



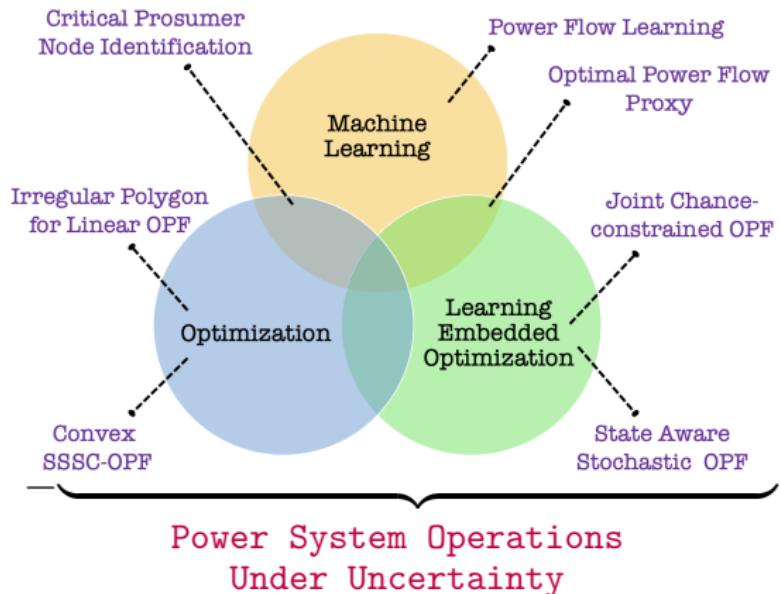
# Bayesian Learning for Power Flow Analysis and Applications

Parikshit Pareek, Ph.D.

Assistant Professor  
Department of Electrical Engineering  
Indian Institute of Technology Roorkee (IIT Roorkee)

[pareek@ee.iitr.ac.in](mailto:pareek@ee.iitr.ac.in)

# Research Interests: $\mathcal{P}^2$ -LAB



- Machine Learning

$$y = \mathcal{M}(\mathbf{x})$$

- Optimization

$$\min c(\mathbf{x})$$

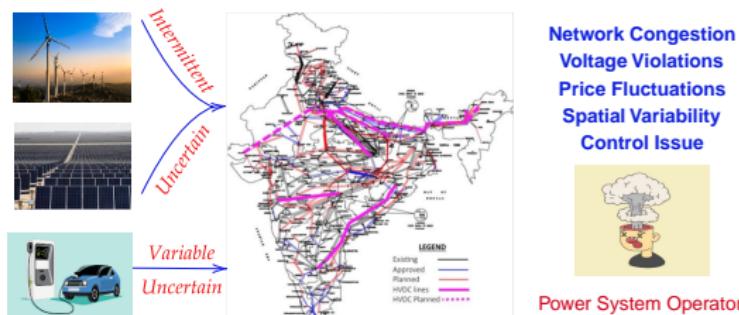
$$s.t. g(\mathbf{x}, y) \leq 0$$

- Learning-Emerged Optimization

$$\min c(\mathbf{x})$$

$$s.t. g(\mathbf{x}, \mathcal{M}(\mathbf{x})) \leq 0$$

# Challenges in Power System Operations due to Characteristics of Renewable Sources & Variable Loads



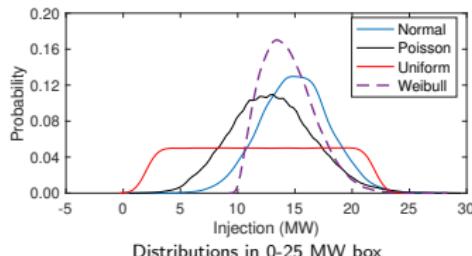
Technological Advancements

Infrastructure Upgrades

Optimal Operations

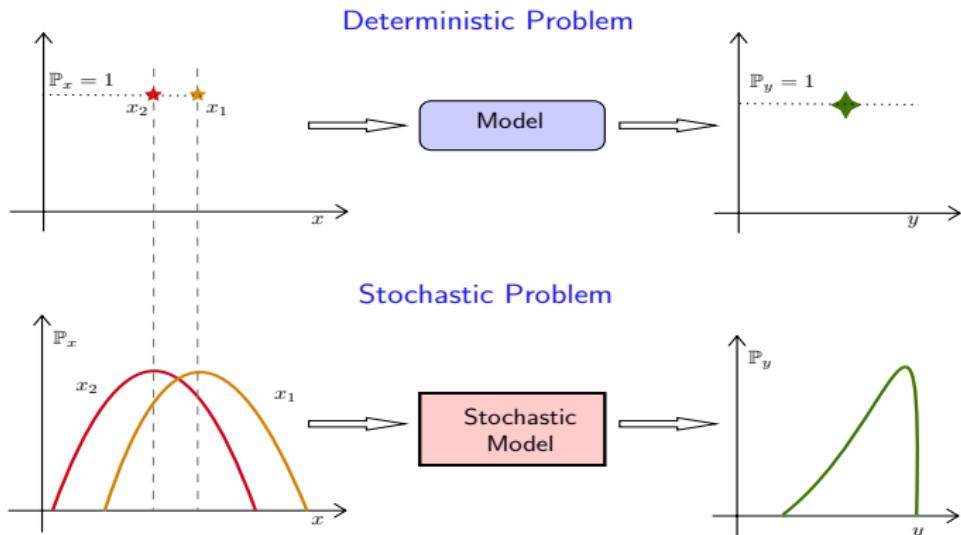
Happy story on left, becoming a Challenge towards right

**NON-PARAMETRIC UNCERTAINTY:** Situation of unknown parameters of probability distributions of random power injections



Models & Methods need to follow non-parametric approach for handling uncertainty

# Problem and Context



## Problem of Risk Estimation & Uncertainty Quantification

- What is the expected value of node voltage violation?
- What are the chances of obtaining node voltage beyond operational limits?

## Problem and Context

Power Flow:  $\Rightarrow S_i = \sum_{j \in \mathcal{N}} y_{ij}^\dagger (v_i v_i^\dagger - v_i v_j^\dagger)$

The diagram illustrates the components of the Power Flow equation. A red bracket labeled "Net Power Injection" points to the term  $S_i$ . A green bracket labeled "Complex Voltage" points to the terms  $v_i v_i^\dagger$  and  $v_i v_j^\dagger$ . A blue bracket labeled "Network Parameter" points to the term  $y_{ij}^\dagger$ .

