Efficient uncertainty estimation with node-based Bayesian neural networks

Trung Trinh

Structure

- Part 1: Node-based Bayesian neural networks (node-based BNNs).

- Part 2: Tackling input corruptions with node-based BNNs.

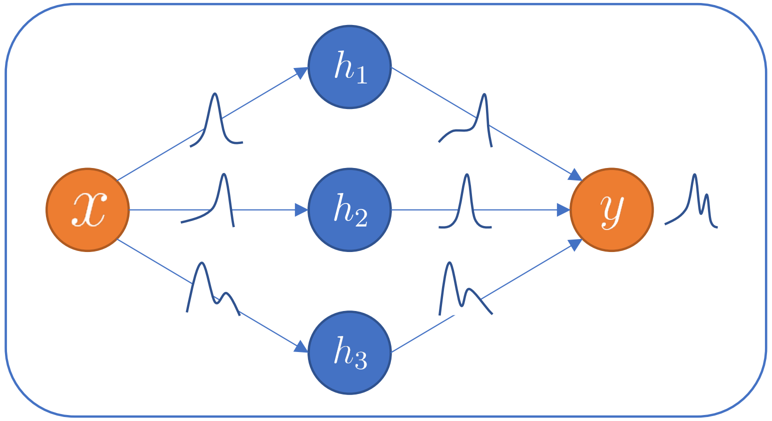
Part 1: Node-based Bayesian neural networks

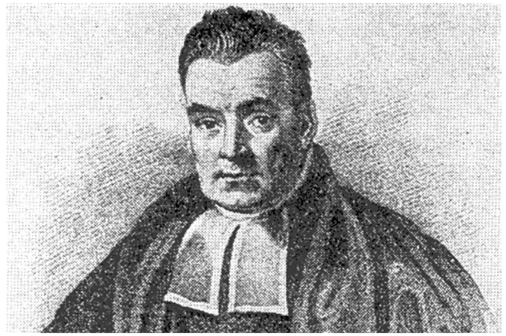
Uncertainty in Deep learning

- Accurate uncertainty estimations are crucial for utilizing machine learning in real world applications.
- Neural networks are overconfident predictors
  - because they cannot represent epistemic uncertainty.
- Two main approaches to represent epistemic uncertainty:
  - Deep ensembles: combine multiple maximum-a-posteriori (MAP) solutions.
  - Bayesian neural networks: probabilistic (Bayesian) representations of epistemic uncertainty.

Bayesian neural networks (BNNs)

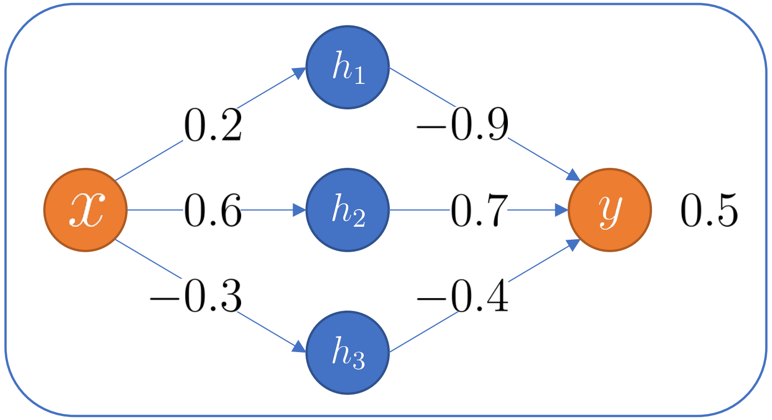
Bayesian neural network (BNN)





Thomas Bayes

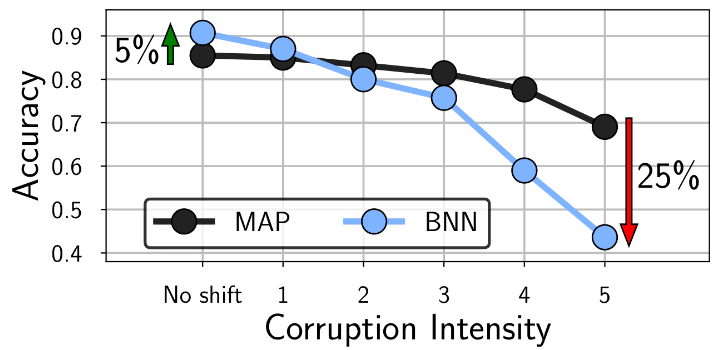
Deterministic neural network (DNN)



Neal (1996). Bayesian Learning for Neural Networks.

BNNs are challenging in practice

- Theoretically, BNNs have better performance than DNNs because they aggregate predictions from multiple hypotheses.
- Practically, however, the results are not great.
  - High-fidelity posterior approximations of BNNs (full batch HMC) are computationally expensive to obtain due to the sizes of these models.
    - Stochastic HMC or variational inference (VI) are used for inference, which requires "sharpening" the target posterior (cold posteriors) to obtain good approximations.[1]
  - Izmailov et al. (2021)[2] used 512 TPUv3 to perform full-batch HMC and discovered that BNNs did worse than DNNs under input corruptions



ResNet-20, CIFAR-10-C

[1] Wenzel et al. (2020). How Good is the Bayes Posterior in Deep Neural Networks Really?
[2] Izmailov et al. (2021). What are Bayesian neural network posteriors really like?

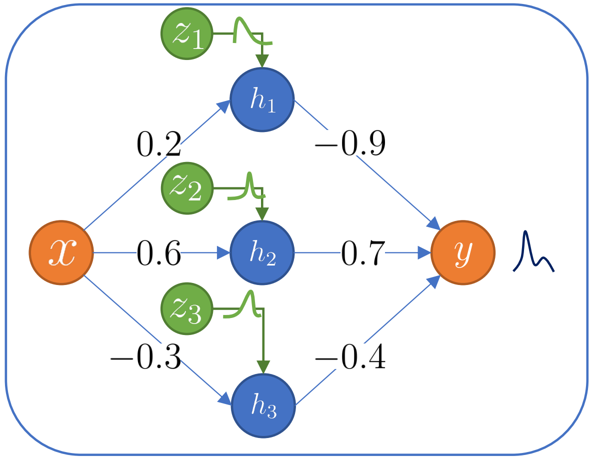
Alternatives to weight-based BNNs

- Function-space inference. (Wang et al, 2019; Sun et al, 2019; D'Angelo et al, 2021)
- Architecture-space inference.
  - Depth uncertainty NNs (Antorán et al, 2020).
- **Activation-space inference (node-based BNNs):**
  - Dropout. (Gal et al, 2016)
  - Rank-1 BNNs. (Dusenberry et al, 2020; Trinh et al, 2022)

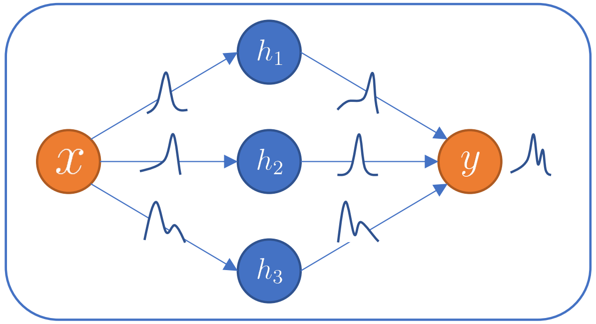
[1] Wang et al. (2019). Function space particle optimization for Bayesian neural networks.
[2] Sun et al. (2019). Functional variational Bayesian neural networks.
[3] D'Angelo et al. (2021). Repulsive Deep Ensembles are Bayesian.
[4] Antorán et al. (2020). Depth Uncertainty in Neural Networks.
[5] Gal et al. (2016). Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning
[6] Dusenberry et al. (2020). Efficient and Scalable Bayesian Neural Nets with Rank-1 Factors.
[7] Trinh et al. (2022). Tackling covariate shift with node-based Bayesian neural networks.

Node-based Bayesian neural networks

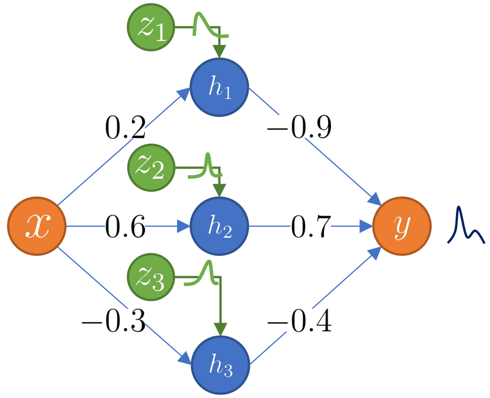
Node-BNNs

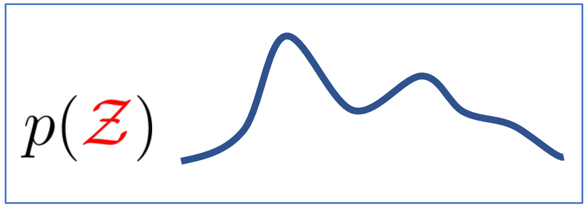


Weight-BNNs



Node-based Bayesian neural networks





An L-layer node-BNN with latent variables $\mathcal{Z} = \{z^{(\ell)}\}_{\ell=1}^{L}$, $z^{(\ell)} = (z_{in}^{(\ell)}, z_{out}^{(\ell)})$:

