UCL Centre for Inverse Problems in Imaging

Bayesian Deep Learning via Subnetwork Inference

Erik Daxberger

Eric Nalisnick

James Allingham

Javier Antorán

José Miguel Hernández-Lobato

UNIVERSITY OF CAMBRIDGE
Max Planck Institute for Intelligent Systems
UNIVERSITY OF AMSTERDAM

Summary

Summary

We show how a Bayesian deep learning method

Summary

We show how a Bayesian deep learning method that does ***expressive inference***

Summary

We show how a Bayesian deep learning method
that does ***expressive inference***
over a carefully chosen ***subnetwork***
within a neural network,

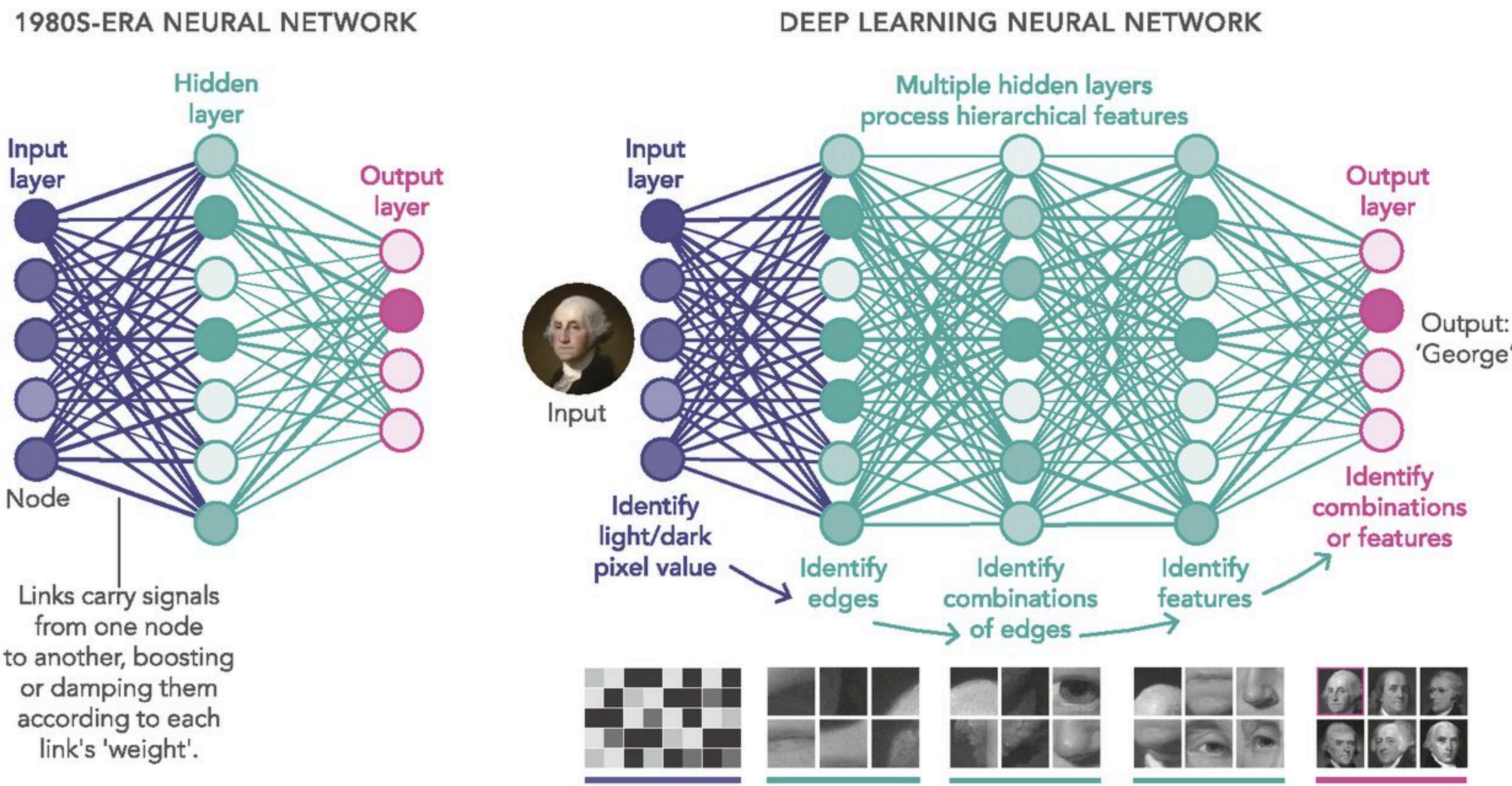
Summary

We show how a Bayesian deep learning method
that does ***expressive inference***
over a carefully chosen ***subnetwork***
within a neural network,
performs better

Summary

We show how a Bayesian deep learning method
that does ***expressive inference***
over a carefully chosen ***subnetwork***
within a neural network,
performs better
than doing crude inference over the full network.

Preliminaries: Deep Learning



Waldrop 2019, "What are the limits of deep learning?"

Issues with Deep Learning

Issues with Deep Learning

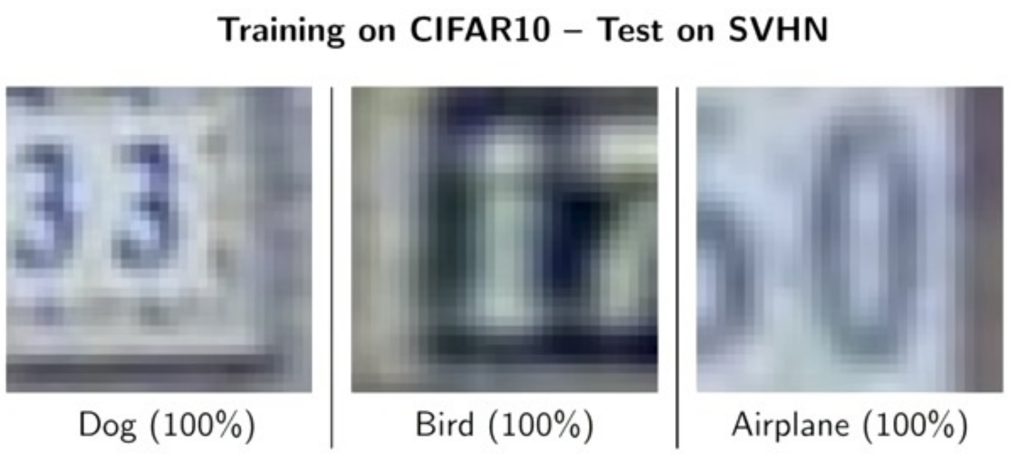
Overconfidence

Training on CIFAR10 – Test on SVHN

https://vitalab.github.io/article/2019/07/11/overconfident.html

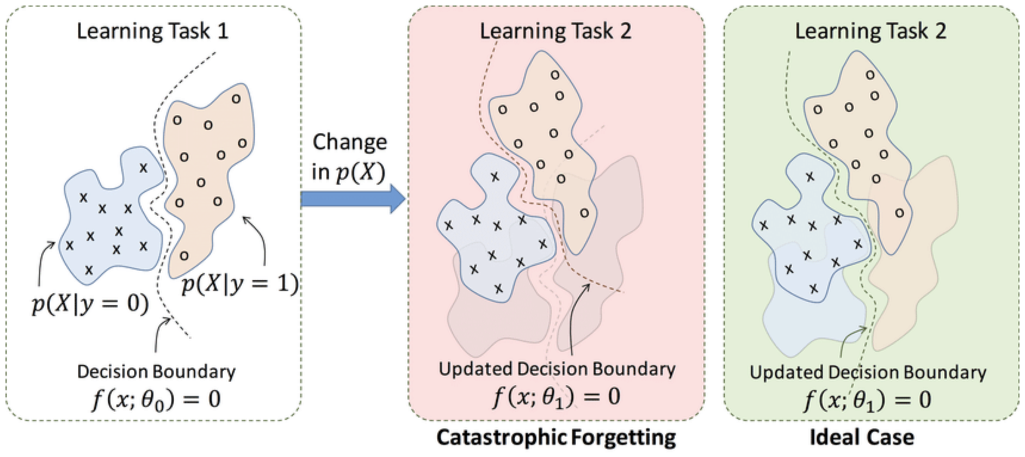
Issues with Deep Learning

Overconfidence



https://vitalab.github.io/article/2019/07/11/overconfident.html

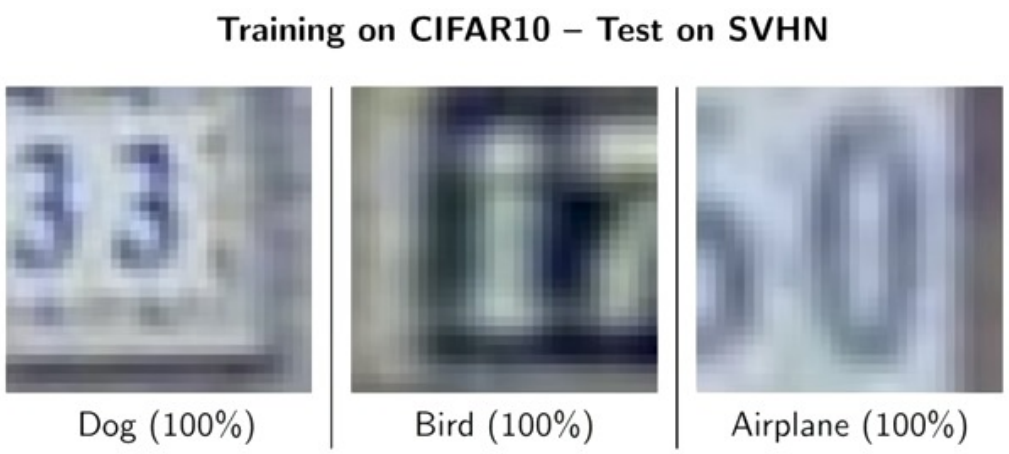
Catastrophic Forgetting



Kolouri et al. 2019, "Attention-Based Selective Plasticity"

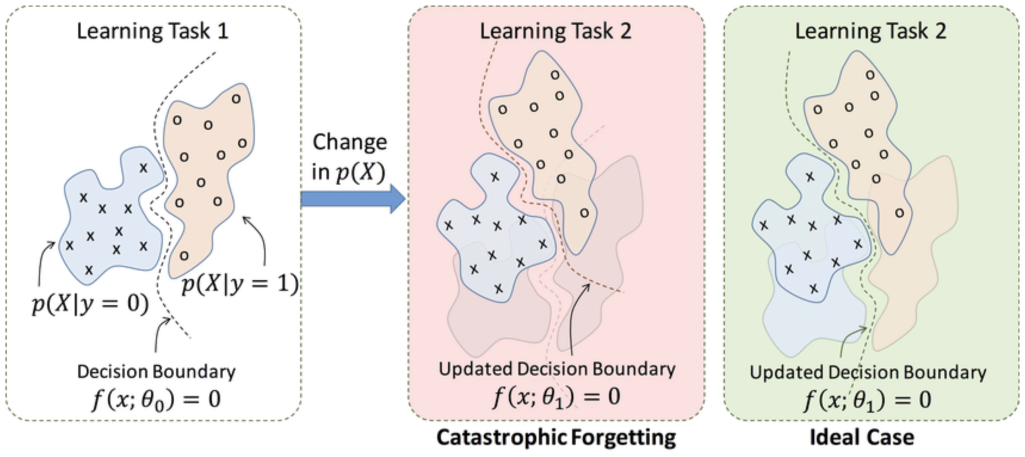
Issues with Deep Learning

Overconfidence



https://vitalab.github.io/article/2019/07/11/overconfident.html

Catastrophic Forgetting

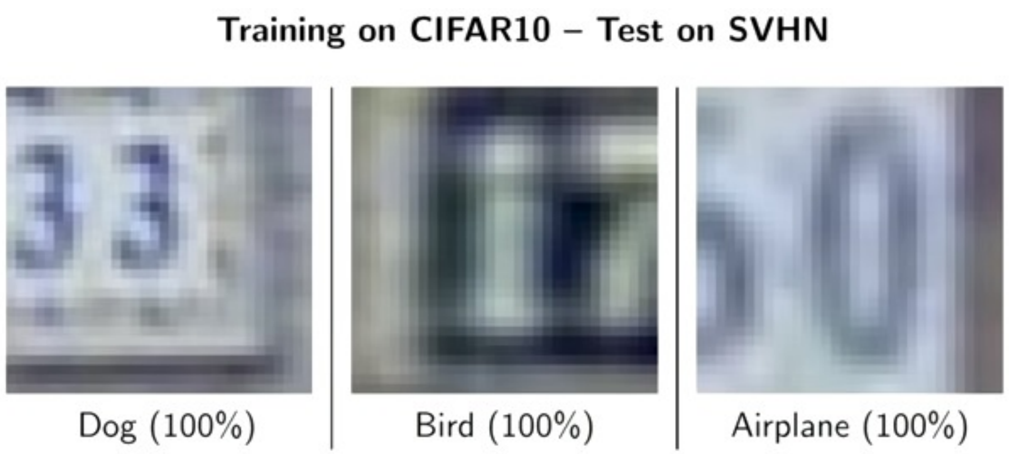


Kolouri et al. 2019, "Attention-Based Selective Plasticity"

Data Inefficiency

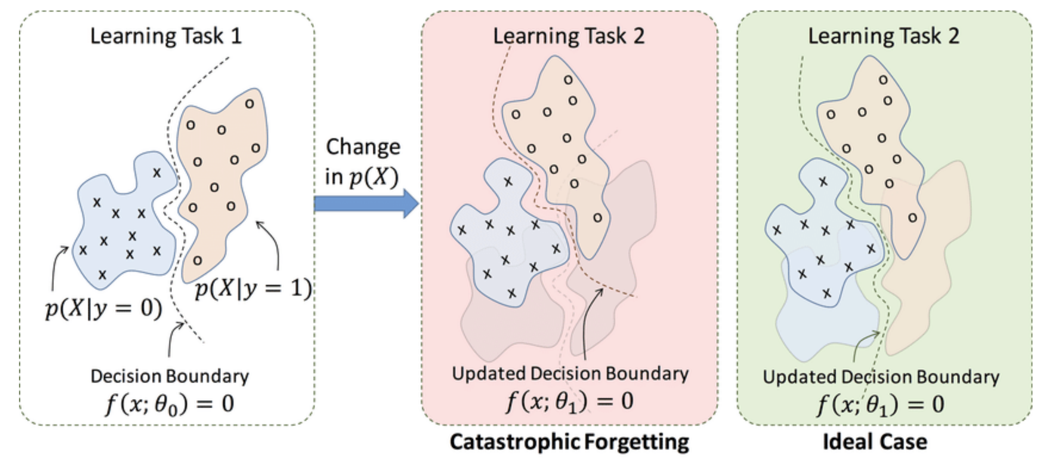
Issues with Deep Learning

Overconfidence



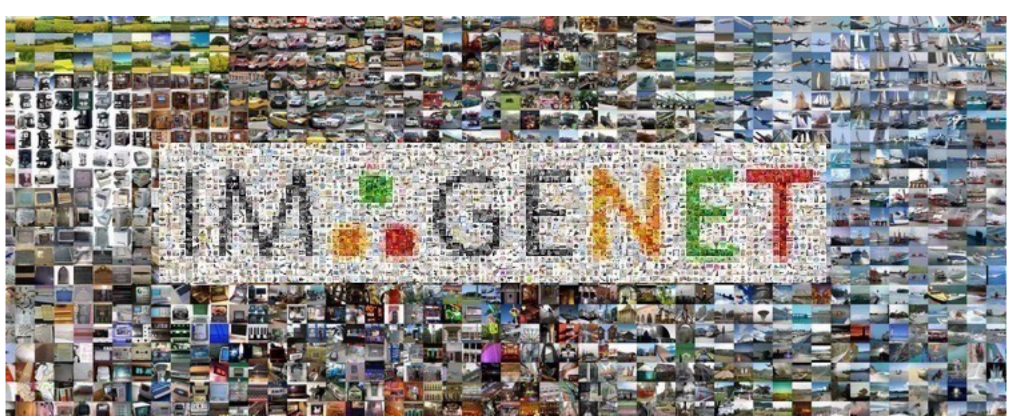
https://vitalab.github.io/article/2019/07/11/overconfident.html

Catastrophic Forgetting



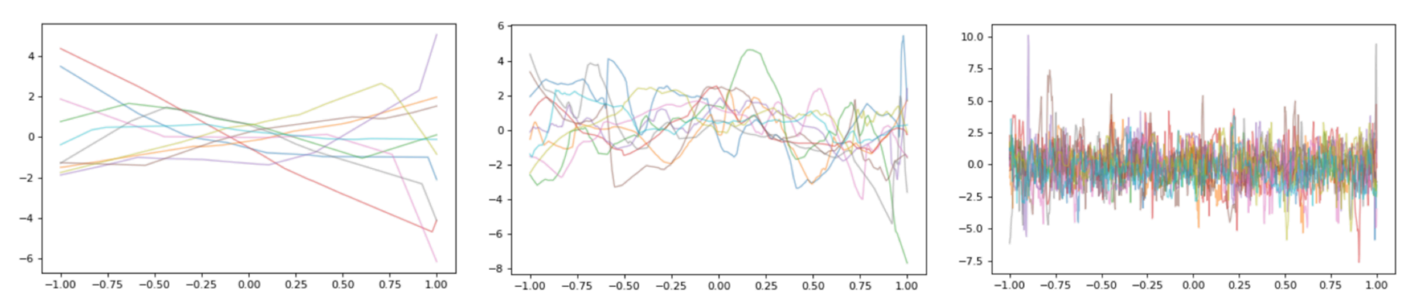
Kolouri et al. 2019, "Attention-Based Selective Plasticity"

Data Inefficiency



Model Selection

1 Hidden Layer | 5 Hidden Layer | 20 Hidden Layer



Probabilistic Inference: A biased coin

Likelihood ↘ ↙ Prior

$$p(\mathbf{W} \mid \mathscr{D}) = \frac{p(\mathscr{D} \mid W)p(W)}{p(\mathscr{D})}$$

Probabilistic Inference: A biased coin

Likelihood

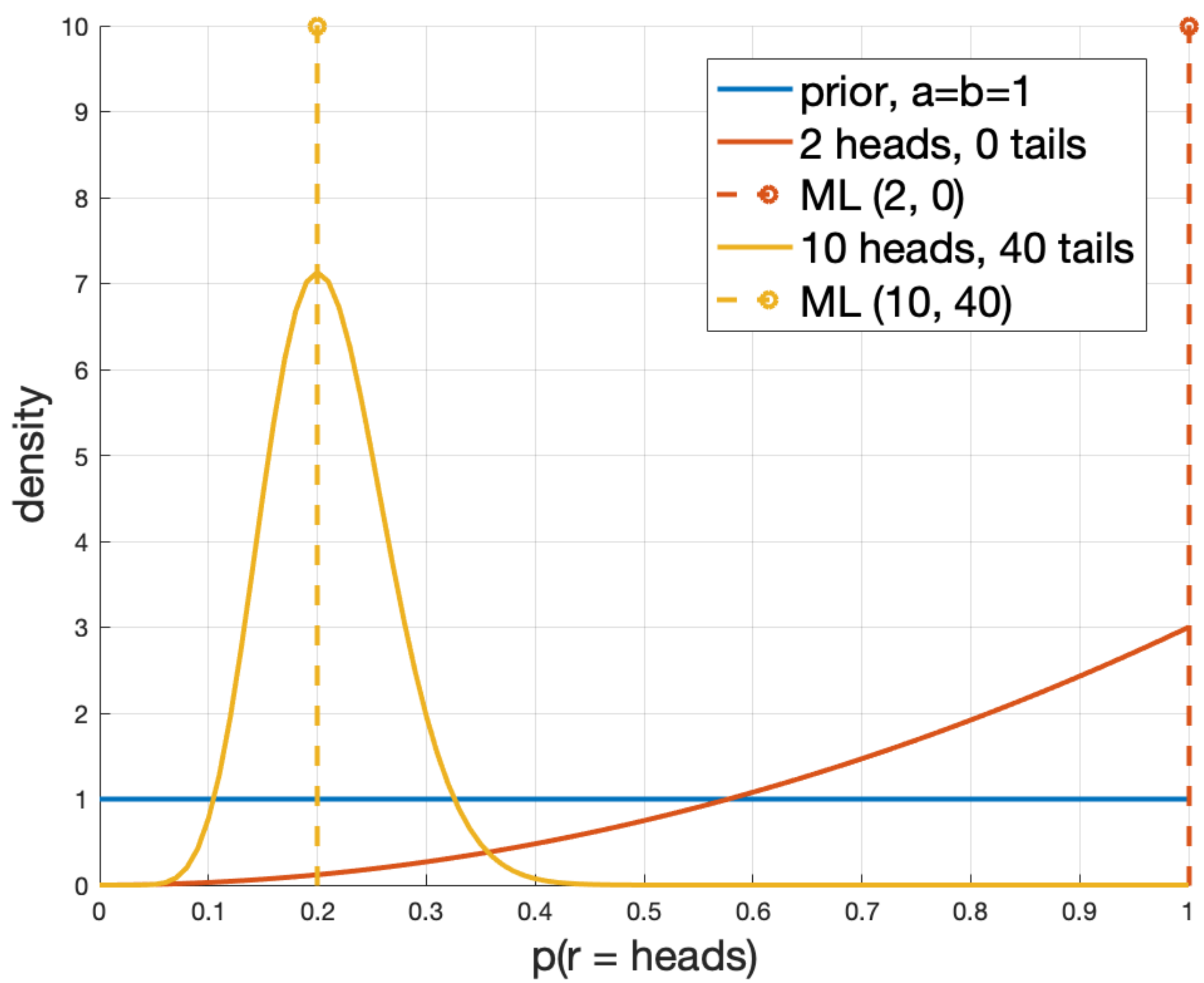
Prior

$$p(\mathbf{W} \mid \mathscr{D}) = \frac{p(\mathscr{D} \mid W)p(W)}{p(\mathscr{D})}$$

Probabilistic Inference: A biased coin

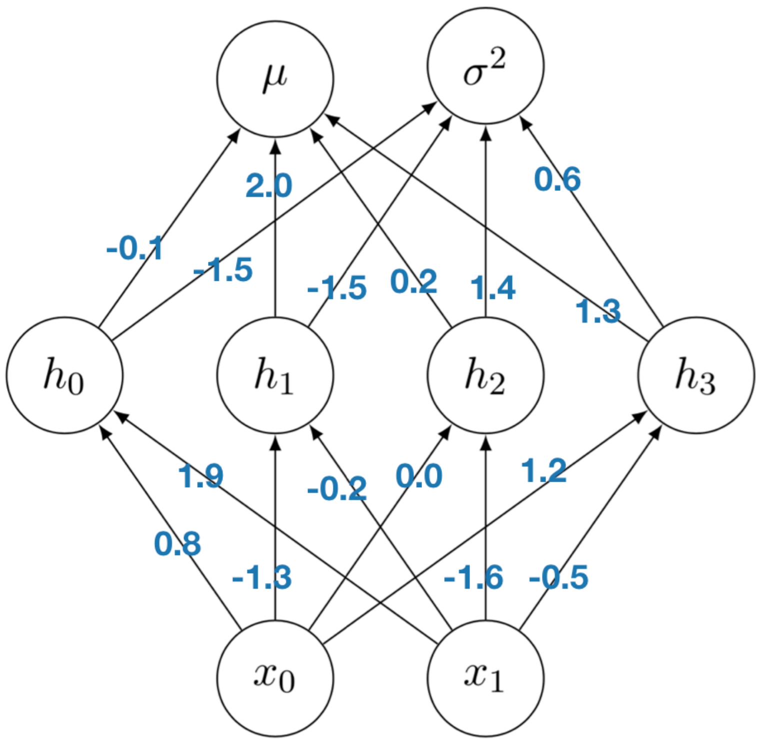
Likelihood ↘ Prior ↙

$$p(\mathbf{W} \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid W)p(W)}{p(\mathcal{D})}$$



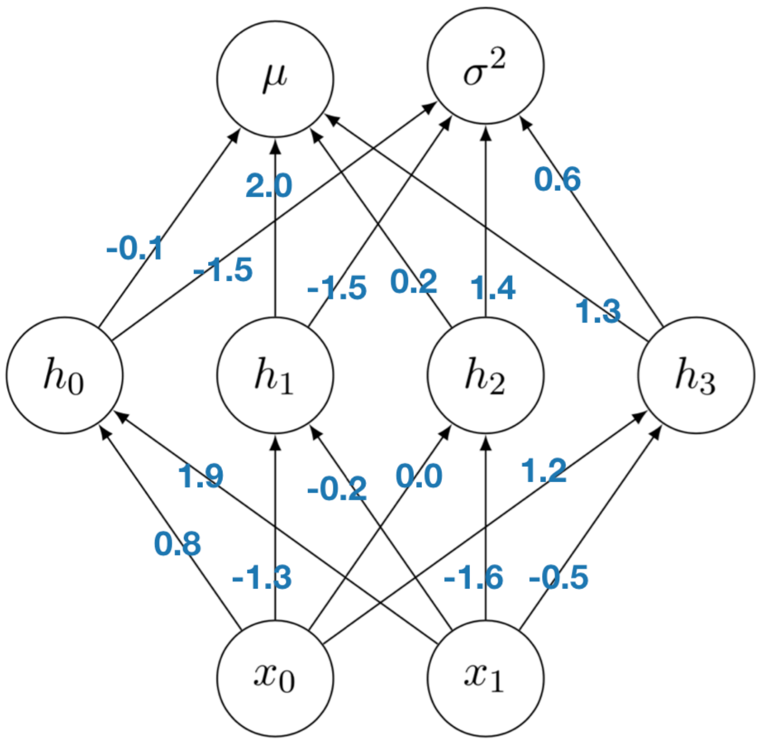
Uncertainty Estimation

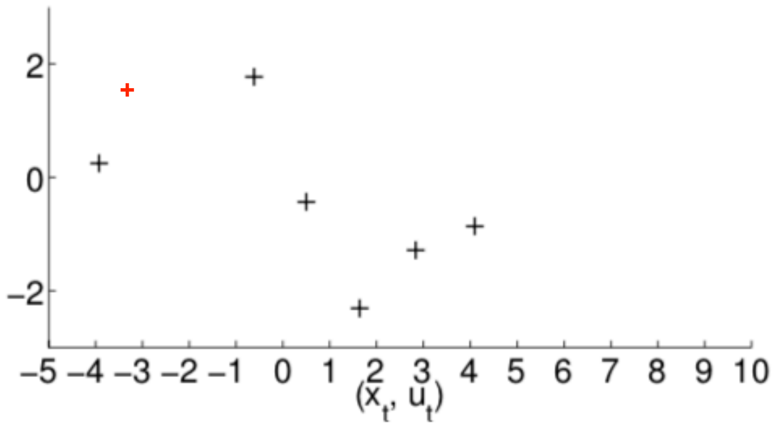
Different Weight Configurations yield Diverse Predictions:



Uncertainty Estimation

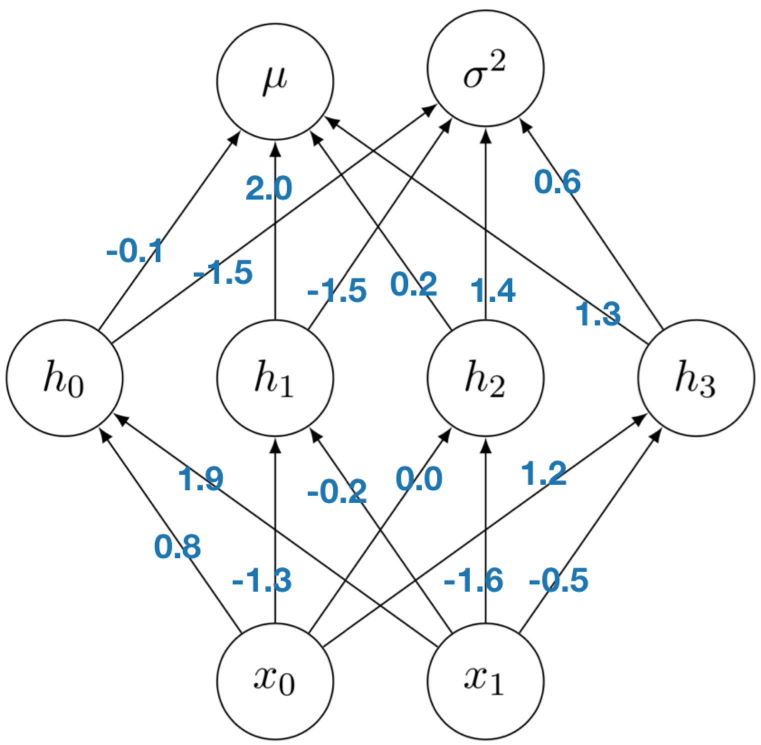
Different Weight Configurations yield Diverse Predictions:

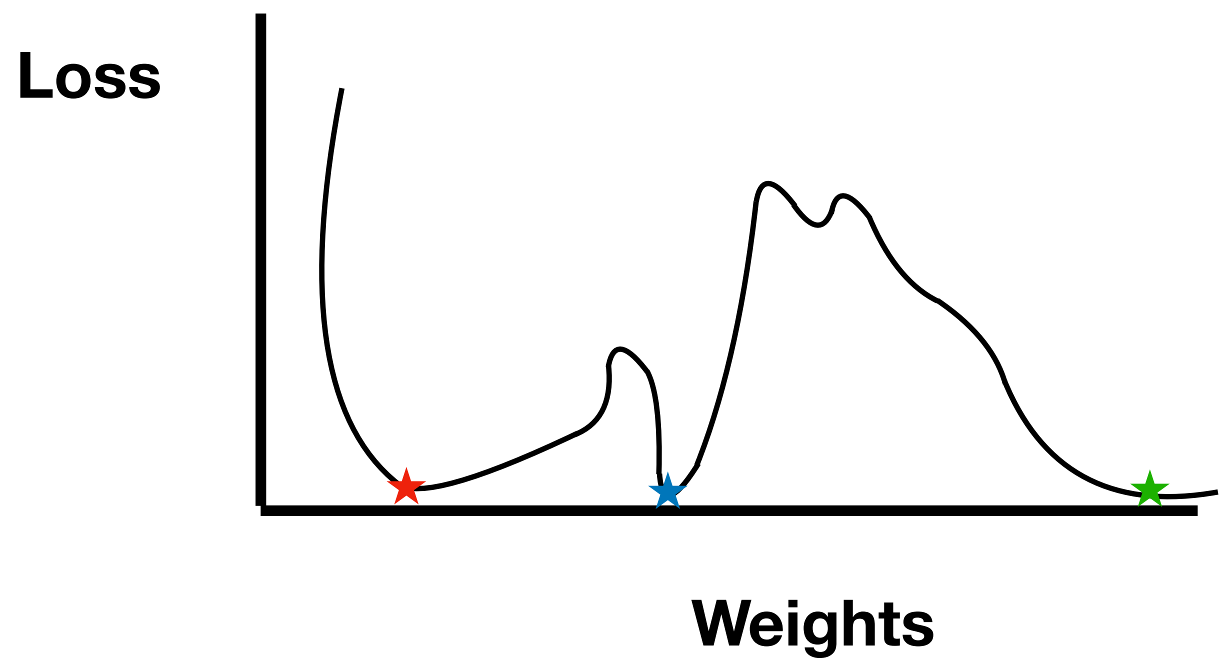


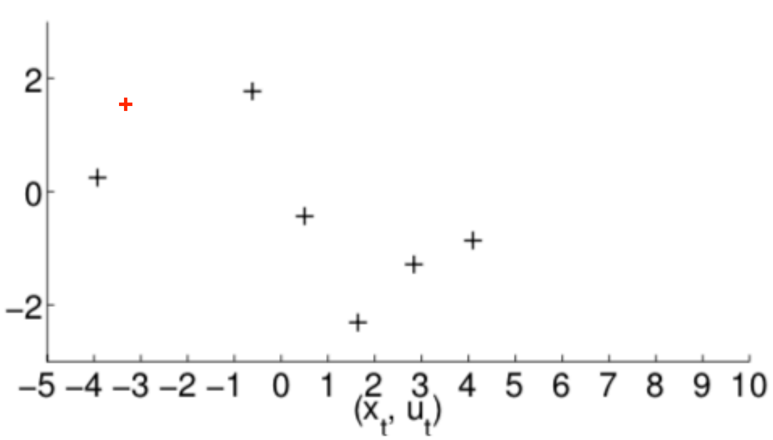


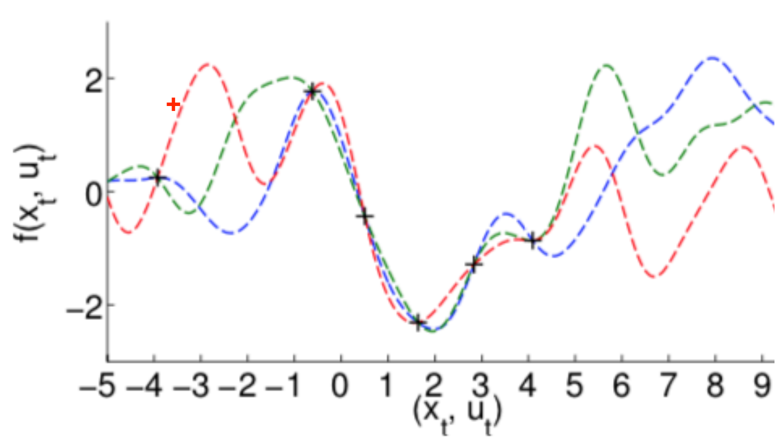
Uncertainty Estimation

Different Weight Configurations yield Diverse Predictions:



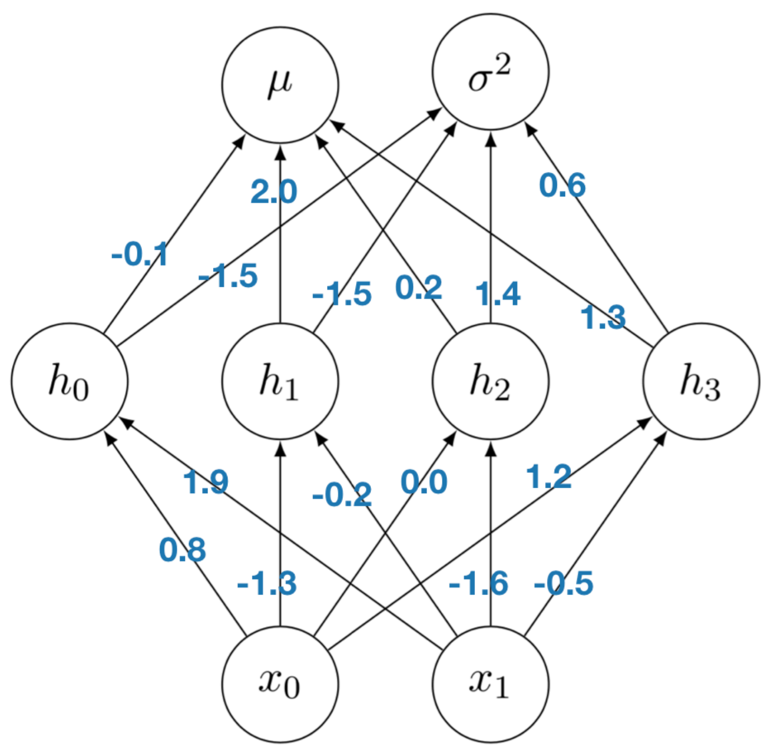


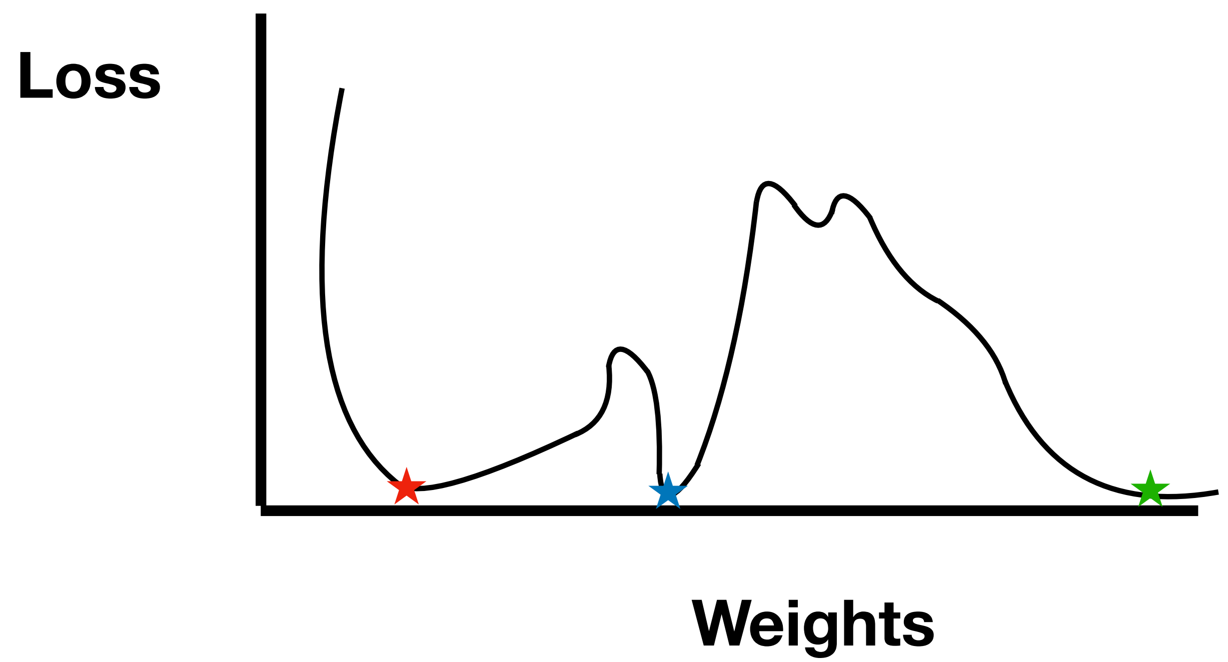


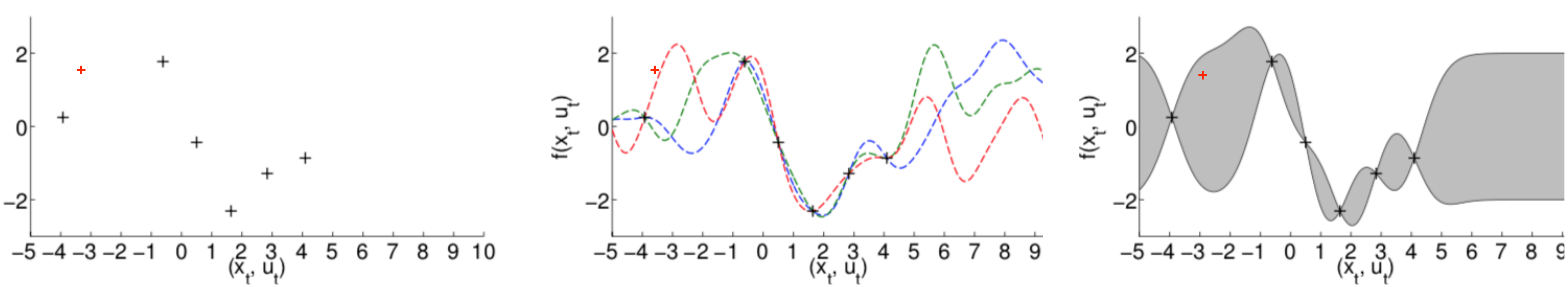


Uncertainty Estimation

Different Weight Configurations yield Diverse Predictions:

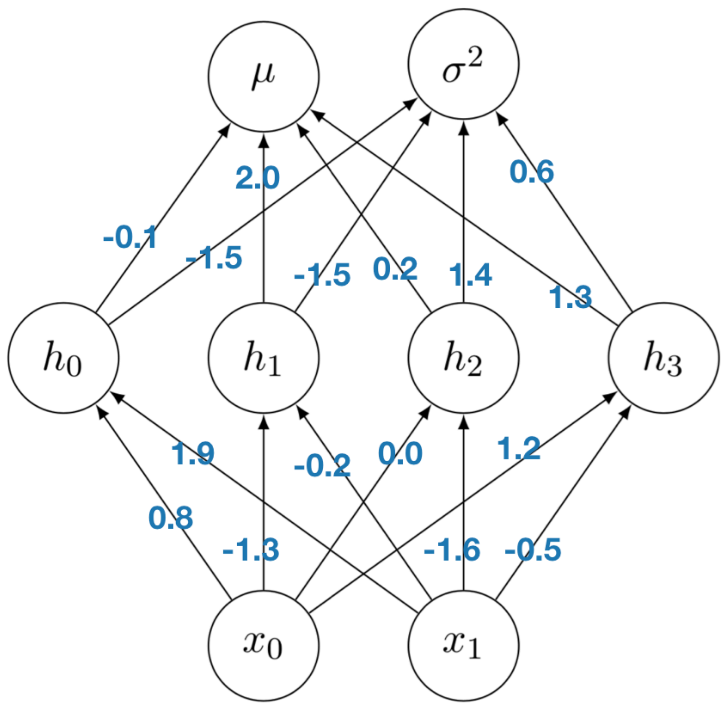




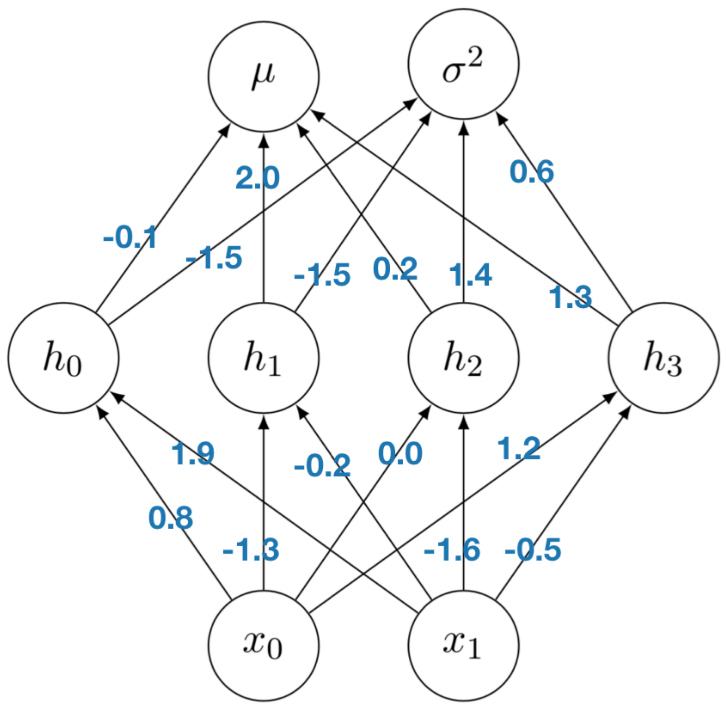


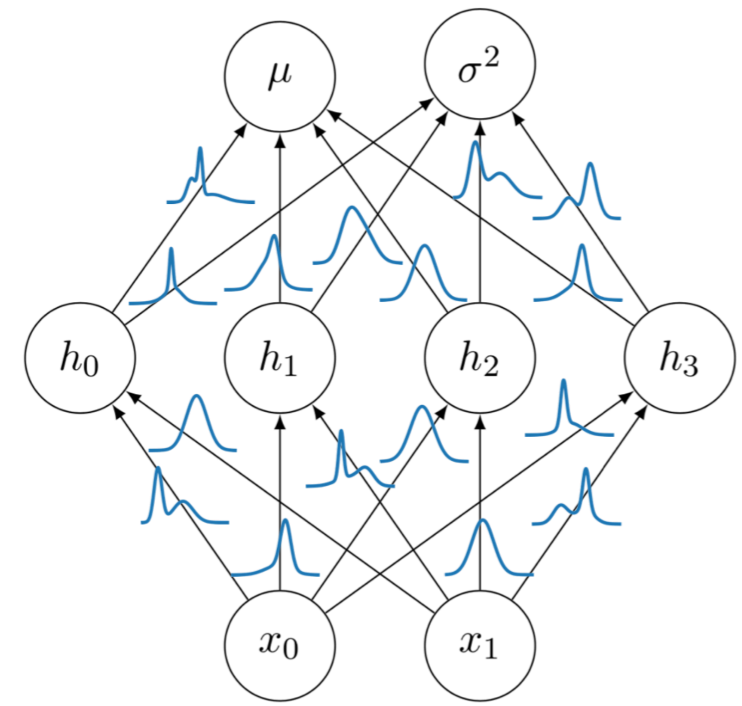
Probabilistic Inference in NNs

Probabilistic Inference in NNs

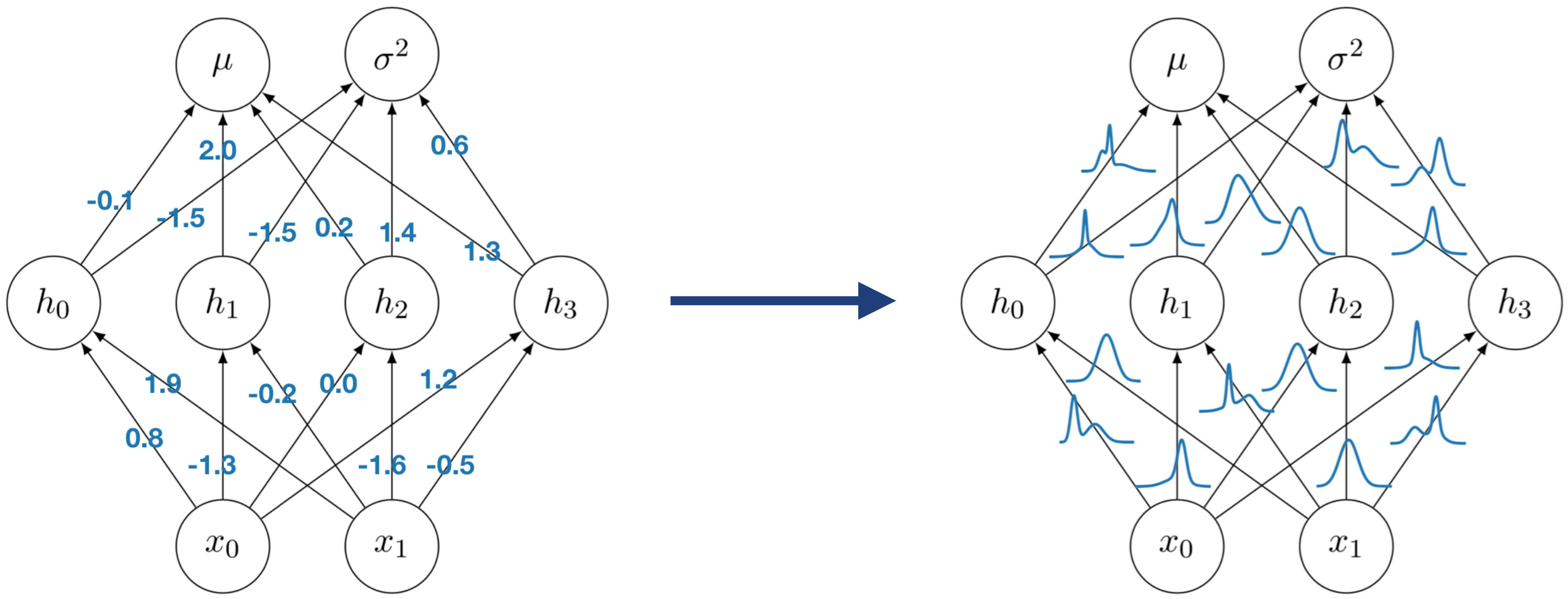


Probabilistic Inference in NNs





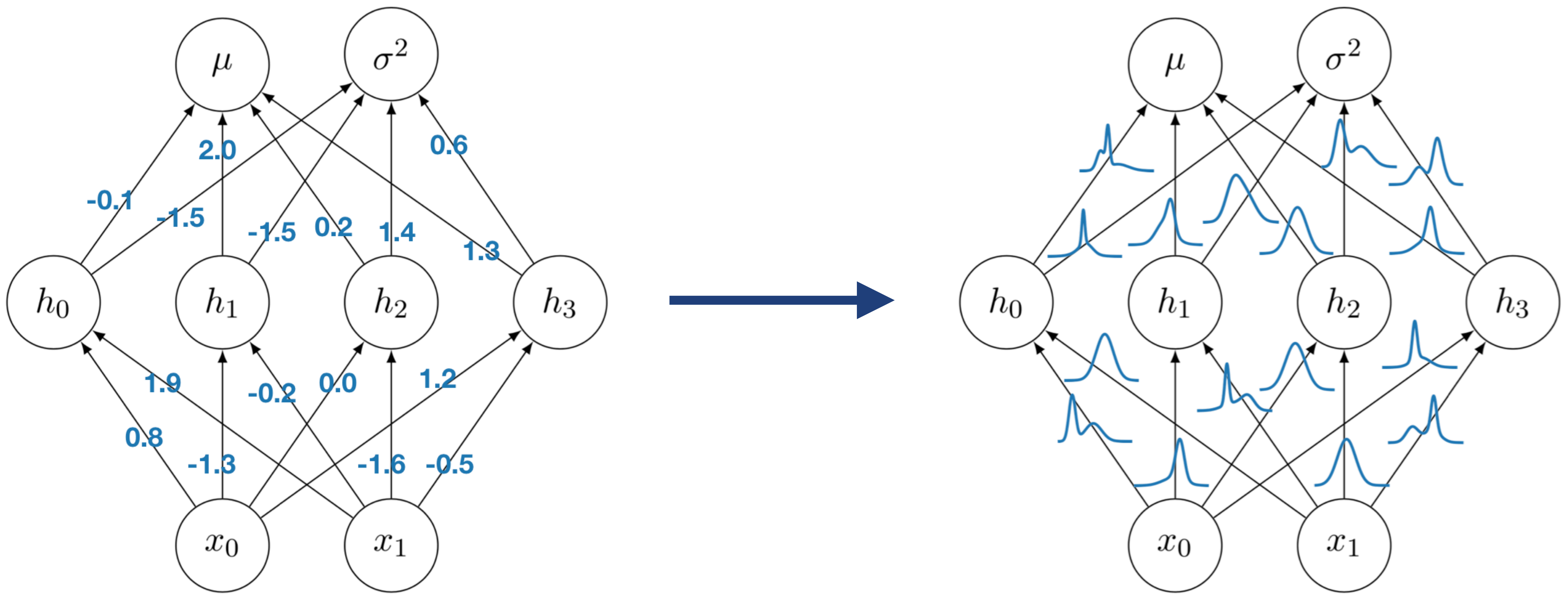
Probabilistic Inference in NNs



1. Obtain posterior distribution over weights

$$p(\mathbf{W} \mid \mathscr{D}) = \frac{p(\mathscr{D} \mid W) p(W)}{p(\mathscr{D})}$$

Probabilistic Inference in NNs



1. Obtain posterior distribution over weights

$$p(\mathbf{W} \mid \mathscr{D}) = \frac{p(\mathscr{D} \mid W)p(W)}{p(\mathscr{D})}$$

2. Marginalise weights to obtain model uncertainty

$$p(\mathbf{Y}^* \mid \mathbf{X}^*, \mathscr{D}) = \int p(\mathbf{Y}^* \mid \mathbf{X}^*, W)p(W \mid \mathscr{D})dW$$

Motivation: Why *Probabilistic* Deep Learning?

Overconfidence

Catastrophic Forgetting

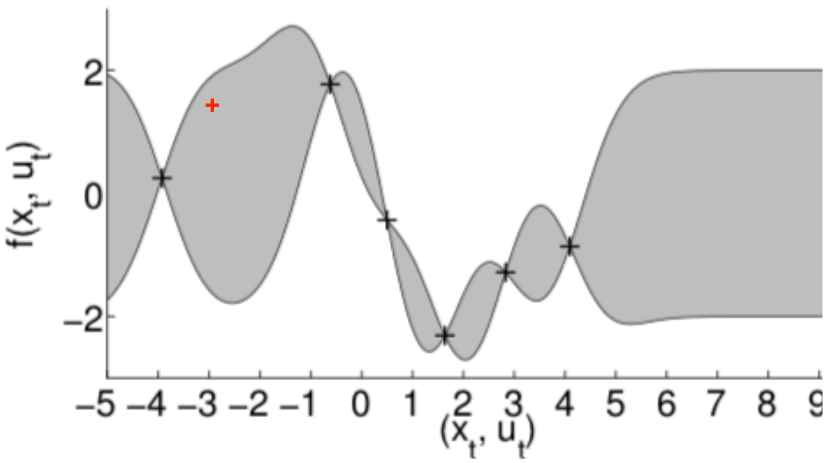
Data Inefficiency

Model Selection

Motivation: Why *Probabilistic* Deep Learning?

~~Overconfidence~~

➡ **Uncertainty Estimation**



Catastrophic Forgetting

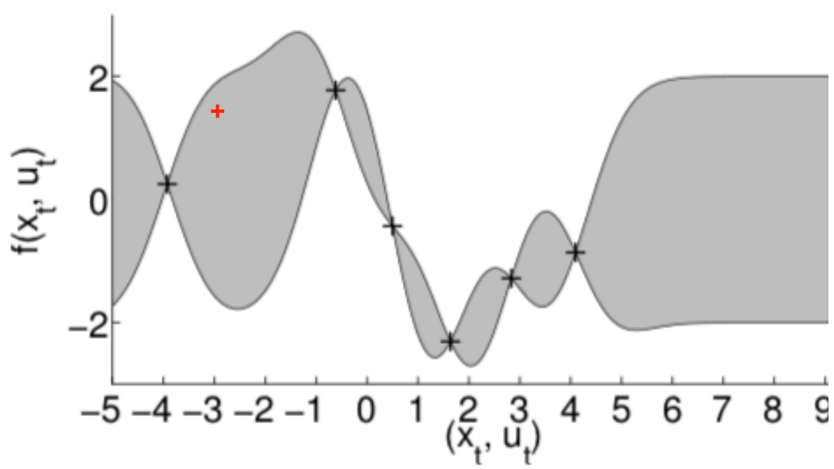
Data Inefficiency

Model Selection

Motivation: Why *Probabilistic* Deep Learning?

