Physics-Informed Graph Learning Acceleration for Large-Scale AC-OPF with Topology Changes

Keunju Song [1] **Kyungnam Park** [1] **Sua Choi** [1] **Seunguk Kim** [2] **Tae-un Kim** [2] **Youngmin Choi** [2] **Sang-Won Min** [3] **Hongseok Kim** [1]

Abstract

In power systems, alternating current optimal power flow (AC-OPF) has been a challenging problem for decades due to its nonconvexity, but fast and efficient solutions are even more needed because of high penetration of large scale renewable generation and load growth. Recently, neural networks (NN) have gained attention in solving AC-OPF, but it is still in an early stage to be applicable for real and large-scale power system operation with topology-changing characteristics. To end this, we propose a novel framework called GraphOPF that considers topology-adaptability, scalability, NN training time, self-supervision, and feasibility altogether. Extensive experiments show that the proposed framework against the baselines is up to 200 times faster in NN training and up to 66 times faster in solving AC-OPF for large-scale power systems including the real Korean power system, while achieving more than 99% feasibility.

1. Introduction

Recently, distributed energy resources (DERs) have been massively deployed in power systems to mitigate climate change while the electric load is rapidly increasing due to the expansion of electric vehicles and data centers. In 2026, DERs are expected to represent 46% of the world's electricity generation and the total electricity consumption of data centers will be more than 1,000 TWh (IEA, 2024), both of which put power systems under high uncertainty and complex operational conditions. To handle this, fast and reliable operation of the power system becomes crucial, which necessitates solving the full AC optimal power flow (OPF). Ever since 1962 when it was first formulated (Carpentier, 1962), AC-OPF has been a fundamental problem in power system operation, aiming to determine the most cost-effective generator dispatch while satisfying physical constraints. Recently, deep learning (DL)-based approaches have been proposed to solve AC-OPF in a fast and reliable way leveraging new neural network (NN) architectures and learning methods. At an early stage, the works in (Pan et al., 2022; Park et al., 2023; Jia et al., 2022; Xu et al., 2025) utilized modern NN architectures, e.g., deep neural networks (DNN), convolutional neural networks (CNN), and Transformers. Very recently, graph neural networks (GNN) have been studied to solve more practical AC-OPF problems (e.g., security-constrained AC-OPF and AC-OPF with topology changes). In (Falconer & Mones, 2022), the authors showed that GNN is effective for topology changes compared to DNN and CNN models. Then, the works in (Liu et al., 2022) and (Yang et al., 2024) analyzed the properties of GNN and proposed a GNN-based learning method for topology changes while keeping model complexity low. Meanwhile, the authors of (Gao et al., 2023) focused on the model architecture, designing the physics-embedded GNN that follows the Gauss-Seidel iteration of AC power flow to solve AC-OPF in $N-1$ contingency.

Other than neural network architectures, learning mechanism is another considerable factor in solving AC-OPF. To minimize the power generation cost and satisfy the physical constraints simultaneously, the physics-informed learning has been studied in two categories: supervised learning and unsupervised learning. The works in (Huang et al., 2021; Park et al., 2023; Jia et al., 2022; Xu et al., 2025; Falconer & Mones, 2022) designed the loss function as mean squared error or mean absolute error, which require labels. However, they may incur expensive costs for NN training because they need labels generated from conventional solvers. Semi-supervised learning (Hien et al., 2025; Chen et al., 2025) has been considered to mitigate this, however, it cannot be fully resolved. To overcome this, the works in (Donti et al., 2021; Kim & Kim, 2024) designed an unsupervised learning method by leveraging the implicit function theorem to enable NN training without labels while satisfying the

[1]Sogang University, Republic of Korea [2]Korea Power Exchange, Republic of Korea [3]Korea Electrotechnology Research Institute, Republic of Korea. Correspondence to: Hongseok Kim <hongseok@sogang.ac.kr>.

Accepted by AI4Science workshop at the 43rd International Conference on Machine Learning, Seoul, South Korea, 2026. Copyright 2026 by the author(s).

constraints.

Even though the aforementioned works showed the potentials of NNs in solving AC-OPF, five practical perspectives such as training time, scalability, feasibility, self supervision, and topology adaptability were not considered altogether because of their trade-offs. In order to overcome all the challenges, we propose GraphOPF, a topology-adaptable unsupervised GNN framework that enables scalable and fast learning. Specifically, GraphOPF utilizes edge-aided GNN (EA-GNN) and a hard-constrained embedded layer. Unlike conventional GNN architecture, EA-GNN consists of multi-edge convolutional layer and Chebyshev graph convolutional layer, which consider edge features of graph data to improve the solution performance (e.g., optimality gap, feasibility) and convergence of NN training further. Moreover, to make it compatible for large-scale AC-OPF, the hard-constrained embedded layer is integrated with EA-GNN to overcome the heavy computational burden of computing AC power flow in implicit layer. Along with those functionalities, the proposed GraphOPF follows an unsupervised learning scheme, so it does not require extra time in preparing labels.

We summarize our key contributions as follows.

- We propose a novel GNN architecture called EA-GNN, which effectively solves AC-OPF problems. By leveraging edge features along with node features of power systems, EA-GNN maintains the quality of AC-OPF solutions (i.e., optimality gap and feasibility) even for national scale power systems. Furthermore, thanks to the graph representation learning in EA-GNN, the proposed GraphOPF is adaptable for topology changes.
- To make it applicable for large-scale AC-OPF, we design a hard-constrained embedded layer to accelerate computation for power flow equations. This ensures that equality constraints in AC-OPF are strictly satisfied. We find that integrating EA-GNN with a hard-constrained embedded layer enables fast convergence in NN training without using labels.
- The proposed GraphOPF outperforms baselines in inference time, NN training time, and constraint satisfaction for large-scale AC-OPF. Specifically, for the real Korean power system having around 4500 buses and 6000 transmission lines, our framework takes around *five minutes* for NN training and shows 99% or more constraint satisfaction on average. Furthermore, even under topology changes during real power system operation, GraphOPF can finish model update within *one minute* by using graph transfer learning.

The rest of this paper is organized as follows. In Section 2, we describe the AC-OPF problem formulation. In Section 3, we present the methodologies of GraphOPF. The experimental settings and results are given in Section 4 and Section 5, followed by the conclusion in Section 7.

2. Problem Formulation

The AC-OPF problem, a nonlinear and nonconvex optimization problem in power systems, aims to minimize the power generation cost while satisfying the physical constraints. For a power network $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ where $\mathcal{N}$ is the set of buses and $\mathcal{E}$ is the set of transmission lines, AC-OPF is formulated as follows:

$$\min_{\mathbf{P}_g, \mathbf{Q}_g, \mathbf{V}, \boldsymbol{\theta}} \quad \sum_{i \in \mathcal{N}} C_i(P_{g,i}) \tag{1a}$$

$$\text{s.t.} \quad P_{g,i}^{\min} \leq P_{g,i} \leq P_{g,i}^{\max}, \forall i \in \mathcal{N}, \tag{1b}$$

$$Q_{g,i}^{\min} \leq Q_{g,i} \leq Q_{g,i}^{\max}, \forall i \in \mathcal{N}, \tag{1c}$$

$$V_i^{\min} \leq V_i \leq V_i^{\max}, \forall i \in \mathcal{N}, \tag{1d}$$

$$p_{ij}^2 + q_{ij}^2 \leq (s_{ij}^{\max})^2, \forall (i,j) \in \mathcal{E}, \tag{1e}$$

