Learning From Less

Optimization Proxies under Time-Sample Constrained Settings

Parikshit Pareek
IIT Roorkee & T-5, LANL

Work done with Kaarthik, Deep and Sidhant

pareek@ee.iitr.ac.in

The ML's Time Equation

$$T_{Total} = T_{Data} + T_{Training} + T_{Prediction}$$

T_{Data}	$T_{Training}$	$T_{Prediction}$
Training Data Generation Validation Data Generation	Hyperparameter Optimization Variational Inference	Prediction Time Validation Time

The Problem

How to Overcome Constraints on Training Time and Data?

Time and Data Constraints

- **Optimization proxies must be trained efficiently within strict time constraints**
  - Operational time constraints
  - Speed requirements for large-scale simulations

Time and Data Constraints

- **Optimization proxies must be trained efficiently within strict time constraints**
  - Operational time constraints
  - Speed requirements for large-scale simulations
- **Obtaining training data is time-consuming or not feasible**
  - Large-scale problem solutions produce only a single data point
  - Limited data related to rare or less probable events

Time and Data Constraints

- **Optimization proxies must be trained efficiently within strict time constraints**
  - Operational time constraints
  - Speed requirements for large-scale simulations
- **Obtaining training data is time-consuming or not feasible**
  - Large-scale problem solutions produce only a single data point
  - Limited data related to rare or less probable events
- **Physical systems require 'some' guarantees on learning quality**
  - Vanilla statistical validation demands a large validation dataset

`Hoeffding's Inequality:` $\varepsilon \propto 1/\sqrt{N}$

Current DNN Proxies: All Permutations & Combinations

- Supervised DNN models with different architectures and settings
  - `Supervised Loss:` Mean Square Error (MSE), Mean Absolute Error (MAE)
  - `Supervised Losses + Penalty :` MSE/MAE + λ[Constraint Violation]
  - `Partial Predict + Power Flow:` Predict P_g, Q_g & solve power flow for $|V|$, θ

- Semi-Supervised DNN models
  - Use Power Flow Jacobian during training
  - Use Power Flow Jacobian at correction stage
  - Last layer– `sigmoid` based output clipping for correction

- Self-Supervised/Unsupervised DNN models
  - Loss Function: Cost + λ[Constraint Violation]
  - Update weights λ in primal-dual fashion

Issues

Large Labeled Data, Large Training & Prediction Time,

Bayesian for Rescue under Low Data

- **What are Bayesian Neural Networks (BNNs)?**
  - BNNs treat weights as probability distributions instead of fixed values
  - Provide a probability distribution over outputs instead of single deterministic predictions.

Bayesian for Rescue under Low Data

- **What are Bayesian Neural Networks (BNNs)?**
  - BNNs treat weights as probability distributions instead of fixed values
  - Provide a probability distribution over outputs instead of single deterministic predictions.
- **Why are BNNs better than DNNs for low data?**
  - Incorporate **prior knowledge** and effectively model uncertainty in data and parameters.
  - Prevent overfitting: **Weight Distribution**, **Regularization via Priors**, **Posterior Averaging**

Bayesian for Rescue under Low Data

- **What are Bayesian Neural Networks (BNNs)?**
  - BNNs treat weights as probability distributions instead of fixed values
  - Provide a probability distribution over outputs instead of single deterministic predictions.
- **Why are BNNs better than DNNs for low data?**
  - Incorporate **prior knowledge** and effectively model uncertainty in data and parameters.
  - Prevent overfitting: **Weight Distribution**, **Regularization via Priors**, **Posterior Averaging**
- **Advantages over DNNs:**
  - Separates two types of uncertainty:
    - **Epistemic uncertainty**: Uncertainty in model parameters $p(w|D)$.
    - **Aleatoric uncertainty**: Noise inherent in data $p(y|x, w)$.
  - Assigns high uncertainty to points far from the training set, avoiding overconfident predictions.

More on BNN

How do BNNs work?

