Decision-Focused Learning with Machine Learning Proxies for Energy Storage Systems

Ruben Smets, Mathieu Tanneau, Jean-François Toubeau, Kenneth Bruninx, Erik Delarue, Pascal Van Hentenryck

Working Paper Series No. ESIM2024-05

Last update: September 27, 2024

Decision-Focused Learning with Machine Learning Proxies for Energy Storage Systems

Ruben Smets, Mathieu Tanneau, Jean-François Toubeau, *Member, IEEE,*, Kenneth Bruninx, *Member, IEEE,* Erik Delarue, *Member, IEEE*, Pascal Van Hentenryck

***Abstract*—As electricity prices become increasingly volatile due to the growing penetration of renewable energy sources, Energy Storage Systems (ESS) are encountering greater opportunities for inter-temporal arbitrage. These can be harnessed through participation in electricity markets using a predict-then-optimize framework: a forecasting model first predicts prices, which are subsequently used in a profit maximization problem. Decision-Focused Learning (DFL) techniques, allowing to train forecasters to directly maximize profits of optimized decisions rather than minimizing statistical price forecasting errors, have been shown to increase profits. However, these techniques are computationally inefficient because they require solving the optimization problem across the entire training set for each training epoch. To address this issue, we introduce a machine learning proxy method mimicking the output of the optimization problem during training. Our approach leverages duality theory to retrieve decisions in two steps: (i) predicting a key dual variable with a pre-trained neural network, and (ii) applying smoothing functions to this prediction to determine the ESS (dis)charge decisions. In a case study involving an ESS participating in the day-ahead market with real-life data, we demonstrate that our proxy method significantly reduces training time compared to benchmark DFL methods, while achieving comparable or superior out-of-sample profits.**

***Index Terms*—Decision-focused learning, Energy storage systems, Price forecasting, Duality theory, Machine learning proxy**

NOMENCLATURE

Decision Variables

μ_t	Dual variable associated with the state of charge update constraint at time step t (€/MWh)
θ	Neural network trainable parameters
c_t	Energy charged at time step t (MWh)
d_t	Energy discharged at time step t (MWh)
e_t	Net energy discharged to the grid at time step t (MWh)
s_t	State of charge at time step t (MWh)

Parameters

η	Charge and discharge efficiency
$\hat{\lambda}_t$	Price forecast at time step t (€/MWh)
$\overline{S}$	Maximum state of charge (MWh)
$\underline{S}$	Minimum state of charge (MWh)
$\overline{E}$	Maximum (dis)charge energy per time step (MWh)

I. INTRODUCTION

The electricity system is undergoing major changes due to increasing shares of intermittent renewable energy generation, as well as the electrification of heating, transportation and industry. These developments are leading to increased volatility in electricity spot market prices [1], posing risks for some market participants while presenting substantial opportunities for others, such as Energy Storage System (ESS) operators. Indeed, since they aim to capitalize on inter-temporal arbitrage opportunities, volatile prices can improve their profitability. In this paper, we propose a new method in the realm of Decision-Focused Learning (DFL) designed to enhance price forecasting for ESS participation in short-term electricity markets.

Battery ESSs, with their short reaction times and high efficiency [2] are well-placed to participate in very short-term markets like Real-Time (RT) and balancing markets. Pumped Hydro Storage (PHS) and Compressed Air Energy Storage (CAES) assets, on the other hand, are slower with lower efficiencies. Participation in the Day-Ahead Market (DAM) has been investigated for BESS [3, 4], PHS [5] and CAES [6] alike, often in a multi-market setting combining DAM participation with reserve [7, 8] or RT [9] markets. Modeling ESS market participation typically involves a predict-then-optimize strategy. Here, ESS operators first predict, e.g., electricity prices, which are subsequently used as input for an Optimization Program (OP) that determines profit-maximizing decisions [3, 10].

Section I-A reviews the current state of electricity price forecasting based on Machine Learning (ML) techniques in the context of predict-then-optimize frameworks. Section I-B covers techniques where machine learning algorithms replace optimization altogether. Finally, Section I-C outlines the key scientific contributions of this manuscript.

A. Price forecasting

In recent years, Machine Learning (ML) techniques like Neural Networks (NN) have emerged as the preferred technique for electricity price forecasting over statistical methods [11]. Specifically, Recurrent Neural Networks [12, 13] have been shown to be well-suited forecasting models because of their ability to accurately capture temporal dependencies. One of the key modeling choices is the loss function that is used for training the forecaster. Traditional choices include the Mean Squared Error (MSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE) for deterministic forecasts [14] or the quantile loss and continuous rank probability score for probabistic forecasts [15]. However, those traditional methods often overlook the insight that in most cases, the forecaster is deployed in a downstream decision problem. DFL uses a loss function that reflects the forecast's value in such decision problems.

A first stream of research in DFL explores traditional statistical loss functions adapted to specific decision problems. This has been implemented for the problem of energy trading in the DAM based on wind forecasts, where [16] proposes a loss function that assigns more importance to large wind power output forecast errors. Other loss-based approaches first formulate a general parametric family of loss functions and use historical data to find the specific function that results in the best downstream performance. Examples include load forecasting to minimize dispatch costs [17] and imbalance price forecasting to maximize profits of an ESS owner [10].

A second stream of DFL research considers an integrated, decision-focused loss function, typically regret [18]. This function quantifies the cost of the forecast error by comparing the downstream performance of decisions based on the predicted values to those with perfect foresight. While specific methods have been developed for such integrated DFL applied to decision tree forecasters [19, 20], we here focus on NN-based forecasters. The primary challenge in training these forecasters to minimize regret is that the prevalent NN training method, gradient descent, depends on the derivative of the optimal decision relative to the forecast inputs, an intricate problem that has seen various solutions. For instance, Implicit Differentiation (ID) of the Karush-Kuhn Tucker (KKT) conditions has proven possible for quadratic [21], conic [22] and ultimately general convex downstream decision problems [23]. However, this method yields unusable gradients for downstream linear programs since they are inherently zero almost everywhere. To overcome this issue, quadratic [24, 25] and log-barrier [26] smoothing functions have been added to the linear objective function. Other approaches involve perturbations of the input parameters [27, 28] or a convex, but non-linear surrogate of the regret loss function [29]. Only the surrogate loss function [29] was used to train a price forecaster for ESS participation in the DAM [30], outperforming an MSE-trained model in terms of obtained profit. However, a major drawback of these integrated approaches is the need to solve the optimization program with every forward pass, which greatly reduces computational efficiency during training.

B. Machine learning-only approaches

As an alternative, Reinforcement Learning (RL) has been put forth to govern ESS decision-making in electricity markets [31, 32, 33]. Here, historical data is used to directly train the best possible ESS policy based on contextual information. While it is similar in spirit to DFL, taking into account ESS decisions and their profitability in the training procedure, it is distinctly different from DFL as it discards the two-step predict-then-optimize philosophy. In that regard, DFL holds the advantage of interpretability and the ease of including technical constraints in the ESS profit maximization problem.

Another recent advancement in ML pertains to proxies for the output of optimization programs. Some optimization programs in power systems, notably large-scale optimal power flow problems, have to be solved in near-RT to ensure stable and efficient grid operation. However, physical laws render this problem highly non-convex, and hence slow to solve. ML proxies can strongly decrease the required time to obtain (near-)optimal solutions. To ensure feasibility of the output variables, differentiable constraint completion and correction mechanisms [34] have been proposed. This technique has been proven successful for large-scale dispatch problems [35].

C. Contributions

In summary, the technique of implicit differentiation of the KKT conditions has not been applied to the problem of profit maximization (of ESSs) in electricity markets. At the same time, there is no clear-cut answer on how to train any integrated decision-focused model efficiently. To address this research gap, we propose a DFL training technique that leverages an ML proxy to replace the downstream optimization program and reduce the computational burden. The scientific contributions of this paper are as follows:

- We apply implicit differentiation of the KKT conditions to the problem of price forecasting for ESS decision-making, improving the generic technique through limited smoothing based on knowledge of the downstream problem.
- We use an ML proxy replacing the optimization oracle in the forward pass of the price forecaster training procedure, considerably improving computational efficiency.
- We present a methodology that utilizes duality theory, reducing the ML proxy's function to predicting a single dual variable, further refined by a smoothing function.
- Within the ML proxy methodology, we define novel smoothing functions that outperform those naturally arising from existing smoothing mechanisms.

