Optimizing Machine Learning Inference Queries for Multiple Objectives

Li, Ziyu; Schonfeld, Mariette; Hai, Rihan; Bozzon, Alessandro; Katsifodimos, Asterios

DOI
10.1109/ICDEW58674.2023.00017

Publication date
2023

Document Version
Final published version

Published in
Proceedings - 2023 IEEE 39th International Conference on Data Engineering Workshops, ICDEW 2023

2023 IEEE 39th International Conference on Data Engineering Workshops (ICDEW) | 979-8-3503-2244-6/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICDEW58674.2023.00017

Optimizing Machine Learning Inference Queries for Multiple Objectives

Ziyu Li* Mariette Schönfeld* Rihan Hai Alessandro Bozzon Asterios Katsifodimos
Delft University of Technology

***Abstract*—Given a set of pre-trained Machine Learning (ML) models, can we solve complex analytic tasks that make use of those models by formulating ML inference queries? Can we mitigate different tradeoffs, e.g., high accuracy, low execution costs and memory footprint, when optimizing the queries? In this work we present different multi-objective ML inference query optimization strategies, and compare them on their usability, applicability, and complexity. We formulate Mixed-Integer-Programming-based (MIP) optimizers for ML inference queries that makes use of different objectives to find Pareto-optimal inference query plans.**

I. Introduction

Machine learning (ML) is used in application domains such as video analytics [10], [23], autonomous driving [21], and content moderation [8]. With the advances in ML research, ML models become more accurate, often requiring both longer training times but also inference times, depending on the architecture of the model.

Combined with the trend for model reusability and reporting, pre-trained models are shared in the form of collections, oftentimes referred to as "model zoos" or "model hubs" such as Hugging Face and PyTorch Hub[1]. Typically these repositories structure the collections with *metadata*, such as the tasks that those models can cover (e.g., object classes that that they can detect within an image), the model's performance (speed, accuracy, recall, etc.) on these tasks, and other metadata. This makes model zoos accessible for users to search for models that suit their needs. Many ML-inference tasks cannot be answered using a single model. For instance, consider the task of retrieving images containing cars or person, `(car ∧ person)`. We term such a query a *machine learning inference query* [13], [22], in the form of Boolean expression consisting of *predicates* connected by conjunctions or disjunctions. Now consider the case where we have models answering either one of the predicates (`car` or `person`) or both (specialized models can be trained on a single class for higher efficiency while maintaining high accuracy). Note that it is possible that one model is selected for multiple predicates. In order to answer the query, we can assign one model per predicate, or one model to answer both.

When a model zoo is available, we could greedily select the most accurate models for each predicate, and evaluate the query. However, in the presence of a model zoo with models of different accuracy/cost characteristics we could sacrifice an acceptable amount of accuracy in order to reduce execution costs significantly. In addition, accuracy may not be the only optimization target for inference queries. For instance, in memory-limited GPUs, one may also want to minimize the model size (memory footprint), and/or the model's execution time. For instance, in edge devices, storage or memory limitations could have priority over accuracy due to limited resource capacity. For complex video queries, the execution time should be prioritized in order for the query to be able to keep up with the 30 frames per second required for real-time object analysis while maintaining accuracy with low cost.

This paper focuses with optimizing complex ML inference queries for multiple objectives. In the context of multi-objective optimization (MOO), a good query plan should balance accuracy and execution cost, as well as any other objectives deemed relevant by the users. Given the multitude of MOO techniques available [2], [14], we cannot choose a single best optimization method and user *preferences* (i.e. the importance of objectives in relation to one another other). In particular, this work explores ways to reconcile the conflicting demands of accuracy, execution time and memory footprint, and examines the efficacy of various MOO approaches in scenarios where objectives are in different preferences.

This work tackles the ML inference query optimization problem by performing model assignment (i.e., assigning models to predicates) from a given model zoo of pre-trained models, and compares several MOO methods on their suitability, usability, and complexity. In short, in this paper we contribute an analysis of applying different MOO methods for the problem of ML inference queries, and propose an optimizer that can generate pareto-optimal ML inference query plans for multiple objectives.

II. Related work

In this work, we tackle the problem of multi-objective ML inference query optimization, whereas existing works often only optimize for one objective during the ML inference process (Sec. II-A). Multi-objective optimization has been studied for database queries, but cannot be extended non-trivially to ML inference queries (Sec. II-B).

