Physics Informed Transformer Architecture for Power Grid Simulation

Vishnu Sankar Manivasakan
Georgia Institute of Technology
vmanivasakan3@gatech.edu

Ilias Baali
Georgia Institute of Technology
ibaali3@gatech.edu

Cole Johnson
Georgia Institute of Technology
cjohnson498@gatech.edu

Priyanshu Mehta
Georgia Institute of Technology
pmehta305@gatech.edu

Abstract

This paper presents an approach to power grid simulation using Transformers combined with physics-informed neural networks (PINNs). Traditional simulation methods, such as Newton-Raphson and Interior Point Methods (IPOPT), face challenges in handling complexity of modern power grids, particularly under time constraints for real-time operations. By leveraging Transformers + PINNs, the proposed method models power grid configurations more efficiently, while incorporating essential physical laws like Kirchhoff's Current Law and Joule's Law. The model aims to predict power flows and node voltages, offering a computationally efficient solution that balances physical accuracy with speed. Evaluation of the model using various metrics, including Mean Absolute Error (MAE) and physical law compliance, demonstrates its robustness across different grid topologies, including out-of-distribution (OOD) scenarios. This work provides a promising pathway for improving the scalability and reliability of power grid simulations in dynamic, real-time environments.

1. Introduction

Electric power transmission networks operate under extremely high currents and voltages, making them susceptible to overload and stability problems. To ensure secure and reliable operation, system operators perform computationally intensive real-time simulations that evaluate risk and inform mitigation strategies. Conventional approaches typically employ nonlinear, non-convex solvers (such as Newton–Raphson methods or interior-point algorithms (IPOPT)) whose runtimes impose practical limits on scenario exploration. For example, a single-timestamp power-flow simulation of the French grid requires approximately 100 ms [2]. In day ahead security assessments, forecasting over a 24-hour horizon refreshed every 15 minutes involves simulating up to 9000 contingencies (90% of the 10,000 possible node failures) within each quarter-hour window. As energy transition uncertainties drive finer time discretizations, potentially to 5-minute intervals, the same 5-minute interval permits only <3000 contingency evaluations, leaving the majority of critical failure modes unexamined. Consequently, the execution time of the solver emerges as a bottleneck that constrains both the breadth and depth of the risk analysis. With power grid topologies and operational regimes becoming increasingly complex, there is an urgent need for faster but equally reliable simulation techniques capable of supporting comprehensive contingency screening in near real time.

2. Related Works

The challenge of simulating power grid operations efficiently has received significant attention in recent years. Traditional methods such as the Newton-Raphson and Interior Point Methods (IPOPT) have long been used to model power flows, but they face challenges in scalability and real-time applications as the complexity of modern grids increases. In particular, the need for faster simulations while maintaining accuracy has led to the exploration of machine learning-based solutions.

Physics-Informed Neural Networks (PINNs) [8] have emerged as a promising technique for incorporating physical laws into deep learning models. These methods directly integrate conservation laws into the loss functions, which helps in enforcing physical constraints during model training. Notable works using PINNs for power systems include Huang and Wang [3], [7], which discuss the use of PINNs in power system optimization, and Lei et al. [6], who apply data-driven methods for optimal power flow using PINNs.

Graph Neural Networks (GNNs) have also gained attention for their ability to model complex network structures, making them a natural choice for power grid simulations. Research by Talebi and Zhou [9] explored the use of GNNs for efficient power flow predictions. Similarly, PINCO [10], which combines GNNs with PINNs, has shown promise in AC optimal power flow problems by incorporating nonlinear constraints into the model. However, the complexity in building this model is prohibitive hence has limited research exploration.

The approach presented in this paper incorporates key ideas from [10], and apply those physics constraints to an existing transformer model [4] in the form of penalty terms in the loss function.

3. Methodology

We frame the task of rapid power-flow prediction as a supervised regression problem in which an input vector $x \in \mathbb{R}^d$, comprising generator outputs, load demands, and network topology features, is mapped to a target vector $y \in \mathbb{R}^k$ of branch currents, line power flows, and bus voltages. Our goal is to learn a function

$$f_\theta : \mathbb{R}^d \to \mathbb{R}^k, \qquad f_\theta(x) \approx y,$$

that approximates the output of traditional numerical solvers at a fraction of the computational cost. We have adapted the transformer model from [4], but have made significant changes to it so that it integrates well with the PINNs approach.

