$\mathcal{P}^2$ LAB
@EE, IIT Roorkee

Bayesian Inference via Prior-Data Fitted Networks

Parikshit Pareek, Ph.D.

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

Time Equation for ML Models

$$T_{data} + T_{train} + T_{predict}$$

Time Equation for ML Models

$$T_{data} + T_{train} + T_{predict}$$

Training Data

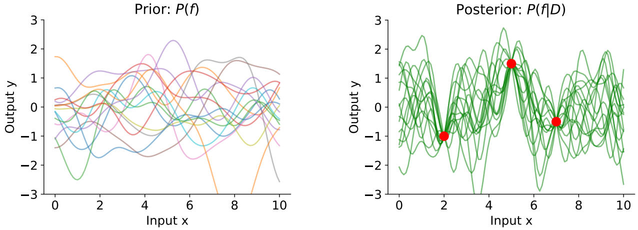
$$\mathcal{M}(\theta_i|\mathcal{D}_i) : \texttt{Need Retraining for Each } \mathcal{D}.$$

Model Weights

The Mechanism: Bayes' Theorem

The Mathematical Engine

$$\underbrace{P(\text{Model}|\text{Data})}_{\text{Posterior}} \propto \underbrace{P(\text{Data}|\text{Model})}_{\text{Likelihood}} \times \underbrace{P(\text{Model})}_{\text{Prior}}$$



The Prior

(Random possibilities)

The Posterior

(Constrained by Data)

Calculating the **Posterior** (Right Image) usually requires slow, complex math (integrals)– Difficult in Higher Dimensions

Comparison: Point Estimates vs. Bayesian Beliefs

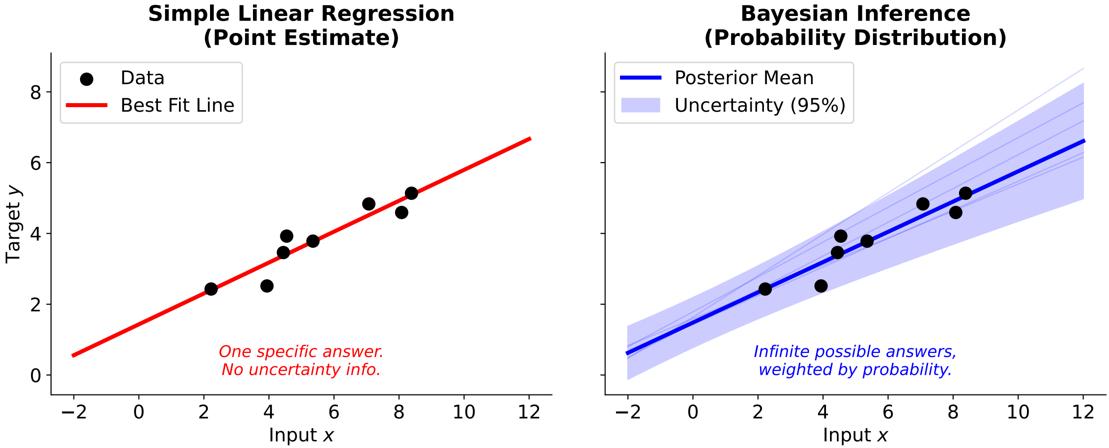
1. Simple Linear Regression

- **Goal:** Find the *single best* line (w, b) that minimizes error.
- **Result:** A Point Estimate.
- **Flaw:** Overconfident. It predicts a precise value even far from data where it should be clueless.

2. Bayesian Inference

- **Goal:** Find the *distribution* of all plausible lines given the data.
- **Result:** A Posterior Distribution.
- **Benefit: Uncertainty Awareness**. It knows when to say "I don't know" (wide shaded region).

Comparison: Point Estimates vs. Bayesian Beliefs

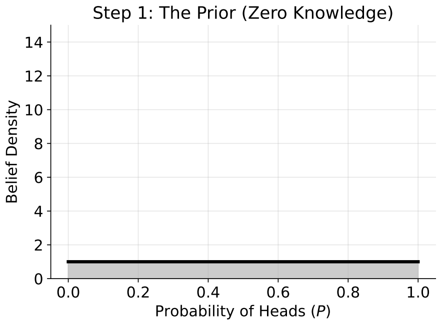


Left: SLR gives one rigid answer. *Right:* Bayesian Inference captures the "cone of uncertainty," growing wider where data is scarce.

Bayesian Inference: Updating Beliefs

The Core Philosophy

- In Bayesian statistics, parameters are not fixed numbers; they are **distributions**.
- We start with a **Prior**: A broad assumption ("Anything is possible").



