TRAINING, ARCHITECTURE, AND PRIOR FOR DETERMINISTIC UNCERTAINTY METHODS

Anonymous authors
Paper under double-blind review

ABSTRACT

Accurate and efficient uncertainty estimation is crucial to build reliable Machine Learning (ML) models capable to provide calibrated uncertainty estimates, generalize and detect Out-Of-Distribution (OOD) datasets. To this end, Deterministic Uncertainty Methods (DUMs) is a promising model family capable to perform uncertainty estimation in a single forward pass. This work investigates important design choices in DUMs: **(1)** we show that *training* schemes decoupling the core architecture and the uncertainty head schemes can significantly improve uncertainty performances. **(2)** we demonstrate that the core *architecture* expressiveness is crucial for uncertainty performance and that additional architecture constraints to avoid feature collapse can deteriorate the trade-off between OOD generalization and detection. **(3)** Contrary to other Bayesian models, we show that the *prior* defined by DUMs do not have a strong effect on the final performances.

1 INTRODUCTION

Safety is critical to the adoption of deep learning in domains such as autonomous driving, medical diagnosis, or financial trading systems. A solution for this problem is to create reliable models capable to estimate the uncertainty of its own predictions. Different uncertainty types are divided in *aleatoric* uncertainty quantified by the inherited noise in the data, thus irreducible; *epistemic* uncertainty quantified by the modeling choice or lack of data, thus reducible; *predictive* uncertainty, a combination of aleatoric and epistemic (Gal, 2016). In practice, high quality uncertainty estimates must be calibrated and able to detect Out-Of-Distribution (OOD) data like anomalies while preserving good Out-Of-Distribution (OOD) generalization performances like on dataset shifts.

Recently, a family of methods for uncertainty estimation named Deterministic Uncertainty Methods (DUMs) have emerged (Postels et al., 2022). Contrary to uncertainty methods such as Ensembles (Lakshminarayanan et al., 2017), MC Dropout (Gal & Ghahramani, 2016) or other Bayesian neural networks on weights (Blundell et al., 2015), which require multiple forward passes to make predictions, DUMs only require a single forward pass, thus making them significantly more computationally efficient. Generally, DUMs are composed of three components with high potential impact on their performances: the *training* procedure which is supposed to optimize the model toward high predictive and uncertainty performances, the core *architecture* which is supposed to define informative embeddings used to make predictions, and the *prior* which is supposed to define the default uncertain predictions. In this work, we investigate the role of these three components on the quality of DUMs uncertainty estimates by evaluating calibration performances, OOD detection, and OOD generalization. Our main contributions are:

- **Training**: We show that *decoupling the learning rates* of the core architecture and uncertainty heads of DUMs, *jointly training* the core architecture and the uncertainty head of DUMs, and *pretraining* with *more data* and *higher data quality* improve uncertainty performances.
- **Architecture**: We demonstrate that the expressiveness of the core architecture defined by the *architecture type*, *architecture size*, and *dimension of the latent space* is crucial for *both* predictive and uncertainty performances. Further, we show that applying additional regularization constraints to avoid *feature collapse* does not find better trade-off between OOD detection and generalization, even sometimes degrading performances.

- **Prior**: In contrast to Bayesian neural networks on weights where the choice of prior is critical (Wenzel et al., 2020; Fortuin et al., 2022; Noci et al., 2021; Kapoor et al., 2022), we empirically show that the choice of prior defined in the training loss or in the uncertainty head of DUMs has a relatively small effect on the final uncertainty performances.

2 Deterministic Uncertainty Methods

We consider a classification task where the goal is to predict the label $y^{(i)} \in \{1, \dots, C\}$ based on the input feature $\boldsymbol{x}^{(i)} \in \mathbb{R}^D$. In this case, a DUM generally performs predictions in two main steps: **(1)** A core *architecture* f_ϕ maps the input features $\boldsymbol{x}^{(i)} \in \mathbb{R}^D$ to a latent representation $\boldsymbol{z}^{(i)} \in \mathbb{R}^H$, i.e. $f_\phi(\boldsymbol{x}^{(i)}) = \boldsymbol{z}^{(i)}$. In practice, important design choices are the latent dimension H and the architecture f_ϕ which should be adapted to the task (see section 4). Further, multiple works proposed to apply additional bi-Lipschitz or reconstruction constraints to enrich the informativeness of the latent representation $\boldsymbol{z}^{(i)}$ (see section 4). **(2)** An *uncertainty head* g_ψ maps the latent representation $\boldsymbol{z}^{(i)}$ to a predicted label $\hat{y}^{(i)}$ and an associated (aleatoric, epistemic, or predictive) uncertainty estimate $u^{(i)}$, i.e. $g_\psi(\boldsymbol{z}^{(i)}) = (\hat{y}^{(i)}, u^{(i)})$. In practice, important design choices are the type of uncertainty head which are generally instantiated with a Gaussian Process (GP) (Liu et al., 2020; van Amersfoort et al., 2021; 2020; Biloš et al., 2019; Charpentier et al., 2022b), a density estimator (Charpentier et al., 2020; 2022a;b; Stadler et al., 2021; Biloš et al., 2019; Mukhoti et al., 2021; Postels et al., 2020a; Winkens et al., 2020; Wu & Goodman, 2020), or an evidential model (Charpentier et al., 2020; 2022a;b; Stadler et al., 2021; Biloš et al., 2019; Malinin & Gales, 2018), and the choice of the prior used by the uncertainty head (see section 5). Beyond core architecture and uncertainty head, another important choice is the training procedure which can either couple or decouple the parameters of the core architecture and uncertainty head (see section 3).

In this work we focus on two recent DUMs which cover different types of uncertainty heads: Natural Posterior Network (NatPN) Charpentier et al. (2022a) which learns an evidential distribution based on density estimation on the latent space, and Deterministic Uncertainty Estimator (DUE) (van Amersfoort et al., 2021) which learns a deep Gaussian Process by parametrizing learnable inducing points in the latent space (see appendix A.1 for further method details). While NatPN is capable to differentiate aleatoric, epistemic, and predictive uncertainty, DUE only outputs the predictive uncertainty. For all the experiments, we use the same default setup: we first *pretrain* the encoder with the cross-entropy loss until convergence, then load the pretrained encoder and jointly *train* the encoder and uncertainty head (see appendices B to D for further compenent details). We report the predictive performance via accuracy, and the uncertainty performances with Brier Score and OOD detection results after averaging over 5 seeds (see appendix A.3 for further metric details). We perform our experiments on MNIST (LeCun et al., 1998), CIFAR10 and CIFAR100 (Krizhevsky, 2009), and Camelyon (Koh et al., 2021). OOD results reported in tables 1 to 3 averages the uncertainty estimation from five OOD datasets: SVHN, STL10, CelebA, Camelyon and SVHN OODom (Netzer et al., 2011; Coates et al., 2011; Liu et al., 2015). Our code is available online[1].

Related work. Previous works survey OOD detection methods (Yang et al., 2021), OOD generazilation methods (Shen et al., 2021), or a wide range of uncertainty estimation methods (Gawlikowski et al., 2021; Psaros et al., 2023; Ulmer, 2021; Abdar et al., 2021) by presenting key methods and challenges. These surveys do not focus on deterministic methods and do not make empirical analysis. Other works propose great empirical studies to compare uncertainty estimation methods under shifts (Ovadia et al., 2019), or analyze the role of the prior in Bayesian neural networks on weights (Wenzel et al., 2020; Fortuin et al., 2022; Noci et al., 2021; Kapoor et al., 2022). These works do not focus on DUMs. Closer to our work, Postels et al. (2022) compares methods in the DUMs family and demonstrate calibration limitations. In contrast, we evaluate the role of *components* in DUMs and show that carefully specifying *training*, *architecture*, or *prior* can improve uncertainty metrics like calibration and OOD detection but also ID and OOD predictive performances.

[1] https://anonymous.4open.science/r/dum-components-EB75

3 TRAINING FOR DUMS

In this section, we study the importance of the training procedure in the performance of DUMs. To this end, we look at *decoupling the learning rates* of the core encoder architecture and the uncertainty head, different *training schemes*, and different *pretraining schemes*.

Decoupling learning rates. We decouple the learning rates of the core architecture and the uncertainty head. We show the validation results for CIFAR100 as ID and SVHN as OOD with the core architecture ResNet18 in fig. 1. *Observation:* We observe that, when using different learning rates for the core architecture and the uncertainty head, NatPN improves Brier Score and OOD epistemic results and DUE significantly improves both predictive and uncertainty results. Hence, this shows that decoupling learning rates can improve results of DUMs, thus suggesting that the core architecture and the uncertainty head have training dynamics which requires different considerations.

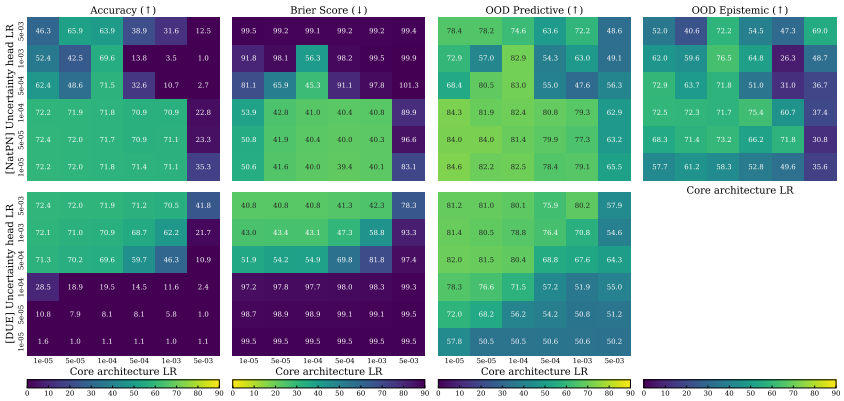


Figure 1: Results of DUMS on CIFAR100 with ResNet18 when **decoupling learning rates** of the core architecture and the uncertainty head. Decoupling learning rates improve DUMs performance.

Training schemes. We compare two settings: the *joint* training in which we jointly train the weights of the core architecture and uncertainty head, and the *sequential* training in which we only the train the uncertainty head by keeping the weights of the pretrained core architecture fixed. For each of the setting, we apply two additional techniques to stabilize the training: adding a *batch normalization* to the last layer of the encoder to enforce latent representations to locate in a normalized region (Ioffe & Szegedy, 2015; Charpentier et al., 2022a), and *resetting the last layer* to retrain its weights to improve robustness to spurious correlation (Kirichenko et al., 2022). We show the results for CIFAR100 as ID and SVHN as OOD with the ResNet18 as core architecture in table 1 and additional results in the appendix table 6. *Observation:* We observe that, compared to its sequential counterpart, joint training consistently improves DUMs performance for most metrics, thus suggesting that joint training should be preferred in practice for DUMs. Furthermore, while the GP-based method DUE does not benefit from stabilization techniques, we observe that they can significantly increase performance of the density-based method NatPN. This behavior is intuitively explained by the practical difficulty to accurately fit densities in high dimensional latent space. This can be significantly improved by using more powerful density estimator (see table 5 in appendix).

