Tackling *covariate shift* with *node-based* Bayesian neural networks

Trung Trinh

Markus Heinonen

Luigi Acerbi

Samuel Kaski

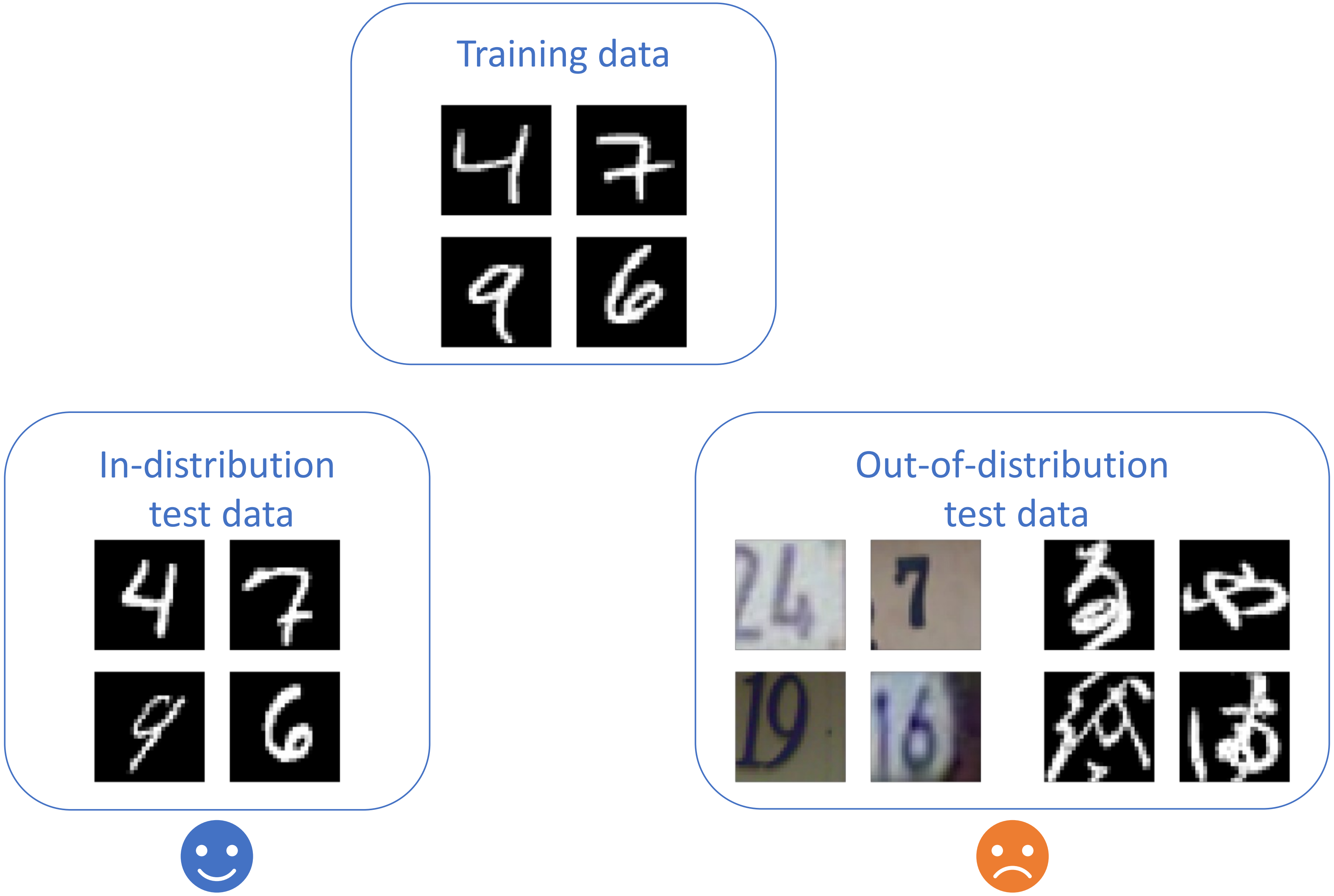
Aalto University School of Science and Technology

UNIVERSITY OF HELSINKI

The University of Manchester

Background

Covariate shift

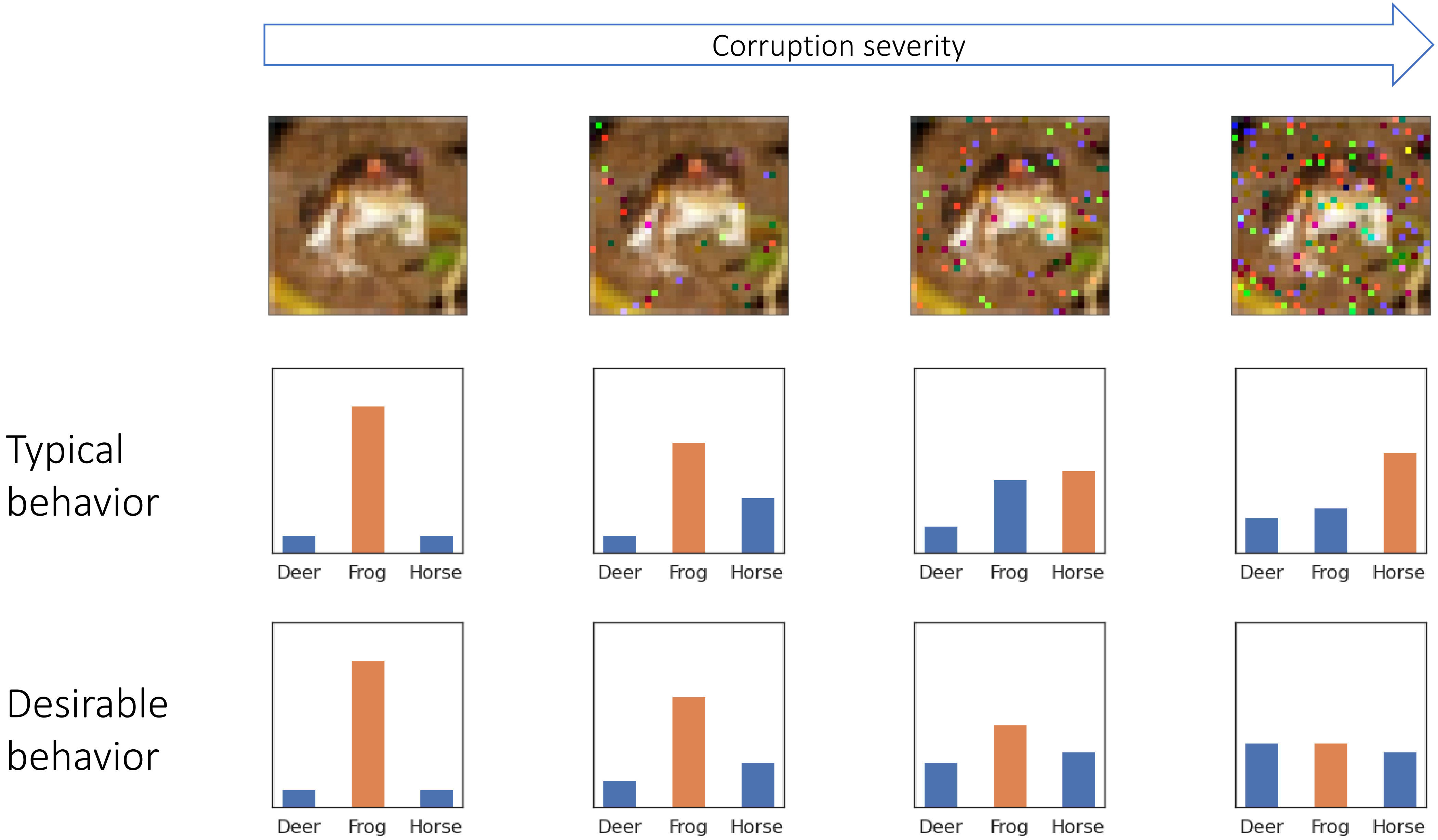


Shift due to corruptions

Shifts due to **corruptions**

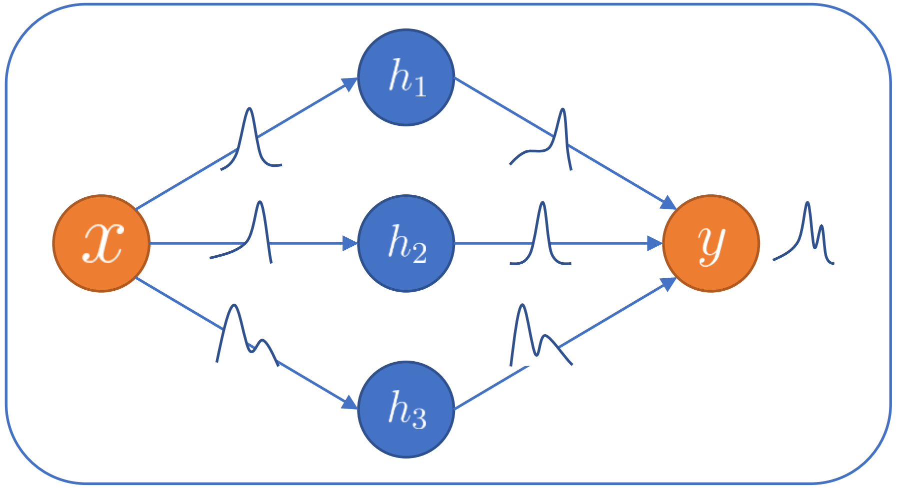
Hendrycks & Dietterich (2019). Benchmarking Neural Network Robustness to Common Corruptions and Perturbations.