$$p_{ij} = -G_{ij} V_i^2 + V_i V_j (G_{ij} \cos\theta_{ij} + B_{ij} \sin\theta_{ij}), \tag{1f}$$

$$q_{ij} = B_{ij} V_i^2 + V_i V_j (G_{ij} \sin\theta_{ij} - B_{ij} \cos\theta_{ij}), \tag{1g}$$

$$P_{g,i} - P_{d,i} = V_i \sum_{j=1}^{N} V_j (G_{ij} \cos\theta_{ij} + B_{ij} \sin\theta_{ij}), \tag{1h}$$

$$Q_{g,i} - Q_{d,i} = V_i \sum_{j=1}^{N} V_j (G_{ij} \sin\theta_{ij} - B_{ij} \cos\theta_{ij}), \tag{1i}$$

where the optimization variables $\mathbf{P}_g, \mathbf{Q}_g, \mathbf{V}, \boldsymbol{\theta}$ are in $\mathbb{R}^{|\mathcal{N}|}$ where $|\cdot|$ is the cardinality of a set, and their i-th element is associated with bus $i \in \mathcal{N}$. For example, $P_{g,i}$ and $Q_{g,i}$ are the active and reactive power of generator $i \in \mathcal{N}$, $P_{d,i}$ and $Q_{d,i}$ are the active and reactive load demands of bus $i \in \mathcal{N}$, V_i is the voltage magnitude of bus $i \in \mathcal{N}$, and $\theta_{ij} = \theta_i - \theta_j$ is the voltage angle differences between bus i and bus j. Let p_{ij} and q_{ij} denote the active and reactive branch power flows from bus i to bus j. Let G_{ij} and B_{ij} denote the conductance and susceptance of the nodal admittance matrix $\mathbf{Y} \in \mathbb{R}^{|\mathcal{N}| \times |\mathcal{N}|}$. The objective function (1a) is the sum of generator cost functions $C_i(P_{g,i}), i \in \mathcal{N}$. The constraints (1b) and (1c) limit the generator output of active and reactive power, (1d) enforces that voltage magnitudes remain within their specified limits, (1e) is the line flow limit, (1f) and (1g) indicate the branch power equations based on Ohm's law, and the constraints (1h) and (1i) are power flow equations based on Kirchhoff's current law.

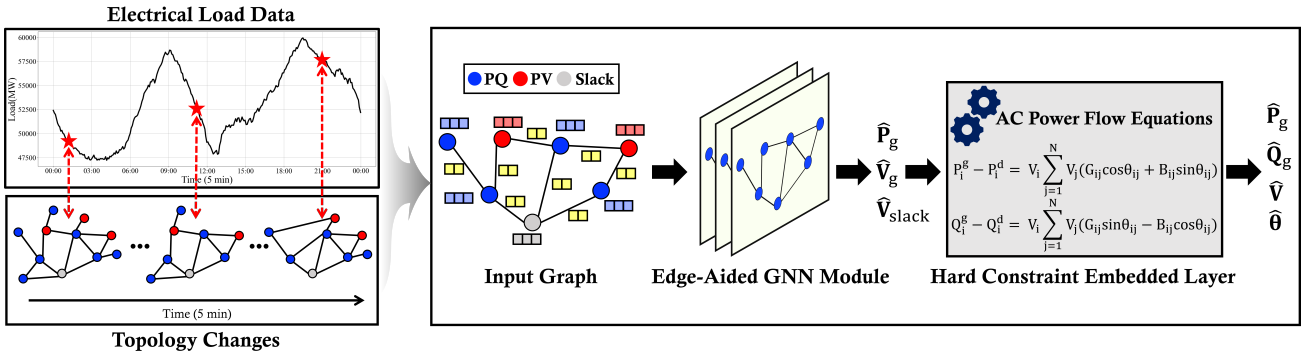

Figure 1. The overall framework of the proposed GraphOPF.

3. Proposed Methodologies

In this section, we describe the overall process of GraphOPF as shown in Figure 1.

3.1. Edge-Aided Graph Neural Networks

The input graph of GNN is a power network $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ where the nodes and edges correspond to the buses and transmission lines. Then, a node feature matrix $\mathbf{X}^{(0)} \in \mathbb{R}^{|\mathcal{N}| \times 2}$ consists of the active and reactive load demands of bus $i \in \mathcal{N}$. In this study, we leverage an edge feature vector $\mathbf{E} \in \mathbb{R}^{|\mathcal{E}| \times 4}$ that consists of the resistance, reactance, capacitance, and the line thermal limit of transmission lines. To handle the node and edge features collectively, we propose EA-GNN by leveraging multi-edge convolutional layer and Chebyshev graph convolutional layer as shown in Fig 2. Details are as follows.

3.1.1. Multi-Edge Convolutional Layer

A multi-edge convolutional layer (MEConv) is the first layer that has the graph data $\mathbf{X}^{(0)} = \{\mathbf{x}_i^{(0)} \mid i \in \mathcal{N}\}$ and $\mathbf{E}$ as inputs. This layer follows the message passing framework (Hamilton, 2020), which is the common mechanism in GNN. Then, the proposed MEConv layer is given by

$$\mu_{ij} = \Phi\big(\kappa \mathbf{x}_i^{(l-1)} || \kappa \mathbf{x}_j^{(l-1)} || \kappa e_{(i,j)}; \Theta_{\text{edge}}^{(l)}\big), \tag{2a}$$

$$\mu_i = \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} \mu_{ij}, \tag{2b}$$

$$\sigma_i = \sqrt{\frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} \mu_{ij}^2 - \mu_i^2}, \tag{2c}$$

$$\mathbf{x}_i^{(l)} = \prod \big(\mu_i + \sigma_i; \Theta_{\text{aggr}}^{(l)}\big), \tag{2d}$$

where Φ is the aggregate function parameterized by $\Theta_{\text{edge}}^{(l)}$ having two multi-layer perceptrons (MLPs), $\kappa = \frac{1}{\sqrt{\deg(i)\deg(j)}}$ denotes the normalization coefficient for the degree of node i and j, and $||$ denotes concatenation. For the message function, we propose a multi-moment aggregation method by designing the projection layer $\prod$ parameterized by $\Theta_{\text{aggr}}^{(l)}$, which takes the sum of mean and standard deviation of the outputs of Φ as an input. Surprisingly, we will see that having multi-moments, i.e., the first and second moments of the distribution of the aggregate function Φ generates, is simple but effective.

Through a multi-moment aggregation method with NNs, our proposed framework first outputs the *topological* feature matrix of power systems for every node $\mathbf{X}_{\text{MEC}} \in \mathbb{R}^{|\mathcal{N}| \times F}$. Note that $\mathbf{X}_{\text{MEC}}$ is a *node* feature matrix derived from the node feature matrix $\mathbf{X}^{(l)}$ and the edge feature vector $\mathbf{E}$, and can be described as the assembled MEConv in the graph perspective such as

$$\mathbf{X}_{\text{MEC}}^{(l)} = MEConv\big(\mathbf{X}^{(l)}, \mathbf{E}; \Theta_{\text{edge}}^{(l)}, \Theta_{\text{aggr}}^{(l)}\big). \tag{3}$$