A. Single-objective ML inference query optimization

The optimization goal of ML pipelines is usually either efficiency or effectiveness. Recent studies [3], [6], [7], [11], [13], [18] train and infer large-scale ML models over the sheer volume of data, e.g., video and large text corpus. They target

[1]https://huggingface.co/, https://pytorch.org/hub/
*Authors have contributed equally to this work.

2473-3490/23/$31.00 ©2023 IEEE
DOI 10.1109/ICDEW58674.2023.00017

improving time-wise *efficiency*, while the accuracy is within a given threshold. NoScope [11] and PP [13] filtered irrelevant frames by training and deploying lightweight binary classifiers, and Tahoma [3] trained specialized models and constructed model cascades to process video frames. However, they only aim to increase the efficiency of query processing without considering any other objectives such as memory or storage footprint. These objectives are paramount in practice, e.g., model inference over edge devices. In this paper, we tackle the problem of ML inference query mitigating the tradeoffs among *multiple* objectives.

B. Multi-objective query optimization in databases

Multi-objective query plan optimization has been studied for database queries [4], [15]. Recent works calculate the Pareto boundary for query plans that are measured using multiple cost functions using the weighted sum method [19], or use the weighted sum method for optimization in mobile-cloud database environments [9]. These methods cannot be extended to optimize multi-objective ML inference queries, because their optimizers prioritize join-ordering and other query operations that are irrelevant to ML inference queries. We therefore need a different optimization algorithm as our basis.

III. Problem definition

We want to assign models to ML inference query predicates while balancing multiple objectives. We start with introducing the notions of *model zoos* and *ML inference queries*.

A. Model zoo

A model zoo is a repository that stores pre-trained ML models [1], [16], [17], [24] (in particular, we consider classification models). In this work, we focus on four types of metadata of a model zoo: inference classes (e.g., the classes that models can identify), accuracy, execution cost, and memory footprint. We formalize the model zoo and its metadata below.

Definition 1. *A **model zoo** M is a set of ML models, and its metadata is a tuple $< P, A, C, S >$, with*
– P is the set of classes the models in M can infer.
– $A \in [0,1]^M$ is a matrix of size $|M| \times |P|$ representing the accuracy of models $m \in M$ on classes $p \in P$.
– $C \in \mathbb{N}^{|M|}$ is a vector representing the execution cost of $m \in M$ in milliseconds. We assume this type of metadata is provided along with the execution environment.
– $S \in \mathbb{N}^{|M|}$ is a vector representing memory footprint of $m \in M$ in bytes.

Example 1. *Consider the query from the Sec. I with the following simplified model zoo example:*

Model name	C	S	A_{car}	A_{human}
model_1	10	3000	80%	0%
model_2	30	8000	95%	0%
model_3	20	5500	0%	85%
model_4	40	9000	0%	95%

TABLE I: A model zoo example

B. ML inference queries

Definition 2. *Given a model zoo M and its metadata $< P, A, C, S >$, an **ML inference query** over M is a first-order sentence of the form*

$$((m_{p_1} \wedge ... \wedge m_{p_j}) \vee ... \vee (m_{p_k} \wedge ... \wedge m_{p_l}))$$

with $p_i \in P$ and $m_{p_i} \in M$. In this work we only consider queries that are in Conjunctive Normal Form (CNF), i.e., conjunctions of disjunction or Disjunctive Normal Form (DNF).

Example 2. *Following the above definition, we reformulate the query from the introduction as follows:*

$$q := model\_2_{car} \wedge model\_4_{human} \tag{1}$$

In short, this means that *model_2* will be used to evaluate the predicate `car` and *model_4* will be used to evaluate the predicate `human`.

C. ML inference query optimization

Our goal is to optimize an ML inference query plan, such that we find an objective-balancing assignment of models from a model zoo to predicates. In this work we focus on three objectives: accuracy, execution cost, and memory footprint.

Example 3. *Continuing with Example 2, this model selection achieves 90.25% accuracy, execution cost of 70ms, and 17000B memory footprint, according to the model zoo in table I, calculated with the objective functions in section IV-A2.*

IV. Approach overview

We solve the ML inference query optimization problem, by formulating the model selection problem as a *Multi-Objective Mixed Integer Program* (MOMIP). An MIP is an optimization structure where the solution space is spanned up by a collection of (preferably linear) equations. An objective function allows optimization algorithms to navigate the solution space, finding the optimal solution [20].