Neural networks under input corruptions



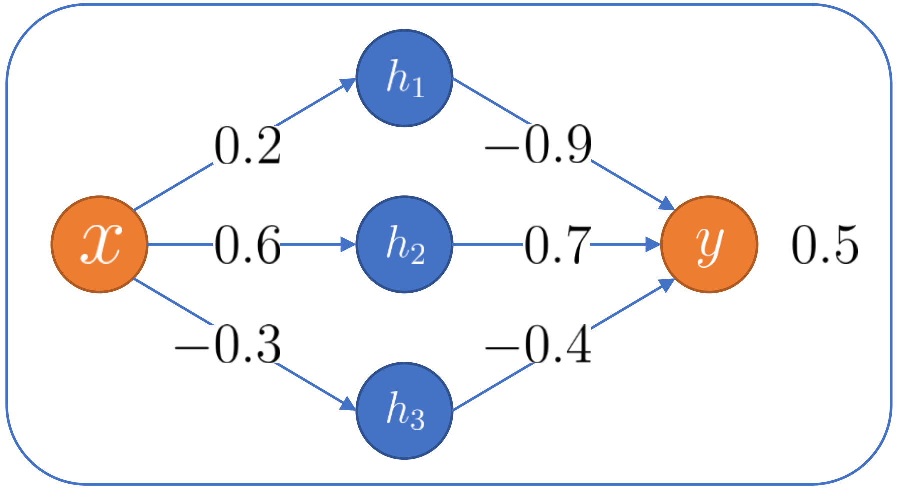
Bayesian neural networks (BNNs)

Bayesian neural network



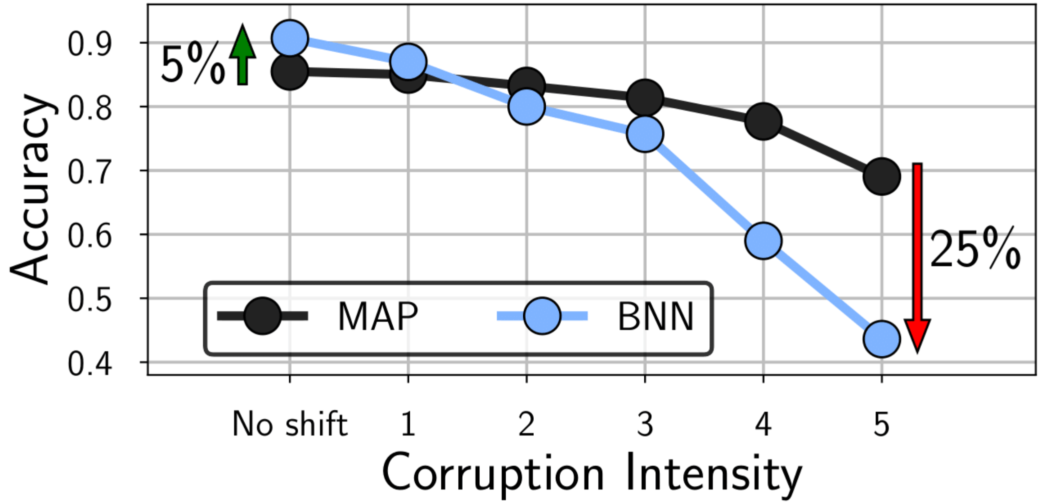
Thomas Bayes

Standard neural network



Neal (1996). Bayesian Learning for Neural Networks.

BNNs perform worse than MAP models under corruptions[1]



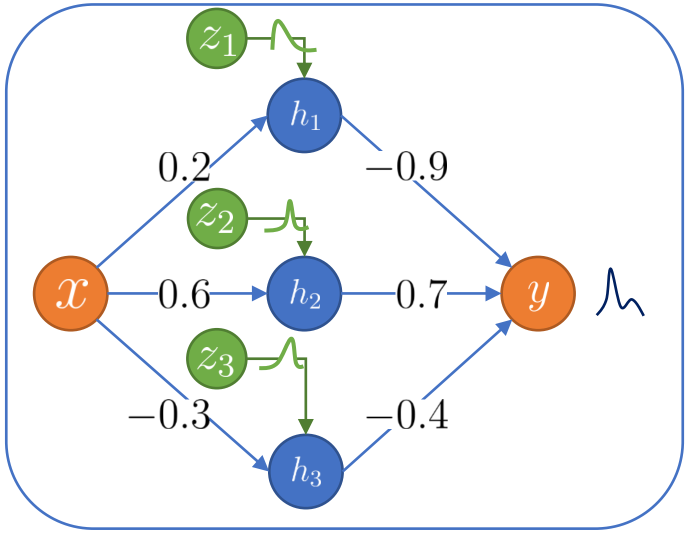
ResNet-20, CIFAR-10-C

Why? Because the standard Gaussian prior does not provide good inductive biases to handle input corruptions.[2]

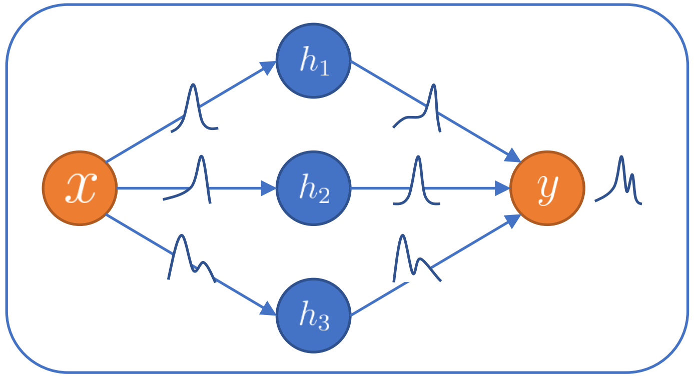
[1] Izmailov et al. (2021). What are Bayesian neural network posteriors really like?
[2] Izmailov et al. (2021). Dangers of Bayesian model averaging under covariate shift?

Node-based Bayesian neural networks

Node-BNNs

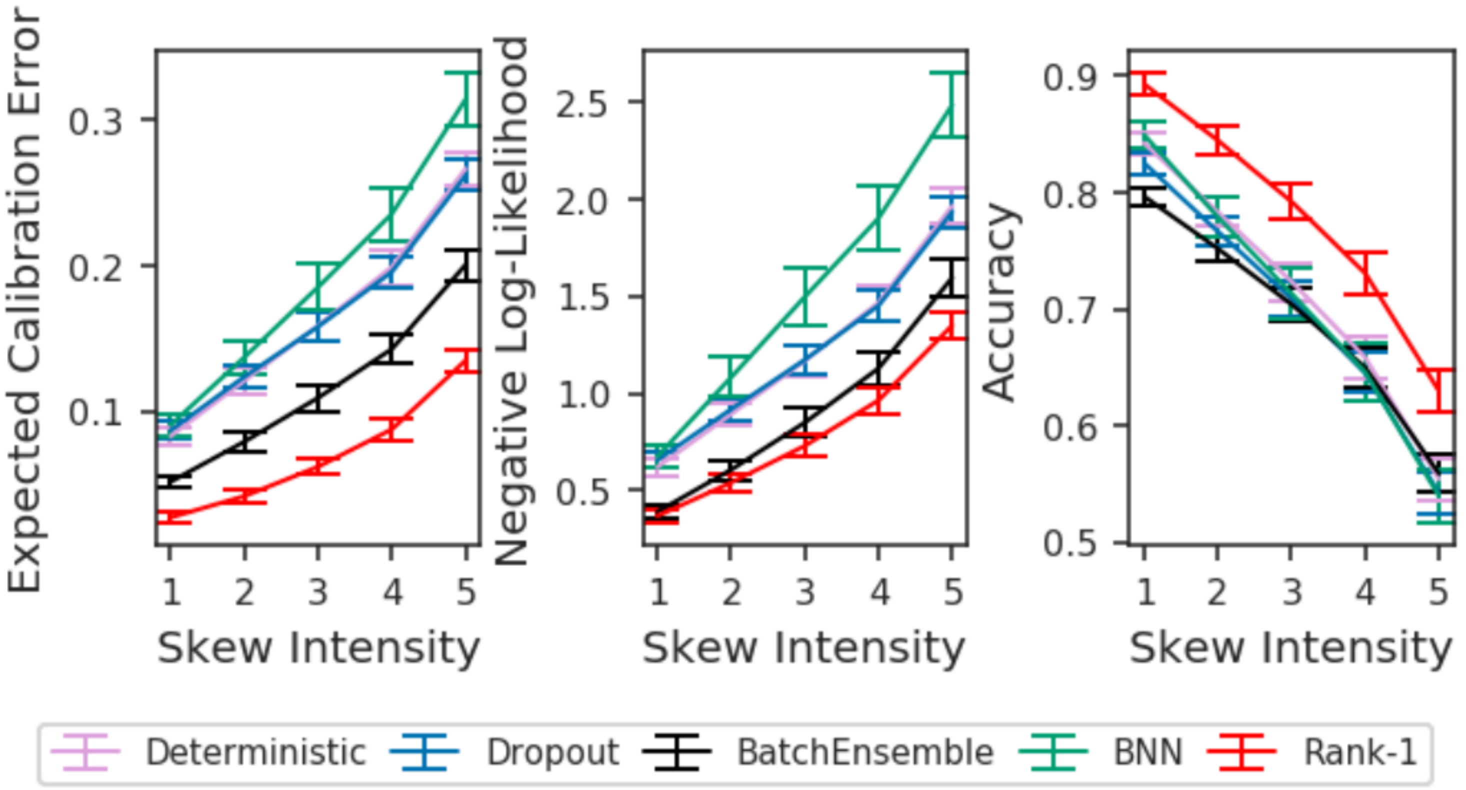


Weight-BNNs



E.g.: MC-Dropout (Gal et al, 2015), Rank-1 BNNs (Dusenberry et al, 2020)