3.1.2. Chebyshev Graph Convolutional Layer

After getting the topological feature $\mathbf{X}_{\text{MEC}}^{(l)}$ from MEConv, we feed it as input to the Chebyshev graph convolutional layer (ChebyConv) (Defferrard et al., 2016) which is parameterized by $\Theta_{\text{Cheby}}^{(l)} = \{\Theta_k^{(l)} \mid k = 1, \cdots, K\}$ so that

$$\mathbf{X}_{\text{Cheby}}^{(l)} = ChebyConv(\mathbf{X}_{\text{MEC}}^{(l)}; \Theta_{\text{Cheby}}^{(l)}) \tag{4}$$

where $\mathbf{X}_{\text{Cheby}}^{(l)} \in \mathbb{R}^{|\mathcal{N}| \times F}$ is the output node feature matrix of the ChebConv layer. Note that $\mathbf{X}_{\text{MEC}}^{(l)}$ from MEConv only considers 1-hop neighbors of each node, so it relies on local connectivity features. Therefore, through the ChebyConv layer, we consider K-hop neighbors (i.e., K Chebyshev filters) of each node for extracting global connectivity features. A ChebyConv layer follows the Chebyshev polynomials,

$$\mathbf{X}_{\text{Cheby}}^{(l)} = \sum_{k=1}^{K} \mathbf{Z}_k^{(l)} \Theta_k^{(l)} \tag{5}$$

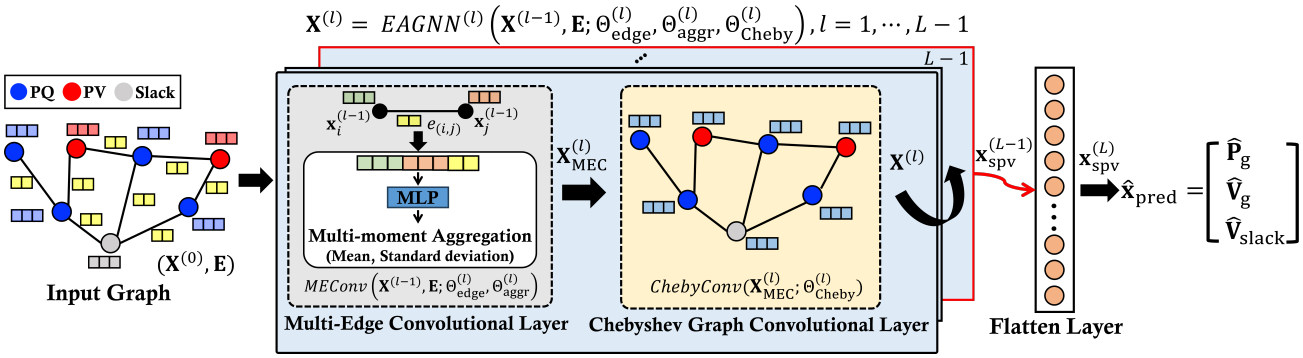

Figure 2. The architecture of the proposed EA-GNN.

where K corresponds to the Chebyshev filter size, and $\mathbf{Z}_k^{(l)}$ is computed recursively as $\mathbf{Z}_k^{(l)} = 2\hat{\mathbf{L}}\mathbf{Z}_{k-1}^{(l)} - \mathbf{Z}_{k-2}^{(l)}$ where $\mathbf{Z}_0^{(l)} = \mathbf{X}_{\text{MEC}}^{(l)}$ and $\mathbf{Z}_1^{(l)} = \hat{\mathbf{L}}\mathbf{X}_{\text{MEC}}^{(l)}$. $\hat{\mathbf{L}} = \frac{2\mathbf{L}}{\lambda_{\max}} - \mathbf{I}$ is the scaled and normalized Laplacian where $\mathbf{L} = \mathbf{D} - \mathbf{A}$ is the Laplacian matrix, which is calculated by the diagonal degree matrix $\mathbf{D} \in \mathbb{R}^{|\mathcal{N}|\times|\mathcal{N}|}$ and the adjacency matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{N}|\times|\mathcal{N}|}$. $\lambda_{\max}$ denotes the largest eigenvalue of $\mathbf{L}$ and $\mathbf{I} \in \mathbb{R}^{|\mathcal{N}|\times|\mathcal{N}|}$ denotes the identity matrix.

By combining the MEConv and ChebyConv layers sequentially, now we can define the assembled EA-GNN that effectively captures the *interactive* topological features $\mathbf{X}^{(l)}$ considering local and global connectivity in the graph data:

$$\mathbf{X}^{(l)} = EAGNN^{(l)}(\mathbf{X}^{(l-1)}, \mathbf{E}; \Theta^{(l)}_{\text{edge}}, \Theta^{(l)}_{\text{aggr}}, \Theta^{(l)}_{\text{Cheby}}), \quad l = 1, \ldots, L-1. \tag{6}$$

3.1.3. Flatten Layer

For the topological features of EA-GNN at the last layer $\mathbf{X}^{(L-1)} \in \mathbb{R}^{|\mathcal{N}|\times F_{\text{out}}}$, we use the reshape function $f_{\text{reshape}} : \mathbb{R}^{|\mathcal{N}|\times F_{\text{out}}} \to \mathbb{R}^{F_{\text{spv}}}$ to extract the features of generator and slack bus in a power network, where $F_{\text{spv}} = |\mathcal{N}_{\text{spv}}| \times F_{\text{out}}$ and $\mathcal{N}_{\text{spv}} \subseteq \mathcal{N}$ is a union of a set of generator buses $\mathcal{N}_{\text{pv}}$ and slack bus $\mathcal{N}_{\text{slack}}$:

$$\mathbf{x}_{\text{spv}}^{(L-1)} = f_{\text{reshape}}(\mathbf{X}^{(L-1)}) \tag{7}$$

where $\mathbf{x}_{\text{spv}}^{(L-1)}$ is the partial topological features of $\mathbf{X}^{(L-1)}$ that consist of generator and slack bus. Then, we input $\mathbf{x}_{\text{spv}}^{(L-1)}$ into the flatten layer to output $\mathbf{x}_{\text{spv}}^{(L)} \in \mathbb{R}^{2|\mathcal{N}_{\text{pv}}|+|\mathcal{N}_{\text{slack}}|}$ so that

$$\mathbf{x}_{\text{spv}}^{(L)} = Flatten(\mathbf{x}_{\text{spv}}^{(L-1)}; \Theta_{\text{flat}}). \tag{8}$$

In (8), $\mathbf{x}_{\text{spv}}^{(L)}$ is within the range of 0 and 1 because a sigmoid function is used as an activation function. Thus, we can finally reconstruct $\hat{\mathbf{x}}_{\text{pred}}$ from $\mathbf{x}_{\text{spv}}^{(L)}$ by using operating constraints $\mathbf{x}^{\max}, \mathbf{x}^{\min} \in \mathbb{R}^{2|\mathcal{N}_{\text{pv}}|+|\mathcal{N}_{\text{slack}}|}$ (i.e., the limit of active power and voltage magnitude of generator and the slack bus), which is the *partial* decision variables in (1):

$$\hat{\mathbf{x}}_{\text{pred}} = \mathbf{x}_{\text{spv}}^{(L)}(\mathbf{x}^{\max}) + (\mathbf{1} - \mathbf{x}_{\text{spv}}^{(L)})(\mathbf{x}^{\min}). \tag{9}$$

Note that $\hat{\mathbf{x}}_{\text{pred}} = [\hat{\mathbf{P}}_g, \hat{\mathbf{V}}_g, \hat{\mathbf{V}}_{\text{slack}}]^\intercal$ consists of the active power generation and voltage magnitude for generator bus and voltage magnitude for slack bus, respectively.

3.2. Hard Constraint Embedded Layer

So far, we have discussed the proposed EA-GNN module as summarized in Fig 2. Now we discuss how to output *full* decision variables in (1) based on the *partial* variables from EA-GNN by using the proposed hard constraint embedded layer, as shown in Fig 1. Specifically, we exploit the advanced implicit layer that can accelerate the computation, having $\hat{\mathbf{x}}_{\text{pred}}$ in (9) as input.

