Nguyen, Tri; Truong, Linh; Arcaini, Paolo; Ishikawa, Fuyuki

Optimizing Multiple Consumer-specific Objectives in End-to-End Ensemble Machine Learning Serving

Published in:
2024 IEEE/ACM 17th International Conference on Utility and Cloud Computing (UCC)

Accepted/In press: 05/11/2024

Document Version
Peer-reviewed accepted author manuscript, also known as Final accepted manuscript or Post-print

Please cite the original version:
Nguyen, T., Truong, L., Arcaini, P., & Ishikawa, F. (in press). Optimizing Multiple Consumer-specific Objectives in End-to-End Ensemble Machine Learning Serving. In *2024 IEEE/ACM 17th International Conference on Utility and Cloud Computing (UCC)*

Optimizing Multiple Consumer-specific Objectives in End-to-End Ensemble Machine Learning Serving

Minh-Tri Nguyen, Hong-Linh Truong
Department of Computer Science, Aalto University
Espoo, Finland
Email: {tri.m.nguyen, linh.truong}@aalto.fi

Paolo Arcaini, Fuyuki Ishikawa
National Institute of Informatics
Tokyo, Japan
Email: {arcaini, f-ishikawa}@nii.ac.jp

***Abstract*—Optimizing the quality of machine learning (ML) services for individual consumers with specific objectives is crucial for improving consumer satisfaction. In this context, end-to-end ensemble ML serving (EEMLS) faces many challenges in selecting and deploying ensembles of ML models on diverse resources across the edge-cloud continuum. This paper provides a method for evaluating the runtime performance of inference services via consumer-defined metrics. We enable ML consumers to define high-level metrics and consider consumer satisfaction in estimating service costs. Moreover, we introduce a time-efficient ensemble selection algorithm to optimize the EEMLS with intricate trade-offs between service quality and costs. Our intensive experiments demonstrate that the algorithm can be executed periodically despite the extensive search space, enabling dedicated optimization for individual consumers in dynamic contexts.**

***Index Terms*—ML Serving, Ensemble Selection, Ensemble ML, End-to-End ML, Performance Evaluation**

I. Introduction

End-to-end ensemble ML serving (EEMLS) has been increasingly deployed in various application domains, utilizing multiple ML models to enhance predictive performance and generalization [1]. Despite such advantages, using ensembles of ML models complicates the optimization while coordinating multiple models and considering trade-offs between service quality and costs. In many cases, the optimization objectives of ML consumers are represented by distinct metrics in specific contexts.

We consider the above-mentioned objectives when ML providers deploy and optimize *dedicated ensembles* of ML models for individual consumers at runtime to enhance consumer satisfaction. First, evaluating the quality of EEMLS is challenging. Different consumers can define multiple high-level metrics, called *consumer-defined metrics* (CDMs), to indicate their optimization objectives. That enables the EEMLS to tailor service quality to individual consumers, ensuring more effective and relevant outcomes to consumer expectations. Currently, most EEMLS platforms support common metrics, such as inference accuracy, response time, and resource usage, but not CDMs [2]. Second, selecting/deploying optimal ensembles is time-consuming due to: (1) The large search space within the massive number of ML models and computing resources in the edge-cloud continuum. (2) Estimating service costs based on consumer satisfaction can significantly increase the complexity of the optimization when considering numerous trade-offs between CDMs and costs. (3) The performance variability of services and ML models in EEMLS across different runtime contexts further complicates the optimization.

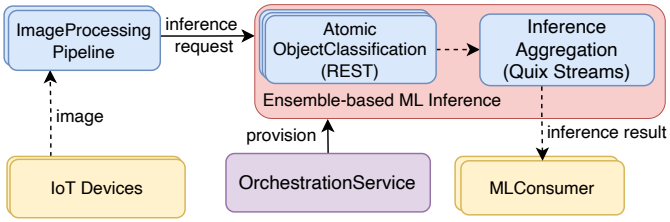

Fig. 1. An EEMLS for Object Classification in Autonomous Driving (OCAD).

To tackle the above-mentioned challenges, this paper presents an approach for optimizing the EEMLS following multiple consumer objectives. Our contributions are:

- We present a method for evaluating the performance of individual inference services based on CDMs. Specifically, we provide a metric specification and develop an evaluation engine to support calculating CDMs from real-time monitoring.
- We introduce a time-efficient algorithm for selecting ensembles and resources following consumer-specific objectives. The algorithm enables continuous quality optimization for individual consumers within dynamic runtime contexts.
- We present analyses on the optimization results in real-world scenarios. We compare our optimization method with the prevalent method used in state-of-the-art EEMLS platforms, including Cocktail [2] and FrugalML [3].