Pretraining schemes. We compare multiple training schemes which differ in terms of *amount* and *quality* of data used for pretraining. Hence, we pretrain the core encoder architecture with 10% of CIFAR100, 100% of CIFAR100, and ImageNet. We show the results for CIFAR100 as ID and SVHN as OOD with ResNet18 as core architecture in table 2 and additional results in the appendix table 7. *Observation:* We observe that, while too few data for pretraining does not improve final performance of DUMs, the overall performance significantly increase when the encoder is pretrained with high quantity and high quality of data. Similarly to Kirichenko et al. (2020), this suggests that the embedding quality is important to improve uncertainty quantification. Here, we show additionally that embeddings pretrained with many high quality data are crucial to facilitate the prediction of the uncertainty head.

Table 1: Results of DUMs on CIFAR100 with ResNet18 under different **training schemes** using *joint/sequential* training with no additional layer, an additional batch norm layer, or resetting the last layer. Gray cells indicate the best between *joint/sequential* while bold numbers indicate the best overall. We observe that joint training works best and stabilization techniques can improve performances.

Method	Train Schema	Accuracy (↑)	Brier Score (↓)	OOD Pred. (↑)	OOD Epis. (↑)
NatPN	joint	71.12 ± 0.18	41.06 ± 0.18	75.17 ± 1.60	63.94 ± 2.80
	joint + bn	71.60 ± 0.14	41.11 ± 0.12	74.22 ± 0.94	66.17 ± 2.55
	joint + reset	71.61 ± 0.18	**40.76 ± 0.18**	**75.35 ± 0.71**	**69.02 ± 1.49**
	sequential	**72.00 ± 0.19**	42.20 ± 0.09	75.09 ± 0.86	53.49 ± 2.56
	sequential + bn	71.98 ± 0.18	42.39 ± 0.11	75.01 ± 0.86	52.34 ± 2.81
	sequential + reset	71.79 ± 0.17	40.95 ± 0.14	74.63 ± 0.85	61.90 ± 2.14
DUE	joint	**72.33 ± 0.11**	**40.80 ± 0.11**	74.74 ± 0.89	-
	joint + bn	72.30 ± 0.09	40.85 ± 0.12	74.63 ± 0.95	-
	joint + reset	71.94 ± 0.12	41.43 ± 0.12	74.89 ± 0.76	-
	sequential	72.07 ± 0.10	41.66 ± 0.10	74.82 ± 0.90	-
	sequential + bn	72.04 ± 0.13	41.73 ± 0.11	74.88 ± 0.95	-
	sequential + reset	71.56 ± 0.14	42.30 ± 0.11	**75.08 ± 1.01**	-

Table 2: Results of DUMs with ResNet50 under different **pretraining schemes** using no pretraining, pretraining on 10% of CIFAR100, 100% of CIFAR100, or ImageNet. Bold numbers indicate best results among all settings. We observe that high quantity and high quality of data can improve performances.

Method	Pretrain Schema	Accuracy (↑)	Brier Score (↓)	OOD Pred. (↑)	OOD Epis. (↑)
NatPN	None	78.45 ± 1.94	30.16 ± 2.57	79.85 ± 3.30	85.81 ± 2.05
	C100 (10%)	67.25 ± 0.71	44.69 ± 0.94	68.10 ± 1.90	78.50 ± 3.27
	C100 (100%)	76.31 ± 0.45	34.32 ± 0.51	78.95 ± 3.19	76.91 ± 2.64
	ImageNet	**84.22 ± 0.12**	**23.67 ± 0.27**	**84.95 ± 1.48**	**89.08 ± 0.70**
DUE	None	72.41 ± 0.24	47.35 ± 0.25	80.04 ± 1.28	-
	C100 (10%)	63.86 ± 0.58	50.94 ± 0.53	72.44 ± 1.32	-
	C100 (100%)	76.38 ± 0.35	36.89 ± 0.50	81.71 ± 1.87	-
	ImageNet	**82.42 ± 0.14**	**28.09 ± 0.19**	**90.24 ± 0.51**	-

4 Architecture for DUMs

In this section, we study the impact of the architecture component in DUMs. To this end, we look at different *latent dimensions*, different *architectural types and size*, and applying different *regularization constraints* to avoid *feature collapse* (van Amersfoort et al., 2021).

Latent dimension. We vary the dimension of the output space of the core architecture. We show the results for each pair of ID dataset and its distribution shifted OOD dataset (MNIST/CMNIST, CIFAR/CIFAR-C, CamelyonID/CamelyonOOD) with the core architecture ResNet18 for MNIST/CIFAR, and WideResNet-28-10 for Camelyon in fig. 2 and additional uncertainty estimation results in the appendix fig. 10. *Observation:* We observe that increasing the latent dimensions leads to improvement for DUMs on ID and OOD datasets with particularly significant improvement for NatPN (see fig. 2). This suggests that higher latent dimensions are more expressive by encoding more information. However, we observe that a too high latent dimension can degrade OOD detection performance by causing numerical instabilities in the training (see fig. 10), suggesting a trade-off between OOD generalization and OOD detection.

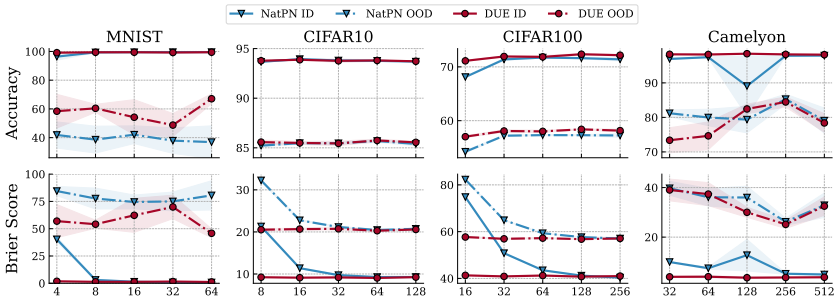


Figure 2: Results of DUMs when varying the **latent dimension size**. We observe that increasing the latent dimension consistently leads to similar or better predictive performance.

Architecture type and size. We compare the influence of the type and size of the core architecture on the performance of DUMs. We consider residual, convolutional, and transformer architectures like ResNet18, ResNet50, EfficientNetV2, and Swin (He et al., 2016; Tan & Le, 2021; Liu et al., 2021). We show the results for DUMs trained on CIFAR100 as ID with the different core architectures in table 3 and additional results at appendix table 8. *Observation:* We observe that models with more parameters achieve better results. In particular, ResNet50 achieves significantly better results than ResNet18. Further, more recent core architectures like EfficientNetV2 and Swin are better calibrated and more expressive leading to a better overall performance. This can be explained by the fact that they are more expressive and provide more informative embeddings for the uncertainty

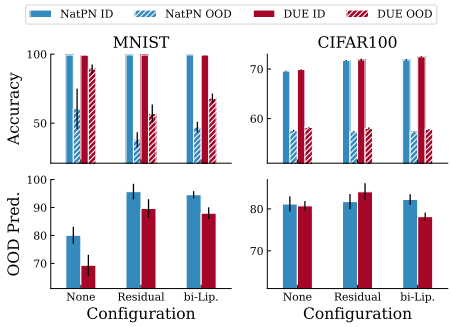


Figure 3: Results OOD generalization and detection of DUMs with none, residual and bi-lipschitz **architecture constraints** on MNIST/CMNIST and CIFAR/CIFAR-C. Bi-lipschitz can improve OOD detection by mitigating feature collapse (see fig. 8a) at the expense of degrading OOD generalization.

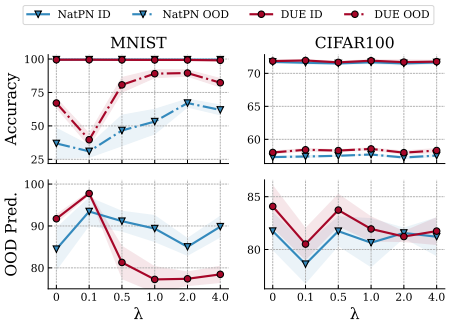


Figure 4: Results OOD generalization and detection of DUMs with reconstruction **architecture constraints** on MNIST/CMNIST and CIFAR/CIFAR-C. Increasing the reconstruction strength λ improves the OOD generalization on simple MNIST/CMNIST dataset but fails for complex datasets. Reconstruction fails to improve OOD detection since it does not avoid feature collapse (see fig. 8b).

head to operate on. This aligns with Minderer et al. (2021) which states that the architecture type is important for the calibration properties.

Table 3: Results of DUMs for different **architecture types.** including residual, convolutional, and transformer architectures. Bold numbers indicate best results among all settings. Larger and more recent architectures are better calibrated with similar or better uncertainty estimation.

Method	Architecture	#Parameters	Accuracy (↑)	Brier Score (↓)	OOD Pred. (↑)	OOD Epis. (↑)
NatPN	ResNet18	11.6M	80.31 ± 0.09	33.69 ± 0.15	87.49 ± 1.90	81.79 ± 1.38
	ResNet50	25.5M	84.22 ± 0.12	23.67 ± 0.27	84.95 ± 1.48	89.08 ± 0.70
	EffNet_V2_S	21.4M	**88.43 ± 0.10**	**17.08 ± 0.10**	**87.79 ± 0.77**	89.47 ± 0.59
	Swin_T	28.2M	87.99 ± 0.09	18.48 ± 0.06	85.91 ± 1.17	**90.23 ± 0.81**
DUE	ResNet18	11.6M	78.85 ± 0.19	36.57 ± 0.15	88.04 ± 0.67	-
	ResNet50	25.5M	82.42 ± 0.14	28.09 ± 0.19	**90.24 ± 0.51**	-
	EffNet_V2_S	21.4M	86.92 ± 0.08	**21.07 ± 0.06**	89.43 ± 0.67	-
	Swin_T	28.2M	**86.93 ± 0.05**	23.23 ± 0.05	89.90 ± 0.36	-

Regularization constraints. *Feature collapse* is a phenomenon where a model may discard important parts of the input information during its training phase, which may degrade OOD detection performance (van Amersfoort et al., 2021). Two techniques to avoid feature collapses are *bi-Lipschitz* constraints via combining residual connections and lipschitz constraints (Liu et al., 2020), and *reconstruction* constraints via adding an additional reconstruction term in the loss (Postels et al., 2020b). We show the results for DUMs trained on the datasets MNIST and CIFAR100 with ResNet18 in figs. 3 and 4 and additional results for other datasets (Toy dataset, CIFAR10, Camelyon) at appendix C. *Observation:* We observe that the reconstruction technique is not capable to avoid feature collapse. Indeed, we show that, even with reconstruction constraints, some (non-discriminative) features can completely collapse (see figs. 8b and 9b for toy examples). Hence, while this method can lead to small OOD improvements on simple tasks (see e.g. MNIST in fig. 4), this benefit does not generalize to more complex tasks (see e.g. CIFAR100 in fig. 4). In contrast, we observe that bilipschitz constraints indeed mitigate the collapse of features (see figs. 8a and 9a for toy examples), leading to similar or higher OOD detection performance (see fig. 3). The mitigation of feature collapse can be mostly assigned to the residual connection constraints. However, bilipschitz constraints can improve OOD detection results on simple tasks (e.g. MNIST, CIFAR10), it degrades OOD generalization performance (see fig. 3) and does not significantly improve OOD detection on more complex tasks (see e.g. Camelyon, CIFAR100 in fig. 6). Intuitively, maintaining features which are not discriminative to the task might introduce spurious correlations, thus degrading performances. E.g. enforces the architecture to encode the color feature in the latent space decreases the performance of the OOD CMNIST datasets after training on the ID MNIST dataset.