The remainder of the paper is structured as follows. Section II elaborates on the methodology of ID of the KKT conditions applied to the problem of ESS profit maximization. Section III details how our proposed method of primal and dual ML proxies can be leveraged to speed up the training process. Then, in Section IV, we show the efficacy of our method by deploying it in a real-life case study of the Belgian DAM. Concluding remarks are provided in Section V.

II. DFL WITH IMPLICIT DIFFERENTIATION

We propose a DFL framework (Figure 1) based on a loss function, e.g., regret, taking into account the decisions following from the forecast, calculated through a differentiable procedure. This leads the way to applying standard gradient descent-based approaches to train the forecaster. However, the requirement of differentiability, especially in cases like (mixed-integer) linear downstream problems, necessitates implementing a modified decision problem during the training phase. During the test phase, the forecasts are applied directly to the original decision problem.

This paper explores two methods of implementing the differentiable decision calculator. While Section III considers an ML proxy to speed up the training procedure, this section explores the use of implicit differentiation of the KKT conditions. Section II-A details the linear optimization program that governs the ESS market participation decisions used in this paper. Section II-B explains the basic principles of implicit differentiation of the KKT conditions. In Section II-C, the need to introduce smoothing terms to ensure non-zero gradients of the optimization program is discussed, and a

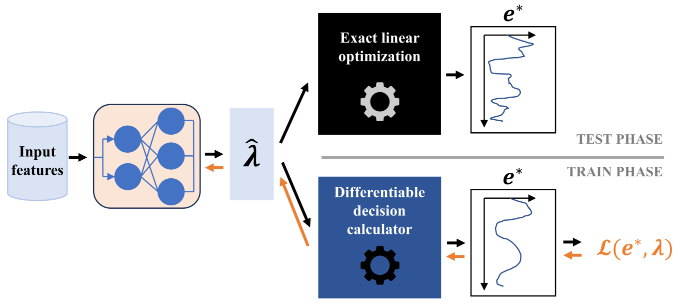
Input features
$\hat{\boldsymbol{\lambda}}$
Exact linear optimization
e^*
TEST PHASE
TRAIN PHASE
Differentiable decision calculator
e^*
$\mathcal{L}(e^*, \lambda)$

Fig. 1: Overview of the method. During training, the forward pass (i) converts input to price forecasts $\hat{\boldsymbol{\lambda}}$ through the NN to be trained and (ii) converts price forecasts to decisions using a differentiable decision calculator. In the backward pass, the NN parameters are updated based on a decision-focused loss $\mathcal{L}$. At test time, the forecasts are directly fed into the exact linear optimization to retrieve the optimal schedules.

limited smoothing approach tailored to the profit maximization problem is introduced.

A. *ESS optimization program*

Following [30, 36], we consider a technology-agnostic Optimization Program (OP) used for ESS market participation, assuming the ESS to be a price-taker:

$$\underset{\boldsymbol{c},\boldsymbol{d},\boldsymbol{s}}{\text{maximize}} \sum_{t=1}^{T} \hat{\lambda}_t \left(\eta d_t - \frac{c_t}{\eta} \right) \tag{1a}$$

subject to:

$$c_t, d_t \geq 0 : \underline{\gamma}_t, \underline{\delta}_t \qquad \forall t \tag{1b}$$
$$c_t, d_t \leq E : \overline{\gamma}_t, \overline{\delta}_t \qquad \forall t \tag{1c}$$
$$\underline{S} \leq s_t \leq \overline{S} : \underline{\sigma}_t, \overline{\sigma}_t \qquad \forall t \tag{1d}$$
$$s_1 = S_0 + c_1 - d_1 : \mu_1 \tag{1e}$$
$$s_t = s_{t-1} + c_t - d_t : \mu_t \qquad \forall t > 1 \tag{1f}$$
$$s_T = S_0 : \mu_{T+1}, \tag{1g}$$

where the decision variables $\boldsymbol{c}$ and $\boldsymbol{d}$ represent the vectors of charge and discharge actions respectively, and $\boldsymbol{s}$ the State of Charge (SoC). Note that bold symbols correspond to vectors. In the objective, the (dis)charge actions are adjusted for efficiency losses η, and multiplied with the price forecast $\hat{\boldsymbol{\lambda}}$ to obtain the expected profit over the look-ahead horizon T. The SoC and (dis)charge actions are subject to box constraints (1b)-(1d). Finally, the SoC update rule is given by Eq. (1e)-(1f), where S_0 is the exogenous parameter representing the SoC at the start of optimization. The cyclic boundary condition (1g), while not inherent to all ESS optimization, is included because of its widespread use, especially in DAM optimization.

B. *Implicit differentiation of KKT conditions*

This work focuses on finding the optimal price forecast $\hat{\boldsymbol{\lambda}}$ in problem (1). The forecasting model is assumed to be a neural network, which is characterized by trainable parameters (weights and biases), collectively referred to as θ:

$$\hat{\boldsymbol{\lambda}} = N^{\lambda}(\boldsymbol{x}; \theta), \tag{2}$$

where $\boldsymbol{x}$ represents the input features. Training the NN corresponds to minimizing a loss function $\mathcal{L}$ over a set of train data by adjusting the NN's trainable parameters:

$$\underset{\theta}{\text{minimize}} \sum_{i \in X_t} \mathcal{L}(\boldsymbol{\lambda}_i, N^{\lambda}(\boldsymbol{x}_i; \theta)), \tag{3}$$

where i denotes a specific example in train set X_t, and $\boldsymbol{\lambda}_i$ corresponds to the ground truth of that example, i.e., the actual historical price.

In DFL, one uses a decision-focused loss, often regret, which quantifies the value of the decisions made based on the forecast in a predict-then-optimize setting. This function corresponds to the OP's objective function, evaluated ex-post for the decisions optimized based on forecasts, compared to those optimized based on the ground truth. In the case of a profit-maximizing ESS, this regret loss function reads:

$$\mathcal{L}^r(\hat{\boldsymbol{\lambda}}, \boldsymbol{\lambda}) = \sum_{t=1}^{T} \lambda_t e_t^*(\boldsymbol{\lambda}) - \lambda_t e_t^*(\hat{\boldsymbol{\lambda}}), \tag{4}$$

where the optimized net discharge, $\boldsymbol{e}^* = \eta \boldsymbol{d}^* - \boldsymbol{c}^*/\eta$, is a function of the price forecast through Problem (1).

Training a NN to minimize a decision-focused loss function requires calculating the gradients of that loss with respect to the trainable parameters of the forecaster. By applying the chain rule, these gradients can be broken down to:

$$\frac{\partial \mathcal{L}^r}{\partial \theta} = \frac{\partial \mathcal{L}^r}{\partial \boldsymbol{e}^*} \frac{\partial \boldsymbol{e}^*}{\partial \hat{\boldsymbol{\lambda}}} \frac{\partial \hat{\boldsymbol{\lambda}}}{\partial \theta}. \tag{5}$$

Here, the first term can straightforwardly be calculated from Eq. (4), and automatic differentiation [37] is used to calculate the gradients within the price forecaster $\frac{\partial \hat{\boldsymbol{\lambda}}}{\partial \theta}$. The challenge in DFL lies in computing the rate of change of the optimal decisions with respect to the forecast, $\frac{\partial \boldsymbol{e}^*}{\partial \hat{\boldsymbol{\lambda}}}$. For most convex problems, the KKT conditions form a system of equations that describe all optimal solutions of the primal problem, see, e.g., [38]. Differentiating these constraints with respect to the predicted variable gives rise to a set of equations, including the term $\frac{\partial \boldsymbol{e}^*}{\partial \hat{\boldsymbol{\lambda}}}$, which can be solved for in the general setting of convex optimization programs. This technique has been made widely available through the *Cvxpylayers* library in *Python* [23]. Training a NN with such implicit differentiation of the KKT conditions boils down to iteratively performing the following procedure:

1) Calculate the price forecast $\hat{\boldsymbol{\lambda}}$ for all the examples in the train set using Eq. (2).
2) Retrieve the optimal decisions by solving Problem (1).
3) Calculate the regret loss with Eq. (4) and use ID of the KKT conditions to calculate the gradients with Eq. (5).
4) Update the NN parameters θ with a gradient descent step.