The rest of this paper is organized as follows: Section II introduces our running example. Sections III and IV present our method for evaluating inference services and an ensemble selection algorithm for EEMLS optimization. Section V and VI explain our experiment and related work. Finally, we conclude the paper in Section VII.

II. Running example & Optimization Framework

In Fig. 1, we illustrate a real-world EEMLS designed for *Object Classification in Autonomous Driving* (OCAD). It is a representative ML application deployed in edge-cloud systems, providing classification services to multiple consumers with varying runtime contexts. These contexts represent multiple utilization purposes of OCAD. For example, analyzing the operating environments (e.g., pedestrian and vehicle density)

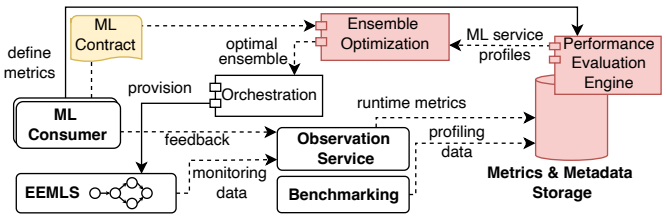

Fig. 2. The workflow and essential components in EEMLS optimization.

around autonomous cars and delivery robots with IoT devices. Depending on the quality requirements, the *OrchestrationService* provisions an optimal ensemble of *AtomicObjectClassification* (*AOC*) as an ensemble of multiple inference services. Services in this OCAD are deployed as microservices across edge-cloud resources. The *AOC*s provide object classification services utilizing 9 variants of VGG16 models whose accuracy is optimized for classifying several specific objects among 13 classes (e.g., pedestrians, cars, and bicycles). We obtained these models and datasets from our collaboration project on autonomous driving. OCAD is a real-world example where the *Ensemble-based ML Inference* must be frequently optimized to maintain the service quality for individual consumers.

From our prior work [4], two components have been developed to support the EEMLS deployments: the *Observation Service* for collecting metrics from the EEMLS system and consumer (*feedback* on service quality) and the *Benchmarking* component is responsible for profiling the inference services before runtime deployments, as shown in Fig. 2. This paper presents new *Performance Evaluation Engine* (built atop the *Metrics & Metadata Storage*) and *Ensemble Optimization* to evaluate and select the optimal ensembles.

III. Runtime Performance Evaluation Method Using Consumer-Defined Metrics

A. Consumer-defined metric definition and specification

1) Definition of Consumer-defined metric (CDM): A CDM is a high-level metric aggregated from other metrics within a certain context. A context refers to a specific utilization purpose of a consumer. The shifts in utilization cause context changes in runtime (or dynamic runtime context). Then, CDMs are performance indicators, allowing consumers to specify the quality requirements and optimization objectives. CDMs are used in runtime performance evaluation and quality estimation, supporting the ensemble selection. Optimizing CDMs, EEMLS can be more responsive to real-time changes for the specific and dynamic utilization of consumers.

Generally, the consumers can define a metric $cdm_i = f_i(M_i \times C_i)$ where f_i is an aggregation function (e.g., sum, max, or consumer's specialized functions). $M_i = \{m_{i1}, \ldots, m_{in}\}$ is a set of input metrics. Since metrics in M_i can be measured in different contexts with different meanings, $C_i = \{c_{i1}, \ldots, c_{in}\}$ is a set of runtime contexts corresponding to the metrics in M_i. A context c_{ij} includes multiple attributes. Each attribute in c_{ij} imposes the utilization condition where m_{ij} is measured. In the simplest case, a CDM can be directly mapped to a common metric without considering context.

2) Specification of CDM : We design a metric specification for consumers to specify complex CDMs. As illustrated in Fig. 3, the consumer can specify a reference to the aggregation $function$ (e.g., function name) and the $input$ when defining a CDM. The $input$ is a list of $\{metric, value, context\}$. The $metric$ (m_{ij}) indicates the metric name and the $value$ can be a specific value or value range (in case the consumers only consider the metrics in a specific value/range). Then, each $context$ (c_{ij}) includes a list of $\{attribute, value\}$, representing a consumer-specific utilization. Our metric specification allows real-time integration/modification of CDMs without significant engineering.

3) Example: Fig. 3 provides an excerpt of the metric specification in OCAD and illustrates the dependency between CDMs and runtime contexts. A consumer defines an *error risk* metric $errorRisk2$ as the function $ER2$ applied on several *miss rates*, i.e., $missRateOfRider$ and $missRateOfCarAsTruck$ (as other CDMs). The *miss rates* are aggregated based on *inference_accuracy* (a common metric) in specific contexts. Here, the contexts indicate the consumer's utilization for classifying *cars* and *trucks* (not the overall *inference_accuracy*). The attributes of the contexts are: *label_class* reported from consumer feedback and *predicted_class* monitored from the EEMLS system.