Pre-activation output

Previous layer's output

Latent node variables

$$f_{in}^{(\ell)} = f^{(\ell-1)}(x; \mathcal{Z}) \circ z_{in}^{(\ell)}$$

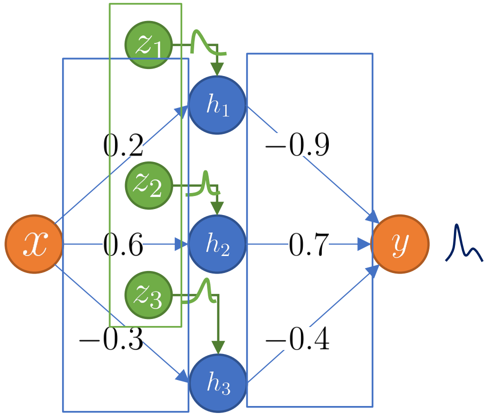
$$h^{(\ell)} = W^{(\ell)} f_{in}^{(\ell)} + b^{(\ell)}$$

$$f^{(\ell)} = \sigma(h^{(\ell)} \circ z_{out}^{(\ell)})$$

For $\mathcal{Z} \sim p(\mathcal{Z})$:

$$f(x; \mathcal{Z}) = f^{(L)}(x; \mathcal{Z})$$

Node-based Bayesian neural networks



Network	Layers	Parameters weights	nodes	w/n ratio
LeNet	5	42K	23	1800x
AlexNet	8	61M	18,307	3300x
VGG16-small	16	15M	5,251	2900x
VGG16-large	16	138M	36,995	3700x
ResNet50	50	26M	24,579	1000x
WideResNet-28x10	28	36M	9,475	3800x

Two types of parameters:

1. Weights and biases $\theta = \{(W^{(\ell)}, b^{(\ell)})\}_{\ell=1}^{L}$

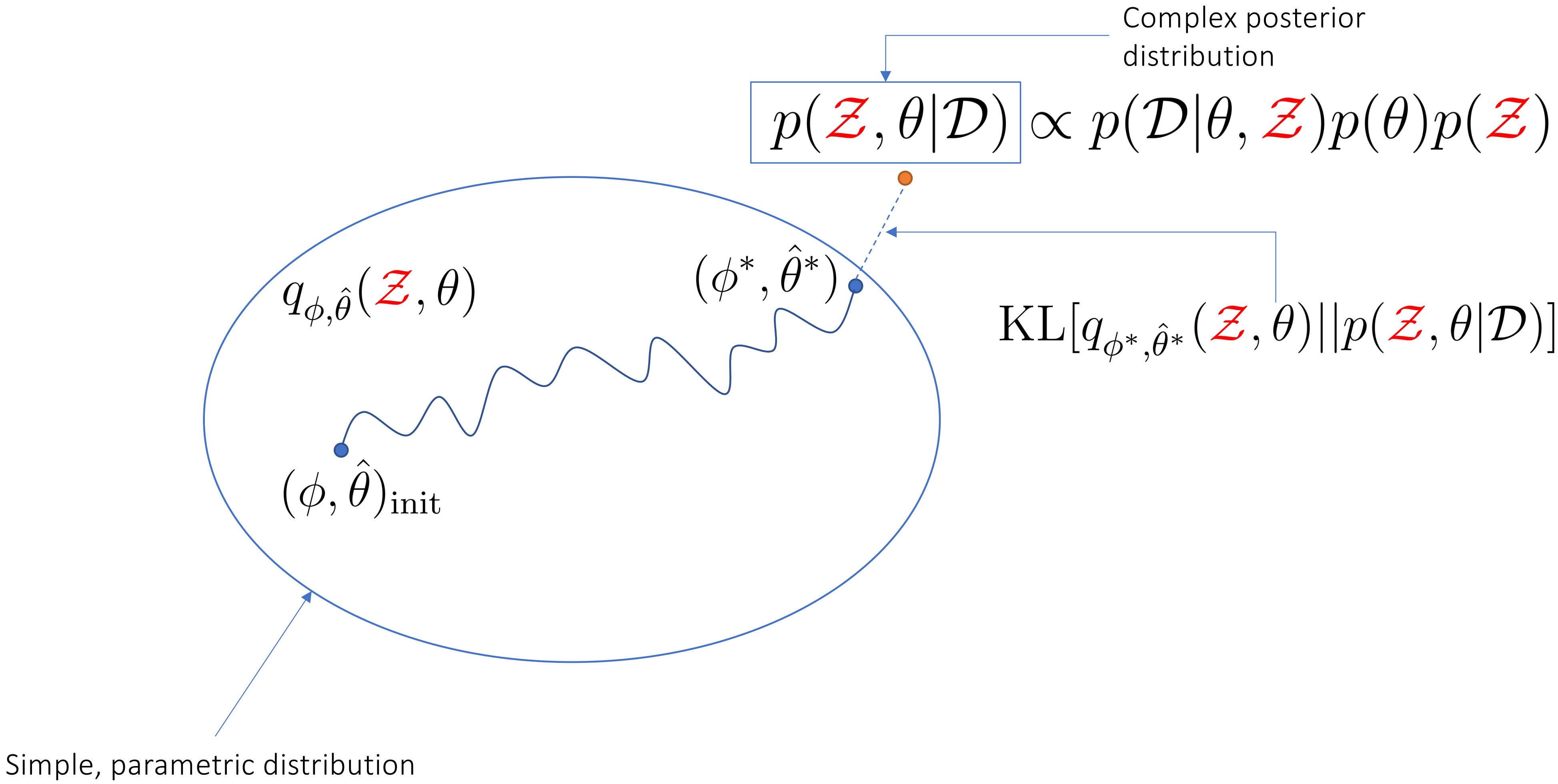
   ➔ Find a MAP estimate.

2. Latent node variables $\mathcal{Z} = \{z^{(\ell)}\}_{\ell=1}^{L}$

   ➔ Infer the posterior distribution.

➔ Node BNNs are efficient alternatives to standard weight-based BNNs.

Training a node BNN: Variational inference



Blei et al. (2017). Variational Inference: A Review for Statisticians.

Variational posterior

$$q_{\phi,\hat{\theta}}(\mathcal{Z}, \theta) = q_{\hat{\theta}}(\theta) q_{\phi}(\mathcal{Z})$$

$$= \delta(\theta - \hat{\theta}) q_{\phi}(\mathcal{Z})$$

Dirac delta measure
(for MAP estimation)

Mixture of Gaussians

MAP estimation of θ

Training objective

We find $(\hat{\theta}, \phi)$ maximizing the following objective using SGD:

expected log-likelihood

$$\mathcal{L}(\hat{\theta}, \phi) = \mathbb{E}_{q_\phi(\mathcal{Z})}[\log p(\mathcal{D}|\hat{\theta}, \mathcal{Z})] - \mathrm{KL}[q_\phi(\mathcal{Z})||p(\mathcal{Z})] + \log p(\hat{\theta})$$

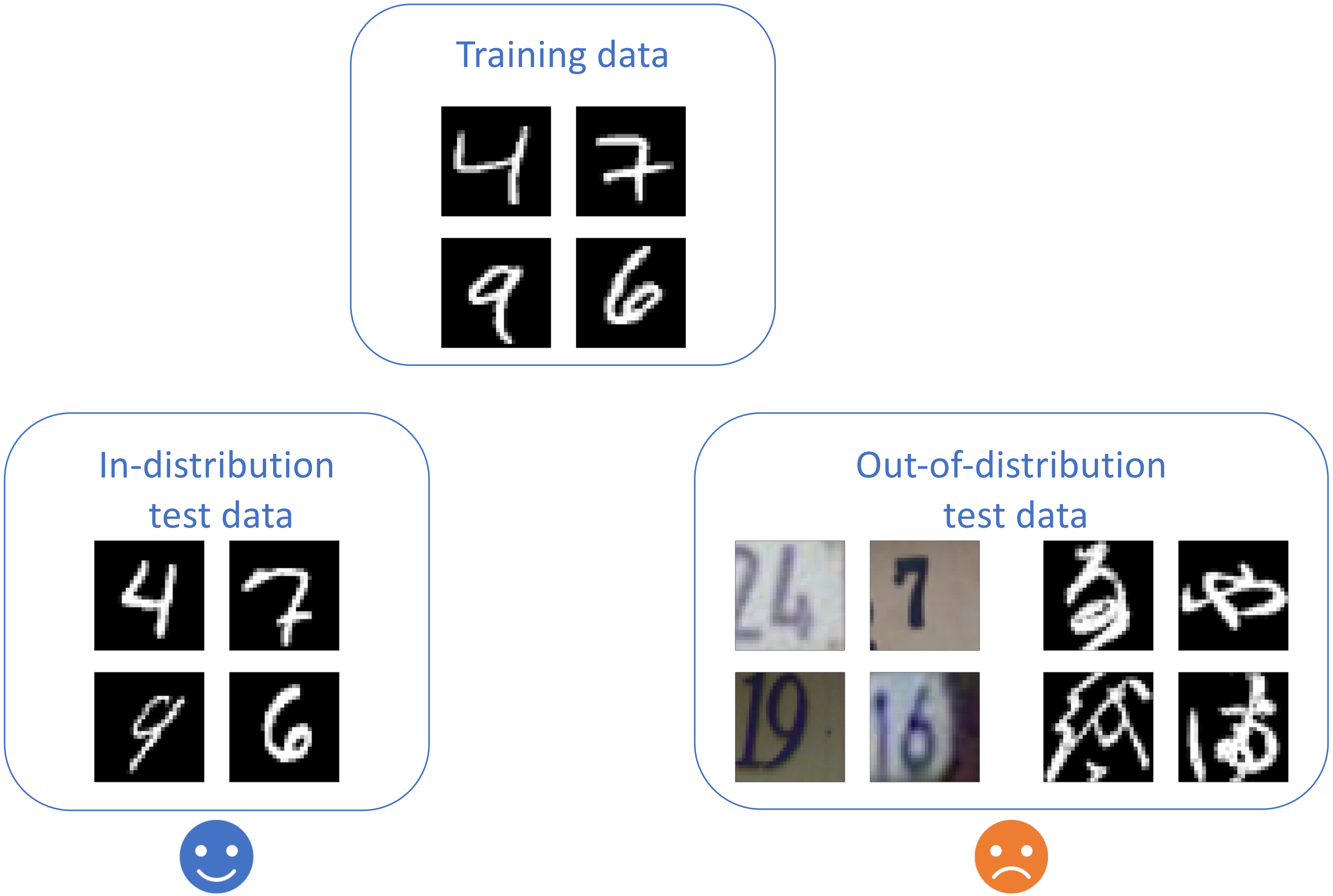
Evidence lower-bound (ELBO)