3.2.1. Advanced Implicit Layer

First, to formulate the hard constraint embedded layer, we express the AC-OPF problem (1) into the standard constrained optimization problem:

$$\min_{\mathbf{y}\in\mathbb{R}^n} f_{\mathbf{d}}(\mathbf{y}) \quad \text{s.t. } g_{\mathbf{d}}(\mathbf{y}) \leq 0, \; h_{\mathbf{d}}(\mathbf{y}) = 0 \tag{10}$$

where $\mathbf{d} \in \mathbb{R}^n \sim \mathcal{D}$ is the input data point from the dataset $\mathcal{D}$, $\mathbf{y} \in \mathbb{R}^n$ has the optimal decision variables, $f_{\mathbf{d}} : \mathbb{R}^n \to \mathbb{R}$ is the objective function, $g_{\mathbf{d}} : \mathbb{R}^n \to \mathbb{R}^{n_{\text{ineq}}}$ denotes the inequality constraints, and $h_{\mathbf{d}} : \mathbb{R}^n \to \mathbb{R}^{n_{\text{eq}}}$ denotes the equality constraints.

In our problem, $\mathbf{y} = [\mathbf{P}_g, \mathbf{Q}_g, \mathbf{V}, \boldsymbol{\theta}]^\intercal$ is a collection of the decision variables in (1), $g_{\mathbf{d}}$ corresponds to (1b)–(1e), and $h_{\mathbf{d}}$ corresponds to (1h) and (1i), which are nonlinear and make our problem nonconvex optimization (Bienstock &

Verma, 2019). Thus, satisfying $h_{\mathbf{d}}$ by using NN is a challenging part as mentioned in (Huang et al., 2021; Zeng et al., 2024). To overcome this problem, the implicit layer method has been considered to ensure that the equality constraints of $h_{\mathbf{d}}$ are always satisfied (Donti et al., 2021; Kim & Kim, 2024; Chen et al., 2025). In our work, the proposed hard constraint embedded layer also follows the implicit layer method. However, directly calculating the Jacobian inverse in the Newton updates of the implicit layer (Donti et al., 2021; Kim & Kim, 2024) becomes computationally expensive for large-scale problems. To address this issue, we leverage the associated linear systems using LU factorization with partial pivoting. We find that this is simple but effectively accelerates the computation of $h_{\mathbf{d}}$ of a large-scale power system while considering implicit differentiation (Donti et al., 2021; Kim & Kim, 2024), resulting in an additional reduction of the overall NN training time.

3.3. Lagrangian Dual-based Loss Function

Due to the hard constraint embedded layer, we can define a Lagrangian dual-based loss function (Kim & Kim, 2024) without considering equality constraints as follows:

$$\mathcal{L}_{\text{loss}}(\hat{\mathbf{y}}) = \mathcal{L}_{\text{obj}}(\hat{\mathbf{y}}) + \mathcal{L}_{\text{ineq}}(\hat{\mathbf{y}}) \tag{11}$$

where $\hat{\mathbf{y}} = [\hat{\mathbf{P}}_g, \hat{\mathbf{Q}}_g, \hat{\mathbf{V}}, \hat{\boldsymbol{\theta}}]^\intercal$, $\mathcal{L}_{\text{obj}}$ is the cost of generation by the predicted active power generation $\hat{\mathbf{P}}_g$, and $\mathcal{L}_{\text{ineq}}$ denotes the amount of total violation. Specifically, with the Lagrange multipliers for each instance in a set of inequality constraints $\mathcal{C}_{\text{ineq}}$, we have $\mathcal{L}_{\text{ineq}}(\hat{\mathbf{y}}) = \sum_{c \in \mathcal{C}_{\text{ineq}}} \lambda_c \max(0, \nu_c(\hat{\mathbf{y}}))$ where $\nu_c(\hat{\mathbf{y}})$ can be determined as

$$\nu_c^L(\hat{P}_{g,i}) = (P_{g,i}^{\min} - \hat{P}_{g,i}), \ \ \forall i \in \mathcal{N}, \tag{12a}$$
$$\nu_c^U(\hat{P}_{g,i}) = (\hat{P}_{g,i} - P_{g,i}^{\max}), \ \ \forall i \in \mathcal{N}, \tag{12b}$$
$$\nu_c^L(\hat{Q}_{g,i}) = (Q_{g,i}^{\min} - \hat{Q}_{g,i}), \ \ \forall i \in \mathcal{N}, \tag{12c}$$
$$\nu_c^U(\hat{Q}_{g,i}) = (\hat{Q}_{g,i} - Q_{g,i}^{\max}), \ \ \forall i \in \mathcal{N}, \tag{12d}$$
$$\nu_c^L(\hat{V}_i) = (V_i^{\min} - \hat{V}_i), \ \ \forall i \in \mathcal{N}, \tag{12e}$$
$$\nu_c^U(\hat{V}_i) = (\hat{V}_i - V_i^{\max}), \ \ \forall i \in \mathcal{N}, \tag{12f}$$
$$\nu_c^L(\hat{s}_{(i,j)}) = (\hat{s}_{(i,j)} - s_{(i,j)}^{\max}), \ \ \forall (i,j) \in \mathcal{E}, \tag{12g}$$

where $\hat{\mathbf{s}}_{(i,j)}$ is the predicted branch power flow over transmission lines, which can be computed by (1f)−(1g) based on $\hat{V}_i$ and $\hat{\theta}_{ij}$. In doing this, (12a)−(12g) induce to satisfy the inequality constraints in AC-OPF (1b)−(1e) during NN training. Finally, we update $\Theta_{\text{edge}}, \Theta_{\text{aggr}}, \Theta_{\text{Cheby}}, \Theta_{\text{flat}}$ (i.e., the learnable parameters of EA-GNN) and $\boldsymbol{\lambda}$, *alternatively* by minimizing (11) and checking the constraint violation. Let $\boldsymbol{\Theta} = \{\Theta_{\text{edge}}, \Theta_{\text{aggr}}, \Theta_{\text{Cheby}}, \Theta_{\text{flat}}\}$ be the collection of the learnable parameters of EA-GNN. Then we have

$$\boldsymbol{\Theta}' = \underset{\boldsymbol{\Theta}}{\operatorname{argmin}}\, \mathcal{L}_{\text{loss}}. \tag{13}$$

$$\boldsymbol{\lambda} = \Big(\lambda_c + \rho \max(0, \nu_c(\hat{\mathbf{y}}')), \quad c \in \mathcal{C}_{\text{ineq}}\Big), \tag{14}$$

where $\hat{\mathbf{y}}'$ indicates the output of the hard constraint embedded layer with EA-GNN parameterized by $\boldsymbol{\Theta}'$. In other words, it is important to compute (14) using $\hat{\mathbf{y}}'$ which is based on a hot-started $\boldsymbol{\Theta}'$ in (13) (Fioretto et al., 2020).

4. Experimental Settings

4.1. Data Preparation

We use four cases from PGLib (version 23.07) (Babaeinejadsarookolaee et al., 2019): GOC-2312, GOC-3970, GOC-4601, and EPIGRIDS-5658 bus systems, and the real Korean power system (denoted as Korea-4492), provided by Korea Power Exchange (KPX). For data generation, the load data is sampled from a uniform distribution within $[80\%, 120\%]$ of the default value on each bus in GOC-2312, GOC-3970, GOC-4601 and EPIGRIDS-5658. For Korea-4492, we set $[80\%, 110\%]$ considering the electric load pattern of the Korean power systems. Then we sample 10,000 training data and 2,000 test data for each case by using MATPOWER (Zimmerman et al., 2010).