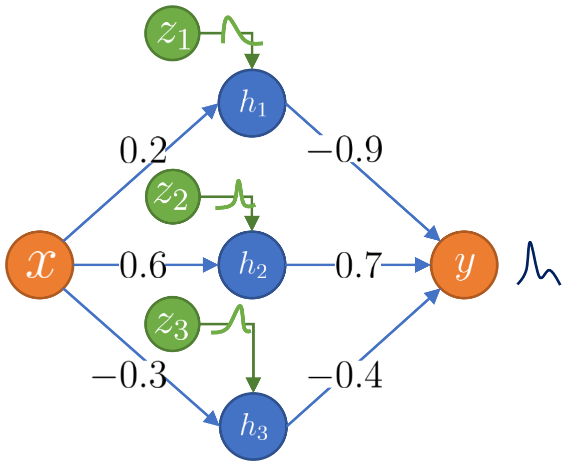
Node-BNNs perform better than MAP models under corruptions

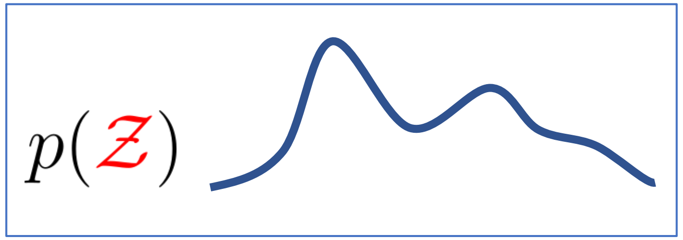


WideResNet-28-10 / CIFAR-10-C

Dusenberry et al. (2020). Efficient and Scalable Bayesian Neural Nets with Rank-1 Factors.

Node-based Bayesian neural networks





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

Previous layer's output

Latent node variables

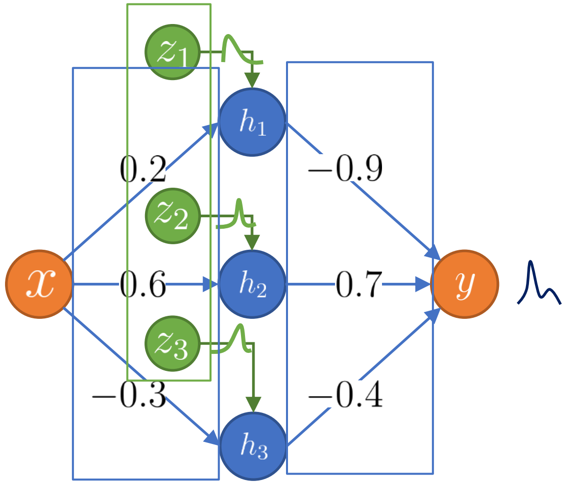
$$f_{in}^{(\ell)} = f^{(\ell-1)}(x; \mathcal{Z}) \circ z^{(\ell)}$$

$$f^{(\ell)}(x; \mathcal{Z}) = \sigma\left(W^{(\ell)} f_{in}^{(\ell)} + b^{(\ell)}\right)$$

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

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

Node-based Bayesian neural networks



		Parameters		
Network	Layers	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.

Our paper’s goals

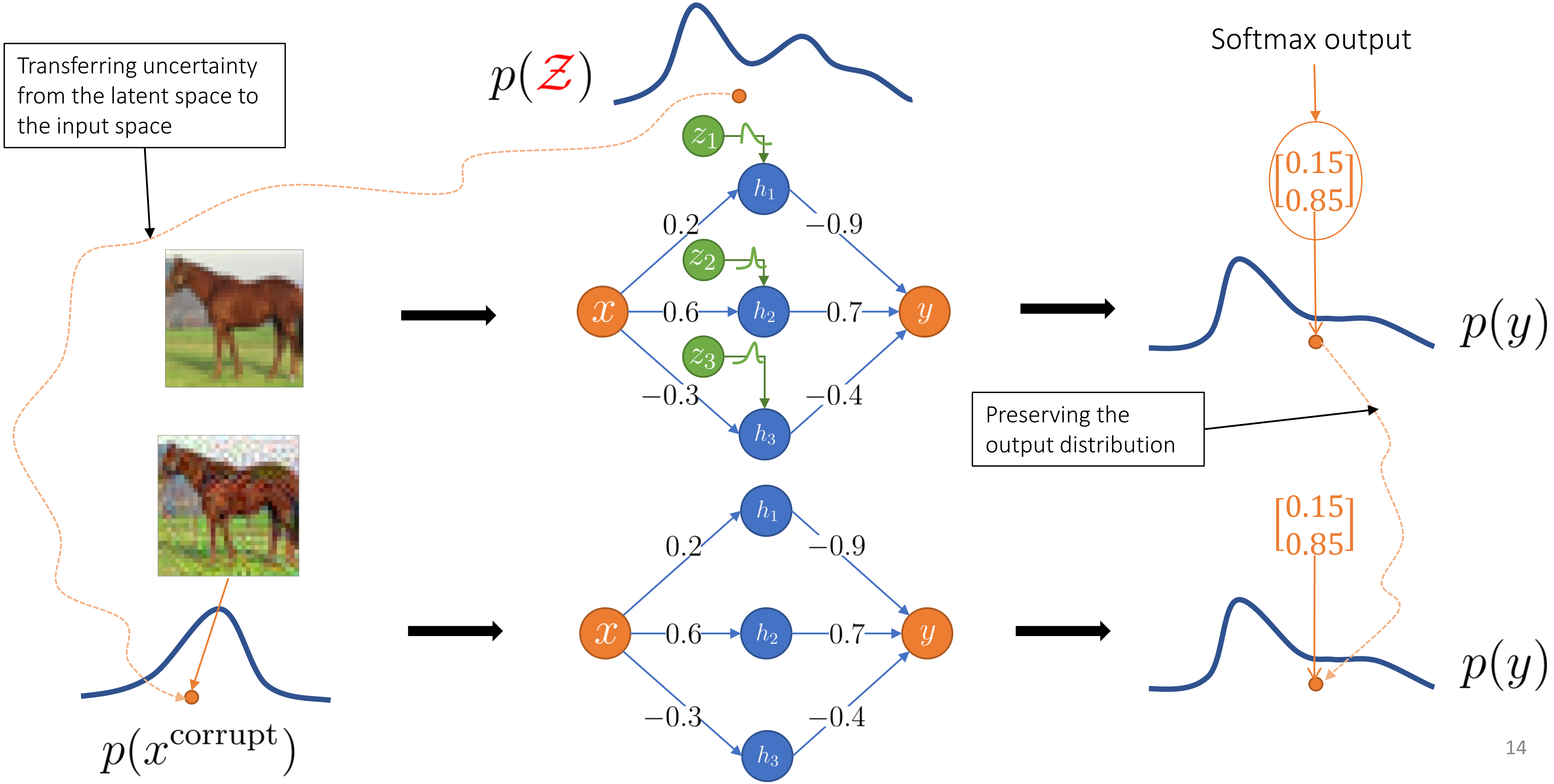
Providing insights into the robustness of node-based BNNs under input corruptions.

↓

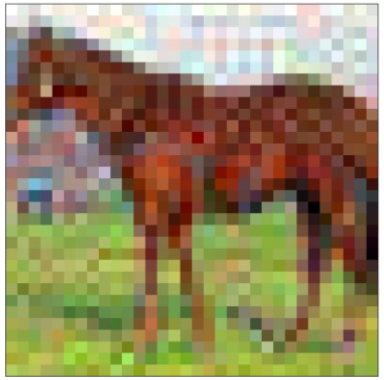
Proposing a method to further improve the robustness of node-based BNNs in this setting.

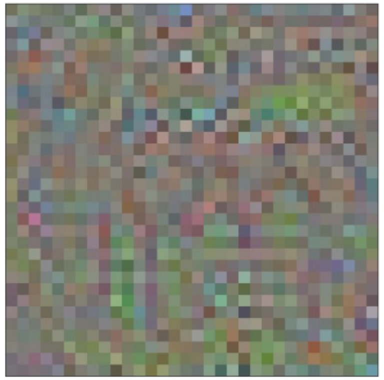
Why do node-based BNNs generalize well under input corruptions?

We hypothesize that the distribution of the latent variables $p(\mathcal{Z})$ induce a distribution of implicit corruptions in the input space $p(x^{\text{corrupt}})$.