- Use Bayes' theorem to update weight distributions:

$$p(w|D) = \frac{p(D|w)\,p(w)}{p(D)} \quad w : \text{weights}, D : \text{data}, p(w) : \text{prior}, p(D|w) : \text{likelihood}.$$

Core Equations of BNNs:

- Prediction distribution:

$$p(y|x, D) = \int p(y|x, w)\,p(w|D)\,dw$$

- Approximated using Monte Carlo or variational techniques.
- Training a BNN is slower than Training a DNN

Our Problem, Target, Motivation and Idea

- **Standard Constrained Optimization Problem**

$$\min_{\mathbf{y}} \quad c(\mathbf{y}) \tag{1a}$$

$$\text{s.t.} \quad g(\mathbf{x}, \mathbf{y}) = 0 \tag{1b}$$

$$h(\mathbf{x}, \mathbf{y}) \leqslant 0 \tag{1c}$$

$$\mathbf{x} \text{ is given (input vector)}$$

Our Problem, Target, Motivation and Idea

- **Standard Constrained Optimization Problem**

$$\min_{\mathbf{y}} \quad c(\mathbf{y}) \tag{1a}$$
$$\text{s.t.} \quad g(\mathbf{x}, \mathbf{y}) = 0 \tag{1b}$$
$$h(\mathbf{x}, \mathbf{y}) \leqslant 0 \tag{1c}$$
$$\mathbf{x} \text{ is given (input vector)}$$

- **Target:**
  Develop a **Fast Evaluating** proxy such that: $\mathbf{y} = \mathcal{M}_w(\mathbf{x})$

Our Problem, Target, Motivation and Idea

- **Standard Constrained Optimization Problem**

$$\min_{\mathbf{y}} \quad c(\mathbf{y}) \tag{1a}$$
$$\text{s.t.} \quad g(\mathbf{x}, \mathbf{y}) = 0 \tag{1b}$$
$$h(\mathbf{x}, \mathbf{y}) \leqslant 0 \tag{1c}$$
$$\mathbf{x} \text{ is given (input vector)}$$

- **Target:**
  Develop a **Fast Evaluating** proxy such that: $\mathbf{y} = \mathcal{M}_w(\mathbf{x})$
- **Motivation:**
  Training-validation data collection is **Expensive** & need to **Train** $\mathcal{M}_w(\mathbf{x})$ **Fast**

Our Problem, Target, Motivation and Idea

- **Standard Constrained Optimization Problem**

$$\min_{\mathbf{y}} \quad c(\mathbf{y}) \tag{1a}$$
$$\text{s.t.} \quad g(\mathbf{x}, \mathbf{y}) = 0 \tag{1b}$$
$$h(\mathbf{x}, \mathbf{y}) \leqslant 0 \tag{1c}$$
$$\mathbf{x} \text{ is given (input vector)}$$

- **Target:**
  Develop a **Fast Evaluating** proxy such that: $\mathbf{y} = \mathcal{M}_w(\mathbf{x})$
- **Motivation:**
  Training-validation data collection is **Expensive** & need to **Train** $\mathcal{M}_w(\mathbf{x})$ **Fast**
- **Idea:**
  Using large unlabeled dataset to enforce **Feasibility** & limited supervised dataset for **Optimality + Feasibility**

The TL;DR: An Optimal Solution Has To Be Feasible

Feasibility Condition

$$\mathcal{F}(\mathbf{y},\mathbf{x}) = \lambda_e \underbrace{\left\|g(\mathbf{x},\mathbf{y})\right\|^2}_{\texttt{Equality Gap}} + \lambda_i \underbrace{\left\|\mathrm{ReLU}[h(\mathbf{x},\mathbf{y})]\right\|^2}_{\texttt{Inequality Gap}}. \quad (2)$$

Feasibility Condition on Network Weights

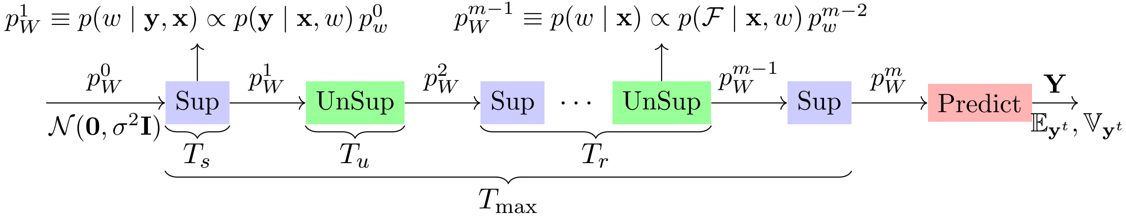
Network weights should be such that, for any input, predicted output provides $\mathcal{F}(\mathbf{y},\mathbf{x}) = 0$