A. MOMIP formulation

1) constraint formulation: We start with an MIP formulation of the model selection problem, where the decision variables are denoted as

$$X_{m,p} = \begin{cases} 1 & \text{when model } m \text{ is assigned to answer predicate } p \\ 0 & \text{elsewhere} \end{cases} \tag{2}$$

for $(m, p) \in M \times P$. To ensure that every predicate is answered by exactly one model, we impose the following constraint:

$$\sum_{m \in M} X_{m,p} = 1, \forall p \in P \tag{3}$$

In our definition of model zoo metadata a lack of ability for a model $m \in M$ to infer on a predicate $p \in P$ is indicated with $A_{m,p} = 0$. We want to avoid such model-predicate combinations, so we introduce the following bound:

$$X_{m,p} \leq \lceil A_{m,p} \rceil, \forall p \in P, m \in M \tag{4}$$

We also define an "indicator" variable B_m to identify whether model m is selected.: B_m is constrained as follows:

$$\begin{array}{rcl} B_m & \geq & X_{m,p} \\ B_m & \leq & \sum_{p \in P} X_{m,p}, \forall m \in M \end{array} \quad (5)$$

2) Objective functions: In this work we consider three objectives, accuracy f_{acc}, execution cost f_{cost}, and memory footprint f_{mem}.

Calculating accuracy. Consider an example DNF query, (`car` ∧ `outdoor`) ∨ (`chair` ∧ `indoor`). The clauses in the query is connected by disjunction. We first estimate the accuracy of the clauses, referred as c_1, c_2. We assume the predicates are independent to each other. Similar assumption has been made in [13]. The accuracy of a conjunctive query, e.g., `car` ∧ `outdoor`, can be estimated by multiplying the accuracy of each models, $a_{car} * a_{outdoor}$. The accuracy of a disjunctive query, e.g., `car` ∨ `bus`, can be computed using inclusion–exclusion principle, as $a_{car} + a_{bus} - a_{car} * a_{bus}$. Following the same manner, we can calculate the accuracy, f_{acc}, of more complex Boolean expressions.

In the canonical form of MOO problems all objectives either have to be minimized or maximized however. Therefore we introduce "accuracy loss":

$$f_{acc\_loss} = 1 - f_{acc} \quad (6)$$

Minimizing f_{acc_loss} is equivalent to maximizing f_{acc}.

Calculating execution cost. The second objective, execution cost f_{cost}, is obtained by summing the execution cost of all used models:

$$f_{cost}(q) = \sum_{m \in M} C_m B_m \quad (7)$$

Calculating memory. The third objective, memory footprint f_{mem} is obtained similarly. We assume all the models are loaded in advance before executing them.

$$f_{mem}(q) = \sum_{m \in M} S_m B_m \quad (8)$$

We derive the following MOMIP:

$$\begin{array}{ll} \min_{y \in Y} & [f_{acc\_loss}, f_{cost}, f_{mem}] \\ \text{such that} & (3), (4), (5) \text{ hold.} \end{array} \quad (9)$$

Where Y denotes the set of valid query plans.

3) Function normalization: Many MOO methods will prioritize optimizing the objectives that naturally take larger values (such as memory that ranges in the thousands, versus accuracy that ranges from 0 to 1). To avoid this 'preferential treatment' we use a normalization method [14]:

$$f_i^{norm}(y) = \frac{f_i(y) - f_i^{\min}}{f_i^{\max} - f_i^{\min}} \quad (10)$$

Which normalizes the objective functions f_i between the minimum and maximum obtainable value.

B. Multi-objective mixed integer optimization methods

Before we introduce the MOO methods, some important definitions need to be introduced [14]:

Definition 3. *A point $\mathbf{y} \in \mathbf{Y}$ is **Pareto optimal** iff there does not exist a point $\mathbf{y}^* \in \mathbf{Y}$ such that $f_i(\mathbf{y}^*) \leq f_i(\mathbf{y}) \, \forall \, 0 \leq i \leq k$ and $f_j(\mathbf{y}^*) < f_j(\mathbf{y})$ for some j. y is **weakly Pareto optimal** iff there does not exist a solution $\mathbf{y}^* \in Y$ such that $\mathbf{f}(\mathbf{y}^*) < \mathbf{f}(\mathbf{y})$.*

Example 4. *The query plan in Example 3 is Pareto optimal: with the models in example 1 we can only achieve lower cost or memory footprint by decreasing accuracy.*

Preference methods. We name the relative importance of objectives *preferences*. Preferences can be indicated in may ways (e.g. weights or hierarchies), which we name.