B. Evaluating inference services based on CDMs

The Evaluation Engine evaluates inference services based on the following steps: (1) Loading all aggregation functions specified in the CDM specification; such functions are consumer-specific implementations or common ones implemented in the engine. (2) Obtaining the metrics and executing the functions (f_i) with corresponding input for calculating individual CDMs, which will be added/updated to the inference service profiles. (3) Filtering the profiles to exclude inference services that do not satisfy the ML contract.

Our engine provides a *configurable evaluation window*, allowing consumers to specify ranges for obtaining metrics to evaluate the performance of inference services (e.g., within a specific time period or a number of recent inferences). Thus, the engine can capture the most relevant monitoring data for calculating CDMs. That enables real-time adjustments to consumer-specific objectives, providing more responsive and effective optimization tailored to the current needs and priorities of individual consumers. By filtering unqualified inference services, the engine also significantly decreases the search space for the ensemble selection algorithm.

IV. Time-efficient Algorithm for Selecting Ensemble in EEMLS Optimization

A. Overview of the ensemble selection algorithm

Let us describe a naive algorithm (in Algorithm 1) whose variables are explained in Table I. An EM is a combination of multiple ML models that work together to improve the inference performance. An ED is a specific deployment of an EM on specific resources (list of inference services). Since no prior work has incorporated CDMs into cost estimation

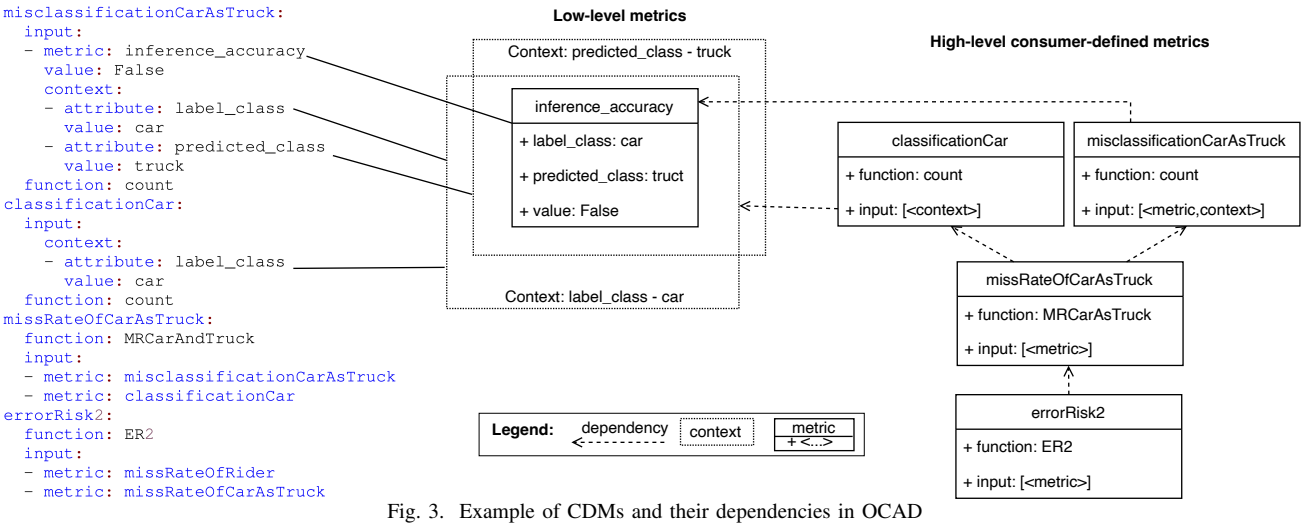

Fig. 3. Example of CDMs and their dependencies in OCAD

Algorithm 1 Naive ensemble selection algorithm

Require: $\mathcal{M}$: $\mathcal{M} = \{\mathfrak{m}_1, \ldots, \mathfrak{m}_m\}$ set of ML models
Require: $\mathcal{D}$: $\mathcal{D} = \{d_1, \ldots, d_d\}$ set of available resources
Require: k: number of ML models in the ensemble
Require: P: the minimum throughput requirement
1: **procedure** EnsembleDeployment($\mathcal{M}$, $\mathcal{D}$, k, P)
2: **for** each EM of k from $\mathcal{M}$ **do**
3: # Find all $ED(s)$ of EM on $\mathcal{D}$ satisfying P
4: $LED \leftarrow$ FindDeployement(EM, $\mathcal{D}$, P)
5: **for** each ED of LED **do**
6: $ES \leftarrow$ ObjectiveFunction(ED)
7: **Store** ES **for this** ED
8: **end for**
9: **end for**
10: **Select** ED **with the highest** ES
11: **end procedure**

and quality optimization, we use the naive algorithm as the baseline for the complexity evaluation. The algorithm includes:

- **Step 1**: Selecting individual $EM(s)$ consisting of k out of m ($m = |\mathcal{M}|$) number of available models (line 2).
- **Step 2**: Finding all $ED(s)$ (or LED) of each EM on available resources $\mathcal{D}$, satisfying throughput requirement P (line 4). Since all ML models in the ensemble must perform the same number of inferences in parallel, they must be deployed with reasonable numbers of replicas (or *scales*) to ensure the throughput satisfies P.
- **Step 3**: Scoring each ED as ES (line 6). We estimate consumer satisfaction with service quality/cost by applying an objective function on the performance of the ED.
- **Step 4**: Selecting the optimal ED (line 10). With consumer satisfaction quantified by a numerical value (ES) for all possible $ED(s)$ of all $EM(s)$ (in **steps 1-3**), we select the optimal ED that achieves the highest ES.

1) Complexity of the naive algorithm: Let's assume that the ML provider has $\mathcal{M} = \{\mathfrak{m}_1, \ldots, \mathfrak{m}_m\}$ ML models and $\mathcal{D} = \{d_1, \ldots, d_d\}$ types of resources. The consumers expect

TABLE I
VARIABLE DESCRIPTION IN ALGORITHM 1

Variable	Description
EM	Ensemble of ML models
ED	Ensemble deployment - A specific deployment of an EM
LED	List of ensemble deployments - List of $ED(s)$
ES	Ensemble score - The score of an ensemble

the EEMLS to utilize a minimum of k_{min} to a maximum of k_{max} ML models. Then, the complexity of the naive algorithm depends on the two first steps. In **Step 1**, we must select combinations of $k \in \{k_{min}, \ldots, k_{max}\}$ out of m models. The number of possible $EM(s)$ is $\sum_{k_{min}}^{k_{max}} {}^{m}C_k$. In **Step 2**, we must estimate the number of replicas (referred to as *scales*) of ML models within each EM and find all possible deployments of these replicas on $\mathcal{D}$, as distinct $ED(s)$. Let S be a set of scales of k ML models in an EM, we have $S = \{s_1, \ldots, s_k\}$ where s_i is estimated as the minimum scale (in the worst-case) of the model $\mathfrak{m}_i$ to satisfy the throughput requirement P. Note that an ED is a specific deployment of an EM on specific resources, so an ED is also a list of inference services. The number of inference services in each ED will be $\sum S$. With $d = |\mathcal{D}|$ types of resources, the number of possible $ED(s)$ for each EM is $d^{\sum S}$. Combining **Step 1** and **Step 2**, the total $ED(s)$ reaches approximately $\sum_{k_{min}}^{k_{max}} {}^{m}C_k \times d^{\sum S}$.

2) Objective function: A significant challenge in EEMLS optimization arises when managing trade-offs among multiple CDMs and service costs. In **Step 3**, we use an objective function to estimate consumer satisfaction with service quality and cost. Output of the function is a number: $ES = quality\_score + cost\_score$. We define $quality\_score = \sum \alpha_i \times f^i_{score}(cdm_i)$ and $cost\_score = \beta \times f^{cost}_{score}(ml\_cost)$. Here, cdm_i is a CDM obtained by estimating the performance of an ED (Section IV-B). The f^i_{score} is a function that maps values of cdm_i to the range [0,1], reflecting consumer satisfaction with the metric. This way, the $quality\_score$ is a quantification of the service quality from the consumer perspective. In the $cost\_score$, f^{cost}_{score} is also a function that maps the service cost (ml_cost in Section IV-C) to the range [0,1], reflecting consumer satisfaction with the service cost. Currently, we provide linear and logarithmic functions for the

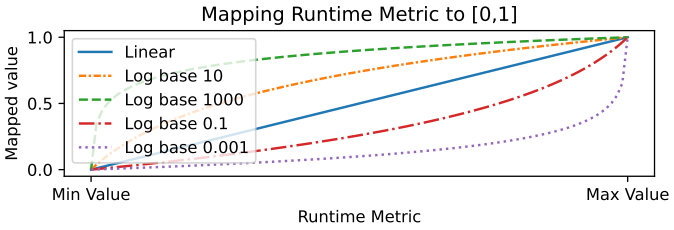

Fig. 4. Mapping metric following linear and logarithmic scales.

