From Sequential to Recursive: Enhancing Decision-Focused Learning with Bidirectional Feedback

Xinyu Wang, Jinxiao Du, Yiyang Peng, Wei Ma*

The Hong Kong Polytechnic University, Hong Kong SAR, China
{xinyuu.wang, jinxiao.du, yiyang.peng}@connect.polyu.hk, wei.w.ma@polyu.edu.hk

Abstract

Decision-focused learning (DFL) has emerged as a powerful end-to-end alternative to conventional predict-then-optimize (PTO) pipelines by directly optimizing predictive models through downstream decision losses. Existing DFL frameworks are limited by their strictly sequential structure, referred to as sequential DFL (S-DFL). However, S-DFL fails to capture the bidirectional feedback between prediction and optimization in complex interaction scenarios. In view of this, we first time propose recursive decision-focused learning (R-DFL), a novel framework that introduces bidirectional feedback between downstream optimization and upstream prediction. We further extend two distinct differentiation methods: explicit unrolling via automatic differentiation and implicit differentiation based on fixed-point methods, to facilitate efficient gradient propagation in R-DFL. We rigorously prove that both methods achieve comparable gradient accuracy, with the implicit method offering superior computational efficiency. Extensive experiments on both synthetic and real-world datasets, including the newsvendor problem and the bipartite matching problem, demonstrate that R-DFL not only substantially enhances the final decision quality over sequential baselines but also exhibits robust adaptability across diverse scenarios in closed-loop decision-making problems.

Introduction

Real-world operational tasks, such as vehicle routing, power generation scheduling, and inventory management, frequently involve decision-making under uncertainties (Donti, Amos, and Kolter 2017; Qi et al. 2023; Sadana et al. 2025; Elmachtoub and Grigas 2022). Conventional *Predict-then-Optimize (PTO)* framework tackles such tasks by first predicting uncertain parameters through machine learning (ML), then solving optimization based on predictions. While widely adopted, PTO often yields suboptimal decisions by minimizing intermediate prediction errors rather than final decision quality (Elmachtoub and Grigas 2022; Mandi et al. 2024; Wang et al. 2025; Bertsimas and Kallus 2020). *Decision-Focused Learning (DFL)* represents a significant advancement by embedding optimization directly into the learning process, thereby minimizing end-to-end decision errors rather than prediction losses (Amos and Kolter 2017; Mandi et al. 2022; Kotary et al. 2021; Wang et al. 2025).

However, traditional DFL maintains a sequential prediction-then-optimize structure, which we term *Sequential DFL (S-DFL)* in this paper. The one-way assumption in S-DFL: predictions inform optimization, but optimization outcomes do not influence subsequent predictions, renders S-DFL inadequate for complex, interactive systems where decisions generate feedback that should recursively refine predictions. Consider the ride-hailing matching problem as a canonical example: when the platform proposes driver-rider matching pairs, user accept/reject decisions provide immediate feedback that should inform the matching decisions (Qin et al. 2020). This scenario exemplifies a broader class of multi-stakeholder adaptive systems, including dynamic pricing scenarios (Levina et al. 2009; Jia, Tong, and Zhao 2014) and task allocation systems (Zhao et al. 2020), which share three common characteristics: (*i*) Initial predictions guide decisions, (*ii*) Decision outcomes generate observable feedback, and (*iii*) The feedback should recursively refine predictions. Such a recursive interaction creates a closed-loop coupling that S-DFL fails to capture, potentially leading to unstable training and suboptimal outcomes.

The first research question arises: How can we effectively model bidirectional prediction-optimization systems with mutually interdependent components? To answer the question, we first time propose *Recursive Decision-Focused Learning (R-DFL)*, an extension of S-DFL that explicitly models the bidirectional feedback between upstream prediction and downstream optimization. As illustrated in Figure 1, R-DFL retains the forward prediction-to-optimization pipeline of S-DFL while innovatively introducing a feedback loop from optimization back to prediction. In this R-DFL framework: (*i*) The optimization module leverages early-stage predictions to generate decisions, and (*ii*) The decision outcomes, in turn, refine the predictions through feedback. This closed-loop architecture explicitly captures the dynamic interdependency characteristics inherent in bidirectional systems, aiming to enhance prediction accuracy and enable more adaptive and robust optimization compared with S-DFL.

The second research question arises: How does gradient propagation operate in the bidirectional structure

*Corresponding Author.

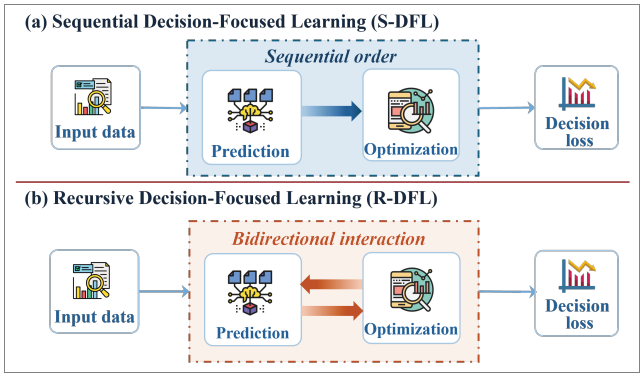

Figure 1: Comparison of S-DFL and R-DFL. R-DFL extends S-DFL by capturing the bidirectional interaction between prediction and optimization, where predictions are informed by both input features and optimization feedback.

of the R-DFL framework? The recursive framework introduces fundamental computational challenges: the cycle between prediction and optimization creates directed cyclic graphs, violating the directed acyclic graph (DAG) assumption underlying conventional deep learning (DL). Therefore, the resulting cyclic dependencies disrupt conventional gradient flow patterns and prevent a straightforward application of standard backpropagation via auto-differentiation (AD), necessitating specialized differentiation methods that preserve the recursive benefits while maintaining training stability and convergence.

To address the backpropagation challenge, we propose two differentiation schemes: explicit unrolling methods and implicit differentiation methods (Monga, Li, and Eldar 2021). The explicit unrolling method handles cyclicity by sequentially unrolling the bidirectional prediction-optimization over a fixed number of iterations during the forward pass, allowing gradients to backpropagate through all unrolled layers via standard AD. While benefiting from straightforward implementation, the explicit unrolling method introduces significant computational overhead that scales with the unrolling depth, making it computationally complicated for deep recursive systems.

In contrast, the implicit differentiation method formulates the coupled system as a fixed-point problem. During the forward pass, equilibrium states are computed using `RootFind` methods, while backward propagation leverages the Implicit Function Theorem (Krantz and Parks 2002) to derive gradients directly at equilibrium points. The implicit differentiation method offers substantial computational advantages by eliminating the need for unrolling computations, hence significantly reducing training time.