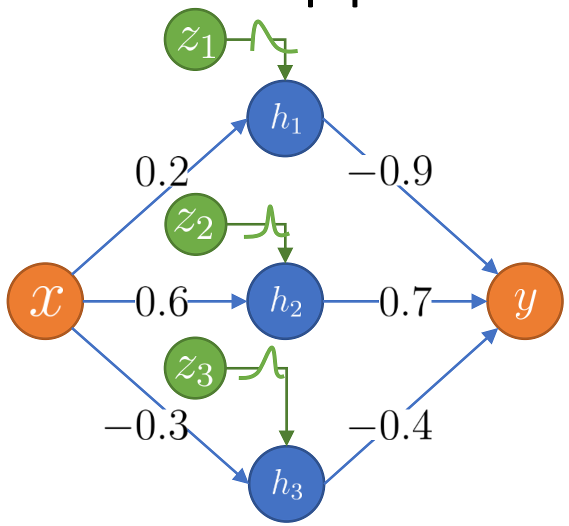
Approximating the implicit corruption



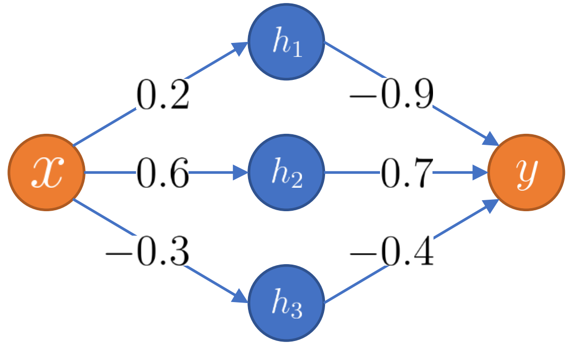


$$x_{corrupt} = x + m$$

Approximating the implicit corruption



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



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

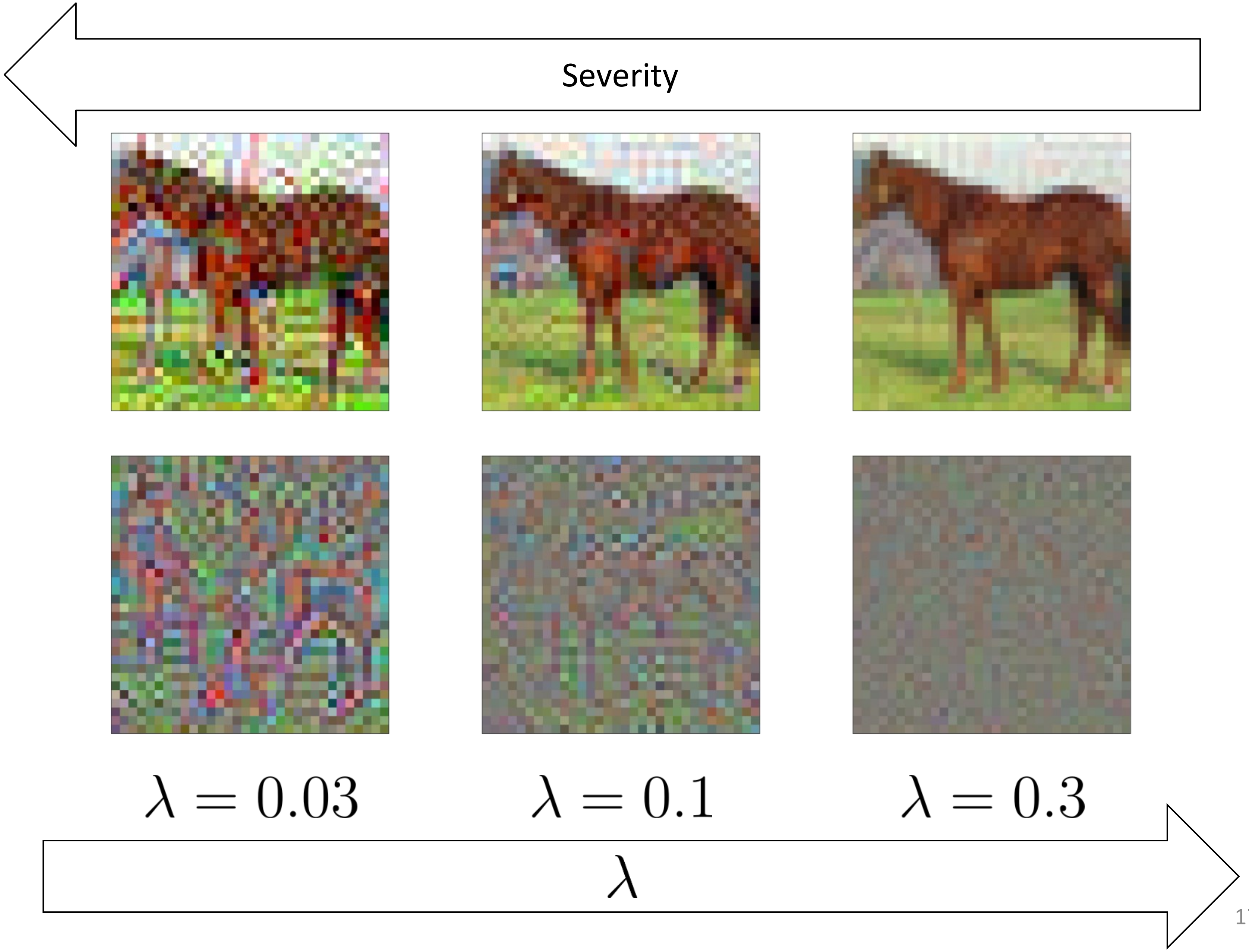
Given $\mathcal{Z} \sim p(\mathcal{Z})$, approximating m minimizing the following loss function using GD:

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

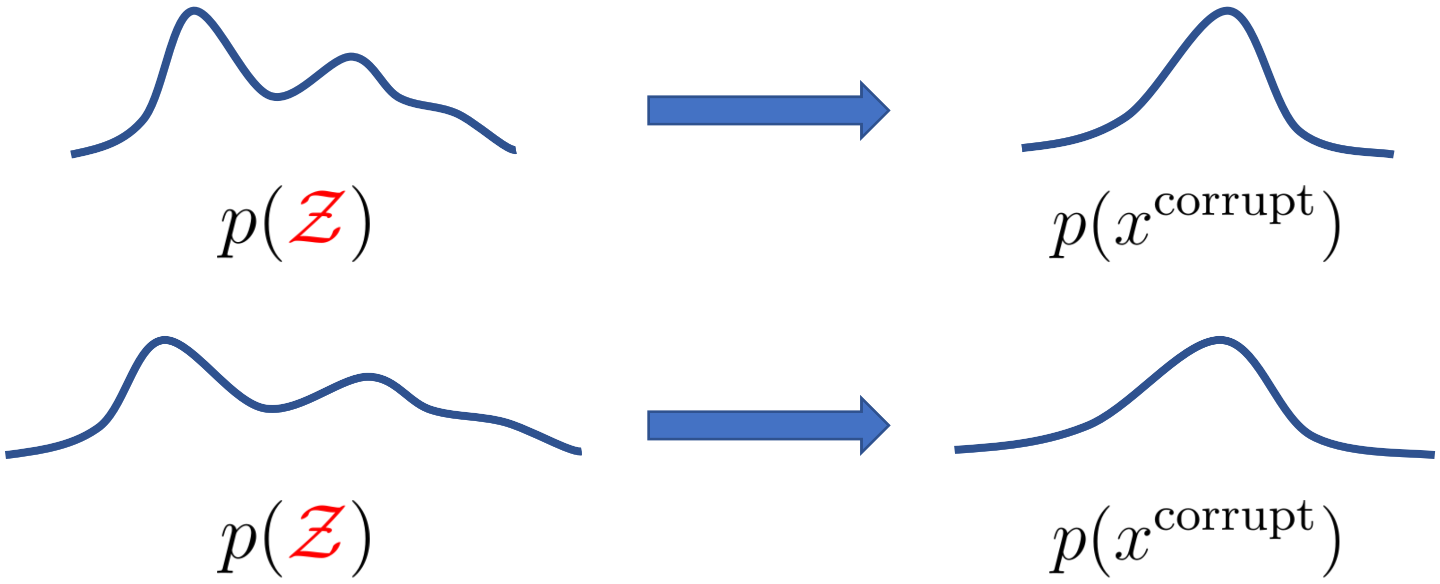
Output matching

L2-regularization

Example of implicit corruptions



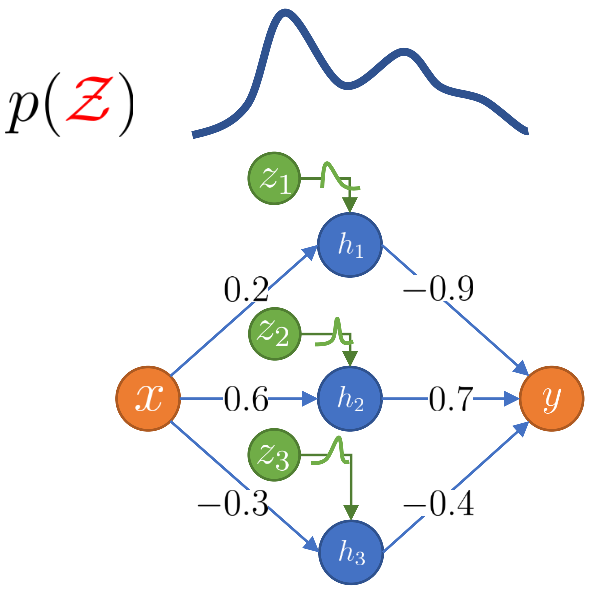
Entropy of latent variables and implicit corruptions



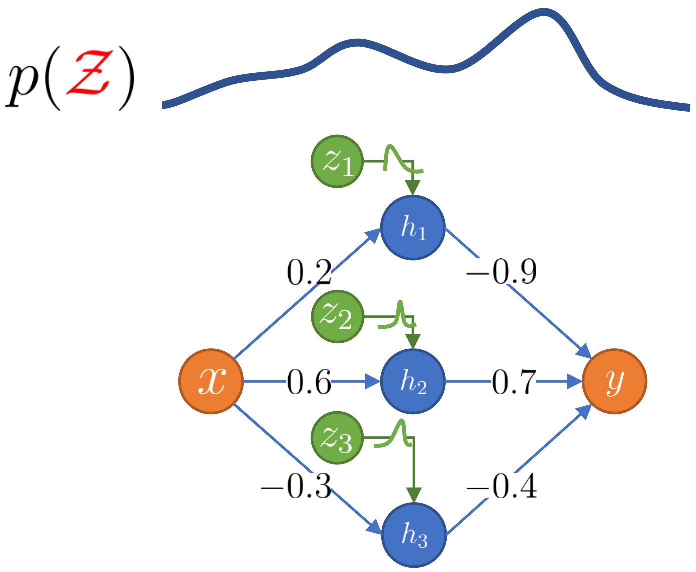
We hypothesize that:

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

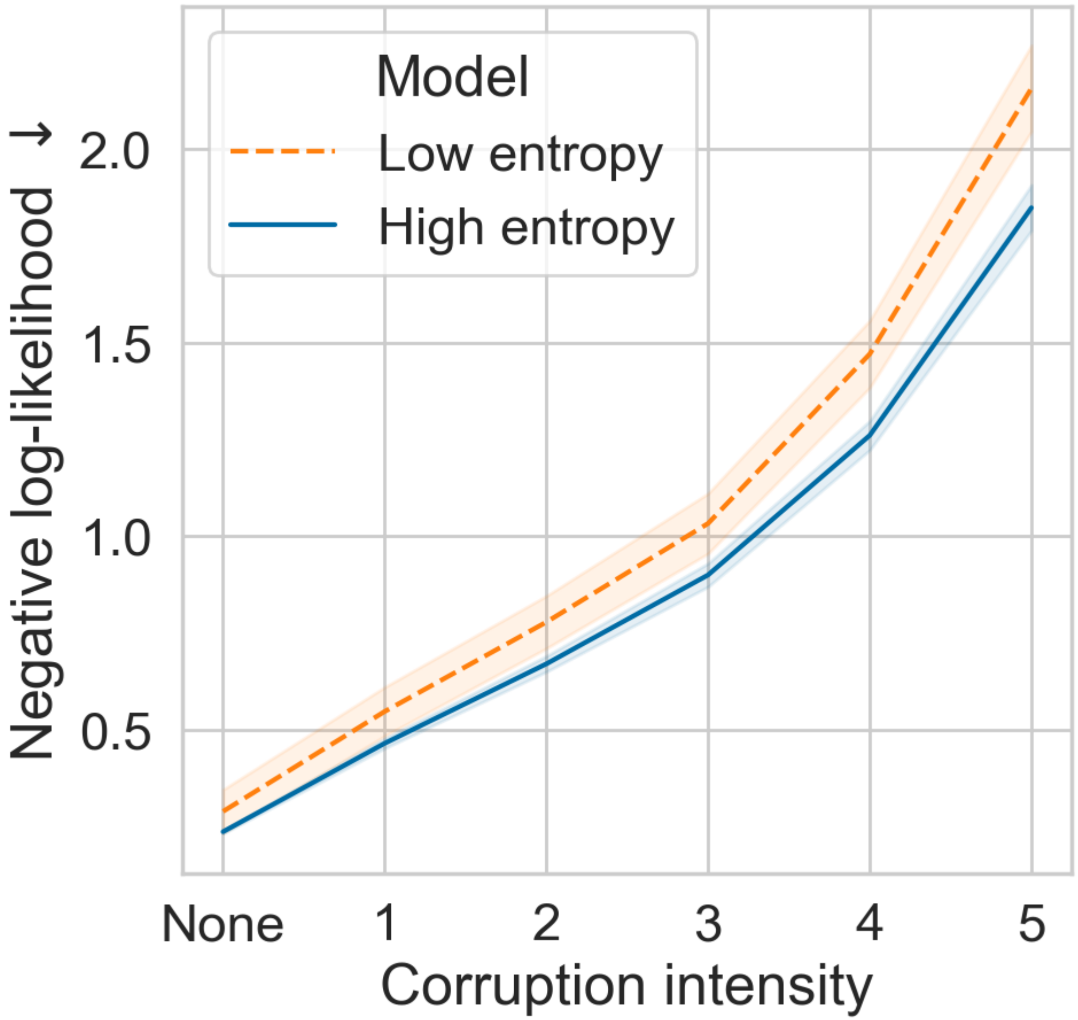
Same ConvNet architecture

Train on CIFAR-10

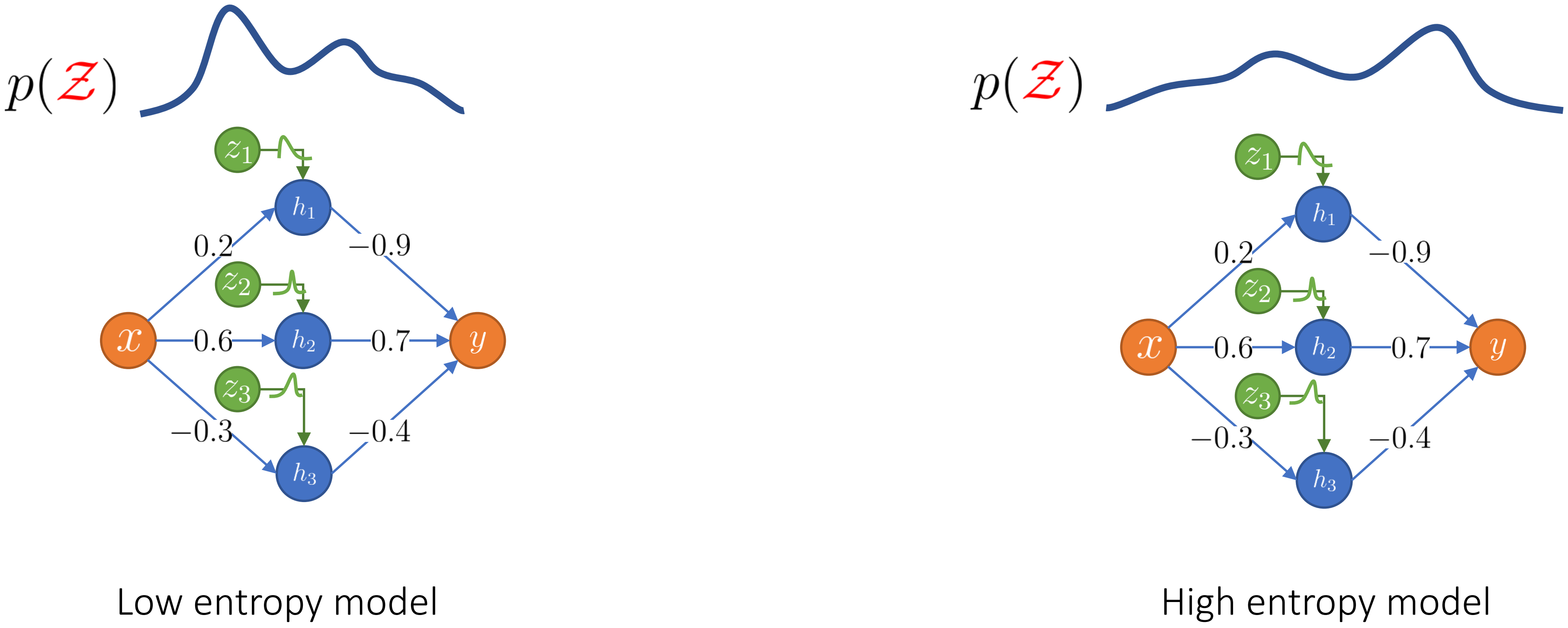
Test on CIFAR-10-C

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

YES!!!



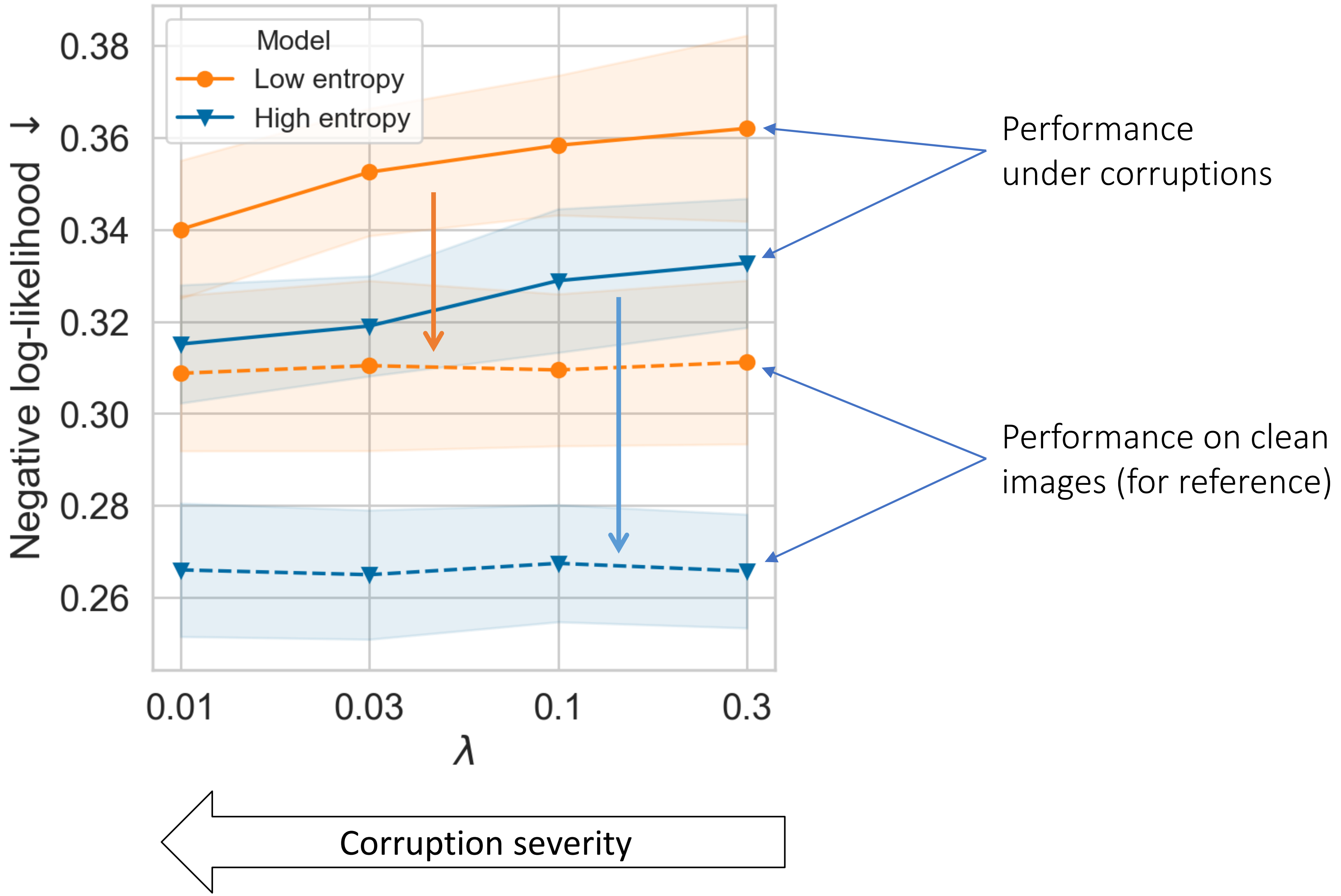
Is a model robust against its own corruptions?