5 PRIOR FOR DUMS

In this section, we study the effect that the prior component has in DUMs. More specifically, we investigate the relationships between aleatoric uncertainty and the prior specified for DUMs. In particular, this is motivated by Kapoor et al. (2022) which shows that using priors that forces model to be confident on the training data points can improve its performance by explicitly accounting for aleatoric uncertainty. To this end, we look at *entropy regularization* defining a training prior in the loss, *prior evidence* and *kernel function* defining a functional prior in the uncertainty head.

Prior. We compare different prior specifications including *entropy regularization* defining a training prior in the loss, *prior evidence* and *kernel function* defining a functional prior in the uncertainty head. Entropy regularization is the entropy term $H(Q)$ in the Bayesian loss used to train NatPN which encourages a (uniform) prior distributions with high entropy (Charpentier et al., 2022a). We control the strength of the regularization factor λ. Further, NatPN also explicitly defines a prior via the parameters χ^{prior} and n^{prior}. While χ^{prior} defines the default categorical prediction via a uniform categorical distribution, the evidence parameter n^{prior} defines the prior number of pseudo-observations and can be varied. Finally, we vary the prior of DUE by using different kernel functions in the learned GP including Matern kernel, RQ kernel, and RBF (Rasmussen & Williams, 2006). *Observation:* Contrary to other Bayesian neural networks (Kapoor et al., 2022), we observe that predictive and uncertainty performances of DUMs are not very sensitive to the prior specification (see figs. 5 and 14 and table 4), thus suggesting a higher robustness to prior mispecification. Nonetheless, a too strong entropy regularization toward an uniform prior degrades more performance of DUMs trained on dataset with low label noise than on high label noise. This suggests that a too high discrepancy between the model prior and the dataset aleatoric uncertainties can impact performance.

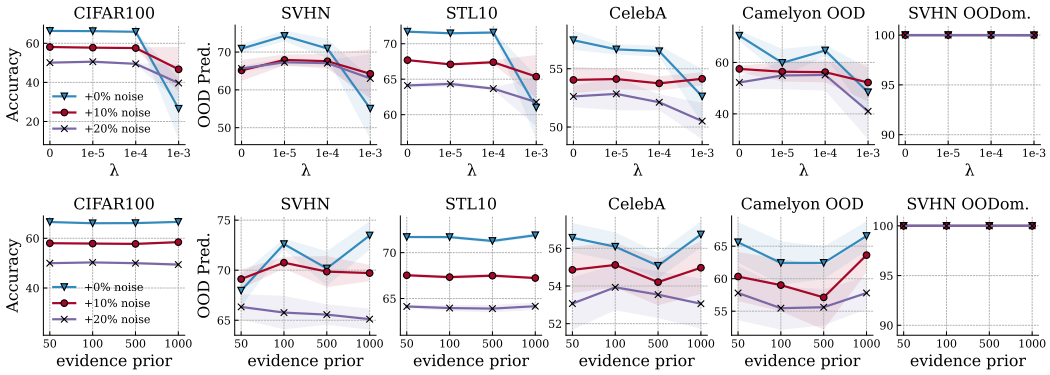


Figure 5: Results of enforcing different **prior** in NatPN on CIFAR100 by changing the (top) *entropy regularization* λ and the (bottom) *evidence prior* n^{prior}. Different priors do not lead consistent results improvements.

Table 4: Results of enforcing different **prior** in DUE on CIFAR100 and Camelyon by changing the kernel function. Different priors lead to similar performance.

	CIFAR100			Camelyon		
Kernel	**Accuracy (↑)**	**Brier Score (↓)**	**OOD Pred. (↑)**	**Accuracy (↑)**	**Brier Score (↓)**	**OOD Pred. (↑)**
Matern52	71.80 ± 0.18	41.37 ± 0.24	75.90 ± 1.18	79.81 ± 2.72	32.46 ± 3.22	58.86 ± 6.20
Matern32	71.80 ± 0.21	41.62 ± 0.22	**76.15 ± 1.18**	80.23 ± 2.71	32.77 ± 3.20	58.50 ± 5.83
Matern12	71.70 ± 0.18	43.10 ± 0.22	75.70 ± 1.19	79.30 ± 2.96	32.67 ± 3.28	**59.13 ± 6.39**
RQ	71.83 ± 0.19	**41.16 ± 0.25**	75.93 ± 1.21	80.31 ± 2.55	32.22 ± 3.14	58.69 ± 6.09
RBF	**71.85 ± 0.19**	41.17 ± 0.24	76.14 ± 1.19	**80.45 ± 2.49**	**32.13 ± 3.11**	58.86 ± 5.91

6 CONCLUSION

We investigate important design choice in DUMs. We show that *training* of DUMs can be improved by decoupling the the optimization of the core architecture and the uncertainty head. We show that expressive core *architecture* can improve DUMs performances. In contrast, additional constraints to avoid feature collapse do not consistently lead to better performance, potentially degrading the OOD generalization and detection trade-off. Finally, we show that the choice of *prior* for DUMs does not lead to important performance improvements.

REFERENCES

Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U. Rajendra Acharya, Vladimir Makarenkov, and Saeid Nahavandi. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 2021.

Martín Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv*, 2019.

Marin Biloš, Bertrand Charpentier, and Stephan Günnemann. Uncertainty on asynchronous time event prediction. In *NeurIPS*, 2019.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *ICML*, 2015.

Bertrand Charpentier, Daniel Zügner, and Stephan Günnemann. Posterior network: Uncertainty estimation without OOD samples via density-based pseudo-counts. In *NeurIPS*, 2020.

Bertrand Charpentier, Oliver Borchert, Daniel Zügner, Simon Geisler, and Stephan Günnemann. Natural posterior network: Deep bayesian predictive uncertainty for exponential family distributions. In *ICLR*, 2022a.

Bertrand Charpentier, Ransalu Senanayake, Mykel Kochenderfer, and Stephan Günnemann. Disentangling epistemic and aleatoric uncertainty in reinforcement learning, 2022b.

Tarin Clanuwat, Mikel Bober-Irizar, Asanobu Kitamoto, Alex Lamb, Kazuaki Yamamoto, and David Ha. Deep learning for classical japanese literature. *arXiv*, 2018.

Adam Coates, Andrew Y. Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In Geoffrey J. Gordon, David B. Dunson, and Miroslav Dudík (eds.), *AISTATS*, 2011.

Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. In *NeurIPS*, 2019.

Vincent Fortuin, Adrià Garriga-Alonso, Sebastian W. Ober, Florian Wenzel, Gunnar Rätsch, Richard E. Turner, Mark van der Wilk, and Laurence Aitchison. Bayesian neural network priors revisited. In *ICLR*, 2022.

Yarin Gal. *Uncertainty in deep learning*. University of Cambridge, 2016.

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

Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna M. Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, Muhammad Shahzad, Wen Yang, Richard Bamler, and Xiao Xiang Zhu. A survey of uncertainty in deep neural networks. *arXiv*, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *SVPR*, 2016.

Dan Hendrycks and Thomas G. Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *ICLR*, 2019.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

Sanyam Kapoor, Wesley J. Maddox, Pavel Izmailov, and Andrew Gordon Wilson. On uncertainty, tempering, and data augmentation in bayesian classification. *arXiv*, 2022.

Polina Kirichenko, Pavel Izmailov, and Andrew G Wilson. Why normalizing flows fail to detect out-of-distribution data. In *NeurIPS*, 2020.

Polina Kirichenko, Pavel Izmailov, and Andrew Gordon Wilson. Last layer re-training is sufficient for robustness to spurious correlations. *arXiv*, 2022.

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, Tony Lee, Etienne David, Ian Stavness, Wei Guo, Berton Earnshaw, Imran S. Haque, Sara M. Beery, Jure Leskovec, Anshul Kundaje, Emma Pierson, Sergey Levine, Chelsea Finn, and Percy Liang. WILDS: A benchmark of in-the-wild distribution shifts. In *ICML*, 2021.

Alex Krizhevsky. Learning multiple layers of features from tiny images. *arXiv*, 2009.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *NeurIPS*, 2017.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 1998.

Jeremiah Z. Liu, Zi Lin, Shreyas Padhy, Dustin Tran, Tania Bedrax-Weiss, and Balaji Lakshminarayanan. Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. In *NeurIPS*, 2020.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *ICCV*, December 2015.

Ilya Loshchilov and Frank Hutter. SGDR: stochastic gradient descent with warm restarts. In *ICLR*, 2017.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

Andrey Malinin and Mark Gales. Predictive uncertainty estimation via prior networks. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2018.

Matthias Minderer, Josip Djolonga, Rob Romijnders, Frances Hubis, Xiaohua Zhai, Neil Houlsby, Dustin Tran, and Mario Lucic. Revisiting the calibration of modern neural networks. In *NeurIPS*, 2021.

Jishnu Mukhoti, Andreas Kirsch, Joost van Amersfoort, Philip H. S. Torr, and Yarin Gal. Deterministic neural networks with appropriate inductive biases capture epistemic and aleatoric uncertainty. *arXiv*, 2021.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

Lorenzo Noci, Kevin Roth, Gregor Bachmann, Sebastian Nowozin, and Thomas Hofmann. Disentangling the roles of curation, data-augmentation and the prior in the cold posterior effect. In *NeurIPS*, 2021.

Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, D. Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *NeurIPS*, 2019.

Janis Postels, Hermann Blum, Cesar Cadena, Roland Siegwart, Luc Van Gool, and Federico Tombari. Quantifying aleatoric and epistemic uncertainty using density estimation in latent space. *arXiv*, 2020a.

Janis Postels, Hermann Blum, Cesar Cadena, Roland Siegwart, Luc Van Gool, and Federico Tombari. Quantifying aleatoric and epistemic uncertainty using density estimation in latent space. *arXiv*, 2020b.

Janis Postels, Mattia Segù, Tao Sun, Luca Daniel Sieber, Luc Van Gool, Fisher Yu, and Federico Tombari. On the practicality of deterministic epistemic uncertainty. In *ICML*, 2022.

Apostolos F. Psaros, Xuhui Meng, Zongren Zou, Ling Guo, and George Em Karniadakis. Uncertainty quantification in scientific machine learning: Methods, metrics, and comparisons. *Journal of Computational Physics*, 2023.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *ICML*, 2015.

Zheyan Shen, Jiashuo Liu, Yue He, Xingxuan Zhang, Renzhe Xu, Han Yu, and Peng Cui. Towards out-of-distribution generalization: A survey, 2021.

Maximilian Stadler, Bertrand Charpentier, Simon Geisler, Daniel Zügner, and Stephan Günnemann. Graph posterior network: Bayesian predictive uncertainty for node classification. In *NeurIPS*, 2021.

Vincent Stimper, Bernhard Schölkopf, and José Miguel Hernández-Lobato. Resampling Base Distributions of Normalizing Flows. In *AISTATS*, 2022.

Mingxing Tan and Quoc V. Le. Efficientnetv2: Smaller models and faster training. In *ICML*, 2021.

Dennis Ulmer. A survey on evidential deep learning for single-pass uncertainty estimation, 2021.

Joost van Amersfoort, Lewis Smith, Yee Whye Teh, and Yarin Gal. Uncertainty estimation using a single deep deterministic neural network. In *ICML*, 2020.

Joost van Amersfoort, Lewis Smith, Andrew Jesson, Oscar Key, and Yarin Gal. On feature collapse and deep kernel learning for single forward pass uncertainty. *arXiv preprint arXiv:2102.11409*, 2021.

Florian Wenzel, Kevin Roth, Bastiaan S. Veeling, Jakub Swiatkowski, Linh Tran, Stephan Mandt, Jasper Snoek, Tim Salimans, Rodolphe Jenatton, and Sebastian Nowozin. How good is the bayes posterior in deep neural networks really? In *ICML*, 2020.

Jim Winkens, Rudy Bunel, Abhijit Guha Roy, Robert Stanforth, Vivek Natarajan, Joseph R. Ledsam, Patricia MacWilliams, Pushmeet Kohli, Alan Karthikesalingam, Simon Kohl, Taylan Cemgil, S. M. Ali Eslami, and Olaf Ronneberger. Contrastive training for improved out-of-distribution detection, 2020.

Mike Wu and Noah D. Goodman. A simple framework for uncertainty in contrastive learning. *arXiv*, 2020.

Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. Generalized out-of-distribution detection: A survey, 2021.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016.

A APPENDIX

A.1 DETERMINISTIC UNCERTAINTY METHODS

We provide a formal definition for the DUMs used in our experiments.

NatPN (Charpentier et al., 2022a) is a deep Bayesian uncertainty model which leverages the exponential family distribution to enable a input-dependent Bayesian update. Given an input $\boldsymbol{x}^{(i)}$, NatPN performs an input-dependent Bayesian update which can be expressed in a closed-form as:

$$\mathbb{Q}(\boldsymbol{\theta}|\boldsymbol{\chi}^{post,(i)}, n^{post,(i)}) = \exp(n^{post,(i)}\boldsymbol{\theta}^T\boldsymbol{\chi}^{post,(i)} - n^{post,(i)}A(\boldsymbol{\theta}))$$
$$\text{where} \qquad \boldsymbol{\chi}^{post,(i)} = \frac{n^{prior}\boldsymbol{\chi}^{prior} + n^{(i)}\boldsymbol{\chi}^{(i)}}{n^{prior} + n^{(i)}}, \quad n^{post,(i)} = n^{prior} + n^{(i)}$$

where $\boldsymbol{\chi}^{prior}, n^{prior}$ are fixed prior parameters. $\boldsymbol{\chi}^{(i)}, n^{(i)}$ are predicted parameters dependent on the input $\boldsymbol{x}^{(i)}$ with a core architecture, and $\boldsymbol{\chi}^{post,(i)}, n^{post,(i)}$ are the input-dependent posterior parameters. The parameters $\boldsymbol{\chi}^{(i)}, n^{(i)}$ are predicted with the core architecture $f_{\boldsymbol{\psi}}$ and a density estimator $\mathbb{P}(.|\boldsymbol{\omega})$. We train NatPN with a Bayesian loss Charpentier et al. (2020) which can be rewritten in closed-form as:

$$\begin{aligned}\mathcal{L}^{(i)} &= -\mathbb{E}_{\boldsymbol{\theta}^{(i)}\sim\mathbb{Q}^{post,(i)}}[log\mathbb{P}(y^{(i)}|\boldsymbol{\theta}^{(i)}] - \mathbb{H}[\mathbb{Q}^{post,(i)}] \\ &\propto \mathbb{E}[\boldsymbol{\theta}^{(i)}]^T u(y^{(i)}) - \mathbb{E}[A(\boldsymbol{\theta}^{(i)})] - \mathbb{H}[\mathbb{Q}^{post,(i)}]\end{aligned} \tag{1}$$

We refer to (Charpentier et al., 2022a) for a more detailed description of the method.

DUE. The deep kernel learning method DUE (van Amersfoort et al., 2021) follows three steps: **(1)** it predicts one latent representation of the input $\boldsymbol{x}^{(i)}$ i.e. $\boldsymbol{z}^{(i)} = f_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})$, **(2)** a Gaussian Process defined from a fixed set of K learnable inducing points $\{\boldsymbol{\phi}_k\}_{k=1}^K$ and a predefined positive definite kernel $\kappa(\cdot,\cdot)$ predicts the mean $\mu(\boldsymbol{x}^{(i)})$ and the variance $\sigma(\boldsymbol{x}^{(i)})$ of a Gaussian distribution, and **(3)** we apply softmax to the mean output $\mu(\boldsymbol{x}^{(i)})$ for the classification prediction, i.e. $\boldsymbol{p}^{(i)} = \text{softmax}(\mu(\boldsymbol{x}^{(i)})$. We train the neural network parameters $\boldsymbol{\theta}$ and the inducing points $\{\boldsymbol{\phi}_k\}_{k=1}^K$ jointly with a variational ELBO loss. We refer to (van Amersfoort et al., 2021) for a more detailed description of the method.

A.2 DATASET DETAILS

We split all the training datasets into train, validation and test sets. For all the datasets, the test set is fixed while the training/validation sets are split in 80/20% respectively. The random split of training/validation sets change depending on the seeds to ensure more diversity.

MNIST (LeCun et al., 1998). Image classification dataset. Similarly as in Arjovsky et al. (2019), we create the CMNIST dataset for domain generalization experiments by expanding the input's size to 3 x 28 x 28 and zeroing one of the three channels. For OOD detection we use the test set of MNIST as ID dataset and compare to: KMNIST (Clanuwat et al., 2018), CIFAR10, CMNIST, and KMNIST OODom, where we scale the input by 255. The batch size used is 512.

CIFAR (Krizhevsky, 2009). Image classification dataset. For domain generalization we use the corrupted version CIFAR-C (Hendrycks & Dietterich, 2019) and report the average metric of 15 corruptions for the level of corruption severity of 1. For OOD detection we use the test set of CIFAR10 as ID dataset and compare to: SVHN (Netzer et al., 2011), STL10 (Coates et al., 2011), CelebA (Liu et al., 2015), Camelyon (Test OOD), and SVHN OODom. Since the Camelyon (Test OOD) dataset is large (85k), we extract only 10k subset of images as the OOD datasets. The batch size used is 128.

Camelyon (Koh et al., 2021). Image classification dataset. For domain generalization the dataset already provide the distribution shifted validation and test splits. For OOD detection we use the ID validation set of Camelyon as ID dataset (the ID test set is not available) and compare to: SVHN, STL10, CelebA, Camelyon (Test OOD), and SVHN OODom. The batch size used is 32.

Each OOD dataset is rescaled to the same size as the ID dataset and normalized with zero mean and unit variance based on the statistics of ID dataset (for the Camelyon dataset we don't apply any normalization as in Koh et al. (2021)).

A.3 METRIC DETAILS

We measure the results using:

Accuracy. The standard accuracy $\frac{1}{N}\sum_i \mathbf{1}[y^{*,(i)} = y^{(i)}]$ is used, where $y^{*,(i)}$ is the true label and $y^{(i)}$ is the predicted label.

Calibration. The Brier score $\frac{1}{C}\sum_i^N ||p^{(i)} - y^{*,(i)}||$ is used, where $p^{(i)}$ is the predicted softmax probability and $y^{*,(i)}$ is the true label. Lower calibration indicates a better calibrated model.

OOD Generalization. We apply accuracy and calibration to the distribution shifted OOD dataset and compare the results with the ID dataset to estimate the model's ability for generalization.

OOD Detection. We treat this task as a binary classification, where we assign class 1 to ID data and class 0 to OOD data using the aleatoric, epistemic, and predictive uncertainty estimates as scores for OOD data. This allows to compute the final scores using the area under the receiver operating characteristic curve (AUC-ROC) to measure the model's ability to detect OOD data.

A.4 MODEL DETAILS

Core architecture. We use the same feature extractor for both the DUMs architecture. The list of core architectures used across the experiments are: *ResNet18 / ResNet50 / EfficientNet / Swin* (He et al., 2016; Tan & Le, 2021; Liu et al., 2021) from the torchvision repository [2] and *Wide-ResNet-28-10* (Zagoruyko & Komodakis, 2016) from the original implementation of DUE.

Uncertainty head. For DUE we use the original implementation [3] with by default we use the RBF kernel function. For NatPN we use the original implementation [4] but change the uncertainty head with a more expressive density estimator. As seen in Table 5, we found that a more expressive normalizing flow with resampled base (Durkan et al., 2019; Stimper et al., 2022) improves significantly the results over a simpler radial normalizing flow (Rezende & Mohamed, 2015) across all the metrics. For all the experiments (except toys where we use radial flow) we use NSF-R with 16 layers.

Table 5: **Normalizing flow expressivity comparison.** Using more expressive normalizing flow significantely improves all the results for NatPN.

	CIFAR100				Camelyon			
Head	**Accuracy (↑)**	**Brier Score (↓)**	**OOD Pred. (↑)**	**OOD Epis. (↑)**	**Accuracy (↑)**	**Brier Score (↓)**	**OOD Pred. (↑)**	**OOD Epis. (↑)**
Radial	71.09 ± 0.21	52.27 ± 0.28	72.84 ± 1.82	50.95 ± 2.16	83.14 ± 0.93	24.55 ± 1.91	60.27 ± 5.29	69.16 ± 7.94
NSF-R	**71.61 ± 0.07**	**43.44 ± 0.11**	**73.54 ± 1.69**	**72.85 ± 1.25**	**89.84 ± 7.93**	**12.52 ± 6.17**	**64.14 ± 10.42**	**81.33 ± 8.78**

B TRAINING FOR DUMS DETAILS

For all the training optimization experiments, we start by pretraining only the core architecture using the cross-entropy loss. Specifically we train a ResNet18 on the CIFAR100 dataset for 200 epochs with SGD optimizer parametrized with learning rate of 1e-1, momentum of 0.9, and a weight decay of 5e-4. The cosine annealing scheduler is also used (Loshchilov & Hutter, 2017) with the minimum learning rate η_{min} of 5e-4. After the pretraining phase is completed, both DUMs can load the same pretrained encoder for joint training. We joint train for 20 epochs with AdamW optimizer (Loshchilov & Hutter, 2019) and the cosine annealing scheduler, which is applied only to the encoder, with the minimum learning of η_{min} of 1e-5. For NatPN training, we follow the original method in Charpentier et al. (2022a). We *warmup* only the NatPN's uncertainty head before the joint training, and *finetune* after the joint training. We use the optimizer AdamW with the same learning rate of the joint training and warmup for 5 epochs and finetune for 100 epochs. In the warmup phase we use a lambda scheduler that increases linearly the learning rate for each epoch until the desired threshold. In the finetune phase we use a multistep scheduler that scales the learning rate

[2] `https://pytorch.org/vision/stable/models.html`
[3] `https://github.com/y0ast/DUE`
[4] `https://github.com/borchero/natural-posterior-network`

by 0.2 at 70% and 90% of the training. The weight decay used is set to 1e-6 for both the DUM's encoder/head. We use the latent dimension 128 for all the training experiments.

Decoupling learning rate. For decoupling learning rate, we simply use different values for the learning rates of the core architecture and of the uncertainty head. After the decoupling learning rate experiment, we choose the best combination of learning rates through model selection via the validation results and apply it to other experiments. E.g., for the joint training schema and pretraining shema experiments, NatPN uses a learning rate of 1e-4/1e-4 for encoder/head respectively, while for DUE it is 1e-5/5e-3.

Pretraining. For the datasets CIFAR, and Camelyon datasets we first pretraining the encoder alone using the cross-entropy loss, while for MNIST we jointly train from scratch as it requires less computation resources. For pretraining the encoders we use the following setting for each dataset:

- **CIFAR.** We train ResNet18 for 200 epochs with SGD optimizer parametrized with learning rate of 1e-1, and momentum of 0.9, and weight decay of 1e-6. The cosine annealing scheduler is used with the minimum learning rate η_{min} of 5e-4.
- **Camelyon.** We train Wide-ResNet-28-10 for 5 epochs with AdamW optimizer parametrized with learning rate of 1e-3, and weight decay of 1e-8. The cosine annealing scheduler is used with the η_{min} of 1e-5.

Training. After the pretraining the encoder, DUMs can simply load the encoder to continue with the joint training. For each dataset and encoder experiment, we use the following settings for joint training:

- **MNIST.** We directly joint train for 20 epochs with AdamW optimizer for both DUMs. NatPN uses a learning rate of 1e-3/1e-4 for encoder/head respectively, and for DUE it is 1e-3/5e-3. The cosine annealing scheduler is used with the η_{min} of 5e-4 applied only to the encoder. The latent dimension used is 16.
- **CIFAR.** we joint train for 20 epochs with AdamW optimizer for both DUMs. NatPN uses a learning rate of 1e-4/1e-4, and DUE 1e-5/5e-3. The cosine annealing scheduler η_{min} is 1e-5 applied only to the encoder. The latent dimension used is 64.
- **Camelyon.** we joint train for 1 epoch with AdamW optimizer for both DUMs. NatPN's learning rate of 5e-6/1e-5, and DUE 1e-5/5e-3. The cosine annealing scheduler η_{min} is 1e-6 applied only to the encoder. The latent dimension used is 128.

Lastly, we *warmup* and *finetune* the normalizing flow as we did in appendix B. We warmup for 0, 5, 0, and we finetune the it for 60, 200, 5 epochs for MNIST, CIFAR, and Camelyon respectively.

C Architecture for DUMs Details

C.1 Regularization constraints

Bi-Lipschitz training details. In the *None* configuration, we removed the residual connection from the architecture for both the pretraining of the encoder and the joint training phase. In the *Residual* configuration, we did not modify anything since both ResNet and WideResNet already uses the residual connection. While for the *bi-Lipschitz* configuration, we added the spectral normalization during both the pretraining and also joint training phase. Following the original method presented in van Amersfoort et al. (2021), we use the same implementation and applied spectral normalization to the linear, convolution, and batch normalization layers. During the model selection, using the validation set results, we find that the best *Lipschitz constant* c for DUE is 4, and for NatPN is 5. For both the models the power iteration parameter is set to 1. For the toy dataset we use an encoder with 4 linear layers of 128 dimension each.

Reconstruction training details. The decoder reconstructs the input extracted from the last residual block of the encoder, before the pooling layer. During the pretraining phase, both the encoder and decoder are trained with the cross-entropy loss plus a MSE reconstruction term. During the joint training phase, we load the pretrained encoder and decoder, and joint train with the DUMs' respective loss plus the MSE reconstruction term. For the toy dataset we use an encoder with 4 linear layers of 128 dimension each.

Table 6: **Train schema OOD detection.** Uncertainty estimation results broken down for each OOD dataset. We observe that NatPN performs significantly better when using *joint*, while DUE is insensitive to the schema used. The ID dataset used for training is CIFAR100.

Model	OOD Data	Train Schema	OOD Alea. (↑)	OOD Epis. (↑)	OOD Pred. (↑)
NatPN	SVHN	joint	80.64 ± 1.22	65.00 ± 3.18	80.64 ± 1.22
		joint + bn	**82.07 ± 0.62**	69.98 ± 4.40	**82.07 ± 0.62**
		joint + reset	80.77 ± 1.63	**74.67 ± 2.37**	80.77 ± 1.63
		sequential	80.76 ± 0.95	43.99 ± 4.16	80.76 ± 0.95
		sequential + bn	80.64 ± 0.83	44.46 ± 5.63	80.64 ± 0.83
		sequential + reset	78.92 ± 1.06	59.54 ± 4.25	78.92 ± 1.06
	STL10	joint	76.63 ± 0.20	58.79 ± 0.24	76.63 ± 0.20
		joint + bn	76.53 ± 0.22	63.51 ± 0.38	76.53 ± 0.22
		joint + reset	76.92 ± 0.36	**64.44 ± 0.52**	76.92 ± 0.36
		sequential	77.57 ± 0.20	42.83 ± 0.62	77.57 ± 0.20
		sequential + bn	**77.64 ± 0.22**	44.26 ± 0.59	**77.64 ± 0.22**
		sequential + reset	77.35 ± 0.15	57.56 ± 0.38	77.35 ± 0.15
	CelebA	joint	51.42 ± 1.29	28.90 ± 1.59	51.42 ± 1.29
		joint + bn	51.45 ± 1.15	27.67 ± 2.45	51.45 ± 1.15
		joint + reset	52.01 ± 0.46	**32.54 ± 0.32**	52.01 ± 0.46
		sequential	52.58 ± 1.08	24.41 ± 1.94	52.58 ± 1.08
		sequential + bn	52.44 ± 1.11	23.65 ± 1.85	52.44 ± 1.11
		sequential + reset	**53.12 ± 0.92**	26.82 ± 1.41	**53.12 ± 0.92**
	Camelyon	joint	**67.17 ± 5.30**	67.03 ± 9.00	**67.17 ± 5.30**
		joint + bn	61.06 ± 2.70	69.69 ± 5.54	61.06 ± 2.70
		joint + reset	67.07 ± 1.12	**73.45 ± 4.22**	67.07 ± 1.12
		sequential	64.54 ± 2.08	56.23 ± 6.07	64.54 ± 2.08
		sequential + bn	64.34 ± 2.12	49.31 ± 5.99	64.34 ± 2.12
		sequential + reset	63.76 ± 2.13	65.59 ± 4.65	63.76 ± 2.13
	SVHN OODom.	joint	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
		joint + bn	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
		joint + reset	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
		sequential	99.99 ± 0.00	100.00 ± 0.00	99.99 ± 0.00
		sequential + bn	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
		sequential + reset	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
DUE	SVHN	joint	-	-	80.75 ± 0.79
		joint + bn	-	-	80.47 ± 1.09
		joint + reset	-	-	80.92 ± 0.70
		sequential	-	-	81.04 ± 0.82
		sequential + bn	-	-	**81.20 ± 0.86**
		sequential + reset	-	-	80.66 ± 1.07
	STL10	joint	-	-	77.20 ± 0.19
		joint + bn	-	-	77.24 ± 0.22
		joint + reset	-	-	77.42 ± 0.25
		sequential	-	-	77.50 ± 0.09
		sequential + bn	-	-	**77.52 ± 0.13**
		sequential + reset	-	-	77.48 ± 0.17
	CelebA	joint	-	-	47.99 ± 1.25
		joint + bn	-	-	47.90 ± 1.13
		joint + reset	-	-	49.10 ± 0.86
		sequential	-	-	48.07 ± 0.99
		sequential + bn	-	-	48.43 ± 1.06
		sequential + reset	-	-	**49.39 ± 1.29**
	Camelyon	joint	-	-	67.76 ± 2.20
		joint + bn	-	-	67.54 ± 2.33
		joint + reset	-	-	67.03 ± 2.00
		sequential	-	-	67.49 ± 2.58
		sequential + bn	-	-	67.23 ± 2.72
		sequential + reset	-	-	**67.85 ± 2.54**
	SVHN OODom.	joint	-	-	100.00 ± 0.00
		joint + bn	-	-	100.00 ± 0.00
		joint + reset	-	-	100.00 ± 0.00
		sequential	-	-	100.00 ± 0.00
		sequential + bn	-	-	100.00 ± 0.00
		sequential + reset	-	-	100.00 ± 0.00

D PRIOR FOR DUMS DETAILS

For all the prior experiments, we use the same training settings as described in B. In these experiments, we vary the *entropy regularization* λ and the (bottom) *evidence prior* n^{prior} for NatPN, and for the choice of kernel for DUE.

Table 7: **Pretrain schema OOD detection.** Uncertainty estimation results broken down for each OOD dataset. We see how ImageNet pretrained encoder consistently performs better than other settings for 4/5 OOD datasets. The ID dataset used in the joint training phase is CIFAR100.

Model	OOD Data	Pretrain Schema	OOD Alea. (↑)	OOD Epis. (↑)	OOD Pred. (↑)
NatPN	SVHN	None	79.95 ± 1.18	77.87 ± 2.54	79.95 ± 1.18
		C100 (10%)	71.81 ± 1.67	75.16 ± 2.20	71.81 ± 1.67
		C100 (100%)	81.76 ± 1.38	65.72 ± 4.91	81.76 ± 1.38
		ImageNet	**89.34 ± 0.66**	**92.02 ± 0.49**	**89.34 ± 0.66**
	STL10	None	80.34 ± 0.77	78.05 ± 2.37	80.34 ± 0.77
		C100 (10%)	74.64 ± 0.72	62.43 ± 2.92	74.64 ± 0.72
		C100 (100%)	80.70 ± 1.33	66.07 ± 2.60	80.70 ± 1.33
		ImageNet	**85.46 ± 0.50**	**90.76 ± 0.31**	**85.46 ± 0.50**
	CelebA	None	73.59 ± 3.78	**76.19 ± 3.32**	73.59 ± 3.78
		C100 (10%)	73.26 ± 1.79	63.31 ± 3.66	73.26 ± 1.79
		C100 (100%)	**73.61 ± 2.16**	58.11 ± 2.85	**73.61 ± 2.16**
		ImageNet	59.30 ± 3.77	64.80 ± 2.15	59.30 ± 3.77
	Camelyon	None	65.37 ± 10.77	96.95 ± 2.01	65.37 ± 10.77
		C100 (10%)	20.84 ± 5.27	91.58 ± 7.56	20.84 ± 5.27
		C100 (100%)	58.69 ± 11.10	94.63 ± 2.85	58.69 ± 11.10
		ImageNet	**90.64 ± 2.47**	**97.82 ± 0.56**	**90.64 ± 2.47**
	SVHN OODom.	None	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
		C100 (10%)	99.96 ± 0.04	100.00 ± 0.00	99.96 ± 0.04
		C100 (100%)	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
		ImageNet	**100.00 ± 0.00**	**100.00 ± 0.00**	**100.00 ± 0.00**
DUE	SVHN	None	-	-	82.66 ± 0.77
		C100 (10%)	-	-	77.24 ± 0.53
		C100 (100%)	-	-	78.76 ± 1.74
		ImageNet	-	-	**92.18 ± 0.11**
	STL10	None	-	-	78.91 ± 1.06
		C100 (10%)	-	-	75.98 ± 0.87
		C100 (100%)	-	-	79.63 ± 1.34
		ImageNet	-	-	**89.56 ± 0.53**
	CelebA	None	-	-	65.64 ± 1.77
		C100 (10%)	-	-	**74.62 ± 1.56**
		C100 (100%)	-	-	74.60 ± 1.11
		ImageNet	-	-	72.19 ± 1.42
	Camelyon	None	-	-	73.00 ± 2.81
		C100 (10%)	-	-	34.84 ± 3.53
		C100 (100%)	-	-	75.58 ± 5.17
		ImageNet	-	-	**97.28 ± 0.47**
	SVHN OODom.	None	-	-	100.00 ± 0.00
		C100 (10%)	-	-	99.51 ± 0.12
		C100 (100%)	-	-	99.99 ± 0.00
		ImageNet	-	-	**100.00 ± 0.00**

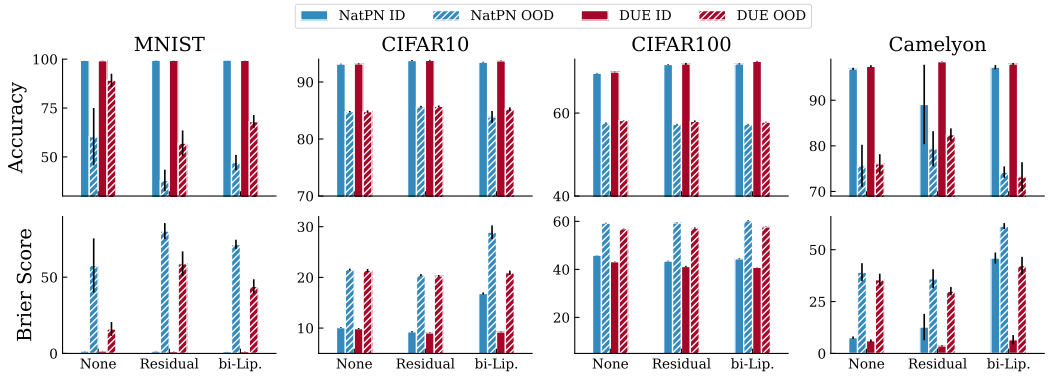


Figure 6: Results OOD generalization and OOD detection results of DUMs with none, residual and bi-lipschitz **architecture constraints**. Bi-lipschitz and more specifically can improve OOD detection by mitigating feature collapse (see fig. 8a) at the expense of degrading OOD generalization.

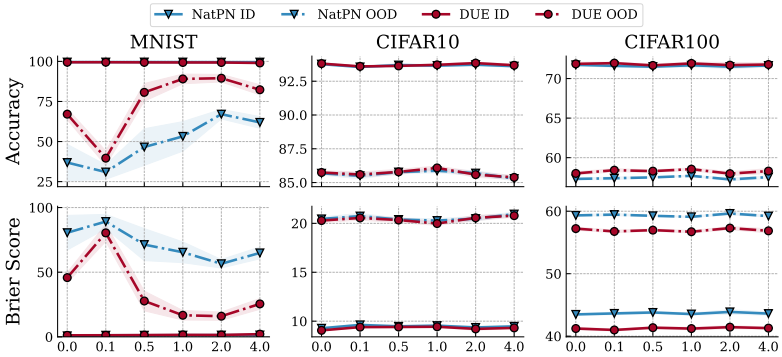
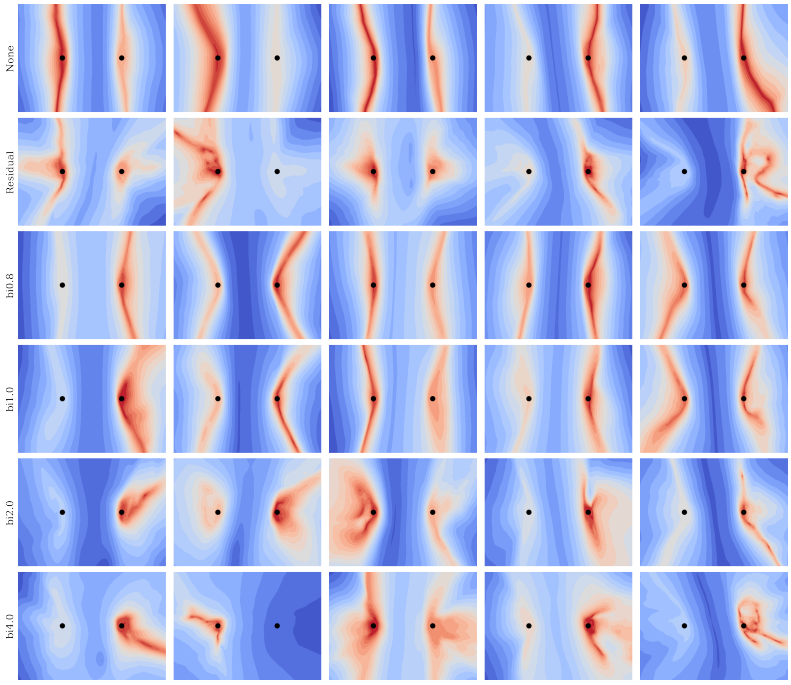


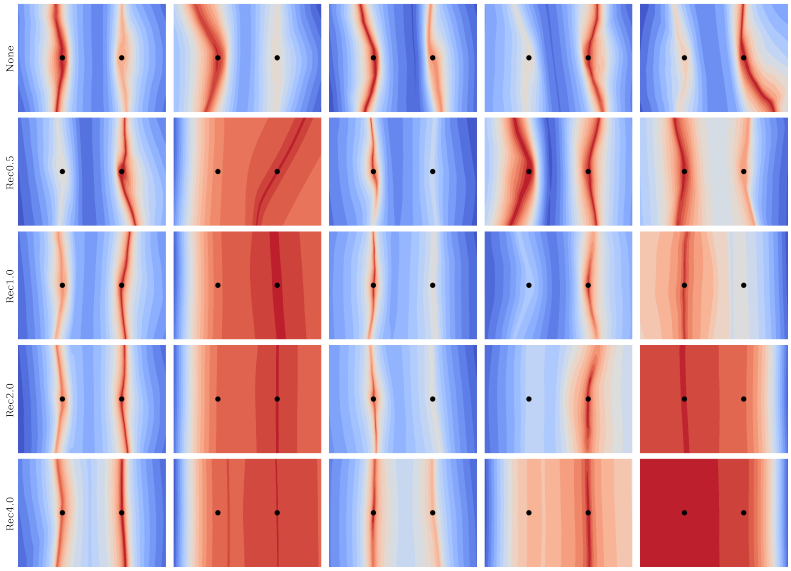
Figure 7: Results OOD generalization and OOD detection results of DUMs with reconstruction **architecture constraints**. Increasing the strength of the reconstruction factor λ improves the OOD generalization only on the simpler MNIST/CMNIST datasets but fails for more complex datasets.

Table 8: **Encoder architecture OOD detection.** For each training dataset we show the uncertainty estimation results on the corresponding OOD dataset. We observe that new architectures (EfficientNet, Swin) have consistently better results. Interestingly, the transformer based model Swin, is not able to detect out-of-domain data, which should be easy in principle.

Model	OOD Data	Architecture	OOD Alea. (↑)	OOD Epis. (↑)	OOD Pred. (↑)
NatPN	SVHN	ResNet18	**89.90 ± 1.22**	78.14 ± 3.13	**89.90 ± 1.22**
		ResNet50	89.34 ± 0.66	92.02 ± 0.49	89.34 ± 0.66
		EfficientNet_V2_S	88.65 ± 0.68	92.52 ± 0.60	88.65 ± 0.68
		Swin_T	87.73 ± 1.36	**94.17 ± 0.74**	87.73 ± 1.36
	STL10	ResNet18	**90.99 ± 0.37**	83.90 ± 0.59	**90.99 ± 0.37**
		ResNet50	85.46 ± 0.50	90.76 ± 0.31	85.46 ± 0.50
		EfficientNet_V2_S	88.68 ± 0.62	91.11 ± 0.47	88.68 ± 0.62
		Swin_T	85.44 ± 0.56	**92.16 ± 0.40**	85.44 ± 0.56
	CelebA	ResNet18	66.46 ± 3.48	50.61 ± 2.14	66.46 ± 3.48
		ResNet50	59.30 ± 3.77	64.80 ± 2.15	59.30 ± 3.77
		EffNet_V2_S	**67.04 ± 1.74**	66.29 ± 1.45	**67.04 ± 1.74**
		Swin_T	63.60 ± 2.16	**72.80 ± 1.00**	63.60 ± 2.16
	Camelyon	ResNet18	90.09 ± 4.43	96.28 ± 1.06	90.09 ± 4.43
		ResNet50	90.64 ± 2.47	97.82 ± 0.56	90.64 ± 2.47
		EfficientNet_V2_S	94.56 ± 0.79	97.42 ± 0.44	94.56 ± 0.79
		Swin_T	**95.43 ± 1.47**	**98.71 ± 0.50**	**95.43 ± 1.47**
	SVHN OODom.	ResNet18	**100.00 ± 0.00**	**100.00 ± 0.00**	**100.00 ± 0.00**
		ResNet50	**100.00 ± 0.00**	**100.00 ± 0.00**	**100.00 ± 0.00**
		EfficientNet_V2_S	**100.00 ± 0.00**	**100.00 ± 0.00**	**100.00 ± 0.00**
		Swin_T	97.37 ± 0.29	93.31 ± 1.42	97.37 ± 0.29
DUE	SVHN	ResNet18	-	-	88.77 ± 0.28
		ResNet50	-	-	92.18 ± 0.11
		EfficientNet_V2_S	-	-	90.95 ± 0.53
		Swin_T	-	-	**93.62 ± 0.39**
	STL10	ResNet18	-	-	**90.66 ± 0.48**
		ResNet50	-	-	89.56 ± 0.53
		EfficientNet_V2_S	-	-	89.08 ± 0.39
		Swin_T	-	-	89.73 ± 0.36
	CelebA	ResNet18	-	-	64.75 ± 1.44
		ResNet50	-	-	72.19 ± 1.42
		EffNet_V2_S	-	-	**72.28 ± 1.76**
		Swin_T	-	-	69.37 ± 0.67
	Camelyon	ResNet18	-	-	96.01 ± 1.13
		ResNet50	-	-	97.28 ± 0.47
		EfficientNet_V2_S	-	-	94.82 ± 0.65
		Swin_T	-	-	**99.33 ± 0.14**
	SVHN OODom.	ResNet18	-	-	**100.00 ± 0.00**
		ResNet50	-	-	**100.00 ± 0.00**
		EfficientNet_V2_S	-	-	**100.00 ± 0.00**
		Swin_T	-	-	97.47 ± 0.22

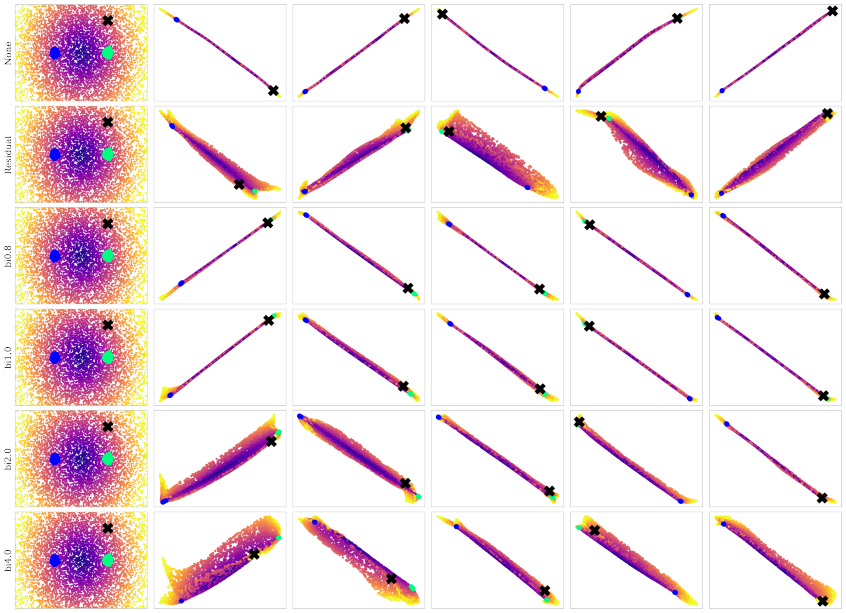


(a) Bi-Lipschitz

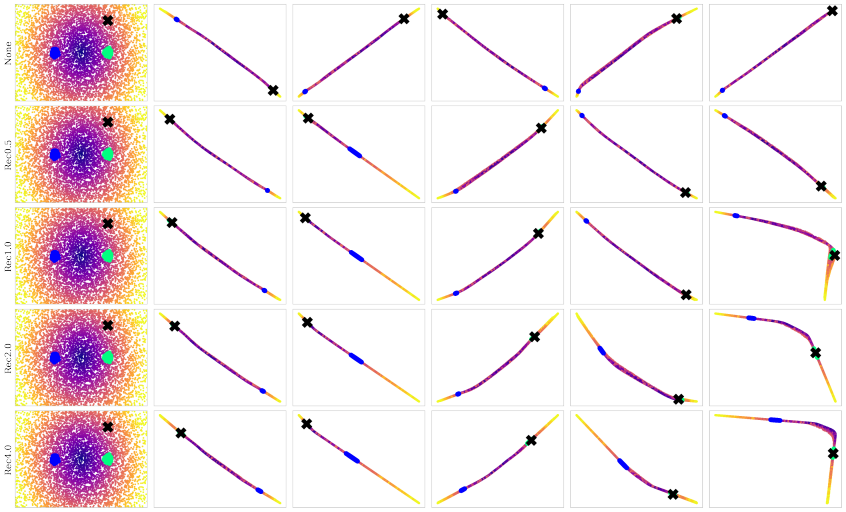


(b) Reconstruction

Figure 8: **Regularization constraint toy dataset uncertainty boundaries with NatPN.** The two black dots represent the center of two different class of Gaussian data, sharing the same y-axis to trigger the *feature collapse* phenomenon. The color represents the likelihood produced by the uncertainty head. Each row is a different setting, e.g. *bi1.0* is the bi-Lipschitz constraint with the Lipschitz constant $c = 1$ and *rec1.0* is the reconstruction term with $\lambda = 1$. Each column is a different seed initialization. (top) **Bi-Lipschitz** experiment shows that the core encoder architecture constrained with a larger *Lipschitz constant* in the last two rows behaves similar to the encoder constrained with only the residual connection (second row) showing that relaxing the spectral normalization constraint falls back to the residual connection, preventing the feature collapse. (bottom) **Reconstruction** experiment shows that it does not help to prevent feature collapse by itself. The core encoder architecture is not constrained with bi-Lipschitz.

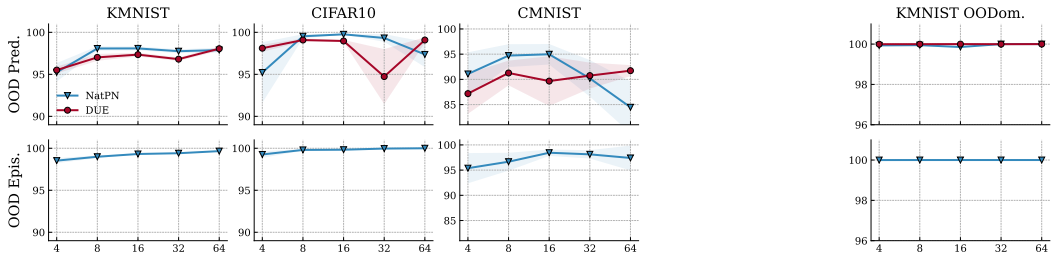


(a) Bi-Lipschitz

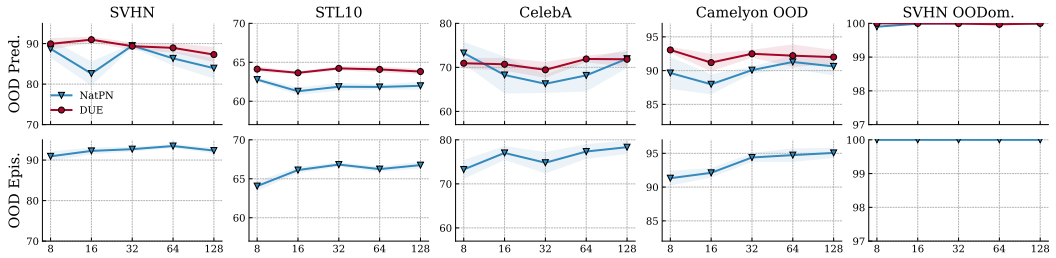


(b) Reconstruction

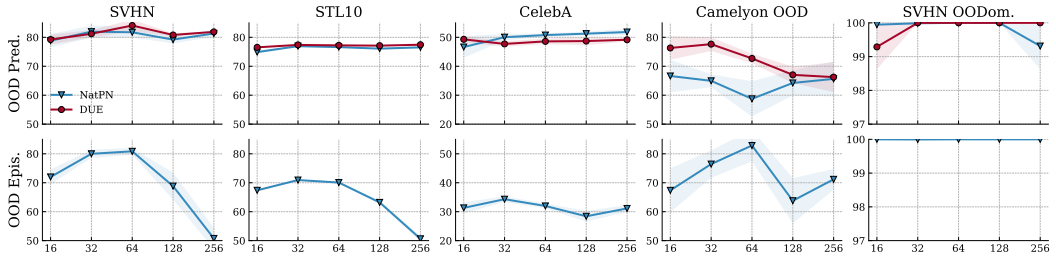
Figure 9: **Regularization constraint toy dataset feature collapse with NatPN.** Similarly to (van Amersfoort et al., 2021), we run the toy experiment where the first column represent two Gaussian class of data, sharing the same y-axis center to trigger the *feature collapse* phenomenon, and a grid of unrelated point to simulate the space distorsion (colors are based on the Gaussian's generating distribution). Each row is a different setting, e.g. *bi1.0* is the bi-Lipschitz constraint with the Lipschitz constant $c = 1$ and *rec1.0* is the reconstruction term with $\lambda = 1$. Each column is a different seed initialization. Results are the same as 8a. Larger Lipschitz constant c reverts back to the residual connection, and reconstruction regularization collapses the 2D dimension into one single dimension.



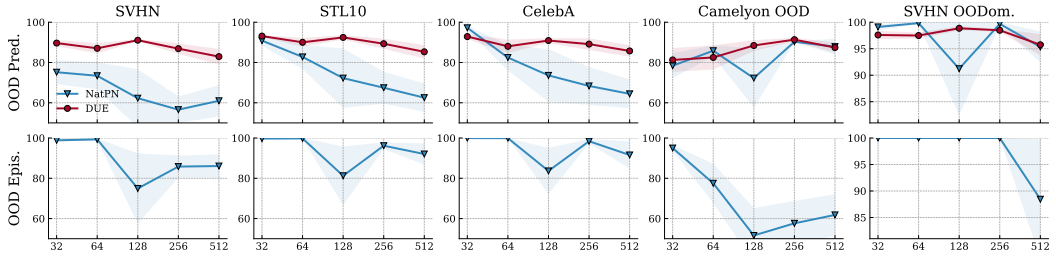
(a) MNIST



(b) CIFAR10



(c) CIFAR100



(d) Camelyon ID

Figure 10: **Latent dimension OOD detection.** For each training dataset we show the uncertainty estimation results on the corresponding OOD dataset. NatPN encounters numerical instabilities with high latent dimension on Camelyon dataset, while DUE is less sensitive to the variation.

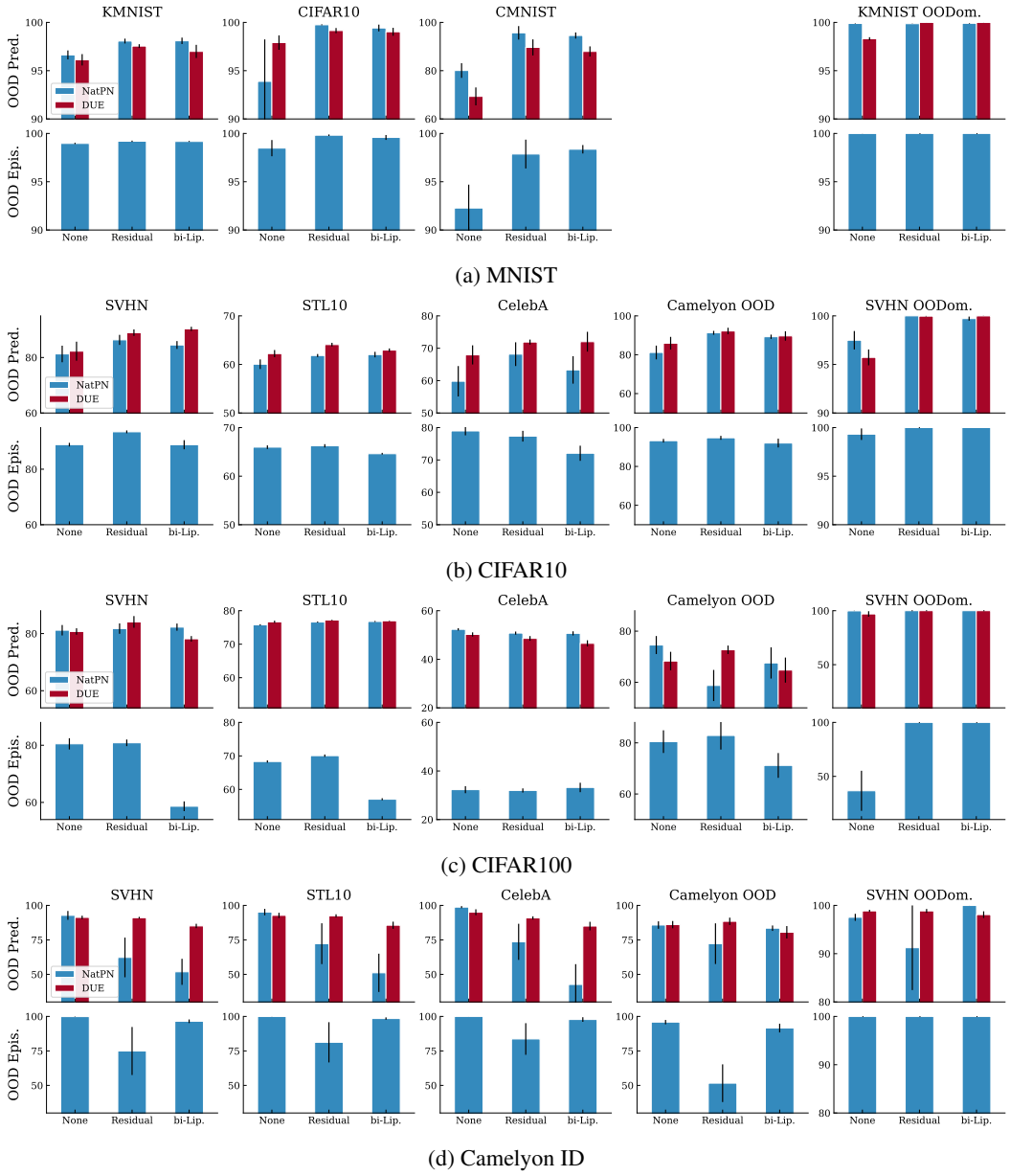


Figure 11: **Bi-lipschitz OOD detection.** For each training dataset we show the uncertainty estimation results on the corresponding OOD dataset. Bi-Lipschitz improvements are not consistent across different OOD datasets.

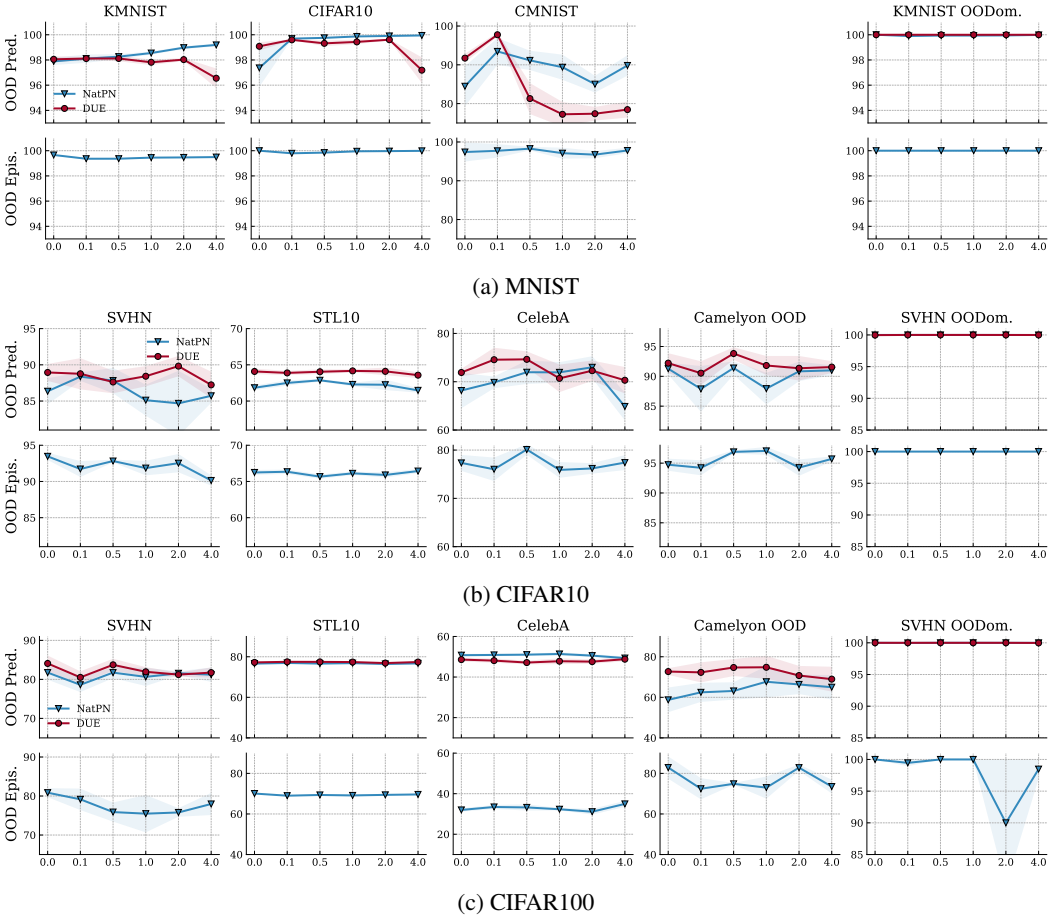


Figure 12: **Reconstruction regularization OOD detection.** Increasing the weight coefficient of the reconstruction loss term improved the OOD detection of NatPN in MNIST. However, we did not observe improvements on more complex datasets such as CIFAR.

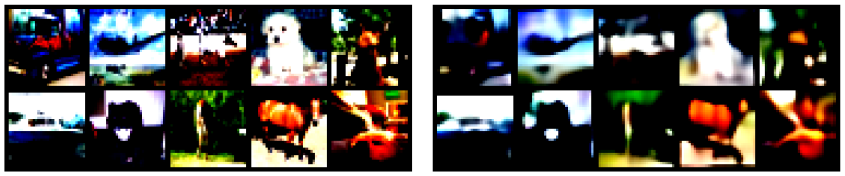


Figure 13: **Reconstruction regularization CIFAR samples.** (left) The original input (right) the reconstructed input after NatPN's joint training phase. The reconstruction discards detailed information compared to the original input.

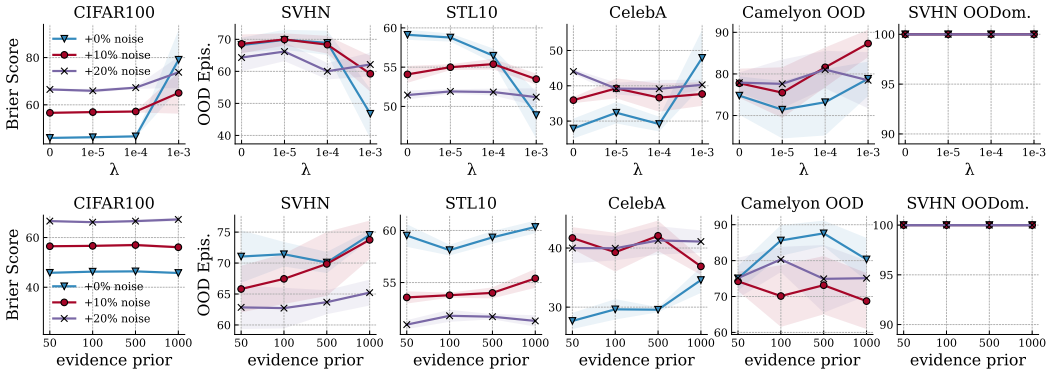


Figure 14: Results of enforcing different **prior** in NatPN on CIFAR100 by changing the (top) *entropy regularization* λ and the (bottom) *evidence prior* n^{prior}. Different priors do not lead consistent results improvements.