To capture the complex, long-range dependencies inherent in electrical networks, we employ a transformer-based architecture. The input vector x is first partitioned into attribute groups (e.g., generator states, load profiles, topology indicators) and each group is projected via a learnable linear mapping into a d_{model}-dimensional embedding. These per-attribute embeddings are concatenated into a sequence of length N, yielding an initial tensor $E_0 \in \mathbb{R}^{N \times d_{\text{model}}}$. We add sinusoidal positional encodings to E_0 so that

$$\text{PE}_{(pos,2i)} = \sin\Big(\frac{pos}{10000^{2i/d_{\text{model}}}}\Big),$$

$$\text{PE}_{(pos,2i+1)} = \cos\Big(\frac{pos}{10000^{2i/d_{\text{model}}}}\Big),$$

thereby providing the model with information about the fixed ordering of attributes.

Our encoder consists of L identical layers, each containing a multi-head self-attention (MHSA) block followed by a position-wise feed-forward network (FFN). In the MHSA, queries Q, keys K, and values V are obtained via learned linear transformations of the input. Each head computes

$$\text{Attention}(Q, K, V) \; = \; \text{softmax}\big(QK^\top / \sqrt{d_k}\big)\, V,$$

where $d_k = d_{\text{model}}/\text{num\_heads}$, and the head outputs are concatenated and linearly projected back to d_{model}. The FFN comprises two dense layers with a GELU activation in between. Both the MHSA and FFN employ residual connections followed by layer normalization.

Beyond standard self-attention, we incorporate two specialized layers. An attention-pooling layer applies MHSA to aggregate sequence features into a single vector via mean pooling across the sequence dimension, followed by an FFN with layer normalization. A cross-attention layer enables one attribute subset (e.g., generator embeddings) to attend to another (e.g., line embeddings), thereby modeling interactions between different component types. After the final encoder layer, we select the last token (with x[:, -1, :]), across the whole sequence, and pass it through a final dense (linear) layer to map to the output dimension ($\mathbb{R}^k$).

We normalized all continuous features and targets to zero mean and unit variance using a standard scaler. We have used the AdamW optimizer, a learning rate schedule with a linear warmup phase followed by exponential decay. Dropout with a rate of 0.1 is applied within the feedforward (FFN) layers of each Transformer encoder block to improve generalization. After the final encoder layer, the model selects the last token from the sequence output before passing it through a final linear layer to generate predictions.

To justify our design choices, we implemented and compared three alternative models: the baseline LeapNet model [2], a pure transformer model, and a physics-informed transformer model. The physics-informed transformer model consistently outperformed the other two, particularly in large networks, by naturally capturing the physics baked into the system.

The entire implementation uses TensorFlow 2.10.16, with custom layers for MHSA, and specialized attention, and integrates with the LIPS framework for data handling, benchmarking, and reproducibility. All code, hyperparameters, and training scripts are version controlled and containerized, with fixed random seeds for NumPy and TensorFlow to ensure reproducibility of results.

3.1. Physics Loss functions

There are eight physics conditions that could be incorporated into the training process. Some of them are penalty terms in the loss function, while others are direct modifications to the model's output. Most of these loss functions are directly adapted from [5], and remodeled in TensorFlow, with which all the codes for this paper was written in.

The first physics constraint (**P1**) is non-negative current magnitudes. This ensures that the predicted magnitudes for current flows at both the origin a_{or_pred} and extremity a_{ex_pred} of power lines are non-negative. We check for $a_{or\_pred} < 0$ and $a_{ex\_pred} < 0$, and we implemented a penalty term by taking the mean wherever the condition is violated.

The second physics constraint (**P2**) is non-negative voltage magnitudes. This ensures that the predicted magnitudes for voltage magnitudes at both the origin v_{or_pred} and extremity v_{ex_pred} of power lines are non-negative. We check for $v_{or\_pred} < 0$ and $v_{ex\_pred} < 0$, and implemented a penalty term by taking the mean wherever the condition is violated.

The third physics constraint **P3** is non-negative power flow sum. This ensures that the sum of active power predicted flowing out of the origin p_{or_pred} and into the extremity p_{ex_pred} is non-negative. $p_{or\_pred} + p_{ex\_pred}$ represents the active power loss on the power lines and this term should always be positive (or zero for an ideal line). We calculate the inequality $p_{or\_pred} + p_{ex\_pred} < 0$ and added a penalty of mean of 1s wherever the condition gets violated.