Power flow equation set allows to obtain the complex voltage values at each network node, given the power injection at each node.

## Problem and Context

Power Flow:  $\Rightarrow S_i = \sum_{j \in \mathcal{N}} y_{ij}^\dagger (v_i v_i^\dagger - v_i v_j^\dagger)$

The diagram shows the power flow equation  $S_i = \sum_{j \in \mathcal{N}} y_{ij}^\dagger (v_i v_i^\dagger - v_i v_j^\dagger)$ . Three annotations are present: 'Net Power Injection' points to the term  $S_i$ ; 'Complex Voltage' points to the terms  $v_i v_i^\dagger$  and  $v_i v_j^\dagger$ ; and 'Network Parameter' points to the term  $y_{ij}^\dagger$ .

Power flow equation set allows to obtain the complex voltage values at each network node, given the power injection at each node.

Given

Operating Conditions  
Load Uncertainty Set

Want

Expected Violation (Risk)  
Probability of Violation

- Time & Sample Complexity
- Varying Operating Conditions
- Arbitrary Uncertainty Sets – Non-parametric Uncertainties

# Why to use ML for power flow learning?



How many Samples ?

## Why to use ML for power flow learning?



### Hoeffding's Inequality

For acceptable error  $\varepsilon = 2 \times 10^{-3}$  & 95% confidence, number of samples needed are

$$M \geq 0.5 \log(2/\beta) \varepsilon^{-2} \geq 200,258$$

$$\implies |\text{True Expected Violation} - \text{Estimated Expected Violation}| < 2 \times 10^{-3}$$

## Why to use ML for power flow learning?



## Hoeffding's Inequality

For acceptable error  $\varepsilon = 2 \times 10^{-3}$  & 95% confidence, number of samples needed are

$$M \geq 0.5 \log(2/\beta) \varepsilon^{-2} \geq 200,258$$

Running Newton-Raphson load flow 200,258 times is challenging  
More so in limited time to quantify ‘operational risk’

## Why to use ML for power flow learning?



Hoeffding's Inequality

For acceptable error  $\varepsilon = 2 \times 10^{-3}$  & 95% confidence, number of samples needed are

$$M \geq 0.5 \log(2/\beta) \varepsilon^{-2} \geq 200,258$$

Running Newton-Raphson load flow 200,258 times is 'challenging'

More so in limited time to quantify 'operational risk'

Solution  $\implies$  A Fast Evaluating ML proxy of Power Flow

# How good an ML tool can be for risk estimation?

If NRLF-based statistical estimation have error  $\varepsilon_h$ , then same number of ML evaluations will achieve  $|\text{True Expected Violation} - \text{Estimated Expected Violation}|$

$$\text{ML Prediction Confidence} \\ \text{ML Prediction Error} \\ < L\varepsilon_m(1 - \delta) + M\delta + \varepsilon_h \\ \text{NRLF Estimation Error} \\ \text{Arbitrarily Large Number}$$

$L\varepsilon_m(1 - \delta) + M\delta + \varepsilon_h$

Annotations:  $L\varepsilon_m(1 - \delta)$  is labeled 'ML Prediction Error' with a red bracket above it.  $M\delta + \varepsilon_h$  is labeled 'NRLF Estimation Error' with a red bracket above it.  $L\varepsilon_m(1 - \delta) + M\delta + \varepsilon_h$  is labeled 'Arbitrarily Large Number' with a green bracket below it.

with confidence  $1 - \beta$ , under a condition.

Performance using 200, 258 NRLFs  $\equiv$  Performance using 801, 029 ML Evaluations

Using Gaussian process (GP) model  $L\varepsilon_m(1 - \delta) + M\delta + \varepsilon_h \approx 2 \times \varepsilon_h$

**Condition:** Probability of the absolute difference between true & predicted voltage (by ML) being larger than  $\varepsilon_m$  is less than or equal to  $\delta$ .

---

Proof follows from the fact that with ML performance condition allows to upper bound difference in true and ML based estimation with  $L\varepsilon_m(1 - \delta) + M\delta$

# Why Gaussian Process (GP) is suitable for Power Flow learning?

GP regression uses **prior knowledge** about a function along with data, to **predict unobserved values** and **quantifying prediction uncertainty**.

- Provide performance guarantee without requiring ground truth solutions
- Has low training data requirement due to prior knowledge infusion

## A Closed-form Power Flow Approximation Framework is

- **Flexible**  $\implies$  Non-linear forms with *complexity-accuracy* trade-off
- **Easy to Evaluate**  $\implies$  Faster numerical calculations
- **Non-parametric**  $\implies$  Works within a power injection range or hypercube
- **Differentiable**  $\implies$  Can be fed into optimization problems
- **Interpretable**  $\implies$  Should provide insights into physical system

Essentially an **explicit expression** of voltage as a function of power injection which provides a **probabilistic performance guarantee** for risk estimation

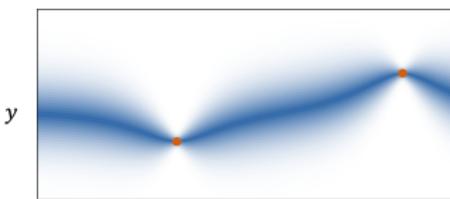
# GP Working Idea: In Brief



- Prior Knowledge: Smooth Function

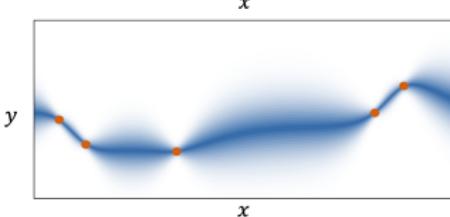
$$k(x_i, x_j) = \tau^2 \exp \left\{ \frac{-\|x_i - x_j\|_2^2}{2\ell^2} \right\}$$