KL divergence

log prior

Part 2: Tackling input corruptions with node-based BNNs

Covariate shift

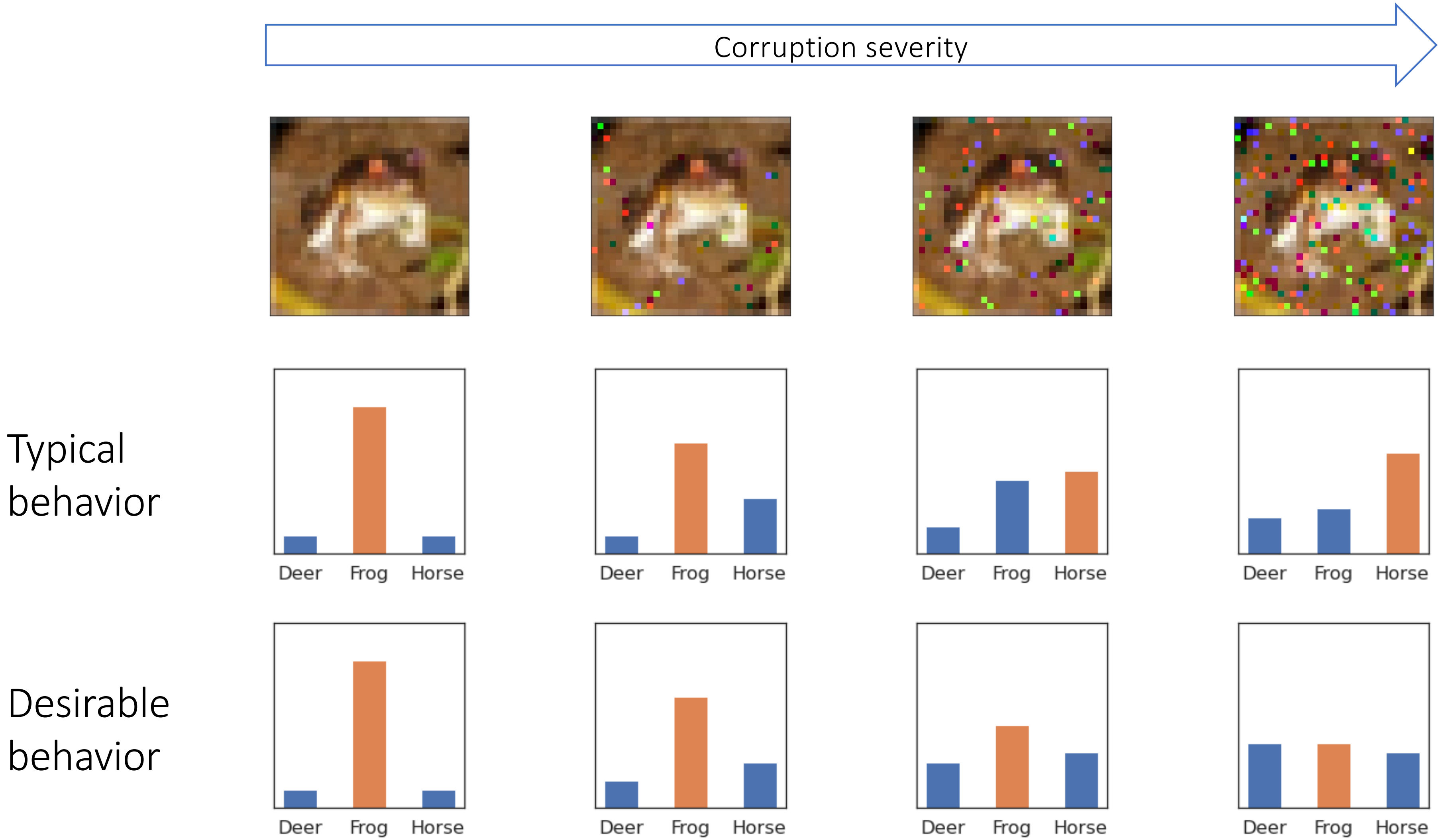


Shift due to corruptions

Shifts due to **corruptions**

Hendrycks & Dietterich (2019). Benchmarking Neural Network Robustness to Common Corruptions and Perturbations.

Neural networks under input corruptions



Neural networks under input corruptions

$$\mathbf{x}^c = \mathbf{x} + \mathbf{g}^{(0)}(\mathbf{x})$$

Corrupted input

Original input

Corruption function
(e.g., shot noise)

The input corruption propagates through the layers, generating a shift in the activation of each layer.

Corrupted output

Clean output

Activation shift

$$\mathbf{g}^{(\ell)}(\mathbf{x}) = \mathbf{f}^{(\ell)}(\mathbf{x}^c) - \mathbf{f}^{(\ell)}(\mathbf{x})$$

$$\approx \mathbf{J}_\sigma \left[\mathbf{h}^{(\ell)}(\mathbf{x})\right] \left(\mathbf{W}^{(\ell)} \mathbf{g}^{(\ell-1)}(\mathbf{x})\right)$$

Jacobian

Pre activation
output

Neural networks under input corruptions

Corrupted output — Clean output

Activation shift →

$$\mathbf{g}^{(\ell)}(\mathbf{x}) = \mathbf{f}^{(\ell)}(\mathbf{x}^c) - \mathbf{f}^{(\ell)}(\mathbf{x})$$

$$\approx \mathbf{J}_\sigma \left[\mathbf{h}^{(\ell)}(\mathbf{x})\right] \left(\mathbf{W}^{(\ell)} \mathbf{g}^{(\ell-1)}(\mathbf{x})\right)$$

Jacobian

Pre activation output

The activation shift depends on:

1) The input: $\mathbf{x}$
2) The corruption: $\mathbf{g}^{(0)}$
3) The weights and biases: $\theta = \{(W^{(\ell)}, b^{(\ell)})\}_{\ell=1}^{L}$

Node-based BNNs simulate shifts during training

$$\mathcal{Z} = \{z^{(\ell)}\}_{\ell=1}^{L} \qquad q(\mathcal{Z})$$

For a sample $\hat{\mathcal{Z}} \sim q(\mathcal{Z})$, define the corresponding simulated shift at one specific layer as:

$$\hat{\mathbf{g}}^{(\ell)}(\mathbf{x}) = \mathbf{f}^{(\ell)}(\mathbf{x}; \hat{\mathcal{Z}}) - \mathbb{E}_{q(\mathcal{Z})}\left[\mathbf{f}^{(\ell)}(\mathbf{x}; \mathcal{Z})\right]$$

The simulated shifts are also functions of the weights and input, similar to shifts caused by actual corruptions.