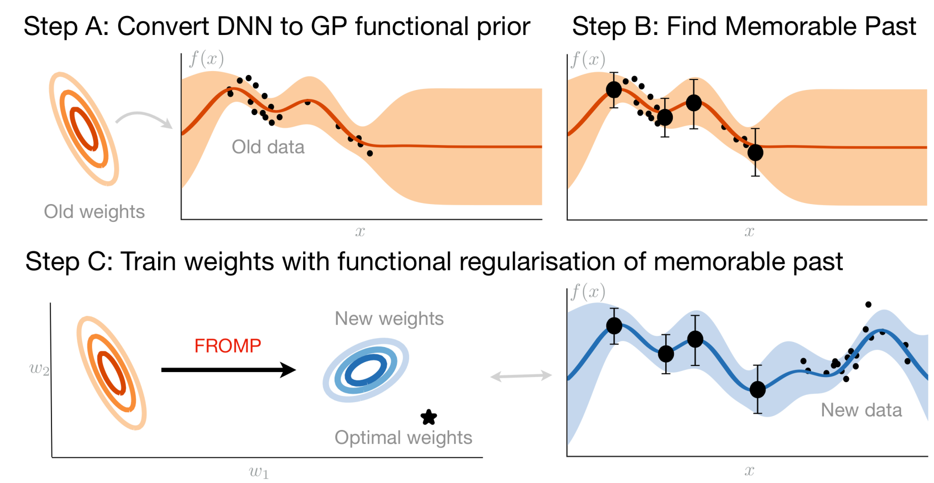
~~Overconfidence~~

Uncertainty Estimation



~~Catastrophic Forgetting~~

Continual Learning



Pan et al. 2020, "Continual Deep Learning by Functional Regularisation of Memorable Past"

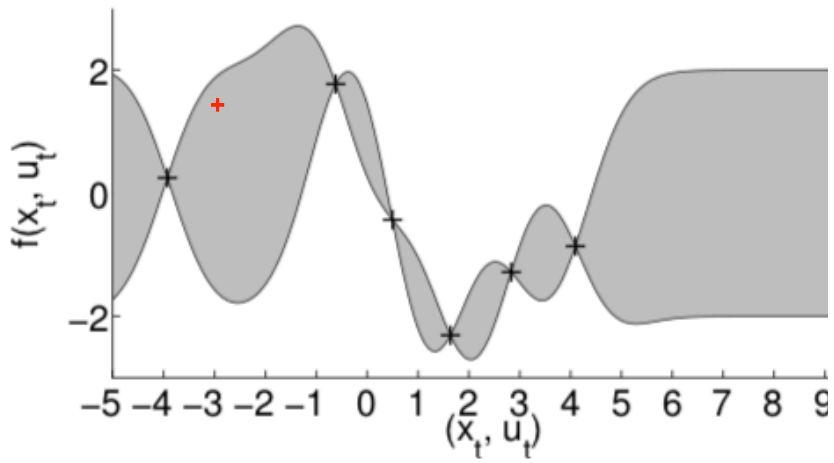
Data Inefficiency

Model Selection

Motivation: Why *Probabilistic* Deep Learning?

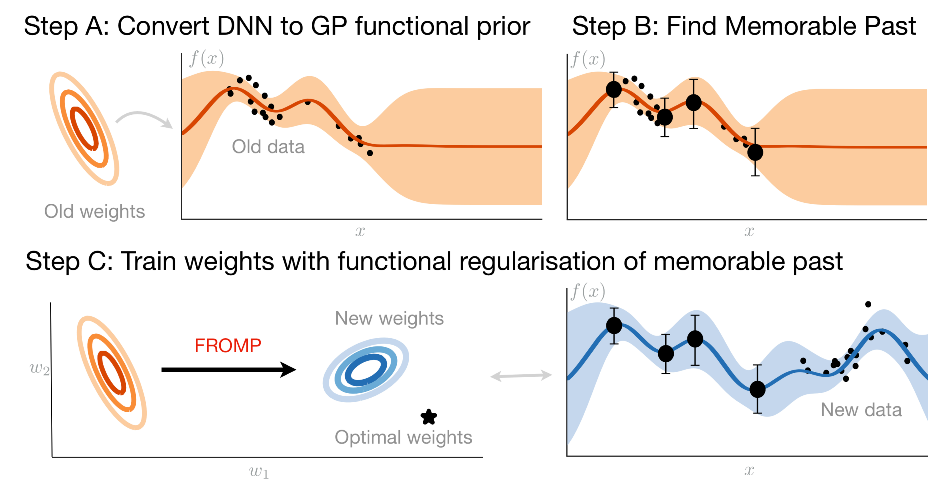
~~Overconfidence~~

→ Uncertainty Estimation



~~Catastrophic Forgetting~~

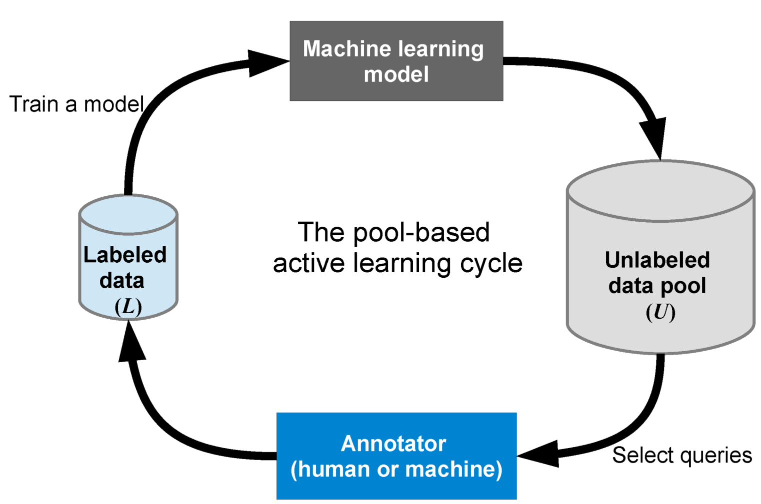
→ Continual Learning



Pan et al. 2020, "Continual Deep Learning by Functional Regularisation of Memorable Past"

~~Data Inefficiency~~

→ Active Learning



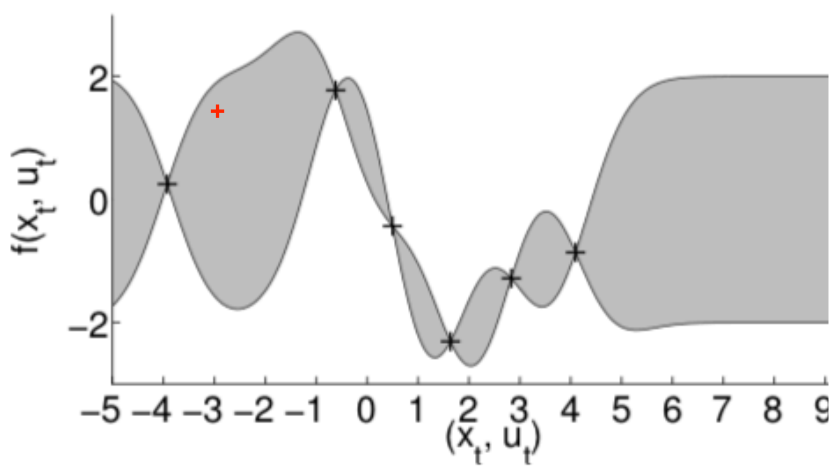
https://medium.com/@kaleajit27/apaperaday-week1-a-meta-learning-approach-to-one-step-active-learning-5ffea59099a2

Model Selection

Motivation: Why *Probabilistic* Deep Learning?

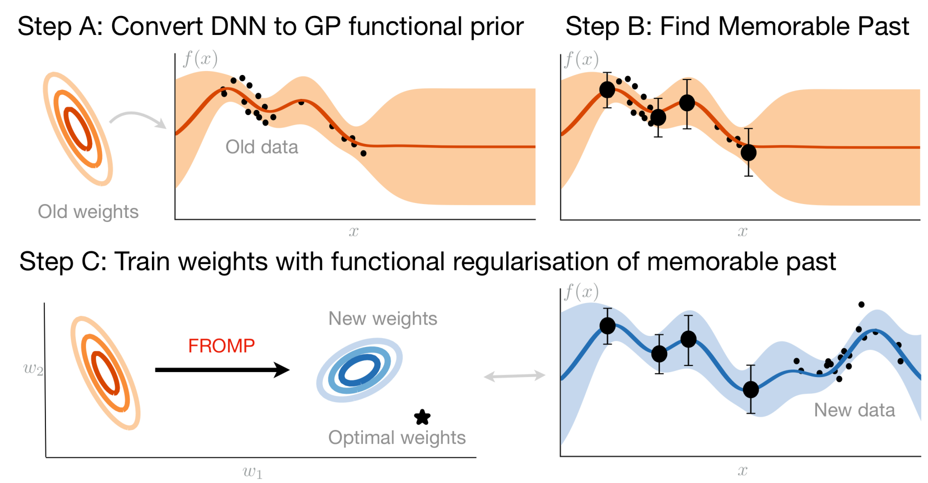
~~Overconfidence~~

➡ Uncertainty Estimation



~~Catastrophic Forgetting~~

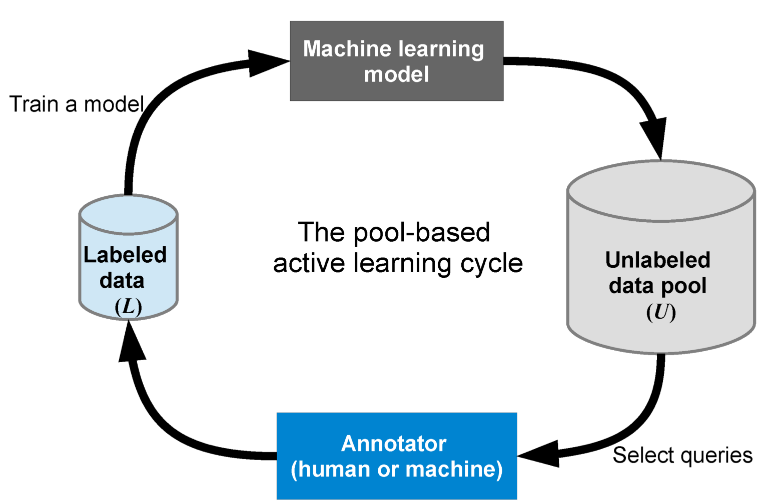
➡ Continual Learning



Pan et al. 2020, "Continual Deep Learning by Functional Regularisation of Memorable Past"

~~Data Inefficiency~~

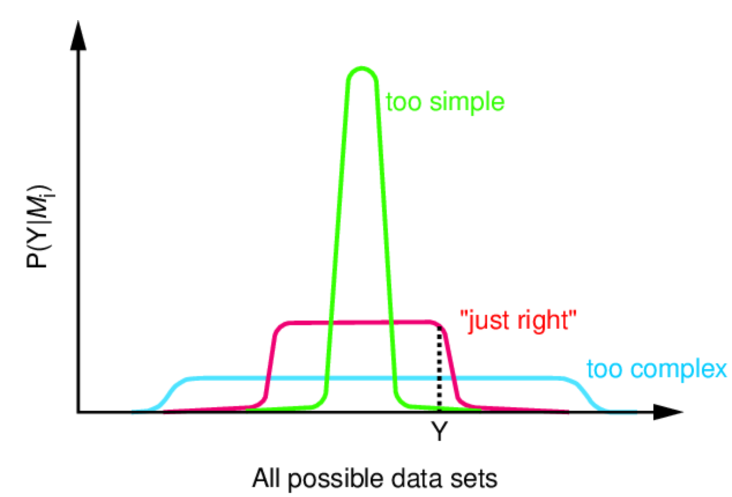
➡ Active Learning



https://medium.com/@kaleajit27/apaperaday-week1-a-meta-learning-approach-to-one-step-active-learning-5ffea59099a2

~~Model Selection~~

➡ Marginal Likelihood



Rasmussen & Ghahramani 2000, "Occam's Razor"

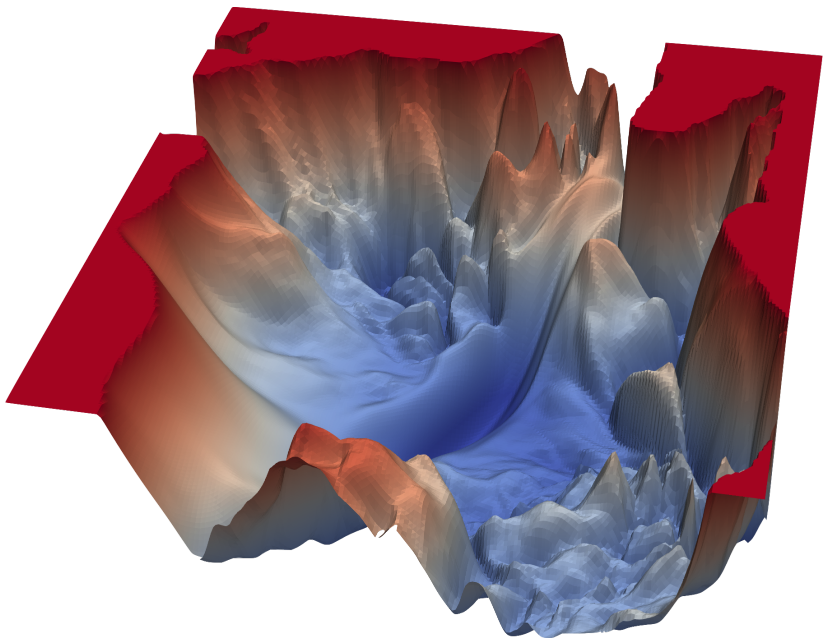
Why This Is Difficult

Why This Is Difficult

The likelihood under a BNN model is very **complex and high dimensional.**

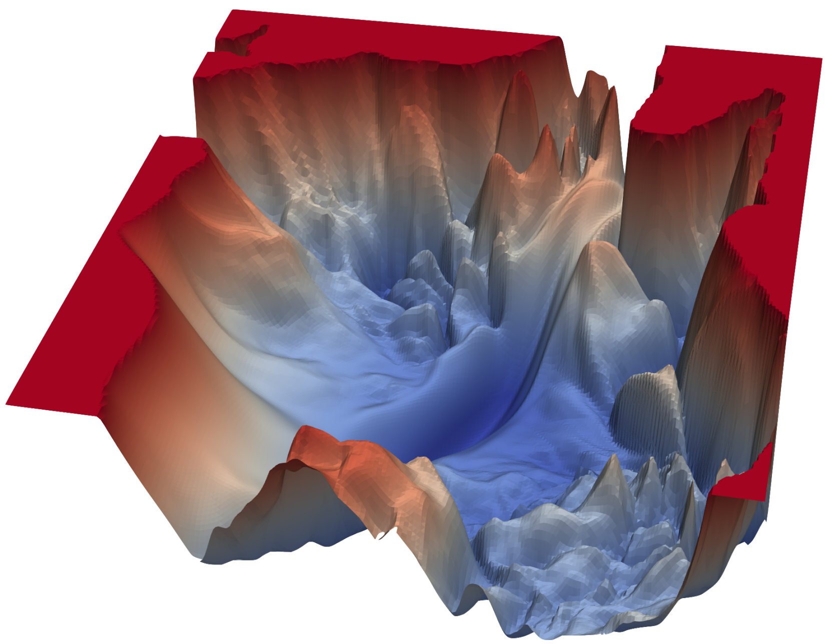
Why This Is Difficult

The likelihood under a BNN model is very **complex and high dimensional.**



Why This Is Difficult

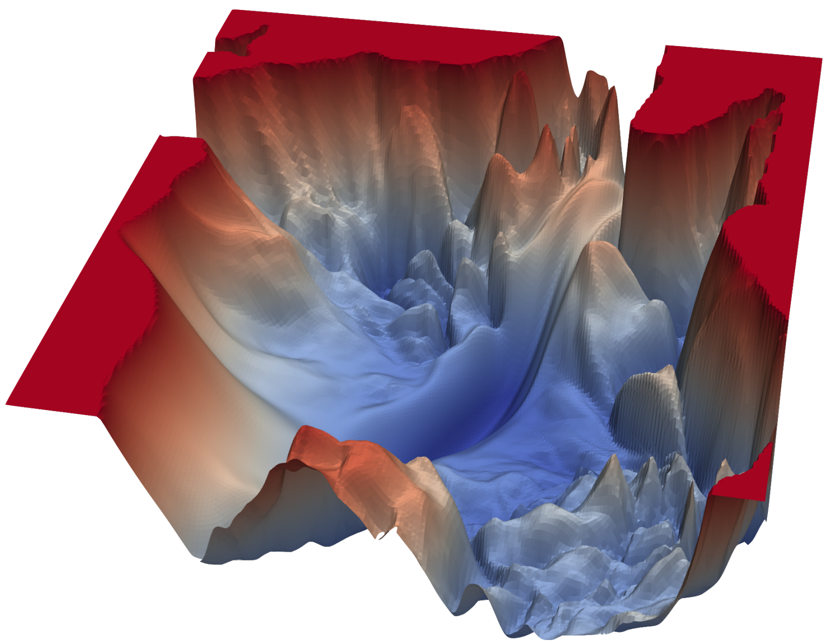
The likelihood under a BNN model is very **complex and high dimensional.**



OpenAI debuts gigantic GPT-3 language model with 175 billion parameters

Why This Is Difficult

The likelihood under a BNN model is very **complex and high dimensional.**



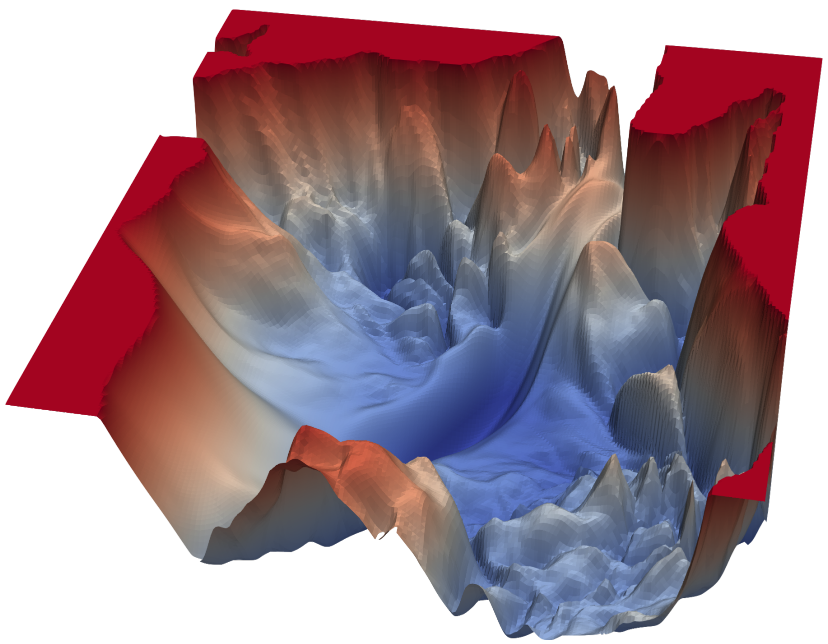
OpenAI debuts gigantic GPT-3 language model with 175 billion parameters

Problem: Modern DNNs are too big!

→ even *approximate* inference is hard!

Why This Is Difficult

The likelihood under a BNN model is very **complex and high dimensional.**



OpenAI debuts gigantic GPT-3 language model with 175 billion parameters

Problem: Modern DNNs are too big!

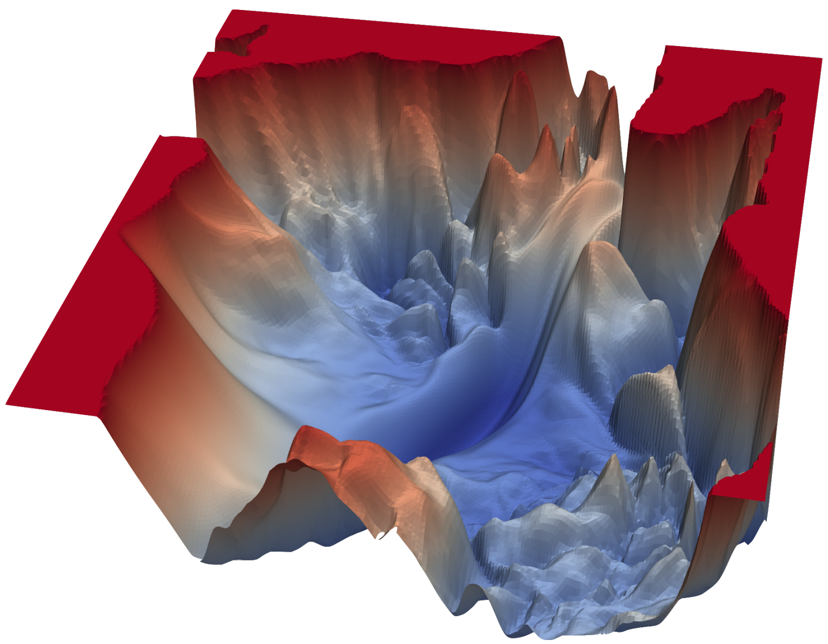
➡ even *approximate* inference is hard!

- **Intractable evidence:**

$$p(\mathscr{D}) = \int p(\mathscr{D} \mid W) p(W) dW$$

Why This Is Difficult

The likelihood under a BNN model is very **complex and high dimensional.**



OpenAI debuts gigantic GPT-3 language model with 175 billion parameters

Problem: Modern DNNs are too big!

→ even *approximate* inference is hard!

- **Intractable evidence:**

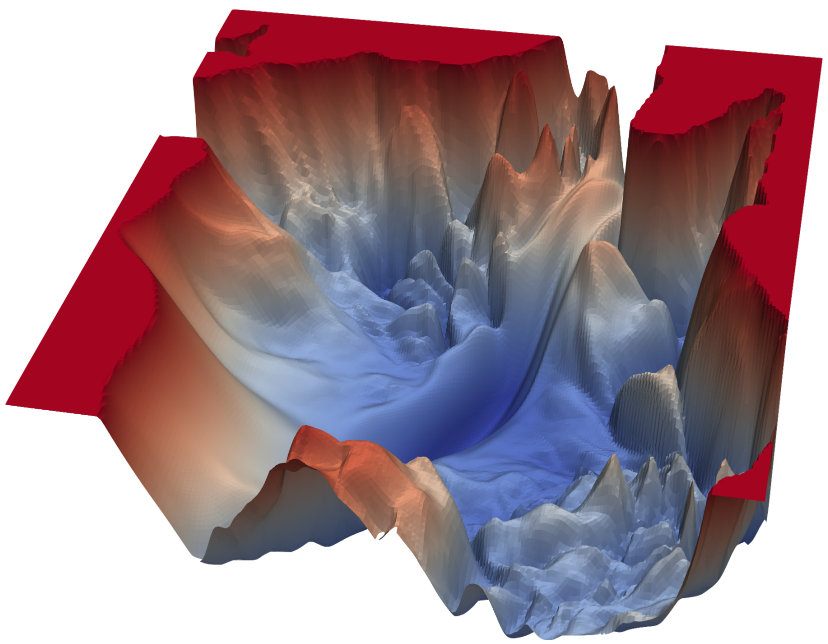
$$p(\mathscr{D}) = \int p(\mathscr{D} \mid W) p(W) dW$$

- **Intractable predictive:**

$$p(\mathbf{Y}^* \mid \mathbf{X}^*, \mathscr{D}) = \int p(\mathbf{Y}^* \mid \mathbf{X}^*, W) p(W \mid \mathscr{D}) dW$$

Why This Is Difficult

The likelihood under a BNN model is very **complex and high dimensional.**



- **Intractable evidence:**

$$p(\mathscr{D}) = \int p(\mathscr{D} \mid W) p(W) dW$$

- **Intractable predictive:**

$$p(\mathbf{Y}^* \mid \mathbf{X}^*, \mathscr{D}) = \int p(\mathbf{Y}^* \mid \mathbf{X}^*, W) p(W \mid \mathscr{D}) dW$$

OpenAI debuts gigantic GPT-3 language model with 175 billion parameters

Problem: Modern DNNs are too big!

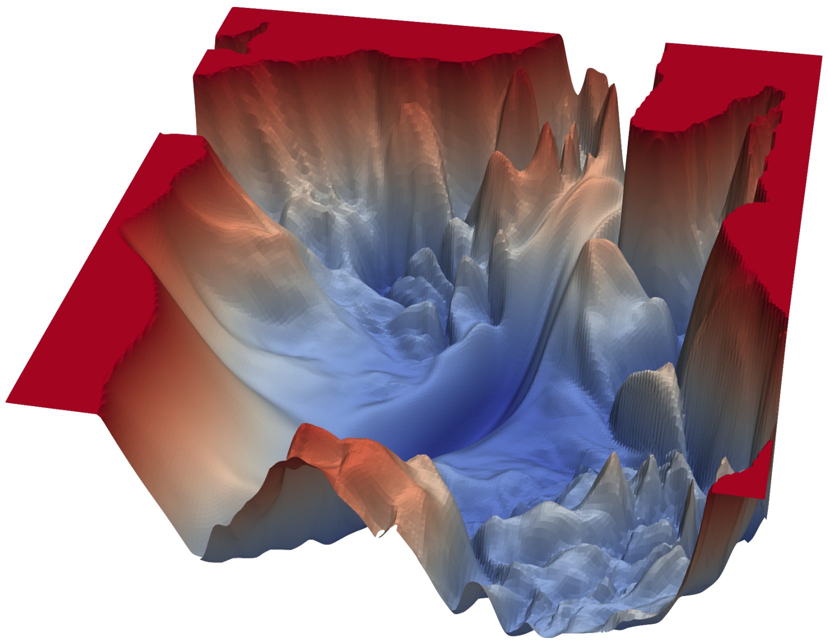
→ even *approximate* inference is hard!

Solution(?): Make strong/unrealistic assumptions on posterior, e.g. full factorisation

$$p(\mathbf{W} | \boldsymbol{y}, \boldsymbol{X}) \approx \prod_{d=1}^{D} q(\mathrm{w}_d)$$

Why This Is Difficult

The likelihood under a BNN model is very **complex and high dimensional.**



OpenAI debuts gigantic GPT-3 language model with 175 billion parameters

Problem: Modern DNNs are too big!

➡ even *approximate* inference is hard!

- **Intractable evidence:**

$$p(\mathscr{D}) = \int p(\mathscr{D} \mid W) p(W) dW$$

Solution(?): Make strong/unrealistic assumptions on posterior, e.g. full factorisation

$$p(\mathbf{W} | \boldsymbol{y}, \boldsymbol{X}) \approx \prod_{d=1}^{D} q(\mathrm{w}_d)$$

- **Intractable predictive:**

$$p(\mathbf{Y}^* \mid \mathbf{X}^*, \mathscr{D}) = \int p(\mathbf{Y}^* \mid \mathbf{X}^*, W) p(W \mid \mathscr{D}) dW$$

➡ Deteriorates quality of induced uncertainties!
(Ovadia 2019, Fort 2019, Foong 2019, Ashukha 2020)