If input sampling is cheap, we can create an **Augmented Labeled Dataset** for **Free**

$$\mathcal{D}^f = \{(\mathbf{x}_j, \mathcal{F}(\cdot,\mathbf{x}) = 0)\}_{j=1}^{M}$$

Sandwich Model

Alternating Training Phases for BNNs in Time-Constrained Bursts



✈ `Selection via Posterior`: Among BNN weights, select the one providing **'Best'** results for the selected criteria

$$W^\star = \arg\min_j \left[\max_i \left| g_i(\mathbf{x}^t, \mathbf{Y}_{\cdot j}) \right| \right] : \quad \text{Minimizes the maximum equality gap}$$

Bounding the Error

$$\mathbb{P}\left\{\left|\mathbb{E}[|e|] - \frac{1}{M}\sum_{i=1}^{M}|e_i|\right| \leqslant \varepsilon\right\} \geqslant 1 - \delta$$

- Error bound ε in PCBs, provided by different concentration inequalities.

Hoeffding's	Empirical Bernstein	Theoretical Bernstein
$R\sqrt{\frac{\log(2/\delta)}{2M}}$	$\sqrt{\frac{2\widehat{\mathbb{V}}_e \log(3/\delta)}{M}} + \frac{3R\log(3/\delta)}{M}$	$\sqrt{\frac{2\mathbb{V}_e \log(1/\delta)}{M}} + \frac{2R\log(1/\delta)}{3M}$

- `Hypothesis:` $\alpha\,\mathrm{MPV} \geqslant \mathbb{V}_e = \underbrace{\mathbb{E}_M\left[\mathbb{V}_W[e]\right] + \mathbb{E}_W\left[\mathbb{V}_M[e]\right]}_{\texttt{Total Variance in Error}} \geqslant \mathbb{V}_{|e|}$

- `Proposed Bound:`

$$\sqrt{\frac{2(2\times\mathrm{MPV})\log\left(1/\delta\right)}{M}} + \frac{2R\log\left(1/\delta\right)}{3M}$$

Matchup: Sandwich, BNN & DNN

Table: Comparative performance results for the ACOPF Problem for **'case57'** with 512 labeled training samples, 2048 unlabeled samples, and $T_{\max} = 600$ sec.

Method	Gap%	Max Eq.	Mean Eq.	Max Ineq.	Mean Ineq.
Sandwich BNN SvP (Ours)	**0.928**	**0.027**	0.006	**0.000**	**0.000**
Sandwich BNN (Ours)	0.964	0.045	**0.005**	0.000	0.000
Supervised BNN SvP (Ours)	3.195	0.083	0.011	0.000	0.000
Supervised BNN (Ours)	3.255	0.130	0.011	0.000	0.000
Naïve MAE	4.029	0.518	0.057	0.000	0.000
Naïve MSE	3.297	0.541	0.075	0.000	0.000
MAE + Penalty	3.918	0.370	0.037	0.000	0.000
MSE + Penalty	3.748	0.298	0.039	0.000	0.000
LD + MAE	3.709	0.221	0.033	0.000	0.000

Matchup: Sandwich, BNN & DNN

Table: Comparative performance results for the ACOPF Problem for **'case118'** with 512 labeled training samples, 2048 unlabeled samples, and $T_{\max} = 600$ sec.

Method	Gap%	Max Eq.	Mean Eq.	Max Ineq.	Mean Ineq.
Sandwich BNN SvP (Ours)	**1.484**	**0.089**	0.018	0.008	**0.000**
Sandwich BNN (Ours)	1.485	0.100	**0.016**	0.008	0.000
Supervised BNN SvP (Ours)	1.568	0.147	0.022	0.013	0.000
Supervised BNN (Ours)	1.567	0.205	0.020	0.013	0.000
Naïve MAE	1.638	2.166	0.187	**0.000**	0.000
Naïve MSE	1.622	3.780	0.242	0.000	0.000
MAE + Penalty	1.577	1.463	0.102	0.000	0.000
MSE + Penalty	1.563	2.637	0.125	0.000	0.000
LD + MAE	1.565	1.284	0.083	0.000	0.000

More Results: Larger Systems

Table: Comparative performance results for the ACOPF Problem for **case500** with 512 labeled training samples, 2048 unlabeled samples and $T_{max} = 600$ sec.

