
Feasibility-Aware Decision-Focused Learning for Predicting Parameters in the Constraints

Jayanta Mandi

Department of Computer Science
KU Leuven, Leuven, Belgium
jayanta.mandi@kuleuven.be

Marianne Defresne*

LAAS-CNRS, Université de Toulouse
CNRS, INSA, Toulouse
marianne.defresne@laas.fr

Senne Berden

Department of Computer Science
KU Leuven, Leuven, Belgium
senne.berden@kuleuven.be

Tias Guns

Department of Computer Science
KU Leuven, Leuven, Belgium
tias.guns@kuleuven.be

Abstract

When some parameters of a constrained optimization problem (COP) are uncertain, this gives rise to a predict-then-optimize (PtO) problem, comprising two stages: the *prediction* of the unknown parameters from contextual information and the subsequent *optimization* using those predicted parameters. Decision-focused learning (DFL) implements the first stage by training a machine learning (ML) model to optimize the quality of the decisions made using the predicted parameters. When the predicted parameters occur in the constraints, they can lead to infeasible solutions. Therefore, it is important to simultaneously manage both feasibility and decision quality. We develop a DFL framework for predicting constraint parameters in a generic COP. While prior works typically assume that the underlying optimization problem is a linear program (LP) or integer LP (ILP), our approach makes no such assumption. We derive two novel loss functions based on maximum likelihood estimation (MLE): the first one penalizes infeasibility (by penalizing predicted parameters that lead to infeasible solutions), while the second one penalizes suboptimal decisions (by penalizing predicted parameters that make the true optimal solution infeasible). We introduce a single tunable parameter to form a weighted average of the two losses, allowing decision-makers to balance suboptimality and feasibility. We experimentally demonstrate that adjusting this parameter provides decision-makers control over this trade-off. Moreover, across several COP instances, we show that adjusting the tunable parameter allows a decision-maker to prioritize either suboptimality or feasibility, outperforming the performance of existing baselines in either objective.

1 Introduction

Many real-world optimization problems involve parameters that are unknown at decision time, such as uncertain customer demand in manufacturing problems or unknown traffic in delivery routing. This gives rise to *predict-then-optimize* (PtO) problems [22], where a machine learning (ML) model first makes point predictions of the unknown parameters from contextual features, after which the resulting instantiated optimization problem is solved.

*Affiliated to KU Leuven during submission of this article

In *prediction-focused learning* (PFL), the predictive model is trained solely to maximize the accuracy of parameter predictions by minimizing standard ML losses. By doing so, PFL does not consider the effect of prediction errors on the solution to the optimization problem. For example, consider a vehicle routing problem, where one first needs to predict unknown customer demands, to then make routing decisions. In this setting, overestimations and underestimations can have differing effects. Overestimating demand may lead to a conservative suboptimal decision, while underestimating demand may lead to an infeasible decision. However, these differing effects are not accounted for by PFL losses.

To address this shortcoming of PFL, *decision-focused learning* (DFL) directly trains the ML model to minimize a *task-specific decision loss* that reflects the quality of the solution produced by using the predicted parameters. With a slight abuse of notation, we will use the term ‘predicted solution’ to denote the solution produced by using the predicted parameters, although the ML model does not *directly* predict the solution, it predicts parameters that are passed to an optimization solver at inference time.

Most existing DFL works [12, 13, 23, 33] focus on cases where the predicted parameters appear *only* in the objective function. In such settings, the decision loss is typically the *regret*, which measures suboptimality of the predicted solution. Instead, we focus on predicting parameters that appear in the constraints of the optimization problem. In this setting too, suboptimal solutions may arise, because the feasible space defined by the predicted parameters may exclude the true optimal solution. However, focusing only on suboptimality is insufficient. This is because the predicted solution may become infeasible with respect to the constraints instantiated by the *true* parameters. Therefore, the ML model should also be trained to minimize the likelihood of such infeasible outcomes.

Focusing only on suboptimality or on infeasibility generally does not lead to a desirable balance between these two objectives. To address this, Hu et al. [14] model decision-making with uncertain constraint parameters as a two-stage process: first, a solution is generated using the predicted parameters; then, if needed, corrective actions – incurring additional costs – are carried out to resolve potential infeasibility. Subsequent works [15, 16] have extended this line of research.

One limitation of their approach is that the corrective action is often problem-specific, and restricted to a specific form of optimization problem (e.g., a linear program). Instead, we avoid the complexity of a second stage altogether, and focus on predicting parameters that minimize both the infeasibility and the suboptimality of the first-stage solution itself. To this end, we first propose two novel loss functions: the Infeasibility Penalty Loss (IPL) and Optimality-Preserving Loss (OPL). The idea behind IPL is that when solving with the predicted parameters leads to a solution that is infeasible according to the true parameters, such infeasibility should be (proportionally) penalized. On the other hand, the OPL loss will penalize predicted parameters that make the true optimal solution infeasible, to avoid obtaining a suboptimal solution when solving with the predicted parameters.

These two losses often contradict each other. The IPL term encourages conservative parameter predictions to guarantee feasibility, even at the expense of producing extremely suboptimal solutions. In contrast, OPL incentivizes loose parameter predictions to ensure the optimal solution is not cut away. Moreover, in practice, a decision-maker may favor more feasibility over optimality or vice versa. To support this subjective preference of one loss over the other, we propose to minimize a weighted average of the two losses. In this way, we introduce a tunable weighting mechanism that adjusts the total loss based on the decision-maker’s preference. We refer to our approach as *Odece*, for “optimizing decisions through end-to-end constraint estimation”.

We compare Odece with the PFL approach of minimizing the mean squared error (MSE), and with some existing DFL approaches. These approaches do not allow the decision-maker to adjust the predicted solution based on their subjective preference between optimality and feasibility. Moreover, the DFL approaches for constraint prediction are limited to linear programs (LPs) or integer or mixed-integer LPs (ILPs and MILPs), or they focus on predicting full constraint systems rather than individual parameters within the constraints. Our experimental results across various optimization problem instances show that, for a single value of the tunable parameter, Odece matches or outperforms existing baselines in terms of both suboptimality and feasibility. More importantly, they show that our approach offers a simple way for decision-makers to *control* the trade-off between suboptimality and infeasibility using a single parameter.

2 Related Literature

Differentiating through optimization problems. Differentiating through optimization problems has gained attention in ML, as it allows solvers to be embedded into gradient-based training. This is often done by ADMM [9, 29] or by implicit differentiation [8] of the optimality conditions. Implicit differentiation techniques have been developed for quadratic programs [2] and conic optimization [1]. However, for combinatorial optimization problems like ILPs, the true gradient is zero almost everywhere because small changes in the parameters usually do not affect the optimal solution. Hence, implicit differentiation techniques, which are applicable to smooth optimization problems, cannot be applied to such combinatorial problems. To address this, Paulus et al. [26] propose CombOptNet, a technique for ILPs to compute pseudo-gradients based on the Euclidean distance between the solution and the constraints.

Predicting objective parameters. Most DFL works focus on predicting parameters in the objective function. Some of these works are based on differentiating through the problem after smoothing it, either using regularization [13, 18, 33] or perturbation [5, 10, 25, 28]. Another approach is to define informative surrogates of the decision loss [4, 12, 19, 20, 23, 31]. Regularization-based methods require differentiable solvers, such as *Cvxpylayers* [1] or *OptNet* [2]. In contrast, perturbation and surrogate-based approaches only need access to solutions. These solutions generally do not need to come from an exact solver and can be obtained by heuristics or alternative methods. For instance, surrogate-based techniques have been applied to planning problems solved via approximate automated planners [21]. We refer readers to a recent survey paper [22] for a more comprehensive overview.

Predicting constraint parameters. Relatively few DFL papers have considered the prediction of parameters in the constraints. Most of these [14–16] model decision-making as a two-stage process. In the first stage, a solution is obtained using the predicted parameters, which may be infeasible with respect to the true parameters. The second stage then applies a corrective action to convert the first-stage solution into a feasible one. Among these works, Hu et al. [15] also recommend a strategy for applying their approach to settings where no corrective action is available at inference time. Their proposal is to assign a very high penalty factor to the corrective action during training. In contrast to this line of work, we do not introduce corrective actions even during training and focus on balancing decision quality and feasibility in a single stage. Another approach [11] simultaneously learns the objective and the constraints, but the constraints are soft and the problem must be formulated as a discrete graphical model. The work by Nandwani et al. [24] aims to learn the full set of constraints from the features. They do this by learning a set of hyperplanes that separate the optimal solution from other *negative assignments*. A key component of their technique is the generation of such negative assignments, which must include both infeasible as well as feasible but suboptimal ones. In contrast to their setting, we focus on learning only the parameters within a known constraint structure. Their method does not easily generalize to our setting, in which the learned parameters must remain compatible with the known constraints to make high-quality solutions. Moreover, both Hu et al. [15] and Nandwani et al. [24] restrict their methods to LPs and ILPs, while the framework proposed in this paper is more general and applicable to a broader class of optimization problems. The prediction of constraints is also central in constraint acquisition, for optimization [3, 17] or constraint satisfaction [6, 32] problems, but the setting differs because the structure of the constraints is unknown and the uncertainty does not depend on any input features.