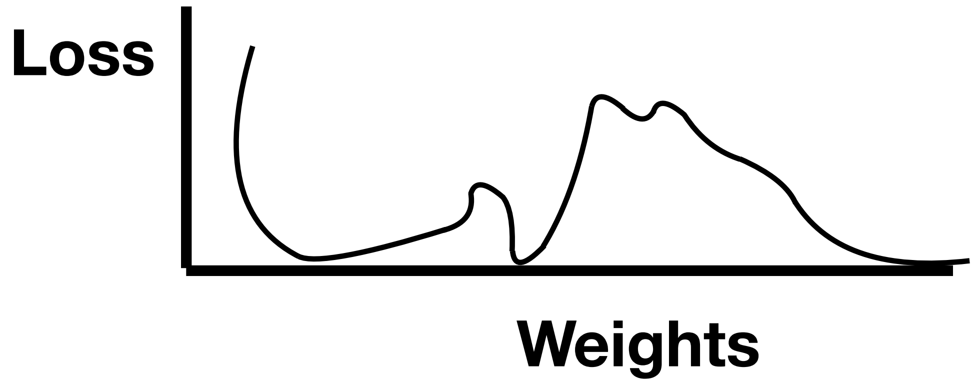
Existing Approaches

Existing Approaches

Variational Inference

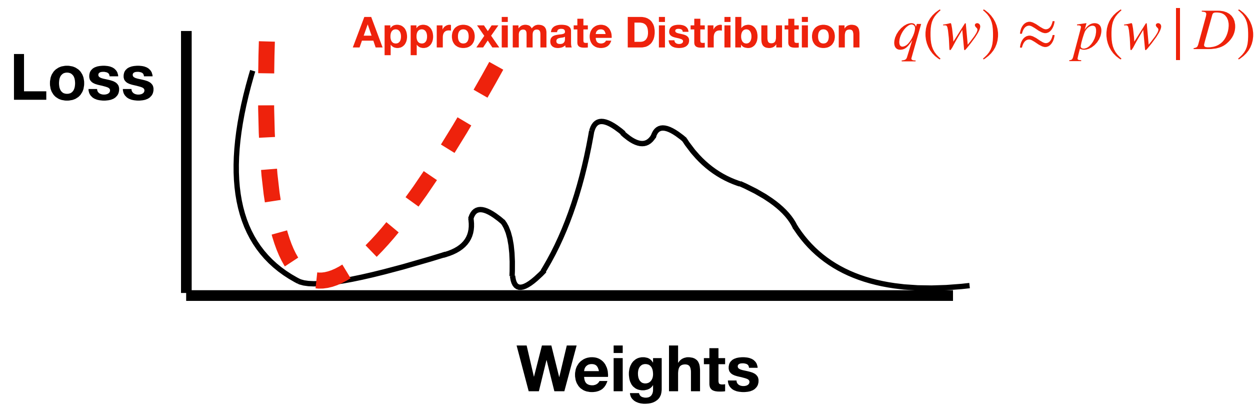
Existing Approaches

Variational Inference



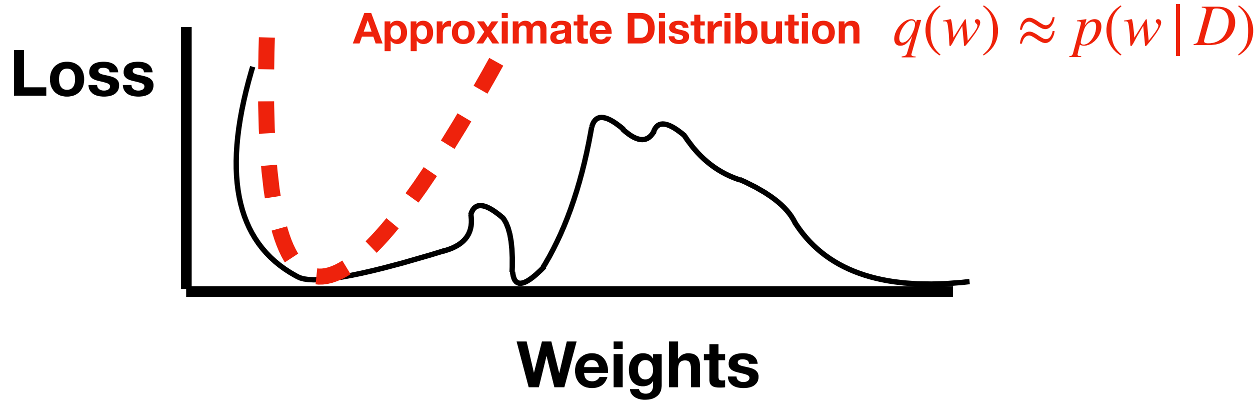
Existing Approaches

Variational Inference

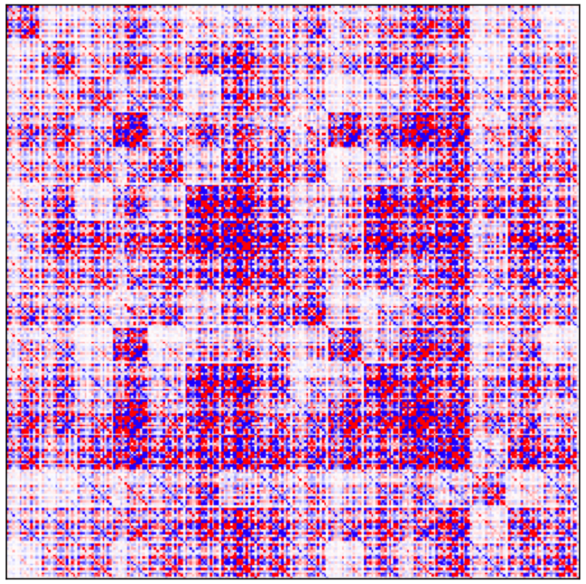


Existing Approaches

Variational Inference



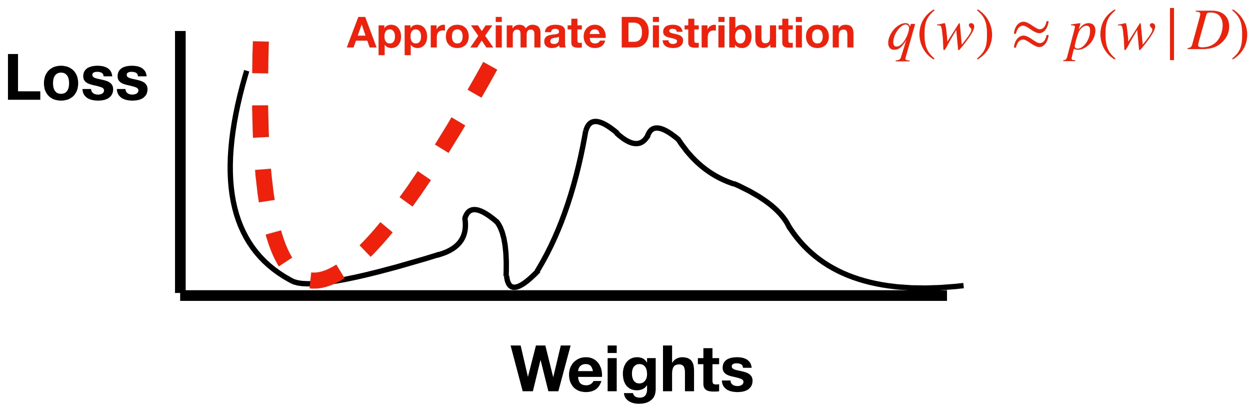
Full Covariance



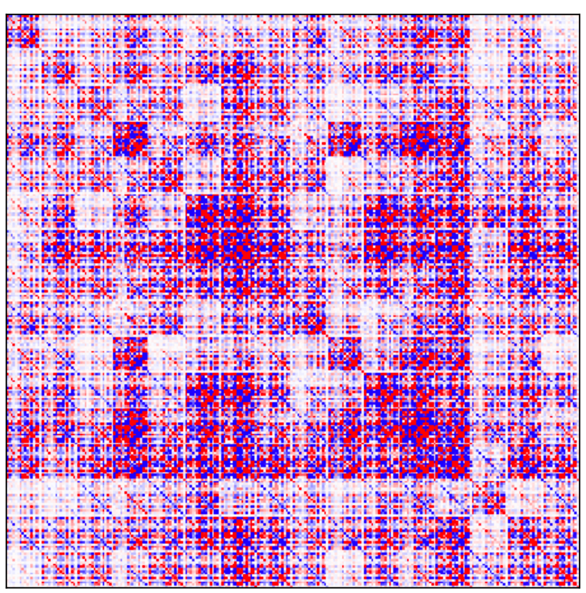
$|W|^2$ **Elements**

Existing Approaches

Variational Inference

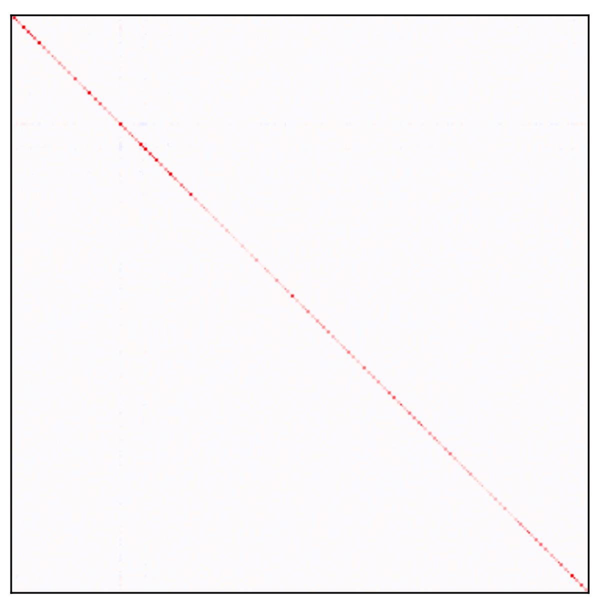


Full Covariance



$|W|^2$ **Elements**

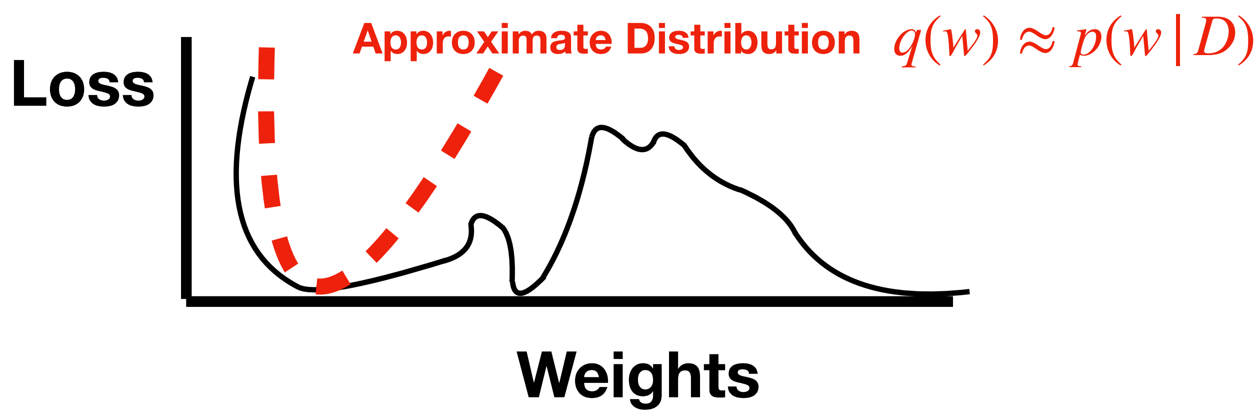
Mean Field



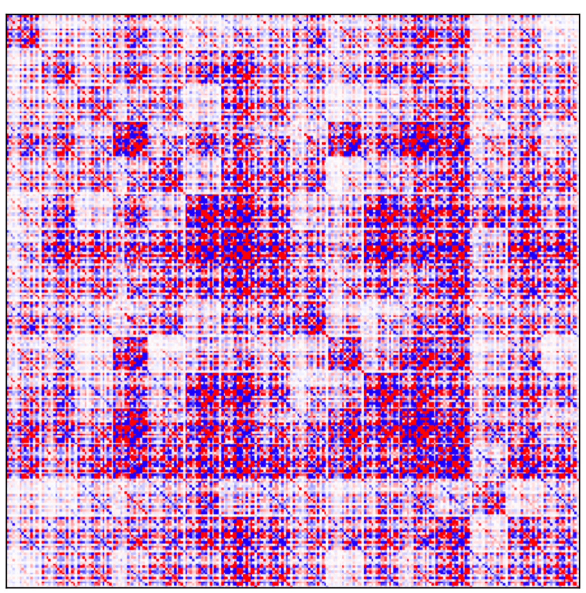
$|W|$ **Elements**

Existing Approaches

Variational Inference

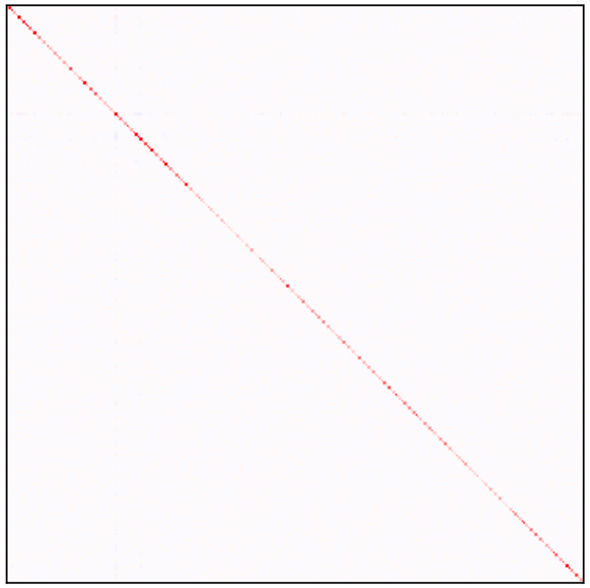


Full Covariance



$|W|^2$ **Elements**

Mean Field

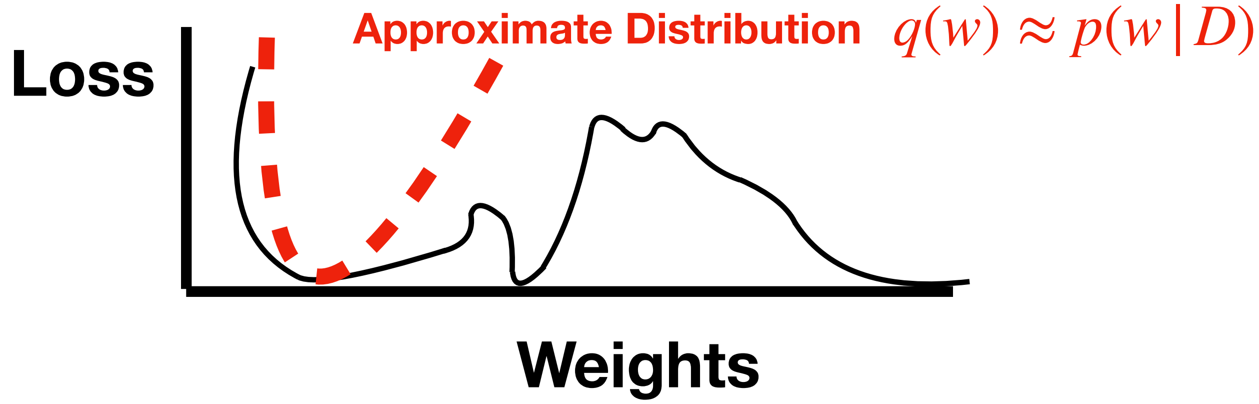


$|W|$ **Elements**

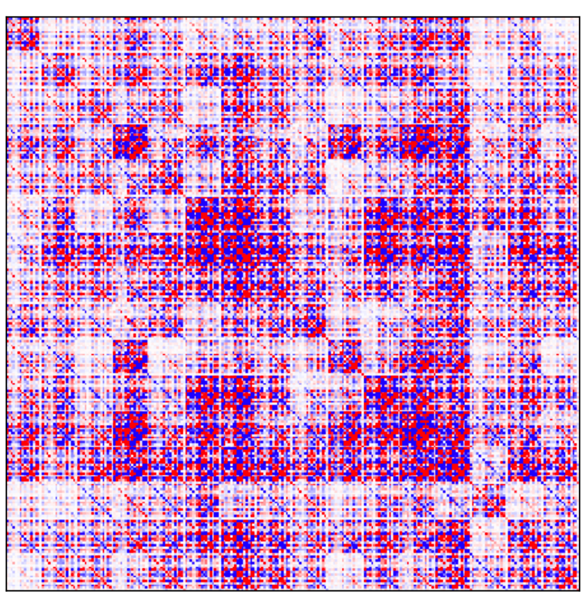
Deep Ensembles

Existing Approaches

Variational Inference

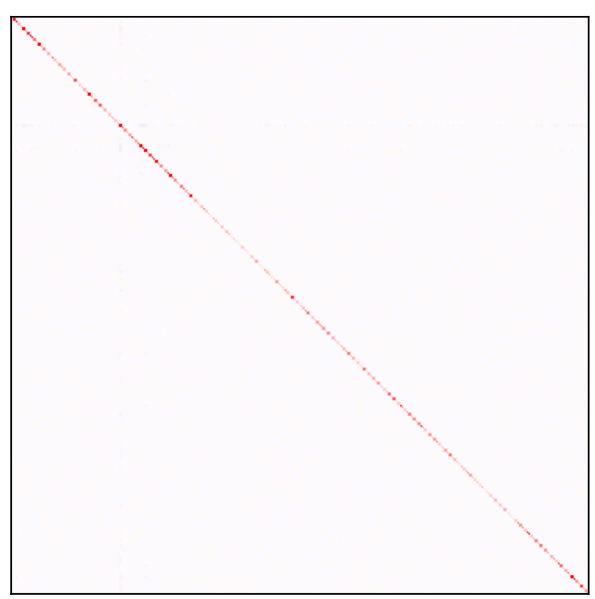


Full Covariance



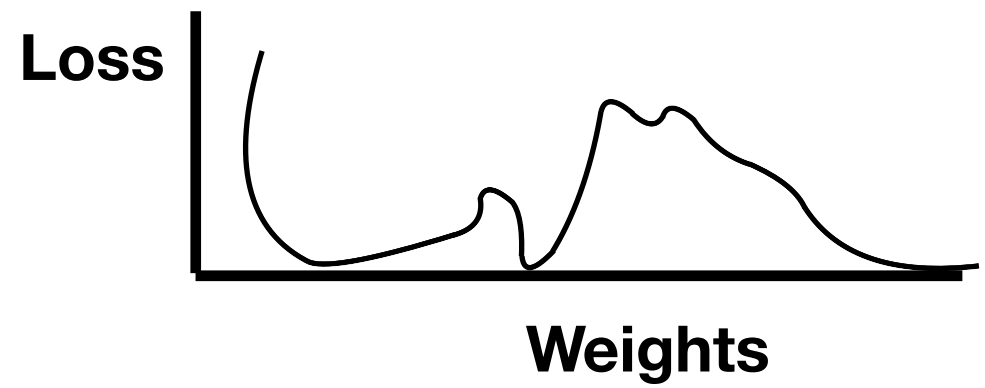
$|W|^2$ **Elements**

Mean Field



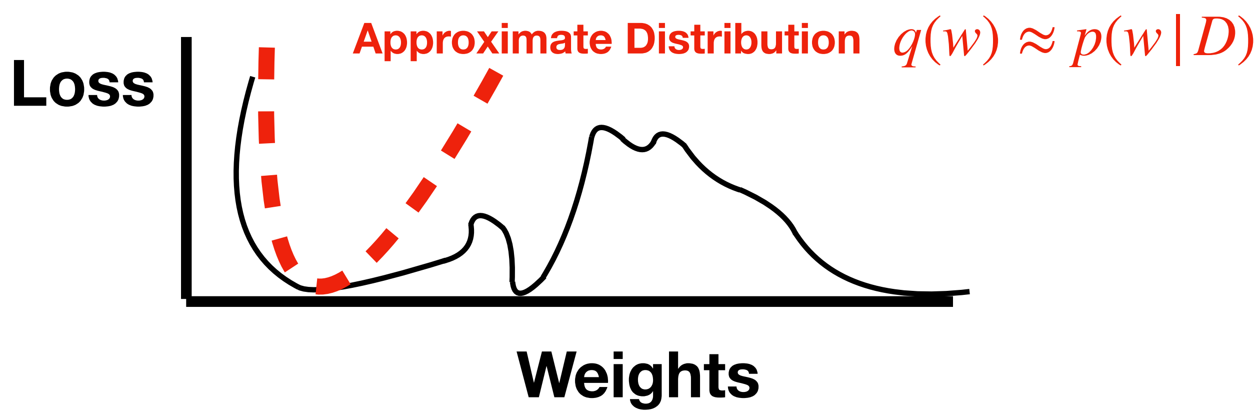
$|W|$ **Elements**

Deep Ensembles

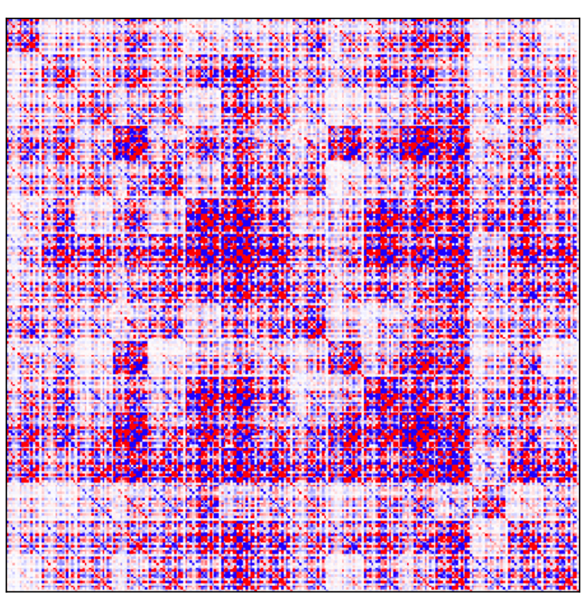


Existing Approaches

Variational Inference

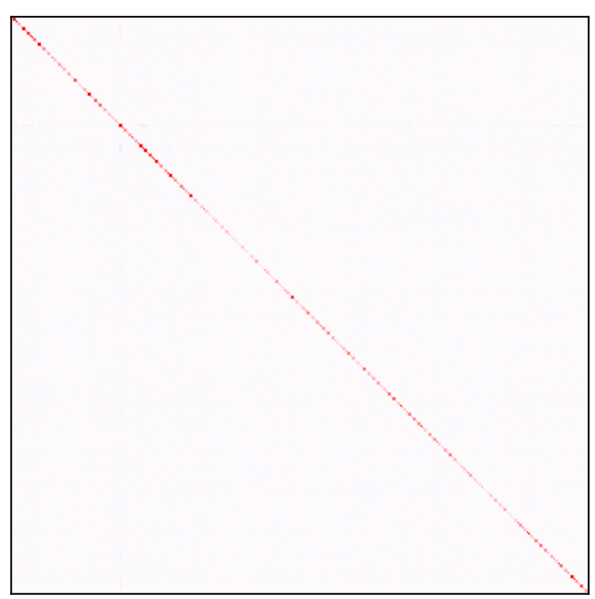


Full Covariance



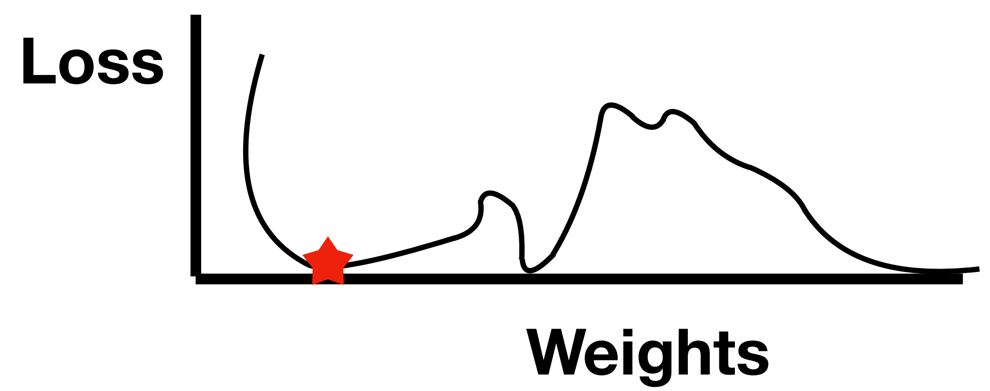
$|W|^2$ **Elements**

Mean Field



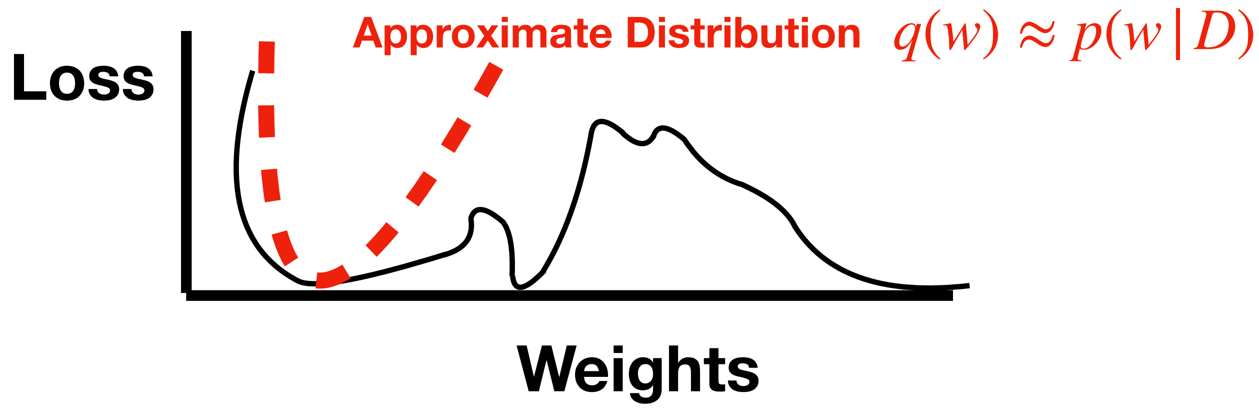
$|W|$ **Elements**

Deep Ensembles

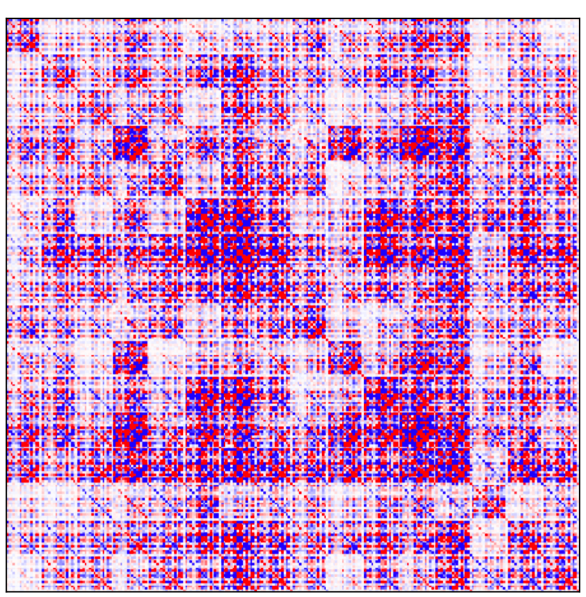


Existing Approaches

Variational Inference

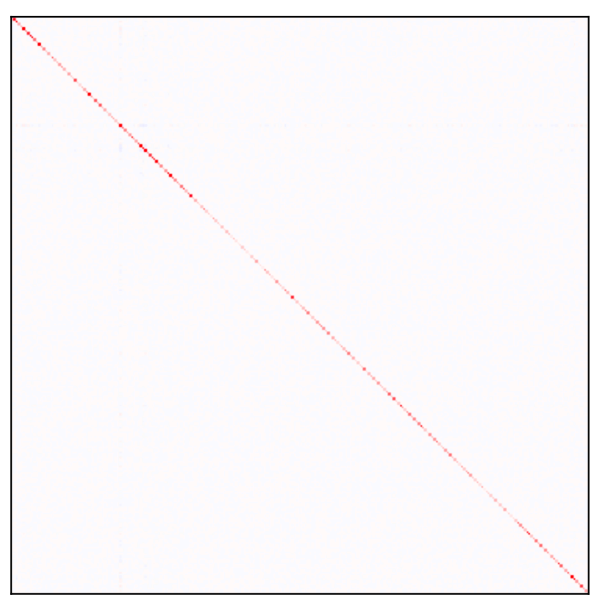


Full Covariance



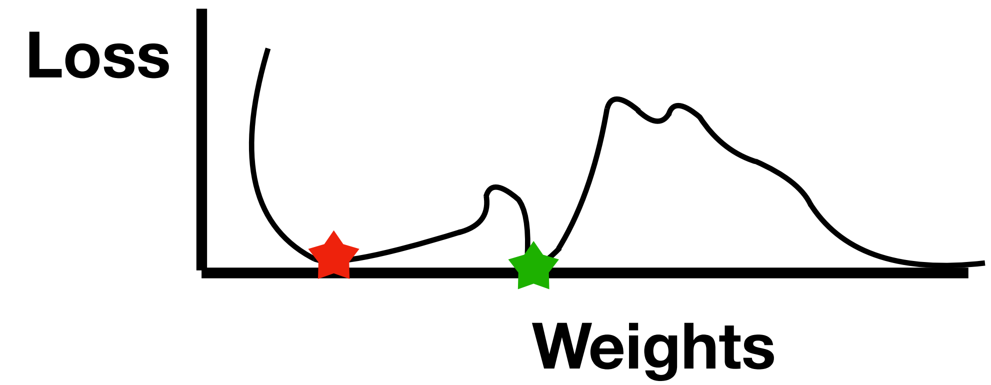
$|W|^2$ **Elements**

Mean Field



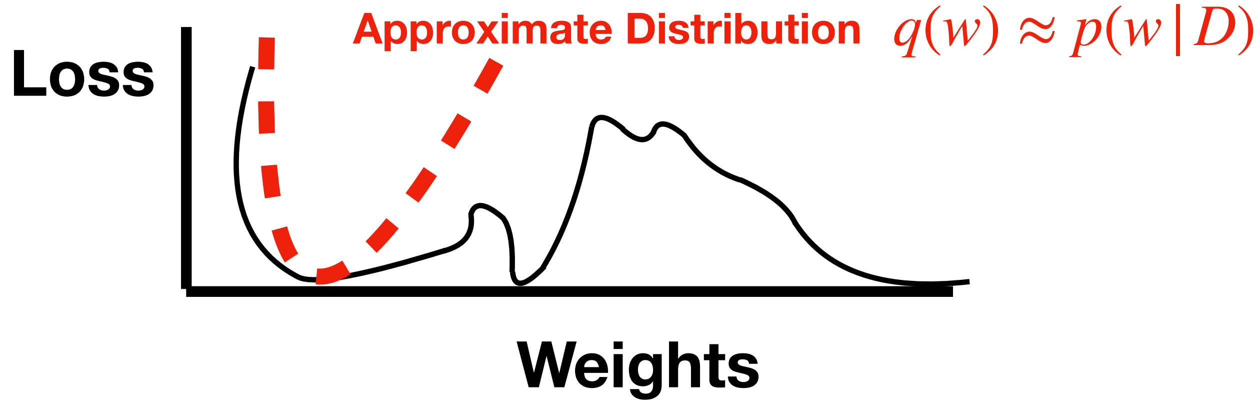
$|W|$ **Elements**

Deep Ensembles

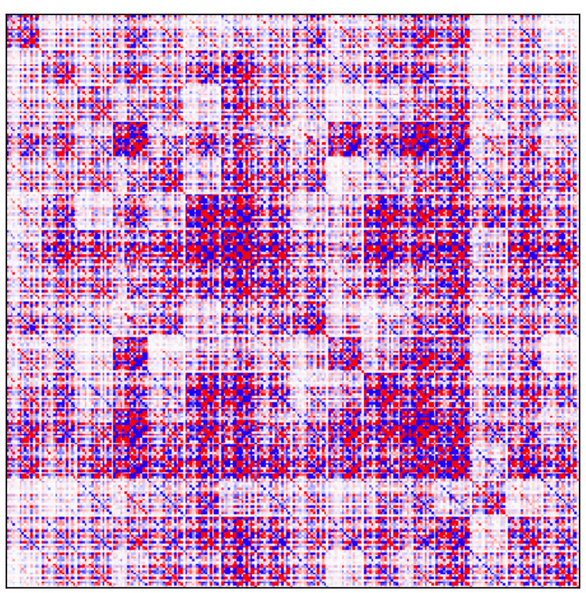


Existing Approaches

Variational Inference

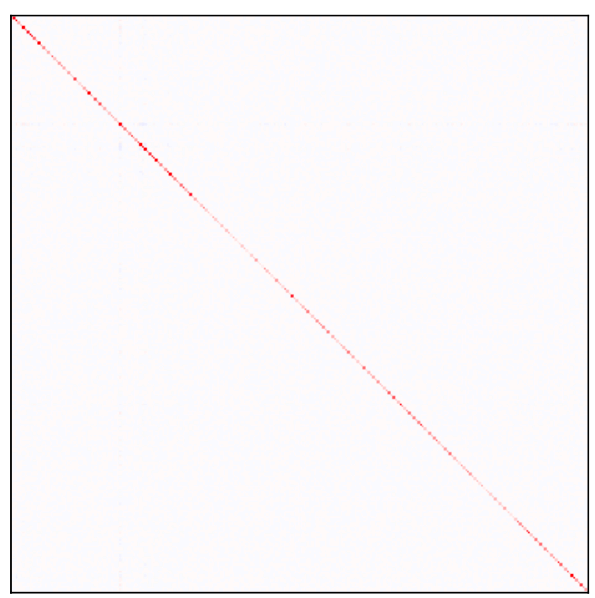


Full Covariance



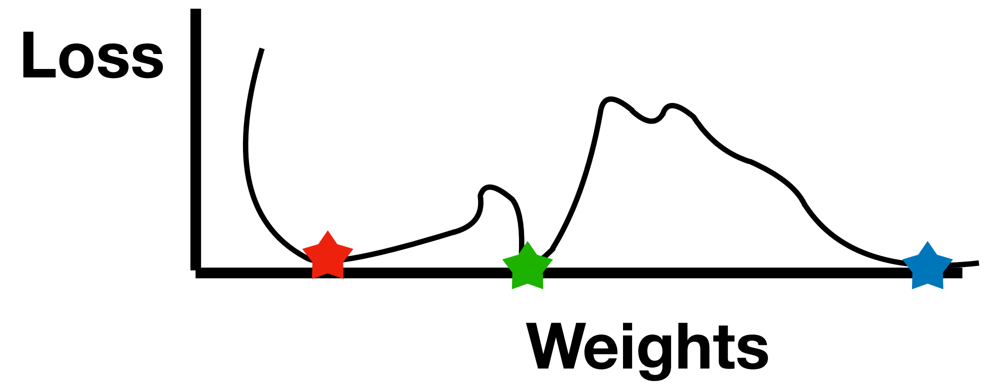
$|W|^2$ **Elements**

Mean Field



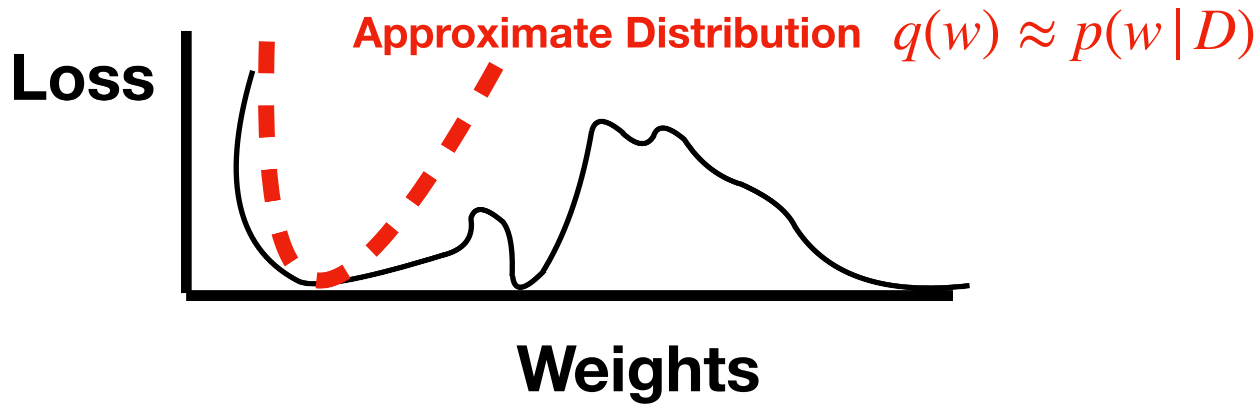
$|W|$ **Elements**

Deep Ensembles

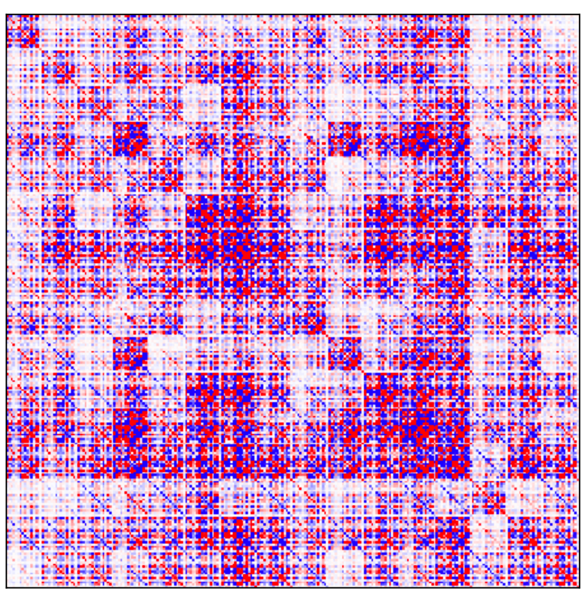


Existing Approaches

Variational Inference

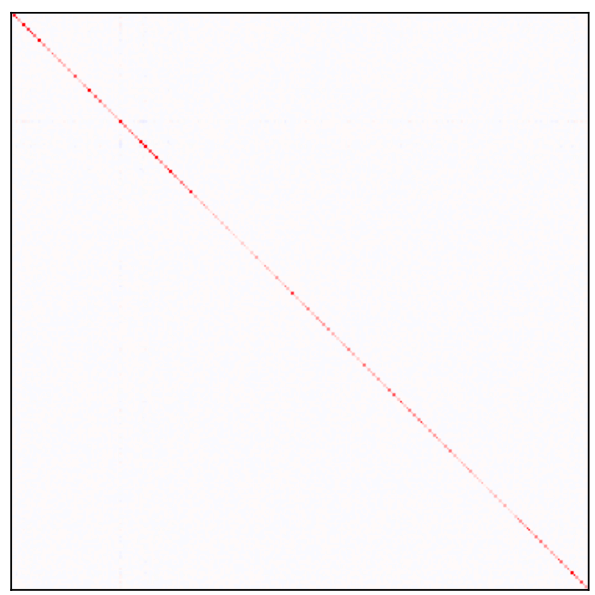


Full Covariance



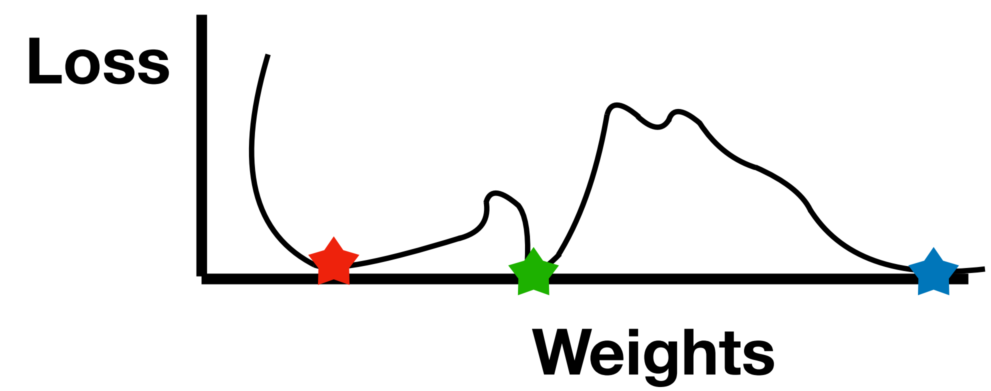
$|W|^2$ **Elements**

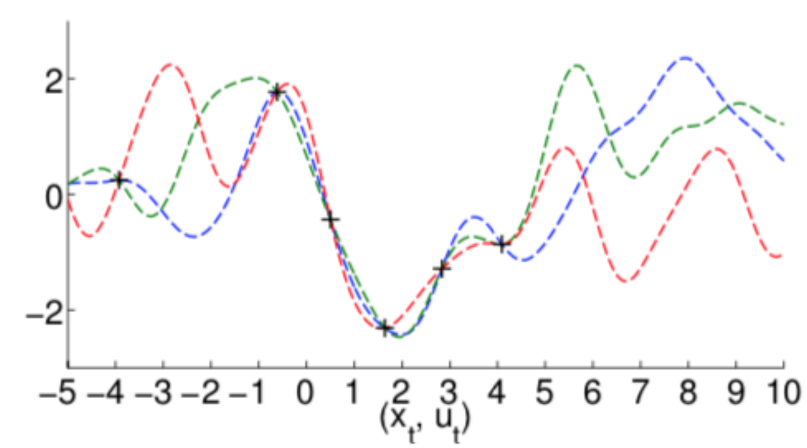
Mean Field



$|W|$ **Elements**

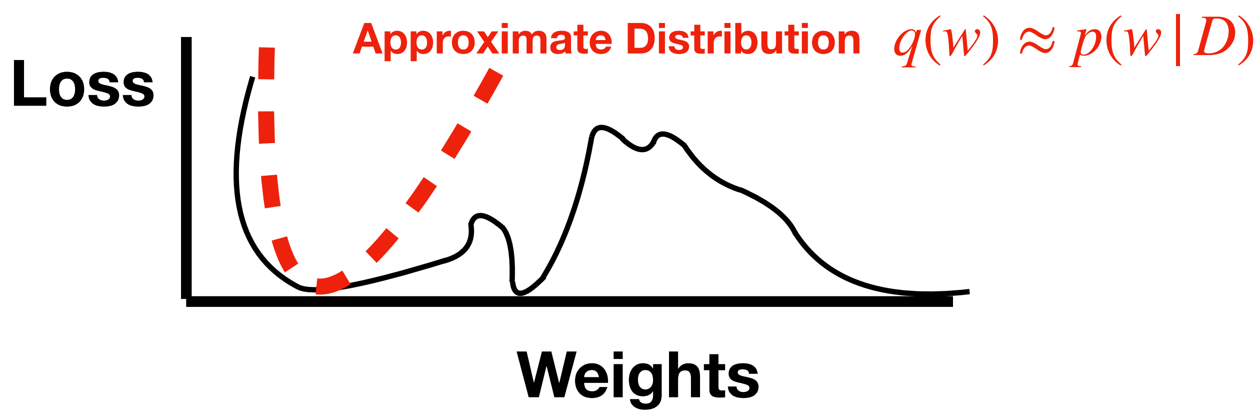
Deep Ensembles



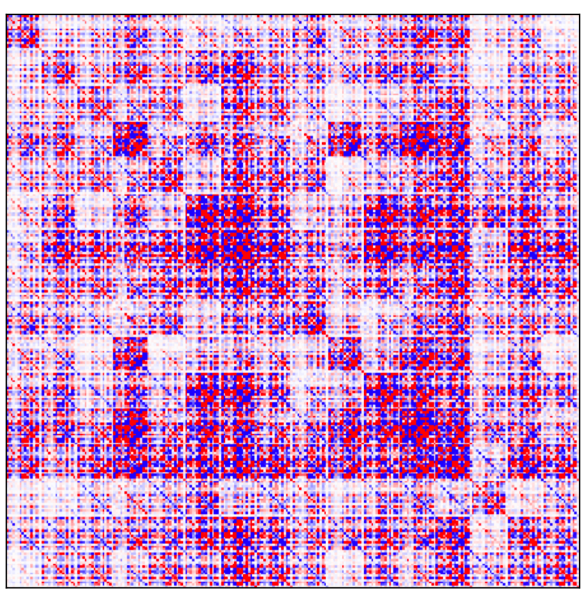


Existing Approaches

Variational Inference

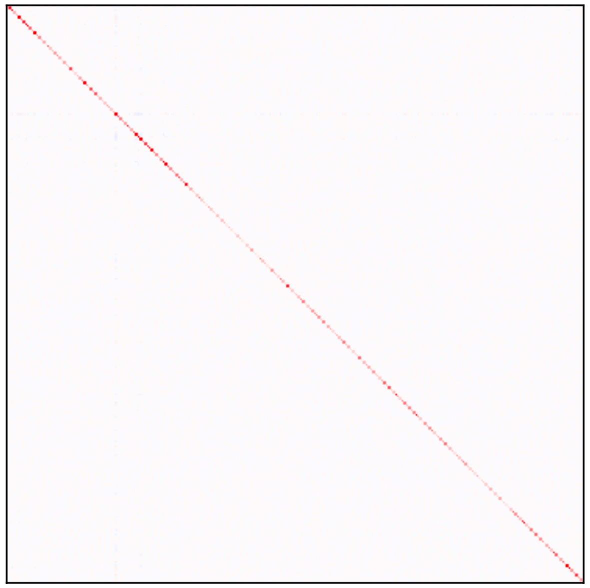


Full Covariance



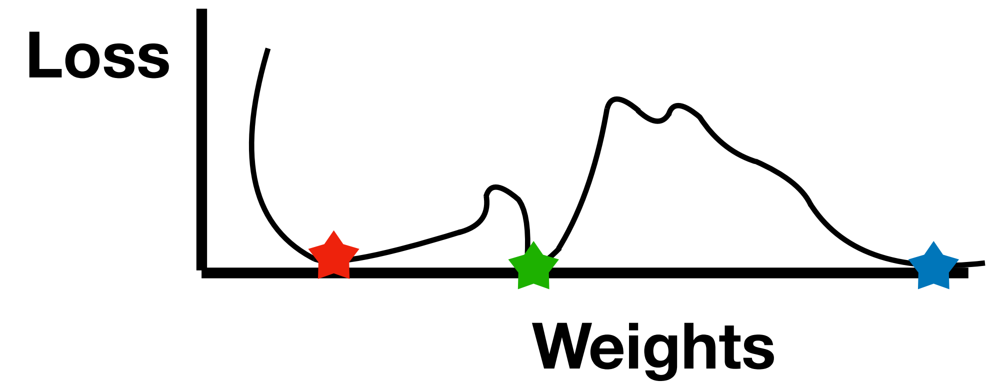
$|W|^2$ **Elements**

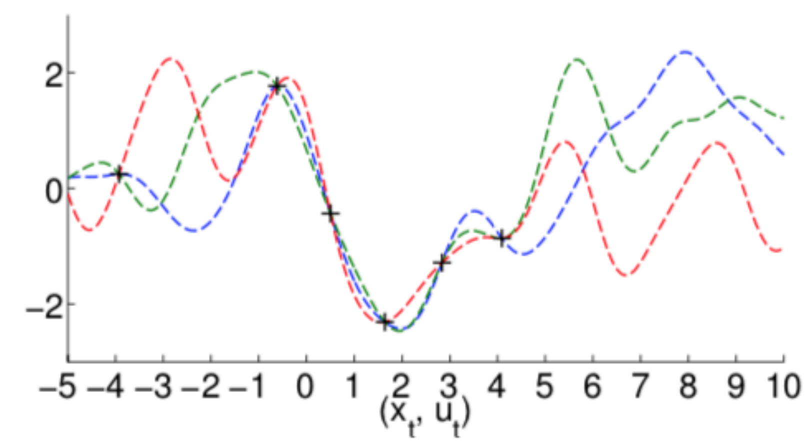
Mean Field

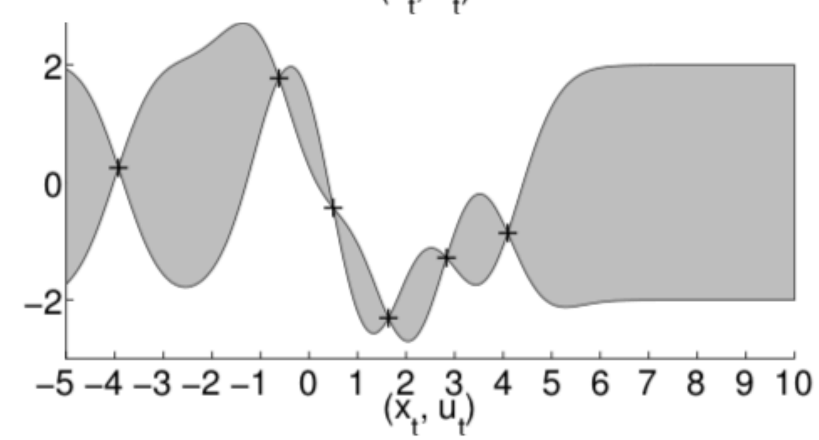


$|W|$ **Elements**

Deep Ensembles



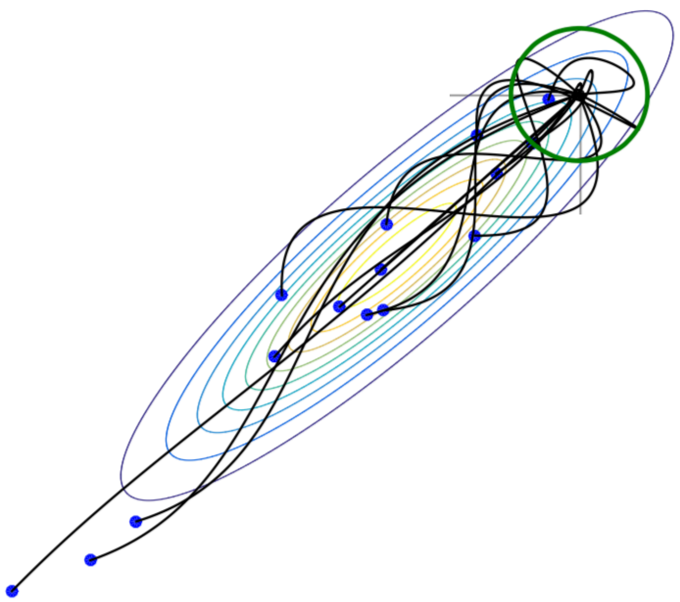




More Existing Approaches

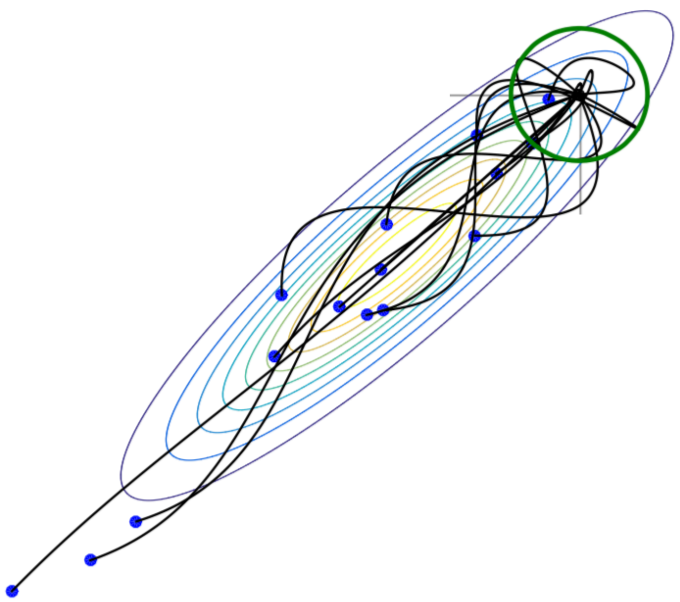
More Existing Approaches

Hamiltonian Monte Carlo

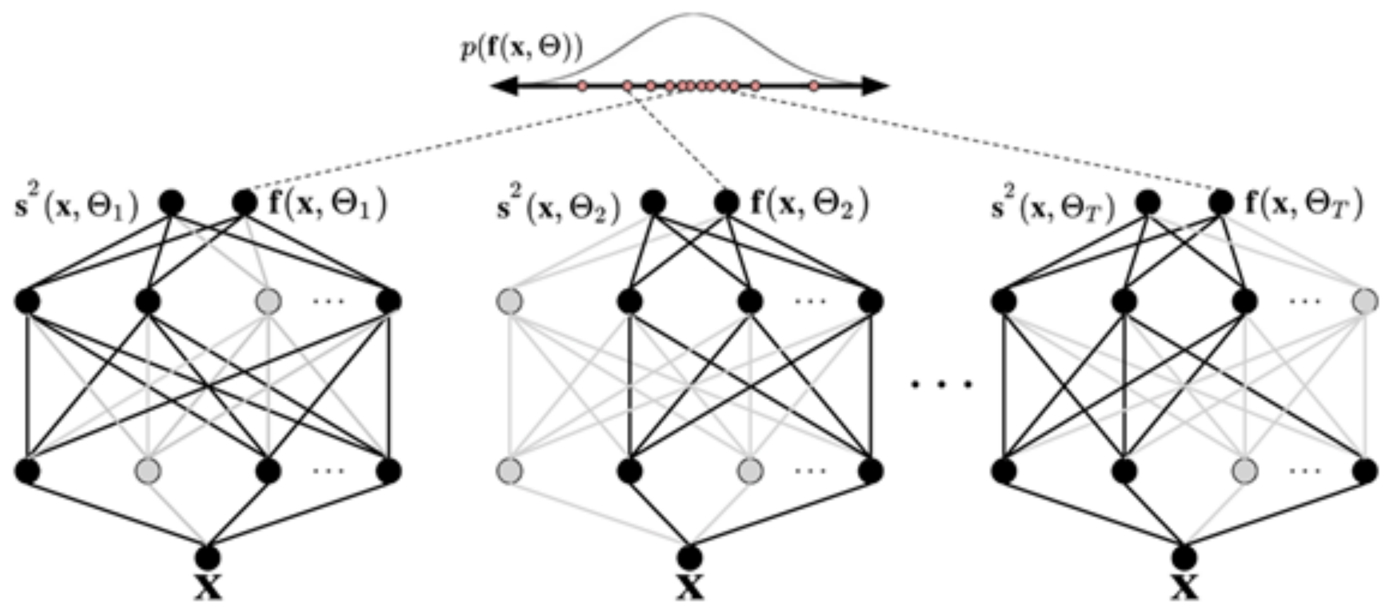


More Existing Approaches

Hamiltonian Monte Carlo

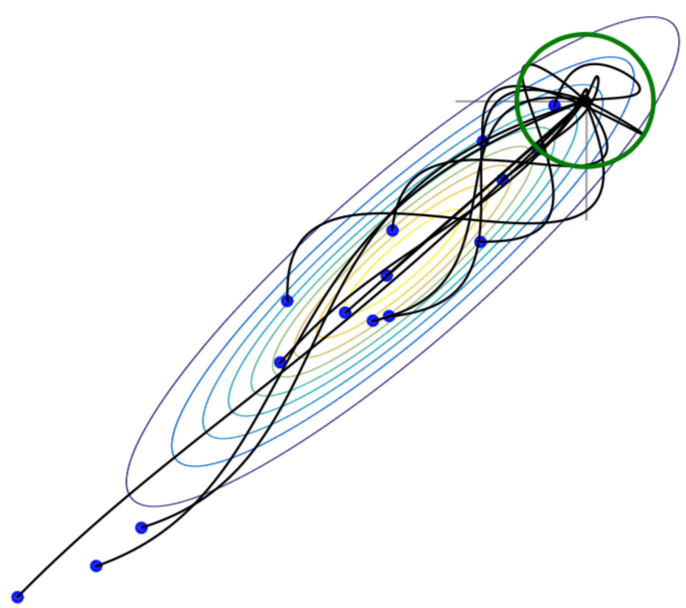


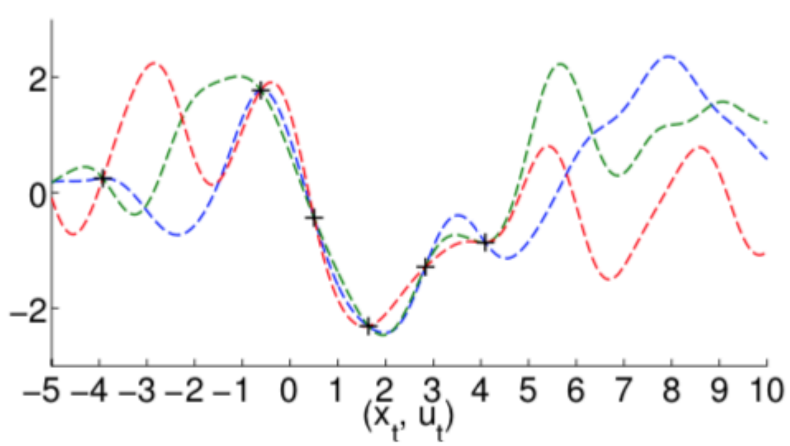
Monte Carlo Dropout



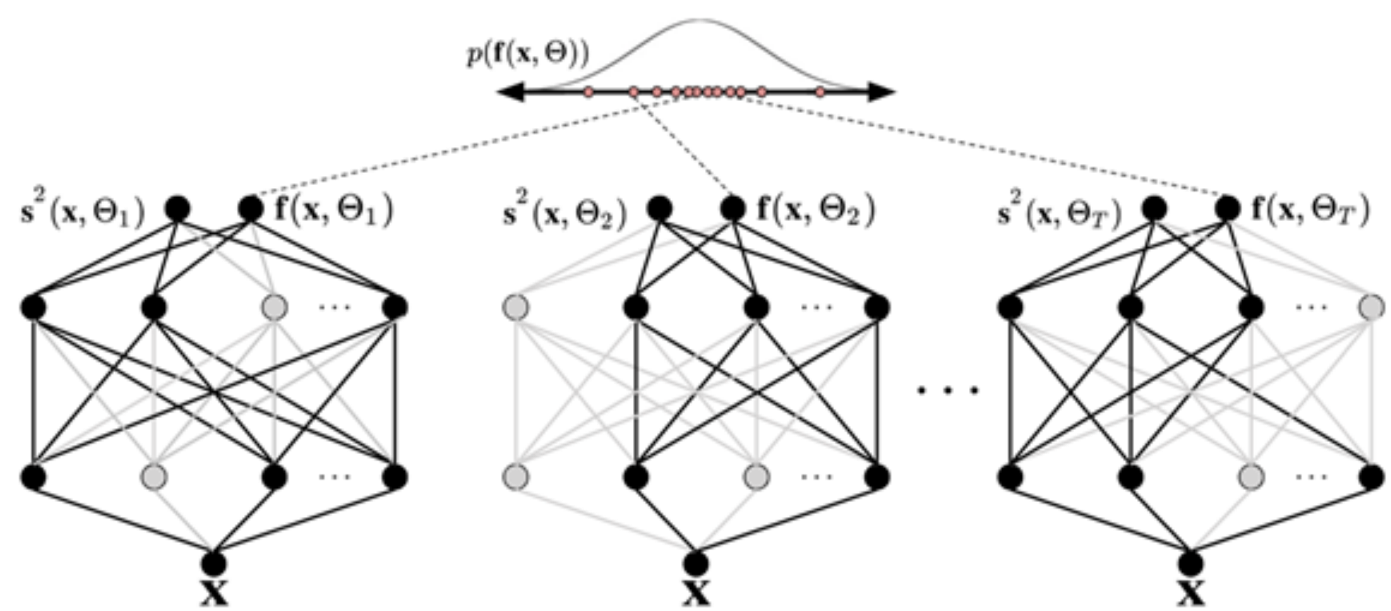
More Existing Approaches

Hamiltonian Monte Carlo



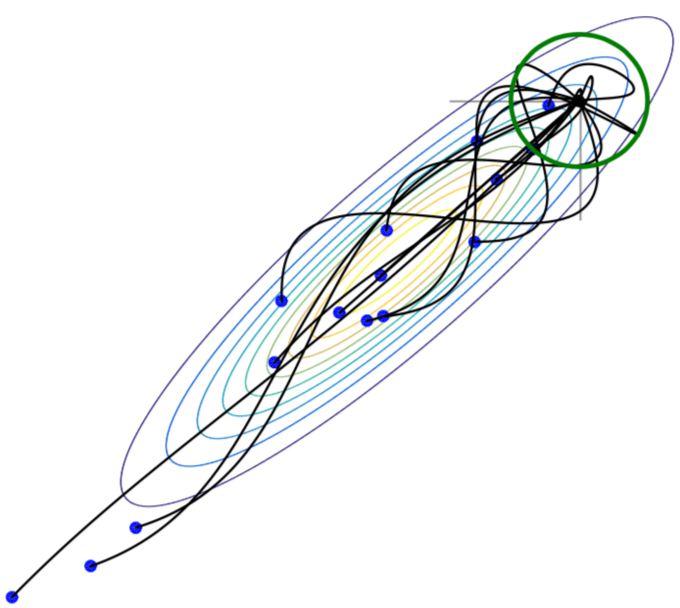


Monte Carlo Dropout

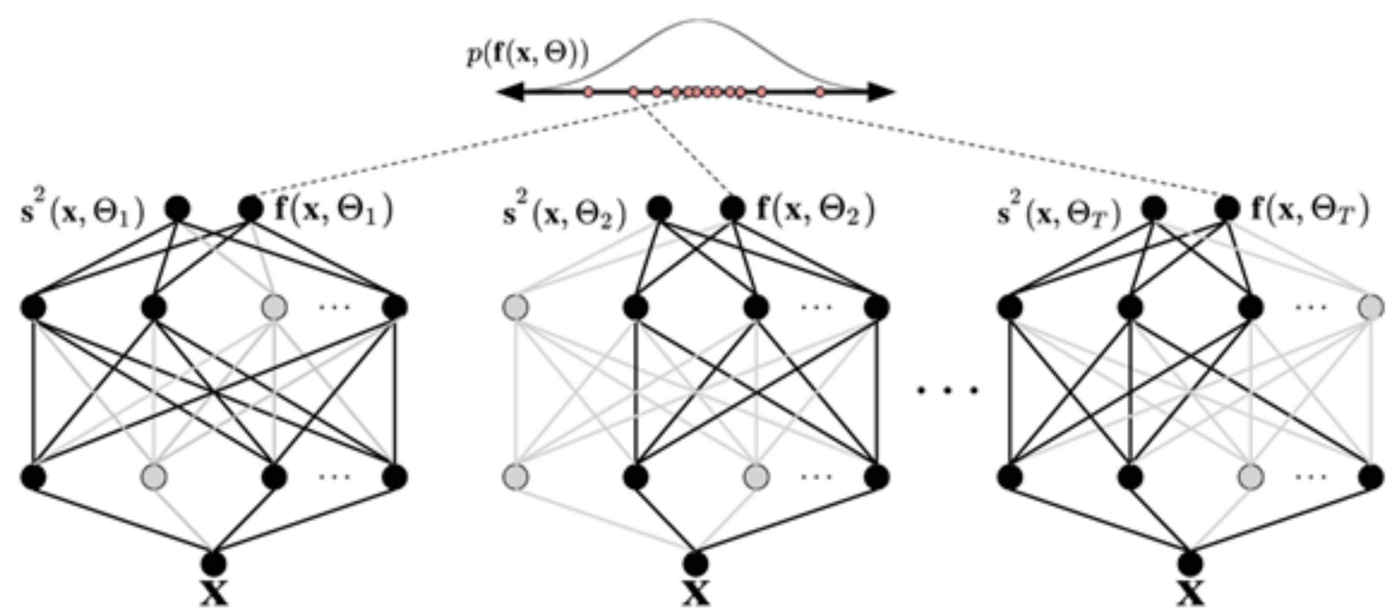


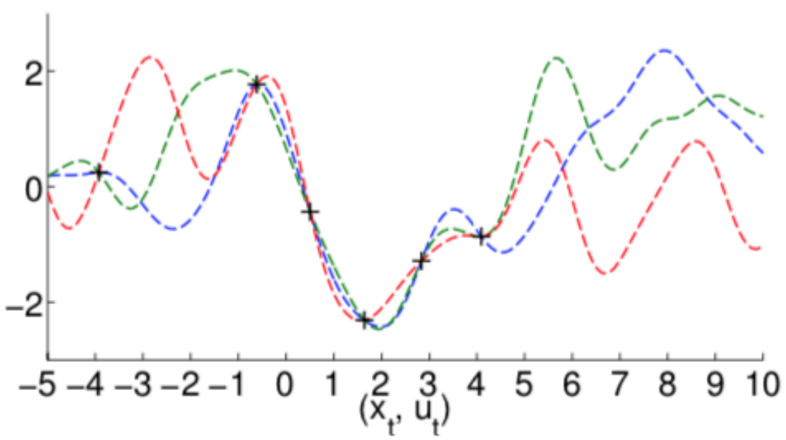
More Existing Approaches

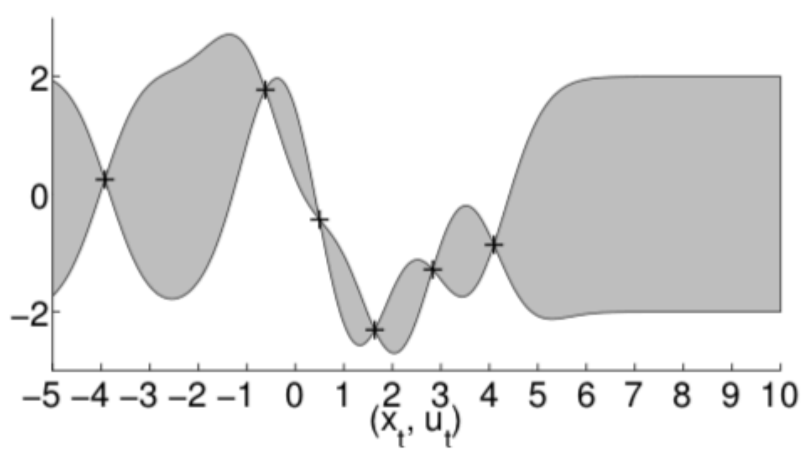
Hamiltonian Monte Carlo



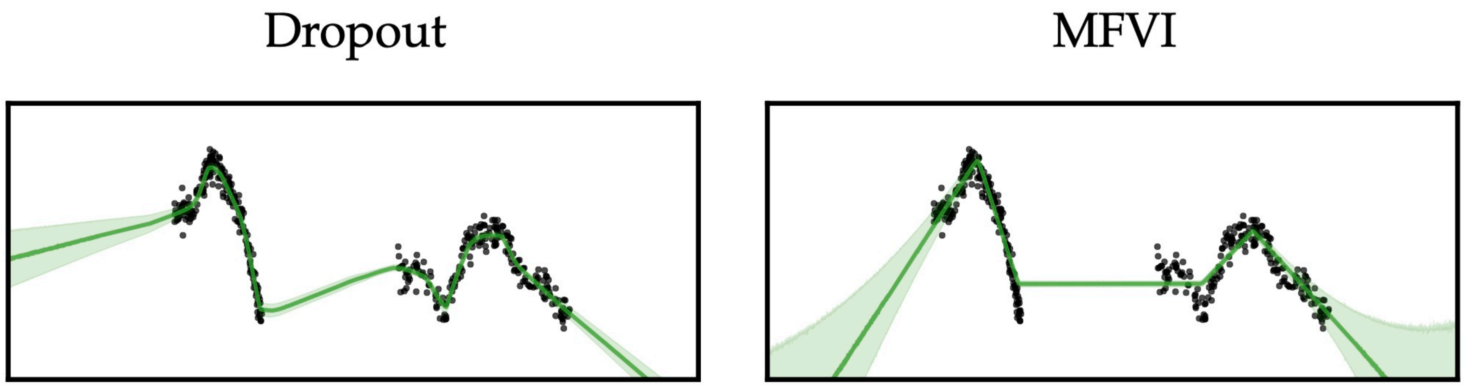
Monte Carlo Dropout



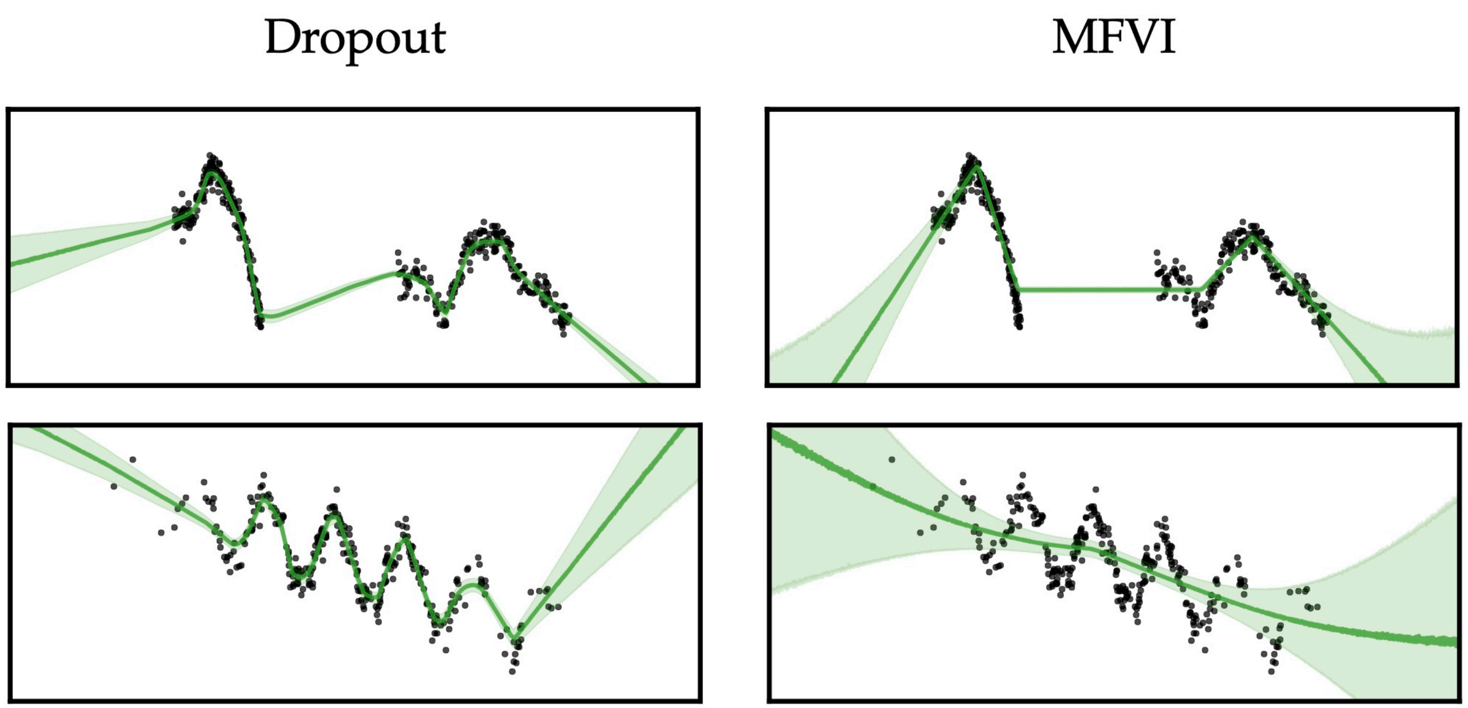




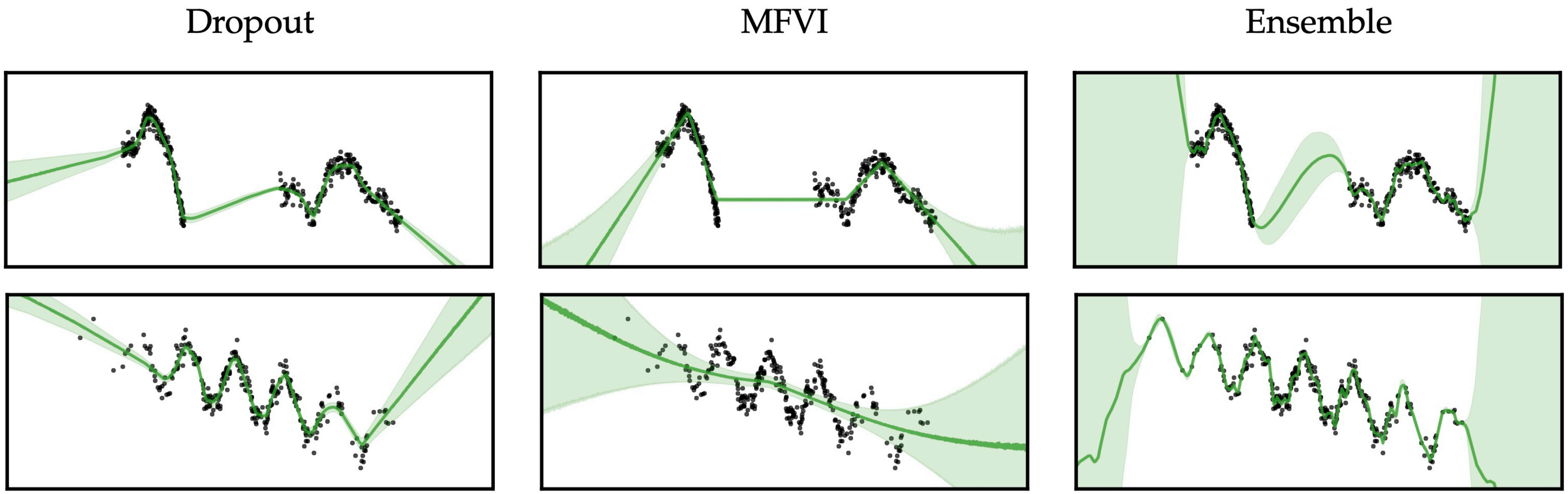
Disappointing Results



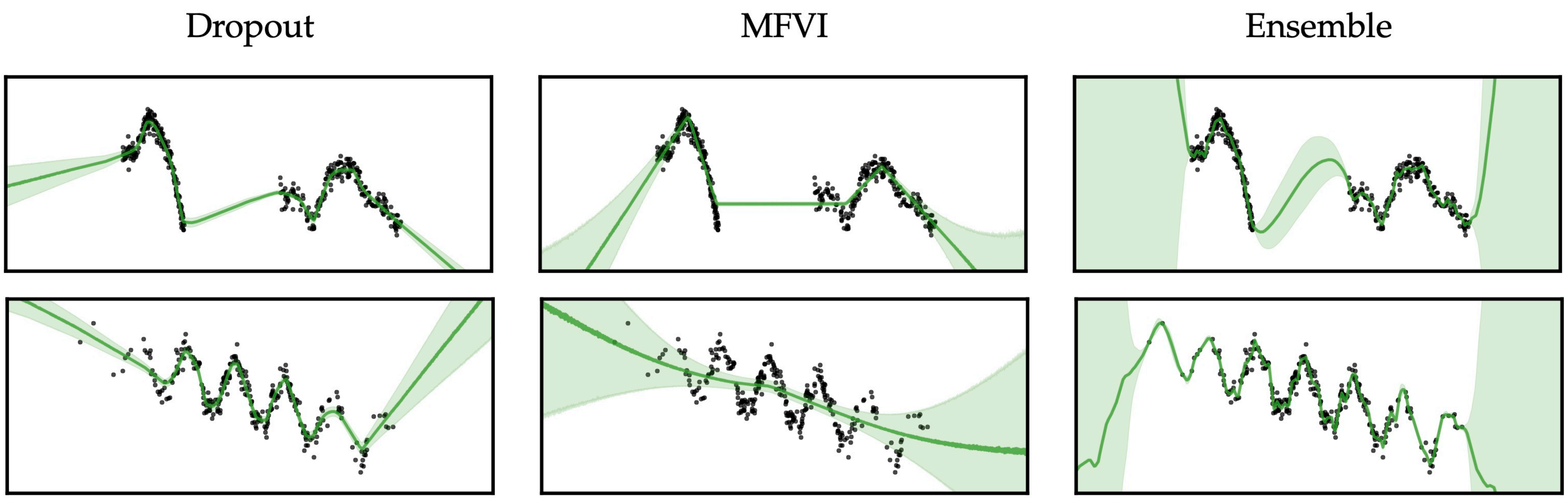
Disappointing Results



Disappointing Results



Disappointing Results

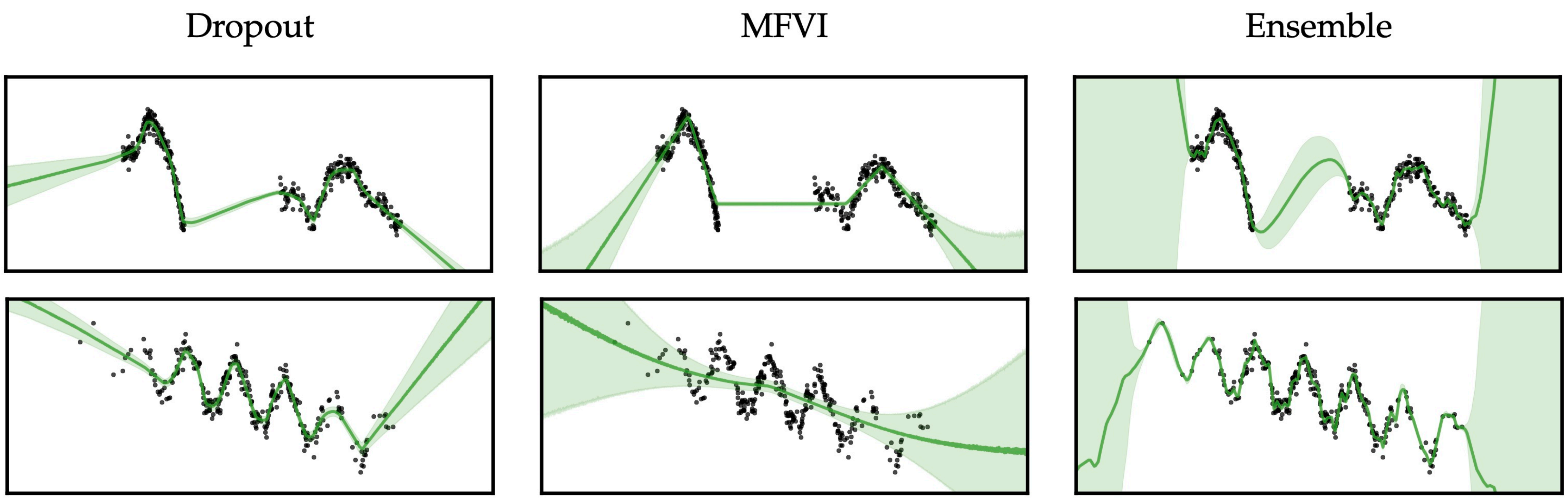


Detailed Studies:

Foong et al. “**On the expressiveness of approximate inference in bayesian neural networks.**” *NeurIPS* (2020).

Wenzel et al. “**How Good is the Bayes Posterior in Deep Neural Networks Really?**” *ICML* (2020)

Disappointing Results



Detailed Studies:

Foong et al. “**On the expressiveness of approximate inference in bayesian neural networks.**” *NeurIPS* (2020).

Wenzel et al. “**How Good is the Bayes Posterior in Deep Neural Networks Really?**” *ICML* (2020)

Try for yourself:

github.com/JavierAntoran/Bayesian-Neural-Networks

Motivation

Motivation

Observation: Almost all Bayesian deep learning methods try to do inference over **all** the weights of the DNN.