Node-based BNNs simulate shifts during training

expected log-likelihood

$$\mathcal{L}(\hat{\theta}, \phi) = \mathbb{E}_{q_\phi(\mathcal{Z})}[\log p(\mathcal{D}|\hat{\theta}, \mathcal{Z})] - \mathrm{KL}[q_\phi(\mathcal{Z})||p(\mathcal{Z})] + \log p(\hat{\theta})$$

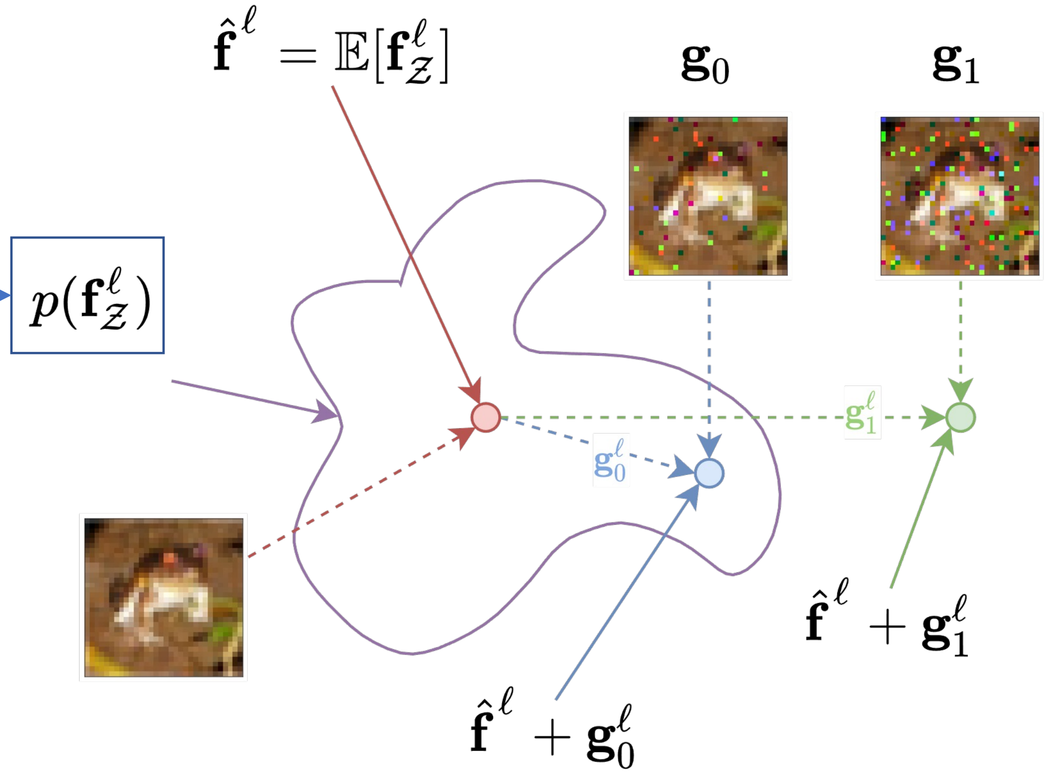
The expected log-likelihood term of the ELBO enforces the model to achieve low loss on the training data despite each layer output being corrupted by noise from $q(\mathcal{Z})$.

➔The model is robust against simulated activation shifts caused by $q(\mathcal{Z})$.

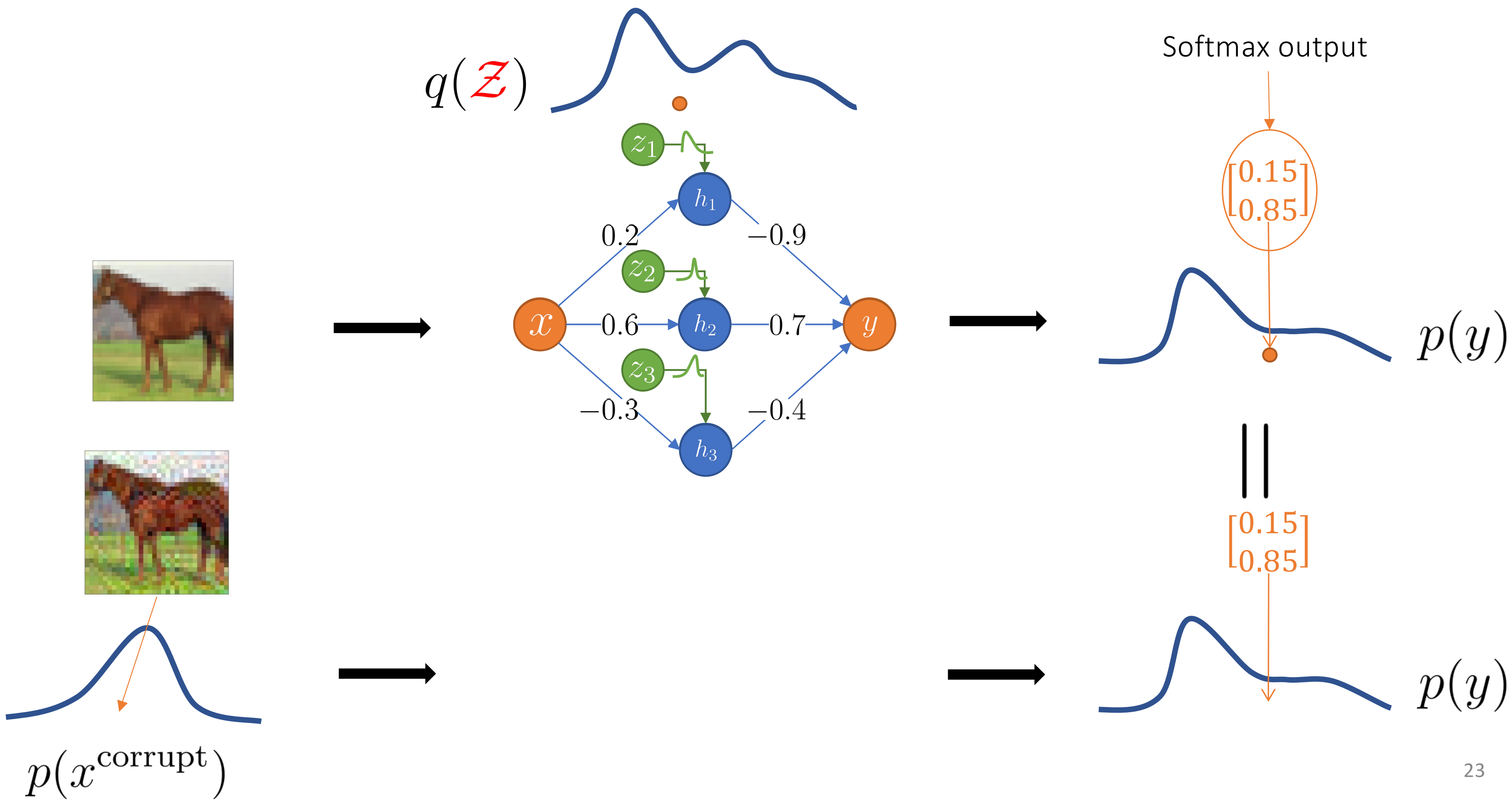
➔The model is robust against activation shifts caused by actual corruptions.

Node-based BNNs simulate shifts during training

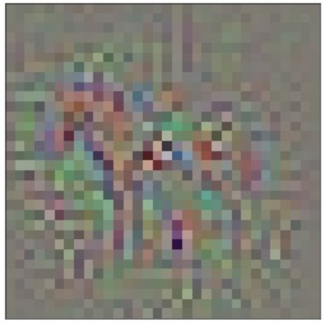
Distribution of layer activation induced by $q(\mathcal{Z})$



The latent posterior $q(\mathcal{Z})$ induces a distribution of corruptions in input space $p(x^{\text{corrupt}})$