C. Smoothing of linear programs

When the downstream problem is linear, there is no local sensitivity of the optimal decision with respect to the OP's parameters, resulting in $\frac{\partial \boldsymbol{e}^*}{\partial \hat{\boldsymbol{\lambda}}} = 0$ nearly everywhere [24, 26]. Through Eq. (5), this renders the method of gradient descent unusable. To overcome this problem, a smoothing approach has been proposed that adds a convex non-linear term to the objective function. In this case, the ESS optimization program becomes:

$$\underset{\boldsymbol{c},\boldsymbol{d},\boldsymbol{s}}{\text{maximize}} \sum_{t=1}^{T} \hat{\lambda}_t \left(\eta d_t - \frac{c_t}{\eta} \right) - \gamma S(\boldsymbol{c},\boldsymbol{d},\boldsymbol{s}) \tag{6a}$$

subject to:

$$\text{(1b) - (1g),} \tag{6b}$$

with $S(\boldsymbol{c},\boldsymbol{d},\boldsymbol{s})$ a convex non-linear function and γ a parameter that determines the degree of smoothing. The value of γ is a trade-off between introducing sufficient gradient information (large γ) and remaining close to the original problem (small γ). In Section III-B, we will demonstrate this indeed induces a sensitivity of the optimal outcome with respect to the forecasted price. Two generic smoothing functions have been proposed in the academic literature. In the first, the square of all decision variables is added to the objective [24]. In the second, log-barrier terms are added for all inequality constraints [26]. In Problem (1), this becomes:

$$S(\boldsymbol{c},\boldsymbol{d},\boldsymbol{s}) = \frac{1}{2} \sum_{t} c_t^2 + d_t^2 + s_t^2 \tag{7a}$$

$$\begin{aligned} S(\boldsymbol{c},\boldsymbol{d},\boldsymbol{s}) = -\sum_{t} &\ln(c_t) + \ln(E - c_t) + \ln(d_t) \\ &+ \ln(E - d_t) + \ln(s_t - \underline{S}) + \ln(\overline{S} - s_t). \end{aligned} \tag{7b}$$

In this paper, we propose an adjustment in the form of limited smoothing functions:

$$S(\boldsymbol{c},\boldsymbol{d},\boldsymbol{s}) = \frac{1}{2} \sum_{t} c_t^2 + d_t^2 \tag{8a}$$

$$\begin{aligned} S(\boldsymbol{c},\boldsymbol{d},\boldsymbol{s}) = -\sum_{t} &\ln(c_t) + \ln(E - c_t) \\ &+ \ln(d_t) + \ln(E - d_t), \end{aligned} \tag{8b}$$

with the prime difference that the SoC variable is not included in the smoothing. The main goal of smoothing the OP is to induce a sensitivity of the optimal decisions with respect to the forecast, but remain close to the original linear OP. While previously proposed smoothings in Eqs. (7) inhibit the optimal solution to have SoC close to $\underline{S}$ and/or $\overline{S}$, this is not the case for our smoothing in Eqs. (8). As such, the limitedly smoothed OP better approximates the original linear OP, while still ensuring sensitivity of $\boldsymbol{c}$ and $\boldsymbol{d}$ with respect to the forecasted price, thus exhibiting non-zero gradients in the DFL training method.

III. Machine Learning Proxies to Speed Up Decision-Focused Learning

A pivotal shortcoming of using loss function (4) is its dependence on the optimized decision values $\boldsymbol{e}^*(\hat{\boldsymbol{\lambda}})$, and hence that an optimization oracle is called for the entire training set for each epoch (Step 2) in Section II, resulting in long train times. This section presents a novel method that addresses this problem by replacing the optimization oracle with an ML proxy. Indeed, the pre-trained NN performs a series of matrix multiplications and non-linear activation functions, which are computationally more efficient than the iterative procedure of exactly solving the optimization problem. We present two distinct approaches, through a primal and dual proxy, see Figure 2. Section III-A outlines the proposed proxy method and the primal proxy procedure. Section III-B presents a dual analysis of the ESS decision problem and introduces the dual proxy method.

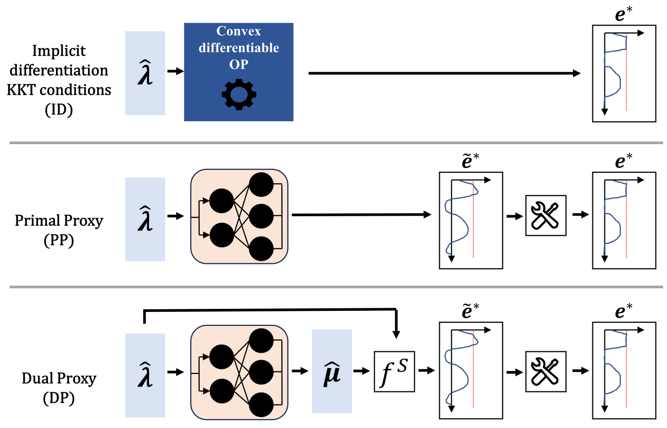
Implicit differentiation KKT conditions (ID)
$\hat{\boldsymbol{\lambda}}$
Convex differentiable OP
$\boldsymbol{e}^*$
Primal Proxy (PP)
$\hat{\boldsymbol{\lambda}}$
$\tilde{\boldsymbol{e}}^*$
$\boldsymbol{e}^*$
Dual Proxy (DP)
$\hat{\boldsymbol{\lambda}}$
$\hat{\boldsymbol{\mu}}$
f^S
$\tilde{\boldsymbol{e}}^*$
$\boldsymbol{e}^*$

Fig. 2: Comparing different methods for implementing a differentiable schedule calculator in DFL training. In the ID method, schedules are computed from prices via convex optimization. The primal proxy method uses a pre-trained proxy NN to generate schedules, followed by a feasibility repair mechanism. In the dual proxy method, the NN outputs a dual variable μ, which is converted to schedules via a smoothing function f^S and then repaired to feasibility.

A. Primal proxy

The proxy method entails a differentiable NN-based proxy function P that yields the optimal decisions based on price forecasts, i.e., $\boldsymbol{e}^* = P(\hat{\boldsymbol{\lambda}})$. In our proposed approach, P is broken down into three distinct steps:

1) Predict the primal variables with a pre-trained NN based on price forecasts: $\tilde{\boldsymbol{c}} = N^c(\hat{\boldsymbol{\lambda}};\theta)$, $\tilde{\boldsymbol{d}} = N^d(\hat{\boldsymbol{\lambda}};\theta)$.
2) Repair the resulting decisions to ensure feasibility: $\boldsymbol{c}^*, \boldsymbol{d}^*, \boldsymbol{s}^* = R(\tilde{\boldsymbol{c}}, \tilde{\boldsymbol{d}})$.
3) Retrieve the net discharge: $\boldsymbol{e}^* = \eta \boldsymbol{d}^* - \boldsymbol{c}^*/\eta$.

Below, we discuss the procedures in the first two steps.

1) Proxy neural network: This NN is pre-trained before the training phase of the price forecaster. As such, the proxy method can only be regarded as more computationally efficient when the speed-up in the DFL training phase of the price forecaster compensates for the required computational effort to train the proxy. Pre-training the proxy requires (i) sampling price data $\hat{\boldsymbol{\lambda}}$ and (ii) for every sample, solving Problem (1) to retrieve the optimized values of $\boldsymbol{c}$ and $\boldsymbol{d}$ serving as targets in the proxy training procedure. When deploying this NN proxy in the price forecaster learning problem, it serves as a fixed mapping from price forecast to decision variable, and its parameters are not updated.

In NN implementations, the final layer typically lacks a non-linear activation function. However, for this application, the box constraints on $\boldsymbol{c}$ and $\boldsymbol{d}$ suggest that it could be advantageous to apply a logistic function to the final layer:

$$\tilde{\boldsymbol{c}} = E \cdot \sigma\left(N^c(\hat{\boldsymbol{\lambda}};\theta);\gamma\right) \tag{9}$$

$$\tilde{\boldsymbol{d}} = E \cdot \sigma\left(N^d(\hat{\boldsymbol{\lambda}};\theta);\gamma\right), \tag{10}$$

with $\sigma : \mathbb{R} \to \mathbb{R}, \quad x \mapsto \frac{1}{1+e^{-\gamma x}}$ the logistic function with steepness parameterized by γ and $N^{c/d}$ a NN without a non-linear activation function in the final layer. This effectively limits the domain of the NN output to the feasible region of $\boldsymbol{c}$ and $\boldsymbol{d}$.