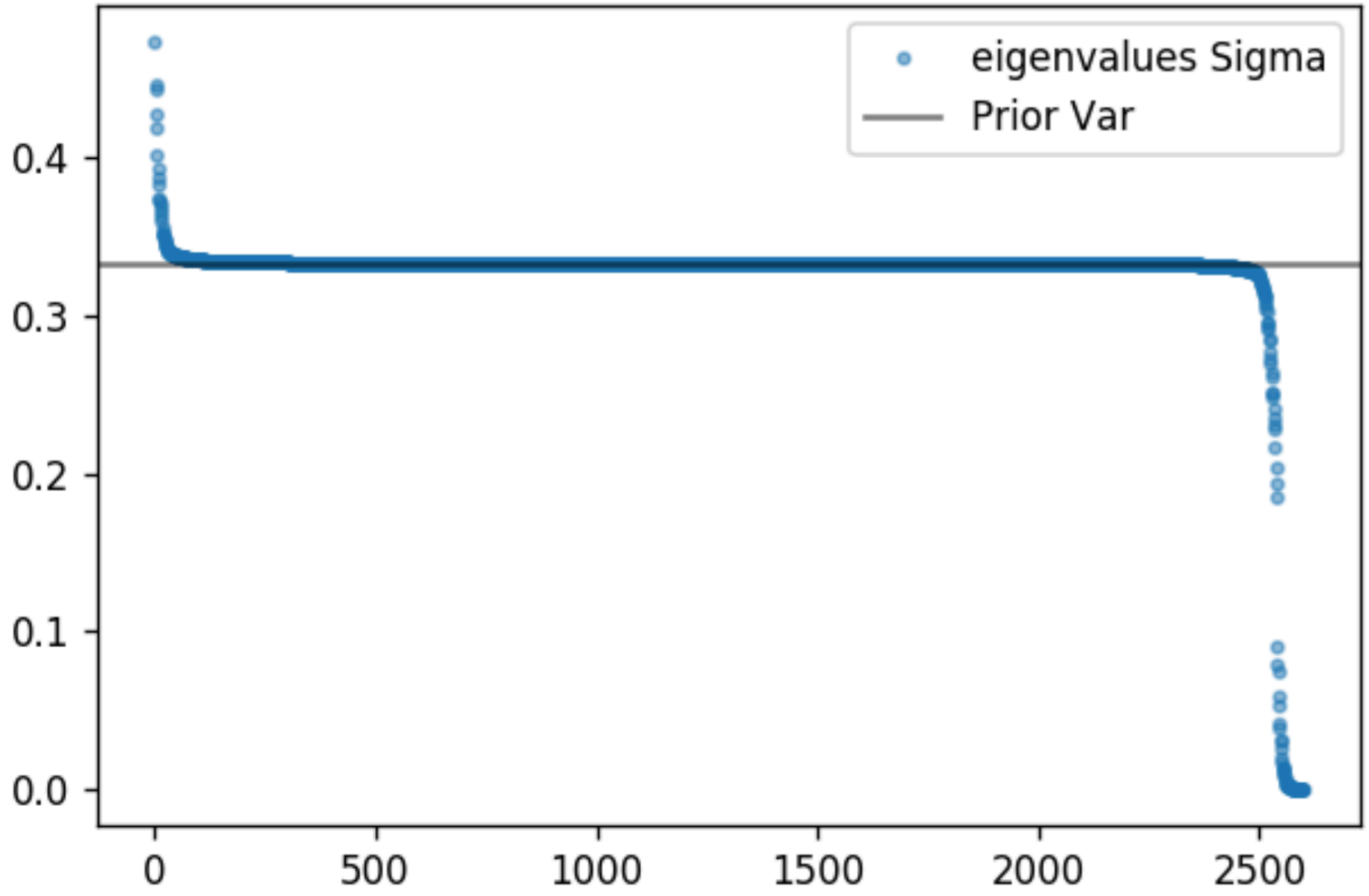
Motivation

Observation: Almost all Bayesian deep learning methods try to do inference over **all** the weights of the DNN.

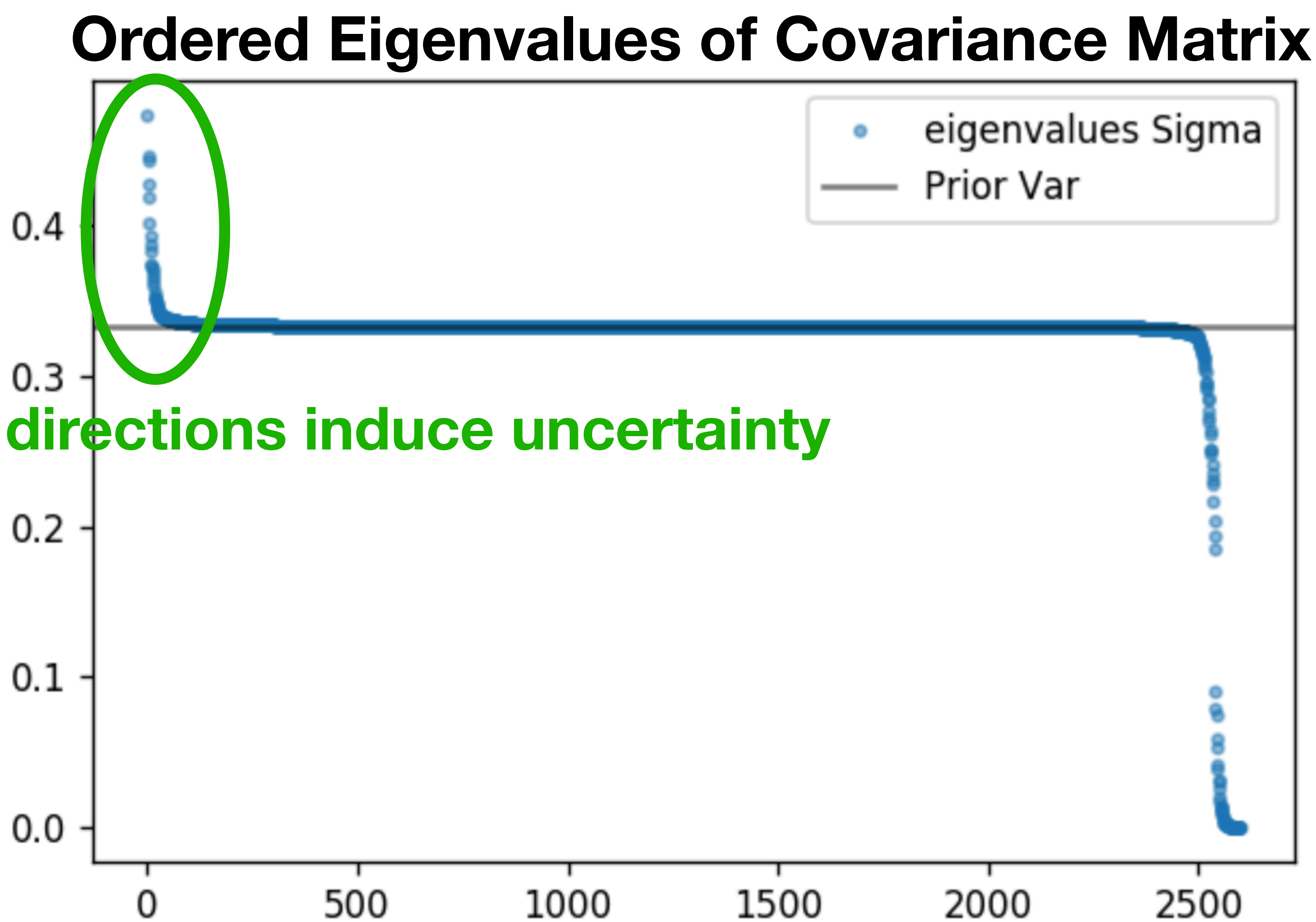
Do we really need to estimate a posterior over **ALL** the weights?!

Motivation

Ordered Eigenvalues of Covariance Matrix

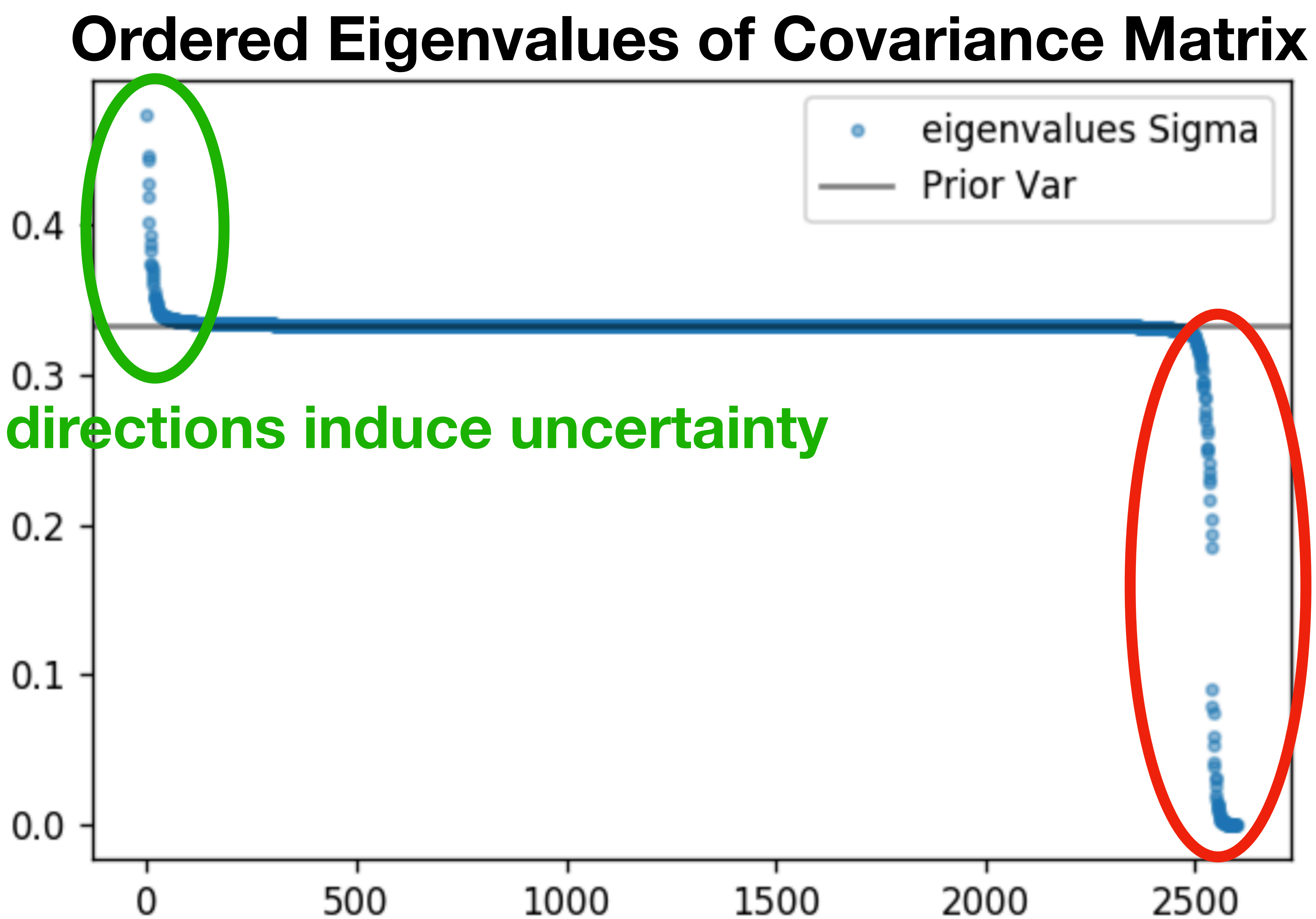


Motivation



Underspecified directions induce uncertainty

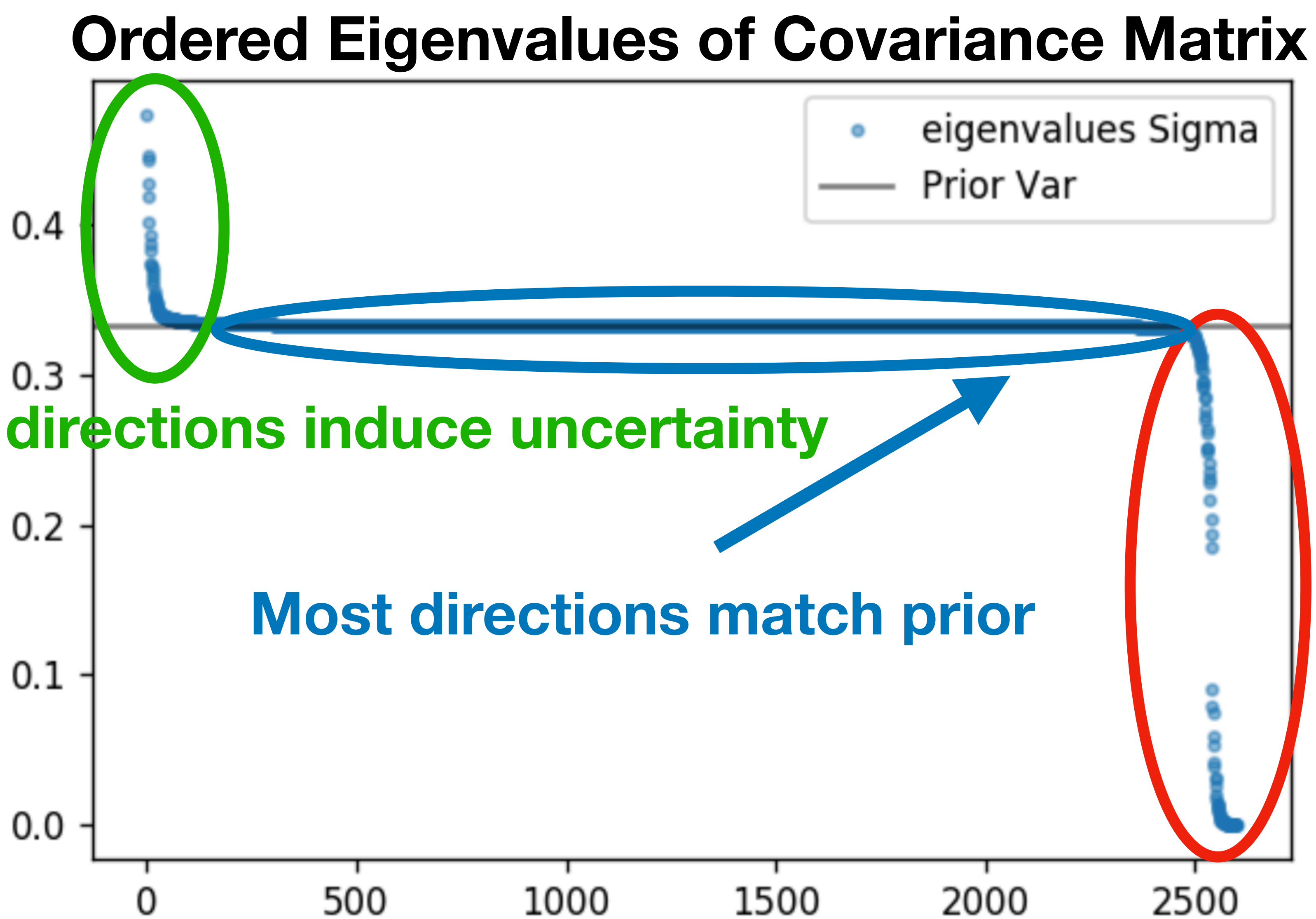
Motivation



Underspecified directions induce uncertainty

Strongly specified directions induce confident predictions

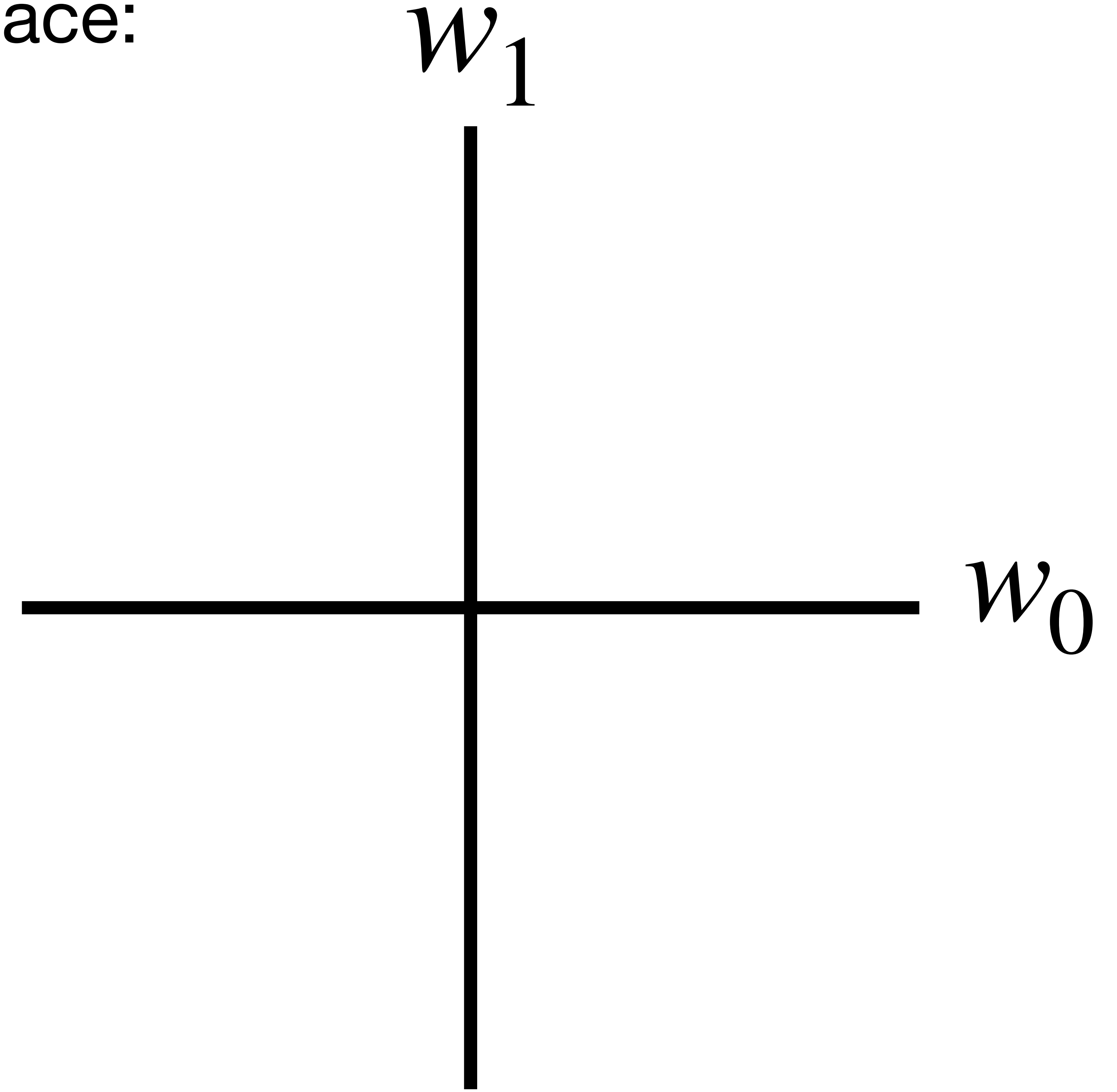
Motivation



Strongly specified directions induce confident predictions

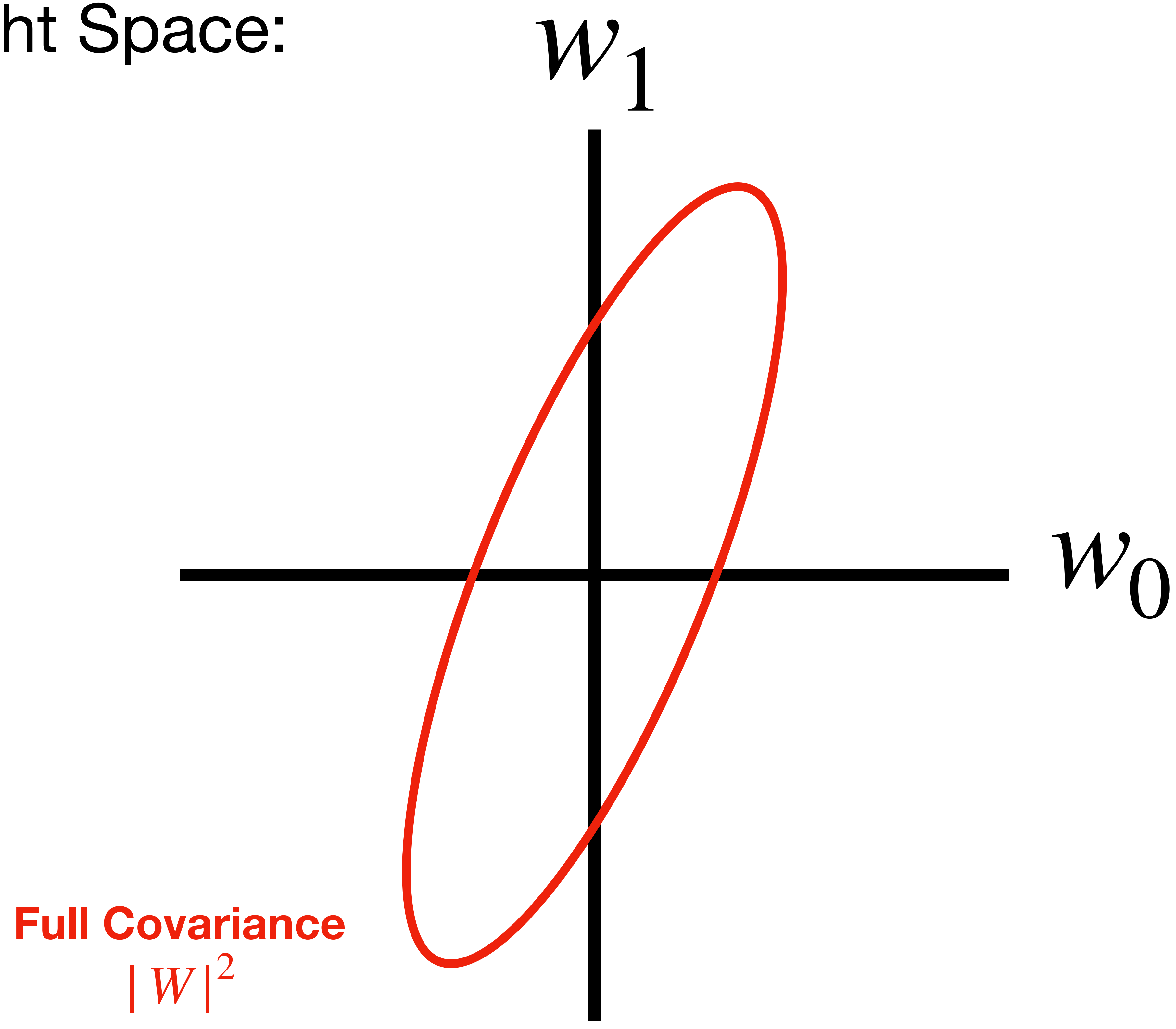
Motivation

Weight Space:



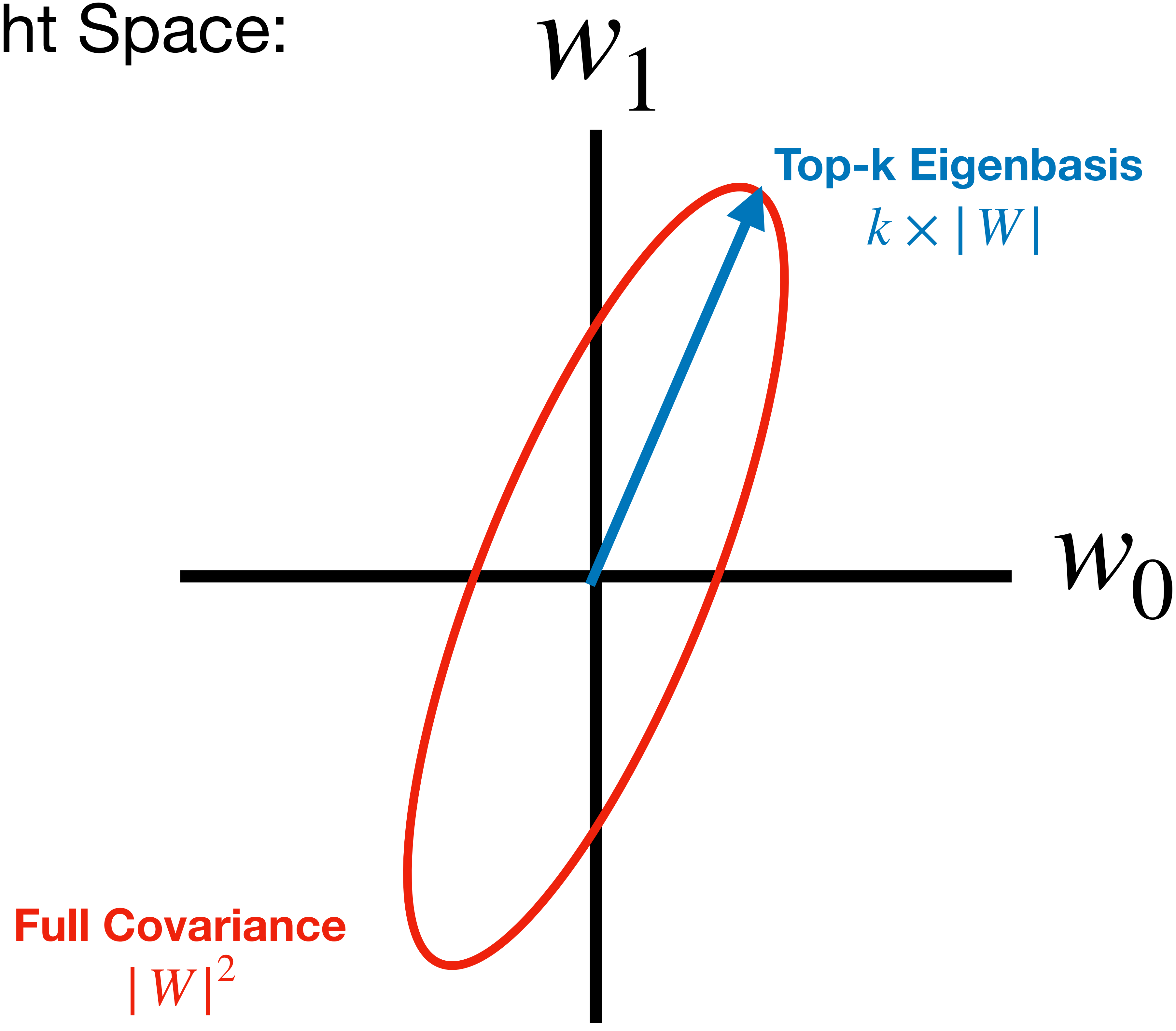
Motivation

Weight Space:



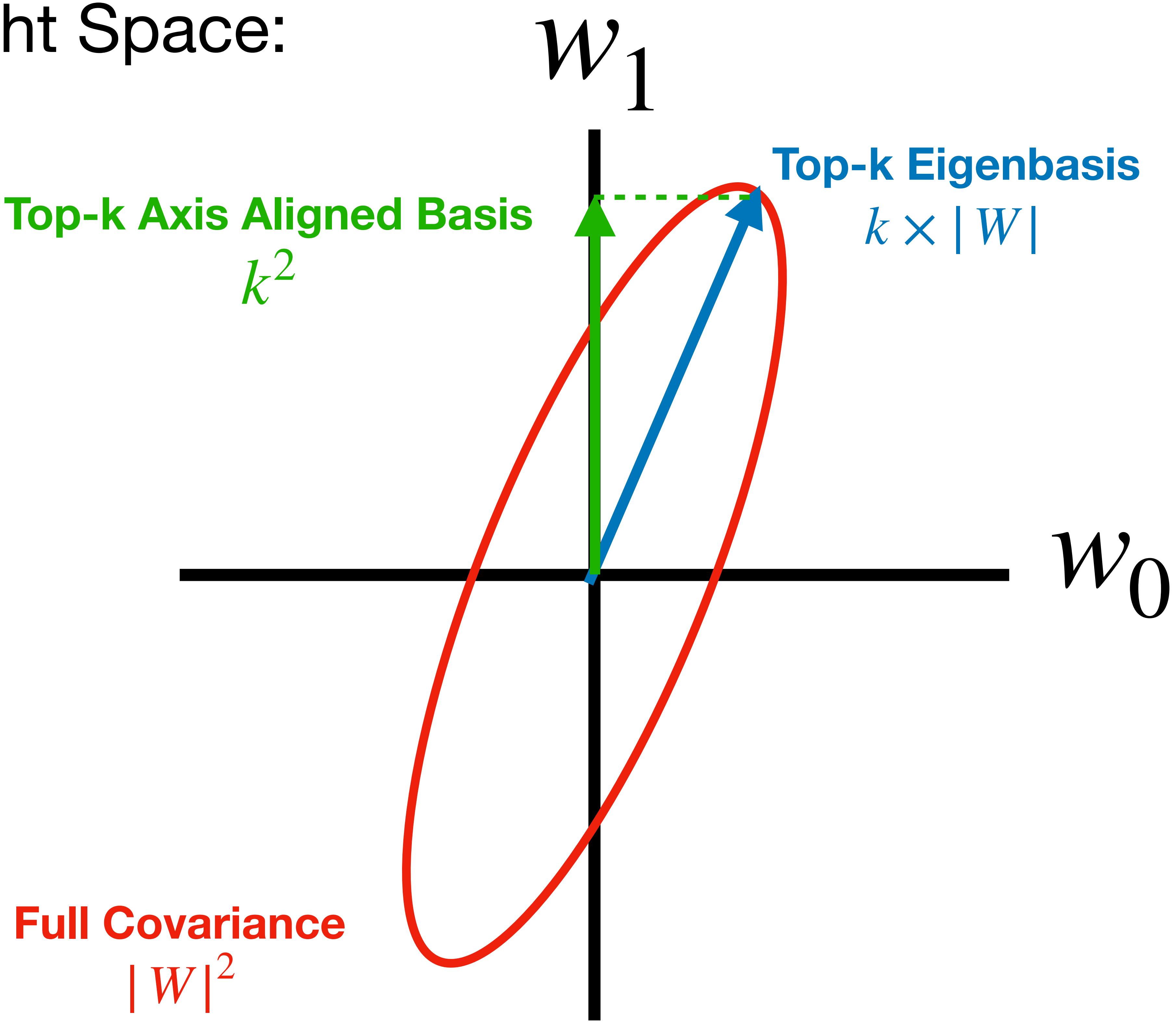
Motivation

Weight Space:



Motivation

Weight Space:



Idea

Idea

Observation: Due to overparameterization, a DNNs **accuracy** is well-preserved by a **small subnetwork**

Idea

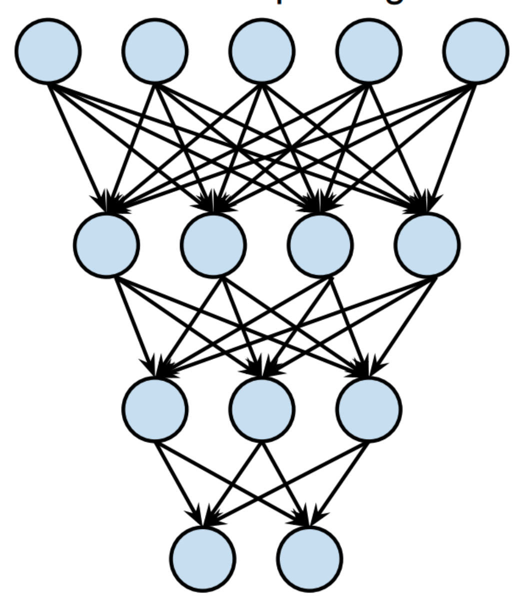
Observation: Due to overparameterization, a DNNs **accuracy** is well-preserved by a **small subnetwork**

How to find those subnetworks? —> DNN **pruning**, e.g. (Frankle & Carbin 2019)

Idea

Observation: Due to overparameterization, a DNNs **accuracy** is well-preserved by a **small subnetwork**

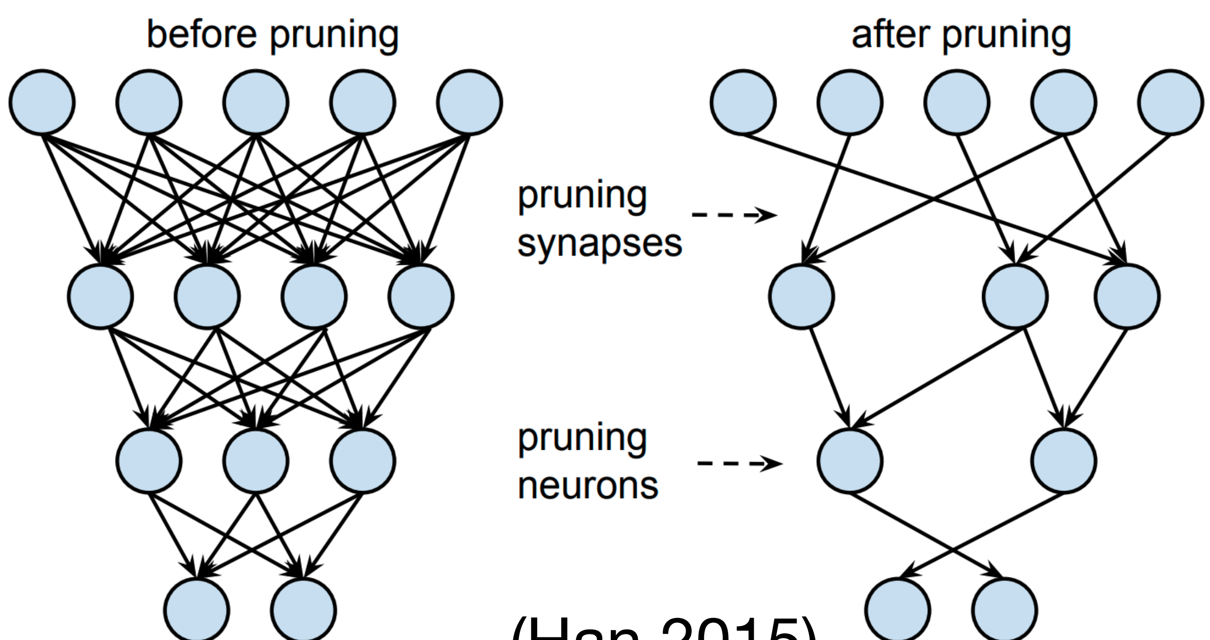
How to find those subnetworks? —> DNN **pruning**, e.g. (Frankle & Carbin 2019)



Idea

Observation: Due to overparameterization, a DNNs **accuracy** is well-preserved by a **small subnetwork**

How to find those subnetworks? —> DNN **pruning**, e.g. (Frankle & Carbin 2019)

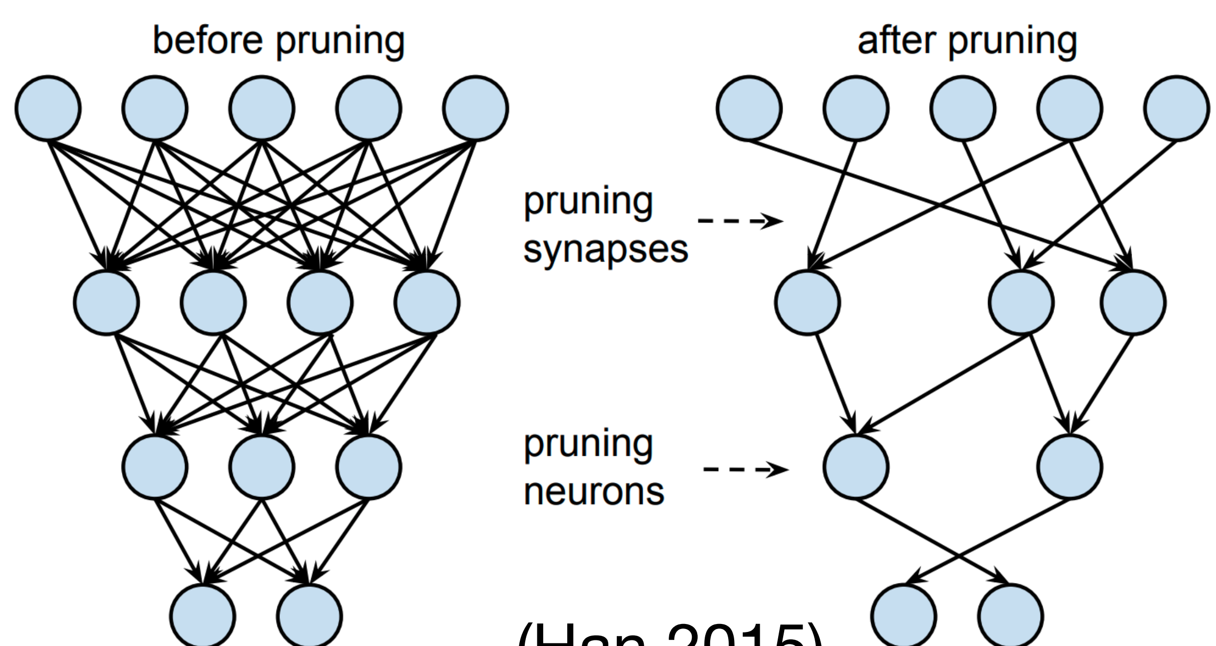


(Han 2015)

Idea

Observation: Due to overparameterization, a DNNs **accuracy** is well-preserved by a **small subnetwork**

How to find those subnetworks? —> DNN **pruning**, e.g. (Frankle & Carbin 2019)



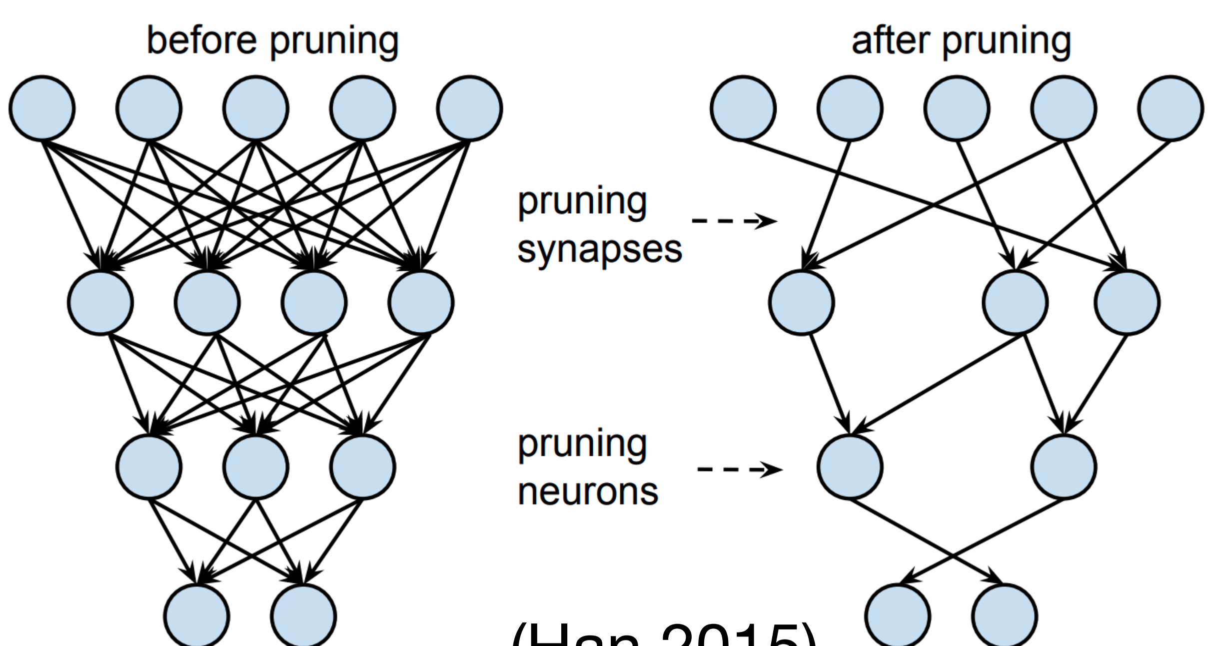
(Han 2015)

Question: **Can a full DNN's *model uncertainty* be well-preserved by a *small subnetwork's* model uncertainty?**

Idea

Observation: Due to overparameterization, a DNNs **accuracy** is well-preserved by a **small subnetwork**

How to find those subnetworks? —> DNN **pruning**, e.g. (Frankle & Carbin 2019)



(Han 2015)

Question: **Can a full DNN's *model uncertainty* be well-preserved by a *small subnetwork's* model uncertainty?**

Answer: This work shows that **Yes!**

Subnetwork Inference

Proposed Posterior Approximation:

$$p(\mathbf{W}|\boldsymbol{y}, \boldsymbol{X}) \approx q(\mathbf{W})$$

Subnetwork Inference

Proposed Posterior Approximation:

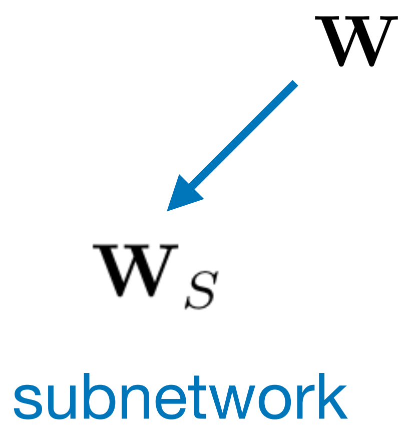
$\mathbf{W}$

$p(\mathbf{W}|\boldsymbol{y}, \boldsymbol{X}) \approx q(\mathbf{W})$

Subnetwork Inference

Proposed Posterior Approximation:

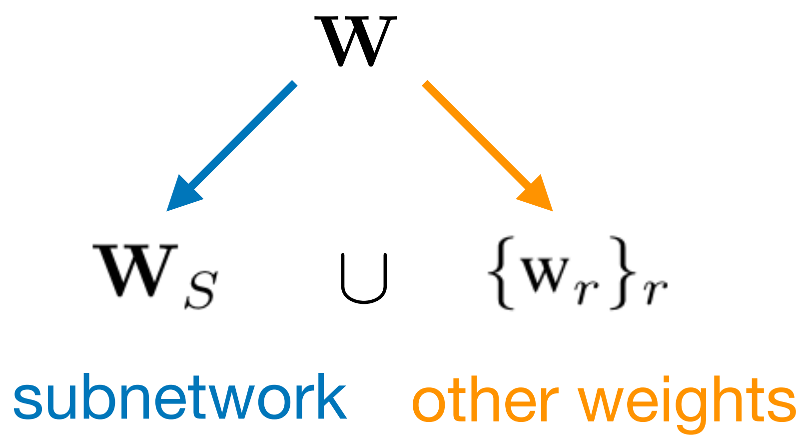
$$p(\mathbf{W}|\boldsymbol{y}, \boldsymbol{X}) \approx q(\mathbf{W})$$



Subnetwork Inference

Proposed Posterior Approximation:

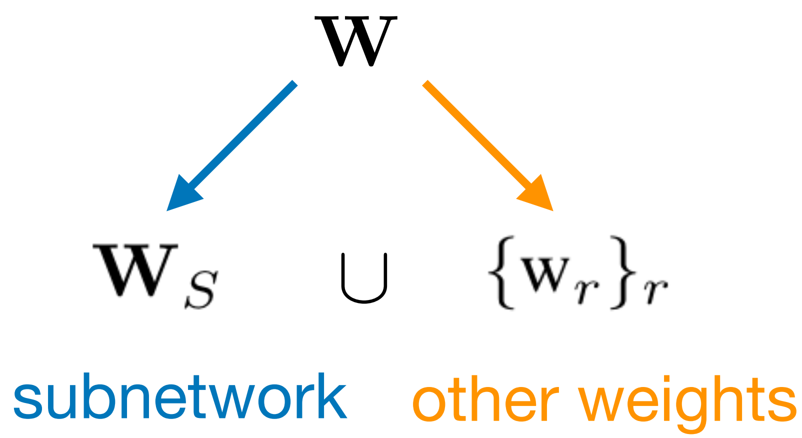
$$p(\mathbf{W}|\boldsymbol{y}, \boldsymbol{X}) \approx q(\mathbf{W})$$



Subnetwork Inference

Proposed Posterior Approximation:

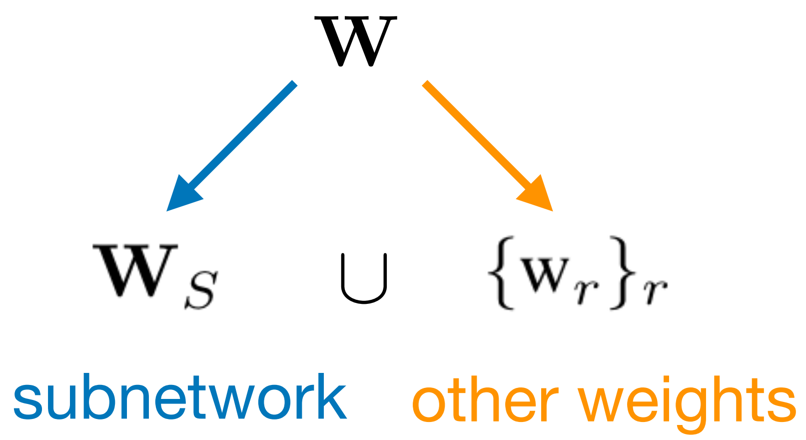
$$p(\mathbf{W}|\boldsymbol{y}, \boldsymbol{X}) \approx q(\mathbf{W}) = p(\mathbf{W}_S|\boldsymbol{y}, \boldsymbol{X}) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_r^*)$$



Subnetwork Inference

Proposed Posterior Approximation:

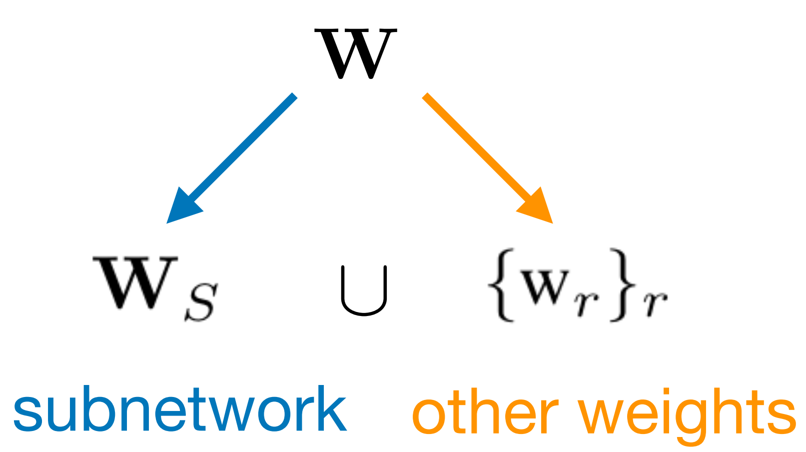
$$p(\mathbf{W}|\boldsymbol{y}, \boldsymbol{X}) \approx q(\mathbf{W}) = p(\mathbf{W}_S|\boldsymbol{y}, \boldsymbol{X}) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_r^*)$$



Subnetwork Inference

Proposed Posterior Approximation:

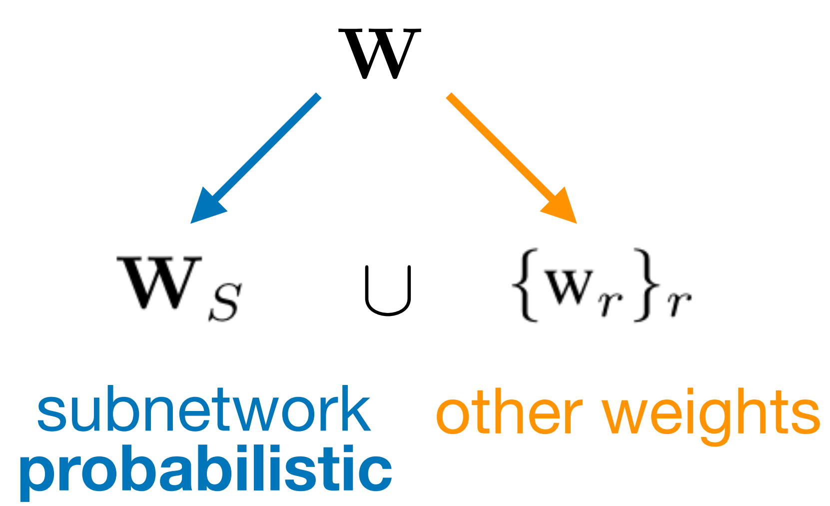
$$p(\mathbf{W}|\boldsymbol{y}, \boldsymbol{X}) \approx q(\mathbf{W}) = p(\mathbf{W}_S|\boldsymbol{y}, \boldsymbol{X}) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_r^*)$$



Subnetwork Inference

Proposed Posterior Approximation:

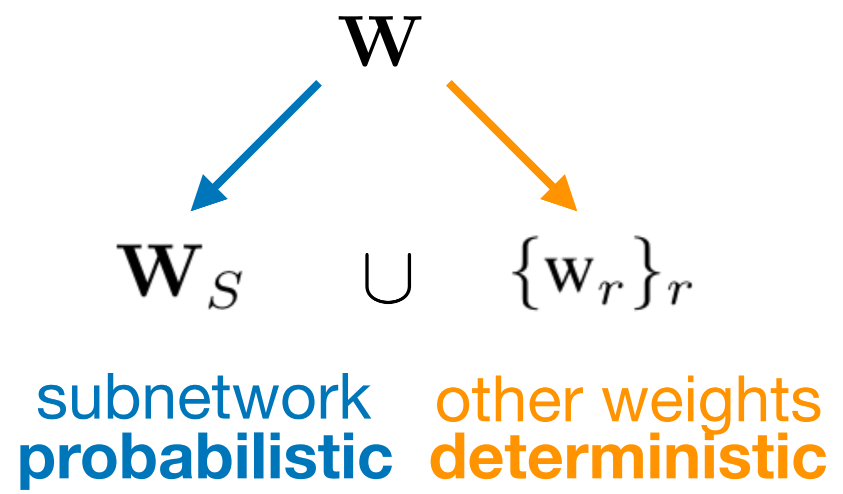
$$p(\mathbf{W}|\boldsymbol{y}, \boldsymbol{X}) \approx q(\mathbf{W}) = p(\mathbf{W}_S|\boldsymbol{y}, \boldsymbol{X}) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_r^*)$$



Subnetwork Inference

Proposed Posterior Approximation:

$$p(\mathbf{W}|\boldsymbol{y}, \boldsymbol{X}) \approx q(\mathbf{W}) = p(\mathbf{W}_S|\boldsymbol{y}, \boldsymbol{X}) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_r^*)$$

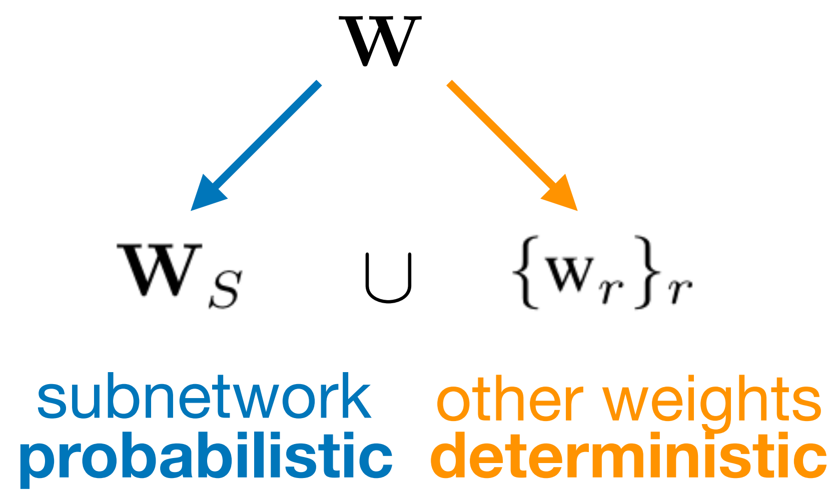


Subnetwork Inference

Proposed Posterior Approximation:

$$p(\mathbf{W}|\boldsymbol{y}, \boldsymbol{X}) \approx q(\mathbf{W}) = p(\mathbf{W}_S|\boldsymbol{y}, \boldsymbol{X}) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_r^*)$$

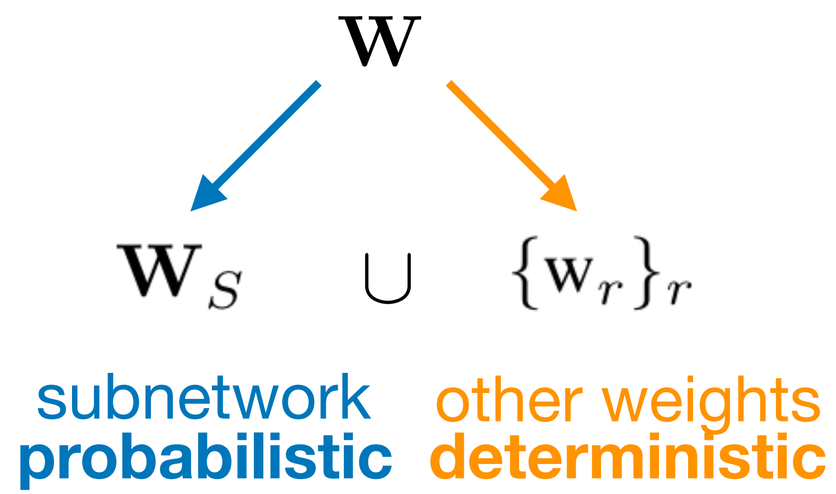
$$\approx q(\mathbf{W}_S) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_r^*)$$



Subnetwork Inference

Proposed Posterior Approximation:

$$
\begin{aligned}
p(\mathbf{W}|\boldsymbol{y}, \boldsymbol{X}) \approx q(\mathbf{W}) &= p(\mathbf{W}_S|\boldsymbol{y}, \boldsymbol{X}) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_r^*) \\
&\approx q(\mathbf{W}_S) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_r^*)
\end{aligned}
$$



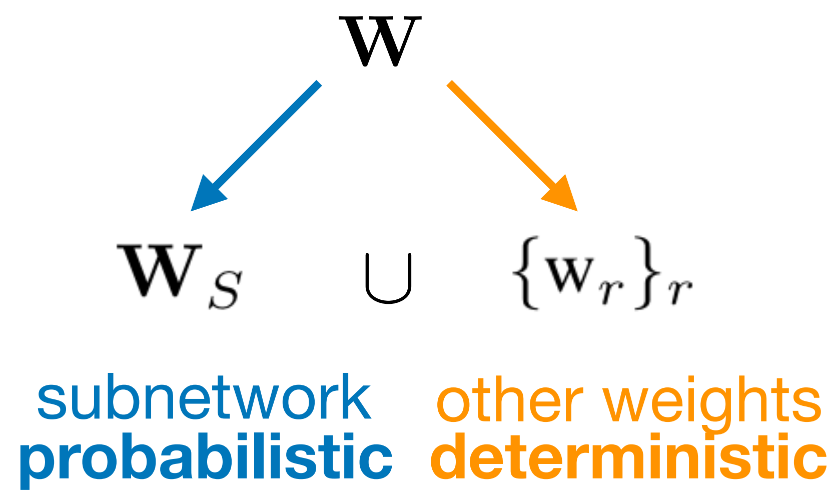
Questions:

Subnetwork Inference

Proposed Posterior Approximation:

$$p(\mathbf{W}|\boldsymbol{y}, \boldsymbol{X}) \approx q(\mathbf{W}) = p(\mathbf{W}_S|\boldsymbol{y}, \boldsymbol{X}) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_r^*)$$

$$\approx q(\mathbf{W}_S) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_r^*)$$



Questions:

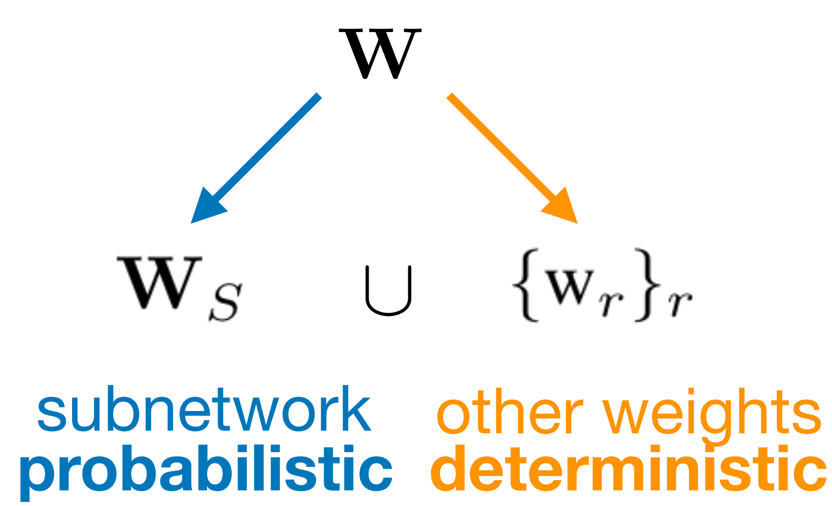
1. How do we choose and infer the subnetwork posterior $q(\mathbf{W}_S)$?