We rigorously establish the equivalence of gradient updates between explicit unrolling and implicit differentiation methods through the Neumann series (Liao et al. 2018; Dimitrov and Coelho 2017). This theoretical result guarantees consistent final accuracy performance regardless of the implementation choice of explicit unrolling and implicit differentiation methods in the R-DFL framework.

In summary, we present four major contributions:

- We first time propose R-DFL, a novel recursive decision-focused learning framework that models bidirectional prediction-optimization feedback within the deep learning architecture, enabling closed-loop decision-making in dynamic systems.
- We develop two differentiation schemes for solving R-DFL with gradient propagation: (*i*) A *multi-layer* explicit unrolling method via auto-differentiation, and (*ii*) A *single-layer* implicit differentiation method that computes equilibrium gradients directly via the Implicit Function Theorem.
- We rigorously establish the *gradient equivalence* between both differentiation schemes, guaranteeing consistent accuracy performance for explicit unrolling and implicit differentiation methods.
- Extensive experiments on both synthetic and real-world datasets, including newsvendor and bipartite matching problems, demonstrate that the R-DFL framework significantly enhances the final decision quality. Comparing the two differentiating methods, while the explicit unrolling methods provide a more straightforward implementation, they often come with higher computational costs. The implicit differentiation methods require manual derivation of gradients but deliver superior computational efficiency.

Related Works

Decision-Focused Learning

DFL has emerged as an effective end-to-end framework to improve the decision quality by integrating predictive (prediction) and prescriptive (optimization) analytics within a unified DL architecture (Bertsimas and Kallus 2020; Mandi et al. 2022; Postek and Hertog 2016). In this paper, we refer to this approach as Sequential DFL (S-DFL). S-DFL diverges from traditional PTO by embedding the optimization module as an implicit differentiable layer within the deep learning architecture, mapping input parameters directly to optimal solutions and incorporating decision quality into the loss function, allowing the predictive model to be trained using task-specific errors. Therefore, gradients can backpropagate efficiently through the optimization layer to parameters in the predictive models (Mandi et al. 2020; Berthet et al. 2020). To improve final decision accuracy, one option is to develop novel surrogate loss functions (Kotary et al. 2022; Elmachtoub and Grigas 2022). Researchers have also developed specialized methods tailored to embedding distinct classes of optimization problems, including discrete optimization (Mandi et al. 2024; Ferber et al. 2020), linear programming (Mandi and Guns 2020), and quadratic programming (Amos and Kolter 2017). For handling non-smooth optimization problems, relaxation and approximation techniques have also been proposed (Wilder, Dilkina, and Tambe 2019; Wang et al. 2019).

The development of specialized software packages has significantly lowered the implementation barrier for S-DFL. Tools like `OptNet` for quadratic programming (Amos and

Kolter 2017), `CvxpyLayers` for convex programming (Agrawal et al. 2019), and `PyEPO` for linear programming (Tang and Khalil 2022) have enabled widespread real-world applications in domains such as inventory management and energy grid scheduling (Elmachtoub and Grigas 2022; Donti, Amos, and Kolter 2017). These advances have established S-DFL as a versatile framework for decision-making under uncertainty. **Despite its successes, S-DFL remains limited to sequential prediction-optimization pipelines and lacks generalization to bidirectional settings where prediction and optimization interact dynamically.**

Explicit Unrolling Methods

The derivation of gradients for solving DFL can be categorized into two distinct approaches based on their differentiation mechanisms: explicit unrolling and implicit differentiation. The explicit unrolling methods have found widespread applications in both time series modeling (e.g., recurrent neural networks like ResNet (He et al. 2016)) and differentiable optimization. The explicit unrolling methods work by expanding either the temporal sequence (for time-series models) or the complete optimization procedure (for differentiable optimization) into an explicit computational graph during the forward pass. Each iteration is treated as a separate layer, creating a finite, unrolled representation that can be processed using standard automatic differentiation frameworks (Kotary, Dinh, and Fioretto 2023). During backpropagation, gradients are calculated by sequentially applying the chain rule through all unrolled layers. Despite its widespread adoption, the explicit unrolling methods present several critical limitations. For problems requiring numerous iterations, the computational graph becomes excessively large, leading to (*i*) substantial memory overhead, (*ii*) increased computational time, and (*iii*) potential numerical instability in gradient flow. These challenges mirror the vanishing and exploding gradient problems commonly observed in deep recurrent neural networks (Monga, Li, and Eldar 2021), significantly constraining the applicability of explicit unrolling methods to complex, iterative problems.

Implicit Differentiation Methods

Implicit differentiation methods offer a powerful alternative to explicit approaches by replacing iterative procedures with a single implicit layer, significantly reducing the complexity of the corresponding computational graph. In this paradigm, the layer output is defined implicitly as the solution to an equilibrium equation rather than through an explicit computational graph. During backpropagation, gradients are computed directly at the solution point using the Implicit Function Theorem, which avoids the need to store intermediate states (Agrawal et al. 2019). Implicit differentiation methods have found successful applications across multiple domains. In time-series modeling, the implicit methods have been employed in Neural ODEs (Chen et al. 2018) and Deep Equilibrium Models (Bai, Kolter, and Koltun 2019), where fixed-point methods identify equilibrium states (El Ghaoui et al. 2021; Zhang et al. 2019; Kazi and Thompson 2017; Li et al. 2020). In differentiable optimization, implicit differentiation has been primarily implemented through differentiation of Karush-Kuhn-Tucker (KKT) conditions (Amos and Kolter 2017; Agrawal et al. 2019; Wilder, Dilkina, and Tambe 2019). Comparative studies highlight several advantages of implicit differentiation methods over explicit unrolling approaches. As noted by Scellier and Bengio (2017), implicit differentiation methods provide better numerical stability while achieving superior memory and computational efficiency due to their compact representation. However, implicit methods typically require manual derivation of gradient expressions. Modern machine learning frameworks like PyTorch have addressed this challenge by providing native support for automatic differentiation and interfaces for custom gradient rules, making implicit differentiation methods more accessible to practitioners (Paszke et al. 2017). **Overall, both explicit and implicit methods for R-DFL are rarely explored.**

Preliminaries

In this section, we briefly introduce notations and mathematical formulations of previous work on S-DFL.

S-DFL

The S-DFL pipeline comprises two essential modules: a predictive model $\mathcal{F}_\theta$ and a convex optimization model $\mathcal{G}$.