The fourth physics constraint **P4** is zero flow on disconnected lines. This enforces disconnected power lines to have zero absolute current and power. The indices corresponding to disconnected power lines are identified, then the sum of absolute currents a_{or_pred}, a_{ex_pred} and absolute powers p_{or_pred}, p_{ex_pred} are calculated at these indices. The penalty is the mean of this sum and it is driven to zero. $P4 = \text{torch.mean}(a_{or\_ex\_pred\_abs\_sum} + p_{or\_ex\_pred\_abs\_sum})\text{.float()}$.

The fifth physics constraint **P5** is energy loss bounds reasonableness. Wherever the total active power loss across all lines $p_{or\_ex\_pred\_sum}$ behaves unexpectedly relative to the total power generation $prod_{p\_data\_sum}$ we add penalty term. Behaving unexpectedly could mean the values are negative or excessively high. This drives the model to find solutions where the relative energy loss is small and positive. We evaluate the ratio energy_loss = $p_{or\_ex\_pred\_sum}/prod_{p\_data\_sum}$ and implemented different penalty functions based on the range of energy_loss. For instance, we penalized heavily if energy_loss $<$ 0 or energy_loss $>$ 0.04, and penalized moderately if 0 $<$ energy_loss $<$ 0.005).

The sixth and seventh physics constraints are the KKT conditions of power balance, inspired from [1], and implemented in [5]. These constraints enforce power balance at each substation in the grid. Making sure the model is consistent with Kirchhoff's Current Law and optimal power flow conditions. These are implemented using the famous Karush-Kuhn-Tucker (KKT) conditions. The final constraint **P8** is to make sure the power load is consistent according to Joule's law. It checks if the predicted active power loss calculated as the difference between origin and extremity power flows, is consistent with the loss calculated using predicted currents and line resistances with the formula I^2R.

The biggest hurdle in implementing all the constraints is that the model takes an enormous amount of time to train. With the limited resources at their end it was impossible to satisfy all the constraints. Though we remodeled and implemented all the constraints, we finally decided to not include constraints **P6**, **P7** and **P8** in the final loss function. Because these three constraints take the longest time to converge below a reasonably lower threshold.

The final formulae for all the physics constraints are given below:

$$P1 = \mathbb{E}_{i,j}\left[1\left[a_{ij}^{\text{or}} < 0 \vee a_{ij}^{\text{ex}} < 0\right]\right]$$

$$P2 = \mathbb{E}_{i,j}\left[1\left[v_{ij}^{\text{or}} < 0 \vee v_{ij}^{\text{ex}} < 0\right]\right]$$

$$P3 = \mathbb{E}_{i,j}\left[1\left[p_{ij}^{\text{or}} + p_{ij}^{\text{ex}} < 0\right]\right]$$

$$P4 = \mathbb{E}_{(i,j)\text{ s.t. }s_{ij}=1}\left[|a_{ij}^{\text{or}}| + |a_{ij}^{\text{ex}}| + |p_{ij}^{\text{or}}| + |p_{ij}^{\text{ex}}|\right]$$

For $P5$

$$e_i = \frac{\sum_{j=1}^{L} 1(p_{ij}^{\text{ex}} + p_{ij}^{\text{or}})}{\sum_k p_{ik}^{\text{gen}}}$$

Penalty function:

$$f(e_i) = \begin{cases} 200(e_i - 0.04) & \text{if } e_i > 0.04 \\ 200(0.005 - e_i) & \text{if } 0 < e_i < 0.005 \\ 500(0.005 - e_i) & \text{if } e_i < 0 \\ 0 & \text{otherwise} \end{cases}$$

Then

$$P_5 = \mathbb{E}_i\left[f(e_i)\right]$$

Let the average predicted current magnitude for line j in sample i be

$$\bar{a}_{ij} = \frac{a_{ij}^{\text{or}} + a_{ij}^{\text{ex}}}{2}$$

Let the scaled power difference term be

$$P8_{,\text{term1}_{ij}} = (p_{ij}^{\text{or}} + p_{ij}^{\text{ex}}) \times 17.498$$