Subnetwork Inference

Proposed Posterior Approximation:

$$p(\mathbf{W}|\boldsymbol{y}, \boldsymbol{X}) \approx q(\mathbf{W}) = p(\mathbf{W}_S|\boldsymbol{y}, \boldsymbol{X}) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_r^*)$$

$$\approx q(\mathbf{W}_S) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_r^*)$$



Questions:

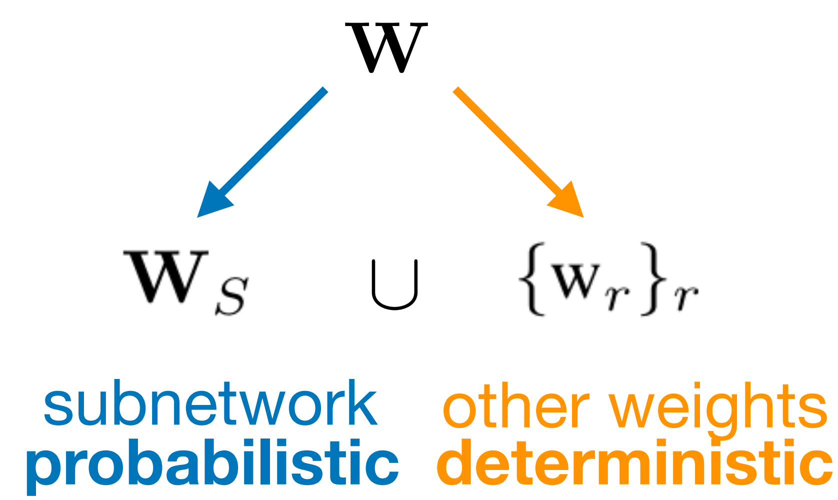
1. How do we choose and infer the subnetwork posterior $q(\mathbf{W}_S)$?
2. How do we set the fixed values $\mathrm{w}_r^* \in \mathbb{R}$ of all remaining weights $\{\mathrm{w}_r\}_r$?

Subnetwork Inference

Proposed Posterior Approximation:

$$p(\mathbf{W}|\boldsymbol{y}, \boldsymbol{X}) \approx q(\mathbf{W}) = p(\mathbf{W}_S|\boldsymbol{y}, \boldsymbol{X}) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_r^*)$$

$$\approx q(\mathbf{W}_S) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_r^*)$$



Questions:

1. How do we choose and infer the subnetwork posterior $q(\mathbf{W}_S)$?

2. How do we set the fixed values $\mathrm{w}_r^* \in \mathbb{R}$ of all remaining weights $\{\mathrm{w}_r\}_r$?

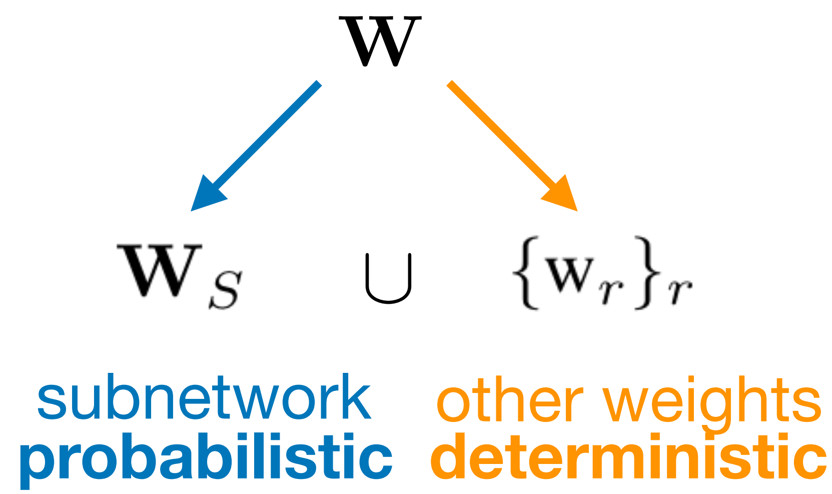
3. How do we select the subnetwork $\mathbf{W}_S$?

Subnetwork Inference

Proposed Posterior Approximation:

$$p(\mathbf{W}|\boldsymbol{y}, \boldsymbol{X}) \approx q(\mathbf{W}) = p(\mathbf{W}_S|\boldsymbol{y}, \boldsymbol{X}) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_r^*)$$

$$\approx q(\mathbf{W}_S) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_r^*)$$



Questions:

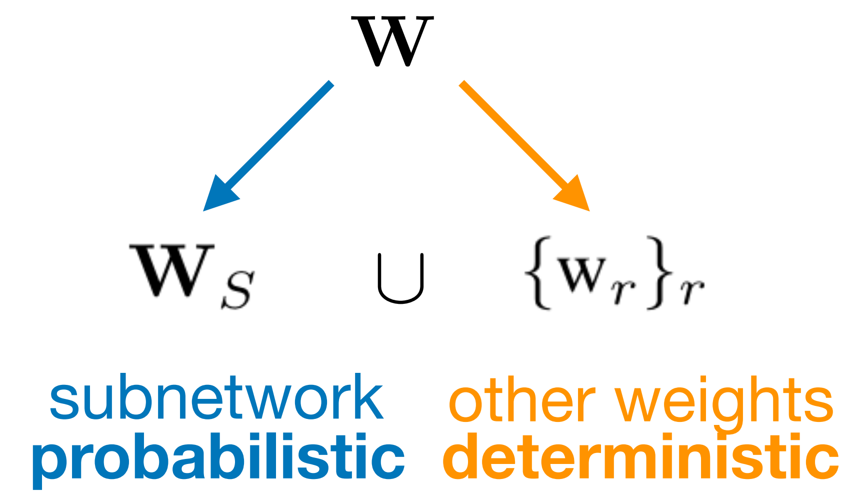
1. How do we choose and infer the subnetwork posterior $q(\mathbf{W}_S)$?
2. How do we set the fixed values $\mathrm{w}_r^* \in \mathbb{R}$ of all remaining weights $\{\mathrm{w}_r\}_r$?
3. How do we select the subnetwork $\mathbf{W}_S$?
4. How do we make predictions with the approximate posterior $q(\mathbf{W})$?

Subnetwork Inference

Proposed Posterior Approximation:

$$p(\mathbf{W}|\boldsymbol{y}, \boldsymbol{X}) \approx q(\mathbf{W}) = p(\mathbf{W}_S|\boldsymbol{y}, \boldsymbol{X}) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_r^*)$$

$$\approx q(\mathbf{W}_S) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_r^*)$$



Questions:

1. How do we choose and infer the subnetwork posterior $q(\mathbf{W}_S)$?

2. How do we set the fixed values $\mathrm{w}_r^* \in \mathbb{R}$ of all remaining weights $\{\mathrm{w}_r\}_r$?

3. How do we select the subnetwork $\mathbf{W}_S$?

4. How do we make predictions with the approximate posterior $q(\mathbf{W})$?

Linearised Laplace Approximation

Linearised Laplace Approximation

Attractive for its post-hoc nature and strong empirical performance!

Linearised Laplace Approximation

Attractive for its post-hoc nature and strong empirical performance!

1. Train a regular NN (with SGD, Adam, etc)

$$\widehat{W} = arg\,max_W \left[\log p(y \,|\, X, W) + \log p(W)\right].$$

Linearised Laplace Approximation

Attractive for its post-hoc nature and strong empirical performance!

1. Train a regular NN (with SGD, Adam, etc)

$$\widehat{W} = arg\,max_W \left[\log p(y \,|\, X, W) + \log p(W)\right].$$

2. Approximate NN with a 1st order Taylor expansion around mode $\widehat{W}$

$$f_{lin}(x, W) = f(x, \widehat{W}) + \hat{J}(x)(W - \widehat{W}); \quad \hat{J}(x) = \frac{\partial f(x, W)}{\partial W}\Big|_{W=\widehat{W}}$$

Linearised Laplace Approximation

Attractive for its post-hoc nature and strong empirical performance!

1. Train a regular NN (with SGD, Adam, etc)

$$\widehat{W} = arg\, max_W \left[\log p(y \,|\, X, W) + \log p(W)\right].$$

2. Approximate NN with a 1st order Taylor expansion around mode $\widehat{W}$

$$f_{lin}(x, W) = f(x, \widehat{W}) + \hat{J}(x)(W - \widehat{W}); \quad \hat{J}(x) = \frac{\partial f(x, W)}{\partial W} |_{W=\widehat{W}}$$

3. Infer a Gaussian posterior for the resulting basis function linear model

Linearised Laplace Approximation

Attractive for its post-hoc nature and strong empirical performance!

1. Train a regular NN (with SGD, Adam, etc)

$$\widehat{W} = arg\,max_W \left[\log p(y \mid X, W) + \log p(W)\right].$$

2. Approximate NN with a 1st order Taylor expansion around mode $\widehat{W}$

$$f_{lin}(x, W) = f(x, \widehat{W}) + \hat{J}(x)(W - \widehat{W}); \quad \hat{J}(x) = \frac{\partial f(x, W)}{\partial W}\Big|_{W=\widehat{W}}$$

3. Infer a Gaussian posterior for the resulting basis function linear model

$$p(W) = \mathcal{N}\left(W; 0, \lambda^{-1} \cdot I\right)$$

Linearised Laplace Approximation

Attractive for its post-hoc nature and strong empirical performance!

1. Train a regular NN (with SGD, Adam, etc)

$$\widehat{W} = arg\,max_W \left[\log p(y \,|\, X, W) + \log p(W)\right].$$

2. Approximate NN with a 1st order Taylor expansion around mode $\widehat{W}$

$$f_{lin}(x, W) = f(x, \widehat{W}) + \hat{J}(x)(W - \widehat{W}); \quad \hat{J}(x) = \frac{\partial f(x, W)}{\partial W}\Big|_{W=\widehat{W}}$$

3. Infer a Gaussian posterior for the resulting basis function linear model

$$p(W) = \mathcal{N}\left(W;\, 0, \lambda^{-1} \cdot I\right) \quad p(W \,|\, \mathcal{D}) \simeq q(W) = \mathcal{N}\left(W;\, \widehat{W}, H^{-1}\right)$$

Linearised Laplace Approximation

Attractive for its post-hoc nature and strong empirical performance!

1. Train a regular NN (with SGD, Adam, etc)

$$\widehat{W} = arg\, max_W \left[\log p(y \,|\, X, W) + \log p(W)\right].$$

2. Approximate NN with a 1st order Taylor expansion around mode $\widehat{W}$

$$f_{lin}(x, W) = f(x, \widehat{W}) + \hat{J}(x)(W - \widehat{W}); \quad \hat{J}(x) = \frac{\partial f(x, W)}{\partial W}\Big|_{W=\widehat{W}}$$

3. Infer a Gaussian posterior for the resulting basis function linear model

$$p(W) = \mathcal{N}\left(W;\, 0, \lambda^{-1} \cdot I\right) \quad p(W \,|\, \mathcal{D}) \simeq q(W) = \mathcal{N}\left(W;\, \widehat{W}, H^{-1}\right) \quad H = J^\top \frac{\partial^2 \log p(y \,|\, f(X, W))}{\partial^2 f(X, W)} J + \lambda \cdot I$$

Linearised Laplace Approximation

Attractive for its post-hoc nature and strong empirical performance!

1. Train a regular NN (with SGD, Adam, etc)

$$\widehat{W} = arg\,max_W \left[\log p(y \,|\, X, W) + \log p(W)\right].$$

2. Approximate NN with a 1st order Taylor expansion around mode $\widehat{W}$

$$f_{lin}(x, W) = f(x, \widehat{W}) + \hat{J}(x)(W - \widehat{W}); \quad \hat{J}(x) = \frac{\partial f(x, W)}{\partial W}|_{W=\widehat{W}}$$

3. Infer a Gaussian posterior for the resulting basis function linear model

$$p(W) = \mathcal{N}\left(W;\, 0, \lambda^{-1} \cdot I\right) \quad p(W \,|\, \mathcal{D}) \simeq q(W) = \mathcal{N}\left(W;\, \widehat{W}, H^{-1}\right) \quad H = J^{\top} \frac{\partial^2 \log p(y \,|\, f(X, W))}{\partial^2 f(X, W)} J + \lambda \cdot I$$

4. The NN's uncertainty is approximated by the uncertainty of the linear model

$$p(y^* \,|\, x^*, \mathcal{D}) = \mathcal{N}(y^*;\, f(x^*, \widehat{W}),\, J^{\top}(x^*)H^{-1}J(x^*) + \sigma^2 I)$$

Linearised Laplace Approximation

Linearised Laplace Approximation

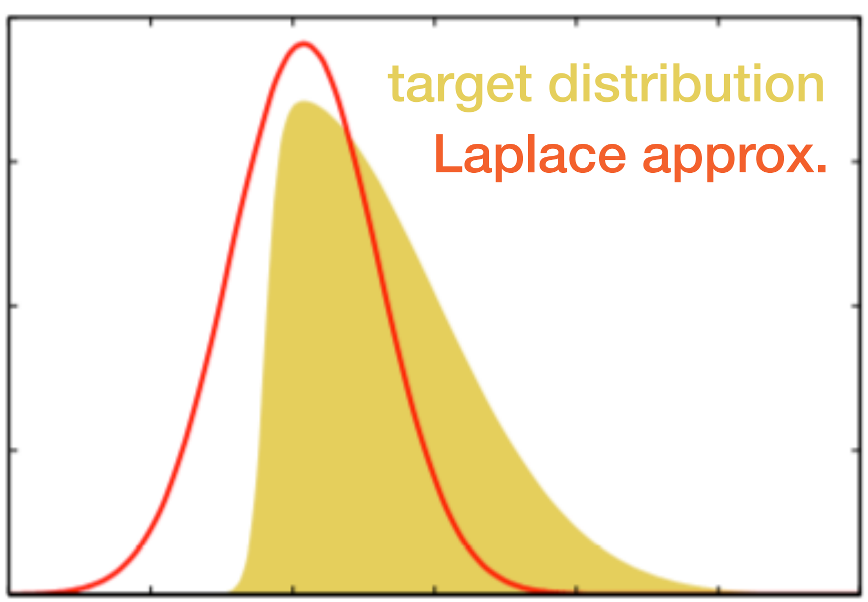
$$p(y^* \,|\, x^*, \mathcal{D}) = \mathcal{N}(y^*; \underbrace{f(x^*, \widehat{W})}_{}, J^\top(x^*)H^{-1}J(x^*) + \sigma^2 I)$$

Preserves mean of pre-trained NN

Linearised Laplace Approximation

$$p(y^* \mid x^*, \mathscr{D}) = \mathscr{N}(y^*; \underbrace{f(x^*, \widehat{W})}, J^\top(x^*)H^{-1}J(x^*) + \sigma^2 I)$$

Preserves mean of pre-trained NN



(Bishop 2006)

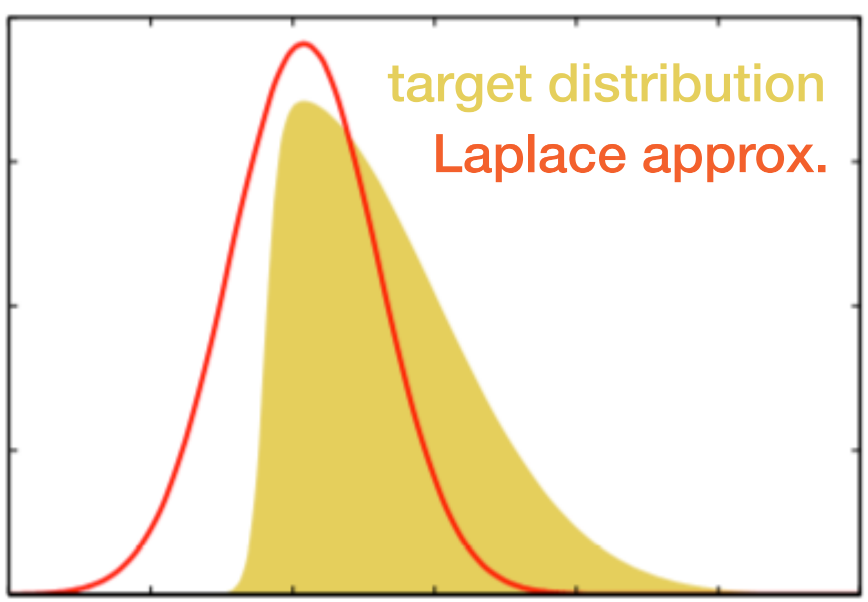
Exact for Gaussian Likelihoods (Regression)

Pretty good for Categorical Likelihoods (Classification)

Linearised Laplace Approximation

$$p(y^* \mid x^*, \mathcal{D}) = \mathcal{N}(y^*; \underbrace{f(x^*, \widehat{W})}, J^\top(x^*) H^{-1} J(x^*) + \sigma^2 I)$$

Preserves mean of pre-trained NN



(Bishop 2006)

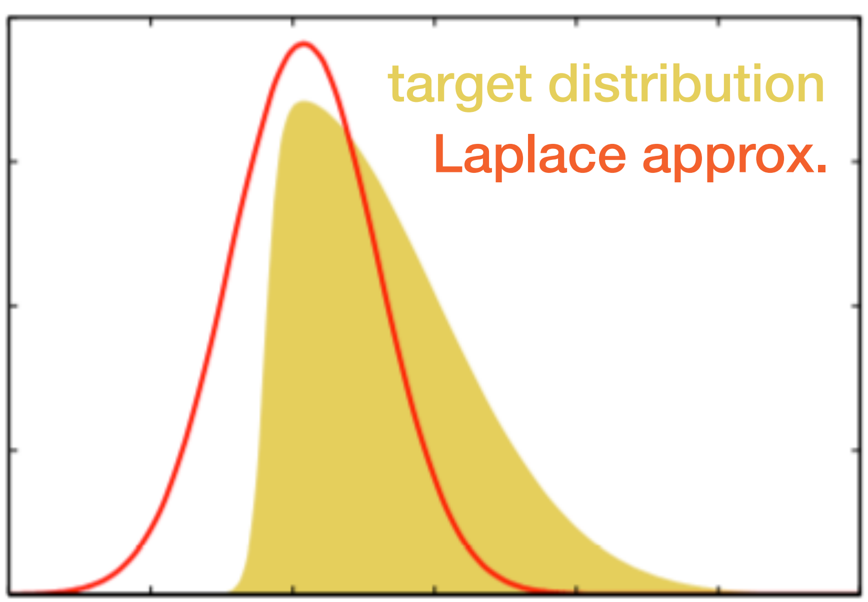
Exact for Gaussian Likelihoods (Regression)

Pretty good for Categorical Likelihoods (Classification)

Problem: H **is too large. Intractable to store and invert**

Linearised Laplace Approximation

$$p(y^* \mid x^*, \mathcal{D}) = \mathcal{N}(y^*; \underbrace{f(x^*, \widehat{W})}_{\text{Preserves mean of pre-trained NN}}, J^\top(x^*) H^{-1} J(x^*) + \sigma^2 I)$$

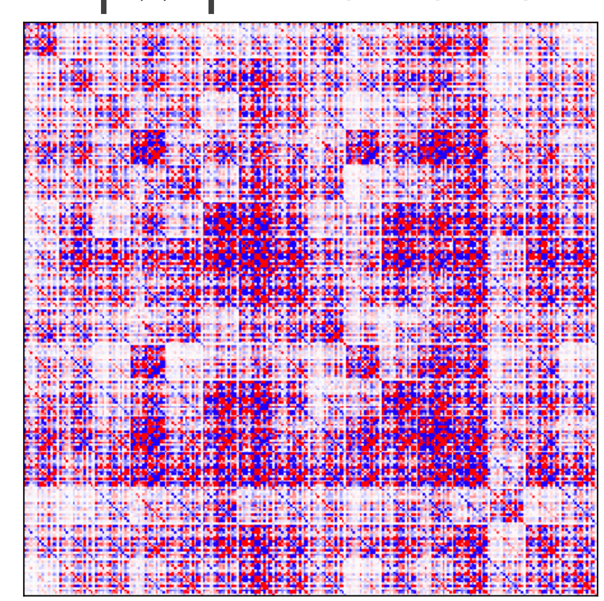


(Bishop 2006)

Exact for Gaussian Likelihoods (Regression)

Pretty good for Categorical Likelihoods (Classification)

$|W|^2$ Elements

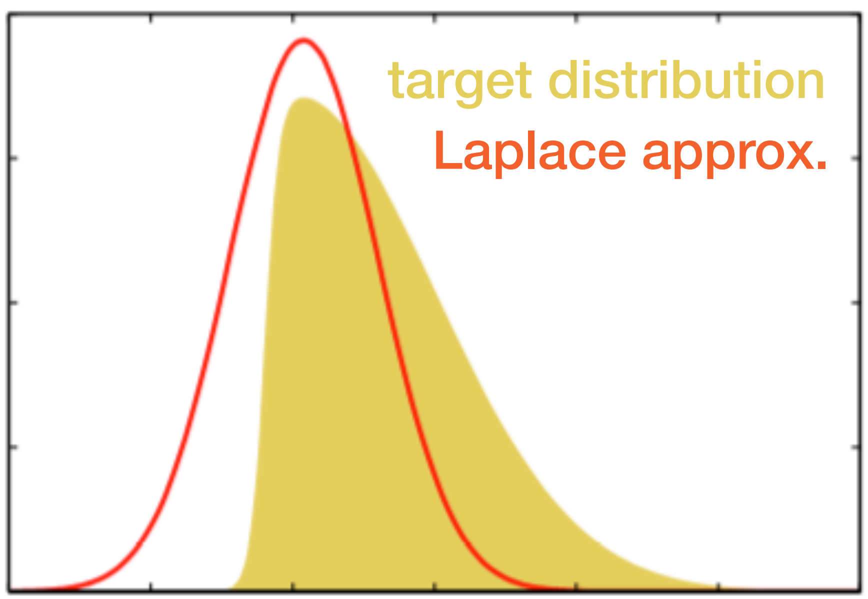


Problem: H is too large. Intractable to store and invert

Linearised Laplace Approximation

$$p(y^* \mid x^*, \mathcal{D}) = \mathcal{N}(y^*; \underbrace{f(x^*, \widehat{W})}_{\text{Preserves mean of pre-trained NN}}, J^\top(x^*)H^{-1}J(x^*) + \sigma^2 I)$$

Preserves mean of pre-trained NN

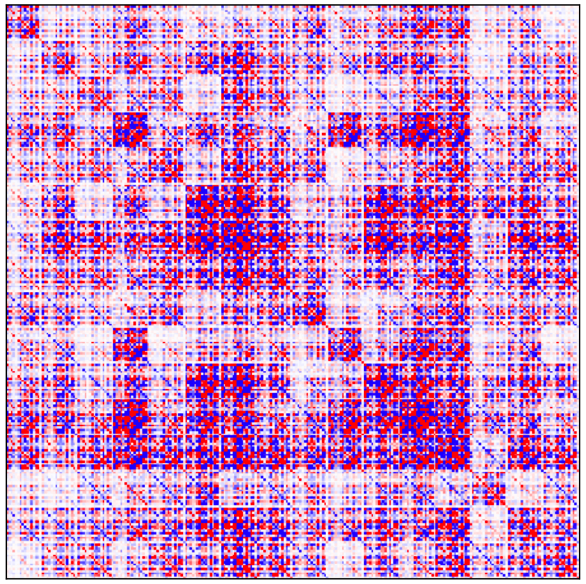


(Bishop 2006)

Exact for Gaussian Likelihoods (Regression)

Pretty good for Categorical Likelihoods (Classification)

$|W|^2$ Elements



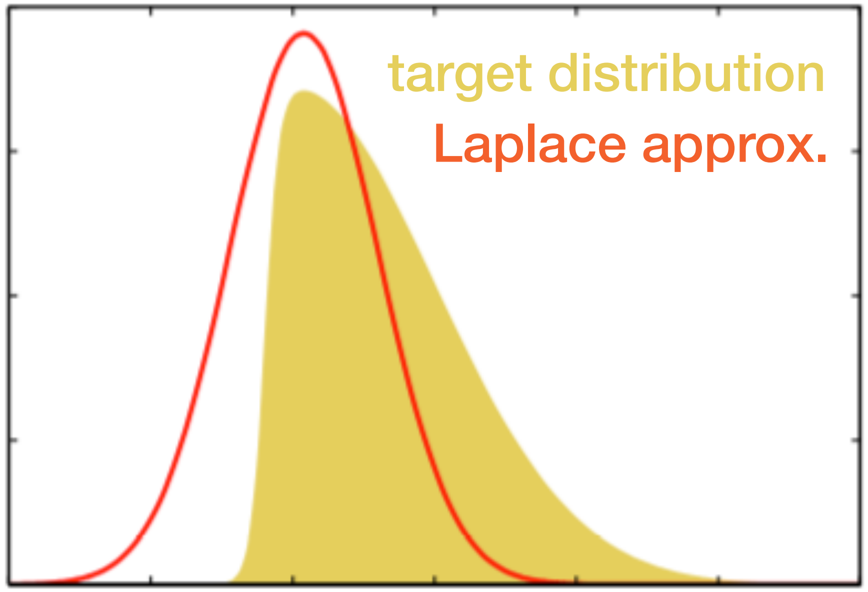
Problem: H **is too large. Intractable to store and invert**

➡ **Do Laplace only over a small subnetwork** $\mathbf{W}_S$

Linearised Laplace Approximation

$$p(y^* \mid x^*, \mathcal{D}) = \mathcal{N}(y^*; \underbrace{f(x^*, \widehat{W})}, J^\top(x^*) H^{-1} J(x^*) + \sigma^2 I)$$

Preserves mean of pre-trained NN

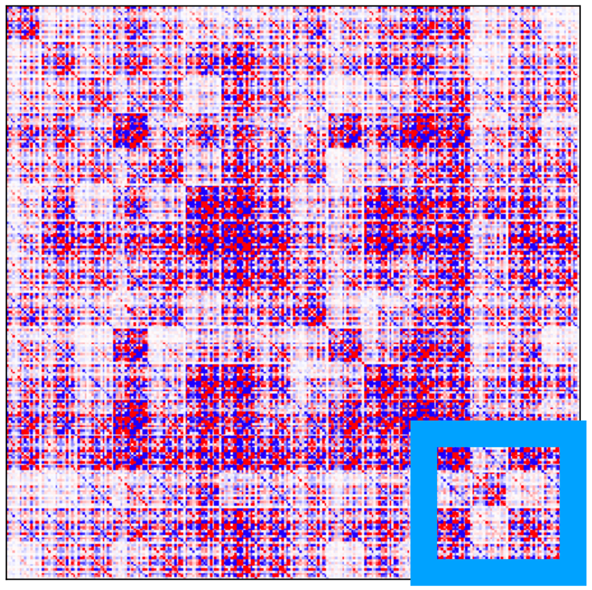


(Bishop 2006)

Exact for Gaussian Likelihoods (Regression)

Pretty good for Categorical Likelihoods (Classification)

$|\mathbf{w}_S'|^2$ Elements



Problem: H is too large. Intractable to store and invert

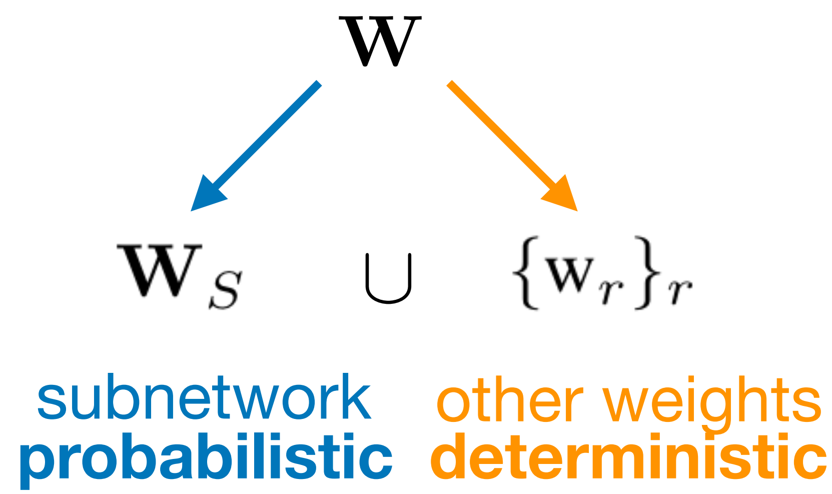
→ **Do Laplace only over a small subnetwork $\mathbf{w}_S$**

Subnetwork Inference

Proposed Posterior Approximation:

$$p(\mathbf{W}|\boldsymbol{y}, \boldsymbol{X}) \approx q(\mathbf{W}) = p(\mathbf{W}_S|\boldsymbol{y}, \boldsymbol{X}) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_r^*)$$

$$\approx q(\mathbf{W}_S) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_r^*)$$



Questions:

1. How do we choose and infer the subnetwork posterior $q(\mathbf{W}_S)$?

2. How do we set the fixed values $\mathrm{w}_r^* \in \mathbb{R}$ of all remaining weights $\{\mathrm{w}_r\}_r$?

3. How do we select the subnetwork $\mathbf{W}_S$?

4. How do we make predictions with the approximate posterior $q(\mathbf{W})$?

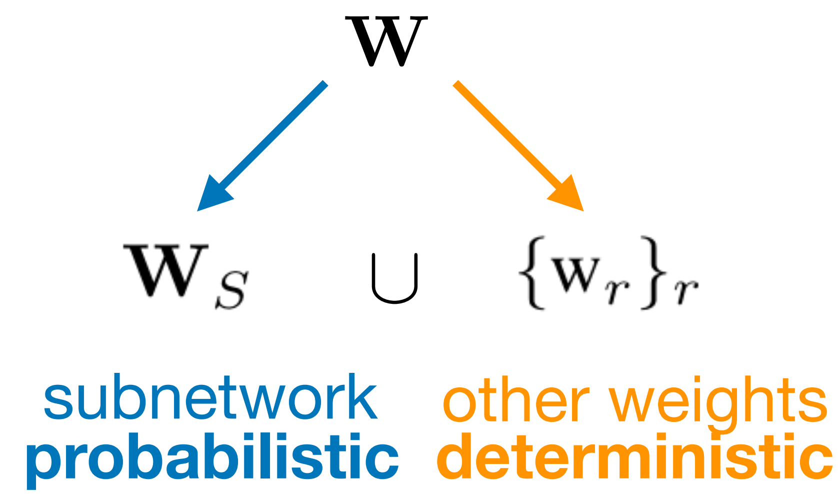
Subnetwork Inference

Proposed Posterior Approximation:

$$p(\mathbf{W}|\boldsymbol{y},\boldsymbol{X}) \approx q(\mathbf{W}) = p(\mathbf{W}_S|\boldsymbol{y},\boldsymbol{X}) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_r^*)$$

$$\approx q(\mathbf{W}_S) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_r^*)$$

$$= \mathcal{N}(\mathbf{W}_S; \boldsymbol{W}_{MAP}^S, H_S^{-1}) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_{MAP}^r)$$



Questions:

1. How do we choose and infer the subnetwork posterior $q(\mathbf{W}_S)$?

2. How do we set the fixed values $\mathrm{w}_r^* \in \mathbb{R}$ of all remaining weights $\{\mathrm{w}_r\}_r$?

3. How do we select the subnetwork $\mathbf{W}_S$?

4. How do we make predictions with the approximate posterior $q(\mathbf{W})$?

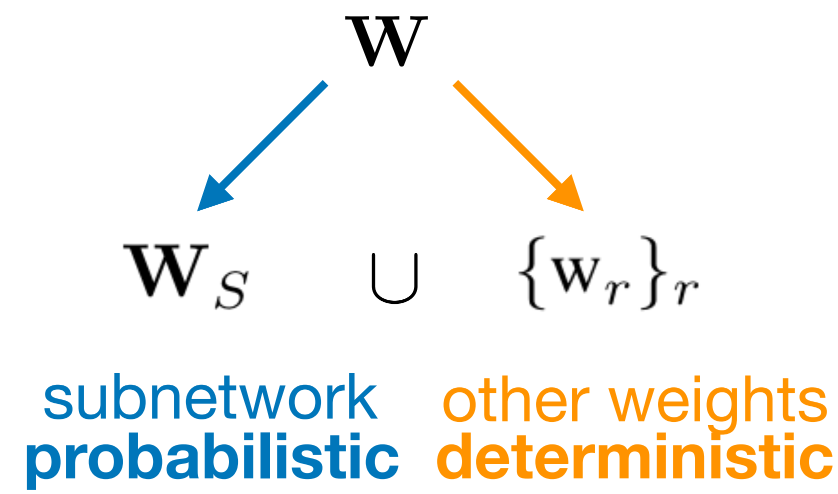
Subnetwork Inference

Proposed Posterior Approximation:

$$p(\mathbf{W}|\boldsymbol{y}, \boldsymbol{X}) \approx q(\mathbf{W}) = p(\mathbf{W}_S|\boldsymbol{y}, \boldsymbol{X}) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_r^*)$$

$$\approx q(\mathbf{W}_S) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_r^*)$$

$$= \mathcal{N}(\mathbf{W}_S; \boldsymbol{W}_{MAP}^S, H_S^{-1}) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_{MAP}^r)$$



Questions:

1. How do we choose and infer the subnetwork posterior $q(\mathbf{W}_S)$?
   —> full-covariance Gaussian via Linearised Laplace approximation
2. How do we set the fixed values $\mathrm{w}_r^* \in \mathbb{R}$ of all remaining weights $\{\mathrm{w}_r\}_r$?
3. How do we select the subnetwork $\mathbf{W}_S$?
4. How do we make predictions with the approximate posterior $q(\mathbf{W})$?

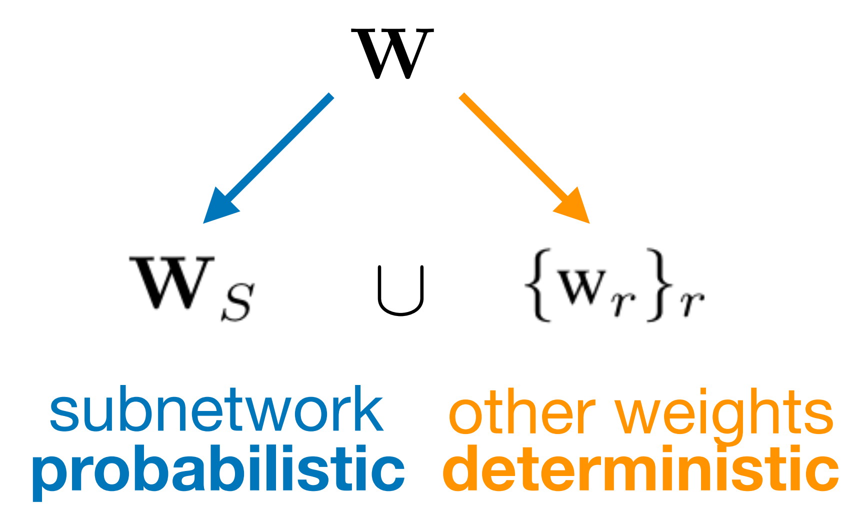
Subnetwork Inference

Proposed Posterior Approximation:

$$p(\mathbf{W}|\boldsymbol{y}, \boldsymbol{X}) \approx q(\mathbf{W}) = p(\mathbf{W}_S|\boldsymbol{y}, \boldsymbol{X}) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_r^*)$$

$$\approx q(\mathbf{W}_S) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_r^*)$$

$$= \mathcal{N}(\mathbf{W}_S; \boldsymbol{W}_{MAP}^S, H_S^{-1}) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_{MAP}^r)$$



Questions:

1. How do we choose and infer the subnetwork posterior $q(\mathbf{W}_S)$?
   —> full-covariance Gaussian via Linearised Laplace approximation

2. How do we set the fixed values $\mathrm{w}_r^* \in \mathbb{R}$ of all remaining weights $\{\mathrm{w}_r\}_r$?

3. How do we select the subnetwork $\mathbf{W}_S$?

4. How do we make predictions with the approximate posterior $q(\mathbf{W})$?

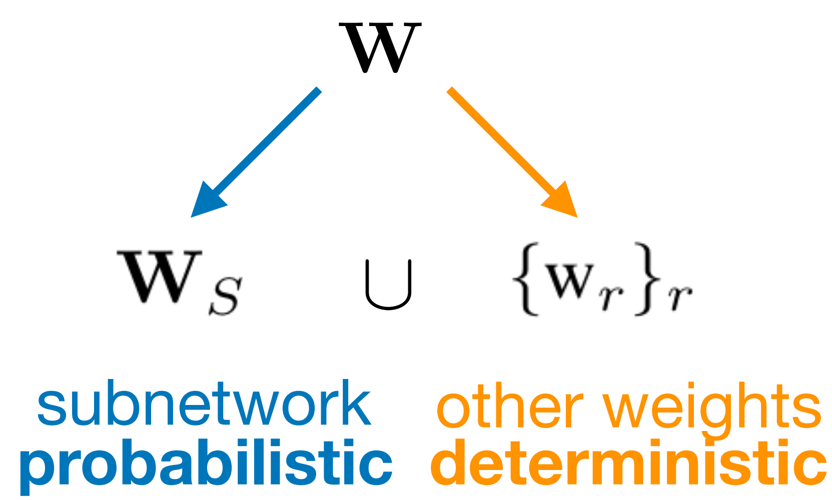
Subnetwork Inference

Proposed Posterior Approximation:

$$p(\mathbf{W}|\boldsymbol{y}, \boldsymbol{X}) \approx q(\mathbf{W}) = p(\mathbf{W}_S|\boldsymbol{y}, \boldsymbol{X}) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_r^*)$$

$$\approx q(\mathbf{W}_S) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_r^*)$$

$$= \mathcal{N}(\mathbf{W}_S; \boldsymbol{W}_{MAP}^S, H_S^{-1}) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_{MAP}^r)$$



Questions:

1. How do we choose and infer the subnetwork posterior $q(\mathbf{W}_S)$?
   —> full-covariance Gaussian via Linearised Laplace approximation
2. How do we set the fixed values $\mathrm{w}_r^* \in \mathbb{R}$ of all remaining weights $\{\mathrm{w}_r\}_r$?
   —> just leave them at their MAP estimates
3. How do we select the subnetwork $\mathbf{W}_S$?
4. How do we make predictions with the approximate posterior $q(\mathbf{W})$?

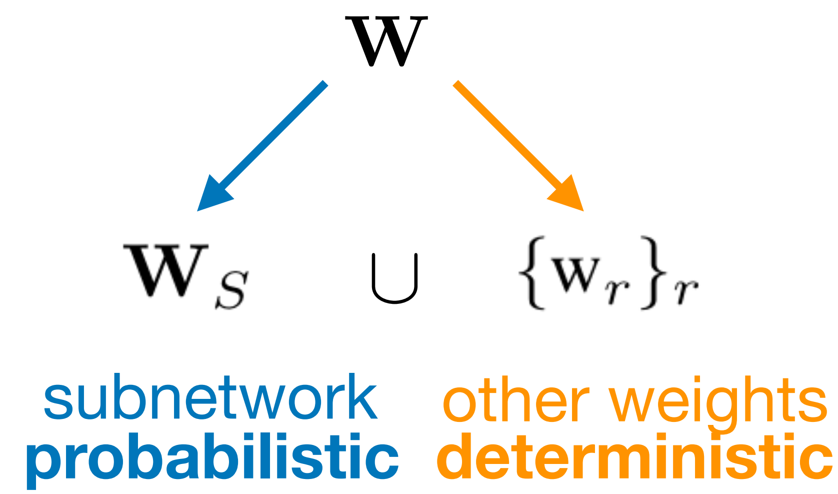
Subnetwork Inference

Proposed Posterior Approximation:

$$p(\mathbf{W}|\boldsymbol{y}, \boldsymbol{X}) \approx q(\mathbf{W}) = p(\mathbf{W}_S|\boldsymbol{y}, \boldsymbol{X}) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_r^*)$$

$$\approx q(\mathbf{W}_S) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_r^*)$$

$$= \mathcal{N}(\mathbf{W}_S; \boldsymbol{W}_{MAP}^S, H_S^{-1}) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_{MAP}^r)$$



Questions:

1. How do we choose and infer the subnetwork posterior $q(\mathbf{W}_S)$?
   —> full-covariance Gaussian via Linearised Laplace approximation
2. How do we set the fixed values $\mathrm{w}_r^* \in \mathbb{R}$ of all remaining weights $\{\mathrm{w}_r\}_r$?
   —> just leave them at their MAP estimates
3. How do we select the subnetwork $\mathbf{W}_S$?
4. How do we make predictions with the approximate posterior $q(\mathbf{W})$?

Subnetwork Selection

Subnetwork Selection

We would like the distribution over functions induced by the subnetwork to match the one induced by the full network.

$$\sup_{n \in \mathcal{N}, X^* \in \mathcal{X}^n} D_{KL}(p_S(y^* \mid X^*, \mathcal{D}) \,||\, p(y^* \mid X^*, \mathcal{D}))$$

Subnetwork Selection

We would like the distribution over functions induced by the subnetwork to match the one induced by the full network.

$$\sup_{n \in \mathcal{N}, X^* \in \mathcal{X}^n} D_{KL}(p_S(y^* \mid X^*, \mathcal{D}) \,||\, p(y^* \mid X^*, \mathcal{D}))$$

Intractable! ✗

Subnetwork Selection

We would like the distribution over functions induced by the subnetwork to match the one induced by the full network.

$$\sup_{n \in \mathcal{N}, X^* \in \mathcal{X}^n} D_{KL}(p_S(y^* \mid X^*, \mathcal{D}) \,||\, p(y^* \mid X^*, \mathcal{D}))$$ **Intractable!** ✘

What about similarity in weight space?

$$p(W \mid \mathcal{D}) \longleftrightarrow p(W_s \mid \mathcal{D}) \prod_r \delta(w_r - \hat{w}_r)$$

Subnetwork Selection

We would like the distribution over functions induced by the subnetwork to match the one induced by the full network.

$$\sup_{n \in \mathcal{N}, X^* \in \mathcal{X}^n} D_{KL}(p_S(y^* | X^*, \mathcal{D}) \, || \, p(y^* | X^*, \mathcal{D}))$$

Intractable! ✘

What about similarity in weight space?

$$p(W | \mathcal{D}) \longleftrightarrow p(W_s | \mathcal{D}) \prod_r \delta(w_r - \hat{w}_r)$$

Most distances are undefined for distributions with **disjoint support** ✘

Subnetwork Selection

We would like the distribution over functions induced by the subnetwork to match the one induced by the full network.

$$\sup_{n \in \mathcal{N}, X^* \in \mathcal{X}^n} D_{KL}(p_S(y^* \,|\, X^*, \mathcal{D}) \,||\, p(y^* \,|\, X^*, \mathcal{D}))$$

Intractable! ✗

What about similarity in weight space?

$$p(W \,|\, \mathcal{D}) \longleftrightarrow p(W_s \,|\, \mathcal{D}) \prod_r \delta(w_r - \hat{w}_r)$$

Most distances are undefined for distributions with **disjoint support** ✗

➡ The **Wasserstein distance** is well defined in this setting. ✓

Subnetwork Selection

Subnetwork Selection

Goal: Find subnetwork whose posterior is **closest to the full network posterior**

Subnetwork Selection

Goal: Find subnetwork whose posterior is **closest to the full network posterior**

$$\min \mathrm{Wass}[\text{ full posterior } \| \text{ subnet posterior }]$$

Subnetwork Selection

Goal: Find subnetwork whose posterior is **closest to the full network posterior**

$$\min \mathrm{Wass}[\text{ full posterior } \| \text{ subnet posterior }]$$

$$= \min \mathrm{Wass}[\ p(\mathbf{W}|\boldsymbol{y}, \boldsymbol{X}) \ \|\ q(\mathbf{W})\ ]$$

Subnetwork Selection

Goal: Find subnetwork whose posterior is **closest to the full network posterior**

$$\min \mathrm{Wass}[\text{ full posterior } \| \text{ subnet posterior }]$$

$$= \min \mathrm{Wass}[\ p(\mathbf{W}|\boldsymbol{y}, \boldsymbol{X}) \ \|\ q(\mathbf{W})\ ]$$

$$\approx \min \mathrm{Wass}[\ \mathcal{N}\left(\mathbf{W}; \boldsymbol{W}_{MAP}, H^{-1}\right)\ \| \qquad]$$

Subnetwork Selection

Goal: Find subnetwork whose posterior is **closest to the full network posterior**

$$\min \mathrm{Wass}[\text{ full posterior } \| \text{ subnet posterior }]$$

$$= \min \mathrm{Wass}[\ p(\mathbf{W}|\boldsymbol{y}, \boldsymbol{X}) \ \| \ q(\mathbf{W})\ ]$$

$$\approx \min \mathrm{Wass}[\ \mathcal{N}\left(\mathbf{W}; \boldsymbol{W}_{MAP}, H^{-1}\right) \ \| \ \mathcal{N}\left(\mathbf{W}_S; \boldsymbol{W}_{MAP}^S, H_S^{-1}\right) \qquad]$$

Subnetwork Selection

Goal: Find subnetwork whose posterior is **closest to the full network posterior**

$$\min \mathrm{Wass}[\text{ full posterior } \| \text{ subnet posterior }]$$

$$= \min \mathrm{Wass}[\ p(\mathbf{W}|\boldsymbol{y}, \boldsymbol{X}) \ \|\ q(\mathbf{W})\ ]$$

$$\approx \min \mathrm{Wass}[\ \mathcal{N}\left(\mathbf{W}; \boldsymbol{W}_{MAP}, H^{-1}\right) \ \|\ \mathcal{N}\left(\mathbf{W}_S; \boldsymbol{W}^S_{MAP}, H_S^{-1}\right) \prod_r \delta(\mathrm{w}_r - \mathrm{w}^r_{MAP})\ ]$$

Subnetwork Selection

Goal: Find subnetwork whose posterior is **closest to the full network posterior**

$$\min \text{Wass}[\text{ full posterior } \| \text{ subnet posterior }]$$

$$= \min \text{Wass}[\ p(\mathbf{W}|\boldsymbol{y}, \boldsymbol{X}) \ \| \ q(\mathbf{W})\ ]$$

$$\approx \min \text{Wass}[\ \mathcal{N}\left(\mathbf{W}; \boldsymbol{W}_{MAP}, H^{-1}\right) \ \| \ \mathcal{N}\left(\mathbf{W}_S; \boldsymbol{W}^S_{MAP}, H_S^{-1}\right) \prod_r \delta(\mathrm{w}_r - \mathrm{w}^r_{MAP})\ ]$$

Intractable, as this depends on **all** entries of the full network Hessian H. ✘

Subnetwork Selection

Goal: Find subnetwork whose posterior is **closest to the full network posterior**

$$\min \mathrm{Wass}[\text{ full posterior } \| \text{ subnet posterior }]$$

$$= \min \mathrm{Wass}[\ p(\mathbf{W}|\boldsymbol{y}, \boldsymbol{X}) \ \|\ q(\mathbf{W})\ ]$$

$$\approx \min \mathrm{Wass}[\ \mathcal{N}\left(\mathbf{W}; \boldsymbol{W}_{MAP}, H^{-1}\right)\ \|\ \mathcal{N}\left(\mathbf{W}_S; \boldsymbol{W}^S_{MAP}, H_S^{-1}\right) \prod_r \delta(\mathrm{w}_r - \mathrm{w}^r_{MAP})\ ]$$

Intractable, as this depends on **all** entries of the full network Hessian H. ✗

➔ Assume that posterior is **factorized** for dependence only on **diagonal** entries. ✓

Subnetwork Selection

Goal: Find subnetwork whose posterior is **closest to the full network posterior**

$$\min \mathrm{Wass}[\ \text{full posterior} \ \|\ \text{subnet posterior}\ ]$$

$$= \min \mathrm{Wass}[\ p(\mathbf{W}|\boldsymbol{y}, \boldsymbol{X}) \ \|\ q(\mathbf{W})\ ]$$

$$\approx \min \mathrm{Wass}[\ \mathcal{N}\left(\mathbf{W}; \boldsymbol{W}_{MAP}, H^{-1}\right) \ \|\ \mathcal{N}\left(\mathbf{W}_S; \boldsymbol{W}_{MAP}^{S}, H_S^{-1}\right) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_{MAP}^{r})\ ]$$

Intractable, as this depends on **all** entries of the full network Hessian H. ✗

➡ Assume that posterior is **factorized** for dependence only on **diagonal** entries. ✓

diag. assumption for **subnetwork selection** $\gg$ diag. assumption for **inference**

Subnetwork Selection

Goal: Find subnetwork whose posterior is **closest to the full network posterior**

$$\min \text{Wass}[\text{ full posterior } \| \text{ subnet posterior }]$$

$$= \min \text{Wass}[\ p(\mathbf{W}|\boldsymbol{y}, \boldsymbol{X}) \ \| \ q(\mathbf{W})\ ]$$

$$\approx \min \text{Wass}[\ \mathcal{N}\left(\mathbf{W}; \boldsymbol{W}_{MAP}, H^{-1}\right) \ \| \ \mathcal{N}\left(\mathbf{W}_S; \boldsymbol{W}_{MAP}^S, H_S^{-1}\right) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_{MAP}^r)\ ]$$

Intractable, as this depends on **all** entries of the full network Hessian H. ✗

➡ Assume that posterior is **factorized** for dependence only on **diagonal** entries. ✓

diag. assumption for **subnetwork selection** $\gg$ diag. assumption for **inference**

⬇

Wasserstein subnetwork selection

Subnetwork Selection

Goal: Find subnetwork whose posterior is **closest to the full network posterior**

$$\min \text{Wass}[\text{ full posterior } \| \text{ subnet posterior }]$$

$$= \min \text{Wass}[\ p(\mathbf{W}|\boldsymbol{y}, \boldsymbol{X}) \ \|\ q(\mathbf{W})\ ]$$

$$\approx \min \text{Wass}[\ \mathcal{N}\left(\mathbf{W}; \boldsymbol{W}_{MAP}, H^{-1}\right) \ \|\ \mathcal{N}\left(\mathbf{W}_S; \boldsymbol{W}^S_{MAP}, H_S^{-1}\right) \prod_r \delta(\mathrm{w}_r - \mathrm{w}^r_{MAP})\ ]$$

Intractable, as this depends on **all** entries of the full network Hessian H. ✗

➡ Assume that posterior is **factorized** for dependence only on **diagonal** entries. ✓

diag. assumption for **subnetwork selection** $\gg$ diag. assumption for **inference**

⬇

Wasserstein subnetwork selection

1) Estimate a **factorized Gaussian** posterior over all weights

Subnetwork Selection

Goal: Find subnetwork whose posterior is **closest to the full network posterior**

$$\min \mathrm{Wass}[\text{ full posterior } \| \text{ subnet posterior }]$$

$$= \min \mathrm{Wass}[\ p(\mathbf{W}|\boldsymbol{y}, \boldsymbol{X}) \ \|\ q(\mathbf{W})\ ]$$

$$\approx \min \mathrm{Wass}[\ \mathcal{N}\left(\mathbf{W}; \boldsymbol{W}_{MAP}, H^{-1}\right) \ \|\ \mathcal{N}\left(\mathbf{W}_S; \boldsymbol{W}_{MAP}^S, H_S^{-1}\right) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_{MAP}^r)\ ]$$

Intractable, as this depends on **all** entries of the full network Hessian H. ✗

➡ Assume that posterior is **factorized** for dependence only on **diagonal** entries. ✓

diag. assumption for **subnetwork selection** $\gg$ diag. assumption for **inference**

⬇

Wasserstein subnetwork selection

1) Estimate a **factorized Gaussian** posterior over all weights
2) Subnetwork = weights with **largest marginal variances**

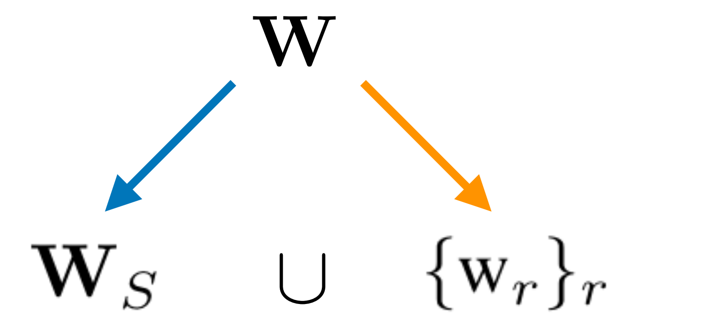
Subnetwork Inference

Proposed Posterior Approximation:

$$p(\mathbf{W}|\boldsymbol{y}, \boldsymbol{X}) \approx q(\mathbf{W}) = p(\mathbf{W}_S|\boldsymbol{y}, \boldsymbol{X}) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_r^*)$$

$$\approx q(\mathbf{W}_S) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_r^*)$$

$$= \mathcal{N}(\mathbf{W}_S; \boldsymbol{W}_{MAP}^S, H_S^{-1}) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_{MAP}^r)$$



Questions:

1. How do we choose and infer the subnetwork posterior $q(\mathbf{W}_S)$?
   —> full-covariance Gaussian via Laplace approximation
2. How do we set the fixed values $\mathrm{w}_r^* \in \mathbb{R}$ of all remaining weights $\{\mathrm{w}_r\}_r$?
   —> just leave them at their MAP estimates
3. How do we select the subnetwork $\mathbf{W}_S$?
4. How do we make predictions with the approximate posterior $q(\mathbf{W})$?

