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

Trung Trinh

Markus Heinonen

Luigi Acerbi

Samuel Kaski

Aalto University
School of Science
and Technology

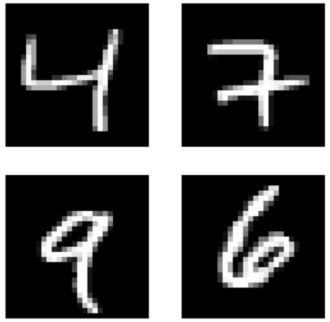
UNIVERSITY OF HELSINKI

MANCHESTER
1824
The University of Manchester

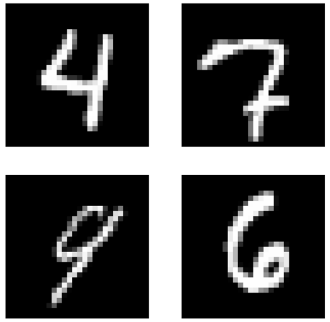
Background

Covariate shift

Training data



In-distribution
test data

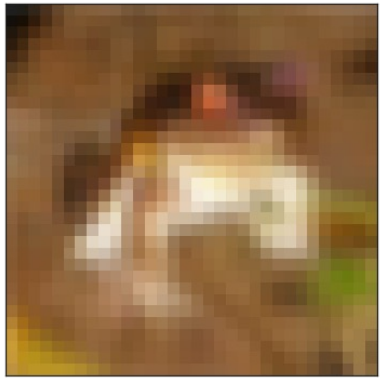


Out-of-distribution
test data

Shift due to corruptions

Shifts due to **corruptions**

Noise

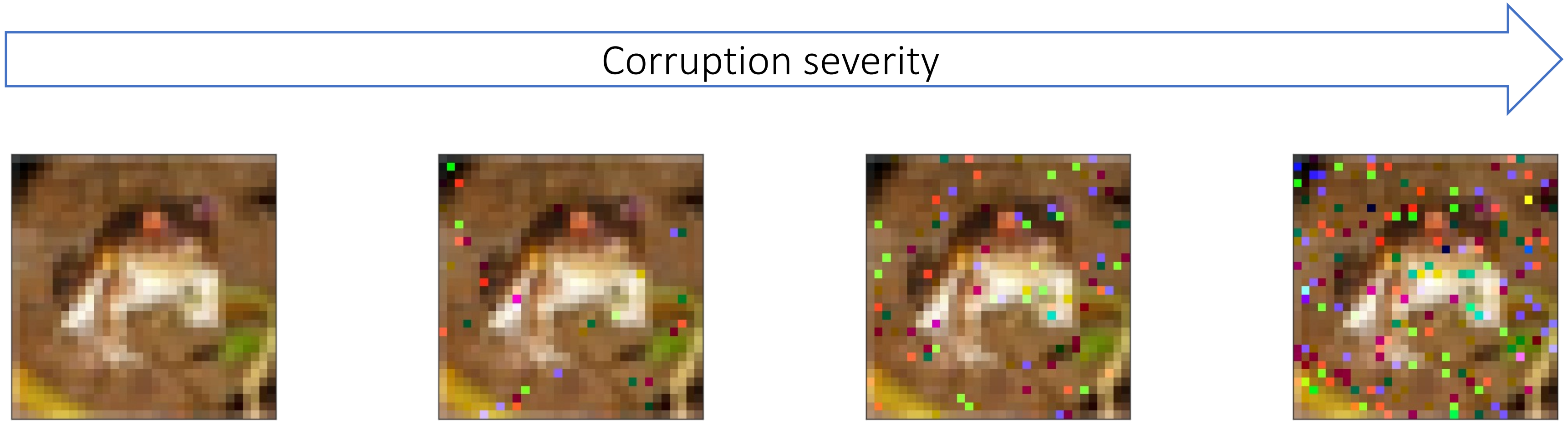


Blur

Saturation

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

Neural networks under input corruptions

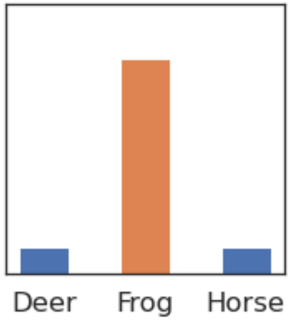


Neural networks under input corruptions

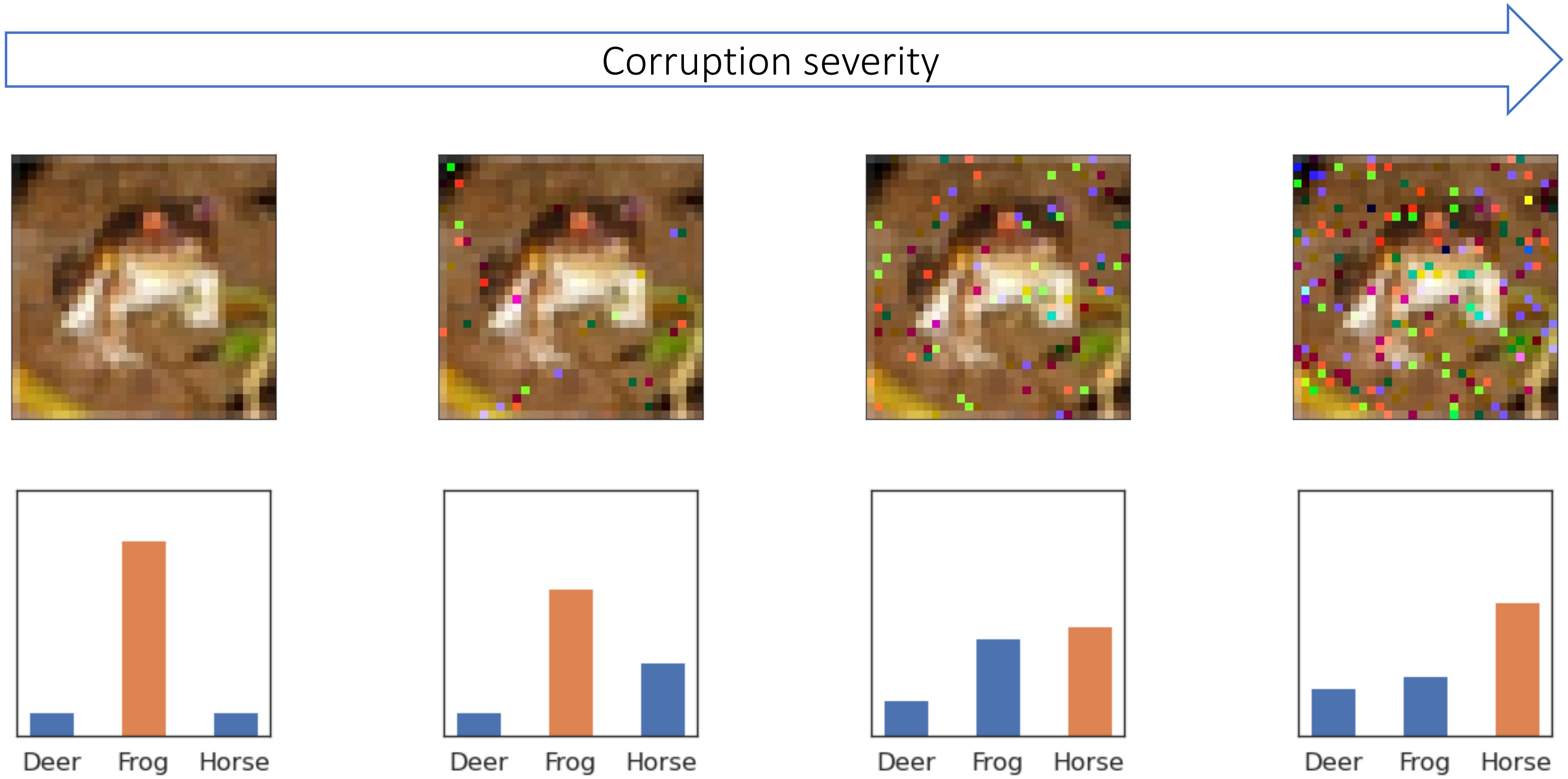
Corruption severity




Typical behavior

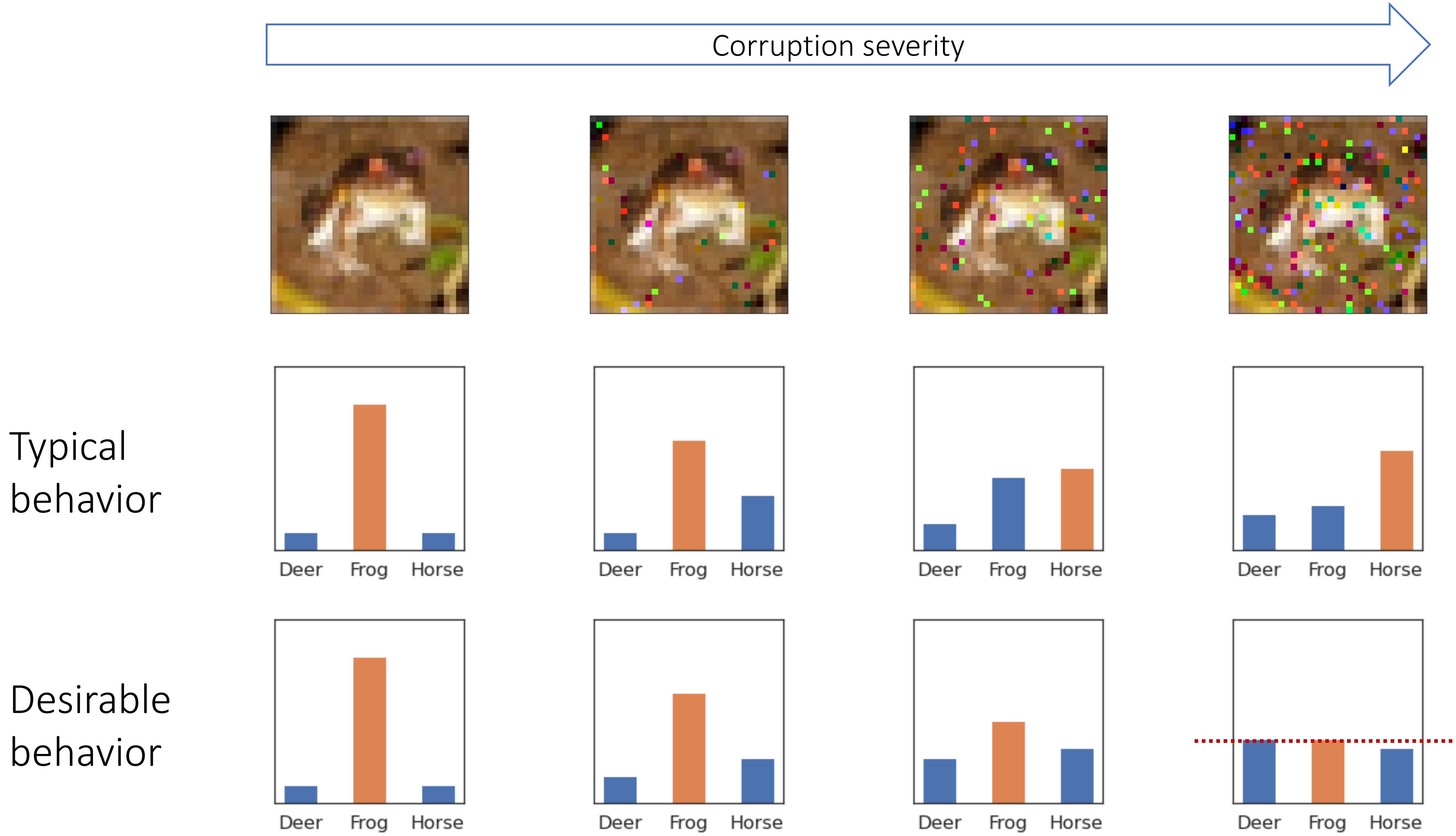


Neural networks under input corruptions



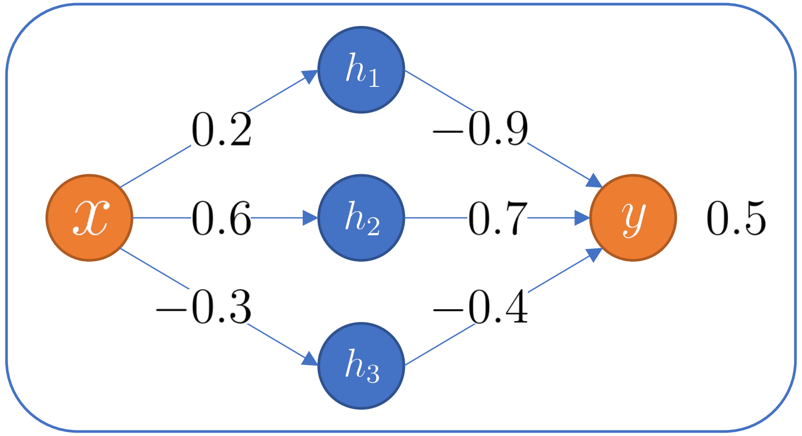
Typical behavior

Neural networks under input corruptions

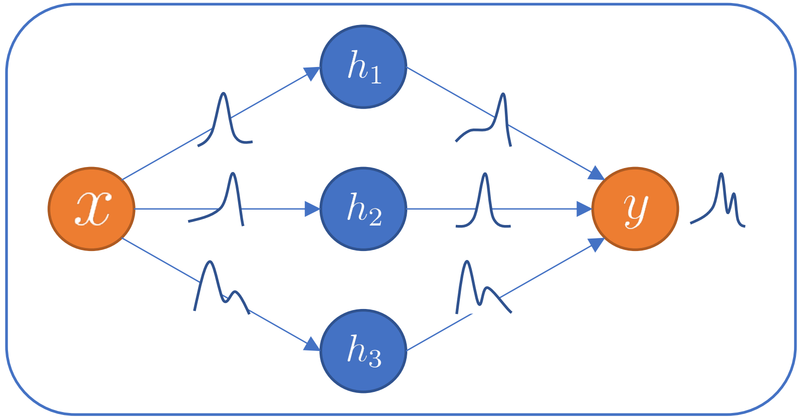


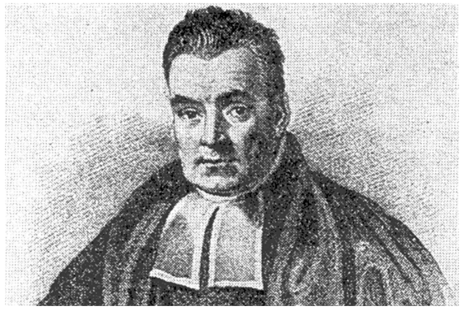
Bayesian neural networks (BNNs)

Standard neural network



Bayesian neural network



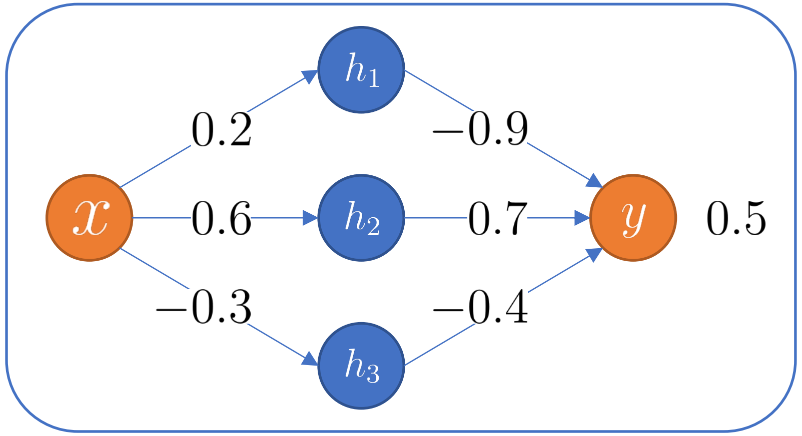


Thomas Bayes

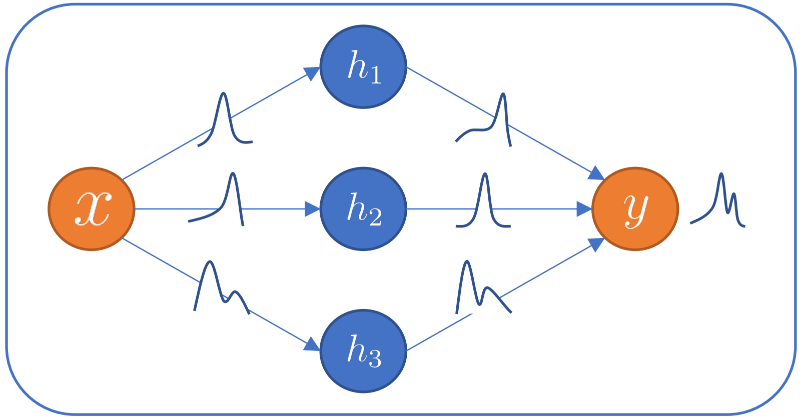
Neal (1996). Bayesian Learning for Neural Networks.

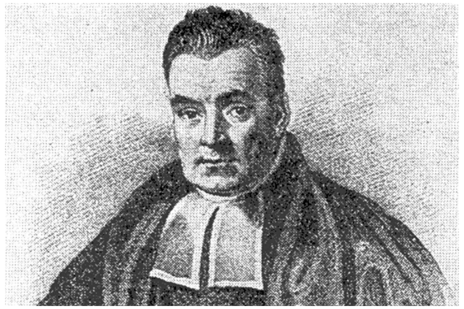
Bayesian neural networks (BNNs)

Standard neural network



Bayesian neural network



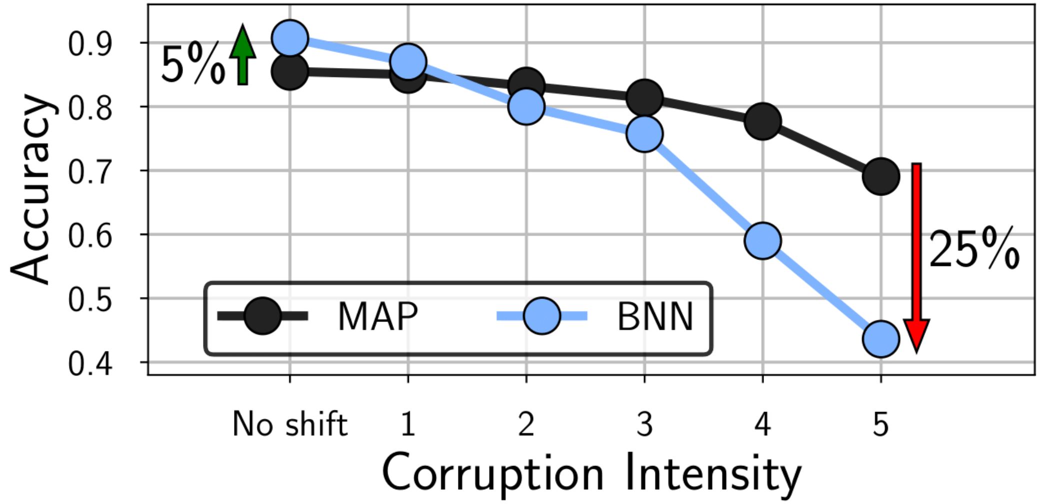


Thomas Bayes

Are BNNs more robust to corruptions?

