to multiple different situations and can drastically improve the performances of a closed system controlled only by the MNO.

In Fig. 9, we compare the accuracy obtained by the different approaches, for the different videos and values of v_{min} considered. In all the scenarios, ATELIER outperforms the other solutions. In some cases, the improvement is limited as the analytics provided by the network operator are already well aligned with the provider metric. This is the case, for instance, of Fig. 9a where $v_{min} = 80$. According to Netflix [30], values around 80 can be considered as *fair*, so in a complex video like `Bunny`, this is already enough to distinguish between these categories. However, in other cases, the difference between the *Baseline* approach and ATELIER could be as high as 37.3%, like in the case of the `Bottle` video for $v_{min} = 99$.

2) Precision

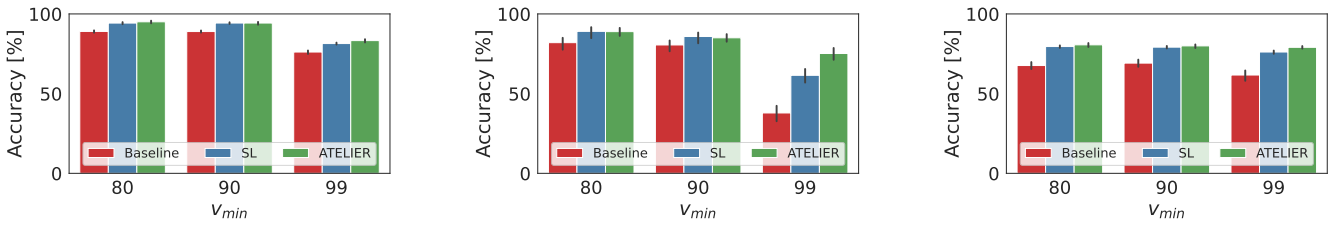
As discussed in Section 1 one of the main goals of the ATELIER RL framework is not only to steer the analytics towards better accuracy but also to do so by improving the performance on the FP cases, which we consider as the most negatively impacting scenario. We showcase ATELIER's effectiveness in this task by analyzing the precision (the ratio between TP and the sum of FP and TP) for all scenarios in Fig. 10c.

The results prove how ATELIER can significantly improve the detection of anomalous flows for all considered scenarios, in some cases achieving a $2\times$ increase compared with the *Baseline* approach. This confirms the ability to obtain different optimal points even with limited trust between the SP and the MNO.

C. ATELIER internals

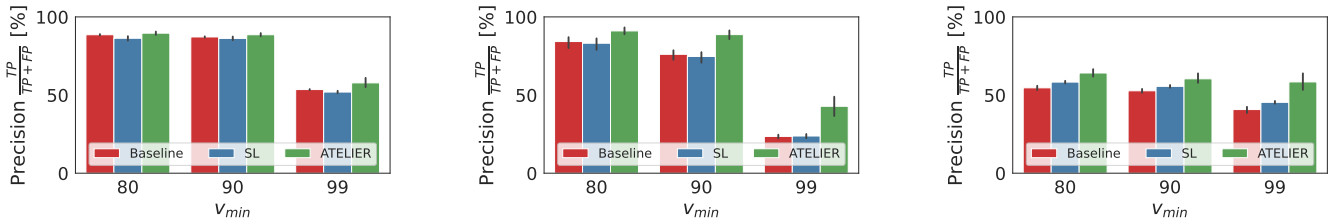
1) RL decision process

We start by discussing the ATELIER RL operation depicted in Fig. 11 for a given experiment. Fig. 11a shows how the average reward r_t of the last $RL_{R_{SGD}}$ steps evolves during consecutive training episodes. The trend is positive until it stabilizes during the last episodes when the training is stopped. This effect is obtained thanks to the corrective actions performed to the η adopted during the training from the SP. Fig. 11b shows how η increases from the initial values up to the ones that yield the best reward. Giving a closer look at π_θ in Fig. 11c we can see that this behaviour during the episode originated from the probability assigned to the different actions in $\mathcal{A}$ by the *actor* network. While a_- suddenly drops, a_+ and $a_=$ exchange their relative weights and find an equilibrium where the most probable actions are keeping η stationary.