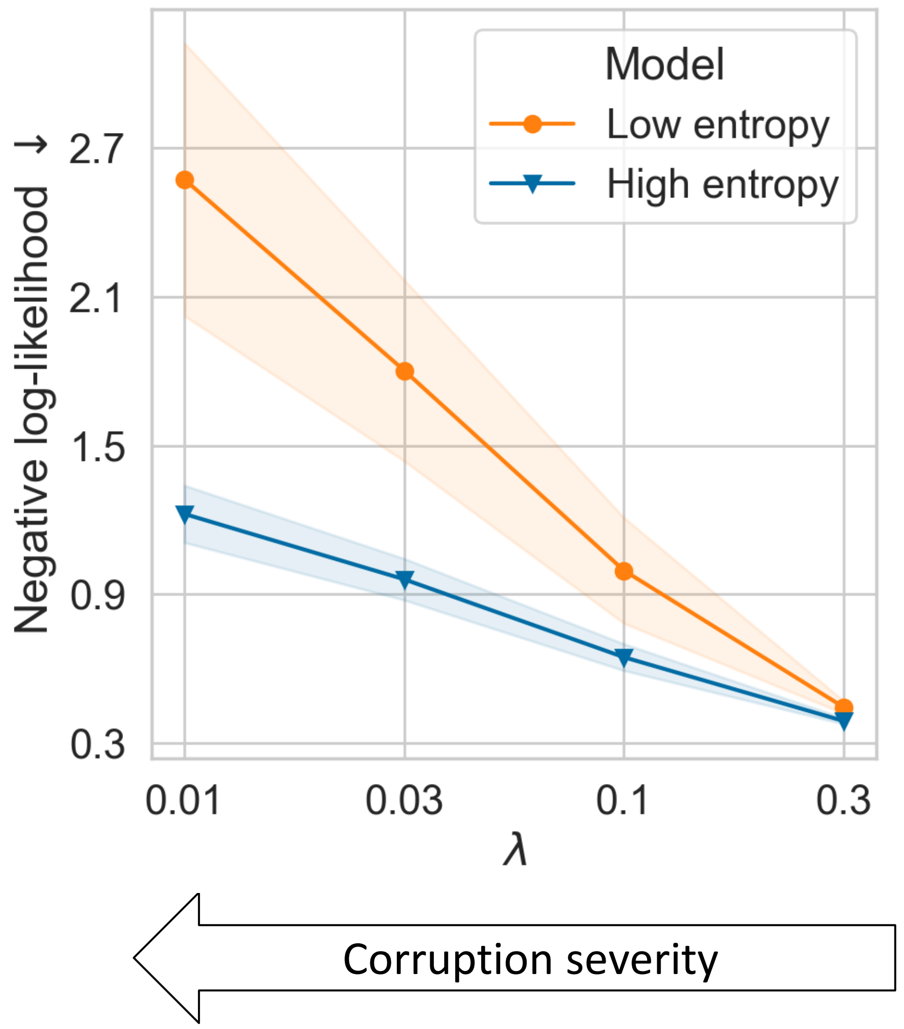
We use each model to generate a set of corrupted test images, then evaluate each model on its own generated corruptions.

Is a model robust against its own corruptions?

YES (in this small experiment)

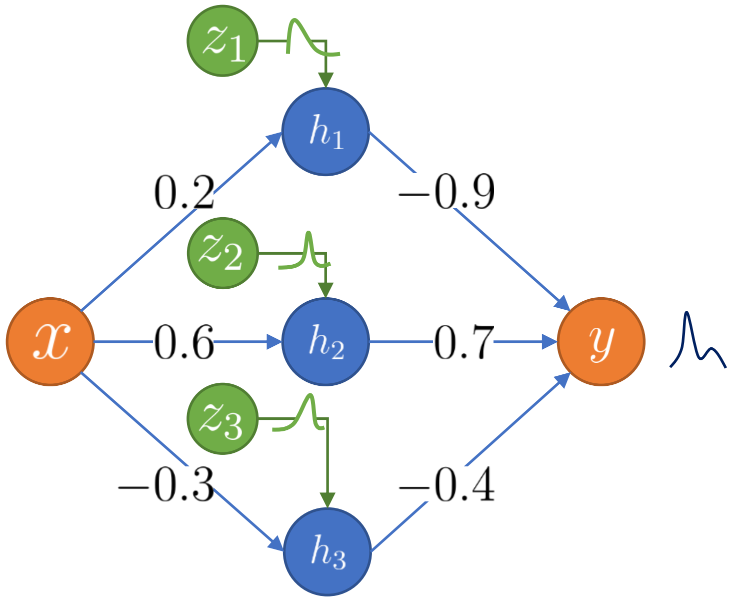


How robust is a model against the other model's corruptions?



How to increase the latent entropy?

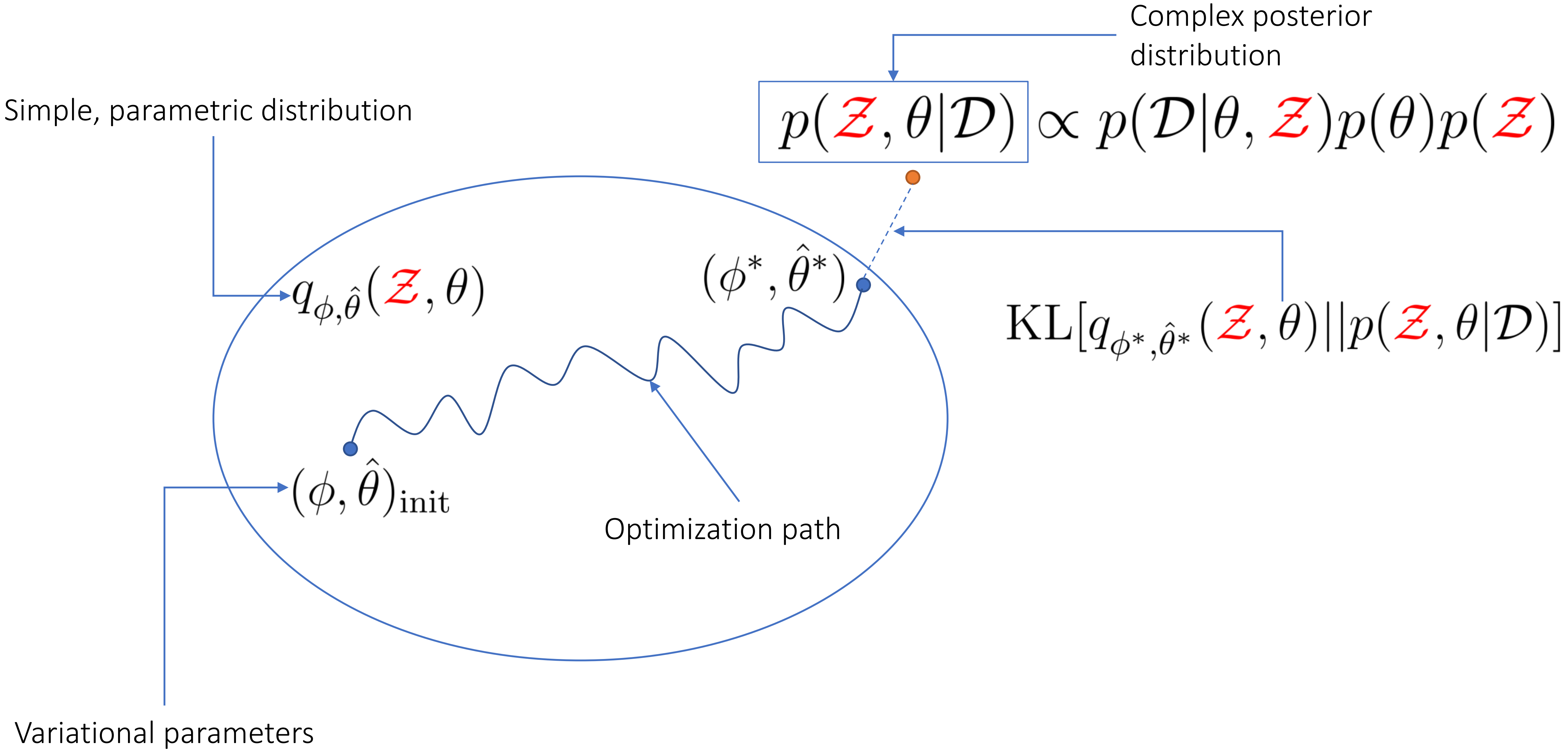
Training a node-based BNN



Two types of parameters:

1. Weights and biases $\theta = \{(W^{(\ell)}, b^{(\ell)})\}_{\ell=1}^{L}$ with a prior $p(\theta)$.
   ➔ Find a MAP estimate.
2. Latent node variables $\mathcal{Z} = \{z^{(\ell)}\}_{\ell=1}^{L}$ with a prior $p(\mathcal{Z})$.
   ➔ Infer the posterior distribution.

