Impact of Noise on Calibration and Generalisation of Neural Networks

Martin Ferianc [*1] **Ondrej Bohdal** [*2] **Timothy Hospedales** [2 3] **Miguel Rodrigues** [1]

Abstract

Noise injection and data augmentation strategies have been effective for enhancing the generalisation and robustness of neural networks (NNs). Certain types of noise such as label smoothing and MixUp have also been shown to improve calibration. Since noise can be added in various stages of the NN's training, it motivates the question of when and where the noise is the most effective. We study a variety of noise types to determine how much they improve calibration and generalisation, and under what conditions. More specifically we evaluate various noise-injection strategies in both in-distribution (ID) and out-of-distribution (OOD) scenarios. The findings highlight that activation noise was the most transferable and effective in improving generalisation, while input augmentation noise was prominent in improving calibration on OOD but not necessarily ID data.

1. Introduction

Noise injection methods have emerged as a promising approach to enhance the generalisation of neural networks (NNs) (Srivastava et al., 2014; Neelakantan et al., 2017). Given the importance of noise for Bayesian NNs (BNNs) (Gal & Ghahramani, 2016; Blundell et al., 2015; Welling & Teh, 2011), we hypothesise that noise injections during training of standard NNs can also positively impact their calibration. Calibration refers to the alignment of prediction's accuracy to their confidence (Guo et al., 2017).

Examples of noise injection approaches include dropout (Srivastava et al., 2014; Gal & Ghahramani, 2016), label smoothing (Szegedy et al., 2016), MixUp (Zhang et al., 2018), Gaussian noise (Blundell et al., 2015), shrinking and perturbing NN weights (Ash & Adams, 2020), and gradient noise (Neelakantan et al., 2017). By introducing noise during the training, these methods encourage active exploration of the parameter space (He et al., 2019) and can be applied to various components of the network, including the input, targets, activations, gradients and the model itself. In this paper, we aim to provide a fair comparison of noise injection methods during training and investigate their impact on both calibration and generalisation of NNs in a computer vision classification setting. We ensure fairness of the comparison through dedicated hyperparameter optimization per noise type and we examine the transferability of found hyperparameters from one dataset or architecture to another. To robustly evaluate both generalisation and calibration we consider testing the methods on both test in-distribution (ID) and out-of-distribution (OOD) data.

The key takeaways from our work are: 1) Activation noise, especially dropout (Srivastava et al., 2014), improves generalisation and marginally also calibration across architectures and datasets. 2) Input augmentation, MixUp (Zhang et al., 2018), improves calibration and generalisation on OOD data but not necessarily ID data. 3) Model noise and gradient noise improve generalisation and calibration, but only to a smaller extent than input or activation noise.

2. Related Work

Standard NNs were shown to lack calibration (Guo et al., 2017), motivating the need for approaches focusing on training NNs such that their confidence matches their accuracy. Bayesian NNs (BNNs) (Blundell et al., 2015; Gal & Ghahramani, 2016; Welling & Teh, 2011) and NN ensembles (Lakshminarayanan et al., 2017) are popular approaches for obtaining well-calibrated models, but they are computationally expensive as they require random sampling and multiple forward passes during test time. Alternative methods have been proposed without increasing computational complexity, particularly during training. They include different loss functions (Kumar et al., 2018; Mukhoti et al., 2020; Bohdal et al., 2021) and temperature scaling (Guo et al., 2017). However, these approaches have their own limitations and may not be suitable for all scenarios. On the other hand, most noise injections are applicable to any NN architecture and any task.

[*]Equal contribution [1]Department of Electronic and Electrical Engineering, University College London [2]School of Informatics, University of Edinburgh [3]Samsung AI Center, Cambridge. Correspondence to: Martin Ferianc <martin.ferianc.19@ucl.ac.uk>, Ondrej Bohdal <ondrej.bohdal@ed.ac.uk>.

Published at the ICML 2023 Workshop on Spurious Correlations, Invariance, and Stability. Baltimore, Maryland, USA.

For **input noise** injection, commonly used are MixUp and Output Diversified Sampling (ODS) methods. MixUp (Zhang et al., 2018) linearly interpolates between two samples and their labels, while ODS (Tashiro et al., 2020) augments the input to diversify predictions and was used in the context of adversarial examples but not calibration. MixUp has been shown to improve calibration and generalisation (Zhang et al., 2022), but its transferability between datasets and architectures has not been explored. Additionally, we investigate naive Gaussian and uniform noise injection, which adds Gaussian or uniform noise to the input during training. In terms of **target noise** injection, frequently used is label smoothing (Pereyra et al., 2017) and MixUp (Zhang et al., 2018) label interpolation. Label smoothing replaces hard targets with soft targets and has already been shown to improve calibration, but not on OOD data (Müller et al., 2019). **Activation noise** injections include Dropout, Gaussian and uniform noise injections. Dropout (Srivastava et al., 2014) randomly sets activations to zero. Gaussian noise injection (Blundell et al., 2015; Camuto et al., 2020; He et al., 2019) adds Gaussian noise to the activations, while uniform noise injection adds uniform noise. In BNNs, these injections are applied both during training and evaluation, whereas in this work we only apply noise during training. Furthermore, **gradient noise** has been shown to improve generalisation through adding annealed Gaussian noise to the gradients during training (Neelakantan et al., 2017; Welling & Teh, 2011). However, it was not benchmarked on calibration, especially without ensembling weights at different training time-steps. Finally for **model noise** injection, recently Gaussian noise injection via shrinking and perturbing weights (Ash & Adams, 2020) at a given epoch frequency was shown to improve retraining generalisation, but calibration on ID or OOD data was not considered.

To the best of our knowledge, the noise injections have been studied 1) separately (Zhang et al., 2022; Müller et al., 2019), 2) orthogonally for generalisation and calibration on ID or OOD data, and 3) without a unified hyperparameter (HP) optimization protocol. This research aims to start the conversation into a comprehensive analysis of the noise injection methods and their relationship to generalisation and calibration, across datasets and NN architectures, providing valuable insights into their effectiveness and practicality.

3. Methodology

This study focuses on training a NN with noise perturbations to investigate their impact on NN's accuracy and calibration, identifying which perturbations are helpful and when. The noise types are divided between **input**, **target**, **gradient** and **model**, and their deployment during training is outlined in Algorithm 1 via blue lines. The probability of applying each noise type to a batch out of B batches is determined by the HP $p \in [0, 1]$, except model noise, which was applied with a selected frequency during the T training epochs. The noise types have associated HPs and tuning ranges.

Algorithm 1 Training of Neural Network with Noise

Require: Training dataset $D = \{(x^b, y^b)\}_{b=1}^{B}$, B batches, learning rate η, number of epochs T, weights θ, hidden states h_i^D, depth D, activations $f(\cdot)$, probability of applying noise to a batch p

Initialize θ randomly
for $t = 1$ to T **do**
  for $b = 1$ to B **do**
    Randomly select (x^b, y^b) from D
    Sample $e \sim U(0, 1)$ {If $e < p$}
    Input noise: Modify x^b
    Target noise: Modify y^b
    Activation noise: Modify $h_i^b = f(h_{i-1}^b)_{i=1}^D$
    Compute hidden states $h_i^b = f(x^b, \theta)$
    Compute predicted output $\hat{y}^i = g(h^i)$
    Compute loss $\mathcal{L}(\hat{y}^i, y^i)$ and gradients $\nabla_\theta \mathcal{L}$
    Gradient noise: Modify $\nabla_\theta L$
    Update weights: $\theta \leftarrow \theta - \eta \nabla_\theta L$
  end for
  if $t \mod \text{frequency} = 0$ **and** $t < 0.75T$ **then**
    Model noise: Modify θ
  end if
end for
return θ

Input noise: The input noise consisted of 2 naive variants and 2 variants which tapped into predictions or the targets to compute the noise. The two naive variants consisted of adding Gaussian or uniformly sampled noise $n \sim U(-\sigma, \sigma); n \sim \mathcal{N}(0, \sigma)$ added to inputs x with standard deviation $\sigma \in [1e^{-4}, 1e^{-1}]$. We considered ODS (Tashiro et al., 2020) with respect to $\epsilon \in [1e^{-4}, 1e^{-1}]$ and temperature $T \in [0.5, 5.0]$, and MixUp (Zhang et al., 2018) with $\alpha \in [0, 1]$ which also modified the targets accordingly. **Target noise:** In addition to MixUp we considered a static noise introduced to the labels y in the form of label smoothing (Müller et al., 2019) with the smoothing factor $l \in [0, 0.25]$. **Activation noise:** The hidden states prior to applying the activation function, $\{h_i^b\}_{i=1}^D$, where D is the depth of the net, could be disturbed by 3 types of activation noise: additive Gaussian or Uniform $n \sim U(-\sigma, \sigma); n \sim \mathcal{N}(0, \sigma)$ with $\sigma \in [1e^{-4}, 1e^{-1}]$ as a tunable HP or multiplicative Dropout (Srivastava et al., 2014) that incorporates a dropout rate $d \in [0, 1]$. The activation noise was used prior to an activation function for all linear or convolutional layers but not in the output layer. **Gradient noise:** The noise applied to the gradients $\nabla_\theta L$ followed (Neelakantan et al., 2017) with the step size $\eta \in [0, 1]$

and the annealing factor $\gamma \in [0, 1]$. **Model noise**: Lastly the model noise follows the idea of shrinking and perturbing the weights θ (Ash & Adams, 2020) with a shrink factor $\mu \in [0.0, 1.0]$ and standard deviation $\sigma \in [0.0, 1e^{-3}]$ with frequency of perturbing every frequency $\in [0, 80]$ epochs, except the last 25% of training epochs.

4. Experiments

Settings We first tune the learning rate and L2 regularisation of a no noise network which are reused when tuning the HPs of each noise injection method on three different combinations: ResNet-18 paired with CIFAR-10 or CIFAR-100 and a fully connected (FC) network paired with SVHN. The tuning is performed using $1/4$ of the training budget over the course of one day, using model-based Tree-structured Parzen Estimator method (Bergstra et al., 2011). With these settings we are able to evaluate about 40 configurations selected using Bayesian Optimisation. Our protocol allows us to optimise the performance of each noise injection method and provide fair comparison. Full experimental details are in Appendix A, including a summary of the identified HPs.

To assess the effectiveness of the noise injection methods, we measure their performance using three metrics: Error [↓, %], Expected Calibration Error (ECE) (Guo et al., 2017) [↓, %], and Negative Log-Likelihood (NLL) [↓] that we report in Appendix B. These metrics provide insights into the accuracy and its match with the confidence of the NNs' predictions. We evaluate the performance on both the ID test set and an augmented OOD set that includes an average over visual corruptions across 19 categories and 5 severities (Hendrycks & Dietterich, 2019). These corruptions include, for example, adding snow or fog to the image, changing the brightness or saturation of the image or blurring the image. We conduct experiments on a series of deployment scenarios where 1) the tuned HPs are directly used on the tuned dataset and architecture, 2) the architecture is changed but the HPs are kept, 3) the HPs come from a different source dataset. The results presented are averaged across 3 seeds and the best results are highlighted in **bold**.

4.1. Analysis

Tuned Hyperparameters In this scenario, we evaluate the performance of the noise injection methods when the HPs are tuned specifically for the dataset and architecture. The results for these experiments are in Tables 1 and 2, and they show that activation noises and input augmentation noises are prominent in improving the accuracy and calibration of the networks across the datasets. Dropout was the most effective for improving ID generalisation in CIFAR-10 and CIFAR-100, while MixUp was the most effective for SVHN. Uniform activation worked the best for improving ID calibration in CIFAR-10 and CIFAR-100, whereas ODS was the best in SVHN. Interestingly, some of the improvements carried to OOD data, for example, where the error on SVHN or CIFAR-100 was the lowest with MixUp or dropout. However, when considering calibration on OOD data, MixUp was dominant for CIFAR-10 and CIFAR-100. On average, dropout improved generalisation and MixUp improved calibration when considering both ID and OOD data. The naive Gaussian and uniform input noise perturbations did not bring significant improvements.

Architecture Transfer In this scenario, we assess the performance of the noise injection methods when the HPs are transferred to a different architecture while keeping the dataset constant. We conduct experiments using SVHN with ResNet-18 with HPs tuned on an FC network. Furthermore, we use HPs tuned for ResNet-18 for both CIFAR-10 and CIFAR-100 and we change the architecture to WideResNet-18. The results are presented in Tables 3 and 4. Considering the performance on ID data, we see that dropout reduced error across architectures and also improved calibration. Contrary to the improvements seen on SVHN when using FC, MixUp did not reduce the error when using ResNet-18 and it even recorded worse performance on OOD data than no noise at all. Switching focus to OOD data, model perturbation moderately improved calibration for CIFAR-100, while activations had a negative impact and led to worse calibration. Even though WideResNet-18 and ResNet-18 are relatively similar, transferring hyperparameters for example for MixUp in CIFAR-100, did not prove efficient as seen in calibration on OOD data which became worse than not using any noise at all. In summary, activation noises, most notably dropout, performed well on improving generalisation on both ID and OOD data and moderately on calibration on ID data. However, no method was able to consistently improve calibration on OOD data after the architecture was changed.

Dataset Transfer Under these settings, we investigate the transferability of hyperparameters by evaluating the noise injection methods on the same architectures but using different datasets. Specifically, we evaluate SVHN with ResNet-18 and HPs from CIFAR-100/ResNet-18, CIFAR-10 with ResNet-18 and CIFAR-100/ResNet-18 HPs, and CIFAR-100 with ResNet-18 but with CIFAR-10/ResNet-18 HPs. The results are shown in Tables 5 and 6. For all SVHN, CIFAR-10, CIFAR-100, the most significant error improvements across ID or OOD data were achieved using dropout and Gaussian noise. Interestingly, the activation Gaussian noise was able to improve calibration on both ID and OOD data on CIFAR-100, but not on the other datasets. MixUp has demonstrated varying results, for example on SVHN or CIFAR-10 the calibration on ID data was worse than not using any noise at all, while in CIFAR-100 there was a marginal improvement. Nevertheless on OOD data MixUp was able to improve calibration across all datasets.

Table 1. Error [↓, %] comparison on in-distribution (ID) and out-of-distribution (OOD) test sets and with tuned hyperparameters.

Noise Type	SVHN		CIFAR-10		CIFAR-100	
	ID	OOD	ID	OOD	ID	OOD
No Noise	17.07	20.77	10.76	33.92	37.86	59.53
Input Gaussian	17.37	20.96	11.05	34.09	37.94	59.48
Input Uniform	17.16	20.92	11.20	33.92	37.92	59.45
Input ODS	15.24	18.73	10.87	31.57	37.95	58.20
Input-Target MixUp	**13.86**	**17.46**	9.82	**28.03**	38.43	58.34
Target Label Smoothing	16.53	20.10	11.50	33.83	38.59	60.45
Activation Gaussian	17.15	20.82	8.99	31.44	34.87	58.12
Activation Uniform	16.97	20.70	8.86	31.83	34.62	57.77
Activation Dropout	14.58	17.97	**8.72**	30.92	**31.39**	**56.58**
Gradient	17.22	20.95	11.26	33.94	38.15	59.89
Model	16.37	20.09	10.77	33.63	38.11	59.80

Table 2. ECE [↓, %] comparison on in-distribution (ID) and out-of-distribution (OOD) test sets and with tuned hyperparameters.

Noise Type	SVHN		CIFAR-10		CIFAR-100	
	ID	OOD	ID	OOD	ID	OOD
No noise	15.47	18.31	5.43	12.72	16.85	15.28
Input Gaussian	15.80	18.50	5.34	12.78	16.91	15.22
Input Uniform	15.62	18.44	5.55	12.85	17.34	15.27
Input ODS	**4.56**	**5.65**	5.63	11.84	16.64	14.99
Input-Target MixUp	9.45	9.93	11.03	**11.23**	15.96	**13.81**
Target Label Smoothing	15.01	14.05	10.14	11.57	28.78	20.26
Activation Gaussian	15.54	18.34	4.20	14.93	11.58	23.32
Activation Uniform	15.43	18.27	**3.83**	15.84	**9.86**	20.62
Activation Dropout	6.49	7.77	5.29	13.40	12.17	25.82
Gradient	15.61	18.49	5.36	13.07	16.38	14.75
Model	13.74	16.29	5.30	12.73	15.40	14.41

Table 3. Error [↓, %] comparison on in-distribution (ID) and out-of-distribution (OOD) test sets and with changed architecture.

Noise Type	SVHN		CIFAR-10		CIFAR-100	
	ID	OOD	ID	OOD	ID	OOD
No Noise	5.23	9.41	14.86	38.19	37.38	61.39
Input Gaussian	5.22	9.44	14.68	38.01	37.37	61.27
Input Uniform	5.23	9.49	14.75	38.31	37.35	61.42
Input ODS	19.80	25.55	14.48	36.09	36.90	60.24
Input-Target MixUp	5.35	13.14	11.80	**31.54**	36.53	59.53
Target Label Smoothing	5.09	9.32	15.24	37.66	36.79	60.93
Activation Gaussian	5.26	9.45	11.96	35.70	43.23	65.98
Activation Uniform	5.25	9.47	12.88	37.90	36.94	61.02
Activation Dropout	**4.23**	**8.01**	**11.12**	33.15	**32.14**	**58.30**
Gradient	5.45	10.02	14.95	37.86	37.30	61.18
Model	4.38	8.50	14.72	38.14	36.49	61.09

Table 4. ECE [↓, %] comparison on in-distribution (ID) and out-of-distribution (OOD) test sets and with changed architecture.

Noise Type	SVHN		CIFAR-10		CIFAR-100	
	ID	OOD	ID	OOD	ID	OOD
No Noise	4.10	7.18	7.30	11.93	18.24	15.54
Input Gaussian	4.14	7.18	7.05	11.77	17.89	15.38
Input Uniform	4.10	7.19	7.08	12.00	18.27	15.25
Input ODS	6.65	9.08	7.11	11.43	17.77	15.39
Input-Target MixUp	10.24	9.93	12.62	11.69	18.92	18.07
Target Label Smoothing	21.24	20.20	12.05	11.54	28.97	19.50
Activation Gaussian	4.20	7.23	**5.60**	11.68	10.50	22.21
Activation Uniform	4.16	7.22	6.48	12.35	**8.58**	19.14
Activation Dropout	3.64	6.61	7.03	**11.05**	9.24	22.26
Gradient	4.21	7.48	7.26	11.36	17.65	15.29
Model	**3.40**	**6.12**	7.14	11.73	15.67	**14.55**

Table 5. Error [↓, %] comparison on in-distribution (ID) and out-of-distribution (OOD) test sets and with changed dataset.

Noise Type	SVHN		CIFAR-10		CIFAR-100	
	ID	OOD	ID	OOD	ID	OOD
No Noise	5.30	9.49	11.56	33.61	38.25	59.85
Input Gaussian	5.17	9.45	11.59	33.42	38.21	59.69
Input Uniform	5.12	9.43	11.18	33.99	38.16	59.85
Input ODS	5.24	9.29	11.45	32.42	37.51	56.85
Input-Target MixUp	5.14	11.65	10.89	**28.16**	37.99	58.69
Target Label Smoothing	5.22	9.39	11.35	34.23	37.31	58.67
Activation Gaussian	5.18	9.32	10.45	31.37	35.47	57.82
Activation Uniform	5.24	9.38	13.60	36.95	35.61	57.79
Activation Dropout	**4.11**	**7.86**	**9.40**	30.00	**32.12**	**56.63**
Gradient	6.11	10.88	11.82	33.38	38.15	60.95
Model	4.42	8.56	12.03	33.77	38.20	59.63

Table 6. ECE [↓, %] comparison on in-distribution (ID) and out-of-distribution (OOD) test sets and with changed dataset.

Noise Type	SVHN		CIFAR-10		CIFAR-100	
	ID	OOD	ID	OOD	ID	OOD
No Noise	4.18	7.19	**4.44**	14.88	17.09	14.88
Input Gaussian	4.10	7.18	4.58	15.11	16.71	15.16
Input Uniform	4.11	7.16	5.52	12.51	17.19	15.10
Input ODS	4.20	7.08	4.73	14.84	16.33	14.82
Input-Target MixUp	6.65	6.83	6.48	**8.87**	16.92	14.33
Target Label Smoothing	15.92	15.22	18.97	14.40	25.71	19.68
Activation Gaussian	4.19	7.19	5.53	18.53	9.28	**12.90**
Activation Uniform	4.16	7.15	4.63	12.63	**8.63**	12.98
Activation Dropout	3.60	6.52	5.64	19.29	9.92	20.81
Gradient	4.68	8.21	4.65	14.88	12.83	13.32
Model	**3.29**	**5.73**	4.81	15.14	16.80	15.01

Summary Different noise injection methods have varying degrees of effectiveness depending on the dataset and architecture. Nevertheless, especially in the tuned regime, certain settings of different noises improved both generalisation and calibration. Activation noise injections demonstrated promising results for error reduction across ID data, while input augmentations seemed to be the most effective for OOD data. Dropout was the most effective in improving error on ID or OOD data, and it proved to be transferable across architectures and datasets. MixUp was the best in improving the performance on OOD data in terms of calibration and accuracy, but not necessarily on the ID data. Interestingly, hidden in its mediocrity, model noise was able to marginally improve accuracy and calibration across majority of considered scenarios. Additional evaluation in terms of NLL, in Appendix B, has shown dropout was the most effective together with model perturbations.

5. Discussion and Conclusions

We investigated the effectiveness of various noise injection strategies for improving the calibration and generalisation of NNs. We found that activation noise was the most transferable and effective in improving generalisation, while input augmentation noise was prominent in improving calibration on OOD but not necessarily ID data. We note that our study was limited in several ways: 1) we did not perform full training when doing hyperparameter optimization, 2) we did not explore all possible combinations of noise types and injection strategies, 3) we focused only on computer vision

classification. Nevertheless, our findings suggest that noise injection can be a promising approach for improving the generalisation and calibration of NNs.

Acknowledgements

Martin Ferianc was sponsored through a scholarship from the Institute of Communications and Connected Systems at UCL.

References

Ash, J. and Adams, R. P. On warm-starting neural network training. In *NeurIPS*, 2020.

Bergstra, J., Bardenet, R., Bengio, Y., and Kégl, B. Algorithms for hyper-parameter optimization. In *NIPS*, 2011.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. Weight uncertainty in neural network. In *ICML*, 2015.

Bohdal, O., Yang, Y., and Hospedales, T. Meta-calibration: Learning of model calibration using differentiable expected calibration error. In *ICML UDL Workshop*, 2021.

Camuto, A., Willetts, M., Simsekli, U., Roberts, S. J., and Holmes, C. C. Explicit regularisation in gaussian noise injections. In *NeurIPS*, 2020.

Gal, Y. and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, 2016.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. In *ICML*, 2017.

He, Z., Rakin, A. S., and Fan, D. Parametric noise injection: Trainable randomness to improve deep neural network robustness against adversarial attack. In *CVPR*, 2019.

Hendrycks, D. and Dietterich, T. Benchmarking neural network robustness to common corruptions and perturbations. In *ICLR*, 2019.

Kumar, A., Sarawagi, S., and Jain, U. Trainable calibration measures for neural networks from kernel mean embeddings. In *ICML*, 2018.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In *NeurIPS*, 2017.

Loshchilov, I. and Hutter, F. Sgdr: Stochastic gradient descent with warm restarts. In *ICLR*, 2017.

Mukhoti, J., Kulharia, V., Sanyal, A., Golodetz, S., Torr, P., and Dokania, P. Calibrating deep neural networks using focal loss. In *NeurIPS*, 2020.

Müller, R., Kornblith, S., and Hinton, G. E. When does label smoothing help? In *NeurIPS*, 2019.

Neelakantan, A., Vilnis, L., Le, Q. V., Kaiser, L., Kurach, K., Sutskever, I., and Martens, J. Adding gradient noise improves learning for very deep networks. In *OpenReview*, 2017.

Pereyra, G., Tucker, G., Chorowski, J., Kaiser, Ł., and Hinton, G. Regularizing neural networks by penalizing confident output distributions. In *ICLR Workshop*, 2017.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 2014.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. In *CVPR*, 2016.

Tashiro, Y., Song, Y., and Ermon, S. Diversity can be transferred: Output diversification for white-and black-box attacks. In *NeurIPS*, 2020.

Welling, M. and Teh, Y. W. Bayesian learning via stochastic gradient langevin dynamics. In *ICML*, 2011.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. mixup: Beyond empirical risk minimization. In *ICLR*, 2018.

Zhang, L., Deng, Z., Kawaguchi, K., and Zou, J. When and how mixup improves calibration. In *ICML*, 2022.

Table 7. NLL [↓] comparison on in-distribution (ID) and out-of-distribution (OOD) test sets with tuned hyperparameters.

Noise Type	SVHN		CIFAR-10		CIFAR-100	
	ID	OOD	ID	OOD	ID	OOD
No noise	2.11	2.42	0.40	1.16	1.88	2.91
Input Gaussian	2.14	2.45	0.41	1.17	1.88	2.91
Input Uniform	2.11	2.43	0.41	1.16	1.89	2.91
Input ODS	0.59	0.69	0.40	1.08	1.87	2.83
Input-Target MixUp	**0.55**	**0.66**	0.41	**0.93**	1.77	**2.67**
Target Label Smoothing	0.73	0.83	0.48	1.14	2.14	3.07
Activation Gaussian	2.11	2.42	0.33	1.13	1.37	2.84
Activation Uniform	2.12	2.43	**0.32**	1.40	1.40	2.76
Activation Dropout	0.56	0.67	**0.32**	1.07	**1.25**	2.86
Gradient	2.13	2.45	0.41	1.17	1.88	2.92
Model	1.52	1.74	0.40	1.16	1.85	2.90

Table 8. NLL [↓] comparison on in-distribution (ID) and out-of-distribution (OOD) test sets with changed architecture.

Noise Type	SVHN		CIFAR-10		CIFAR-100	
	ID	OOD	ID	OOD	ID	OOD
No Noise	0.31	0.57	0.53	1.25	1.83	2.95
Input Gaussian	0.31	0.57	0.53	1.24	1.82	2.94
Input Uniform	0.31	0.57	0.53	1.25	1.83	2.95
Input ODS	0.84	1.10	0.52	1.19	1.80	2.89
Input-Target MixUp	0.27	0.50	0.48	**1.03**	1.72	2.77
Target Label Smoothing	0.40	0.52	0.60	1.24	2.05	3.06
Activation Gaussian	0.32	0.58	**0.41**	1.15	1.65	3.25
Activation Uniform	0.32	0.58	0.43	1.20	1.44	2.89
Activation Dropout	0.33	0.60	**0.41**	1.08	**1.23**	**2.84**
Gradient	0.33	0.59	0.54	1.23	1.81	2.93
Model	**0.23**	**0.42**	0.53	1.24	1.72	2.89

Table 9. NLL [↓] comparison on in-distribution (ID) and out-of-distribution (OOD) test sets with changed dataset.

Noise Type	SVHN		CIFAR-10		CIFAR-100	
	ID	OOD	ID	OOD	ID	OOD
No Noise	0.32	0.57	0.39	1.18	1.91	2.92
Input Gaussian	0.32	0.57	0.40	1.18	1.89	2.92
Input Uniform	0.31	0.57	0.41	1.16	1.90	2.93
Input ODS	0.32	0.56	0.39	1.15	1.83	2.75
Input-Target MixUp	0.23	0.43	0.39	**0.90**	1.77	2.70
Target Label Smoothing	0.34	0.46	0.57	1.19	2.02	2.96
Activation Gaussian	0.33	0.58	0.36	1.24	1.56	2.72
Activation Uniform	0.32	0.57	0.43	1.19	1.54	2.70
Activation Dropout	0.32	0.59	**0.35**	1.25	**1.23**	**2.67**
Gradient	0.36	0.67	0.40	1.17	1.79	2.91
Model	**0.20**	**0.37**	0.41	1.19	1.89	2.91

A. Experimental Settings Details

We used stochastic gradient descent with momentum 0.9 to train all the networks. The learning rate $\eta = [1e-4, 1e-1]$ together with L2 regularisation $\lambda = [1e-7, 1e-1]$ were initially tuned and then reused for each noise injection method. We used cosine annealing learning rate schedule without restarts (Loshchilov & Hutter, 2017) for all experiments. For each dataset, we only used normalization without any further data augmentations. We used gradient norm clipping of 5.0 to stabilise the training. The batch size was set to 256 for all experiments. 10% of the training data was used as the validation set to select the best model. The tuning was performed with 1 seed and the winning hyperparameters were retrained 3 times with different seeds. The final results are reported as the average of the 3 runs. We used cross-entropy loss for all the experiments. In all cases we trained the networks for 200 epochs.

We used a fully connected network with 4 hidden layers of 150 units followed by ReLU activations and ResNet-18 with [32, 64, 128, 256] channels in 4 stages with [2, 2, 2, 2] blocks with strides [1, 2, 2, 2]. For WideResNet-18 we used the same channel, stride and block configuration as for ResNet-18, but with the Bottleneck block, expansion factor 4, base width 32, base width multiplier 2 and a single group. Both residual architectures use batch normalisation and ReLU activations. We used the default PyTorch weight initialization for all layers. We used 10 bins to measure ECE and a small eta $1e^{-8}$ which was added to the output softmax probabilities to avoid NaNs. The found hyperparameters for each dataset architecture pair are in Table 10. The descriptions of the hyperparameters as well as the search ranges are provided in Section 3.

B. Negative Log-Likelihood

We provide additional Tables 7, 8 and 9 that report the Negative Log-Likelihood for the different experiments that we have conducted. Comparing the NLL results to the previously collected results on the error and calibration error, there are subtle differences. In Table 7 the NLL for SVHN is the lowest for MixUp and not ODS for ID and OOD data as suggested by ECE. Furthermore, we can see in Table 8 that when we change the architecture, the model perturbation achieved the lowest NLL on both ID and OOD data for SVHN, while the dropout activation was the best in terms of the error. Dropout retained the lowest NLL in CIFAR-100 and CIFAR-10 on ID data. Looking at Table 9 that reports results with the changed dataset, surprisingly model perturbation was the most dominant in NLL on SVHN, in contrast again to dropout as suggested by the error. Even though for CIFAR-10 the calibration on ID data was the lowest without using any noise, dropout was able to improve the NLL over no noise settings. In summary and contrast to the previous results, in NLL improvements dropout was perceived as the most effective together with model perturbations.

Table 10. Found hyperparameters on SVHN, CIFAR-10 and CIFAR-100 datasets.

NOISE TYPE	**HYPERPARAMETER**	**SVHN**	**CIFAR-10**	**CIFAR-100**
NO NOISE	LEARNING RATE	0.042	0.0082	0.060
	L2	1.3E-07	0.029	0.0031
INPUT GAUSSIAN	σ	0.0029	0.00011	0.00061
	p	0.56	0.83	0.78
INPUT UNIFORM	σ	0.00086	0.0038	0.0013
	p	0.28	0.63	0.61
INPUT ODS	T	1.37	0.85	2.26
	η	0.047	0.0056	0.00080
	p	0.96	0.066	0.67
INPUT-TARGET MIXUP	α	0.99	0.55	0.56
	p	0.91	0.72	0.69
TARGET LABEL SMOOTHING	l	0.24	0.11	0.18
	p	0.99	0.65	0.96
ACTIVATION GAUSSIAN	σ	0.0035	0.0064	0.099
	p	1.0	0.54	0.47
ACTIVATION UNIFORM	σ	0.0014	0.014	0.058
	p	0.02	0.70	0.63
ACTIVATION DROPOUT	d	0.16	0.31	0.26
	p	0.78	0.55	0.75
GRADIENT	η	0.020	0.26	0.32
	γ	0.63	0.99	0.99
	p	0.0083	0.19	0.021
MODEL	μ	0.32	0.75	0.57
	σ	9.7E-05	0.00087	0.00016
	FREQUENCY	52	28	12