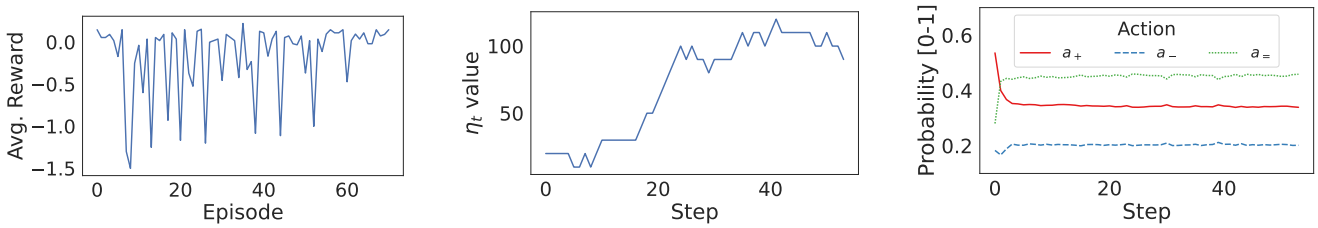
(a) `Bunny` (b) `Bottle` (c) `Scarlet`

FIGURE 9: ATELIER accuracy compared with Baseline and SL. 50 experiments average and 95% confidence interval.



(a) `Bunny` (b) `Bottle` (c) `Scarlet`

FIGURE 10: ATELIER precision compared with Baseline and SL. 50 experiments average and 95% confidence interval.



(a) RL-training, Avg. reward evolution (b) Margin evolution (c) Act. p. evolution

FIGURE 11: Video `Bottle`, $v_{min} = 80$, exp. number 8

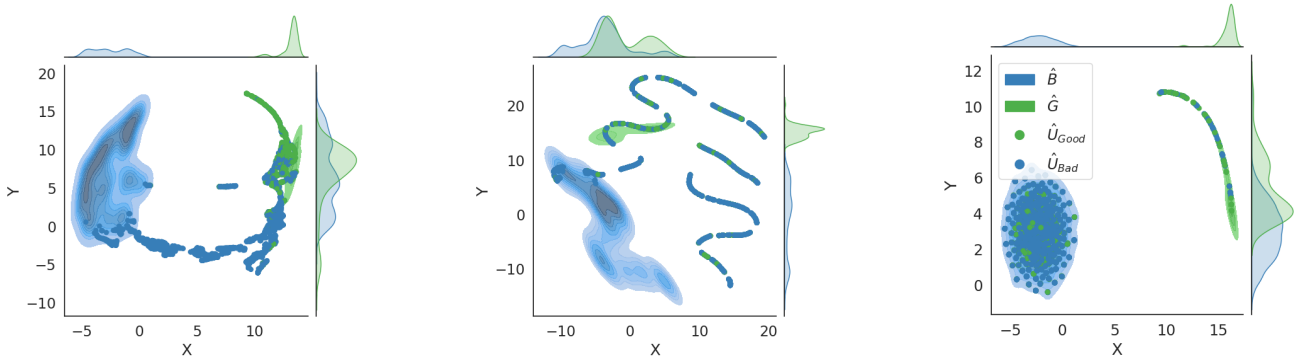
2) ATELIER embeddings

We conclude the evaluation results by understanding *how* ATELIER obtains superior performance when compared to the benchmarks. As discussed in Section C, by steering η, ATELIER can effectively exclude samples combination from the training and tell others further apart. This is reflected by the position in the latent space of the embeddings generated by the SNN. Due to their high dimensionality (4D), we represent them in 2D using the Uniform Manifold Approximation and Projection (UMAP) [31] algorithm in Fig. 12.

Fig. 12a shows the embedding placement for the *Baseline* approach, where many undecided samples (dots) are located outside the areas where the *good* and *bad* embedding are placed (the green and blue heatmaps), leading to classification errors. While the SL, shown in Fig. 12b, improves this situation by better distributing undecided samples, the situation is completely different in Fig. 12c, where the ambiguity between the two cases has completely disappeared and the anchor dataset is effectively split into samples that are closer to the *good* area and samples that are closer to the *bad* one, with only a few samples that belong to the wrong set, leading to the increased performance discussed in Section 1.

VI. Related Work

The view discussed in this paper goes well beyond the one of network slicing proposed by 5G standards. Allowing the (3rd party) SPs to personalize the network behaviour (such as the analytics production) according to their needs extends the network-slicing concept by enabling competition in a limited trust environment. Besides the analytics production, this concept has a number of applications ranging from the Internet of Things, especially in an industrial environment (such as the architectural framework proposed by the 5G-ACIA [2] organization) or to allow a wider QoS management



(a) Baseline (b) SL (c) ATELIER

FIGURE 12: Uniform Manifold Approximation and Projection (UMAP) embedding representation comparison after training over the same identical datasets, Video `Bottle`, $v_{min} = 99$. Dots color represent the expected ground truth label.

directly using open API, as promoted by the CAMARA [3] project.

Our study spreads across multiple domains, from the mobile network QoE forecasting capabilities to topics like adaptive loss functions, which are more closely related to the machine learning sphere. Each state-of-the-art class is then described separately in the following.

Network services alignment The alignment of analytics/forecasting systems to metrics that are out of the MNO domain is a novel explored concept that presents only a few available solutions. More specifically, the solutions that are more closely related to our work are: (i) [32], where the authors propose the concept of having different models working together to optimize a *network service* based on external feedback, and (ii) [16], where the authors also tackle the problem of possibly unrelated QoE metrics with the QoS and network parameters. However, in both cases, they do not assume that such optimization shall happen in a *limited trust* environment considering the sharing of sensible information. (as a matter of fact, the proposed optimization is happening within the same operator MANO entity).

Customized loss functions The usage of customized loss function in the mobile network domain is becoming more and more frequent to tackle the specific requirements imposed by the environment. For instance, the usage of a parametrized loss function like the one proposed in [33] and refined in [34] yields performance gains up to 50% in terms of reduced resource utilization. The authors in such papers presented an approach that takes advantage of domain knowledge as a regularization term but also embeds monetary costs in the loss function to obtain better predictions on anticipatory MANO decisions.

However, one of the main inspiration points for our work is the one presented by Huang *et al.* [35], where multiple AI/ML models are coupled in a parent-child scenario in order to cooperate and outperform other deep-learning approaches in tasks like image recognition. More specifically, the scenario proposed in [35] uses RL algorithms to correctly tune loss function parameters to improve the learning phase on different specific tasks. They also show how this approach could be widely adapted to different scenarios and different loss functions.

Data-bias and -uncertainty Mobile networks, and networks in general are designed to satisfy the user QoS enforcing mechanisms such as efficient resource provisioning. Hence it is very likely that when training Deep Learning (DL)-based solutions, the network datasets contain very biased data for *good* network behaviour.

This fact is not good news for DL systems that are known to suffer performance degradation in such situations, as described in [21]. Other than that, the capability to take advantage of uncertain situations is mandatory for unlabelled datasets, like the approach described in [36], where the authors assume that just a tiny fraction of the dataset is labeled and exploit it to obtain "pseudo"-labels on the remaining part of the dataset. Through multiple iterations, they show improvements in the prediction of such "pseudo"-labels. Other approaches, like the one firstly proposed by Benigo *et al.* [37] try to overcome dataset limitations by gradually increasing the dataset difficulty during the training, e.g. introducing new data-classes in sequential steps. All these arrangements could benefit the MNO DL infrastructure, which has no clue of the real QoE value associated with the network KPIs.

Limited-trust network infrastructure As we defined in Section I, one of the most stringent assumptions for our work is the *limited-trust* requirement, where the SP cannot disclose any raw data to other parties, especially business-related information.

In [4] we introduce such concepts, defining the network components and APIs that can be used to achieve a broader, but still secure, communication between MNOs and SPs to obtain tailored analytics. On the other hand, the assumptions made in [4] strictly limit the evaluation only to the MNO do-

main, without actually presenting and addressing the possible mismatch between analytics and business metrics. This paper digs deeper into such assumptions and better models the gap between MNO deployed solutions and the SP expected outcome, providing a fully-fledged solution, ATELIER, to the problem at hand.

VII. Conclusion

In this paper, we presented ATELIER, a framework for aligning the analytics provided by network operators to the actual QoE metrics needed by SPs. In our work, we presented the general problem and then discussed the solution (based on similarity learning and reinforcement learning techniques) for the alignment of anomaly detection analytics to the business metric related to a video streaming service.

By using open-source tools for Mobile network function implementations, we tested ATELIER over a set of scenarios representative of a multi-service network environment. In this way, ATELIER showcases the adaptability and generalization to different applications (i.e., distinct videos using different QoE levels), providing analytics that align the QoS measured by MNOs with the QoE assessed by SPs. In all the benchmarked scenarios ATELIER outperforms the alternative solutions.

Acknowledgment

The European Union-NextGenerationEU has partly funded this work through the UNICO 5G I+D SORUS project and by Smart Networks and Services Joint Undertaking (SNS JU) European Union's Horizon Europe research and innovation programme under Grant Agreement No. 101139270 (ORIGAMI). This publication is part of the project 6GINSPIRE PID2022-137329OB-C42, funded by MCIN/ AEI/10.13039/501100011033/. This work was also supported by the German Federal Ministry of Education and Research (BMBF) project 6G-ANNA under Grant Agreement No. 16KIS077K.

REFERENCES

[1] A. Garcia-Saavedra and X. Costa-Pérez, "O-RAN: Disrupting the Virtualized RAN Ecosystem," *IEEE Communications Standards Magazine*, vol. 5, no. 4, pp. 96–103, 2021.

[2] "5G-ACIA, 5G Alliance for Connected Industries , a Working Party of ZVEI (German Electrical and Electronic Manufacturers' Association)." https://www.5g-acia.org/.

[3] Linux Foundation, "Camara Project." https://camaraproject.org/. [Accessed 27-Jun-2023].

[4] M. Milani, D. Bega, M. Gramaglia, and C. Mannweiler, "Optimizing predictive analytics in 5g networks through zero-trust operator-customer cooperation," in *2023 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pp. 123–128, 2023.

[5] V. Ziegler, P. Schneider, H. Viswanathan, M. Montag, S. Kanugovi, and A. Rezaki, "Security and trust in the 6g era," *IEEE Access*, vol. 9, pp. 142314–142327, 2021.

[6] Y. Wang, R. Forbes, C. Cavigioli, H. Wang, A. Gamelas, A. Wade, J. Strassner, S. Cai, and S. Liu, "Network management and orchestration using artificial intelligence: Overview of etsi eni," *IEEE Communications Standards Magazine*, vol. 2, no. 4, pp. 58–65, 2018.

[7] T. Li, K. Zhu, N. C. Luong, D. Niyato, Q. Wu, Y. Zhang, and B. Chen, "Applications of multi-agent reinforcement learning in future internet: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 2, pp. 1240–1279, 2022.

[8] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 2031–2063, 2020.

[9] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," *arXiv preprint arXiv:1712.07557*, 2017.

[10] C. Dwork, "Differential privacy," in *International colloquium on automata, languages, and programming*, pp. 1–12, Springer, 2006.

[11] H. Fereidooni, S. Marchal, M. Miettinen, A. Mirhoseini, H. Möllering, T. D. Nguyen, P. Rieger, A.-R. Sadeghi, T. Schneider, H. Yalame, *et al.*, "Safelearn: Secure aggregation for private federated learning," in *2021 IEEE Security and Privacy Workshops (SPW)*, pp. 56–62, IEEE, 2021.

[12] M. Naseri, J. Hayes, and E. De Cristofaro, "Local and central differential privacy for robustness and privacy in federated learning," *arXiv preprint arXiv:2009.03561*, 2020.

[13] A. Blanco-Justicia, D. Sánchez, J. Domingo-Ferrer, and K. Muralidhar, "A critical review on the use (and misuse) of differential privacy in machine learning," *ACM Computing Surveys*, vol. 55, no. 8, pp. 1–16, 2022.

[14] 3GPP, "Study on network slice management capability exposure," Technical Report (TR) 28.824, 3rd Generation Partnership Project (3GPP), July 2023. Version 18.0.1.

[15] 3GPP, "Architecture enhancements for 5G System (5GS) to support network data analytics services," Technical Specification (TS) 23.288, 3rd Generation Partnership Project (3GPP), March 2023. Version 18.1.0.

[16] A. Collet, A. Bazco-Nogueras, A. Banchs, M. Fiore, *et al.*, "Automanager: a meta-learning model for network management from intertwined forecasts," in *IEEE International Conference on Computer Communications*, 2023.

[17] M. A. Garcia-Martin, M. Gramaglia, and P. Serrano, "Network Automation and Data Analytics in 3GPP 5G Systems," *IEEE Network*, pp. 1–1, 2023.

[18] R. C. Streijl, S. Winkler, and D. S. Hands, "Mean opinion score (MOS) revisited: methods and applications, limitations and alternatives," *Multimedia Systems*, vol. 22, pp. 213–227, Dec. 2014.

[19] Z. Li, A. Aaron, I. Katsavounidis, A. Moorthy, and M. Manohara, "Toward a practical perceptual video quality metric," *The Netflix Tech Blog*, vol. 6, no. 2, p. 2, 2016.

[20] Z. Li, C. Bampis, J. Novak, A. Aaron, K. Swanson, A. Moorthy, and J. Cock, "VMAF: The journey continues," 2018.

[21] S. Thudumu, P. Branch, J. Jin, and J. J. Singh, "A comprehensive survey of anomaly detection techniques for high dimensional big data," *Journal of Big Data*, vol. 7, July 2020.

[22] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823, 2015.

[23] C.-Y. Wu, R. Manmatha, A. J. Smola, and P. Krahenbuhl, "Sampling matters in deep embedding learning," in *Proceedings of the IEEE international conference on computer vision*, pp. 2840–2848, 2017.

[24] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 539–546, IEEE, 2005.

[25] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, pp. 1735–1742, IEEE, 2006.

[26] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification.," *Journal of machine learning research*, vol. 10, no. 2, 2009.

[27] F. Gringoli, P. Serrano, I. Ucar, N. Facchi, and A. Azcorra, "Experimental qoe evaluation of multicast video delivery over ieee 802.11aa wlans," *IEEE Transactions on Mobile Computing*, vol. 18, no. 11, pp. 2549–2561, 2019.

[28] S. Bhatnagar, R. S. Sutton, M. Ghavamzadeh, and M. Lee, "Natural actor–critic algorithms," *Automatica*, vol. 45, no. 11, pp. 2471–2482, 2009.

[29] A.-J. Gallego, J. Calvo-Zaragoza, and R. B. Fisher, "Incremental unsupervised domain-adversarial training of neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 11, pp. 4864–4878, 2020.
[30] Netflix Technology Blog, "VMAF: The Journey Continues — netflixtechblog.com." https://netflixtechblog.com/vmaf-the-journey-continues-44b51ee9ed12. [Accessed 20-Dec-2023].
[31] L. McInnes, J. Healy, and J. Melville, "UMAP: Uniform manifold approximation and projection for dimension reduction," 2020.
[32] A. Collet, A. Banchs, and M. Fiore, "Lossleap: Learning to predict for intent-based networking," in *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*, pp. 2138–2147, 2022.
[33] D. Bega, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Perez, "Deepcog: Cognitive network management in sliced 5g networks with deep learning," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, pp. 280–288, 2019.
[34] D. Bega, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Perez, "Aztec: Anticipatory capacity allocation for zero-touch network slicing," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, pp. 794–803, 2020.
[35] C. Huang, S. Zhai, W. Talbott, M. B. Martin, S.-Y. Sun, C. Guestrin, and J. Susskind, "Addressing the loss-metric mismatch with adaptive loss alignment," in *International conference on machine learning*, pp. 2891–2900, PMLR, 2019.
[36] D. Cai, S. Wang, Y. Wu, F. X. Lin, and M. Xu, "Federated few-shot learning for mobile NLP," in *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*, pp. 1–17, 2023.
[37] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48, 2009.
[38] I. Gomez-Miguelez, A. Garcia-Saavedra, P. D. Sutton, P. Serrano, C. Cano, and D. J. Leith, "SrsLTE: An Open-Source Platform for LTE Evolution and Experimentation," in *Proceedings of the Tenth ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization*, WiNTECH '16, (New York, NY, USA), p. 25–32, Association for Computing Machinery, 2016.
[39] N. Apostolakis, M. Gramaglia, and P. Serrano, "Design and validation of an open source cloud native mobile network," *IEEE Communications Magazine*, vol. 60, no. 11, pp. 66–72, 2022.
[40] C. Grecos and M. Y. Yang, "Fast inter mode prediction for P slices in the H264 video coding standard," *IEEE Transactions on Broadcasting*, vol. 51, no. 2, pp. 256–263, 2005.
[41] S. Casner, R. Frederick, V. Jacobson, and H. Schulzrinne, "RFC 3550, RTP: A Transport Protocol for Real-Time Applications," *Network Working Group*, 2003.
[42] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

Mattia Milani Is an industrial Ph.D. student currently enrolled at the University Carlos III de Madrid and emplyeed at Nokia Strategy & Technology. He received his M.Sc (2020) in Computer Science from University of Trento.

Dario Bega is a Network System Automation Researcher at Nokia Bell Labs. He received his M.Sc. (2013) in Telecommunications Engineering from the University of Pisa, Italy, and his Ph.D. (2020) in Telematics Engineering from the University Carlos III of Madrid and IMDEA Networks Institute.

Pablo Serrano (M'09, SM'16) is an Associate Professor at the University Carlos III de Madrid. He has over 100 scientific papers in peer-reviewed international journals and conferences. He currently serves as Editor for IEEE Open Journal of the Communication Society

Marco Gramaglia is a Visiting Professor at the University Carlos III of Madrid, where he received M.Sc (2009) and Ph.D (2012) degrees in Telematics Engineering.

Christian Mannweiler is Department Head of Network Architecture Research at Nokia Strategy & Technology. He received his M.Sc and Ph.D degrees in Electrical Engineering from University of Kaiserslautern in 2008 and 2014, respectively.

Appendix A
Experimental setup

A. Mobile Network Deployment

The mobile network is based on `srsRAN` [38], an open-source software implementation of Rel. 16 compliant UE, gNodeB (gNB), and Core network. While this software is usually configured to be used with a Software Define Radio (SDR) card as the Radio Unit, `srsRAN` also offers the opportunity to stream the fronthaul traffic over a socket [39], a configuration we used for our experiments to allow the virtualization of the system end to end.

The lifecycle management of the system is handled through `Docker` running each of the elements in the network (i.e., UE, gNB, 5GC). Videos are streamed using the `ffmpeg` server (running in the 5GC docker to emulate the cloud environment) and client (running in the UE) software.

Network impairments are emulated by inserting them in the Core to Radio Access Network (RAN) interface through the `tc` software, mostly by delaying and dropping packets there. This will also emulate possible impairments in the air interface obtained e.g., by monitoring the HARQ process in the gNB.

Finally, the flow features are extracted by emulating a Deep Packet Inspection (DPI) solution running in the network, by sniffing traffic through `Wireshark`.

B. Videos and data engineering

We use the well-known VMAF tool to analyze the perceived QoE for the user watching the videos in the UE. Videos are split into shorter samples (of duration w s) that represent partial feedback about the experienced QoE by the user, and VMAF is averaged over these shorter intervals.

To showcase the adaptability of our framework for different services, we stream three different videos following a methodology similar to the one adopted in [27]. Namely, we use three videos `Bunny`, `Bottle`, and `Scarlet` that have different characteristics in terms of bandwidth, as they are encoded with the `h264` which produces different bitrates according to the complexity of the scenes recorded in the video [40].

This affects, among other things, the complexity of the QoE to QoS mapping. We demonstrate this experimentally in Fig. 3. Both Figs. 3a and 3b show how the overall VMAF is affected depending on the applied drop rate and delay, respectively. While for both metrics the high-level trends are similar (higher VMAF for low impairment rate), the behaviour for each of them is i) highly non-linear and ii) video dependent. For instance, while the delay mildly affects `Bunny` and `Bottle`, it negatively affects `Scarlet` VMAF and, most importantly, with a very large variance. Also, the effect of packet drops is negatively affecting the QoE in different ways for the selected videos.

Hence, to train the network described in Section III, we compute the VMAF (used by the SP to calculate the reward function discussed in Eq. (3)) and gather QoS metrics over the same time interval w. For this case study, we resort to 6 features used to compute the KPI vector $\mathcal{K}$:

- Two features related to packet losses: PDR and Byte Drop Rate (BDR).
- Four features related to the delay: the average, standard deviation, skewness, and kurtosis of the Inter-Packet Time (IPT)

Then, we set conservative values for $\mathcal{G}_k$ and $\mathcal{B}_k$ (detailed in Table 4) and create the tree sets $\hat{G}$, $\hat{B}$, and $\hat{U}$ that are used to start the training of ATELIER. The system hyperparameters are detailed in Table 5.

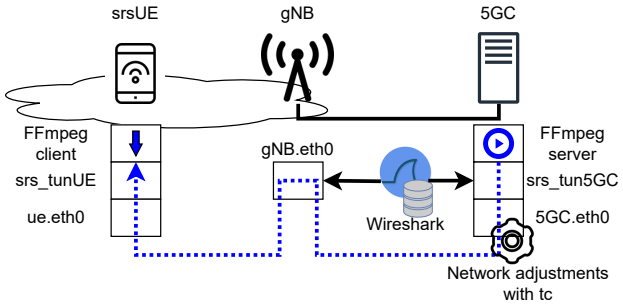


FIGURE 13: real-time video stream docker environment representation

TABLE 2: Streaming scenarios

Scenario	Impairment
No Impairment	None
Packet drop	[0.5 - 3]% packet drop rate with 50% correlation
Packet delay	[100 - 600]± 50ms delay with a normal distribution

C. Experiment Management

The experiment manager we designed generates docker-composer environments on the fly following the configuration provided by the user. we specifically designed the docker images to use *srsRAN*[8] to emulate mobile networks. Using this infrastructure we will distribute a real-time video stream to the final user through *FFmpeg*[9]. The software will then evaluate both network-related KPIs and QoE metrics putting those together into the same dataset.

In our case, we decided to use one UE, one gNB and one 5G Core network (5GC) interconnected through the docker network. Fig. 13 shows all the components that we used during the data-generation process. We used Wireshark[10] to sniff the traffic both at the 5GC and the gNB interfaces and compute the network-related KPIs.

[8]https://www.srsran.com/
[9]https://ffmpeg.org/
[10]https://www.wireshark.org/

TABLE 3: Files used during the real-time video-stream experiments obtained from the CDVL[11] database

Title	ID	Run Time[s]	Peculiarities
Big Buck Bunny	`Bunny`	24	Cartoon with scene changes and text overlays
Bottle	`Bottle`	15	Zoom out, showing a stack of brightly colored bottles on a smoothly gradated background
Scarlet-Oak	`Scarlet`	24	Camera recordings, with computer-generated text

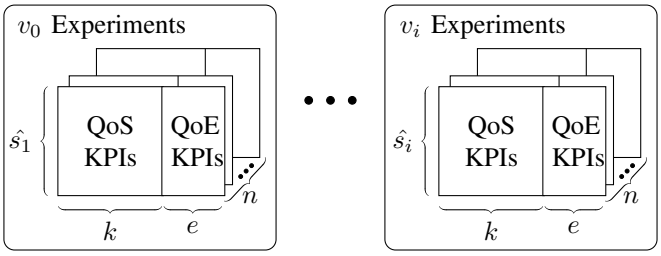


FIGURE 14: Dataset structure visual representation for each video v_i

We use the real-time video distribution protocol Real-time Transport Protocol (RTP) [41] through `FFmpeg` both on the server side and on the client. `FFmpeg` is used to generate the data format needed to compute the VMAF score [19], [20]. The videos transmitted during the experiments are taken from the Consumer Digital Video Library (CDVL) database and listed in Table 3.

To introduce network KPIs perturbations used to obtain variability in the QoS we used `tc`[12] to impair the 5GC. The different scenarios and `tc` configurations are presented in Table 2.

Given a set of different videos $\mathcal{V} = \{v_0, \ldots, v_i\}$ used during the experiments, Fig. 14 provides a visual representation of the dataset at the end of the generation process. For each video v_i the generator produces n experiments of length $\hat{s}_i$, which is a video-dependent property that describes the length of a video in seconds. In fact, each second of transmission is aggregated, both for the QoE and QoS KPIs. All the experiments, independently from the video, have k KPIs related to the QoS and e with the QoE. In total, we have generated 3600 experiments. The dataset D is then passed to the Neural Network (NN) Framework for the pre-processing and evaluation.

[11] https://www.cdvl.org/
[12] TC manual

TABLE 4: Features thresholds used in $\mathcal{G}_k$ and $\mathcal{B}_k$

KPI	$\mathcal{G}_k$	$\mathcal{B}_k$
Packet Drop Ratio	0.0	0.1
Byte Drop Ratio	0.03	0.08
Avg. interarrival time [ms]	0.02	0.06
Std. interarrival time [ms]	0.02	0.06
Skw. interarrival time	7	55
Kur. interarrival time	8	55

Appendix B
Deep Learning Details

A. Hyper-parameters

Table 4 shows the default parameters used to select $\mathcal{G}_k$ & $\mathcal{B}_k$.

Table 5 shows the default hyperparameters used in the ATELIER Deep Learing framework.

TABLE 5: Hyper parameters

Parameter	Value	Usage
w	9 [s]	Number of seconds that composes a single sample
b	100	Batch dimension
k	10	Number of epoch used during each SNN trainng cycle
η	20.0	Default margin value
a_Δ	10.0	Delta used when applying a_+ or a_-
T_{SL}	10	Number of training iterations when applying the SL technique
K	5	Stochastic Gradient Descent (SGD) iterations between one RL episode and the following one
β	0.001	Beta value used in the entropy of A2C
γ'	0.5	Gamma value used during the cumulative sum of $\mathcal{M}$
γ	0.8	Gamma value used for the reward R cumulative sum
T_{RL}	10	Number of episodes considered for the convergence of RL
T_{RL}^{max}	200	Number of episodes before stopping RL
M_Δ	0.01	Delta required to consider two $\mathcal{M}_i$ and $\mathcal{M}_j$ different
RL_c	0.05	RL convergence threshold
RL_{adam}	0.0008	Adam strength for RL
Env_{acc}	50 [%]	Environment accuracy threshold
Env_{exit}	20	Number of SGD with an accuracy value below Env_{acc} before interruption
f_{adam}	0.0001	Adam strength for the f model

B. Deep Learning architecture

The layer infrastructure of the SNN is the same for every experiment, given that we compare KPIs time series the first

layer is a Long Short Time Memory (LSTM) that is then flattened and passed to a series of dense layers. We have chosen to use Adam as optimizer [42] with a learning rate of 0.0001. The training dataset D_t is passed as an input to the network normalized and separated in batches of dimension b. The RL network is structured as a common model main-corpus with an input and a dense layer. Then the output is split into two output layers, the first one is the action output of dimension $|\mathcal{A}|$ with a softmax activation, and the second one is a single neuron output that acts as the critic value. Also, the RL network uses Adam as an optimizer with a learning rate of 0.0008.