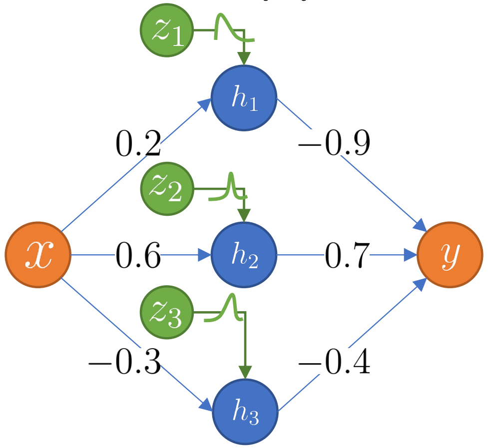


Approximating the implicit corruption

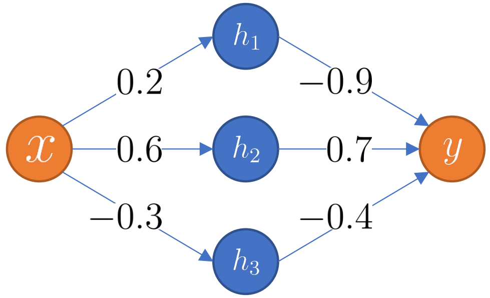
 $=$  $+$ 

$$x_{corrupt} = x + m$$

Approximating the implicit corruption



$$f(x; \mathcal{Z})$$



$$\hat{f}(x) = f(x; \mathcal{Z} = \mathbf{1})$$

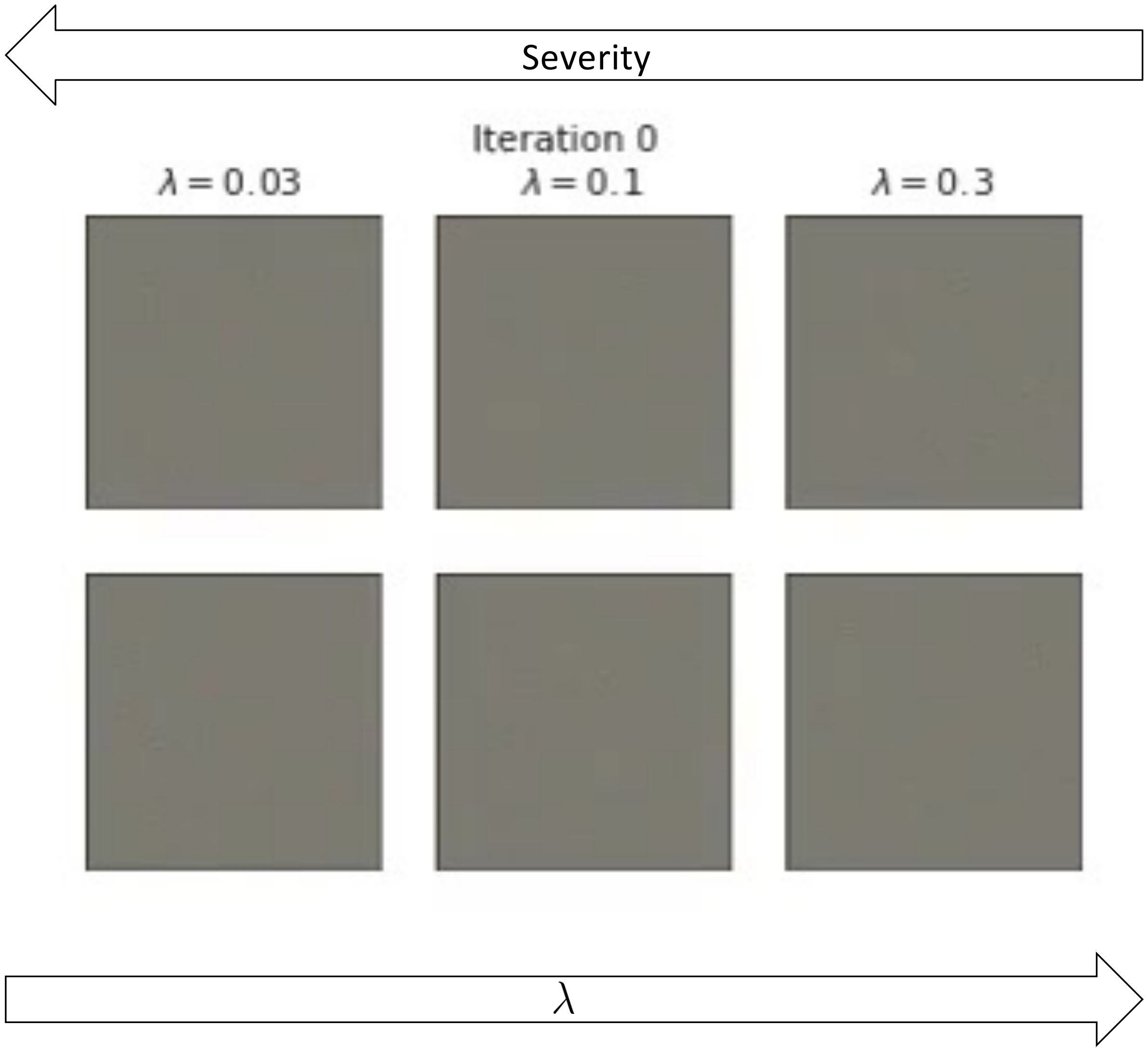
Given $\mathcal{Z} \sim q(\mathcal{Z})$, approximating m minimizing the following loss function using GD:

$$\frac{1}{2}\left|\left|f(x; \mathcal{Z}) - \hat{f}(x+m)\right|\right|_2^2 + \frac{\lambda}{2}||m||_2^2$$

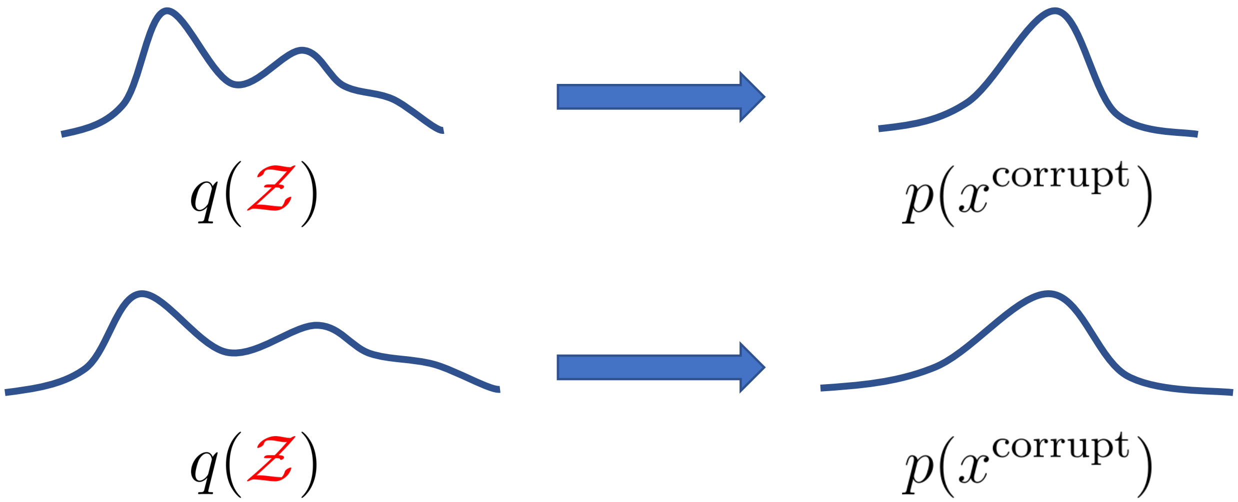
Output matching

L2-regularization

Example of implicit corruptions



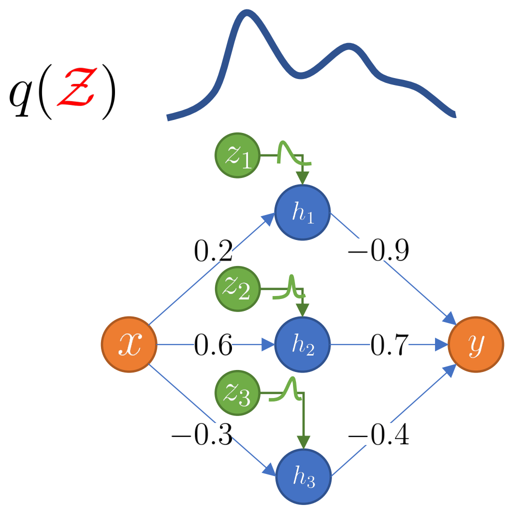
Entropy of latent variables and implicit corruptions



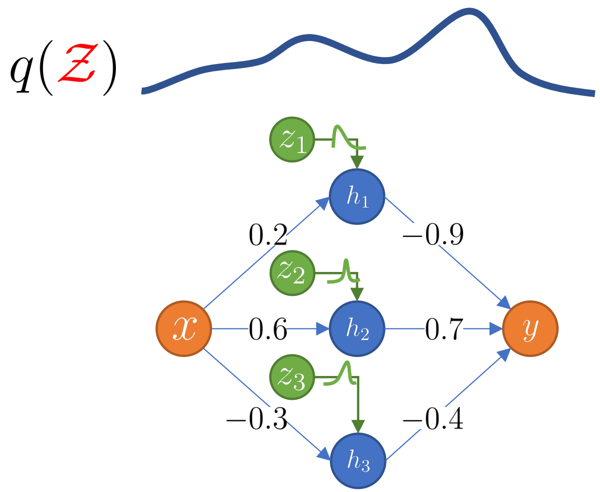
We hypothesize that:

1. Increasing the entropy of the latent variables $\mathcal{Z}$ increase the diversity of the implicit corruptions.
2. By training under more diverse implicit corruptions, node-based BNNs become more robust against natural corruptions.

Is it true that “higher entropy = more robust node-based BNNs”?



Low entropy model



High entropy model

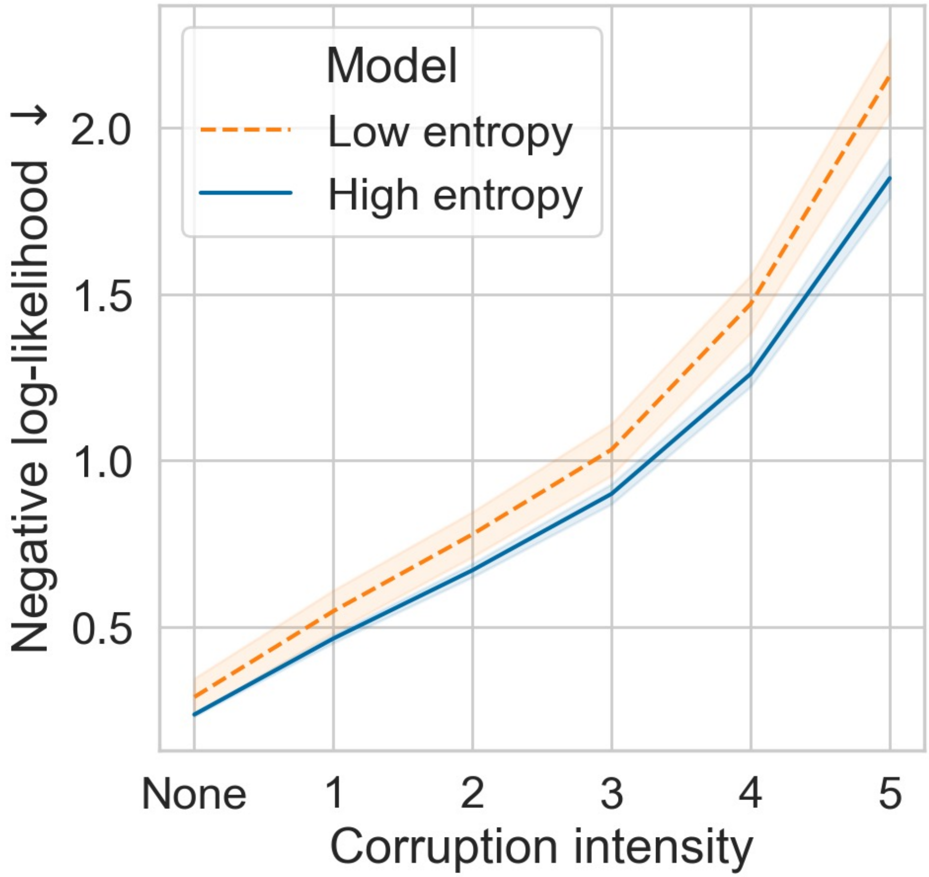
Same ConvNet architecture

Train on CIFAR-10

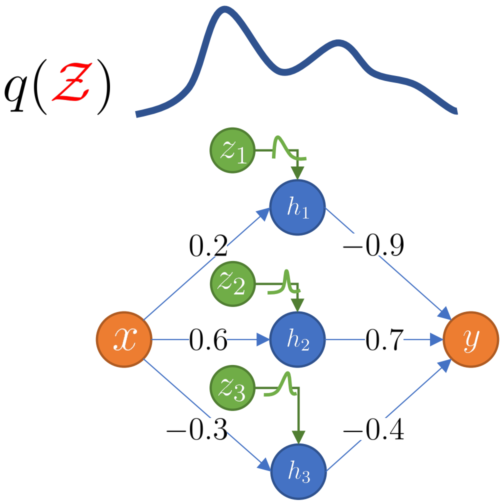
Test on CIFAR-10-C

Is it true that “higher entropy = more robust node-based BNNs”?

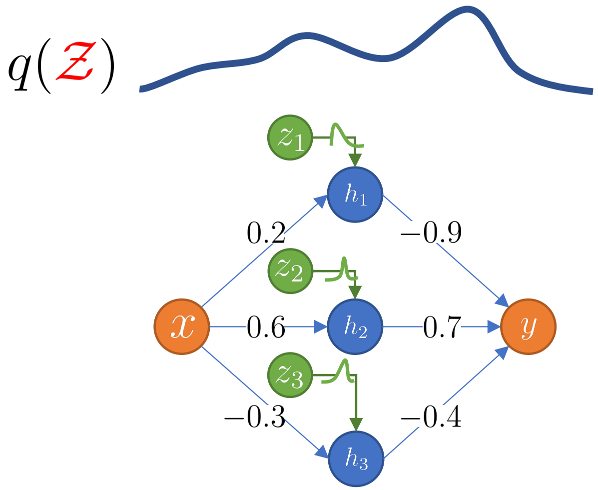
YES!!!



Is a model robust against its own corruptions?



Low entropy model

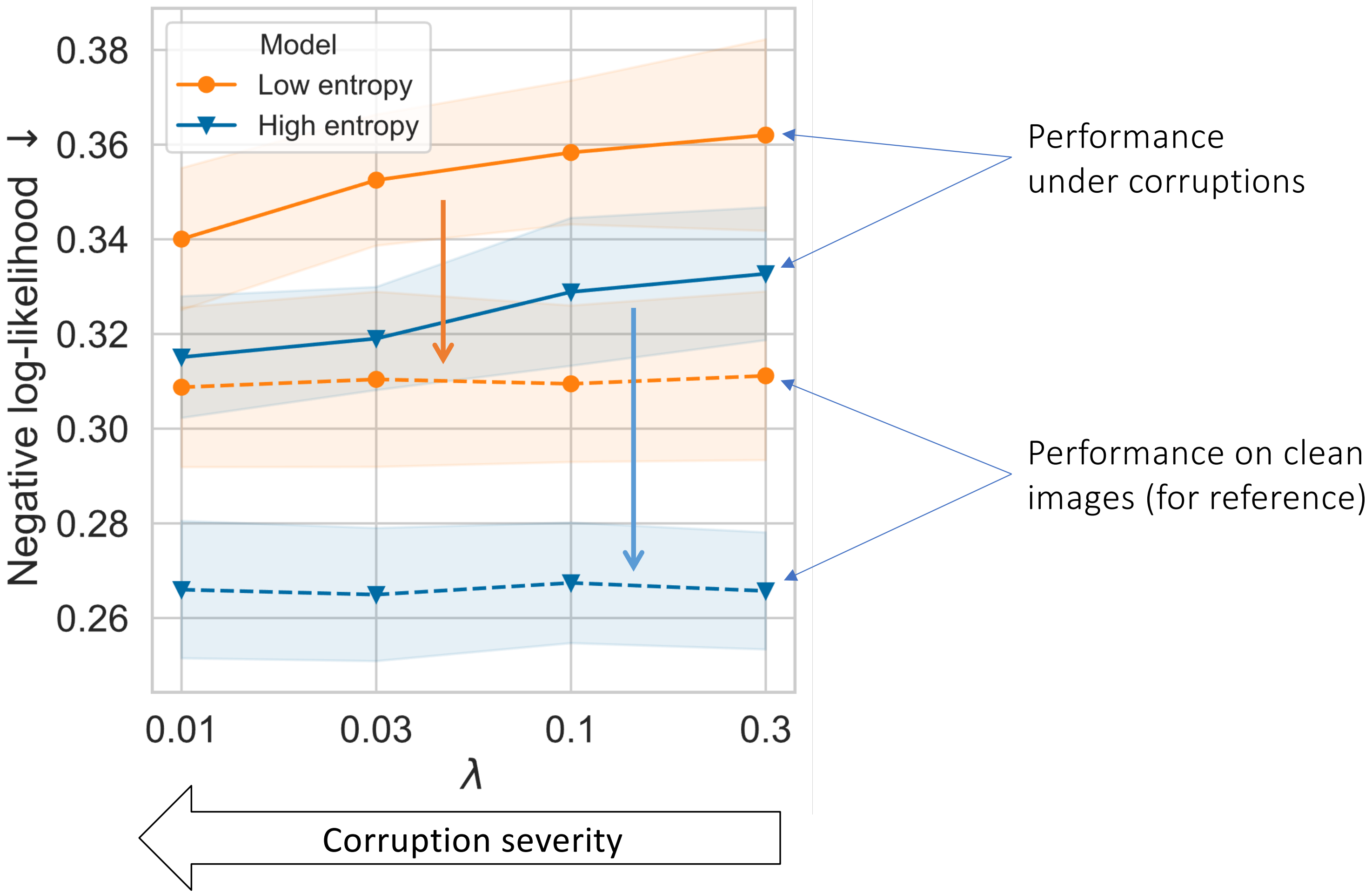


High entropy model

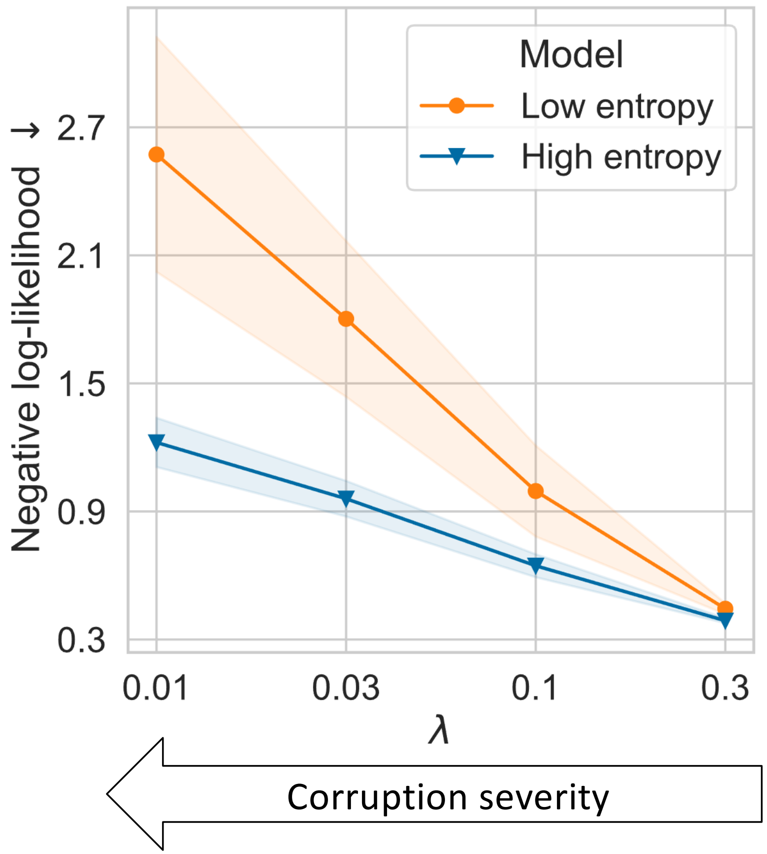
We use each model to generate a set of corrupted test images, then evaluate each model on its own generated corruptions.

Is a model robust against its own corruptions?

YES (in this small experiment)



How robust is a model against the other model's corruptions?



Increasing the latent entroy: Entropic regularization

$$\gamma > 0$$

$$\mathcal{L}_\gamma(\hat{\theta}, \phi) = \mathcal{L}(\hat{\theta}, \phi) + \gamma \mathbb{H}[q_\phi(\mathcal{Z})]$$

The γ-ELBO

The original ELBO

The γ entropy
(increase the entropy
of the latent variables)

The γ – ELBO = tempered posterior

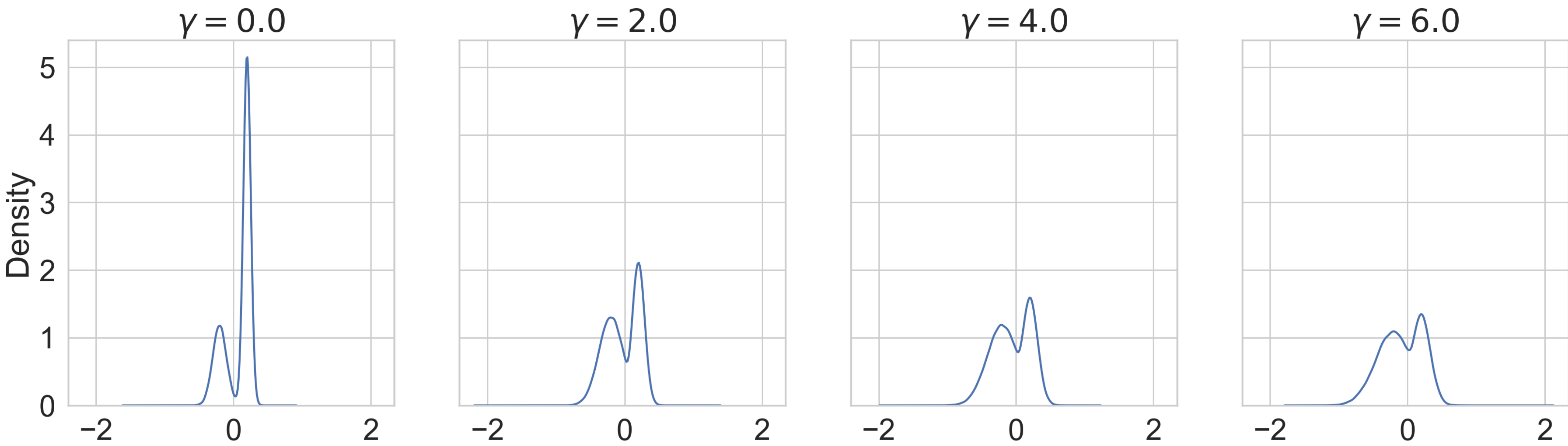
Maximizing the γ – ELBO is equivalent to minimizing:

$$\mathrm{KL}[q_{\phi,\hat{\theta}}(\mathcal{Z},\theta)||p_\gamma(\mathcal{Z},\theta|\mathcal{D})]$$

$$p_\gamma(\mathcal{Z},\theta|\mathcal{D}) \propto p(\mathcal{D}|\mathcal{Z},\theta)^{\frac{1}{\gamma+1}} p(\mathcal{Z},\theta)^{\frac{1}{\gamma+1}}$$

Temperature $\tau = \gamma + 1$

Effects of $\gamma > 0$ on the target posterior.



$\gamma > 0$ → temperature $\tau > 1$ → 'hot' posterior

A justification for hot posterior

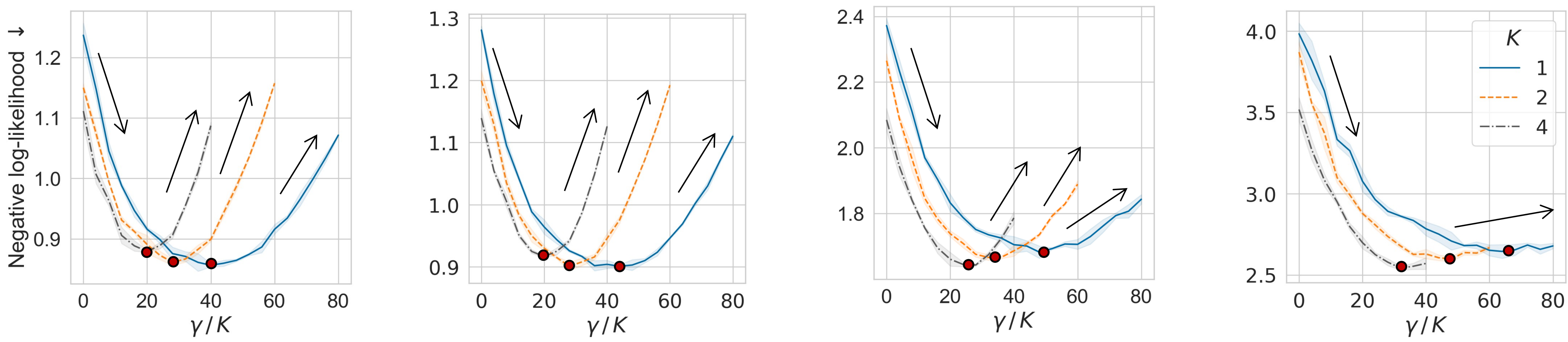
1) Neither the model definition or the dataset accounts for input corruptions.

2) Variational inference only converges to a posterior whose entropy is calibrated for the variability in the training data.

➔ By increasing the entropy of the posterior, we also account for the variability caused by input corruptions.

Ablation study

Effects of γ on corruption robustness



Validation

Test

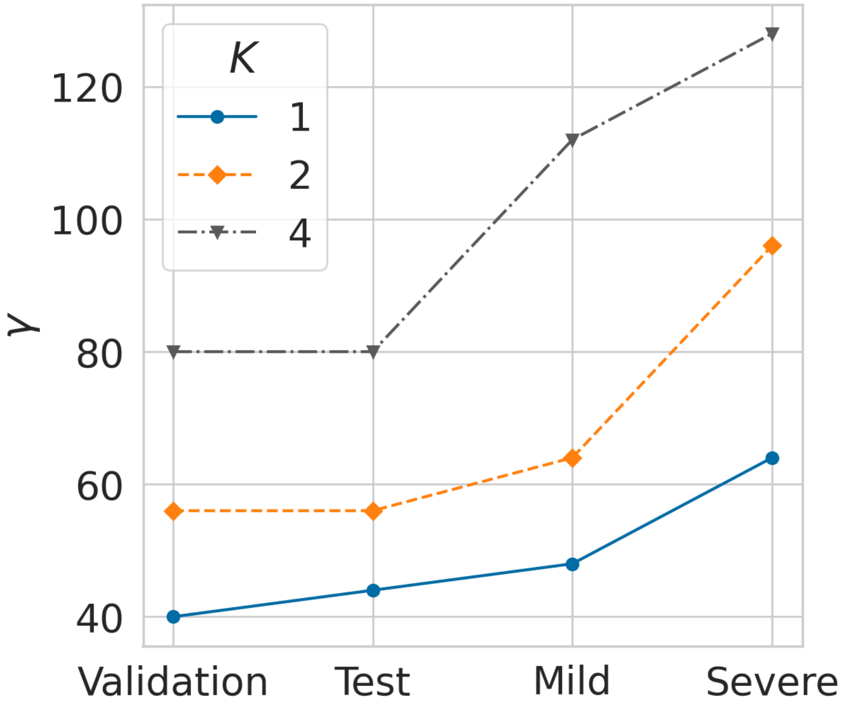
Mild corruption

Severe corruption

VGG16 / CIFAR-100. Test on CIFAR-100-C

K: number of Gaussian components in $q_\phi(\mathcal{Z})$.

Effects of γ on corruption robustness



Optimal γ

More severe corruptions require higher optimal γ

Robust learning under label noise

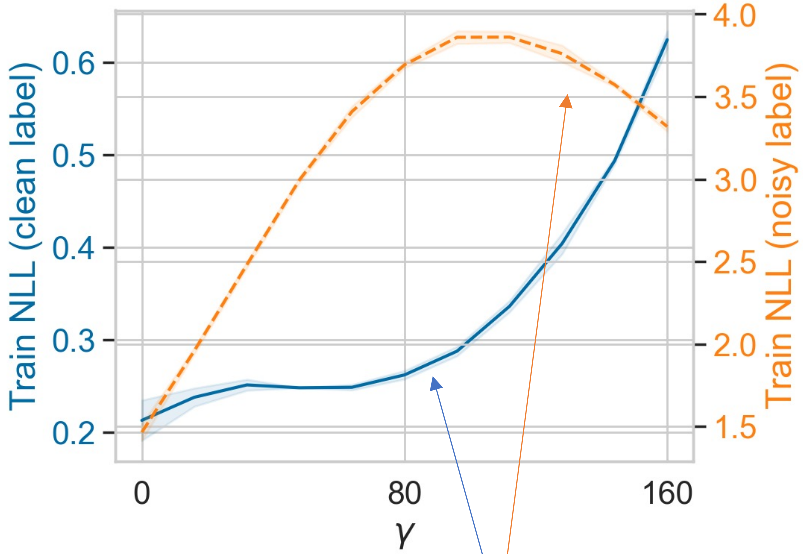
Memorizing random labels **is harder than** learning generalizable patterns[1]

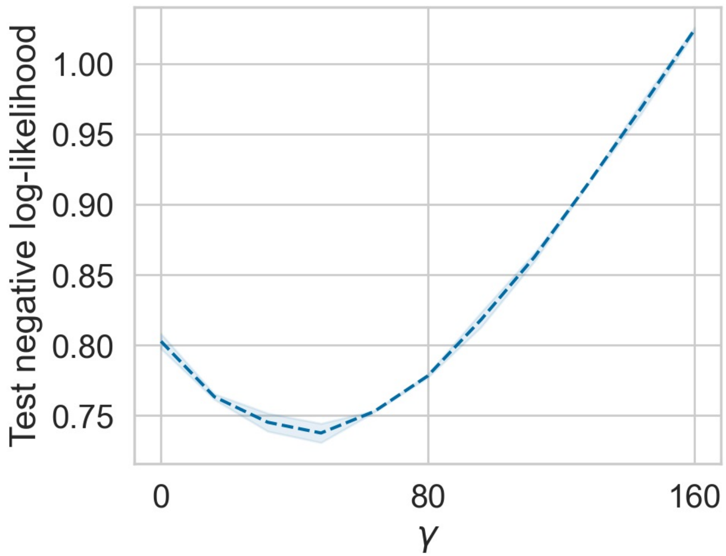
↓

If a sample with a wrong label is corrupted with sufficiently diverse corruptions, the model fails to memorize this wrong label.

[1]Arpit et al. (2017). A closer look at memorization in deep networks.

Robust learning under label noise



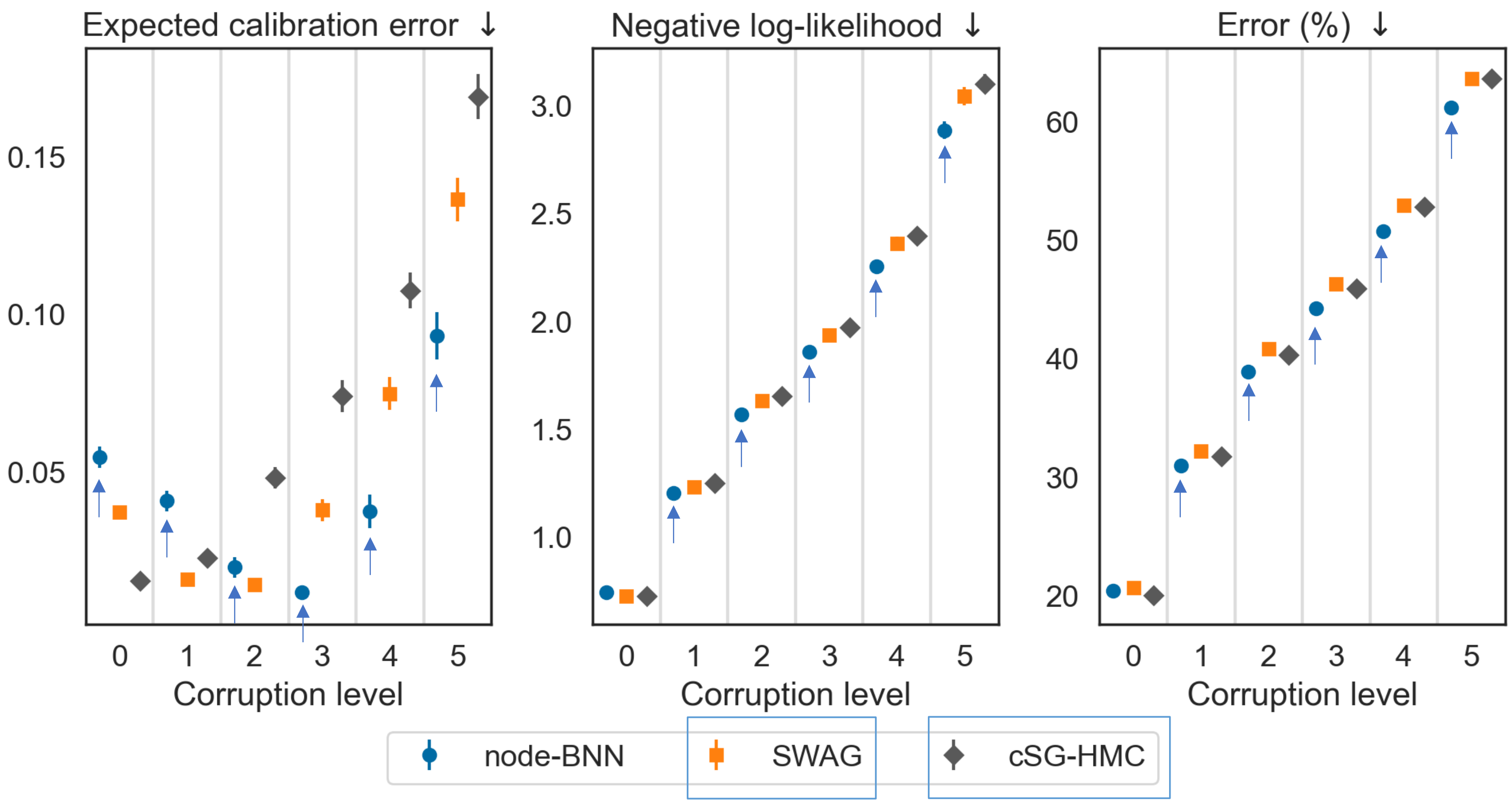


NLL on clean test set of CIFAR-10

Train NLL of wrongly labelled samples (in orange) increase much faster than the train NLL of correctly labelled samples (in blue)

ResNet18 / CIFAR-10
40% of training labels are corrupted

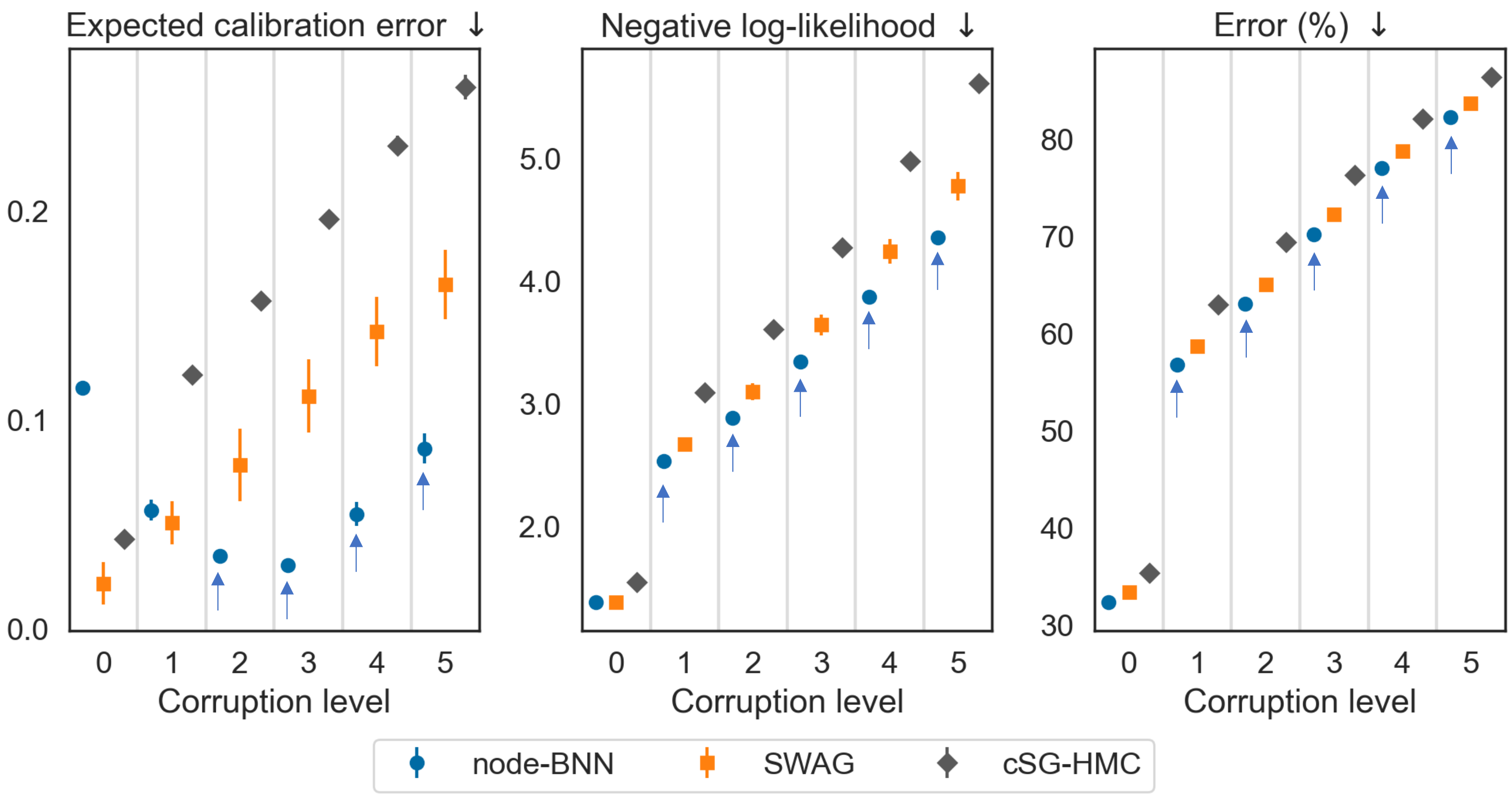
Benchmark comparison



ResNet18 / CIFAR-100

Maddox et al. (2019). A Simple Baseline for Bayesian Uncertainty in Deep Learning.
Zhang et al. (2020). Cyclical Stochastic Gradient MCMC for Bayesian Deep Learning.

Benchmark comparison



PreActResNet18 / TinyImageNet

Conclusion

1) Node-based BNNs are efficient alternative to standard weight-based BNNs that are effective against input corruptions.

2) Node-based BNNs can be made more robust against corruptions by increasing the entropy of the latent posterior.

More information is available at https://aaltopml.github.io/node-BNN-covariate-shift/