Neal (1996). Bayesian Learning for Neural Networks.

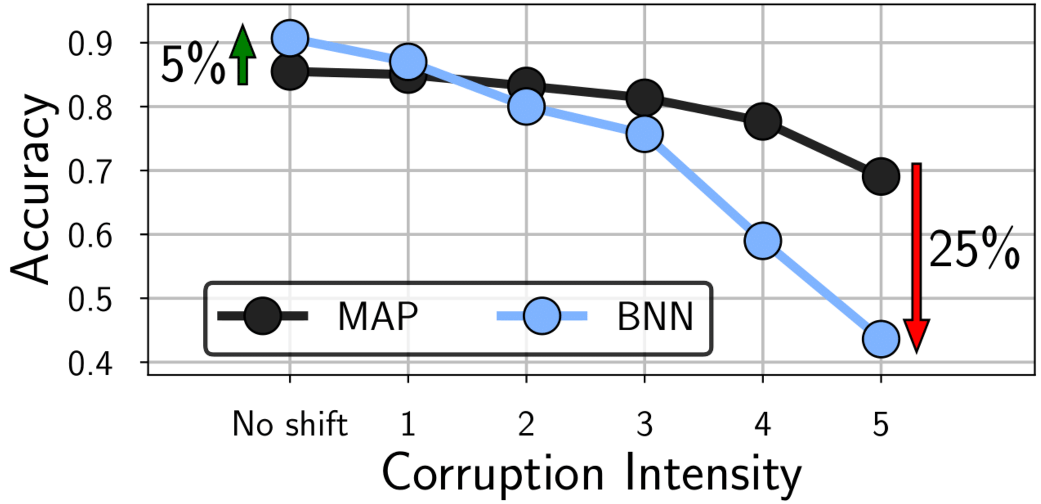
BNNs perform worse than MAP models under corruptions[1]



ResNet-20, CIFAR-10-C

[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?

BNNs perform worse than MAP models under corruptions[1]



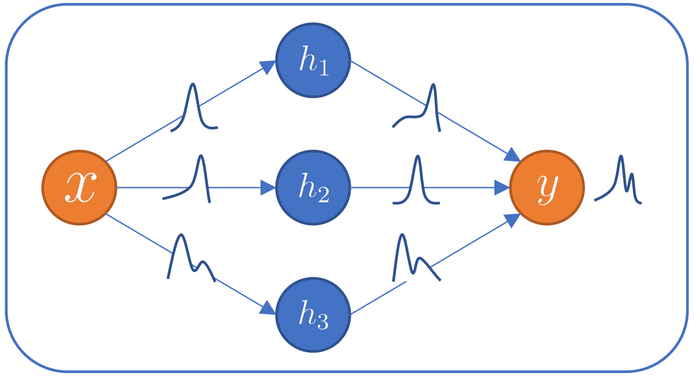
ResNet-20, CIFAR-10-C

Gaussian prior does not provide useful inductive biases to handle input corruptions.[2]

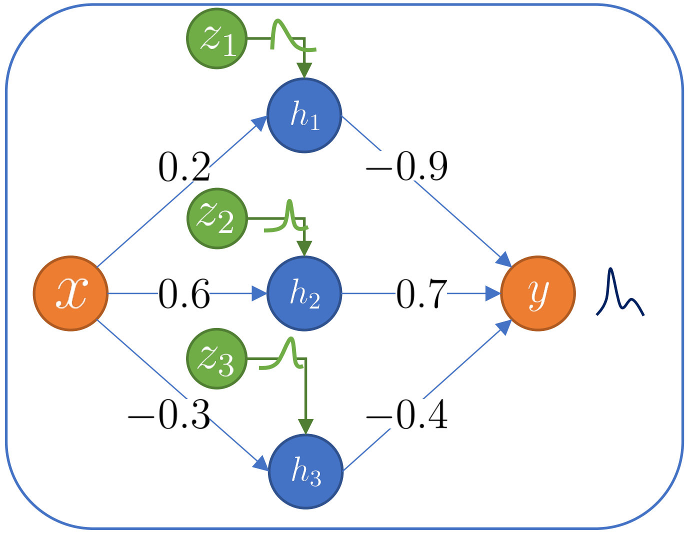
[1] Izmailov et al. (2021). What are Bayesian neural network posteriors really like?
[2] Izmailov et al. (2021). Dangers of Bayesian model averaging under covariate shift?

Node-based Bayesian neural networks

Weight-BNNs



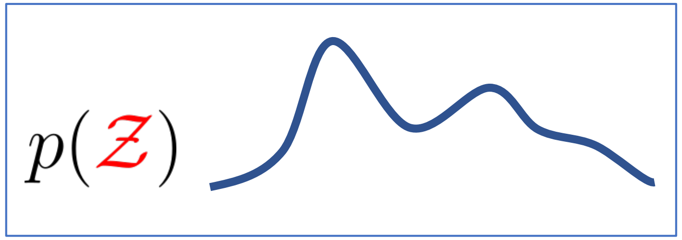
Node-BNNs



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

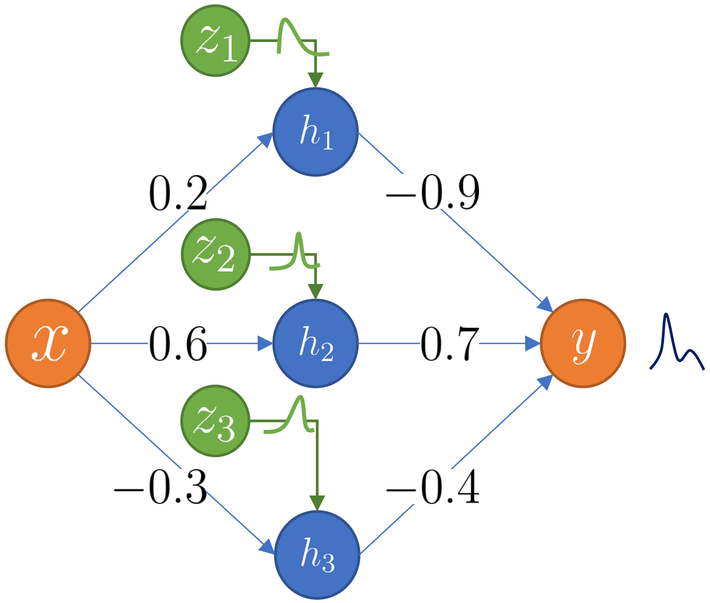
Node BNNs

NodeBNN with latent variables $\mathcal{Z} = \{z^{(\ell)}\}_{\ell=1}^{L}$



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

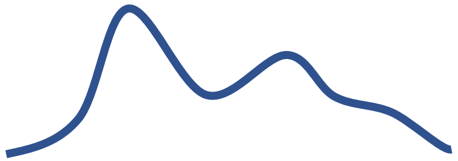
$$f_{in}^{(\ell)} = f^{(\ell-1)}(x; \mathcal{Z}) \circ z^{(\ell)}$$

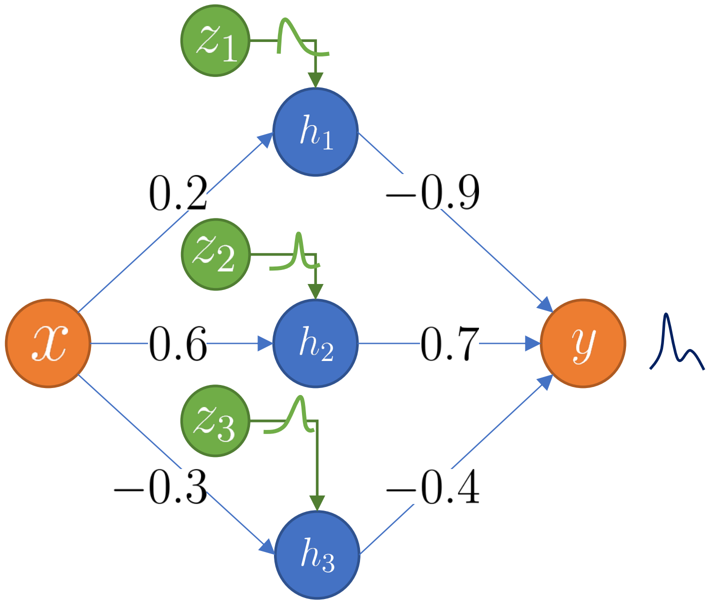


Node BNNs

NodeBNN with latent variables $\mathcal{Z} = \{z^{(\ell)}\}_{\ell=1}^{L}$

$p(\mathcal{Z})$





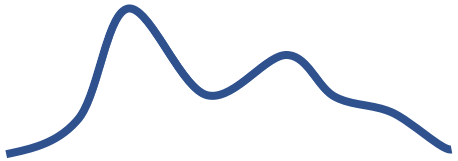
$$f^{(\ell)}(x; \mathcal{Z}) = \sigma\left(W^{(\ell)} f_{in}^{(\ell)} + b^{(\ell)}\right)$$

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

Previous layer's output

Node BNNs

NodeBNN with latent variables $\mathcal{Z} = \{z^{(\ell)}\}_{\ell=1}^{L}$

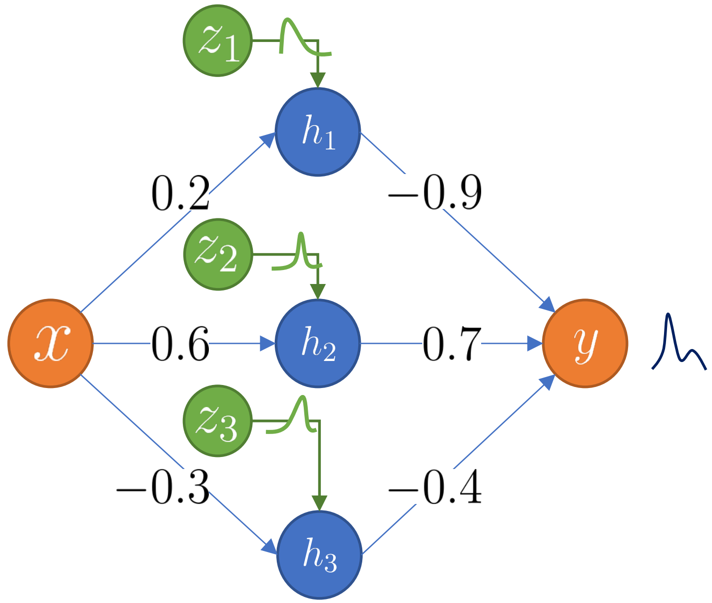
$p(\mathcal{Z})$ 

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

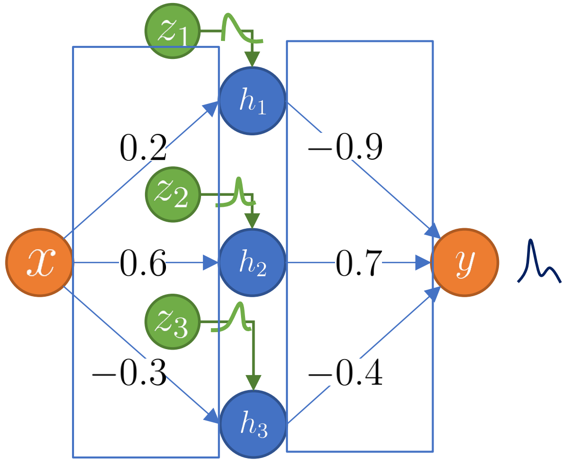
$$f_{in}^{(\ell)} = f^{(\ell-1)}(x;\mathcal{Z}) \circ z^{(\ell)}$$

Previous layer's output

Latent node variables



Node BNNs

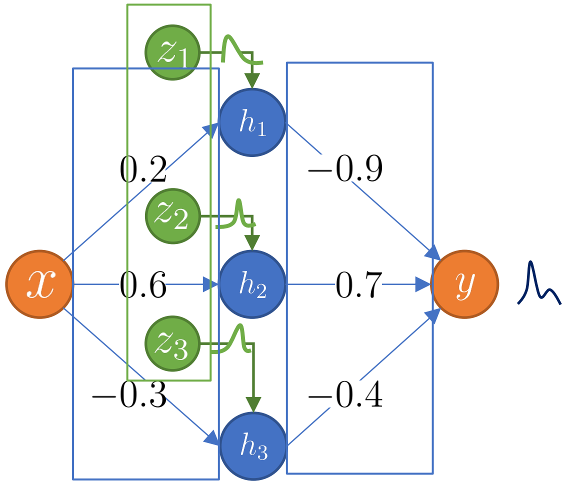


Two types of parameters:

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

   ➔ Pretrained or MAP solution

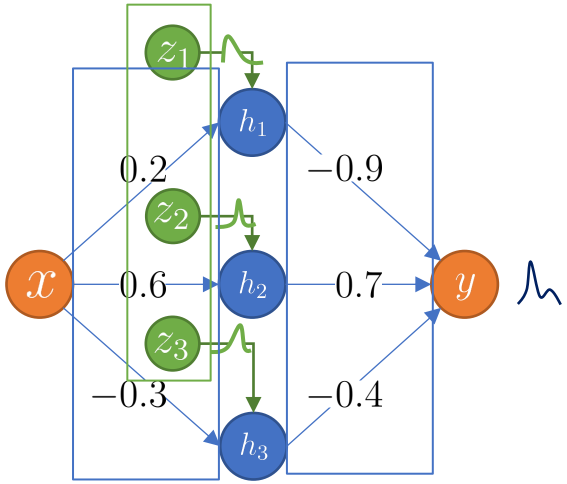
Node BNNs



Two types of parameters:

1. Weights and biases $\theta = \{(W^{(\ell)}, b^{(\ell)})\}_{\ell=1}^{L}$
   ➔ Pretrained or MAP solution
2. Node variables $\mathcal{Z} = \{z^{(\ell)}\}_{\ell=1}^{L}$
   ➔ Infer posterior

Node BNNs

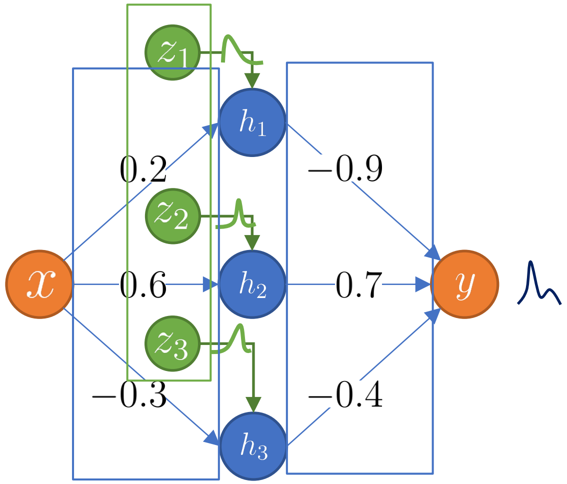


Network	Layers	Parameters: weights	nodes	w/n ratio
LeNet	5	42K	23	1800x
AlexNet	8	61M	18,307	3300x
VGG16-small	16	15M	5,251	2900x
VGG16-large	16	138M	36,995	3700x
ResNet50	50	26M	24,579	1000x
WideResNet-28x10	28	36M	9,475	3800x

Two types of parameters:

1. Weights and biases $\theta = \{(W^{(\ell)}, b^{(\ell)})\}_{\ell=1}^{L}$
   ➔ Pretrained or MAP solution
2. Node variables $\mathcal{Z} = \{z^{(\ell)}\}_{\ell=1}^{L}$
   ➔ Infer posterior