3 Preliminaries

Problem setup. We will consider a constrained optimization problem (COP) of the following form:

$$\begin{aligned} \min_{\mathbf{x} \in \Omega} \quad & f(\mathbf{x}; \mathbf{q}) \\ \text{subject to} \quad & g_i(\mathbf{x}; \boldsymbol{\rho}) \leq 0, \quad i \in \{1, \dots, M\} \end{aligned} \tag{1a} \tag{1b}$$

where Ω defines the domain of the decision variable vector $\mathbf{x}$. Ω may be a subset of real numbers or integers, depending on the type of optimization problem. The upper and lower bounds for each decision variable are also specified by Ω . We will assume that the constraint functions $g_i(\mathbf{x}; \boldsymbol{\rho})$ are differentiable functions. We do not explicitly express equality constraints, since any constraint

of the form $h_j(\mathbf{x}; \boldsymbol{\rho}) = 0$ can be represented by two inequality constraints: $h_j(\mathbf{x}; \boldsymbol{\rho}) \leq 0$ and $-h_j(\mathbf{x}; \boldsymbol{\rho}) \leq 0$. To make the notation simpler, we denote the set of constraints by $\mathcal{S}$ and the feasible set when the constraint parameters take the value $\boldsymbol{\rho}$ by $\mathcal{F}(\boldsymbol{\rho})$. More formally,

$$\mathcal{F}(\boldsymbol{\rho}) = \{\mathbf{x} \in \Omega : g_i(\mathbf{x}; \boldsymbol{\rho}) \leq 0, \quad \forall i \in \mathcal{S}\} \quad (2)$$

We use $\mathbf{x}^*(\mathbf{q}, \boldsymbol{\rho})$ to denote a (possibly non-unique) optimal solution to the optimization problem for a given objective parameter $\mathbf{q}$ and constraint parameter $\boldsymbol{\rho}$. When multiple optimal solutions exist, we define the set of all optimal solutions as $\mathbf{W}(\mathbf{q}, \boldsymbol{\rho}) = \{\mathbf{x} : f(\mathbf{x}; \mathbf{q}) \leq f(\mathbf{x}'; \mathbf{q}); \forall \mathbf{x}' \in \mathcal{F}(\boldsymbol{\rho})\}$. Note that $\mathbf{W}(\mathbf{q}, \boldsymbol{\rho})$ is a subset of the feasible region $\mathcal{F}(\boldsymbol{\rho})$. So, if $\mathcal{F}$ is empty (i.e., no feasible solution exists), $\mathbf{W}$ is also empty. In the special case of a unique optimal solution, $\mathbf{W}(\mathbf{q}, \boldsymbol{\rho})$ is a singleton. However, our implementation relies on a solver that returns a single solution. When multiple optimal solutions exist, the commercial solvers we use break ties using a predefined rule.

Example. Consider the multi-dimensional 0-1 knapsack problem (MDKP) [27]. The objective of this problem is to choose a subset with maximal value from a given set of items, subject to M capacity constraints. Let ρ_i be the capacity in dimension i . There are N items, the value of each item is q_n , and $\rho_{(n,i)}$ is the weight of item n in dimension i . Each decision variable x_n can be either zero or one, hence $\Omega = \{0, 1\}^N$. This optimization problem can be modeled as an ILP as follows:

$$\min_{\mathbf{x}_{1:N}} \sum_{n=1}^N (-q_n)x_n \quad \text{such that} \quad x_n \in \{0, 1\} \quad \forall n \in [N], \quad \sum_{n=1}^N \rho_{ni}x_n \leq \rho_i \quad \forall i \in [M] \quad (3)$$

Here, $\mathcal{S}$ consists of the M capacity constraints.

Predict-then-Optimize problems. In the PtO formulation for uncertainty in the constraints, some or all components of the parameter vector $\boldsymbol{\rho}$ are unknown and must be estimated before solving the COP. For example, in the knapsack problem, the item weights $\rho_{(n,i)}$ may be unknown, but the capacity in each dimension ρ_i might be known. Additionally, we observe a set of values correlated with $\boldsymbol{\rho}$, which we refer to as *features* and denote by ϕ . These features are used to make estimates of the unknown parameters. For the estimation task, a predictive model $\mathcal{M}_\theta$, parameterized by θ , is trained using training data of the form $\mathcal{D} \equiv \{(\phi_\kappa, \boldsymbol{\rho}_\kappa, \mathbf{q}_\kappa, \mathbf{x}^*(\mathbf{q}_\kappa, \boldsymbol{\rho}_\kappa))\}_{\kappa=1}^K$. Formally, the goal is to learn a mapping from the feature space to the parameter space. In this work, we assume that the cost vector $\mathbf{q}_\kappa$ is instance-specific but known and not used to learn the mapping. Note that the structure of the COP is fixed, i.e., the number of constraints and the domains of decision variables are known. Once trained, given a new feature vector ϕ and a known cost vector $\mathbf{q}$, the model produces an estimated parameter vector $\hat{\boldsymbol{\rho}} = \mathcal{M}_\theta(\phi)$ using ϕ . Since this is a PtO task, $\mathcal{M}_\theta$ is ultimately evaluated based on the quality of the solution $\mathbf{x}^*(\mathbf{q}, \hat{\boldsymbol{\rho}})$ induced by the predicted parameters.

A maximum likelihood perspective on DFL. We will develop a framework based on maximum likelihood estimation (MLE) to predict constraint parameters. Our goal is to maximize the log-likelihood of the observed data $\mathcal{D}$. In PFL, the parameters θ of the predictive model would be estimated to increase the likelihood of the observations $\boldsymbol{\rho}_\kappa$ given the features ϕ_κ . However, in DFL training, the focus is on the likelihood of $\mathbf{x}^*(\mathbf{q}_\kappa, \boldsymbol{\rho}_\kappa)$ being the optimal solution, given $\mathbf{q}_\kappa$ and ϕ_κ . To put it another way, the MLE perspective of DFL consists of learning a θ to predict the constraint parameter vector $\hat{\boldsymbol{\rho}}_\kappa$ so that $\mathbf{x}^*(\mathbf{q}_\kappa, \hat{\boldsymbol{\rho}}_\kappa)$ is the most likely solution to the parameter $\hat{\boldsymbol{\rho}}_\kappa = \mathcal{M}_\theta(\phi_\kappa)$.

4 Methodology

We start by defining a function $UNSAT_i(\mathbf{x}, \boldsymbol{\rho}_\kappa)$ to indicate whether the i -th constraint is violated by an assignment $\mathbf{x}$ when the constraint parameter is set to the vector $\boldsymbol{\rho}_\kappa$. It is defined as:

$$UNSAT_i(\mathbf{x}, \boldsymbol{\rho}_\kappa) := \mathbb{1}\{g_i(\mathbf{x}, \boldsymbol{\rho}_\kappa) > 0\}$$

Similarly, we define $SAT_i(\mathbf{x}, \boldsymbol{\rho}_\kappa)$ to indicate that the i -th constraint is satisfied. Therefore, $SAT_i(\mathbf{x}, \boldsymbol{\rho}_\kappa) := 1 - UNSAT_i(\mathbf{x}, \boldsymbol{\rho}_\kappa)$. Using these definitions, we introduce $Feas(\mathbf{x}, \boldsymbol{\rho}_\kappa)$, an indicator variable equal to 1 if the solution $\mathbf{x}$ satisfies all the constraints.

$$Feas(\mathbf{x}, \boldsymbol{\rho}_\kappa) = \prod_{i \in \mathcal{S}} SAT_i(\mathbf{x}, \boldsymbol{\rho}_\kappa) \quad (4)$$

We use $Infeas(\mathbf{x}, \boldsymbol{\rho}_\kappa)$ to denote that $\mathbf{x}$ is infeasible with respect to $\boldsymbol{\rho}_\kappa$. Note that $\mathbf{x}$ is an infeasible assignment if it violates any one of the constraints. Hence, we can write:

$$Infeas(\mathbf{x}, \boldsymbol{\rho}_\kappa) = \max_{i \in \mathcal{S}} UNSAT_i(\mathbf{x}, \boldsymbol{\rho}_\kappa) \quad (5)$$

As we want to maximize the likelihood of obtaining a feasible solution with optimal objective value, we first model the probability of violating the constraints using the predicted parameters. Given predicted parameters $\mathcal{M}_\theta(\phi_\kappa)$, we model the parametric probability $P_\theta(UNSAT_i(\mathbf{x}, \mathcal{M}_\theta(\phi_\kappa)) | \mathcal{M}_\theta(\phi_\kappa))$ in the following manner:

$$P_\theta(UNSAT_i(\mathbf{x}, \mathcal{M}_\theta(\phi_\kappa)) | \mathcal{M}_\theta(\phi_\kappa)) = \frac{1}{1 + \exp(-g_i(\mathbf{x}, \mathcal{M}_\theta(\phi_\kappa)))} \quad (6)$$