- Predict unobserved values: Mean



Thickest Shade of Blue

- Prediction uncertainty: Variance



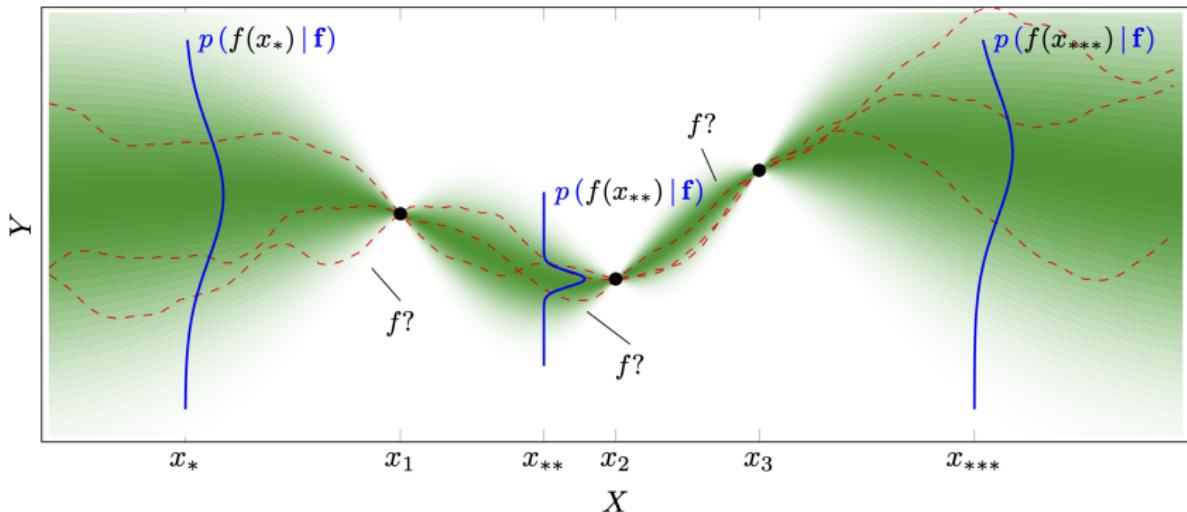
Lighter blue – lower probability

- Performance Guarantee<sup>†</sup>

$$\mathbb{P}\left\{\left|V(\mathbf{s}) - [\mu(\mathbf{s}) + \gamma\sigma(\mathbf{s})]\right| \geq \varepsilon_m\right\} \leq \delta(\gamma)$$

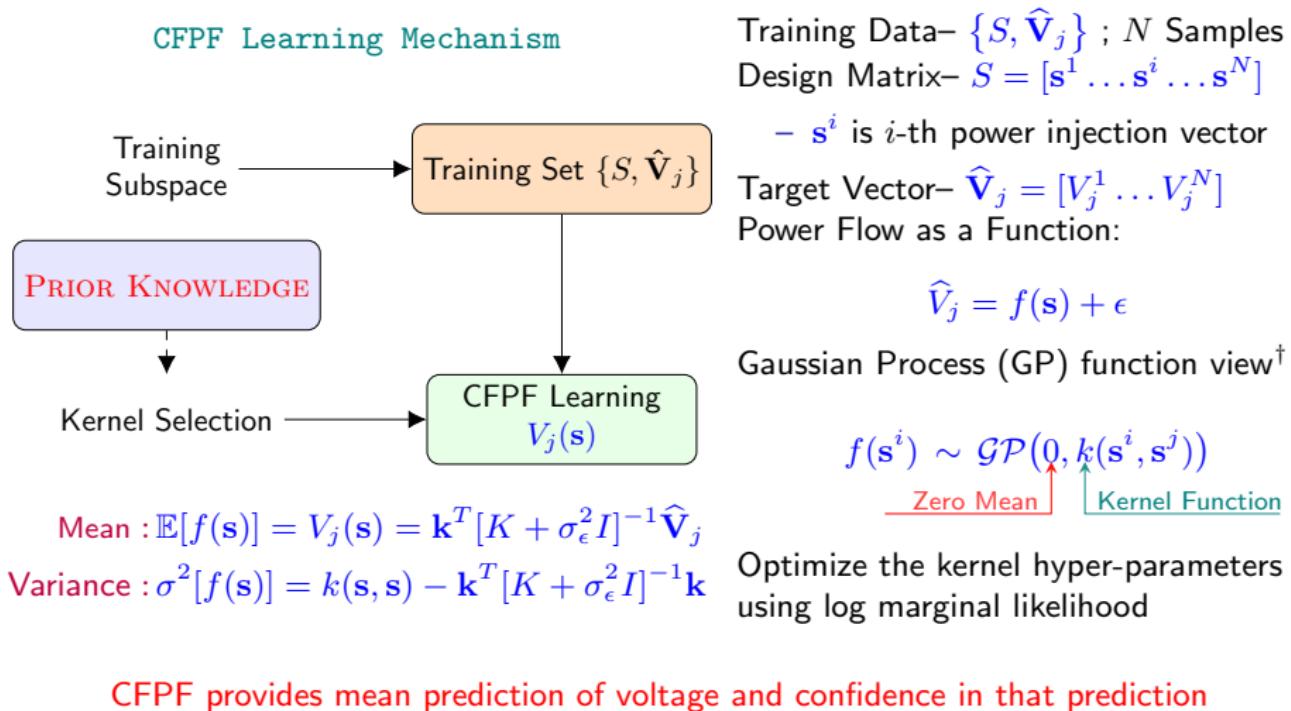
<sup>†</sup> Considering the power flow function is a GP.

## GP Assumption



The distribution of  $f(x^*)$ ,  $f(x^{**})$  and  $f(x^{***})$  for the three inputs  $x^*$ ,  $x^{**}$  and  $x^{***}$ , conditional on the observed values  $f(x_1)$ ,  $f(x_2)$  and  $f(x_3)$ .

# CFPF Learning: Using Gaussian Process



<sup>†</sup> C. K. Williams and C. E. Rasmussen, Gaussian processes for machine learning. MIT press Cambridge, MA, 2006, vol. 2, no. 3.

# CFPF: Forms

$$V_j(\mathbf{s}) = [k(\mathbf{s}^1, \mathbf{s}) \ \dots \ k(\mathbf{s}^N, \mathbf{s})] \alpha_j$$

Training Data  
Variable  $\mathbf{s} = [\mathbf{p}; \mathbf{q}]$       Constant

How Do FORMS LOOK THEN?

*Simple, Standard Forms*

- Linear:  $\mathbf{v} = A\mathbf{s} + b$
- Quadratic:  $V_j(\mathbf{s}) = \mathbf{s}^T M \mathbf{s} + \mathbf{m}^T \mathbf{s} + r$

*More Complex but Accurate Forms*

$$V_j(\mathbf{s}) = \sum_{i=1}^N \alpha_j(i) \beta^i$$

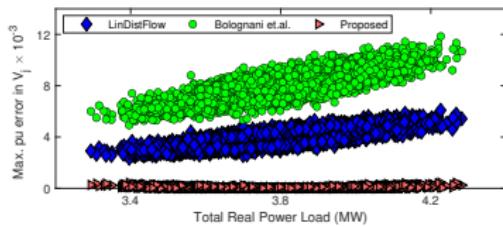
where,  $\beta^i = \tau^2 \exp\left(-\|\mathbf{s}^i - \mathbf{s}\|^2 / 2l^2\right)$  : Gaussian Kernel

$\mathbf{v}$  : Node Voltage Vector;  $M$  : Sensitivity Matrix;  $\mathbf{m}$  : Slope Vector;

$\mathbf{s}$  : Node Power Injection Vector

# Performance of CFPF

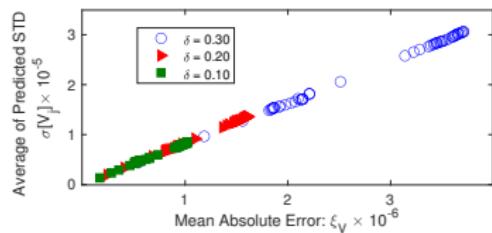
- Subspace-wise Approximations & Non-Parametric



Error Not a Function of Load

For 69-Bus System

- Inherent Accuracy Indicator



Use of Predictive Variance

For 56-Bus System

## Non-parametric Nature

Distribution	Max. MAE (pu)
Normal	1.86E-05
Beta	2.39E-05
Laplace	2.29E-05
Mixed	1.74E-05

For 33-Bus System

- Differentiable Functions
- Faster Numerical Evaluations
- Model Interpretability via Hyper-parameters
- Independent from Network Type Assumptions

# Graph-Structured Kernel for Low Sample Complexity CFPF Learning

Combining Physics and Intuition into GP

# Standard CFPF Learning: Limitations & Solutions

## ✖ Curse-of-Dimensionality

Complexity  $\mathcal{O}(N^3)$  with  $N$  training samples; Systems  $\leq 100$ -Bus

## ✖ Mesh Network Flows

Power injection-voltage relationship is not direct; Higher order nonlinearity

## ✖ Limited Use

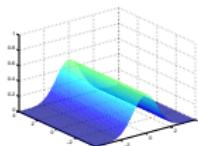
Full GP is restrictive to be used within Bayesian Optimization

- Learn individual low-dimensional sub-function & combine  
Lower Curse-of-Dimensionality
- Capturing localized voltage-power injection relationship will be easier  
Suitable for Mesh Network Flows
- Low-dimensional GPs  
Useful as surrogate in Bayesian Optimization approach

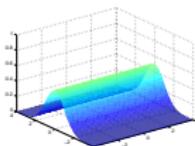
An Ideal Situation Would be if

A high-dimensional voltage function is breakable into low-dimensional sub-functions

# Additive Gaussian Processes

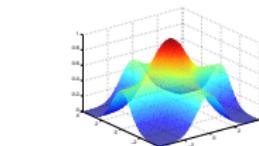


$k_1(x_1, x'_1)$   
1-D Kernel

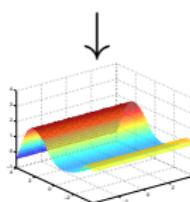


$k_2(x_2, x'_2)$   
1-D Kernel

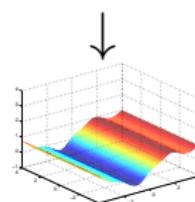
=



$k_1(x_1, x'_1) + k_2(x_2, x'_2)$   
Additive Kernel

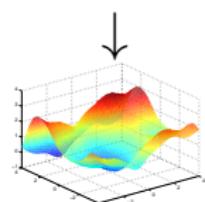


$f_1(x_1)$   
1-D GP Prior



$f_2(x_2)$   
1-D GP Prior

=



$f_1(x_1) + f_2(x_2)$   
Additive GP Prior

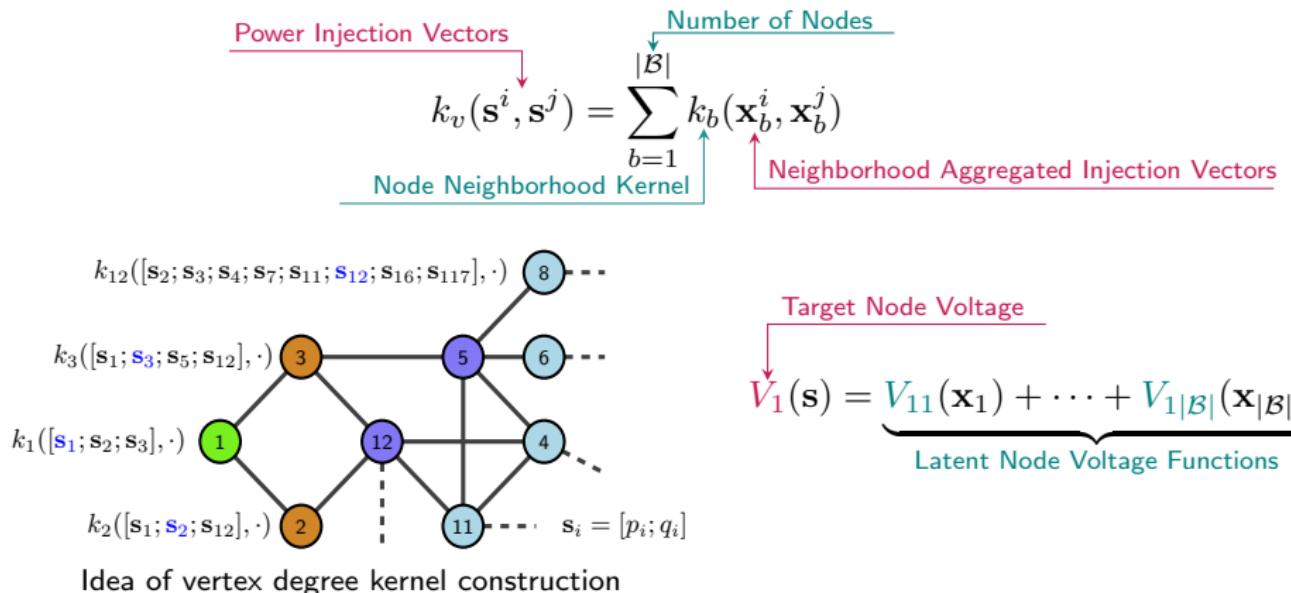
$$V_j(\mathbf{s}) = V_j^1(\mathbf{s}_1) + \cdots + V_j^m(\mathbf{s}_m)$$

$$\equiv \mathcal{GP}_1\left(0, K_1(\mathbf{s}_1, \cdot)\right) + \cdots + \mathcal{GP}_m\left(0, K_m(\mathbf{s}_m, \cdot)\right)$$

$\equiv$  Additive Gaussian Process

# Using Network-Structure to Achieve Additive GP Architecture

- Neighboring power injections have highly correlated effect on node voltages
- Effect of far away power injections is approximately equal to sum of individual effects



## Active Learning: Lower Samples and Stopping Criteria

- ★ **What:** Learning by successively selecting the next training point ‘intelligently’
- ★ **Why:** To speed-up the learning process using unlabeled data i.e. only input data needed  $\implies$  Low Sample Complexity  $\implies$  Less Power Flow Samples Needed
- ★ **Concept:** Next training point is the one which has maximum uncertainty in underlying function

$$\mathbf{s}^{t+1} = \arg \max_{\mathbf{s} \in \mathcal{L}} \sigma_f^t(\mathbf{s}) \rightarrow \text{Only Function Evaluation}$$

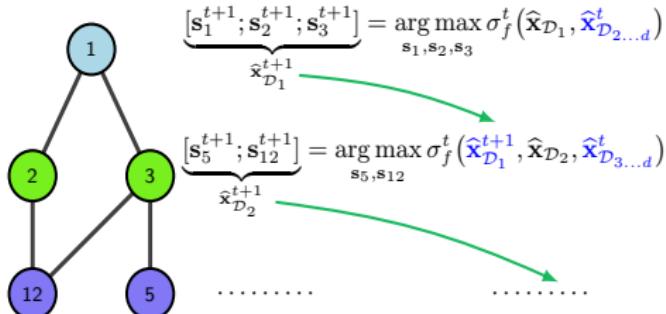
Provides a Proxy of Error  $\rightarrow$  Stopping Criteria

Finding maximum variance point for large-dimensional input space is hard  
 $\implies$  Used mostly up to 20-dimensions  
 $\implies$  Power systems have 100s of uncertain power injections

# Network-swipe Active Learning

$$\text{Variance: } \sigma^2[f(\mathbf{s})] = \underbrace{\sum_{b=1}^{|\mathcal{B}|} k_b(\mathbf{x}_b, \mathbf{x}_b)}_{\text{Constant}} - \underbrace{\left[ \sum_{b=1}^{|\mathcal{B}|} k_b(X_b, \mathbf{x}_b) \right]^T}_{\text{Variable Vector}} \underbrace{[K_1 + \dots + K_{|\mathcal{B}|}]^{-1}}_{\text{Constant Matrix}} \underbrace{\left[ \sum_{b=1}^{|\mathcal{B}|} k_b(X_b, \mathbf{x}_b) \right]}_{\text{Variable Vector}}$$

Neighborhood Aggregated Injection Vectors  $\mathbf{x}_b$ 's have Overlap



Idea of network-swipe algorithm for AL.

---

## Algorithm 1 Network-Swipe Algorithm for AL

**Require:**  $T, \mathcal{D}, \{\mathbf{s}^1, V^1\}, \sigma_{threshold}^2, T_{max}$

- 1: Initialize GP model (6) with VDK (9)
- 2: **while**  $\sigma_f^2(\mathbf{s}^t) \geq \sigma_{threshold}^2$  **do**
- 3:     Solve (11) for  $\hat{\mathbf{x}}_{D_i}^{t+1}$ , sequentially for  $i = 1 \dots d$
- 4:     Solve ACPF for load  $\mathbf{s}^{t+1}$  to get  $V^{t+1}$
- 5:     Update GP model with  $(\mathbf{s}^{t+1}, V^{t+1})$ ;  $t = t + 1$
- 6:     **If**  $\text{runtime} > T_{max}$  **then break**
- 7: **end while**

**Output:** Compute  $\mu_f(\mathbf{s}), \sigma_f^2(\mathbf{s})$  for final GP

---

# Benchmarking Performance

Comparison of MAE performance of different methods for 118-Bus System

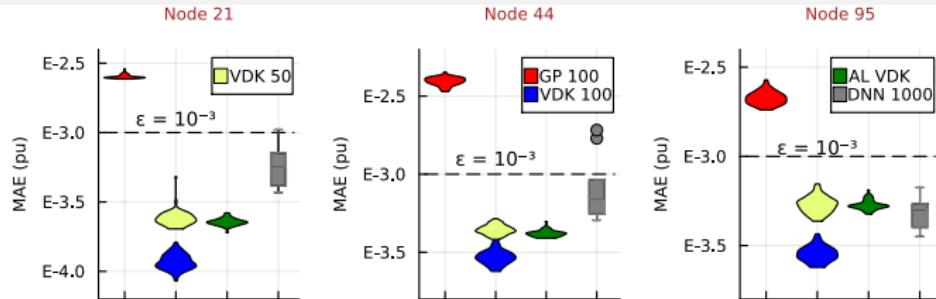


Table: Sample and Time Requirement for Risk Estimation in 500-Bus System

Node	Samples	Time(s)	MAE (pu) $\times 10^{-4}$	$\Delta VE \times 10^{-4}$
4	67 - 70	28 - 30	$3.11 \pm 1.0$	$7.8 \pm 0.5$
268	102 - 109	53 - 58	$4.68 \pm 1.5$	$7.9 \pm 0.2$
320	72 - 76	30 - 33	$4.66 \pm 1.8$	$7.8 \pm 0.4$
321	70 - 77	30 - 33	$4.96 \pm 2.5$	$6.8 \pm 0.5$
-	Mean evaluation time for 80100 samples is 33.2 sec			

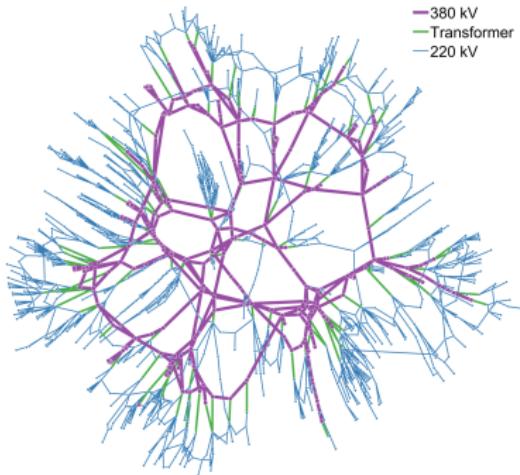
VDK-AL performs the risk estimation for any node in  $\approx 120$  sec

NRLF running time for 20025 samples is  $\approx 4205$  sec

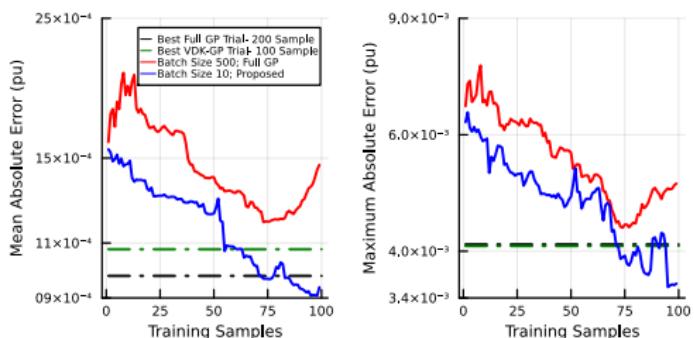
Demonstrating the efficiency and low sample-complexity of AL-VDK & Usefulness for power system operation under uncertainty

# Active Learning Performance for 1354-Bus System

- 100 samples are sufficient to learn a node voltage function with 2236 random power injections



1,354-Buses, 260 generators, & 1,991-lines.

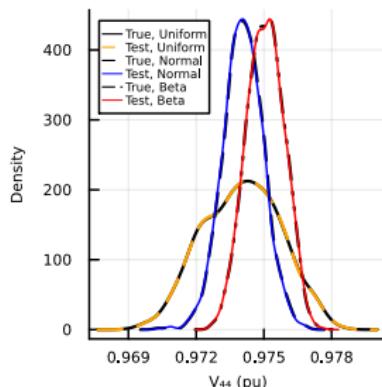
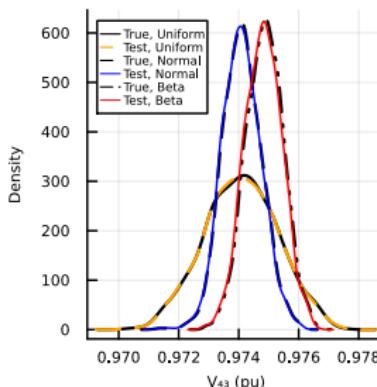
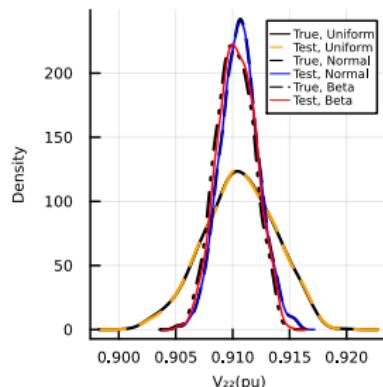


Learning  $V_5$  for 1354-Bus system

#PF	Learning Method (Training Samples)		
	Full GP (200)	VDK-GP (100)	AL (100)
	5000	2500	100

A random trial of active learning outperforms large number of passive learning attempts  
Reduced time complexity of 25-50 times

# Application: Uncertainty Quantification (UQ)



Density functions for UQ in 118-Bus system for node voltages  $V_{22}$ ,  $V_{43}$  &  $V_{44}$

Voltage Violation probability(VP)  $\Rightarrow$  Probability of voltage value being out of limits

Voltage Violation probability (VP) estimation with various input distributions for 33-bus system

Distribution	Normal	Beta	Laplace	Mixed	Comments
Max. Error in VP	0.0026	0.0028	0.0029	0.0009	Monte-Carlo – Proposed
Time Proposed (s)	$1.04 + 1.66 + 0.70 \times 4 = 5.5$				Train Once; Evaluate Multiple
Time MCS (s)	105.2	103.6	100.4	108.3	Each Time 10000 PF Solving

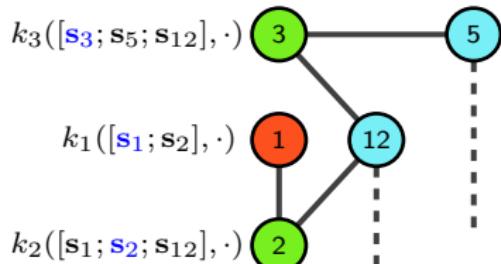
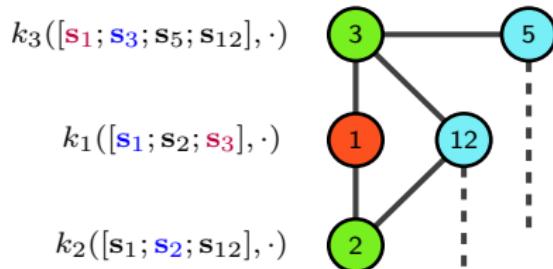
Single model works for different distributions without the need for retraining

# Multi-Task Vertex Degree Kernel for CFPF under Network Contingencies

Using 'Available' Knowledge to Reduce Training Sample Requirement

# CFPF Learning under Network Contingencies

Using VDK's Ability to Accommodate Graph Structure & Existing Models to Learn CFPF for 'New' Network Topology



Different sub-kernels of VDK for different topologies of a part of 118-Bus system.

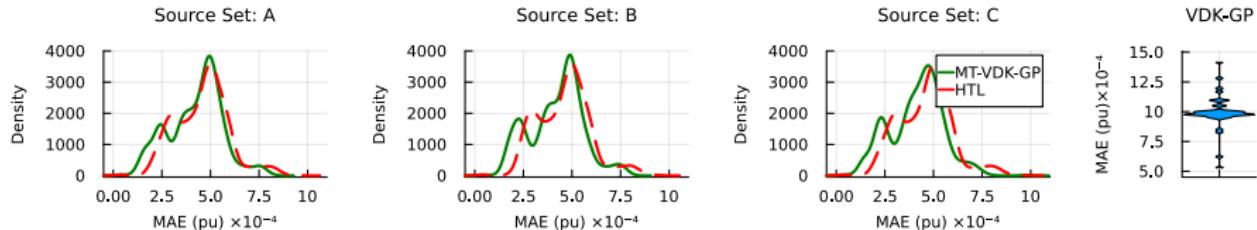
## Multi-Task Vertex Degree Kernel (MT-VDK)

$$k_{\mathcal{E}_T}(\mathbf{s}^i, \mathbf{s}^j) = \underbrace{\sum_{b=1}^{|\mathcal{B}|} k_b(\mathbf{x}_b^i, \mathbf{x}_b^j)}_{\text{VDK on Target } \mathcal{E}_T} + \omega \underbrace{\sum_{m=1}^{M_s} k_{\mathcal{E}_m}(\mathbf{s}^i, \mathbf{s}^j)}_{\text{VDKs on Source } \mathcal{E}_m \text{'s}}$$

Annotations for the equation:

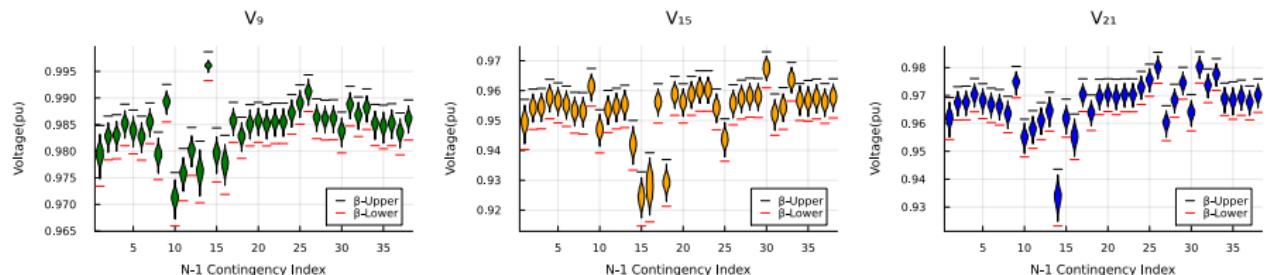
- A red bracket labeled "Power Injection Vectors" spans the first term  $\sum_{b=1}^{|\mathcal{B}|} k_b(\mathbf{x}_b^i, \mathbf{x}_b^j)$ .
- A green bracket labeled "Weight" spans the second term  $\omega \sum_{m=1}^{M_s} k_{\mathcal{E}_m}(\mathbf{s}^i, \mathbf{s}^j)$ .
- A blue bracket labeled "Number of Source Networks" spans the term  $M_s$ .
- A green bracket labeled "Target Network Graph" spans the entire equation  $k_{\mathcal{E}_T}(\mathbf{s}^i, \mathbf{s}^j)$ .

## More Results



Mean absolute error (MAE) densities for 24 node voltages for 38 different networks (N-1 contingencies).

Average error HALVED using MT-VDK-GP with same number of training sample



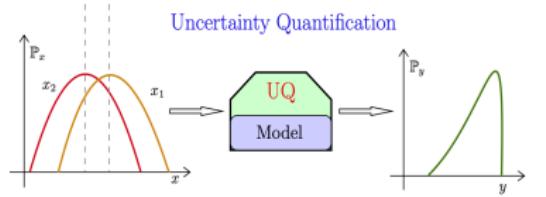
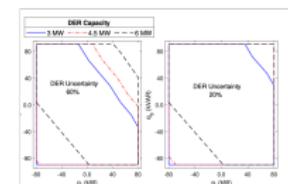
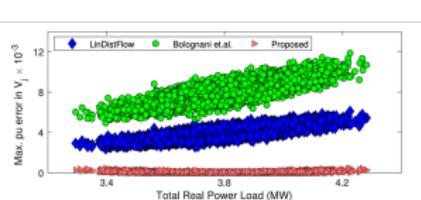
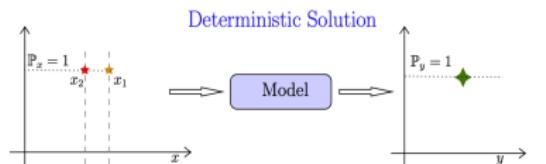
Probabilistic Voltage Envelopes (PVEs) for node voltages & 38 different structures of 30-Bus network

# Summary & Applications

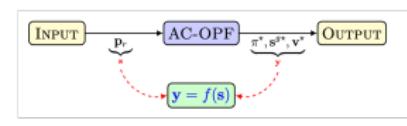
Standard GP → VDK-GP & MT-VDK-GP → Network-Swipe AL

Application Specific Physics-inspired Kernels can help build Interpretable Learning Models in Low Data Regimes.

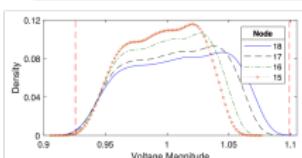
## Applications of Physics-inspired Power Flow Learning



## Linear Power Flow under Uncertainty



## Closed-form OPF Proxy



## Privacy-preserving Feasibility Assessment

All above mentioned works incorporate uncertainties in non-parametric manner

# Thank You!

## Collaborators

- Dr. Sidhant Misra
- Prof. Lalit Goel
- Dr. L.P.M.I. Sampath
- Dr. Deepjyoti Deka
- Prof. Ashu Verma
- Prof. Hung D. Nguyen
- Dr. Abhijeet Jayakumar
- Dr. Carlton Coffrin
- Dr. Weng Yu

# $\mathcal{P}^2$ LAB

@EE, IIT Roorkee



Our group is currently working on  
**Machine Learning applications in Power Systems.**

We have a few **fully funded PhD positions**  
for motivated candidates interested in this area.

*In case you know someone who might be a good fit,  
please feel free to refer them to us.*

More info: <https://psquare-lab.github.io/people/>

# ML Performance Bounds

## Theorem

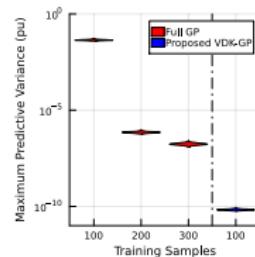
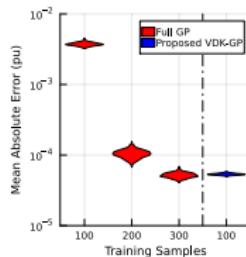
Given for an ML model  $\mathbb{P}\{|V(\mathbf{s}^i) - \hat{V}(\mathbf{s}^i)| > \varepsilon_m\} \leq \delta$  with  $\delta \in (0, 1)$  for any  $\mathbf{s} \in \mathcal{S}$ , and  $h(\cdot)$  is a Lipschitz continuous function with Lipschitz constant  $L$ , then the error in expected value estimation using the ML model, for any  $\mathbf{s} \in \mathcal{S}$ , is bounded with probability greater than  $1 - \beta$  as

$$|\mathbb{E}[h_p(\mathbf{s})] - \hat{\mathbb{E}}[h_m(\mathbf{s})]| \leq L\varepsilon_m(1 - \delta) + M\delta + \varepsilon_h$$

where,  $\beta \in (0, 1)$ ,  $\varepsilon_h = \sqrt{\frac{\log(2/\beta)}{2N}}$ ,  $M > |h_p(\mathbf{s}) - h_m(\mathbf{s})|$ , and  $N$  number of samples.

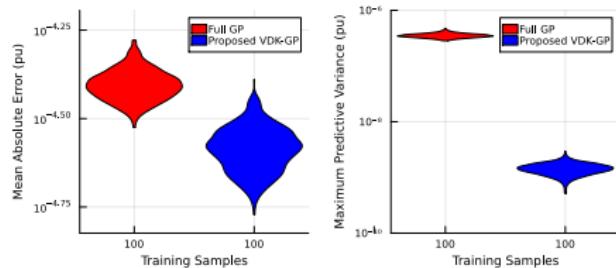
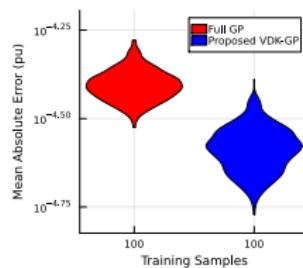
# Benchmarking

500 independent trials with 100 training samples; Testing: 1000 unique samples



$V_1$  within  $\pm 10\%$  hypercube for 500-Bus system.

Three Times Lower Sample Complexity



$V_1$  within  $\pm 10\%$  hypercube for 118-Bus system.

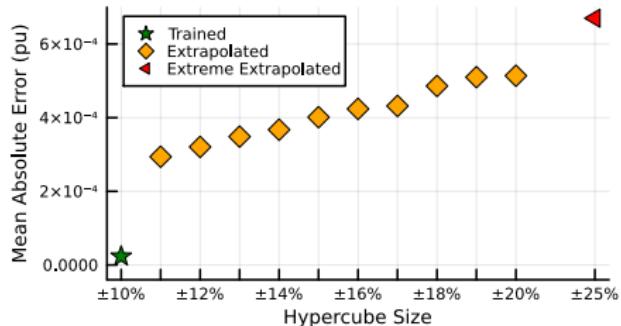
50% Lower Error & 100 Times Confident Model

Proposed VDK-GP outperforms a 3-layer, 1000-neuron Deep Neural Network for using 100 training samples in 118-Bus system

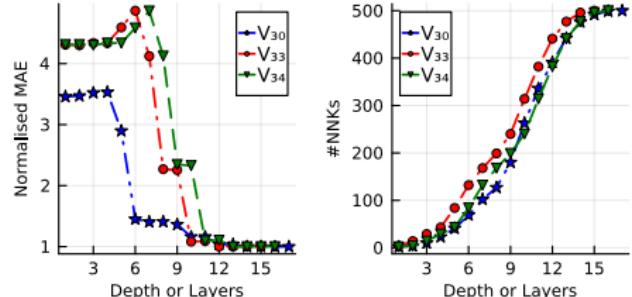
Accurate, loading independent power flow model with extremely low sample complexity  
Useful for power system operation under uncertainty

Node	MAE (pu)	
	Proposed	DNN
1	$5.22 \times 10^{-5}$	$1.89 \times 10^{-4}$
43	$8.70 \times 10^{-5}$	$9.77 \times 10^{-4}$
117	$2.26 \times 10^{-5}$	$9.05 \times 10^{-4}$

## More Insights: Extrapolation & Depth Effect

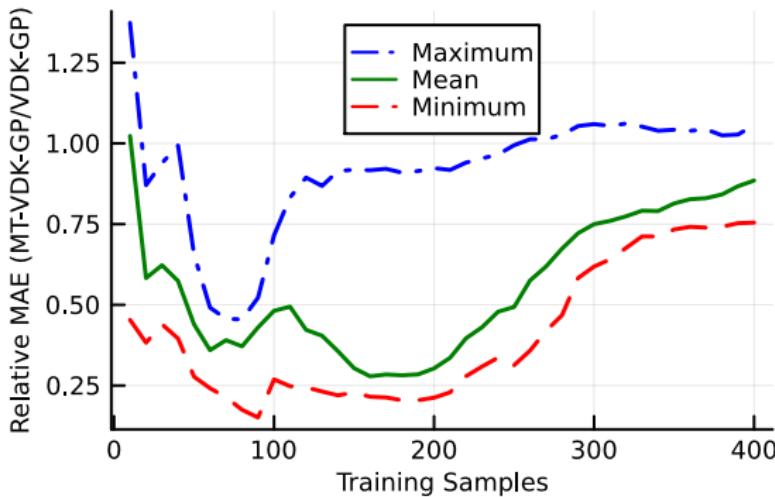


Extrapolation of VDK-GP model trained within  $\pm 10\%$  hypercube for 118-Bus system.



Effect of depth on learning quality of three different voltage function in 500-Bus system.

## VDK-GP V/S MT-VDK-GP



Relative MAE variation with respect to training samples— defined as the ratio of MAE obtained using MT-VDK-GP to VDK-GP.

Valleys in the relative error curves reflect the superior performance of the proposed MT-VDK-GP at low data regimes.