Let the I^2R term be

$$P_{8,\text{term2}_{ij}} = (\bar{a}_{ij})^2 \times R_{ij}$$

Then

$$P8 = \left|\mathbb{E}_{i,j}\left[P_{8,\text{term1}_{ij}} - P8_{8,\text{term2}_{ij}}\right]\right|$$

where

- N = batch size
- $L = 186$ = number of lines
- $a^{\text{or}}, a^{\text{ex}}, p^{\text{or}}, p^{\text{ex}}, v^{\text{or}}, v^{\text{ex}}$ are predicted tensors (shape $N \times L$)
- s is the `line_status_data` tensor (shape $N \times L$), where $s_{ij} = 1$ if line j is disconnected in sample i
- p^{gen} is the `prod_p_data` tensor (shape $N \times$ num_gen)

- R is the `RBus_selected` tensor (shape $N \times L$), resistance for each line per sample
- $\mathbb{E}_{i,j}[\cdot]$ = mean over all samples i and lines j
- $\mathbb{E}_{i}[\cdot]$ = mean over all samples i
- $\mathbb{E}_{(i,j)\text{ s.t. }C}[\cdot]$ = mean over all sample/line pairs (i, j) satisfying condition C
- $1[\cdot]$ = indicator function (1 if condition true, 0 otherwise)

The final loss function is

$$L = 10MSE + P_1 + P_2 + P_3 + P_4 + P_5$$

In the pure Transformer model, MSE is computed directly between the raw network output and target values. No physics modifications are applied before loss calculation, and the optimizer focuses solely on minimizing this MSE. In the Pysics-Informer Transformer model the MSE is calculated after applying physics-based modifications which are the KKT adjustments and disconnected line zeroing to the Transformer's predictions. So the basic formula for MSE remains the same in both cases, in the PINN, MSE is part of a multi-objective loss after adjustments.

4. Data

The dataset used in this project is sourced from the LIPS (Latent Intervention for Power Systems) benchmarking framework, which provides standardized datasets for power grid simulations. This dataset is designed specifically for evaluating neural surrogate models in power systems and includes various power grid topologies with dynamic perturbations.

The dataset contains four primary subsets: a training set, a validation set, and two test sets. The training set includes 300,000 samples, while the validation set comprises 100,000 samples. The first test set, representing in-distribution data, contains 100,000 samples, and the second test set, which is out-of-distribution (OOD) and designed to evaluate the model's generalization capabilities, includes 200,000 samples. This design ensures comprehensive evaluation across diverse grid scenarios and topologies.

Each sample in the dataset includes key attributes such as production and load powers, voltages, and power grid topology attributes, including the status of each power line and node connections. These attributes are essential for the prediction of power flows and node voltages in the simulation. The dataset also includes optional physical variables that could be used for physics-based modeling.

Preprocessing steps include normalization of the input features, ensuring that the data adheres to consistent scales for neural network models. The dataset does not undergo extensive augmentation due to the domain-specific nature of the task, but random perturbations in grid topology are introduced during training to improve the model's robustness.

Known limitations of the dataset include potential biases in grid topologies, as it primarily represents a set of common power grid structures. Additionally, while the dataset provides a good representation of power grid scenarios, outlier or rare topologies may not be well-represented, which can affect the model's ability to generalize to highly atypical grids.

5. Experiments

Six different experiments were performed in this project. The first experiment was on varying different parameters and hyper-parameters of the transformer model itself. This kind of experimentation has several sub-parts to it as listed below

- Learning Rate and Schedule
- Gradient Clipping vs. No Gradient Clipping
- Model Architecture
- Optimizer choice