Note that the above is a smooth version of the following non-smooth distribution:

$$P_\theta = \begin{cases} 1, & \text{if } g_i(\mathbf{x}, \mathcal{M}_\theta(\phi_\kappa)) \geq 0 \\ 0, & \text{otherwise} \end{cases} \quad (7)$$

As $P_\theta(SAT_i(\mathbf{x}, \mathcal{M}_\theta(\phi_\kappa)) | \mathcal{M}_\theta(\phi_\kappa)) = 1 - P_\theta(UNSAT_i(\mathbf{x}, \mathcal{M}_\theta(\phi_\kappa)) | \mathcal{M}_\theta(\phi_\kappa))$, we can write:

$$P_\theta(SAT_i(\mathbf{x}, \mathcal{M}_\theta(\phi_\kappa)) | \mathcal{M}_\theta(\phi_\kappa)) = \frac{1}{1 + \exp(g_i(\mathbf{x}, \mathcal{M}_\theta(\phi_\kappa)))} \quad (8)$$

4.1 Loss Function to Penalize Infeasibility-Inducing Predictions

We first propose a loss function that ensures that true infeasible assignments remain infeasible under the predicted parameters. Let $\mathbf{x}_{neg}$ be an assignment that is infeasible under the true parameters. As it is easy to determine which constraints are violated by $\mathbf{x}_{neg}$, we develop an MLE framework to maximize the likelihood of violating those constraints under the predicted parameters. Given a $\mathbf{x}_{neg}$, the log-likelihood of violating these constraints can be written in the following form:

$$\sum_{i \in \mathcal{S}} UNSAT_i(\mathbf{x}_{neg}, \boldsymbol{\rho}_\kappa) \log P_\theta(UNSAT_i(\mathbf{x}_{neg}, \mathcal{M}_\theta(\phi_\kappa)) | \mathcal{M}_\theta(\phi_\kappa)) \quad (9)$$

Note that $UNSAT_i(\mathbf{x}_{neg}, \boldsymbol{\rho}_\kappa)$ can be viewed as a mask that equals 1 when $\mathbf{x}_{neg}$ violates constraint i under the true parameters $\boldsymbol{\rho}_\kappa$. Using the definition of $P_\theta(UNSAT_i(\mathbf{x}, \mathcal{M}_\theta(\phi_\kappa)) | \mathcal{M}_\theta(\phi_\kappa))$ from Eq. 6, we can write the following:

$$\log P_\theta(UNSAT_i(\mathbf{x}_{neg}, \mathcal{M}_\theta(\phi_\kappa)) | \mathcal{M}_\theta(\phi_\kappa)) = -\log(1 + \exp(-g_i(\mathbf{x}_{neg}, \mathcal{M}_\theta(\phi_\kappa)))) \quad (10)$$

We now propose a loss function to maximize this likelihood (i.e., minimize the negative log-likelihood). We include a margin parameter $\Upsilon > 0$ to encourage the constraints to be violated with a buffer. Since this loss penalizes cases where an infeasible assignment $\mathbf{x}_{neg}$ is considered feasible under the predicted parameters, we refer to it as the Infeasibility-Aware Loss (IAL).

$$\mathcal{L}^{IAL}(\mathbf{x}_{neg}, \mathcal{M}_\theta(\phi_\kappa)) = \sum_{i \in \mathcal{S}} UNSAT_i(\mathbf{x}_{neg}, \boldsymbol{\rho}_\kappa) \text{softplus}(\Upsilon - g_i(\mathbf{x}_{neg}, \mathcal{M}_\theta(\phi_\kappa))) \quad (11)$$

where we write $\log(1 + \exp(\cdot))$ as $\text{softplus}(\cdot)$.

Intuition behind IAL. Note that softplus is a smooth approximation to the Relu function. If we replace the softplus function with a Relu function, Eq. 11 takes the following form:

$$\overline{\mathcal{L}^{IAL}}(\mathbf{x}_{neg}, \mathcal{M}_\theta(\phi_\kappa)) = \sum_{i \in \mathcal{S}} UNSAT_i(\mathbf{x}_{neg}, \boldsymbol{\rho}_\kappa) \max(0, \Upsilon - g_i(\mathbf{x}_{neg}, \mathcal{M}_\theta(\phi_\kappa))) \quad (12)$$

From this expression, we see that for a truly violated constraint, i.e., of which $UNSAT_i(\mathbf{x}_{neg}, \boldsymbol{\rho}_\kappa) = 1$, the loss becomes zero only if $g_i(\mathbf{x}_{neg}, \mathcal{M}_\theta(\phi_\kappa)) > \Upsilon$. In other words, the predicted constraint must also be violated. $\overline{\mathcal{L}^{IAL}}(\mathbf{x}_{neg}, \mathcal{M}_\theta(\phi_\kappa))$ is zero if this holds for all constraints violated by $\mathbf{x}_{neg}$.

Selecting $\mathbf{x}_{neg}$. The IAL loss formulation is based on $\mathbf{x}_{neg}$, a candidate assignment which violates at least one constraint with respect to the true parameter ρ_κ . This leads to the question of how to obtain such a $\mathbf{x}_{neg}$. We adopt the following approach: for each instance, after predicting $\hat{\rho}_\kappa$, we solve the COP using $\hat{\rho}_\kappa$ and obtain a solution $\mathbf{x}^*(\mathbf{q}_\kappa, \hat{\rho}_\kappa)$. If this solution is infeasible under the true parameters ρ_κ , we use it as $\mathbf{x}_{neg}$. Otherwise, we set the loss to zero, since the predicted solution satisfies all the true constraints, which is the preferred outcome. We call this final loss function the Infeasibility Penalty Loss (IPL), as it penalizes the predicted parameters when the resulting predicted solution violates the true constraints.

$$\mathcal{L}^{IPL}(\rho_\kappa, \hat{\rho}_\kappa) = (1 - Feas(\mathbf{x}^*(\mathbf{q}_\kappa, \hat{\rho}_\kappa), \rho_\kappa)) \mathcal{L}^{IAL}(\mathbf{x}^*(\mathbf{q}_\kappa, \hat{\rho}_\kappa), \hat{\rho}_\kappa) \quad (13)$$

We can also define $\overline{\mathcal{L}^{IPL}}$ where $\mathcal{L}^{IAL}$ in Eq. 13 is replaced with $\overline{\mathcal{L}^{IAL}}$.

Lemma 1. For a true parameter ρ_κ and a predicted parameter $\hat{\rho}_\kappa$, $\overline{\mathcal{L}^{IPL}}(\rho_\kappa, \hat{\rho}_\kappa)$ is zero if and only if $\mathbf{x}^*(\mathbf{q}_\kappa, \hat{\rho}_\kappa)$, an optimal solution for $\hat{\rho}_\kappa$, is feasible with respect to ρ_κ .

Proof. $\overline{\mathcal{L}^{IPL}}$ is by construction zero when $\mathbf{x}^*(\mathbf{q}_\kappa, \hat{\rho}_\kappa)$ is feasible with respect to ρ_κ . Because if $\mathbf{x}^*(\mathbf{q}_\kappa, \hat{\rho}_\kappa)$ is feasible with respect to ρ_κ , $Feas(\mathbf{x}^*(\mathbf{q}_\kappa, \hat{\rho}_\kappa), \rho_\kappa)$ is equal to one, which makes $\overline{\mathcal{L}^{IPL}} = 0$. Hence,

$$Feas(\mathbf{x}^*(\mathbf{q}_\kappa, \hat{\rho}_\kappa), \rho_\kappa) = 1 \implies \overline{\mathcal{L}^{IPL}} = 0$$

To prove the other direction, let us assume, $\overline{\mathcal{L}^{IPL}} = 0$, but $Feas(\mathbf{x}^*(\mathbf{q}_\kappa, \hat{\rho}_\kappa), \rho_\kappa) \neq 1$. Hence, $\overline{\mathcal{L}^{IAL}}(\mathbf{x}^*(\mathbf{q}_\kappa, \hat{\rho}_\kappa), \hat{\rho}_\kappa)$ must be zero. Note that, $\overline{\mathcal{L}^{IAL}}(\mathbf{x}^*(\mathbf{q}_\kappa, \hat{\rho}_\kappa), \hat{\rho}_\kappa)$ can be zero only if $g_i(\mathbf{x}^*(\mathbf{q}_\kappa, \hat{\rho}_\kappa), \hat{\rho}_\kappa) \geq \Upsilon > 0$ for all the violated constraints. However, as $\mathbf{x}^*(\mathbf{q}_\kappa, \hat{\rho}_\kappa)$ is the optimal solution with $\hat{\rho}_\kappa$ being the parameter, $g_i(\mathbf{x}^*(\mathbf{q}_\kappa, \hat{\rho}_\kappa), \hat{\rho}_\kappa)$ must be less than zero for all the constraints and we arrived at a contradiction. So,

$$\overline{\mathcal{L}^{IPL}} = 0 \implies Feas(\mathbf{x}^*(\mathbf{q}_\kappa, \hat{\rho}_\kappa), \rho_\kappa) = 1$$

□

4.2 Loss Function for Preserving Optimal Solutions

The IPL loss penalizes predicted parameters when the predicted solution becomes infeasible with respect to the true parameters. However, minimizing *only* $\mathcal{L}^{IPL}$ can lead to learning a feasible region that is much stricter than the true feasible region, as any solution therein will be feasible under the true parameters. For example, when learning the capacity parameter in a knapsack problem, minimizing only $\mathcal{L}^{IPL}$ might result in capacity values so small that no items are selected. While such solutions are always feasible with respect to the true parameters, they are suboptimal and of no practical value. Therefore, we also need a loss function to penalize cases where the learned parameters turn out to be too strict. To do this, we define a loss function that penalizes predicted parameters when they render known optimal solutions infeasible. Within the MLE framework, this corresponds to maximizing the likelihood that the true solution is feasible.