Predictive Model Let $\mathcal{F}_\theta : \mathbb{R}^d \to \mathbb{R}^n$ be a parametric function with parameters $\theta \in \mathbb{R}^d$ that maps input features $\boldsymbol{v} \in \mathbb{R}^d$ to predicted outputs $\hat{\boldsymbol{c}} := \mathcal{F}_\theta(\boldsymbol{v})$. $\hat{\boldsymbol{c}} \in \mathbb{R}^n$ then serve as parameters for the downstream optimization problem $\mathcal{G}$.

Convex Optimization Model Define a convex program $\mathcal{G} : \mathbb{R}^n \to \mathbb{R}^n$ with objective g and constraints $\boldsymbol{h}$ that takes input parameters $\hat{\boldsymbol{c}}$ and returns optimal solutions $\boldsymbol{x}^*(\hat{\boldsymbol{c}})$ as:

$$\mathcal{G} : \boldsymbol{x}^*(\hat{\boldsymbol{c}}) = \underset{\boldsymbol{x} \in \mathcal{A}}{\operatorname{argmin}}\ g(\boldsymbol{x}; \hat{\boldsymbol{c}}), \tag{1}$$

where the differentiable convex objective function is $g : \mathbb{R}^n \to \mathbb{R}$, $\boldsymbol{x} \in \mathcal{A}$ is the feasible region with linear constraints $\boldsymbol{h}(\boldsymbol{x}) \leq \mathbf{0}$, $\boldsymbol{h} : \mathbb{R}^n \to \mathbb{R}^m$. For simplicity, we omit the unpredictable parameters in the notation of $\mathcal{G}$.

Loss Function The goal of S-DFL is to minimize the expected decision regret (Elmachtoub and Grigas 2022): $\mathcal{R}(\hat{\boldsymbol{c}}, \boldsymbol{c}) = g(\boldsymbol{x}^*(\hat{\boldsymbol{c}}), \boldsymbol{c}) - g(\boldsymbol{x}^*(\boldsymbol{c}), \boldsymbol{c})$. If given a dataset $\mathcal{D} = \{(v_n, c_n)\}_{n=1}^N$ with N samples, the empirical loss function is: $\mathcal{L}_{(\boldsymbol{v},\boldsymbol{c})\in\mathcal{D}} = \frac{1}{N}\sum_{n=1}^N \mathcal{R}(\mathcal{F}_\theta(\boldsymbol{v}_n), \boldsymbol{c}_n)$.

Gradient Computation To update the parameters θ in the predictive model via gradient descent methods, one can derive the gradient through the chain rule: $\frac{\partial \mathcal{L}}{\partial \theta} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{x}^*}\frac{\partial \boldsymbol{x}^*}{\partial \hat{\boldsymbol{c}}}\frac{\partial \hat{\boldsymbol{c}}}{\partial \theta}$, where $\frac{\partial \mathcal{L}}{\partial \boldsymbol{x}^*}$ is the gradient of loss function w.r.t. the optimal decision, $\frac{\partial \boldsymbol{x}^*}{\partial \boldsymbol{c}}$ is obtained from the KKT condition by differentiating the convex optimization problem $\mathcal{G}$ or by unrolling the optimization procedures, and $\frac{\partial \hat{\boldsymbol{c}}}{\partial \theta}$ is the gradient of the output in the predictive model $\mathcal{F}_\theta$ w.r.t. parameters θ.

Recursive Decision-Focused Learning

In this section, we introduce the proposed Recursive Decision-Focused Learning (R-DFL) framework with two

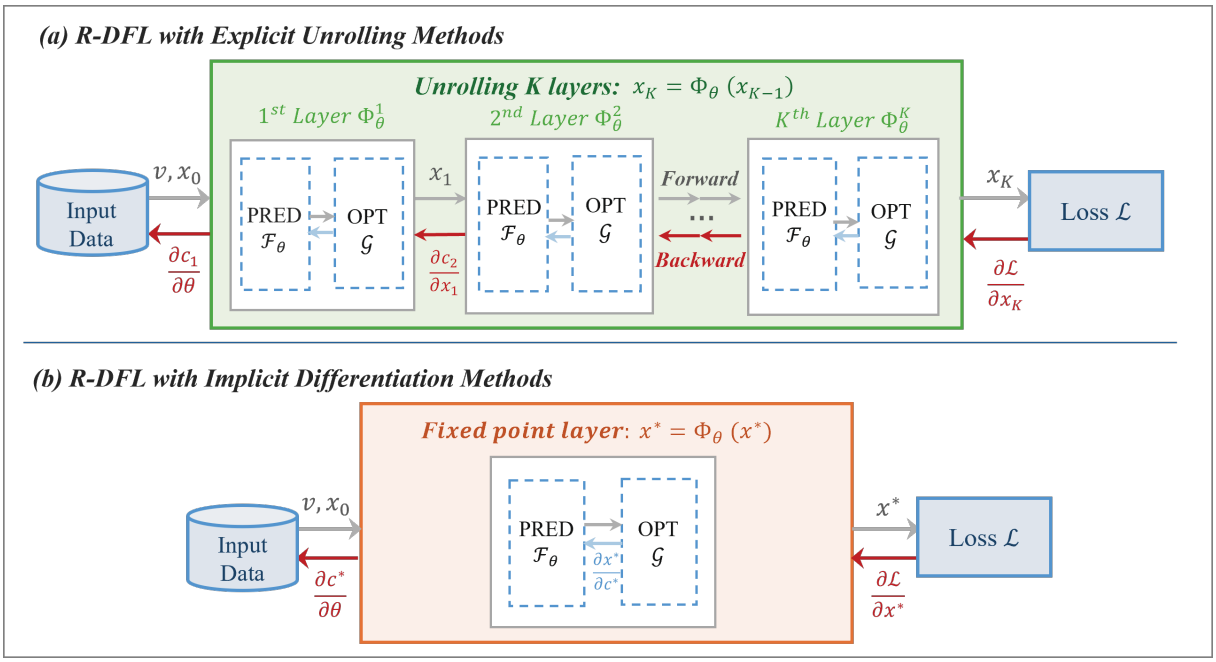

Figure 2: Illustration of the R-DFL framework with explicit unrolling and implicit differentiation methods.

gradient propagation schemes: explicit unrolling and implicit differentiation, as illustrated in Figure 2. Here we use the same notations of $\mathcal{F}_\theta$, $\mathcal{G}$, and g in both S-DFL and R-DFL to smooth the presentation. Unlike conventional sequential approaches, the R-DFL models prediction and optimization with bidirectional feedback.The predictive model $\mathcal{F}_\theta$ takes inputs from both feature vector $\boldsymbol{v}$ and optimization results $\boldsymbol{x}$, then generates outputs $\hat{\boldsymbol{c}}$ to optimization model $\mathcal{G}$ as parameters.