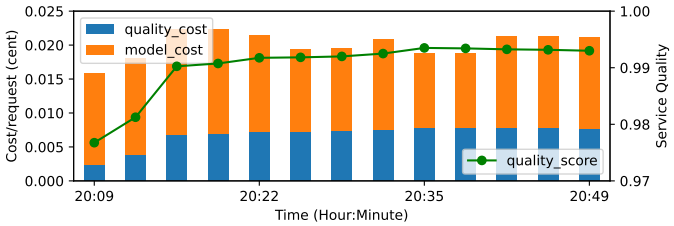

Fig. 5. The patterns of service quality and cost in OCAD.

mapping (explained in Section IV-C). Consumers can specify α_i and β as weighted factors to prioritize the metrics or service cost in their trade-offs. These weighted factors are currently fine-tuned based on heuristic observations.

B. Estimating performance of an ensemble deployment

We need to evaluate the performance of individual $ED(s)$ to calculate the $quality\_score$ and $cost\_score$. Before that, we must identify and evaluate the performance of all possible inference services (by deploying $\mathcal{M}$ on $\mathcal{D}$ as $\mathcal{M} \times \mathcal{D}$). Any ED including unqualified inference services will be excluded. Since precise performance estimation of an ED can be challenging, we estimate performance in the worst scenario using common aggregation methods (e.g., max, min, and $average$). That ensures the performance of the $ED(s)$ always satisfies consumer requirements at runtime. The estimation results are aggregated CDMs, indicating the performance of an ED when combining certain inference services.

C. Estimating cost of an ensemble deployment

In our work, we only consider ml_cost as the sum of $model\_cost$ (the cost of utilizing ML models) and $quality\_cost$ (the cost for certain inference quality, representing consumer satisfaction). Specifically, the $model\_cost$ is static and available on the marketplaces [5]. However, the $quality\_cost$ is dynamic, estimated based on metrics observed on each inference and the current performance of associated inference services. To estimate the $quality\_cost$, we map the values of each cdm_i to $mValue_i \in [0, 1]$, using a logarithmic or linear function, illustrated in Fig. 4. The consumer and provider will agree upon the mapping function, the expected value range (maximum/minimum), and the maximum cost ($qCost_i^{max}$) that the consumer must pay for the quality indicated by cdm_i. The cost incurred by cdm_i is $qCost_i$, that equals to $qCost_i^{max} \times mValue_i$ if cdm_i needs to be maximized, or $qCost_i^{max} \times (1 - mValue_i)$ if cdm_i needs to be minimized. The $qCost_i$ being 0 indicates that the inference quality on cdm_i does not meet the consumer minimum requirement. If the $qCost_i$ equals to $qCost_i^{max}$, the inference quality on cdm_i completely meets the consumer requirement. Eventually, the $quality\_cost$ of an inference is the sum of all $qCost_i$ corresponding to all CDMs on all inference services.

D. Scale Reduction Algorithm

We notice that the reason for the rapid-expanding search space in the naive algorithm is the increase of the scale S when increasing P. Therefore, we present the scale reduction algorithm to enhance the naive algorithm by reducing S.

In **Step 2**, we need to perform two tasks: estimating the scale S and selecting resources to deploy the ML models. To reduce the number of $ED(s)$, we divide the scales of all ML models by the minimum value in S. Specifically, $S_{reduced} = ceil(S/min(S))$. For example, if $S = \{15, 20, 25, 22\}$, $min(S)$ is 15 and $S_{reduced} = ceil(S/15) = \{1, 2, 2, 2\}$. We proceed with resource selection and calculate the ES as usual with the new $S_{reduced}$. This way, we reduce $\sum S = 82$ to $\sum S_{reduced} = 7$. Thus, the number of $ED(s)$ has decreased exponentially. After selecting the optimal ED, as the result is a list of inference services associated with $S_{reduced}$, these inference services must be scaled up by a factor of $min(S)$ (used previously) to satisfy P. When we group these inference services by their ML models, we obtain the new scales as $S_{up} = min(S) \times S_{reduced}$. Using the previous example, $S_{up} = \{15, 30, 30, 30\}$. Then, the scales of most ML models in S_{up} will be larger than their original versions in S, causing throughput redundancy. Hence, for each ML model, we remove redundant inference services in descending order of ml_cost. The remaining inference services are the output of the *Ensemble Optimization*.

V. Experiments

We conduct experiments with the OCAD running example (Section II) on multiple devices of Raspberry Pi 4, Jetson Nano, Jetson Xavier AGX, Beelink BT3, Thinkstation P620, and VMs on GCP (Google Cloud Platform) with different configurations. The dataset is open under our git repository[1].

A. Experiment 1 - Evaluating the EEMLS via CDMs & optimizing the EEMLS in real-time

We experiment with a consumer whose objective is to maximize inference accuracy in classifying cars and minimize the miss rate in misclassifying cars as trucks. We simulate the dynamic input data by randomly changing the object distribution (the number of images belonging to different object classes within a specific period) every 10 minutes. Then, we monitor the service cost and quality (via $quality\_score$).