Node BNNs



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

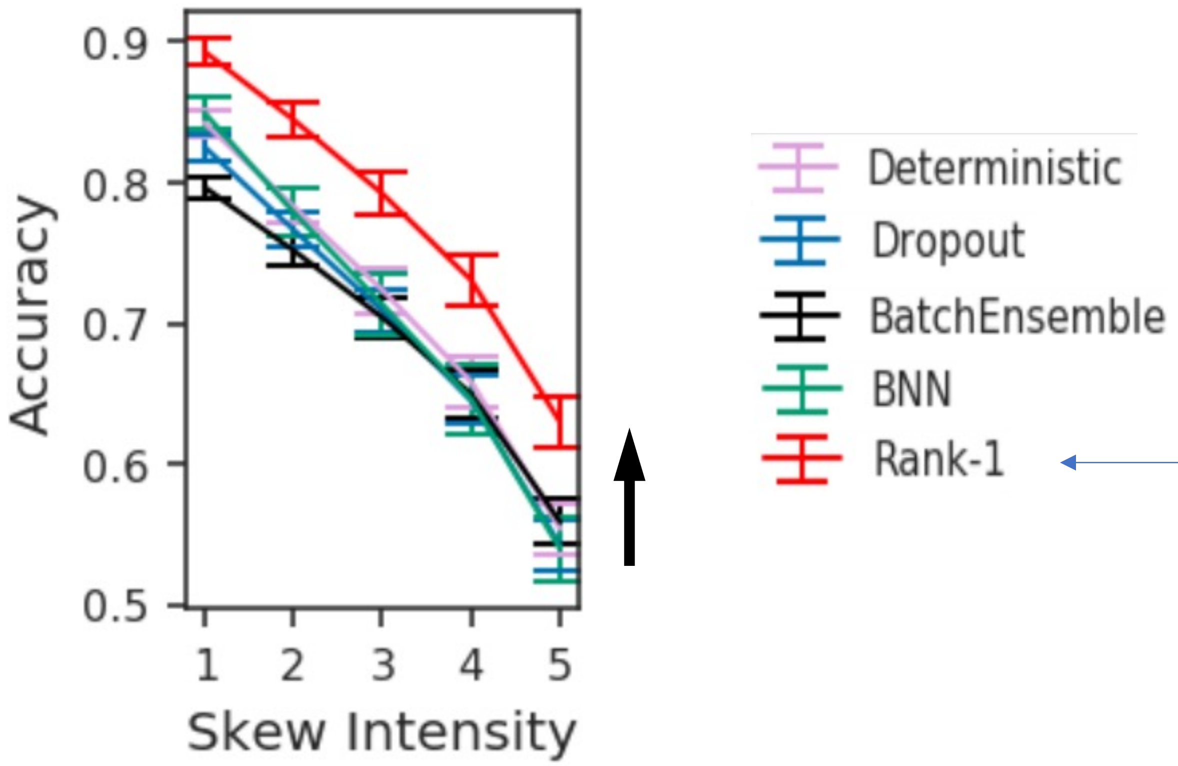
   ➔ Pretrained or MAP solution

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

   ➔ Infer posterior

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

Node-BNNs outperform MAP under corruptions



Node-BNN

WideResNet-28-10 / CIFAR-10-C

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

Our paper’s goals

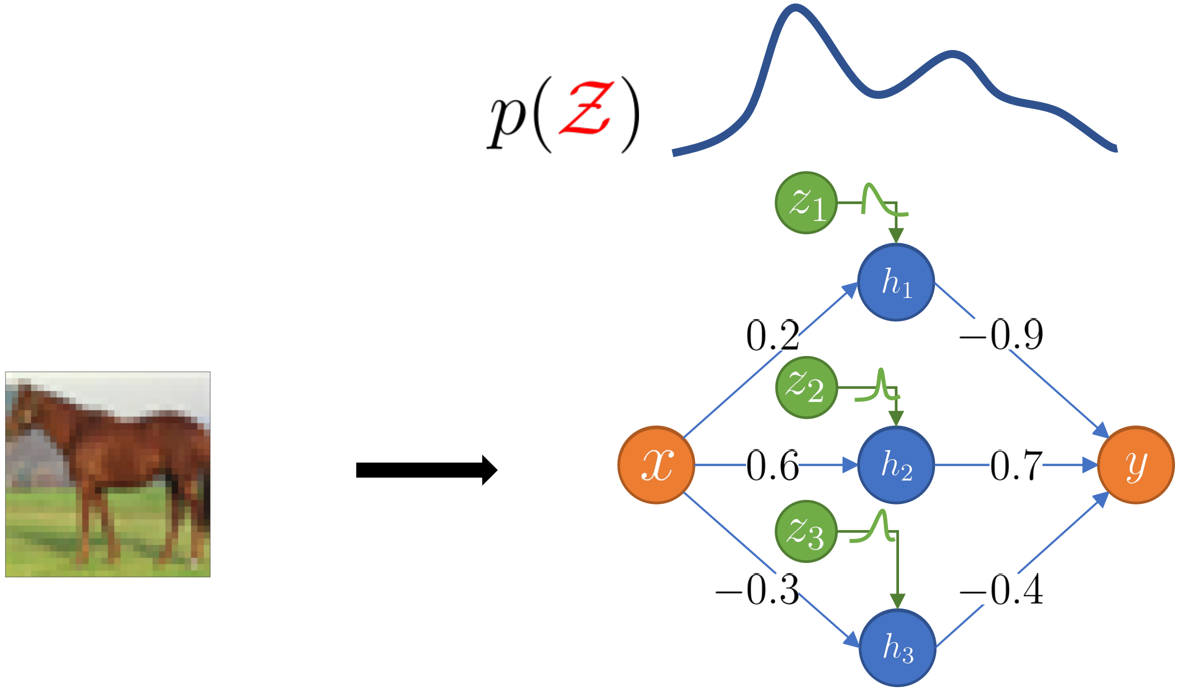
Providing insights into the robustness of node-BNNs under input corruptions.

↓

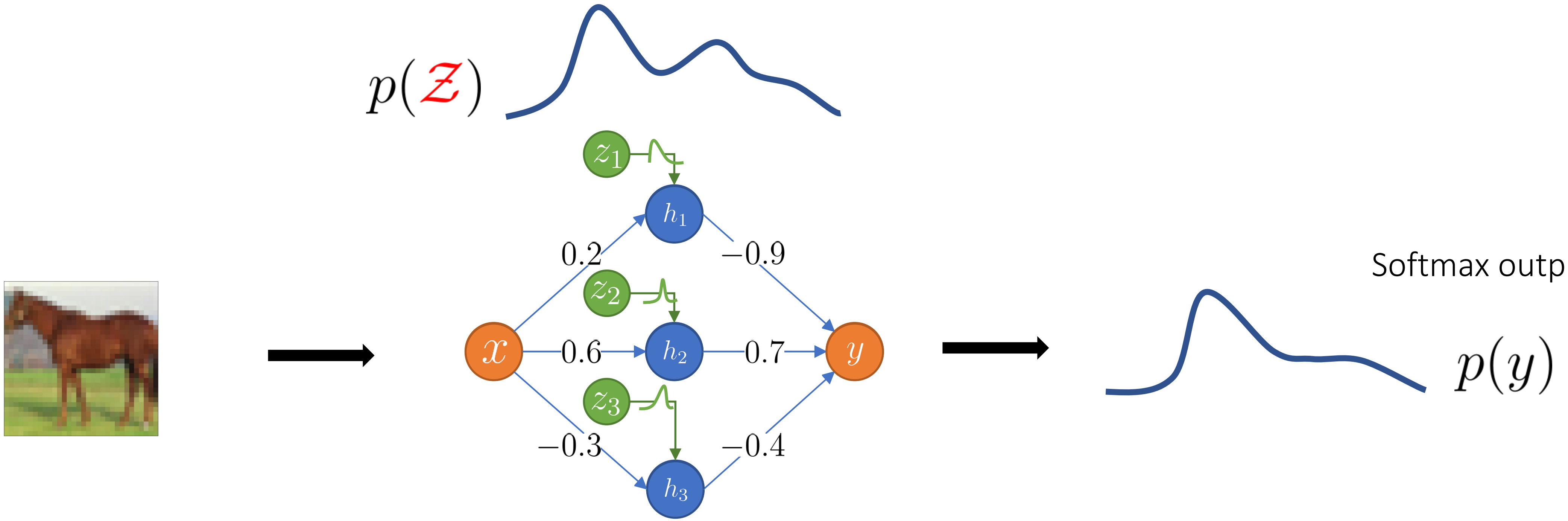
Proposing a method to improve the robustness of node-BNNs

Why do node-BNNs generalize better under input corruptions?

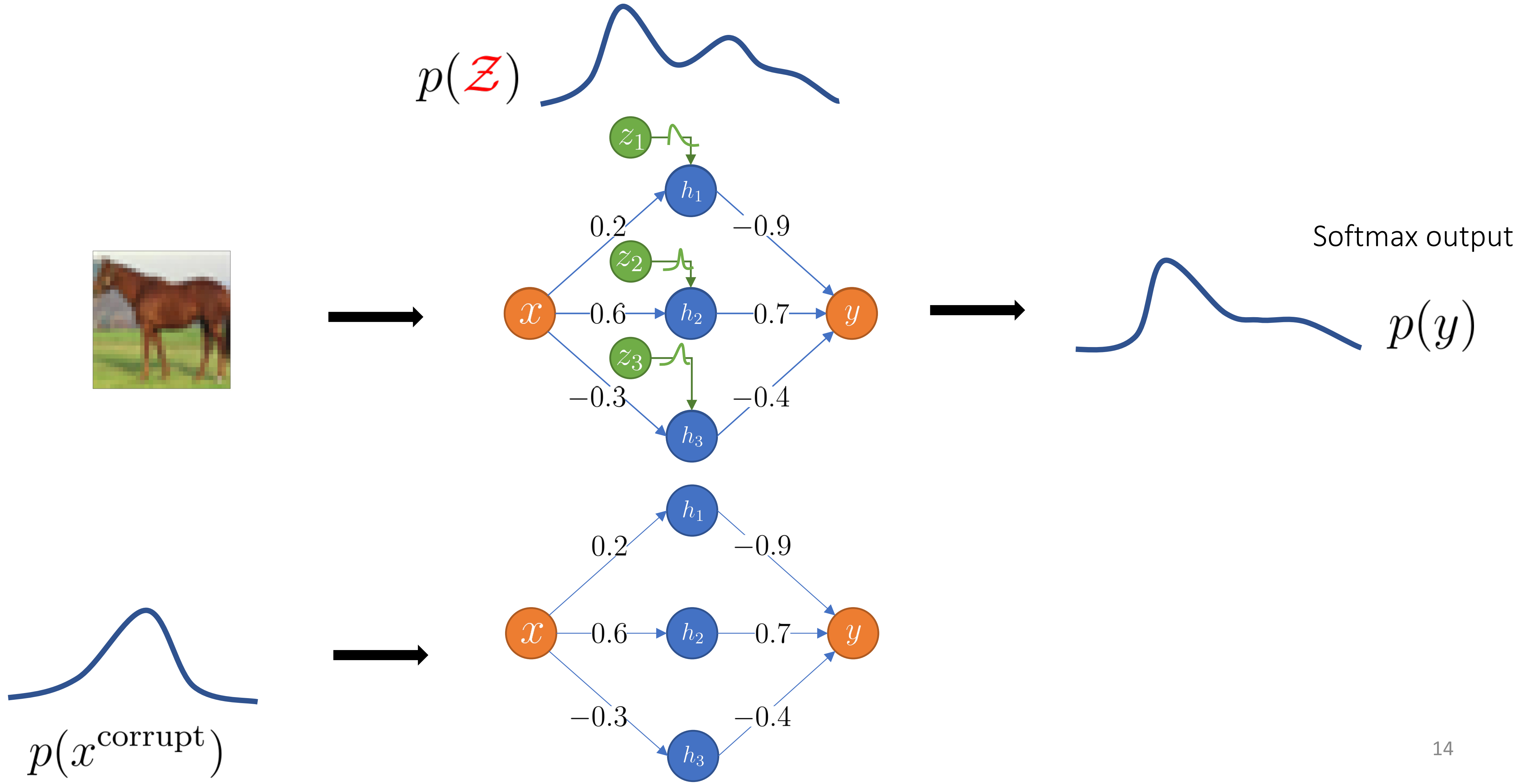
The latent distribution $p(\mathcal{Z})$ induces implicit input distribution $p(x^{\text{corrupt}})$



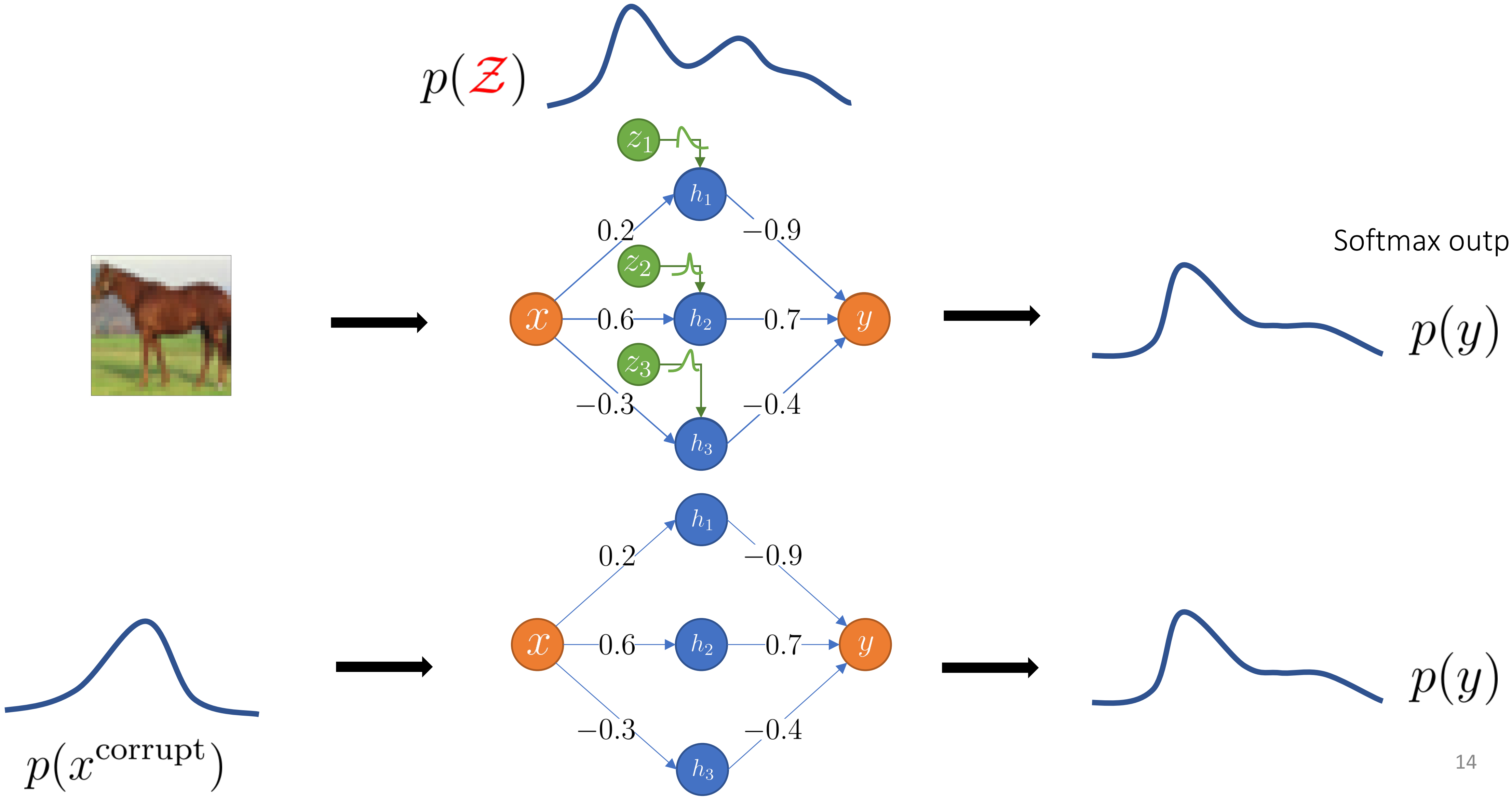
The latent distribution $p(\mathcal{Z})$ induces implicit input distribution $p(x^{\text{corrupt}})$



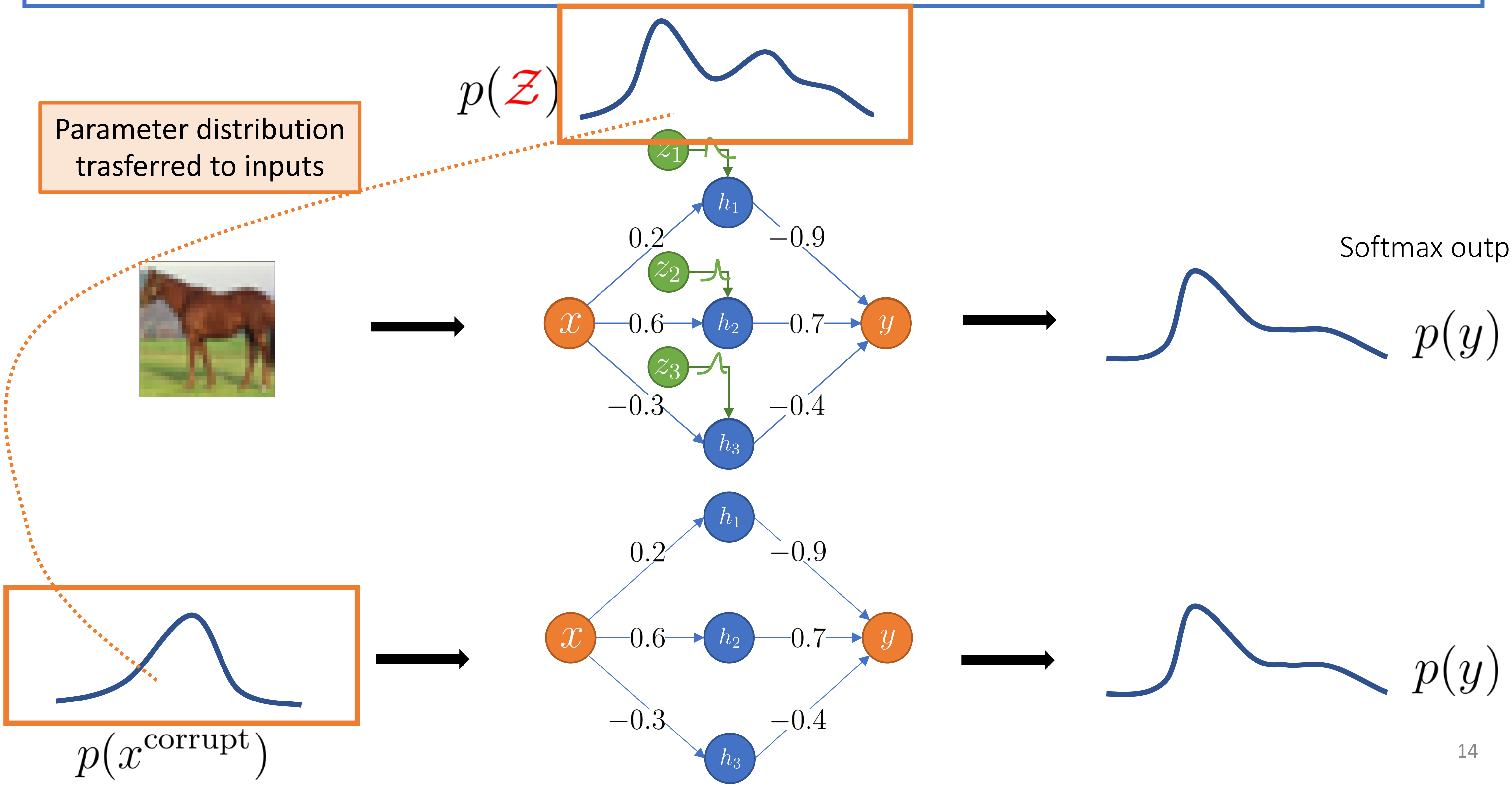
The latent distribution $p(\mathcal{Z})$ induces implicit input distribution $p(x^{\text{corrupt}})$



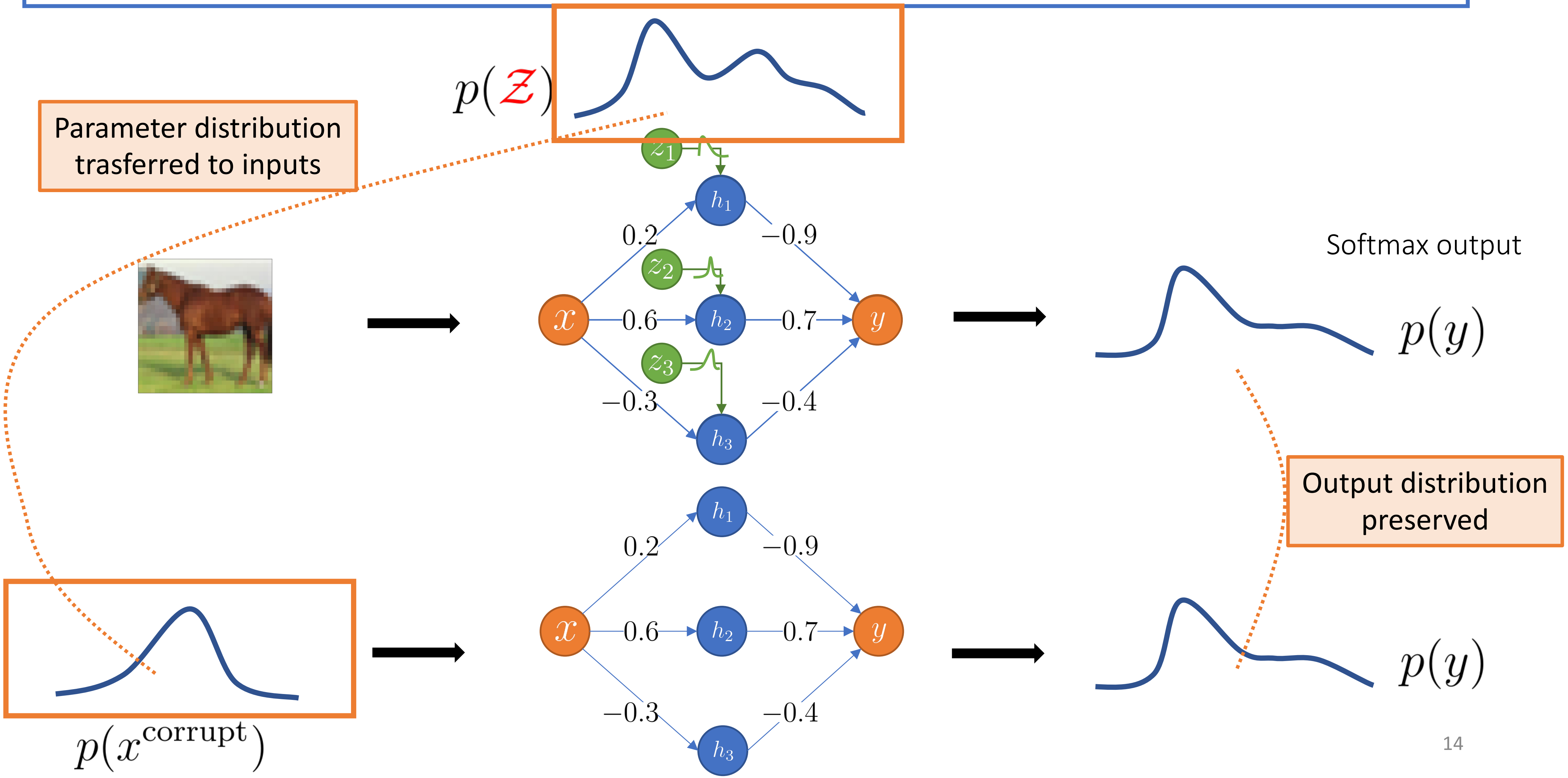
The latent distribution $p(\mathcal{Z})$ induces implicit input distribution $p(x^{\text{corrupt}})$



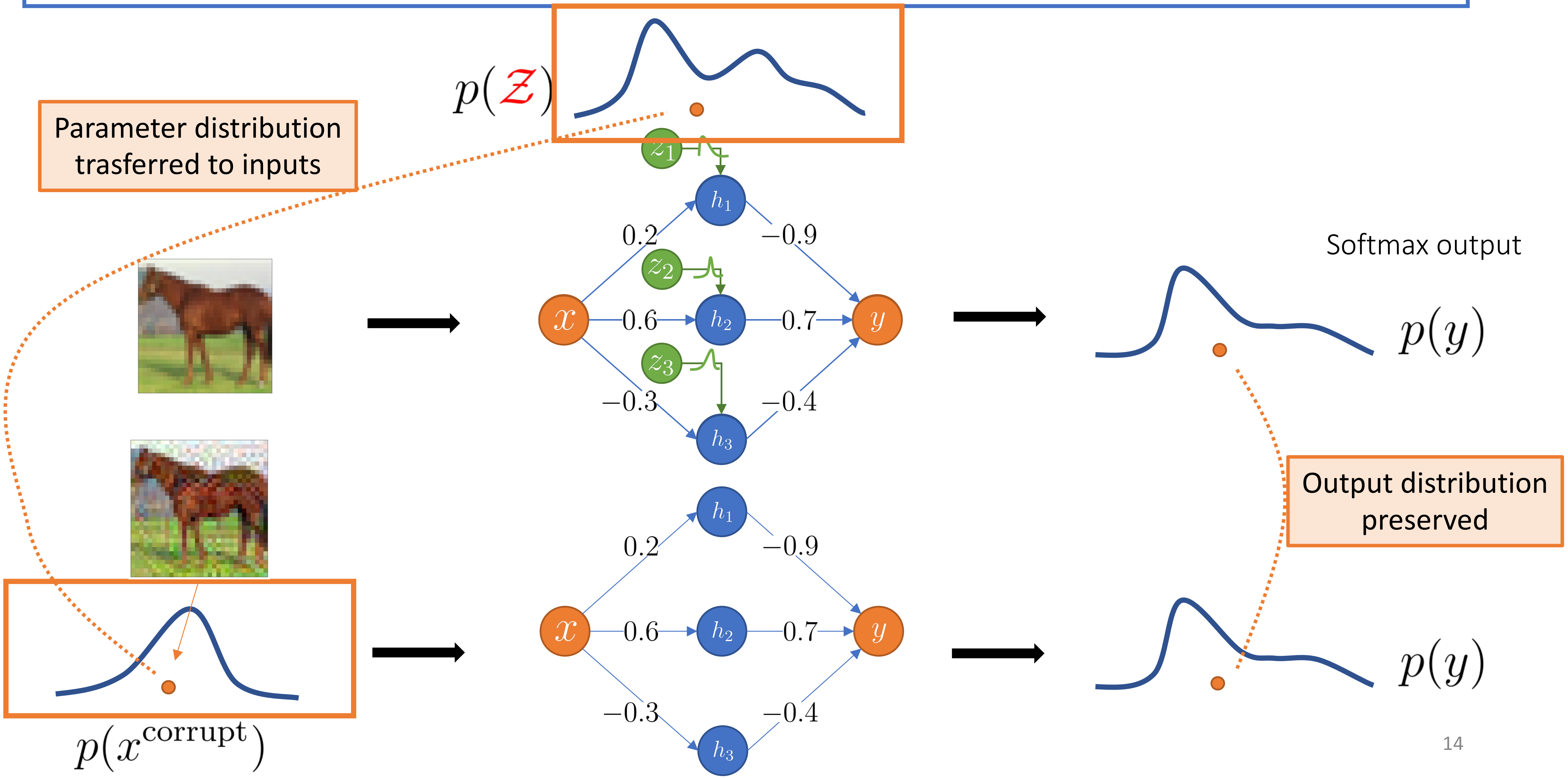
The latent distribution $p(\mathcal{Z})$ induces implicit input distribution $p(x^{\text{corrupt}})$



The latent distribution $p(\mathcal{Z})$ induces implicit input distribution $p(x^{\text{corrupt}})$



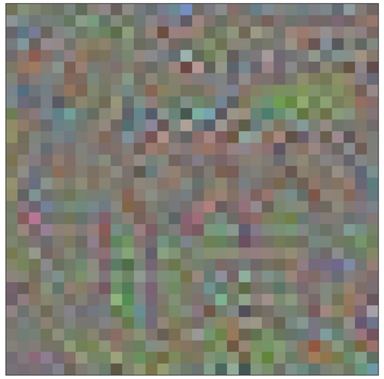
The latent distribution $p(\mathcal{Z})$ induces implicit input distribution $p(x^{\text{corrupt}})$



Finding the implicit corruption

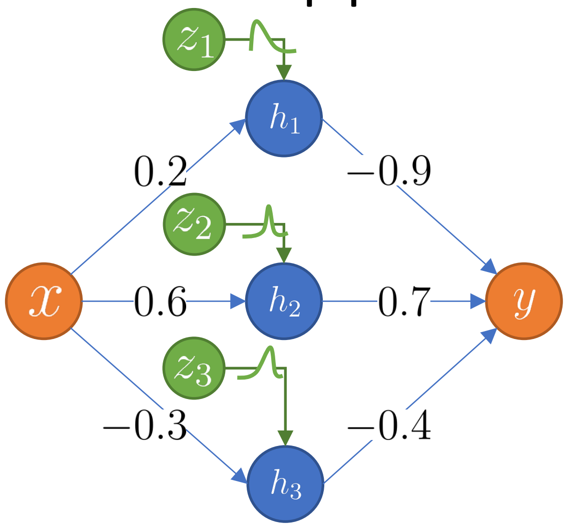
$=$

$+$

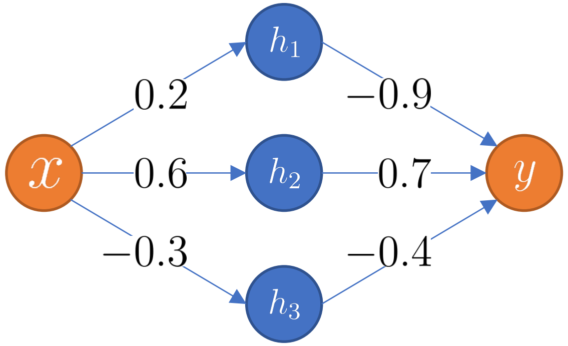


$$x_{corrupt} \qquad x \qquad m$$

Approximating the implicit corruption



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

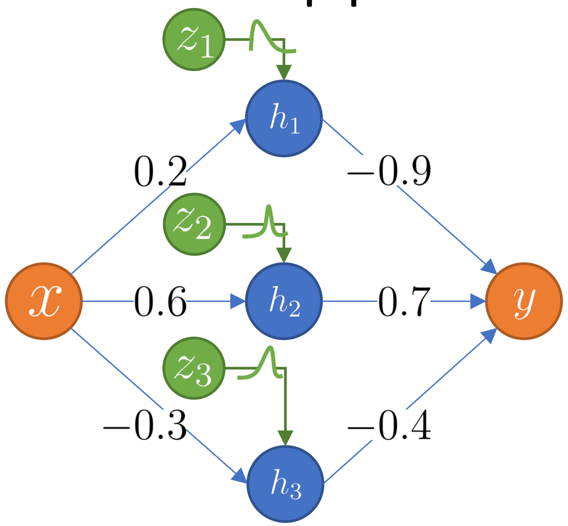


$$\hat{f}(x) = f(x; \mathcal{Z} = \mathbf{1})$$

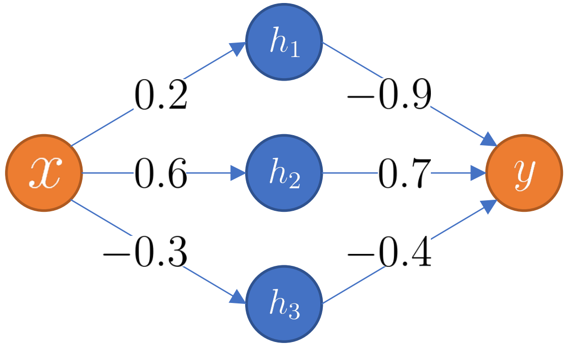
Given $\mathcal{Z} \sim p(\mathcal{Z})$, approximate m by minimizing

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

Approximating the implicit corruption



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



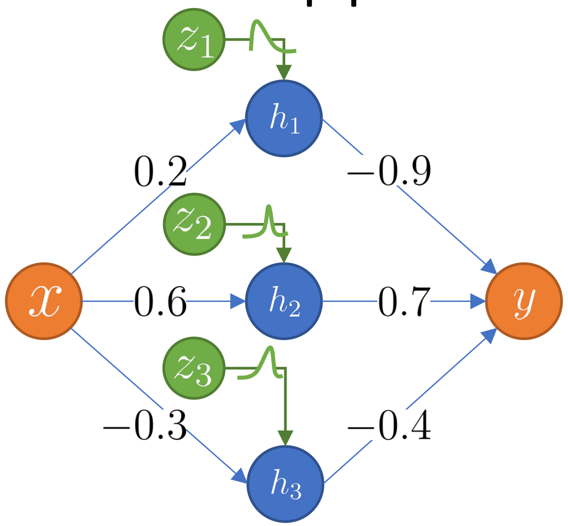
$$\hat{f}(x) = f(x; \mathcal{Z} = \mathbf{1})$$

Given $\mathcal{Z} \sim p(\mathcal{Z})$, approximate m by minimizing

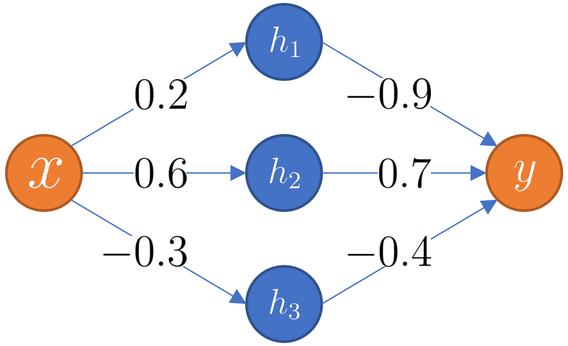
$$\boxed{\frac{1}{2}\left|\left|f(x; \mathcal{Z}) - \hat{f}(x+m)\right|\right|_2^2} + \frac{\lambda}{2}||m||_2^2$$

Output matching

Approximating the implicit corruption



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



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

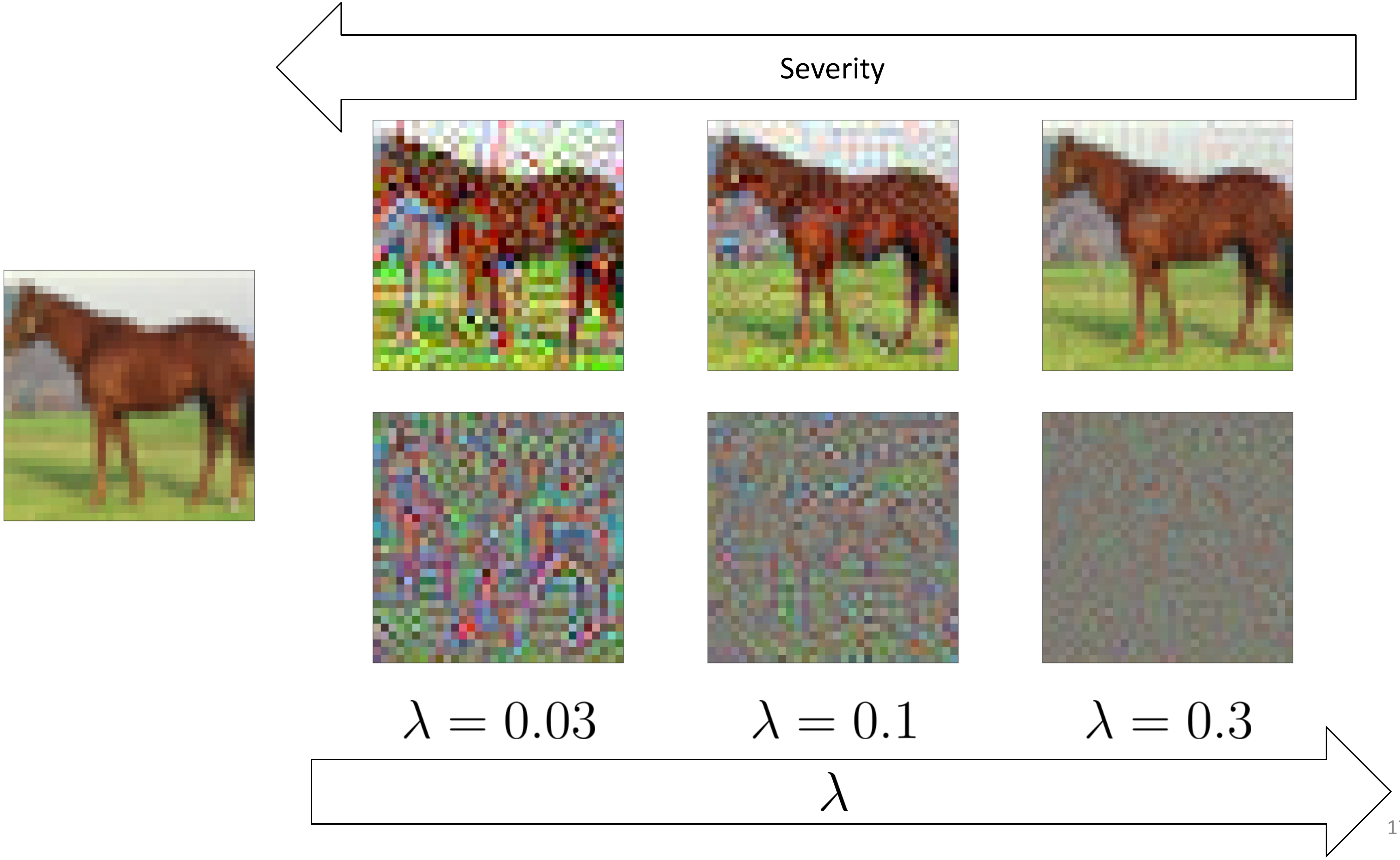
Given $\mathcal{Z} \sim p(\mathcal{Z})$, approximate m by minimizing

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

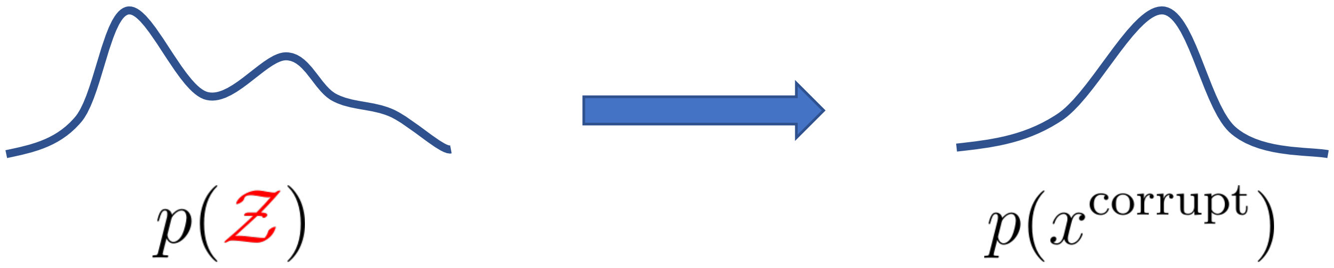
Output matching

L2-regularization

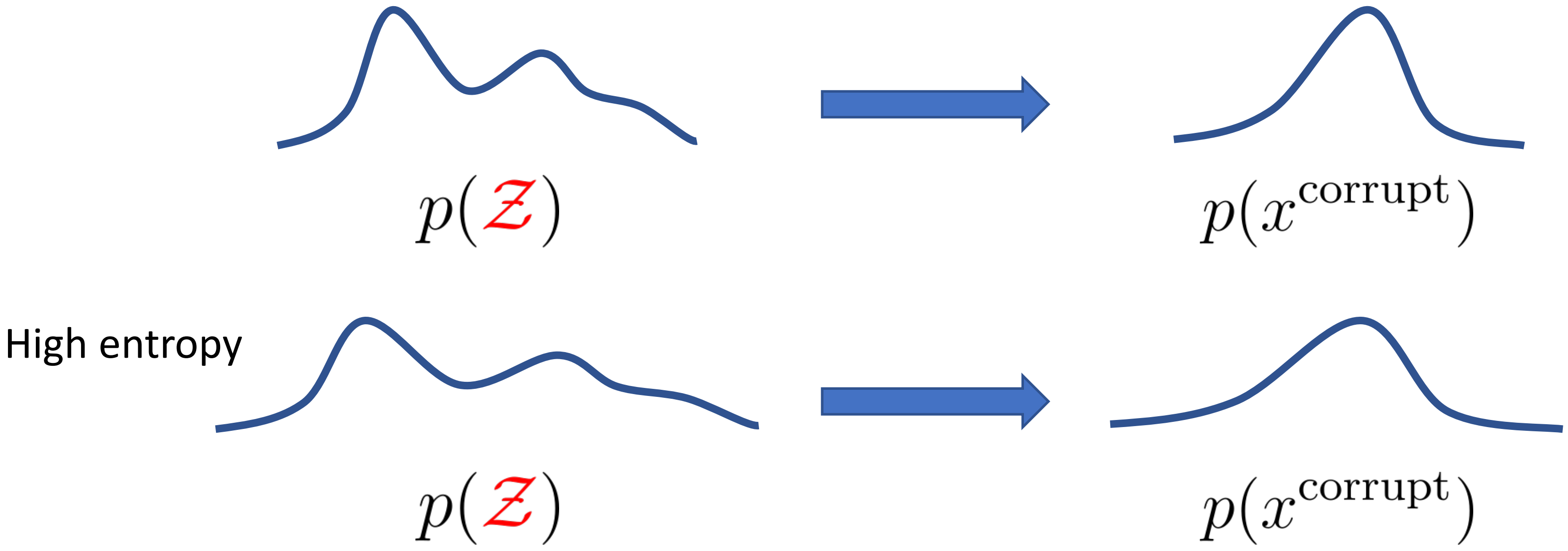
Example of implicit corruptions



Entropy of latent variables and implicit corruptions



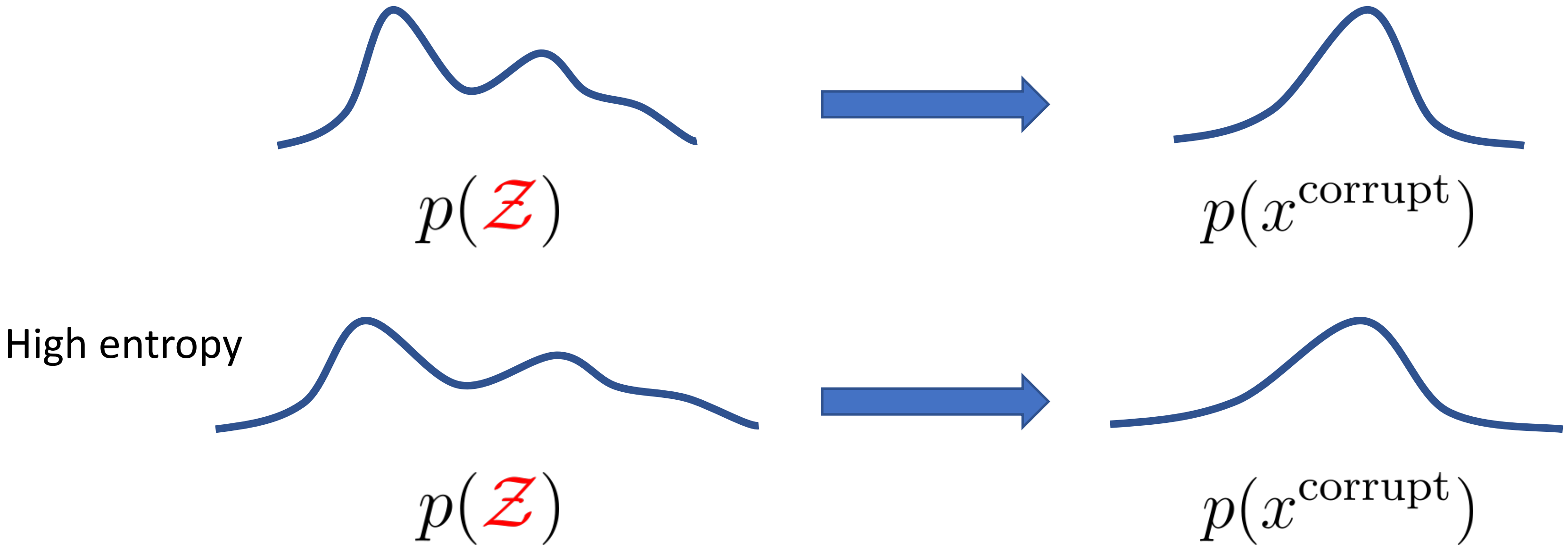
Entropy of latent variables and implicit corruptions



We show:

1. Increasing entropy of latent variables z increase the diversity of implicit corruptions

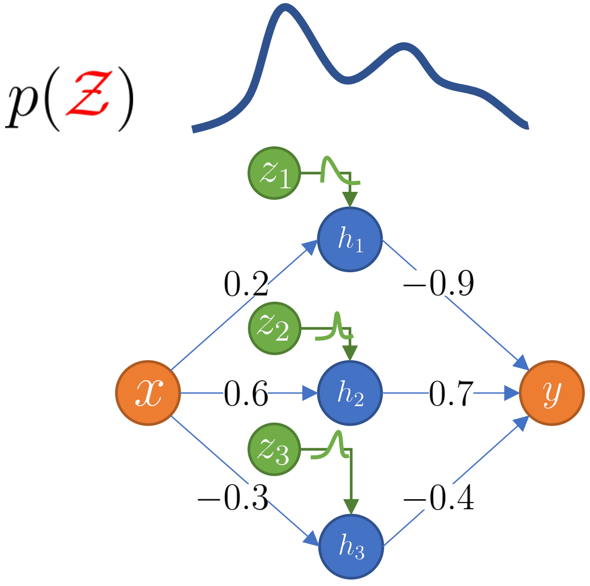
Entropy of latent variables and implicit corruptions



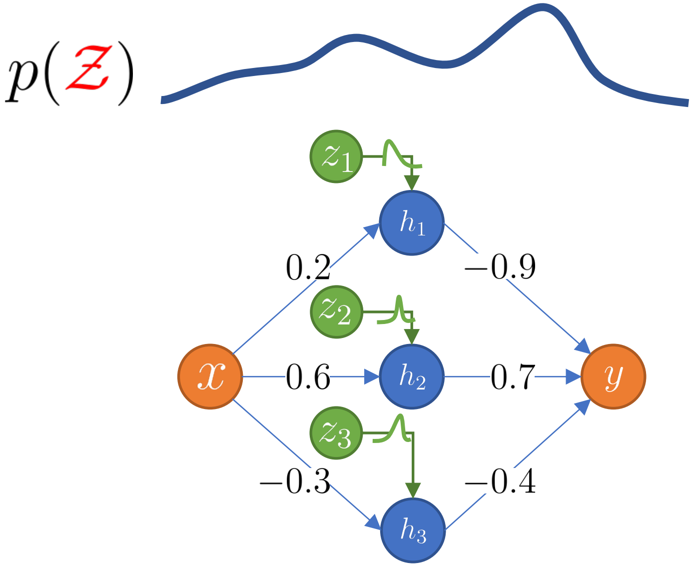
We show:

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

High entropy = more robust node-BNNs?



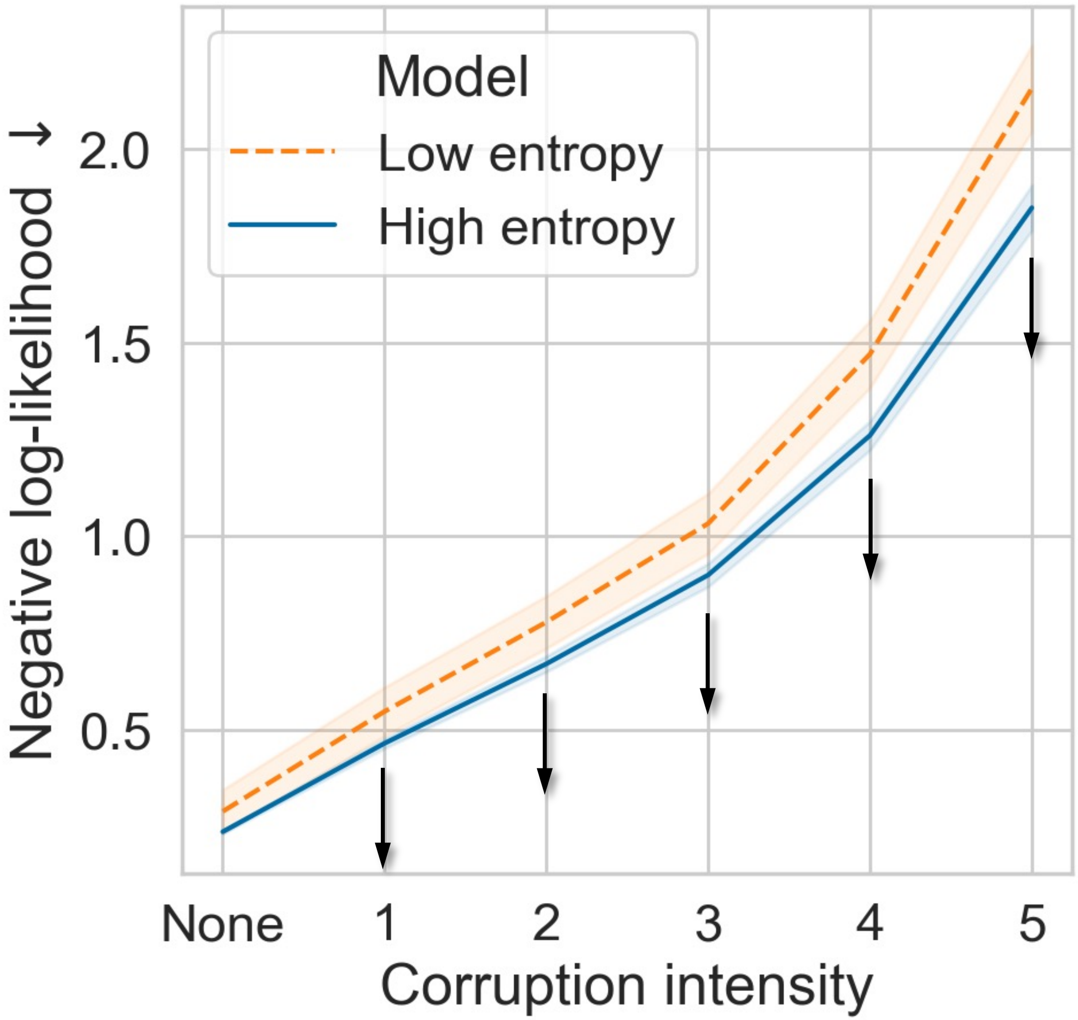
Low entropy model



High entropy model

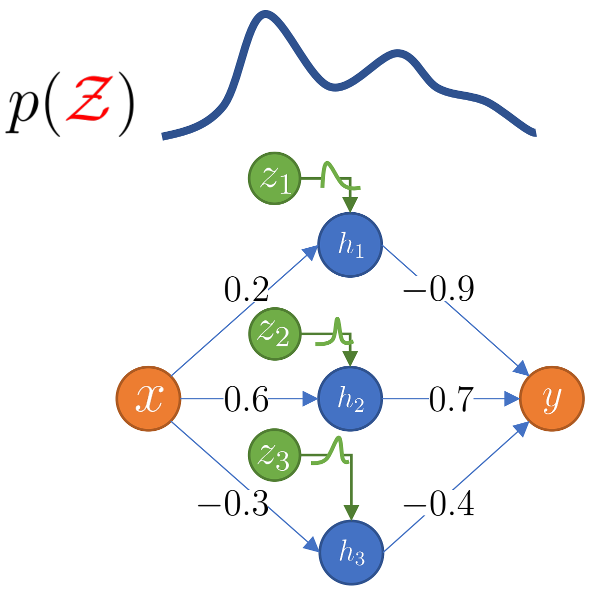
Same ConvNet architecture
Train on CIFAR-10
Test on CIFAR-10-C

High entropy = more robust node-BNNs?

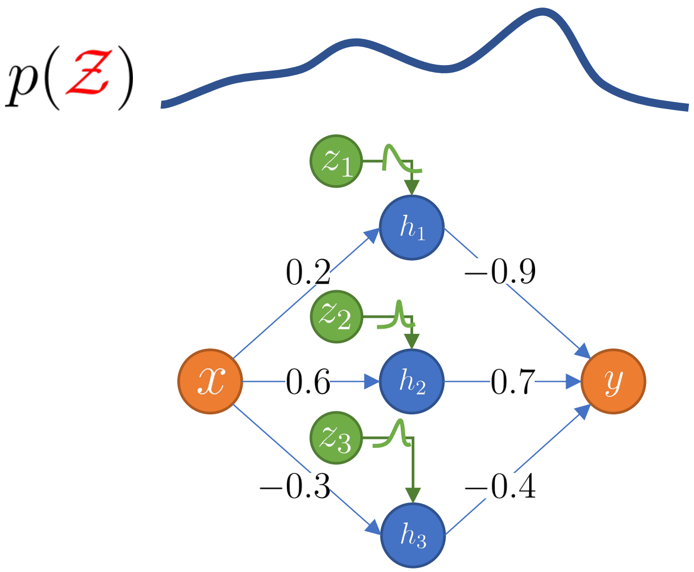


YES

Is a model robust against its own corruptions?



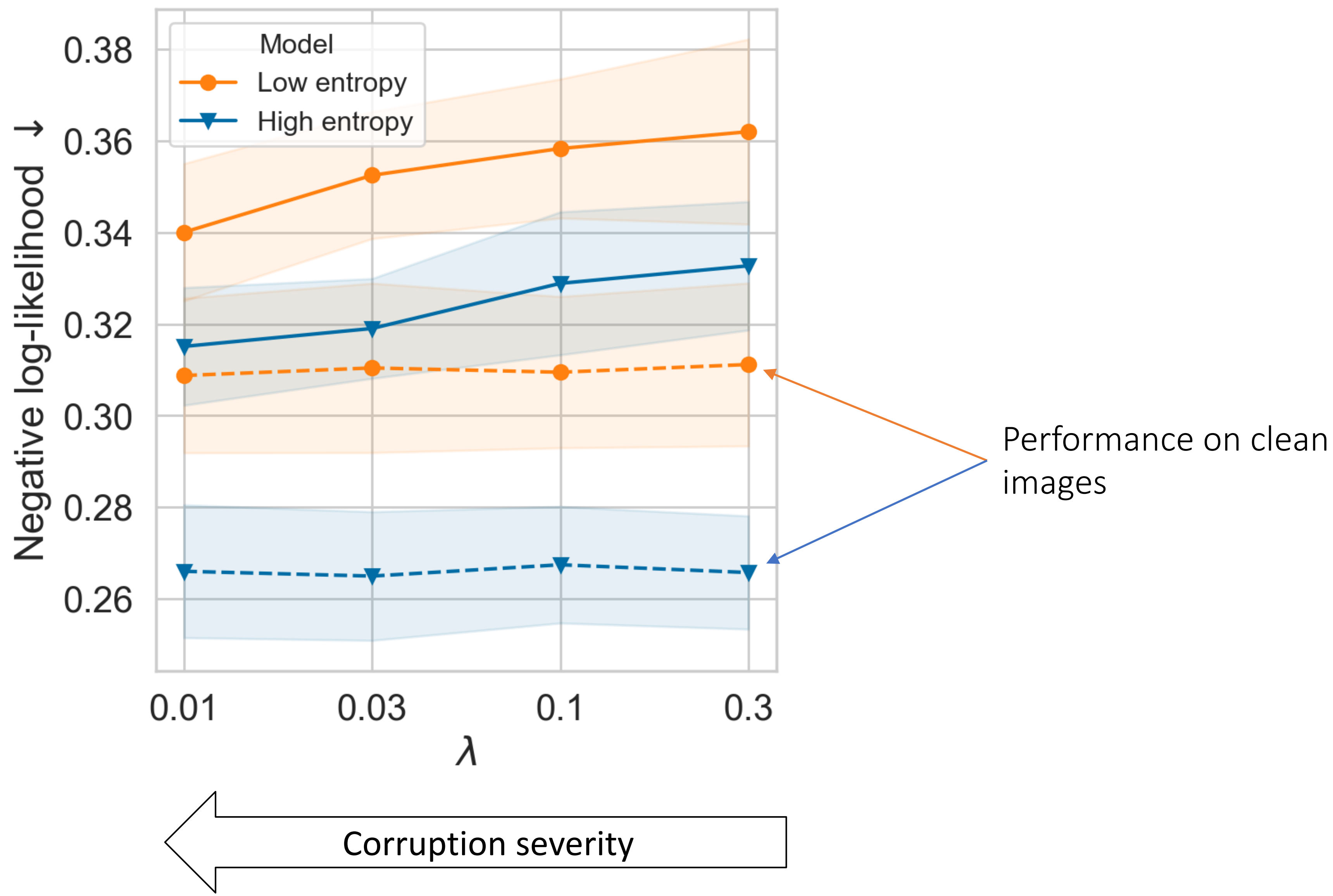
Low entropy model



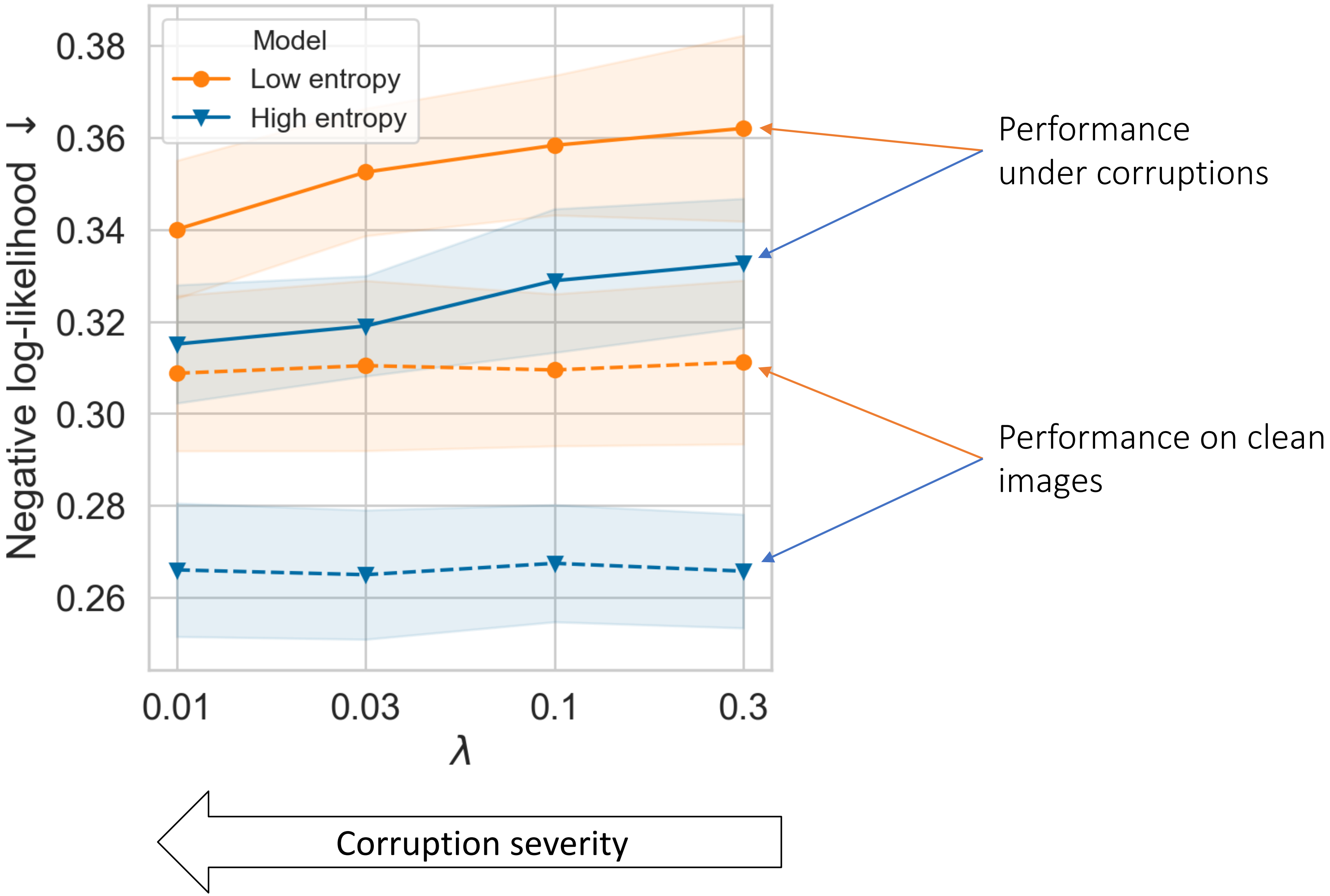
High entropy model

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?

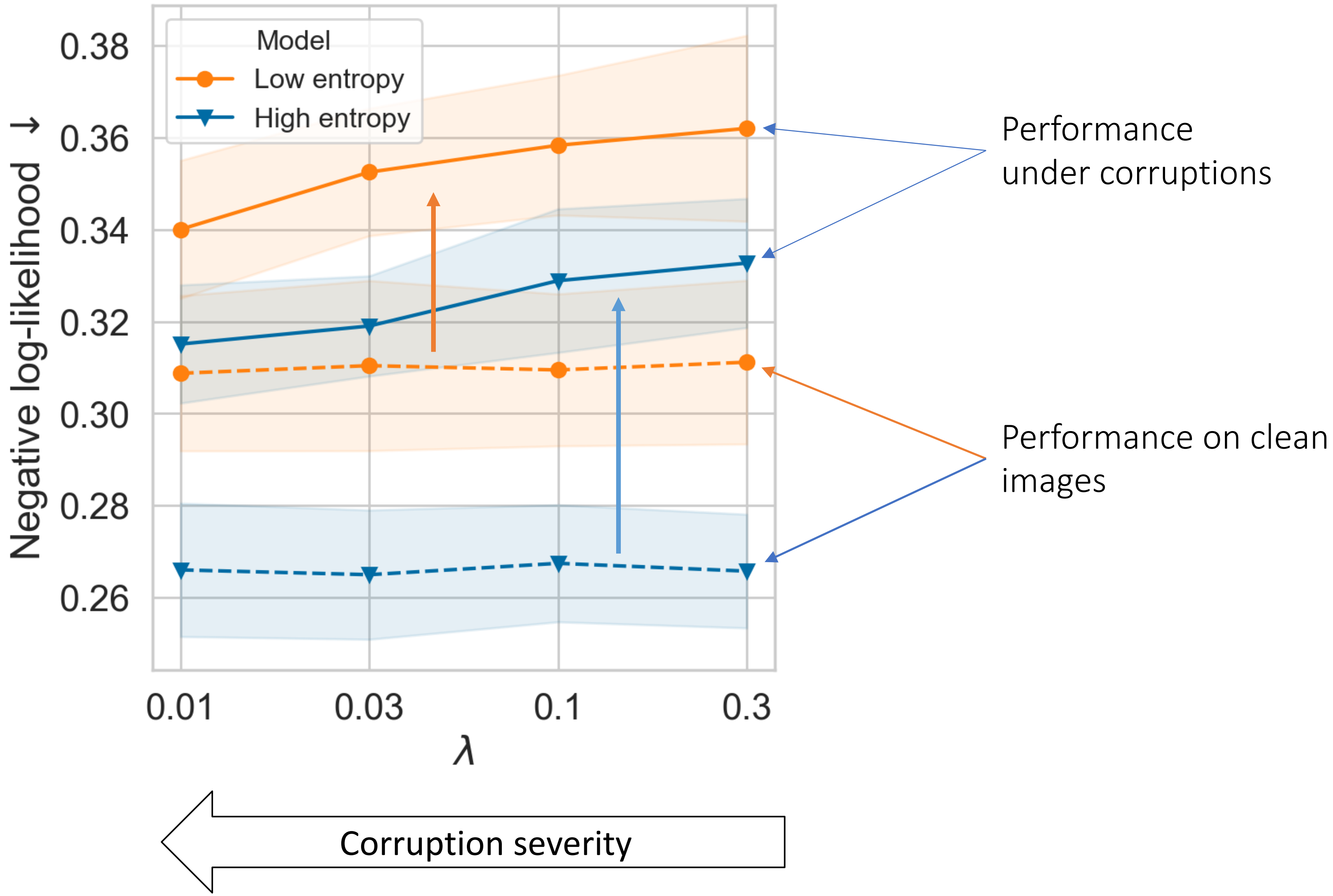


Is a model robust against its own corruptions?

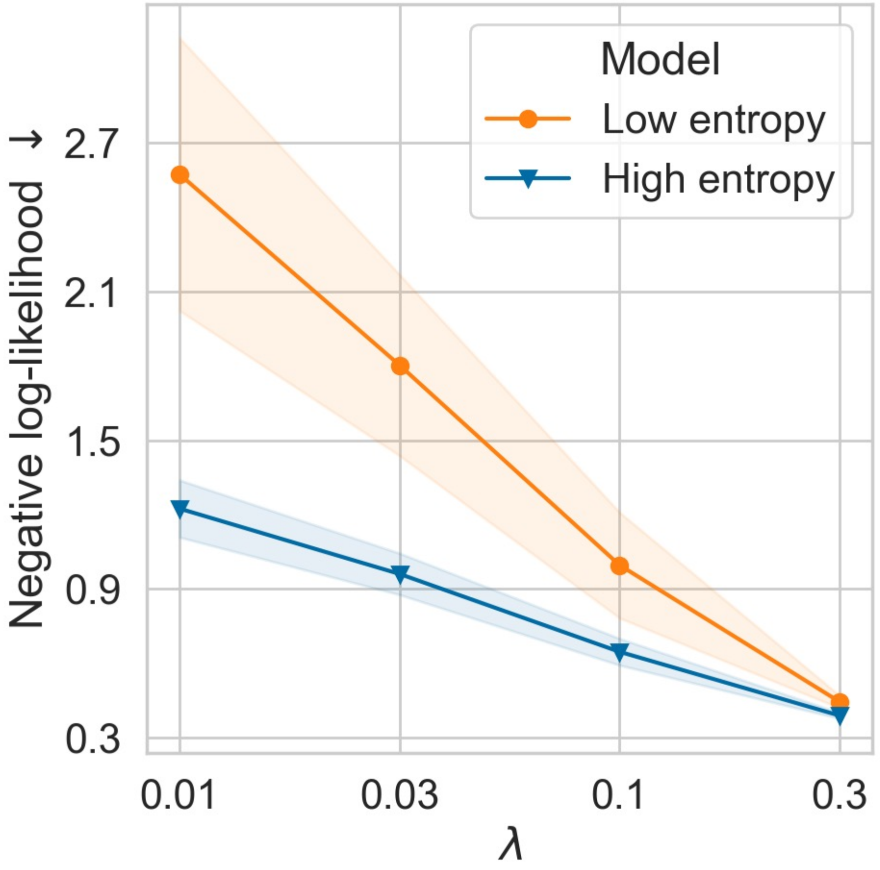


Is a model robust against its own corruptions?

Yes! (in this small experiment)



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



Corruption severity

The high-entropy model can handle corruptions better

How to increase the latent entropy?

Variational inference

Complex posterior distribution

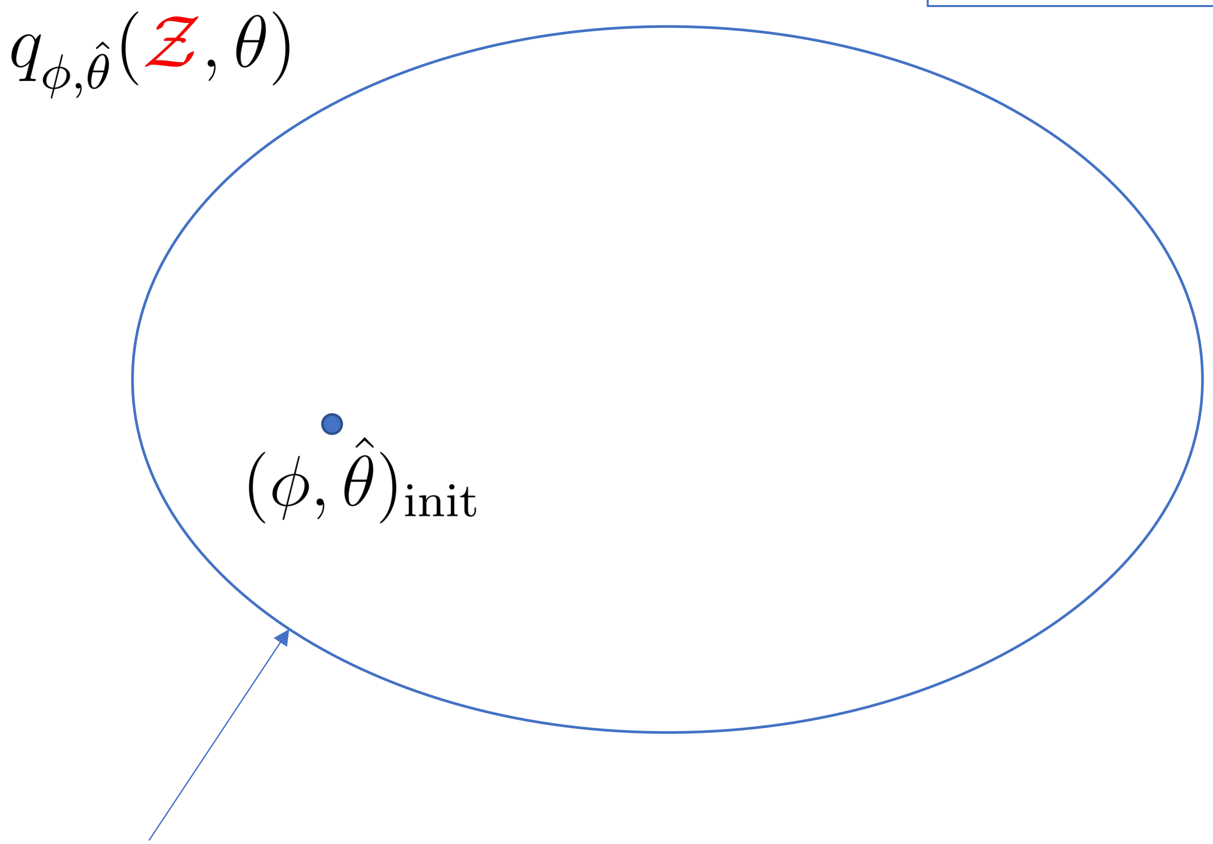
$$\boxed{p(\mathcal{Z}, \theta|\mathcal{D})} \propto p(\mathcal{D}|\theta, \mathcal{Z})p(\theta)p(\mathcal{Z})$$

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

Variational inference

Complex posterior distribution

$$p(\mathcal{Z}, \theta | \mathcal{D}) \propto p(\mathcal{D}|\theta, \mathcal{Z})p(\theta)p(\mathcal{Z})$$



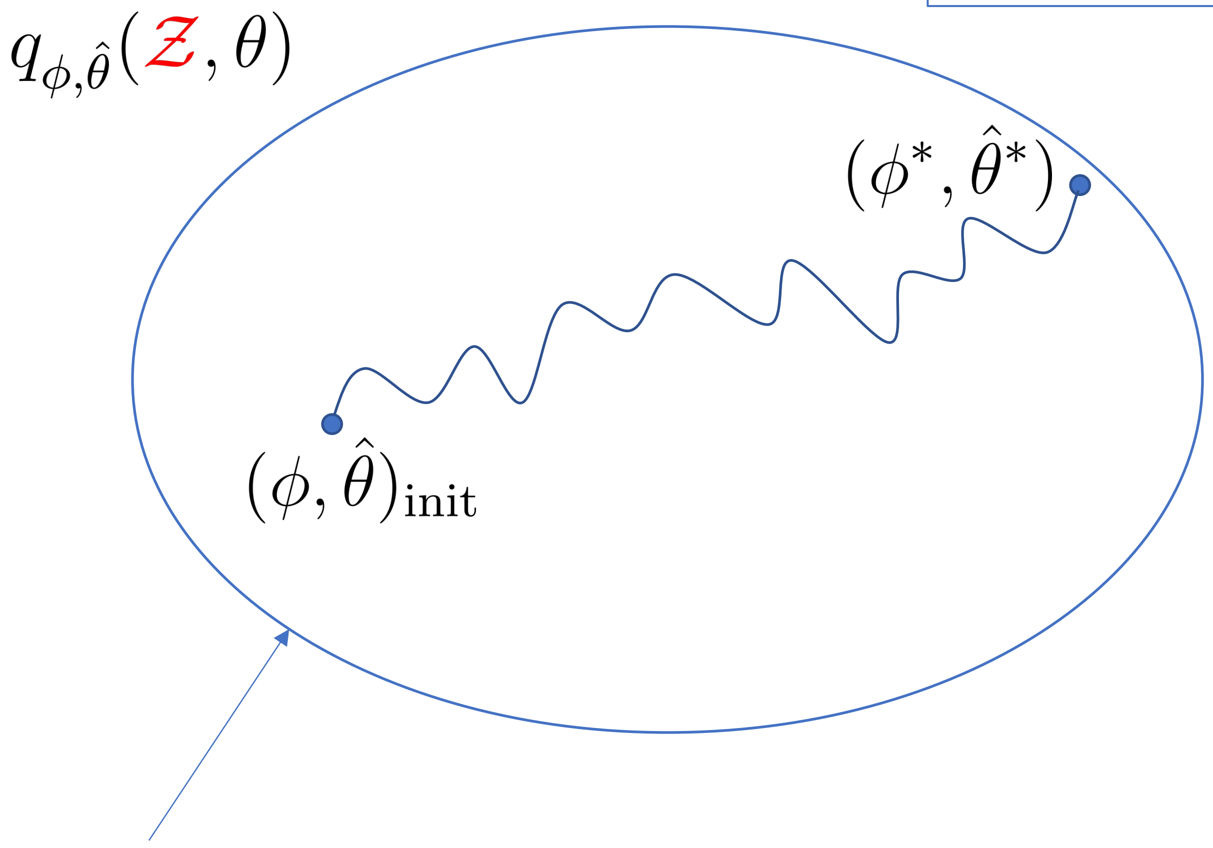
Simple, parametric distribution

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

Variational inference

Complex posterior distribution

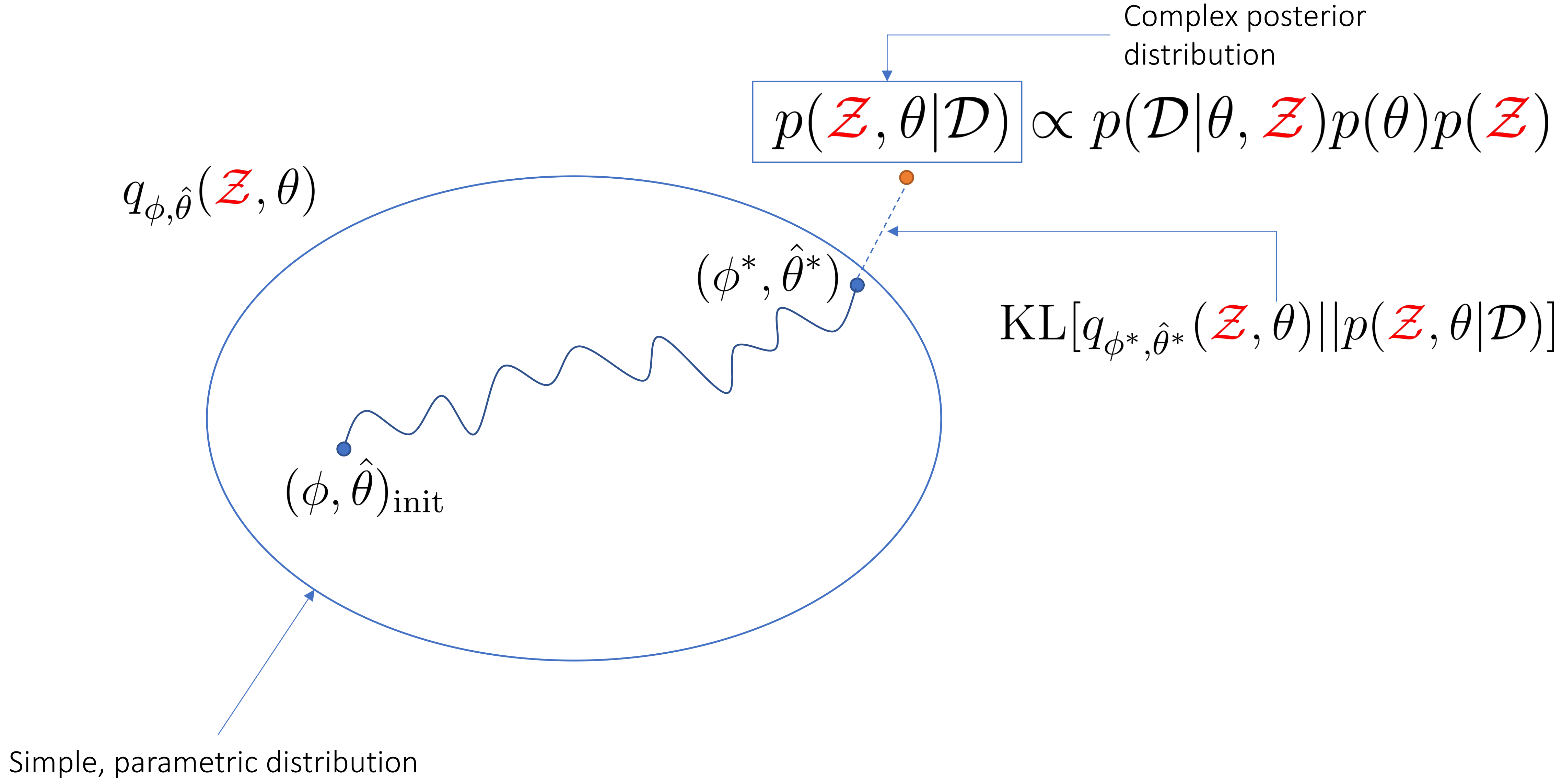
$$p(\mathcal{Z}, \theta | \mathcal{D}) \propto p(\mathcal{D} | \theta, \mathcal{Z}) p(\theta) p(\mathcal{Z})$$



Simple, parametric distribution

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

Variational inference



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

Variational posterior

$$
\begin{aligned}
q_{\phi,\hat{\theta}}(\mathcal{Z}, \theta) &= q_{\hat{\theta}}(\theta) q_{\phi}(\mathcal{Z}) \\
&= \delta(\theta - \hat{\theta}) q_{\phi}(\mathcal{Z})
\end{aligned}
$$

Variational posterior

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

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

Dirac delta measure
(for MAP estimation)

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

ELBO optimization of $(\hat{\theta}, \phi)$

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

Evidence lower-bound (ELBO)

ELBO optimization of $(\hat{\theta}, \phi)$

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

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

The original ELBO

Entropic regularization

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

The original ELBO

The γ entropy

The γ – ELBO = tempered posterior

Maximizing the γ – ELBO is equivalent to minimizing:

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

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

Mandt et al (2016). Variational Tempering

The γ – ELBO = tempered posterior

Maximizing the γ – ELBO is equivalent to minimizing:

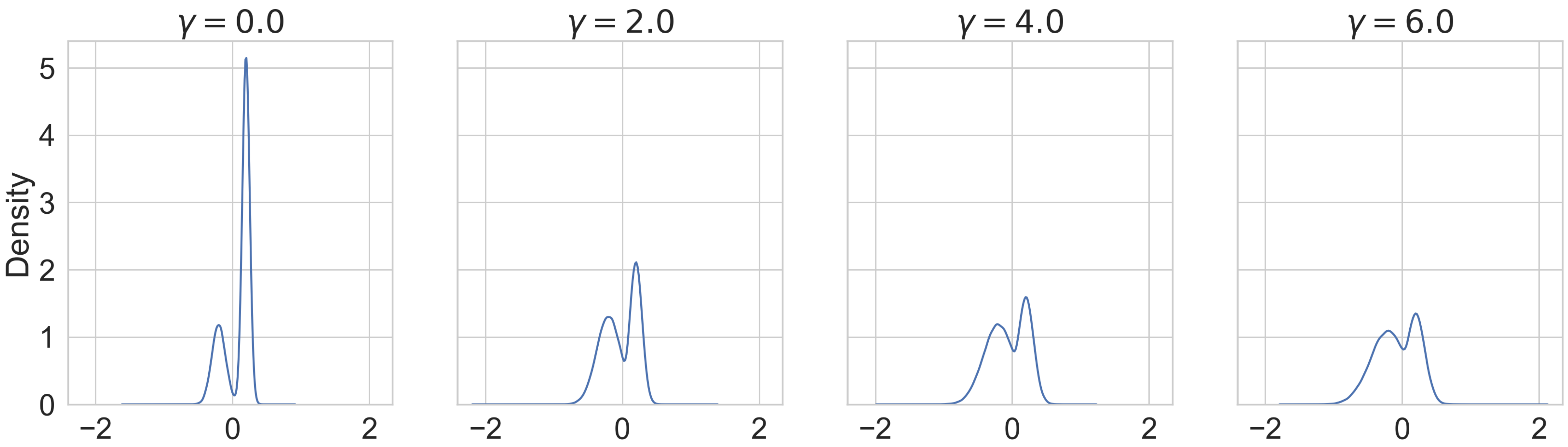
$$\mathrm{KL}[q_{\phi,\hat{\theta}}(\mathcal{Z},\theta)||p_\gamma(\mathcal{Z},\theta|\mathcal{D})]$$

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

Temperature $\tau = \gamma + 1$

Mandt et al (2016). Variational Tempering

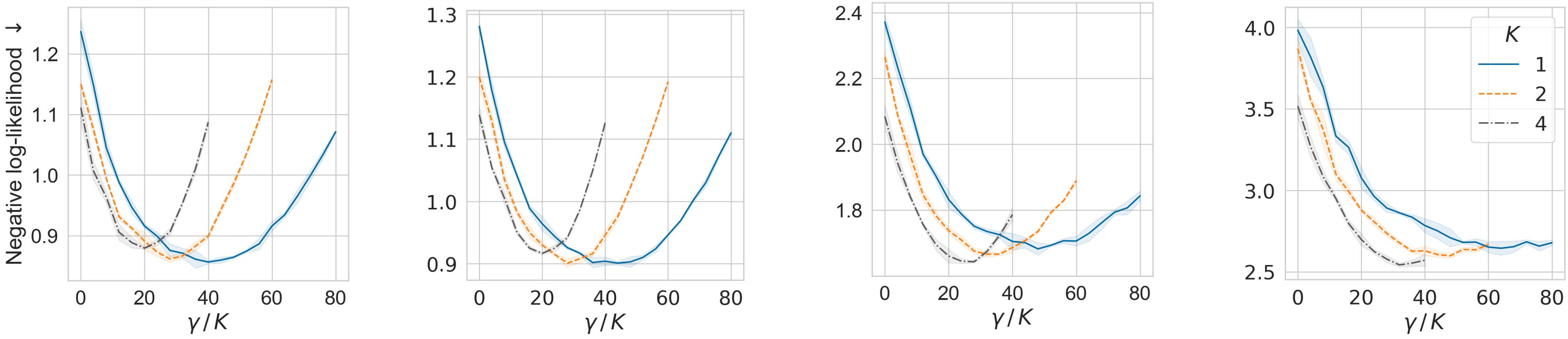
$\gamma > 0$ enlargens posterior



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

Experiments

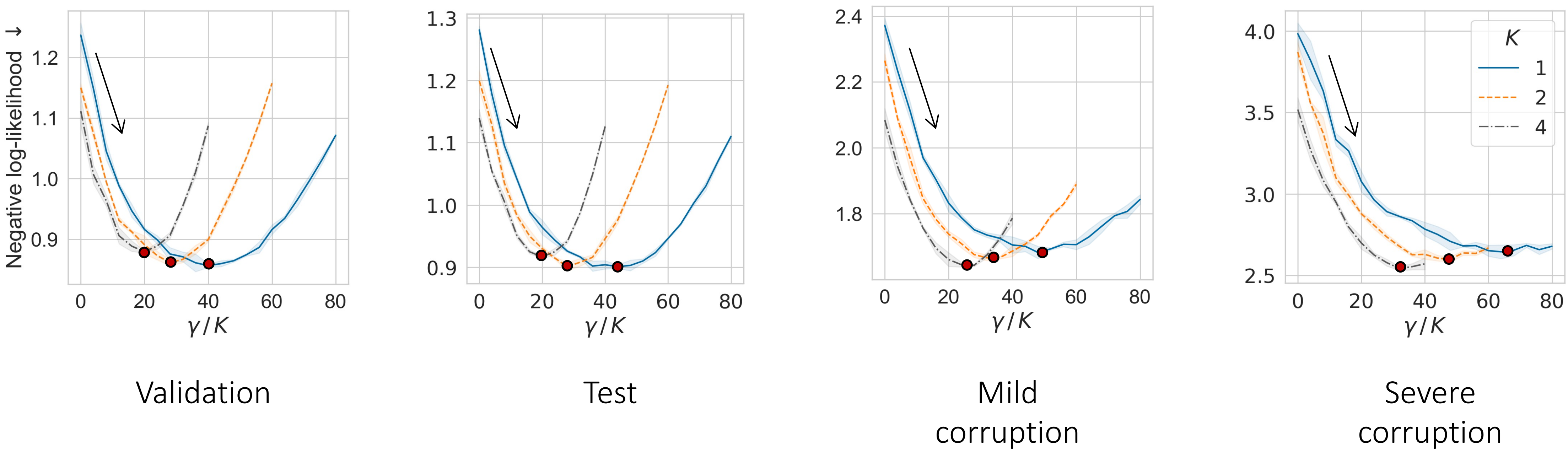
Effects of γ on corruption robustness



Validation — Test — Mild corruption — Severe corruption

Network: VGG16
Train: CIFAR-100
Test: CIFAR-100-C
K: number of Gaussian components in $q_\phi(\mathcal{Z})$

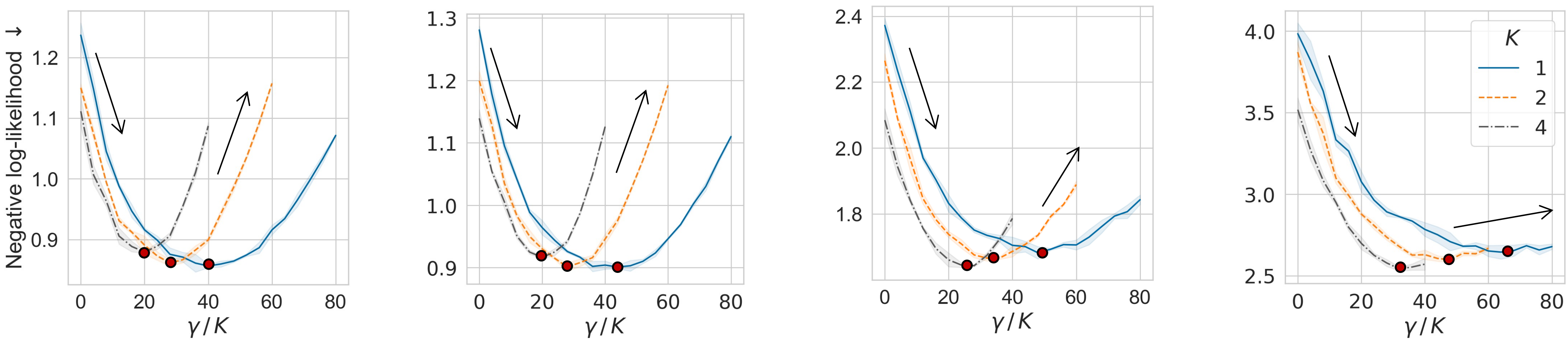
Effects of γ on corruption robustness



Network: VGG16
Train: CIFAR-100
Test: CIFAR-100-C
K: number of Gaussian components in $q_\phi(\mathcal{Z})$

1. High'ish entropy provides best performance

Effects of γ on corruption robustness



Validation

Test

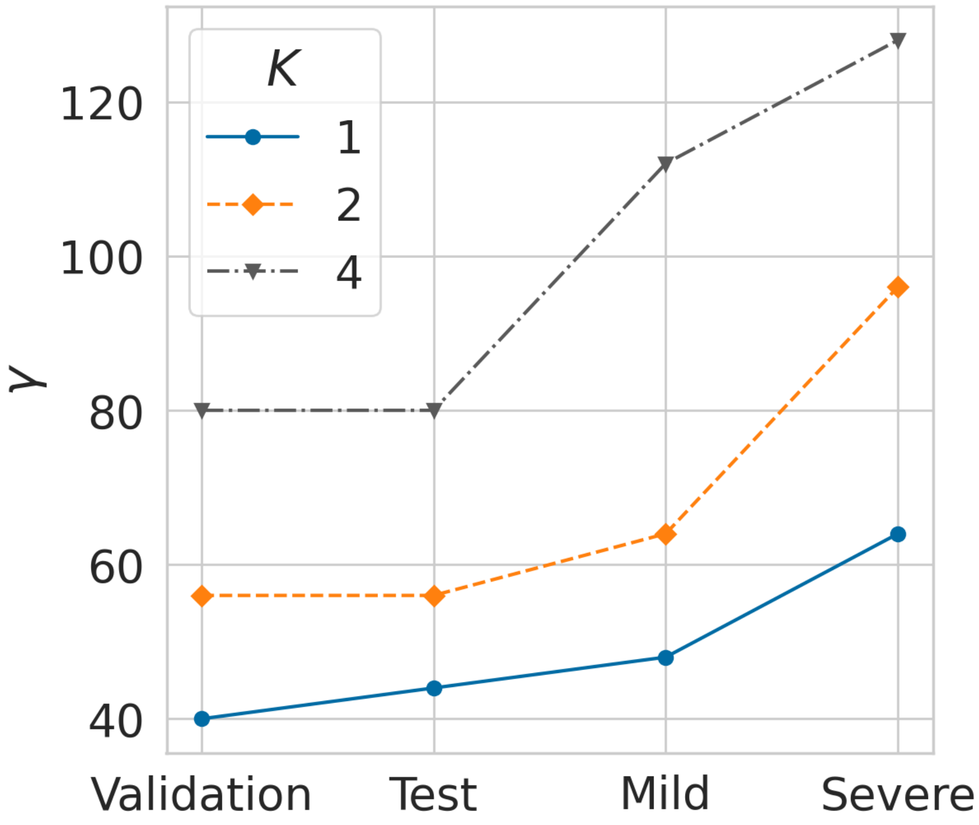
Mild corruption

Severe corruption

Network: VGG16
Train: CIFAR-100
Test: CIFAR-100-C
K: number of Gaussian components in $q_\phi(\mathcal{Z})$

1. High'ish entropy provides best performance
2. Optimising too much entropy worsens

More severe corruptions require higher optimal γ



Optimal γ

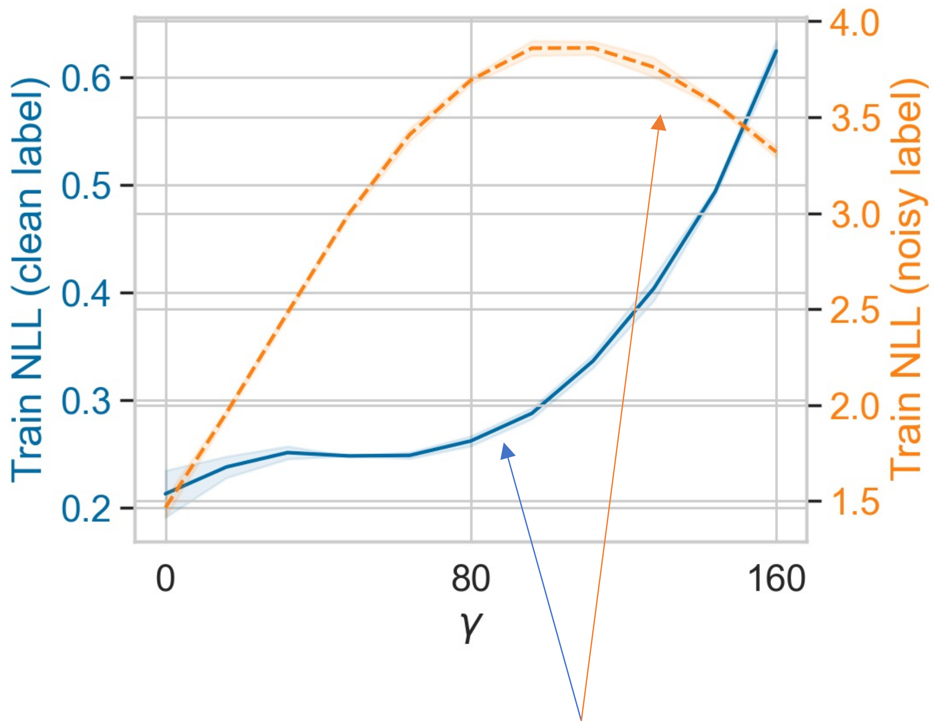
Robust learning under label noise

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

Wrongly labelled sample can’t be memorized if we add enough corruptions

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

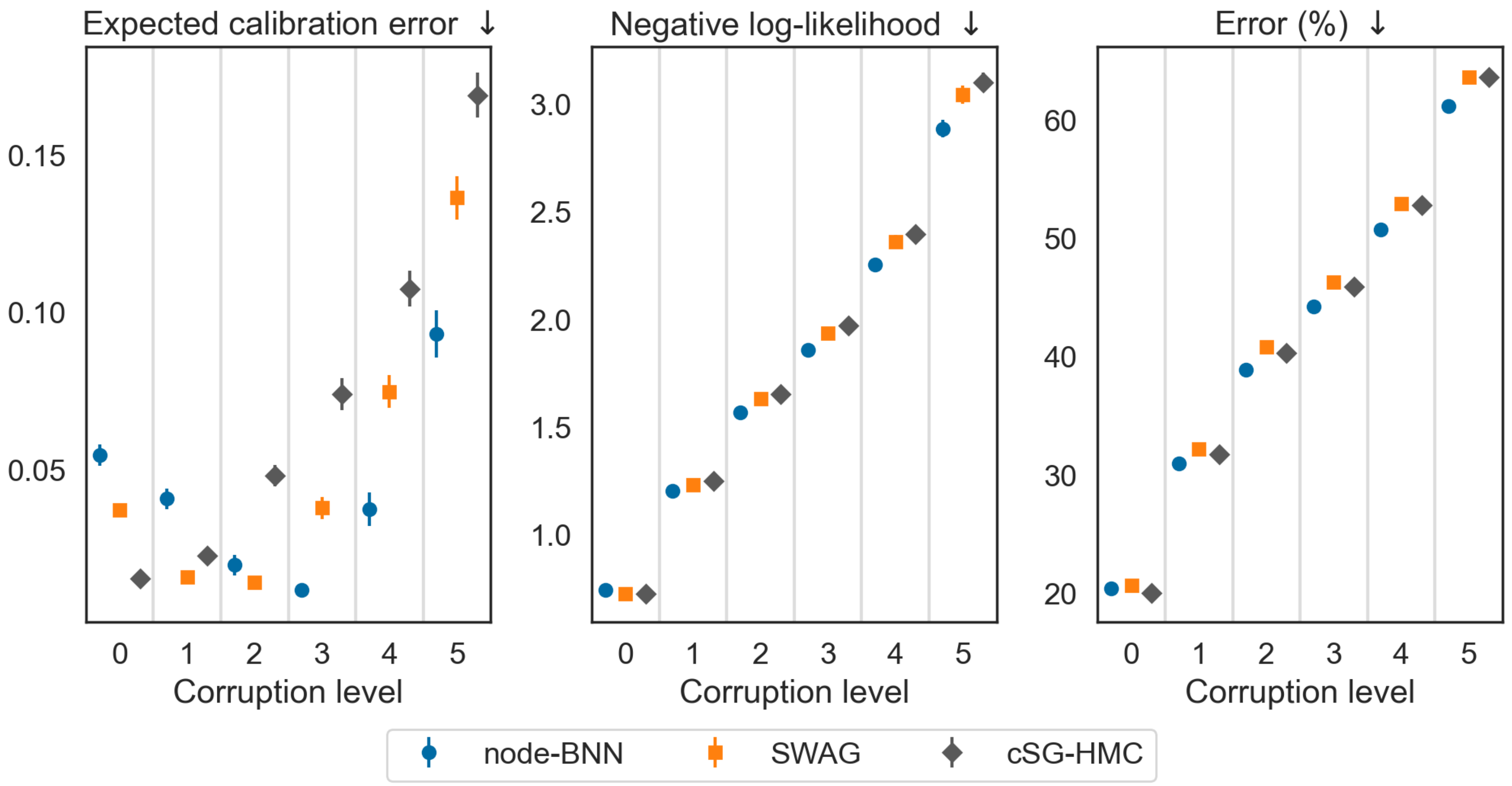
Robust learning under label noise



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

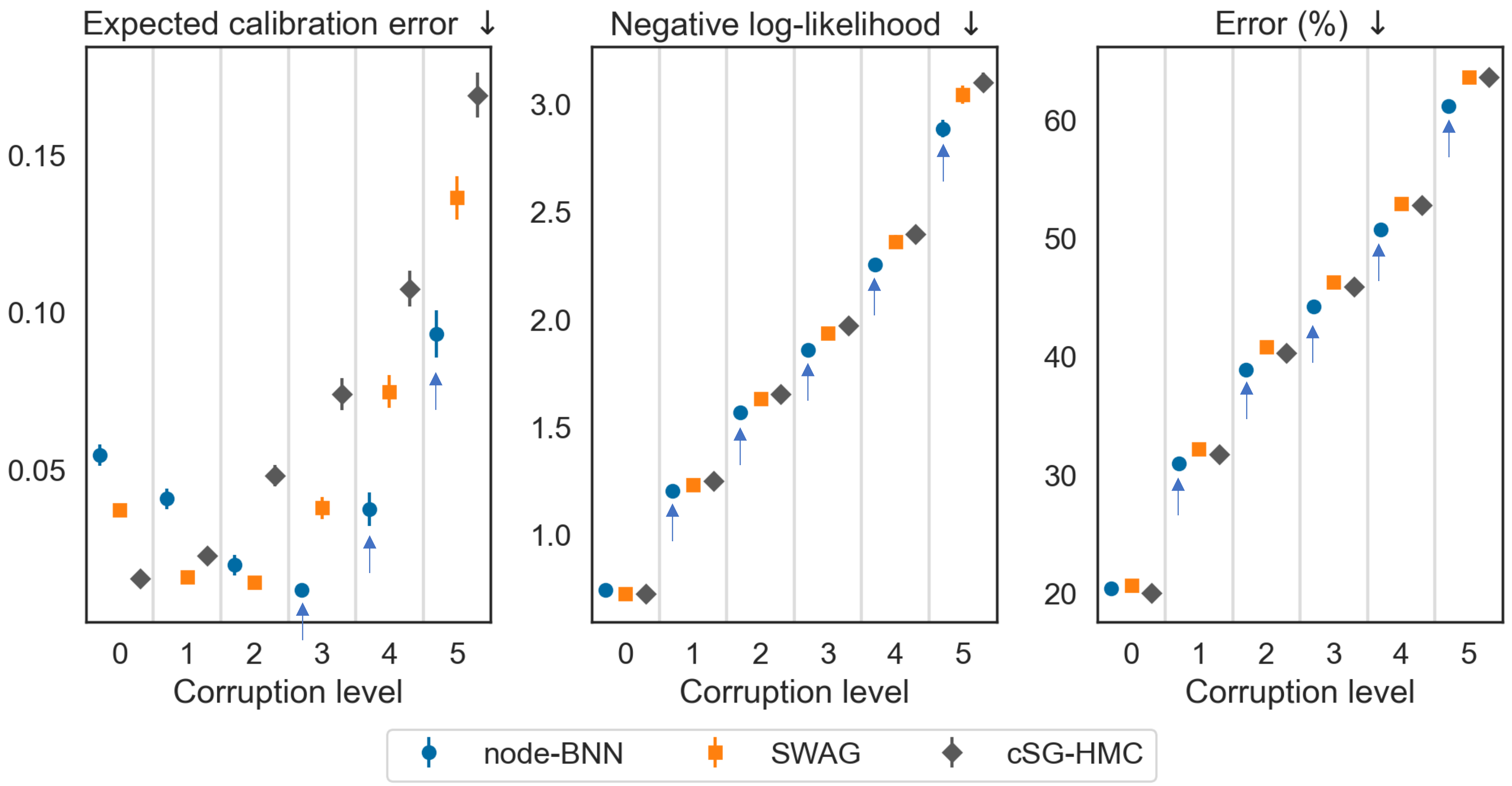
Benchmark comparison



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

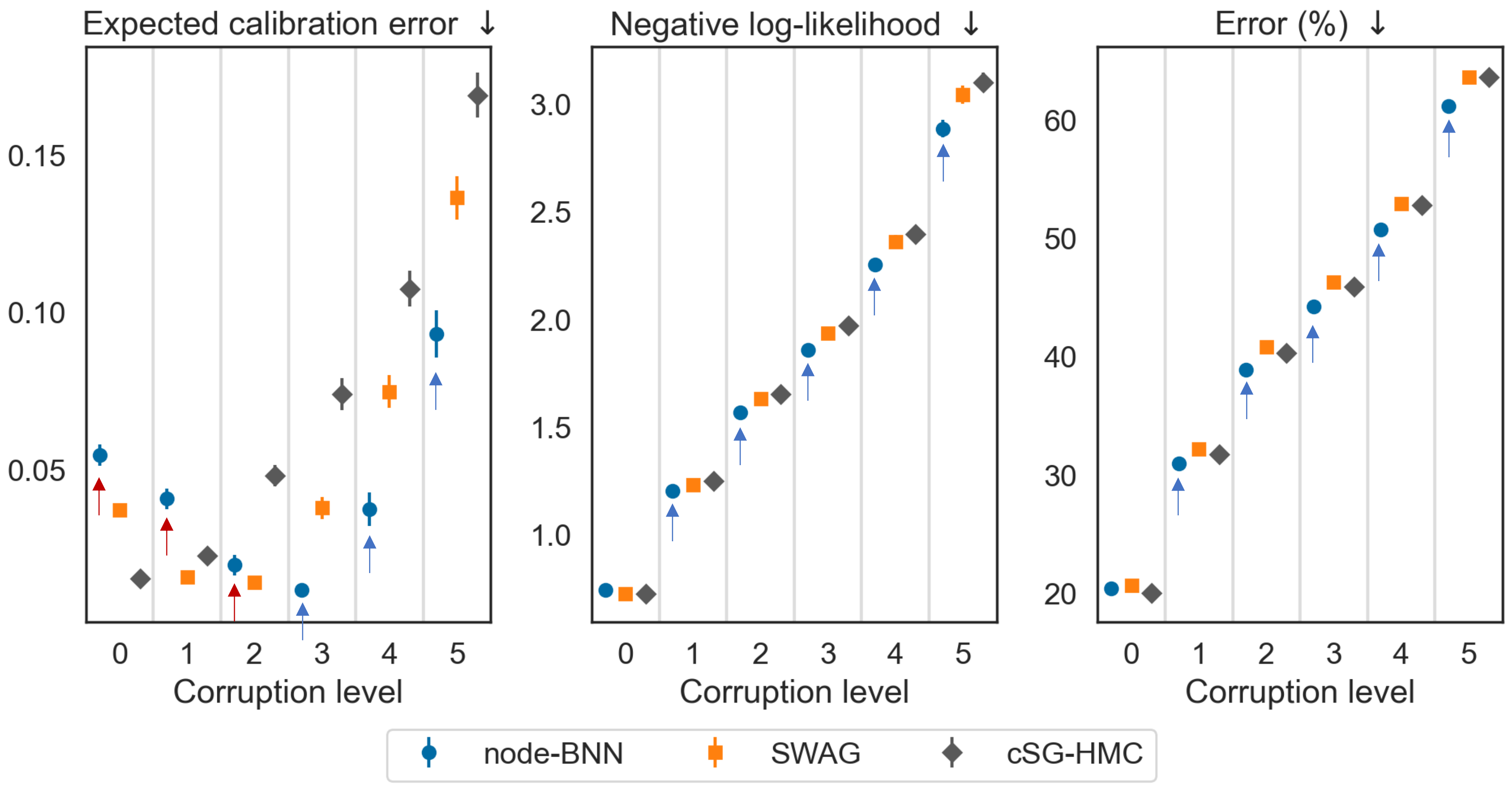
Benchmark comparison



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

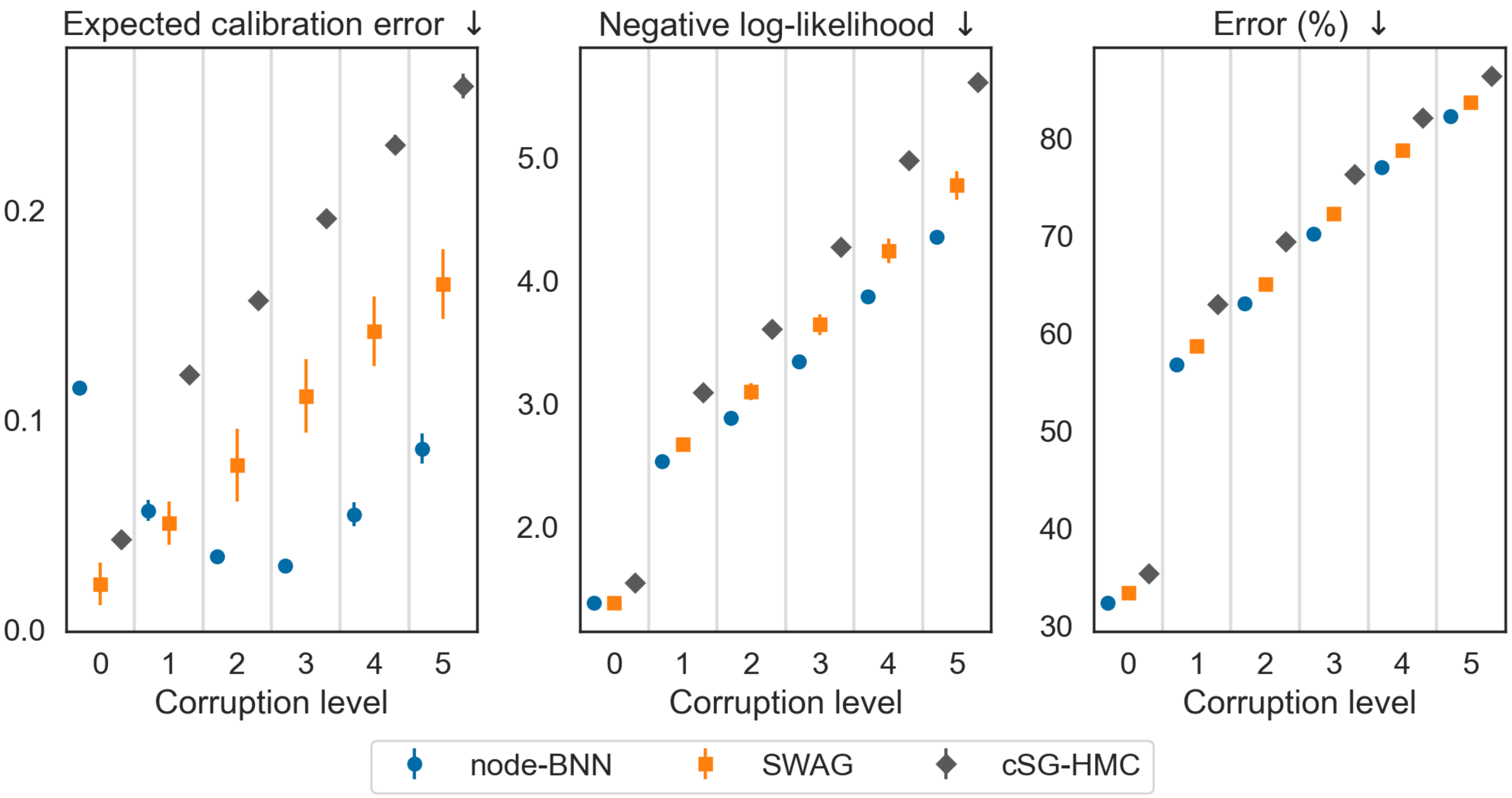
Benchmark comparison



ResNet18 / CIFAR-100

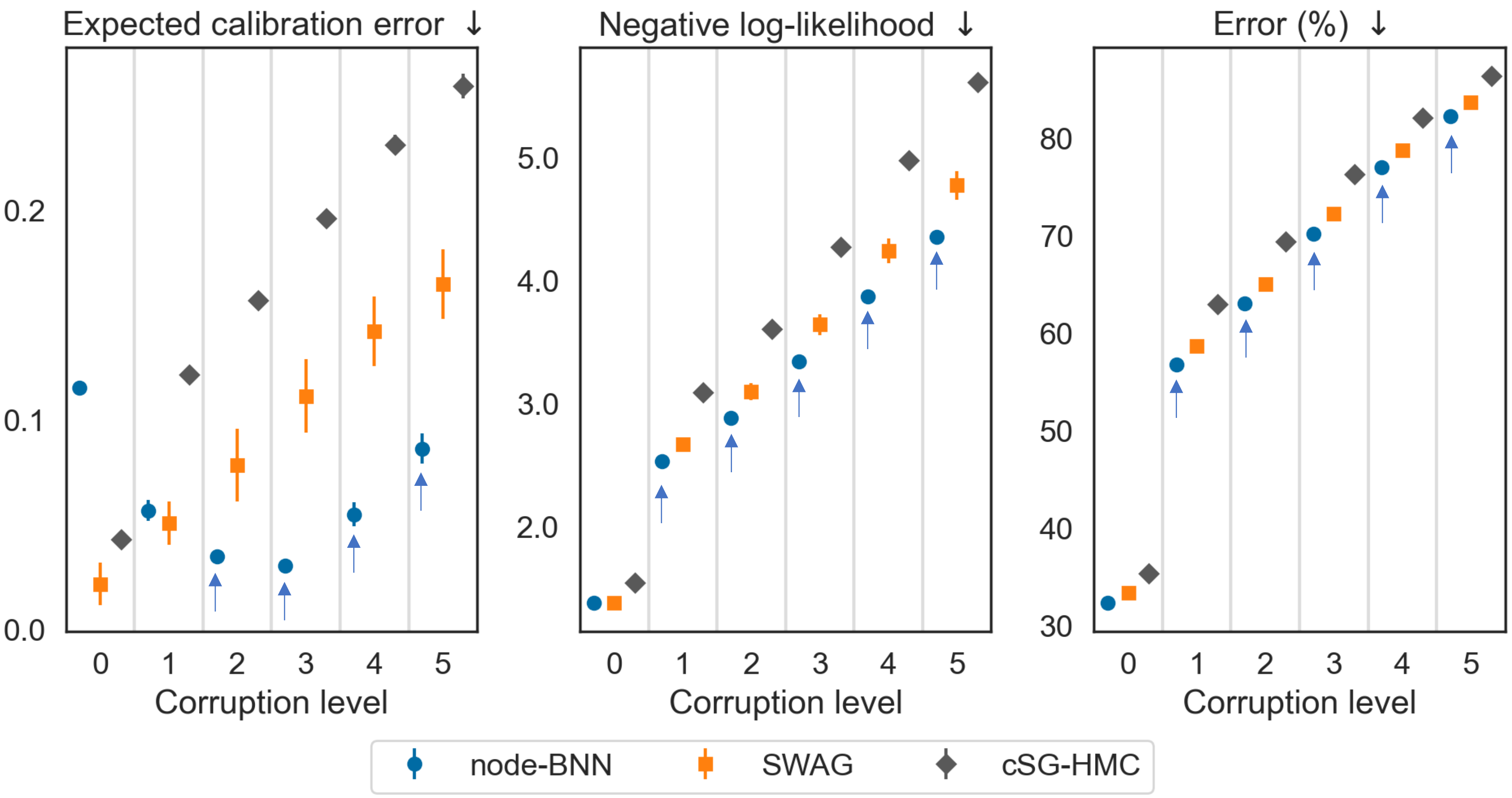
Maddox et al. (2019). A Simple Baseline for Bayesian Uncertainty in Deep Learning
Zhang et al. (2020). Cyclical Stochastic Gradient MCMC for Bayesian Deep Learning

Benchmark comparison



PreActResNet18 / TinyImageNet

Benchmark comparison



PreActResNet18 / TinyImageNet

Conclusion

1 Latent variables simulate a set of implicit corruptions, and implicitly training under these corruptions, node-based BNNs become robust against natural corruptions.

Code is available at https://github.com/AaltoPML/node-BNN-covariate-shift

Conclusion

1. Latent variables simulate a set of implicit corruptions, and implicitly training under these corruptions, node-based BNNs become robust against natural corruptions.

2. Maximizing entropy of the latent variables increases diversity of implicit corruptions, and thus node-BNN robustness.

Code is available at https://github.com/AaltoPML/node-BNN-covariate-shift

Conclusion

1. Latent variables simulate a set of implicit corruptions, and implicitly training under these corruptions, node-based BNNs become robust against natural corruptions.

2. Maximizing entropy of the latent variables increases diversity of implicit corruptions, and thus node-BNN robustness.

3. 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.

Code is available at https://github.com/AaltoPML/node-BNN-covariate-shift

Conclusion

1. Latent variables simulate a set of implicit corruptions, and implicitly training under these corruptions, node-based BNNs become robust against natural corruptions.

2. Maximizing entropy of the latent variables increases diversity of implicit corruptions, and thus node-BNN robustness.

3. 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.

Code is available at https://github.com/AaltoPML/node-BNN-covariate-shift