4.2. Comparative Models

We extensively compare the proposed GraphOPF with 6 state-of-the-art algorithms, which are summarized as follows.

- PowerModels.jl (Coffrin et al., 2018): PowerModels.jl is an open-source library in Julia for steady-state power system optimization. It is implemented based on the JuMP (Dunning et al., 2017) package, which is a modeling language for mathematical optimization. For solving AC-OPF problems, we use the sparse linear solver MA57 in IPOPT (denoted as IPOPT-MA57 hereafter).

- ExaModels+MadNLP (Shin et al., 2024): ExaModels is an algebraic modeling tool for mathematical optimization, specialized for single instruction multiple data (SIMD) abstraction of nonlinear programs. Thus, it supports parallel processing on GPU accelerators. MadNLP is a nonlinear programming solver based on IPOPT. It provides not only various types of sparse and dense linear solvers but also GPU-based solvers (e.g., cuDSS). In our experiments, we use ExaModels and MadNLP in terms GPU.

- OPF-DNN (Fioretto et al., 2020): OPF-DNN follows supervised learning based on a DNN architecture. It outputs $\mathbf{P}_g, \mathbf{Q}_g, \mathbf{V}$, and $\boldsymbol{\theta}$ for given active and reactive load data by minimizing the Lagrangian dual-based loss function.

- DC3 (Donti et al., 2021): DC3 is a DL-based non-convex optimization solver that follows unsupervised learning. In AC-OPF, it first outputs partial variables $\mathbf{P}_g, \mathbf{V}_g, \mathbf{V}_{\text{slack}}$ from a DNN model having active/reactive load data as input, and constructs full variables $\mathbf{P}_g, \mathbf{Q}_g, \mathbf{V}, \boldsymbol{\theta}$ by using the implicit layer. Note that the AC-OPF problem formulated in (Donti et al., 2021) did not consider the line flow limit, i.e., (1e).

- DeepLDE (Kim & Kim, 2024): DeepLDE is an unsupervised learning-based AC-OPF solver based on a MLP architecture. Same as DC3, it leverages the implicit layer to construct full variables from partial variables predicted by MLP. To satisfy the inequality constraints, DeepLDE leverages a Lagrangian dual-based loss function.

- PG-GNN (Yang et al., 2024): PG-GNN is an unsupervised GNN-based AC-OPF solver that leverages the Lagrangian dual-based loss function (Fioretto et al., 2020). It outputs $\mathbf{P}_g, \mathbf{Q}_g, \mathbf{V}$, and $\boldsymbol{\theta}$ for given input features $\mathbf{P}_d, \mathbf{Q}_d, \mathbf{P}_g^{\min}, \mathbf{P}_g^{\max}.\mathbf{Q}_g^{\min}, \mathbf{Q}_g^{\max}$. In our experiments, we leverage ChebNet-4 model in (Yang et al., 2024) that consists of 3 ChebyConv layers with 4 Chebyshev filters.

4.3. Performance Metrics

To evaluate the performance of each model, we use four metrics as follows.

- **Optimality gap** $\mathcal{F}_{opt}$: The optimality gap is computed as $100 \times \frac{f(\hat{\mathbf{P}}_g) - f(\mathbf{P}_g^\star)}{f(\mathbf{P}_g^\star)}$ where $f(\cdot)$ is the generation cost function and $\hat{\mathbf{P}}_g$ is the predicted active power generation of generators. $\mathbf{P}_g^\star$ is the cost calculated by the MIPS solver in MATPOWER, which is not necessarily globally optimal but serves as the reference value for performance evaluation.

- **Feasibility violation**: To check the feasibility of the predicted solution, we compute the feasibility violation (FV) of the inequality constraints (denoted as $\mathbf{P}_g^{FV}, \mathbf{Q}_g^{FV}, \mathbf{V}^{FV}$, and $\mathbf{S}^{FV}$) and the equality constraints (denoted as Active PF^{FV} and Reactive PF^{FV}). For inequality constraints, FV is calculated through (12a)−(12g). For equality constraints, FV is the residual between the left and right terms in (1h) and (1i), respectively.

- **Feasibility satisfaction**: We also check the ratios of buses, generators, and transmission lines that satisfy the corresponding inequality (denoted as $\mathbf{P}_g^{FS}, \mathbf{Q}_g^{FS}, \mathbf{V}^{FS}$, and $\mathbf{S}^{FS}$) and equality (denoted as Active PF^{FS} and Reactive PF^{FS}) constraints. We set the tolerance as 10^{-2} to the Active PF^{FV} and Reactive PF^{FV} when calculating the Active PF^{FS} and Reactive PF^{FS}, respectively.

- **Computational time**: In the case of DL-based methods, NN training time T_{train} and the inference time T_{inf} are measured. In the case of mathematical models, the time to get a solution is measured as T_{opt}.

Table 1. Performance comparison with mathematical optimization models for large-scale PGLib power systems.

BUS SYSTEM	PERFORMANCE METRICS	POWERMODELS.JL (IPOPT-MA57)	EXAMODELS+MADNLP (CUDSS)	GRAPHOPF
GOC-2312	T_{inf}/T_{opt} (ms)	4493.90	577	**68.07**
	COST	**452259.44**	452264.19	463081.55
GOC-3970	T_{inf}/T_{opt} (ms)	10827.36	1022.72	**168.94**
	COST	**3830521.07**	3830537.23	3868293
GOC-4601	T_{inf}/T_{opt} (ms)	11467.28	1193.87	**179.27**
	COST	**3435540.55**	3435597.09	3462860.50
EPIGRIDS-5658	T_{inf}/T_{opt} (ms)	11279.51	830.76	**249.73**
	COST	**10150409.46**	10150725.90	10271291.50

5. Experimental Results

In this section, we provide experimental results for two types of problems: large-scale PGLib power systems and the case study of real Korean power system. Our simulation setup is based on five NVIDIA GPUs (i.e., four NVIDIA RTX 4090 TI GPUs and the NVIDIA A100 GPU) and a Windows desktop with the AMD Ryzen 9 5900X 12-Core Processor and 16 GB RAM. We use five NVIDIA GPUs for NN training, and only use the NVIDIA RTX 4090 TI GPU for the NN test phase and the GPU-based mathematical optimization solver. In the following tables, the values with bold font show the best performance, and red font for unacceptably poor performance, if any.

5.1. Performance Comparison Results for Large-scale PGLib Power Systems

5.1.1. MATHEMATICAL OPTIMIZATION MODELS

We compare GraphOPF with PowerModels.jl and ExaModels+MadNLP in terms of average computational time and objective cost. As shown in Table 1, GraphOPF shows up to $66.02\times$ speedup in computation and $0.80\% - 2.39\%$ of optimality gap for all PGLib power systems. This indicates that GraphOPF can compute substantially faster than the baselines while maintaining the low optimality gap in large-scale AC-OPF. One might think that ExaModels+MadNLP (Shin et al., 2024) shows reasonably fast computation and also trustworthy solutions in large-scale AC-OPF, which questions the need of AI-based AC-OPF solutions. However, mathematical optimization models often fail in finding a solution, which will be discussed in Section 5.2.2 in detail.

5.1.2. DL-BASED OPTIMIZATION MODELS

In Table 2, we compare GraphOPF with PG-GNN (Yang et al., 2024), DeepLDE (Kim & Kim, 2024), DC3 (Donti

Table 2. Computational time, Optimality gap, and Feasibility satisfaction comparison for large-scale PGLib power systems (DL-based optimization models).

