Self-Certifying Primal-Dual Optimization Proxies for Large-Scale Batch Economic Dispatch

Michael Klamkin, Mathieu Tanneau, Pascal Van Hentenryck

Georgia Institute of Technology

Suggested Citation

M. Klamkin, M. Tanneau, and P. Van Hentenryck, "Self-Certifying Primal-Dual Optimization Proxies for Large-Scale Batch Economic Dispatch," in *Proceedings of the PowerUp Conference*, 2026.

BibTeX Entry

```
@inproceedings{klamkin2026selfcertifying,
  author = {Klamkin, Michael and Tanneau, Mathieu and Van Hentenryck, Pascal},
  title = {Self-Certifying Primal-Dual Optimization Proxies for Large-Scale Batch
      Economic Dispatch},
  booktitle = {Proceedings of the PowerUp Conference},
  year = {2026},
}
```

Self-Certifying Primal-Dual Optimization Proxies for Large-Scale Batch Economic Dispatch

Michael Klamkin[†‡], Mathieu Tanneau[†], Pascal Van Hentenryck[†]

***Abstract*—Recent research has shown that optimization proxies can be trained to high fidelity, achieving average optimality gaps under 1% for large-scale problems. However, worst-case analyses show that there exist in-distribution queries that result in orders of magnitude higher optimality gap, making it difficult to trust the predictions in practice. This paper aims at striking a balance between classical solvers and optimization proxies in order to enable *trustworthy deployments with interpretable speed-optimality tradeoffs based on a user-defined optimality threshold*. To this end, the paper proposes a hybrid solver that leverages duality theory to efficiently bound the optimality gap of predictions, falling back to a classical solver for queries where optimality cannot be certified. To improve the achieved speedup of the hybrid solver, the paper proposes an alternative training procedure that combines the primal and dual proxy training. Experiments on large-scale transmission systems show that the hybrid solver is highly scalable. The proposed hybrid solver achieves speedups of over 1000x compared to a parallelized simplex-based solver while guaranteeing a maximum optimality gap of 2%.**

***Index Terms*—deep learning, economic dispatch, power systems, optimization proxies**

I. Introduction

In order to plan and operate modern power grids, Transmission Systems Operators (TSOs) regularly have to solve large numbers of parametric optimization problems, e.g., multi-year simulations for transmission and capacity planning [1], or Monte-Carlo simulations for short-term risk analysis [2], [3]. Parametric optimization problems also appear in stochastic optimization formulations, which have received growing interest as renewable generation and distributed energy resources are integrated into the grid [4], [5].

The computational cost of solving large batches of similar instances (in the tens or hundreds of thousands) has fueled significant interest in *optimization proxies*, Machine Learning-based (ML) surrogate models that emulate the behavior of optimization solvers. Once trained, optimization proxies can produce close-to-optimal solutions to large-scale problems in milliseconds [6], [7], bringing performance guarantees to proxy-based market clearing and risk analysis [3]. Nevertheless, although proxies enjoy inference speeds that are orders of magnitude faster than classical optimization algorithms, their lack of strong worst-case performance guarantees has hindered their widespread adoption in real-life settings.

To address this limitation, the paper proposes a hybrid solver framework that, for the first time, combines the speed of data-driven proxies with the worst-case guarantees of optimization solvers. A core component of this framework is to jointly learn primal *and* dual feasible solutions, thus providing a self-contained mathematical certificate of performance in the form of a provable duality gap. This unique self-certifying capability enables the systematic detection of poor predictions *online*, without ground truth information. Namely, when the duality gap of a predicted primal-dual solution exceeds a user-prescribed optimality tolerance, the corresponding instance is solved exactly using a classical optimization solver. This strategy ensures worst-case optimality guarantees while providing a large overall speedup since, in practice, only a small fraction of samples exhibit poor performance. In addition, it provides an interpretable way to tradeoff optimality and speedup. Such optimality guarantees are crucial in the context of market clearing with performance requirements and real-time risk analysis based on proxies [3], ensuring that the simulated system states are consistent with actual operations.

A. Related Works

1) Optimization Proxies: Recent research has explored how to design *feasible* optimization proxies: architectures that guarantee by construction the feasibility of the solution [8]–[13]. In particular, [10] proposes a differentiable repair layer based on proportional response to efficiently produce primal-feasible solutions to the economic dispatch problem. Nevertheless, despite substantial progress on the theory and practice of optimization proxies, their widespread adoption remains hindered by their lack of worst-case performance guarantees. Indeed, [14] show that, while proxies may achieve low optimality gaps on average (e.g. under 1%), their worst-case performance can be orders of magnitude larger (e.g. above 100%), well beyond practically reasonable tolerances.

2) Neural Network Verification: In order to enable responsible adoption of proxies in practice, recent work has began to consider how to bound the sub-optimality of trained proxies by adapting techniques from neural network verification (NNV) [15]. [16] applies MIP reformulation techniques for verifying the worst-case infeasibility of ACOPF proxies. [17] proposes primal acceleration techniques, using state-of-the-art NNV solvers [18] for verifying DCOPF proxies. [14] considers primal-feasible proxies for economic dispatch, proposing a first-order method to approximate the worst-case sub-optimality.

It is important to note that [14], [16], [17] all show that worst-case performance in trained optimization proxies is often orders of magnitude worse than average performance; in other words, optimization proxies are not immune to adversarial attacks. Thus, relying on neural network verification tools as a screening procedure for proxy adoption, besides being computationally expensive, leads to overly pessimistic conclusions.

3) ML-accelerated Optimization: Another line of work considers warm-starting schemes, where machine learning

[†]Affiliated with NSF AI Institute for Advances in Optimization, Georgia Institute of Technology, Atlanta, Georgia, United States.

[‡]Corresponding Author. Email: klam@gatech.edu.

predictions are used to set the initialization of a classical solver. Then, the overall scheme inherits both feasibility and optimality guarantees from the termination of the classical solver. Prior works in this direction report speedups on the order of 2-25x [19]–[22]. This paper strikes a balance between warm-starting and directly using predictions; it proposes a scheme that allows to avoid the solver entirely for a majority of queries, unlocking further speedup on the order of 1000x.

B. Contributions and Outline

The paper contributions are summarized as follows:

1) It introduces a principled framework that fuses AI and optimization and provides, for the first time, *controllable* worst-case optimality guarantees without 1) requiring any expensive offline verification step or 2) imposing restrictions on the underlying architectures. Importantly, the optimality tolerance is prescribed by users rather than being derived from properties of the underlying models.
2) It improves the training of primal-dual proxies by taking advantage of their self-certifying nature and uses a dedicated loss function.
3) It presents a complete implementation of the proposed framework for the economic dispatch problem and conducts extensive numerical experiments on large-scale instances based on the European transmission system.
4) It demonstrates state-of-the-art performance for solving large batches of industry-scale economic dispatch problems. On the 9241_pegase test case, the hybrid solver achieves up to 925x speedup over a parallelized simplex-based solver while guaranteeing worst-case optimality below 1%, and over 1000x at 2%.