MOO methods. We consider three methods commonly used in MOMIPs [2] and a naive greedy method of our own contribution.

1) The weighted sum method: The weighted sum method minimizes the weighted sum of all objectives. It guarantees a Pareto optimal solution. The weighted sum model selection MOMIP is formulated as follows:

$$\begin{array}{ll} \min_{y \in Y} & w_{acc\_loss} f_{acc\_loss}^{norm} + w_{cost} f_{cost}^{norm} + w_{mem} f_{mem}^{norm} \\ \text{such that} & (3), (4), (5) \text{ hold.} \end{array} \quad (11)$$

2) The weighted min-max method: In the weighted min-max method, also known as the weighted Tchebycheff method, an ancillary variable λ is introduced as an upper bound to every (weighted) objective, which is then minimized. It generates weakly Pareto optimal solution. The weighted min-max MOMIP model selection is formulated as follows:

$$\begin{array}{lccl} \min_{y \in Y} & \lambda & & \\ \text{such that} & (3), (4), (5), & & \\ & w_i f_i^{norm} & \leq & \lambda, \\ & i \in \{acc\_loss, cost, mem\} & \text{hold.} & \end{array} \quad (12)$$

3) The lexicographic method: In the lexicographic method a hierarchy of objectives is used to convey their importance. First the program is solved for the most importance objective. The objective value is then added as an upper bound to that objective, and next the second most important objective is used as the objective function. This means that the MOMIP needs to be solved several times, and the method is inefficient for finding compromise solutions. The lexicographic method guarantees Pareto optimal solutions.

4) Greedy-MOO (Baseline): Our greedy baseline pairs predicates to models using a basic weighted sum utility function:

$$\begin{array}{rcl} U(m,p) & = & w_{acc\_loss} \frac{1 - A_{m,p}}{1 - \min\{A_{n,p} | n \in M\}} \\ & + & w_{cost} \frac{C_m}{\max C} + w_{mem} \frac{D_m}{\max D} \end{array} \quad (13)$$

which is then used to calculate the following model selection:

$$X_{m,p} = \begin{cases} 1 \text{ if } U(m,p) = \min\{U(n,p) | n \in M, A_{n,p} > 0\} \\ 0 \text{ elsewhere} \end{cases} \quad (14)$$

Greedy-MOO does not guarantee Pareto optimal solutions, which can be shown using simple counterexamples.

Example 5. *We visualize a sample search space with solutions found by the different MOO methods in Fig. 1. Note how the solutions vary in their objective values, even when they use similar preference profiles.*

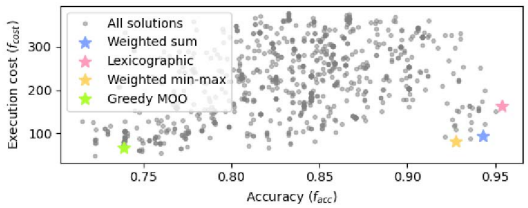


Fig. 1: MOO solutions in an example search space

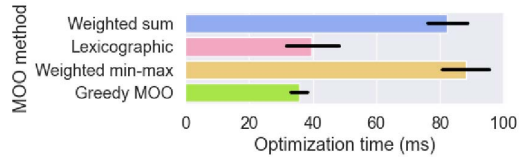


Fig. 2: Optimization time over different MOO methods

V. Results

A. Setup

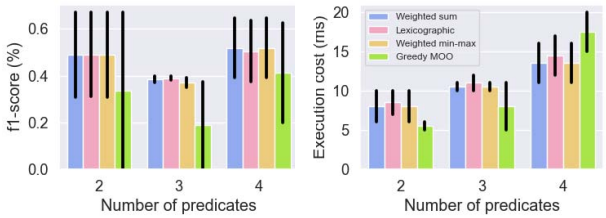
1) Data: The data that we are using for our experiments are the COCO [12] dataset and Tweeteval [5] corpus. COCO is a dataset aiming for object detection (OD) task, containing 123k images, covering 80 classes. Tweeteval is a corpus of 50k tweets, covering natural language processing (NLP) tasks such as sentiment analysis and topic classification, with 17 classes in total. Each instance in both datasets contain multiple labels, making them suitable for complex queries.

2) Model zoos: we have collected pretrained models from several public model zoos (Hugging Face and the PyTorch Hub). For the NLP and OD tasks we collected 47 and 31 models respectively. We finetune some NLP models to fit Tweeteval to perform different tasks.