BUS SYSTEM	PERFORMANCE METRICS	PG-GNN	DEEPLDE	DC3	OPF-DNN	**GRAPHOPF**
GOC-2312	$T_{train}(sec)/N_{train}$	2593/8000	7466/1000	13021/1000	7346/7800	**962**/100
	$\mathcal{F}_{opt}(\%)$	26.56	2.86	**2.35**	24.26	2.39
	$\mathbf{P}_g^{FS}(\%)$	**100**	**100**	**100**	**100**	**100**
	$\mathbf{Q}_g^{FS}(\%)$	**100**	98.78	98.15	**100**	98.88
	$\mathbf{V}^{FS}(\%)$	**100**	**100**	**100**	**100**	**100**
	$\mathbf{S}^{FS}(\%)$	**100**	99.89	96.11	98.89	99.97
	ACTIVE PF$^{FS}(\%)$	42.50	**100**	**100**	39.16	**100**
	REACTIVE PF$^{FS}(\%)$	27.28	**100**	**100**	71.05	**100**
GOC-3970	$T_{train}(sec)/N_{train}$	5904/8000	17100/1000	24660/1000	17040/7800	**2280**/100
	$\mathcal{F}_{opt}(\%)$	5.91	4.84	**0.17**	1.97	0.99
	$\mathbf{P}_g^{FS}(\%)$	**100**	99.79	99.71	**99.99**	99.46
	$\mathbf{Q}_g^{FS}(\%)$	**100**	88.32	91.06	**99.99**	97.48
	$\mathbf{V}^{FS}(\%)$	**100**	**100**	98.68	**100**	**100**
	$\mathbf{S}^{FS}(\%)$	**100**	**100**	99.94	99.99	**100**
	ACTIVE PF$^{FS}(\%)$	28.02	**100**	**100**	51.69	**100**
	REACTIVE PF$^{FS}(\%)$	59.27	**100**	99.90	91.63	**100**
GOC-4601	$T_{train}(sec)/N_{train}$	6656/8000	24549/1000	25320/1000	18060/7800	**2271**/100
	$\mathcal{F}_{opt}(\%)$	4.41	7.93	**0.21**	5.76	0.80
	$\mathbf{P}_g^{FS}(\%)$	**100**	99.84	**100**	**100**	**100**
	$\mathbf{Q}_g^{FS}(\%)$	**100**	86.25	87.22	**100**	98.29
	$\mathbf{V}^{FS}(\%)$	**100**	**100**	99.19	**100**	99.95
	$\mathbf{S}^{FS}(\%)$	**100**	**100**	99.97	**100**	**100**
	ACTIVE PF$^{FS}(\%)$	24.17	**100**	99.96	49.88	**100**
	REACTIVE PF$^{FS}(\%)$	62.24	**100**	99.96	89.69	**100**
EPIGRIDS-5658	$T_{train}(sec)/N_{train}$	15209/8000	18372/1000	7451/1000	18809/7800	**1991**/100
	$\mathcal{F}_{opt}(\%)$	47.93	**1.01**	30.62	17.58	1.19
	$\mathbf{P}_g^{FS}(\%)$	**100**	99.84	99.05	**100**	99.82
	$\mathbf{Q}_g^{FS}(\%)$	**100**	76.04	53.02	**100**	96.13
	$\mathbf{V}^{FS}(\%)$	**100**	99.64	**100**	**100**	**100**
	$\mathbf{S}^{FS}(\%)$	52.80	99.90	99.80	98.31	**100**
	ACTIVE PF$^{FS}(\%)$	5.94	**100**	99.94	32.49	**100**
	REACTIVE PF$^{FS}(\%)$	9.66	**100**	99.91	49.39	**100**

Table 3. Performance comparison with mathematical optimization models for Korea-4492 system.

BUS SYSTEM	PERFORMANCE METRICS	POWERMODELS.JL (IPOPT-MA57)	EXAMODELS+MADNLP (CUDSS)	**GRAPHOPF**
KOREA-4492	T_{inf}/T_{opt} (ms)	5630.70	591.14	**259.16**
	COST	**121279.59**	121279.71	121317.21

et al., 2021), and OPF-DNN (Fioretto et al., 2020) in terms of average NN training time, optimality gap, and feasibility satisfaction (FS). Note that distributed data parallelism is used for NN training in the EPIGRIDS-5658 bus system with four NVIDIA RTX 4090 TI GPUs. For all the PGLib power systems, GraphOPF guarantees around 99% FS on average for the inequality and equality constraints of the generators, buses, and transmission lines. In addition, GraphOPF does not have unacceptably poor performance cases (used red letters in Table 2) while other baselines have quite a few cases; for example, PG-GNN exhibits eleven poor cases. This is because the baselines are only verified on small or medium-scale systems, indicating vulnerability to large-scale systems. More importantly, GraphOPF achieves high sample efficiency, leading to fast NN training. As shown in Table 2, DeepLDE achieves $1.04\times$ speedup in NN training time on average for all test cases compared to DC3, indicating the effectiveness of the advanced implicit layer over vanilla one. Notably, GraphOPF can further accelerate this process in $7.5\times - 10.81\times$ speedup by reducing the number of training data (denoted as N_{train}) at least ten times less than the baselines through EA-GNN and the advanced implicit layer.

5.2. Case Study of Real Power System: Korea Power Exchange

5.2.1. AC-OPF IN A SINGLE TIME INSTANCE

To verify the performance of GraphOPF more practically, we test GraphOPF on the real Korean power system (Korea-4492) with the help of KPX (the only ISO in Korea) in the same way in Section 5.1. In Table 3, GraphOPF shows up to $21.73\times$ speedup in computation and 0.03% of optimality gap, indicating that GraphOPF especially performs low optimality gap in real-world data. In addition, as shown in Table 4, GraphOPF outperforms DL-based optimization models in terms of NN training time, optimality gap, and FS. Surprisingly, GraphOPF only takes around five minutes for NN training in the Korea-4492 bus system. Thus, this clearly demonstrates that GraphOPF ensures high feasibility, fast training and near-optimal performance even in the real power system, which underpins the practicality of GraphOPF.

5.2.2. AC-OPF WITH TOPOLOGY CHANGES

The most challenging part of AI AC-OPF comes from the topology-fragile nature. To verify the adaptability against

Table 4. Computational time, Optimality gap, and Feasibility satisfaction comparison for Korea-4492 system (DL-based optimization models).

BUS SYSTEM	PERFORMANCE METRICS	PG-GNN	DEEPLDE	DC3	OPF-DNN	**GRAPHOPF**
KOREA-4492	$T_{train}(sec)/N_{train}$	12370/8000	14715/1000	78713/1000	24977/7800	**393**/100
	$\mathcal{F}_{opt}(\%)$	7.37	0.32	4.54	2.06	**0.03**
	$\mathbf{P}_g^{FS}(\%)$	**100**	99.82	99.85	**100**	99.83
	$\mathbf{Q}_g^{FS}(\%)$	**100**	**100**	**100**	**100**	**100**
	$\mathbf{V}^{FS}(\%)$	**100**	99.71	98.71	**100**	**100**
	$\mathbf{S}^{FS}(\%)$	99.91	**100**	**100**	99.98	**100**
	ACTIVE PF$^{FS}(\%)$	31.56	**99.99**	**99.99**	48.84	**99.99**
	REACTIVE PF$^{FS}(\%)$	8.50	**99.91**	**99.91**	78.39	**99.91**

Table 5. Performance results of GraphOPF for AC-OPF with topology changes (Korean power system).