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)

MAP estimation of θ

Mixture of Gaussians

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

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

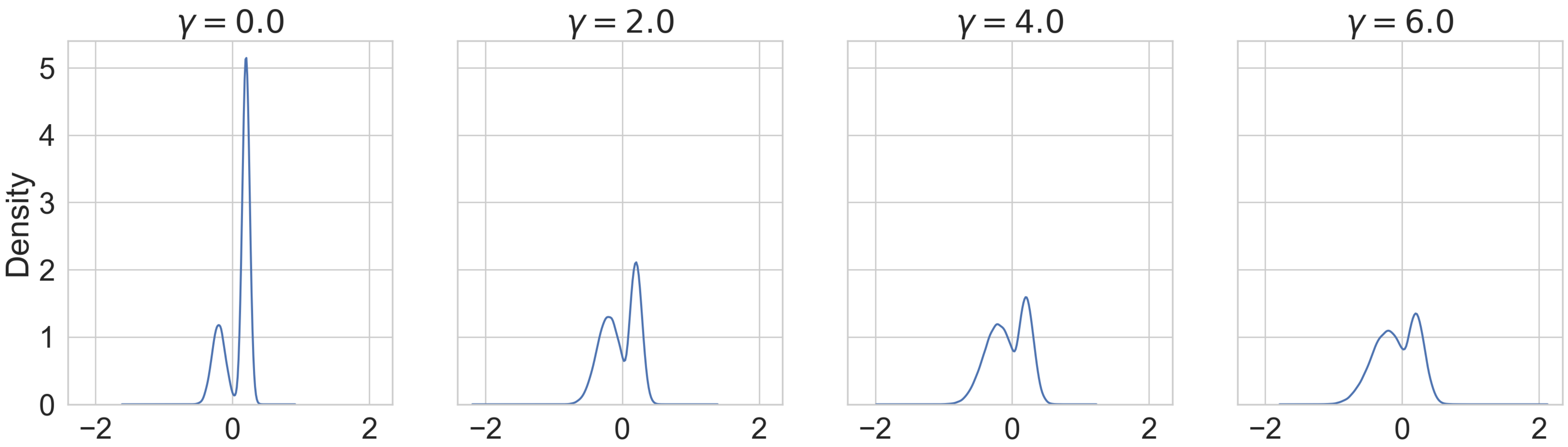
Maximizing the γ – ELBO is equivalent to minimizing:

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

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

Temperature $\tau = \gamma + 1$

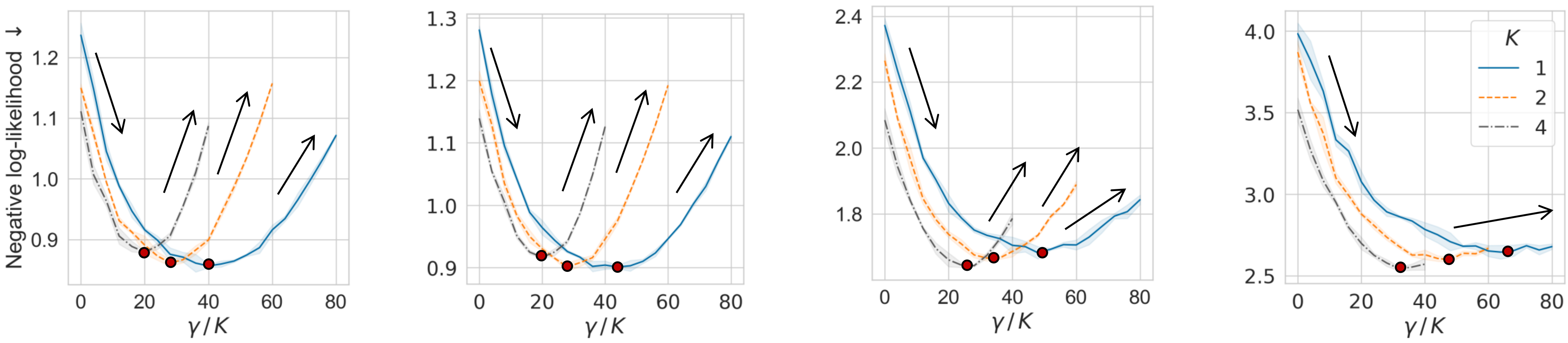
Effects of $\gamma > 0$ on the target posterior.



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

Ablation study

Effects of γ on corruption robustness



Validation

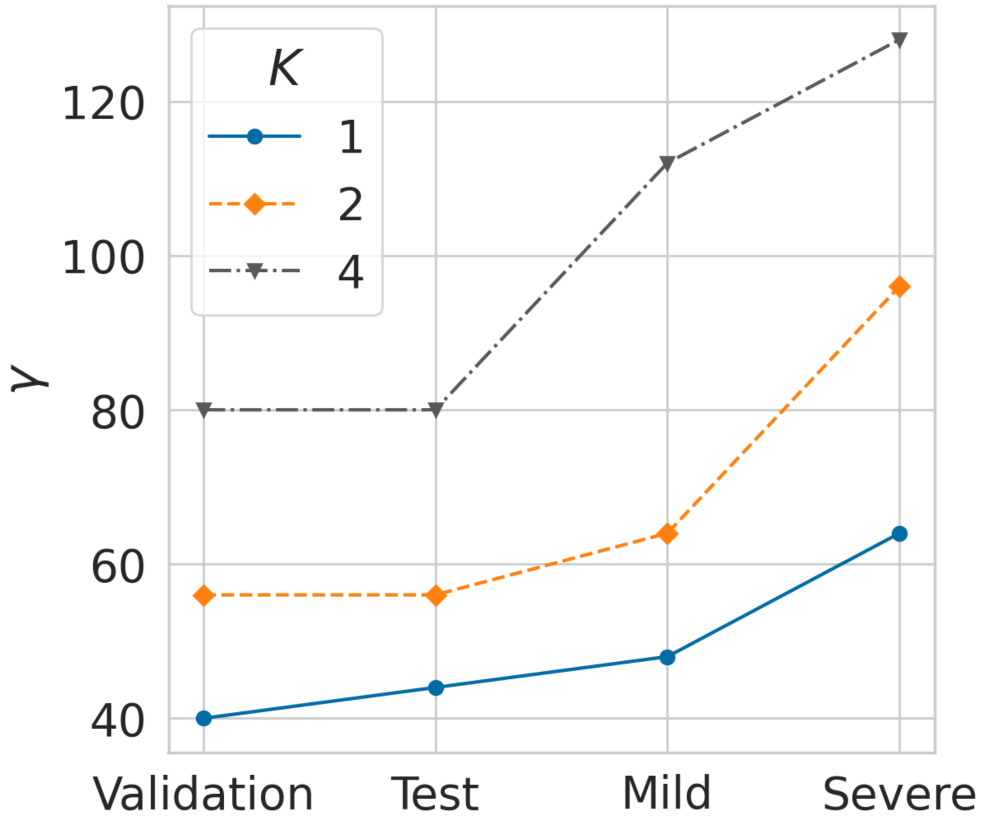
Test

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

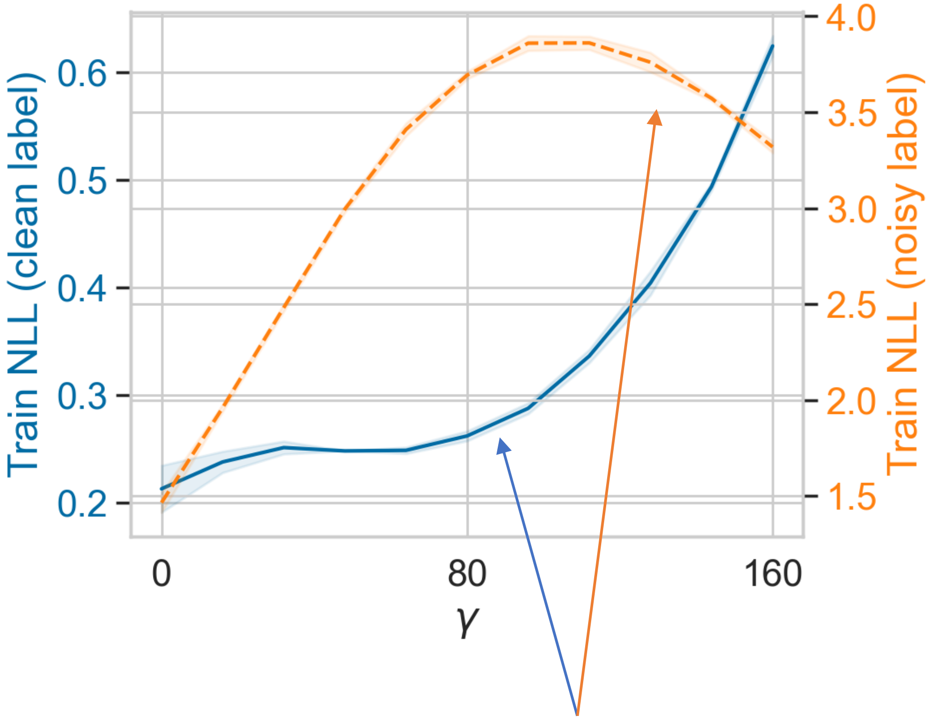
Robust learning under label noise

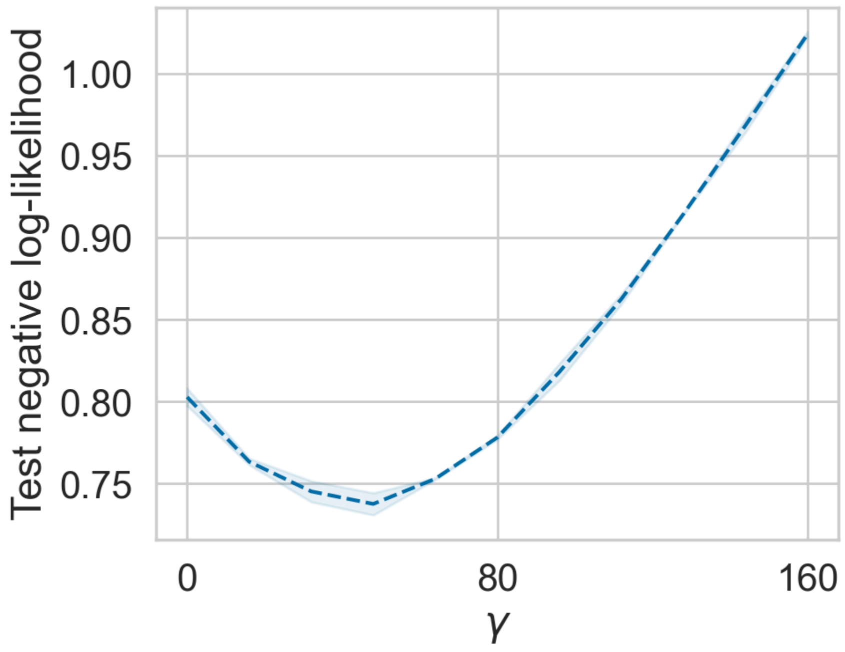
Memorizing random labels **is harder than** learning generalizable patterns[1]