2) Repair mechanism: Although Eqs. (9)-(10) ensure that the box constraints on the (dis)charge actions are met, this is not necessarily the case for the constraints on the SoC (Eqs. (1d))-((1g)). To adhere to the SoC constraints, we propose two distinct repair mechanisms. The first mechanism, Rc, involves rescaling the (dis)charge decisions to address imbalances between the total charge and discharge decisions, see Algorithm 1. This approach ensures that the resulting schedule adheres to the SoC update rule and cyclic boundary condition, but it does not necessarily ensure that the output schedule remains within the SoC box constraints (1d). The second mechanism, Rf, applies a two-fold iterative repair procedure to ensure meeting all the SoC constraints, see Algorithm 2. Both repair functions apply a greedy clipping method. For adhering to the SoC box constraints, the first function loops through the timesteps chronologically and clips the decisions leading to SoC box constraint violations. To meet the cyclic boundary condition, the second function processes the timesteps in reverse order, clipping the charge or discharge decisions until the constraint is satisfied.

While the case study confirms the necessity of a repair mechanism for achieving acceptable outcomes, we emphasize that these procedures are only two of many potential solutions for ensuring feasibility, and that it does not guarantee optimality of the repaired decisions.

Algorithm 1 Repair mechanism ensuring that the output schedule adheres to the cyclic boundary condition by rescaling the (dis)charge decisions.

```
Function Rc(c, d, s):
1: $\phi_d = \min(\frac{\sum_t c_t}{\sum_t d_t}, 1)$
2: $\phi_c = \min(\frac{\sum_t d_t}{\sum_t c_t}, 1)$
3: $s_0 = S_0$
4: for $t = 1 : T$ do
5:     $c_t \leftarrow \phi_c c_t$
6:     $d_t \leftarrow \phi_d d_t$
7:     $s_t = s_{t-1} + c_t - d_t$
8: end for
9: return $\boldsymbol{c}, \boldsymbol{d}, \boldsymbol{s}$
```

Algorithm 2 Repair mechanism ensuring feasibility of the proxy output schedule by greedily adjusting the (dis)charge decsisions to adhere to (i) the SoC box constraints and (ii) the cyclic boundary condition.

```
Function Rf(c, d):
1: $\tilde{\boldsymbol{c}}, \tilde{\boldsymbol{d}}, \tilde{\boldsymbol{s}}$ = RepairBox($\boldsymbol{c}, \boldsymbol{d}$)
2: $\boldsymbol{c}^f, \boldsymbol{d}^f, \boldsymbol{s}^f$ = RepairCBCGreedy($\tilde{\boldsymbol{c}}, \tilde{\boldsymbol{d}}, \tilde{\boldsymbol{s}}$)
3: return $\boldsymbol{c}^f, \boldsymbol{d}^f, \boldsymbol{s}^f$
Function RepairBox(c, d):
1: $s_0 = S_0$
2: for $t = 1 : T$ do
3:     $c_t \leftarrow \min(c_t, \overline{S} - s_{t-1})$
4:     $d_t \leftarrow \min(d_t, s_{t-1} - \underline{S})$
5:     $s_t = s_{t-1} + c_t - d_t$
6: end for
7: return $\boldsymbol{c}, \boldsymbol{d}, \boldsymbol{s}$
Function RepairCBC(c, d, s):
1: $t = T$
2: while $s_T < S_0$ or $s_T > S_0$ do
3:     $x = \max(0, \min(c_t, s_T - S_0))$
4:     $y = \max(0, \min(d_t, S_0 - s_T))$
5:     $c_t \leftarrow c_t - x$
6:     $d_t \leftarrow d_t - y$
7:     for all $\tau \in [t, T]$ do
8:         $s_\tau \leftarrow s_\tau - x + y$
9:     end for
10:    $t \leftarrow t - 1$
11: end while
12: return $\boldsymbol{c}, \boldsymbol{d}, \boldsymbol{s}$
```

B. Dual proxy

In this section, we first leverage duality theory to gain a deeper understanding in the optimal solutions to both the linear and smoothed ESS decision problems. This is subsequently used to provide an alternative, dual, proxy method.

1) Dual analysis: The dual of Problem (1) is given by:

$$\underset{\boldsymbol{y}}{\text{minimize}} \sum_t \left(E(\overline{\delta}_t + \overline{\gamma}_t) + \overline{S}\overline{\sigma}_t - \underline{S}\underline{\sigma}_t\right) + S_0(\mu_1 + \mu_{T+1}) \tag{11a}$$

subject to:

$$\underline{\delta}_t, \underline{\gamma}_t, \overline{\delta}_t, , \overline{\gamma}_t, \underline{\sigma}_t, \overline{\sigma}_t \geq 0 \qquad \forall t \tag{11b}$$

$$-\underline{\delta}_t + \overline{\delta}_t + \mu_t = \lambda_t \eta \qquad \forall t \tag{11c}$$

$$-\underline{\gamma}_t + \overline{\gamma}_t - \mu_t = -\lambda_t/\eta \qquad \forall t \tag{11d}$$

$$-\underline{\sigma}_t + \overline{\sigma}_t + \mu_t - \mu_{t+1} = 0 \qquad \forall t \tag{11e}$$

with $\boldsymbol{y} = \{\underline{\boldsymbol{\delta}}, \underline{\boldsymbol{\gamma}}, \overline{\boldsymbol{\delta}}, , \overline{\boldsymbol{\gamma}}, \underline{\boldsymbol{\sigma}}, \overline{\boldsymbol{\sigma}}, \boldsymbol{\mu}\}$ the set of dual variables. From constraints (11c) and (11d), and using the complementary slackness conditions [38], we find an explicit expression of the optimal (dis)charge decision:

$$\forall t : e_t^* = \begin{cases} -E/\eta & \text{if } \hat{\lambda}_t < \eta\mu_t^* \\ 0 & \text{if } \hat{\lambda}_t \in (\eta\mu_t^*, \mu_t^*/\eta) \\ \eta E & \text{if } \hat{\lambda}_t > \mu_t^*/\eta. \end{cases} \tag{12}$$

From this, we conclude that the optimal (dis)charge at time t can be obtained from the value of a single dual variable μ_t and forecasted price $\hat{\lambda}_t$. Notice from the conditions in (12) that μ_t represents the value of storage energy in each time step: its value relative to the price forecast $\hat{\lambda}_t$ determines the action. If the forecast is sufficiently small compared to μ_t, the optimal decision is to charge, whereas a discharge action follows from large price forecasts compared to μ_t. The optimal decisions are showcased as the black lines in Fig. 3. Notice from Eqs. (12) that the optimal decision is a piecewise-constant function of the forecasted price, and we find for the resulting gradients $\frac{\partial \boldsymbol{e}^*}{\partial \hat{\boldsymbol{\lambda}}} = 0$ nearly everywhere, as was mentioned in Section II-C.

When considering the smoothed optimization problem (6), Equations (11c)-(11d) in the dual problem become:

$$-\underline{\delta}_t + \overline{\delta}_t + \mu_t + \gamma \frac{\partial S}{\partial d_t} = \hat{\lambda}_t \eta \qquad \forall t \tag{13a}$$

$$-\underline{\gamma}_t + \overline{\gamma}_t - \mu_t + \gamma \frac{\partial S}{\partial c_t} = -\hat{\lambda}_t / \eta \qquad \forall t. \tag{13b}$$

A similar analysis again leads to an expression of the optimal decision as a function of $\boldsymbol{\mu}$ and $\hat{\boldsymbol{\lambda}}$. For quadratic smoothing ($\frac{\partial S}{\partial d_t} = \gamma d_t$; $\frac{\partial S}{\partial c_t} = \gamma c_t$), we find:

$$\forall t : e_t^* = \begin{cases} -E/\eta & \text{if } \hat{\lambda}_t < \eta(\mu_t - \gamma E) \\ \frac{\hat{\lambda}_t/\eta - \mu_t}{\gamma \eta} & \text{if } \hat{\lambda}_t \in [\eta(\mu_t - \gamma E), \eta \mu_t] \\ 0 & \text{if } \hat{\lambda}_t \in (\mu_t \eta, \mu_t/\eta) \\ \eta \frac{\eta \hat{\lambda}_t - \mu_t}{\gamma} & \text{if } \hat{\lambda}_t \in [\mu_t/\eta, (\mu_t + \gamma E)/\eta] \\ \eta E & \text{if } \hat{\lambda}_t > (\mu_t + \gamma E)/\eta. \end{cases} \tag{14}$$

Alternatively, for the log-barrier smoothing ($\frac{\partial S}{\partial d_t} = \frac{1}{d_t} - \frac{1}{E - d_t}$; $\frac{\partial S}{\partial c_t} = \frac{1}{c_t} - \frac{1}{E - c_t}$), the optimal decisions can be written as:

$$\begin{aligned} e_t^* = \eta & \frac{A_t^d E - 2\gamma + \sqrt{(A^d D - 2\gamma)^2 + 4 A_t^d \gamma E}}{2A^d} \\ & - \frac{A^c E - 2\gamma + \sqrt{(A_t^c D - 2\gamma)^2 + 4 A^c \gamma E}}{2 A_t^c \eta}, \end{aligned} \tag{15}$$

with $A_t^d = \hat{\lambda}_t \eta - \mu_t$ and $A_t^c = \mu_t - \hat{\lambda}_t/\eta$. These solutions are illustrated for different values of the smoothing strength γ in Figure 3. Notice how introducing a non-zero γ indeed induces a local sensitivity of the optimal decision with respect to forecasted price, and that Eqs. (14) and (15) reduce to the piece-wise constant solution of (12) when $\gamma \to 0$.