Thus, we formulate the probability of an assignment $\mathbf{x}$ being a feasible point, given the parameter $\mathcal{M}_\theta(\phi_\kappa)$. Note that $\mathbf{x}$ is a feasible assignment only if it satisfies all the constraints. While the probabilities of satisfying constraints i and j are *not* independent, they are conditionally independent [7, Chapter 8] given the constraint parameter vector $\mathcal{M}_\theta(\phi_\kappa)$. This is because the constraint parameter vector determines the feasibility of each constraint independently. Once $\mathcal{M}_\theta(\phi_\kappa)$ is known, the probability of satisfying constraint i depends only on $\mathcal{M}_\theta(\phi_\kappa)$, and any additional information about constraint j being satisfied does not affect that probability. Because of the conditional independence, we can express the conditional probability of the assignment, $\mathbf{x}$ being feasible given $\mathcal{M}_\theta(\phi_\kappa)$, by:

$$P_\theta(Feas(\mathbf{x}, \mathcal{M}_\theta(\phi_\kappa)) | \mathcal{M}_\theta(\phi_\kappa)) = \prod_{i \in S} P_\theta(SAT_i(\mathbf{x}, \mathcal{M}_\theta(\phi_\kappa)) | \mathcal{M}_\theta(\phi_\kappa)) \quad (14)$$

Now using Eq. 8, we can model $P_\theta(Feas(\mathbf{x}, \mathcal{M}_\theta(\phi_\kappa)) | \mathcal{M}_\theta(\phi_\kappa))$, as

$$P_\theta(Feas(\mathbf{x}, \mathcal{M}_\theta(\phi_\kappa)) | \mathcal{M}_\theta(\phi_\kappa)) = \prod_{i \in S} \frac{1}{1 + \exp(g_i(\mathbf{x}, \mathcal{M}_\theta(\phi_\kappa)))} \quad (15)$$

Thus, we obtain the log-likelihood function of $\mathbf{x}$ being feasible, in the form of:

$$-\sum_{i \in \mathcal{S}} \log(1 + \exp(g_i(\mathbf{x}, \mathcal{M}_\theta(\phi_\kappa)))) = -\sum_{i \in \mathcal{S}} \text{softplus}(g_i(\mathbf{x}_\kappa^*, \mathcal{M}_\theta(\phi_\kappa))) \quad (16)$$

We point out that in the training data, we know that $\mathbf{x}^*(\mathbf{q}_\kappa, \boldsymbol{\rho}_\kappa)$ is a feasible solution. Denoting $\mathbf{x}_\kappa^*$ as the known feasible point, we can formulate a loss function by minimizing the negative log-likelihood. In our implementation, we add a margin parameter Υ as we have done for $\mathcal{L}^{IPL}$. Since this loss is designed to maximize the probability that the optimal solution remains feasible, we refer to it as the Optimality-Preserving Loss (OPL). We express it in the following form:

$$\mathcal{L}^{OPL}(\boldsymbol{\rho}_\kappa, \mathcal{M}_\theta(\phi_\kappa)) = \sum_{i \in \mathcal{S}} \text{softplus}(\Upsilon + g_i(\mathbf{x}^*(\mathbf{q}_\kappa, \boldsymbol{\rho}_\kappa), \mathcal{M}_\theta(\phi_\kappa))) \quad (17)$$

Intuition behind OPL. By replacing the *softplus* in Eq. 17 with a *Relu* function, we can write

$$\overline{\mathcal{L}^{OPL}}(\boldsymbol{\rho}_\kappa, \hat{\boldsymbol{\rho}}_\kappa) = \sum_{i \in \mathcal{S}} \max((\Upsilon + g_i(\mathbf{x}^*(\mathbf{q}_\kappa, \boldsymbol{\rho}_\kappa), \hat{\boldsymbol{\rho}}_\kappa)), 0) \quad (18)$$

where we write $\mathcal{M}_\theta(\phi_\kappa)$ as $\hat{\boldsymbol{\rho}}_\kappa$ for brevity. From this expression, it is easy to see that $\mathcal{L}^{OPL}$ attains its minimum value of zero when all constraint values are less than or equal to $-\Upsilon$, i.e., when all the constraints are satisfied with some margin.

Lemma 2. For a true parameter $\boldsymbol{\rho}_\kappa$ and a predicted parameter $\hat{\boldsymbol{\rho}}_\kappa$, both $\overline{\mathcal{L}^{OPL}}(\boldsymbol{\rho}_\kappa, \hat{\boldsymbol{\rho}}_\kappa)$ and $\overline{\mathcal{L}^{IPL}}(\boldsymbol{\rho}_\kappa, \hat{\boldsymbol{\rho}}_\kappa)$ will be zero if and only if $\mathbf{x}^*(\mathbf{q}_\kappa, \hat{\boldsymbol{\rho}}_\kappa)$, a solution of $\hat{\boldsymbol{\rho}}_\kappa$, is an optimal solution with respect to $\boldsymbol{\rho}_\kappa$.

Proof. Let $\overline{\mathcal{L}^{OPL}}(\boldsymbol{\rho}_\kappa, \hat{\boldsymbol{\rho}}_\kappa)$ and $\overline{\mathcal{L}^{IPL}}(\boldsymbol{\rho}_\kappa, \hat{\boldsymbol{\rho}}_\kappa)$ be equal to zero. We have proved in Lemma 1 that $\overline{\mathcal{L}^{IPL}}$ being zero implies that a solution of $\hat{\boldsymbol{\rho}}_\kappa$ is feasible with respect to κ . So, $\mathbf{x}^*(\mathbf{q}_\kappa, \hat{\boldsymbol{\rho}}_\kappa)$, a solution of $\hat{\boldsymbol{\rho}}_\kappa$ lies in the set $\mathcal{F}(\boldsymbol{\rho}_\kappa)$.

Because $\mathbf{x}^*(\mathbf{q}_\kappa, \boldsymbol{\rho}_\kappa)$ is an optimal solution,

$$f(\mathbf{x}^*(\mathbf{q}_\kappa, \boldsymbol{\rho}_\kappa); \mathbf{q}_\kappa) \leq f(\mathbf{x}'; \mathbf{q}_\kappa) \quad \forall \mathbf{x}' \in \mathcal{F}(\boldsymbol{\rho}_\kappa)$$

As $\mathbf{x}^*(\mathbf{q}_\kappa, \hat{\boldsymbol{\rho}}_\kappa) \in \mathcal{F}(\boldsymbol{\rho}_\kappa)$, from the above argument, we can write

$$f(\mathbf{x}^*(\mathbf{q}_\kappa, \boldsymbol{\rho}_\kappa); \mathbf{q}_\kappa) \leq f(\mathbf{x}^*(\mathbf{q}_\kappa, \hat{\boldsymbol{\rho}}_\kappa); \mathbf{q}_\kappa) \quad (19)$$

Now, $\overline{\mathcal{L}^{OPL}}(\boldsymbol{\rho}_\kappa, \hat{\boldsymbol{\rho}}_\kappa)$ being zero implies that $\mathbf{x}^*(\mathbf{q}_\kappa, \boldsymbol{\rho}_\kappa)$, an optimal solution of the true parameter, $\boldsymbol{\rho}_\kappa$ lies in $\mathcal{F}(\hat{\boldsymbol{\rho}}_\kappa)$. Note,

$$f(\mathbf{x}^*(\mathbf{q}_\kappa, \hat{\boldsymbol{\rho}}_\kappa); \mathbf{q}_\kappa) \leq f(\mathbf{x}'; \mathbf{q}_\kappa) \quad \forall \mathbf{x}' \in \mathcal{F}(\hat{\boldsymbol{\rho}}_\kappa)$$

As $\mathbf{x}^*(\mathbf{q}_\kappa, \boldsymbol{\rho}_\kappa) \in \mathcal{F}(\hat{\boldsymbol{\rho}}_\kappa)$

$$f(\mathbf{x}^*(\mathbf{q}_\kappa, \hat{\boldsymbol{\rho}}_\kappa); \mathbf{q}_\kappa) \leq f(\mathbf{x}^*(\mathbf{q}_\kappa, \boldsymbol{\rho}_\kappa); \mathbf{q}_\kappa) \quad (20)$$

Hence, Eq. 19 and Eq. 20 suggests

$$f(\mathbf{x}^*(\mathbf{q}_\kappa, \hat{\boldsymbol{\rho}}_\kappa); \mathbf{q}_\kappa) = f(\mathbf{x}^*(\mathbf{q}_\kappa, \boldsymbol{\rho}_\kappa); \mathbf{q}_\kappa)$$

As $\mathbf{x}^*(\mathbf{q}_\kappa, \hat{\boldsymbol{\rho}}_\kappa)$ is a feasible solution with respect to $\boldsymbol{\rho}_\kappa$ and its objective value is equal to the optimal objective value, it is an optimal solution.