5.1. Learning Rate and Schedule

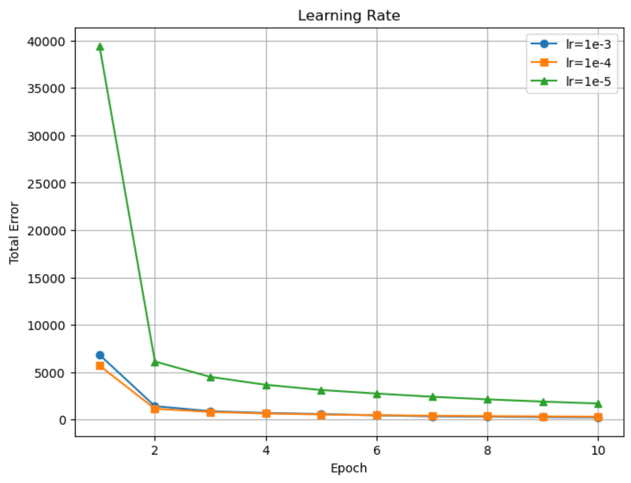

Figure 1. Effect of different learning rates

From the plot, we can observe that a learning rate of $1e^{-5}$ has the slowest converging rate. Initially $1e^{-4}$ seems to be performing better than $1e^{-3}$ but within 10 epochs $1e^{-3}$ takes better strides in reducing the error. This calls for a dynamic learning rate approach. But as we know we might require smaller learning rates during the final phase of training for more accuracy and better stability. So we implemented learning rate schedule with warmup phase followed by exponential decay.We used *tf.keras.optimizers.schedules.ExponentialDecay* for the decay part and combined it with a linear warmup phase using a custom schedule class. We set the $peak\_learning\_rate$ as $1e^{-3}$ as it initially converges faster. We also tried running with a learning rate of $1e^{-2}$ but the training was too unstable and the loss function was 100 order of magnitudes higher relatively.

5.2. Gradient Clipping vs. No Gradient Clipping

An experiment with and without gradient clipping was also performed. It was noticed that gradient without clipping initially had better convergence but then it had stability issues after 30 epochs once the algorithm started gravitating towards a minimum. The reason behind this observation is initially the MSE error is four orders of magnitude higher than the physics constraints $(P_1, P_2, P_3, P_4, P_5)$ so the model makes use of large gradients to quickly reduce the MSE error. But once the MSE error is about the two orders of magnitude as the physics error, the slightly larger gradients from MSE error caused the optimizer to overshoot, and the error in physics constraints started increasing. Specifically the P_3 error was overshooting. With gradient clipping this issue was however not observed and the model stabilizes the training process. Although with gradient clipping the training process was slower, the stability it provided outweighed the drawback.

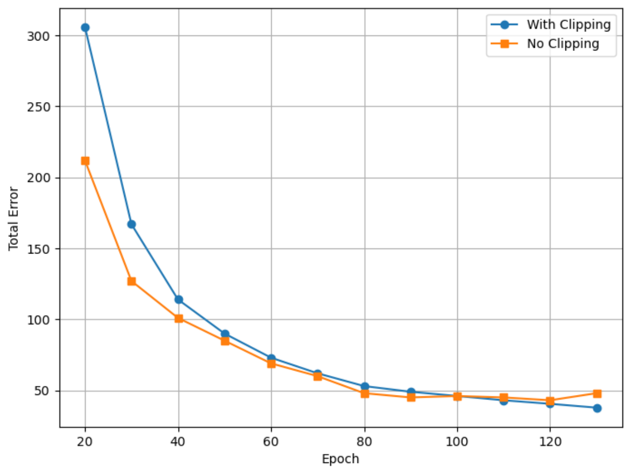

Figure 2. With and without Gradient Clipping

5.3. Optimizer Choice

The comparison between AdamW and Adam clearly shows that AdamW outperforms Adam. The common reason explaining this phenomena is that AdamW decouples weight decay from the gradient updates. In standard Adam, weight decay is used in the gradient calculation which slightly distorts the optimization path and slows down the convergence. Since AdamW applies weight decay separately it seems to work better in generalization which is why error curve AdamW is slightly lower in fig:3