The rest of the paper is organized as follows: Section II proposes the hybrid solver framework, Section III describes the proposed training algorithm, Section IV presents the numerical experiments, and Section V concludes the paper.

II. Self-Certifying Proxies

This section introduces the notations used throughout the paper by reviewing linear programming, duality, and optimization proxies. Then, it presents the hybrid solver, which exploits the self-certifying property of primal-dual feasible solutions.

A. Linear Programming and Duality

For ease of reading, the proposed framework is stated for general linear programming (LP) problems. Namely, consider a parametric LP problem in primal-dual form

$$P_\theta: \quad \min_{\mathbf{x}} \quad \left\{c_\theta^\top \mathbf{x} \;\middle|\; A_\theta \mathbf{x} \geq b_\theta\right\}, \tag{1a}$$

$$D_\theta: \quad \max_{\mathbf{y}\geq 0} \quad \left\{b_\theta^\top \mathbf{y} \;\middle|\; A_\theta^\top \mathbf{y} = c_\theta\right\}, \tag{1b}$$

where $\theta \in \mathbb{R}^p$ is the vector of parameters, which takes value in $\Theta \subseteq \mathbb{R}^p$, and $A_\theta \in \mathbb{R}^{m\times n}$ $b_\theta \in \mathbb{R}^m$ and $c_\theta \in \mathbb{R}^n$ are the (parametric) left-hand side matrix, right-hand side, and objective, respectively. For $\theta \in \Theta$, the primal and dual feasible sets are

$$\mathcal{X}_\theta = \{\mathbf{x} \in \mathbb{R}^n \mid A_\theta \mathbf{x} \geq b_\theta\}, \tag{2a}$$

$$\mathcal{Y}_\theta = \{\mathbf{y} \in \mathbb{R}^m \mid A_\theta^\top \mathbf{y} = c_\theta, \mathbf{y} \geq 0\}. \tag{2b}$$

The paper assumes that $\mathcal{X}_\theta \neq \emptyset$ and $\mathcal{Y}_\theta \neq \emptyset, \forall \theta \in \Theta$, i.e., the primal-dual feasible set is always non-empty. This mild technical assumption ensures that strong duality always holds.

The primal (resp. dual) objective value of a primal-feasible (resp. dual-feasible) solution $\mathbf{x} \in \mathcal{X}_\theta$ (resp. $\mathbf{y} \in \mathcal{Y}_\theta$) is denoted by $\phi_\theta(\mathbf{x}) = c_\theta^\top \mathbf{x}$ (resp. $\psi_\theta(\mathbf{y}) = b_\theta^\top \mathbf{y}$). The optimal value of P_θ (resp. D_θ) is denoted by $\phi_\theta^\star$ (resp. $\psi_\theta^\star$). Optimal solutions to the primal and dual problems are denoted by $\mathbf{x}_\theta^\star$ and $\mathbf{y}_\theta^\star$. By LP duality, the primal- and dual-optimal values are equal, and any primal (resp. dual) feasible solution yields a valid upper (resp. lower) bound on the optimal value, i.e.,

$$\forall \theta \in \Theta, \forall (\mathbf{x}, \mathbf{y}) \in \mathcal{X}_\theta \times \mathcal{Y}_\theta, \;\; \psi_\theta(\mathbf{y}) \leq \psi^\star = \phi^\star \leq \phi_\theta(\mathbf{x}).$$

Finally, given $(\mathbf{x}, \mathbf{y}) \in \mathcal{X}_\theta \times \mathcal{Y}_\theta$, define the *duality gap*

$$\Gamma_\theta(\mathbf{x}, \mathbf{y}) = \phi_\theta(\mathbf{x}) - \psi_\theta(\mathbf{y}) \geq 0. \tag{3}$$

The duality gap provides a measure of how close to optimum a primal-dual solution is. Namely,

$$\Gamma_\theta(\mathbf{x}, \mathbf{y}) \geq \phi_\theta(\mathbf{x}) - \phi_\theta^\star \geq 0, \tag{4a}$$

$$\Gamma_\theta(\mathbf{x}, \mathbf{y}) \geq \psi_\theta^\star - \psi_\theta(\mathbf{y}) \geq 0. \tag{4b}$$

It is important to note that the duality gap only requires the knowledge of a primal-dual *feasible* solution, i.e., $\Gamma_\theta(\mathbf{x}, \mathbf{y})$ *can be computed without knowing the problem's optimal value*.

Remark: The LP setting is not a very restrictive assumption, as the proposed framework readily applies to nonlinear convex problems using conic duality. Readers are referred to [23] for a more detailed review of conic optimization and duality.

B. Hybrid Solver

The paper assumes access to primal and dual optimization proxies that produce primal and dual solutions, respectively. Formally, let p_α and d_β denote the primal and dual proxy models respectively, and let $\hat{\mathbf{x}} = p_\alpha(\theta)$ and $\hat{\mathbf{y}} = d_\beta(\theta)$ denote the predicted primal and dual solutions for parameter $\theta \in \Theta$. α (resp. β) refers to the primal (resp. dual) proxy's trainable parameters. It is further assumed that both proxies produce feasible solutions, i.e.,

$$\forall \theta \in \Theta, (p_\alpha(\theta), d_\beta(\theta)) \in \mathcal{X}_\theta \times \mathcal{Y}_\theta. \tag{5}$$

The assumption of primal-dual feasibility in Eq. (5) is not too restrictive. Indeed, state-of-the-art neural network architectures such as ΠNet [24] or RAYEN [11] can impose arbitrary convex constraints on their outputs. In the specific case of economic dispatch problems, dedicated repair layers have been proposed that ensure primal feasibility [10]. In addition, [25] show how to build dual-feasible proxies for convex problems with bounded variables, which is the case for the economic dispatch problems considered in this work. Section IV-B presents the detailed architectures used for primal and dual proxies.

The proposed hybrid solver is presented in Algorithm 1. The solver takes as input the instance's parameter value θ and a user-defined optimality tolerance ϵ. First, a primal-dual solution $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ is predicted using the primal and dual proxies, whose duality gap $\hat{g}$ is computed using (3). If the duality gap

Algorithm 1 Hybrid solver inference

Input: query θ, optimality threshold ϵ
Output: feasible, ϵ-optimal prediction $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$

```
Predict (x̂, ŷ) ∈ X_θ × Y_θ
Compute ĝ = Γ_θ(x̂, ŷ)
if ĝ ≤ ε then
    return (x̂, ŷ)
end if
return (x⋆_θ, y⋆_θ)
```

$\hat{g}$ is below the optimality tolerance ϵ, the predicted solution $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ is returned. Otherwise, an optimal solution $(\mathbf{x}^\star_\theta, \mathbf{y}^\star_\theta)$ is computed using an optimization solver. Thus, the worst-case optimality gap of the hybrid solver is at most ϵ.

It is important to note that the core enabler of the proposed framework is the availability of $\hat{g}$ as a self-contained certificate of sub-optimality, by leveraging LP duality and the feasibility of the predicted solution $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$. *This key result makes it possible to detect "bad" predictions online, without ground truth information, and call the optimization solver only when needed.* To the best of the authors' knowledge, this is the first data-driven hybrid framework that provides *controllable* worst-case optimality guarantees without any offline verification step.

III. Joint Primal-Dual Training

The standard self-supervised optimization proxy training performs empirical risk minimization (maximization) using the primal (dual) objective value [7]. Using a finite set of samples $\mathcal{D}$, the primal proxy training problem reads

$$\min_{\alpha} \quad \frac{1}{|\mathcal{D}|} \sum\nolimits_{\theta \in \mathcal{D}} c_\theta^\top \left(p_\alpha(\theta)\right) \tag{6}$$

Optimization proxies are typically relatively small (under 10M parameters), making it tractable to train the primal and dual models in parallel, on a single consumer GPU even for large scale power systems[1]. Training the primal and dual models in parallel has several benefits, as discussed next. The parallel empirical risk minimization problem reads

$$\min_{\alpha,\beta} \quad \frac{1}{|\mathcal{D}|} \sum\nolimits_{\theta \in \mathcal{D}} \Gamma_\theta\big(p_\alpha(\theta),\, d_\beta(\theta)\big) \tag{7}$$

where α and β represent the trainable parameters of the neural networks in the primal and dual optimization proxies respectively. Note that in terms of training gradients, the problem is totally separable in α and β. Furthermore, it recovers exactly the primal (dual)-only training problems due to the definition of Γ_θ.

Although prior work has considered learning schemes that produce estimates of both primal and dual solutions, to the authors' knowledge, this work is the first to directly use the duality gap as the training loss. Enabled by the feasibility of the predictions, and in contrast to prior work based on alternating primal-dual updates [26], [27] and penalty-based losses [28], [29], the proposed duality gap loss method yields a streamlined implementation by introducing no additional hyperparameters. Furthermore, the duality gap loss results in more stable and predictable training due to its separable form, since the primal (resp. dual) training gradient does not depend on the current performance of the dual (resp. primal) network. The streamlined and stabilized training is important for applications such as real-time risk analysis and market clearing where rapid automated re-training is required [30].

[1]The largest case considered in the experiments, `9241_pegase`, requires only 8GB of GPU memory for parallel training.

A. Normalization

A key feature of the primal-dual setting is that it allows for principled normalization. Besides not relying on the optimal value, which is unknown at training and inference time, it is important that the normalization retains the self-certifying property – that the duality gap bounds from above the true optimality gaps – when such a guarantee is needed. In these cases, e.g. to implement Algorithm 1 for normalized optimality thresholds $\epsilon\%$, the predicted dual objective value is used, since it bounds from below the optimal value. This yields the (proper) normalized gap

$$\bar{\Gamma}_\theta(\mathbf{x}, \mathbf{y}) \coloneqq \frac{\Gamma_\theta(\mathbf{x}, \mathbf{y})}{\psi_\theta(\mathbf{y})} \tag{8}$$

which bounds from above the ground truth optimality gaps of the primal and dual predictions, i.e. replacing Γ_θ with $\bar{\Gamma}_\theta$ in Step 2 of Algorithm 1 allows to certify

$$\frac{\phi_\theta(\mathbf{x}) - \phi^\star_\theta}{\phi^\star_\theta} \leq \epsilon\%, \quad \text{and} \quad \frac{\psi^\star_\theta - \psi_\theta(\mathbf{y})}{\psi^\star_\theta} \leq \epsilon\%.$$

In contexts where an exact bound on the true normalized optimality gap is not required, e.g. for normalizing the loss during training, estimates of the optimal value can be used. In particular, the experiments in Section IV-C normalize the training loss using the midpoint:

$$\tilde{\Gamma}_\theta(\mathbf{x}, \mathbf{y}) \coloneqq \frac{\Gamma_\theta(\mathbf{x}, \mathbf{y})}{(\phi_\theta(\mathbf{x}) + \psi_\theta(\mathbf{y}))/2} \tag{9}$$

Note that the denominator is treated as a constant w.r.t. α and β when performing automatic differentiation during training, and validation metrics use the proper normalized gap (8).

B. Targeting an optimality tolerance

When jointly training primal-dual feasible optimization proxies for later use in a hybrid solver, practitioners often already have a target optimality threshold ϵ in mind. In this case, the following ReLU/hinge loss more directly optimizes the speedup of the hybrid solver:

$$\min_{\alpha,\beta} \quad \frac{1}{|\mathcal{D}|} \sum\nolimits_{\theta \in \mathcal{D}} \Gamma^\epsilon_\theta \tag{10a}$$

$$\text{where} \quad \Gamma^\epsilon_\theta \coloneqq \max(\Gamma_\theta\big(p_\alpha(\theta),\, d_\beta(\theta)\big) - \epsilon, 0) \tag{10b}$$

since it focuses on improving the proxies at points θ where $\Gamma_\theta\big(p_\alpha(\theta),\, d_\beta(\theta)\big) > \epsilon$, which would trigger the fallback to the classical solver in Algorithm 1. The experiments compare Γ^0 to $\Gamma^{1\%}$; note that Γ^0 recovers (7) due to (3).

C. Sampling without replacement

Optimization proxies are typically trained for thousands of epochs with $|\mathcal{D}|$ in the tens to hundreds of thousands [6], where one epoch corresponds to one "complete traversal" through $\mathcal{D}$; in other words, each sample is seen many times during training. However, unlike the supervised learning setting where training data instances must be solved apriori – representing a considerable computational burden – in the self-supervised setting, generating new samples requires minimal computation. Thus, the paper proposes to *resample $\mathcal{D}$ at each epoch so that each query is only seen once throughout training*.

This is equivalent to using a very large finite dataset, though can be implemented much more efficiently. In particular, the need for storage and CPU-GPU transfer of training data batches is eliminated since batches can be sampled on-demand, directly on the GPU. Although efficient and simple to implement, such implementations are uncommon due to the lack of a dependable convergence measure in the primal (dual)-only setting. Indeed, even self-supervised learning implementations are often based around finite datasets in order to leverage pre-computed optimal solutions to implement features such as performance-based dynamic scheduling of learning rate, checkpointing, and termination, necessitating labeling at least the validation set samples and increasing the overall training time. Training the primal and dual proxies jointly enables implementing these crucial features using reliable metrics based on the (normalized) duality gap, fully eliminating the requirement for labeled data to begin training. Importantly, this enables immediate and trustworthy (re-)training within larger workflows.

IV. Numerical Experiments

This section first describes the economic dispatch formulation used in this work as well as the architectures used to implement the primal and dual proxies. Then, experimental results are presented, using the hybrid solver framework to accelerate solving batches of economic dispatch instances on large-scale power system benchmarks 1354_pegase, 2869_pegase, and 9241_pegase from PGLib [31]. Summary statistics for each benchmark are included in Table II.

A. Economic Dispatch Formulation

Consider a power grid comprising N buses, E branches, D loads and G generators. The economic dispatch problem is formulated as the parametric LP problem

$$\begin{aligned}
\min_{\mathbf{p}^{\mathrm{g}},\mathbf{p}^{\mathrm{f}},\boldsymbol{\xi}} \quad & c^\top \mathbf{p}^{\mathrm{g}} + M e^\top \boldsymbol{\xi} && && \text{(11a)}\\
\text{s.t.} \quad & \Phi A_g \mathbf{p}^{\mathrm{g}} - \mathbf{p}^{\mathrm{f}} = \Phi A_d p^{\mathrm{d}} && [\boldsymbol{\pi}] && \text{(11b)}\\
& e^\top \mathbf{p}^{\mathrm{g}} = e^\top p^{\mathrm{d}} && [\boldsymbol{\lambda}] && \text{(11c)}\\
& \underline{p} \le \mathbf{p}^{\mathrm{g}} \le \overline{p} && [\underline{\mathbf{z}}, \overline{\mathbf{z}}] && \text{(11d)}\\
& \underline{f} - \boldsymbol{\xi} \le \mathbf{p}^{\mathrm{f}} \le \overline{f} + \boldsymbol{\xi} && [\underline{\boldsymbol{\mu}}, \overline{\boldsymbol{\mu}}] && \text{(11e)}\\
& \boldsymbol{\xi} \ge 0 && [\mathbf{y}] && \text{(11f)}
\end{aligned}$$

parametrized by the vector of active power demands $p^{\mathrm{d}} \in \mathbb{R}^D$. Decision variables $\mathbf{p}^{\mathrm{g}} \in \mathbb{R}^G$, $\mathbf{p}^{\mathrm{f}} \in \mathbb{R}^E$ and $\boldsymbol{\xi} \in \mathbb{R}^E$ denote active power dispatch, active power flow and line overflows, respectively. The notation e is used to denote an appropriately-sized vector whose entries are all equal to 1. The objective (11a) minimizes active power generation costs and thermal violation penalties, where M is a large positive constant, set to 150,000 in the experiments. Constraint (11b) expresses power flows using the network's Power Transfer Distribution Factor (PTDF) matrix Φ and generator and load incidence matrices A_g and A_d. Constraint (11c) enforces global power balance. Constraints (11d) and (11e) enforce bounds on active power dispatch and power flows, respectively. Dual variables associated to each constraint are indicated between brackets. The dual of the economic dispatch reads

$$\begin{aligned}
\max_{\substack{\boldsymbol{\lambda},\boldsymbol{\pi},\underline{\mathbf{z}},\overline{\mathbf{z}},\\ \underline{\boldsymbol{\mu}},\overline{\boldsymbol{\mu}},\mathbf{y}}} \quad & \boldsymbol{\lambda} e^\top p^{\mathrm{d}} + (\Phi A_d d)^\top \boldsymbol{\pi} \\
& \quad + \underline{f}^\top \underline{\boldsymbol{\mu}} - \overline{f}^\top \overline{\boldsymbol{\mu}} + \underline{p}^\top \underline{\mathbf{z}} - \overline{p}^\top \overline{\mathbf{z}} && \text{(12a)}\\
\text{s.t.} \quad & \boldsymbol{\lambda} e + (\Phi A_g)^\top \boldsymbol{\pi} + \underline{\mathbf{z}} - \overline{\mathbf{z}} = c && \text{(12b)}\\
& -\boldsymbol{\pi} + \underline{\boldsymbol{\mu}} - \overline{\boldsymbol{\mu}} = 0 && \text{(12c)}\\
& \underline{\boldsymbol{\mu}} + \overline{\boldsymbol{\mu}} + \mathbf{y} = Me && \text{(12d)}\\
& \underline{\boldsymbol{\mu}}, \overline{\boldsymbol{\mu}}, \underline{\mathbf{z}}, \overline{\mathbf{z}}, \mathbf{y} \ge 0 && \text{(12e)}
\end{aligned}$$

To generate the training, validation, and testing samples, the PGLearn [32, Algorithm 1 and Table 1] method is used to sample p^{d} such that a wide range of total power demand is covered, including the challenging high-load regime. Note that unlike the PGLearn datasets, which include optimal solutions, as described in Section III-C the implementation in this paper does not collect solutions for training nor validation samples, allowing it to start training immediately.

To implement the CPU solver baseline as well as the fallback in Algorithm 1, a lazy constraint approach is used. Algorithm 2 describes the procedure, which is implemented in JuMP 1.28.0 [33] using the HiGHS 1.11.0 [34] solver. Using lazy constraints for the thermal limits drastically improves solve times; for 1354_pegase, Algorithm 2 is approximately 15x faster than an equivalent sparse phase-angle formulation, for 2869_pegase 45x faster, and for 9241_pegase 250x faster. Note that testing samples are solved individually offline to facilitate reporting results for different levels of ϵ, e.g. Figure 1. Furthermore, model building time is excluded from the solve time, and perfect sample-wise parallelism is emulated with 24 CPUs, i.e., the solve time for a batch of N samples each with solve times t_i is computed using the ideal makespan bound $\max(\frac{1}{24}\sum_i t_i, \max_i(t_i))$.

B. Feasibility Guarantees

In order to guarantee the feasibility of the primal predictions, the experiments implement the following inference procedure, based on the proportional response layer [10]:

1) Predict $\tilde{\mathbf{p}}^{\mathrm{g}} \in \{\mathbf{p}^{\mathrm{g}} \mid \underline{p} \le \mathbf{p}^{\mathrm{g}} \le \overline{p}\}$
2) Recover $\hat{\mathbf{p}}^{\mathrm{g}}$ by using the proportional response layer [10, Eq. 4] to project $\tilde{\mathbf{p}}^{\mathrm{g}}$ onto the hypersimplex $\{\mathbf{p}^{\mathrm{g}} \mid e^\top \mathbf{p}^{\mathrm{g}} = e^\top p^{\mathrm{d}},\ \underline{p} \le \mathbf{p}^{\mathrm{g}} \le \overline{p}\}$, i.e.

$$\hat{\mathbf{p}}^{\mathrm{g}} \leftarrow \begin{cases} (1-\eta^{\uparrow})\tilde{\mathbf{p}}^{\mathrm{g}} + \eta^{\uparrow}\overline{p} & \text{if } e^\top \tilde{\mathbf{p}}^{\mathrm{g}} < e^\top p^{\mathrm{d}} \\ (1-\eta^{\downarrow})\tilde{\mathbf{p}}^{\mathrm{g}} + \eta^{\downarrow}\underline{p} & \text{if } e^\top \tilde{\mathbf{p}}^{\mathrm{g}} \ge e^\top p^{\mathrm{d}} \end{cases}$$

Algorithm 2 Lazy PTDF solver

Input: query p^{d}
Output: optimal solution $(\mathbf{p}^{\mathrm{g}}, \mathbf{p}^{\mathrm{f}}, \boldsymbol{\xi})$

Initialize $\mathcal{C} \leftarrow \emptyset$
while true **do**
  Solve

$$\begin{aligned} \min_{\mathbf{p}^{\mathrm{g}},\mathbf{p}^{\mathrm{f}},\boldsymbol{\xi}} \quad & c^\top \mathbf{p}^{\mathrm{g}} + M e^\top \boldsymbol{\xi} \\ \text{s.t.} \quad & (\Phi A_g \mathbf{p}^{\mathrm{g}})_i - \mathbf{p}^{\mathrm{f}}_i = (\Phi A_d p^{\mathrm{d}})_i \quad \forall i \in \mathcal{C} \\ & \text{(11c)} - \text{(11f)} \end{aligned}$$

  Compute $\tilde{\mathbf{p}}^{\mathrm{f}} \leftarrow \Phi A_g \mathbf{p}^{\mathrm{g}} - \Phi A_d p^{\mathrm{d}}$
  Compute $\tilde{\boldsymbol{\xi}} = \max\big(\max(0, \tilde{\mathbf{p}}^{\mathrm{f}} - \overline{f}), \max(0, \underline{f} - \tilde{\mathbf{p}}^{\mathrm{f}})\big)$
  for each branch i in $1{:}E$ **do**
    if $\tilde{\boldsymbol{\xi}}_i > 0$ **and** $i \notin \mathcal{C}$ **then**
      $\mathcal{C} \leftarrow \mathcal{C} \cup \{i\}$ {add constraint i}
    end if
  end for
  if no constraints were added **then**
    return $(\mathbf{p}^{\mathrm{g}}, \tilde{\mathbf{p}}^{\mathrm{f}}, \tilde{\boldsymbol{\xi}})$
  end if
end while

where $\eta^{\uparrow} := \dfrac{e^\top p^{\mathrm{d}} - e^\top \tilde{\mathbf{p}}^{\mathrm{g}}}{e^\top \overline{p} - e^\top \tilde{\mathbf{p}}^{\mathrm{g}}}$ and $\eta^{\downarrow} := \dfrac{e^\top \tilde{\mathbf{p}}^{\mathrm{g}} - e^\top p^{\mathrm{d}}}{e^\top \tilde{\mathbf{p}}^{\mathrm{g}} - e^\top \underline{p}}$

3) Recover $\hat{\mathbf{p}}^{\mathrm{f}} = \Phi A_g \hat{\mathbf{p}}^{\mathrm{g}} - \Phi A_d p^{\mathrm{d}}$.
4) Recover $\hat{\boldsymbol{\xi}} = \max\big(\max(0, \hat{\mathbf{p}}^{\mathrm{f}} - \overline{f}), \max(0, \underline{f} - \hat{\mathbf{p}}^{\mathrm{f}})\big)$.

To guarantee dual-feasiblility, the paper adopts Smoothed Self-Supervised Learning (S3L) [35], which for the economic dispatch reads:

1) Predict $\hat{\boldsymbol{\lambda}} \in \mathbb{R}$ and $\hat{\boldsymbol{\pi}} \in [-M, M]^E$
2) Recover $\hat{\underline{\boldsymbol{\mu}}} = \frac{\mu}{\overline{f} - \underline{f}} + \frac{\hat{\boldsymbol{\pi}}}{2} + \sqrt{(\frac{\mu}{\overline{f} - \underline{f}})^2 + (\frac{\hat{\boldsymbol{\pi}}}{2})^2}$
3) Recover $\hat{\overline{\boldsymbol{\mu}}} = \frac{\mu}{\overline{f} - \underline{f}} - \frac{\hat{\boldsymbol{\pi}}}{2} + \sqrt{(\frac{\mu}{\overline{f} - \underline{f}})^2 + (\frac{\hat{\boldsymbol{\pi}}}{2})^2}$
4) Compute $z = c - \hat{\boldsymbol{\lambda}} e - (\Phi A_g)^\top \hat{\boldsymbol{\pi}}$
5) Recover $\hat{\underline{\mathbf{z}}} = \frac{\mu}{\overline{\mathbf{p}^{\mathrm{g}}} - \underline{\mathbf{p}^{\mathrm{g}}}} + \frac{z}{2} + \sqrt{(\frac{\mu}{\overline{\mathbf{p}^{\mathrm{g}}} - \underline{\mathbf{p}^{\mathrm{g}}}})^2 + (\frac{z}{2})^2}$
6) Recover $\hat{\overline{\mathbf{z}}} = \frac{\mu}{\overline{\mathbf{p}^{\mathrm{g}}} - \underline{\mathbf{p}^{\mathrm{g}}}} - \frac{z}{2} + \sqrt{(\frac{\mu}{\overline{\mathbf{p}^{\mathrm{g}}} - \underline{\mathbf{p}^{\mathrm{g}}}})^2 + (\frac{z}{2})^2}$

Note that this procedure is only used during training. During inference, Dual Lagrangian Learning (DLL) [25] is used since it guarantees optimal completion given fixed $\hat{\boldsymbol{\lambda}}$ and $\hat{\boldsymbol{\pi}}$:

1) Predict $\hat{\boldsymbol{\lambda}} \in \mathbb{R}$ and $\hat{\boldsymbol{\pi}} \in [-M, M]^E$
2) Recover $\hat{\underline{\boldsymbol{\mu}}} = \max(0, \hat{\boldsymbol{\pi}})$.
3) Recover $\hat{\overline{\boldsymbol{\mu}}} = \max(0, -\hat{\boldsymbol{\pi}})$.
4) Recover $\hat{\underline{\mathbf{z}}} = \max(0, c - \hat{\boldsymbol{\lambda}} e - (\Phi A_g)^\top \hat{\boldsymbol{\pi}})$.
5) Recover $\hat{\overline{\mathbf{z}}} = \max(0, (\Phi A_g)^\top \hat{\boldsymbol{\pi}} + \hat{\boldsymbol{\lambda}} e - c)$.

Note that both the primal and dual recovery procedures require bounded predictions as part of Step 1; the experiments use the "double-softplus" function to enforce the element-wise bound constraints $l \leq x \leq u$ on the neural network predictions: $\tilde{x} := \ln(1 + e^{x-l}) - \ln(1 + e^{x-u}) + l$.

Finally, note that although the hybrid solver framework is applicable for any architecture which produces primal-dual feasible predictions, its performance depends on the inference time of the models, including any repair steps. The procedures described above are chosen due to their computational efficiency, enabling timely training and fast inference.

C. Results

The experiments compare the hybrid solver to the parallelized classical solver described in Section IV-A. For each case, two types of hybrid solvers are evaluated, differing only in the loss function used to train the underlying proxy models: Γ^0 refers to the standard ERM-based loss (7) and $\Gamma^{1\%}$ refers to the "ReLU" loss (10) with a target optimality threshold of $\epsilon{=}1\%$. All other hyperparameters are kept constant: proxies are trained using the Adam optimizer with batch size 1024 and learning rate $10^{-3}{\rightarrow}10^{-5}$, scheduled using the normalized mean duality gap on the validation set. Specifically, the learning rate is multiplied by 0.95 when the gap does not decrease by at least 0.01% for 50 consecutive epochs. The primal and dual proxies each use 4 hidden layers with dimension 256, learned batch-normalization, and softplus activations. Training is run for 5000 epochs with a (re-sampled, as discussed in Section III-C) dataset size of 20,480. Validation is run every epoch using the same 10,240 (unlabeled) samples, and checkpoints are saved based on the best mean normalized duality gap on the validation set. Results are presented on a batch of 240,000 unseen testing samples. This batch size is chosen to mimic performing a 1-day hourly-granularity simulation with 10,000 scenarios.

The training and evaluation is run on an NVIDIA H200 GPU, implemented using PyTorch 2.8.0 [36]. GPU inference uses `torch.compile` including the proxy inference, repair, and objective value calculation. This straightforward inference setup is used to keep the implementation simple; further speedup can likely be achieved by improving this aspect. Furthermore, time spent on data movement (inputs to GPU and outputs to CPU) is *included* in the inference time measurements.

Table II includes the column $T^{\text{Opt}}_{240\text{k}}$, reporting the solve time of the perfectly-parallelized classical solver for the test set. The column $T^{\text{ML}}_{240\text{k}}$ reports the inference time of the proxy models; across all cases, the ML models return predictions for all 240,000 samples in milliseconds, far exceeding the throughput of the parallelized classical solver. The column $T^{\text{ML}}_{\text{Train}}$ shows the training time for the ML models; about 30 minutes for 1354_pegase and 2869_pegase and just over an hour for 9241_pegase. Finally, the $|\alpha|{+}|\beta|$ column reports the total number of trainable parameters across both the primal and dual models; the largest case uses under 7.5M parameters.

The "Nx ϵ%" metric is used to analyze the performance of the proposed hybrid solver; it captures at what percent optimality tolerance ϵ% the hybrid solver achieves an Nx speedup. In other words, Pareto-optimal hybrid solvers achieve the largest Nx speedups at the lowest ϵ% tolerances. Note that, in contrast to NNV-based approaches which only provide a global worst-case bound, the hybrid solver allows practitioners to choose the target optimality threshold ϵ%, then evaluate the speedup. Table I includes such evaluations for $\epsilon{=}\{0.5\%, 1\%, 2\%\}$, as well as "inverse" results showing at what ϵ% level an $N{=}\{100, 500, 1000\}$-times speedup is achieved. Furthermore, Figure 1 shows N as a function of ϵ.

TABLE I
AGGREGATED PERFORMANCE RESULTS ACROSS DIFFERENT WEIGHT INITIALIZATIONS

Case	Loss	100x ϵ%	500x ϵ%	1000x ϵ%	Nx 0.5%	Nx 1%	Nx 2%
1354_pegase	Γ^0	**0.43**$\pm$**0.01**	0.96$\pm$0.17	1.36$\pm$0.20	**168.17**$\pm$**35.60**	588.97$\pm$199.74	1646.03$\pm$199.75
1354_pegase	$\Gamma^{1\%}$	0.65$\pm$0.07	**0.75**$\pm$**0.08**	**0.80**$\pm$**0.09**	1.90$\pm$0.54	**1782.84**$\pm$**297.92**	**2163.81**$\pm$**2.29**
2869_pegase	Γ^0	**0.47**$\pm$**0.03**	1.17$\pm$0.05	1.49$\pm$0.05	**109.01**$\pm$**6.99**	332.89$\pm$26.66	1672.71$\pm$119.89
2869_pegase	$\Gamma^{1\%}$	0.56$\pm$0.04	**1.10**$\pm$**0.03**	**1.36**$\pm$**0.03**	50.27$\pm$34.33	**388.00**$\pm$**32.21**	**1798.52**$\pm$**19.63**
9241_pegase	Γ^0	**0.53**$\pm$**0.03**	0.87$\pm$0.02	2.88$\pm$0.07	**82.33**$\pm$**18.62**	707.22$\pm$57.06	958.89$\pm$18.36
9241_pegase	$\Gamma^{1\%}$	0.57$\pm$0.03	**0.85**$\pm$**0.04**	**1.18**$\pm$**0.04**	64.04$\pm$13.71	**925.23**$\pm$**39.92**	**1113.27**$\pm$**1.30**

TABLE II
CASE SIZES, TIMING STATISTICS, AND PARAMETER COUNTS

Case name	$\lvert p^{\mathrm{d}}\rvert$	$\lvert \mathbf{p}^{\mathrm{g}}\rvert$	$\lvert \mathbf{p}^{\mathrm{f}}\rvert$	$T^{\mathrm{Opt}}_{240\mathrm{k}}$	$T^{\mathrm{ML}}_{240\mathrm{k}}$	$T^{\mathrm{ML}}_{\mathrm{Train}}$	$\lvert\alpha\rvert+\lvert\beta\rvert$
1354_pegase	673	260	1991	90.8s	0.03s	28min	1.3M
2869_pegase	1491	510	4582	199.7s	0.09s	32min	2.5M
9241_pegase	4895	1445	16049	711.2s	0.64s	70min	7.4M

Table I shows the hybrid solver performance across cases and when using different training losses for the underlying proxies. Notably, all configurations achieve substantial speedups at practical optimality tolerance levels, all achieving 1000x speedup below 3% tolerance. When using the $\Gamma^{1\%}$ loss configurations, 100x is achieved below 0.7% across all cases and 1000x is achieved below 1.4% for all cases. Furthermore, the difference between the Γ^0 and $\Gamma^{1\%}$ rows highlights the tradeoff associated with targeting an optimality tolerance; speedup for tolerance levels below the target is traded for speedup at and above the target. Notably, $\Gamma^{1\%}$ consistently outperforms Γ^0 at the target tolerance (corresponding to the Nx 1% column), achieving speedups over 1500x for 1354_pegase, over 350x for 2869_pegase, and over 900x for 9241_pegase. With a higher 2% threshold, speedup is over 1000x for all cases; above 2000x for 1354_pegase.

Figure 1 further explores the impact of using a non-zero target threshold, comparing the speedup-optimality tradeoff curves when using different underlying proxy models, one trained using Γ^0, labeled ERM, and the other using $\Gamma^{1\%}$, labeled ReLU. The figure shows that, at the 1% target tolerance and above, using the $\Gamma^{1\%}$ models results in larger speedups. At optimality tolerances below the target, Γ^0 begins to outperform $\Gamma^{1\%}$. While $\Gamma^{1\%}$ only achieves 10x speedup for thresholds below 0.6%, Γ^0 achieves over 100x speedup while guaranteeing at most 0.5% sub-optimality.

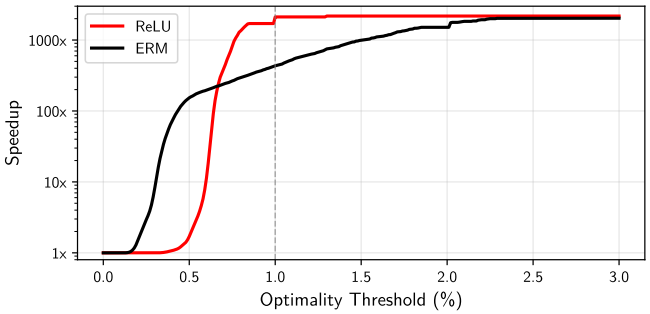

Fig. 1. Speedup of hybrid solver on 1354_pegase as a function of the optimality threshold ϵ.

V. CONCLUSION

This paper has proposed a novel hybrid framework that accelerates large-scale computations in power systems operations and planning by integrating data-driven machine learning proxies and classical optimization solvers in a principled way. The proposed hybrid solver offers substantial speedups *and* worst-case performance guarantees, thus synergizing several recent developments in machine learning for power systems. This approach further closes the gap between theoretical advances and the tight requirements of practical deployments, enabling trustworthy proxy-based market clearing with performance guarantees and real-time risk analysis for large-scale power systems.

Unlike neural network verification-based approaches, the proposed methodology does not impose strong restrictions on activation functions, does not require any post-training analysis, and provides *adaptable* worst-case optimality guarantees. The paper has conducted extensive computational experiments on large-scale power grids, corresponding to economic dispatch problems with over 10^5 decision variables and constraints, and over 10^4 parameters. Numerical results demonstrate the scalability of the approach, achieving speedups on the order of 1000x over optimized classical solvers.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 2112533 and Grant No. DGE-2039655, and partially funded by Los Alamos National Laboratory's Directed Research and Development project, "Artificial Intelligence for Mission (ArtIMis)." Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the sponsors.

AI was not used in neither the research nor writing of this paper.

REFERENCES

[1] Y. Gu and J. McCalley, "Market-based transmission expansion planning," in *2011 IEEE/PES Power Systems Conference and Exposition*, IEEE, 2011.

[2] O. Stover, P. Karve, and S. Mahadevan, "Reliability and risk metrics to assess operational adequacy and flexibility of power grids," *Reliability Engineering & System Safety*, vol. 231, p. 109018, 2023.

[3] W. Chen, M. Tanneau, and P. Van Hentenryck, "Real-time risk analysis with optimization proxies," *Electric Power Systems Research*, vol. 235, p. 110822, 2024.

[4] B. Knueven *et al.*, "Stochastic look-ahead commitment: A case study in MISO," in *2023 IEEE Power & Energy Society General Meeting (PESGM)*, IEEE, 2023.

[5] P. Brahmbhatt *et al.*, "Benders decomposition using graph modeling and multi-parametric programming," *arXiv:2508.01100*, 2025.

[6] H. Khaloie *et al.*, "Review of machine learning techniques for optimal power flow," *SSRN 4681955*, 2024.

[7] P. Van Hentenryck, "Trustworthy optimization learning: a brief overview," in *De Gruyter Proceedings in Mathematics*, p. 121, 2025.

[8] B. Amos and J. Z. Kolter, "Optnet: Differentiable optimization as a layer in neural networks," in *International conference on machine learning*, PMLR, 2017.

[9] M. Kim and H. Kim, "Projection-aware deep neural network for dc optimal power flow without constraint violations," in *2022 IEEE SmartGridComm*, pp. 116–121, 2022.

[10] W. Chen *et al.*, "End-to-end feasible optimization proxies for large-scale economic dispatch," *IEEE Transactions on Power Systems*, 2023.

[11] J. Tordesillas, J. P. How, and M. Hutter, "Rayen: Imposition of hard convex constraints on neural networks," *arXiv:2307.08336*, 2023.

[12] Y. Min and N. Azizan, "Hardnet: Hard-constrained neural networks with universal approximation guarantees," *arXiv:2410.10807*, 2024.

[13] G. E. Constante-Flores, H. Chen, and C. Li, "Enforcing hard linear constraints in deep learning models with decision rules," *NeurIPS*, 2025.

[14] W. Chen, H. Zhao, M. Tanneau, and P. Van Hentenryck, "Compact optimality verification for optimization proxies," in *ICML*, 2024.

[15] C. Liu *et al.*, "Algorithms for verifying deep neural networks," *Foundations and Trends® in Optimization*, 2021.

[16] R. Nellikkath and S. Chatzivasileiadis, "Physics-informed neural networks for minimising worst-case violations in dc optimal power flow," in *2021 IEEE SmartGridComm*, 2021.

[17] R. Nellikkath, M. Tanneau, P. Van Hentenryck, and S. Chatzivasileiadis, "Scalable exact verification of optimization proxies for large-scale optimal power flow," *arXiv:2405.06109*, 2024.

[18] S. Wang *et al.*, "Beta-crown: Efficient bound propagation with per-neuron split constraints for neural network robustness verification," *Advances in neural information processing systems*, vol. 34, pp. 29909–29921, 2021.

[19] K. Xu *et al.*, "Explainable warm-start point learning for ac optimal power flow using a novel hybrid stacked ensemble method," *Sustainability*, 2025.

[20] W. Dong *et al.*, "Smart-pgsim: Using neural network to accelerate ac-opf power grid simulation," in *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, IEEE, 2020.

[21] S. Park, W. Chen, T. W. Mak, and P. Van Hentenryck, "Compact optimization learning for ac optimal power flow," *IEEE Transactions on Power Systems*, 2023.

[22] F. Diehl, "Warm-starting ac optimal power flow with graph neural networks," in *NeurIPS*, 2019.

[23] A. Ben-Tal and A. Nemirovski, *Lectures on modern convex optimization*. SIAM, 2001.

[24] P. D. Grontas *et al.*, "Pinet: Optimizing hard-constrained neural networks with orthogonal projection layers," *arXiv:2508.10480*, 2025.

[25] M. Tanneau and P. V. Hentenryck, "Dual lagrangian learning for conic optimization," in *NeurIPS*, 2024.

[26] S. Park and P. Van Hentenryck, "Self-supervised primal-dual learning for constrained optimization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, pp. 4052–4060, 2023.

[27] B. Li, L. Yang, Y. Chen, S. Wang, Q. Chen, H. Mao, Y. Ma, A. Wang, T. Ding, J. Tang, *et al.*, "Pdhg-unrolled learning-to-optimize method for large-scale linear programming," *arXiv preprint arXiv:2406.01908*, 2024.

[28] A. Iftakher, R. Golder, B. N. Roy, and M. F. Hasan, "Physics-informed neural networks with hard nonlinear equality and inequality constraints," *Computers & Chemical Engineering*, p. 109418, 2025.

[29] S. Arvind, R. Pomaje, and R. V. Bhat, "Karush-kuhn-tucker condition-trained neural networks (kkt nets)," *arXiv preprint arXiv:2410.15973*, 2024.

[30] O. Stover, P. Karve, S. Mahadevan, W. Chen, H. Zhao, M. Tanneau, and P. Van Hentenryck, "Just-in-time learning for operational risk assessment in power grids," *arXiv:2209.12762*, 2022.

[31] S. Babaeinejadsarookolaee *et al.*, "The Power Grid Library for benchmarking AC optimal power flow algorithms," *arXiv:1908.02788*, 2019.

[32] M. Klamkin, M. Tanneau, and P. V. Hentenryck, "PGLearn - an open-source learning toolkit for optimal power flow," 2025.

[33] M. Lubin *et al.*, "JuMP 1.0: Recent improvements to a modeling language for mathematical optimization," *Mathematical Programming Computation*, 2023.

[34] Q. Huangfu and J. A. J. Hall, "Parallelizing the dual revised simplex method," *Mathematical Programming Computation*, 2018.

[35] M. Klamkin, M. Tanneau, and P. V. Hentenryck, "Dual interior point optimization learning," in *IISE Annual Conference*, 2025.

[36] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *NeurIPS*, 2019.

I. Review #37A

A. Paper summary

This paper designs a primal-dual based optimization proxy to predict solutions to large scale economic dispatch problems. Two neural networks are trained in parallel, and the optimizer's goal is to drive down the optimality gap. The approach depends on a feasibility projection on the back side of the NNs which predict primal and dual solutions.

B. Comments for authors

This paper presents a nice innovation: using an optimizer to drive down the optimality gap associated with primal and dual NN predictions. The entire approach, however, hinges on an assumption that isn't really discussed in a motivating way: unsupervised learning is more efficient than supervised learning. A similar paper could be written where parallelized, fast LP solvers generate primal and dual solutions, and then NNs are trained, in parallel, to predict these solutions. Why is this simpler approach not benchmarked against? Overall, the performance of their method is excellent, and the test cases are relevant. This paper would be improved with better framing and softer, more specific claims. Furthermore, the proposed economic dispatch formulation uses penalized, rather than hard, line flow limits. This makes the primal problem trivial to project feasible, which the authors' approach hinges on.

- The sentence: "This paper aims at striking a balance between classical solvers and optimization proxies in order to enable trustworthy deployments with interpretable speed-optimality tradeoffs based on a user-defined optimality threshold." is very unclear for a sentence in an abstract.
- Generally, the paper uses a lot of italicized text, which isn't necessary in scientific papers.
- Also, the paper makes many strong claims about being the "first" to propose an innovation. This feels like salesmanship. Indeed, if a paper isn't the first to propose its method, then the paper doesn't need to be written: research novelty should be self-certifying.
- "Thus, relying on neural network verification tools as a screening procedure for proxy adoption, besides being computationally expensive, leads to overly pessimistic conclusions." This claim should be qualified. Verification is a tool which can be applied flexibly, e.g., to determine where the your method will perform poorly (i.e., where the backup solver will need to be consistently used).
- "Optimization proxies are typically relatively small (under 10M parameters)," This is a fairly subjective claim. Scaling laws show: more compute, more data, bigger model -¿ better performance. Please add a citation.
- Why is bound normalization needed? The technical description of the approach is clear, but the motivation is lacking.
- "However, unlike the supervised learning setting where training data instances must be solved apriori – representing a considerable computational burden – in the self-supervised setting, generating new samples requires minimal computation." Why is this true? In the self-supervised setting, the GPU has to implicitly solve economic dispatch; there is no free lunch. Rather than using an efficient simplex solver, or the lazy constraint solver, you make the GPU solve the problem, buried under a training problem. If Alg. 2 is really so fast, if could be used to generate parallel training data.
- "Importantly, this enables immediate and trustworthy (re-)training within larger workflows." This is unclear. Why is the approach trustworthy?
- "Algorithm 2 is approximately 15x faster than an equivalent sparse phase-angle formulation..." This is a very nice result in itself!
- It is very disappointing to see that (11) is essentially an unconstrained problem, in the sense that line flow limits are penalized, and the remainder of the limits are trivial. This limits the applicability of the proposed innovation.
- Test results appear excellent.

C. Summarize your recommendation in one to two sentences

The paper proposes a nice technical innovation with self-certifying primal-dual learning, but the claims of the paper are slightly oversold, and the specific motivation and framing is lacking. A rewrite is suggested.

— Anonymous reviewer

II. Review #37B

A. Paper summary

The paper introduces a principled hybrid solver that uses learned primal–dual proxies together with a fallback classical solver, allowing users to prescribe an optimality tolerance and obtain worst-case guarantees without expensive offline verification or restrictions on neural architectures. The scheme can often avoid invoking the solver entirely, yielding major runtime speedups with up to three orders of magnitude.

B. Comments for authors

Strengths:

1) Well written paper combining state of the art learning to optimize (L2O) methodology with well established duality theory frok constrained optimization.
2) Self-supervised L2O algorithm with user-prescribed tolerance and no restrictive neural architecture or expensive post-processing or verification.
3) Clean integration with existing solvers and use of lazy constraints for efficiency.
4) Case study implemented with a high degree of technical sophistication and with the use of state of the art tools, e.g. JuMP, PGLearn, and HiGHS solver.
5) The case study is addressing real operational needs such as real-time market clearing, risk analysis.
6) Strong empirical scalability on industry-scale test cases with guaranteed feasibility and 1000x speedups.

Weaknesses:

1) Open-source code would be highly beneficial for the adoption, reproducibility, and transparency of the obtained results. I hope authors will release the code after the publication.

C. Summarize your recommendation in one to two sentences

Strong accept: I like this paper a lot. I can highly recommend it for the PowerUP conference.

— Jan Drgona

III. Review #37C

A. Paper summary

This paper solves the reliability problem of learned optimization proxies by introducing a hybrid primal-dual solver for the economic dispatch problem. The key idea is to train two neural networks so that the dual network provides an upper bound on the primal value. If the predicted solution meets a user-defined optimality tolerance, it is accepted; otherwise, a fallback classical solver is invoked. The authors test on large-scale benchmarks (1354, 2869, 9241-bus European grids) and report dramatic speedups: e.g., on the 9241-bus case hybrid solver achieves $\approx$925$\times$ faster solution times than a parallel simplex solver at a 1% gap, with worst-case gap below that threshold. The framework is novel, the methodology is sound, and the results suggest a promising way to marry machine learning with rigorous optimization guarantees.

B. Comments for authors

Strengths Novelty and Practicality: The concept of self-certifying optimization proxies is a significant contribution. By marrying neural predictions with duality bounds, the authors address the reliability gap in learned solvers, while the user-defined optimality tolerance provides much-needed flexibility for real-world operations. Methodological Rigor and Impact: The hybrid solver is theoretically grounded in LP duality, and the joint training of primal and dual models on the duality gap is a principled approach. Achieving 1,000 times speedups on large-scale systems (up to 9,241 buses) demonstrates high practical value for both machine learning researchers and power systems operators.

Weaknesses/Points for Improvement Transparency on Overhead and Fallbacks: The paper should provide more detail on the offline training cost (time/epochs) and, crucially, the fallback frequency. Reporting how often the classical solver is triggered at specific tolerances (e.g., 1%) is essential for assessing the true reliability-speed trade-off. Generalization and Robustness: While the results on the Pegase systems are impressive, the study would be strengthened by testing out-of-distribution performance via perturbed topologies or extreme loads. Additionally, the authors should discuss the scalability of this framework to non-linear AC optimal power flow or convex relaxations.

C. Summarize your recommendation in one to two sentences

I recommend acceptance. This paper addresses a significant problem with a well-justified solution. The methodology is novel, and the empirical evidence is strong.

— Anonymous reviewer

IV. Brief response to reviewers (optional)

V. Brief summary of changes made to the final paper