2) Dual proxy implementation: From the above discussion, we can conclude that in both the linear and smoothed cases, a closed-form expression exists that can be used to calculate the optimal primal decisions based on the forecasted price and optimal value of a single dual variable. Based on this insight, we adjust the first step of the primal proxy method to:

1) a) Using the price forecast, predict the dual variable with a pre-trained NN: $\hat{\boldsymbol{\mu}} = N^{\mu}(\hat{\boldsymbol{\lambda}})$.
   b) Apply a pre-determined smoothing function to obtain smoothed decisions: $\tilde{\boldsymbol{c}} = f_c^S(\hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\mu}})$; $\tilde{\boldsymbol{d}} = f_d^S(\hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\mu}})$.

This is visualized on the bottom of Figure 2. One of the advantages to this approach is that the output dimension of the NN reduces, inherently making the learning problem easier.

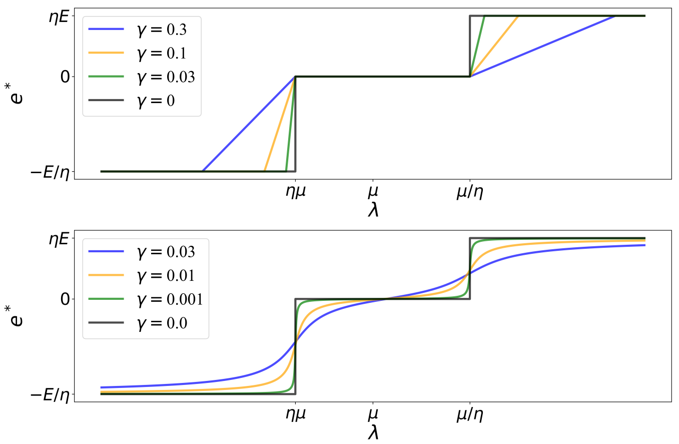

Fig. 3: Optimal decision of Problem (6) with quadratic (top) and log-barrier (bottom) smoothing for one specific instance.

Indeed, instead of expecting the NN to produce values of both d_t and c_t for every timestep, we can leverage the above expressions to reconstruct the decision based on the prediction of a single dual variable μ_t for every timestep.

Introducing a smoothing function offers three key advantages. First, it ensures the feasibility of the (dis)charge decisions, specifically adherence to box constraints (1b)-(1c). Second, it allows for directly modeling the impact of changing the price forecast. For example, when f^S is given by Eq. (14) or Eq. (15), increasing the price forecast will result in an increased net discharge. In the case of a primal proxy, this is not possible to control since the proxy NN itself captures the mapping from price forecast to the optimal decision. Third, the smoothed decision approach the optimal decisions (12) of the linear OP, while exhibiting a non-zero gradient. Our proposed proxy approach provides the modeler the opportunity to explicitly formulate a custom smoothing function f^S, without needing information on an equivalent convex optimization program.

We propose two alternatives to existing smoothings (Eqs. (14) and (15)). The first (Eq. (16)) closely relates to the quadratic smoothing, but assumes the smoothed areas center around the linear decision thresholds $\hat{\lambda}_t = \mu_t \eta$ and $\hat{\lambda}_t = \mu_t / \eta$:

$$f^S(\hat{\lambda}, \mu) = \begin{cases} -E/\eta & \text{if } \hat{\lambda} < \eta(\mu - \delta) \\ \frac{\hat{\lambda}/\eta - \mu - \delta}{\gamma \eta} & \text{if } \hat{\lambda} \in [\eta(\mu - \delta), \eta(\mu_t + \delta)] \\ 0 & \text{if } \hat{\lambda} \in (\eta(\mu + \delta), (\mu - \delta)/\eta) \\ \eta \frac{\eta \hat{\lambda} - \mu + \delta}{\gamma} & \text{if } \hat{\lambda} \in [(\mu - \delta)/\eta, (\mu + \delta)/\eta] \\ \eta E & \text{if } \hat{\lambda} > (\mu + \delta)/\eta, \end{cases} \tag{16}$$

with $\delta = \frac{\gamma E}{2}$, where we omitted the time dependency subscript $\cdot_t$ for notational simplicity. We will henceforth refer to this as the *quadratic-centered* smoothing. The second proposed smoothing function, termed *cubic-centered* smoothing (Eq. (17)), also centers around the decision thresholds, but exhibits a quadratic instead of linear sensitivity of the optimal decision with respect to the price forecast, such that gradients have the

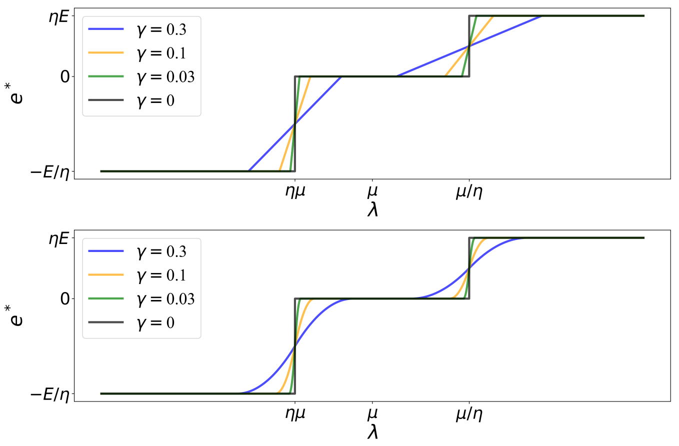

Fig. 4: Optimal decision of Problem (6) with quadratic-centered (top) and cubic-centered (bottom) smoothing for one specific instance.

highest absolute value at the threshold points:

$$f^S(\hat{\lambda}, \mu) = \begin{cases} -E/\eta & \text{if } \hat{\lambda} < \eta\mu - \delta \\ \frac{1}{\eta} q(\hat{\lambda}, -a, -b, c, x^c) & \text{if } \hat{\lambda} \in [\eta\mu - \delta, \eta\mu] \\ \frac{1}{\eta} q(\hat{\lambda}, a, -b^c, c^c, x^c) & \text{if } \hat{\lambda} \in [\eta\mu, \eta\mu + \delta] \\ 0 & \text{if } \hat{\lambda} \in \eta\mu + \delta, \mu/\eta - \delta \\ \eta q(\hat{\lambda}, a, b, c, x^d) & \text{if } \hat{\lambda} \in [\mu/\eta - \delta, \mu/\eta] \\ \eta q(\hat{\lambda}, -a, b, c, x^d) & \text{if } \hat{\lambda} \in [\mu/\eta, \mu/\eta + \delta] \\ \eta E & \text{if } \hat{\lambda} > (\mu + \gamma P)/\eta, \end{cases} \tag{17}$$

with $q(\hat{\lambda}, a, b, c, x) = a(\hat{\lambda} - x)^2 + b(\hat{\lambda} - x) + c$, $a = \frac{E}{2\delta^2}$, $b = \frac{E}{\delta}$, $c = \frac{E}{2}$, $x^c = \eta\mu$, and $x^d = \frac{\mu}{\eta}$.

The intuition behind these alternative smoothing functions (Figure 4) is that they selectively incorporate advantages of the quadratic and log-barrier smoothing functions. Like the quadratic smoothing, they exhibit regions with zero gradients, which is desirable to not further update weights when the resulting decisions are optimal. Conversely, like the log-barrier-smoothing, the regions exhibiting sensitivity center around the threshold point of the optimal decisions of linear OP (1). This results in output decisions that are more aligned with the linear decisions than those produced by quadratic smoothing.

IV. Case Study: Day-ahead Participation

This section assesses the effectiveness of the proposed DFL method and its different components by applying it to the problem of an ESS operating in the Belgian DAM. Section IV-A outlines the case study's main design choices. Through a specific example, Section IV-B demonstrates the working mechanism of the proxy models. Section IV-C reviews the profit performance, while Section IV-D compares the training times of different DFL training methods.