3) Metrics: We split the dataset into evaluation set and test set, where we use evaluation set to retrieve metadata regarding the models, and test set to measure the performance of optimizers on complex queries. To test the efficacy of the optimizers, we assume the data distribution of evaluation set and test set is the same. For accuracy we measured the quality of the models in f1-score. The execution cost is measured as the time it takes to inference on a single data point in milliseconds. The cost of models range from a few milliseconds to a few hundreds.

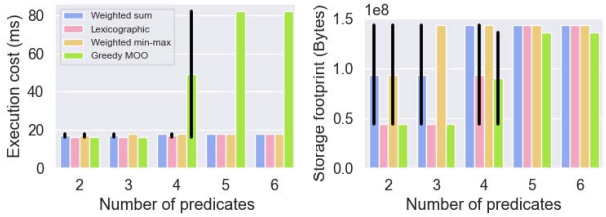
4) Queries: To run our experiments we formulated 10 queries for the OD model zoo and 6 for the NLP model zoo, half in CNF and half in DNF. The queries vary from 2 to 6 predicates, with 1 to 2 predicates per group. Queries are similar to (`person` $\wedge$ (`car` $\vee$ `bike`) $\wedge$ `emergency_light`).

5) Optimization: We generate query plans for every query over two preference profiles (see V-B2), execute them over the test set, and record the resulting f1-score, and execution cost. A preference profile consists of a hierarchy of the three objectives, where the most important objective gets a weight of $\frac{1}{2}$, the second $\frac{1}{3}$, and the last $\frac{1}{6}$. We also compare the methods on the time it takes to generate a query plan, for which we use a larger number of randomly generated queries.



(a) f1-score (b) Execution cost

Fig. 3: Query plan performance for the NLP scenario



(a) Execution cost (b) Memory footprint

Fig. 4: Query plan performance for the OD scenario

B. Results

1) Optimization time: For optimization time, visualized in Fig. 2 with aggregated results from different queries, we see that the greedy-MOO and the lexicographic method perform significantly better than the weighted sum and weighted min-max method, that perform comparatively.

2) Query execution: To compare query execution performance, we spotlight two use case scenarios and compare query plans calculated with our 4 MOO methods on their two most important objectives.

- NLP: accuracy $>$ execution cost $>$ memory;
- OD: execution cost $>$ memory $>$ accuracy.

We see that for both scenarios in Fig. 3 and 4 that the greedy method performs poorly. Even for the most important objective (accuracy in the NLP scenario, execution cost in the OD scenario) it has bad scores. The lexicographic method manages to score well for its most important objective, but underperforms for the others. The weighted sum and weighted min-max method perform very comparably, finding highly similar query plans in most cases and balancing objectives adequately. Due to its slightly lower computation time and guarantee for Pareto-optimal query plans, the weighted sum method would be the better choice for our optimizer.

C. Conclusion

We investigated the multi-objective ML inference query optimization problem and formulated as MOMIP that optimizes for accuracy, execution cost, and memory footprint. We tested several commonly used MOO methods and compared them on their theoretical suitability and tested their performance in different experimental settings. We note that the weighted-sum based optimizer can process user preferences and balance objectives accordingly, outperforming naive methods. Future work can investigate effect of the assumptions.

References

[1] A. Akbik, T. Bergmann, D. Blythe, K. Rasul, S. Schweter, and R. Vollgraf, "Flair: An easy-to-use framework for state-of-the-art nlp," in *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics (demonstrations)*, 2019, pp. 54–59.

[2] M. J. Alves and J. Climaco, "A review of interactive methods for multiobjective integer and mixed-integer programming," *European Journal of Operational Research*, vol. 180, no. 1, pp. 99–115, 2007.

[3] M. R. Anderson, M. Cafarella, G. Ros, and T. F. Wenisch, "Physical representation-based predicate optimization for a visual analytics database," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, IEEE, 2019, pp. 1466–1477.

[4] W. Balke and U. Güntzer, "Multi-objective query processing for database systems," in *(e)Proceedings of the Thirtieth International Conference on Very Large Data Bases, VLDB 2004, Toronto, Canada, August 31 - September 3 2004*, M. A. Nascimento, M. T. Özsu, D. Kossmann, R. J. Miller, J. A. Blakeley, and K. B. Schiefer, Eds., Morgan Kaufmann, 2004, pp. 936–947. DOI: 10.1016/B978-012088469-8.50082-6. [Online]. Available: http://www.vldb.org/conf/2004/RS24P1.PDF.