Overview of R-DFL Framework

The R-DFL pipeline comprises two essential modules: a predictive model $\mathcal{F}_\theta$ and a convex optimization model $\mathcal{G}$.

Predictive Model Let $\mathcal{F}_\theta : \mathbb{R}^{d+n} \to \mathbb{R}^n$ be a parametric predictive model with parameter $\theta \in \mathbb{R}^{d+n}$ that maps feature vectors $\boldsymbol{v} \in \mathbb{R}^d$ and optimization result $\boldsymbol{x} \in \mathbb{R}^n$ to predicted results $\hat{\boldsymbol{c}} \in \mathbb{R}^n$: $\hat{\boldsymbol{c}} = \mathcal{F}_\theta(\boldsymbol{x}, \boldsymbol{v})$.

Convex Optimization Model Define a convex optimization model $\mathcal{G}$ with differentiable objective function g and linear constraints $\boldsymbol{h}$, takes the input from the predictive model $\hat{\boldsymbol{c}}$, and generates the optimal results:

$$\mathcal{G} : \boldsymbol{x}^*(\hat{\boldsymbol{c}}) = \underset{\boldsymbol{x} \in \mathcal{A}}{\operatorname{argmin}}\ g(\boldsymbol{x}; \hat{\boldsymbol{c}}), \tag{2}$$

where $g : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is a differentiable convex function in $\boldsymbol{x}$ for fixed $\hat{\boldsymbol{c}}$, $\boldsymbol{x} \in \mathcal{A}$ is the feasible region with linear constraints $\boldsymbol{h}(\boldsymbol{x}) \leq \boldsymbol{0}$, $\boldsymbol{h} : \mathbb{R}^n \to \mathbb{R}^m$, $\boldsymbol{x}^*(\hat{\boldsymbol{c}})$ is the optimal solution. We consider the case with two assumptions in the optimization model $\mathcal{G}$:

- *Single-valued, differentiable and convex*. The optimization model $\mathcal{G}$ induces a (potentially multi-valued) mapping from parameters to solutions. In our framework, we restrict $\mathcal{G}$ to be single-valued, differentiable, and convex (Diamond and Boyd 2016).
- *Parameter only in objective*. The predicted parameter $\hat{\boldsymbol{c}}$ appears exclusively in the objective function, while constraints remain fixed, guaranteeing that the KKT conditions yield a well-defined differentiable mapping from predictions to decisions.

Loss Function To minimize the expected decision regret, the loss function of R-DFL is defined as:

$$\mathcal{L} = \mathbb{E}_{\boldsymbol{c},\boldsymbol{v}}\left[\mathcal{R}\big(\mathcal{F}_\theta(\boldsymbol{v}, \boldsymbol{x}^*(\hat{\boldsymbol{c}})), \boldsymbol{c}\big)\right]. \tag{3}$$

Gradient Computation To update the parameters θ in the predictive model, we need to compute the gradient $\frac{\partial \mathcal{L}}{\partial \theta}$ in the R-DFL framework. However, the recursive interaction between $\mathcal{F}_\theta$ and $\mathcal{G}$ in R-DFL prevents direct application of the chain rule like S-DFL. We address it through explicit unrolling and implicit differentiation methods below.

Explicit Unrolling Methods

As illustrated in Figure 2(a), the first explicit differentiation method unrolls K iterations of the prediction-optimization cycle, treating each iteration as a separate layer in the computational graph. This allows gradient backpropagation through the entire unrolled sequence.

Forward Pass In each separate layer in the K iterations, we compute predicition and optimization in sequence:

$$\hat{\boldsymbol{c}}_i = \mathcal{F}_\theta(\boldsymbol{x}_{i-1}, \boldsymbol{v}), \quad \boldsymbol{x}_i = \mathcal{G}(\hat{\boldsymbol{c}}_i), \quad i = 1, 2, \ldots, K \tag{4}$$

Define the composed prediction-optimization layer as $\Phi_\theta = \mathcal{G} \circ \mathcal{F}_\theta$, such that each layer in the unrolling process in Equation 4 is compactly expressed to:

$$\boldsymbol{x}_i = \mathcal{G}(\mathcal{F}_\theta(\boldsymbol{x}_{i-1}, \boldsymbol{v})) = \Phi_\theta(\boldsymbol{x}_{i-1}, \boldsymbol{v}), \quad i = 1, 2, \ldots, K \tag{5}$$

where $\boldsymbol{x}_0$ is initialized randomly, and all K layers in Φ_θ share the same θ.

Then the full unrolling computation of a total of K layers in the forward pass yields (Omit $\boldsymbol{v}$ for simplicity) :

$$\boldsymbol{x}_K = \Phi_\theta(\boldsymbol{x}_{K-1}) \circ \ \Phi_\theta(\boldsymbol{x}_{K-2}) \circ \cdots \circ \Phi_\theta(\boldsymbol{x}_0). \quad (6)$$

The final output $\boldsymbol{x}_K$ serves as input to the loss function $\mathcal{L}$.

Backward Pass The gradient backpropagates sequentially through each composite layer Φ_θ from layer K to layer 1 in the computational graph. The complete gradient $\frac{\partial \mathcal{L}}{\partial \theta}$ for the explicit unrolling methods is formally given in Theorem 1.

Theorem 1 (**Gradient of Explicit Unrolling Methods**). *Define the Jacobian of function Φ_θ at $\boldsymbol{x}_i$ to be:*

$$J_{\Phi_\theta}|_{\boldsymbol{x}_i} = \frac{\partial \boldsymbol{x}_{i+1}}{\partial \boldsymbol{x}_i} = \frac{\partial \Phi_\theta(\boldsymbol{x}_i)}{\partial \boldsymbol{x}_i}, \ \forall i \in \{1, ..., K-1\}, \quad (7)$$

The Jacobian $J_{\Phi_\theta}|_{\boldsymbol{x}_i}$ is computed as:

$$J_{\Phi_\theta}|_{\boldsymbol{x}_i} = J_{\mathcal{G}}|_{\boldsymbol{c}_{i+1}} \cdot J_{\mathcal{F}_\theta}|_{\boldsymbol{x}_i}. \ \forall i \in \{1, ..., K-1\} \quad (8)$$

Then the loss gradient $\mathcal{L}$ w.r.t. θ at the final K^{th} layer is:

$$\frac{\partial \mathcal{L}}{\partial \theta} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{x}_K} \Sigma_{i=1}^{K} \left(\left(\prod_{j=i+1}^{K} J_{\Phi_\theta}|_{\boldsymbol{x}_{j-1}} \right) \frac{\partial \Phi_\theta(\boldsymbol{x}_{i-1})}{\partial \theta} \right). \quad (9)$$

Proof. See Appendix A.1.

Theorem 1 presents the general form of the task-specific loss gradient in the backward pass of the explicit methods through the chain rule by AD. Appendix B.1 shows the pseudocode of the explicit unrolling methods. Below, we give brief explanations:

- The product term $\prod_{j=i+1}^{K} J_{\Phi_\theta}|_{\boldsymbol{x}_{j-1}}$ captures the indirect cumulative gradients from path $\boldsymbol{x}_K \to \boldsymbol{x}_i$.
- The partial derivative $\frac{\partial \Phi_\theta(\boldsymbol{x}_{i-1})}{\partial \theta}$ refers to the direct gradient dependence of $\boldsymbol{x}_i$ on θ at each unrolling step.
- Derivation of Equation 8 is provided in Appendix A.5. The $J_{\mathcal{G}}|_{\boldsymbol{c}_i}$ is obtained from conducting sensitivity analysis on the KKT condition of optimization model $\mathcal{G}$, and $J_{\mathcal{F}_\theta}|_{\boldsymbol{x}_i}$ is the gradients in the predictive model.

Implicit Differentiation Methods

The implicit differentiation method bypasses layer-by-layer unrolling by deriving directly at the equilibrium point through fixed-point methods, as illustrated in Figure 2(b).

Forward Pass The forward pass computes the equilibrium state $\boldsymbol{x}^*$ through `RootFind` of the composed model Φ_θ. At convergence, the equilibrium point satisfies:

$$\boldsymbol{x}^* = \mathcal{G}(\mathcal{F}_\theta(\boldsymbol{x}^*, \boldsymbol{v})) = \Phi_\theta(\boldsymbol{x}^*, \boldsymbol{v}). \quad (10)$$

Let $\mathcal{H}_\theta$ denote the fixed-point layer (Omit $\boldsymbol{v}$ for simplicity):

$$\mathcal{H}_\theta = \boldsymbol{x}^* - \Phi_\theta(\boldsymbol{x}^*) \to 0, \quad (11)$$

where the equilibrium point $\boldsymbol{x}^*$ is the root of $\mathcal{H}_\theta$. At convergence, prediction results $\boldsymbol{c}^*$ is obtained as: $\boldsymbol{c}^* = \mathcal{H}_\theta(\boldsymbol{x}^*)$.

Note that alternative methods can be employed to achieve faster convergence guarantees, rather than relying on fixed-point iterations. For instance, if the composite function $\mathcal{H}$ is differentiable and convex, Newton's method or quasi-Newton methods can be used.

Backward Pass The gradient of the implicit methods directly back propagates at the equilibrium point.

Theorem 2 (**Gradient of Implicit Differentiation Methods**). *The loss gradient back propagates at the equilibrium point $\boldsymbol{x}^*$ w.r.t. θ is given as:*

$$\frac{\partial \mathcal{L}}{\partial \theta} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{x}^*} (I - J_{\Phi_\theta}|_{\boldsymbol{x}^*})^{-1} \frac{\partial \Phi_\theta(\boldsymbol{x}^*)}{\partial \theta}, \quad (12)$$

where $J_{\Phi_\theta}|_{\boldsymbol{x}^*}$ is the Jacobian of Φ_θ evaluated at $\boldsymbol{x}^*$.

Proof. See Appendix A.2.

Theorem 2 presents the general form of the total loss gradient of the implicit method derived through the Implicit Function Theorem (Krantz and Parks 2002). See Appendix B.2 for pseudocode of the implicit methods. Below, we give brief explanations:

- In the forward `RootFind` procedures, we **disable** forward gradient tracking until finding the equilibrium point, ensuring the backward propagation operates directly from the equilibrium point $\boldsymbol{x}^*$ to the parameter θ.
- The implicit differentiation method computes exact gradients through the inverse Jacobian term $(I - J_{\Phi_\theta}|_{\boldsymbol{x}^*})^{-1}$, which implicitly captures the full unrolling structure of the forward pass.

Gradient Equivalence of Unrolling and Implicit Differentiation Methods

We first present the convergence of explicit unrolling methods through Polyak-Łojasiewicz Inequality (PL) condition (Xiao, Lu, and Chen 2023), then prove the equivalent gradient of the explicit unrolling and implicit methods using the Neumann Series (Dimitrov and Coelho 2017).

Lemma 1 (**Convergence of Explicit Unrolling Methods**). *Suppose the gradient of the differentiable function Φ_θ is Lipschitz smooth, and Φ_θ satisfies the PL condition with $\mu > 0$:*

$$\frac{1}{2} \|\nabla \Phi_\theta(\boldsymbol{x})\|^2 \geq \mu(\Phi_\theta(\boldsymbol{x}) - \Phi_\theta^*), \quad (13)$$

where $\Phi_\theta^* = \inf_{\boldsymbol{x}} \Phi_\theta(\boldsymbol{x}) = \boldsymbol{x}^*$.

Then Φ_θ will converge to the fixed point $\boldsymbol{x}^*$ after infinite K iterations $\lim_{K \to \infty} \boldsymbol{x}_K = \boldsymbol{x}^*$.

Proof. See Appendix A.4.

Theorem 3 (**Gradient Equivalence of Explicit Unrolling and Implicit Differentiation Methods**). *Assume that after infinite K iterations, $\boldsymbol{x}_K$ converges to a fixed point $\boldsymbol{x}^*$,:*

$$\lim_{K \to \infty} \boldsymbol{x}_K = \boldsymbol{x}^*, \quad \boldsymbol{x}^* = \mathcal{G}\big(\mathcal{F}_\theta(\boldsymbol{x}^*)\big) = \Phi_\theta(\boldsymbol{x}^*). \quad (14)$$

Then for any smooth loss $\mathcal{L}$, if the spectral radius holds $\rho(J_{\Phi_\theta}|\boldsymbol{x}^*) < 1$, the loss gradients sasitfy:

$$\frac{\partial \mathcal{L}}{\partial \theta} = \lim_{K \to \infty} \frac{\partial \mathcal{L}}{\partial \boldsymbol{x}_K} \sum_{i=1}^{K} \left(\left(\prod_{j=i+1}^{K} J_{\Phi_\theta}|\boldsymbol{x}_{j-1} \right) \frac{\partial \Phi_\theta(\boldsymbol{x}_{i-1})}{\partial \theta} \right) \quad (15)$$

$$= \frac{\partial \mathcal{L}}{\partial \boldsymbol{x}^*} (I - J_{\Phi_\theta}|_{\boldsymbol{x}^*})^{-1} \frac{\partial \Phi_\theta(\boldsymbol{x}^*)}{\partial \theta}. \quad (16)$$

Proof. See Appendix A.5.

Theorem 3 states that the gradient obtained by unrolling infinite steps in Equation 9 agrees with the one given by implicit differentiation via fixed point in Equation 12.

Dataset	Newsvendor Problem						Bipartite Matching Problem					
Scalability	Small		Mid		Large		Small		Mid		Large	
Decision variable	10		50		100		16		225		900	
Constraints	32		152		302		57		706		2761	
Jacobian matrix	32×32		152×152		302×302		57×57		706×706		2761×2761	
Metrics	RMSE	Time	RMSE	Time	RMSE	Time	RMSE	Time	RMSE	Time	RMSE	Time
PTO	12.771	-	12.747	-	12.684	-	0.412	-	0.232	-	0.190	-
S-DFL	12.245	-	12.536	-	12.649	-	0.408	-	0.231	-	0.187	-
R-DFL-U	8.983	135	9.173	369	9.343	422	**0.396**	65	0.222	432	**0.170**	2704
R-DFL-I	**8.831**	**118**	**9.106**	**254**	**9.327**	**369**	0.398	**26**	**0.220**	**65**	0.171	**1867**

Table 1: Performance of R-DFL framework with explicit unrolling and implicit differentiation methods on the newsvendor and bipartite matching problems. Jacobian size indicates dimension of matrix $J_{\mathcal{G}}|_{\boldsymbol{c}_{i+1}}$). Unit for time: seconds.

Numerical Experiments

We evaluate the effectiveness of the proposed R-DFL on two problems using various datasets at different scales: a modified classical multi-product newsvendor problem (MPNP) on a synthetic dataset and a bipartite matching problem (BMP) on a real-world dataset. All experiments are repeated 5 times with different random seeds, with average results reported.

Benchmark Problems and Datasets

Multi-Product Newsvendor Problem with Synthetic Data We extend the classical MPBP (Donti, Amos, and Kolter 2017; Cristian et al. 2023; Turken et al. 2012) to a recursive form (R-MPNP) with two stakeholders (retailers and suppliers). Suppliers determine order costs $\boldsymbol{c} \in \mathbb{R}^n$ responding to both retailer order quantities $\boldsymbol{x} \in \mathbb{R}^n$ and contextual features $\boldsymbol{v} \in \mathbb{R}^d$. Retailers determine optimal order quantities $\boldsymbol{x}^*$ while facing adaptive cost parameters set by suppliers. See Appendix C.1 for the mathematical formulations. Synthetic datasets across three scales ($n = 10, 50, 100$) are generated, corresponding to Jacobian matrices $J_{\mathcal{G}}|_{\boldsymbol{c}_{i+1}}$ of size 32×32, 152×152, and 302×302 respectively in gradient computation.

Bipartite Matching Problem with Real-World Data Bipartite matching has broad applications in resource allocation, such as ride-hailing systems and housing assignments (Wilder, Dilkina, and Tambe 2019; Benabbou et al. 2018; Zhao et al. 2019). We examine the recursive BMP (R-BMP) in ride-hailing, which involves two parties: a centralized platform and individual participants (drivers and riders). The platform generates an allocation decision $\boldsymbol{x} \in \mathbb{R}^n$ to optimize driver-rider matching while estimating potential participant regret $\boldsymbol{c} \in \mathbb{R}^n$. Upon receiving their assignments, both drivers and riders experience realized regret based on the specific matches $\boldsymbol{x}$, creating a feedback loop. See Appendix C.2 for mathematical formulations. We evaluate R-DFL using real-world matching data from the NYC TLC trip dataset, under problem scales of $(4, 15, 30)$ matches, corresponding to sizes of optimization problems $n = 16, 225, 900$ and Jacobian matrices 57×57, 706×706, and 2761×2761, respectively.

Baselines

We evaluate our R-DFL framework with two differentiation methods against the state-of-the-art S-DFL framework and PTO framework. All frameworks use identical predictive architectures and equivalent convex programming formulations. The baselines are:

- **R-DFL-U**: The recursive DFL with the explicit unroll methods through finite unrolling steps.
- **R-DFL-I**: The recursive DFL with the implicit differentiation methods backpropagating at the equilibrium point.
- **S-DFL**: The sequential DFL where the predictive model uses only exogenous features, without decision feedback.
- **PTO**: The conventional two-step approach with separate prediction and optimization, where the predictive model uses only exogenous features, without decision feedback.

Evaluation Metrics

We evaluate the baselines across two dimensions: (1) Decision accuracy, (2) Computational efficiency. While all baselines are evaluated on accuracy, the efficiency metrics focus specifically on the two differentiation methods of R-DFL.

- **Accuracy**: RMSE loss for the final decision $\boldsymbol{x}$.
- **Efficiency**: Average training time per epoch (in seconds).

Results

The results in Table 1 reveal three findings: (1) The proposed R-DFL framework with two differentiation methods consistently outperforms the S-DFL and PTO baselines across all datasets in accuracy, demonstrating that modeling of the recursive structure will significantly enhance the final decision quality in recursive decision-making problems with bidirectional feedback. (2) While maintaining comparable accuracy, the implicit differentiation methods achieve significantly less training time per epoch, 1.5 times faster than explicit unrolling methods in the large-scale problem of bipartite matching, highlighting their superior computational efficiency. (3) The predictive model in the newsvendor problem involves more predictive parameters than the bipartite matching problem, leading to longer training time despite smaller Jacobian matrices. Full table see Appendix D.3.

Accuracy Comparison of the R-DFL

Figure 3 compares the decision result distributions of the R-DFL framework using two differentiation methods across training and test datasets in the newsvendor problem. The QQ plots reveal strong alignment between the two methods, particularly for small-scale datasets, indicating consistent decision outcomes. While minor deviations emerge in the largest dataset, the overall agreement suggests the robustness of both differentiation methods across varying problem scales, and either differentiation method can reliably support the R-DFL framework in practical applications.

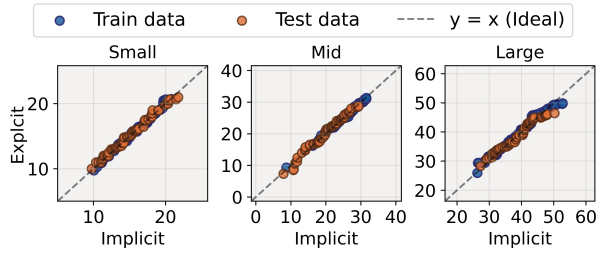

Figure 3: Accuracy comparison between R-DFL-U and R-DFL-I on newsvendor dataset across three scales.

Sensitivity Analysis on Unrolling Layers

Figure 4 presents a sensitivity analysis examining the impact of varying numbers of unrolling layers $\{5, 10, 15, 20, 25\}$ of both differentiation methods on the two datasets. Results demonstrate that: (1) Regarding accuracy, both the explicit and implicit methods achieve comparable RMSE values. (2) Regarding computational efficiency, the implicit methods exhibit superior performance compared to explicit unrolling with consistently less training time, particularly with the increment of unrolling layers. (3) Specifically, increasing the number of unrolling layers for explicit unrolling methods yields marginal accuracy improvements and results in substantial computational overhead, significantly prolonging training time.

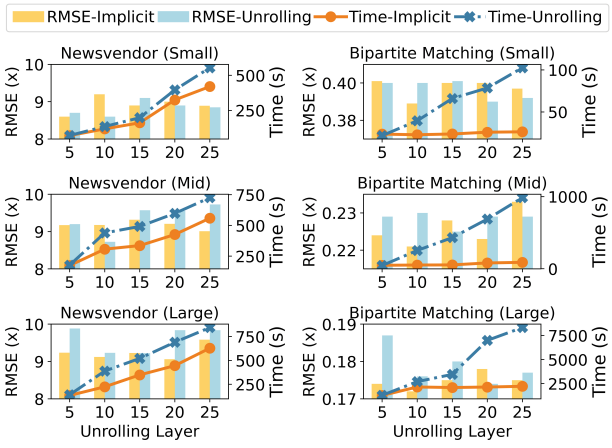

Figure 4: Sensitivity analysis of unrolling layers.

Robustness Check of the R-DFL

Table 2 reports results of three widely-used predictive models in two datasets, including `LSTM` (Hochreiter and Schmidhuber 1997), `RNN` (Elman 1990), and `Transformer` (Vaswani et al. 2017) with 10 unrolling layers. The R-DFL framework with both differentiation methods shows higher accuracy than S-DFL and PTO frameworks in both experiments, representing the effectiveness and robustness of the proposed R-DFL framework. Full table see Appendix D.3.

Newsvendor Problem (Large)						
Model	LSTM		RNN		Transformer	
Metrics	RMSE	Time	RMSE	Time	RMSE	Time
PTO	12.034	-	12.842	-	14.071	-
S-DFL	12.693	-	12.583	-	14.040	-
R-DFL-U	10.112	531	10.872	510	**11.231**	561
R-DFL-I	**10.104**	**355**	**10.810**	**340**	11.332	**360**
Bipartite Matching Problem (Large)						
Model	LSTM		RNN		Transformer	
Metrics	RMSE	Time	RMSE	Time	RMSE	Time
PTO	0.219	-	0.185	-	0.193	-
S-DFL	0.230	-	0.187	-	0.188	-
R-DFL-U	0.176	2704	0.176	2821	0.172	2821
R-DFL-I	**0.174**	**2093**	**0.174**	**2079**	**0.166**	**2078**

Table 2: Comparison with different predictive models.

Conclusions

While S-DFL has shown promise in improving decision quality by integrating optimization models in deep learning, its sequential structure fails to capture the critical interactions between prediction and optimization in closed-loop decision-making problems, hence limiting its applications. This paper introduces R-DFL, a novel framework that establishes bidirectional interaction between optimization and prediction, together with two mathematically-derived differentiation methods: explicit unrolling methods via auto-differentiation and implicit differentiation methods based on fixed-point analysis. Theoretical and numerical analyses demonstrate that considering the recursive structure substantially enhances the decision quality in problems with bidirectional feedback, regardless of prediction models. Furthermore, this paper reveals a trade-off between the two differentiation methods: while explicit unrolling methods offer a more straightforward implementation, implicit differentiation methods achieve superior computational efficiency. Beyond these specific applications, R-DFL represents a significant advancement in decision-focused learning by creating a truly unified prediction-prescription pipeline capable of modeling complex bidirectional systems, with promising extensions to broader classes of closed-loop decision-making problems under uncertainty. Future directions include extending R-DFL to handle stochastic recursive environments and developing a more versatile framework capable of addressing integer problems.

Acknowledgments

The work described in this paper is supported by grants from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. PolyU/15206322 and PolyU/15227424), the Otto Poon Charitable Foundation Smart Cities Research Institute (SCRI) (U-CDC4), and the Research Centre for Digital Transformation of Tourism (RCDTT) (1-BBGU) at the Hong Kong Polytechnic University. The contents of this article reflect the views of the authors, who are responsible for the facts and accuracy of the information presented herein.

References

Agrawal, A.; Amos, B.; Barratt, S.; Boyd, S.; Diamond, S.; and Kolter, J. Z. 2019. Differentiable convex optimization layers. *Advances in Neural Information Processing Systems*, 32.

Amos, B.; and Kolter, J. Z. 2017. Optnet: Differentiable optimization as a layer in neural networks. In *International conference on machine learning*, 136–145. PMLR.

Bai, S.; Kolter, J. Z.; and Koltun, V. 2019. Deep equilibrium models. *Advances in neural information processing systems*, 32.

Benabbou, N.; Chakraborty, M.; Ho, X.-V.; Sliwinski, J.; and Zick, Y. 2018. Diversity constraints in public housing allocation. In *17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2018)*.

Berthet, Q.; Blondel, M.; Teboul, O.; Cuturi, M.; Vert, J.-P.; and Bach, F. 2020. Learning with differentiable pertubed optimizers. *Advances in Neural Information Processing Systems*, 33: 9508–9519.

Bertsimas, D.; and Kallus, N. 2020. From predictive to prescriptive analytics. *Management Science*, 66(3): 1025–1044.

Chen, R. T.; Rubanova, Y.; Bettencourt, J.; and Duvenaud, D. K. 2018. Neural ordinary differential equations. *Advances in neural information processing systems*, 31.

Cristian, R.; Harsha, P.; Perakis, G.; Quanz, B. L.; and Spantidakis, I. 2023. End-to-end learning for optimization via constraint-enforcing approximators. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, 7253–7260.

Diamond, S.; and Boyd, S. 2016. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83): 1–5.

Dimitrov, V.; and Coelho, D. 2017. On the computation of Neumann series. *arXiv preprint arXiv:1707.05846*.

Donti, P.; Amos, B.; and Kolter, J. Z. 2017. Task-based end-to-end model learning in stochastic optimization. *Advances in neural information processing systems*, 30.

El Ghaoui, L.; Gu, F.; Travacca, B.; Askari, A.; and Tsai, A. 2021. Implicit deep learning. *SIAM Journal on Mathematics of Data Science*, 3(3): 930–958.

Elmachtoub, A. N.; and Grigas, P. 2022. Smart "predict, then optimize". *Management Science*, 68(1): 9–26.

Elman, J. L. 1990. Finding structure in time. *Cognitive science*, 14(2): 179–211.

Ferber, A.; Wilder, B.; Dilkina, B.; and Tambe, M. 2020. Mipaal: Mixed integer program as a layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 1504–1511.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hochreiter, S.; and Schmidhuber, J. 1997. Long short-term memory. *Neural computation*, 9(8): 1735–1780.

Jia, L.; Tong, L.; and Zhao, Q. 2014. An online learning approach to dynamic pricing for demand response. *arXiv preprint arXiv:1404.1325*.

Kazi, M.; and Thompson, B. 2017. Implicitly-defined neural networks for sequence labeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 172–177.

Kotary, J.; Dinh, M. H.; and Fioretto, F. 2023. Backpropagation of Unrolled Solvers with Folded Optimization. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

Kotary, J.; Fioretto, F.; Van Hentenryck, P.; and Wilder, B. 2021. End-to-end constrained optimization learning: A survey. *arXiv preprint arXiv:2103.16378*.

Kotary, J.; Fioretto, F.; Van Hentenryck, P.; and Zhu, Z. 2022. End-to-end learning for fair ranking systems. In *Proceedings of the ACM Web Conference 2022*, 3520–3530.

Krantz, S. G.; and Parks, H. R. 2002. *The implicit function theorem: history, theory, and applications*. Springer Science & Business Media.

Levina, T.; Levin, Y.; McGill, J.; and Nediak, M. 2009. Dynamic pricing with online learning and strategic consumers: An application of the aggregating algorithm. *Operations research*, 57(2): 327–341.

Li, J.; Yu, J.; Nie, Y.; and Wang, Z. 2020. End-to-end learning and intervention in games. *Advances in Neural Information Processing Systems*, 33: 16653–16665.

Liao, R.; Xiong, Y.; Fetaya, E.; Zhang, L.; Yoon, K.; Pitkow, X.; Urtasun, R.; and Zemel, R. 2018. Reviving and improving recurrent back-propagation. In *International conference on machine learning*, 3082–3091. PMLR.

Mandi, J.; Bucarey, V.; Tchomba, M. M. K.; and Guns, T. 2022. Decision-focused learning: Through the lens of learning to rank. In *International conference on machine learning*, 14935–14947. PMLR.

Mandi, J.; and Guns, T. 2020. Interior point solving for lp-based prediction+ optimisation. *Advances in Neural Information Processing Systems*, 33: 7272–7282.

Mandi, J.; Kotary, J.; Berden, S.; Mulamba, M.; Bucarey, V.; Guns, T.; and Fioretto, F. 2024. Decision-focused learning: Foundations, state of the art, benchmark and future opportunities. *Journal of Artificial Intelligence Research*, 80: 1623–1701.

Mandi, J.; Stuckey, P. J.; Guns, T.; et al. 2020. Smart predict-and-optimize for hard combinatorial optimization problems.

In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 1603–1610.

Monga, V.; Li, Y.; and Eldar, Y. C. 2021. Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing. *IEEE Signal Processing Magazine*, 38(2): 18–44.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in pytorch.

Postek, K.; and Hertog, D. d. 2016. Multistage adjustable robust mixed-integer optimization via iterative splitting of the uncertainty set. *INFORMS Journal on Computing*, 28(3): 553–574.

Qi, M.; Shi, Y.; Qi, Y.; Ma, C.; Yuan, R.; Wu, D.; and Shen, Z.-J. 2023. A practical end-to-end inventory management model with deep learning. *Management Science*, 69(2): 759–773.

Qin, Z.; Tang, X.; Jiao, Y.; Zhang, F.; Xu, Z.; Zhu, H.; and Ye, J. 2020. Ride-hailing order dispatching at didi via reinforcement learning. *INFORMS Journal on Applied Analytics*, 50(5): 272–286.

Sadana, U.; Chenreddy, A.; Delage, E.; Forel, A.; Frejinger, E.; and Vidal, T. 2025. A survey of contextual optimization methods for decision-making under uncertainty. *European Journal of Operational Research*, 320(2): 271–289.

Scellier, B.; and Bengio, Y. 2017. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in computational neuroscience*, 11: 24.

Tang, B.; and Khalil, E. B. 2022. Pyepo: A pytorch-based end-to-end predict-then-optimize library with linear objective function. In *OPT 2022: Optimization for Machine Learning (NeurIPS 2022 Workshop)*.

Turken, N.; Tan, Y.; Vakharia, A. J.; Wang, L.; Wang, R.; and Yenipazarli, A. 2012. The multi-product newsvendor problem: Review, extensions, and directions for future research. *Handbook of newsvendor problems: Models, extensions and applications*, 3–39.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Wang, P.-W.; Donti, P.; Wilder, B.; and Kolter, Z. 2019. Satnet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In *International Conference on Machine Learning*, 6545–6554. PMLR.

Wang, Y.; Wang, J.; Zhang, H.; and Song, J. 2025. Bridging Prediction and Decision: Advances and Challenges in Data-Driven Optimization. *Nexus*.

Wilder, B.; Dilkina, B.; and Tambe, M. 2019. Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 1658–1665.

Xiao, Q.; Lu, S.; and Chen, T. 2023. An alternating optimization method for bilevel problems under the Polyak-Łojasiewicz condition. *Advances in Neural Information Processing Systems*, 36: 63847–63873.

Zhang, Z.; Kag, A.; Sullivan, A.; and Saligrama, V. 2019. Equilibrated recurrent neural network: Neuronal time-delayed self-feedback improves accuracy and stability. *arXiv preprint arXiv:1903.00755*.

Zhao, B.; Xu, P.; Shi, Y.; Tong, Y.; Zhou, Z.; and Zeng, Y. 2019. Preference-aware task assignment in on-demand taxi dispatching: An online stable matching approach. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 2245–2252.

Zhao, Y.; Zheng, K.; Yin, H.; Liu, G.; Fang, J.; and Zhou, X. 2020. Preference-aware task assignment in spatial crowdsourcing: From individuals to groups. *IEEE Transactions on Knowledge and Data Engineering*, 34(7): 3461–3477.