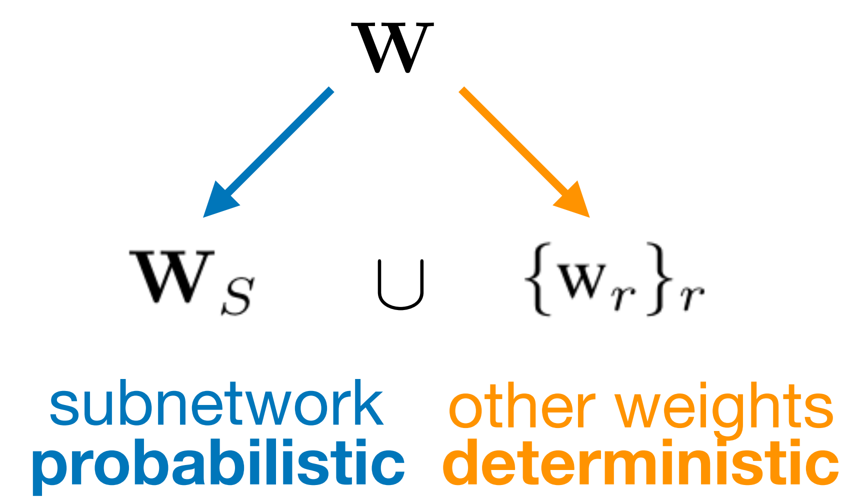
Subnetwork Inference

Proposed Posterior Approximation:

$$p(\mathbf{W}|\boldsymbol{y},\boldsymbol{X}) \approx q(\mathbf{W}) = p(\mathbf{W}_S|\boldsymbol{y},\boldsymbol{X}) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_r^*)$$

$$\approx q(\mathbf{W}_S) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_r^*)$$

$$= \mathcal{N}(\mathbf{W}_S; \boldsymbol{W}_{MAP}^S, H_S^{-1}) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_{MAP}^r)$$



Questions:

1. How do we choose and infer the subnetwork posterior $q(\mathbf{W}_S)$?
   —> full-covariance Gaussian via Laplace approximation
2. How do we set the fixed values $\mathrm{w}_r^* \in \mathbb{R}$ of all remaining weights $\{\mathrm{w}_r\}_r$?
   —> just leave them at their MAP estimates
3. How do we select the subnetwork $\mathbf{W}_S$?
   —> min. Wass. distance between subnetwork posterior & full posterior
4. How do we make predictions with the approximate posterior $q(\mathbf{W})$?

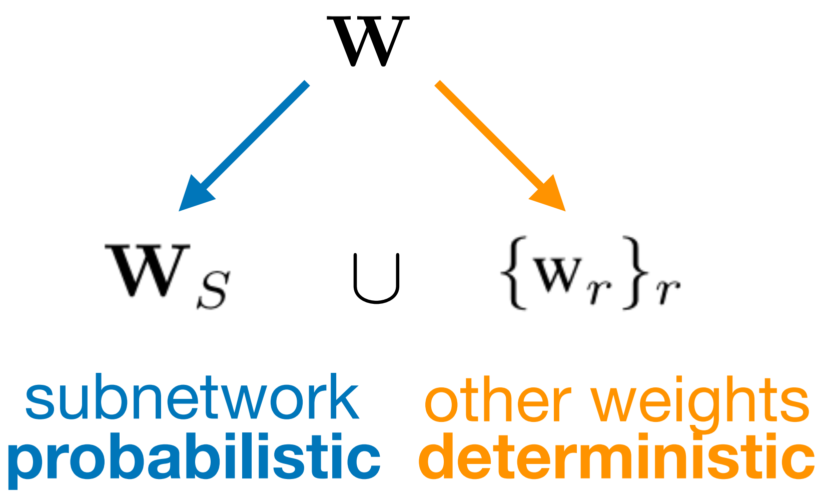
Subnetwork Inference

Proposed Posterior Approximation:

$$p(\mathbf{W}|\boldsymbol{y}, \boldsymbol{X}) \approx q(\mathbf{W}) = p(\mathbf{W}_S|\boldsymbol{y}, \boldsymbol{X}) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_r^*)$$

$$\approx q(\mathbf{W}_S) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_r^*)$$

$$= \mathcal{N}(\mathbf{W}_S; \boldsymbol{W}_{MAP}^S, H_S^{-1}) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_{MAP}^r)$$



Questions:

1. How do we choose and infer the subnetwork posterior $q(\mathbf{W}_S)$?
   —> full-covariance Gaussian via Laplace approximation
2. How do we set the fixed values $\mathrm{w}_r^* \in \mathbb{R}$ of all remaining weights $\{\mathrm{w}_r\}_r$?
   —> just leave them at their MAP estimates
3. How do we select the subnetwork $\mathbf{W}_S$?
   —> min. Wass. distance between subnetwork posterior & full posterior
4. How do we make predictions with the approximate posterior $q(\mathbf{W})$?

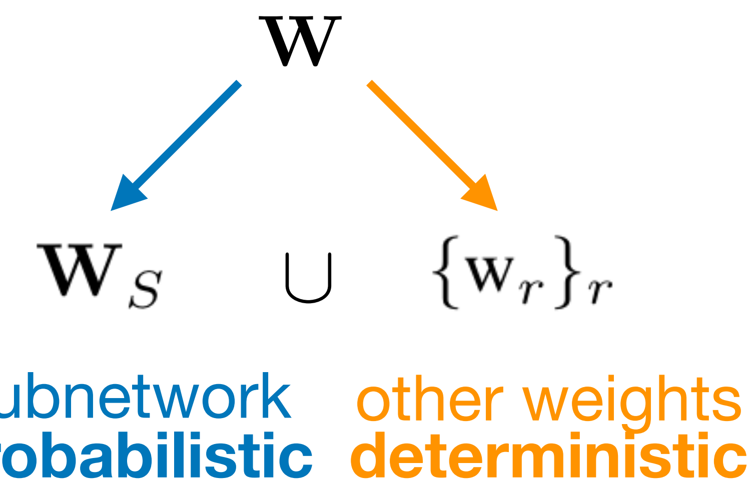
Subnetwork Inference

Proposed Posterior Approximation:

$$p(\mathbf{W}|\boldsymbol{y},\boldsymbol{X}) \approx q(\mathbf{W}) = p(\mathbf{W}_S|\boldsymbol{y},\boldsymbol{X}) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_r^*)$$

$$\approx q(\mathbf{W}_S) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_r^*)$$

$$= \mathcal{N}(\mathbf{W}_S; \boldsymbol{W}_{MAP}^S, H_S^{-1}) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_{MAP}^r)$$



Questions:

1. How do we choose and infer the subnetwork posterior $q(\mathbf{W}_S)$?
   —> full-covariance Gaussian via Laplace approximation
2. How do we set the fixed values $\mathrm{w}_r^* \in \mathbb{R}$ of all remaining weights $\{\mathrm{w}_r\}_r$?
   —> just leave them at their MAP estimates
3. How do we select the subnetwork $\mathbf{W}_S$?
   —> min. Wass. distance between subnetwork posterior & full posterior
4. How do we make predictions with the approximate posterior $q(\mathbf{W})$?
   —> use all weights: integrate out subnetwork & keep others fixed

Making Predictions

	Full Laplace	Subnetwork Laplace
Predictive $p(\mathbf{y}^*\|\mathbf{x}^*, \mathcal{D})$ **for Regression**		
Predictive $p(\mathbf{y}^*\|\mathbf{x}^*, \mathcal{D})$ **for Classification**		

Making Predictions

	Full Laplace	Subnetwork Laplace
Predictive $p(\mathbf{y}^*\|\mathbf{x}^*, \mathcal{D})$ **for Regression**	$\mathcal{N}(\mathbf{y}^*; \boldsymbol{f}(\mathbf{x}^*, \widehat{\mathbf{w}}), \Sigma(\mathbf{x}^*) + \sigma^2 I)$	
Predictive $p(\mathbf{y}^*\|\mathbf{x}^*, \mathcal{D})$ **for Classification**		

Making Predictions

	Full Laplace	Subnetwork Laplace
Predictive $p(\mathbf{y}^*\|\mathbf{x}^*, \mathcal{D})$ **for Regression**	$\mathcal{N}(\mathbf{y}^*; \boldsymbol{f}(\mathbf{x}^*, \widehat{\mathbf{w}}), \Sigma(\mathbf{x}^*)+\sigma^2 I)$	$\mathcal{N}(\mathbf{y}^*; \boldsymbol{f}(\mathbf{x}^*, \widehat{\mathbf{w}}), \Sigma_S(\mathbf{x}^*)+\sigma^2 I)$
Predictive $p(\mathbf{y}^*\|\mathbf{x}^*, \mathcal{D})$ **for Classification**		

Making Predictions

	Full Laplace	Subnetwork Laplace
Predictive $p(\mathbf{y}^*\|\mathbf{x}^*, \mathcal{D})$ **for Regression**	$\mathcal{N}(\mathbf{y}^*; \boldsymbol{f}(\mathbf{x}^*, \widehat{\mathbf{w}}), \Sigma(\mathbf{x}^*)+\sigma^2 I)$	$\mathcal{N}(\mathbf{y}^*; \boldsymbol{f}(\mathbf{x}^*, \widehat{\mathbf{w}}), \Sigma_S(\mathbf{x}^*)+\sigma^2 I)$
Predictive $p(\mathbf{y}^*\|\mathbf{x}^*, \mathcal{D})$ **for Classification**		

Predictive Cov. Matrix $\Sigma(\mathbf{x}^*) = \widehat{\boldsymbol{J}}(\mathbf{x}^*)^T \widetilde{H}^{-1} \widehat{\boldsymbol{J}}(\mathbf{x}^*)$

Making Predictions

	Full Laplace	Subnetwork Laplace
Predictive $p(\mathbf{y}^*\|\mathbf{x}^*, \mathcal{D})$ **for Regression**	$\mathcal{N}(\mathbf{y}^*; \boldsymbol{f}(\mathbf{x}^*, \widehat{\mathbf{w}}), \Sigma(\mathbf{x}^*)+\sigma^2 I)$	$\mathcal{N}(\mathbf{y}^*; \boldsymbol{f}(\mathbf{x}^*, \widehat{\mathbf{w}}), \Sigma_S(\mathbf{x}^*)+\sigma^2 I)$
Predictive $p(\mathbf{y}^*\|\mathbf{x}^*, \mathcal{D})$ **for Classification**		

Predictive Cov. Matrix $\Sigma(\mathbf{x}^*) = \widehat{\boldsymbol{J}}(\mathbf{x}^*)^T \widetilde{H}^{-1} \widehat{\boldsymbol{J}}(\mathbf{x}^*)$ $\Sigma_S(\mathbf{x}^*) = \widehat{\boldsymbol{J}}_S(\mathbf{x}^*)^T \widetilde{H}_S^{-1} \widehat{\boldsymbol{J}}_S(\mathbf{x}^*)$

Making Predictions

	Full Laplace	Subnetwork Laplace
Predictive $p(\mathbf{y}^*\|\mathbf{x}^*, \mathcal{D})$ **for Regression**	$\mathcal{N}(\mathbf{y}^*; \boldsymbol{f}(\mathbf{x}^*, \widehat{\mathbf{w}}), \Sigma(\mathbf{x}^*)+\sigma^2 I)$	$\mathcal{N}(\mathbf{y}^*; \boldsymbol{f}(\mathbf{x}^*, \widehat{\mathbf{w}}), \Sigma_S(\mathbf{x}^*)+\sigma^2 I)$
Predictive $p(\mathbf{y}^*\|\mathbf{x}^*, \mathcal{D})$ **for Classification**	$\mathrm{softmax}\left(\frac{\boldsymbol{f}(\mathbf{x}^*, \widehat{\mathbf{w}})}{\sqrt{1+\frac{\pi}{8}\mathrm{diag}(\Sigma(\mathbf{x}^*))}}\right)$	

Predictive Cov. Matrix

$$\Sigma(\mathbf{x}^*) = \widehat{\boldsymbol{J}}(\mathbf{x}^*)^T \widetilde{H}^{-1} \widehat{\boldsymbol{J}}(\mathbf{x}^*)$$

$$\Sigma_S(\mathbf{x}^*) = \widehat{\boldsymbol{J}}_S(\mathbf{x}^*)^T \widetilde{H}_S^{-1} \widehat{\boldsymbol{J}}_S(\mathbf{x}^*)$$

Making Predictions

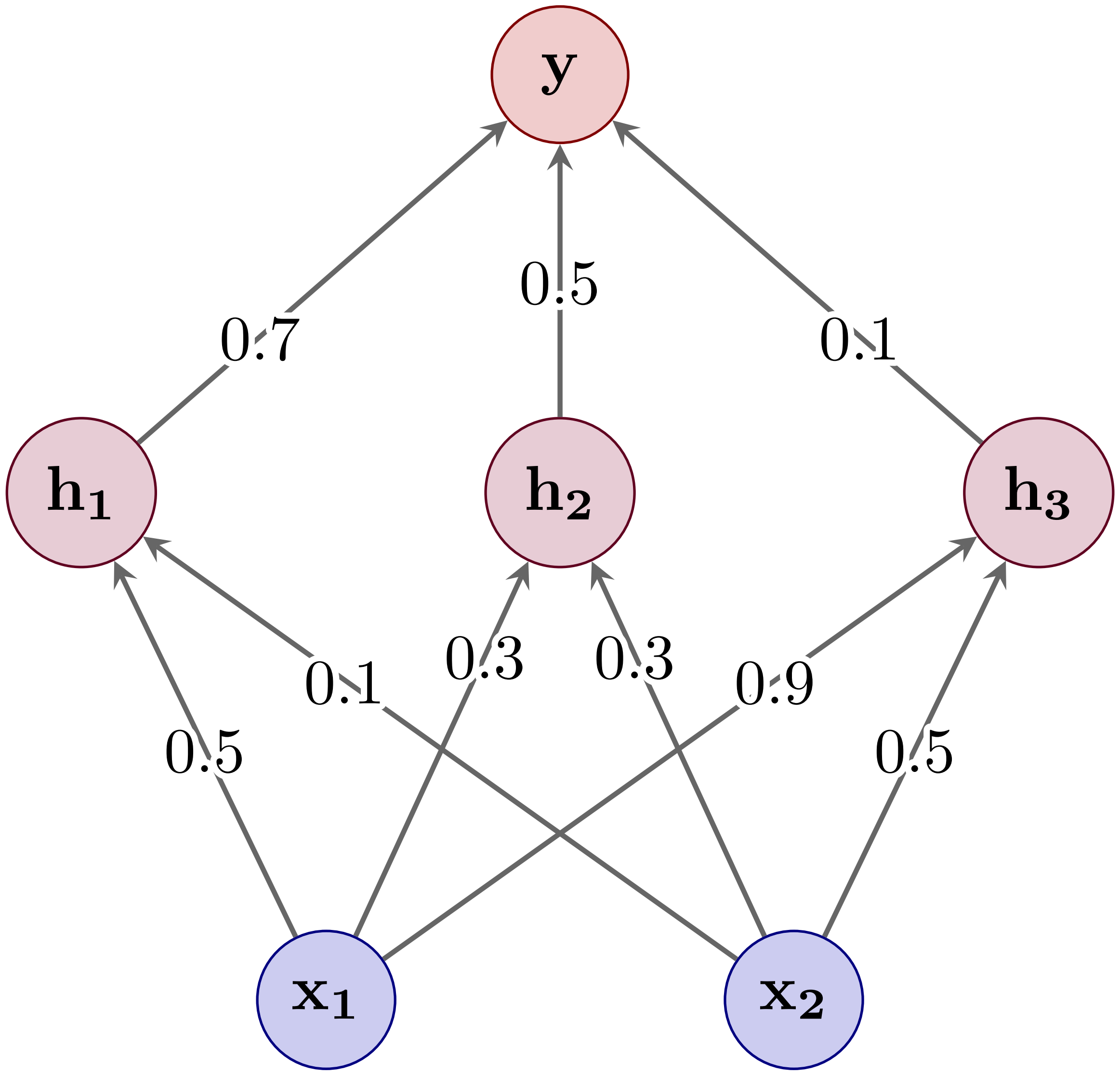
	Full Laplace	Subnetwork Laplace
Predictive $p(\mathbf{y}^*\|\mathbf{x}^*, \mathcal{D})$ **for Regression**	$\mathcal{N}(\mathbf{y}^*; \boldsymbol{f}(\mathbf{x}^*, \widehat{\mathbf{w}}), \Sigma(\mathbf{x}^*)+\sigma^2 I)$	$\mathcal{N}(\mathbf{y}^*; \boldsymbol{f}(\mathbf{x}^*, \widehat{\mathbf{w}}), \Sigma_S(\mathbf{x}^*)+\sigma^2 I)$
Predictive $p(\mathbf{y}^*\|\mathbf{x}^*, \mathcal{D})$ **for Classification**	$\text{softmax}\left(\frac{\boldsymbol{f}(\mathbf{x}^*, \widehat{\mathbf{w}})}{\sqrt{1+\frac{\pi}{8}\text{diag}(\Sigma(\mathbf{x}^*))}}\right)$	$\text{softmax}\left(\frac{\boldsymbol{f}(\mathbf{x}^*, \widehat{\mathbf{w}})}{\sqrt{1+\frac{\pi}{8}\text{diag}(\Sigma_S(\mathbf{x}^*))}}\right)$

Predictive Cov. Matrix $\quad \Sigma(\mathbf{x}^*) = \widehat{\boldsymbol{J}}(\mathbf{x}^*)^T \widetilde{H}^{-1} \widehat{\boldsymbol{J}}(\mathbf{x}^*) \quad \Sigma_S(\mathbf{x}^*) = \widehat{\boldsymbol{J}}_S(\mathbf{x}^*)^T \widetilde{H}_S^{-1} \widehat{\boldsymbol{J}}_S(\mathbf{x}^*)$

Subnetwork Inference

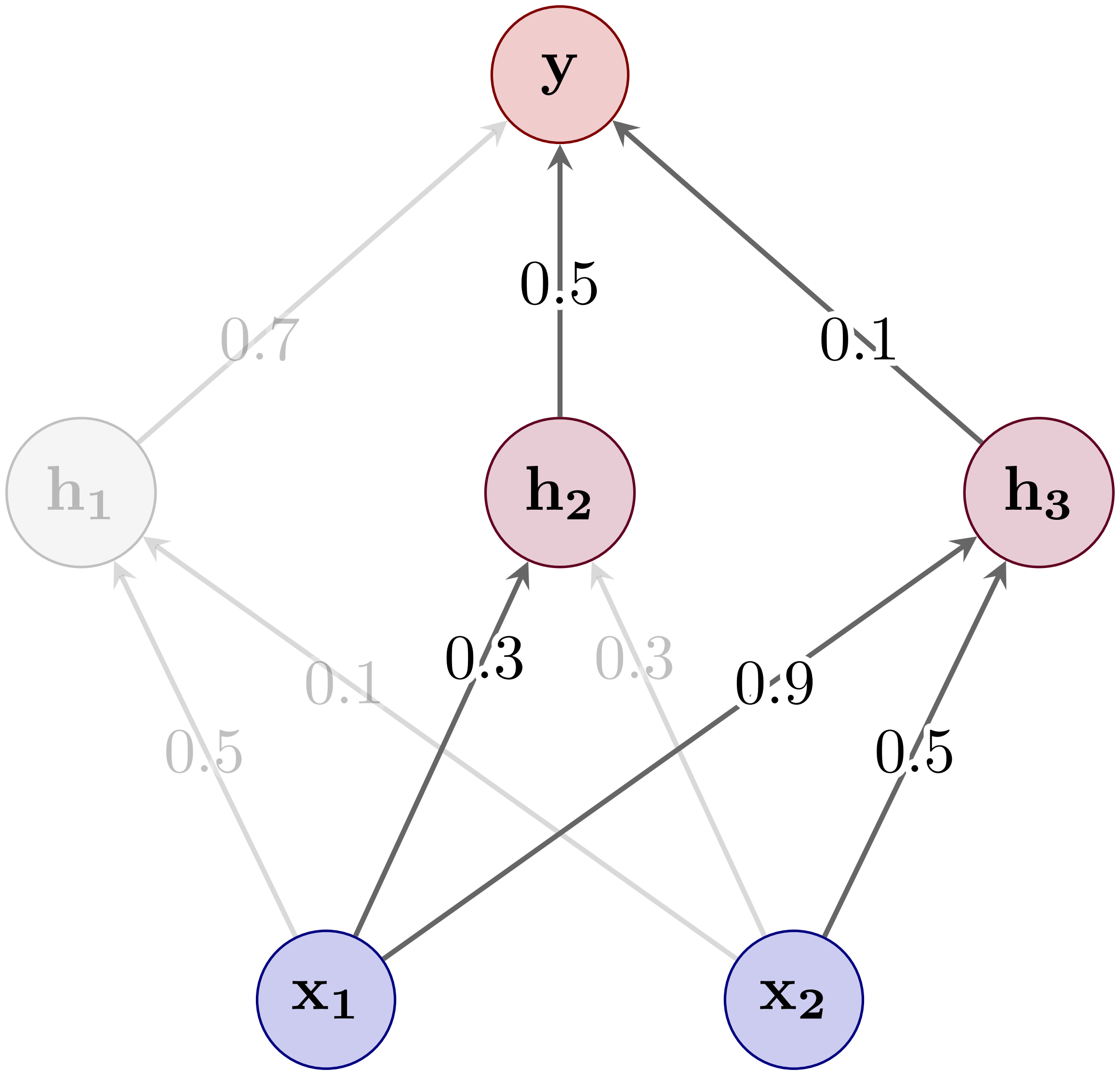
Subnetwork Inference

1 MAP Estimation



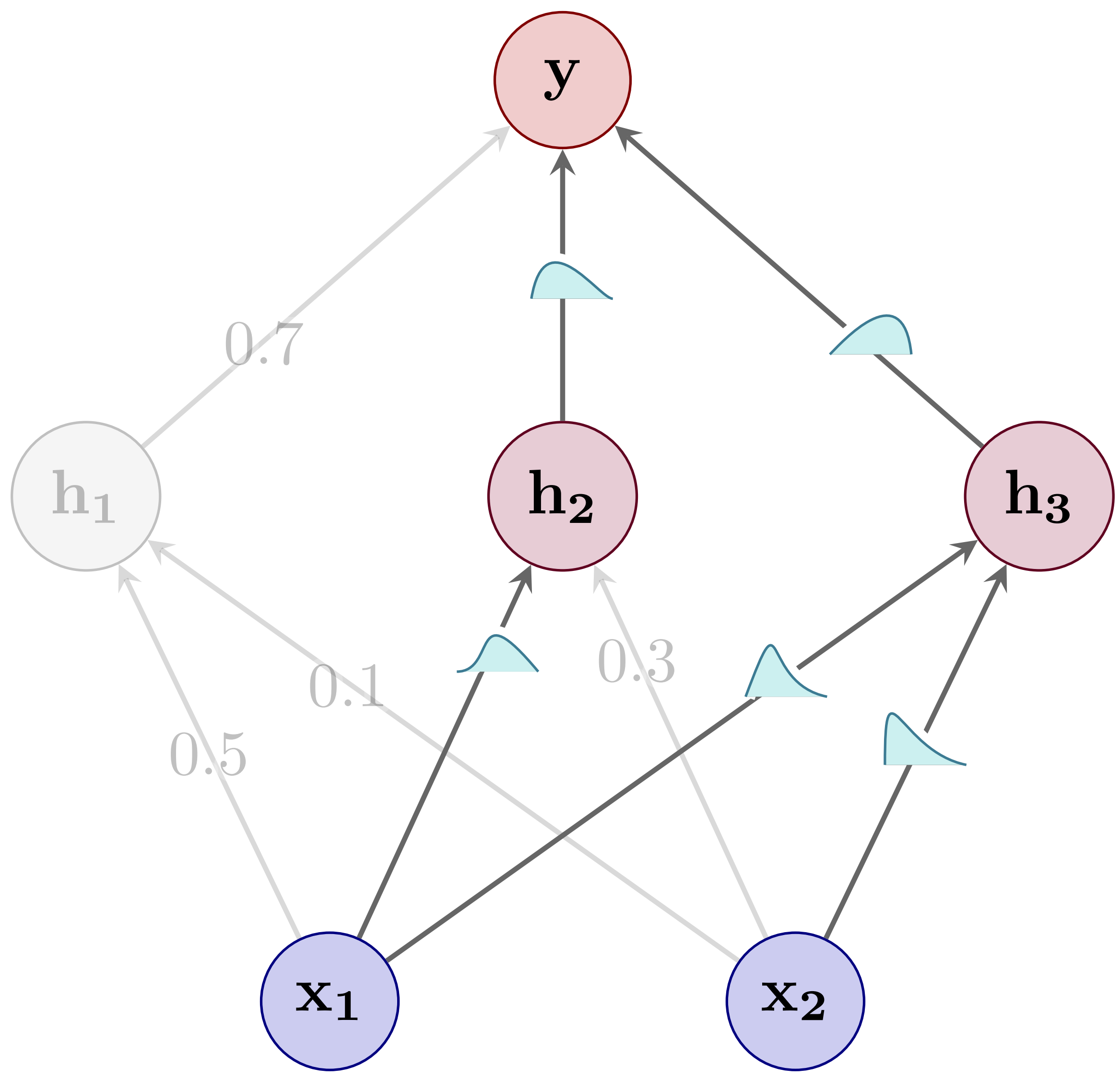
Subnetwork Inference

1. MAP Estimation
2. Subnet Selection



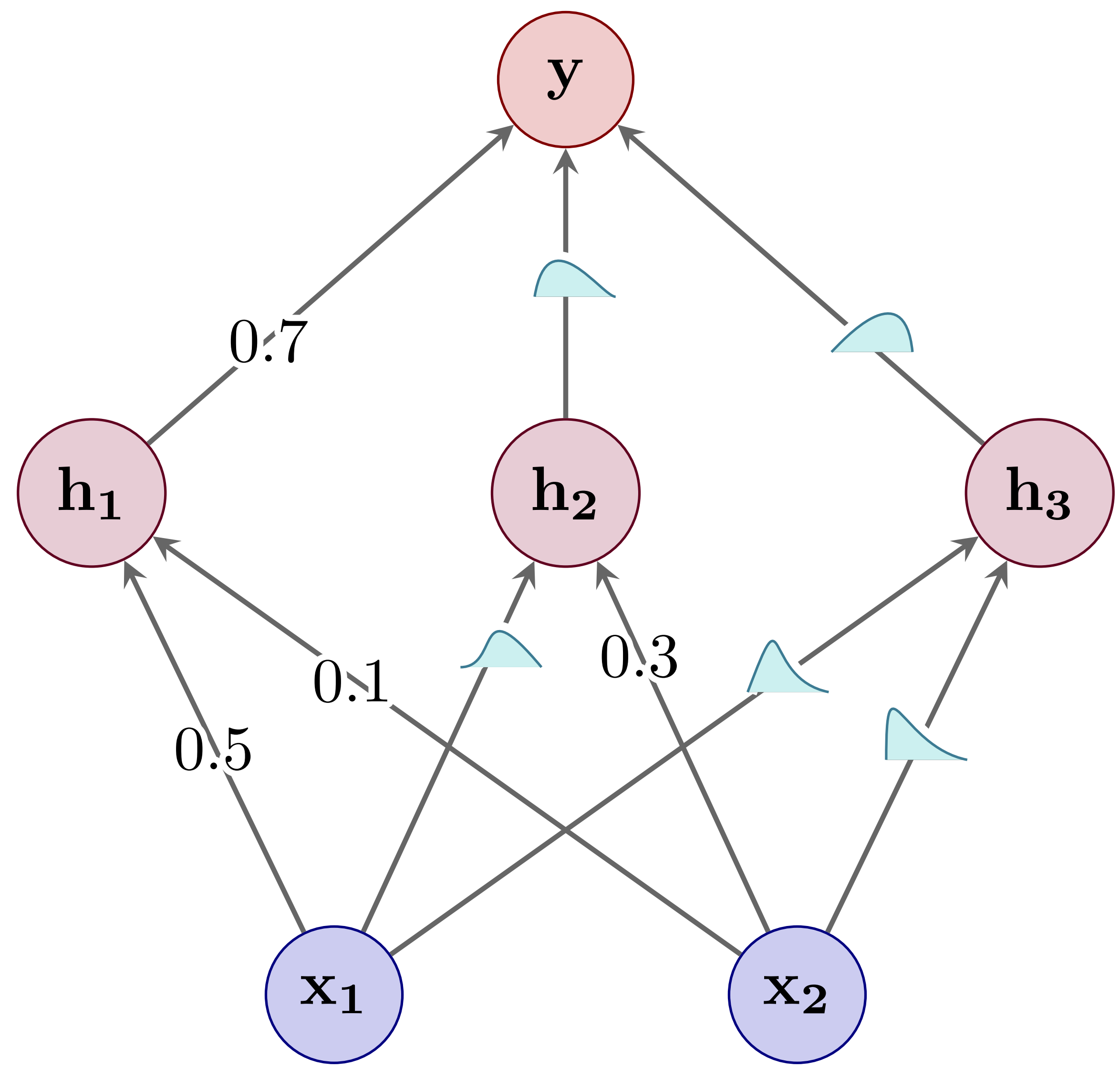
Subnetwork Inference

1. MAP Estimation
2. Subnet Selection
3. Bayes. Inference



Subnetwork Inference

1. MAP Estimation
2. Subnet Selection
3. Bayes. Inference
4. **Prediction**



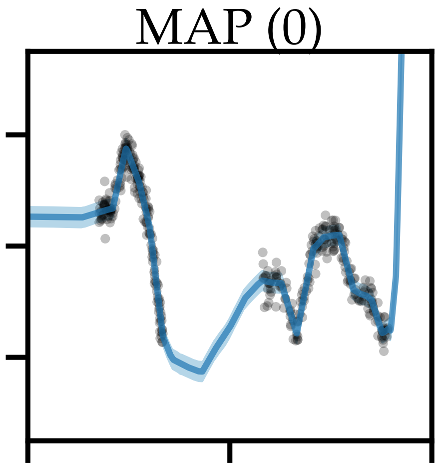
1D Regression

1D Regression

Model: 2 hidden layer, fully-connected NN
with a total of 2600 weights

1D Regression

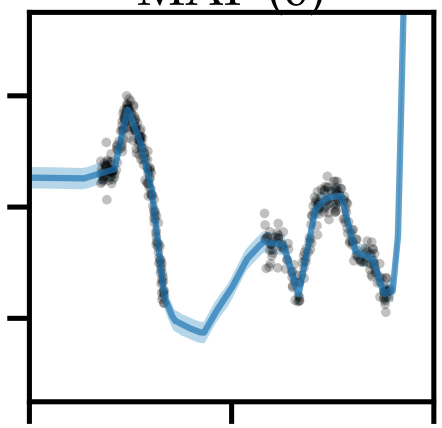
Model: 2 hidden layer, fully-connected NN
with a total of 2600 weights



1D Regression

Model: 2 hidden layer, fully-connected NN with a total of 2600 weights

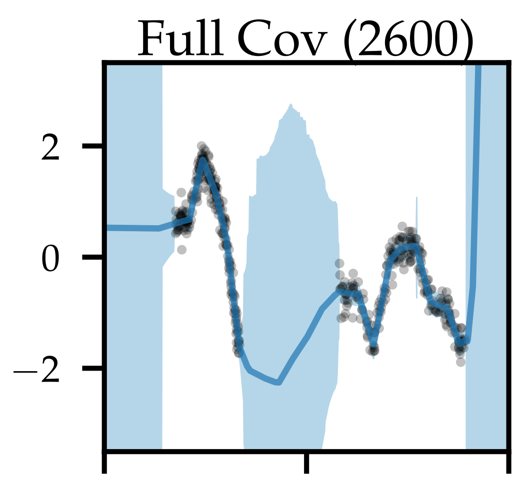
Goal: test 'in-between' predictive uncertainty (Foong 2019)

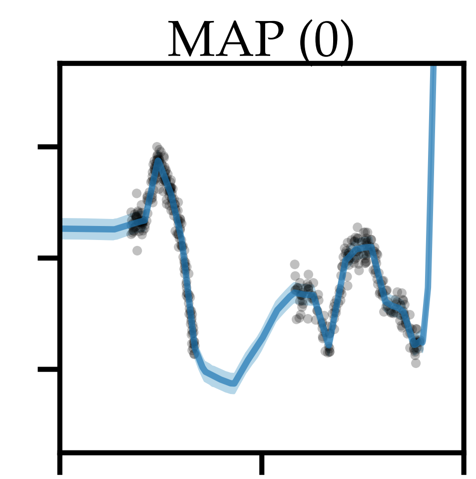


1D Regression

Model: 2 hidden layer, fully-connected NN with a total of 2600 weights

Goal: test ‘in-between’ predictive uncertainty (Foong 2019)

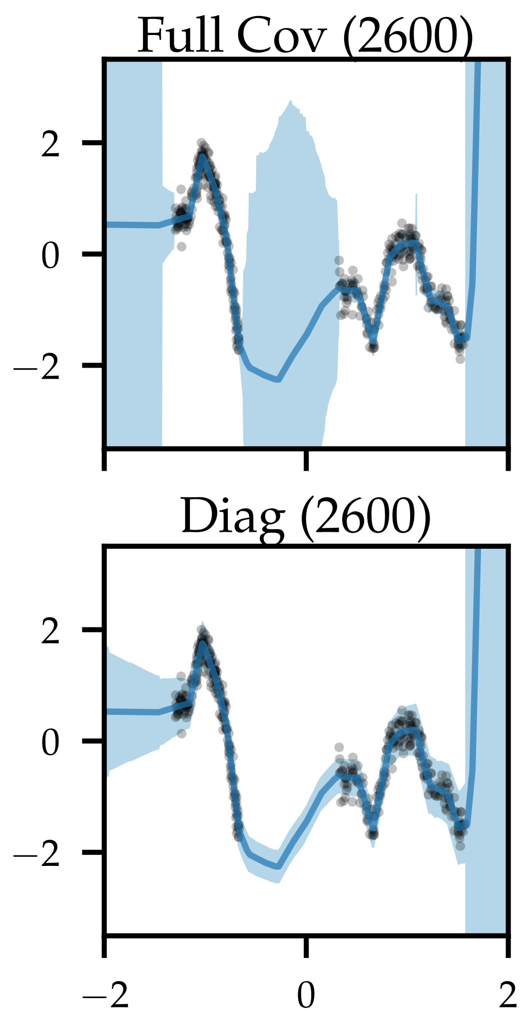


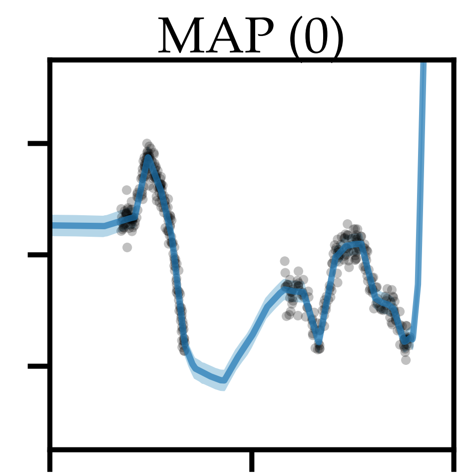


1D Regression

Model: 2 hidden layer, fully-connected NN with a total of 2600 weights

Goal: test ‘in-between’ predictive uncertainty (Foong 2019)

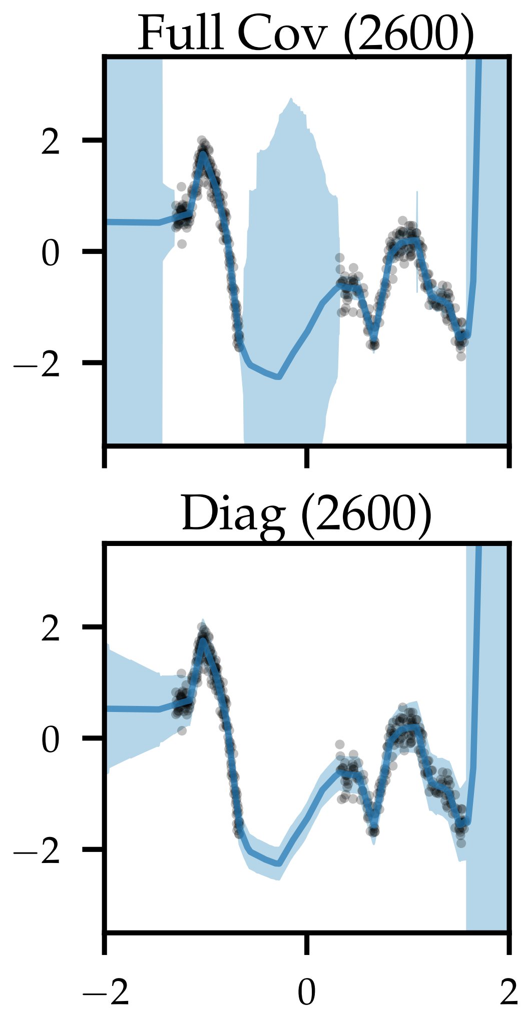


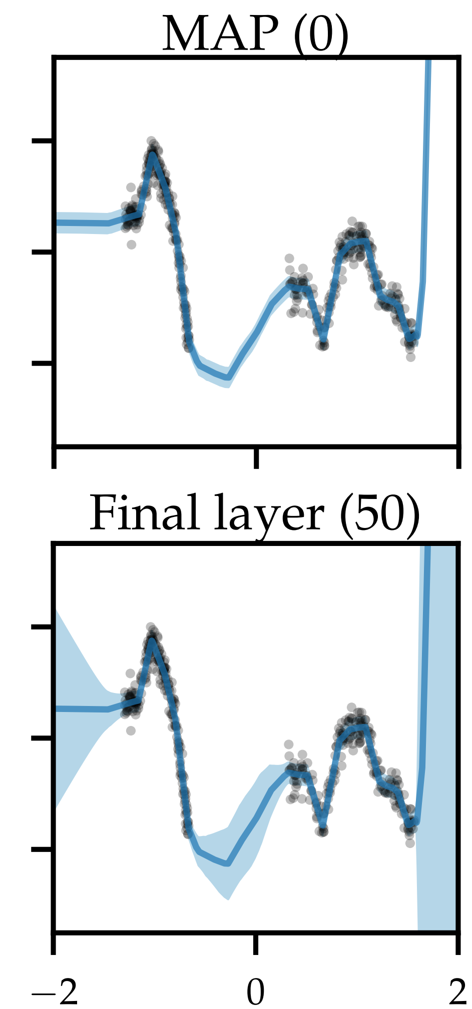


1D Regression

Model: 2 hidden layer, fully-connected NN with a total of 2600 weights

Goal: test ‘in-between’ predictive uncertainty (Foong 2019)

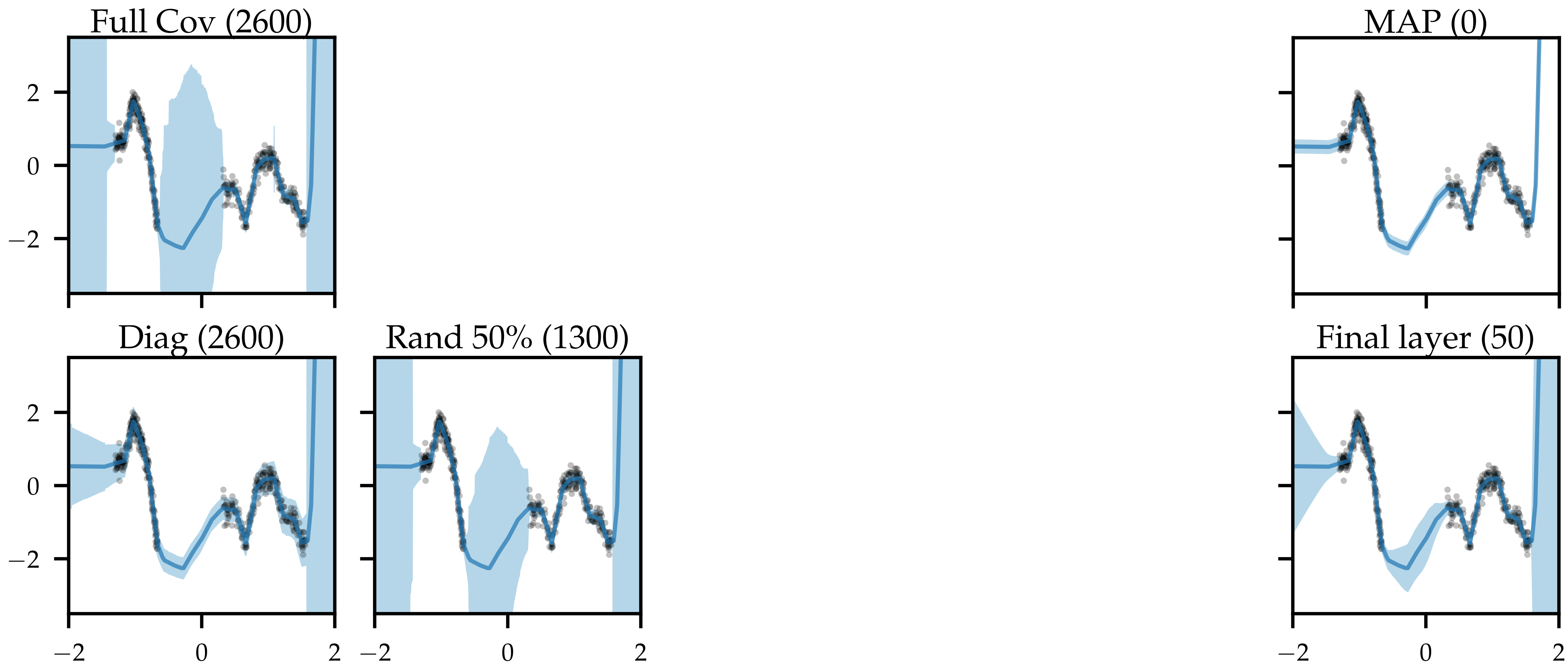




1D Regression

Model: 2 hidden layer, fully-connected NN with a total of 2600 weights

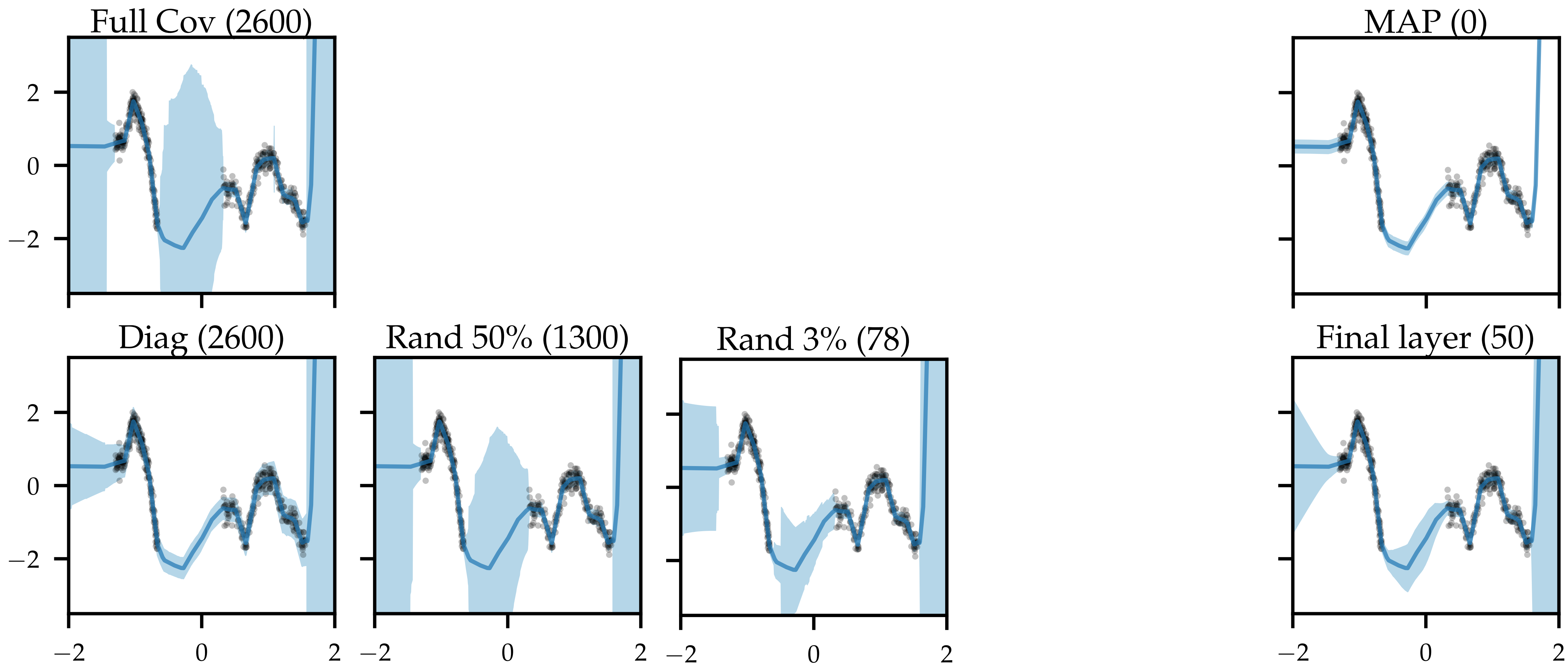
Goal: test 'in-between' predictive uncertainty (Foong 2019)



1D Regression

Model: 2 hidden layer, fully-connected NN with a total of 2600 weights

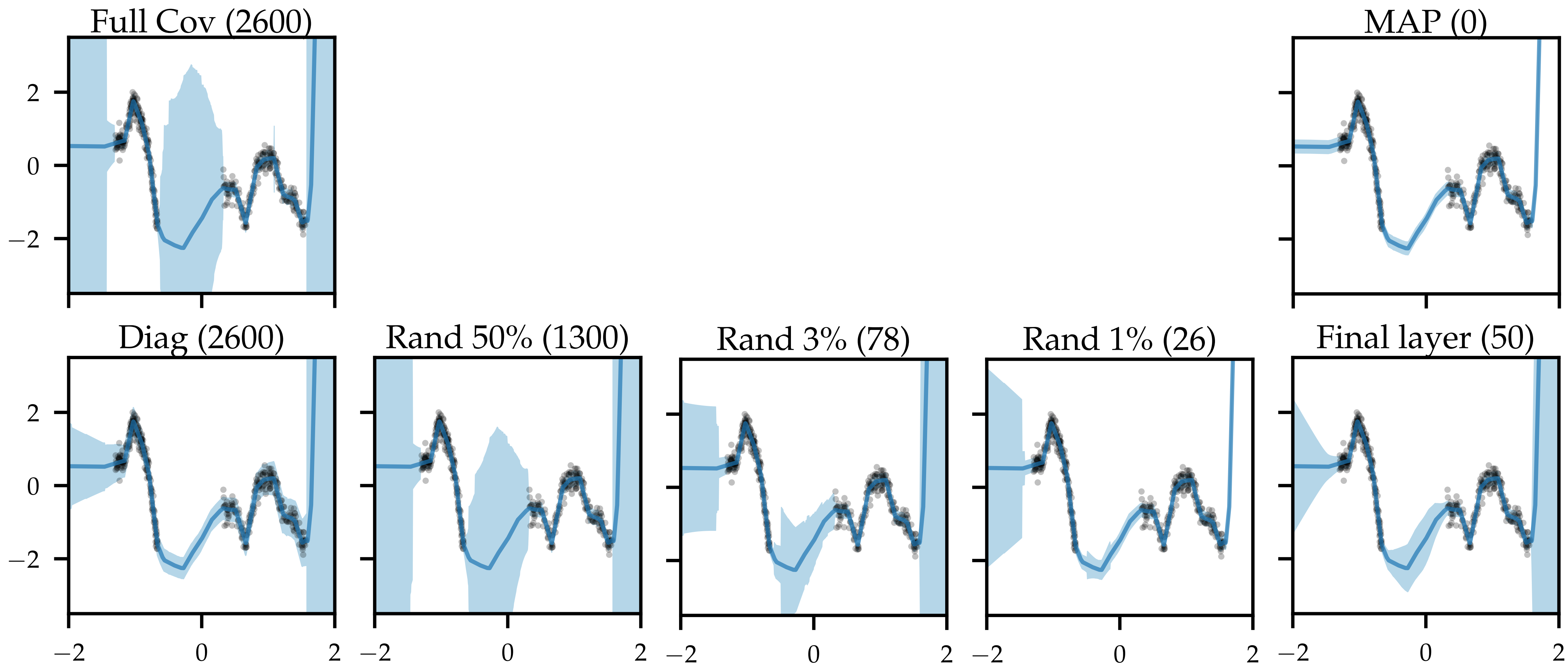
Goal: test 'in-between' predictive uncertainty (Foong 2019)



1D Regression

Model: 2 hidden layer, fully-connected NN with a total of 2600 weights

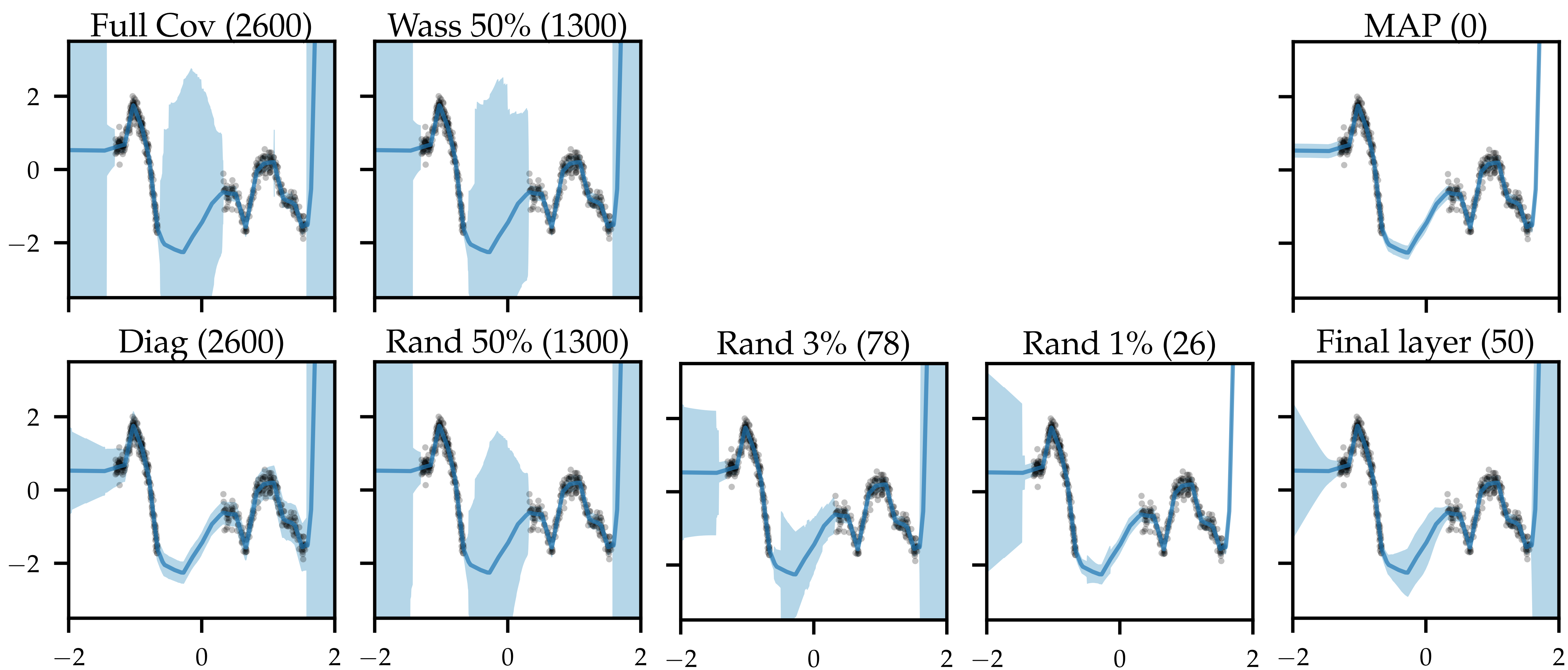
Goal: test 'in-between' predictive uncertainty (Foong 2019)