A. Case study design

We examine the case where a 1MW ESS participates in the Belgian DAM. Consequently, as the DAM has a granularity of 1h, this means $E = 1$ in Problem (1). The Energy-to-Power (E/P) ratio is varied between 1 and 24, and the round-trip efficiency is set at 81% ($\eta = 0.9$). Our methodology incorporates two neural networks: the first transforms input features into price forecasts, whereas the second translates these forecasts into optimal primal or dual variables. Both networks operate as time series forecasters, processing data from 24 time steps to generate a single output per time step.

For the price forecasting network, following [11], we use as input features publicly available forecasts of the load and generation in Belgium, as well as in its interconnected neighbouring countries, temporal information, and prices of the preceding day. The dataset spans from April 2019 to March 2023. The train set comprises April 2019-March 2022, while days of April 2022-March 2023 are assigned to the validation and test set in alternating order. For the price forecaster, we use an attention-based encoder-decoder RNN, see e.g. [39], where the encoder captures how input features affect prices and the decoder combines the encoder output with current day forecasts to predict prices. Prior to training the forecaster with the proposed DFL methods described in Sections II and III, we warm start the price forecaster by pre-training the NN using a Loss Tuning (LT) approach as introduced in [10]. Here, we consider a parametric family to sample MSE-like loss functions including a variability component. The loss function leading to the highest ex-post profit on the validation set is selected as the warm start.

On the other hand, the goal of the proxy network is to mimic the outcome of the ESS optimization problem. Its input features are daily prices, while its target is the corresponding optimal values of $\boldsymbol{c}$ and $\boldsymbol{d}$ (primal proxy) or $\boldsymbol{\mu}$ (dual proxy). The actual prices of the train dataset comprise the input features. Solving OP (1) with those prices yields the optimal primal and dual variables for all examples, which are the targets of the proxy NNs. We use a single decoder LSTM RNN, since information of prior days is irrelevant in obtaining the optimal outcome of the OP, and as such an encoder-decoder structure would be obsolete. The primal proxy is implemented as a single RNN yielding both $\boldsymbol{c}$ and $\boldsymbol{d}$, having an output which is twice the size of that of the dual proxy, yielding $\boldsymbol{\mu}$. For both the primal and dual proxies, the same 120 hyper-parameter combinations, varying the learning rate, number of hidden layers and neurons were used for training 120 distinct NNs.

To test the efficacy of the proposed methodological novelties, we compare the performance of several different price forecasting models trained with different traditional or DFL-based procedures, summarized in Table I. For the ID and DP methods, there is the choice of smoothing function, for which the cases of quadratic (14), log-barrier (15), quadratic-centered (16) and cubic-centered (17) smoothing are considered. The latter two are only applicable to the DP training methods. For the PP method, both a model with and without sigmoid activation function on the final layer are implemented.

B. Proxy and repair mechanism

Here, we show the working principles of the dual proxy method and the different repair mechanism through a specific

TABLE I: Overview of price forecaster training methods.

Abbr.	DFL	Model
MSE	✗	Forecaster trained to minimize MSE
LT	✗	Loss Tuning-based warm start of forecaster proposed in [10]
IDF	✓	Implicit Differentiation with full smoothing, as in Eqs. (7)
IDR	✓	Implicit Differentiation with reduced smoothing as in Eqs. (8)
PP	✓	Primal Proxy yielding (dis)charge decisions without repair
DP	✓	Dual Proxy yielding dual variable μ without repair
XPRc	✓	Primal or Dual proxy with repair on cyclic boundary conditions
XPRf	✓	Primal or Dual ML proxy with repair ensuring full feasibility
PF	-	Perfect Foresight model on electricity price

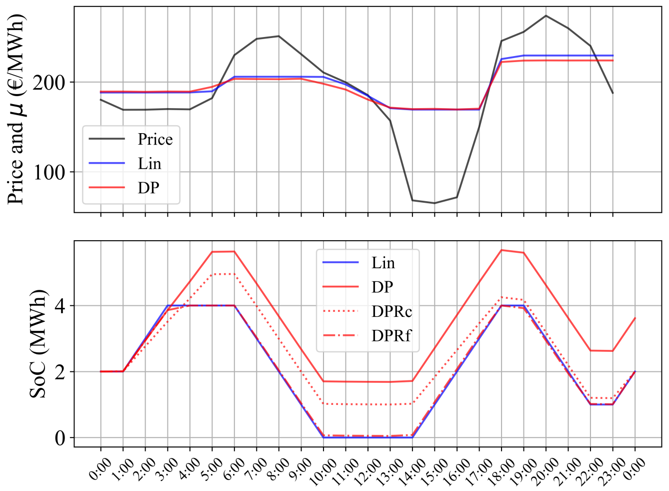

Fig. 5: Price and μ as output of the linear OP (1) compared to the output of the dual ML proxy (top), and the corresponding schedules for the output of the linear optimization (Lin) and the proxy method comparing different repair mechanisms (bottom). For the schedule, log-Barrier smoothing with $\gamma = 0.1$ was implemented. Data for May 16, 2022.

example. Figure 5 shows the values of the dual $\boldsymbol{\mu}$, comparing the optimal outcome of linear OP (1) and the output of the trained dual proxy model. It also shows the resulting SoC schedule by applying a log-barrier smoothing function with $\gamma = 0.1$, for the different repair mechanisms.

Note the strong correspondence between the optimized values of μ and the ML model outcome. However, as the lower figure shows, overestimation of μ at 04:00 and 05:00 leads to an infeasible schedule, exceeding the SoC limit of 4 MWh, resulting from the DP model. On the other hand, the decisions align with those resulting from the linear OP for the rest of the day. Ultimately, this also leads to a final SoC of 3.5 MWh, as such not respecting the cyclic boundary condition (1g). Note, however, that the repaired schedule closely matches the optimized schedule.

C. *Profit performance*

Table II shows the out-of-sample profit results for the different training models, employing various smoothing functions and a sensitivity on the E/P ratio. For each DFL-based model, 36 combinations of the learning rate and γ were implemented. Notably, for all models, the value of γ leading to the best forecaster lies between 0.1 and 30. Then, the hyper-parameter combination yielding the highest validation profit was retrained 12 times using different random seeds. This two-step procedure is only performed on the DFL-based methods to validate if they reliably outperform the best traditional benchmarks. It is worth stressing that these profit values are retrieved following the method presented in Fig. 1, where the different forecasting models are trained with their respective methods, and the trained price forecasters are applied to the exact linear optimization program on the test set to find the feasible and optimal decisions, as well as their implied out-of-sample profit.

As a first observation, the majority of integrated DFL models exhibit significant improvements over the pre-trained forecaster with loss tuning, ranging from 8% to 18% for the best performer. Interestingly, the advantage of DFL models decreases as the E/P ratio increases. This is likely due to the precision required in timing local optima at smaller E/P ratios.

Secondly, the models trained with a dual proxy (DP) consistently outperform those trained with a primal proxy (PP). This can be attributed to its incorporation of a direct link between changing price forecast values and optimal (dis)charge decisions through the smoothing function.

Thirdly, the utility of the ML proxy method allowing for custom smoothing functions becomes apparent from comparing the results of the models with quadratic smoothing to those with quadratic-centered and cubic-centered smoothing. Indeed, the DPRf forecaster with the latter two smoothings consistently outperforms that with the quadratic smoothing. This is a result from the more intuitive smoothing, see Figures 3 and 4. Interestingly, the cubic-centered smoothing even yields the best-performing model overall for $E/P = 4$ and $E/P = 12$. It is worth emphasizing that the innovation of an ML proxy was essential to achieve such custom smoothing, as the ID training method requires explicitly finding a convex optimization program leading to corresponding smoothed decisions, which may not exist for the specific functions proposed in (16) and (17).

Fourthly, the ML proxy method including repair mechanism almost always outperforms its counterpart without repair, which can be attributed to a discrepancy between the anticipated decisions by the proxy model and those derived from exact optimization, see, e.g., Fig. 5. A notable exception is the primal proxy model with linear activation of the final layer. This result follows from the observation that the decisions resulting from that proxy model violate the SoC constraints to lesser extent, leading to relatively small differences between the original and repaired schedules. For all other proxy models, incorporating a repair mechanism consistently improves out-of-sample profit by upholding the cyclic boundary conditions of the proxy schedule, as demonstrated by the results of the DPRc and PPRc methods. Additionally, integrating a repair mechanism for SoC box constraints in the DPRf method further enhances performance, particularly in models with smaller Energy-to-Power (E/P) ratios, where these constraints are more likely to be violated.