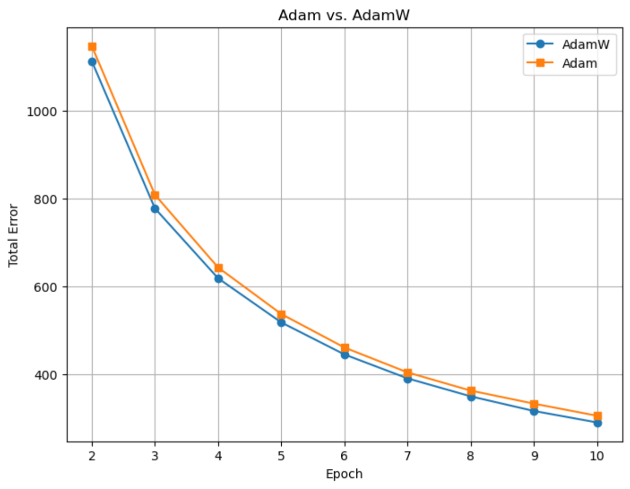

Figure 3. Optimizer comparison at lr=$1e^{-4}$

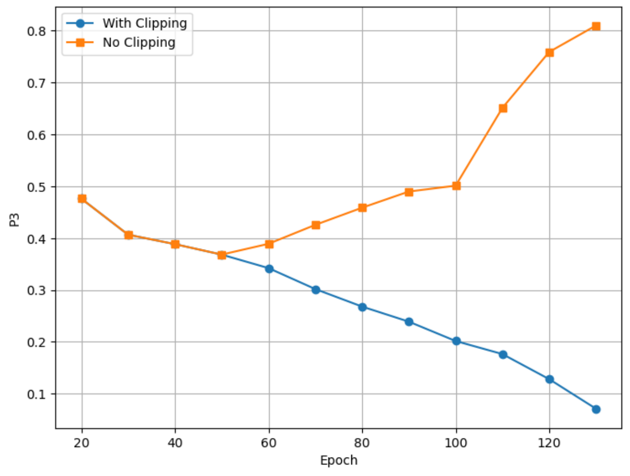

Figure 4. Effect of Gradient Clipping with P3

5.4. Effect of Number of Attention Heads

In this experiment, we compared the effect of using 2 versus 4 attention heads in the Transformer model. The 2-head model has lower total error during early training phase. The reason could be that with fewer heads it has fewer parameters and generalizes faster. The 4-head model has higher number of parameters an requires more training epochs until it starts getting better. In fig:5, we can see that after about 25 epochs, the model with 4 heads is able to drive the errors much lower than the one with 2 heads.

6. Results

The Physics-Informed Transformer model, with about 1,534,188 parameters, was trained for 500 epochs to reach

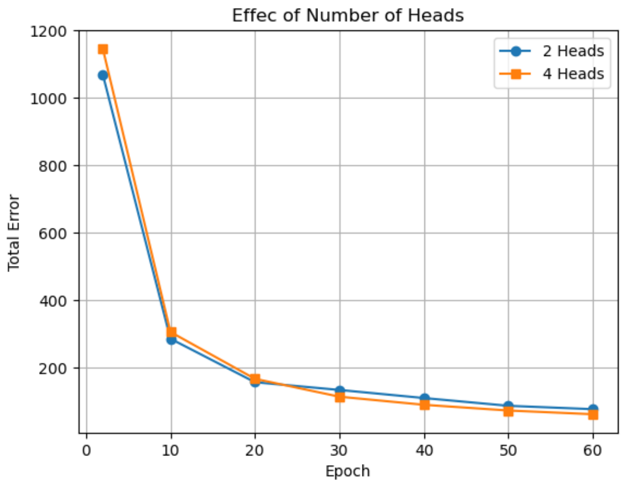

Figure 5. Effect of Number of Attention Heads

convergence. It performs much better than the baseline LeapNet model. LeapNet uses dense layers that operate locally on node features, but Transformers with self-attention allow the model to learn long-range relationships between all components in the power grid. This is crucial for electrical systems where changes in one part of the network can have far-reaching effects elsewhere. Adding PINN losses in terms of penalties for violating physical laws like KCL, energy conservation, and positivity constraints guides the model to produce physically meaningful outputs, even for unseen perturbations (OOD cases).

Category	**Metric**	**Violation**
Current Positivity	a_or	3.86%
	a_ex	4.35%
Voltage Positivity	v_or	1.79%
	v_ex	1.36%
Loss Positivity	loss_criterion	41.69%
Disconnected Lines	all	100.00%
Physical Laws	Energy Loss	25.11%
	Global Conservation	99.98%
	Local Conservation	95.20%
	Joule Law	96.29%

Table 1. Evaluation metrics on the OOD test dataset (**LeapNet**)

However, since we did not incorporate the global joule heating constraint in the final model, the Joule Law is violated to a high degree but not as high as the LeapNet model. The global conservation error is about 33.26%, and this is expected as it directly corresponds to **P3** which was not entirely driven to zero. Since it was given a lower weight compared to the MSE, the optimizer prioritized it over **P3**. The local conservation is also high because it direclty correlates with **P6** and **P7** which were not implemented as well.