[5] F. Barbieri, J. Camacho-Collados, L. Neves, and L. Espinosa-Anke, "Tweeteval: Unified benchmark and comparative evaluation for tweet classification," *arXiv preprint arXiv:2010.12421*, 2020.

[6] Z. Cai, M. Saberian, and N. Vasconcelos, "Learning complexity-aware cascades for pedestrian detection," *IEEE transactions on pattern analysis and machine intelligence*, vol. 42, no. 9, pp. 2195–2211, 2019.

[7] J. Cao, R. Hadidi, J. Arulraj, and H. Kim, "Thia: Accelerating video analytics using early inference and fine-grained query planning," *arXiv preprint arXiv:2102.08481*, 2021.

[8] T. Gillespie, "Content moderation, ai, and the question of scale," *Big Data & Society*, vol. 7, no. 2, p. 2 053 951 720 943 234, 2020.

[9] F. Helff, L. Gruenwald, and L. d'Orazio, "Weighted sum model for multi-objective query optimization for mobile-cloud database environments.," in *EDBT/ICDT Workshops*, Citeseer, 2016.

[10] J. Jiang, G. Ananthanarayanan, P. Bodik, S. Sen, and I. Stoica, "Chameleon: Scalable adaptation of video analytics," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, 2018, pp. 253–266.

[11] D. Kang, J. Emmons, F. Abuzaid, P. Bailis, and M. Zaharia, "Noscope: Optimizing neural network queries over video at scale," *arXiv preprint arXiv:1703.02529*, 2017.

[12] T.-Y. Lin, M. Maire, S. Belongie, *et al.*, "Microsoft coco: Common objects in context," in *European conference on computer vision*, Springer, 2014, pp. 740–755.

[13] Y. Lu, A. Chowdhery, S. Kandula, and S. Chaudhuri, "Accelerating machine learning inference with probabilistic predicates," in *Proceedings of the 2018 International Conference on Management of Data*, 2018, pp. 1493–1508.

[14] R. T. Marler and J. S. Arora, "Survey of multi-objective optimization methods for engineering," *Structural and multidisciplinary optimization*, vol. 26, no. 6, pp. 369–395, 2004.

[15] C. H. Papadimitriou and M. Yannakakis, "Multiobjective query optimization," in *Proceedings of the Twentieth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 21-23, 2001, Santa Barbara, California, USA*, P. Buneman, Ed., ACM, 2001. DOI: 10.1145/375551.375560. [Online]. Available: https://doi.org/10.1145/375551.375560.

[16] Y. Shu, Z. Kou, Z. Cao, J. Wang, and M. Long, "Zoo-tuning: Adaptive transfer from a zoo of models," in *International Conference on Machine Learning*, PMLR, 2021, pp. 9626–9637.

[17] F. P. Such, V. Madhavan, R. Liu, *et al.*, "An atari model zoo for analyzing, visualizing, and comparing deep reinforcement learning agents," in *IJCAI*, 2019.

[18] Y. Sun, X. Wang, and X. Tang, "Deep convolutional network cascade for facial point detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 3476–3483.

[19] I. Trummer and C. Koch, "Multi-objective parametric query optimization," *The VLDB Journal*, vol. 26, no. 1, pp. 107–124, 2017.

[20] L. A. Wolsey, "Mixed integer programming," *Wiley Encyclopedia of Computer Science and Engineering*, pp. 1–10, 2007.

[21] J. Wu, Z. Liu, J. Li, C. Gu, M. Si, and F. Tan, "An algorithm for automatic vehicle speed detection using video camera," in *2009 4th International Conference on Computer Science & Education*, IEEE, 2009, pp. 193–196.

[22] Z. Yang, Z. Wang, Y. Huang, Y. Lu, C. Li, and X. S. Wang, "Optimizing machine learning inference queries with correlative proxy models," *arXiv preprint arXiv:2201.00309*, 2022.

[23] H. Zhang, G. Ananthanarayanan, P. Bodik, M. Philipose, P. Bahl, and M. J. Freedman, "Live video analytics at scale with approximation and delay-tolerance," in *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, 2017, pp. 377–392.

[24] W.-Y. Zhou, G.-W. Yang, and S.-M. Hu, "Jittor-gan: A fast-training generative adversarial network model zoo based on jittor," *Computational Visual Media*, vol. 7, no. 1, pp. 153–157, 2021.