Now, to prove the opposite direction, assume that $\mathbf{x}^*(\mathbf{q}_\kappa, \hat{\boldsymbol{\rho}}_\kappa)$ is an optimal solution with respect to $\boldsymbol{\rho}_\kappa$. This implies that $\mathbf{x}^*(\mathbf{q}_\kappa, \hat{\boldsymbol{\rho}}_\kappa)$ is a feasible solution with respect to $\boldsymbol{\rho}_\kappa$. Therefore, from Lemma 1, $\overline{\mathcal{L}^{IPL}}(\boldsymbol{\rho}_\kappa, \hat{\boldsymbol{\rho}}_\kappa) = 0$. As $\mathbf{x}^*(\mathbf{q}_\kappa, \hat{\boldsymbol{\rho}}_\kappa)$ adheres to all the constraints when $\hat{\boldsymbol{\rho}}_\kappa$ is the constraints parameter value,

$$\sum_{i \in \mathcal{S}} \max((\Upsilon + g_i(\mathbf{x}^*(\mathbf{q}_\kappa, \hat{\boldsymbol{\rho}}_\kappa), \hat{\boldsymbol{\rho}}_\kappa)), 0) = 0 \quad (21)$$

As $\mathbf{x}^*(\mathbf{q}_\kappa, \hat{\boldsymbol{\rho}}_\kappa)$ is an optimal solution with respect to $\boldsymbol{\rho}_\kappa$, $\mathbf{x}^*(\mathbf{q}_\kappa, \hat{\boldsymbol{\rho}}_\kappa)$ is one of the $\mathbf{x}^*(\mathbf{q}_\kappa, \boldsymbol{\rho}_\kappa)$. If we replace $\mathbf{x}^*(\mathbf{q}_\kappa, \hat{\boldsymbol{\rho}}_\kappa)$ with $\mathbf{x}^*(\mathbf{q}_\kappa, \boldsymbol{\rho}_\kappa)$ in Eq. 21, $\overline{\mathcal{L}^{OPL}}(\boldsymbol{\rho}_\kappa, \hat{\boldsymbol{\rho}}_\kappa) = 0$ by definition (Eq. 18). $\square$

4.3 Minimizing a Convex Combination of $\mathcal{L}^{OPL}$ and $\mathcal{L}^{IPL}$

Lemma 2 suggests that by minimizing both $\mathcal{L}^{IPL}$ and $\mathcal{L}^{OPL}$ to zero, we can obtain a solution $\mathbf{x}^*(\mathbf{q}_\kappa, \hat{\boldsymbol{\rho}}_\kappa)$, which is optimal with respect to $\boldsymbol{\rho}_\kappa$. However, simultaneously minimizing $\mathcal{L}^{IPL}$ and $\mathcal{L}^{OPL}$ presents a challenge. This is because $\mathcal{L}^{IPL}$ and $\mathcal{L}^{OPL}$ represent two conflicting goals. Minimizing only $\mathcal{L}^{OPL}$ encourages the model to predict loose constraints so that all true feasible solutions remain feasible, hence making even infeasible solutions appear feasible. Conversely, minimizing $\mathcal{L}^{IPL}$ encourages stricter constraints, which may render even true optimal solutions infeasible. We thus propose to minimize a convex combination of these losses, in the following form:

$$\alpha \mathcal{L}^{IPL}(\boldsymbol{\rho}_\kappa, \hat{\boldsymbol{\rho}}_\kappa) + (1 - \alpha) \mathcal{L}^{OPL}(\boldsymbol{\rho}_\kappa, \hat{\boldsymbol{\rho}}_\kappa) \quad (22)$$

$\alpha \in [0, 1]$ is a hyperparameter we call the *infeasibility-aversion coefficient*. It reflects how much importance the decision-maker places on avoiding infeasible solutions (through IPL) versus maintaining the feasibility of known feasible ones (through OPL). The two extremes are $\alpha = 1$ and $\alpha = 0$, both of which represent unrealistic preferences. $\alpha = 1$ means the decision-maker prioritizes only feasibility, possibly ending up with trivial solutions (e.g., picking no items in a knapsack problem). α near 0 implies a focus on high-reward solutions, even if they are infeasible (e.g., selecting all items).

In this section, we have proposed a novel approach for ‘optimizing decisions through end-to-end constraint estimation’ (Odece), involving two novel loss functions: $\mathcal{L}^{IPL}$ and $\mathcal{L}^{OPL}$. We have proposed to consider a weighted average of the two losses to reflect the decision-maker’s subjective preference of infeasibility over suboptimality. Next, we will experimentally investigate how Odece performs compared to other existing approaches for predicting constraint parameters.

5 Experimental Evaluation

We experiment with predicting constraint parameters in the following optimization problems:

Multi-dimensional knapsack problem. The first is the 0-1 multi-dimensional knapsack problem (MDKP), introduced in Section 3. In the experimental section, we consider MDKP instances with 50 items and constraints in three dimensions. We consider two settings: 1) predicting the item weight vectors while keeping the capacity constraints known, and 2) predicting the capacity vectors with known item weights.

Brass Alloy Production. Our third experimental setting involves the Brass alloy production problem. We reproduce this experimental setup from the work by Hu et al. [15]. It is a covering LP problem. We use the publicly available Brass alloy data from their repository.² The production of the Brass alloy requires two metals: Copper (Cu) and Zinc (Zn). A factory needs to purchase ores that contain these two metals. There are 10 potential suppliers, each offering ores at a different price per unit. The exact metal content in each supplier’s ore is unknown and must be predicted. The goal is to buy a combination of ores from the 10 suppliers at the lowest possible cost, while ensuring that the total purchase contains at least 627.54 units of Cu and 369.72 units of Zn.

5.1 Dataset Description

Synthetic parameter generation of the MDKP. We adopt the synthetic data generation process described by Elmachtoub and Grigas [12]. This data generation approach introduces a non-linear relationship between input features and the unknown parameters. A linear model is used to predict the parameters from features. The motivation behind using a linear model is to demonstrate that DFL methods can produce high-quality solutions despite model misspecification. This approach has been widely used as a benchmark in DFL studies [12, 20, 22, 30, 31].

Each knapsack instance contains 50 items, with all items correlated with 10 input features. Item values, which appear in the objective function, are sampled independently from a Gumbel distribution with location 100 and scale 20. For each run, 1500 instances are generated, split into 900 for training, 100 for validation, and 500 for testing. We took extra caution to ensure that the synthetically-generated parameters produce non-trivial optimal solutions, avoiding cases where all or none of the items are

²https://github.com/Elizabethxyhu/NeurIPS_Two_Stage_Predict-Optimize/

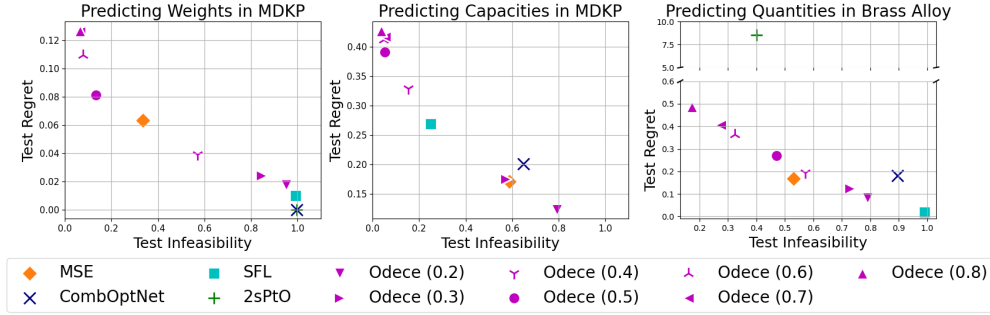


Figure 1: Infeasibility ratio and regret of feasible solutions on Test instances (best viewed in colors).

selected in the knapsack. To achieve this, we clip the capacity in each dimension to be less than half the total weight of all items and ensure that individual item weights remain below the respective capacity values.

Brass Alloy Production. For this experiment, we use the dataset from Hu et al. [15]. In this dataset, the Cu and Zn contents in each supplier's ore are predicted using separate sets of 4096 features each. Out of the available 500 instances, we use 350, 50, and 100 for training, validation, and testing, respectively. We use a fully-connected neural network with one hidden layer of 512 neurons for prediction.