State 1: Maximum Uncertainty

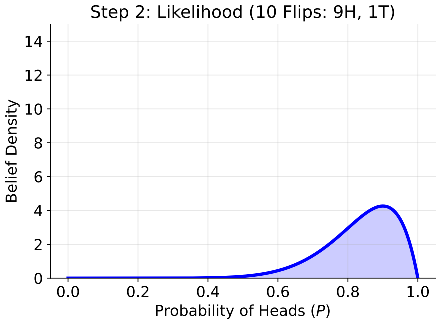
The Example: Is this coin biased?

- *Start:* I know nothing. It could be fair or biased.

Bayesian Inference: Updating Beliefs

The Core Philosophy

- In Bayesian statistics, parameters are not fixed numbers; they are **distributions**.
- We start with a **Prior**: A broad assumption ("Anything is possible").
- We observe **Data** (Likelihood).



State 2: Learning begins...

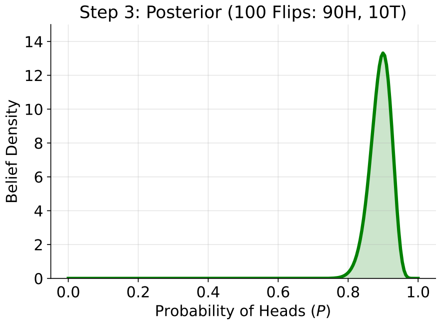
The Example: Is this coin biased?

- *Start:* I know nothing. It could be fair or biased.
- *Observation:* I flip it 10 times, get 9 Heads.

Bayesian Inference: Updating Beliefs

The Core Philosophy

- In Bayesian statistics, parameters are not fixed numbers; they are **distributions**.
- We start with a **Prior**: A broad assumption ("Anything is possible").
- We observe **Data** (Likelihood).
- We update to a **Posterior**: A sharper, more confident belief.



State 3: Strong Belief formed!

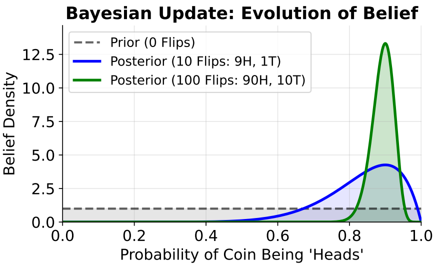
The Example: Is this coin biased?

- *Start:* I know nothing. It could be fair or biased.
- *Observation:* I flip it 10 times, get 9 Heads.
- *Result:* I am now fairly sure it favors Heads.

Bayesian Inference: Updating Beliefs

The Core Philosophy

- In Bayesian statistics, parameters are not fixed numbers; they are **distributions**.
- We start with a **Prior**: A broad assumption ("Anything is possible").
- We observe **Data** (Likelihood).
- We update to a **Posterior**: A sharper, more confident belief.



The Example: Is this coin biased?

- *Start:* I know nothing. It could be fair or biased.
- *Observation:* I flip it 10 times, get 9 Heads.
- *Result:* I am now fairly sure it favors Heads.

Prior-Data Fitted Networks (PFNs)

Goal: Perform BAYESIAN INFERENCE in a single forward pass.

Prior-Data Fitted Networks (PFNs)

Goal: Perform BAYESIAN INFERENCE in a single forward pass.

Why? Bayesian inference is slow, and repeated retraining makes the computational cost prohibitive

Prior-Data Fitted Networks (PFNs)

Goal: Perform Bayesian Inference in a single forward pass.

Why? Bayesian inference is slow, and repeated retraining makes the computational cost prohibitive

How? Learn to approximate the *Posterior Predictive Distribution (PPD)* $p(y|x, \mathcal{D})$

Amortized Bayesian Inference

Given a new dataset $\mathcal{D}_{\text{context}}$ and query point x_{test}, it outputs

$$q_{\theta^\star}(y_{\text{test}} \mid x_{\text{test}}, \mathcal{D}_{\text{context}}) \approx p(y_{\text{test}} \mid x_{\text{test}}, \mathcal{D}_{\text{context}})$$

in a single forward pass, where $\theta^\star$ are the optimal PFN parameters.

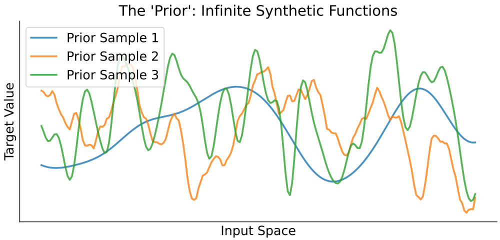
The Core Idea: Learning from Priors

What is a PFN?

- PFNs are not trained on your target dataset (like ImageNet or Power Grid data).
- They are **Meta-Learners** trained on **Priors**

The "Prior" Definition

- A mathematical recipe that generates infinite synthetic datasets (e.g., Gaussian Processes).
- The PFN learns the *statistical behavior* of this prior, not specific data points.



The model observes millions of random functions during training, effectively memorizing how to interpolate data.

The Core Idea: Learning from Priors

What is a PFN?

- PFNs are not trained on your target dataset (like ImageNet or Power Grid data).
- They are **Meta-Learners** trained on **Priors**

The "Prior" Definition

- A mathematical recipe that generates infinite synthetic datasets (e.g., Gaussian Processes).
- The PFN learns the *statistical behavior* of this prior, not specific data points.

Example: GP Prior Generator

```
def get_batch():
  # 1.  Sample Physics
  ℓ ∼ Uniform(0.1, 2.0)
  kernel = RBF(lengthscale=ℓ)

  # 2.  Sample Data
  X = rand(N, D)
  y ∼ MultivariateNormal(0, K(X))

  return X, y
```

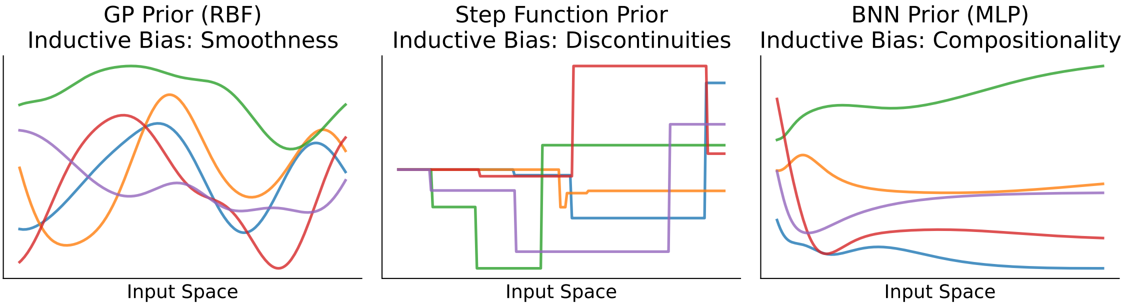
A Menagerie of Priors: What can we model?

Since the PFN learns to approximate the *Bayesian Posterior* of the generator, changing the generator changes the inference engine.

- **Gaussian Process (GP) Priors**
  - Generator: Sample kernel hyperparameters $(\ell, \sigma_f, \sigma_n) \to$ Sample GP.
  - **Result:** A neural network that mimics exact GP inference but 80x faster.
- **Bayesian Neural Network (BNN) Priors**
  - Generator: Sample MLP weights $W \sim \mathcal{N}(0, I)$.
  - **Result:** PFN approximates the complex weight-space posterior of a BNN.
- **Structural Causal Models (TabPFN)**
  - Generator: Complex causal graphs, sparse interactions, non-linear activations.
  - **Result:** A foundation model for tabular data that beats XGBoost on small datasets.

Why Prior Selection Matters: Inductive Bias

The "No Free Lunch" Theorem: No model works best on all data.
The PFN Solution: Embed the correct *Inductive Bias* via the Prior.

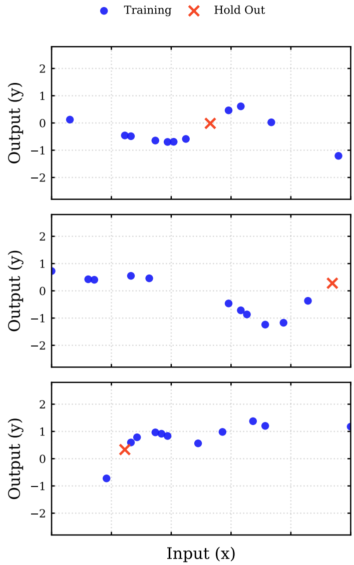


If you train on this...
Model assumes world is smooth (Power Systems).

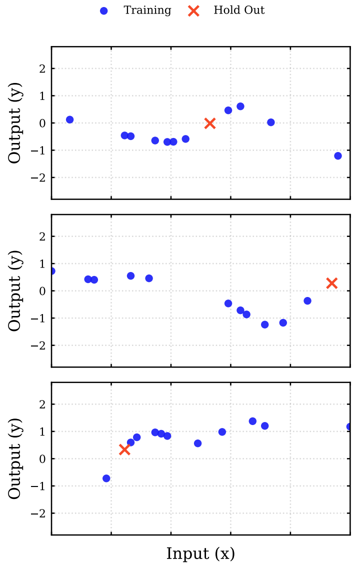
If you train on this...
Model assumes world has jumps (Digital Logic).

If you train on this...
Model assumes world is compositional.

PFN Idea



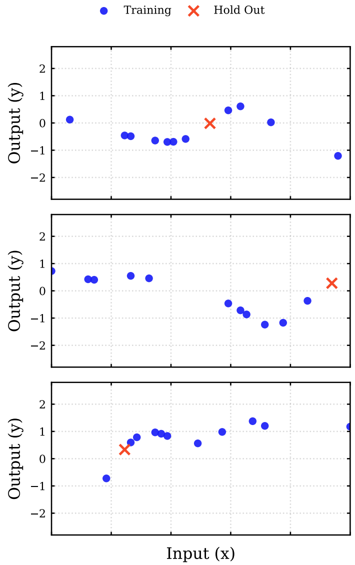
PFN Idea



Train PFN using Negative Log-Likelihood Loss over hold-out samples

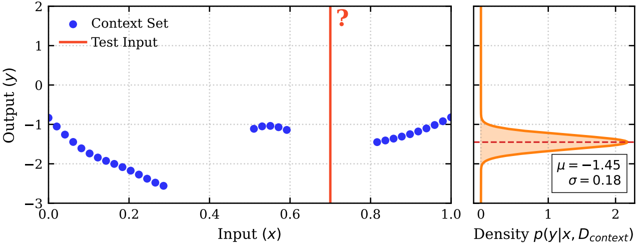
$$\ell_\theta = \sum_{k=1}^{K} \left[-\log q_\theta(\mathbf{y}^k \mid \mathbf{x}^k, \mathcal{D}^k) \right]$$

PFN Idea



Train PFN using Negative Log-Likelihood Loss over hold-out samples

$$\ell_\theta = \sum_{k=1}^{K} \left[-\log q_\theta(\mathbf{y}^k \mid \mathbf{x}^k, \mathcal{D}^k) \right]$$



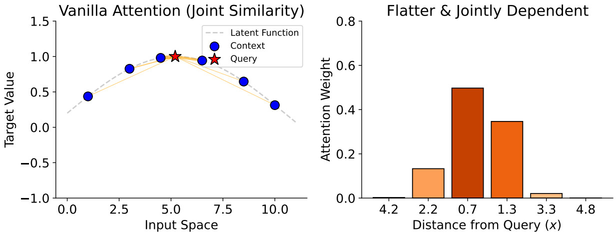
How PFNs "Think"

- The model does not use fixed weights. It uses **In-Context Learning**.
- The **Attention Mechanism** acts as a learned similarity kernel.

The Calculation For a query $x_?$, prediction $\hat{y}$ is a weighted sum:

$$\hat{y} \approx \sum_{i=1}^{N} \underbrace{\alpha_i(x_?, x_i)}_{\text{Similarity}} \cdot \underbrace{y_i}_{\text{Value}}$$

- **High** α_i**:** Context is close/relevant.
- **Low** α_i**:** Context is ignored.



Visualizing Attention: The model assigns higher weights (thicker lines) to context points closer to the query, effectively "interpolating" the smooth underlying function.

PFN: Training & Inference

Sample prior datasets $D^{(i)} \sim p(\mathcal{D})$

$$D^{(1)} = D^{(1)}_{train} \cup \{(x^{(1)}_{test}, y^{(1)}_{test})\}$$

$$\vdots$$

$$D^{(K)} = D^{(K)}_{train} \cup \{(x^{(K)}_{test}, y^{(K)}_{test})\}$$

Train the PFN by minimizing

$$-\sum_{i=1}^{K} \log q_\theta(y^{(i)}_{test} | x^{(i)}_{test}, D^{(i)}_{train})$$

PFN with parameters θ^*

Actual context dataset & test input

$$(D_{context}, x_{test})$$

$$q_{\theta^*}(y_{test} | x_{test}, D_{context}) \approx p(y_{test} | x_{test}, D_{context})$$

Samuel Müller, Noah Hollmann, *et.al* "Transformers can do bayesian inference". ICLR 2022.

One + Two Questions

Scalability

Scalability: Can PFNs perform GP-style inference for 50D regression?

One + Two Questions

Scalability

Scalability: Can PFNs perform GP-style inference for 50D regression?

No - standard PFNs don't scale beyond $\sim$10D

One + Two Questions

Scalability

Scalability: Can PFNs perform GP-style inference for 50D regression?

No - standard PFNs don't scale beyond ~10D

Attention Encoding: Localization

Why is everyone encoding $x + y$ in PFNs? How does this affect localization?

One + Two Questions

Scalability

Scalability: Can PFNs perform GP-style inference for 50D regression?

No - standard PFNs don't scale beyond $\sim$10D

Attention Encoding: Localization

Why is everyone encoding $x + y$ in PFNs? How does this affect localization?

Localization: Nearby points provide more info about the value at an unknown point than distant ones.

One + Two Questions

Scalability

Scalability: Can PFNs perform GP-style inference for 50D regression?

No - standard PFNs don't scale beyond $\sim$10D

Attention Encoding: Localization

Why is everyone encoding $x + y$ in PFNs? How does this affect localization?

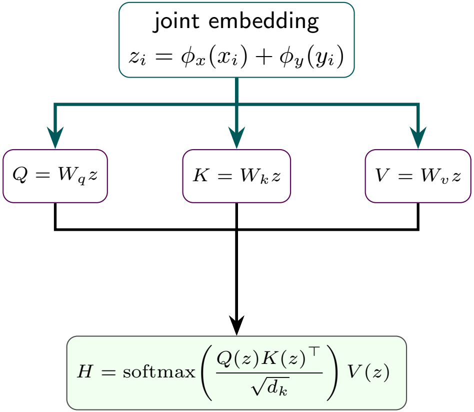
Localization: Nearby points provide more info about the value at an unknown point than distant ones.

Backbone vs Attention

Are transformers all we need, or can PFNs also be built using CNNs etc.?

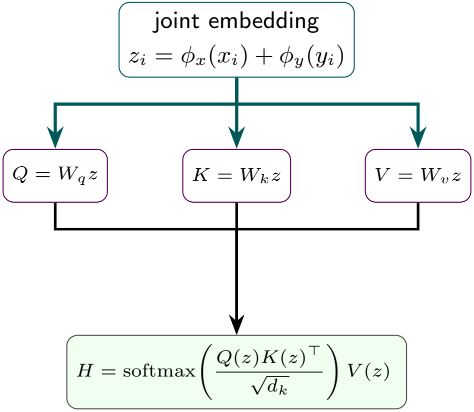
Main Idea: Decouple Input and Output

Vanilla Attention



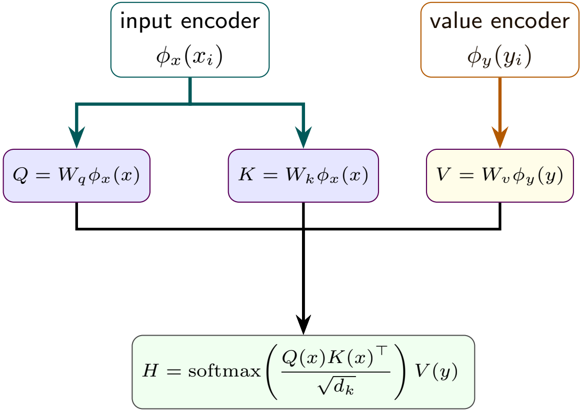
Main Idea: Decouple Input and Output

Vanilla Attention



Mixing Input & Output

Decoupled-Value Attention (DVA)



Keeping Input & Output Separate

Theorem: Enforcing Localization

Theoretical Result

For DVA with linear embeddings, the attention weight $\alpha_i(x_\star)$ (for a test input $x_\star$) assigned to a context point x_i is proportional to a **Mahalanobis RBF Kernel**:

$$\alpha_i(x_\star) \propto \exp\left(-\frac{1}{2\tau}\|(x_\star - x_i)\|_A^2\right)$$

Implication:

- As the distance $\|x_\star - x_i\|$ increases, the attention weight decays **exponentially**.
- This mathematically forces the Model to behave like a GP Kernel: localization.
- This holds true for any backbone architecture.

$\|u\|_A^2 = u^T Au$

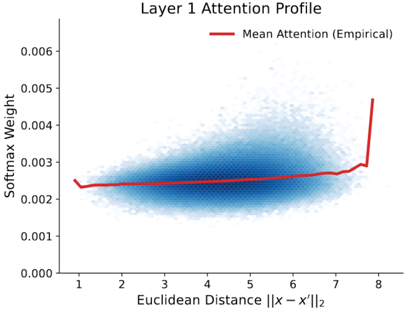
Empirical Evidence: 10D Input Space

Does the theory hold in practice? We look at the attention weights in a 10D task.

Empirical Evidence: 10D Input Space

Does the theory hold in practice? We look at the attention weights in a 10D task.

Vanilla Attention

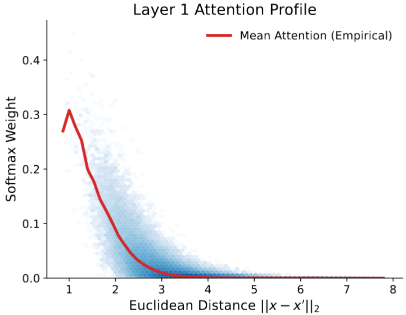


No Localization

Weights are flat/uniform.

The model fails to localize inputs.

DVA (Ours)

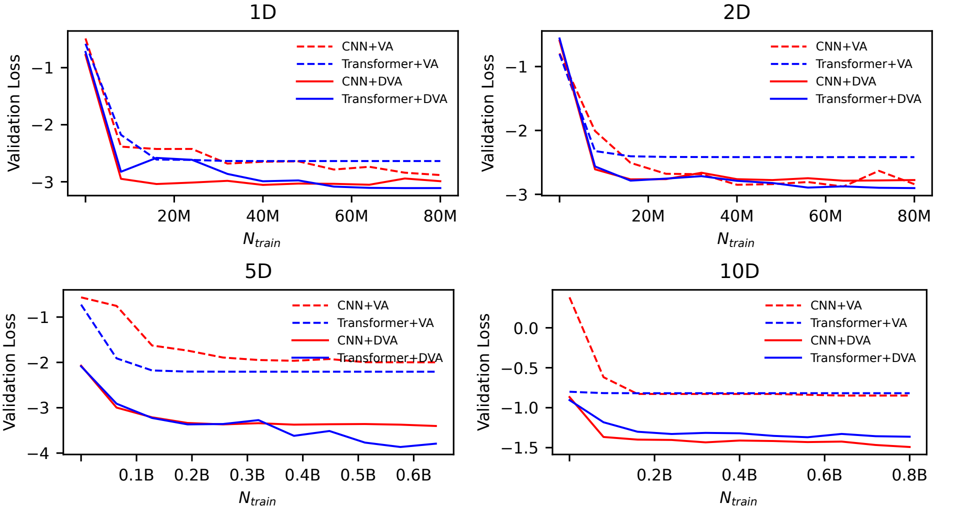


Exponential Decay

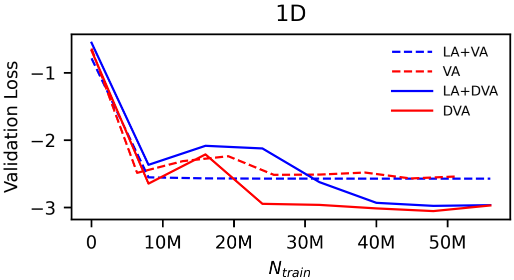
Weights drop as distance increases.

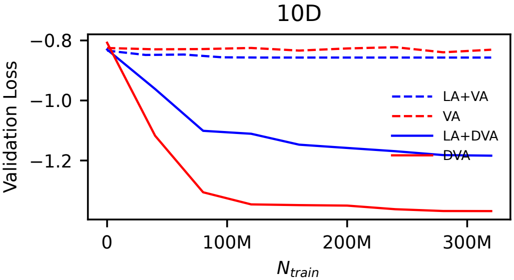
(Matches Theorem 1)

DVA Outperforms VA: Scale and Error

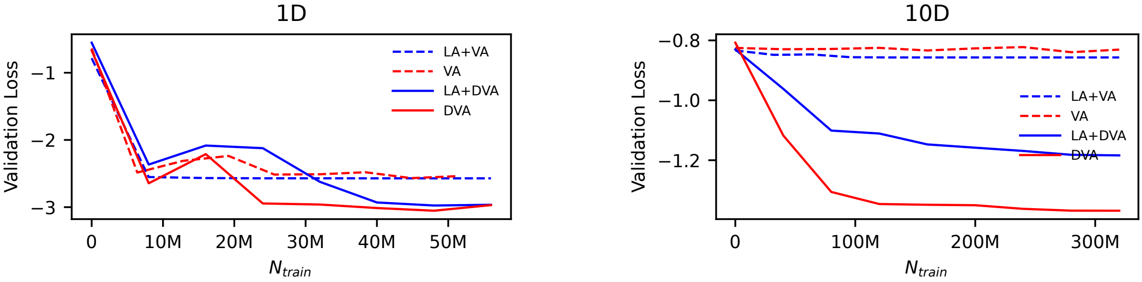


Decoupling Helps with Linear Attention Too



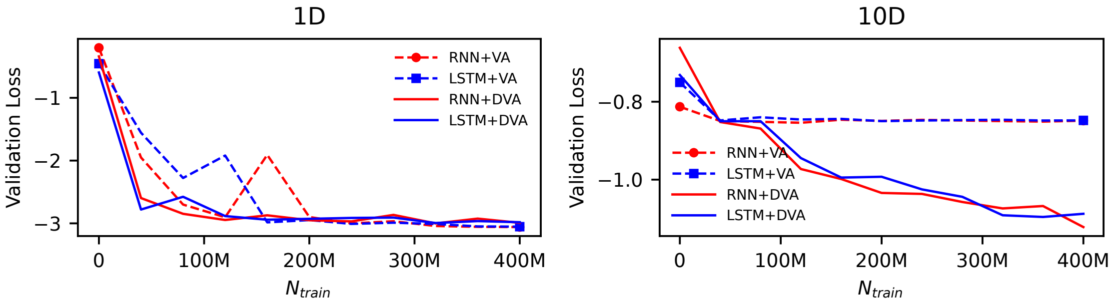


Decoupling Helps with Linear Attention Too



`Softmax DVA` $\gg$ `Linear DVA` $\gg$ `Softmax VA` $\gg$ `Linear VA`

Attention Agnostic: RNN/LSTM Results



With DVA, PFN training performance is nearly identical across backbones
: CNN, RNN, LSTM, Transformer.

64D Power-Flow Learning

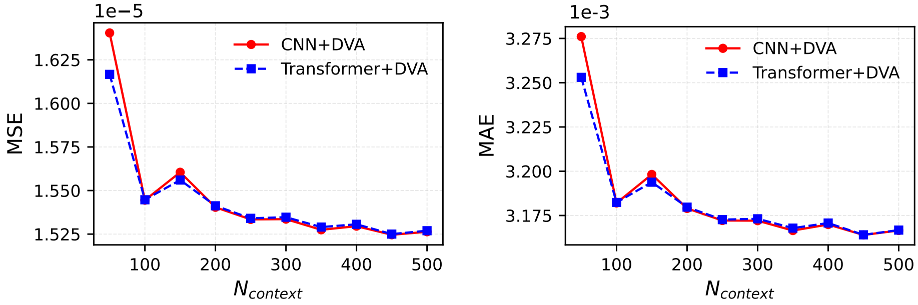
- Model: IEEE 33-bus AC power flow (64D inputs: 32 real $+$ 32 reactive loads) $\rightarrow$ 32 bus voltages.

64D Power-Flow Learning

- Model: IEEE 33-bus AC power flow (64D inputs: 32 real + 32 reactive loads) $\rightarrow$ 32 bus voltages.
- Exact GP (per-bus) yields lowest MSE/MAE but is *slow* for real-time (needs 32 GPs).
- CNN+DVA and Transformer+DVA PFNs trade small accuracy loss (MAE $\sim 10^{-3}$) for $\sim 80\times$ faster inference.

64D Power-Flow Learning

- Model: IEEE 33-bus AC power flow (64D inputs: 32 real + 32 reactive loads) $\rightarrow$ 32 bus voltages.
- Exact GP (per-bus) yields lowest MSE/MAE but is *slow* for real-time (needs 32 GPs).
- CNN+DVA and Transformer+DVA PFNs trade small accuracy loss (MAE $\sim 10^{-3}$) for $\sim 80\times$ faster inference.