1D Regression

Model: 2 hidden layer, fully-connected NN with a total of 2600 weights

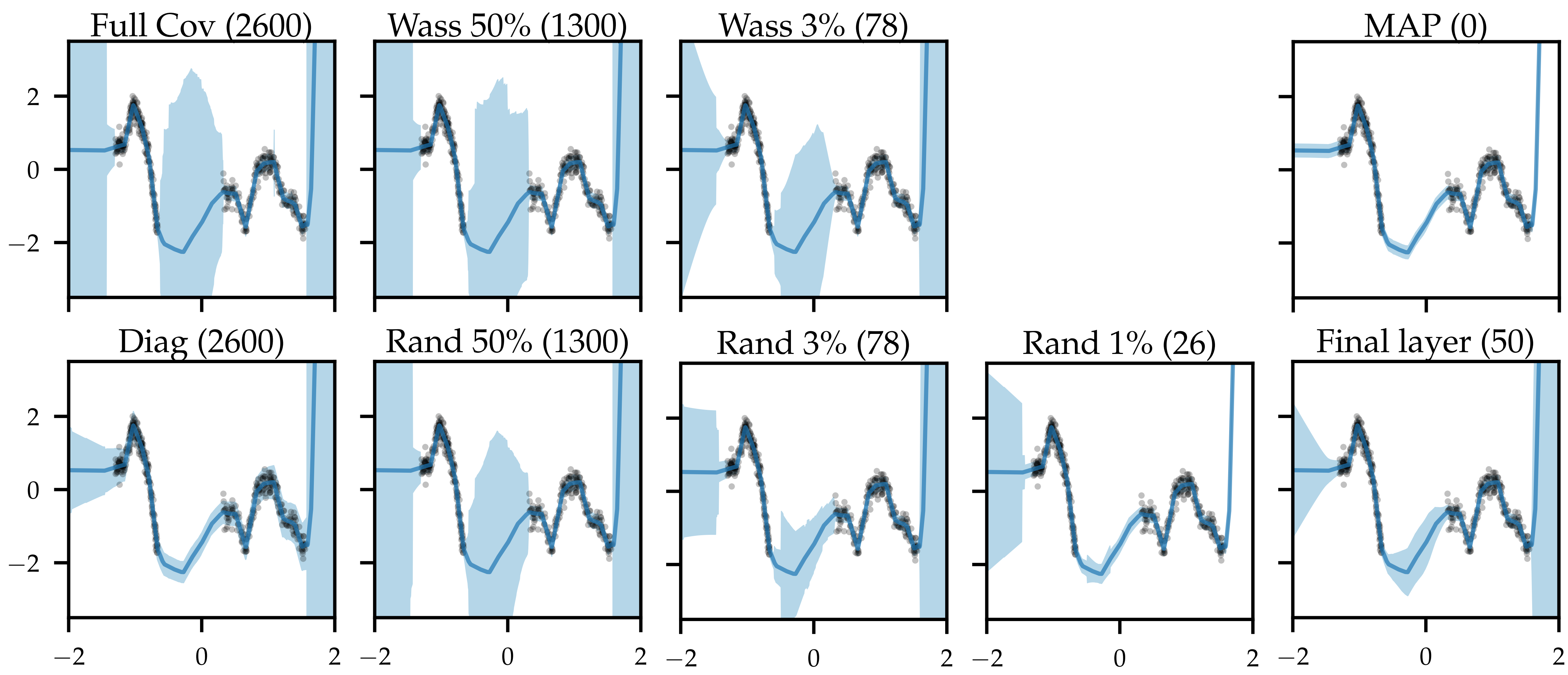
Goal: test 'in-between' predictive uncertainty (Foong 2019)



1D Regression

Model: 2 hidden layer, fully-connected NN with a total of 2600 weights

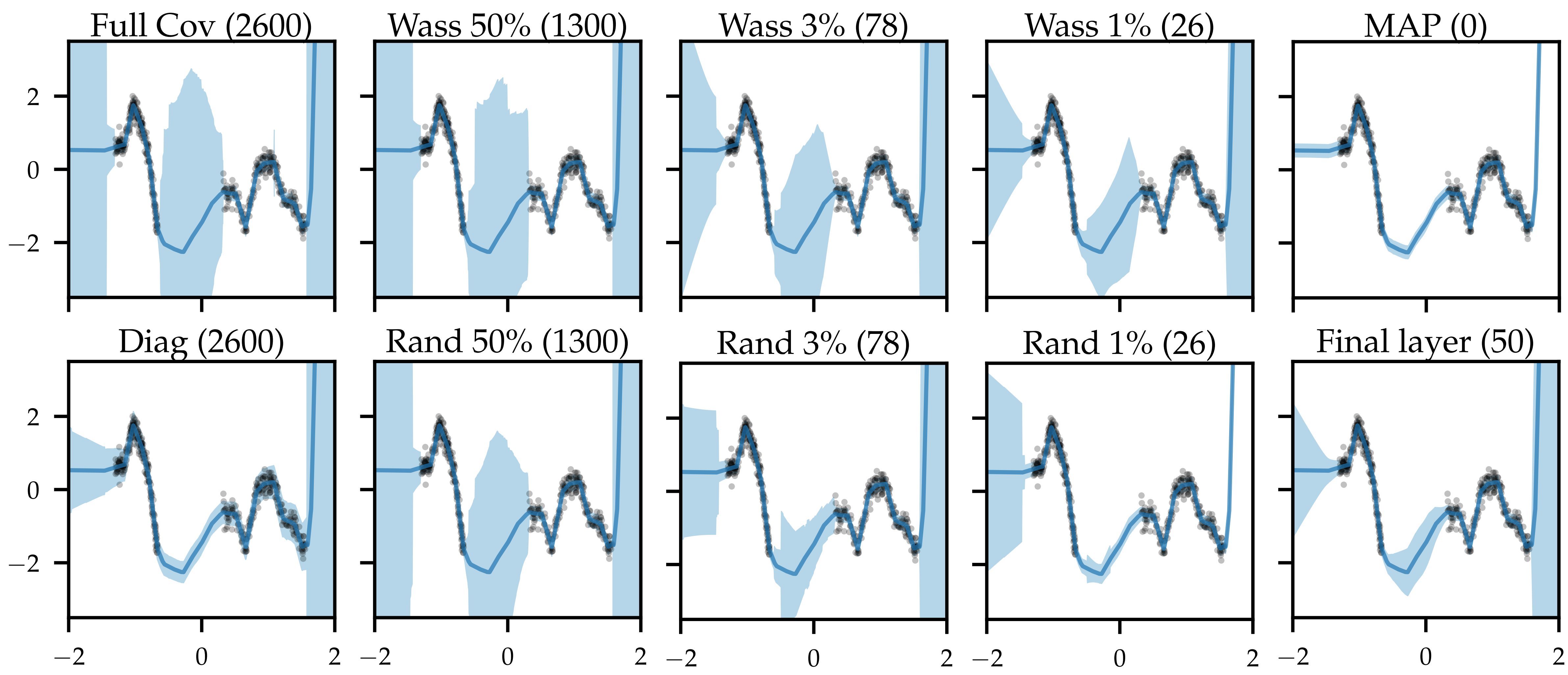
Goal: test ‘in-between’ predictive uncertainty (Foong 2019)



1D Regression

Model: 2 hidden layer, fully-connected NN with a total of 2600 weights

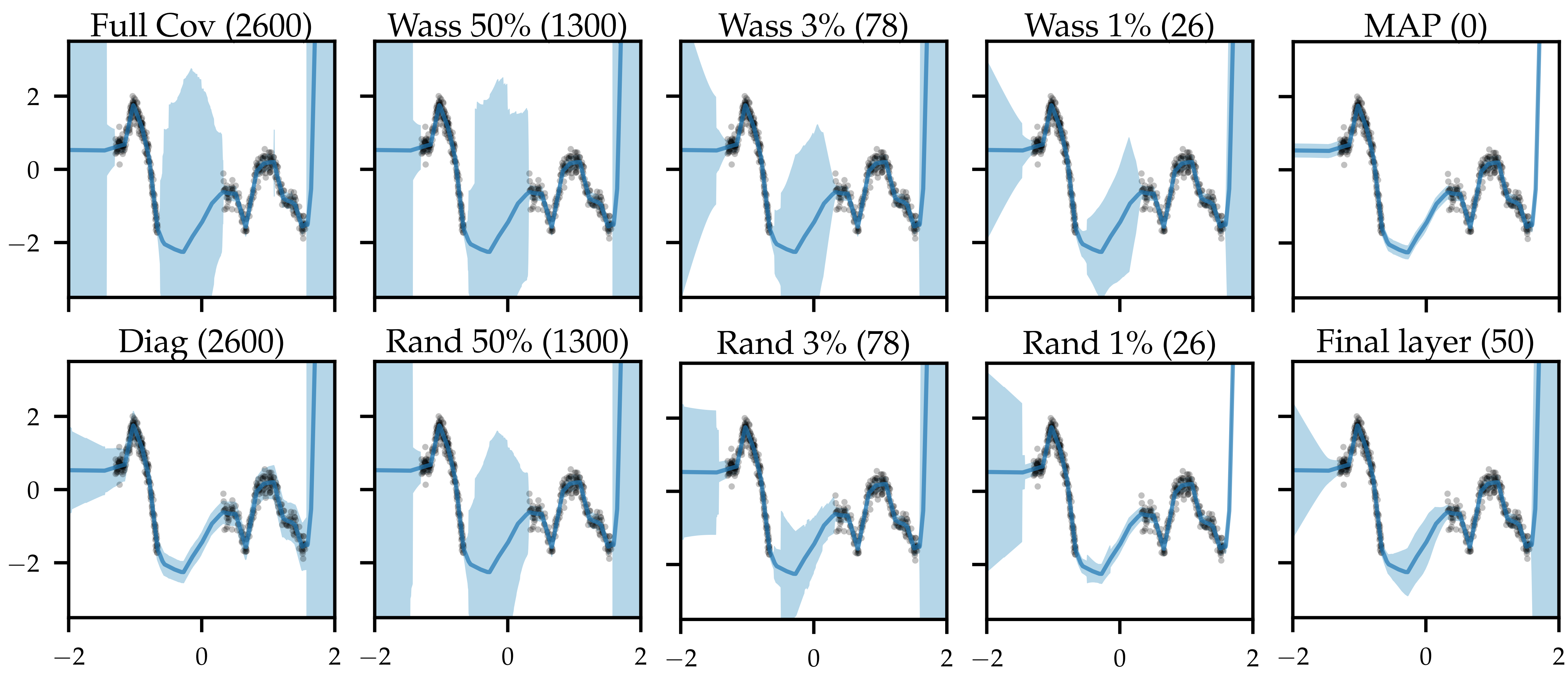
Goal: test 'in-between' predictive uncertainty (Foong 2019)



1D Regression

Model: 2 hidden layer, fully-connected NN with a total of 2600 weights

Goal: test 'in-between' predictive uncertainty (Foong 2019)



→ Expressive inference over a small subnetwork preserves **more predictive uncertainty** than crude inference over the full network!

Interaction Between Network Size and Subnetwork Size

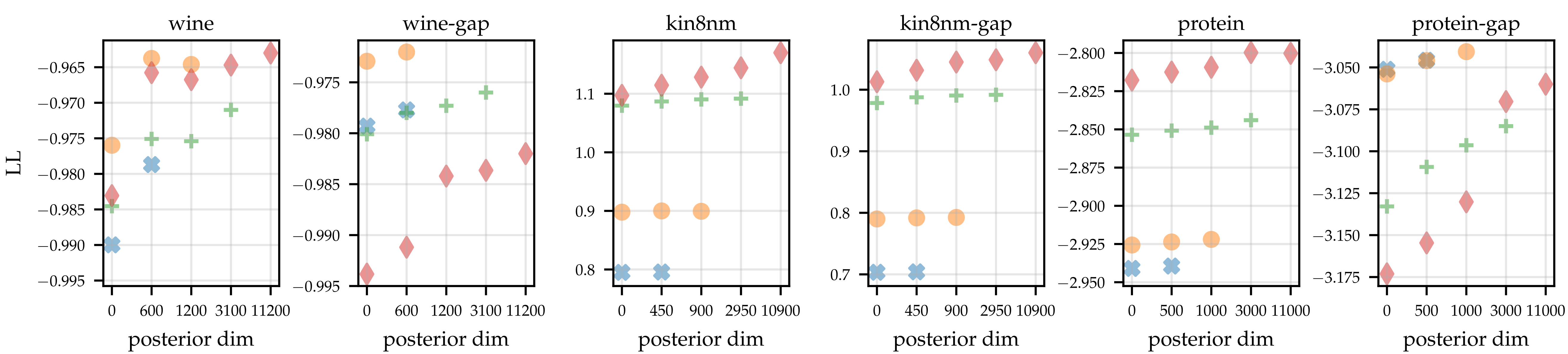
We compare 4 models:

1. 50 hidden units, 1 hidden layer $w_i{:}100,\ h_i{:}1$
2. 100 hidden units, 1 hidden layer $w_i{:}50,\ h_i{:}1$
3. 50 hidden units, 2 hidden layer $w_i{:}50,\ h_i{:}2$
4. 100 hidden units, 2 hidden layer $w_i{:}100,\ h_i{:}2$

Interaction Between Network Size and Subnetwork Size

We compare 4 models:

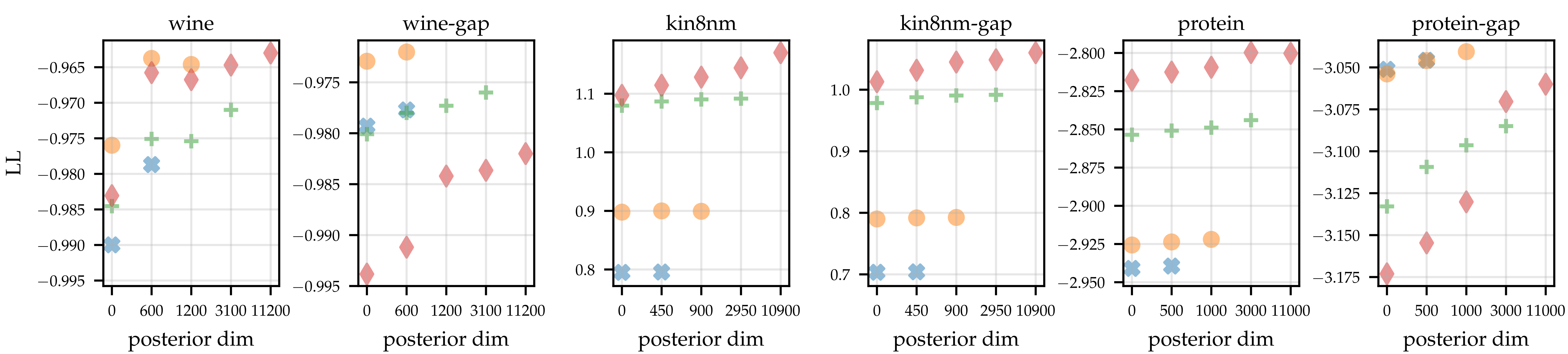
1. 50 hidden units, 1 hidden layer — w_i:100, h_i:1
2. 100 hidden units, 1 hidden layer — w_i:50, h_i:1
3. 50 hidden units, 2 hidden layer — w_i:50, h_i:2
4. 100 hidden units, 2 hidden layer — w_i:100, h_i:2



Interaction Between Network Size and Subnetwork Size

We compare 4 models:

1. 50 hidden units, 1 hidden layer — ● w_i:100, h_i:1
2. 100 hidden units, 1 hidden layer — ✖ w_i:50, h_i:1
3. 50 hidden units, 2 hidden layer — ✚ w_i:50, h_i:2
4. 100 hidden units, 2 hidden layer — ♦ w_i:100, h_i:2



➡ Given the same amount of compute, **larger models benefit more from subnetwork inference.**

Image Class. under Distribution Shift

Image Class. under Distribution Shift

Model:
ResNet-18 with **11M** weights

Image Class. under Distribution Shift

Model:
ResNet-18 with **11M** weights

⬇

Wasserstein subnetwork inference
subnet of just **42K (0.38%)** weights

Image Class. under Distribution Shift

Model:
ResNet-18 with **11M** weights

⬇

Wasserstein subnetwork inference
subnet of just **42K (0.38%)** weights

Baselines:

Image Class. under Distribution Shift

Model:
ResNet-18 with **11M** weights

⬇

Wasserstein subnetwork inference
subnet of just **42K (0.38%)** weights

Baselines:

- MAP

Image Class. under Distribution Shift

Model:
ResNet-18 with **11M** weights

Wasserstein subnetwork inference
subnet of just **42K (0.38%)** weights

Baselines:
- MAP
- Diagonal Laplace

Image Class. under Distribution Shift

Model:
ResNet-18 with **11M** weights

↓

Wasserstein subnetwork inference
subnet of just **42K (0.38%)** weights

Baselines:
- MAP
- Diagonal Laplace
- MC Dropout (Gal 2016)

Image Class. under Distribution Shift

Model:
ResNet-18 with **11M** weights

⬇

Wasserstein subnetwork inference
subnet of just **42K (0.38%)** weights

Baselines:

- MAP
- Diagonal Laplace
- MC Dropout (Gal 2016)
- Deep Ensembles (Lakshminarayanan 2017)

Image Class. under Distribution Shift

Model:
ResNet-18 with **11M** weights

⬇

Wasserstein subnetwork inference
subnet of just **42K (0.38%)** weights

Baselines:
- MAP
- Diagonal Laplace
- MC Dropout (Gal 2016)
- Deep Ensembles (Lakshminarayanan 2017)
- SWAG (Maddox 2019)

Image Class. under Distribution Shift

Model:
ResNet-18 with **11M** weights

Wasserstein subnetwork inference
subnet of just **42K (0.38%)** weights

Baselines:
- MAP
- Diagonal Laplace
- MC Dropout (Gal 2016)
- Deep Ensembles (Lakshminarayanan 2017)
- SWAG (Maddox 2019)

Rotated MNIST (Ovadia 2019)

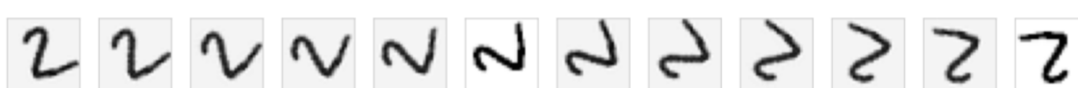


Image Class. under Distribution Shift

Model:
ResNet-18 with **11M** weights

Wasserstein subnetwork inference
subnet of just **42K (0.38%)** weights

Baselines:
- MAP
- Diagonal Laplace
- MC Dropout (Gal 2016)
- Deep Ensembles (Lakshminarayanan 2017)
- SWAG (Maddox 2019)

Rotated MNIST (Ovadia 2019)

Corrupted CIFAR10 (Ovadia 2019)

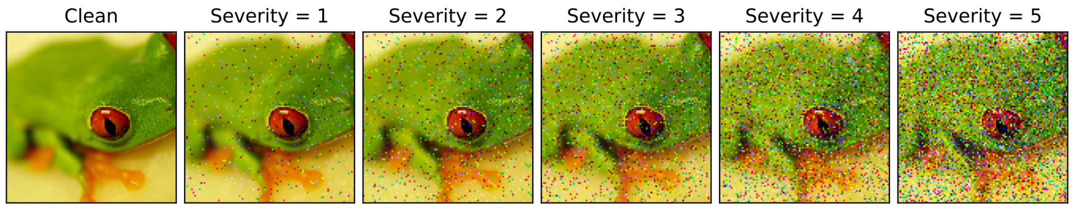


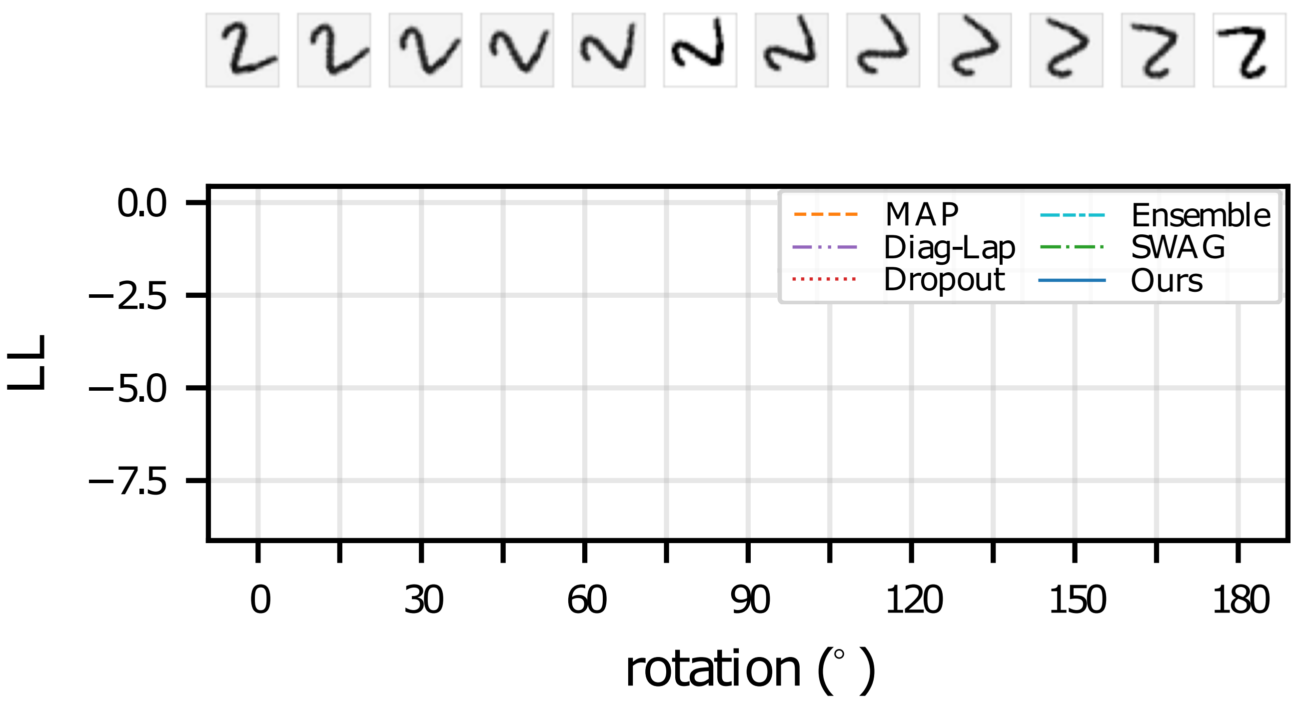
Image Class. under Distribution Shift

Model:
ResNet-18 with **11M** weights

Wasserstein subnetwork inference
subnet of just **42K (0.38%)** weights

Baselines:
- MAP
- Diagonal Laplace
- MC Dropout (Gal 2016)
- Deep Ensembles (Lakshminarayanan 2017)
- SWAG (Maddox 2019)

Rotated MNIST (Ovadia 2019)



Corrupted CIFAR10 (Ovadia 2019)

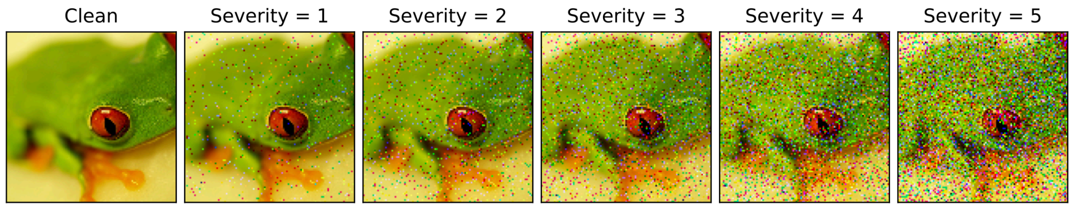


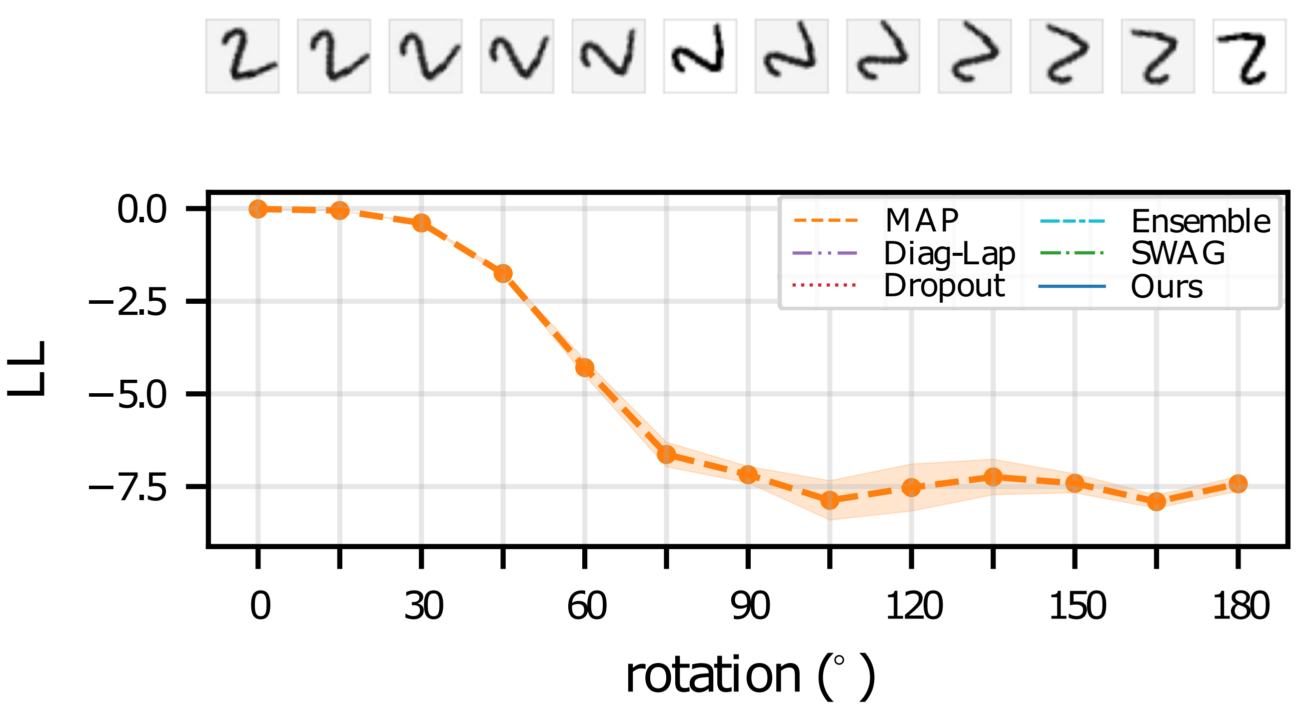
Image Class. under Distribution Shift

Model:
ResNet-18 with **11M** weights

Wasserstein subnetwork inference
subnet of just **42K (0.38%)** weights

Baselines:
- MAP
- Diagonal Laplace
- MC Dropout (Gal 2016)
- Deep Ensembles (Lakshminarayanan 2017)
- SWAG (Maddox 2019)

Rotated MNIST (Ovadia 2019)



Corrupted CIFAR10 (Ovadia 2019)

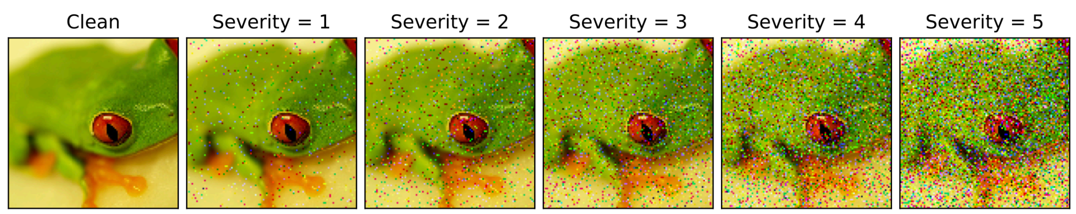


Image Class. under Distribution Shift

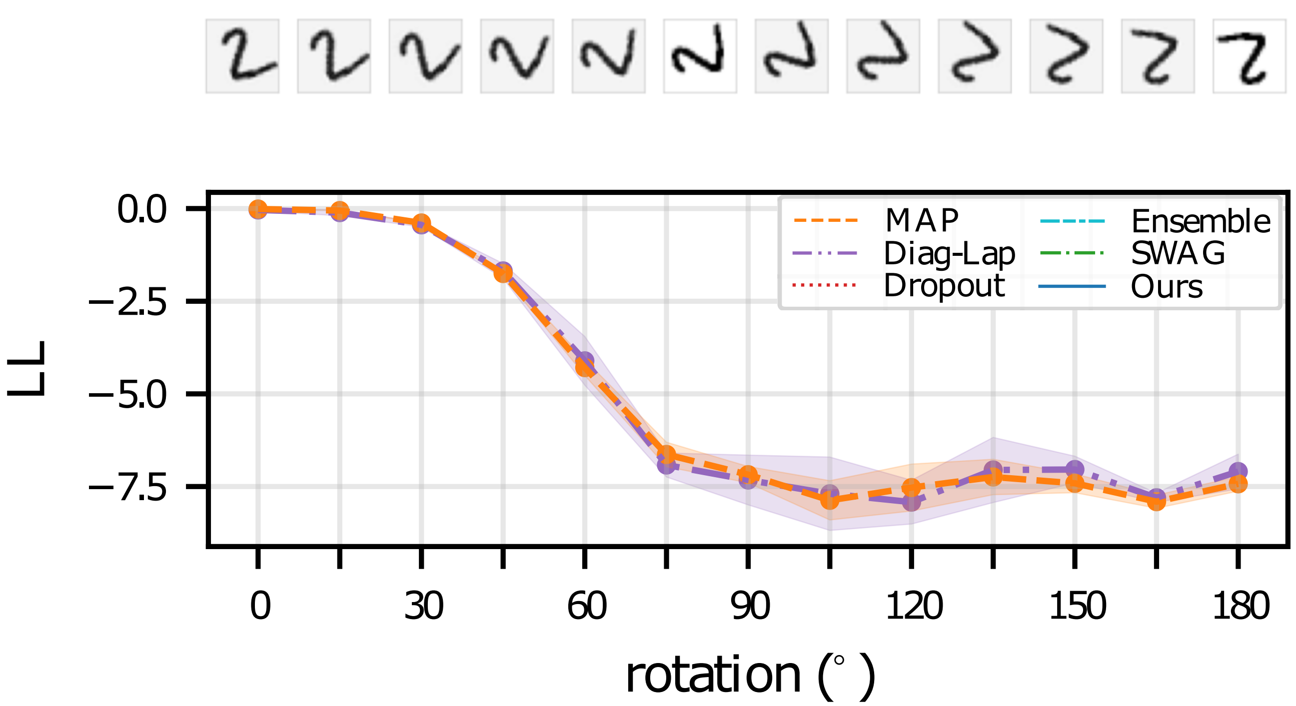
Model:
ResNet-18 with **11M** weights

⬇

Wasserstein subnetwork inference
subnet of just **42K (0.38%)** weights

Baselines:
- MAP
- Diagonal Laplace
- MC Dropout (Gal 2016)
- Deep Ensembles (Lakshminarayanan 2017)
- SWAG (Maddox 2019)

Rotated MNIST (Ovadia 2019)



Corrupted CIFAR10 (Ovadia 2019)

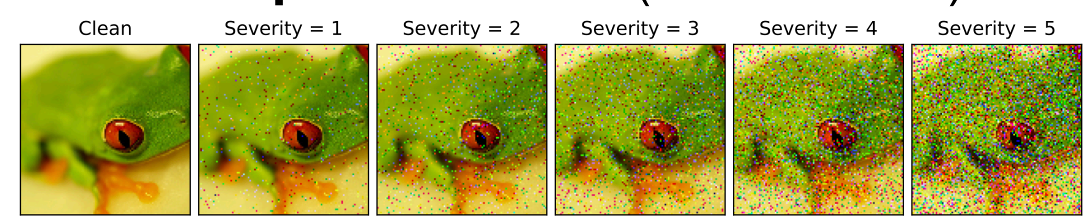


Image Class. under Distribution Shift

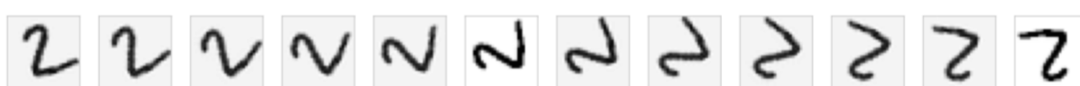
Model:
ResNet-18 with **11M** weights

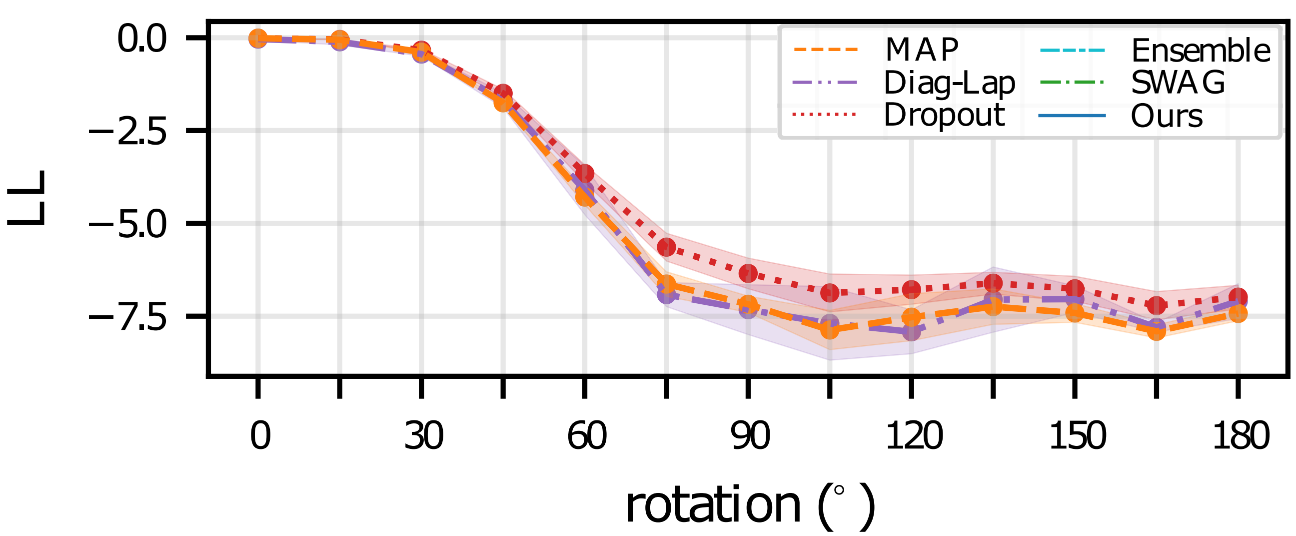
Wasserstein subnetwork inference
subnet of just **42K (0.38%)** weights

Baselines:

- MAP
- Diagonal Laplace
- MC Dropout (Gal 2016)
- Deep Ensembles (Lakshminarayanan 2017)
- SWAG (Maddox 2019)

Rotated MNIST (Ovadia 2019)





Corrupted CIFAR10 (Ovadia 2019)

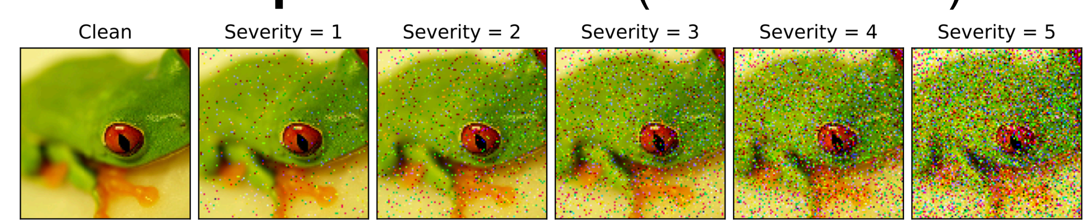


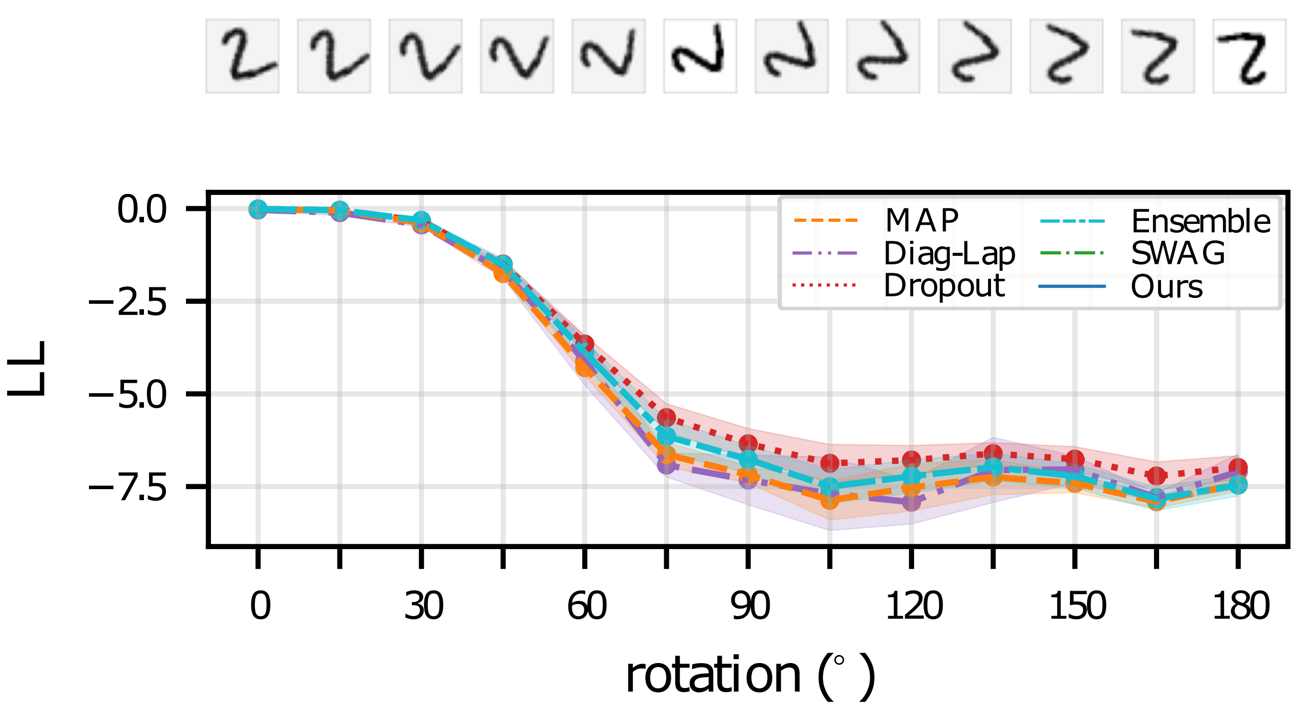
Image Class. under Distribution Shift

Model:
ResNet-18 with **11M** weights

Wasserstein subnetwork inference
subnet of just **42K (0.38%)** weights

Baselines:
- MAP
- Diagonal Laplace
- MC Dropout (Gal 2016)
- Deep Ensembles (Lakshminarayanan 2017)
- SWAG (Maddox 2019)

Rotated MNIST (Ovadia 2019)



Corrupted CIFAR10 (Ovadia 2019)

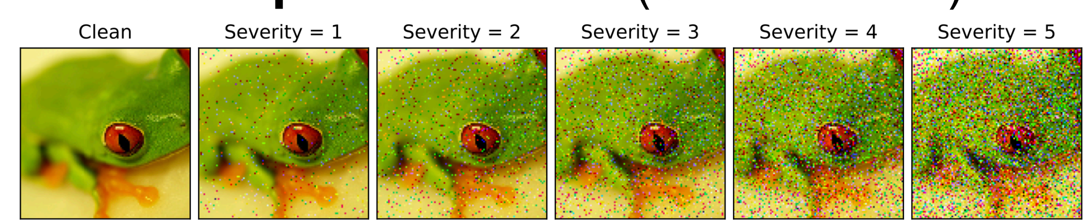


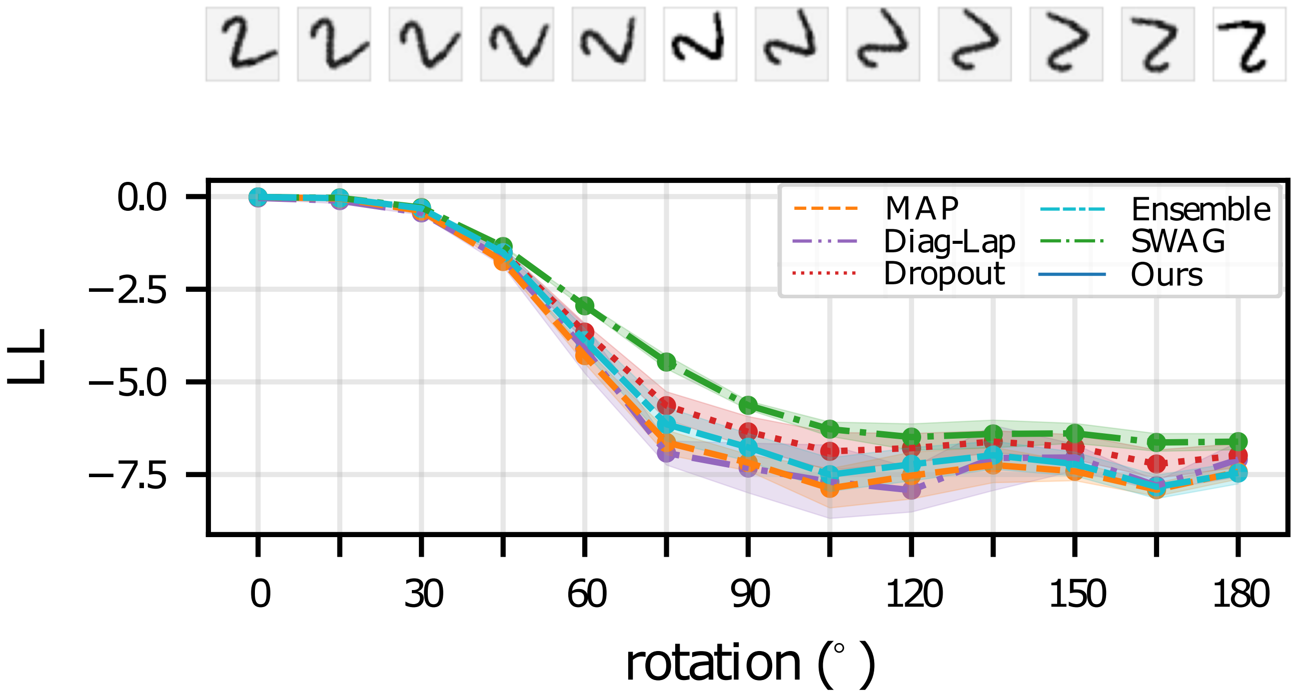
Image Class. under Distribution Shift

Model:
ResNet-18 with **11M** weights

Wasserstein subnetwork inference
subnet of just **42K (0.38%)** weights

Baselines:
- MAP
- Diagonal Laplace
- MC Dropout (Gal 2016)
- Deep Ensembles (Lakshminarayanan 2017)
- SWAG (Maddox 2019)

Rotated MNIST (Ovadia 2019)



Corrupted CIFAR10 (Ovadia 2019)

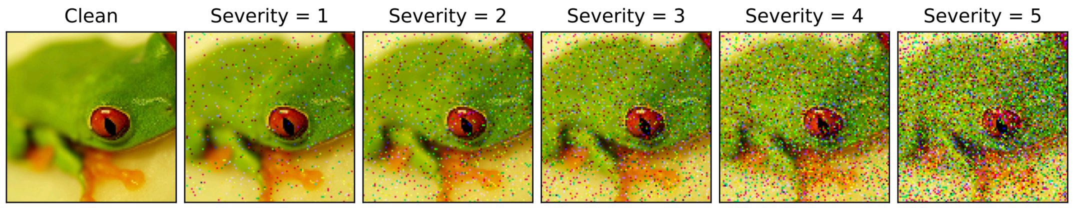


Image Class. under Distribution Shift

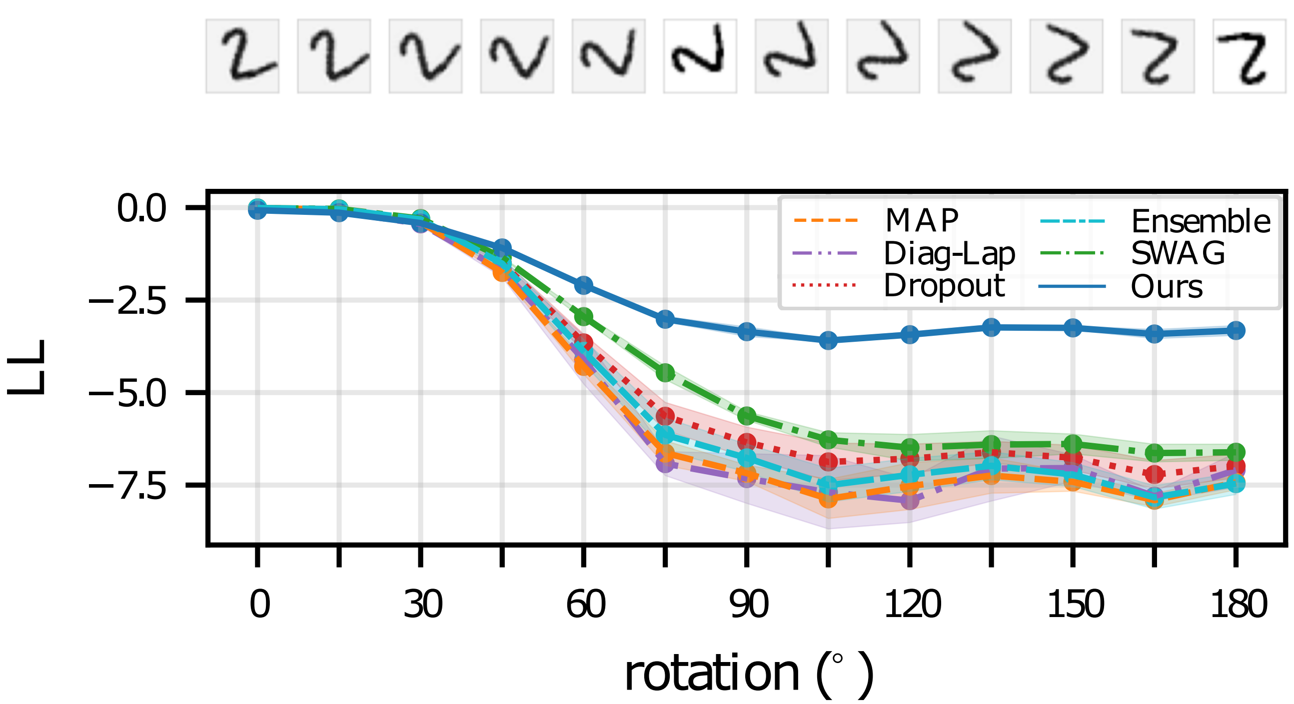
Model:
ResNet-18 with **11M** weights

🡇

Wasserstein subnetwork inference
subnet of just **42K (0.38%)** weights

Baselines:
- MAP
- Diagonal Laplace
- MC Dropout (Gal 2016)
- Deep Ensembles (Lakshminarayanan 2017)
- SWAG (Maddox 2019)

Rotated MNIST (Ovadia 2019)



Corrupted CIFAR10 (Ovadia 2019)

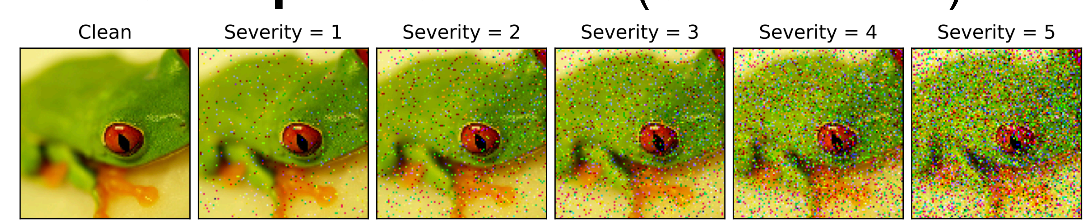


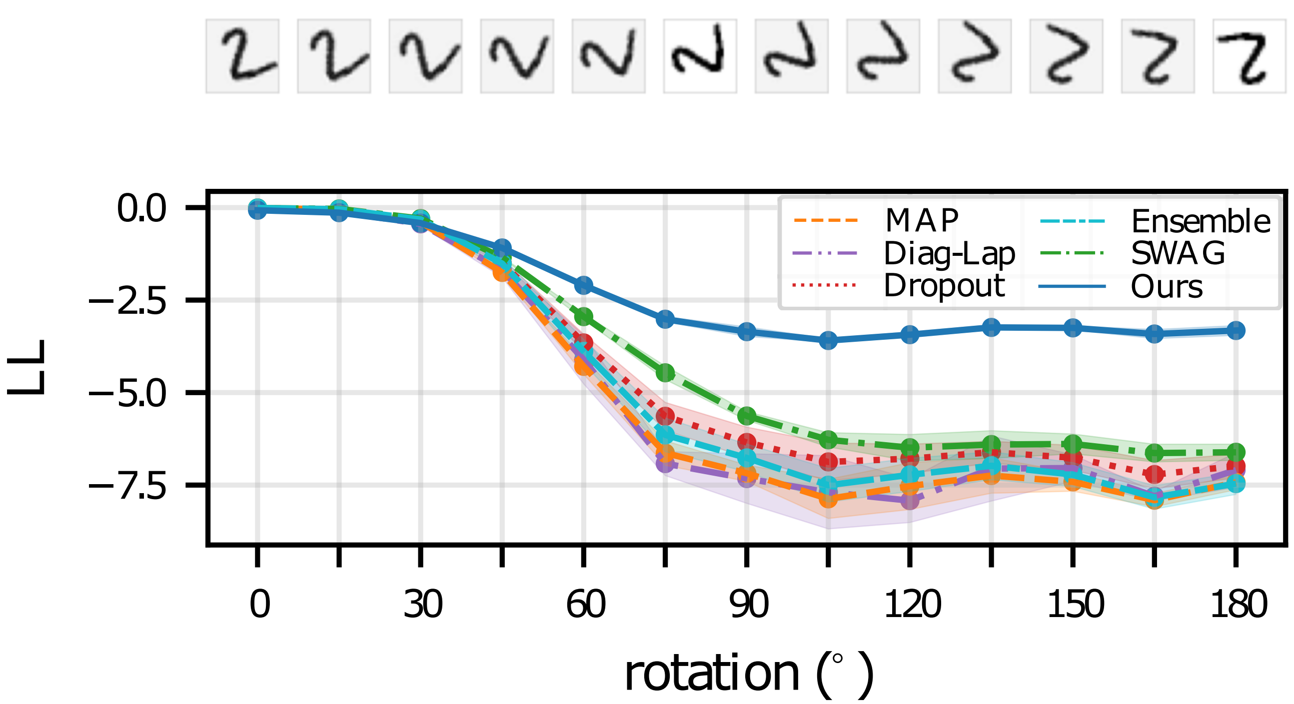
Image Class. under Distribution Shift

Model:
ResNet-18 with **11M** weights

Wasserstein subnetwork inference
subnet of just **42K (0.38%)** weights

Baselines:
- MAP
- Diagonal Laplace
- MC Dropout (Gal 2016)
- Deep Ensembles (Lakshminarayanan 2017)
- SWAG (Maddox 2019)

Rotated MNIST (Ovadia 2019)



Corrupted CIFAR10 (Ovadia 2019)

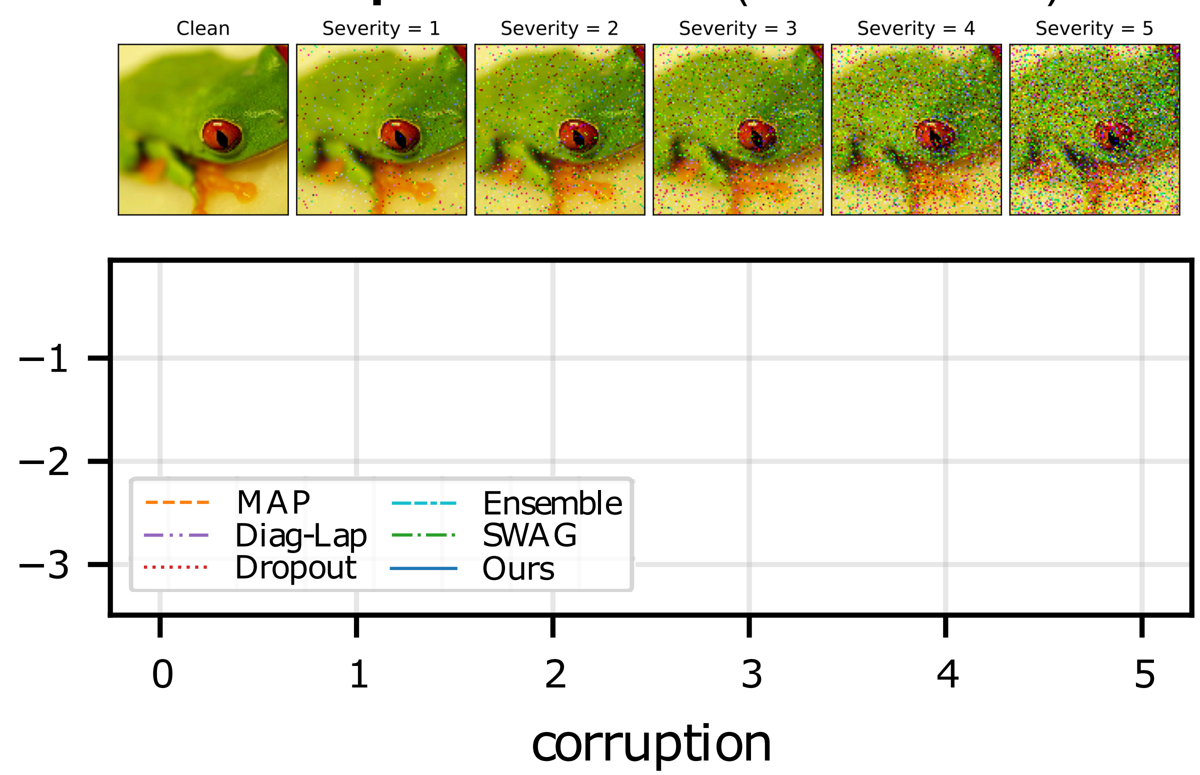


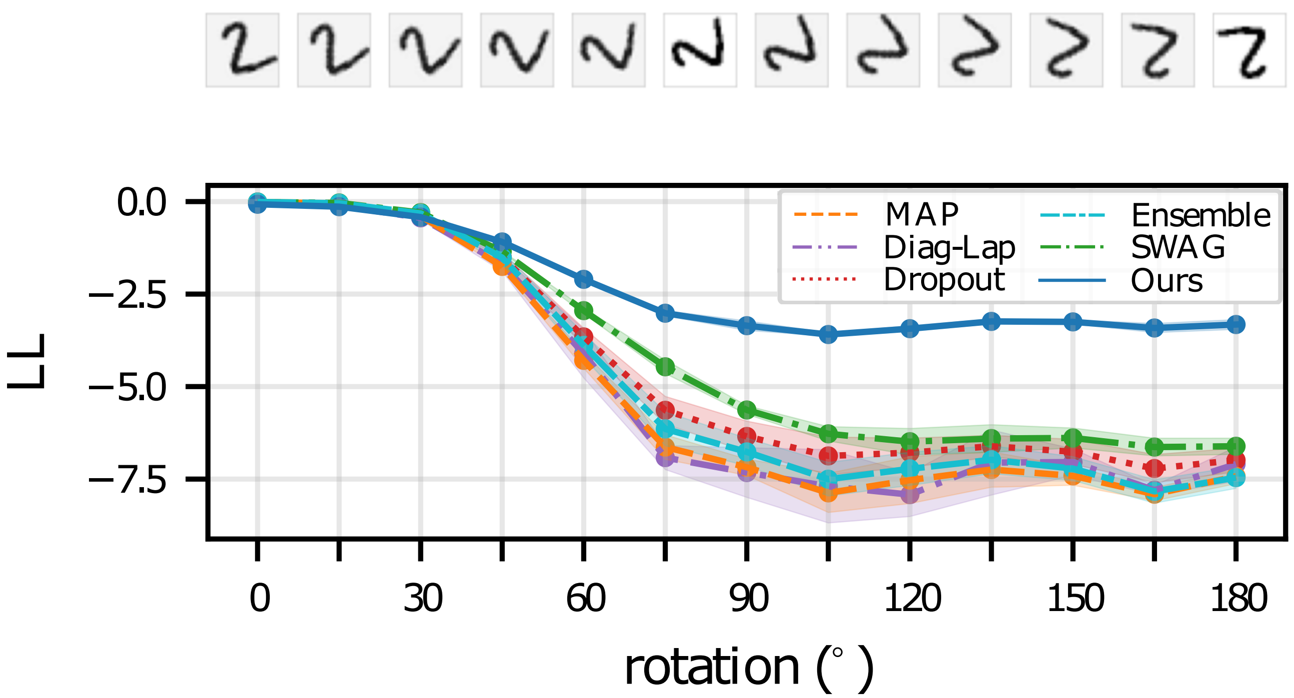
Image Class. under Distribution Shift

Model:
ResNet-18 with **11M** weights

Wasserstein subnetwork inference
subnet of just **42K (0.38%)** weights

Baselines:
- MAP
- Diagonal Laplace
- MC Dropout (Gal 2016)
- Deep Ensembles (Lakshminarayanan 2017)
- SWAG (Maddox 2019)

