End-to-end electricity system forecasting via approximate message passing

Anthony Degleris
Department of Electrical Engineering
Stanford University
Stanford, CA 94305, USA
degleris@stanford.edu

Akshay Sreekumar
Department of Electrical Engineering
Stanford University
Stanford, CA 94305, USA
akshay81@stanford.edu

Kamran Tehranchi
Department of Civil & Environmental Engineering
Stanford University
Stanford, CA 94305, USA
ktehranchi@stanford.edu

Ram Rajagopal
Department of Electrical Engineering
Stanford University
Stanford, CA 94305, USA
ramr@stanford.edu

Abstract

Accurately forecasting electricity system outcomes, such as power flow schedules and nodal prices, is crucial to integrating renewable generation and battery storage technologies into the grid. These outcomes are the result of a large-scale constrained optimization problem and must satisfy various physical constraints. We propose combining a neural network with a differentiable approximate *proximal message passing* solver to produce an end-to-end model for grid forecasting. Our method does not require expensive implicit differentiation steps and generalizes to new system topologies unobserved during training. Initial experiments on a Western U.S. grid dataset suggest our method can improve upon traditional architectures in both constraint satisfaction and generalization to unseen problem data.

1 Introduction

Reliably and efficiently integrating renewable energy sources into the power grid is key to decarbonizing the electricity sector. Unfortunately, many renewable sources, such as wind and solar, are inherently intermittent and uncertain, creating a need for short-term forecasting models that can accurately predict power schedules and prices. These forecasts can be used, for example, to inform battery storage operations, generator commitment decisions, and flexible load scheduling, all of which must satisfy intertemporal constraints and therefore often need to be planned in advance. Improving short-term forecasts helps better schedule these decisions around renewable production, reducing curtailments and, hence, reliance on fossil fuels. Consequently, improved short-term electricity forecasting is essential to electricity system decarbonization for climate change mitigation.

Both power schedules (e.g., generator or line flows) and prices are the outcomes of a constrained optimization problem. This problem, often a variant of *DC-linearized optimal power flow* (DC OPF) (Frank & Rebennack, 2016), is solved by a system operator to minimize energy costs and set market clearing prices. Accordingly, both power schedules and prices satisfy a set of physical and economic constraints that are either partially or fully known in advance. Vanilla deep neural networks (DNNs), although highly expressive, often struggle

to make predictions that consistently satisfy constraints or take advantage of known structure in the optimization problem. This motivates the need for deep learning models that can explicitly handle constraints and leverage optimization problem structure.

Contribution. We propose a hybrid approach that integrates a differentiable, GPU-accelerated *proximal message passing* solver for the DC-OPF problem (Degleris et al., 2024) as a layer within a neural network. Our approach leverages the strengths of deep learning to generate accurate power and price predictions while using *approximate* unrolled optimization steps to encode the physical structure of the problem and encourage physically feasible forecasts. In particular, our method does not require fully observed problem data during inference, enforcing only known constraints exactly, and approximating unknown ones. Moreover, the optimization steps are modular and can be incorporated into arbitrary deep learning-based forecast models. We demonstrate our method with a forecasting task on a realistic Western U.S. electricity system dataset; initial results suggest our approach improves upon vanilla deep learning architectures in both constraint satisfaction and prediction accuracy.

1.1 Related Work

Machine learning has been widely applied to forecasting tasks in power systems, such as predicting load, renewable generation, and market prices (Khodayar et al., 2017; Morais et al., 2023). However, comparatively less attention has been given to end-to-end approaches that explicitly incorporate physical structure into forecasting power flows and locational marginal prices. Our work is motivated by related ideas in *optimization-aware* machine learning, a term we use to refer to methods for incorporating the structure of optimization problems into machine learning models. In particular, we draw from two related areas of research.