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



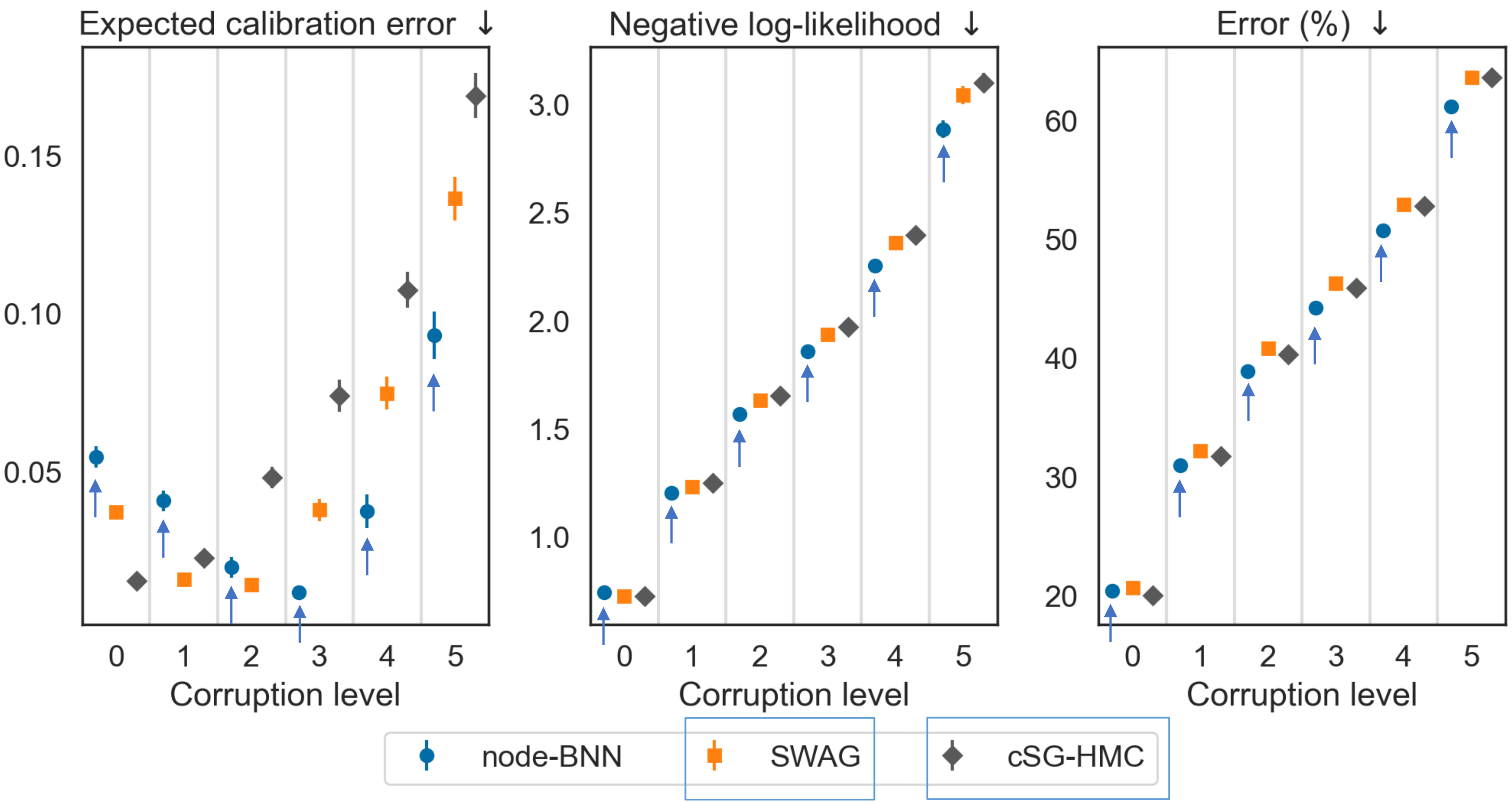


NLL on clean test set of CIFAR-10

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

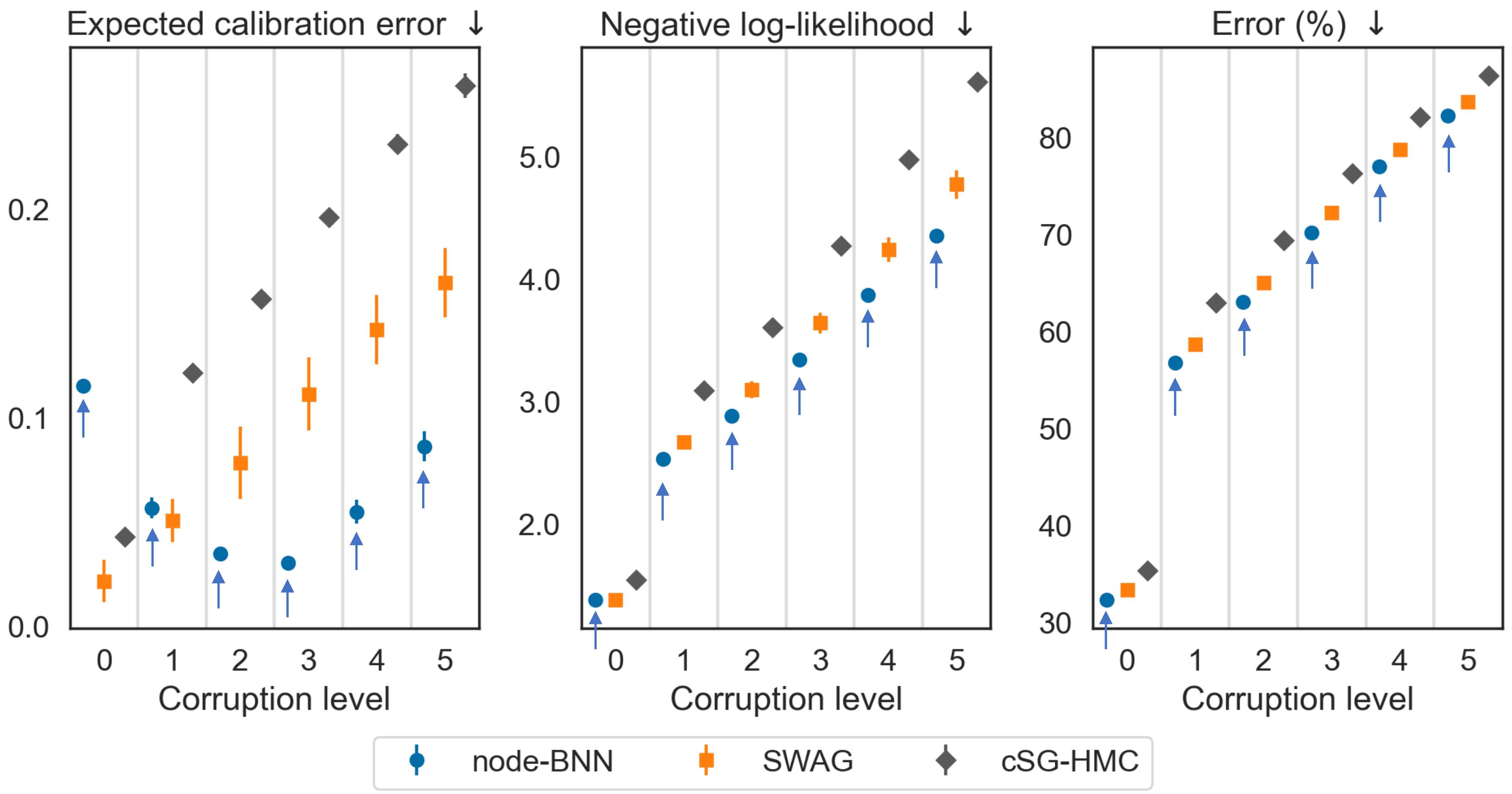
Benchmark comparison



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. We showed that the latent variables simulated a set of implicit corruptions, and by training under these corruptions, node-based BNNs become robust against natural corruptions.

2. By maximizing the entropy of the latent variables, we increase the diversity of the implicit corruptions and thus improve the robustness of node-based BNNs.

3. We demonstrated that the latent entropy controls the trade-off between in-distribution performance and performance under corruptions, with more severe corruptions require higher optimal latent entropy which decreases the in-distribution performance.

4. As a side effect, our method also provides robustness against noisy training labels.

For more information visit: