Graph Structure Learning with Interpretable Bayesian Neural Networks

Max Wasserman
Dept. of Computer Science
University of Rochester
mwasser6@ur.rochester.edu
github.com/maxwass

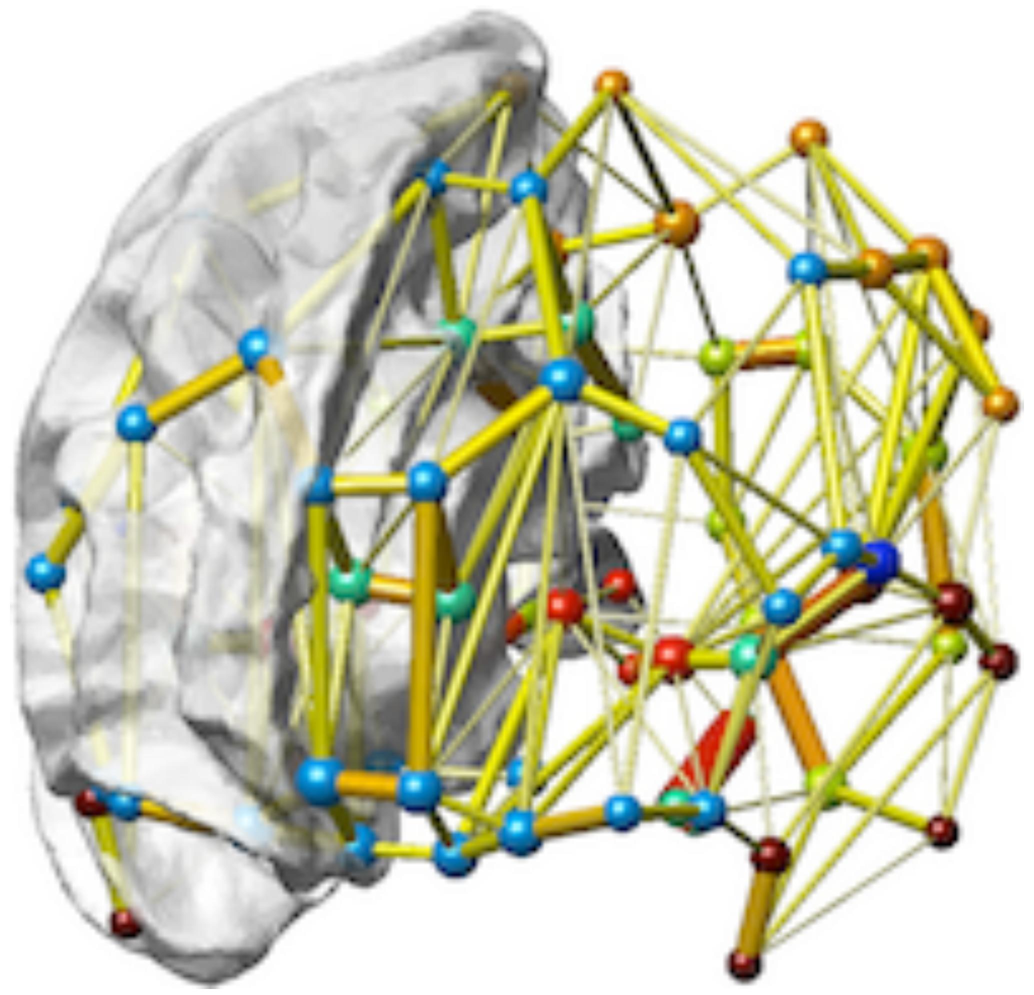
Collaborator: Gonzalo Mateos

Delft, Netherlands, June 25, 2024

UNIVERSITY of ROCHESTER

What is this talk about?

Edge Predictions with Uncertainty Quantification



- Traditional GSP: study signals/filters with known graph
    - But often graph not observed
- Graph Structure Learning (GSL): Learning graphs from nodal observations. 2 Main Approaches.
    1. Model Based
        - Solve an optimization problem [Friedman’08], [Kalofolias’16], [Saboksayr’21]
    2. Unrolling Based
        - Constructs deep network using Model Based solution iterations [Pu’21], [Wasserman’22]
    - Both only provide point estimates of graph structure
- **Goal**: point & uncertainty estimates of graph structure from nodal observation

Talk Outline

Edge Predictions with Uncertainty Quantification

- Develop a point estimate function
  - Pose and solve inverse optimization problem to estimate graph structure from nodal observations
  - ‘Unroll’ solution iterations to form a deep network
- Make it **Bayesian**
  - Parameter priors, inference, derive predictive point & uncertainty estimates over unobserved edges

Graph Signal Processing

Notation & Background

- Given graph $\mathscr{G}$ with adjacency matrix $A \in \mathbb{R}^{N \times N}$
- Collect node signals $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_p] \in \mathbb{R}^{N \times p}$, where $\bar{\mathbf{x}}_i^\top$ denotes it's i-th row.
- Form Euclidean Distance Matrix $\mathbf{E} \in \mathbb{R}^{N \times N}$, where $E_{ij} := \|\bar{\mathbf{x}}_i^\top - \bar{\mathbf{x}}_j^\top\|^2$
- Work with undirected graphs without self-loops. Reduce dimensionality.
  - $\mathbf{a} = \text{vec}[\text{triu}[\mathbf{A}]] \in \mathbb{R}^{N(N-1)/2}$, $\mathbf{e} = \text{vec}[\text{triu}[\mathbf{E}]] \in \mathbb{R}^{N(N-1)/2}$
- Total Variation of $\mathbf{X}$ w.r.t. $\mathscr{G}$ $:= \text{Trace}(\mathbf{X}^\top \mathbf{L} \mathbf{X}) = \|\mathbf{A} \circ \mathbf{E}\|_1 = 2\mathbf{a}^\top \mathbf{e}$ [Kalofolias'16]

Smoothness <—> Sparsity!

Learning Graphs from Smooth Signals

Point Estimate with Convex Optimization

- Goal: Identify undirected graph $\mathscr{G}$ such that signals $\mathbf{X}$ are smooth on $\mathscr{G}$
- Why? Many real world graph signals are smooth (i.e. Total Variation is small)
  - Examples: Sensor measurements [Chepuri'17], product ratings [Huang'18]
- How? Formulate and solve convex inverse problem

$$\boldsymbol{a}^* \in \arg\min_{\boldsymbol{a} \in \mathcal{C}} \left\{ \underbrace{\mathcal{L}_{\mathrm{data}}(\boldsymbol{a}, \boldsymbol{e})}_{\text{Data fidelity}} + \underbrace{\mathcal{L}_{\mathrm{reg}}(\boldsymbol{a})}_{\text{Regularization}} \right\}$$

Point Estimate Approaches via Optimization

SOTA Convex Formulation

Nodal Degrees
Vector: $\mathbf{A1} = \mathbf{Sa}$

$$\boldsymbol{a}^*(\boldsymbol{e}, \alpha, \beta) = \underset{\boldsymbol{a} \in \mathbb{R}^{N(N-1)/2}}{\arg\min} \left\{ 2\boldsymbol{a}^\top \boldsymbol{e} - \alpha \mathbf{1}^\top \log(\boldsymbol{S}\boldsymbol{a}) + \frac{\beta}{2}\|\boldsymbol{a}\|_2^2 + \mathbb{I}\{\boldsymbol{a} \geq 0\} \right\},$$

Data Fidelity — Smoothness

Regularizer — Isolated Nodes

Regularizer — Small Edge

Constraint — Edge Non-negativity

We can reparameterize from (α, β) to (θ, δ)

[Kalofolias'16]

$$\boldsymbol{a}^*(\boldsymbol{e}, \alpha, \beta) = \sqrt{\frac{\alpha}{\beta}}\boldsymbol{a}^*\left(\frac{1}{\sqrt{\alpha\beta}}\boldsymbol{e}, 1, 1\right) = \delta \boldsymbol{a}^*(\theta \boldsymbol{e}, 1, 1)$$

$$= \boxed{\delta} \underset{\boldsymbol{a} \in \mathbb{R}^{N(N-1)/2}}{\arg\min} \left\{ 2\boxed{\theta}\boldsymbol{a}^\top \boldsymbol{e} - \mathbf{1}^\top \log(\boldsymbol{S}\boldsymbol{a}) + \frac{1}{2}\|\boldsymbol{a}\|_2^2 + \mathbb{I}\{\boldsymbol{a} \geq 0\} \right\}$$

Point Estimate Approaches via Optimization

DPG's Independent Interpretability

Algorithm 1 Dual Proximal Gradient Descent

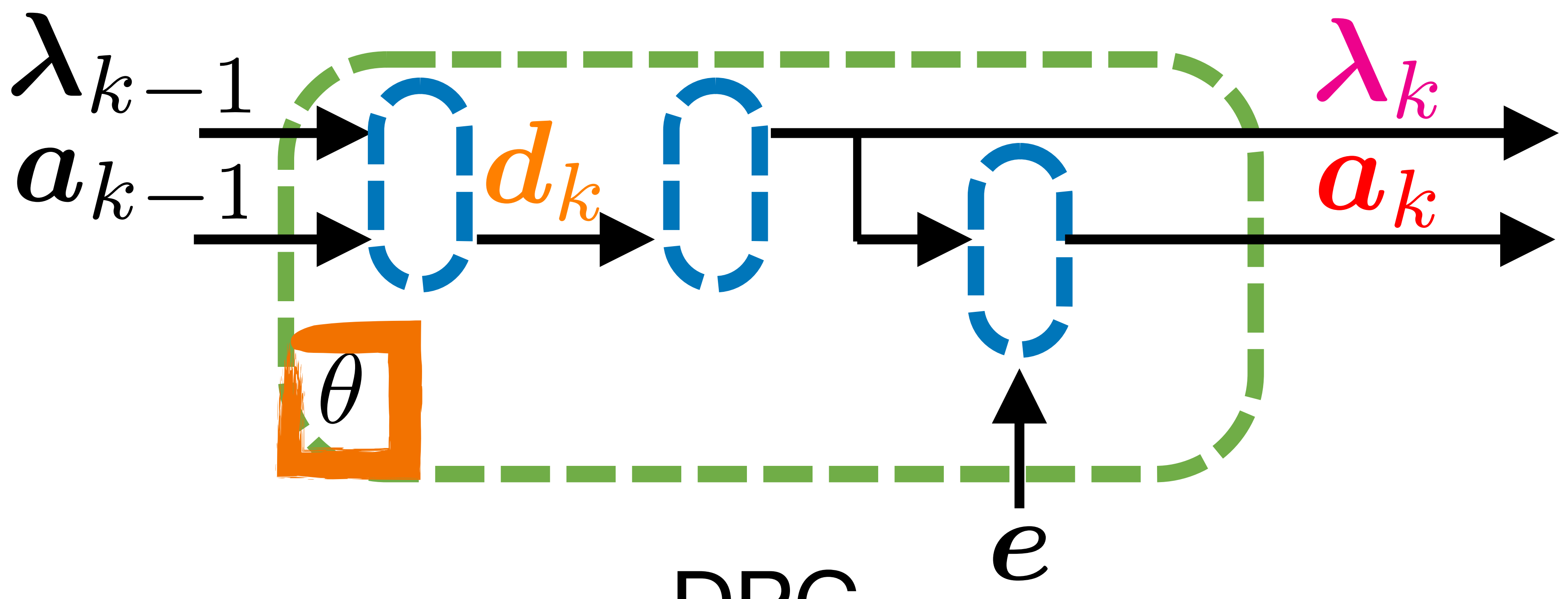
Inputs: Fixed parameters $\theta, \delta \in \mathbb{R}$ and data $\boldsymbol{e}$
Initialize: $\boldsymbol{a}_0$ and $\boldsymbol{\lambda}_0$ at random.
for $k = 1, 2, \ldots$ **do**

$$\boldsymbol{d}_k = \boldsymbol{S}\boldsymbol{a}_{k-1} - (N-1)\boldsymbol{\lambda}_{k-1}$$

$$\boldsymbol{\lambda}_k = \frac{-1}{2(N-1)}\left(\boldsymbol{d}_k - \sqrt{\boldsymbol{d}_k{}^2 + 4(N-1)\mathbf{1}}\right)$$

$$\boldsymbol{a}_k = \max\left(\mathbf{0}, \tfrac{1}{2}\boldsymbol{S}^\top\boldsymbol{\lambda}_k - \boxed{\theta}\,\mathrm{e}\right)$$

end for
Return: $\boxed{\delta}\,\boldsymbol{a}_k$



DPG

Iterations ONLY contain $\boxed{\theta}$

Sparsity pattern of solution determined by θ independently of all other parameters

Higher $\theta \rightarrow$ Higher sparsity

Defn: θ is independently interpretable w.r.t. sparsity of recovered graph

A bridge between *prior information* on sparsity and the *value of* θ

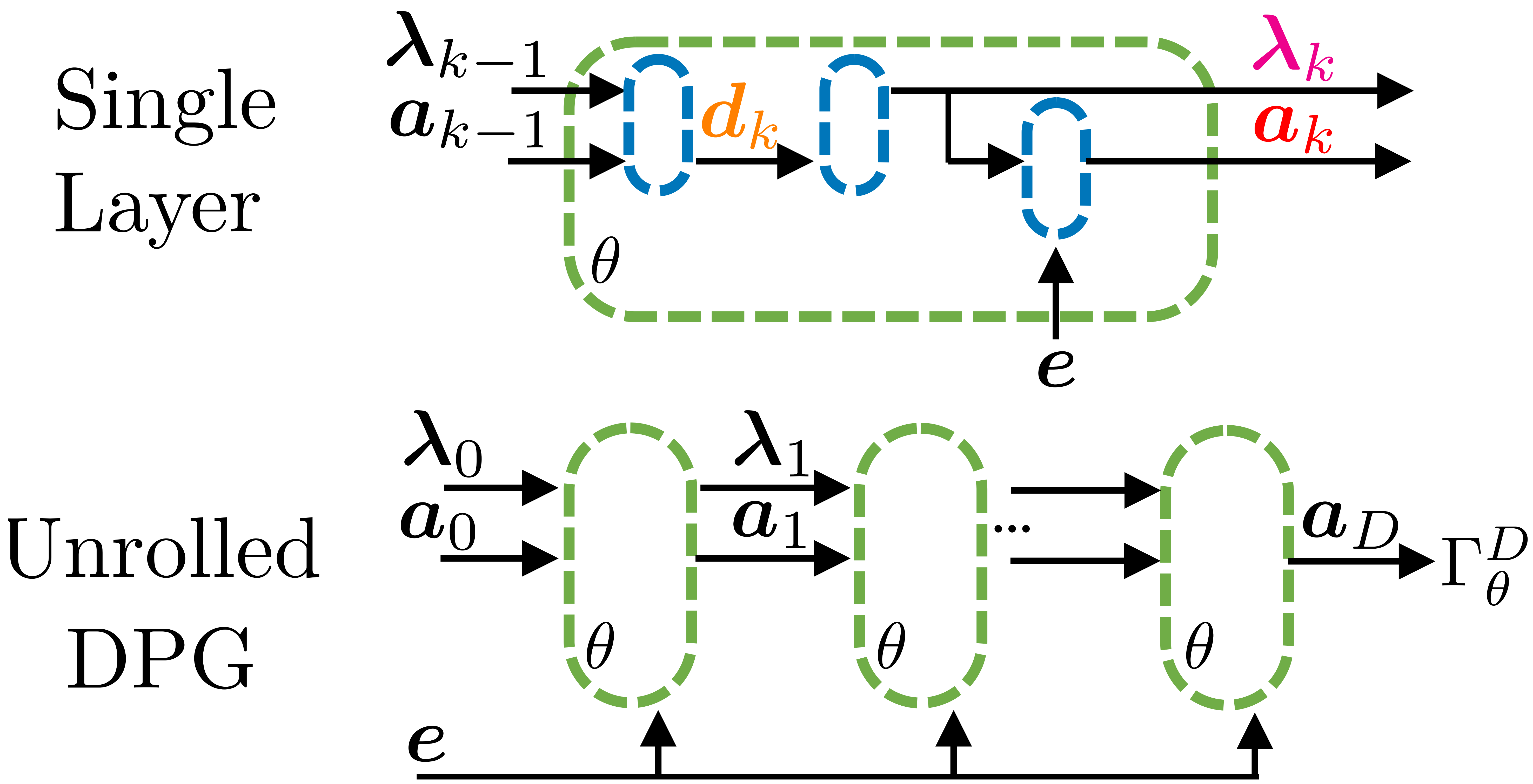
Point Estimate Approaches via Deep Unrollings

1. Assume generative process $\mathbf{X} \sim \mathscr{F}(\mathbf{A})$
   - ties data to the graph
2. Propose optimization framework & iterative solution procedure for inverse problem $\mathbf{A} = \mathscr{F}^{-1}(\mathbf{X})$

Model Based GSL

3. Unroll iterative algorithm to motivate deep network architecture

Point Estimate Approaches via Deep *Neural Network* Unrollings



- Regularization parameters —> Learnable parameters with backprop.
- Truncate after D DPG iterations. We now approximate solutions.
- The first GSL *Neural Network*!
  - Layers of linear transformations and point wise non-linearities.

Unrolled DPG: A Graph Valued NN

With Interpretability!

- Unrolled DPG is a neural network function $\Gamma_\theta^D : \boldsymbol{e} \to \boldsymbol{a}$
- Gradient w.r.t. parameters well-defined!
- Dataset $\mathcal{T} = \{\mathcal{T}_e, \mathcal{T}_a\} = \{\boldsymbol{e}^{(t)}, \boldsymbol{a}^{(t)}\}_{t=1}^T$
- We use unweighted graphs $\boldsymbol{a}^{(t)} \in \{0,1\}^{|\mathcal{E}|}$. Subtract mean b. Drive through sigmoid σ.

Final 3 Parameter
GSL Neural Network

$$\boxed{\sigma(\delta\Gamma_\theta^D(\boldsymbol{e}) - b); \Theta = \{\theta, \delta, b\}}$$

- Bernoulli likelihood: Unrolling encodes the mean.

Bayesian Neural Networks (BNN)
Background

- A Bayesian NN: a NN with stochastic weights.
- Posterior Distribution: Distribution over weights conditioned on observed data.
- Pushing posterior distribution through the NN produces a distribution over predictions.
- We can use this distribution to derive a measure of uncertainty.
- Key Ingredients
  - Weight Prior $p(\Theta)$
  - Likelihood $p(\mathcal{T}_a \mid \mathcal{T}_e, \Theta)$
  - Posterior $p(\Theta \mid \mathcal{T})$

$$p(\Theta \mid \mathcal{T}) \propto p(\mathcal{T}_a \mid \mathcal{T}_e, \Theta) p(\Theta)$$

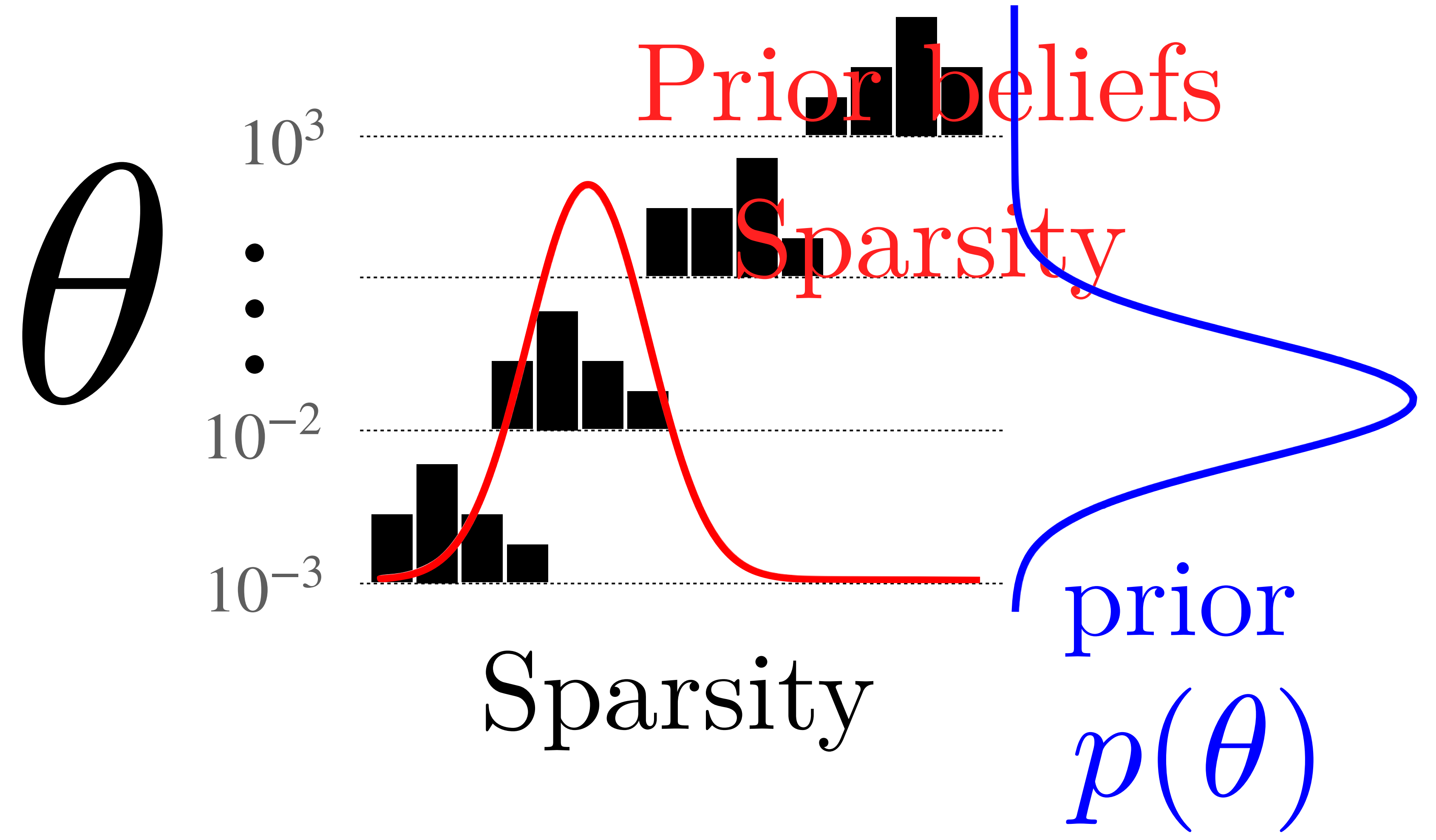
$$p(\tilde{\boldsymbol{a}} \mid \tilde{\boldsymbol{e}}, \mathcal{T}) = \int p(\tilde{\boldsymbol{a}} \mid \tilde{\boldsymbol{e}}, \Theta) p(\Theta \mid \mathcal{T}) \mathrm{d}\Theta$$

Difficulties
1. How do we set the prior?
2. How do we approximate the posterior?

Producing a Bayesian Neural Network

Informative Prior Over Parameters

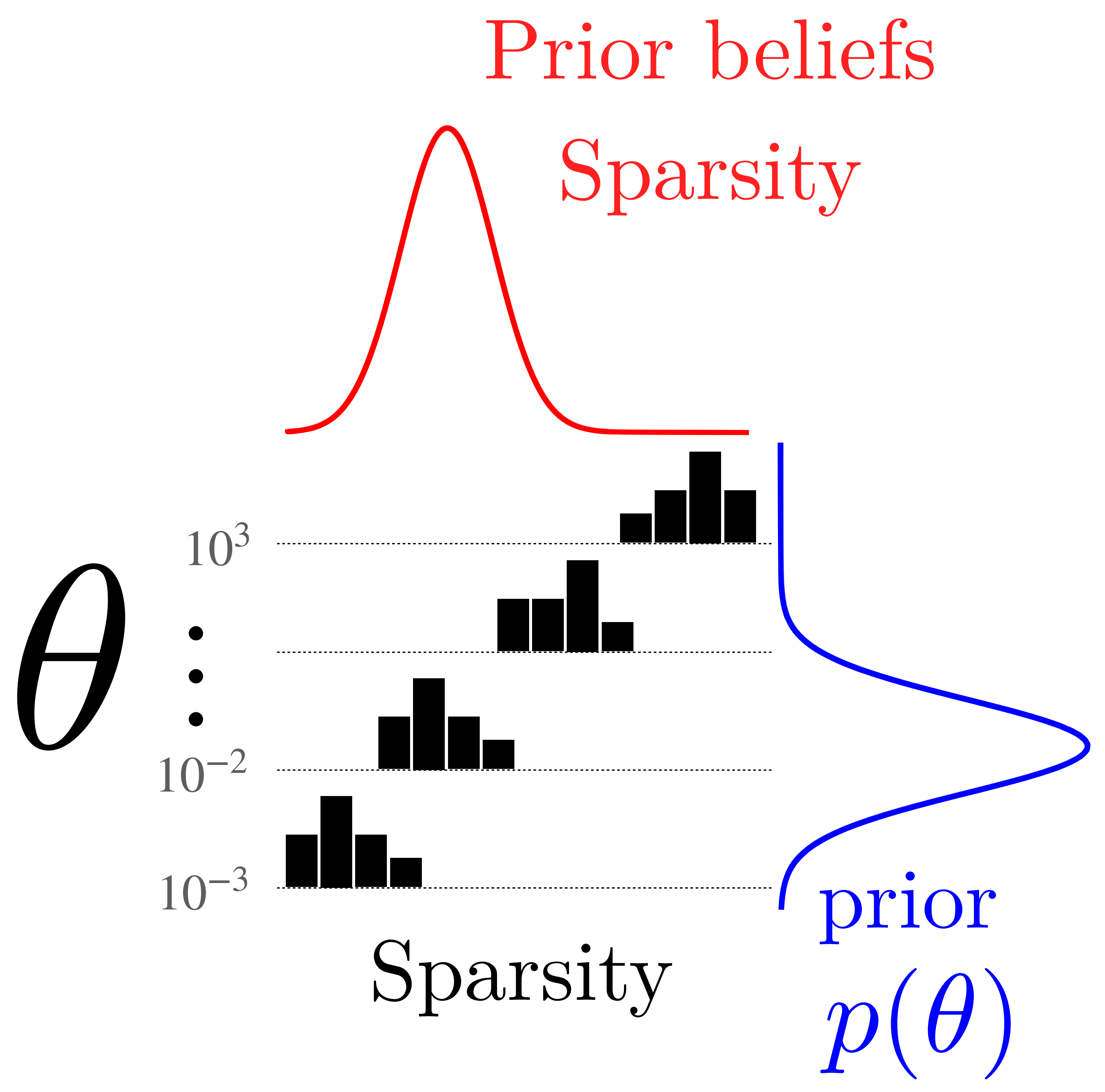
- BNNs require parameter priors $p(\Theta)$.
- To construct $p(\Theta)$:
  - Use independent interpretability of θ.
  - Subset of inputs $\mathcal{T}_e$.
  - prior beliefs over sparsity —> prior distribution over θ.
- Weakly informative prior for $p(\delta, b)$.
  - Inspect edge weight magnitudes at performant θ's.



Producing a Bayesian Neural Network

Informative Prior Over Parameters

- BNNs require parameter priors $p(\Theta)$.
- To construct $p(\Theta)$:
  - Use independent interpretability of θ.
  - Subset of inputs $\mathcal{T}_e$.
  - prior beliefs over sparsity —> prior distribution over θ.
- Weakly informative prior for $p(\delta, b)$.
  - Inspect edge weight magnitudes at performant θ's.



GSL with Interpretable BNNs

Bayesian Modeling: Posterior Predictive

1. Inference: condition on the data $\underline{\Theta^{(m)}} \sim p(\Theta \mid \mathcal{T})$ "Posterior Samples"

   Via Hamiltonian Monte Carlo

2. Marginalize out parameters $p(\tilde{\boldsymbol{a}} \mid \tilde{\boldsymbol{e}}, \mathcal{T}) \approx \frac{1}{M} \sum_{m=1}^{M} p(\tilde{\boldsymbol{a}} \mid \tilde{\boldsymbol{e}}, \underline{\Theta^{(m)}})$ "Posterior Predictive" over test sample $(\tilde{\boldsymbol{e}}, \tilde{\boldsymbol{a}})$

3. Which we sample from … $\tilde{\boldsymbol{a}}^{(m)} \sim p(\tilde{\boldsymbol{a}} \mid \tilde{\boldsymbol{e}}, \underline{\Theta^{(m)}})$

4. To produce edge-wise point and uncertainty estimates!

Point ('pred. mean')

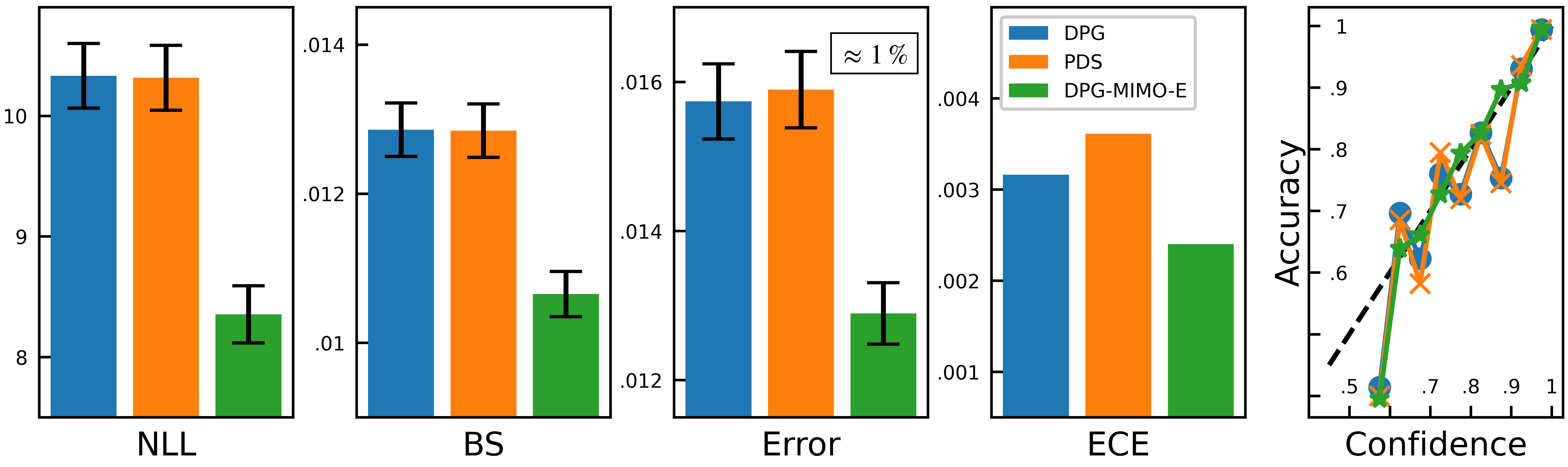
$$\boxed{\mathbb{E}[\tilde{a}_i \mid \tilde{\boldsymbol{e}}, \mathcal{T}] \approx \frac{1}{M} \sum_{m=1}^{M} \tilde{a}_i^{(m)}}$$

Uncertainty ('pred. stdv.')

$$\boxed{\mathrm{Var}[\tilde{a}_i \mid \tilde{\boldsymbol{e}}, \mathcal{T}]^{\frac{1}{2}} \approx \left[\frac{1}{M} \sum_{m=1}^{M} (\tilde{a}_i^{(m)} - \mathbb{E}[\tilde{a}_i \mid \tilde{\boldsymbol{e}}, \mathcal{T}])^2\right]^{\frac{1}{2}}}$$

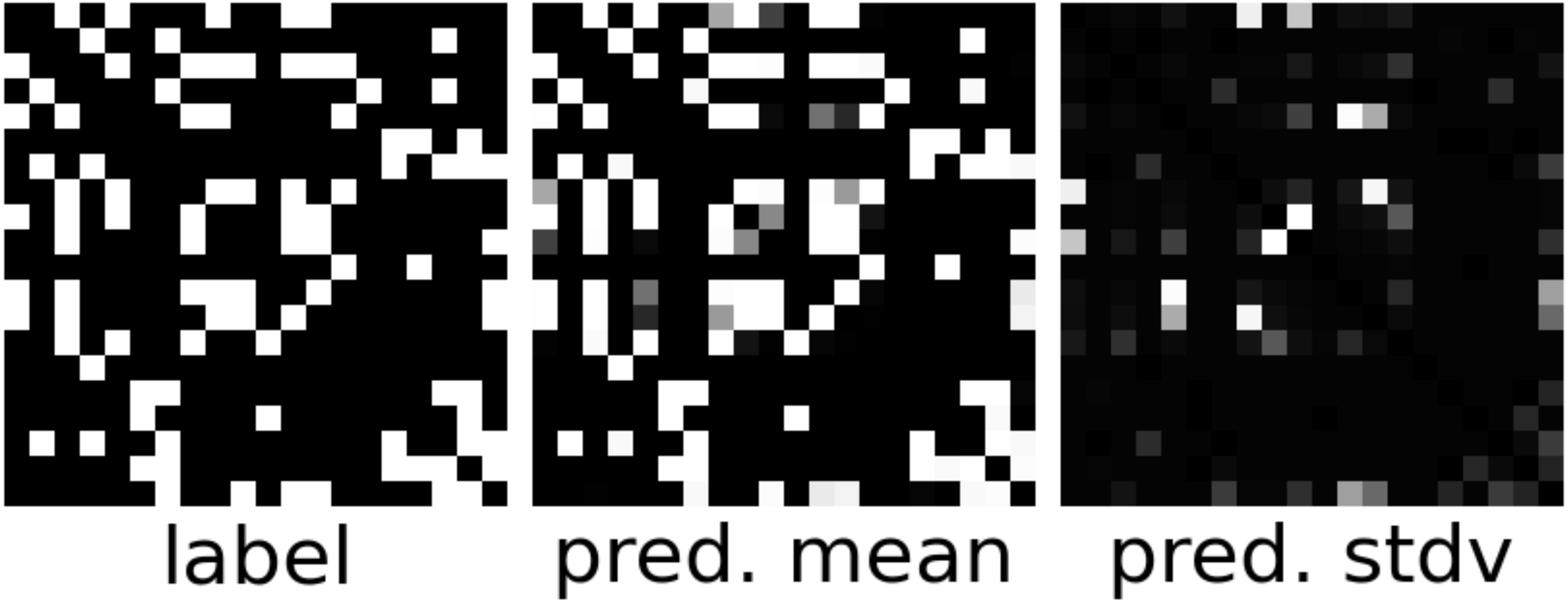
GSL with Interpretable BNNs

Synthetic Evaluation



GSL with Interpretable BNNs

Synthetic Evaluation

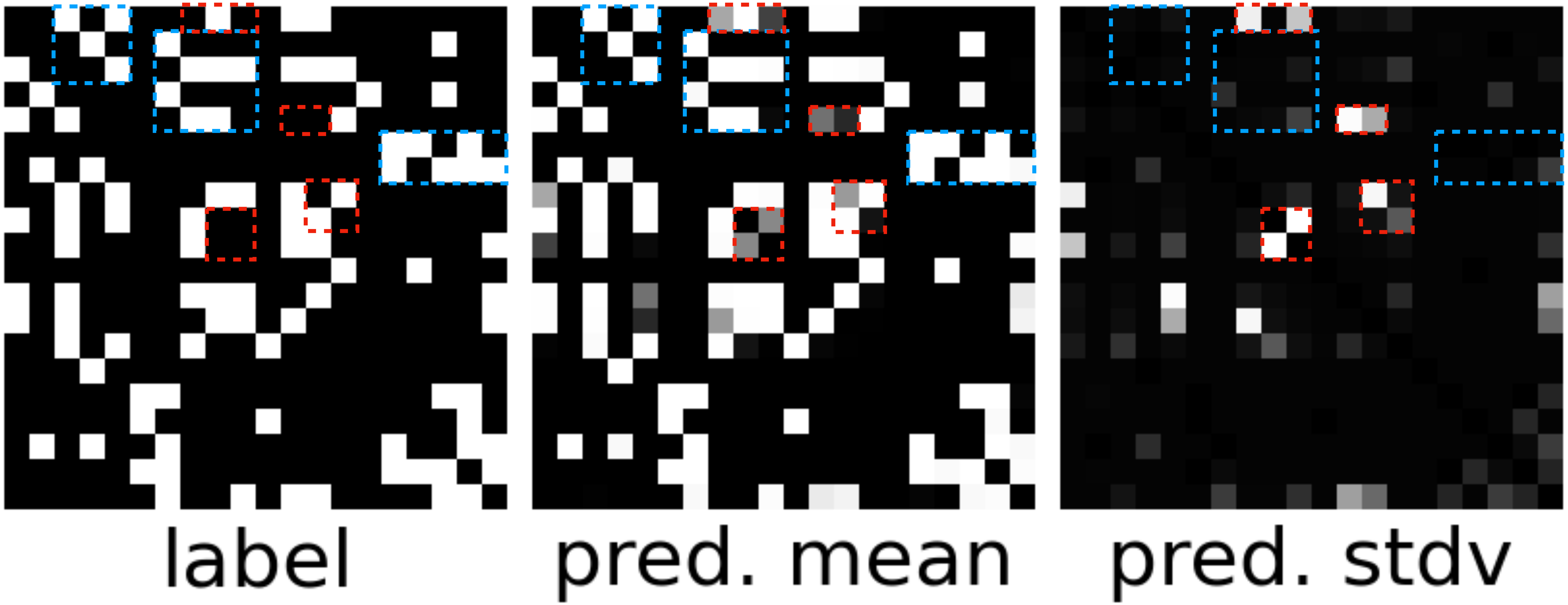


GSL with Interpretable BNNs

Synthetic Evaluation

Strong Recovery

Some Mistakes

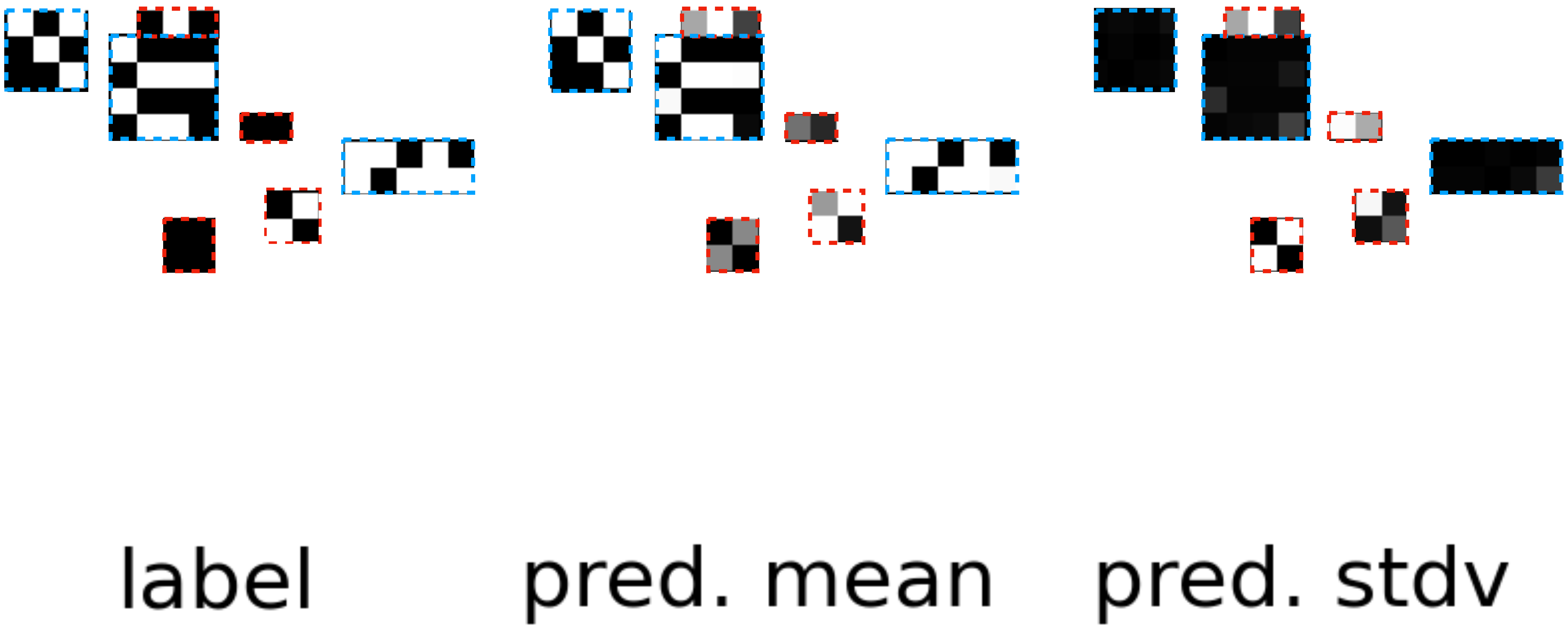


GSL with Interpretable BNNs

Synthetic Evaluation

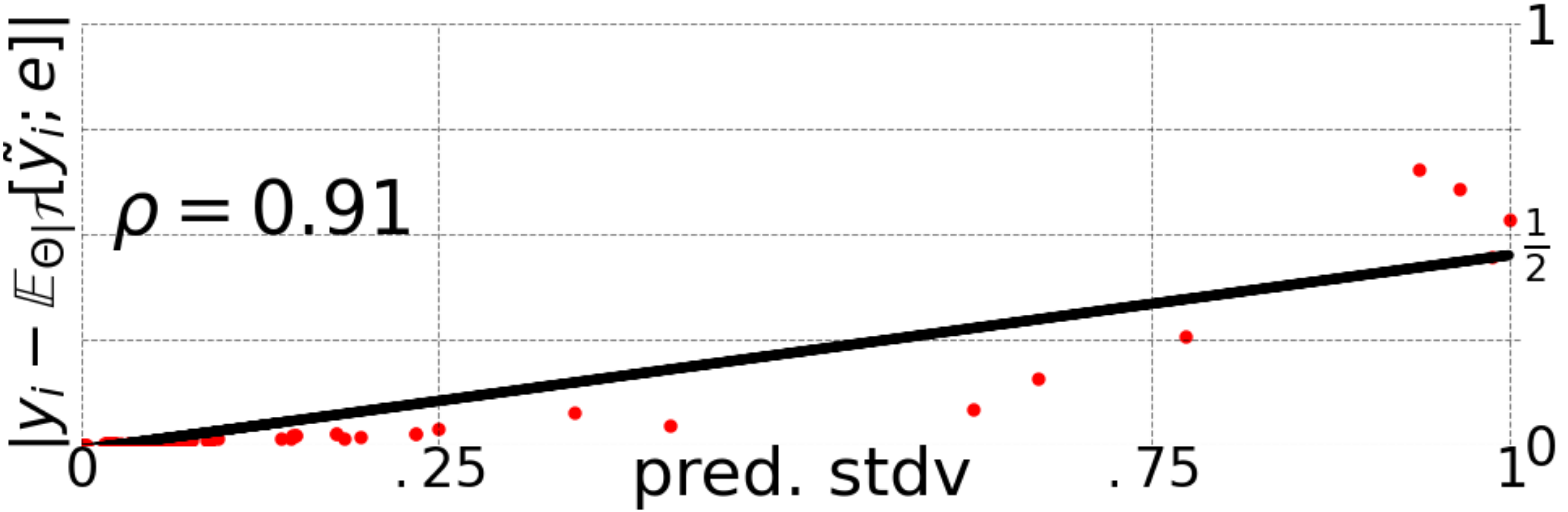
Strong Recovery

Some Mistakes



GSL with Interpretable BNNs

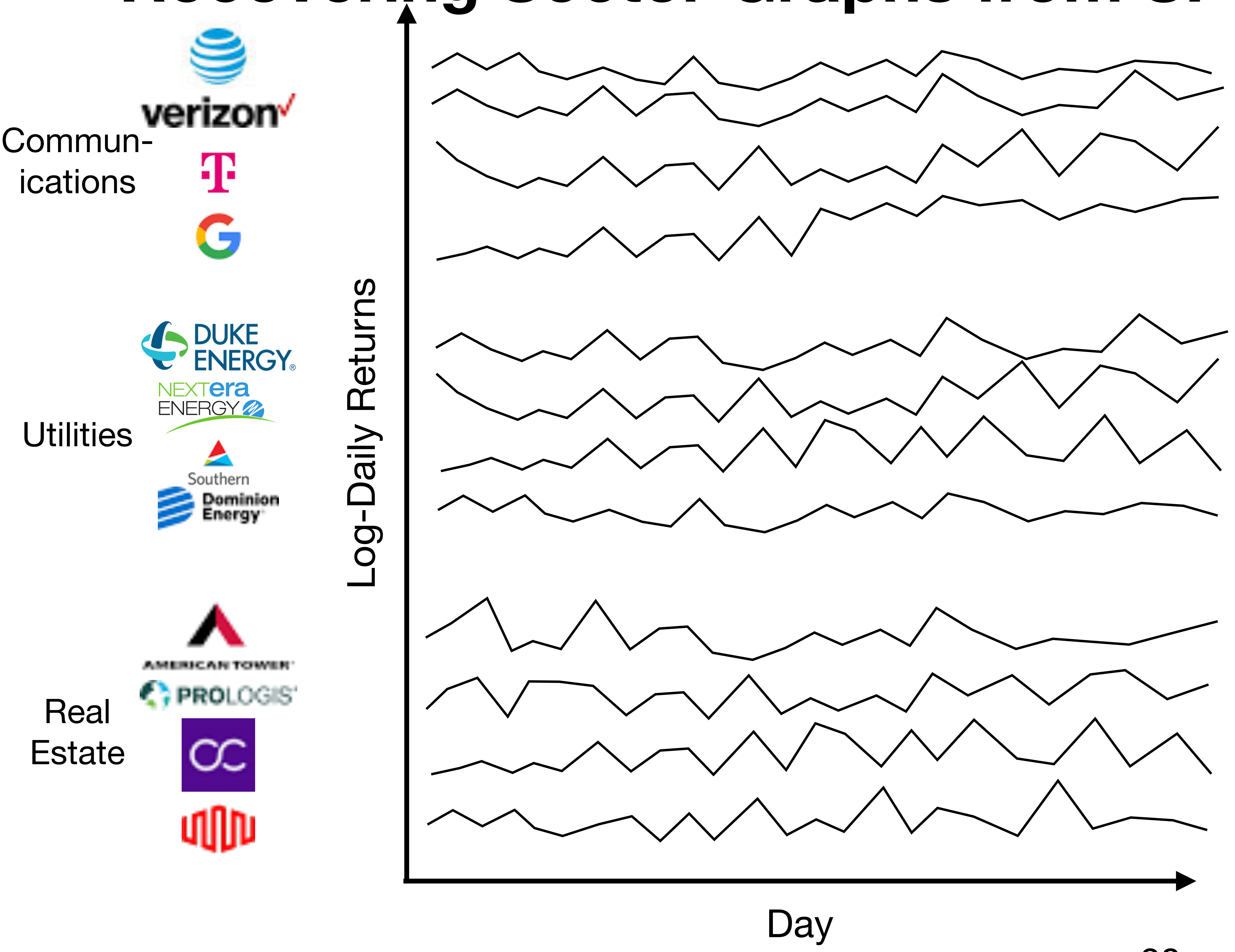
Synthetic Evaluation



GSL with Interpretable BNNs

Recovering Sector Graphs from SP500 stock time series

[Cardoso'23]

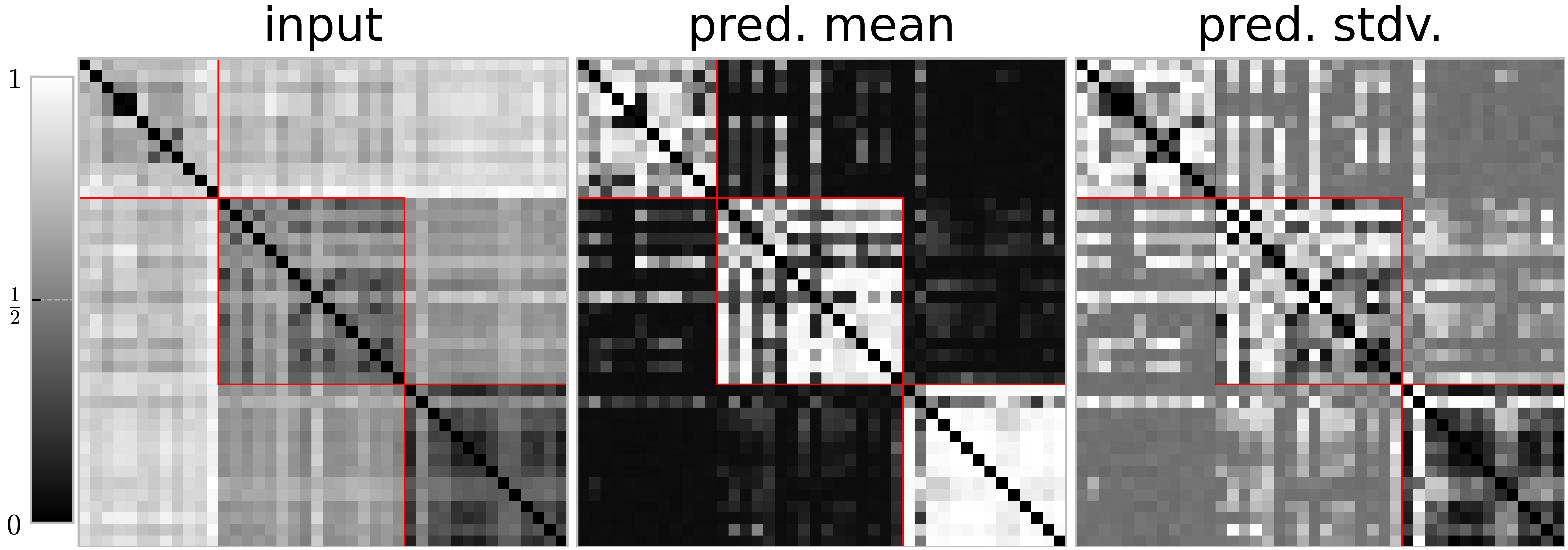


- Split stocks evenly into train/test
- For each split
  - Input: 1 - |Pearson correlation matrix|
  - Label: $A_{ij} = 1$ if stocks i & j in same sector, 0 otherwise.
    - = Block Diagonal with 3 blocks
- Run inference on DPG using this single training sample

GSL with Interpretable BNNs

Evaluation: Recovering Sector Graphs from SP500 stock time series

- Error and Uncertainty Pearson Correlation: **0.70**



Closing Remarks

- We analyze recovering graphs from nodal observations
- DPG: simple iterations which produce a true neural network when unrolled
- Structure of DPG makes a parameter independently interpretable
- Use structure to construct informative parameter priors using priors on sparsity
- Conditioning on observed data —> posterior distribution $p(\Theta \mid \mathcal{T})$
- On unseen nodal observations X, push posterior distribution through Unrolled DPG to produce a distribution over edges $p(\tilde{a} \mid \mathcal{T}; \tilde{e})$
  - From which we recover point and uncertainty estimates
- Future Work: Scaling & Variational Inference

For more…

- ArXiv
  - https://arxiv.org/abs/2406.14786
- Github repo
  - github.com/maxwass/gsl-bnn

Graph Structure Learning with Interpretable Bayesian Neural Networks

Max Wasserman *mwasser6@cs.rochester.edu*
Department of Computer Science
University of Rochester

Gonzalo Mateos *gmateosb@ece.rochester.edu*
Department of Electrical and Computer Engineering
University of Rochester

Abstract

Graphs serve as generic tools to encode the underlying relational structure of data. Often this graph is not given, and so the task of inferring it from nodal observations becomes