The PPD of Power Flow

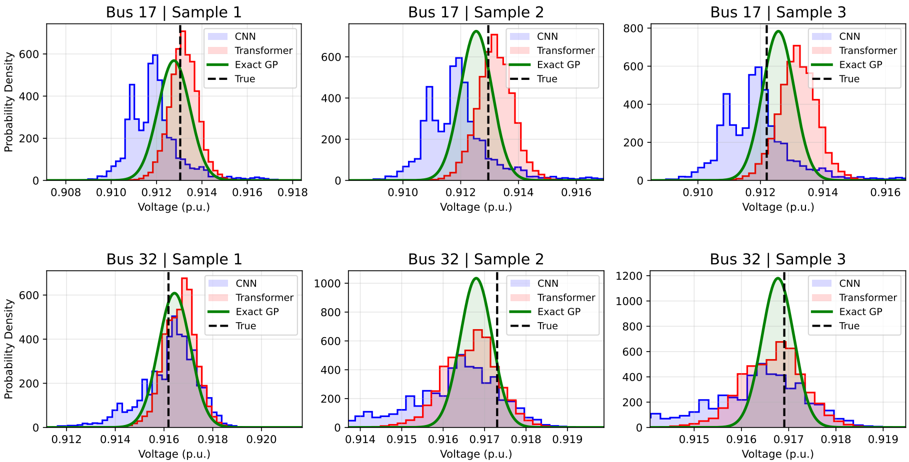
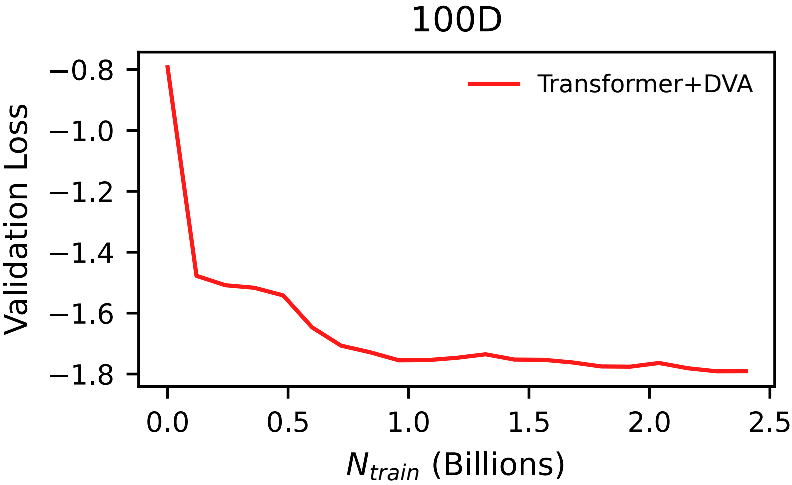


Figure: Comparing the PPD for three distinct samples for (Top) Bus 18. (Bottom) Bus 33

100D is Possible Too



Summary & Takeaways

- **Decoupled-Value Attention:** A specialized attention rule that preserves input locality and mirrors GP inference.
- **Bias Reduction:** DVA cuts PFN bias by $> 50\%$ in high-D tasks (5D, 10D) compared to vanilla attention.
- **Attention $\gg$ Backbone:** CNN/RNN/LSTM PFNs with DVA perform on par with Transformers, despite far fewer parameters.
- **Scaling to 64D:** DVA-enabled PFNs successfully learn 64D physical equations at $\sim 80\times$ GP speed.

Summary & Takeaways

- **Decoupled-Value Attention:** A specialized attention rule that preserves input locality and mirrors GP inference.
- **Bias Reduction:** DVA cuts PFN bias by $> 50\%$ in high-D tasks (5D, 10D) compared to vanilla attention.
- **Attention $\gg$ Backbone:** CNN/RNN/LSTM PFNs with DVA perform on par with Transformers, despite far fewer parameters.
- **Scaling to 64D:** DVA-enabled PFNs successfully learn 64D physical equations at $\sim 80\times$ GP speed.

Big Question

What if we could make a PFN which works for all ND GP Regressions?
A Foundation Model for Regression

Best Part of The Work!

My Coauthors are Third Year UG Students @IITR!

Kaustubh Sharma
B.Tech Electrical Engg.
kaustubh202.github.io

Simardeep Singh
B.Tech Metallurgical Engg.
linkedin.com/in/simar7220

Lab Website

Full Pre-Print

Best Part of The Work!

My Coauthors are Third Year UG Students @IITR!

Kaustubh Sharma
B.Tech Electrical Engg.
kaustubh202.github.io

Simardeep Singh
B.Tech Metallurgical Engg.
linkedin.com/in/simar7220

Lab Website

Full Pre-Print

Funding Support

Faculty Initiation Grant

Anusandhan National Research Foundation

PM Early Career Research Grant-2025

Join Our Team: PhD Openings Available!

We are looking for motivated PhD candidates to join us in July 2026!

ML for Engineering

Developing robust Machine Learning models for physical systems.

- Physics-Informed Machine Learning
- Surrogate Modeling (PFNs, GPs)
- High-Dimensional Uncertainty Quantification

AI Agents for Engineering

Building autonomous agents for complex engineering tasks.

- Reasoning & Decision Making
- LLMs for Scientific Discovery
- Control & Optimization Agents

Interested?

Send your CV and a brief statement of interest to:

`pareek@ee.iitr.ac.in`

https://psquare-lab.github.io