BUS SYSTEM	$\mathcal{F}_{opt}$ (%)	$\mathbf{P}_g^{FS}$ (%)	$\mathbf{Q}_g^{FS}$ (%)	$\mathbf{V}^{FS}$ (%)	$\mathbf{S}^{FS}$ (%)	ACTIVE PF^{FS} (%)	REACTIVE PF^{FS} (%)	$T_{train}/T_{inf}/T_{sol}$ (sec)	N_{train}, N_{test}	$\lvert\mathcal{N}_g\rvert$	$\lvert\mathcal{N}\rvert$	$\lvert\mathcal{E}\rvert$
0100_49364MW	0.17	99.90	100	100	100	99.99	99.89	54/0.24/3.73	15,450	158	4487	5970
0200_48129MW	0.19	99.66	99.87	100	100	99.99	99.88	54/0.24/3.72	15,450	144	4486	5970
0300_47372MW	0.20	99.71	100	100	100	99.99	99.90	53/0.24/3.54	15,450	145	4486	5970
0400_47320MW	0.21	99.80	100	100	100	99.99	99.90	54/0.24/3.60	15,450	147	4486	5970
0500_47943MW	0.18	99.86	100	100	100	99.99	99.89	53/0.24/3.71	15,450	153	4485	5969
0600_50000MW	0.31	99.90	100	99.97	100	99.99	99.90	54/0.24/3.74	15,450	168	4486	5970
0700_53639MW	0.23	99.88	100	99.98	100	99.99	99.90	54/0.24/3.71	15,450	185	4524	6030
0800_55646MW	-	99.94	99.96	99.98	100	99.99	99.88	102/0.24/∞	15,450	203	4529	6029
0900_58364MW	0.11	99.82	100	99.97	99.99	99.99	99.89	105/0.25/4.39	15,450	212	4526	6027
1000_56322MW	0.22	99.90	99.98	99.96	99.99	99.99	99.90	57/0.25/4.16	15,450	214	4522	6018
1100_53234MW	0.25	99.94	99.73	100	100	99.99	99.90	103/0.24/4.62	15,450	193	4510	6006
1204_49120MW	-	99.92	100	99.95	100	99.98	99.90	102/0.24/∞	15,450	185	4514	6009
1304_49184MW	-	99.74	100	99.91	100	99.99	99.90	109/0.25/∞	15,450	192	4515	6011
1404_50855MW	-	99.96	100	99.91	100	99.99	99.90	54/0.24/∞	15,450	202	4516	6013
1503_50930MW	0.15	99.80	100	100	100	99.99	99.90	102/0.24/4.76	15,402	214	4522	6016
1603_52305MW	0.84	99.86	99.97	99.95	100	99.99	99.90	48/0.21/4.50	15,450	224	4527	6028
1703_54300MW	0.84	99.79	99.78	99.91	100	99.99	99.90	56/0.23/4.51	15,450	226	4537	6038
1803_56622MW	0.24	99.90	99.98	100	100	99.99	99.90	48/0.21/4.25	15,450	219	4536	6040
1903_58834MW	0.18	99.97	100	99.98	100	99.99	99.90	49/0.21/4.32	15,448	218	4538	6044
2004_58790MW	0.32	99.81	100	100	100	99.99	99.90	91/0.21/4.26	15,442	213	4538	6046
2104_57559MW	-	99.95	99.97	99.95	100	99.99	99.91	48/0.21/∞	15,450	203	4505	5994
2203_56467MW	0.51	99.75	99.98	100	100	99.99	99.91	49/0.21/4.38	15,450	199	4503	5992
2304_55651MW	0.29	99.80	100	100	100	99.99	99.88	103/0.23/4.39	15,450	184	4496	5986

topology change, we test GraphOPF on the Korean power system for entire one day of historical operation. Table 5 summarizes the performance of GraphOPF in terms of average NN training time, optimality gap, and FS. As can be seen, the numbers of generators (denoted as $|\mathcal{N}_g|$) also changes ranging from 144 to 226. Similarly, the number of buses and transmission lines change.

Considering GNN architecture can solve AC-OPF with topology changes (Falconer & Mones, 2022; Liu et al., 2022; Gao et al., 2023; Yang et al., 2024), we leverage transfer learning with frozen layers that only transfer the learnable parameters of EA-GNN layers (i.e., fine-tuning the flatten layer). In doing this, we find that NN training of GraphOPF can be further accelerated and thus finished less than one minute; surprisingly, only 15 training samples were enough for fine-tuning in the case of Korean power system. We also find that GraphOPF achieves 99% FS on average for the five cases where the mathematical solvers fail to converge, and thus the solving time denoted by T_{sol} is recorded as ∞ (red letters in Table 5). In addition, the cost gap (denoted by $\mathcal{F}_{opt}$) between the GraphOPF and the MATPOWER solver cannot be recorded for those five cases. This shows that GraphOPF can be an alternative method for critical situations during power system operation, which may happen frequently in future power systems due to the high penetration of DERs and load growth.

6. Ablation Studies

GraphOPF consists of EA-GNN and a hard constraint embedded layer. Especially, EA-GNN is designed with MEConv and ChebyConv layers, cooperatively capturing generalized topological features even with a small amount of data on large-scale power systems by additionally using edge features. To support this, ablation studies are conducted by eliminating an MEConv layer of EA-GNN (called M1 hereafter).

6.1. Performance Comparison Results

Figure 3 shows the performance results of optimality gap and FV for GOC-2312, GOC-4601, EPIGRIDS-5658, and Korea-4492 bus systems. Note that M1 fails in model train-

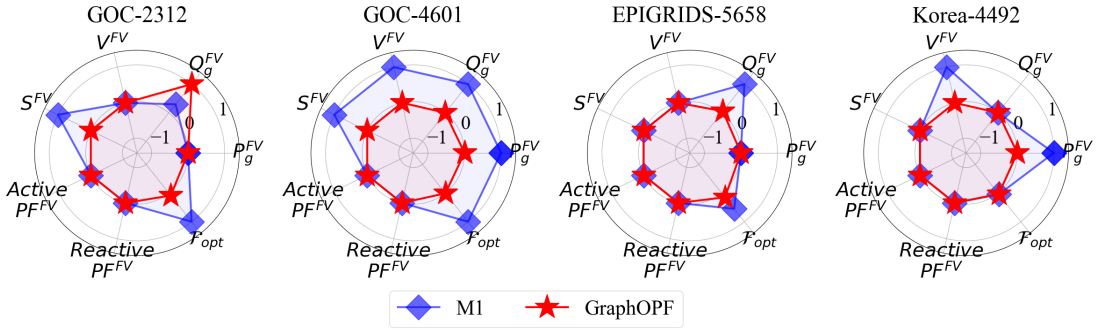

Figure 3. Radar charts for optimality gap and feasibility violation results of ablation studies.

Table 6. Feature drift results on two operating scenarios.

MODEL	NORMALIZED L2 NORM (%)	SINKHORN-WASSERSTEIN DISTANCE
FIXED TOPOLOGY		
EA-GNN	**0.0026**	**0.0023**
M1	6.89	7.39
M1-1000	5.21	5.52
TOPOLOGY CHANGES		
EA-GNN	**0.065**	**0.0024**
M1	13.83	13.48
M1-1000	16.32	11.80

Table 7. Two operating scenarios of AC-OPF in Korea power system.