1) Evaluating EEMLS: We evaluate the performance of individual inference services from the consumer perspective. In Fig. 5, we illustrate the service quality of the EEMLS via the $quality\_score$. At the beginning, the service quality is relatively low due to the use of randomly initialized ED and insufficient monitoring data for the evaluation. Later, as the EEMLS is optimized according to CDMs within a specific

[1]https://github.com/rdsea/ROHE/tree/main/datasets/UCC2024

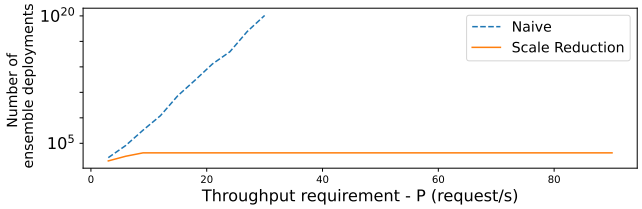

Fig. 6. The reduction in the number of ensemble deployments (search space) of the ensemble selection algorithms.

runtime context, the *quality_score* remains consistently high from the third data point. Then, the *quality_cost* follows the same pattern, optimized for consumer-specific objectives via CDMs and reflecting consumer satisfaction.

2) Optimizing the EEMLS in real-time: In Fig. 6, we report the number of possible $ED(s)$ as the throughput requirement (P) increases. With larger P, the scales of ML models increase, and the potential deployments across different types of resources also multiply. This number in the naive algorithm quickly reaches 10^{20}. With the scale reduction algorithm, the scales (S) of ML models in the ensemble are always divided by the smallest scale. The sum of the reduced scales ($\sum S_{reduced}$) does not change considerably and has an upper bound. That keeps the number of $ED(s)$ under 10^5, allowing our optimization to be performed frequently (short execution).

B. Experiment 2 - Optimizing EEMLS for individual consumers based on consumer-specific objectives

This experiment has two consumers. The objective of *Consumer 1* is to maximize accuracy in classifying cars, maximize confidence in classifying cars, and minimize the miss rate in misclassifying cars as trucks. The objective of *Consumer 2* is to maximize accuracy in classifying pedestrians, maximize confidence in classifying pedestrians, and minimize the miss rate in misclassifying pedestrians as riders. We compare *our optimization* (**C**onsumer-specific **O**ptimization - *CO*) and the ***P**revalent **O**ptimization* (*PO*) used in state-of-the-art EEMLS platforms [2], [3]. The CO optimizes the EEMLS based on consumer satisfaction with service quality via CDMs and cost within specific runtime contexts, and PO optimizes the EEMLS based on overall accuracy and cost. For simulating real-world scenarios, we vary the object distribution to reflect different time periods of a day when varying numbers of objects. The input object distribution changes every 10 minutes, and the request frequency is 5 req/s. All metrics are evaluated over the latest 1500 requests.

In Figs. 7 and 8, we illustrate the optimization effectiveness for the two consumers on different metrics, observed from *CO* (• marked lines) and *PO* (× marked lines). The *overall accuracy* in both optimizations fluctuates as the object distribution constantly changes. With EEMLS optimized based on consumer-specific objectives, the *overall accuracy* in *CO* drops several times for *Consumer 1* because the input data has fewer car images in those periods, but it remains higher than 0.965. For *Consumer 2*, in the second half of the experiment, the *overall accuracy* in *CO* even surpasses *PO* where pedestrians dominate the distribution of the input data. Without consumer-specific objectives in *PO*, the risks of misclassifying are significant. By varying weighted factors (α_i and β), we achieve different optimization effects with high *accuracy* in classifying cars (Fig. 7) as well as pedestrians (Fig. 8) and reduce *miss rate*(s) close to 0 after 10 minutes. Therefore, the *CO* outperforms *PO* from the perspective of individual consumers.

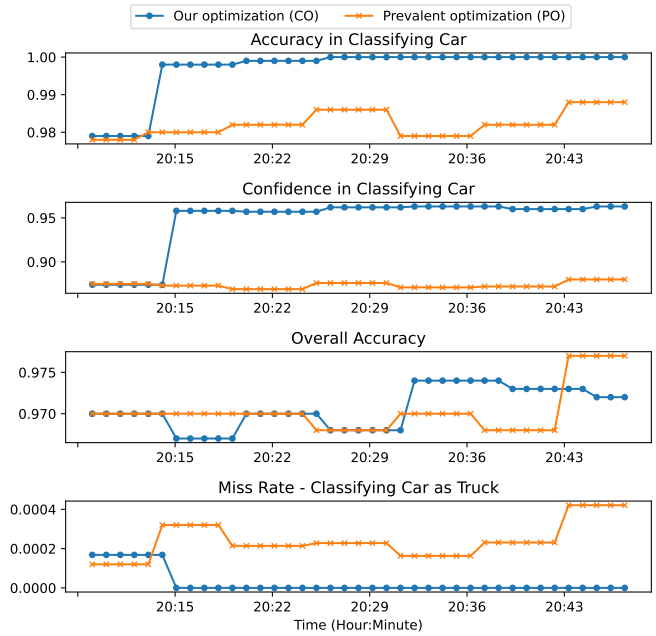

Fig. 7. Compare optimization effectiveness of the EEMLS for *Consumer 1* with objectives on classifying cars.

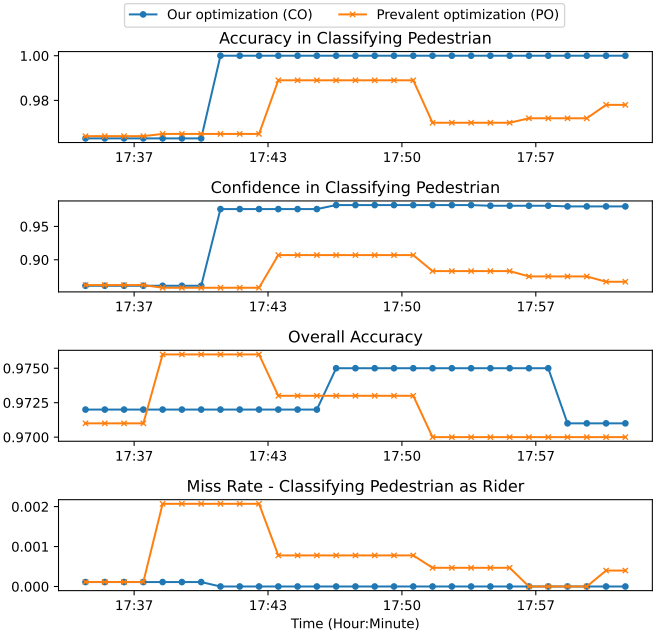

Fig. 8. Compare optimization effectiveness of the EEMLS for *Consumer 2* with objectives on classifying pedestrians.

VI. Related Work

Evaluating the quality of inference services using CDMs: In existing works, the optimization objectives are indicated via common metrics [6]. Most studies, such as [7] and [8], evaluate the quality of inference services based on throughput,

response time, and resource utilization. Like our study, Cocktail [2] employs ensembles of ML models but only focuses on accuracy and latency. Meanwhile, existing commercial platforms supporting EEMLS, like Azure ML Studio [9] and AWS Autogluon [10], support developers in monitoring end-to-end metrics obtained from common ML tools/libraries. However, the runtime contexts and metric aggregations have not been addressed in the quality/cost evaluation. Similarly, frameworks for MLaaS (e.g., Sinan [11], InFaas [12], and FrugalML [3]) limit the inference service quality to several common metrics. Thus, high-level CDMs have not been supported. All above-mentioned studies evaluate inference services for general purposes using common metrics defined by the ML provider, so they have not fully considered the consumer perspectives in the evaluation. In contrast, our work allows consumers to define high-level metrics within complex dependencies and contexts. *Optimizing the quality of EEMLS following consumer-specific objectives:* The most relevant works prior to ours are Cocktail [2] and FrugalML [3]. Specifically, Cocktail aims to reduce deployment costs and latency with the trade-off in accuracy, and FrugalML optimizes the ML utilization of multiple public inference services with budget constraints. In other studies, the service cost is estimated based on resource utilization without considering consumer satisfaction with service quality [1]. MLaaS platforms like Clipper [13] and Sagemaker [14] support ensemble ML serving with optimized latency and resource utilization without considering ML-specific metrics. Rafiki [15] is a framework focusing on the trade-off between accuracy and latency. Many public ML services like Google Vision [16] apply service costs for individual inferences, but the cost is fixed regardless of consumer satisfaction and runtime contexts. Here, we optimize the EEMLS by considering complex relations among multiple consumer-specific objectives and service costs in diverse runtime contexts.

VII. Conclusion

Optimizing service quality for individual consumers is essential for improving consumer satisfaction. However, this raises significant challenges in evaluating and optimizing the EEMLS, given the diversity of CDMs, ML models, and underlying computing resources within the edge-cloud continuum. In this paper, we introduce a metrics specification and evaluation engine to address these challenges. By this means, we present an efficient algorithm for optimizing EEMLS, where service costs are estimated based on consumer satisfaction with the delivered service quality. That is a step towards developing a comprehensive framework for EEMLS, ensuring reliability in service quality for multiple consumers across various domains while optimizing resource utilization for the providers. This framework enhances the responsiveness of EEMLS by enabling consumers to dynamically adjust/prioritize optimization objectives following varying contexts or serving conditions (e.g., in autonomous systems, manufacturing, and other IoT domains). Since the current optimization still depends on numerous configurations, assumptions about the systems, and consumer requirements, that opens many potential research directions for our future studies in automation optimization.