The experiments were executed on an *Intel i7-13800H* (20 cores) CPU with 32GB RAM. Details on the problems and datasets used in the three experiments are provided in Appendix A. The source code for reproducing our results is publicly available at: <https://github.com/JayMan91/OdeceDFLforConstraintsNeurips25>.

5.2 Experimental Results

We denote the model as $\text{Odece}(\alpha)$ (e.g., $\text{Odece}(0.1)$) to indicate Odece is trained with that specific value of α . We compare the proposed Odece against four competitors: i) *MSE*: a PFL approach which trains the ML model to minimize the MSE loss over the parameter predictions, ii) *CombOptNet*: the technique proposed by Paulus et al. [26] to compute the gradient of ILP parameters, iii) *SFL*: the solver-free learning proposed by Nandwani et al. [24] for learning parameters of an ILP, iv) *2sPtO*: the two-stage predict+optimize approach proposed by Hu et al. [15]. We implement 2sPtO using a high penalty factor, as recommended in Appendix A.2 of their paper. We could not apply 2sPtO to predict the knapsack capacity, as it is designed to predict only the left-hand side parameters in linear constraints. More specifically, it can compute gradients with respect to b in constraints of the form $b^\top x \leq c$, but not with respect to c . For CombOptNet we compute the L1 loss of the predicted solution and the true solution and backpropagate the L1 loss through CombOptNet.

We evaluate performance using two metrics: the proportion of infeasible solutions and the normalized regret on the test data. Each technique is run five times with different random seeds. Figure 1 reports the average proportion of infeasible solutions (x-axis) and average normalized regret (y-axis) across these runs. Detailed results, including averages and standard deviations across the five runs, are provided in Appendix B. We compute regret only over predicted solutions that are feasible under the true parameters. More formally, out of K test instances, let K' denote the number of instances where the solution $x^*(q_\kappa, \hat{\rho}_\kappa)$ is feasible under ρ_κ . The proportion of infeasible solutions is then $\frac{K'}{K}$. The normalized regret is defined as follows:

$$\frac{1}{K'} \sum_{\kappa=1}^{K'} \frac{f(x^*(q_\kappa, \hat{\rho}_\kappa); q_\kappa) - f(x_\kappa^*; q_\kappa)}{f(x_\kappa^*; q_\kappa)} \quad (23)$$

Note that normalized regret can be misleading in this setting. This is because in extreme cases, where most predicted solutions are infeasible, regret is computed over only a small subset of instances.

Results. The first observation from Figure 1 is that CombOptNet exhibits relatively poor performance across the three problems. For both MDKP weight prediction and the Brass Alloy problem, the proportion of infeasible solutions generated by CombOptNet is very high ($> 85\%$). In MDKP capacity prediction, the proportion of infeasible solutions is slightly lower but still high ($> 65\%$). The performance of SFL is similar to CombOptNet for MDKP weight prediction and the Brass Alloy problem. However, in the MDKP capacity prediction problem, SFL performs significantly better, with the proportion of infeasible solutions around 25%. We suspect this is because SFL relies more on c than b to separate the true optimal solution from the *negative* assignments, resulting in more informative gradients for c and less informative gradients for b . (For example, in a 3-dimensional MDKP where the optimal solution is $[0, 0, 1]$, negative samples such as $[1, 1, 1]$, $[1, 1, 0]$, $[0, 1, 1]$ and $[1, 0, 1]$ can be classified as negative based on the capacity, i.e., the right-hand side parameter, c .)

For MDKP weight prediction, 2sPtO performs very poorly (the proportion of infeasible solutions is greater than 99%). It is not applicable for MDKP capacity prediction. Figure 1 might suggest that 2sPtO achieves a relatively lower proportion of infeasible solutions for the Brass Alloy problem, but this is misleading for two reasons. First, it results in extremely high normalized regret ($> 850\%$). Note that we have to break the y-axis in Figure 1 to accommodate 2sPtO's regret values. More importantly, its performance is highly unstable, which is not noticeable in Figure 1, where the average of five runs is plotted. Table 3 in the Appendix reveals that the standard deviation across the five runs is very high. In some runs, the proportion of infeasible solutions is near 0, while in others it reaches $\approx 99\%$. This indicates unstable learning.

Our closest competitor is MSE, but minimizing MSE does not let decision-makers adjust the trade-off between optimality and feasibility. MSE has infeasibility rates of 33%, 60%, and 53% across the three problems, respectively. For a decision-maker, these infeasibility rates may be unacceptably high. Odece allows a decision-maker to attain lower infeasibility rate by varying α , at the cost of higher normalized regret. For instance, Odece(0.8) brings the infeasibility rates down to 6%, 4%, and 18%, across the three problems, respectively. We have also conducted statistical significance tests to determine whether the test regret and infeasibility of Odece are lower than those of MSE. Since each run was conducted using the same random seed for both models, we used a paired t-test with the alternative hypothesis that MSE has higher regret and higher infeasibility than Odece. The p-values from the paired t-tests for different values of α are reported in Table 4, 5 and 6 in Appendix B. The results of the statistical significance test reveal that for $\alpha > 0.5$, Odece produces a statistically significantly lower proportion of infeasible solutions than MSE. On the other hand, for $\alpha < 0.3$, the normalized regret on feasible instances generated by Odece is statistically significantly lower than that of MSE. This holds true across all three problems.

6 Conclusion

We proposed Odece, a novel DFL technique for predicting parameters appearing in the constraints of COPs. Unlike existing DFL methods for constraint predictions – which either depend on problem-specific second-stage corrective actions or which are designed to predict the full set of constraints – Odece directly optimizes for feasibility and solution quality in a single stage. Our experiments showed that by tuning Odece's α parameter, decision-makers can flexibly manage the trade-off between infeasibility and suboptimality, allowing them to align solutions with their individual preferences.

One limitation of the current Odece implementation is that it solves the COP for each instance using the predicted parameters during training. To speed up training, future work could explore caching candidate assignments, as done by Mulamba et al. [23] for predicting objective parameters. Odece could also be extended to jointly predict both constraint and objective parameters. Finally, we plan to apply Odece to a wider range of optimization problems, such as combinatorial problems with non-linear objective and constraint functions, as well as larger, real-world applications.

Acknowledgments and Disclosure of Funding

This research received funding from the European Research Council (ERC) under the European Union's Horizon 2020 Research and Innovation Programme (Grant No. 101002802, CHAT-Opt), and from the Research Foundation Flanders (FWO) project G0G3220N. Senne Berden is a fellow of the Research Foundation-Flanders (FWO-Vlaanderen, 11PQ024N).

References

- [1] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and J. Z. Kolter. Differentiable convex optimization layers. *Advances in neural information processing systems*, 32, 2019.
- [2] B. Amos and J. Z. Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, pages 136–145. PMLR, 2017.
- [3] S. Berden, M. Kumar, S. Kolb, and T. Guns. Learning MAX-SAT models from examples using genetic algorithms and knowledge compilation. In *28th International Conference on Principles and Practice of Constraint Programming (CP 2022)*, pages 8–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2022.
- [4] S. Berden, A. İ. Mahmutoğulları, D. Tsouros, and T. Guns. Solver-free decision-focused learning for linear optimization problems. *arXiv preprint arXiv:2505.22224*, 2025.
- [5] Q. Berthet, M. Blondel, O. Teboul, M. Cuturi, J.-P. Vert, and F. Bach. Learning with differentiable perturbed optimizers. *Advances in neural information processing systems*, 33:9508–9519, 2020.
- [6] C. Bessiere, C. Carbonnel, and A. Himeur. Learning constraint networks over unknown constraint languages. In *IJCAI 2023-32nd International Joint Conference on Artificial Intelligence*, pages 1876–1883. International Joint Conferences on Artificial Intelligence Organization, 2023.
- [7] C. M. Bishop and N. M. Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.
- [8] M. Blondel, Q. Berthet, M. Cuturi, R. Frostig, S. Hoyer, F. Llinares-Lopez, F. Pedregosa, and J.-P. Vert. Efficient and modular implicit differentiation. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 5230–5242. Curran Associates, Inc., 2022.
- [9] A. Butler and R. H. Kwon. Efficient differentiable quadratic programming layers: an ADMM approach. *Comput. Optim. Appl.*, 84(2):449–476, 2023.
- [10] G. Dalle, L. Baty, L. Bouvier, and A. Parmentier. Learning with combinatorial optimization layers: a probabilistic approach, 2022. URL <https://arxiv.org/abs/2207.13513>.
- [11] M. Defresne, S. Barbe, and T. Schiex. Scalable Coupling of Deep Learning with Logical Reasoning. In *Thirty-second International Joint Conference on Artificial Intelligence, IJCAI’2023*, 2023.
- [12] A. N. Elmachetoub and P. Grigas. Smart “predict, then optimize”. *Management Science*, 68(1): 9–26, 2022.
- [13] A. Ferber, B. Wilder, B. Dilkina, and M. Tambe. Mipaal: Mixed integer program as a layer. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(02):1504–1511, Apr. 2020.
- [14] X. Hu, J. C. Lee, and J. H. Lee. Predict+ optimize for packing and covering lps with unknown parameters in constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 3987–3995, 2023.
- [15] X. Hu, J. C. Lee, and J. H. Lee. Two-stage predict+optimize for MILPs with unknown parameters in constraints. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL <https://openreview.net/forum?id=0tnhFpyWjb>.
- [16] X. Hu, J. C. H. Lee, and J. H. man Lee. Branch & learn with post-hoc correction for predict+optimize with unknown parameters in constraints. In *Integration of AI and OR Techniques in Constraint Programming*, 2023.
- [17] M. Kumar, S. Kolb, S. Teso, and L. De Raedt. Learning max-sat from contextual examples for combinatorial optimisation. *Artificial Intelligence*, 314:103794, 2023. ISSN 0004-3702. doi: <https://doi.org/10.1016/j.artint.2022.103794>. URL <https://www.sciencedirect.com/science/article/pii/S0004370222001345>.

- [18] J. Mandi and T. Guns. Interior point solving for lp-based prediction+optimisation. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 7272–7282, 2020.
- [19] J. Mandi, E. Demirović, P. J. Stuckey, and T. Guns. Smart predict-and-optimize for hard combinatorial optimization problems. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(02):1603–1610, Apr. 2020.
- [20] J. Mandi, V. Bucarey, M. M. K. Tchomba, and T. Guns. Decision-focused learning: Through the lens of learning to rank. In *International conference on machine learning*, pages 14935–14947. PMLR, 2022.
- [21] J. Mandi, M. Foschini, D. Höller, S. Thiébaux, J. Hoffmann, and T. Guns. Decision-focused learning to predict action costs for planning. In *Proceedings of the European Conference on Artificial Intelligence (ECAI-24)*, pages 4060–4067. IOS Press, 2024.
- [22] J. Mandi, J. Kotary, S. Berden, M. Mulamba, V. Bucarey, T. Guns, and F. Fioretto. Decision-focused learning: Foundations, state of the art, benchmark and future opportunities. *Journal of Artificial Intelligence Research*, 80:1623–1701, 2024.
- [23] M. Mulamba, J. Mandi, M. Diligenti, M. Lombardi, V. Bucarey, and T. Guns. Contrastive losses and solution caching for predict-and-optimize. In Z.-H. Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 2833–2840. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/390. Main Track.
- [24] Y. Nandwani, R. Ranjan, Mausam, and P. Singla. A solver-free framework for scalable learning in neural ILP architectures. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.
- [25] M. Niepert, P. Minervini, and L. Franceschi. Implicit mle: Backpropagating through discrete exponential family distributions. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 14567–14579. Curran Associates, Inc., 2021.
- [26] A. Paulus, M. Rolinek, V. Musil, B. Amos, and G. Martius. Comboptnet: Fit the right np-hard problem by learning integer programming constraints. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8443–8453. PMLR, 18–24 Jul 2021.
- [27] D. Pisinger. Where are the hard knapsack problems? *Computers & Operations Research*, 32(9): 2271–2284, 2005.
- [28] M. V. Pogančić, A. Paulus, V. Musil, G. Martius, and M. Rolinek. Differentiation of blackbox combinatorial solvers. In *International Conference on Learning Representations*, 2020.
- [29] H. Sun, Y. Shi, J. Wang, H. D. Tuan, H. V. Poor, and D. Tao. Alternating differentiation for optimization layers. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*, 2023.
- [30] B. Tang and E. B. Khalil. Pyepo: A pytorch-based end-to-end predict-then-optimize library for linear and integer programming, 2023.
- [31] B. Tang and E. B. Khalil. Cave: A cone-aligned approach for fast predict-then-optimize with binary linear programs. In B. Dilkina, editor, *Integration of Constraint Programming, Artificial Intelligence, and Operations Research - 21st International Conference, CPAIOR 2024, Uppsala, Sweden, May 28-31, 2024, Proceedings, Part II*, volume 14743 of *Lecture Notes in Computer Science*, pages 193–210. Springer, 2024.
- [32] D. Tsouros, S. Berden, and T. Guns. Learning to learn in interactive constraint acquisition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 8154–8162, 2024.

- [33] B. Wilder, B. Dilkina, and M. Tambe. Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019*, pages 1658–1665. AAAI Press, 2019.

A Problem and Dataset Description

A.1 Multi-dimensional knapsack problem

The formulation of the optimization problem is as follows:

$$\min_{x_{1:N}} \sum_{n=1}^N (-q_n) x_n \quad \text{such that} \quad x_n \in \{0, 1\}, \forall n \in [N], \quad \sum_{n=1}^N \rho_{ni} x_n \leq \rho_i, \forall i \in [M] \quad (24)$$

Here, q_n and ρ_{ni} are the values and weights in dimension i of item n . We synthetically generate 1500 MDKP instances for each run. Of these, 900 are used for training, 100 for validation, and 500 for testing. We represent the dataset as $\{(\phi_\kappa, \rho_\kappa, q_\kappa, \mathbf{x}^*(q_\kappa, \rho_\kappa))\}_{\kappa=1}^K$ where ϕ_κ is the feature vector, ρ_κ is the concatenated vector of weights and capacity, q_κ is the objective parameter, and $\mathbf{x}^*(q_\kappa, \rho_\kappa)$ denotes the solution to instance κ . In our experiment, ϕ_κ is of dimension 10. The feature vectors are sampled from a multivariate Gaussian distribution with zero mean and unit variance, i.e., $\phi_\kappa \sim \mathbf{N}(0, \mathbf{I}_{10})$.

A.1.1 Data Generation for Unknown Weight Vectors

We generate our dataset similarly to Elmachetoub and Grigas [12] and Tang and Khalil [30]. While these works consider predicting parameters in the objective function, we focus on predicting parameters in the constraints. We use the same data generation process to synthetically create constraint parameters from features.

To generate the weight vector, first a matrix $B \in \mathbb{R}^{M \times N \times 10}$ is generated, which represents the true underlying model, unknown to the decision-maker. Each entry of the matrix B is sampled from independent Bernoulli distributions of probability 0.5. The weight $\rho_{ni}^{(\kappa)}$, for κ -th instance, is then generated according to the following formula:

$$\rho_{ni}^{(\kappa)} = \left[\frac{1}{3.5^{\text{Deg}}} \left(\frac{1}{\sqrt{10}} (\phi_\kappa^\top B[i, n]) + 3 \right)^{\text{Deg}} + 1 \right] \xi_{ni}^{(\kappa)} \quad (25)$$

The Deg is ‘model misspecification’ parameter. This is because a linear model is used as a predictive model in the experiment and a higher value of Deg indicates the predictive model deviates more from the true underlying model and larger the prediction errors. $\xi_{ni}^{(\kappa)}$ is a multiplicative noise term sampled randomly from the uniform distribution $\xi_{ni}^{(\kappa)} \sim U[1 - w, 1 + w]$. We report results for $\text{Deg} = 6$ and $w = 0.25$.

We generated the capacity $\rho_i^{(\kappa)}$, the the following ways:

$$\rho_i^{(\kappa)} = r * \xi_i^{(\kappa)} \left[\sum_{n=1}^N \sum_{\iota=1}^{10} B[i, n, \iota] \right] \quad (26)$$

The term inside the summation represents an upper bound on the total weight along dimension i . Therefore, setting the capacity vector equal to this summation would always allow all items to be selected in the true solution, resulting in a trivial case. To avoid this, we add $r \in [0, 1]$. It tightens the capacity constraints, thereby preventing trivial solutions where all items are feasibly selected. In our experiments, we set r to 0.2. Due to the multiplicative noise term $\xi_i^{(\kappa)} \sim U[1 - w, 1 + w]$, the capacity vectors for each instance are not exactly the same.

A.1.2 Data Generation for Unknown Capacity Vectors

We again follow an approach similar to Tang and Khalil [30] to construct this dataset. To generate the capacity vector, we first create a matrix, $B \in \mathbb{R}^{M \times 10}$ is generated, which represents the true underlying model. Each entry of the matrix B is sampled from independent Bernoulli distributions of probability 0.5 like before. The i -th dimensional capacity $\rho_i^{(\kappa)}$, for κ -th instance, is then generated according to the following formula:

$$\rho_i^{(\kappa)} = \left[\frac{1}{3.5^{\text{Deg}}} \left(\frac{1}{\sqrt{10}} (\phi_\kappa^\top B[i]) + 3 \right)^{\text{Deg}} + 1 \right] \xi_i^{(\kappa)} \quad (27)$$

Table 1: Test Regret and Infeasibility of the Models on MDKP Weight Prediction.

Model	Infeasibility		Regret	
	Avg	Sd	Avg	Sd
MSE	0.335	0.015	0.063	0.003
CombOptNet	0.996	0.003	0.000	0.000
SFL	0.992	0.004	0.010	0.013
2sPtO	0.996	0.003	0.000	0.000
Odece(0.2)	0.952	0.020	0.017	0.002
Odece(0.3)	0.845	0.019	0.024	0.002
Odece(0.4)	0.570	0.050	0.039	0.002
Odece(0.5)	0.134	0.023	0.081	0.004
Odece(0.6)	0.079	0.027	0.110	0.009
Odece(0.7)	0.066	0.021	0.126	0.011
Odece(0.8)	0.065	0.025	0.127	0.007

Table 2: Test Regret and Infeasibility of the Models on MDKP Capacity Prediction.

Model	Infeasibility		Regret	
	Avg	Sd	Avg	Sd
MSE	0.588	0.024	0.171	0.010
CombOptNet	0.647	0.092	0.202	0.024
SFL	0.251	0.124	0.269	0.044
Odece(0.2)	0.791	0.077	0.123	0.022
Odece(0.3)	0.572	0.269	0.175	0.086
Odece(0.4)	0.153	0.184	0.330	0.097
Odece(0.5)	0.053	0.020	0.392	0.039
Odece(0.6)	0.049	0.024	0.412	0.058
Odece(0.7)	0.055	0.020	0.417	0.042
Odece(0.8)	0.038	0.008	0.428	0.042

However, directly using this capacity value would result in capacities that are too low for most instances, leading to solutions where no items are selected. To obviate such scenarios, we scale the capacity vector by a factor $r = 0.5 * N$, where N is the number of items. Further, to preserve the dependency between the feature vector and the capacity vector, we multiply the feature vector by the same scaling factor r .

A.2 Brass alloy production problem

We reproduce this experimental setup from the work by Hu et al. [15]. It is actually a covering LP problem. The production of the Brass alloy requires two metals – Copper (Cu) and Zinc (Zn). A factory needs to purchase ores that contain these two metals. These metals must be sourced from multiple suppliers, each offering metal ores at different prices per unit. The factory’s goal is to purchase a combination of ores from these suppliers that minimizes the total cost while ensuring that the total quantity of each metal meets the production requirements. The optimization problem can be formally written as:

$$\min_{x_{1:N}} \sum_{n=1}^N q_n x_n \quad \text{such that} \quad x_n \geq 0 \quad \forall n \in [N], \quad \sum_{n=1}^N \rho_{ni} x_n \geq \rho_i, \quad \forall i \in [M] \quad (28)$$

where the decision variable x_n represents the amount of ore to be purchased from supplier n , and q_n is the cost per unit from that supplier. ρ_{ni} denotes amount of metal i in one unit of ore from supplier n . The constraints ensure that the total acquired amount of each metal i across all suppliers sums up more than ρ_i , the required quantity for production.

Table 3: Test Regret and Infeasibility of the Models on Alloy Production Problem.

Model	Infeasibility		Regret	
	Avg	Sd	Avg	Sd
MSE	0.532	0.005	0.169	0.004
CombOptNet	0.896	0.233	0.181	0.002
SFL	0.991	0.013	0.019	0.001
2sPtO	0.400	0.548	8.539	6.691
Odece(0.2)	0.790	0.064	0.081	0.033
Odece(0.3)	0.729	0.078	0.123	0.025
Odece(0.4)	0.571	0.085	0.195	0.041
Odece(0.5)	0.471	0.123	0.270	0.054
Odece(0.6)	0.324	0.042	0.365	0.045
Odece(0.7)	0.276	0.026	0.406	0.037
Odece(0.8)	0.174	0.029	0.486	0.037

In the Brass alloy problem instances, there are 10 customers and the factory needs at least 627.54 units of Cu and 369.72 units of Zn. We consider the data for Brass alloy, which is publicly available in the repository ³. The dataset contains 500 instances. In each instance, there are 4096-dimensional feature vector for predicting the values of each ρ_{ni} . We use 350 instances for training, 50 for validation, and 100 for testing.

B Additional Experimental Results

In this section of the Appendix, we provide additional details on the experimental results that could not be included in the main text due to space limitations. Specifically, we report the average and standard deviation of test regret and infeasibility across five runs for each model. Table 1, Table 2 and Table 3 present the average average and standard deviation of test regret and infeasibility across five runs for MDKP weight prediction, MDKP capacity prediction and Brass alloy production problem respectively.

α	p-value (infeasibility)	p-value (regret)
0.8	3×10^{-6}	0.999
0.7	3×10^{-5}	0.999
0.6	5×10^{-6}	0.999
0.5	5×10^{-5}	0.999
0.4	0.999	1.50×10^{-4}
0.3	0.999	4×10^{-6}
0.2	0.999	3×10^{-6}

Table 4: P-values from paired t-tests comparing MSE and Odece for infeasibility and regret with different values of α for MDKP weight predictions.

Test of statistical significance. For predicting capacity, we observe in Table 5, Odece has significantly lower infeasibility and higher regret till $\alpha = 0.4$. We also observe the same for the alloy production problem in Table 6. We conducted statistical significance tests to determine whether the test regret and infeasibility of Odece are lower than those of MSE. Since each run was conducted using the same random seed for both models, we used a paired t-test with the alternative hypothesis that MSE has higher regret and higher infeasibility than Odece. The p-values from the paired t-tests for different values of α for MDKP weight predictions are reported in Table 4. It tells that for $\alpha \geq 0.5$, Odece has significantly lower infeasibility and higher regret than MSE. The opposite is true when α goes below 0.4.

³https://github.com/Elizabethxyhu/NeurIPS_Two_Stage_Predict-Optimize/

α	p-value (infeasibility)	p-value (regret)
0.8	6×10^{-7}	0.995
0.7	2×10^{-7}	0.995
0.6	9×10^{-7}	0.999
0.5	1×10^{-6}	0.997
0.4	0.003	0.999
0.3	0.450	0.545
0.2	0.997	0.040

Table 5: P-values from paired t-tests comparing MSE and Odece for infeasibility and regret with different values of α for MDKP capacity predictions.

α	p-value (infeasibility)	p-value (regret)
0.8	4×10^{-6}	0.999
0.7	7×10^{-6}	0.999
0.6	0.002	0.999
0.5	0.160	0.993
0.4	0.848	0.882
0.3	0.999	0.007
0.2	0.999	0.002

Table 6: P-values from paired t-tests comparing MSE and Odece for infeasibility and regret with different values of α for Alloy Production Problem.

C Details about Hyperparameter Configuration

Finally, in Table 7, we provide the hyperparameter configurations used for each model to reproduce the results reported above.

Table 7: Hyperparameter configurations for each model and task.

Task	Hyperparameter	MSE	Odece	CombOptNet	SFL	2sPtO
MDKP Weight Prediction	Learning Rate	0.05	0.05	0.05	0.05	0.05
	τ (CombOptNet)			0.50		
	Temperature (SFL)				0.50	
	damping (2sPtO)					0.01
	thr (2sPtO)					0.1
MDKP Capacity Prediction	Learning Rate	0.005	0.005	0.005	0.01	
	τ (CombOptNet)			0.1		
	Temperature (SFL)				0.1	
Brass Alloy	Learning Rate	0.001	0.001	0.005	0.001	0.05
	τ (CombOptNet)			0.5		
	Temperature (SFL)				0.5	
	damping (2sPtO)					0.01
	thr (2sPtO)					0.1

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [\[Yes\]](#)

Justification: Yes, the main claims in the abstract and introduction accurately reflect the paper's contributions and scope. Yes, the main claims in the abstract and introduction accurately reflect the paper's contributions and scope. The paper is motivated by the need to develop a novel approach for predicting unknown parameters of an optimization problem using correlated features, while explicitly accounting for the subjective preferences of the decision-maker regarding infeasibility and suboptimality. This motivation, along with the core contributions, is clearly stated in both the abstract and the introduction.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: Yes, the paper provides a brief summary of the limitations, which we aim to address in future work.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Yes, we have provided proofs for the theorems proposed in the paper.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We are providing the code and data to reproduce the main experimental results of the paper.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [\[Yes\]](#)

Justification: We will make the data and code publicly accessible upon acceptance of the paper.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [\[Yes\]](#)

Justification: Yes, we have described the training, test, and validation split, as well as the approach used for hyperparameter selection. For detailed information on the hyperparameters, we refer the reader to the Appendix C.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [\[Yes\]](#)

Justification: In the main text, we report only average of five runs for all the techniques. But we report standard deviation and conduct statistical significance in Appendix B.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [\[Yes\]](#)

Justification: We have provided sufficient information on the computational resources used.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [\[Yes\]](#)

Justification: We do not resort to crowdsourcing or contract work for our research.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [\[NA\]](#)

Justification: Our work is on fundamental research and we do not find any direct path to any negative societal impacts.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: The paper presents a fundamental research and does not release any model or data that can be easily misused.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: Yes, we cite the source of the data used in the paper

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [\[Yes\]](#)

Justification: We will be making our implementation publicly available along with documentation to support reproducibility.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [\[NA\]](#)

Justification: No human subjects are involved in this research.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [\[NA\]](#)

Justification: No human subjects are involved in this research.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: Our research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.