SCENARIOS	LOAD (MW)	$\lvert\mathcal{N}_g\rvert$	$\lvert\mathcal{N}\rvert$	$\lvert\mathcal{E}\rvert$
FIXED TOPOLOGY				
CASE1	51408	182	4492	5969
CASE2	50064			
TOPOLOGY CHANGES				
CASE1	58834	218	4538	6044
CASE2	47320	147	4486	5970

ing for GOC-3970. As can be seen, GraphOPF outputs near-optimal and feasible solutions, showing that MEConv layer is effective for large-scale AC-OPF.

6.2. Feature Drift Analysis for Generalization Performance

Table 6 presents the generalization performance by measuring the feature drift on two operating scenarios of AC-OPF in Korea power system: maximum and minimum load cases in a fixed topology and topology changing cases with different loads. The details of operating scenarios are in Table 7. First, the features are extracted by EA-GNN layers from GraphOPF and ChebyConv layers from M1 for each case in the scenario, respectively. Then, feature drift is computed for case 1 and 2 by using graph mean pooling (Hamilton, 2020) and Sinkhorn-Wasserstein distance (Cuturi, 2013). For graph mean pooling, we use the ratio of normalized L2 norm to compute feature drift as $100 \times \frac{2\|\mathbf{F}^{(1)}-\mathbf{F}^{(2)}\|_2}{\|\mathbf{F}^{(1)}\|_2+\|\mathbf{F}^{(2)}\|_2}$, where $\mathbf{F}^{(1)}$ and $\mathbf{F}^{(2)}$ are graph-level features for case 1 and 2 of operating scenario. In Table 6, EA-GNN shows invariant feature drift, demonstrating robust generalization performance against diverse operating scenarios. Interestingly, increasing the number of training samples to tenfold for M1 (denoted as M1-1000) is helpful to reduce feature drift. Nevertheless, EA-GNN still outperforms M1-1000. Hence, it demonstrates that EA-GNN is enough to achieve generalization by using only 100 training samples, and fine-tuning on flatten layer while freezing EA-GNN layers can be effective on topology changes.

7. Conclusion

In this paper, we propose a fast, topology-adaptable, scalable, and unsupervised physics-informed graph learning framework called GraphOPF that considers large-scale AC-OPF. By combining EA-GNN with a hard constraint embedded layer, the NN training time of GraphOPF was substantially accelerated while satisfying the constraint feasibility. Extensive experimental results demonstrated that GraphOPF achieves feasible and near-optimal AC-OPF solutions in large-scale power systems, including the real Korean power system. Furthermore, GraphOPF shows remarkable performance in AC-OPF problems with topology changes by leveraging transfer learning in EA-GNN layers. This improves sample efficiency in GraphOPF to accelerate NN training further, reducing it less than one minute in Korean power system. In our future work, we will consider graph transfer learning in GraphOPF to build an OPF foundation model.

Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT under Grant RS-2025-02215243.

References

Babaeinejadsarookolaee, S., Birchfield, A., Christie, R. D., Coffrin, C., DeMarco, C., Diao, R., Ferris, M., Fliscounakis, S., Greene, S., Huang, R., et al. The power

grid library for benchmarking ac optimal power flow algorithms. *arXiv preprint arXiv:1908.02788*, 2019.

Bienstock, D. and Verma, A. Strong np-hardness of ac power flows feasibility. *Operations Research Letters*, 47 (6):494–501, 2019.

Carpentier, J. Contribution a l'etude du dispatching economique. *Bull. Soc. Fr. Elec. Ser.*, 3:431, 1962.

Chen, K., Bose, S., and Zhang, Y. Physics-informed gradient estimation for accelerating deep learning-based ac-opf. *IEEE Transactions on Industrial Informatics*, 2025.

Coffrin, C., Bent, R., Sundar, K., Ng, Y., and Lubin, M. Powermodels. jl: An open-source framework for exploring power flow formulations. In *2018 Power Systems Computation Conference (PSCC)*, pp. 1–8. IEEE, 2018.

Cuturi, M. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26, 2013.

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016.

Donti, P. L., Rolnick, D., and Kolter, J. Z. Dc3: A learning method for optimization with hard constraints. *arXiv preprint arXiv:2104.12225*, 2021.

Dunning, I., Huchette, J., and Lubin, M. Jump: A modeling language for mathematical optimization. *SIAM review*, 59(2):295–320, 2017.

Falconer, T. and Mones, L. Leveraging power grid topology in machine learning assisted optimal power flow. *IEEE Transactions on Power Systems*, 38(3):2234–2246, 2022.

Fioretto, F., Mak, T. W., and Van Hentenryck, P. Predicting ac optimal power flows: Combining deep learning and lagrangian dual methods. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 630–637, 2020.

Gao, M., Yu, J., Yang, Z., and Zhao, J. A physics-guided graph convolution neural network for optimal power flow. *IEEE Transactions on Power Systems*, 39(1):380–390, 2023.

Hamilton, W. L. *Graph representation learning*. Morgan & Claypool Publishers, 2020.

Hien, D. T., Song, K., Kim, K., and Kim, H. Alternative learning architecture for solving ac-opf via supervised relaxation and cross encoder. In *NeurIPS Workshop on GPU-Accelerated and Scalable Optimization*, 2025.

Huang, W., Pan, X., Chen, M., and Low, S. H. Deepopf-v: Solving ac-opf problems efficiently. *IEEE Transactions on Power Systems*, 37(1):800–803, 2021.

IEA. Electricity 2024. https://www.iea.org/reports/electricity-2024, 2024.

Jia, Y., Bai, X., Zheng, L., Weng, Z., and Li, Y. Convopf-dop: A data-driven method for solving ac-opf based on cnn considering different operation patterns. *IEEE Transactions on Power Systems*, 38(1):853–860, 2022.

Kim, M. and Kim, H. Unsupervised deep lagrange dual with equation embedding for ac optimal power flow. *IEEE Transactions on Power Systems*, 40(1):1078–1090, 2024.

Liu, S., Wu, C., and Zhu, H. Topology-aware graph neural networks for learning feasible and adaptive ac-opf solutions. *IEEE Transactions on Power Systems*, 38(6): 5660–5670, 2022.

Pan, X., Chen, M., Zhao, T., and Low, S. H. Deepopf: A feasibility-optimized deep neural network approach for ac optimal power flow problems. *IEEE Systems Journal*, 17(1):673–683, 2022.

Park, S., Chen, W., Mak, T. W., and Van Hentenryck, P. Compact optimization learning for ac optimal power flow. *IEEE Transactions on Power Systems*, 39(2):4350–4359, 2023.

Shin, S., Anitescu, M., and Pacaud, F. Accelerating optimal power flow with gpus: Simd abstraction of nonlinear programs and condensed-space interior-point methods. *Electric Power Systems Research*, 236:110651, 2024.

Xu, K., Dai, X., and Qiu, L. Opformer: Real-time optimal power flow with cnn-based transformer. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. IEEE, 2025.

Yang, M., Qiu, G., Liu, J., Liu, Y., Liu, T., Tang, Z., Ding, L., Shui, Y., and Liu, K. Topology-transferable physics-guided graph neural network for real-time optimal power flow. *IEEE Transactions on Industrial Informatics*, 20(9): 10857–10872, 2024.

Zeng, S., Kim, Y., Ren, Y., and Kim, K. Qcqp-net: Reliably learning feasible alternating current optimal power flow solutions under constraints. In *6th Annual Learning for Dynamics & Control Conference*, pp. 1539–1551. PMLR, 2024.

Zimmerman, R. D., Murillo-Sánchez, C. E., and Thomas, R. J. Matpower: Steady-state operations, planning, and analysis tools for power systems research and education. *IEEE Transactions on power systems*, 26(1):12–19, 2010.