Category	**Metric**	**Violation**
Current Positivity	a_or	1.26%
	a_ex	2.39%
Voltage Positivity	v_or	0.61%
	v_ex	0.85%
Loss Positivity	loss_criterion	0.015%
Disconnected Lines	all	0.0%
Physical Laws	Energy Loss	13.12%
	Global Conservation	33.26%
	Local Conservation	60.81%
	Joule Law	79.29%

Table 2. Evaluation metrics on the OOD test dataset (**Pysics-Informer Transformers**)

7. Conclusion

LeapNet or pure Transformer model, or any model for a matter of fact does not directly encode any physics laws in them, and hence they are often not sufficient to meet real physics constraints. Data alone cannot teach physics to a model. That's exactly why we need PINN like models, and we incorporate it in a Transformer approach which simulates the physics-based reasoning throughout training, leading to better generalization and more reliable behavior in physically perturbed or unseen scenarios. Overall, combining a globally expressive architecture (Transformer) with strong physics regularization (PINN) yields models that are both more accurate and more robust. However, they do come at a very high training cost. On average, the training cost is **4-5x** higher than other models even without including all the constraints. If one were to implement the local constraints **P6** and **P7** the training cost could potentially shoot-up. Though increased computational burden is expensive, the substantial improvements in physical fidelity and out-of-distribution robustness make PINN-augmented (Transformer) models a highly promising approach for critical applications where safety, reliability, and physics-consistency are non-negotiable.

8. Team Contributions

Please refer to the next page

References

[1] Hao Chen, Gonzalo E Constante Flores, and Can Li. Physics-informed neural networks with hard linear equality constraints. *Computers & Chemical Engineering*, 189:108764, 2024. 3

[2] Benjamin Donnot, Balthazar Donon, Isabelle Guyon, Zhengying Liu, Antoine Marot, Patrick Panciatici, and Marc Schoenauer. Leap nets for power grid perturbations. *arXiv preprint arXiv:1908.08314*, 2019. 1, 2

[3] Bin Huang and Jianhui Wang. Applications of physics-informed neural networks in power systems-a review. *IEEE Transactions on Power Systems*, 38(1):572–588, 2022. 1

[4] Nina Inalgad. Powergrid self-attention. `https://github.com/Ninalgad/powergrid-self-attention`, 2023. Accessed: 2025-04-13. 2

[5] Jiang Jiang. Leap-pinn: Learning effective abstract physics through self-attention. `https://github.com/JiangJiang65/LEAP-PINN`, 2022. Accessed: 2025-04-13. 2, 3

[6] Xingyu Lei, Zhifang Yang, Juan Yu, Junbo Zhao, Qian Gao, and Hongxin Yu. Data-driven optimal power flow: A physics-informed machine learning approach. *IEEE Transactions on Power Systems*, 36(1):346–354, 2020. 1

[7] Rahul Nellikkath and Spyros Chatzivasileiadis. Physics-informed neural networks for ac optimal power flow. *Electric Power Systems Research*, 212:108412, 2022. 1

[8] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019. 1

[9] Seyedamirhossein Talebi and Kaixiong Zhou. Graph neural networks for efficient ac power flow prediction in power grids. *arXiv preprint arXiv:2502.05702*, 2025. 1

[10] Anna Varbella, Damien Briens, Blazhe Gjorgiev, Giuseppe Alessio D'Inverno, and Giovanni Sansavini. Physics-informed gnn for non-linear constrained optimization: Pinco a solver for the ac-optimal power flow. *arXiv preprint arXiv:2410.04818*, 2024. 2

Student Name	Contributed Aspects	Details
Vishnu Sankar Manivasakan	PINN Integration with Transformer, Literature Review, Model Evaluation, Model Optimization	Implemented the core Transformer + PINN architecture and optimized it for power grid simulation tasks
Ilias Baali	Model Design, Implementation of LeapNet Architecture	Implemented the core LeapNet (baseline) architecture and optimizer it for power grid simulation tasks, contributed to data gathering and analysis of relevant research, along with evaluating model performance and results
Cole Johnson	PINN Integration with Transformer, Model Optimization	Assisted with integrating the physics-based constraints into the neural network, and worked on optimizing the model's computational efficiency.
Priyanshu Mehta	Data Collection, Data Preprocessing, Performance Evaluation, Report Writing	Handled data preprocessing, evaluated model performance using various metrics, and contributed to the final report and documentation.

Table 3. Contributions of team members.