The first area uses facts about the optimization problem to design *completion-correction layers* that complete missing variables determined by equality constraints and correct the solution to satisfy inequality constraints (Chen et al., 2021; Donti et al., 2021; Pan et al., 2023; Klamkin et al., 2024; Rosemberg et al., 2024). These methods often scale well to large problems and can generally be trained in a self-supervised manner, completely eliminating the need to ever explicitly solve a constrained optimization problem (Donti et al., 2021; Pan et al., 2023). However, completion-correction techniques are almost always used to predict problem solutions when the problem inputs are fully known and, generally, are not straightforward to extend to uncertain problem inputs.

The second area explicitly integrates parametrized constrained optimization problems within a neural network using *optimization layers* (Chen et al., 2021; Amos & Kolter, 2017; Agrawal et al., 2019). These layers are evaluated exactly using a standard constrained optimization solver (e.g., an interior point method) and differentiated by applying *implicit differentiation* to the problem's optimality conditions. Although this approach is highly flexible and can

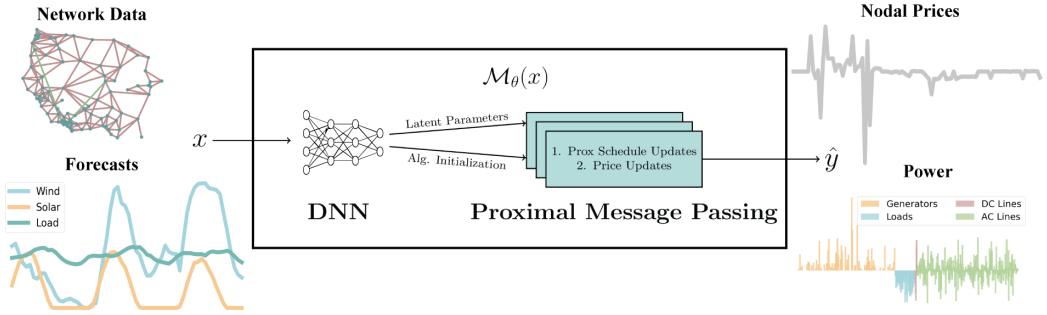


Figure 1: Diagram of proposed architecture. System information and forecasts are fed through a deep neural network followed by several steps of proximal message passing to predict power schedules and prices.

exactly enforce problem constraints, the computational expense of solving a linear system during implicit differentiation generally prohibits this approach from scaling to large datasets and high-dimensional problem instances (Donti et al., 2021).

Instead of using exact optimization layers, our approach explicitly unrolls iterations of a *proximal message passing solver* (Degleris et al., 2024) specially designed for electricity network optimization problems. Unlike other unrolled solvers for constrained optimization that require expensive factorization steps (Schubiger et al., 2020; Blondel et al., 2021; Bishop et al., 2024), proximal message passing takes advantage of the problem's network structure and is entirely factorization-free. Each iteration of the algorithm can be viewed as a correction-completion step, exactly enforcing known constraints and approximately enforcing unknown ones.

2 Model

In this section, we first describe the convex optimization problem used to generate power and price schedules in an electricity market. Then, we describe the resulting forecasting problem considered in this paper.

Electricity market model. We use a general model for *DC optimal power flow* (DC OPF) problems from Kraning (2014) that considers networks with N nodes connected to electrical devices (loads, generators, transmission lines, etc.) via T terminals. The topology of the network is given by the network incidence matrix $A \in \{0,1\}^{N \times T}$, which has entries $A_{nt} = 1$ if terminal t connects to node n and $A_{nt} = 0$ otherwise. The cost of operation is given by the convex function $c_z(\cdot,\cdot) : \mathbf{R}^T \times \mathbf{R}^T \to \mathbf{R} \cup \{\infty\}$, which is parameterized by the problem data $z \in \mathbf{R}^R$. The function c_z is extended real-valued, meaning it can take on infinite values in order to model device-specific constraints, e.g., a transmission line's flow limit or a generator's maximum production capacity. Given problem data z, the grid operator seeks to minimize the device costs subject to power balance and phase consistency constraints,

$$\begin{array}{lll} \text{minimize} & c_z(p,v) & \\ \text{subject to} & Ap = 0, \quad A^T v = u & (\lambda, \omega) \end{array} \tag{1}$$

where the variables are the terminal powers $p \in \mathbf{R}^T$ and phase angles $v \in \mathbf{R}^T$, and the nodal phase angles $u \in \mathbf{R}^N$. The power balance and phase consistency constraints are analogous to Kirchhoff's current and voltage laws, respectively, and impose physical limits on power flows across the network. The dual variables (i.e., Lagrange multipliers) of the power balance and phase consistency constraints are $\lambda \in \mathbf{R}^N$ and $\omega \in \mathbf{R}^T$; the vector λ also defines the *locational marginal prices* (LMPs) of the network at optimality. Our framework is highly general and can model many common devices such as loads, generators, AC transmission lines, and DC transmission lines. See Appendix A for additional details and refer to Boyd & Vandenberghe (2004) for background on optimization in general.

Electricity system forecasting problem. Assume, without loss of generality, that Problem (1) has a unique solution for every $z \in \mathbf{R}^R$. Then let $p_z^*, v_z^*, u_z^*, \lambda_z^*, \omega_z^*$ be the optimal primal and dual variables of Problem (1) as a function of the problem data, and define $y_z^* = (p_z^*, v_z^*, \lambda_z^*, \omega_z^*)$ as the problem's *solution map*. We assume z is drawn from some unknown random distribution with density $p(z)$. The forecaster does not observe z directly, but instead sees $x \in \mathbf{R}^K$ drawn from $p(x|z)$. The observation x may be, for example, a noisy, masked, or reduced version of the true state z; in general, the conditional distribution $p(x|z)$ may involve nonlinear transformations and complex sources of noise.

Our goal is to learn a model $\hat{y}_\theta = \mathcal{M}_\theta(x)$ (parameterized by θ) that predicts the solution y_z^* of Problem (1) as a function of the observations x, i.e.,

$$\mathcal{M}_\theta(x) \approx y_z^* \quad \text{for all} \quad z, x \sim p(z)p(x|z). \tag{2}$$

Notably, the model only sees the observation x, not the full problem data z, during inference; this diverges from the problems considered in Donti et al. (2021); Pan et al. (2023) and others. During training, we assume access to full samples $(x_1, z_1, y_1), \ldots, (x_S, z_1, y_S)$; this is

realistic since many system operators release full historical market data. We seek forecasting models that (i) are accurate, (ii) respect the power imbalance, phase consistency, and device constraints (implied in c_z), and (iii) generalize well to unseen problem data (for example, new transmission topologies).

2.1 Proximal Message Net

Kraning (2014) uses the *alternating direction method of multipliers (ADMM)* to derive an efficient algorithm for Problem (1) that alternates between two steps,

1. *Prox schedule step.* $(p^{(i+1)}, v^{(i+1)}) = \mathbf{prox}_{c_z,\rho}\left(\mathbf{resid}(p^{(i)}) - \lambda^{(i)}, \mathbf{avg}(v^{(i)}) - \omega^{(i)}\right),$
2. *Price update.* $\lambda^{(i+1)} = \lambda^{(i)} + \mathbf{avg}(p^{(i+1)}),$
   $\omega_n^{(i+1)} = \omega^{(i)} + \mathbf{resid}(v^{(i+1)}),$

where $\mathbf{avg}(x)$ and $\mathbf{resid}(x)$ are linear operators and $\mathbf{prox}_{f,\rho}$ is the prox operator of f,

$$\mathbf{prox}_{f,\rho}(b) = \mathbf{argmin}_u \left(f(u) + \frac{\rho}{2}\|u - b\|_2^2\right).$$

The prox operators for most common devices can be evaluated analytically and parallelized across devices of the same type. This algorithm, called *proximal message passing*, converges to an optimal solution of (1), does not require solving linear systems, and has an efficient GPU implementation available in PyTorch (Degleris et al., 2024). For more algorithm details, see Appendix B.

Proposed approach. We propose combining a traditional deep neural network (DNN) $\mathcal{D}_\theta$ with L iterations of proximal message passing. Specifically, we first use the DNN to predict the algorithm initialization, $y_\theta^{(0)}(x) = \left(p_\theta^{(0)}(x), v_\theta^{(0)}(x), \lambda_\theta^{(0)}(x), \omega_\theta^{(0)}(x)\right)$. Then, we use the DNN to estimate the device cost functions $c_z(\cdot,\cdot)$ in one of two ways:

1. **Predict the latent parameters directly.** We use a DNN $\hat{z}_\theta(x)$ to directly estimate z, then use $c_{\hat{z}_\theta(x)}$ in the proximal power schedule updates.
2. **Proxy cost functions.** We replace $c_z(\cdot,\cdot)$ with a proxy function, $\hat{c}_{\theta,x}(p,v) = \|p - p_\theta^{(0)}(x)\|_2^2 + \|v - v_\theta^{(0)}(x)\|_2^2$ to penalize schedules for deviating from their initialization. (In theory, $\hat{c}_{\theta,x}$ could be any function class parameterized by a neural network.)

Furthermore, suppose $c_z(p,v) = c_{z_1}^1(p_1, v_1) + c_{z_2}^2(p_2, v_2)$ where $z = (z_1, z_2)$ and z_2 is fully observed (i.e., z_2 is part of x). In this case, we apply the above estimates to c_{z_1} only and use the true c_{z_2} during proximal updates. Full observation is commonly the case for transmission system parameters, for example, which are often made available to grid participants.

Let $\mathcal{P}_L(c, y^{(0)})$ be L iterations of proximal message passing applied to initialization $y^{(0)}$ for cost function c. In summary, our proposed architectures are

$$\mathcal{M}_\theta^{\text{state}}(x) = \mathcal{P}_L\left(c_{\hat{z}_\theta(x)},\ y_\theta^{(0)}(x)\right), \qquad \mathcal{M}_\theta^{\text{proxy}}(x) = \mathcal{P}_L\left(\hat{c}_{\theta,x},\ y_\theta^{(0)}(x)\right),$$

where $\hat{z}_\theta(x)$ and $y_\theta^{(0)}(x)$ are the outputs of $\mathcal{D}_\theta(x)$. The function $\mathcal{D}_\theta(x)$ can be any differentiable model, i.e., the proposed approach could be incorporated into a state-of-the-art forecasting architecture. Each iteration of message passing has approximately the same computational cost as a single dense layer to train and evaluate. Therefore, when L is small (say, $L = 5$), our approach has similar computational requirements to a vanilla DNN.

3 Experiments

We study joint power and price prediction tasks on a synthetic 100-node Western U.S. electricity system from the PyPSA-USA dataset (Tehranchi et al., 2024). The dataset contains 5 years of hourly historical generation and load data from 2019 to 2023 (43,800 samples). We consider three versions of the data: (**Perfect**) the model inputs include the

Metric	Perfect			Noisy			Outage		
	Dense	State	Proxy	Dense	State	Proxy	Dense	State	Proxy
Loss	0.132	**0.122**	0.132	**0.124**	0.130	0.141	0.142	**0.126**	0.141
Price MSE	0.137	0.092	**0.076**	**0.081**	0.141	0.083	0.141	0.113	**0.079**
Power MSE	0.121	**0.042**	0.100	0.125	**0.089**	0.117	0.140	**0.045**	0.106
Power Imbal.	**0.009**	0.039	0.017	**0.007**	0.022	0.018	**0.009**	0.041	0.024
DCPF Infeas.	40.0	**15.1**	23.4	45.9	**26.8**	27.5	42.8	**17.3**	25.7
Gen. Viol.	0.026	**0.00**	0.161	**0.030**	0.354	0.193	0.024	**0.00**	0.293
Line Viol.	0.004	**0.00**	**0.00**	0.006	**0.00**	**0.00**	0.030	**0.00**	**0.00**

Table 1: Test loss, relative power and price error, and various constraint satisfaction metrics on all three datasets.

full problem data, i.e., $x = z$, (**Noisy**) the generation and load forecasts have added noise, and (**Outage**) a fraction of the transmission line capacities are randomly de-rated during each time interval (but are still observed).

We compare $L = 5$ iterations of the latent parameter (**State**) and proxy cost function (**Proxy**) models from Section 2.1 against a simple DNN baseline (**Dense**). All three models use a dense neural network backbone with 2 hidden layers and train on a weighted sum of prediction accuracy and constraint violations. The **State** model also employs a residual connection when predicting latent parameters $\hat{z}_\theta(x)$. For fairness, in the **Perfect** dataset, generation and load data is only an input to the neural network backbone and not explicitly fed into the message passing steps. All models use generation and load forecasts alongside all transmission line data as features. See Appendix C for full experiment details.

Results. We report each model's test set performance in Table 1, recording the test loss, price and power mean square error (normalized by the magnitude of the true price and power data), and the violation of several problem constraints. In particular, we record the power imbalance, the DC power flow infeasibility for AC lines (see Appendix A), and the violation of transmission line and generator capacity constraints. We find the **Proxy** and **State** models improve upon the **Dense** baseline in overall loss, power and price accuracy, line violations, and DC power flow infeasibility for both the *Perfect* and *Outage* datasets. The **Proxy** model does not perform as well in enforcing generator constraints or power imbalance as the **Dense** baseline, while the **State** model is worse than the **Dense** baseline in enforcing power balance but better in enforcing generator constraints for the *Perfect* and *Outage* datasets. The reduced performance against the baseline in these metrics is possibly because of the challenge inherent in enforcing multiple constraints. Overall, we find that both **State** and **Proxy** lead to improved price forecasts in the *Perfect* and *Outage* cases; see Figure 2 for a representative sample. For the *Noisy* dataset, the **Dense** baseline outperforms the **State** and **Proxy** models across several metrics, suggesting that these architectures are not as robust to injected noise.

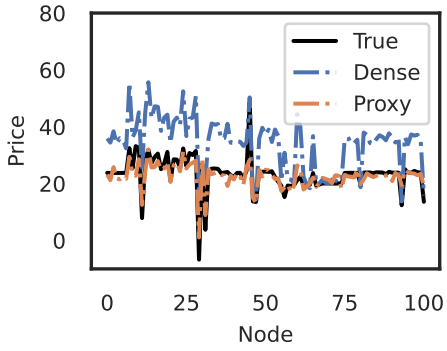


Figure 2: Representative price sample from **Dense** model and **Proxy** on the **Outage** dataset. **Dense** fails to generalize to new system topology leading to poor price forecasts.

4 Conclusion

We introduce a framework for combining an approximate constrained optimization solver within a neural network. Initial results indicate this method has the potential to improve upon standard deep learning architectures. Future work includes designing improved model architectures, e.g., replacing $\hat{c}_{\theta,x}$ with more complex functions.

References

Akshay Agrawal, Brandon Amos, Shane T Barratt, Stephen P Boyd, Steven Diamond, and J Z Kolter. Differentiable convex optimization layers. *Neural Information Processing Systems*, 32:9558–9570, October 2019.

Brandon Amos and J Z Kolter. OptNet: Differentiable optimization as a layer in neural networks. *International Conference on Machine Learning*, 70:136–145, March 2017.

Arun L Bishop, John Z Zhang, Swaminathan Gurumurthy, Kevin Tracy, and Zachary Manchester. ReLU-QP: A GPU-accelerated quadratic programming solver for model-predictive control. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 13285–13292. IEEE, May 2024.

Mathieu Blondel, Quentin Berthet, Marco Cuturi, Roy Frostig, Stephan Hoyer, Felipe Llinares-L'opez, Fabian Pedregosa, and Jean-Philippe Vert. Efficient and modular implicit differentiation. *Neural Information Processing Systems*, abs/2105.15183:5230–5242, May 2021.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, England, March 2004.

Bingqing Chen, Priya Donti, Kyri Baker, J Zico Kolter, and Mario Berges. Enforcing policy feasibility constraints through differentiable projection for energy optimization. *arXiv preprint arXiv*, May 2021.

Anthony Degleris, Abbas El Gamal, and Ram Rajagopal. GPU accelerated security constrained optimal power flow. *arXiv [math.OC]*, October 2024.

Priya L Donti, David Rolnick, and J Zico Kolter. DC3: A learning method for optimization with hard constraints. *arXiv [cs.LG]*, April 2021.

Stephen Frank and Steffen Rebennack. An introduction to optimal power flow: Theory, formulation, and examples. *IIE transactions: industrial engineering research & development*, 48(12):1172–1197, December 2016.

Mahdi Khodayar, Okyay Kaynak, and Mohammad E Khodayar. Rough deep neural architecture for short-term wind speed forecasting. *IEEE Transactions on Industrial Informatics*, 13(6):2770–2779, December 2017.

Michael Klamkin, Mathieu Tanneau, and Pascal Van Hentenryck. Dual interior-point optimization learning. *arXiv [cs.LG]*, February 2024.

Matt Kraning. Dynamic network energy management via proximal message passing. *Foundations and trends® in optimization*, 1(2):73–126, 2014.

Lucas Barros Scianni Morais, Giancarlo Aquila, Victor Augusto Durães de Faria, Luana Medeiros Marangon Lima, José Wanderley Marangon Lima, and Anderson Rodrigo de Queiroz. Short-term load forecasting using neural networks and global climate models: An application to a large-scale electrical power system. *Applied Energy*, 348(121439): 121439, October 2023.

Xiang Pan, Minghua Chen, Tianyu Zhao, and Steven H Low. DeepOPF: A feasibility-optimized deep neural network approach for AC optimal power flow problems. *IEEE systems journal*, 17(1):673–683, March 2023.

Andrew Rosemberg, Mathieu Tanneau, Bruno Fanzeres, Joaquim Garcia, and Pascal Van Hentenryck. Learning optimal power flow value functions with input-convex neural networks. *Electric power systems research*, 235(110643):110643, October 2024.

Michel Schubiger, Goran Banjac, and John Lygeros. GPU acceleration of ADMM for large-scale quadratic programming. *Journal of parallel and distributed computing*, 144:55–67, October 2020.

Kamran Tehranchi, Trevor Barnes, Martha Frysztacki, and Ines Azevedo. Pypsa-usa: A flexible open-source energy system model and optimization tool for the united states. *Available at SSRN 5029120*, 2024.

A Device Node Model

We use the DC OPF formulation from Kraning (2014); Degleris et al. (2024). Below, we restate the main details of the formulation for convenience.

Consider a network with N nodes and D devices. Nodes in the network have (voltage) phase angle $u \in \mathbf{R}^N$ that determine their electrical potential within the system. Each device has a vector of power flows $p_d \in \mathbf{R}^{\tau_d}$ and a vector of phase angles $v_d \in \mathbf{R}^{\tau_d}$, where $\tau_d \geq 1$ is the number of *terminals* connected to the device. Every terminal t connects to node $n_d(t) \in \{1, \dots, N\}$; we represent this succinctly through the device's incidence matrix $A_d \in \{0,1\}^{N \times \tau_d}$, where $(A_d)_{nt} = 1$ if $n = n_d(t)$ and $(A_d)_{nt} = 0$ otherwise.

The cost of operating a device with power flows p_d and phase angles v_d is given by a convex function $c_d(\cdot, \cdot; z_d) : \mathbf{R}^{\tau_d} \times \mathbf{R}^{\tau_d} \to \mathbf{R} \cup \{\infty\}$ parameterized by the device data $z_d \in \mathbf{R}^{r_d}$; the functions c_d are extended real-valued, meaning they can take on infinite values in order to model device-specific constraints. Given problem data $z = (z_1, \dots, z_D) \in \mathbf{R}^R$, the grid operator solves the DC OPF problem, which seeks to minimize the device costs subject to power balance and phase consistency constraints,

$$\begin{array}{lll} \text{minimize} & \sum_d c_d(p_d, v_d; z_d) & \\ \text{subject to} & \sum_d A_d p_d = 0, & (\lambda) \\ & A_d^T v_d = u, \quad d = 1, \dots, D & (\omega_d) \end{array} \tag{3}$$

where the variables are $p = (p_1, \dots, p_D)$, $v = (v_1, \dots, v_D)$, and $u \in \mathbf{R}^N$. The dual variables of the power balance and phase consistency constraints are $\lambda \in \mathbf{R}^N$ and $\omega = (\omega_1, \dots, \omega_D)$, respectively; the vector λ also defines the *locational marginal prices* (LMPs) of the network at optimality. This framework is highly general and can model many common devices such as loads, generators, AC transmission lines, and DC transmission lines. We list two examples below.

Generators and loads. Both generators and load can be modeled with simple quadratic costs and a box constraint on power injections,

$$c^{\text{injector}}(p_1, v_1; a, b, p^{\max}, p^{\min}) = p_1^T \mathbf{diag}(a) p_1 + b^T p_1 + \mathcal{I}\{p^{\min} \leq p_1 \leq p^{\max}\},$$

where $\mathcal{I}$ is the convex indicator function on a constraint, which evaluates to ∞ when the constraint is not satisfied and zero otherwise. Here the problem data are the quadratic and linear cost coefficients $a, b \in \mathbf{R}$ and the minimum and maximum power injection $p^{\min}, p^{\max} \in \mathbf{R}$.

AC transmission line. AC transmission lines have device cost function,

$$f(p_1, p_2, v_1, v_2; b, u) = \mathcal{I}\{-u \leq p_1 \leq u\} + \mathcal{I}\{p_1 + p_2 = 0\} + \mathcal{I}\{p_1 = b(v_2 - v_1),$$

where $u \in \mathbf{R}_+$ is the line capacity and $b \in \mathbf{R}_+$ is the line susceptance. The AC transmission line cost is a sum of only indicator terms and is either infinite or zero. The third constraint, $p_1 = b(v_2 - v_1)$, is called the *DC power flow constraint*.

B Proximal Message Passing

Kraning (2014) uses the *alternating direction method of multipliers (ADMM)* to derive an efficient algorithm for DC optimal power flow (Problem (3)) that alternates between two steps,

1. *Proximal schedule updates.* For each device $d = 1, \dots, D$,

$$(p_d^{(i+1)}, v_d^{(i+1)}) = \mathbf{prox}_{c_d, \rho}\left(\mathbf{resid}_d(p^{(i)}) - \lambda^{(i)}, \mathbf{avg}_d(v^{(i)}) - \omega^{(i)}; z_d\right)$$

2. *Dual updates.* For each node $n = 1, \ldots, N$,

$$\lambda_n^{(i+1)} = \lambda_n^{(i)} + \mathbf{avg}_n(p^{(i+1)}),$$
$$\omega_n^{(i+1)} = \omega_n^{(i)} + \mathbf{resid}_n(v^{(i+1)}),$$

where $\mathbf{avg}(x) = \mathbf{diag}(\sum_d A_d \mathbf{1})^{-1} \cdot \sum_d A_d x_d$ and $\mathbf{resid}(x) = x - \mathbf{avg}(x)$. Above, $\mathbf{prox}_{f,\rho}$ is the proximal operator of f, defined as

$$\mathbf{prox}_{f,\rho}(b; z) = \mathbf{argmin}_u \left(f(u; z) + \frac{\rho}{2} \|u - b\|_2^2 \right).$$

The parameter $\rho > 0$ is called the *penalty parameter* and determines how rapidly the power schedules change as a function of the prices. Degleris et al. (2024) shows to implement proximal message passing efficiently on the GPU by vectorizing all proximal operators across devices of the same type and implementing the **avg** and **resid** operators using GPU scatter and gather kernels, respectively. The framework also extends to contingency-constrained DC OPF problems as well.

C Experiment Details

C.1 Data

The power system network data used in these experiments are constructed using the PyPSA-USA dataset (Tehranchi et al., 2024). The model represents the U.S. Western Interconnection for historical years 2019 through 2023. As detailed in Tehranchi et al. (2024), historical data for each region is reconstructed from climate reanalysis data, balancing authority reported demand, historical fuel prices, and unit-level efficiencies. Detailed generator, demand, and transmission representation is clustered to 101 terminals, 488 generators, 251 AC lines, 3 HVDC links, and 101 loads. Generators are clustered to nodal aggregations of combined-cycle turbines, open-cycle turbines, nuclear, solar, onshore wind, biomass, waste, hydroelectric, and coal plants. Temporal data is represented at hourly resolution.

The **Perfect** dataset corresponds to the case when we observe the full problem data, i.e., $x = z$. For the **Noisy** dataset, the generation and load forecasts have been corrupted by noise. More precisely, for a given wind, solar, or load resource, the forecast becomes:

$$y(t) = \max\left(0, \min\left(x(t) + \eta(t), b\right)\right), \tag{4}$$

where $\eta(t) \sim \mathcal{N}(0, \alpha \cdot \mathrm{std}(x))$. Here, $x(t)$ is the original forecast, $\eta(t)$ is Gaussian noise with standard deviation scaled by α, and b is an upper bound ($b = 1$ for wind/solar, $b = \infty$ for load). In our experiments, $\alpha = 0.5$. In the **Outage** dataset, we dynamically and randomly de-rate (i.e., decrease their capacity) 10 AC transmission lines by 50% for 1 month at a time. All three datasets use a train-val-test split of 80%, 10%, and 10%, and split the data in a sequential fashion for the forecasting task.

C.2 Model and Experiment Setup

Each model considers the following set of features:

1. Load forecasts
2. Generator forecasts
3. Generator capacities

When using message passing layers, we explicitly use network topology information as well. The training objective for all models consists of a weighted sum of prediction accuracy and constraint violations given by:

$$\begin{aligned}\mathcal{L} = {} & w_{\text{power}} p_{\text{mse}} + w_{\text{price}} \lambda_{\text{mse}} + w_{\text{imbalance}} p_{\text{imbalance}} \\ & + w_{\text{gen}} p_{\text{genviolations}} + w_{line} p_{\text{lineviolations}} + w_{cost} c_{\text{system}}\end{aligned} \tag{5}$$

- p_{mse} is the mean squared error for power predictions
- λ_{mse} is the mean squared error for price predictions
- $p_{\text{imbalance}}$ measures the mean absolute power imbalance
- $p_{\text{genviolations}}$ penalizes violations of generator capacity constraints
- $p_{\text{lineviolations}}$ penalizes violations of line capacity constraints.
- c_{system} represents the system cost

All three models use a dense neural network architecture consisting of an input layer of dimension 1077, two hidden layers of dimension 200, and a final output layer. In all cases, the hidden layers use batch normalization and ReLU activations.

The output dimension of this dense neural network (DNN) varies across the three different models. For the **Dense** model, the output layer is of dimension 1199 as we directly predict powers for all the devices and prices across all nodes. In the case of the **Proxy** model, the output layer is of dimension 1803 as we predict powers, prices, and phase information. In the case of the **State** model, the output layer is of dimension 2392 as we additionally predict the latent parameters such as load and generation. The outputs of both **State** and **Proxy** are used in the message passing layers, where we complete $L = 5$ message passing steps. All model architectures are implemented in PyTorch.

We train all models for 500 epochs using AdamW with a batch size of 64, a learning rate of 1e-5, and a weight decay 0.01. The learning rate is reduced by a factor of 2 if the validation error has not improved over 10 epochs.