Metric meta-learning in deep learning models for intent-based networking

by

Alan Collet

A dissertation submitted in partial fulfillment of the requirements
for the
degree of Doctor of Philosophy in

Telematic Engineering

Universidad Carlos III de Madrid

Advisors:
Marco Fiore and Albert Banchs

Tutor:
Marco Fiore

September 2024

Metric meta-learning in deep learning models for intent-based networking

Prepared by:
Alan Collet, IMDEA Networks Institute, Universidad Carlos III de Madrid
contact: alan.collet@imdea.org

Under the advice of:
Marco Fiore, IMDEA Networks Institute
and
Albert Banchs, IMDEA Networks Institute, Universidad Carlos III de Madrid

This work has been supported by:

Acknowledgements

Completing this thesis was both challenging and, at times, overwhelming, but I believe it has profoundly changed me for the better. Overcoming these challenges has made this journey incredibly valuable, and I would probably never had the chance to accomplish that without people who are now important to me.

I would first like to thank my supervisors, Dr. Marco Fiore and Dr. Albert Banchs, for their continuous support during this research adventure. I am especially grateful for their guidance in various research directions and for allowing me to pursue this PhD within the Network Data Science Research Group. Your mentorship has been invaluable, and I deeply appreciate all the support you have provided.

A very special thanks to my closest workmates, who have also become dear friends, from the first generation of the Network Data Science research group that made the task so much easier during those 4 years: Leonardo Lo Schiavo, Sachit Mishra, Sergi Alcala-Marin, Orlando Martinez-Durive, Aristide Akem, André Zanella, Dr Antonio Bazco-Nogueras and Dr. Michele Gucciardo.

More generally, I would like to thanks all the IMDEA Networks staff. Being part of this community has been a truly wonderful experience.

I would like to extend my gratitude to my family, who supported me during this long journey. Finally, I would like to sincerely thank my girlfriend, Pauline, for her unlimited support over the past four years. Your constant encouragement and belief in me have been invaluable, and I am deeply grateful for your love. You have been my greatest source of strength throughout this journey. I love you more than words can express. I love you.

Published and Submitted Content

This thesis is based on the following published papers:

[1] **Alan Collet**, Albert Banchs, Marco Fiore. LossLeaP: Learning to Predict for Intent-Based Networking. Published in *the 41th IEEE International Conference on Computer Communications (IEEE INFOCOM 2022)*, 2-5 May 2022, London, UK. https://doi.org/10.1109/INFOCOM48880.2022.9796918

- This work is fully included and its content is reported in Chapters 3, 4 and 6.
- The author's role in this work is focused on the design, implementation and experimentation of the proposed methodology.
- The material from this source included in this thesis is not singled out with typographic means and references.

[2] **Alan Collet**, Antonio Bazco-Nogueras, Albert Banchs, Marco Fiore. AutoManager: a Meta-Learning Model for Network Management from Intertwined Forecasts. Published in *the 42th IEEE International Conference on Computer Communications (IEEE INFOCOM 2023)*, 17-20 May 2022, New York, USA. https://doi.org/10.1109/INFOCOM53939.2023.10229064

- This work is fully included and its content is reported in Chapters 3, 5 and 6.
- The author's role in this work is focused on the design, implementation and experimentation of the proposed methodology.
- The material from this source included in this thesis is not singled out with typographic means and references.

[3] Serly Moghadas, Claudio Fiandrino, **Alan Collet**, Giulia Attanasio, Marco Fiore, Joerg Widmer. Spotting Deep Neural Network Vulnerabilities in Mobile Traffic Forecasting with an Explainable AI Lens. Published in *the 42th IEEE International Conference on Computer Communications (IEEE INFOCOM 2023)*, 17-20 May 2022, New York, USA.https://doi.org/10.1109/INFOCOM53939.2023.10228989

- This work is partially included and its content is reported in Chapter 7

- The author's role in this work is focused on the design, implementation, and experimentation of the explainability algorithms (LRP).
- The material from this source included in this thesis is not singled out with typographic means and references.

[4] **Alan Collet**, Antonio Bazco-Nogueras, Albert Banchs, Marco Fiore. Explainable and Transferable Loss Meta-Learning for Zero-Touch Anticipatory Network Management. Published in IEEE Transactions on Network and Service Management journal https://doi.org/10.1109/TNSM.2024.3377442

- This work is fully included and its content is reported in Chapters 3, 4, 5 and 6.
- The author's role in this work is focused on the design, implementation and experimentation of the proposed methodology.
- The material from this source included in this thesis is not singled out with typographic means and references.

[] **Alan Collet**, Antonio Bazco-Nogueras, Albert Banchs, Marco Fiore. Explainable and Learning to Learn How to Manage Network Resources with Loss Function Meta-Learning. Published in IEEE Communications magazine

- This work is fully included and its content is reported in Chapters 3, 4, 5 and 6.
- The author's role in this work is focused on the design, implementation and experimentation of the proposed methodology.

Abstract

The evolution of communication networks towards self-configuring systems requires the development of anticipatory approaches for network management to realize the envisioned concept of zero-touch network orchestration. For this task, Intent-Based Networking (IBN) represents a transformative approach to network management, aiming to bridge the gap between high-level, human-understandable intents and their translation into effective, automated network operations.

Intent-Based Networking aims to greatly improve network management by leveraging automation, artificial intelligence, and machine learning to align network configurations with user-specified outcomes. Rather than detailing how tasks should be accomplished, users simply define their desired outcomes, such as secure connections between locations. The IBN system interprets these intents and automatically configures the network, continuously monitoring and adjusting settings to ensure goals are met. This approach simplifies network management by abstracting complex configurations, enhancing agility, and enabling faster deployment of services. Ultimately, IBN offers a more intuitive, efficient, and business-aligned method of managing networks.

Central to this is the requirement for network orchestrators to leverage suitable automated decision models, which are essential to accurately actualize these intent-based objectives. This becomes especially critical in the realm of anticipatory networking, where it is imperative to employ a prediction model that is not only accurate, but also aligned with the complex and nuanced objectives set forth by the original intent.

Traditional prediction models in network demand and management optimization, however, have been restricted by their generic, inflexible, and manually set objectives, meaning that they require perfect knowledge of the relationship between the management decisions and the consequent system performance. However, in anticipatory networking, there exist many tasks where characterizing such a relationship in advance is not possible. In such tasks, it is possible to measure the resulting performance of a management decision a posteriori, but we cannot know it a priori. An example of such tasks is the maximization of the monetary profit when allocating resources to end users: when taking a certain resource allocation decision, it is possible to measure the gain afterwards, but it would be extremely difficult to determine a priori the resulting monetary profit.

These limitations contrast with the sophisticated needs of anticipatory IBN, underscoring the growing demand for an advanced predictive model. This model should be capable of autonomously understanding and adapting to the interplay between its predictions and the targeted network management objectives.

The initial section of the thesis introduces a model designed to bridge the gap between the loss function and the performance metric in regression challenges. This architecture represents a significant advancement in the field of automated machine learning. This model's effectiveness is underscored through real-world examples where its architectural approach proves not just advantageous but indispensable. These case studies highlight how the model effectively aligns its predictions with the detailed objectives of IBN. This alignment showcases the model's adaptability and its capability to meet the demands of anticipatory networking tasks.

The first interest of this model delves into this "loss-metric mismatch" problem in machine learning. This issue arises when the loss function, used for training deep neural networks (DNNs), does not align with the actual performance metric, or in the case of IBN, the translated anticipatory Management and Network Orchestration objective (MANO).

However, in the rapidly evolving landscape of network management and broader technological domains, a second aspect gaining increasing prominence is the imperative to align disparate entities towards a common objective. Traditional methodologies fall short of addressing this need. These models often operate in parallel, prioritizing their individual optimization goals. This leads to competitive behaviors, where the pursuit of one model's optimization comes at the expense of others, potentially deteriorating the overall system performance. Such dynamics are particularly problematic in complex scenarios where problems and solutions are deeply interlinked and require a collaborative and holistic approach rather than a competitive one.

Addressing this gap, the second part of this thesis builds on the results in the first part and introduces an improved model that seeks to fulfill the requirement of coordinating multiple concurrent and independent predictions so that they achieve a common goal with minimum assumptions. Crucially, the proposed model maintains a solution to the issue of loss metric mismatch while extending it to multiple predictors, hence substantially widening the range of potential network management applications. Our exhaustive testing, with both controlled environments and real-world applications, demonstrates the superior performance of the proposed approach. This compelling evidence underlines the model's efficacy in meeting the intricate demands of anticipatory IBN.

The final part of this thesis is dedicated to exploring the explainability of DNN models. In today's world, where ethical considerations and enhanced understanding are paramount, there is a growing emphasis on the explainability of models, particularly in the context of DNNs. Explainability is critical to comprehending a model's behavior, and

discerning normal functioning from potential malfunctions, whether due to inherent flaws or as a result of targeted attacks using malicious data. The ability to explain a model's decisions and processes is not just a technical requirement but also an ethical imperative, ensuring transparency and trust in automated systems. Special attention is given to its significance in the context of spatio-temporal models used for detecting potential attacks. The thesis examines the explainability aspects of the models discussed in the first two parts and demonstrates their inherent transparent nature in contrast to other technique as Reinforcement Learning (RL). This exploration is crucial for providing insights into the model's decision-making processes, enhancing trustworthiness, and facilitating the identification and mitigation of vulnerabilities. By focusing on explainability, the thesis aims to contribute to the development of more transparent, reliable, and ethically sound machine learning models, particularly in the realms of network management and security.

Overall, this thesis presents a comprehensive exploration of advanced network management strategies. It identifies and addresses critical challenges of aligning the predictive capabilities of machine learning models, particularly DNNs, with the nuanced objectives of IBN. The proposed models, which aim to resolve the "loss-metric mismatch" and ensure unified goal achievement, mark significant advancements in automated network management. These models are not only theoretically relevant but are also validated through extensive real-world applications, demonstrating their practical efficacy and adaptability in dynamic networking environments. Moreover, the thesis emphasizes the importance of explainability in machine learning, highlighting its crucial role in ensuring the reliability, transparency, and ethical integrity of automated systems. By delving into the explainability of DNN models, the research contributes significantly to the development of more transparent and trustworthy machine-learning solutions.

Contents

List of Tables

List of Figures

List of Acronyms

AF Application Functions

AI Artificial Intelligence

AVST Average Uninterrupted Service Time

CLR Cyclic Learning Rate

DNN Deep Neural Network

DUs Distributed Units

IBN Intent-Based Networking

ICT Information and Communications Technology

INR Implicit Neural Representation

IR Individual Regressor

LRP LayeR-wise backPropagation

KPIs Key Performance Indicators

LSTM Long-Short Term Memory

MAE Mean Absolute Error

MANO Management and Network Orchestration

MLP Multi-Layer Perceptron

MLOps Machine Learning Operations

MOS Mean Opinion Score

MSE Mean Squared Error

NSSI Network Slice Subnet Instance

OPEX monetary OPerating EXpenses

QoE Quality of Experience

RL Reinforcement Learning

RNN Recurrent Neural Network

RUs Remote Units

SLA Service Level Agreement

VIM Virtual Infrastructure Manager

VM Virtual Machine

VNFs Virtual Network Functions

XAI EXplainable AI

ZSM Zero-touch networking and Service Management

Part I

Preliminaries

1 Introduction

Mobile networks are becoming more complex from a variety of facets, including the size and increasingly distributed nature of the infrastructure, the heterogeneity of the technologies and service requirements, and the emergence of new paradigms. As a result, network management is a more challenging task than ever, requiring rapid or anticipatory actions in response to the dynamics of a tangled environment. Thus, the automation of network management plays a core role in the vision for 6G [6]. Moreover, the increasing softwarization and programmability of mobile networks, and the redesign of network functions for cloud-native operation lay the foundations to automation paradigms like Zero-touch networking and Service Management (ZSM) [7] and Intent-Based Networking (IBN) [8]. Data-driven models will be important enablers in this ecosystem, and standard-defining organizations are integrating Machine Learning Operations (MLOps) into Management and Network Orchestration (MANO) frameworks [9–11].

The ultimate vision for network management automation is IBN. In IBN, human controllers only dictate intent policies that define (e.g., in natural language) what the network should do in terms of high-level objectives, without specifying how to achieve them [12, 13]. As illustrated in Figure 1.1, intents go through a translation stage, where they are rendered (e.g., via natural language processing) into a form that is consumable by the network management entities. Those entities are then responsible for the activation stage, where suitable management decisions are taken to meet the target requirements. A final assurance stage includes monitoring of the system, verification that specifications are met, and triggering of potential adjustments. Ultimately, as intents are transparent to the underlying hardware and portable across technologies, IBN pushes humans out of the loop as much as possible while preserving as much as possible the models robust, transparent, and understandable by humans. It limits humans' role to that of providers of conceptual operation guidelines which the system is then expected to interpret and achieve in a fully automated manner.

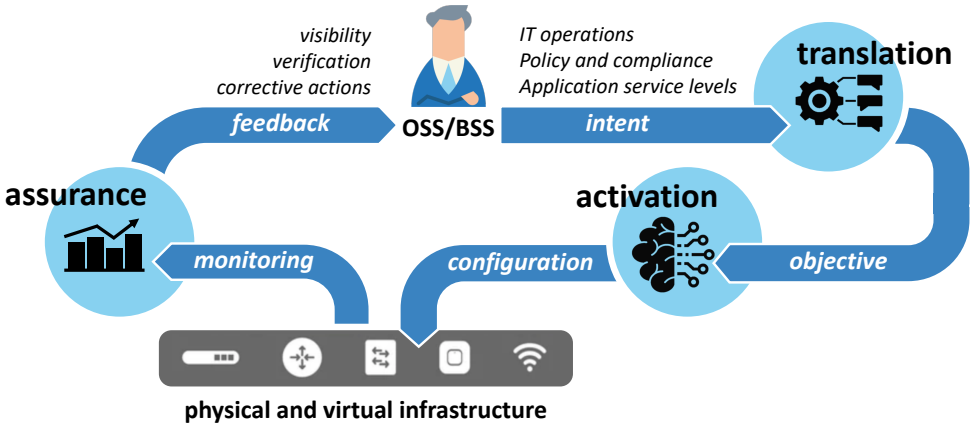


Figure 1.1. High-level workflow of Intent-Based Networking.

IBN is today in its infancy, with standardization efforts that are still in progress [7, 14–16]. While IBN platforms are being advertised [17], commercial solutions touting support for the paradigm are mainly network visibility tools that allow operators to retrieve monitoring information by means of intent-based declarations; they are quite far from the promises of autonomous remediation or intent-based orchestration and do not provide enough transparency.

1.1. Context

While the attention is usually drawn to the IBN translation stage, where networks are controlled through human-understandable commands, it is arguable from a networking viewpoint that the essence and the primary technical challenges of IBN actually reside in the activation stage. Indeed, it is during activation that network orchestrators and controllers have to automatically design and implement the appropriate decision model that aligns with the intended management objective. The process of automated model design is, by itself, a complex task. In IBN, the challenge is even harder, as objectives are machine-translated from an intent policy, hence may easily take forms that are inherently difficult to handle. These objectives might combine a multitude of Key Performance Indicators (KPIs) or exhibit characteristics that are inherently complex to address, such as lack of convexity or differentiability. These aspects set IBN activation apart from traditional human-in-the-loop policing, where the management problem is manually formalized in a way that is tractable and an appropriate decision-making model is then carefully designed to solve it.

In scenarios where management decisions need to be made preemptively, the implementation of the IBN activation stage involves the automatic customization and resolution of a prediction problem that is specifically tailored to the target management

objective. This requires not only a deep understanding of the network's intricacies but also a nuanced approach to model design and implementation, ensuring that the network's behavior aligns precisely with the overarching management goals.

As a toy example, let us consider the following and deliberately simple intent policy: "ensure high reliability to all streaming services from the Fusion Arena in Philadelphia in the next hour". The intent would be machine-translated into a set of objectives warranting that enough transport and compute resources are proactively allocated to all streaming network slices in the end-to-end path between the Remote Units (RUs) covering the Fusion Arena and the mobile network gateway and so, without disturbing other services. For instance, one such objective would target the virtualized Radio Access Network (vRAN) Far-Edge Site serving the RUs above, and require that CPU capacity is reserved in Distributed Units (DUs) of the far-edge site to run radio functions (e.g., demodulation, decoding, forward error correction) for the entirety of the expected slice demand and so, without disturbing other services. Here, IBN activation consists in producing a forecast of the CPU resources needed to serve the streaming services demand in the next hour and isolate them in the ES for exclusive use by that slice.

Assuming that a blueprint is readily available for a prediction model that solves the exact task above is not realistic: this is just one of the infinite anticipatory network management problems that may occur in practice, and building (and maintaining) a comprehensive catalog of fine-tuned models that tackle each and all of them is not possible in practical contexts. Sticking with this toy example, the original intent could change to, e.g., maximizing the revenue of the operator from streaming services slices, or allocating the minimum capacity that keeps streaming services users satisfied; these variants completely reorient the decision process, altering the notion of the correct amount CPU resources, and requiring a different logic to predict it.

Instead, to truly manage anticipatory IBN activation challenges, there is a compelling need for a more advanced approach: the ability to automatically create a forecasting model on-demand that is precisely aligned with any specific management objective it is presented with. This capacity ensures preparedness to cope with the requisites of any intent, even if not known a-priori and/or completely entangled with one another.

At the same time, in recent years, there has been a growing emphasis on trust and resilience in Information and Communication Technology. This shift has led to a dynamic regulatory environment, with changes occurring at both national and international levels. Several initiatives in this evolving landscape are focused on incorporating explainable Artificial Intelligence (AI) into systems [18]. The distinction between explainability and model interpretability is critical here. While interpretability is concerned with the transparency of the internal workings of a generic AI model, explainability delves deeper. It aims to provide stakeholders with tailored insights that enable them to understand how AI makes its decisions. This aspect of AI is critical for various stakeholders. For

end-users, explainability is key to building trust in AI decisions, particularly in scenarios where these decisions have significant impacts. As an example, for governmental agencies, explainable AI is a tool to ensure the protection of citizens' rights and compliance with laws. This ensures that AI systems are not just efficient and effective but also operate within ethical and legal boundaries.

1.2. Challenges towards deploying IBN

As previously outlined, achieving alignment with the targeted IBN objectives requires addressing several specific needs. A part of those requirements is the focus on the following challenges, which are key for the successful implementation of IBN in real-world scenarios.

1.2.1. Loss-metric mismatch

A fundamental issue addressed in this thesis is the loss-metric mismatch in machine learning. This problem arises when the loss function employed in training Deep Neural Network (DNN) does not correspond to the actual performance metric or, in the context of IBN, the machine-translated management objective. Standard traffic predictors, for instance, fail to resolve this mismatch and consequently produce outputs that do not directly align with the intended objective. The visionary ability set out above is obviously extremely difficult to realize in practice.

Some surrogate losses are often used as a proxy for discontinuous or otherwise challenging prediction-metric relationships that cannot be directly used to drive the learning process. These surrogates are typically task-specific and handcrafted, which involves significant manual effort for each new task. Furthermore, they require prior understanding of the relationship between the prediction and the metric.

To remove the need for time-consuming human expert intervention, recent proposals have explored the possibility of meta-learning surrogates. Solutions in this space have been proposed to express the performance metric as a function of a set of simple surrogates [19], or to compose a loss function from primitive mathematical operators [20]. A more elaborated strategy in this direction is that of learning the loss as a convex-by-construction function within a given parametric family (e.g., MSE or other quadratic operations of the loss input) and identifying the exact parameters providing the best performance [21].

All the proposals above still require a priori knowledge of the original relationship to identify a relevant set or family of surrogates, hence do not answer to the need of learning the loss at model runtime upon deployment in the target system that IBN aims to achieve.

1.2.2. Common objective for intertwined predictions

The second significant challenge explored in this thesis revolves around the management of complex, intertwined predictions aimed at optimizing a collective, global objective. This is a challenging and open problem in loss meta-learning that is particularly relevant in the context of MANO. Essentially any multiple-input MANO problem falls in this category. Examples of such multiple-input problems include predictive scheduling, where limited capacity must be efficiently distributed among various users, and proactive admission control, which involves managing multiple traffic flows requesting access to a network service. Additionally, this challenge adresses the anticipatory allocation of shared resources, a task that requires foresight and strategic planning to ensure optimal use of available resources.

This problem follows directly the loss-metric mismatch challenge as it is not feasible to manually craft loss functions in multidimensional contexts involving multiple predictors. This situation is essentially a more general scenario of the loss-metric mismatch problem, where the complexity is increased due to the multidimensional nature of the tasks. Traditional methodologies often isolate forecasting from decision-making processes. potentially overlooking the complex interplay between predictions and final objectives. As a consequence, such an approach can lead to outcomes that are suboptimal. The complexities of these relationships are not just about individual predictions but also about how these predictions interact and collectively contribute to the final outcome.

This challenge demands a more integrated approach, that recognizes and accommodates the different interactions between various predictive elements and decision-making processes.

1.2.3. Explainability

Finally, the last challenge tackled in this thesis is the explainability of most advance data-driven approaches. Particularly, mobile traffic prediction is particularly challenging due to the dynamic and unpredictable nature of traffic loads, which vary significantly across different locations and times. Recent advancements in deep learning have proven successful in addressing these complexities [22].

However, a prominent issue with these sophisticated architectures lies in their opaqueness, as the logic they apply is not intuitively understandable to humans. This lack of explainability in DNN models poses significant challenges, particularly in the context of deploying them in production networks. The core of the problem is the absence of a clear understanding of the decision-making processes within these models which complicates troubleshooting and makes them more vulnerable to adversarial attacks.

These are well known to occur when adversaries ingeniously design perturbations to the original input. These perturbations, while often imperceptible to the human eye, are

potent enough to drastically undermine the accuracy of an AI model during inference [23]. In the specific context of spatio-temporal mobile traffic forecasting, such perturbations could entail artificially altering the load or jamming a certain number of base stations over time.

Addressing this challenge of explainability is key to apply most of these advanced techniques for real-world applications. This is essential not only for ensuring robustness against potential adversarial attacks but also for fostering trust and reliability in their deployment within production environments. Hence, in this thesis, we do not only design advanced models for traffic prediction through deep learning but also underscore the paramount importance of bridging the gap between sophisticated technology and its explainability, a key to its successful implementation in practical, real-world scenarios.

1.3. Contributions

In particular, this thesis introduces a deep-learning model to address these issues: aligning the loss function with real-world performance, and handling intertwined predictions for complex objectives. These models are validated through experiments and real-world datasets, demonstrating their effectiveness and industry relevance. Enhanced explainability is a key feature, crucial for understanding, trust, and security, making these models suitable for real-world applications where AI decision transparency is vital.

1.3.1. Identifying limitations in state-of-the-art method for regression for IBN

As mentioned before, the need for innovative approaches based on AI for regression problems is becoming indispensable today and particularly in networking, with problems such as traffic forecasting. Modern methods, incorporating an array of techniques like decision trees, reinforcement learning, and/or deep neural networks, represent the most promising results. However, these state-of-the-art models, despite their sophistication, often fall short of delivering concrete solutions for real applications. The limitations stem primarily from the inherent design of these approaches. For instance, decision trees may struggle with capturing complex nonlinear relationships in data, while reinforcement learning can be harmed by its inherent lack of explainability and transparency.

This scenario presents a significant challenge, as these shortcomings limit the models' ability to fully meet the evolving and demanding objectives of network management. As networks grow in complexity and the volume of data increases, the need for models that are not only accurate but also efficient, interpretable, and scalable becomes more acute.

1.3.2. Designing deep-learning predictors for IBN

In order to address the challenges outlined earlier, such as the loss-metric mismatch, the intricate task of managing intertwined predictions in scenarios with common objectives, and the prevalent issue of explainability in contemporary methods, this thesis introduces two distinct models.

These models are meticulously designed to mitigate these issues, thereby narrowing the gap between theoretical AI and its practical applications in real-world scenarios. The first model introduces an innovative approach to harmonize the loss function with real-world performance indicators, ensuring that the training process is directly relevant to practical applications.

The second model focuses on handling intertwined predictions in cases where a single objective is dependent on multiple predictions. It facilitates more accurate and reliable predictions in complex scenarios.

Together, these models represent a significant stride towards bridging the gap between theoretical AI concepts and their deployment in everyday practical situations, addressing key challenges in the field.

1.3.3. Validating the predictors in realistic application use cases

In this study, the efficiency of those proposed models is rigorously validated through a series of controlled experiments, and comparing them with the most advanced state-of-the-art approaches. This comparison is crucial, as it not only benchmarks our models against existing methodologies but also highlights their relative strengths and improvements.

Furthermore, a strong emphasis is placed on the practical applicability of our models by utilizing real-world scenarios and datasets. This approach is vital for two reasons: firstly, it demonstrates the models' effectiveness in realistic and complex environments, which are often more challenging and unpredictable compared to controlled experimental settings. Secondly, it underscores the relevance of our models in addressing actual industry needs and problems.

For these real-world evaluations, we employ extensive real-world datasets. This is key in illustrating the superiority of our models. Notably, our models not only perform better in typical scenarios but also enable feasible solutions in situations where standard approaches prove inadequate.

This dual approach of theoretical validation and practical application testing ensures a comprehensive evaluation of the models. It highlights their robustness, scalability, and adaptability, ultimately confirming their potential to achieve IBN prerogatives.

1.3.4. Enhancing explainability of deep-learning predictors

We emphasize the growing necessity of explainability in AI models, especially in the context of today's complex and rapidly evolving world. This need for explainability is twofold: firstly, it enhances human understanding, allowing users and stakeholders to comprehend the decision-making process of AI models. This understanding is crucial for fostering trust and facilitating more effective human-AI collaboration. Secondly, explainability plays a pivotal role in security. By understanding how an AI model arrives at its conclusions, we can better identify potential vulnerabilities or biases, thereby fortifying the model against manipulations and adversarial attacks.

To concretely illustrate the importance of explainability, we present a use case utilizing real-world data in a comprehensive spatio-temporal analysis that demonstrates the practical necessity of transparency and interpretability in a complex, real-world environment. Tools are developed and presented in order to address the lack of explainability in this scenario.

Finally, the enhanced transparency and explainability of the models developed in the thesis will be demonstrated, especially when compared to more conventional approaches. A key aspect of our models is their inner design, which allows for the representation of the behavior of predictors in various scenarios through distinct shapes. This innovative representation is crucial, as it not only makes the models' operations more intuitive and understandable but can also serve a vital function in identifying anomalies.

Particularly in the context of networking, the ability to visualize and understand the behavior of predictors is crucial. In normal operating conditions, the models exhibit shapes representing the behavior of the system for standard predictors' activity. However, in the event of an attack or any unusual activity, these patterns may change noticeably. Moreover, the impact on a metric system will be way more impactful and representative than the impact on a standard loss followed by artificial makeup to align with the system prerequisites. This feature is especially significant as it provides a clear and immediate indication of potential security breaches or malicious activities.

This makes the models developed in the thesis particularly suited for real-world applications where understanding and trust in AI decisions are critical.

1.4. Outline of the thesis

This thesis is split into three main sections. Part I has the role of introducing and describing the big limitation of actual models regarding the IBN objectives that are today's targeted in networking management tasks. It is composed of the Introduction and Chapter 2. Part II is composed of Chapter 3, Chapter 4 and Chapter 5 and its objective is to present a model that can achieve all IBN objectives described above, this

will be highlighted through its detailed definition and multiple use cases in real world scenarios. Finally, Part III of this thesis describes the importance of explainability and present in what manner the model depicted in Part II is able to fulfill the IBN objective of increased explainability. This represents Chapter 6 and Chapter 7. We describe below the specific contributions and findings of every chapter.

Chapter 2. This chapter presents state-of-the-art models that are used for networking management scenarios, their inherent constraints, and ongoing efforts to address these constraints partially to align with the prerequisites outlined by IBN.

Chapter 3. This chapter introduces an innovative model, that adeptly satisfies the three prerequisites outlined in the preceding Introduction, namely addressing the loss-metric discrepancy, effectively coordinating multiple predictors towards a unified objective, and augmenting the level of explainability. This chapter consists of describing the concept behind it as well as a comprehensive overview of its different components.

Chapter 4. This chapter provides a comprehensive exploration of how the model outlined in Chapter 3 effectively tackles the challenge of loss metric mismatch. Through a series of diverse examples and real-world applications, it elucidates the model's capability in handling this issue. Moreover, the chapter delves into an in-depth analysis of different features of the model, offering insights into their characteristics and functionalities.

Chapter 5. This Chapter has the role of exploring the limitations of the simple approach given in Chapter 4 for the management of common objectives problems. Additionally, it elucidates strategies for overcoming these constraints by illustrating various real-world use cases. The challenge lies in integrating this additional characteristic with the solution proposed in Chapter 3, thus effectively addressing all primary objectives of IBN outlined in the Introduction simultaneously.

Chapter 6. This chapter illustrates how the model depicted and used in the previous Chapters 3, 4, and 5, helps to tackle another critical problem in today's AI-driven approaches: their inherent lack of explainability and interpretability. This is a property inherent to this model thanks to its particular architecture.

Chapter 7. This chapter presents why the need for explainability is gaining popularity, particularly underscored through a comprehensive analysis of a spatio-temporal use case. Its importance is highlighted to understand how both the models and the system are working and also, by helping to fortify their resilience against potential malicious exploits.

Chapter 8. This Chapter summarizes the different results across the thesis and discusses possible future research directions and perspectives.

2 State-of-the-art and limitations of current methods

This thesis lies at the interface of computer networking and machine learning, and contributes to advancing the state of the art in both domains. In this chapter we discuss how this is the case, by presenting current frameworks for automated anticipatory management in mobile networks, and techniques for meta-learning in DNN architectures. We argue that the inherent problem underlying current state-of-the-art approaches to automated anticipatory networking is that they build on an inflexible NI design that only targets a single objective and does not apply to other (even similar) problems.

2.1. Anticipatory network management for 6G systems

Three main strategies for anticipatory networking can be identified in the literature, one being more traditional and vastly more popular, and the other two are representing a relatively new proposal. They discussed them next.

2.1.1. Mobile network prediction.

Predictors for mobile network traffic have traditionally relied on statistical models, mainly based on autoregression [24–28]. Approaches based on tools from Markovian [29] or information [30] theory have also been proposed. More recently, as in many other research and engineering domains, deep learning has gained momentum in the design of data-driven network automation solutions [31]. Forecasting is no exception, and many recent works have employed a variety of Deep Neural Network DNN architectures to anticipate future mobile network loads [32–36], showing improved performance over previous methods.

All these predictors aim at producing a forecast that deviates as little as possible from the future traffic demand, by minimizing legacy error metrics such as Mean Absolute Error (MAE) or Mean Squared Error (MSE). In DNN models, as exemplified in Figure 2.1, this is achieved by using MAE or MSE as the loss function, i.e., the expression that the neural network learns to minimize during training. The output provided by these predictors

is completely agnostic to the (manually defined or intent-based) network management objective. Therefore, the prediction does not offer a solution to IBN activation, rather is a mere input to the actual decision-making process.

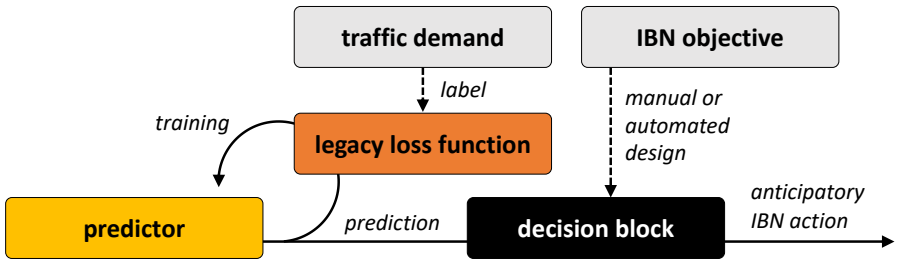


Figure 2.1. Anticipatory IBN activation with traditional mobile network traffic prediction. The predictor is trained to output a pure traffic forecast that minimises the distance from the future traffic demand, measured via a legacy loss function such as MAE or MSE. A separate and subsequent decision block must then encode the actual network management objective, and post-process the prediction to produce the action.

The inherent problem of the approach is that predictions are inevitably imperfect, and yield errors whose impact on the downstream decision-making process is not trivial. As a simple yet representative example, a typical (unbiased) traffic forecasting model incurs in roughly equivalent probability of committing positive and negative errors in the estimation of future demands. Yet, while positive errors result in unnecessary but not too harmful over-dimensioning by the decision-maker, negative errors can cause critical underprovisioning and service disruption. To avoid the latter, the decision-making module must somehow compensate for the prediction inaccuracy, and yet it has no information about if, when, and in what way (e.g., the error is positive or negative) the forecast is inaccurate. Ultimately, this creates a cumbersome operation where the decision-making solution must be designed to fix, usually in very simplistic sub-optimal ways (e.g., by introducing a static large over-dimensioning to mitigate underestimation in the example above).

2.1.2. Loss customization

While the classical traffic predictors above factually decouple the problems of forecasting and activation, recent works on deep learning for network management have proved that jointly solving the two problems is a much more effective approach. So-called capacity forecasting does not predict future traffic demands but directly anticipates the amount of resources needed to serve them [5]. As illustrated in Figure 2.2, a capacity forecast is achieved by training a IBN model with a loss function that encodes a specific resource management objective. In cases where meeting the IBN objective only needs allocating resources, the forecasting model addresses the activation stage as a whole, as

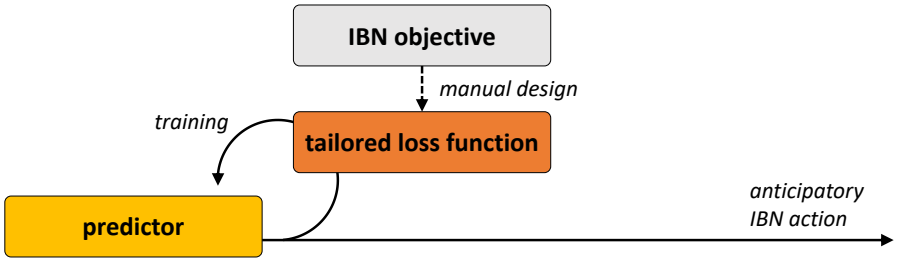


Figure 2.2. Anticipatory IBN activation with capacity forecasting. The network management objective is manually encoded into a tailored loss function by human (networking and machine learning) experts. Once trained with the dedicated loss function, the predictor directly outputs the anticipatory IBN action.

it directly outputs the decision needed to meet the management intent.

However, state-of-the-art models for capacity forecasting employ loss functions that are designed manually, based on expert knowledge [37]. This strategy has several limitations that make it unsuitable for IBN: (A) it requires human intervention, hence is not aligned with the vision of a full-fledged IBN activation stage where the whole decision process is automated; (B) it assumes that an effective loss function can be devised by hand, which is not the case when the relationship between the actionable network parameters and the management objective is, e.g., not known a priori, or especially tangled[1]; (C) it must abide by the requisite that loss functions be differentiable, so that popular optimizers based on gradient descent can be used for training [38] whereas machine-translated objectives may not be differentiable.

Current definitions of tailored loss functions for capacity forecasting aim at avoiding all underprovisioning by assigning a very high cost to predicting values below the demand, while penalizing additional overprovisioning [5, 37]. In fact, this suits perfectly the IBN activation toy example set out in Chepter 1: the capacity prediction avoids underprovisioning of CPU resources, hence meeting the intent objective. Yet, this is achieved via human-defined loss functions, and the approach suffers from the problem (A) above. Problems (B) and (C) do not apply to our toy example, as the objective is fairly simple; however, they can easily emerge in more complicated settings.

2.1.3. Reinforcement learning

The general problem of automated decision-making during IBN activation naturally lends itself to be solved via Reinforcement Learning (RL). Indeed, RL does not require defining a loss function and is the standard approach to deal with abstract but measurable objectives. However, RL is not well suited for the specific, forecasting type of task we are

[1]For instance, loss functions proposed for capacity forecasting present a pie-cewise linear design, and cannot capture non-linear or multivariate objectives.

tackling: it operates on a limited set of well-defined discrete actions, which in our context unnecessarily restricts decisions to quantized levels. As a matter of fact, previous works where RL is employed for prediction are set in the stock market ecosystem, where the aim is anticipating price fluctuations to take simple buy or sell decisions [39,40]. Different from stock markets, network management benefits from predictions on a continuous space, which ensures that, e.g., the exact required amount of resources are allocated in concert with fluctuations in the demand. Even if RL methods for regression-type problems exists, those did not really emerge and their results are not convincing. In fact, as later discussed in Section 3, this thesis' proposal can be interpreted as a way to reconcile RL with the continuous-value forecasting task that is needed for IBN activation.

2.2. Meta-learning for deep neural networks

Meta-learning, also referred to as learning-to-learn, overcomes the limitations of fixed learning-based models and allows automatically tuning different aspects of the learning algorithm to the target task [41]. Meta-learning has been successfully applied to, e.g., distillation [42], augmentation [43] or batching [44] of training data, initialization [45] or optimization [46] of the model parameters, tuning [47] of its hyper-parameters, and discovery [48] of the actual architecture, possibly as a composition of modules [49].

One of the focuses of this thesis is on meta-learning of loss functions, which aims at learning the parameters, components, or shape of the loss to be used to train the actual model. The problem can be seen as an instance of a hierarchical optimization, where a meta-model is optimized under a constraint represented by the main model optimization [50]. We stress that this is instead semantically different from meta-learning optimization schedules in iterative and alternate optimization processes [51].

Three main approaches to loss meta-learning have been explored to date in the literature, and we detail them next.

2.2.1. Learning to parametrize a predefined loss

The majority of works on loss meta-learning propose dedicated models to infer the most suitable configuration of a predefined, parametrizable loss function. Here, a number of studies have investigated the use of decision networks to select among a set of predefined (family of) loss functions [52, 53], or to learn the parameters of known and differentiable meta-losses [54]. Other relevant investigations have focused on multi-part loss functions, where the goal is setting [55, 56] and possibly dynamically updating [57] the function weights based on (live) performance metrics. Also related to the same concept are strategies such as training a network to correct the optimization trajectory produced by a fixed loss [58], or introducing general loss functions that contain hyper-parameters to be learned during training along with the neural network parameters [59].

In all these cases, the loss independently of whether it is expressed as a tunable function or set of primitives must be designed or selected manually, which is not possible when the performance metric of interest is not known a priori. Instead, we seek a solution that can learn a clean-slate loss.

2.2.2. Learning surrogate loss functions

Surrogate losses are often used as a proxy for discontinuous or otherwise challenging prediction-metric relationships that cannot be directly used to drive the learning process. Surrogates are typically task-specific and handcrafted, which involves significant manual effort for each new task and of course requires prior knowledge about the relationship of interest. To remove the need for time-consuming human expert intervention, recent proposals have explored the possibility of meta-learning surrogates. Such solutions have proposed to express the performance metric as a function of a set of simple surrogates [19], or to compose a loss function from primitive mathematical operators [20]. A more elaborated strategy is that of learning the loss as a convex-by-construction function within a given parametric family (e.g., MSE or other quadratic operations of the loss input) and identify the exact parameters providing the best performance [21].

All the proposals above still require a-priori knowledge of the original relationship to identify a relevant set or family of surrogates, hence do not answer to the need of learning the loss at model runtime upon deployment in the target system, which is our main target. A closer design to the one we adopt in this thesis is a clean-slate surrogate learning based on a neural network modeling of the loss [60]. Yet, the approach is intended for classification tasks only and is not adapted for regression: e.g., it explicitly makes the result invariant to the ordering of the minibatch samples, which is discrepant e.g., with a time series forecasting goal; or, it adopts a bilevel programming optimization of the model parameters that we later show not to perform well in our target regression tasks.

2.2.3. Learning to teach

The concept of representing the loss function via a neural network has been in fact explored beyond the context of surrogate losses, as the so-called learning-to-teach paradigm. In an early work, the use of a teacher network was proposed to dynamically train parameters of a loss function that adapts to the learning stage of the main model [61]; yet, this approach still relies on a generic known loss function to be parametrized by the teacher. The seminal idea of a trainable task-parametrized and clean-slate loss generator was introduced for reinforcement and supervised learning by the meta-critic model, where an action-value function neural network learns to criticize the actions in a specified task [62, 63]. However, the meta-critic model is only applied to supervised learning problems as a tool for pre-training in the few-shot learning of new tasks (e.g., by

generalizing to unseen value ranges in the same domain), and is explicitly indicated by its authors as inappropriate for single tasks like the ones we consider.

It is also worth noting that meta-critic and its extensions [19, 64–66] are dedicated to discrete-space classification tasks or ranking problems. The same is also true of recent proposals to employ genetic programming tools to learn clean-slate loss functions [67, 68].

2.3. Explainability

In recent years, the interest in promoting trust and resilience in Information and Communications Technology (ICT) systems has gained momentum. In response, the landscape of regulations at both national and international bodies is continuously evolving and several initiatives involve EXplainable AI (XAI) [18].

2.3.1. Techniques and visualization tools

To date, various XAI techniques and visualization tools have been developed, primarily focusing on computer vision and natural language processing domains. These techniques can be broadly classified into two groups: model-agnostic and model-specific. Model-agnostic methods such as SHAP [69], Lime [70], and Eli5 [71] explain predictions by perturbing model inputs to gauge the significance of features. These techniques vary in how they compute relevance scores. In contrast, LayeR-wise backPropagation (LRP) [72] is a model-specific approach that identifies relevant neurons based on input data, offering insights into influential input features.

Complementing these techniques are visualization tools aiding in the identification of input elements steering predictions and monitoring changes in hidden states. For instance, TSViz [73] provides a 3D visualization tool tailored for convolutional deep learning models. Long-Short Term Memory (LSTM)-Vis [74] and Sequence to Sequence (Seq2Seq)-Vis [75] are tailored for LSTM and Seq2Seq models, respectively, catering to Natural Language Processing (NLP) applications. In contrast, ML-EXray [76] focuses on identifying preprocessing errors and optimizing model performance.

2.3.2. Adversarial attacks

The concept of adversarial attacks on neural networks was introduced in the seminal work by Szegedy et al. [77] showcasing how a slight perturbation in input can effectively deceive a classifier. This notion was exemplified by instances such as the placement of a tape strip on a speed limit sign, causing a classifier to misinterpret it and accelerate instead of braking. Moreover, the study revealed that these input perturbations possess a distinct pattern rather than being random occurrences. Remarkably, when the same

perturbation is applied to a different neural network trained on a different subset of data, it similarly misclassifies the input.

Under the realm of Adversarial attack techniques, perturbation serves as a fundamental tool for assessing the resilience of models against such attacks. These attacks can be categorized into white-box, gray-box, or black-box methods, depending on the amount of information the attacker has. White-box attacks presume full knowledge of training data, model architecture, and parameters, whereas black-box attacks lack any such information, with gray-box attacks occupying an intermediary position.

The pioneering adversarial attack, known as the Fast Gradient Sign Method (FGSM), was introduced in 2014 [78]. This method involves introducing an imperceptibly small perturbation to an image, aligning the perturbation's element values with the sign of the elements of the cost function gradient, thereby increasing the classification error. An iterative version of FGSM was proposed later in [79] and achieves higher effectiveness in crafting adversarial inputs at the expense of higher computational cost. Although created for images, the two methods have been tested for univariate and multi-variate time-series [80].

Adversarial attacks can further be categorized as targeted or untargeted, with the former aimed at modifying specific predictions and the latter directed towards degrading overall model accuracy. In this context, [81] proposes novel strategies involving perturbation masking and tuning-and-scaling, tailored for data and model poisoning.

Part II

MetaLoss model, applications and performance

3 MetaLoss model design and implementation

Overall, prior studies on loss meta-learning have focused on (i) parametrizable loss functions or (ii) clean-slate losses for classification tasks. Little attention has been paid to the meta-learning of clean-slate losses for regression. Part of the reason comes from the fact that loss meta-learning has been considered to create an indirection that makes single-task regression less efficient [62], under the assumption that losses such as Mean Absolute Error (MAE) or Mean Squared Error (MSE) can already optimally drive training in that case. This part of the thesis challenges this assumption and shows that there exist practical use cases, e.g., in system engineering, where loss meta-learning benefits single-task predictions. By investigating meta-learning solutions that target loss functions for regression tasks, this sheds new light on the advantages that this emerging paradigm can bring to a class of machine-learning problems where loss meta-learning has been overlooked.

The proposed approach consists of a loss-function-agnostic regressor that performs twofold learning. First, it must learn the output values that minimize a certain loss function. Second, it must also learn which is the said loss function according to a posteriori system measurements. This approach is particularly beneficial for resolving both the Loss-Metric mismatch issue and the challenge of aligning multiple predictors towards a common objective. Further elaboration on each of these problems will be provided in subsequent chapters, detailing the capabilities of the model in addressing them.

3.1. High level concept

A novel approach based on machine-learning for regression under imperfect knowledge of the loss-metric relationship is proposed, i.e., when we cannot fully characterize a priori the mapping between the decision triggered by the model and its impact on the objective performance but we are able to measure the system performance based on a metric a posteriori. This is due to the two main problems depicted in the Introduction (Chapter 1) of this thesis, i.e., the loss-metric mismatch and the common goal objective.

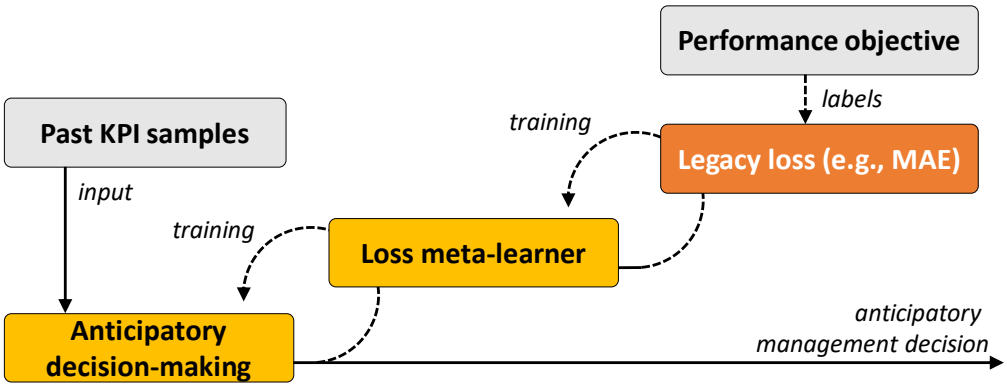


Figure 3.1. Anticipatory IBN activation with **MetaLoss**. The network management objective is learned and encoded into a *loss meta-learner*. This block then serves as the loss function to train the *predictor*, so that it directly outputs the anticipatory IBN action. The whole process is automated.

This design, named **MetaLoss**, is built on an elemental idea: we do not pre-select or assume a specific predefined shape or equation for the loss function; instead, we let the model find through learning the function that best characterizes the corresponding network management objective, providing full freedom to explore the possible relationships. This freedom is achieved thanks to the fact that feedforward neural networks are universal approximators [82] and great shape-learners.

In practical terms, we compose the *predictor* with another ML block that takes the role of a loss function meta-learner, as illustrated in Figure 3.1. In other words, this piece of the system aims at apprehending the relationship existing between the target network management objective and the prediction (i.e., anticipatory decision) made directly by the system. In such a manner, once the loss meta-learner has finished its training and has learned a certain function, it becomes an automatically tailored loss function that can assess the quality of the decision taken by the *predictor* given a certain system state and the considered network management objective.

As detailed later, the implementation chosen in the subsequent analyses for the previously described *predictor* and loss meta-learner is Deep Neural Network (DNN)-based, although such a choice is not a limitation of the model and other choices are possible. Particularly, we use a specific kind of DNN named Implicit Neural Representation (INR), described later in this chapter.

It is worth noting that, as shown in Figure 3.1, the loss meta-learner is trained so as to minimize a legacy standard loss function (e.g., MAE or MSE) of its output with respect to the network management objective.

This choice of a standard loss function is based on a twofold rationale. First, it is aligned with the idea that the meta-learner shall be trained to simply minimize the difference between the estimated and actual performance of the management decision: in

particular, the meta-learner does so by directly using performance measurements collected in the target system, removing all need to formalize its operation as a mathematical function as it happens instead with recent custom loss strategies [5, 83]. For the different experiments and use cases of this manuscript, the MAE function has been chosen as the loss training the *loss shaper* block.

Second, it makes the approach general and applicable to many different tasks, as it does not involve any (application-specific) expert knowledge: we will show later in the manuscript that the model can recover the performance of expert knowledge-based solutions without explicitly utilizing such knowledge. It will also be demonstrated that even using multiple intertwined signals where multiple predictions have an impact on each other (game theory problem), `MetaLoss` is capable of giving a persuasive answer to the problem by maximizing the overall output of the system.

As a result, `MetaLoss` handles the fundamental limitations of previous approaches: (*i*) it can learn the function mapping the regression decisions with the management objective directly from measurements with no human intervention; (*ii*) it does not require prior system knowledge to correctly characterize entangled, non-linear and multivariate common objectives that characterize network management tasks in particular. Finally (*iii*), its design allows for a greater explainability and security as well as a broader understanding of the system it is used on. This last point will be developed in Chapter 6.

3.2. Relation to reinforcement learning

While it lies within the category of supervised learning methods, the approach proposed above has certain conceptual similarities to Reinforcement Learning (RL), from the viewpoint that the learning process is based on observations of the outcome of the taken decisions in both cases. Yet, there are also important differences and some advantages for `MetaLoss`, which we describe next.

First, RL is known to be best suited for discrete-space decisions and many solutions suffer from scalability issues. However, `MetaLoss` is a method for regression problems (and especially applied on forecasting tasks) that is naturally designed for the continuous input and output spaces that are often encountered in network management tasks, while also being compatible with discrete spaces.

A second crucial aspect is that our model inherently separates the logical components of (*i*) anticipatory decision-making, implemented by the *predictor* element, and (*ii*) relationship between the decision and the resulting performance, embedded by the loss meta-learner that steers the learning process. This logical detachment allows us to isolate the learned loss function at the end of training, which is not possible with existing RL techniques. Isolating of the loss has two main advantages.

- **Explainability**: we can explore (e.g., by injecting controlled input) the trained neural network that implements the meta-learner in order to discover how different decisions impact the network performance, thus obtaining precious insights on the system operation that are not known at design time. More generally, this allows revealing how to the decision process occurs and makes the "reasoning" of the deep learning model way easier to explain, which is a much-demanded feature absent in the vast majority of complex black-box data-driven models proposed for zero-touch network management. This aspect will be described in Chapter 6.

- **Portability via transfer or few-shot learning**: once the loss is learned, the logical independence of the loss meta-learner allows reusing the learned loss in different settings where the decision-metric relationship is expected to be the same, but the statistics or correlations between inputs may vary: for instance, in scenarios where the same network management task needs to be performed in presence of diverse traffic demands, such as in different cities, urban versus rural areas, or across countries. Furthermore, this advantage can be exploited in cases where the loss is expected to be similar but not exactly the same as the learned one: as an example if an identical energy-optimization management task is to be run in the presence of hardware that entails different power consumption profiles. In such situations, a pre-trained loss meta-learner can be further fine-tuned in the new setting via few-shot learning approaches [84]. This transfer learning aspect will be explored in Chapter 4.

3.3. Problem formulation and notation

Let us denote the space of system state variables as $\mathcal{S} \subset \mathbb{R}^s$. The agent is a *predictor*, and we denote the input space of the *predictor* as $\mathcal{X} \subset \mathbb{R}^{n_1 \times n_2}$ and the output space as $\mathcal{Y} \subset \mathbb{R}^m$, such that the *predictor* can be modeled as $f_{\theta^p} : \mathcal{X} \rightarrow \mathcal{Y}$, where θ^p represents the parameters of the *predictor*. For a given input matrix[1] $\boldsymbol{X}_i = (\boldsymbol{x}_{i,1}^\top, \ldots, \boldsymbol{x}_{i,n_2}^\top) \in \mathcal{X}$, the model makes a decision denoted as $\hat{\boldsymbol{y}}_i = f_{\theta^p}(\boldsymbol{X}_i) = (\hat{y}_{i,1}, ..., \hat{y}_{i,m})^\top \in \mathcal{Y}$, where i represents the sample index.

For a certain input $\boldsymbol{X}_i$, we denote the performance cost of the *predictor*'s decision as $\mathcal{M}_i = f_{\mathcal{M}}(\hat{\boldsymbol{y}}_i, \boldsymbol{v}_i)$, where $\boldsymbol{v}_i \in \mathcal{S}$ denotes the external observations that may impact $\mathcal{M}_i$. Hence, in the common scenario in which the loss function (i.e., $f_{\mathcal{M}}(\hat{\boldsymbol{y}}_i, \boldsymbol{v}_i)$) is known, the *predictor*'s objective is to minimize the performance cost of its decisions based on $f_{\mathcal{M}}$,

[1] For the sake of clarity, the *predictor*'s input space is described as a regular $\mathbb{R}^{n_1 \times n_2}$ space. Nevertheless, for an input matrix $\boldsymbol{X}_i$ the proposed approach does not limit the different vectors $\boldsymbol{x}_{i,k} \in \mathbb{R}^{n_1}$, $k \in \{1, \ldots, n_2\}$, to have the same size.

i.e., to solve the following optimization:

$$\min_{\theta^p} \quad \sum_i (f_{\mathcal{M}}(f_{\theta^p}(\boldsymbol{X}_i), \boldsymbol{v}_i) \tag{3.1}$$

However, we recall that the objective is characterized by an *a priori unknown* expression. Note that, even if this expression is not known, the performance is assumed to be measurable, and samples of $f_{\mathcal{M}}(\cdot)$ can be obtained by observing the outcome of the *predictor*'s output on the system. This assumption holds true nearly always in scenarios involving networking management problems.

The relation $f_{\mathcal{M}}(\cdot)$ is uncharted at first, and the model must learn it by means of a second optimization function describing the loss meta-learner task. Such loss meta-learner takes as inputs (i.e., as coordinates of the learned loss) the *predictor* decision $\hat{\boldsymbol{y}}_i$ and the current observations $\boldsymbol{v}_i$; then, it casts a metric estimate $\widetilde{\mathcal{M}}_i = f_{\theta^\ell}(\hat{\boldsymbol{y}}_i, \boldsymbol{v}_i)$, where θ^ℓ represents the parameters of the loss meta-learner process.

Consequently, **MetaLoss** is composed of two optimization problems. First, the loss meta-learner task aims at correctly characterizing $f_{\mathcal{M}}(\hat{\boldsymbol{y}}_i, \boldsymbol{v}_i)$ through the estimated $f_{\theta^\ell}(\hat{\boldsymbol{y}}_i, \boldsymbol{v}_i)$. This is done by computing a simple legacy loss function, e.g., MAE or MSE.[2] It follows that the loss meta-learner optimizer's objective is defined by

$$\min_{\theta^\ell} \quad \sum_i \; || \; f_{\theta^\ell}(\hat{\boldsymbol{y}}_i, \; \boldsymbol{v}_i), f_{\mathcal{M}}(\hat{\boldsymbol{y}}_i, \; \boldsymbol{v}_i) \; ||. \tag{3.2}$$

In turn, the *predictor*'s objective is to minimize the performance cost of its decisions w.r.t. the predicted performance $f_{\theta^\ell}(\hat{\boldsymbol{y}}_i, \boldsymbol{v}_i)$, i.e., to solve

$$\min_{\theta^p} \quad \sum_i \; f_{\theta^\ell}(f_{\theta^p}(\boldsymbol{X}_i), \boldsymbol{v}_i). \tag{3.3}$$

Note that, for the cases in which the loss function is known, the first optimization in (3.2) is not needed and (3.3) becomes $\min_{\theta^p} \; f_{\mathcal{M}}(\boldsymbol{y}_i, f_{\theta^p}(\boldsymbol{X}_i), \boldsymbol{v}_i)$, which is the standard definition of a regression problem.

3.4. The MetaLoss model

The proposed approach, named **MetaLoss**, is a general-purpose loss-function-agnostic regressor that realizes a bifold learning, as previously mentioned: On one hand, (*i*) it learns to predict multiple actions to jointly optimize a specific global objective. Yet, since the model is unaware of such optimization goal, (*ii*) it must also learn the appropriate loss function that will correctly optimize that goal from a-posteriori system measurements.

[2] The loss meta-learner block aims to mimic as closely as possible $f_{\mathcal{M}}(\cdot)$: hence, a generic loss function minimizing the error between $f_{\theta^\ell}(\cdot)$ and $f_{\mathcal{M}}(\cdot)$ suits well the purpose.

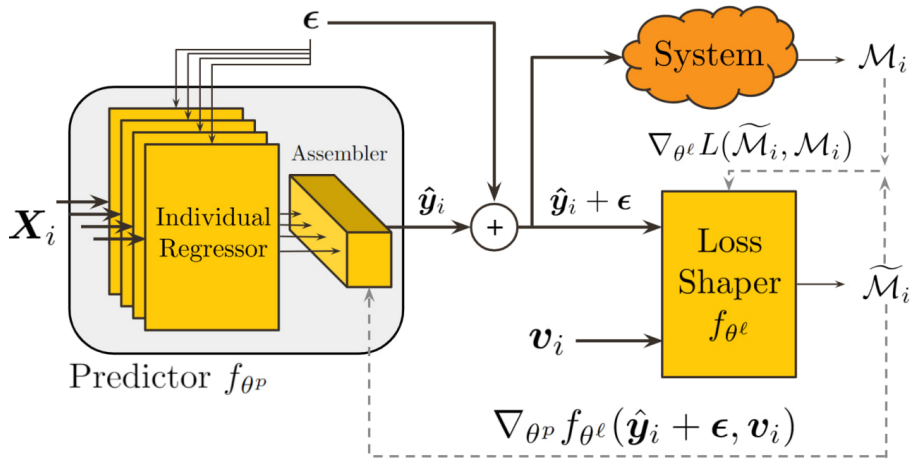


Figure 3.2. Dateiled architecture and training backpropagation process of the `MetaLoss` model.

We first provide a model overview to later describe the detailed implementation.

`MetaLoss` adopts a data-driven deep-learning approach that follows the general design for loss meta-learning outlined in Figure 3.2 to jointly solve the optimization problems (3.2)-(3.3). To this end, we distinguish two main blocks, as illustrated in Fig. 3.2: The first one, on the left, encloses the actual *predictor* that manages to solve (3.3), providing an output that fits the metric indicated by the second block, on the right, that represents the *loss shaper* block and that handles (3.2). That is, this second block learns and encodes the initially unknown relationship between the objective and the output of the *predictor*. Without loss of generality, both blocks are built upon DNN and, more specifically, the *loss shaper* uses an Implicit Neural Representation (INR) architecture. The whole process is automated.

An important aspect of the `MetaLoss` design is the inner structure of the *predictor*. As depicted in Fig. 3.2, the *predictor* actually consists of a set of different Individual Regressors (IRs) followed by one assembler. The rationale for this structure is that many practical management or engineering problems require the prediction of multiple system variables at once; such variables are often intertwined, i.e., take values that depend on each other. For instance, in multi-user resource allocation tasks, the system has a maximum physical capacity that cannot be exceeded, and predictions on the optimal resources reserved for each user are contingent on those booked for other users at the same time. Formally, in this exemplifying task, $\boldsymbol{X}_i$ can be decomposed in a set of vectors $(\boldsymbol{x}_{i,1}^\top, \ldots, \boldsymbol{x}_{i,m}^\top)$ for different user demands, and the corresponding resource reservations $\hat{\boldsymbol{y}}_i = (\hat{y}_{i,1}, \ldots, \hat{y}_{i,m})^\top$, are inter-dependent. In this setting, IRs are separate parallel layers that aim at predicting the correct output of their respective inputs; the assembler receives such estimated values and learns how they are both intertwined among them and related to the performance objective taught by the *loss shaper*.

MetaLoss is a conceptual model, and the focus of our work is not on improving the design of the *predictor* block over state-of-the-art regression algorithms such as N-BEATS [85] or DeepAR [86], but on addressing the loss-metric mismatch. In fact, the IR blocks in Fig. 3.2 can implement any prediction model, including those mentioned above. Yet, even a perfect *predictor* would not minimize a practical performance metric if trained with standard error-reduction losses.

The output of all IR is then gathered by a single *aggregator* block, which is an elemental part of the *predictor*: The *aggregator* learns the interaction between each of the input variables and how they are intertwined among them and with the performance metric, exploiting the across-dimension knowledge to provide predictions $\hat{\boldsymbol{y}}_i$ that optimize the network management objective.

Finally, the *loss shaper* implements $\widetilde{\mathcal{M}}_i = f_{\theta^\ell}(\hat{\boldsymbol{y}}_i, \boldsymbol{v}_i)$, i.e., it realizes the loss meta-learner described in the conceptual model in Figure 3.1. The purpose of the *loss shaper* is to meta-learn the loss to train the *predictor*, and therefore it is not active during inference. It generates the estimated cost that the decision $\hat{\boldsymbol{y}}_i$ produces in the system. Thus, its purpose is to learn as accurately as possible this cost of performance, and hence its loss must just capture the sheer distance between them.

As already noted, the internal structure of the *predictor* is an essential novelty that allows for forecasting from intertwined variables and provides key advantages:

- The challenging task of optimizing f_{θ^p} is sliced into simpler learning sub-tasks thanks to the logical structure based on parallel IR all feeding the aggregator. This is an architechture that is compared to other kind of similar conceptual ideas aiming to achieve the same result in Chapter 5.

- **MetaLoss** naturally lends itself to support modular transfer learning, and it does so in a twofold manner:

  - (i) For IR: it allows for separated pre-training of each IR on legacy loss functions to reduce the training time. This modular structure improves the scalability of the model (cf. Chapter 4), and its portability, since each pre-trained IR can be applied in other tasks with different loss functions but where the input variable is the same. In addition, during pre-training, each IR could be trained with a different standard/handcrafted loss function, which is particularly pertinent in scenarios where different IR handle inputs that are different in nature. Conversely, when pre-training is not considered and **MetaLoss** is trained as a whole, the IR are not directly trained with a specific loss function: the IR output is just the input of the aggregator, whose output is evaluated by the *loss shaper* to compute the loss.

- (*ii*) For *loss shaper*: since this block learns the unknown loss function that defines the *predictor*'s performance-output relationship irrespectively of the used regressor, the trained *loss shaper* block can be employed to train any *predictor* that faces the same complex problem, and it can be also used as an initial state for a new *loss shaper* in presence of different unknown functions that are expected to be similar.

Concerning the **MetaLoss** concept and training, the following important remarks are in order.

- The meta-learning model outlined above and detailed next is able to approximate a non-differentiable objective $\mathcal{M}$ by a *differentiable alternative* $f_{\theta^\ell}(\cdot)$, which is implemented by the *loss shaper* DNN upon training. This allows optimizing the *predictor* DNN under metrics that could not be directly used as losses, via a suitable approximation of the same.

- The **MetaLoss** design allows for co-training the *predictor* and loss estimator during the same gradient descent iteration so that each DNN is informed of (and can learn from) the improvements of the other. This makes the learned loss $f_{\theta^\ell}(\cdot)$ adapted to the inherent forecasting limits of the *predictor* as well as the *predictor* will be trained according to this same *loss shaper*.

It's worth noting that the preceding discussion primarily outlines the theoretical underpinnings of the model. In practical settings, training the predictor as a single entity and independently considering both the assembler and each individual IR to fulfill their respective tasks isn't feasible. Particularly, relying solely on one of these IRs, even if there is use of pertraining as described in the following section 3.6.1 won't yield the precise output required for addressing the specific problem associated with that IR. Similarly, expecting real predictions to serve as inputs for the assembler and generate a global configuration to resolve related common objectives is not feasible. This is specifically something that can be an interesting aspect and is developed in the potential future works in Section 8.1.

3.5. Learning the loss function

We explore here the two major features in order to learn the loss function properly, i.e., the nature of the loss shaper block and the loss exploration mechanism inspired by the exploration-exploitation mechanism present in reinforcement learning.

3.5.1. Loss shaper nature

As depicted earlier, for the *loss shaper*, we adopt an INR architecture with the motivation that this component does not aim at directly identifying the loss as a pure

mathematical expression, but rather as a geometric multi-dimensional shape based on coordinates. It thus makes sense to implement it with a model intended for shape reconstruction. INR frameworks have shown significant capability [87] to encode the functional relationship between a data sample and its coordinates (e.g., mapping a pixel position to its value) through a simple Multi-Layer Perceptron (MLP) architecture. INR frameworks were initially applied to 3D graphics and rendering [88], and they have recently gained popularity for other domains, such as computer vision [89] audio [90], video [91], or time series analysis [92], thanks to their ability to perform compression, super-resolution, and handling of missing values.

Formally, an INR approximates the mapping from the coordinate space $\mathcal{T} \subset \mathbb{R}^n$ to the signal or feature space $\mathcal{X}(t) \subset \mathbb{R}^m$ (for a single coordinate $t \in \mathcal{T}$) via an MLP f_θ with weights θ. As an example, in the image domain, the $\mathcal{T}$ space is the set of (a, b) pixel locations, the $\mathcal{X}$ space is the RGB color space, and an image $\boldsymbol{X}_i$ is the mapping $\mathcal{T} \to \mathcal{X}$. It can be extended to a multi-variate setting, where the coordinates $\mathcal{T}$ correspond to the different indexes, the output $\mathcal{X}$ is the multi-variate feature space. Each element can be seen as a function $\boldsymbol{X}_i$ from the coordinate space $\{j\}_{j=1,\dots,J}$ to the signal space and is given by a collection of pairs $d_i = \{(j, X_i(j))\}_{j \in 1..J}$ referring to both coordinates and values. Most Advanced INR models are so-called Sinusoidal Representation Networks SiReN [87]. SiREN models contain an extra hyperparameter ω_0 and are defined by the following periodic activation functions between layer h^l and h^{l+1}. We define the layer h^l with its weights W^l and bias b^l:

$$h^{l+1} = \sin(\omega_0(W^l h^l + b^l)) \tag{3.4}$$

Some works on shape reconstruction prefer to use differentiable signed distance functions (SDFs) [87] as target, for shape reconstruction purposes, i.e., learning the distance to the closest point of the loss for each prediction. However, such an approach is not suitable for the problem here considered, as it would imply either knowing in advance the shape of the loss or modeling it using purely random samples at first, which is not feasible for a real system. Therefore, in this thesis, we consider instead the direct approach of learning the shape of the loss.

The use of a neural network as a loss provides another very important feature that is necessary to perform the training of the *predictor* DNN. Given its intrinsic nature it can be represented as a complex polynomial expression, augmented by a combination of activation functions. This ensures the function's differentiability, a pivotal requirement for effective training. This property enables seamless integration into the training process of the predictor through the straightforward employment of gradient descent algorithms.

3.5.2. Loss exploration

The input of the *loss shaper* INR is implemented by a different expression than that indicated in the formal problem definition of Section 7.1. Specifically, as portrayed in Figure 3.2, instead of providing the computed action $\hat{\boldsymbol{y}}_i$ to the *loss shaper* INR, rather it is fed with a disturbed version of it, $\hat{\boldsymbol{y}}_i + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} = (\epsilon_1, ..., \epsilon_m)^\top \sim \mathcal{N}_m(0, \Sigma_t)$, and Σ_t evolving during the training process. Therefore, the weight updates at time t, for both the *predictor*'s DNN (θ_t^p) and the *loss shaper* INR (θ_t^ℓ) are computed as usual by gradient descent using the following expressions.

$$\theta_{t+1}^p = \theta_t^p - \alpha_t^p \, \frac{\partial f_{\theta^\ell}(\hat{\boldsymbol{y}}_i + \boldsymbol{\epsilon},\ \boldsymbol{v}_i)}{\partial \theta_t^p} \tag{3.5}$$

$$\theta_{t+1}^\ell = \theta_t^\ell - \alpha^\ell \, \frac{\partial L(f_{\theta^\ell}(\boldsymbol{y}_i,\ \hat{\boldsymbol{y}}_i + \boldsymbol{\epsilon},\ \boldsymbol{v}_i),\ f_{\mathcal{M}}(\boldsymbol{y}_i,\ \hat{\boldsymbol{y}}_i + \boldsymbol{\epsilon}, \boldsymbol{v}_i))}{\partial \theta_t^\ell} \tag{3.6}$$

The learning rates of the *predictor* and the *loss shaper* DNN are, respectively, α_t^p and α^ℓ. The dependency of α_t^p on t is due to the use of CLR mentioned above in Section 3.6.2.

The noise $\boldsymbol{\epsilon}$ is only used in training, and it is set to 0 once the expression of the loss $f_{\theta^\ell}(\cdot)$ is learnt, during testing. In this regard, a critical design feature of **MetaLoss** is that $\boldsymbol{\epsilon}$ is also input to the regressor DNN: during training, this lets the prediction block learn the correlation between such input and the added disturbance to its output. Then, during inference, setting $\boldsymbol{\epsilon}$ to 0 allows producing outputs $\hat{\boldsymbol{y}}_i$ that are not biased by the loss exploration used in training.

The goal of the random variable $\boldsymbol{\epsilon}$ is to allow for further exploration of the input values, supplying the *loss shaper* with a broader observation of the input domain beyond that provided by the training samples, and improving the reliability of the characterization of the loss function over the continuous domain. There is in fact an intuitive analogy between this additive noise and the approach applied in reinforcement learning RL of taking a non-optimal typically random decision with some probability, creating an exploration-exploitation trade-off. Similar to what occurs in RL, we observe that the noise is most beneficial when it exponentially decays as the training advances. More details for practical use cases are presented in the next Chapter 4 and Chapter 5.

3.6. Training process

In this section, we delve into the different features of the training process of the **MetaLoss** model. Those are optional but improve results in the different use case scenarios presented in Chapter 4 and Chapter 5.

3.6.1. Pre-training of individual processing units

`MetaLoss`'s design is steered towards facing the challenge of decreasing the learning time, because the particular structure of the *predictor* block makes it possible to individually pre-train IRs: we can optimize each $\mathrm{IR_k}$, $k \in \{0, ..., m\}$ to minimize $L(\hat{y}_k, y_k)$, i.e., training each IR as a standard regressor. The definition of loss L can be any loss function, from manually crafted by expert humans to legacy loss function. This speed-up will depend on the similarity between the loss used to pre-train the model and the final loss that the *loss shaper* will estimate. Note that, even if such pre-training is carried out with a legacy $L^2(\cdot)$ function, it turns out that it accelerates the learning process in most realistic settings. For the different experiments and use cases presented in this thesis, the nature of the different IR will be Long-Short Term Memory (LSTM) as we are targeting specifically forecasting problems. It will be interesting to use more advanced models such as Transformers for representing the IR. More details are presented in the Section 8.1

3.6.2. Cyclic learning rate

We incorporate the Cyclic Learning Rate (CLR) method [93] into the design. CLR consists of dynamically adapting the learning rate during training to better explore the properties of the gradient of $f_{\theta^\ell}(\hat{\boldsymbol{y}}_i, \boldsymbol{v}_i)$ during the process of optimizing the weights θ^p of the *predictor*. This method is beneficial especially because, in realistic use cases, the representation of $f_{\theta^\ell}(\hat{\boldsymbol{y}}_i, \boldsymbol{v}_i)$ learned by the *loss shaper* (or loss meta-learner) is a multivariate high-degree polynomial that does not have saddle points thanks to its DNN nature.

CLR makes the learning rate vary within a certain range, where the extreme values of such range are preselected hyperparameters. In this manner, CLR prevents Stochastic Gradient Descent (SGD) from getting stuck in local minima. For simplicity, in our experiments, we consider a basic triangular version of CLR, with 3 cycles across the full training phase. The changes in the learning rate are defined by the designer, e.g., by defining that it follows a triangular function between two values, or an exponential decay function. Eventually, CLR enables faster automatic convergence for any shape of the network management objective function $f_{\mathcal{M}}$, and it also gratefully aligns with the AutoML principles embraced by `MetaLoss` because it automates part of the machine learning hyperparameter configuration: it reduces the model sensitivity to the initial value of the learning rate by autonomously varying it, preventing the learning from being stuck due to a wrong default learning rate choice.

3.6.3. Co-training of predictor and loss shaper

`MetaLoss` is implemented as two cascaded DNN, as illustrated in Fig. 3.2. The *loss shaper* block is fed by the current observations and the *predictor*'s output. This allows

jointly optimizing the two blocks through the same backpropagation process. The weights of the two DNN are optimized during training as follows.

First, during the forward pass, the *predictor* f_{θ^p} is fed with a set of observations of the system state and outputs its decision $\boldsymbol{y}_i$. External observations $\boldsymbol{v}_i$ that may impact the metric, but not directly the prediction are measured and both are passed to the *loss shaper*, which computes the estimated performance function $\widetilde{\mathcal{M}}_i = f_{\theta^\ell}(f_{\theta^p}(\boldsymbol{X}_i, \boldsymbol{\epsilon}) + \boldsymbol{\epsilon}, \boldsymbol{v}_i)$. At the same time, the actual performance of the decision $\mathcal{M}_i = f_{\mathcal{M}}(f_{\theta^p}(\boldsymbol{X}_i, \boldsymbol{\epsilon}) + \boldsymbol{\epsilon}, \boldsymbol{v}_i)$ is measured.

Then, the mismatch between estimated and true performance is evaluated via a legacy or standard loss function, and backpropagated first to the *loss shaper* INR. Here, the *loss shaper* updates its weights θ^ℓ to better capture the relation between $\mathcal{M}_i$ and the combined values of the prediction $\hat{\boldsymbol{y}}_i$ and the system state $\boldsymbol{v}_i$. Within the same iteration, the updated loss is sequentially backpropagated to the *predictor* DNN, which allows improving the alignment of the prediction with the optimal decision that minimizes $\mathcal{M}_i$.

This design increases the efficiency of the training phase with respect to the case where each block is optimized independently, e.g., by feeding the *loss shaper* block with random predictions and, once the loss has been learned, using it to train the *predictor*. Indeed, co-training allows learning a loss $f_{\theta^\ell}(\cdot)$ that is adapted to the intrinsically limited accuracy of the *predictor*; as an example, co-training may lead to learning diverse shapes of the loss depending on the magnitude of the target variable if the quality of the prediction is found to be affected by the absolute value of the target variable. We will observe practical situations where this type of adaptation occurs in the following Chapters 4 and 5.

It is worth noting that such a co-training represents a major novelty of our model with respect to previous related proposals [60–62]. Indeed, the end-to-end backpropagation training was not possible in prior models, and the two elements (the learning-to-act block and the learning-to-correct block) were trained either iteratively or in a nested manner.

3.6.4. Complexity

Computational complexity is an important metric to evaluate the efficiency of anticipatory decision-making algorithms for network performance optimization, especially in the context of large-scale infrastructures and services with stringent latency requirements. In order to assess the overall complexity of **MetaLoss**, we quantify its training time across its two main components, i.e., the *predictor* and the *loss shaper*. Let us define as N_{Pred} (resp. N_{LS}) the number of input features to the *predictor* (resp. *loss shaper*), as L_{Pred} (resp. L_{LS}) the number of layers in the neural networks, as M_{Pred} (resp. M_{LS}) the average number of neurons in each layer, and as T_{Pred} (resp. T_{LS}) the number of training samples; then, the *predictor* neural network has a total training time complexity of $C_{Pred} = O(T_{Pred} \cdot N_{Pred} \cdot L_{Pred} \cdot M_{Pred})$, whereas the *loss shaper* has a complexity $C_{LS} = O(T_{LS} \cdot N_{LS} \cdot L_{LS} \cdot M_{LS})$.

As both parts are trained together using a single backpropagation, the full training time of the model can be described by $C_{\texttt{MetaLoss}} = C_{Pred} + C_{LS} + c$, where c is the residual time caused for some transformation between the output of the *predictor* and the input of the *loss shaper*. For the predictions, only the *predictor* is used and thus the model does not take longer inference than any other DNN model. A detailed time aspect is defined on a specific use case in Chapter 5.

3.7. Summary

This chapter introduced a novel model named `MetaLoss`, designed specifically to address emerging requirements crucial for achieving the objectives of Intent-Based Networking (IBN). These objectives primarily include resolving issues such as the mismatch between loss functions and system metrics and establishing common objectives across different predictors. These functionalities are imperative for the successful execution of new standard Management and Network Orchestration (MANO) tasks. The subsequent chapters delve deeper into these problems, offering additional examples and use cases to illustrate their significance.

4 Loss metric mismatch and single predictor scenarios

The primary objective of this chapter is to focus into the mechanisms by which the model, as outlined in Chapter 3, addresses the issue of loss metric mismatch, fulfilling the objectives delineated by the IBN objectives.

In tackling this challenge, we narrow our focus to scenarios where the model's *predictor* described in Chapter 3 comprises a singular IR. Consequently, we temporarily set aside the complexities associated with intertwined forecasts, reserving them for detailed exploration in the next Chapter 5.

This chapter proceeds by elucidating how the model's efficiency in rectifying loss metric discrepancies is empirically verified through controlled experiments with comparison with established benchmarks. Furthermore, its real-world applicability is underscored through practical scenarios where its adoption not only proves beneficial but also potentially indispensable.

Additionally, this chapter performs an in-depth examination of various facets of the model, including its aptitude for transfer learning and its noise exploration process. These extended analyses serve to furnish a comprehensive understanding of the model's capabilities, thereby enriching the discourse surrounding its deployment and optimization.

In order to achieve different experiment described below, the architecture used was the following: The regressor block is implemented with 3 LSTM layers using respectively (128, 128, 64) neurons. The activation function used in every layer is the ReLu one. The loss shaper is composed of 3 layers, with respectively (256, 256, 128) units in it.

4.1. Measurement datasets

Before presenting the different analyses and use cases that we consider, we describe the datasets that we use in the remainder of the document for the sake of clarity, as we make use of these datasets in several sections. We have three different datasets, each containing different metrics and considering different use cases. Each dataset is unevenly used across the use cases. The first one, containing real-world measurements about traffic

demand of services in a commercial country-wide cellular network, is the main dataset, and it is used for most of the different analyses. The other two datasets (of power grid demand, and generic time series) are also considered here to generalize the results and offer a broader view of the implications and potential of the proposed approach. These datasets are described as follows.

- **Dataset** `TrafficApp`. This dataset contains real-world mobile data traffic demand generated by twelve popular service providers in a large metropolitan area during several consecutive months [94]. The data was collected and aggregated by the network operator using passive measurement probes, resulting in traffic levels (in bytes) every five minutes, for more than 22,000 samples for each of the services. Individual IP sessions were mapped to specific services using Deep Packet Inspection (DPI) and commercial traffic classifiers deployed by the operator. The data was geo-referenced via User Location Information (ULI) extracted from the Packet Data Protocol (PDP) Contexts and Evolved Packet System (EPS) Bearers in the GPRS Tunneling Protocol control plane (GTP-C).

  The data was processed in secure premises by the operator, in compliance with international regulations, and under the supervision of the relevant data protection officers. For our study, we had access to de-personalized aggregates of traffic time series at core network and Edge datacenters. `TrafficApp` is the main dataset in the subsequent experiments because of the quality, quantity, and relevance of the data that it contains. In general, 9 weeks of data are employed for training, 1 week for validation, and 1 week for testing.

- **Dataset** `PowerGrid`. This open-source dataset describes the hourly energy consumption in a part of the Eastern Interconnection grid in the USA between 2002 and 2018. The data comes from a regional transmission organization (RTO) [95], and it incorporates information from different power providers, each one of them managing a different geographical area. Overall, the time series consists of more than 145,000 samples.

- **Dataset** `M4data`. The M4 dataset is an open-source data collection created for the 4th Makridakis forecasting Competition [96]. It is composed of 100,000 time series with different seasonality m: yearly, quarterly, monthly, weekly, daily and hourly series. `M4data` was collected by randomly sampling 100,000 time series from the ForeDeCk database, and the time series were anonymized to guarantee objectivity, included time stamps.

4.2. Controlled experiments

In this section, we highlight the performance of the `MetaLoss` model using at first a controlled experiment and comparing it with multiple benchmarks.

4.2.1. Controlled environment use cases

As here the focus is on the relative performance of the different models, we perform tests in a controlled environment that favors the interpretability of the results. Specifically, we assume a setting where the objective of IBN activation, i.e., $f_{\mathcal{M}}$ in the notation of Section 3.4, is known and simple enough to be expressed in a closed form. This allows manually designing a loss function tailored to the objective, and use it in a baseline benchmark. To test generalization, we consider four different use cases with diverse forms of $f_{\mathcal{M}}$, as follows.

- **Traffic forecasting under absolute error.** The objective is a traditional traffic forecasting, i.e., predicting a $\hat{y}_t$ that matches the future data traffic volume x_{t+1}, with an error cost that is linearly proportional to the absolute discrepancy. Formally, $\mathcal{M}_{t+1} = |\hat{y}_t - x_{t+1}|$, which is optimized by a MAE loss.

- **Traffic forecasting under squared error.** Same, but larger errors tend to have an increasingly higher cost for the operator, with a quadratic growth. Formally, $\mathcal{M}_{t+1} = (\hat{y}_t - x_{t+1})^2$, which is minimized by a legacy MSE loss function.

- **Resource allocation with probabilistic guarantees.** The IBN objective is providing a probabilistic guarantee on the anticipatory allocation of resources, so as to accommodate the future traffic demand x_{t+1} a fraction τ of the time. Formally, $\mathcal{M}_{t+1} = \tau \cdot \mathcal{R}\left(x_{t+1} - \hat{y}_t\right) + (1-\tau) \cdot \mathcal{R}\left(\hat{y}_t - x_{t+1}\right)$, where $\mathcal{R}(x) = x \cdot \mathbb{1}_{x \geq 0}$, and $\mathbb{1}_C$ is an indicator function that takes value 1 if condition C is verified and 0 otherwise. This is a quantile forecast that can be optimized via a pinball loss [97].

- **Capacity forecasting.** The goal of the operator is anticipating the capacity needed to (i) avoid an expensive monetary fee α incurred for non-serviced future traffic x_{t+1}, and (ii) limit unnecessary overdimensioning beyond x_{t+1}.
  Formally, $\mathcal{M}_{t+1} = \alpha \cdot \mathbb{1}_{\hat{y}_t < x_{t+1}} + (\hat{y}_t - x_{t+1}) \cdot \mathbb{1}_{\hat{y}_t \geq x_{t+1}}$. This IBN objective maps in fact to the capacity forecasting problem addressed by previous works in the literature, for which the expert-designed α-OMC loss function is available [5].

All experiments are run on real-world traffic generated by Facebook Live, and transiting in a core network datacenter of an operational 4G mobile network; see Section 4.1 above for more details on the data and its collection. We employ 9 weeks of data for training, 1 week for validation, and 1 week for testing.

4.2.2. Benchmarks

We compare `MetaLoss` with a wide range of benchmarks that include baselines, state-of-the-art models for loss learning, and variants of our proposed scheme.

4.2.2.1. Baseline approaches

Two different solutions are used as a basis for our comparative performance evaluation.

- `Manual` - The *predictor* described in Section 3.4 is trained with a loss function designed manually to fulfill the specific target objective. As anticipated, MAE, MSE, pinball and α-OMC losses are used in the four use cases.

- `Disjoint` - The prediction and loss-learning functionalities are logically separated: first, the *loss shaper* is trained in isolation, by inputting uniform random noise and measuring the resulting performance, so as to learn the correct loss for the objective; then, the *predictor* is trained using the loss learned by the *loss shaper*.

4.2.2.2. Loss-learning models

The Adaptive Loss Alignment (ALA) method is a state-of-the-art approach for loss learning in classification tasks [98]. While ALA has been originally proposed for discrete classification tasks, we adapt it for regression by: (i) changing the type of characteristics used for validation, replacing the (logarithmic) probabilities that the input pertains to each class with the first- and second-order statistics of the regression values; and, (ii) swapping the set of classification-oriented loss functions originally used by ALA with expressions that are suitable for regression. We test two ALA models, which differ by the expression used.

- `ALA-manual` applies ALA on a linear combination of all manually designed loss functions that are used separately in the `Manual` approach depicted above. The rationale is having ALA automatically select the correct loss function for each use case, by tuning the linear combination weights.

- `ALA-moldable` applies ALA on a single loss function with a highly parametrizable shape. The function, illustrated in Figure 4.1, can mimic any of the losses for the controlled environment use cases, and the experiment aims at testing if ALA can learn the correct values of the parameters x_0, x_1, x_2, y_1 and y_2.

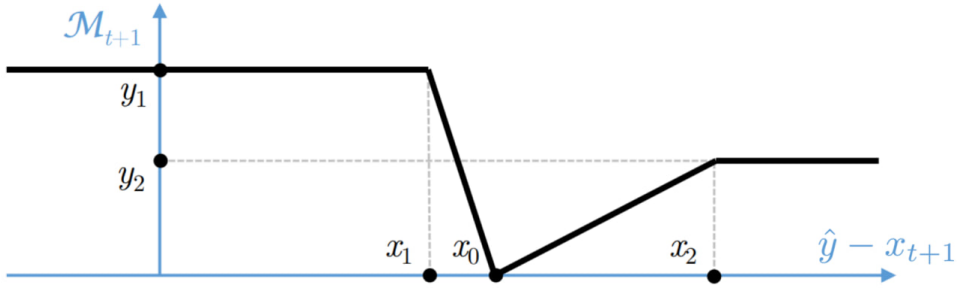


Figure 4.1. Parametrizable loss function used by ALA-moldable.

4.2.2.3. Variations of `MetaLoss`

As a form of ablation study, we test several variants of our proposed model, as follows.

- **`Fixed-rate`** operates as **`MetaLoss`**, but is trained with a legacy fixed learning rate, instead of an automated CLR.

- **`Noiseless`** uses the same architecture as **`MetaLoss`**, but does not include ϵ as an input to the *predictor*, which thus cannot learn the impact of the noise on its output; note that if no noise is added to the prediction either, the noisy exploration is completely skipped during training.

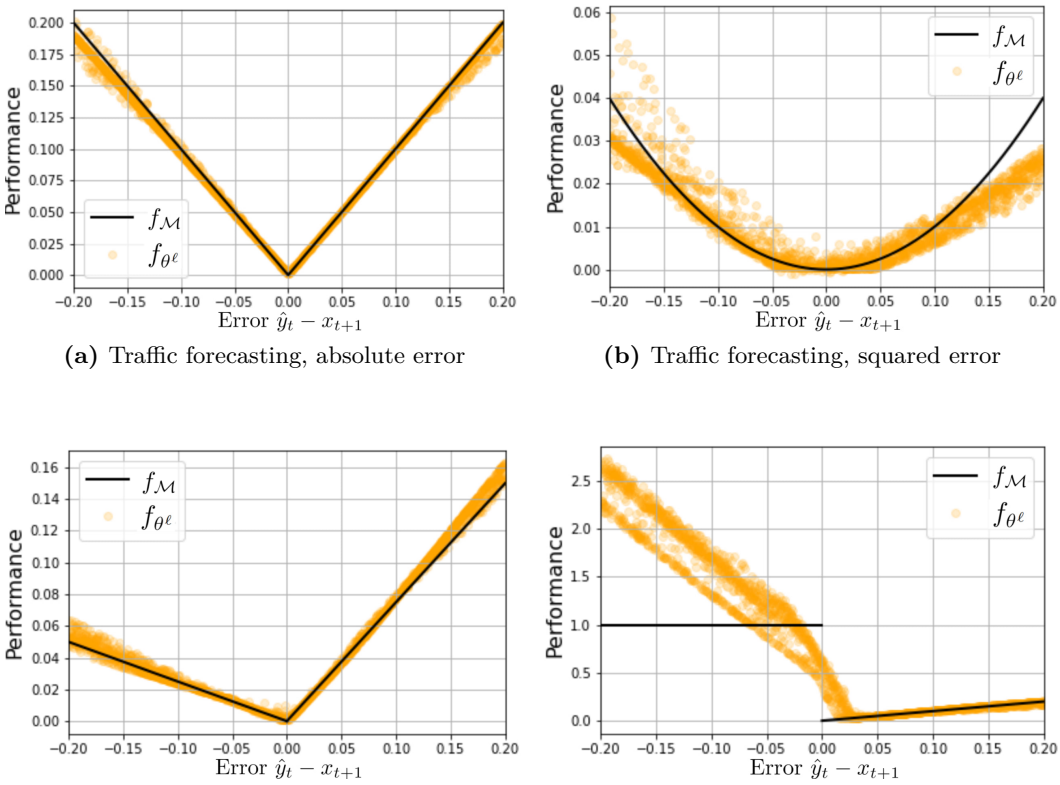
- **`Iterative`** adopts an alternating training strategy, instead of **`MetaLoss`** co-training. Specifically: the *predictor* block is trained in isolation during odd iterations, using the loss currently implemented by the loss-learning block; then, the *loss-shaper* block is trained during even iterations, by adding noise to the output of the *predictor*.

- **`Half`** is a complementary technique that can be adopted in combination with the **`Noiseless`** and **`Iterative`** approaches above. In a first moment, it trains the *predictor* and *loss-shaper* DNN jointly, and as mandated by either model; then, it freezes the loss-learning block and only keeps training the *predictor* for better convergence.

4.2.3. Controlled experiments

We first illustrate **`MetaLoss`**'s capability to learn a suitable loss function in the considered use cases. Figure 4.2 portrays the four loss functions f_{θ^ℓ} learned by our model, which map the error $y_t - x_{t+1}$ into the target system performance. The corresponding objective $f_{\mathcal{M}}$ is superposed to the learned loss to facilitate the interpretability of the result. The two shapes are well aligned in all cases, and thus the match is good. The only significant difference, which emerges in the case of capacity forecasting, is due to the fact that the original objective is not differentiable or even continuous, and hence it cannot be

directly used as a loss function: as mentioned in Section 3.4, and as a desirable by-product of co-training, **MetaLoss** learns a differentiable version of $f_{\mathcal{M}}$, so that the latter can be used to train the *predictor*.



(a) Traffic forecasting, absolute error

(b) Traffic forecasting, squared error

(c) Resource allocation, $\tau = 0.75$

(d) Capacity forecasting, $\alpha = 1.0$

Figure 4.2. Loss functions f_{θ^ℓ} learned by the loss meta-learner INR of **MetaLoss**, under diverse objectives $f_{\mathcal{M}}$.

Complete results from the controlled environment use cases are summarized in Table 4.1. Across all settings, **MetaLoss** stands out as the model with best performance, or a close second; more precisely, when not yielding the best result, our solution is typically within the variance of the method ranking first. A closer inspection of the exact figures reveals several important observations, as follows.

- The models that perform close to **MetaLoss** are those that involve human intervention, which is needed to define a tailored (and possibly parametrizable) loss function for the specific goal, such as **Manual** or **ALA-moldable**; instead, the training of **MetaLoss** is fully automated.

- The models performing best for some use cases tend to have highly fluctuating performance under other objectives, where they generate poor predictions; instead, **MetaLoss** performs consistently well across all target use cases, which demonstrates its flexibility and generality.

- **MetaLoss** produces results with sensibly lower standard deviation, which elicits a more consistent quality of anticipatory decisions.

- Juxtaposing **MetaLoss** with its variants proves that all design elements in Section 3.4 contribute to the performance of the model, and removing co-training (**Iterative**), noisy exploration (**Noiseless**), or learning rate adaptiveness (**fixedLR**) deteriorates results.

Overall, the results obtained in the controlled environments clearly showcase the gain of **MetaLoss** over other methods, in terms of sheer performance and flexibility. Importantly, this is attained while also reducing the need for human intervention.

4.2.4. Advantages of MetaLoss for plain traffic forecasting

We now evaluate the performance of the proposed solution for simple time series forecasting with standard loss functions such as MAE or MSE. This analysis aims at bringing light to one of the key aspects of **MetaLoss**: the fact that it can actually train the *predictor* to optimize a given loss function in such way that it obtains better results than if the *predictor* was directly trained with the true static loss function. As it is shown in Chapter 6, this is thanks to **MetaLoss** flexibility in constructing the loss function, being able to adapt the shape of the loss function for different values of the input data.

We challenged our model with the winner of the well-known M4 competition [99], i.e., the **ES-RNN** model [100], which combines exponential smoothing and a Recurrent Neural Network (RNN). The results are computed without the ensemble-learning part (for computational limits) and compared to our model by evaluating both solutions on the same M4 dataset. The M4 competition defined the winner according to three metrics: the mean absolute scaled error (MASE), the symmetric mean absolute percentage error (sMAPE) and the Overall Weighted Average (OWA), formally expresses as follows.

$$sMAPE = \frac{200}{H} \sum_{i=1}^{H} \frac{|y_i - \hat{y}_i|}{|y_i| + |\hat{y}_i|} \tag{4.1}$$

$$MASE = \frac{1}{H} \sum_{i=1}^{H} \frac{|y_i - \hat{y}_i|}{\frac{1}{H-m} \sum_{j=m+1}^{H} |y_j - y_{j-m}|} \tag{4.2}$$

$$OWA = \frac{1}{2} [\frac{sMAPE}{sMAPE_{Naive}} + \frac{MASE}{MASE_{Naive}}] \tag{4.3}$$

where H represents the number of samples in the subset of the dataset corresponding to a certain seasonality, and m the period of the season (e.g., 24 for hourly, 7 for daily, etc.).

Model		Noise ϵ	**Traffic forecasting, absolute**		**Traffic forecasting, squared**		**Resource allocation, $\tau = 0.75$**		**Capacity forecasting, $\alpha = 1$**	
			Train ($\times 10^{-2}$)	Test ($\times 10^{-2}$)	Train ($\times 10^{-4}$)	Test ($\times 10^{-4}$)	Train ($\times 10^{-3}$)	Test ($\times 10^{-3}$)	Train ($\times 10^{-2}$)	Test ($\times 10^{-2}$)
`Manual`		-	1.32 ± 0.12	1.62 ± 0.21	**2.64 ± 0.43**	4.04 ± 0.24	4.81 ± 0.35	5.99 ± 0.67	**3.30 ± 0.53**	**3.62 ± 0.61**
`Disjoint`		0.1	1.45 ± 0.11	1.80 ± 0.14	4.28 ± 0.85	6.50 ± 1.50	6.54 ± 0.42	8.26 ± 0.83	9.50 ± 1.10	14.80 ± 2.40
`ALA-manual`		-	2.28 ± 0.33	2.35 ± 0.36	6.80 ± 1.20	7.90 ± 1.70	11.20 ± 1.70	14.20 ± 2.04	8.91 ± 2.43	13.35 ± 3.17
`ALA-moldable`		-	1.17 ± 0.04	1.46 ± 0.04	2.71 ± 0.22	**3.87 ± 0.23**	5.03 ± 0.42	6.41 ± 0.55	40.04 ± 30.21	40.32 ± 30.78
`Noiseless`		0	1.71 ± 0.59	2.01 ± 0.58	5.12 ± 1.96	5.83 ± 2.29	6.13 ± 0.37	7.60 ± 1.13	6.21 ± 1.58	8.62 ± 3.55
		0.01	1.21 ± 0.13	**1.39 ± 0.14**	3.17 ± 0.90	5.30 ± 1.19	5.28 ± 0.41	6.43 ± 0.95	6.10 ± 1.27	8.88 ± 3.01
		0.1	1.19 ± 0.10	1.49 ± 0.15	3.32 ± 0.85	5.20 ± 1.20	18.04 ± 5.66	19.03 ± 6.87	17.16 ± 1.06	17.31 ± 1.30
`Iterative`	*plain*	0.01	1.49 ± 0.18	1.99 ± 0.21	4.10 ± 1.11	4.90 ± 1.33	6.29 ± 0.96	8.40 ± 1.90	6.67 ± 0.70	9.32 ± 1.66
		0.1	1.48 ± 0.15	2.01 ± 0.18	4.02 ± 1.02	5.82 ± 1.15	5.69 ± 0.69	7.34 ± 1.37	7.02 ± 0.63	8.12 ± 0.44
	Half	0.01	1.53 ± 0.11	2.01 ± 0.15	5.21 ± 1.31	15.96 ± 6.14	6.95 ± 1.98	9.09 ± 3.47	6.63 ± 1.02	7.96 ± 1.21
		0.1	1.54 ± 0.08	2.08 ± 0.16	4.85 ± 1.14	7.22 ± 2.47	6.10 ± 0.99	7.90 ± 1.65	7.65 ± 1.40	8.41 ± 1.59
`Fixed-rate`	*plain*	0.01	1.21 ± 0.09	1.54 ± 0.10	3.17 ± 0.75	5.80 ± 1.12	5.06 ± 0.22	6.31 ± 0.39	5.99 ± 0.80	7.94 ± 1.61
		0.1	1.31 ± 0.11	1.57 ± 0.10	4.28 ± 0.12	5.90 ± 1.01	5.25 ± 0.17	6.50 ± 0.26	6.28 ± 1.09	8.04 ± 1.33
	Half	0.01	1.24 ± 0.03	1.51 ± 0.02	3.72 ± 0.85	5.10 ± 0.85	5.33 ± 0.47	6.81 ± 0.61	6.05 ± 1.35	7.53 ± 1.21
		0.1	1.26 ± 0.03	1.55 ± 0.05	6.32 ± 2.18	7.79 ± 3.36	6.01 ± 0.52	6.42 ± 0.60	5.71 ± 1.04	7.94 ± 2.17
`MetaLoss`		0.01	**1.15 ± 0.02**	1.44 ± 0.01	2.77 ± 0.27	3.96 ± 0.20	**4.54 ± 0.05**	**5.61 ± 0.06**	3.72 ± 0.23	3.86 ± 0.27
		EDecay	**1.13 ± 0.03**	**1.36 ± 0.02**	**2.68 ± 0.19**	**3.84 ± 0.16**	**4.48 ± 0.06**	**5.57 ± 0.06**	**3.34 ± 0.29**	**3.53 ± 0.31**

Table 4.1. Comparative evaluation summary. The **best** and **second best** performing models are highlighted for each objective.

Metric	Algorithm	Hour	Day	Week	Month	Quarter	Year
sMAPE	`ES-RNN` [100]	12.397	3.032	9.188	12.592	**10.102**	13.508
	`MetaLoss`	**12.304**	**3.022**	**9.185**	**12.572**	10.227	**13.300**
MASE	`ES-RNN` [100]	1.346	1.129	2.603	**0.936**	**1.174**	3.046
	`MetaLoss`	**1.322**	**1.128**	**2.581**	**0.936**	1.260	**2.980**
OWA	`ES-RNN` [100]	0.618	0.980	1.057	0.876	**0.887**	0.796
	`MetaLoss`	**0.611**	**0.979**	**1.052**	**0.872**	0.907	**0.781**

Table 4.2. Comparison of our proposed model against the winner of the M4 competition computed on the M4 dataset.

The results are summarized in Table 4.2 and show that `MetaLoss` obtains better performance for all the considered seasonalities except for quarterly, where both values are comparable and lie within the confidence interval of each other. This result proves that `MetaLoss` achieves state-of-the-art performance even if its design is not focused on plain forecasting (but rather on joint loss-learning and forecast).

4.3. Hyper-parametrization and transfer learning

In this section, the objective is to focus into the fundamental features of the `MetaLoss` model by illustrating its functionality within a practical, real-world scenario. Through this simple, yet important use case, we aim to elucidate the significant role played by various hyperparameters in building the performance of the model. Additionally, we highlight `MetaLoss`'s great capacity by design for executing high-quality transfer learning, a capability that greatly enhances its adaptability and effectiveness across diverse tasks and datasets. By dissecting these aspects, we provide valuable insights into the inner workings and versatility of the `MetaLoss` model, offering a deeper understanding of its potential applications and implications in contemporary machine learning contexts.

Here, we consider a controlled yet realistic use case to finalize the detailed characterization of the proposed approach.

Capacity forecasting for resource allocation. We consider a capacity forecasting problem, along the lines of that introduced in Section 4.2.1. We consider different streaming mobile services, i.e., Facebook Live, Twitch and YouTube; in addition, we consider heterogeneous datacenters, each serving a geographically separated group of communication antennas, what will be later explained. Each combination of service and datacenter entails dissimilar patterns in the user demand for network resources [94]. We consider that the operator needs to predict the required capacity resources for the next 5 minutes, which lies within the standard operation time scale of modern network function orchestrators [101]. In this case, the performance metric matches the asymmetric expression of Section 4.2.1–use case (d). As mentioned there, recent studies in the

Datacenter D1 (values x10e-1)

Loss function	Facebook	Twitch	YoutTube
MSE	4.651±0.748	4.741±0.502	4.867±0.685
MSLE	4.503±0.697	4.405±0.3873	4.478±0.369
`MetaLoss`	**0.877±0.013**	**1.354±0.014**	**1.188±0.003**
Surrogate α-OMC	1.186±0.015	1.685± 0.018	1.563±0.023

Datacenter D2 (values x10e-1)

Loss function	Facebook	Twitch	YoutTube
MSE	4.095±0.790	4.265±0.625	4.709±0.319
MSLE	4.203± 0.906	4.385± 0.348	5.060±0.628
`MetaLoss`	**0.880±0.029**	**1.361±0.004**	**1.302±0.008**
Surrogate α-OMC	1.151±0.010	1.632± 0.016	1.497± 0.008

Datacenter D3 (values x10e-1)

Loss function	Facebook	Twitch	YoutTube
MSE	4.567±0.637	4.175±0.286	4.591±0.226
MSLE	4.469±0.961	4.119±0.304	5.185±0.607
`MetaLoss`	**1.102±0.031**	**1.376±0.010**	**1.480±0.016**
Surrogate α-OMC	1.365±0.0192	1.555±0.023	1.683±0.018

Datacenter D4 (values x10e-1)

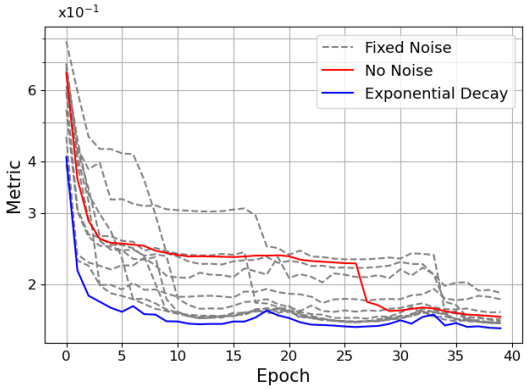
Loss function	Facebook	Twitch	YoutTube
MSE	4.751±0.631	4.492±0.523	4.620±0.540
MSLE	4.513 ±0.598	4.434±0.603	4.738±0.625
`MetaLoss`	**0.940±0.013**	**1.601±0.010**	**1.305±0.012**
Surrogate α-OMC	1.230±0.014	1.712± 0.016	1.483±0.017

Table 4.3. Performance results for three different services (Facebook, Twitch, and Youtube), for four different datacenters and four different loss functions.

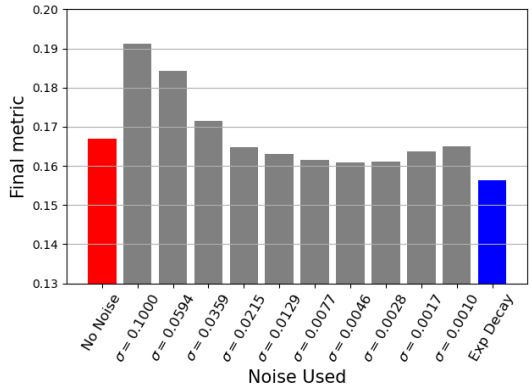
computer networking literature have proposed an expert-designed loss function, α-OMC, which can be considered a handcrafted, differentiable surrogate of the true performance metric [5]. We thus employ α-OMC as a state-of-the-art benchmark for comparison with `MetaLoss` in this use case.

4.3.1. Performance evaluation

We provide in Table 4.3 the performance results for the preemptive network resource allocation problem depicted above, summarizing the performance figures of all services in separate tables for each datacenters. Results are aligned with the ones presented in the manuscript, with an important gap between standard MSE and Mean Squared Logarithmic Error (MSLE) losses and more advanced techniques. In all cases, `MetaLoss` outperforms the expert-designed surrogate loss α-OMC by at least 15%, which corresponds to significant operating expense cuts in mobile network infrastructure management. The most important part is that it is doing so even without prior knowledge of the relationship between prediction and performance metric.



(a) Learning curve for different exploration noise's variances. Grey lines correspond to the fixed variances shown in Figure 4.3b.



(b) Loss metric results as function of the exploration noise variance. For the exponential decay case, the variance decreases as training epochs advance.

Figure 4.3. Impact of the variance of the exploration noise variance for accuracy and learning convergence.

4.3.2. Sensitivity to hyper-parameters

The model hyper-parameters correspond to both the learning rate and the loss exploration noise, and our implementation makes the model robust to those hyper-parameters. Experiments are run according to the first use case on capacity forecasting for resource allocation depicted earlier.

Learning rate We employ CLR, which is known to reduce the sensitivity of models to the choice of learning rate, and we recall that the ablation studies of Table 4.1 showed how CLR is beneficial in most settings, even if it is not the most impactful hyperparameter in terms of performance, the CLR helps, with the fact that the loss used by the predictor evolves during the training, to avoid the model of being stuck in local minima during the training phase, thus, makes the convergence of the model more consistent.

Loss exploration noise We make the exploration noise decay exponentially over time. This (i) avoids the need to fine-tune a fixed noise hyper-parameter, and (ii) improves the overall performance with respect to any fixed noise value. To demonstrate that this is the case, we consider a zero-mean Gaussian noise and analyze the performance for different values of the noise variance.

First, we consider the case with no noise (i.e., zero variance), which serves as a benchmark and shows the performance of a baseline model without this parameter. Later, we consider ten different variances, such that the standard deviation follows a

geometric progression from 0.001 to 0.1. These extreme values are selected as meaningful ranges after a detailed evaluation. Finally, we consider the exponential decay case in which the variance decreases as the number of epochs increases, borrowing the idea from the exploration-exploitation trade-off of reinforcement learning [102]. In particular, we consider that the standard deviation at epoch t is equal to $\frac{0.2}{(0.1t)^{\phi}}$, where ϕ is selected to ensure that the value at the first epoch is 0.1 and the value at the last epoch is 0.001, for a fair comparison with the fixed-variance cases.

Results are shown in Figure 4.3. In Figure 4.3b, we represent the final metric evaluated for the different noise variance, averaged over several realizations. We can observe how the results are quite sensitive to the variance of the noise, and adding noise can also be detrimental, whereas the exponential decay case obtains the best performance.

This last adaptive approach is beneficial not only because it achieves the best performance, but also because it allows us to avoid the hyper-parameter tuning required in view of the sensitiveness of the results to the variance. In Figure 4.3a, we show the learning curve (averaged over training realizations), where we only highlight the no-noise and the exponential decay curves for the ease of visualization. The exponential decay approach provides a much faster and more stable convergence. It has also been observed that it reduces the probability of obtaining a diverging solution.

4.3.3. Transfer learning

As the traffic serviced by each base station can differ substantially, each data center experiences quite diverse temporal dynamics of the demand [94]. Yet, the performance metric is the same for all datacenters, and the resource allocation predictors dedicated to each datacenter shall all be trained to optimize it.

This is an ideal setting where to test the capability of `MetaLoss` for transfer learning. Specifically, we evaluate how a predictor trained with the `MetaLoss` approach (i.e., by jointly training the predictor and the *loss shaper* block) performs with respect to a predictor that is trained with a not-varying *loss shaper* block that has been previously trained with the data from another datacenter. It is important to note that the *loss shaper* block can also be trained as well starting from an already trained configuration, in order to adapt to the predictor behavior.

Results are shown in Tables 4.4, 4.5 and 4.6. Each column indicates the datacenter in which the *loss shaper* block has been trained, whereas the rows indicate the datacenter for which that same *loss shaper* block, once trained for the column datacenter data, is used to train from scratch the predictor. Quite naturally, the values on the diagonal reflect the best performance, as they refer to cases where the full `MetaLoss` model trained on the traffic measured at a specific datacenter is employed to predict new demands in the same datacenter. However, by looking at values beyond the diagonal, where a loss function trained on traffic in one datacenter is used to train a new predictor for a different

datacenter, performance remains on the same level. Specifically, by comparing the values in Table 4.4 with those in Table 4.3, transfer learning via `MetaLoss` yields performances in new datacenters that are still better than or comparable with those obtained with the expert-designed α-OMC function in use case presented above in Section 4.2.1. Also, the same performance remains much better than that of legacy mismatched losses like MSE or MSLE.

Predictor trained at↓	Metric est. trained at D1	Metric est. trained at D2	Metric est. trained at D3	Metric est. trained at D4
D1	**0.877±0.013**	1.028±0.050	1.021±0.021	1.044±0.040
D2	1.002±0.048	**0.880±0.029**	1.001±0.036	1.035±0.047
D3	1.226±0.012	1.243±0.050	**1.102±0.031**	1.245±0.023
D4	1.088±0.037	1.081±0.028	1.076±0.022	**0.940±0.013**

Table 4.4. Transfer learning results, when run in different network datacenters serving diverse Facebook traffic demands. All values are $\times 10^{-1}$.

Predictor trained at↓	Metric est. trained at D1	Metric est. trained at D2	Metric est. trained at D3	Metric est. trained at D4
D1	**1.354±0.014**	1.522±0.012	1.497±0.012	1.530±0.016
D2	1.479±0.005	**1.361±0.004**	1.489±0.008	1.490±0.015
D3	1.487±0.014	1.487±0.017	**1.376±0.010**	1.491±0.024
D4	1.671±0.014	1.670±0.015	1.684±0.011	**1.601±0.010**

Table 4.5. Transfer learning results, when run in different network datacenters serving diverse Twitch traffic demands. All values are $\times 10^{-1}$.

Predictor trained at↓	Metric est. trained at D1	Metric est. trained at D2	Metric est. trained at D3	Metric est. trained at D4
D1	**1.188±0.003**	1.401±0.013	1.382±0.018	1.389±0.021
D2	1.433±0.006	**1.302±0.008**	1.423±0.013	1.443±0.017
D3	1.693±0.016	1.684±0.021	**1.480±0.016**	1.686±0.022
D4	1.396±0.013	1.345±0.022	1.388±0.020	**1.305±0.012**

Table 4.6. Transfer learning results, when run in different network datacenters serving diverse Youtube traffic demands. All values are $\times 10^{-1}$.

4.4. Real-world use cases

After analyzing and decomposing in detail the proposed solution, in this section we evaluate the performance of `MetaLoss` in realistic scenarios that are closer to real-world tasks, and which are considerably more demanding and complex. Specifically, we consider two practical case studies, detailed next. In both analyses, we employ real-world traffic measurements to ensure credibility of the results.

4.4.1. Power grid management

Even though this is not a networking problem, this use case is interesting as it demonstrates the generality of `MetaLoss`. Moreover this use case uses an additional input defined as $\boldsymbol{v}_i$ depicted in Chapter 3.

4.4.1.1. Problem definition

The complex field of power grid management and the study of smart grids are ruled by a significant number of diverse Key Performance Indicators (KPIs) [103–105]. One of the fundamental dimensions that define the performance in such scenarios is the reliability of the network, i.e., how often the network fails to provide the required power. Interestingly, the reliability in power management is not only measured by the frequency of power cuts due to the under-provisioning but also by the duration of these cuts [103]. The smart grid manager is especially interested in preventing under-estimations, as in the previous use case of network resource allocation; however, the relevant performance metric also incorporates memory about past outages. Specifically, in the presence of underestimation, if the previous forecast was also underestimated, the current cost is added to the precedent cost in a recursive manner. Thus, the metric depends on the previous state of the system and involves discrete (thus discontinuous) jumps when consecutive outages occur. Manually designing a surrogate version of the multi-dimensional loss function matching the metric above is not trivial, and we are not aware of any such attempt in the literature. `MetaLoss` removes this barrier by learning the correct loss in an automated way.

We employ α-OMC as a state-of-the-art loss benchmark. Although initially crafted for network-related applications, its underlying principle remains consistent in our context: tolerating a minimal level of surplus to minimize disruptions. The temporal aspect of the metric is however not possible to adapt in this function.

4.4.1.2. Results

We summarize the evaluation results in Table 4.7. We observe that `MetaLoss` significantly outperforms traditional training methods based on standard loss functions. While the advantage over generic loss roots in the loss-metric mismatch and is easily understood, the improvement with respect to α-OMC is explained as `MetaLoss` effectively adapts to varying demands. The model learns a loss that compensates for different accuracies in predicting power consumption values of diverse magnitudes.

In this first use case, we are particularly interested in presenting how `MetaLoss` can adapt to a recursive cost relationship. Figure 4.4 shows the learned loss function. The result shows the potential of the proposed approach to characterize unknown and non-trivial loss functions for regression problems. More details on this aspect are present in

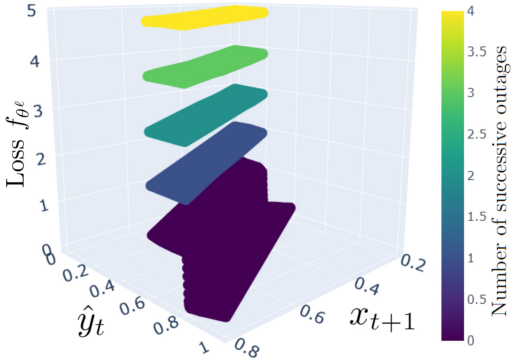


Figure 4.4. A 3-dimensional representation of the 4-dimensional learned loss.

Table 4.7. Normalized performance results for use case I for four solutions: A predictor DNN trained with (*i*) MSE and (*ii*) MSLE loss functions, the proposed **`MetaLoss`** model, and an expert-designed surrogate loss from [5].

Loss function	Metric
MSE	1.804±0.262
MSLE	1.813±0.253
`MetaLoss`	**0.056±0.002**
α-OMC	0.114±0.007

Chapter 6. In this particular case, **`MetaLoss`** learns a differentiable approximation of a step-wise discrete function.

4.4.2. Reserving virtual machines at a core network datacenter

In this use case, we address the efficient management of data center resources, which is critical for minimizing operational costs and ensuring optimal performance. By accurately predicting the number of Virtual Machines (VMs) required for each service, the data center can avoid over-provisioning (which leads to unnecessary costs) and under-provisioning (which leads to poor service quality). This balance is crucial for maintaining cost-effective operations and high-quality service delivery, particularly in environments with high and variable demand, such as video streaming services.

4.4.2.1. Problem definition

We consider a core network datacenter setting, where we assume that each video streaming service is assigned a dedicated Network Slice Subnet Instance (NSSI). The machine-translated intent involves that the Virtual Infrastructure Manager (VIM) responsible for controlling the datacenter resources must predict the number of VMs that need to be allocated in advance to each NSSI, so as to run the Virtual Network Functions (VNFs) required to serve the demand generated by the corresponding mobile service. Clearly, every VM has an operating cost (e.g., due to power consumption) so it is desirable that only the strictly necessary set of VMs is reserved for each NSSI.

Clearly, we do not have access to information about the actual operating costs of the datacenter, or about the local VM management strategies. This forces us to emulate that part of the system behavior to perform our experiments. To that end, we proceed as follows: (*i*) we define a credible expression $f_{\mathcal{M}}$ for the target management objective; (*ii*) we use $f_{\mathcal{M}}$ to generate observations $\mathcal{M}_{t+1}$ in response to predictions $\hat{y}_t$; and, (*iii*) we use observations $\mathcal{M}_{t+1}$ as ground-truth measurements to train and test **`MetaLoss`** and

benchmark models. A very important remark is that $f_{\mathcal{M}}$ is only an artifice we adopt to generate measurement-like data, and it is unknown to `MetaLoss`, which has only access to the observations.

Formally, let all VM have equivalent performance, so that each has operating cost κ, and can serve a maximum volume of traffic C_σ that may depend on the slice σ. Also, $\eta \gg \kappa$ is a high penalty triggered in case insufficient VMs are allocated in advance. The observations are computed as:

$$\mathcal{M}_{t+1} = f_{\mathcal{M}}(\hat{y}_t, \boldsymbol{x}_{t+1}) = \eta \cdot \mathcal{R}\left(x_{t+1} - C_\sigma \cdot |\lfloor \hat{y}_t \rfloor|\right) + \ \kappa \cdot \mathcal{R}\left(C_\sigma \cdot |\lfloor \hat{y}_t \rfloor| - x_{t+1}\right). \tag{4.4}$$

This expression basically forces a high handicap if insufficient VMs are allocated, and a milder one for VM overprovisioning.

The VM orchestration takes place at every 5 minutes, which is thus the forecast horizon of the predictor. We feed `MetaLoss` with past traffic information $\boldsymbol{x}_t = \{x_{t-N}, \ldots, x_t\}$, with $N = 6$, and let it learn to produce a forecast $\hat{y}_t$ that minimizes (4.4), from observations $\mathcal{M}_{t+1}$. We remark that the objective is not linear nor differentiable over the whole domain, as it includes floor and absolute value operators.

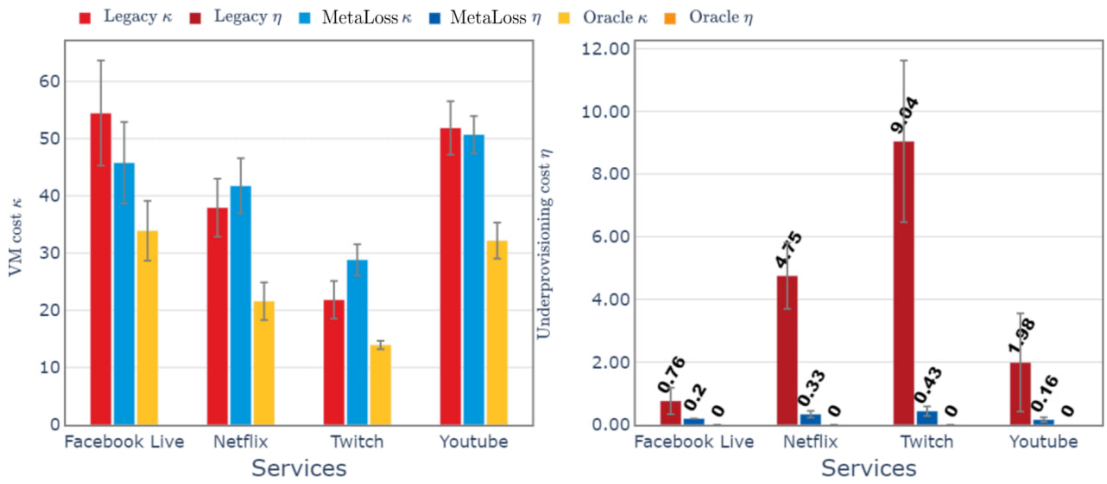
4.4.2.2. Results

We consider two benchmarks for comparative analysis.

- An **Oracle** predictor, which has perfect knowledge of the future, and always allocates the optimal minimum number of VMs to serve the upcoming demand without providing any Service Level Agreement (SLA) violation.

- A **Legacy** traffic forecasting model, implemented as the predictor part of `MetaLoss`, and trained to minimize an MSE loss. As discussed in Section 2, this model requires an additional downstream block in charge of taking the management decision. In this use case, we manually design a VM allocation algorithm that (*i*) applies an overprovisioning factor to the predicted traffic so as to avoid expensive underdimensioning, and (*ii*) uses C_σ to compute the number of VMs to serve the inflated demand. Upon extensive tests, we set the overprovisioning factor to 1.6, which enforces 60% safety margin.

Figure 4.5 summarizes the performance, when the C_σ and κ are set to 10% and 1% of η. Even when fine-tuned, a decision making policy based on a **Legacy** prediction is substantially less efficient than `MetaLoss`. Our model allocates around the same VMs as **Legacy**, but with an extremely limited underprovisioning that is one or two orders of magnitude lower than that of **Legacy**. The extra VMs allocated for Netflix and Twitch traffic comes with the fact that those traffics are less regular and more spiky than Youtube and Facebook Live, thus, `MetaLoss` tends to increase the safety margin for



(a) VMs reserved **(b)** Underprovisioning time (%)

Figure 4.5. VM reservation for diverse slices. (a) Reserved VMs. (b) Fraction of time when the slice demand cannot be served.

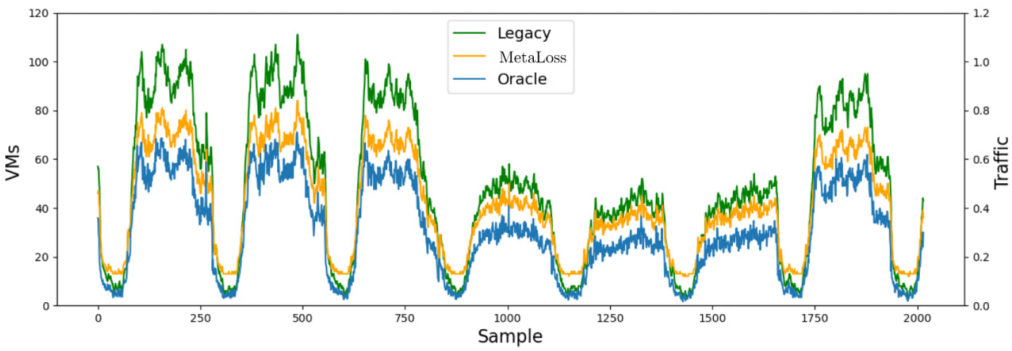


Figure 4.6. Example of VM reservation for the Facebook slice.

such traffic. This comes from the fact that the *loss-shaper* and the *predictor* are trained together and both learn from the other as described in Section 3.6.3. The reason is illustrated in Figure 4.6, for one sample slice: `MetaLoss` anticipates a constant reasonable overdimensioning at all times; instead, the solution based on the **Legacy** predictor tends to allocate excess VMs during the high-traffic daylight hours, and does not leave a wide enough safety margin overnight, when it allocates less VMs than **Oracle**, hence not servicing part of the demand. Instead, `MetaLoss` learns that a static overprovisioning factor is not a good strategy to cope with the inherent forecast inaccuracy, and automatically identifies a better loss to minimize the (unknown) expression in (4.4). By doing so, our model performs in fact fairly close to the **Oracle**.

4.4.3. Minimizing video streaming OPEX at the Edge

4.4.3.1. Problem definition

We now move to a mobile Edge environment, where a set of computational facilities, each serving between 70 and 130 base stations, are located close to the radio access. We assume again that of the four video streaming services presented before has a dedicated NSSI at the Edge facilities. Here, the management intent aims at minimizing the monetary OPerating EXpenses (OPEX) associated to running the video streaming slices at the network Edge. This maps to an IBN objective of periodically and preemptively rescaling the compute resources assigned to each NSSI in a facility, to smoothly run the needed VNFs.

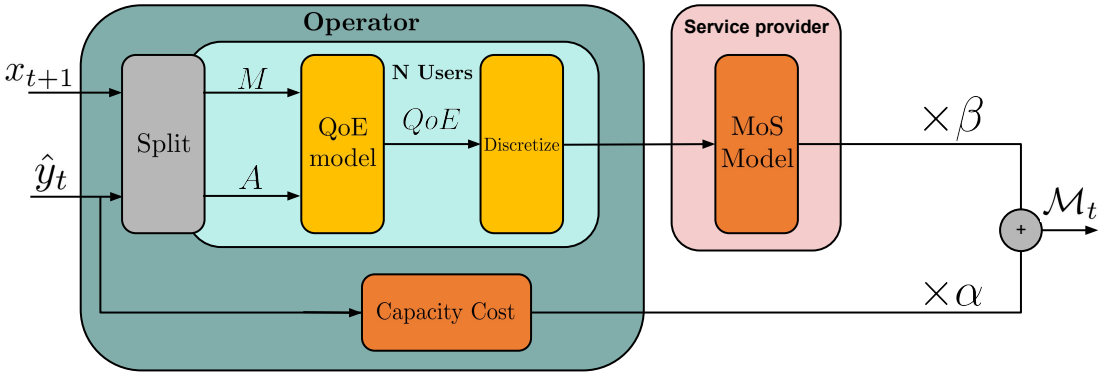
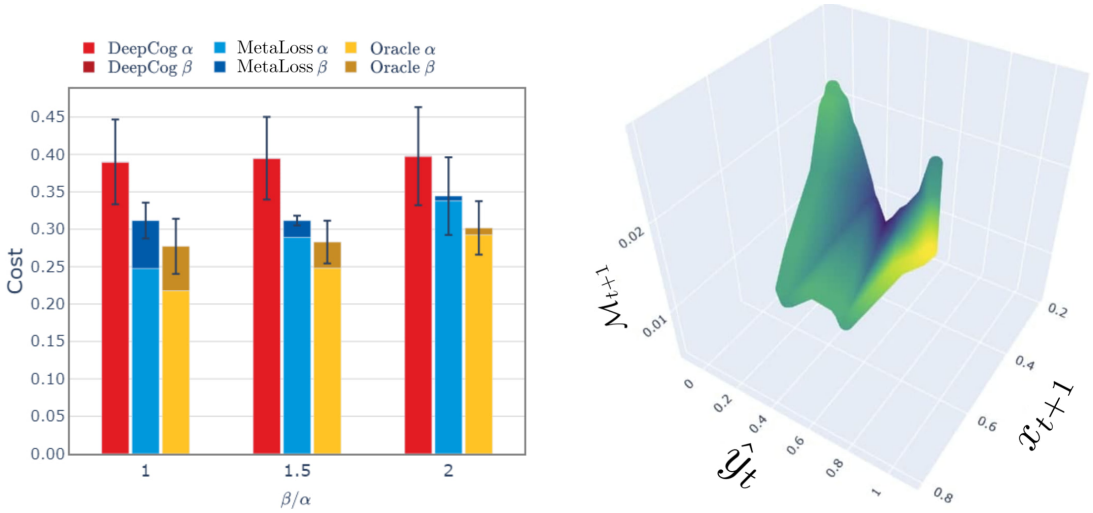


Figure 4.7. Emulation pipeline for the monetary OPEX.

We have to emulate the ground-truth OPEX, by developing a sensible model that relates OPEX to the system variables. Here, we go a step further in terms of complexity, and, rather than defining a single expression for $f_{\mathcal{M}}$, we employ a whole pipeline to mimic the objective. Figure 4.7 offers a high-level view of the pipeline. First, the slice-level allocated computing capacity $\hat{y}_t$ and the future traffic to be served x_{t+1} are split across users according to a probability distribution that describes the imbalance of traffic generated by different users. The per-user maximum requested (M, from x_{t+1}) and available (A, from $\hat{y}_t$) compute powers are fed to an empirical non-linear model that maps such metrics into a user-level video streaming Quality of Experience (QoE) [106]. The QoE value is then discretized in line with real-world QoE metrics, and passed to a module that converts the QoE into a Mean Opinion Score (MOS). Based on the MOSs of the users, the slice tenant can communicate to the operator if the SLA has been violated, which entails a monetary fee β per violation. This cost is summed to that generated by the need to reserve the compute resources d_t to accommodate the slice traffic, derived in the bottom block, at a price α per capacity unit.

Interestingly, not only the expression of $f_{\mathcal{M}}$ is now much more involved than before,



(a) Overall performance

(b) Learned loss for a ratio $\beta/\alpha = 1$

Figure 4.8. OPEX performance in the Facebook Live slice case. (a) Overall cost. (b) Loss function learned by `MetaLoss`.

but it is also not observable by the operator in practical cases. Indeed, the VNF Manager (VNFM) must take the anticipatory decision on the computing capacity allocation has, in the best case, partial visibility of how the computational capacity allocated to an NSSI results into its economic performance. Typically, the VNFM only has data about the current allocated resources and slice traffic demand from the Virtual Infrastructure Manager VIM. In the best case, it may consume QoE information provided by the Network Data Analytics Function (NWDAF) [107] that interfaces with Application Functions (AF), but the MOS part of the pipeline remains unknown. Ultimately, this use case sets forth a scenario where a network manager would not have the necessary system knowledge to manually design a solution to the problem.

4.4.3.2. Results

We let `MetaLoss` learn the whole pipeline above in a setting where the compute capacity reconfiguration occurs every 5 minutes. We compare it against two benchmarks, as follows.

- An **Oracle** predictor that knows the future demand and the full $f_{\mathcal{M}}$ pipeline, and returns an optimal allocation.

- **DeepCog**, a state-of-the-art capacity predictor [5] that we configure to cope with the problem in the best way possible, by setting its loss function parameter to β/α.

To avoid repetitions, only results for the Facebook Live slice are shown, in Figure 4.8a, however, performances are homogeneous across services. All costs are relative to that of the minimum static capacity allocation that always services the full demand, and are shown for different β/α ratios. Despite the fact that we feed it with information about the true α and β values, **DeepCog** is still constrained by its unflexible loss function. Instead, `MetaLoss` can autonomously learn a much better loss, which results in a reduced cost closer to the **Oracle** one. Figure 6.2a offers a glance at the fairly complex loss function captured by the loss shaper of `MetaLoss`: even if full knowledge of the pipeline in Figure 4.7 were available, designing manually devise this shape would be an exacting task. Our approach effectively automates such design, which lays an important stone in the path to full-fledged IBN.

4.5. Summary

Throughout this chapter, we have elucidated the efficacy of `MetaLoss` in addressing the challenge of loss-metric mismatch. By conducting multiple controlled experiments and analyzing real-world use cases, `MetaLoss` has demonstrated its superiority over existing state-of-the-art models in applicable scenarios. Furthermore, `MetaLoss` offers additional flexibility and freedom in DNN model-based implementation, thereby enhancing their adaptability to diverse contexts and its potential to advance the field of networking management.

5 Global optimization for multiple predictors scenarios

In this chapter, we focus on enhancing the model outlined in the preceding Chapter 4 to tackle the resolution of common objective problems characterized by intertwined predictions. This endeavor stands as the second primary objective of this thesis, aimed at fulfilling the achieving goals of IBN.

The first section of this chapter delves into showcasing the efficiency of this novel architecture. This is achieved through controlled experiments, wherein our enhanced model is tested against alternative methodologies striving to achieve similar objectives. By comparing the performance metrics, we aim to underscore the superiority of our proposed approach. To highlight the significance of this enhancement, the use case is described as a simple toy example, though, impossible to achieve with actual methods. Through this illustrative scenario, we unveil why accommodating intertwined predictions is a crucial feature. We elucidate that relying solely on the architecture delineated in the previous Chapter 4 renders this task unfeasible.

Finally, the narrative extends to achieve real-world use cases that, thus far, have remained elusive for traditional data-driven approaches. Through detailed analyses and case studies, we aim to demonstrate how this enhanced model transcends existing limitations, thereby enabling the resolution of challenges that IBN goal aims to achieve.

In order to achieve the different experiments described below, the architecture used was the following: The regressor block is implemented with k 3-layer parallel LSTMs block, k being the number of IR used for the problem, using respectively (128, 128, 64) followed by 4 fully connected layers of respectively (256, 256, 128, 64) neurons for the assembler. The activation function used in every layer is the ReLu one. The loss shaper INR has 5 Siren layers, with respectively (256, 256, 256, 256, 128) units in it using as depicted in 3.4. This more complex architecture is explained by the fact that the use cases in this chapter are more complex.

The problem of solving common objectives across different actions is not new, in fact it represents a famous area of research in Game Theory. The most famous example to illustrate this is the Prisoner's dilemma. It is a classic concept, used to illustrate

situations where individuals acting in their own self-interest can lead to suboptimal outcomes for all involved. If we imagine that two criminals are arrested and placed in separate interrogation rooms. The police offer each prisoner a deal:

- If Prisoner A confesses and testifies against Prisoner B, while B remains silent, A will go free and B will receive a harsh sentence (10 years).
- If both prisoners confess, they both receive moderate sentences (5 years each).
- If both prisoners remain silent, lacking sufficient evidence, they will both be convicted of a lesser charge and receive light sentences (1 year each).

Payoff Matrix	**Prisoner B Stays Silent**	**Prisoner B Confesses**
Prisoner A Stays Silent	A: 1 year, B: 1 year	A: 10 years, B: 0 years
Prisoner A Confesses	A: 0 years, B: 10 years	A: 5 years, B: 5 years

Table 5.1. Prisoner's Dilemma Payoff Matrix.

The dilemma arises because while both prisoners would be better off cooperating (remaining silent), the temptation to defect (confess) is strong due to the potential for a better individual outcome as seen in Table 5.1. However, this leads to a situation where both end up worse off than if they had cooperated, i.e., the Nash Equilibrium.

The focus here is similar but rather than discrete problems as the prisoner dilemma we instead focus on continuous type problems, i.e., continuous games. The theory behind it is however very similar, the fact that individual optimum does not imply the optimum for an individual. This is a very important research topic as that kind of game, trying to achieve a common objective exists in many different areas, from environmental studies, animal behaviors, politics, and obviously networking management.

5.1. Controlled experiment

We provide a comprehensive analysis of the performance of the proposed approach for several controlled experiments, with the aim of describing in detail the contributions of each of its components and its overall advantage over previous proposals for loss meta-learning network management applications.

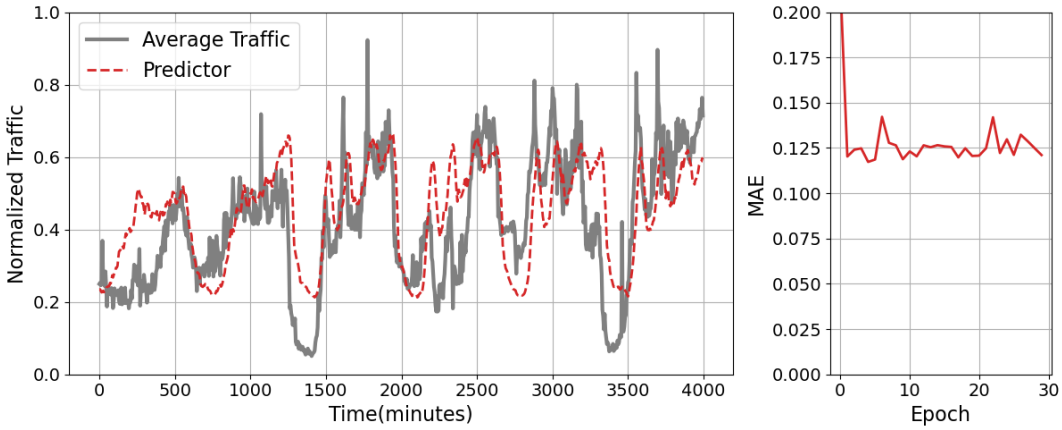
We first focus on proving that the proposed structure is indeed required and crucial for the model. For that, we compare the performance of the architecture presented in Chapter 3 against three alternative architectures in a simple toy example. Second, we provide an ablation study and comparison with state-of-the-art approaches for several loss functions with different levels of complexity, so as to prove the advantage of loss meta-learning, and we do so for single-variable problems for the sake of comprehension. After that, we provide results on generic time-series forecasting problems and, finally, we provide a detailed analysis for realistic yet simple use cases.

5.1.1. Toy example

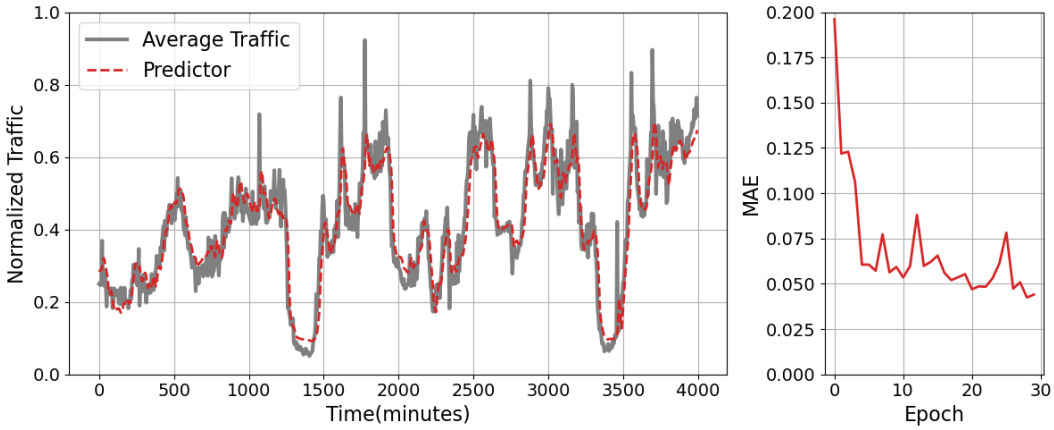
Intertwined predictions are problematic for current solutions for loss meta-learning regressors. We demonstrate the issue via a toy example, where two predictors a and b are fed with past values of traffic time series $\boldsymbol{x}^a$ and $\boldsymbol{x}^b$ up to time t, and they have to forecast the mean of the two next traffic values, i.e., $(x^a_{t+1} + x^b_{t+1})/2$.

While fictional, this is a naive case where the two predictions are intertwined, as their output depends on both flows $\boldsymbol{x}^a$ and $\boldsymbol{x}^b$. Abiding by the principles above, the expression of the prediction target is not known a priori, and the model must (i) meta-learn that the averaging function is the right loss for the task, and (ii) use it to train the predictors.

Figure 5.1a shows the performance in the simple task above of a predictor implementing the only architecture needed to achieve the loss metric mismatch problem depicted in the previous Chapter 4.



(a) Standard model



(b) `MetaLoss`

Figure 5.1. Performance of (a) a Standard forecasting model and (b) **`MetaLoss`** in the toy use case of mean value forecasting. Left: predictions and target average. Right: forecasting error over train epochs.

The forecast is clearly misaligned with respect to the mean traffic target (left), and results in a poorly converged error (right). The reasons for such poor performance are detailed in Section 5.1.2.

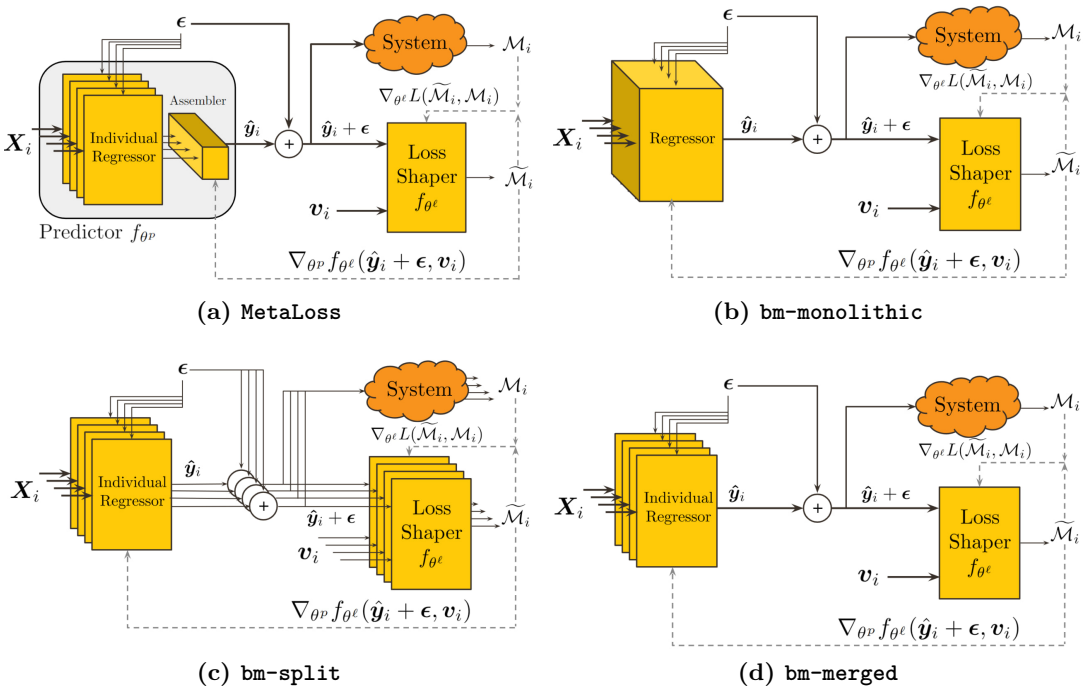


Figure 5.2. Detailed architectures of the (a) `MetaLoss`, (b) `bm-monolithic`, (c) `bm-split` and (d) `bm-merged` benchmarks.

5.1.2. Ablation study of `MetaLoss` logical architecture

First, we compare the proposed architecture with three alternative architectures, illustrated in Figure 5.2 and defined as:

- The `bm-monolithic` approach, presented in Figure 5.2b, implements the predictor with a single block composed of fully connected LSTM and MLP layers, in such way that it receives the whole input $\boldsymbol{X}_i$ to also output all forecasts $\hat{\boldsymbol{y}}_i$. Hence, this model is the most naive generalization from a single-input single-output predictor.
- The `bm-split` model applies a parallelization of the multi-dimensional forecasting task along single input-output tuples, decomposing the problem into n_{in} single-variable problems, i.e., independent prediction instances, as in Figure 5.2c.
- The `bm-merged` model is an architecture compromising between the previous solutions: it is composed of multiple parallel predictors that feed a single performance cost estimator during training, as illustrated in Figure 5.2d.

The aforementioned benchmarks represent in some way simplified versions of **MetaLoss**: while **bm-monolithic** merges the internal IPU-based architecture of the predictor in Figure 3.2 into gathered LSTM layers, **bm-merged** withdraws the other key element, the aggregator. Thus, these approaches allow us to deliver an ablation study for the proposed model. We also remark that **bm-split** and **bm-merged** implicitly assume that $n_{in} = n_{out}$ and that $\hat{\boldsymbol{y}}_i$ can be inferred from $\boldsymbol{X}_t$ only, and consequently the generality of both is notably reduced with respect to **MetaLoss**, which removes such assumptions.

We also compare the whole architecture against versions of it that miss some of the key components. Namely:

- **ReLu** identifies the impact of considering the ReLu activation function instead of the SiREN activation function for the *loss shaper*.
- The **Noiseless** design, which consists in setting $\boldsymbol{\epsilon} = 0$.
- The **fixed-Noise** design, in which the noise variance Σ is fixed throughout the training. This variance is set to 3% of the average value of the data, which is picked as giving the best results with a fixed amount of noise.
- The **fixed-LR**, where the learning rate is fixed.
- The **exp-LR**, with exponential decay learning rate.
- Finally, **Grabocka** is an approach inspired from [60], using a similar training approach as the bilevel programming optimization of the model parameters adopted by [60], where the loss model and predictor are alternatively trained for some epochs, instead of being trained simultaneously in the same gradient descent iteration as in **MetaLoss**.

We use this simple yet illustrative toy example depicted above where the unknown objective is producing an average of all inputs in the next time slot, to test the four models. Formally, applying the notation introduced in Section 3.4, the predictor shall output a scalar[1] $\hat{y}_t = \frac{1}{n_{in}} \sum_{i=1}^{n_{in}} x_{t+1}^{(i)}$. This is an uninvolved problem with intertwined predictions,[2] as the output depends on predictions for all inputs. This use case allows for a straightforward way to experiment with the different models and is not intended to model any practical network management task (which will be analyzed in Section 5.2).

We summarize the results in Table 5.2. **bm-split** and **bm-merged** both suffer from the decoupling of each output $\hat{y}_t^{(i)}$ from all inputs $\boldsymbol{X}_t$, and **MetaLoss** reduces the error generated by these approaches between 75% and 90%. This proves, in particular, that the

[1]The prediction of **bm-split** and **bm-merged** is n_{in} dimensional, hence each $\hat{y}_i^{(i)}$ shall match the same expression in the main text.

[2]We pay attention to input uncorrelated time series so that their mean cannot be simplified as $\hat{y}_{t+1} = K\boldsymbol{X}_t$, which would artificially ease the task.

n_{in}	MetaLoss	ReLu	bm-merged	bm-split	bm-monolithic
2	**0.018±0.014**	0.024±0.012	0.101±0.081	0.119±0.094	0.067±0.042
4	**0.020±0.012**	0.023±0.010	0.095±0.073	0.116±0.089	0.047±0.026
6	**0.018±0.010**	0.021±0.008	0.098±0.076	0.090±0.073	0.043±0.026
12	**0.010±0.006**	0.012±0.003	0.093±0.079	0.088±0.096	0.038±0.024
n_{in}	Noiseless	fixed-Noise	fixed-LR	exp-LR	Grabocka
2	0.025±0.010	0.024±0.019	0.028±0.013	0.024±0.013	0.029±0.015
4	0.026±0.010	0.025±0.017	0.026±0.013	0.022±0.011	0.026±0.012
6	0.023±0.009	0.022±0.012	0.024±0.009	0.022±0.008	0.022±0.011
12	0.015±0.003	0.013±0.006	0.013±0.004	0.013±0.004	0.015±0.006

Table 5.2. Results for the controlled experiment in terms of average MAE, for our model and all benchmarks. The goal is to learn that the loss function is the MAE of the estimate with respect to $\hat{y}_t = \frac{1}{n_{in}} \sum_{i=1}^{n_{in}} x_{t+1}^{(i)}$.

inner structure of the predictor, with the separated IR that are fed into the aggregator, is crucial to combine the contribution of each variable, making sure that each IR learns its (potentially different) role towards achieving the objective and ensuring that the task is learned. Additionally, the poor performance of the **bm-monolithic** model, whose MAE more than doubles the one attained by **MetaLoss**, showcases the importance of parallel individual IR to manage independently the input time series that are not naively correlated, since **bm-monolithic** treats all the input time series as a single vector, reducing the performance of the LSTM layers. Finally, the simultaneous training of metric estimator and predictor through the same backpropagation iteration improves the performance up to 25%, as observed by comparing **MetaLoss** against **Grabocka**. Overall, **MetaLoss** yields largely superior performance over the state-of-the-art and competitor architectures even in a simple (for intertwined variables) toy example.

5.2. Real-world use cases

In this section we evaluate the performance of **MetaLoss** in very realistic scenarios that are real-world problems, and which are considerably more demanding and complex. Specifically, we consider two practical case studies, detailed next. In both analyses, we employ real-world traffic measurements to ensure credibility of the results.

5.2.1. Overbooking of network slices

Multi-tenancy is a defining paradigm in 5G systems, which are expected to fully support network slicing providing flexible policing and strong guarantees to heterogeneous services run by different tenants. The anticipatory allocation of isolated and customized resources to individual slices is a key functionality to ensure that the Quality of Experience (QoE) that each tenant request is met. This functionality also offers new opportunities for operators to optimize their revenues: for instance, *slice overbooking* takes advantage of

slice demand multiplexing to reduce resource utilization while fulfilling (a priori unknown) QoE [35]. Inspired by the overbooking approach, we consider a case where anticipatory admission control (AC) and resource reservation (RR) of network slices must be steered to optimize QoE-based revenues for the network operator.

5.2.1.1. MANO objective

We consider that the QoE level determines certain monetary revenues and costs for the network operator, where satisfying the QoE level agreed between the tenant and the operator results in a revenue for the operator, whereas failing to provide the required QoE incurs in costs and fees for the operator. A Service-Level Agreements (SLA) sets the performance target of a given network slice in terms of the requested end-user QoE, and specifies the amount offered by the tenant for implementing the slice. The SLA also defines how such an amount is reduced for lower QoE levels, including an economic fee for the operator in case the QoE falls below a minimum acceptable threshold.

Formally, let us denote the traffic demand generated by slice $s \in \mathcal{S}$ at time t as $d_s(t)$ and its peak traffic demand as D_s, and let $c_s(t)$ be the amount of resources that the operator allocates to such a slice. We further denote the normalized demand and allocation as $\bar{d}_s(t) \triangleq \frac{d_s(t)}{D_s}$ and $\bar{c}_s(t) \triangleq \frac{c_s(t)}{D_s}$, respectively.

In case the slice is accepted, the tenant pays for the implementation of the slice an amount $R_s(t) = \gamma d_s(t)$, which is proportional to the traffic to be accommodated by some constant γ factor (in \$/byte). However, the amount $R_s(t)$ is paid in full only if a target end-user QoE level is attained; if instead the end users suffer from lower QoE, the operator incurs into a proportionally reduced payment by the tenant.

In our experiments, we assume that the QoE requirement is met if $\delta_s(t) \triangleq \bar{d}_s(t) - \bar{c}_s(t) \leq 0$, i.e., enough resources are reserved by the operator to serve the whole traffic demand of the slice. Otherwise, the QoE (hence the revenues for the operator) drop according to a sigmoid function of $\delta_s(t)$, as recommended by standard models [108] in revenue management and prospect theory [109]. Below a threshold QoE, the sigmoid becomes negative, meaning that the operator receives no payment but must start paying a fee to the tenant. If then the QoE further falls below a minimum acceptable value, the fee jumps to a large amount $K_s(t) = \beta R_s(t)$ that is proportional to the requested resources for the slice. The resulting QoE-based cost of the accepted slice for the operator is

$$QoE(\delta_s(t)) \triangleq \Big(\tfrac{K_s(t)}{2}\big(\tanh(\alpha\delta_s(t) + \beta) + 1\big) - R_s(t)\Big), \tag{5.1}$$

where α, β are constants that determine how the sigmoid function exactly scales with the underprovisioning $\delta_S(t)$. Besides the revenue from the SLA, the total economic advantage of the operator is affected by two other elements. On the one hand, the overprovisioning of the reserved resources determines an increase of operating expenses (OPEX): therefore,

whenever $\delta_s(t) < 0$, the operator incurs an OPEX penalty $-\gamma\delta_s(t)$ that grows linearly with the quantity of unnecessarily reserved resources. On the other hand, committing to allocate more resources than those available in the network infrastructure leads to a global performance drop and possible service outages, with a huge cost M in terms of, e.g., customer churn.

The final performance cost associated to the slice MANO is

$$\begin{aligned}\mathcal{M}(t) \;=\; & \sum_{s\in\mathcal{S}} \Big(\mathbb{1}(\bar{c}_s(t)) \cdot QoE(\delta_s(t)) - \mathbb{1}(-\delta_s(t)) \cdot \gamma\delta_s(t)\Big) \\ & + M \cdot \mathbb{1}\Big(\big(\sum_{s\in\mathcal{S}} c_s(t)\big) - C\Big), \end{aligned} \tag{5.2}$$

where C is the total capacity of the system, and $\mathbb{1}(\cdot)$ is the step function that takes value one if the argument is less than 0, and value one otherwise: thus, $\mathbb{1}\left(\bar{c}_s(t)\right)$ takes value one only if the slice is admitted and resources are allocated to it. The second term seeks to keep $\delta_s(t)$ as close to zero as possible where the last term represents a high penalty (M) in case that the total amount of allocated resources exceeds the total capacity C.

5.2.1.2. Solution based on `MetaLoss`

A key observation is that, in practice, the network operator cannot analytically know the relation between the slice resource allocation and the QoE of end users of the specific tenant. This is a very complex function that depends on many operational parameters and whose characterization depends on application-level data (e.g., user feedback) that the operator does not have. Therefore, (5.1) is not known a priori, in both its whole formulation and specific parametrization of α and β. In turn, this makes it impossible for the operator to model the whole performance cost in (5.2) in advance.

In this scenario, the operator can apply **`MetaLoss`** to address the MANO task of deciding (i) which slices to accept and (ii) how many resources to reserve to accepted slices at the start of each orchestration interval. Indeed, once deployed in the system, **`MetaLoss`** can learn the expression of the performance cost in (5.2), and optimize the prediction of $\bar{c}_s(t)$ to minimize such cost. Note that the value of $\bar{c}_s(t)$ determines both the slice admission control, via $\mathbb{1}(\bar{c}_s(t))$, and the allocation of resources, which match its value if positive.

The detailed integration of **`MetaLoss`** in the network architecture for this use case is outlined in Figure 5.3. Our model sits in the Network Function Virtualization Orchestrator (NFVO) of the MANO framework, as proposed by current standards [11]. It can then leverage the Management Data Analytics Function (MDAF) [110] to access information exposed via the Application Function (AF) [111]: based on the operator-tenant agreement, such information can include live QoE measurements and slice revenue data, which let **`MetaLoss`** observe the results of its decisions on the QoE and costs. Our

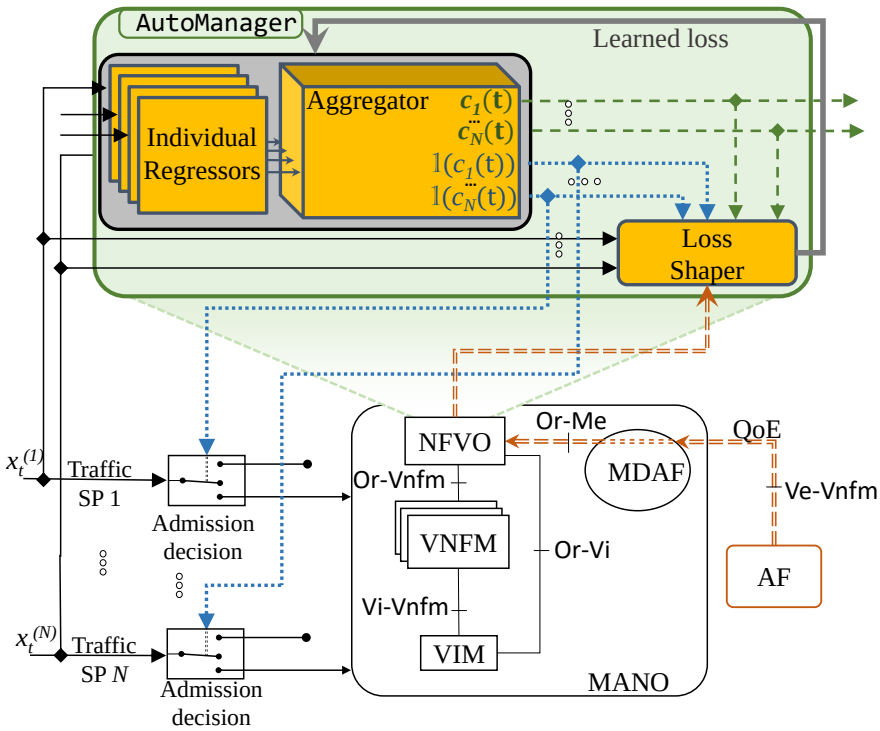


Figure 5.3. Implementation of `MetaLoss` in the network architecture.

model is then in a position to learn from experience the overall cost in (5.2), and forecast $\bar{c}_s(t), \forall s \in \mathcal{S}$ to minimize it.

The meta-learning task is in fact not trivial. We provide an intuition of the complexity of the problem in Figure 5.4, which illustrates the expression in (5.2), in the naive case of a single slice. Despite the fact that we are considering the simplest version possible of the cost, and the only one that can be represented in only three dimensions, the shape of the relationship between the MANO objective and the prediction is tangled and also highly sensitive to the value of the input traffic demand. Scaling to realistic multidimensional versions of the cost thus implies very strong meta-learning capabilities.

5.2.1.3. Benchmark

For the best of our knowledge, there does not exist any automated solution for this problem. Because of that, we compare `MetaLoss` against a solution that follows the legacy approach presented in Figure 2.1 depicted in the state-of-the-art chapter. Specifically, we consider that each slice $s \in \mathcal{S}$ is handled separately by one predictor with a tailored loss function, which is responsible of forecasting the resources to be reserved for slice s (denoted by $c'_s(t)$) if the slice is accepted. The admission control is instead handled apart, and formulated as an optimization problem solved starting from all $c'_s(t)$ predictions. The optimization program tries to find the best binary admission control variables $x_s(t)$, which

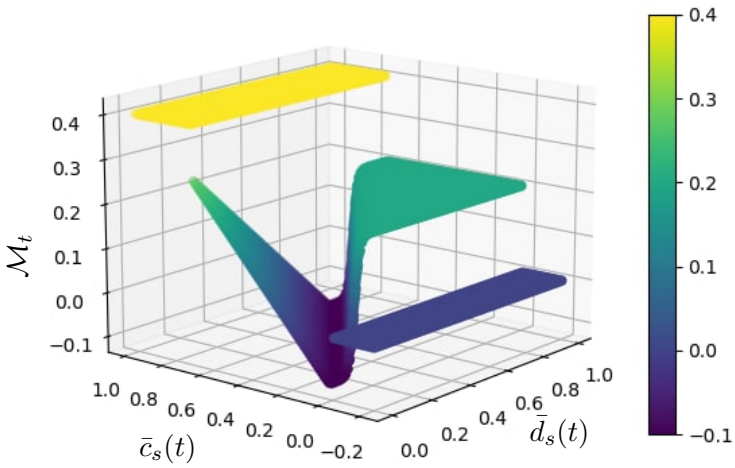


Figure 5.4. Sample of the final performance cost, when a single slice $s = \mathcal{S}$ is present in the network.

take value one if slice s is accepted. Formally,

$$\max_{x_s(t)} \sum\nolimits_{s\in\mathcal{S}} R_s(t)x_s(t) \quad \text{s.t.} \sum\nolimits_{s\in\mathcal{S}} c'_s(t)x_s(t) \leq C. \tag{5.3}$$

This is in fact the well-known and NP-hard Knapsack problem (KP). Hence, we name the benchmark **`knapsack`**. The individual predictors of the benchmark are trained in a similar way as for the **`MetaLoss`** implementation, with the difference that it only receives data for the corresponding slice. Therefore, the learning block of the benchmark only takes care of the resource prediction, but not the admission control, and it is thus less general and flexible than the solution based on **`MetaLoss`**.

5.2.1.4. Results

We make use of the traffic data from up to 12 different applications in `TrafficApp` dataset. We assume that each application is requesting a dedicated slice s.

We first provide a qualitative illustration of the results in Figure 5.5. **`MetaLoss`** (top plot) learns to keep the allocated capacity below the maximum capacity of the edge datacenter (gold) even if the actual total demand (dotted red) exceeds such value. When the maximum capacity is above the total traffic, **`MetaLoss`** offers a very precise forecast of the resources needed to accommodate the demand. The bottom plot focuses instead on two slices (gray and red), showing their requested capacity (solid) and the allocated resources by our solution (dotted). During periods of low demand, after hour 72, both requests are precisely predicted and fully served. When capacity is scarce, before hour 72, the gray slice is often rejected (with zero reserved resources). Also, the resources allocated to the red slice do not match anymore the request, but are slightly lower: here, **`MetaLoss`**

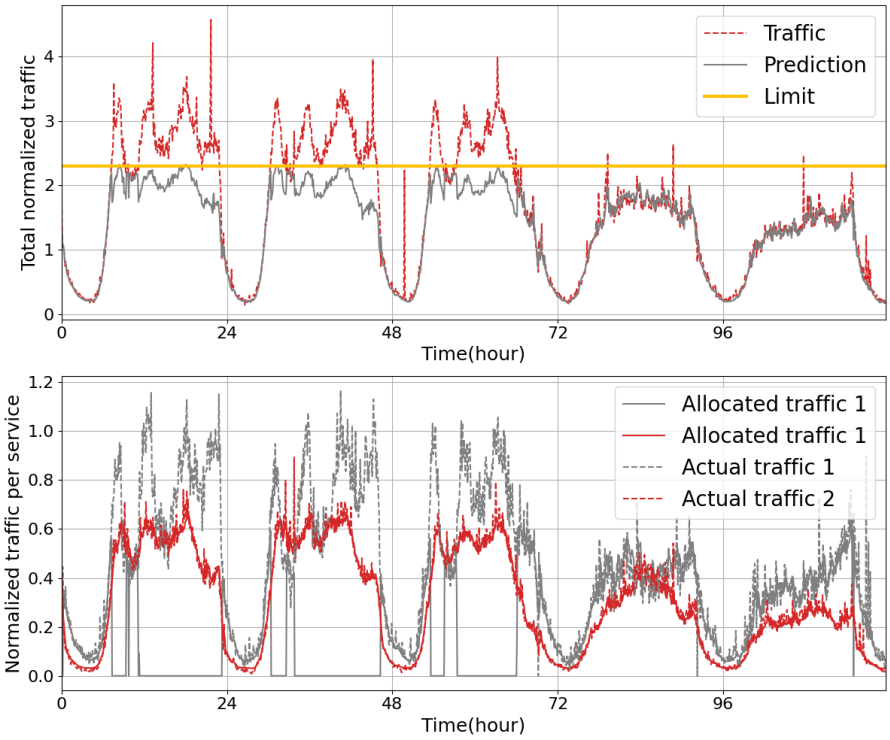


Figure 5.5. Illustration of the `MetaLoss` operation for a 4 slices overbooking scenario.

Slices	`MetaLoss`			`Knapsack`	
2	**0.1195**	(177,91%)	**1.28**	0.0430	1.00
4	**0.2382**	(47.04%)	**2.76**	0.1620	2.60
6	**0.3827**	(43.81%)	**5.67**	0.2661	4.36
8	**0.4215**	(20.57%)	**7.10**	0.3496	4.83
12	**0.5424**	(60.76%)	**10.80**	0.3374	8.94

Table 5.3. Results for use case in Section 5.2.1. We report the final Gain with Percent increase over the benchmark, and average number of slices Admitted. Slices vary from 2 to 12.

learns and exploits the sigmoid function in (5.1), which entails a negligible penalty if the QoE of the end users is reduced only slightly. Indeed, it is more profitable for the operator to gain a bit less on the red slice, and free up space for additional tenants.

Table 5.3 demonstrates quantitatively these results by reporting the performance of `MetaLoss` and the benchmark for different numbers of slices. We focus on non-trivial periods where the management task plays a role, i.e., when the slice demands exceed the capacity available at each Edge datacenter. Overall, `MetaLoss` allows the operator to effectively increase its revenues through overbooking by up to 60% when serving simultaneously 12 slices, and to admit an average of almost 2 slices more than the benchmark. We recall that it does so without any prior knowledge of the system or of the objective, which shows its meta-learning capabilities and the practical advantages entailed in solving the network management task.

	MetaLoss		Benchmark [35]	
Slice	Hours	% of day (24h)	Hours	% of day (24h)
1	7.28	(30.34%)	7.56	(31.5%)
2	24.00	(100%)	8.06	(33.58%)
3	24.00	(100%)	7.27	(30.25%)
4	19.41	(80.88%)	7.05	(29.33%)
Average	18.67	(77.80%)	7.485	(31,18%)

Table 5.4. AVST for the four services in the QoE-driven AC-RR use case.

Moreover, for the particular 4-slice scenario, we also analyze the temporal dynamics of the AC-RR decisions against a more sophisticated benchmark [35]. It uses a Holt-winters algorithm to predict the future slice traffic, and then solves a disjoint optimization problem aiming at maximizing the revenue of the operator. We note that the expert-designed benchmark assumes knowledge of the relationship between revenue and decisions that may not be available in operational settings whereas our approach does not.

For that, we define the Average Uninterrupted Service Time (AVST) as the continued time interval during which a slice is served upon acceptance; the interval is thus concluded once the slice is torn down by the operator. The metric is averaged on a daily basis and expressed in hours, with a range from 0 (the slice is never allocated) to 24 (the slice is served all the time); it provides insights on the slice reliability, as service providers are interested in maximizing the uninterrupted time. Table 5.4 presents the AVST for each slice. Our model demonstrates substantial enhancements in AVST, positively impacting overall quality of experience QoE, despite not being the direct focus of our model. Our solution is able to serve two of the slices continuously, as it solves a regression problem and has implicit temporal memory; instead, the benchmark incurs many interruptions as it solves an independent optimization problem at each time step.

For this use-case, the training times are presented in Table 5.5, where the three columns of each row correspond to $C_{\texttt{MetaLoss}}$, C_{Pred}, and C_{LS}, respectively. Those are the complexities depicted in Section 3.6.4. The difference between the first column and the sum of the other two columns corresponds to the residual time c. This experiment is run using an NVIDIA A100 GPU. These results highlight that the predictor's training takes most of the training time, and the percentage of training time that it consumes increases proportionally to the number of slices. It is understandable as the predictor's architecture considerably varies with the addition of new services (IR are added) while only the number of inputs differs for the *loss shaper*. For this use case, the comparison is more complex as our model enables this kind of design (multiple input flux) that is not feasible with traditional approaches. This is an interesting point as it proves that the scalability of the `MetaLoss` architecture is not much more limiting than the one of the *predictors* itself.

	Training time (s±10)		
Slices	Total	*Predictor*	*Loss shaper*
2	**1680**	990	560
4	**2250**	1430	620
6	**2920**	2010	680
8	**3530**	2550	770
12	**4720**	3690	900

Table 5.5. Training time for use case of Section 5.2.1 for different number of concurrent services.

5.2.2. Energy-prudent vRAN management

In the second use case, we focus on virtualized RAN (vRAN) management, where the operator must decide on the functional split of the PHY/MAC function pipeline between Distributed Units (DU) residing closer to the radio access, and more powerful Central Units (CU) towards the Edge [112]. We assume that the primary goal of the operator is ensuring the RAN sustainability, by maximizing its energy efficiency [113].

5.2.2.1. MANO objective

Let us assume a specific scenario where N DUs handle the aggregated traffic generated from a set of Remote Units (RU) each. The DUs are associated to a single CU through high-speed mid-haul connectivity. Processing each PHY/MAC function has a different energy cost at the DUs and CU. The energy consumption at each DU is directly proportional to the traffic volume that it has to handle at a given time, since the equipment must be always active. Conversely, the CU runs a number of physical machines (PMs) that can be dynamically turned on/off based on the incoming demand by the DUs [113]. All PMs are identical, and thus have the same energy model [114]: activating a new PM incurs a start-up cost, then the energy consumption increases proportionally to the computing load of the PM[3].

Given the traffic demand at one DU, different functional splits entail diverse computational loads at the DU and CU, as they move PHY/MAC functions from the former to the latter. The MANO objective is then deciding, for each DU and decision time, what functional split shall be used to minimize the total energy cost of the vRAN system. It is important to stress that, since this is a preemptive decision, it must be based on a forecast of the mobile data traffic that the DU will observe during the next decision interval.

Once those decisions have been taken, the CU will only activate the minimum required number of PMs to serve the total incoming demand from all DUs. For that, at the start of each decision time, the CU receives the estimated traffic and functional split per DU, and

[3]We assume that the increment of energy per traffic unit is lower at PMs than at DUs, as the CU feature more expensive and scalable equipment.

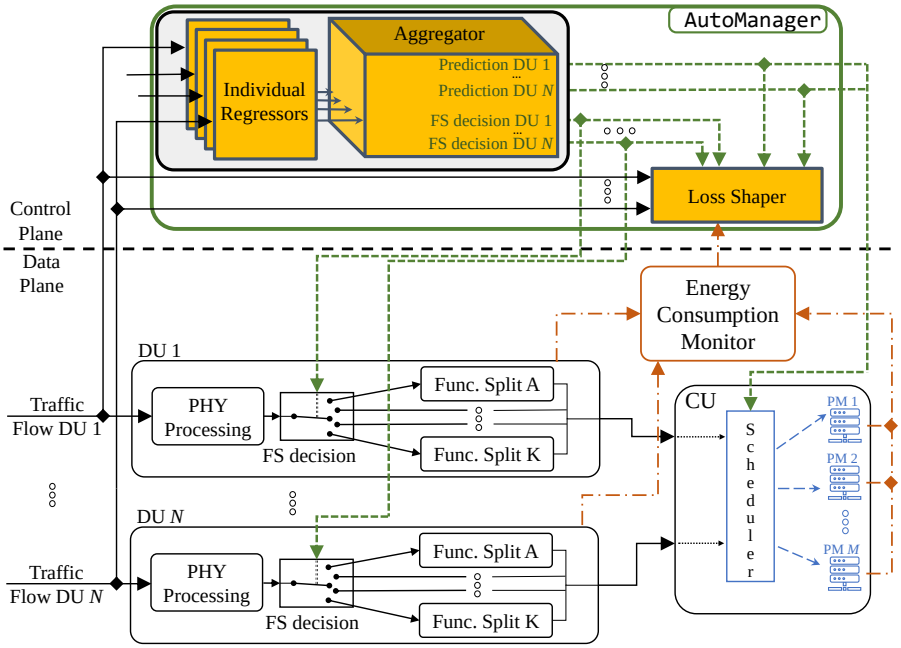


Figure 5.6. Implementation of **`MetaLoss`** in use case II.

applies a bin-packing [115] near-optimal heuristic algorithm to calculate the minimum amount of PMs required and the associated assignment of PMs to DUs. Moreover, since this a preemptive decision, the cost of operation does not depend only on the energy consumption, but also on the quality of the forecast.

5.2.2.2. Solution based on `MetaLoss`

We first remark that, in practice, the network operator does not know the policy on PM activation decided by the cloud provider hosting the CU, or the exact power consumption behavior of the PMs and DUs equipment. Still, the operator must take functional split decisions and reserve compute capacity in the CU without such information. In addition, the fact that the PM activation decision depends on all DUs sets the stage for a clear instance of intertwined predictions under unknown objectives.

In this use case, we can use **`MetaLoss`** to implement the network function that takes care of both (*i*) deciding on the functional split at each DU, and (*ii*) forecasting the traffic at each DU used to request compute capacity at the CU. We can see in Figure 5.6 how the detailed model of **`MetaLoss`** from Figure 3.2 is integrated in the control plane. In this case, the input to **`MetaLoss`** is composed of N different traffic flows, one for each one of the DUs. **`MetaLoss`** receives the past samples of these flows, and it outputs two sets of values: the first set is a variable per DU that states the specific functional split that is selected, and the second set contains the amount of processing from each DU required at the CU as result of the selected functional split and predicted load. This decision is then

		fullDU		fullCU	
DUs	MetaLoss	Power	Increase	Power	Increase
2	**138.4**	180.3	(30.3%)	162.1	(17.1%)
4	**339.2**	365.6	(7.8%)	415.2	(22.4%)
6	**496.8**	542.7	(9.2%)	566.8	(14.1%)
8	**624.4**	663.7	(6.3%)	682.4	(9.3%)
10	**824.7**	861.9	(4.5%)	886.6	(7.5%)

Table 5.6. Performance summary for use case II. Mean power consumption (W) at all DUs and the CU, and increase incurred by the benchmarks with respect to `MetaLoss`.

implemented and the energy consumption is measured a posteriori, so that it can be fed back to `MetaLoss` to train the algorithm and learn the appropriate loss function.

5.2.2.3. Benchmarks

We compare the performance of `MetaLoss` in the use case above versus that of two benchmarks. The first one, named `FullDU`, consists in selecting the functional split that makes the DUs handle all the processing of their corresponding RUs; the second benchmark, named `FullCU`, is the case in which all the processing load allowed by the lowest functional possible is transferred to the CU. These two extreme policies must be fed with forecasts of the traffic load at each DU. Since these are traditional predictions of mobile traffic, we employ individual legacy LSTM neural networks [32] to anticipate the load of each DU in the next decision interval. The use of more complex models, such as meta-learning ones, is not necessary for such an obvious prediction task.

We choose these simple approaches because deriving the optimal functional split is not feasible: we would need to solve an extremely complex optimization problem, where the bin-packing algorithm running at the CU, which is itself a NP-hard problem [115], is only a subset of the overall MANO objective.

5.2.2.4. Results

We run tests based on the same measurement data as the precedent use case i.e., `TrafficApp`. In this case, we assume that a variable number of DUs, from 2 to 10 are deployed in the region, and are assigned to a single CU. Each DU aggregates the total traffic generated by all applications at a disjoint subset of RUs, which we map to base stations. The operator predicts the upcoming traffic at each DU and recomputes the associated functional split at every 5 minutes. The energy consumption of each PM is based on real-world server specifications [116], such that a PM has an energy consumption is 80 W when idle, which linearly grows to 200 W at full load of 200 Mbps.

A view of the overall results of the experiment is provided in Table 5.6, where we list the mean power consumption in W, aggregated over all DUs and the CU. The figures clearly show how **MetaLoss** learns the energy cost of different functional splits, unravelling the complex relationship between the anticipatory decision variables and the power consumption at the CU. As a result, **MetaLoss** takes MANO decisions on the split and requested resources at the CU that yield a reduction of the mean power consumption in the 4%–30% range with respect to the two benchmarks. It is worth noting that the fact that the performance gain of our solution diminishes with the number of DUs is not a sign that **MetaLoss** is not scalable; rather, it is due to the maximum gain being bounded between the power consumption of the PMs used by the **FullDU** model and that one PM less. Then, a larger set of DUs implies that more PMs are required at the CU, and that the maximum gain is inherently reduced.

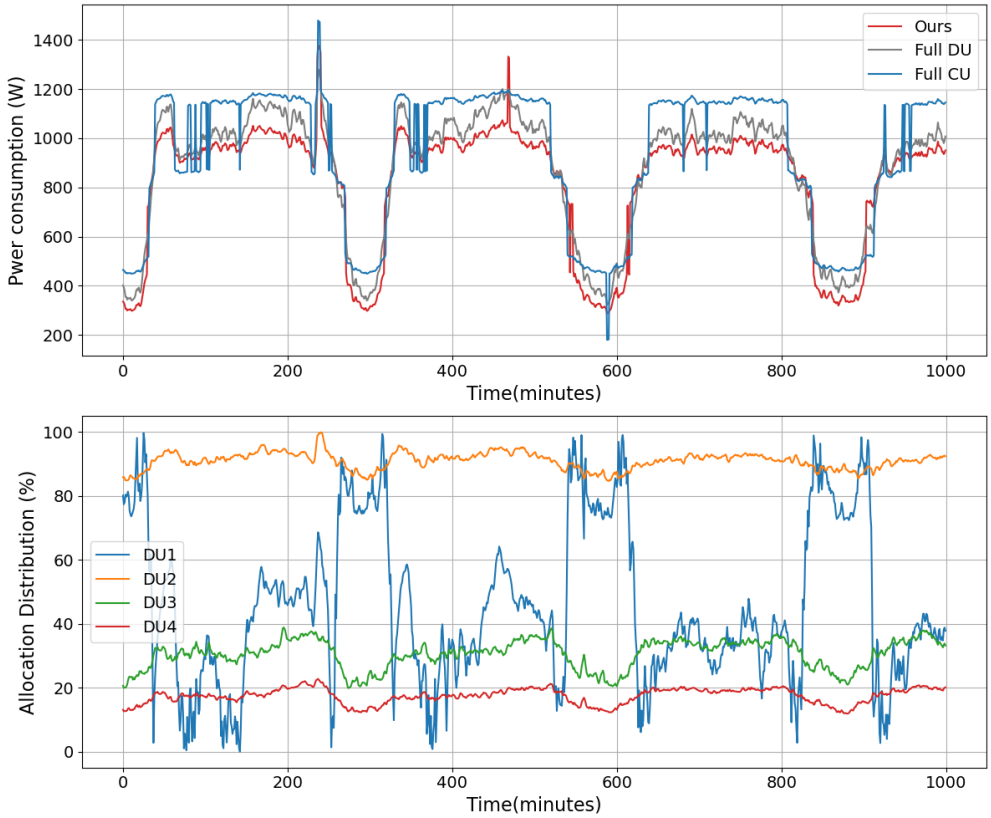


Figure 5.7. Temporal detail of the performance of **MetaLoss** and the **FullDU** and **FullCU** benchmarks in use case II.

We also provide details on the temporal variance of such results in the top plot of Figure 5.7. The lower power consumption granted by our solution is steady over the wide fluctuations of mobile demands over consecutive days. Also, the performance of **MetaLoss** is more stable: e.g., the **FullCU** approach determines jumps in the power consumption, as minimum traffic variations can trigger the (de-)activation of an extra PM for a very

short amount of time.

The bottom plot in Figure 5.7 illustrates instead the evolution of the CU resource reservation requests issued by `MetaLoss` over time, for a sample of 4 DUs. The abscissa represents the percentage of the total computing load arriving at the i-th DU that is moved to the CU for processing. We observe how `MetaLoss` redistributes the load of the DUs to a different extent over time, dynamically adapting the traffic fluctuations. In particular, the model acts on the decision at DU 3 to reach the optimal operation point; when more demand is moved from DU 3 to the CU, the PM resources requested by the other three DUs are slightly reduced to ensure the load of the CU stays at the correct level that minimizes the power consumption.

5.3. Summary

Finally, it has been demonstrated in this Chapter, through a consequant array of experiments and diverse use cases, the unparalleled proficiency of `MetaLoss` in navigating complex scenarios. These scenarios entail intertwined predictions directed towards a common overarching objective, all while adeptly managing the resolution of the loss metric mismatch described in the precedent Chapter 4. This work is a pioneer as up to now, such a global outcome problem was previously unattainable by existing methodologies, which primarily addressed adversarial scenarios. Remarkably, `MetaLoss` not only resolves these challenges but also augments the transparency and interpretability of the system, marking a pioneering advancement in the field.

Part III

Explainability

6 Explainability in MetaLoss

In this Chapter, we focus on how the **MetaLoss** architecture can enhance the explainability and transparency of DNN models. Explainability has become a paramount concern, growing in importance daily. Global regulations, such as the General Data Protection Regulation (GDPR) in Europe, now mandate transparency and explainability in AI models, reflecting this strengthen focus.

But explainability offers numerous other advantages beyond regulatory compliance. For instance, it helps to build trust among users and stakeholders by providing clear insights into how decisions are made. This is especially crucial in high-stakes domains like healthcare, finance, and law enforcement, where decisions can have profound impacts on individuals' lives.

Explainability also aims to increase the adoption of AI as transparent and explainable AI systems are more likely to be accepted and used. When users understand how AI makes decisions, they are more likely to trust and adopt these technologies, facilitating smoother integration into everyday life and reducing resistance to new AI applications.

Lastly, explainability is vital for identifying and correcting errors, biases, or unintended behaviors in AI models. Explainability techniques can reveal which features are most influential in model predictions, enabling more effective debugging and refinement. This ongoing evaluation and improvement process enhances model performance and ensures the reliability and accuracy of AI systems over time.

In this context, the **MetaLoss** model aims to have a significantly improved explainability over standard AI models, making it a perfect fit for real-world engineering problems.

6.1. Explainability and security

The advantage of representing the loss function by a neural network (the *loss shaper*) in **MetaLoss** is twofold, it offers significantly enhanced explainability and security of predictive models.

First, employing a neural network as the foundation for the loss function offers a notable advantage that extends beyond the training phase. Even post-training, this approach enables the depiction of the system's behavior through a diverse array of complex shapes, exemplified in the different use cases and in the next sections. These shapes serve as representations of the system's response to varying inputs, facilitating a deeper understanding of the predictor's behavior. Notably, users can easily generate different shapes by inputting diverse values directly through the loss shaper, without affecting the predictor itself. By offering a clear window into the model's decision-making process, this transparency empowers users to make informed decisions and anticipate the predictor's responses across a spectrum of potential circumstances, thus enhancing both comprehension and confidence in the model's capabilities.

Secondly, the use of a neural network as the loss function provides a substantial boost to the security of the entire system. One significant advantage lies in its ability to avoid being targeted by the most sophisticated attacks, such as the BIM as described in Chapter 7 as those methods need the use of the loss function knowledge, In most actual scenarios, those are simple enough to be brute forced as they only rely on legacy loss functions or simple custom-made ones. With `MetaLoss`, the complexity of the neural network renders such attacks virtually impossible. Unless the *loss shaper* is known in advance, identifying the loss function utilized by the predictor remains an insurmountable task, thereby preventing adversaries from executing these advanced attacks. Also, even in the face of more standard and simpler attacks, the model's capacity to mitigate conflicts, as elaborated upon in Chapter 5, further enhances the system's security. The model will prolong the time required for attackers to compromise the system. Compared to conventional models, the resilience afforded by the neural network-based *loss shaper* ensures a higher threshold before breaking, thereby enhancing the overall robustness and longevity of the system against adversarial threats.

6.2. System adaptation

In this section, we explore how the results from `MetaLoss` are interpretable in the different possible scenarios, setting it apart from other methods like Reinforcement Learning.

6.2.1. Learning the uncertainty of the predictor

In this first example, we use the simple scenario for `MetaLoss` presented in Section 4.3. This scenario aims to optimize a non-symmetric metric for a capacity forecasting problem. The key to the improved performance presented in the different tables in section 4.3 of the loss meta-learning solutions is unveiled in Figure 6.1. There, the manual α-OMC loss is designed as a bi-dimensional function of the error between the predicted capacity and

the actual capacity required by the service (i.e., the abscissa values), which indeed mimics the ground-truth metric.

Instead, our proposed approach (automatically) learns a more complex three-dimensional function over the complete space of all combinations of predicted and needed capacity. This allows `MetaLoss` to adapt to fluctuating traffic demands. The model is designed to learn a loss that adjusts for varying levels of precision in predicting traffic of different scales. This is illustrated in the plot, where, for the sake of clarity, we present a simplified view of the meta-learned loss for high and low demand volumes only. Our model finds different adapted curves to the two cases: in presence of high traffic loads it shifts the minimum loss point to the right to offer a safer margin against the higher errors incurred by the predictor in such traffic conditions.

This type of adjustments in the loss function are impossible to ascertain by just looking at the performance metric, as they inherently depend on the prediction quality. Yet, co-training the predictor and loss shaper block as done by our proposed model enables discovering a loss that is optimized for different absolute values of the predicted variable.

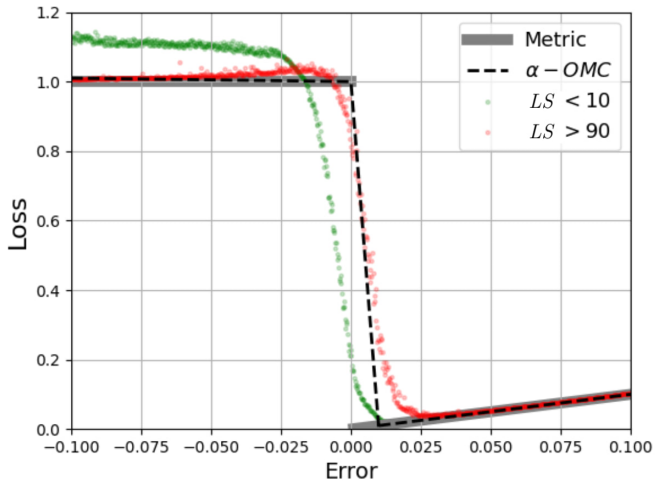


Figure 6.1. Normalized cost (loss function) for this first use case: (*i*) ground-truth metric defining the relationship between decisions and performance (gray), (*ii*) expert-designed loss function that models the metric in `DeepCog` (α-OMC, dashed black), and (*iii*) fully automated loss meta-learning solution (LS). For LS, we show the learned function by the loss shaper for the top (red) and bottom (green) deciles of the predicted values, i.e., in presence of high and low traffic demands, respectively.

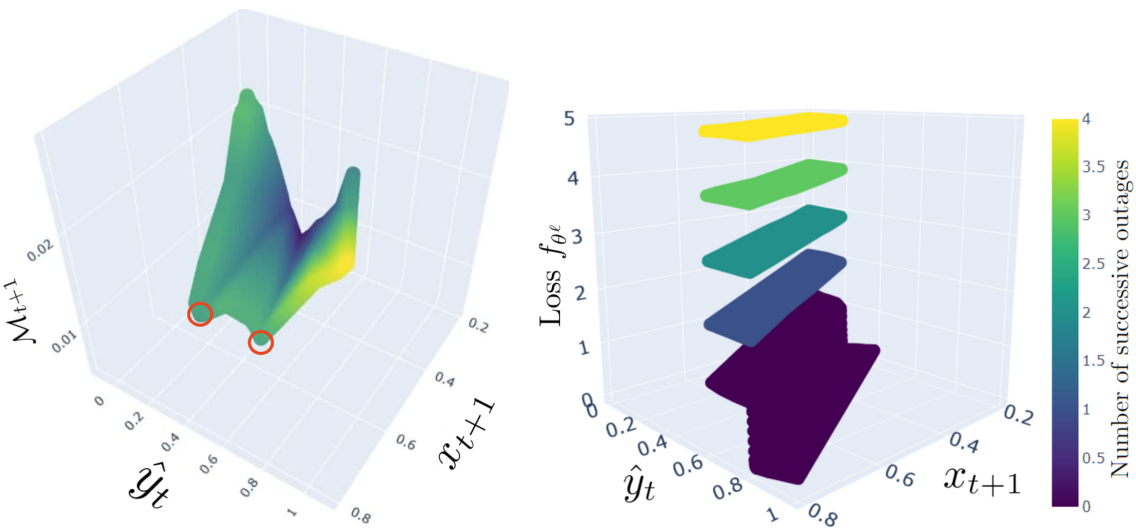
6.2.2. Visualizing complex loss-metric

The major strength of `MetaLoss` for enhanced explainability is its ability to easily visualize the learned loss by the loss shaper. This is something that is already depicted in the previous subsection 6.2.1. But the model's capability extends to more complex scenarios. Here we are focusing first on two particular use cases: The minimization of the

OPEX developed in Section 4.4.3, and the power grid management from Section 4.4.1.

In the first use case, approximating a metric that considers the QoE of end-users is nearly impossible. While we can assume that increasing the received data will generally make users happier, the extent of this effect is indeterminate. However, using the `MetaLoss` model, the derived loss function, shown in Figure 6.2a, provides significant insights into the system's behavior. For instance, there is a second, nearly equivalent minimum for the highest future traffic values (highlighted with red circles). This insight is crucial for scenarios where an event necessitates a reduction in throughput for a specific service. We can infer that in such cases, the QoE of end-users will degrade (with notations we can assume it will downgrade from 5 to 4 stars). The new optimal minimum expenditure would then be this second minimum, as increasing traffic further would not improve QoE and would result in unnecessary expenses.

Regarding power grid management, the metric evolves with consecutive power outages, forming a complex 4-dimensional shape that is nearly impossible to manually design (Figure 6.2b). In this context, the evolution of the metric with respect to the number of consecutive outages is accurately depicted. It is clear that as the number of consecutive outages increases, the final metric value also increases.

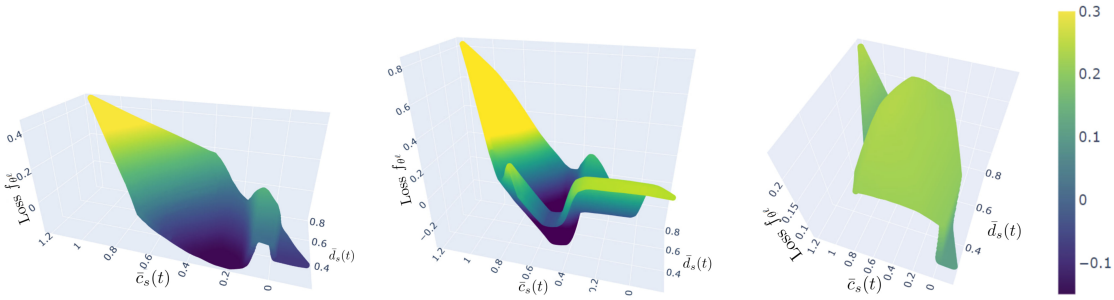


(a) OPEX optimization scenario **(b)** Power Grid Management

Figure 6.2. Loss function learned by `MetaLoss` in two different complex real-world scenarios.

Finally, we can use `MetaLoss` to model even more complex situations. To this end we use the overbooking use case depicted in Section 5.2.1. For this use case, we present the learned loss for a four-service scenario in Figure 6.3. This figure presents three specific perspectives of the complete eight-dimensional loss (two dimensions per slice), taking into account anticipatory decisions and actual demand. The other dimensions have been set up for different scenarios (low traffic demand, medium traffic demand and high traffic demand). This visualization underscores the impracticality of manually designing such a

complex loss; yet, with **MetaLoss**, it is both achievable and interpretable. From the shape at those different levels, we can easily identify that in the case of (a) low background, the slice can be accepted and with sufficient capacity to fulfill the demand easily no matter the importance of it, while in the (c) high background case, the demand must be extremely high in order for the slice to be accepted, and would be rejected otherwise. In the (b) medium background scenario, both behaviors are present, meaning that the slice demand must be consequent enough in order to be accepted. Those information helps to understand how the system will behave in different scenarios.



(a) Low background **(b)** Medium background **(c)** High background

Figure 6.3. Learned loss for the AC-RA task in sliced networks. We show the learned cost (z axis) for varying anticipatory decisions (x axis) and traffic demands (y axis) of one slice, while all other slices have fixed values. We represent the cases where the other slices generate (a) low, (b) medium, and (c) high traffic loads.

6.3. Summary

To wrap up, the explainability and transparency of the **MetaLoss** model are significantly enhanced by its distinctive architecture, setting it apart from standard AI approaches. This design improves **MetaLoss**'s resilience against potential malicious attacks and ensures models' interpretability by representing complex systems in an understandable form. Such improved explainability is not just beneficial but essential in contemporary scenarios and for IBN adoption.

7 Explainability in DNN predictors

The objective of this chapter is to underscore the fundamental importance of explainability within networking contexts, particularly illustrated through a spatio-temporal use case. This will emphasize the importance of the explainability features of the model depicted in the previous Chapter 6.

The aim is to highlight the necessity of enhancing robustness and resilience in DL-driven networking issues. To achieve this, we narrow our focus to a specific aspect of the problem. Conventional attacks such on spatio-temporal-based DNN models are impractical, as they would necessitate modifications across numerous base stations utilized by the model, potentially numbering in the thousands depending on input size. Instead, we pose the question: "Can we identify the most influential base station for forecasting?" If so, it becomes possible to ascertain whether altering the behavior of a limited number of impactful strategic points is adequate to deceive the predictor. Answering this query entails integrating EXplainable AI (XAI) to discern the most influential base station for the model from a spatio-temporal standpoint. This endeavor entails addressing two major challenges.

Firstly, the *semantic interpretation*. Current XAI methods lack the ability to offer thorough explanations of model operations. Although they provide insights specific to the model (such as how neurons respond to inputs), what is required are explanations with tangible significance (such as identifying the most influential base station).

The other key challenge is to focus on *utility*. Beyond semantic interpretation, the more comprehensive explanations should strike a balance between being meaningful and practical in their level of detail. Visualization tools often fall short in this regard, either due to their specificity to particular models or because they inundate users with excessive information, as observed in TSViz [73].

7.1. Problem formulation

7.1.1. Prediction

The primary aim of deep neural networks (DNNs) employed in mobile traffic forecasting is to anticipate the traffic volume at time $t+1$ based on historical traffic data or other quantities related to this as it can be seen in the use cases of the previous Chapters 4 and 5 of this thesis. Formally, considering $\boldsymbol{X} = (\boldsymbol{X}^1, \boldsymbol{X}^2, \ldots, \boldsymbol{X}^T)$ as the tensor of sequence of traffic snapshots at time $t = 1, 2, \ldots, T$, each snapshot $\boldsymbol{X}^t$ encompasses data from geographically dispersed base stations, each identified by its coordinates (r, c) of dimensions $R \times C \in \mathbb{N}^2$.

$$\boldsymbol{X}^t = \begin{bmatrix} x^t_{1,1} & \cdots & x^t_{1,C} \\ x^t_{2,1} & \cdots & x^t_{2,C} \\ \vdots & \ddots & \vdots \\ x^t_{R,1} & \cdots & x^t_{R,C} \end{bmatrix}. \tag{7.1}$$

Therefore, $x^t_{r,c}$ measures the traffic volume at the base station located at (r, c) at time t. Let $\boldsymbol{X}^{S,t} \in \mathbb{R}^{R \times C \times S}$ be the set of historical S past traffic observations at time t: $\boldsymbol{X}^{S,t} = (\boldsymbol{X}^{t-S+1}, \boldsymbol{X}^{t-S+2}, \ldots, \boldsymbol{X}^t)$. Note that S is known as history and $S \ll T$. Then, the forecast $\hat{\boldsymbol{X}}^{t+1}$ of the spatio-temporal traffic volume at time $t+1$ is:

$$\hat{\boldsymbol{X}}^{t+1} = f(\boldsymbol{X}^{S,t}) \tag{7.2}$$

where f represents a generic prediction function. Specifically, during the design phase of DNN models, the objective is to synthesize f as f_θ by determining the appropriate weights θ. The training process involves iteratively evaluating a loss function $L(\hat{\boldsymbol{X}}^{t+1}, \boldsymbol{X}^{t+1})$ and updating the model weights θ. The choice of L can be tailored based on the predictor's objectives. For assessment purposes, we will employ loss functions designed for standard traffic estimation and capacity forecasting, as detailed in Section 7.1.3. These losses are relatively straightforward compared to the objectives of the diverse use cases presented in previous chapters of this thesis, as the primary aim here is to emphasize the critical importance of explainability in the models.

7.1.2. LRP algorithm

The existing implementations that are publicly available for LayeR-wise backPropagation (LRP) are designed for sentiment analysis with LSTM, which is not suitable for spatio-temporal forecasting. Therefore, an alternate version has been designed for it. In analogy with computer vision where the objective is to understand the relevance of each pixel of an image at each point in time t, the objective here is to characterize the relevance of each base station by assigning scores to $x^t_{r,c}$. We need to take into

account that each prediction $\hat{\boldsymbol{X}}^{t+1}$ depends on the past sequence of observations $\boldsymbol{X}^{S,t}$. Call $\boldsymbol{Z}^t = (\boldsymbol{Z}^{t-S+1}, \boldsymbol{Z}^{t-S+2}, \ldots, \boldsymbol{Z}^t)$ the relevance scores associated to the prediction at $t+1$. Then, at each timestamp t, $z^t_{r,c}$ defines the relevance of each traffic volume observed at the base station located in (r, c). In general,

$$\boldsymbol{Z}^t = \begin{bmatrix} z^t_{1,1} & \cdots & z^t_{1,C} \\ z^t_{2,1} & \cdots & z^t_{2,C} \\ \vdots & \ddots & \vdots \\ z^t_{R,1} & \cdots & z^t_{R,C} \end{bmatrix}. \tag{7.3}$$

LRP assigns a score to all the inputs of a predictor and this score indicates the extent of their contribution to the predictor. The scores are computed by tracking back from the output the individual activation a_i of each neuron i and its contribution to neuron j with weight $\theta_{i,j}$ in subsequent layers of the neural network h and $h+1$. Formally:

$$\boldsymbol{Z}^{(h)}_{i \leftarrow j} = \boldsymbol{Z}^{(h+1)}_j \sum_{i,j} \frac{a_i \cdot \theta_{i,j}}{\sum_k a_k \cdot \theta_{k,j}}. \tag{7.4}$$

LRP follows a conservation principle for which the total amount of relevance distributed in layer $h+1$ remains unaltered in layer h. When the backpropagation reaches the input layer, the relevance is distributed to the input. By averaging over the different previous time instants, we obtain a compact and useful metric that uniquely identifies the temporal relevance of each base station $\boldsymbol{Z}_{r,c}$, thereby addressing the two challenges presented in the Introduction of this Chapter 7.

7.1.3. Dataset

For the experiment, we rely on the following dataset, whose attributes and properties are described thereafter.

- **Milan Dataset.** The Telecom Italia dataset contains mobile traffic data from two areas in Italy, Milan and Trentino, collected in 2014 [117]. This is the state-of-the-art dataset used in the literature (e.g., [81]). The data comes from 1728 base stations and is aggregated in a grid comprising square cells, e.g., 10 000 cells for Milan. The data contains SMS, voice calls, and "Internet activities" at a 10 minutes granularity. Similar to other works that rely on this dataset [118], we use "Internet activities" as a proxy for mobile traffic volume.

- **EU Metropolitan Area (EUMA) Dataset.** The second dataset contains traffic volumes generated by a set of popular mobile applications like YouTube, Facebook, Netflix, Twitch, and Whatsapp, among others. The data was collected in a production LTE network that provides service to a major metropolitan region

in Europe in 2019. The dataset describes service-level traffic volumes at each of over 400 base stations. As in the case of the Milan dataset, the traffic information is aggregated over 10-minutes intervals and mapped to a regular grid of 3 400 cells using the same Voronoi-based methodology [119]. We remark that, in order to make the scenarios comparable, grid cells in the Milan and EUMA datasets have the same size, i.e., 325×325 m^2.

7.2. Prediction algorithm

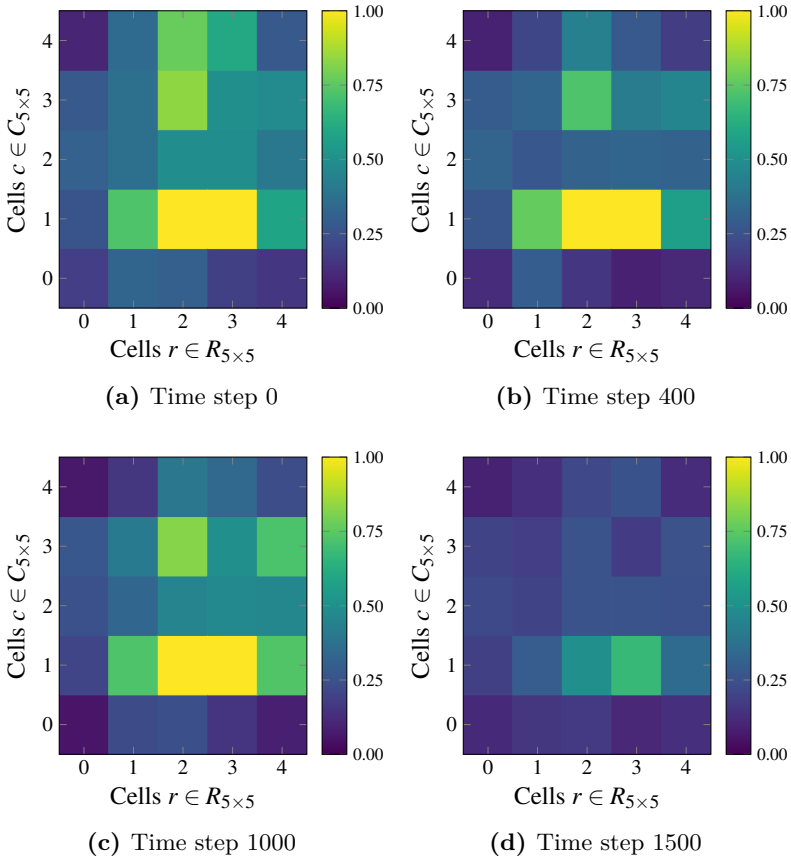
For the prediction algorithm, we use DeepCog [120]. It was designed for capacity forecasting using spatio-temporal data and it aims at allocating sufficient resources for the operator to jointly minimize overprovisioning and penalty for non-served demands (i.e., SLA violations). We train small models on 5×5 grids and each model forecasts the capacity of the central cell only. This renders the analysis of the vulnerability more practical as the state-of-the-art attacks would craft perturbations on a few base stations and not all of those in the bigger areas. Finally, we use, for the predictor $S = 3$ as the number of past observations.

7.3. Highlighting weaknesses

Here, we showcase that across the spatio-temporal domain, not all base stations contribute equally to the prediction. Figure 7.1 shows the relevance scores for one of the 5×5 grids over different time steps (e.g., 0 corresponds to the first sample of the **Milan** test set). This highlights the fact that the relevance scores for a given cell fluctuate over time across different instances of the test set.

We break down the global relevance matrix into the individual components Z^{t-s} with $s \in S$ (in our case, $S = 3$) in the **EUMA** dataset (see Figure 7.2). The relevance scores in the corresponding grid fade out with the increase of s, i.e., moving into the past. This is naturally understood as the more recent the sample will be, the more relevant it will be to predict the next time stamp (without taking care of any seasonality aspect). Relevance scores within each historical step S generally exhibit a fading trend, albeit with some minor deviations.

Finally, these relevance scores do not entirely align with traffic dynamics, highlighting the necessity for explainable artificial intelligence (XAI) tools. This underscores that DNNs captures more intricate dynamics beyond mere instantaneous traffic volumes, indicating that these volumes alone cannot serve as a proxy for base station relevance.



(a) Time step 0 **(b)** Time step 400

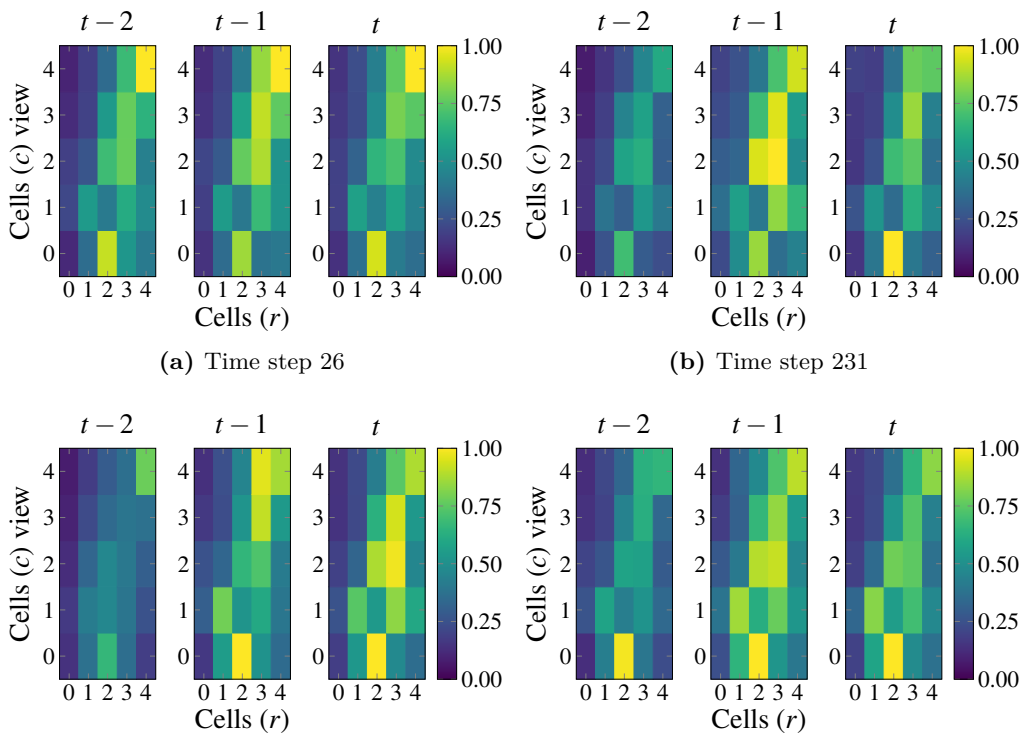
(c) Time step 1000 **(d)** Time step 1500

Figure 7.1. Relevance scores from the analysis of the **Milan** dataset with the capacity forecasting predictor.

7.4. Exploiting weeknesses

Concerning the state-of-the-art attacks, we use FGSM and BIM [80]. Those are whitebox attack, meaning that the model must be none by the attacker. FGSM computes the gradient of the cost function relative to the neural network input and crafts adversarial inputs $\overline{\boldsymbol{X}}^t = \boldsymbol{X}^t + \eta$ with $\eta = \epsilon \cdot \text{sign}(\bigtriangledown L(\boldsymbol{X}^t, \hat{\boldsymbol{X}}^t))$, where $\boldsymbol{X}^t$ is the real input, $\overline{\boldsymbol{X}}^t$ the adversarial one, L the loss function of the model and $\bigtriangledown$ the gradient of the model computed with respect to the ground truth $\boldsymbol{X}^t$.BIM employs FGSM for a specified number of iterations, with each step allowing a perturbation of up to ϵ. Recognizing that simultaneously jamming multiple base stations is less feasible than injecting load (e.g., via mobile devices), we adapt FGSM and BIM so that when the gradient is negative, the perturbation is nullified. This modification ensures that the attacks solely inject traffic rather than subtracting traffic volumes.

In our scenario, perturbing $\boldsymbol{X}^t$ within the context of capacity forecasting involves applying baseline attacks across the entire grid of cells used for predicting the central one.



(a) Time step 26 **(b)** Time step 231

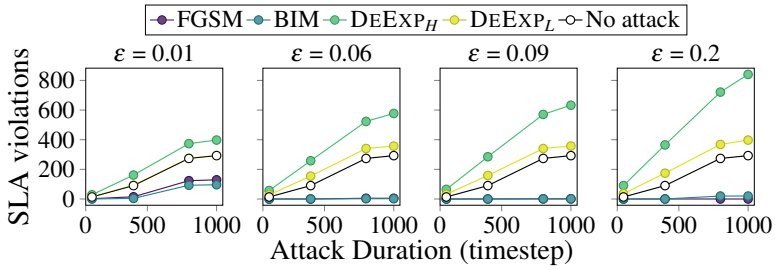
(c) Time step 248 **(d)** Time step 394

Figure 7.2. Relevance scores from the analysis of the **EUMA** dataset with the capacity forecasting predictor for a grid with high capacity.

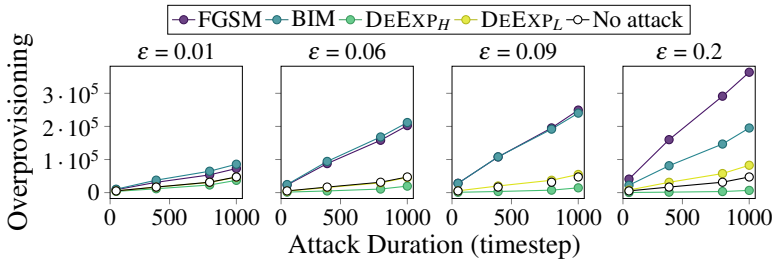
In contrast, utilizing the relevance scores, we can identify the most significant cells in the grid for perturbation. Consequently, we directly perturb the time series of these identified cells. To ensure a fair comparison, we ensure that the same amount of traffic B is injected. To achieve this, we define the duration and steps of the attack as $D = [d_1, d_2, \ldots, d_N]$, where $N = |D|$, and determine $B = \sum_{d=1}^{N} \eta_d$ while keeping ϵ fixed for FGSM/BIM. Thus, we define:

- DeExp_H as the strategy that always perturbs the most relevant cell for the prediction during D.
- DeExp_L as the strategy that always perturbs the least relevant cell for the prediction during D.

For each dataset, we vary the amount of injected traffic B by fixing 4 different values of ϵ (i.e., $\epsilon = \{0.01, 0.06, 0.09, 0.2\}$). We run BIM with 200 iterations. For each B, we set 4 attack durations (i.e., $D = \{100, 144, 250, 350\}$, where D is expressed as the number samples of 10 minutes each) and select 4 different attack start times and 4 different cell grids in the test set of the datasets. For brevity, we average the results obtained when varying the attack start times and the cell grids. The attacks to the capacity forecasting



(a) SLA violations



(b) Overprovisioning

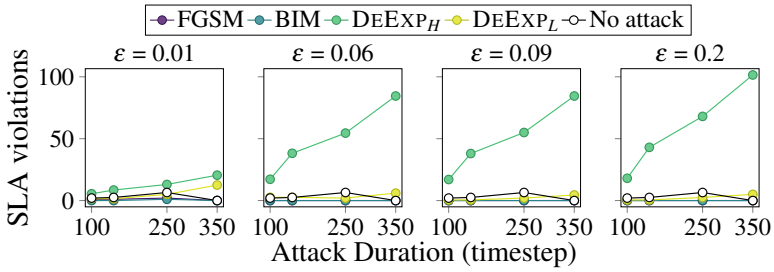
Figure 7.3. Capacity forecasting analysis for the **Milan** dataset. DeExp$_{H/L}$ denote respectively the perturbation attack upon having identified with DeExp the most relevant and the least relevant cell to perturb.

predictor are measured in terms of SLA violations and overprovisioning. Results are represented in Figures 7.3 for the **Milan** dataset and in Figure 7.4 for **EUMA** dataset.

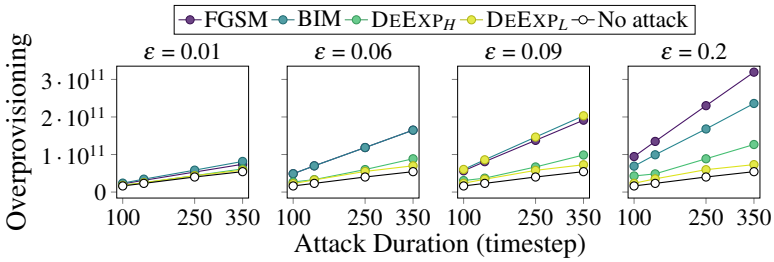
In all scenarios, DeExp$_H$ emerges as the strategy yielding the highest damage to the predictor. As anticipated, when compared to the more "aggressive" approach of DeExp$_H$, DeExp$_L$ results in lesser accuracy degradation. In the absence of any attack, the predictor aims for an equilibrium to minimize overprovisioning while avoiding incurring costly penalties for SLA violations. Notably, state-of-the-art attacks such as FGSM and BIM induce overprovisioning: by introducing traffic across all base stations, the predictor responds by provisioning additional capacity, as expected. However, DeExp$_H$ frequently leads to underprovisioning of the required capacity by the predictor, resulting in numerous SLA violations that are more economically burdensome for a Mobile Network Operator compared to overprovisioning.

7.5. Summary

In conclusion, this chapter has delved into the pressing challenge of evaluating the robustness and resilience of DNN models utilized in mobile traffic forecasting. Through this examination, we have observed that existing white-box adversarial attacks, relying on knowledge of model weights, can introduce perturbations across the entire spatio-temporal domain. This analysis underscores the vulnerability of DNN models, primarily



(a) SLA violations



(b) Overprovisioning

Figure 7.4. Capacity forecasting analysis for the **EUMA** dataset. DeExp$_{H/L}$ denote respectively the perturbation attack upon having identified with DeExp the most relevant and the least relevant cell to perturb.

due to the significant influence of certain base stations at specific moments. This inherent vulnerability presents an opportunity for adversaries to devise more sophisticated and disruptive attack strategies. As a result, urgent attention is required to develop countermeasures to mitigate these vulnerabilities. Additionally, this underscores the critical importance of explainability in contemporary models, emphasizing the need for transparency and understanding in the face of evolving threats.

8 Conclusion

Intent-Based Networking (IBN) represents a highly promising paradigm for shaping future network standards, necessitating significant advancements within networking environments. While a considerable body of research interests to enhance state-of-the-art standards primarily for forecasting purposes or for facilitating text-to-command applications through Natural Language Processing (NLP), this thesis takes a divergent path. Rather than solely focusing on these conventional areas, it seeks to extend data-driven approaches into previously unexplored realms, constrained by the limitations of existing models.

It primarily focuses on addressing two major conceptual problems encountered with actual used models, the *loss-metric mismatch* and facilitating the orchestration of *multiple predictions to attain a globally optimized outcome*. Additionally, it aims to enhance both *explainability* and security when compared to standard models, while also *minimizing human intervention* in the whole process.

To achieve these goals, an innovative approach to anticipatory network management is introduced, building upon recent discoveries that highlight the efficacy of custom loss functions in guiding predictors toward improved decision-making. This novel strategy, named `MetaLoss`, steers this approach of handcrafting adapted loss functions to its extreme and allows a total loss-agnostic approach by automatically constructing a loss-function adapted to the needs of the users and the system it is used on. Furthermore, `MetaLoss` can seamlessly incorporate available expert knowledge or expert-designed custom architectures, whose characterization of the problem may not be complete by themselves, but can be synergistically combined with `MetaLoss`.

It is also proven through a very comprehensive range of experiments, both in controlled environments and with real-world measurement data, that `MetaLoss` yields improved network management decisions in comparison with standard approaches while reducing human intervention. It also provides interesting and desirable properties, such as enhanced explainability of the decision-making process through meta-learned loss exploration, and transfer learning capabilities via reuse of a trained loss meta-learner.

Furthermore, the applicability of the `MetaLoss` principle extends beyond networking domains, as evidenced by practical implementations in other fields such as power grid management, underscoring its versatility and potential for broader impact for regression problems.

In summary, this thesis not only identifies critical challenges in network management but also offers practical solutions that align technical advancements with ethical considerations. It has provided a thorough examination of the challenges and advancements in network management, particularly in the context of evolving communication networks and the concept of zero-touch network orchestration. The evolution towards self-configuring systems necessitates anticipatory approaches, and the use of the `MetaLoss` model emerges as a transformative tool in bridging the gap between the theoretical IBN objectives and its real adoption.

8.1. Short-term advancement of `MetaLoss`

From what have been discussed throughout this thesis manuscript, there are multiple multiple research directions that can be explored.

Exploration of the IRs architechture. The first aspect, is to explore diverse architecture for the predictors. While extensive efforts have been devoted to designing various architectures for the *loss-shaper* block, the exploration of different IR models has remained relatively. This was intentional as the goal of the thesis is not to enhance the state of the art for regression purposes. However, during this thesis, numerous new models for forecasting have been discovered, the most advanced one leveraging Transformer architectures [121–123].

Transformers, since their creation, have given significant breakthroughs across a multitude of domains, and the realms of forecasting and regression are no exception. Their ability to capture long-range dependencies and contextual information has enhanced predictive modeling. An interesting perspective would be to experiment with the efficiency of such new architecture as the IRs of the `MetaLoss` model.

Another very interesting aspect would be to mix different architectures and data types and explore how the model can adapt to very diverse and complex scenarios. As an example, we could imagine one IR would still manage numerical data while another one will use embeddings on text-type data or images.

Exploit the architecture design for distributed architechture. A second aspect that could lead to a variety of applications is to use the inner design of `MetaLoss` for a distributed architecture. While, as depicted in Chapter 3, the output of the different IRs can not be identified as the hypothetical output without the coordination process, we can still split the architecture of the *predictor* into smaller subpieces.

In fact, we can easily imagine implementing the different IRs layers of the *predictor*

among different devices and using a central unit to hold the assembler part and the *loss-shaper* block. This could allow us to use the powerful **MetaLoss** model on minimal configurations with the concession of only minimal information exchanges between the different devices. This could be particularly useful with IoT devices as an example.

Explore the game theory aspect. Although Chapter 5 sheds light on the connection with game theory problems, there exists a notable gap in the exploration of this subject. delving deeper into this would be an interesting perspective as we can parallelize the **MetaLoss** approach to a continuous game, i.e., a game with continuous action space.

Moreover, an interesting approach would be to use the mixed strategy approach in game theory to adapt the **MetaLoss** model for classification purposes. However, this approach would come with a disadvantage: the necessity to compute hypothetical metrics for diverse non-happening actions during training poses computational challenges, potentially limiting the scalability of the model. Despite this limitation, the prospect of adapting the model's capabilities for classification tasks through a game-theoretic lens would be interesting for further investigation.

Explore real-world unknown metrics scenarios. While numerous real-world scenarios have been rigorously tested, showcasing the great efficiency of the **MetaLoss** model when compared to state-of-the-art methodologies, there remains an intriguing gap in the exploration. Specifically, we have yet to tackle a use case where the metric is truly unknown. Even in the most intricate scenarios, such as those detailed in Sections 4.4.3 and 5.2.1, where the metric undergoes a complex process and renders its adaptation into a handcrafted loss function impossible, it is still modeled to some extent.

A very interesting opportunity lies in deploying the **MetaLoss** approach in real-world conditions where modeling the metric is impractical but can be measured in practice. This endeavor would serve as a testament to the model's superiority over conventional methods and mark a significant stride forward for the implementation of IBN.

8.2. MetaLoss implementation in 6G networks

AI implementations in 6G networks and beyond is going to revolutionize communication technology by enhancing efficiency, speed, and connectivity. AI will play a critical role in optimizing network management, enabling real-time data processing, and facilitating dynamic resource allocation. This integration will support ultra-low latency and high reliability required for advanced applications like autonomous vehicles, smart cities, and immersive augmented reality experiences. Additionally, data-driven predictive maintenance and security measures will ensure network robustness and resilience against cyber threats.

One of the key advantages of AI in 6G networks is its ability to manage complex

and dense network environments. By utilizing machine learning algorithms, networks can predict traffic patterns, adjust to changing conditions, and optimize bandwidth usage dynamically. This ensures that data flows smoothly even in highly congested areas, enhancing user experience and reducing latency.

The proliferation of IoT devices will significantly increase the amount of data generated within 6G networks. This massive influx of diverse data makes AI approaches particularly suitable, as they can efficiently analyze and manage large volumes of information. AI's ability to process and interpret vast datasets in real-time is crucial for maintaining network performance and enabling advanced applications.

In those directions, the abilities provided by the `MetaLoss` model depicted throughout the thesis seem to be a perfect fit. Its capability to manage complex systems and address scenarios where the optimal solution is neither trivial nor *measurable a priori* makes it ideal for tackling intricate networking challenges. This is especially relevant for 6G networks, where the complexity of network management will be significantly higher.

Furthermore, `MetaLoss` is efficient at handling multiple predictors with varied data types, enabling it to satisfy a common objective from diverse data sources, which is essential for effective IoT data management. In 6G networks, the proliferation of IoT devices will lead to an unprecedented increase in data volume and variety. The `MetaLoss` model's proficiency in handling and analyzing heterogeneous data streams ensures accurate and usefull predictions, enhancing overall network performance and reliability.

Additionally, `MetaLoss`'s ability to represent systems through different shapes enhances the explainability and interpretability of the AI model's predictions. This distinct feature sets it apart from other AI models, making it exceptionally well-suited for integration into 6G network architectures. By providing clear and understable insights, the `MetaLoss` model helps stakeholders understand the reasoning behind AI decisions, facilitating better collaboration and more informed decision-making. Overall, the `MetaLoss` model is uniquely positioned to address the challenges of next-generation communication networks.

References

[1] Alan Collet, Albert Banchs, and Marco Fiore. Lossleap: Learning to predict for intent-based networking. In *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*, pages 2138–2147, 2022.

[2] Alan Collet, Antonio Bazco-Nogueras, Albert Banchs, and Marco Fiore. Automanager: a meta-learning model for network management from intertwined forecasts. In *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications*, pages 1–10, 2023.

[3] Serly Moghadas, Claudio Fiandrino, Alan Collet, Giulia Attanasio, Marco Fiore, and Joerg Widmer. Spotting deep neural network vulnerabilities in mobile traffic forecasting with an explainable ai lens. In *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications*, pages 1–10, 2023.

[4] Alan Collet, Antonio Bazco-Nogueras, Albert Banchs, and Marco Fiore. Explainable and transferable loss meta-learning for zero-touch anticipatory network management. *IEEE Transactions on Network and Service Management*, pages 1–1, 2024.

[5] Dario Bega, Marco Gramaglia, Marco Fiore, Albert Banchs, and Xavier Costa-Perez. Deepcog: Cognitive network management in sliced 5g networks with deep learning. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, pages 280–288, 2019.

[6] The 5G Infrastructure Association. European Vision for the 6G Network Ecosystem, 2021.

[7] 3GPP. Technical specification group services and system aspects, "tr:28.812 – study on scenarios for intent driven management services for mobile networks, telecommunication management. 2020.

[8] ETSI. Zero touch network and service management (zsm) means of automation. 2018.

[9] ITU-T. Recommendation – Architectural framework for machine learning in future networks including IMT-2020, 2019.

[10] ONAP. Technical Specification – AI/ML Workflow Description and Requirements v01.02.02, 2020.

[11] ETSI. GR NFV-IFA 041 – Network Functions Virtualisation (NFV) Release 4; Management and Orchestration; Report on enabling autonomous management in NFV-MANO, 2021.

[12] C Janz, N Davis, D Hood, M Lemay, D Lenrow, L Fengkai, F Schneider, J Strassner, and A Veitch. Intent nbi–definition and principles. *Open Networking Foundation, Version*, 2, 2015.

[13] D. Schulz. Intent-based automation networks: Toward a common reference model for the self-orchestration of industrial intranets. *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, pages 4657–4664, 2016.

[14] Alexander Clemm, Laurent Ciavaglia, Lisandro Granville, and Jeff Tantsura. Intent-based networking-concepts and definitions. *IRTF draft work-in-progress*, 2020.

[15] ETSI. Experiential networked intelligence (eni), "context-aware policy management gap analysis.". 2018.

[16] STANDARDIZATION SECTOR. Focus group on machine learning for future networks including 5g ml5g-i-125-r3.

[17] Yiming Wei, Mugen Peng, and Yaqiong Liu. Intent-based networks for 6g: Insights and challenges. *Digital Communications and Networks*, 6(3):270–280, 2020.

[18] Shen Wang, M. Atif Qureshi, Luis Miralles-Pechuán, Thien Huynh-The, Thippa Reddy Gadekallu, and Madhusanka Liyanage. Explainable ai for b5g/6g: Technical aspects, use cases, and research challenges, 2021.

[19] Qijia Jiang, Olaoluwa Adigun, Harikrishna Narasimhan, Mahdi Milani Fard, and Maya Gupta. Optimizing black-box metrics with adaptive surrogates, 2020.

[20] Hao Li, Tianwen Fu, Jifeng Dai, Hongsheng Li, Gao Huang, and Xizhou Zhu. Autoloss-zero: Searching loss functions from scratch for generic tasks, 2021.

[21] Sanket Shah, Kai Wang, Bryan Wilder, Andrew Perrault, and Milind Tambe. Decision-focused learning without differentiable optimization: Learning locally optimized decision losses, 2022.

[22] C. Zhang, P. Patras, and H. Haddadi. Deep learning in mobile and wireless networking: A survey. *IEEE Communications Surveys Tutorials*, 21(3):2224–2287, Q3 2019.

[23] Damilola Adesina, Chung-Chu Hsieh, Yalin E. Sagduyu, and Lijun Qian. Adversarial machine learning in wireless communications using RF data: A review, 2021.

[24] Fengli Xu et al. Big Data Driven Mobile Traffic Understanding and Forecasting: A Time Series Approach. *IEEE Transactions on Services Computing*, 9(5):796–805, Sep. 2016.

[25] Mingyang Zhang et al. Understanding Urban Dynamics From Massive Mobile Traffic Data. *IEEE Transactions on Big Data*, 5(2):266–278, Nov. 2017.

[26] S. Tom Au et al. Automatic forecasting of double seasonal time series with applications on mobility network traffic prediction. *JSM Proceedings, Business and Economic Statistics Section*, Jul. 2011.

[27] Stavros Ntalampiras et al. Forecasting mobile service demands for anticipatory MEC. In *Proc. of IEEE WoWMoM*, pages 14–19, Chania, Greece, Jun. 2018.

[28] Carolina Gijon, Matías Toril, Salvador Luna-Ramírez, María Luisa Marí-Altozano, and José María Ruiz-Avilés. Long-term data traffic forecasting for network dimensioning in lte with short time series. *Electronics*, 10(10), 2021.

[29] M. Zubair Shafiq et al. Characterizing and modeling internet traffic dynamics of cellular devices. In *Proc. of ACM SIGMETRICS*, pages 265–276, San Jose, CA, USA, Jun. 2011.

[30] Rongpeng Li et al. The prediction analysis of cellular radio access network traffic: From entropy theory to networking practice. *IEEE Communications Magazine*, 52(6):234–240, Jun. 2014.

[31] Chaoyun Zhang et al. Deep Learning in Mobile and Wireless Networking: A Survey. *arXiv:1803.04311 [cs.NI]*, Mar. 2018.

[32] Jing Wang et al. Spatiotemporal modeling and prediction in cellular networks: A big data enabled deep learning approach. In *Proc. of IEEE INFOCOM*, pages 1–9, Atlanta, GA, USA, May 2017.

[33] Ali Yadavar Nikravesh et al. An Experimental Investigation of Mobile Network Traffic Prediction Accuracy. *Services Transactions on Big Data*, 3(1):1–16, Jan. 2016.

[34] Chaoyun Zhang et al. Long-Term Mobile Traffic Forecasting Using Deep Spatio-Temporal Neural Networks. In *Proc. of ACM MobiHoc*, pages 231–240, Los Angeles, CA, USA, Jun. 2018.

[35] Josep Xavier Salvat et al. Overbooking Network Slices Through Yield-driven End-to-end Orchestration. In *Proc. of ACM CoNEXT*, pages 353–365, Heraklion, Greece, Dec. 2018.

[36] C. Gutterman et al. RAN resource usage prediction for a 5G slice broker. In *Proc. of ACM Mobihoc*, Catania, Italy, Jul. 2019.

[37] Dario Bega, Marco Gramaglia, Marco Fiore, Albert Banchs, and Xavier Costa-Perez. Aztec: Anticipatory capacity allocation for zero-touch network slicing. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, pages 794–803, 2020.

[38] Diederik P Kingma et al. Adam: A method for stochastic optimization. *arXiv:1412.6980*, Dec. 2014.

[39] Parag C. Pendharkar and Patrick Cusatis. Trading financial indices with reinforcement learning agents. *Expert Systems with Applications*, 103:1–13, 2018.

[40] Jae Won Lee. Stock price prediction using reinforcement learning. In *ISIE 2001. 2001 IEEE International Symposium on Industrial Electronics Proceedings (Cat. No.01TH8570)*, volume 1, pages 690–695 vol.1, 2001.

[41] T. M. Hospedales, A. Antoniou, P. Micaelli, and A. J. Storkey. Meta-learning in neural networks: A survey. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (01):1–1, may 2021.

[42] Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A. Efros. Dataset distillation, 2020.

[43] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[44] Yang Fan, Fei Tian, Tao Qin, Xiang-Yang Li, and Tie-Yan Liu. Learning to teach. In *International Conference on Learning Representations*, 2018.

[45] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *ICML'17*, page 1126–1135. JMLR.org, 2017.

[46] Marcin Andrychowicz, Misha Denil, Sergio Gomez Colmenarejo, Matthew W. Hoffman, David Pfau, Tom Schaul, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems (NIPS 2016)*, volume 29, pages 3981–3989. NIPS Proceedings, 2016.

[47] Paul Micaelli and Amos Storkey. Non-greedy gradient-based hyperparameter optimization over long horizons, 2020.

[48] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*, 2019.

[49] Ferran Alet, Erica Weng, Tomás Lozano Pérez, and Leslie Pack Kaelbling. *Neural Relational Inference with Fast Modular Meta-Learning*. Curran Associates Inc., Red Hook, NY, USA, 2019.

[50] Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. Bilevel programming for hyperparameter optimization and meta-learning. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1563–1572. PMLR, July 2018.

[51] Haowen Xu, Hao Zhang, Zhiting Hu, Xiaodan Liang, Ruslan Salakhutdinov, and Eric Xing. Autoloss: Learning discrete schedule for alternate optimization. In *International Conference on Learning Representations*, 2019.

[52] Qingliang Liu and Jinmei Lai. Stochastic loss function. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):4884–4891, Apr. 2020.

[53] Giulia Denevi, Carlo Ciliberto, Dimitris Stamos, and Massimiliano Pontil. Learning to learn around a common mean. In *Advances in Neural Information Processing Systems (NIPS 2018)*, volume 31. NIPS Proceedings, 2018.

[54] Sarah Bechtle, Artem Molchanov, Yevgen Chebotar, Edward Grefenstette, Ludovic Righetti, Gaurav Sukhatme, and Franziska Meier. Meta learning via learned loss. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 4161–4168, 2021.

[55] Chen Huang, Shuangfei Zhai, Walter Talbott, Miguel Ángel Bautista, Shih-Yu Sun, Carlos Guestrin, and Joshua M. Susskind. Addressing the loss-metric mismatch with adaptive loss alignment. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of International Conference on Machine Learning, ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 2891–2900. PMLR, June 2019.

[56] Sen Zhao, Mahdi Milani Fard, Harikrishna Narasimhan, and Maya Gupta. Metric-optimized example weights. In *International Conference on Machine Learning*, pages 7533–7542. PMLR, 2019.

[57] A. Ali Heydari, Craig A. Thompson, and Asif Mehmood. Softadapt: Techniques for adaptive loss weighting of neural networks with multi-part loss functions, 2019.

[58] Chen Huang, Shuangfei Zhai, Pengsheng Guo, and Josh M. Susskind. Metricopt: Learning to optimize black-box evaluation metrics. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 174–183. Computer Vision Foundation / IEEE, 2021.

[59] Jonathan T. Barron. A general and adaptive robust loss function, 2019.

[60] Josif Grabocka, Randolf Scholz, and Lars Schmidt-Thieme. Learning surrogate losses, 2019.

[61] Lijun Wu, Fei Tian, Yingce Xia, Yang Fan, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. Learning to teach with dynamic loss functions. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 6467–6478, Red Hook, NY, USA, 2018. Curran Associates Inc.

[62] Flood Sung, Li Zhang, Tao Xiang, Timothy M. Hospedales, and Yongxin Yang. Learning to learn: Meta-critic networks for sample efficient learning. *arxiv*, abs/1706.09529, 2017.

[63] Wei Zhou, Yiying Li, Yongxin Yang, Huaimin Wang, and Timothy M. Hospedales. Online meta-critic learning for off-policy actor-critic methods, 2020.

[64] Jun Shu, Qian Zhao, Keyu Chen, Zongben Xu, and Deyu Meng. Learning adaptive loss for robust learning with noisy labels. *arXiv preprint arXiv:2002.06482*, 2020.

[65] Boyan Gao, Henry Gouk, and Timothy M. Hospedales. Searching for robustness: Loss learning for noisy classification tasks, 2021.

[66] Francesco Marchetti, Sabrina Guastavino, Michele Piana, and Cristina Campi. Score-oriented loss (sol) functions, 2021.

[67] Santiago Gonzalez and Risto Miikkulainen. Improved training speed, accuracy, and data utilization through loss function optimization. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, 2020.

[68] Santiago Gonzalez and Risto Miikkulainen. Optimizing loss functions through multi-variate taylor polynomial parameterization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '21, page 305–313. Association for Computing Machinery, 2021.

[69] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Proc. of NIPS*, pages 4768–4777, 2017.

[70] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why Should I Trust You?": Explaining the predictions of any classifier. In *Proc. of ACM SIGKDD*, page 1135–1144, 2016.

[71] M. Korobov and K. Lopuhin. ELI5 is a python library - v. 0.11, 2021. Available at (accessed 26/10/2021): https://eli5.readthedocs.io/en/latest/.

[72] Grégoire Montavon, Alexander Binder, Sebastian Lapuschkin, Wojciech Samek, and Klaus-Robert Müller. *Layer-Wise Relevance Propagation: An Overview*, pages 193–209. Springer International Publishing, 2019.

[73] Shoaib Ahmed Siddiqui, Dominique Mercier, Mohsin Munir, Andreas Dengel, and Sheraz Ahmed. TSViz: Demystification of deep learning models for time-series analysis. *IEEE Access*, 7:67027–67040, 2019.

[74] Hendrik Strobelt, Sebastian Gehrmann, Hanspeter Pfister, and Alexander M. Rush. LSTMVis: A tool for visual analysis of hidden state dynamics in recurrent neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):667–676, 2018.

[75] Hendrik Strobelt, Sebastian Gehrmann, Michael Behrisch, Adam Perer, Hanspeter Pfister, and Alexander M. Rush. Seq2seq-Vis: A visual debugging tool for sequence-to-sequence models. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):353–363, 2019.

[76] Hang Qiu, Ioanna Vavelidou, Jian Li, Evgenya Pergament, Pete Warden, Sandeep Chinchali, Zain Asgar, and Sachin Katti. ML-EXray: Visibility into ML deployment on the edge, 2021.

[77] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *ICLR*, Apr 2014.

[78] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv*, 2014.

[79] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv*, 2017.

[80] Gautam Raj Mode and Khaza Anuarul Hoque. Adversarial examples in deep learning for multivariate time series regression. In *Proc. of IEEE AIPR*, pages 1–10, 2020.

[81] Tianhang Zheng and Baochun Li. Poisoning attacks on deep learning based wireless traffic prediction. In *Proc. of IEEE INFOCOM*, pages 1–10, 2022.

[82] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[83] Yarin Perry, Felipe Vieira Frujeri, Chaim Hoch, Srikanth Kandula, Ishai Menache, Michael Schapira, and Aviv Tamar. DOTE: Rethinking (predictive) WAN traffic engineering. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 1557–1581, Boston, MA, April 2023. USENIX Association.

[84] Qianru Sun, Yaoyao Liu, Tat-Seng Chua, and Bernt Schiele. Meta-transfer learning for few-shot learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[85] Boris Oreshkin, Dmitri Carpo, Nicolas Chapados, and Yoshua Bengio. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. 04 2020.

[86] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3):1181–1191, 2020.

[87] Vincent Sitzmann, Julien N.P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Proc. Adv. in Neural Inform. Processing Syst. (NeurIPS)*, 2020.

[88] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3D-structure-aware neural scene representations. In *Proc. Adv. in Neural Inform. Processing Syst. (NeurIPS)*, 2019.

[89] Vishwanath Saragadam, Daniel LeJeune, Jasper Tan, Guha Balakrishnan, Ashok Veeraraghavan, and Richard G Baraniuk. WIRE: Wavelet implicit neural representations. In *Proc. IEEE/CVF Conf. Comput. Vis. and Pattern Recog. (CVPR)*, pages 18507–18516, 2023.

[90] Filip Szatkowski, Karol J. Piczak, Przemysław Spurek, Jacek Tabor, and Tomasz Trzciński. HyperSound: Generating implicit neural representations of audio signals with hypernetworks. In *Proc. NeurIPS Workshop on Meta-Learning*, 2022.

[91] Hao Chen, Bo He, Hanyu Wang, Yixuan Ren, Ser Nam Lim, and Abhinav Shrivastava. NeRV: Neural representations for videos. In *Proc. Adv. in Neural Inform. Processing Syst. (NeurIPS)*, 2021.

[92] Emilien Dupont, Hrushikesh Loya, Milad Alizadeh, Adam Golinski, Yee Whye Teh, and Arnaud Doucet. COIN++: Neural compression across modalities. *Trans. Machine Learning Research*, 2022.

[93] Leslie N Smith. Cyclical learning rates for training neural networks. In *2017 IEEE winter conference on applications of computer vision (WACV)*, pages 464–472. IEEE, 2017.

[94] Cristina Marquez, Marco Gramaglia, Marco Fiore, Albert Banchs, Cezary Ziemlicki, and Zbigniew Smoreda. Not all apps are created equal: Analysis of spatiotemporal heterogeneity in nationwide mobile service usage. In *Proceedings of the 13th International Conference on Emerging Networking EXperiments and Technologies*, CoNEXT '17, pages 180–186. Association for Computing Machinery, 2017.

[95] Rob Mulla. Pjm hourly energy consumption data (version 3). 2018.

[96] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The m4 competition: Results, findings, conclusion and way forward. *International Journal of Forecasting*, 34(4):802–808, 2018.

[97] Roger Koenker and Kevin F. Hallock. Quantile regression. *Journal of Economic Perspectives*, 15(4):143–156, December 2001.

[98] Chen Huang, Shuangfei Zhai, Walter Talbott, Miguel Bautista Martin, Shih-Yu Sun, Carlos Guestrin, and Josh Susskind. Addressing the loss-metric mismatch with adaptive loss alignment. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2891–2900. PMLR, 09–15 Jun 2019.

[99] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The M4 Competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1):54–74, 2020. M4 Competition.

[100] Slawek Smyl. A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting*, 36(1):75–85, 2020. M4 Competition.

[101] Juliver Gil Herrera and Juan Felipe Botero. Resource Allocation in NFV: A Comprehensive Survey. *IEEE Transactions on Network and Service Management*, 13(3):518–532, Sep. 2016.

[102] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[103] IEEE guide for electric power distribution reliability indices. *IEEE Std 1366-2012 (Revision of IEEE Std 1366-2003)*, pages 1–43, 2012.

[104] Enrique Personal, Juan Ignacio Guerrero, Antonio Garcia, Manuel Pena, and Carlos Leon. Key performance indicators: A useful tool to assess smart grid goals. *Energy*, 76:976–988, 2014.

[105] W.J. Harder. Key performance indicators for smart grids. Master's thesis, University of Twente, 2017.

[106] James Nightingale, Pablo Salva-Garcia, Jose M. Alcaraz Calero, and Qi Wang. 5g-qoe: Qoe modelling for ultra-hd video streaming in 5g networks. *IEEE Transactions on Broadcasting*, 64(2):621–634, 2018.

[107] 3GPP. Architecture enhancements for 5g system (5gs) to support network data analytics services (3gpp ts 23.288 version 16.4.0 release 16). 2020.

[108] Zhiqing Tang, Fuming Zhang, Xiaojie Zhou, Weijia Jia, and Wei Zhao. Pricing model for dynamic resource overbooking in edge computing. *IEEE Transactions on Cloud Computing*, 2022. early access.

[109] Daniel Kahneman and Amos Tversky. Prospect theory: An analysis of decision under risk. In *Handbook of the fundamentals of financial decision making: Part I*, pages 99–127. World Scientific, 2013.

[110] 3GPP TS 28.533 v16. Management and Orchestration of Networks and Network Slicing; Management and Orchestration Architecture (Rel. 16), June 2019.

[111] 3GPP TS 29.517 v16.2.0. 5G System; Application Function Event Exposure Service; Stage 3 (Rel. 16), March 2020.

[112] Line M. P. Larsen, Aleksandra Checko, and Henrik L. Christiansen. A survey of the functional splits proposed for 5g mobile crosshaul networks. *IEEE Communications Surveys & Tutorials*, 21(1):146–172, 2019.

[113] Rajkarn Singh, Cengis Hasan, Xenofon Foukas, Marco Fiore, Mahesh K. Marina, and Yue Wang. Energy-efficient orchestration of metro-scale 5g radio access networks. In *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, pages 1–10, 2021.

[114] Paola Soto, Danny De Vleeschauwer, Miguel Camelo, Yorick De Bock, Koen De Schepper, Chia-Yu Chang, Peter Hellinckx, Juan F. Botero, and Steven Latré. Towards autonomous VNF auto-scaling using deep reinforcement learning. In *International Conference on Software Defined Systems (SDS)*, pages 01–08, 2021.

[115] David S. Johnson. *Near-optimal bin packing algorithms*. PhD thesis, Massachusetts Institute of Technology, 1973.

[116] Soha Rawas. Energy, network, and application-aware virtual machine placement model in SDN-enabled large scale cloud data centers. *Multimedia Tools and App.*, 80(10):15541–15562, 2021.

[117] Gianni Barlacchi, Marco De Nadai, Roberto Larcher, Antonio Casella, Cristiana Chitic, Giovanni Torrisi, Fabrizio Antonelli, Alessandro Vespignani, Alex Pentland, and Bruno Lepri. A multi-source dataset of urban life in the city of Milan and the province of Trentino. *Scientific data*, 2015.

[118] I. Alawe, A. Ksentini, Y. Hadjadj-Aoul, and P. Bertin. Improving traffic forecasting for 5G core network scalability: A machine learning approach. *IEEE Network*, 32(6):42–49, Nov 2018.

[119] Sebastian Troia, Gao Sheng, Rodolfo Alvizu, Guido Alberto Maier, and Achille Pattavina. Identification of tidal-traffic patterns in metro-area mobile networks via matrix factorization based model. In *Proc. of IEEE PerCom Workshops*, pages 297–301, 2017.

[120] D. Bega, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Pérez. DeepCog: Optimizing resource provisioning in network slicing with AI-based capacity forecasting. *IEEE JSAC*, 38(2):361–376, 2020.

[121] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting?, 2022.

[122] Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting, 2024.

[123] Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y. Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and Qingsong Wen. Time-llm: Time series forecasting by reprogramming large language models, 2024.