Method	Gap%	Max Eq.	Mean Eq.	Max Ineq.	Mean Ineq.
Sandwich BNN SvP (Ours)	2.009	**0.770**	0.066	0.190	**0.000**
Sandwich BNN (Ours)	2.002	0.781	**0.056**	0.191	0.000
Supervised BNN SvP (Ours)	1.191	2.204	0.088	0.141	0.000
Supervised BNN (Ours)	**1.191**	2.401	0.072	**0.140**	0.000
Naïve MAE	1.208	20.818	0.905	0.000	0.000
Naïve MSE	1.201	24.089	1.031	0.000	0.000
MAE + Penalty	1.205	11.833	0.580	0.000	0.000
MSE + Penalty	1.215	10.314	0.475	0.000	0.000
LD + MAE	1.279	11.166	0.532	0.000	0.000

Hypothesis Testing

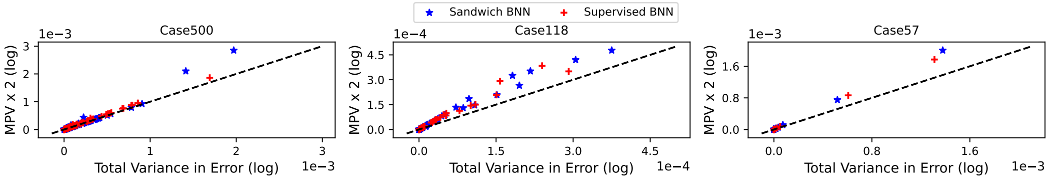
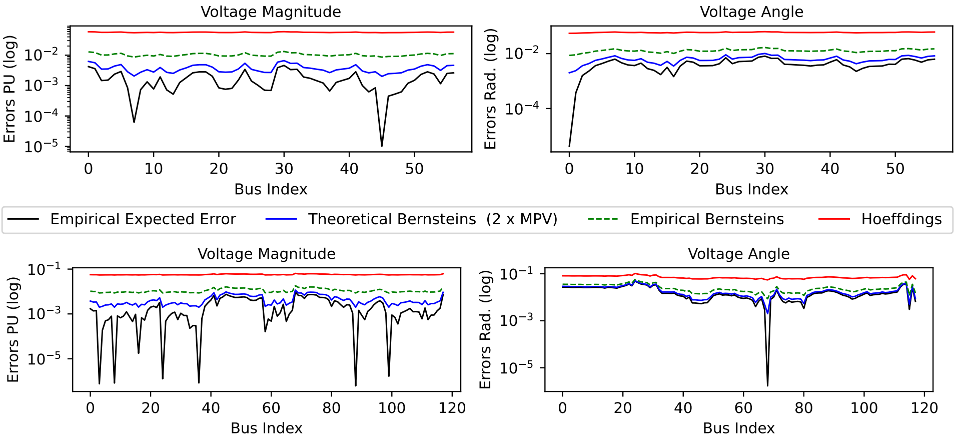


Figure: Empirical study comparing total variance in error $\widehat{\mathbb{V}}_e$ with $2 \times$ MPV across different cases of ACOPF and the proposed learning mechanisms.

Probabilistic Bounds



- Theoretical Bernstein bounds with 2 $\times$MPV are tightest.

We consider $\delta = 0.95$ and 1000 *out-of-sample* testing data points i.e. $M = 1000$

Conclusions & Future Questions

- BNNs outperform DNNs in low data and low training time settings
- Sandwich BNNs generally better at enforcing feasibility compared to Supervised BNNs
- ? Will GPUs make BNNs better, as they are harder to train?
- ? How to put cost function in unsupervised stage?
- ? What if DNN is used for unsupervised stage while BNN for supervised?

Write to me at `pareek@ee.iitr.ac.in`

Related Papers:

- P. Pareek, K. Sundar, D. Deka, & S. Misra (2024) "Optimization Proxies using Limited Labeled Data and Training Time–A Semi-Supervised Bayesian Neural Network Approach", ArXiv:2410.03085.
- P. Pareek, K. Sundar, D. Deka, & S. Misra (2024) "Learning from Less: Bayesian Neural Networks for Optimization Proxy using Limited Labeled Data", NeurIPS 2024 Workshop on Bayesian Decision-making and Uncertainty.