Rotated MNIST (Ovadia 2019)



Corrupted CIFAR10 (Ovadia 2019)

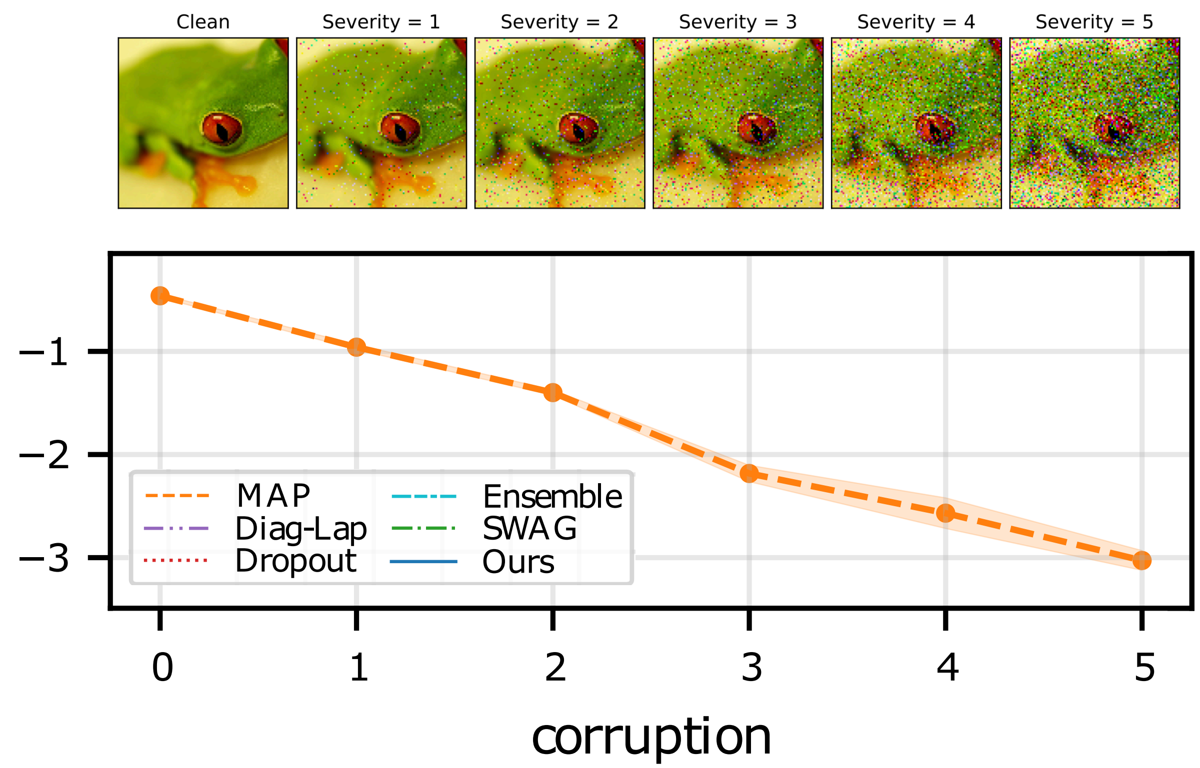


Image Class. under Distribution Shift

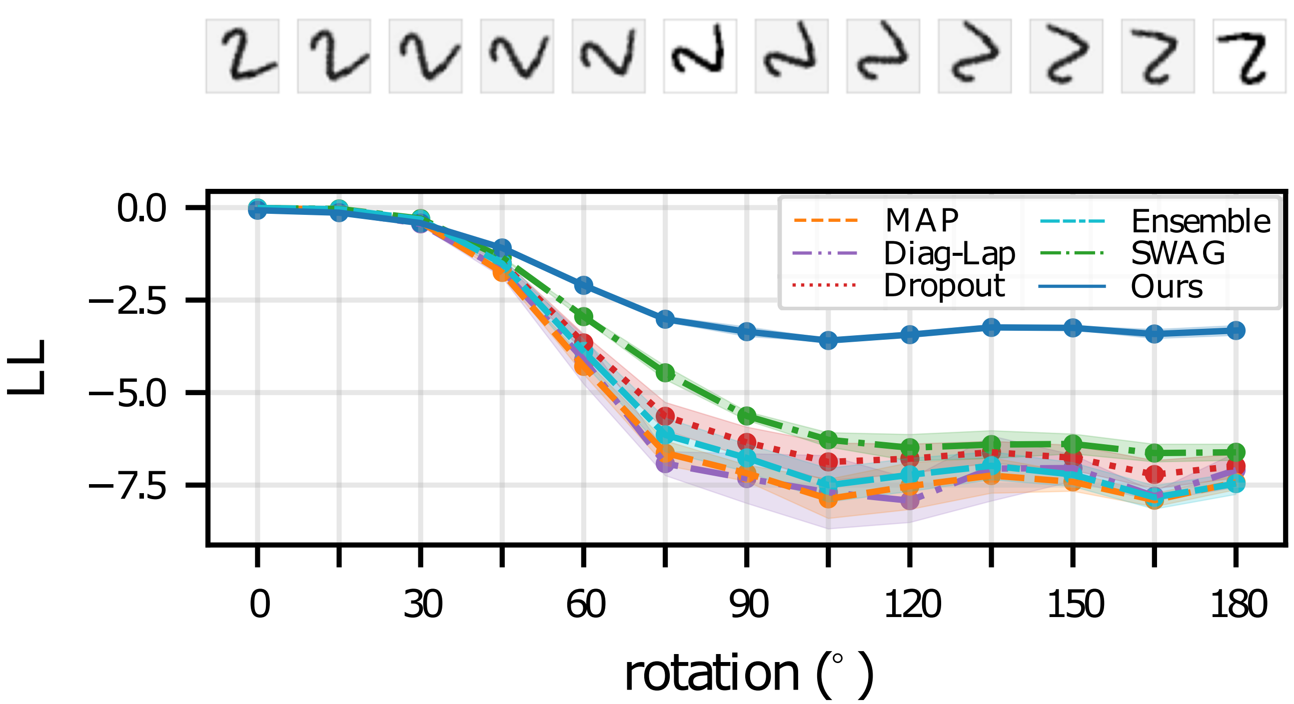
Model:
ResNet-18 with **11M** weights

⬇

Wasserstein subnetwork inference
subnet of just **42K (0.38%)** weights

Baselines:
- MAP
- Diagonal Laplace
- MC Dropout (Gal 2016)
- Deep Ensembles (Lakshminarayanan 2017)
- SWAG (Maddox 2019)

Rotated MNIST (Ovadia 2019)



Corrupted CIFAR10 (Ovadia 2019)

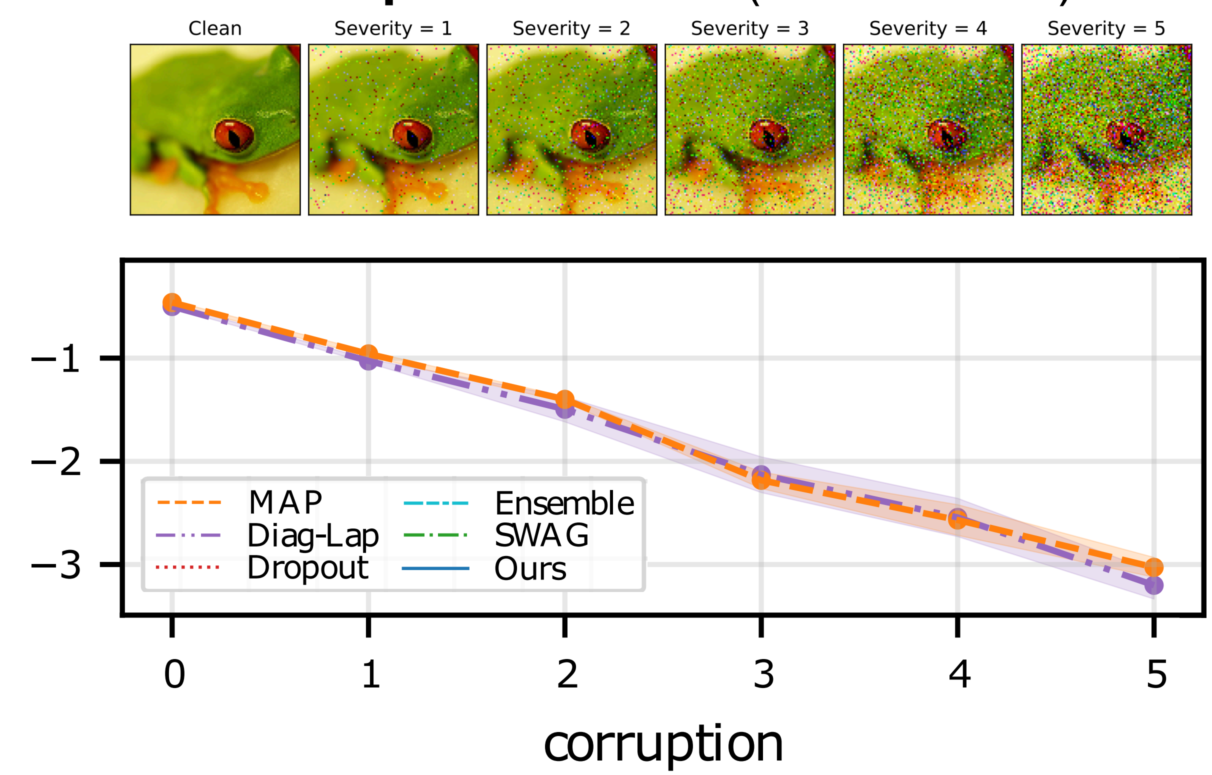


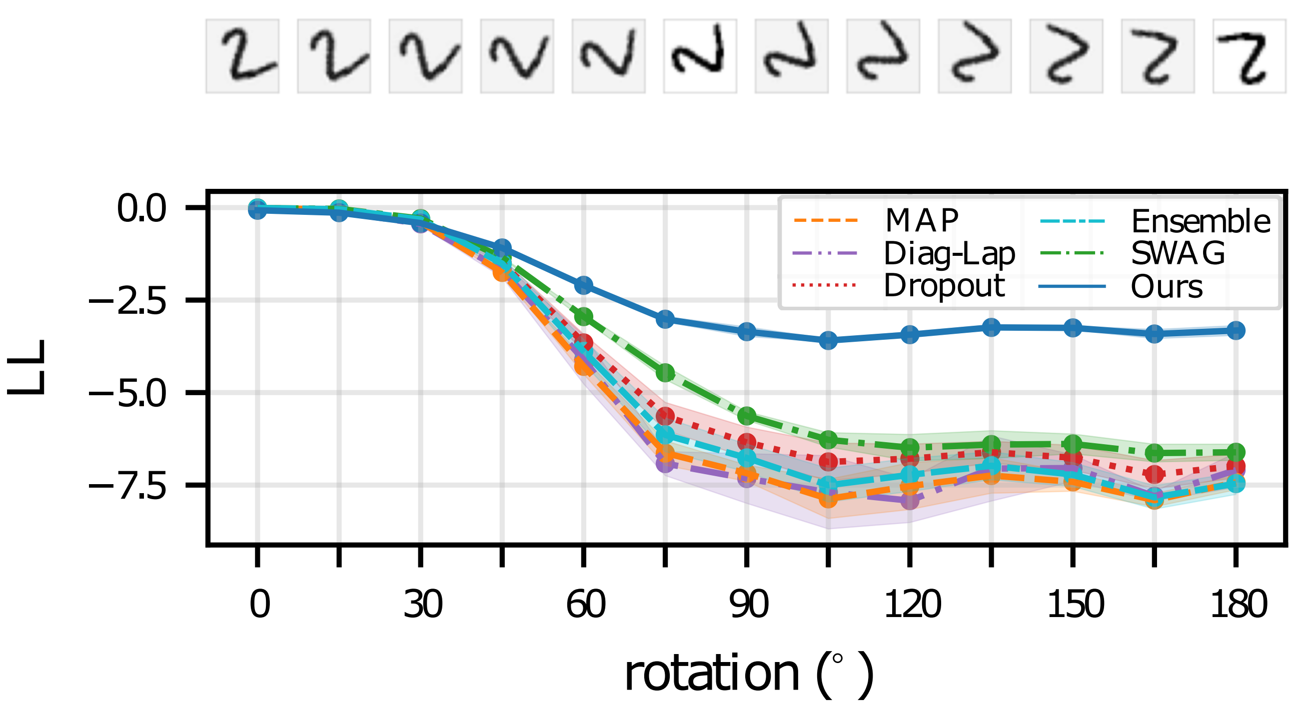
Image Class. under Distribution Shift

Model:
ResNet-18 with **11M** weights

Wasserstein subnetwork inference
subnet of just **42K (0.38%)** weights

Baselines:
- MAP
- Diagonal Laplace
- MC Dropout (Gal 2016)
- Deep Ensembles (Lakshminarayanan 2017)
- SWAG (Maddox 2019)

Rotated MNIST (Ovadia 2019)



Corrupted CIFAR10 (Ovadia 2019)

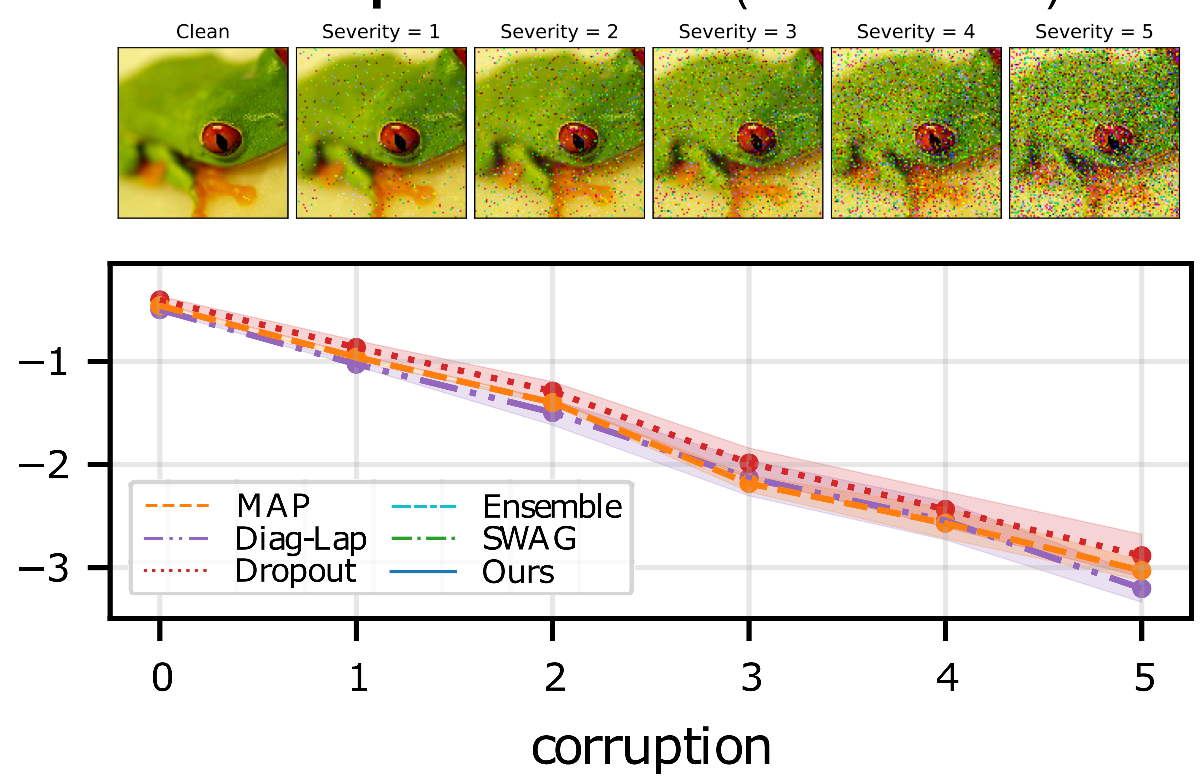


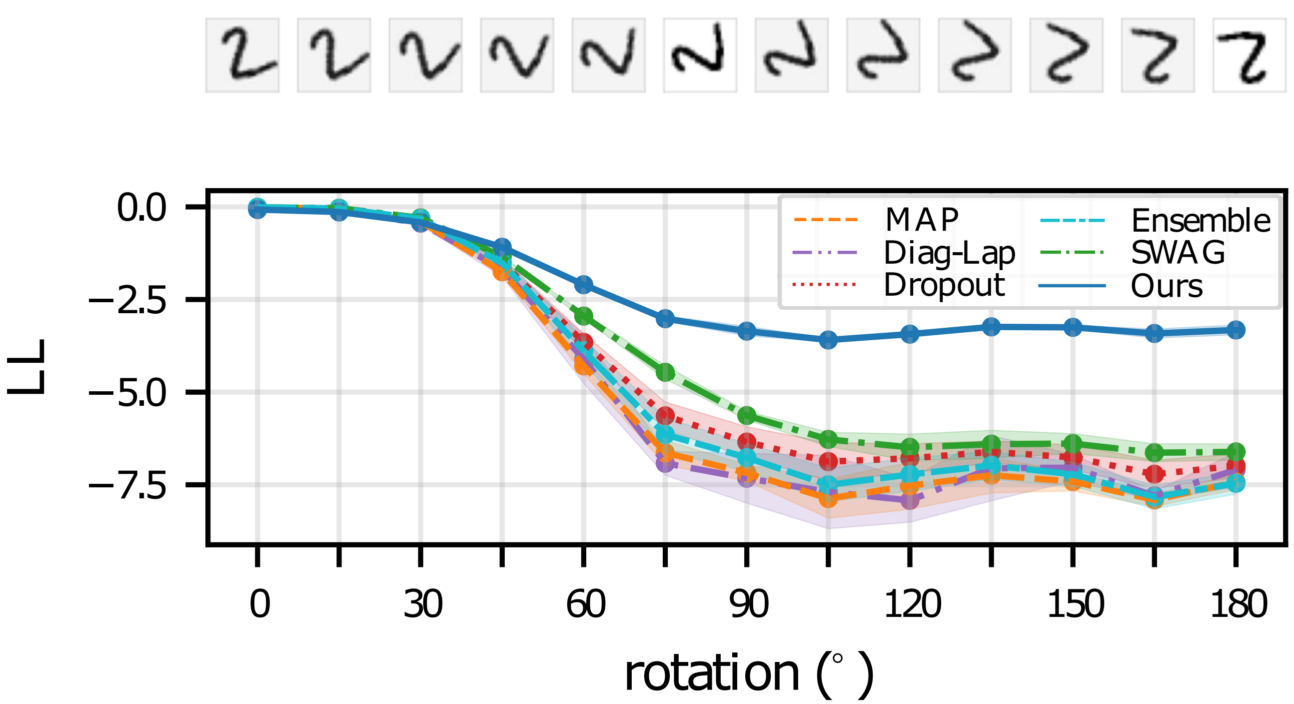
Image Class. under Distribution Shift

Model:
ResNet-18 with **11M** weights

Wasserstein subnetwork inference
subnet of just **42K (0.38%)** weights

Baselines:
- MAP
- Diagonal Laplace
- MC Dropout (Gal 2016)
- Deep Ensembles (Lakshminarayanan 2017)
- SWAG (Maddox 2019)

Rotated MNIST (Ovadia 2019)



Corrupted CIFAR10 (Ovadia 2019)

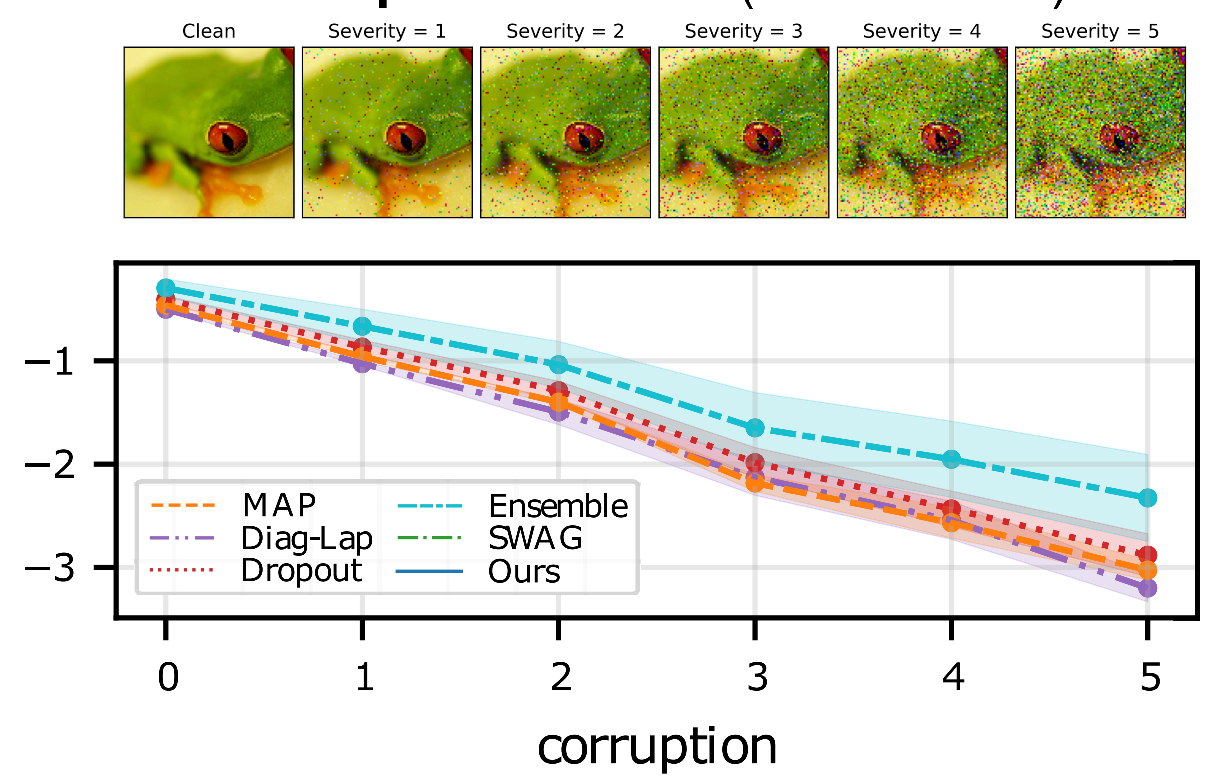


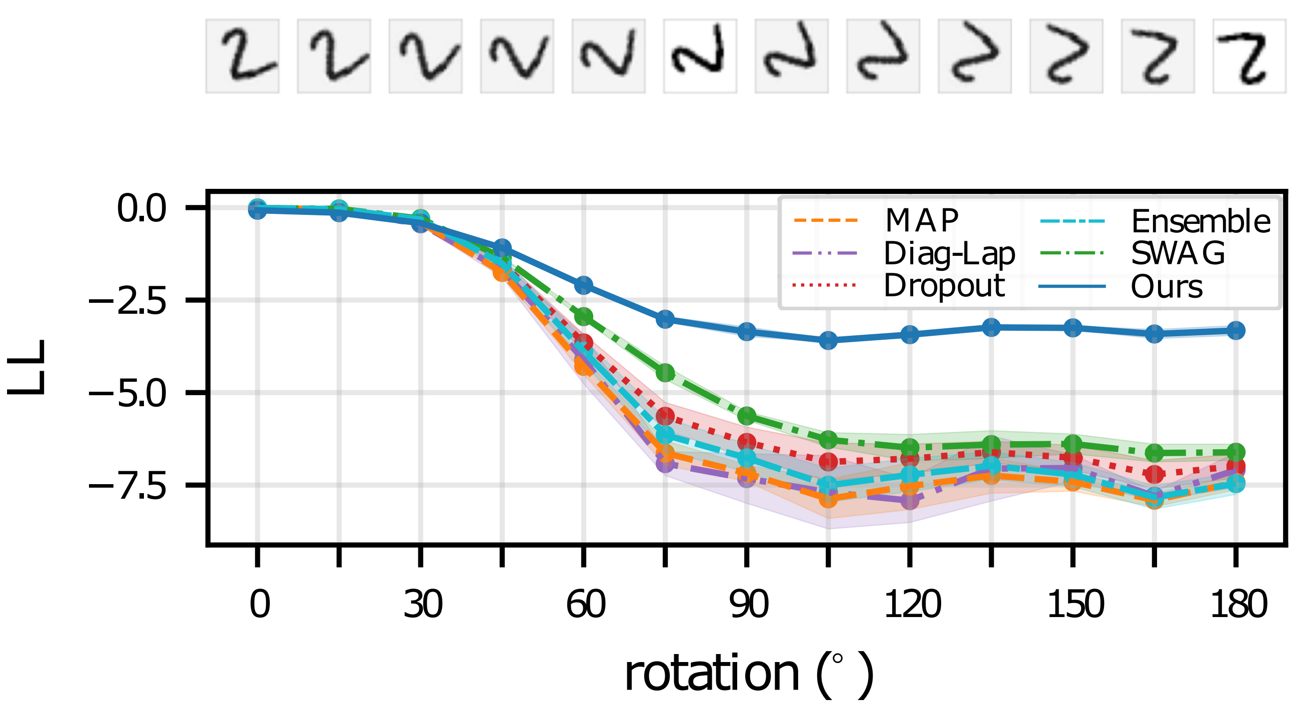
Image Class. under Distribution Shift

Model:
ResNet-18 with **11M** weights

Wasserstein subnetwork inference
subnet of just **42K (0.38%)** weights

Baselines:
- MAP
- Diagonal Laplace
- MC Dropout (Gal 2016)
- Deep Ensembles (Lakshminarayanan 2017)
- SWAG (Maddox 2019)

Rotated MNIST (Ovadia 2019)



Corrupted CIFAR10 (Ovadia 2019)

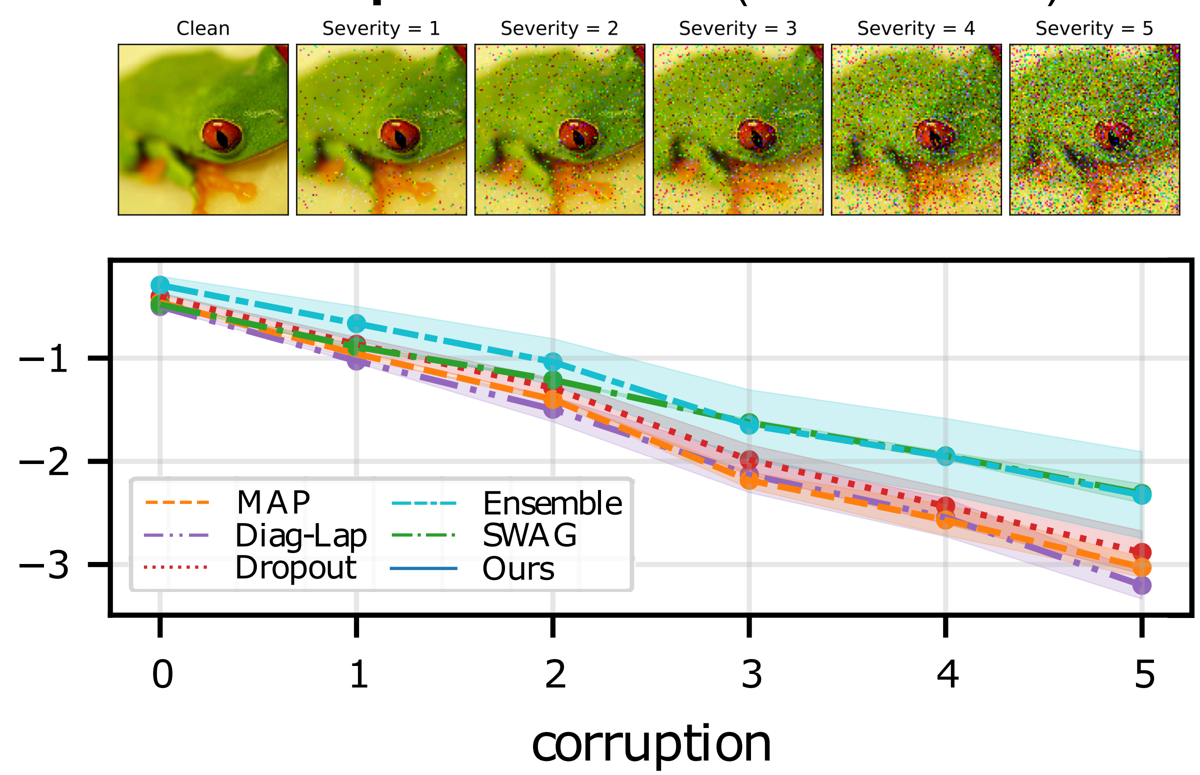


Image Class. under Distribution Shift

Model:
ResNet-18 with **11M** weights

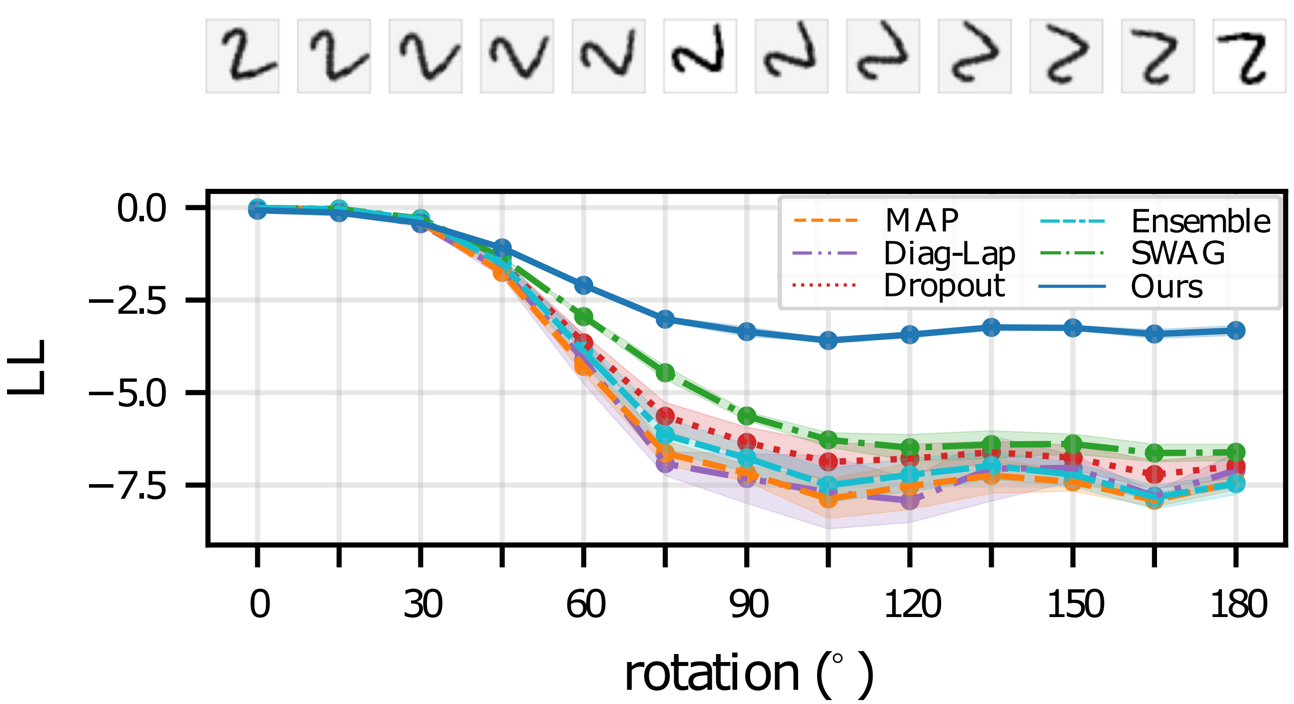
⬇

Wasserstein subnetwork inference
subnet of just **42K (0.38%)** weights

Baselines:

- MAP
- Diagonal Laplace
- MC Dropout (Gal 2016)
- Deep Ensembles (Lakshminarayanan 2017)
- SWAG (Maddox 2019)

Rotated MNIST (Ovadia 2019)



Corrupted CIFAR10 (Ovadia 2019)

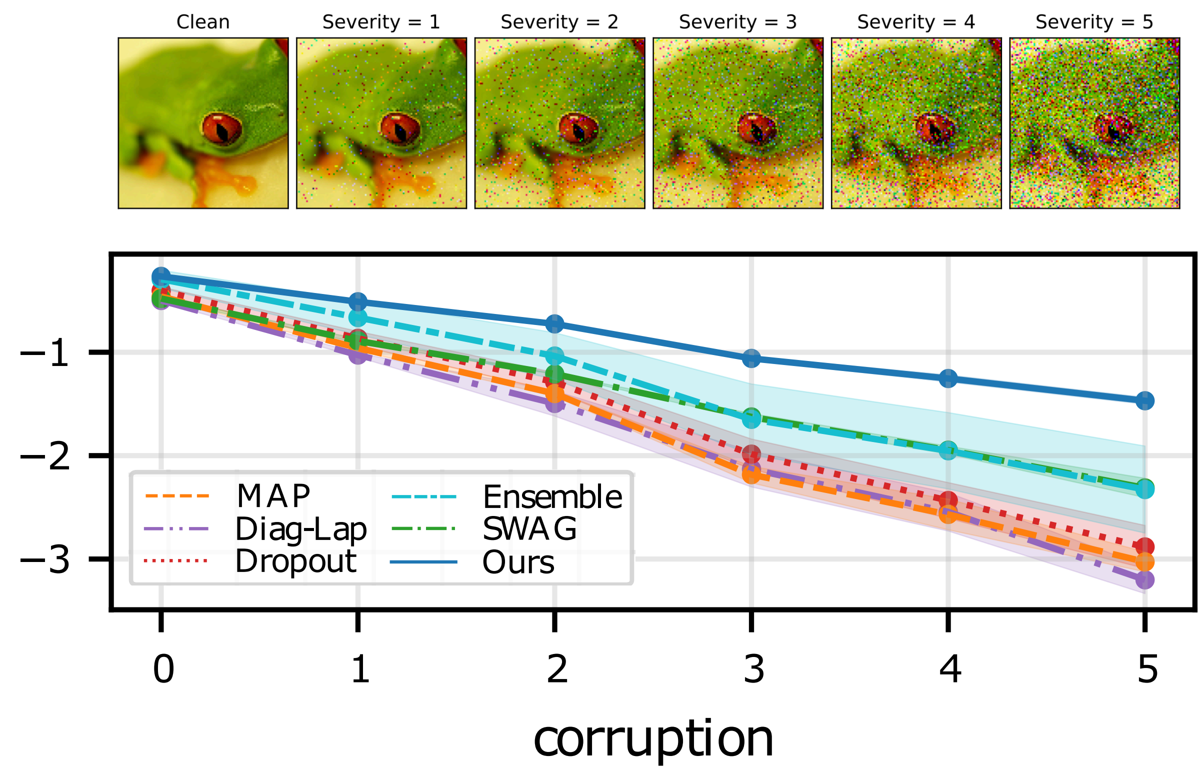


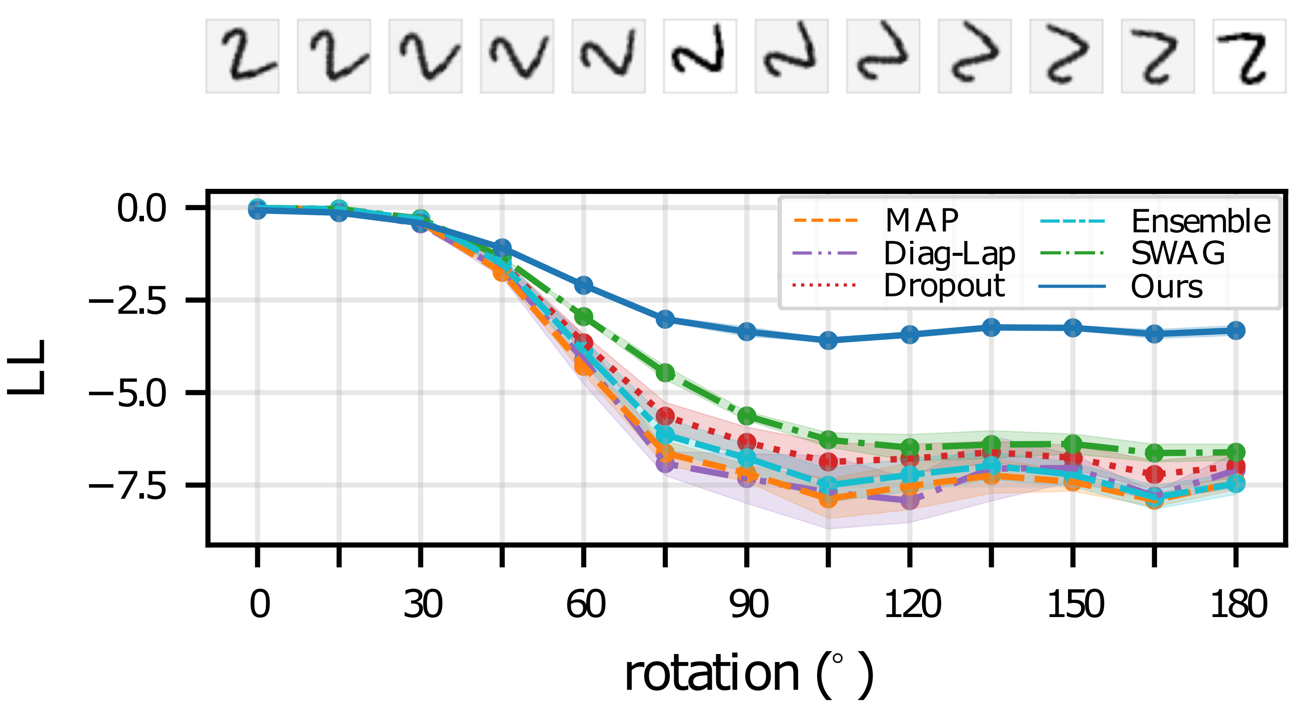
Image Class. under Distribution Shift

Model:
ResNet-18 with **11M** weights

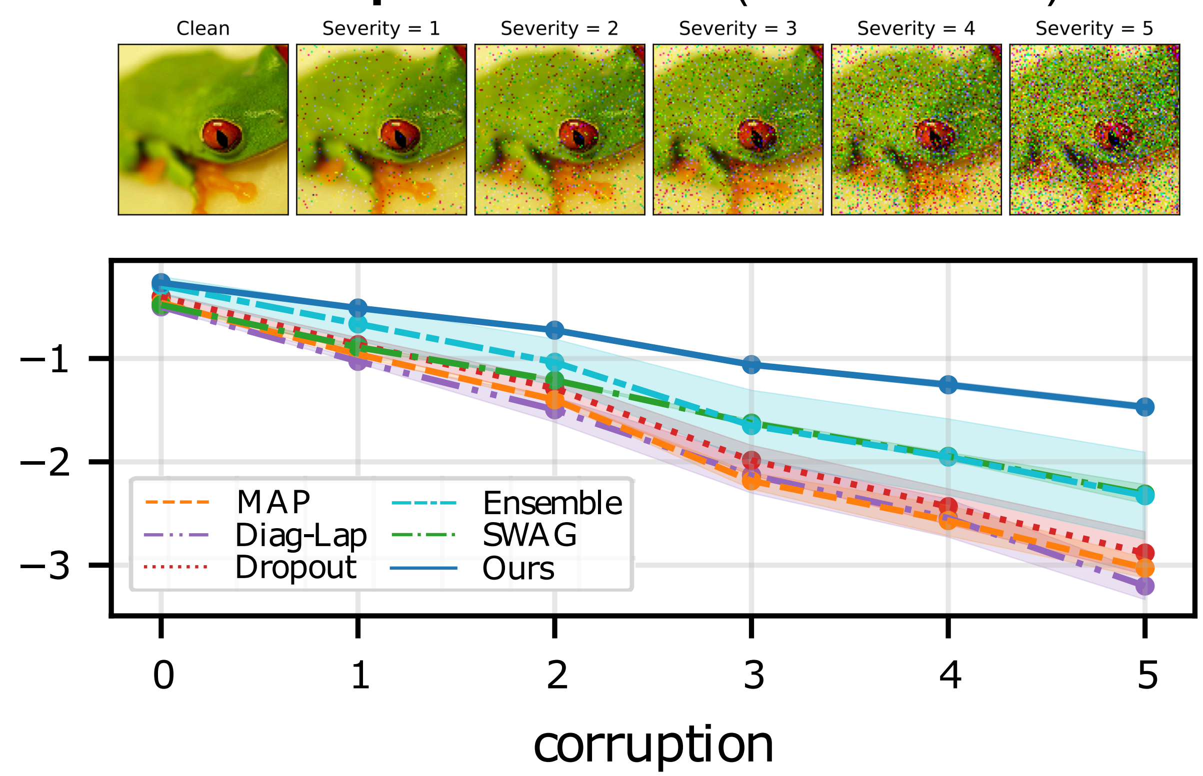
Wasserstein subnetwork inference
subnet of just **42K (0.38%)** weights

Baselines:
- MAP
- Diagonal Laplace
- MC Dropout (Gal 2016)
- Deep Ensembles (Lakshminarayanan 2017)
- SWAG (Maddox 2019)

Rotated MNIST (Ovadia 2019)



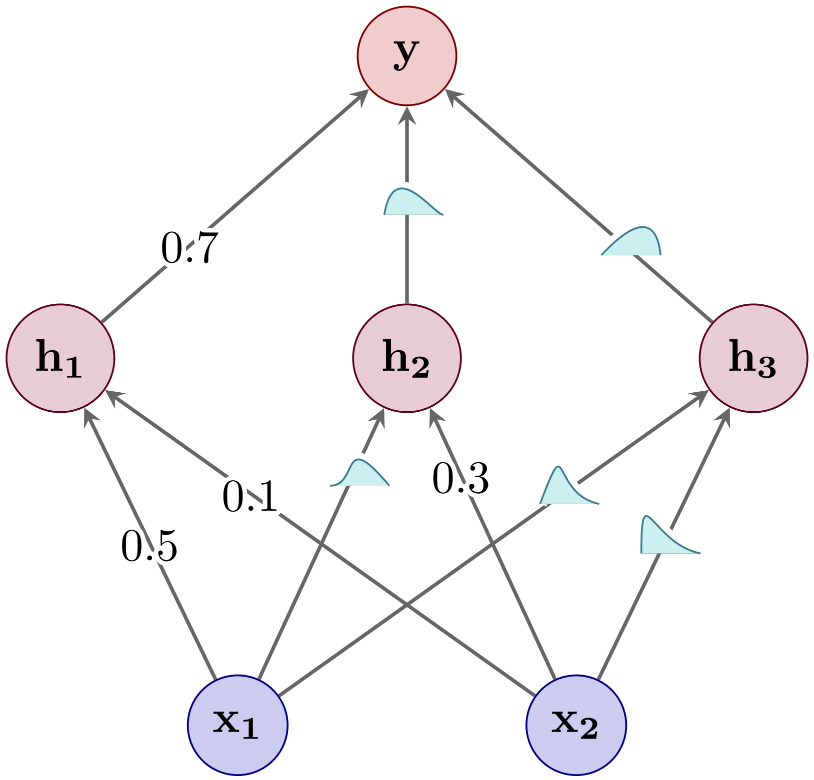
Corrupted CIFAR10 (Ovadia 2019)



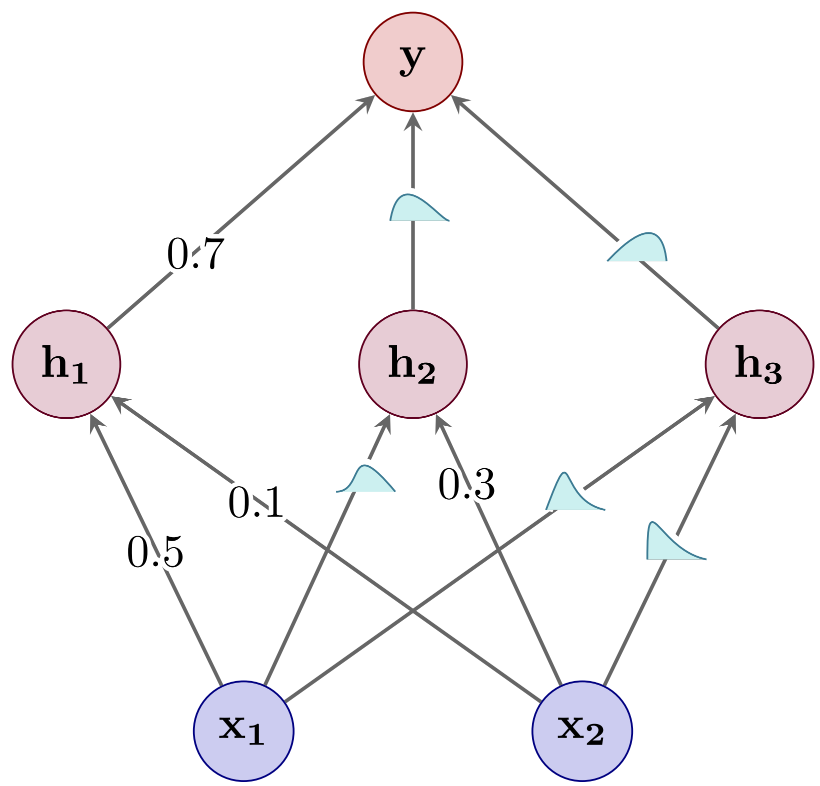
Subnet inference is **more robust to distribution shift** than popular baselines!

Take-Home Message

Take-Home Message

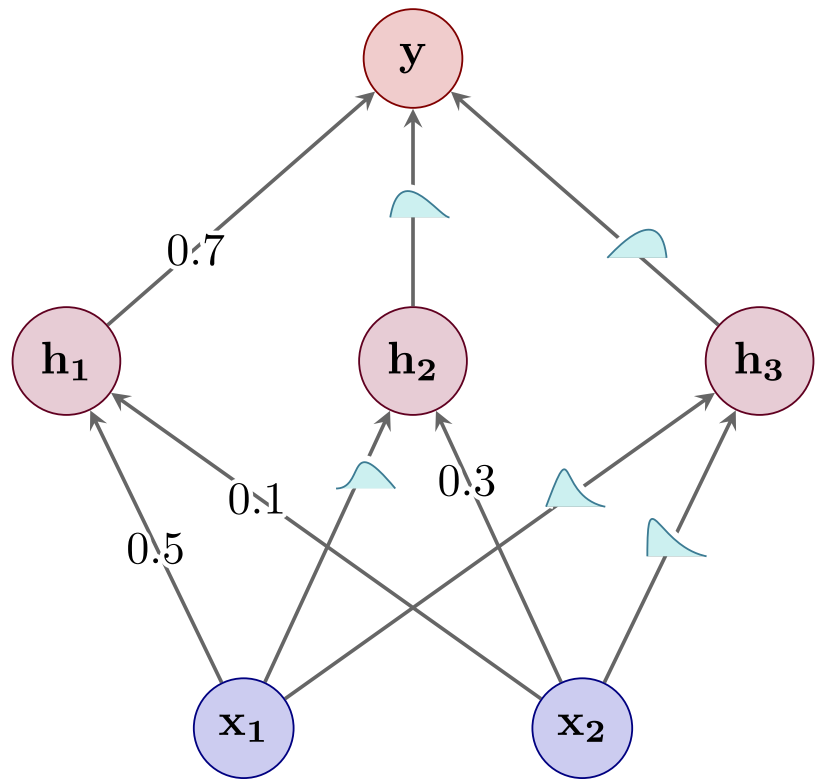


Take-Home Message



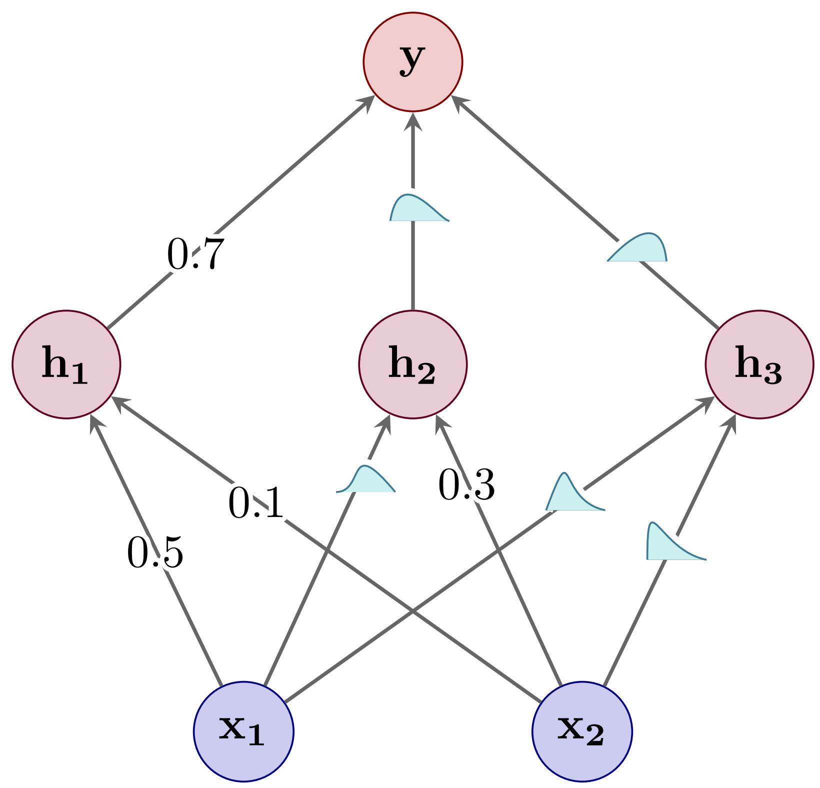
We propose a Bayesian deep learning method

Take-Home Message



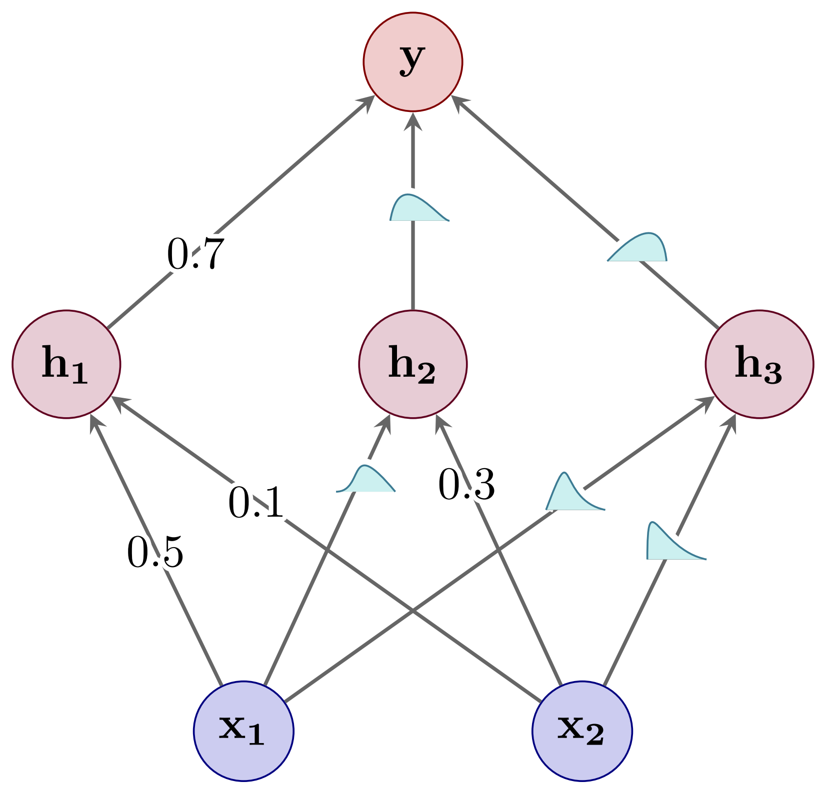
We propose a Bayesian deep learning method
that does ***expressive inference***

Take-Home Message



We propose a Bayesian deep learning method
that does ***expressive inference***
over a carefully chosen ***subnetwork***
within a neural network,

Take-Home Message



We propose a Bayesian deep learning method
that does ***expressive inference***
over a carefully chosen ***subnetwork***
within a neural network,
and show that this ***performs better*** than
doing crude inference over the full network.