Finally, the IDR methods outperform IDF methods in all but one case. As discussed in Section II-C, reduced smoothing was introduced in IDR to maintain sensitivity of optimal decisions

TABLE II: Average profit on the test set and its standard deviation (between brackets), for trained models with different initial random seeds, employing different smoothing functions and training methods with a sensitivity on the ESS E/P ratio; all results in €/MW/qh.

f^S	**Model**	**Energy-to-Power ratio**				
		1	**4**	**8**	**12**	**24**
-	**TMSE**	0.84	2.45	3.30	3.69	3.72
	LT	0.90	2.68	3.67	3.94	3.94
Quad	**IDF**	1.06 (0.02)	2.90 (0.05)	3.91(0.05)	4.17 (0.07)	4.18 (0.09)
	IDR	1.04 (0.02)	3.00 (0.04)	3.92 (0.06)	4.26 (0.05)	4.21 (0.17)
	DP	0.90 (0.00)	2.68 (0.00)	3.67 (0.00)	3.94 (0.00)	3.94 (0.00)
	DPRc	0.93 (0.02)	2.88 (0.15)	3.76 (0.10)	4.24 (0.05)	4.22 (0.06)
	DPRf	0.95 (0.03)	2.99 (0.04)	3.89 (0.06)	4.20 (0.06)	4.21 (0.07)
LogBar	**IDF**	1.06 (0.02)	3.01 (0.03)	3.67 (0.00)	3.94 (0.0)	3.94 (0.00)
	IDR	**1.06** (0.02)	3.03 (0.05)	**3.97** (0.03)	4.28 (0.06)	4.29 (0.07)
	DP	0.90 (0.00)	2.68 (0.00)	3.67 (0.00)	3.94 (0.00)	3.94 (0.00)
	DPRc	0.98(0.02)	3.00 (0.05)	3.90 (0.06)	4.26 (0.05)	**4.33** (0.04)
	DPRf	1.05 (0.04)	3.03 (0.02)	3.89 (0.06)	4.23 (0.03)	4.06 (0.13)
Quad-c	**DP**	0.91 (0.02)	2.68 (0.00)	3.67 (0.00)	3.94 (0.00)	3.94 (0.00)
	DPRc	0.94 (0.03)	2.97 (0.10)	3.90 (0.02)	4.23 (0.09)	4.28 (0.06)
	DPRf	1.06 (0.05)	3.02 (0.04)	3.93 (0.04)	4.28 (0.08)	4.32 (0.05)
Cube-c	**DP**	0.90 (0.00)	2.68 (0.02)	3.67 (0.00)	3.94 (0.00)	3.94 (0.00)
	DPRc	0.97 (0.03)	2.96 (0.05)	3.92 (0.03)	**4.29** (0.07)	4.24 (0.09)
	DPRf	1.06 (0.01)	**3.03** (0.03)	3.90 (0.03)	4.26 (0.09)	4.32 (0.07)
Linear	**PP**	1.02 (0.03)	2.98 (0.05)	3.86 (0.06)	4.21 (0.05)	4.17 (0.09)
	PPRc	1.03 (0.01)	2.97 (0.04)	3.89 (0.03)	4.23 (0.07)	4.21 (0.09)
	PPRf	1.00 (0.05)	2.97 (0.03)	3.86 (0.05)	4.22 (0.06)	4.26 (0.06)
Sigmoid	**PP**	0.93 (0.03)	2.72 (0.07)	3.68 (0.02)	3.94 (0.03)	3.93 (0.03)
	PPRc	0.94 (0.05)	2.97 (0.05)	3.90 (0.05)	4.25 (0.07)	4.29 (0.06)
	PPRf	1.01 (0.03)	2.99 (0.03)	3.89 (0.05)	4.25 (0.05)	4.26 (0.04)
-	**PF**	1.51	3.96	5.25	5.78	5.87

with respect to price forecasts while avoiding penalization of specific SoC values, thereby remaining closer to the original linear optimization problem. These findings underscore the importance of considering the specifics of the downstream problem when implementing a convex smoothing in the ID technique. This approach is particularly effective with log-barrier smoothing, where the IDF method fails to improve upon the LT warm start for ESS with $E/P > 4$.

D. Train times

Table III compares the required train times of the different decision-focused methods. All experiments were implemented in Pytorch 1.8 utilizing CUDA 11.2 for parallel GPU training, and conducted on a system equipped with an NVIDIA Quadro 600 GPU and 64 GB of memory. As the dual proxy was shown to outperform the primal proxy in terms of out-of-sample profit, Table III reports only the training times for the dual proxy with quadratic and log-barrier smoothing. However, it is worth noting that the training times in case of using other smoothing functions and a primal proxy are similar, as all proxy approaches use the same neural network architecture and repair mechanisms.

The first observation is that, as expected, the ML proxy methods outperform the ID methods in terms of computational efficiency, being at least 1.9 times faster for quadratic smoothing and 9.6 times for log-barrier smoothing. While the log-barrier smoothing is considerably slower than the quadratic smoothing due to the more strenuous solution procedure for the ID method, it has nearly no effect on the speed of the proxy method. Combining these results with the insights from

TABLE III: Train time in seconds per epoch of the price forecaster in the different decision-focused methods applied to an ESS with $E/P = 4$, for quadratic and log-barrier smoothing. A breakdown is provided of time required to compute the forward pass (FW), backward pass (BW) and validation performance (VAL).

	Quadratic				**Log-barrier**			
	FW	**BW**	**VAL**	**TOT**	**FW**	**BW**	**VAL**	**TOT**
IDF	11.9	5.4	0.7	**18.0**	93.3	29.1	0.6	**123.1**
IDR	11.6	4.1	0.7	**16.4**	76.3	11.0	0.7	**87.9**
DP	2.0	0.9	0.7	**3.6**	2.5	1.2	0.6	**4.4**
DPRc	2.9	1.2	0.7	**4.9**	3.0	1.4	0.7	**5.2**
DPRf	3.9	3.9	0.7	**8.5**	4.2	4.3	0.6	**9.2**

Section IV-C, we can conclude that the DPRf method, especially with cubic-centered smoothing, tends to outperform the ID method with quadratic smoothing while being considerably faster. Although the DPRf method in some cases yields slightly lower profit performance than the IDR model with log-barrier smoothing, it is much more computationally efficient.

Interestingly, imposing the iterative repair mechanism to avoid violation of the SoC box constraints put forth in Algorithm 2 slows down the training procedure for the proxy method significantly. Only employing the repair mechanism on the cyclic boundary condition as in Algorithm 1 increases the speed-up in training compared to the IDR model to 3.3 times in case of quadratic smoothing, and 16.9 times when log-barrier smoothing is used. This suggests potential for future research into more efficient repair mechanisms that comply with SoC box constraints without sacrificing training speed.

For the specific implementation in this case study, the DPRf method reduces the training time by 26 minutes for 200 epochs per model compared to training the forecaster with the IDR approach. This can be compared to the required train time for the proxy model, amounting 17 minutes. Crucially, the time invested in ML proxy training is a one-time cost, offering long-term efficiency benefits for repeated or continuous model training across various hyperparameters, particularly the smoothing value γ. This reusability of the ML proxy model for ongoing training presents substantial opportunities for further computational savings.

V. Conclusion

In electricity markets, ESS operators can enhance profits using a predict-then-optimize framework, where price forecasts feed into a downstream optimization problem. DFL allows training forecasters to maximize the profit of optimized decisions rather than minimizing statistical errors on the price output. As opposed to RL, this approach allows the ESS owner to retain interpretability of the optimized decisions. However, the prevalent DFL technique using implicit differentiation of the KKT conditions of the downstream problem exhibits long train times, as it requires solving the downstream optimization in every forward pass of the training procedure. Moreover, for linear downstream problems, this method is constrained by its reliance on convex smoothings of the optimization problem.

In this paper, we proposed an ML proxy method as an alternative to the implicit differentiation training technique. The optimization in the forward pass was replaced with a pre-trained neural network, as such significantly speeding up the training process. In addition, the proposed approach based on duality theory allows formulating a proxy for the dual problem and custom smoothing functions, offering greater flexibility in modeling how prices affect ESS decisions compared to implicit differentiation.

In a case study of an ESS participating in the Belgian day-ahead market, we re-affirmed the utility of DFL as it increases out-of-sample profit with 8% to 18% compared to traditional ML training methods, depending on the size of the ESS. Compared to the technique of implicit differentiation, the ML proxy method demonstrated a speed-up by a factor of 1.9 to 9.6, depending on the smoothing function used, while matching the obtained out-of-sample profit. Finally, we highlighted the efficacy of custom smoothing functions, reporting consistently higher profit performance compared to the smoothing resulting form the implicit differentiation method.

References

[1] David Wozabal, Christoph Graf, and David Hirschmann. “The effect of intermittent renewables on the electricity price variance”. In: *OR Spectrum* 38 (2016), pp. 687–709.

[2] Kendall Mongird et al. *Energy storage technology and cost characterization report*. Tech. rep. Pacific Northwest National Lab. (PNNL), Richland, WA (United States), 2019.

[3] Juan Arteaga and Hamidreza Zareipour. “A Price-Maker/Price-Taker Model for the Operation of Battery Storage Systems in Electricity Markets”. eng. In: *IEEE Transactions on Smart Grid* 10.6 (2019), pp. 6912–6920.

[4] Emma Wessel, Ruben Smets, and Erik Delarue. “Risk-aware participation in day-ahead and real-time balancing markets for energy storage systems”. In: *Electric Power Systems Research* 235 (2024), p. 110741.

[5] Uğur Yıldıran and İsmail Kayahan. “Risk-averse stochastic model predictive control-based real-time operation method for a wind energy generation system supported by a pumped hydro storage unit”. In: *Applied Energy* 226 (2018), pp. 631–643.

[6] Roohallah Khatami, Konstantinos Oikonomou, and Masood Parvania. “Look-ahead optimal participation of compressed air energy storage in day-ahead and real-time markets”. In: *IEEE Transactions on Sustainable Energy* 11.2 (2019), pp. 682–692.

[7] Jean-François Toubeau et al. “Data-driven scheduling of energy storage in day-ahead energy and reserve markets with probabilistic guarantees on real-time delivery”. In: *IEEE Transactions on Power Systems* 36.4 (2020), pp. 2815–2828.

[8] K. Pandžić, K. Bruninx, and H. Pandžić. “Managing Risks Faced by Strategic Battery Storage in Joint Energy-Reserve Markets”. In: *IEEE Transactions on Power Systems* 36.5 (2021), pp. 4355–4365. DOI: 10.1109/TPWRS.2021.3058936.

[9] Dheepak Krishnamurthy et al. “Energy Storage Arbitrage Under Day-Ahead and Real-Time Price Uncertainty”. eng. In: *IEEE Transactions on Power Systems* 33.1 (2018), pp. 84–93.

[10] Ruben Smets et al. “Value-Oriented Forecasting of Imbalance Prices Through Loss Function Tuning for Optimal Control of Energy Storage Systems”. In: *Available at SSRN 4648734* (2024).

[11] Jesus Lago, Fjo De Ridder, and Bart De Schutter. “Forecasting spot electricity prices: Deep learning approaches and empirical comparison of traditional algorithms”. In: *Applied Energy* 221 (2018), pp. 386–405.

[12] Siyu Zhou et al. “An optimized heterogeneous structure LSTM network for electricity price forecasting”. In: *Ieee Access* 7 (2019), pp. 108161–108173.

[13] Jeremie Bottieau et al. “Interpretable Transformer Model for Capturing Regime Switching Effects of Real-Time Electricity prices”. eng. In: *IEEE Transactions on Power Systems* 38.3 (2023), pp. 1–14.

[14] Qi Wang et al. “A Comprehensive Survey of Loss Functions in Machine Learning”. eng. In: *Annals of Data Science* 9.2 (2022), pp. 187–212.

[15] Jakub Nowotarski and Rafał Weron. “Recent advances in electricity price forecasting: A review of probabilistic forecasting”. In: *Renewable and Sustainable Energy Reviews* 81 (2018), pp. 1548–1568.

[16] Jiale Wang et al. “Risk-Averse Optimal Combining Forecasts for Renewable Energy Trading under CVaR Assessment of Forecast Errors”. eng. In: *IEEE Transactions on Power Systems* (2023), pp. 1–13.

[17] Jialun Zhang, Yi Wang, and Gabriela Hug. “Cost-oriented load forecasting”. eng. In: *Electric Power Systems Research* 205 (2022), p. 107723.

[18] Jayanta Mandi et al. “Decision-focused learning: Foundations, state of the art, benchmark and future opportunities”. In: *arXiv preprint arXiv:2307.13565* (2023).

[19] Adam N. Elmachtoub, Jason Cheuk Nam Liang, and Ryan Mcnellis. “Decision Trees for Decision-Making under the Predict-then-Optimize Framework”. In: *Proceedings of the 37th International Conference on Machine Learning*. Vol. 119. Proceedings of Machine Learning Research. PMLR, 13–18 Jul 2020, pp. 2858–2867.

[20] Akylas Stratigakos et al. “Prescriptive Trees for Integrated Forecasting and Optimization Applied in Trading of Renewable Energy”. eng. In: *IEEE Transactions on Power Systems* 37.6 (2022), pp. 1–1.

[21] Brandon Amos and J Zico Kolter. “Optnet: Differentiable optimization as a layer in neural networks”. In: *International Conference on Machine Learning*. PMLR. 2017, pp. 136–145.

[22] Akshay Agrawal et al. “Differentiating through a cone program”. In: *arXiv preprint arXiv:1904.09043* (2019).

[23] Akshay Agrawal et al. “Differentiable convex optimization layers”. In: *Advances in Neural Information Processing Systems* 32 (2019).

[24] Bryan Wilder, Bistra Dilkina, and Milind Tambe. “Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 1658–1665.

[25] Aaron Ferber et al. “Mipaal: Mixed integer program as a layer”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 02. 2020, pp. 1504–1511.

[26] Jayanta Mandi and Tias Guns. “Interior point solving for lp-based prediction+ optimisation”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 7272–7282.

[27] Quentin Berthet et al. “Learning with differentiable pertubed optimizers”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 9508–9519.

[28] Guillaume Dalle et al. “Learning with combinatorial optimization layers: a probabilistic approach”. In: *arXiv preprint arXiv:2207.13513* (2022).

[29] Adam N Elmachtoub and Paul Grigas. “Smart “predict, then optimize””. In: *Management Science* 68.1 (2022), pp. 9–26.

[30] Linwei Sang et al. “Electricity price prediction for energy storage system arbitrage: A decision-focused approach”. In: *IEEE Transactions on Smart Grid* 13.4 (2022), pp. 2822–2832.

[31] Jinhao Li et al. “Temporal-Aware Deep Reinforcement Learning for Energy Storage Bidding in Energy and Contingency Reserve Markets”. In: *arXiv preprint arXiv:2402.19110* (2024).

[32] Jaeik Jeong, Seung Wan Kim, and Hongseok Kim. “Deep reinforcement learning based real-time renewable energy bidding with battery control”. In: *IEEE Transactions on Energy Markets, Policy and Regulation* (2023).

[33] Seyed Soroush Karimi Madahi, Bert Claessens, and Chris Develder. “Distributional Reinforcement Learning-based Energy Arbitrage Strategies in Imbalance Settlement Mechanism”. In: *arXiv preprint arXiv:2401.00015* (2023).

[34] Priya L Donti, David Rolnick, and J Zico Kolter. “DC3: A learning method for optimization with hard constraints”. In: *arXiv preprint arXiv:2104.12225* (2021).

[35] Wenbo Chen, Mathieu Tanneau, and Pascal Van Hentenryck. “End-to-end feasible optimization proxies for large-scale economic dispatch”. In: *IEEE Transactions on Power Systems* (2023).

[36] Thomas Mercier, Mathieu Olivier, and Emmanuel De Jaeger. “The value of electricity storage arbitrage on day-ahead markets across Europe”. In: *Energy Economics* 123 (2023), p. 106721.

[37] Adam Paszke et al. “Automatic differentiation in pytorch”. In: (2017).

[38] Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[39] Jeremie Bottieau et al. “Very-Short-Term Probabilistic Forecasting for a Risk-Aware Participation in the Single Price Imbalance Settlement”. eng. In: *IEEE Transactions on Power Systems* 35.2 (2020), pp. 1218–1230.

Energy Systems Integration & Modeling Group (ESIM):

Web-page: `https://www.mech.kuleuven.be/esim`

LinkedIn: `https://www.linkedin.com/company/esimrg`

Latest preprints and accepted publications: `https://www.mech.kuleuven.be/esim-publications`