Acknowledgment

The authors would like to thank Nguyen Vuong for her support in developing the OCAD application. This work was partially supported by Engineerable AI Techniques for Practical Applications of High-Quality Machine Learning-based Systems Project (Grant Number JPMJMI20B8), JST-Mirai.

References

[1] I. D. Mienye and Y. Sun, "A survey of ensemble learning: Concepts, algorithms, applications, and prospects," *IEEE Access*, vol. 10, pp. 99 129–99 149, 2022.

[2] J. R. Gunasekaran, C. S. Mishra, P. Thinakaran, B. Sharma, M. T. Kandemir, and C. R. Das, "Cocktail: A multidimensional optimization for model serving in cloud," in *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, 2022, pp. 1041–1057.

[3] L. Chen, M. Zaharia, and J. Y. Zou, "FrugalML: How to use ML prediction APIs more accurately and cheaply," *Advances in neural information processing systems*, vol. 33, pp. 10 685–10 696, 2020.

[4] M. Nguyen and H. Truong, "On optimizing resources for real-time end-to-end machine learning in heterogeneous edges," *Software: Practice and Experience*, 2024.

[5] L. Chen, P. Koutris, and A. Kumar, "Towards model-based pricing for machine learning in a data marketplace," in *Proceedings of the 2019 International Conference on Management of Data*, 2019, pp. 1535–1552.

[6] H. Mo, L. Zhu, L. Shi, S. Tan, and S. Wang, "HetSev: Exploiting heterogeneity-aware autoscaling and resource-efficient scheduling for cost-effective machine-learning model serving," *Electronics*, vol. 12, no. 1, p. 240, 2023.

[7] J. Cho, D. Zad Tootaghaj, L. Cao, and P. Sharma, "Sla-driven ml inference framework for clouds with heterogeneous accelerators," *Proceedings of Machine Learning and Systems*, vol. 4, pp. 20–32, 2022.

[8] N. Mahmoudi and H. Khazaei, "MLProxy: SLA-aware reverse proxy for machine learning inference serving on serverless computing platforms," *arXiv preprint arXiv:2202.11243*, 2022.

[9] Microsoft Azure, "Multiclass Decision Forest component ensemble machine learning serving in Azure ML studio," 2024, accessed on 24-08-2024. [Online]. Available: https://learn.microsoft.com/en-us/azure/machine-learning/component-reference/multiclass-decision-forest

[10] N. Erickson, J. Mueller, A. Shirkov, H. Zhang, P. Larroy, M. Li, and A. Smola, "AutoGluon-Tabular: Robust and accurate AutoML for structured data," *CoRR*, vol. abs/2003.06505, 2020. [Online]. Available: https://arxiv.org/abs/2003.06505

[11] Y. Zhang, W. Hua, Z. Zhou, G. E. Suh, and C. Delimitrou, "Sinan: ML-based and QoS-aware resource management for cloud microservices," in *Proceedings of the 26th ACM international conference on architectural support for programming languages and operating systems*, 2021, pp. 167–181.

[12] F. Romero, Q. Li, N. J. Yadwadkar, and C. Kozyrakis, "INFaaS: Automated model-less inference serving," in *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, 2021, pp. 397–411.

[13] D. Crankshaw, X. Wang, G. Zhou, M. J. Franklin, J. E. Gonzalez, and I. Stoica, "Clipper: A low-latency online prediction serving system," in *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, 2017, pp. 613–627.

[14] Hardt, Michaela et al., "Amazon SageMaker Clarify: Machine learning bias detection and explainability in the cloud," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, ser. KDD '21. New York, NY, USA: Association for Computing Machinery, 2021, pp. 2974–2983. [Online]. Available: https://doi.org/10.1145/3447548.3467177

[15] W. Wang, J. Gao, M. Zhang, S. Wang, G. Chen, T. K. Ng, B. C. Ooi, J. Shao, and M. Reyad, "Rafiki: machine learning as an analytics service system," *Proc. VLDB Endow.*, vol. 12, no. 2, pp. 128–140, oct 2018. [Online]. Available: https://doi.org/10.14778/3282495.3282499

[16] Google Cloud, "Computer vision extract insights from images, documents, and videos." 2023, accessed on 24-08-2024. [Online]. Available: https://cloud.google.com/vision