6 SUPPLEMENT

We here provide additional theoretical background, implementation and optimization details, additional results, comparisons and ablation studies. In particular, we outline a theoretical justification of the poor calibration of DUMs in Sec. 6.1, and report additional details on the theoretical background necessary to understand DUMs' design choices in Sec. 6.2. A description of Lipschitz regularization techniques can be found in Sec. 6.2.1, while Sec. 6.2.3 summarizes common choices of uncertainty estimation techniques for discriminative and generative DUMs.

Sec. 6.3.1/Sec. 6.3.2 show additional results for image classification/semantic segmentation. We report optimization/implementation details in Sec. 6.4/Sec. 6.5. Furthermore, we describe the quantification of uncertainty for image classification Sec. 6.5.3 and semantic segmentation Sec. 6.5.4. Finally, we provide details on the data collection process in CARLA and examples from the sequences collected for semantic segmentation Sec. 6.6.

6.1 THEORETICAL JUSTIFICATION

Given a neural network NN trained on a dataset $D = (X, Y)$ with $X = \{x_i\}_{i \in |D|}$ and $Y = \{y_i\}_{i \in |D|}$, some DUMs (42; 20; 19) require modeling of the intermediate activations $p(z|X, Y)$ of the NN to derive estimates of the predictive uncertainty. For this reason, different types of regularization over the feature space are applied with the aim of making the latent distribution representative of the input one. Since this only captures the data distribution through the lense of a fixed set of model parameters, we argue that a key ingredient is missing to account for the total variability over latent distribution.

Taking into account the distribution $p(\theta|X, Y)$ over networks' parameters θ, which is approximated by a surrogate distribution $q(\theta)$, we obtain for the distribution $p(z|X, Y)$:

$$\begin{aligned} p(z|X,Y) &= \int_x p(z|x,X,Y)p(x|X,Y)dx \\ &= \int_x \left(\int_\theta p(z|x,\theta)p(\theta|X,Y)d\theta \right) p(x|X,Y)dx \\ &= \int_x \left(\int_\theta p(z|x,\theta)q(\theta)d\theta \right) p(x|X,Y)dx \\ &= \int_\theta \left(\int_x p(z|x,\theta)p(x|X,Y)dx \right) q(\theta)d\theta \end{aligned} \tag{1}$$

DUMs typically assume that network weights are fixed, i.e. they are distributed according to a Dirac delta function:

$$p(\theta|X,Y) \approx q(\theta) = \delta(\theta) = \begin{cases} +\infty, & \theta = \hat{\theta} \\ 0, & \theta \neq \hat{\theta} \end{cases} \tag{2}$$

subject to $\int_{-\infty}^{\infty} \delta(x)\, dx = 1$. Then,

$$\begin{aligned} p(z|X,Y) &= \int_\theta \left(\int_x p(z|x,\theta)p(x|X,Y)dx \right) q(\theta)d\theta \\ &= \int_x \left(p(z|x,\hat{\theta}) \right) p(x|X,Y)dx \end{aligned} \tag{3}$$

Intuitively, DUMs only approximate the inner integral over the distribution of inputs x, since only the distribution over the input space is taken into account and weights $\hat{\theta}$ are fixed. While this justifies the good performance of DUMs on OOD detection, failing to model the weights distribution may not account for the total variability over latent distributions, thus underestimating the output predictive uncertainty and making it a bad hint for networks' expected error.

Incorporating the lack of knowledge over the network's weights is a promising direction to make DUMs well calibrated.

6.2 DUMs - FUNDAMENTALS

We here introduce the theoretical background necessary for the understanding of DUMs. Initially (Sec. 6.2.1, Sec. 6.2.2 and Sec. 6.2.3) we maintain a modularized perspective of DUMs by describing individual components including their advantages and disadvantages. Subsequently (Sec. 6.2.4), we shed light on each DUMs individually using insights from the modularized considerations.

In particular, We describe regularization techniques, including Lipschitz regularization (Sec. 6.2.1) for enforcing distance awareness and informative representations (Sec. 6.2.2). Sec. 6.2.3 summarizes discriminative and generative approaches to uncertainty estimation in DUMs. Finally, Sec. 6.2.4 discusses each individual method used in our empirical comparison.

6.2.1 REGULARIZATION TECHNIQUES - DISTANCE AWARENESS

The fundamental idea of distance-aware hidden representations is to avoid feature collapse by enforcing distances between latent representations to mirror distances in the input space. This can be achieved by constraining the Lipschitz constant, as it enforces a lower and an upper bound to expansion and contraction performed by an underlying neural network. More formally, given any pair of inputs x_1 and x_2 the following lower and upper bounds must hold for the resulting activation of a feature extractor f_θ with parameters θ: $c_1||x_1 - x_2||_I \leq ||f_\theta(x_1) - f_\theta(x_2)||_F \leq c_2||x_1 - x_2||_I$. c_1 and c_2 denote respectively the lower and upper bound for the Lipschitz constant, and $||\cdot||_I$ and $||\cdot||_F$ are the chosen metrics in the input and feature space respectively.

Recent proposals have primarily adopted two methods to impose the bi-Lipschitz constraint.

Gradient Penalty. First introduced to regularize the Lipschitz constant in GAN training (78), a two-sided gradient penalty is used as an additional loss term to enforce detectability in the feature space of changes in the input by DUQ (13). The gradient penalty is formulated as an additional loss term that regularises the Frobenius norm $||J||_F$ of the Jacobian J of a NN to enforce a bi-Lipschitz constraint. Therefore, the training loss of a NN is typically enhanced with the absolute difference between $||J||_F$ and some chosen positive constant.

Given a model g and an input x, regularising the Frobenius norm $||J||_F$ of its Jacobian J constraints its Lipschitz constant. Therefore, the following two-sided gradient penalty is used: $\lambda \left[||\nabla_x g(x)||_F - 1\right]^2$, where λ is the regularization strength, $||\cdot||_2$ is the L_2 norm, the target bi-Lipschitz constant is 1. For more details, refer to (13).

Spectral Normalization. The two-sided gradient penalty described above requires backpropagating through the Jacobian of a NN and is, thus, computationally demanding. A more efficient technique is SN (49). For each layer $g : \mathbf{h}_{in} \rightarrow \mathbf{h}_{out}$, SN normalizes the weights W of each layer using their spectral norm $sn(W)$ to constrain the bi-Lipschitz constant. Thus, weight matrices are normalized according to: $W_{sn} = \frac{W}{c \cdot sn(W)}$. This effectively constrain the layer's Lipschitz norm $||g||_{Lip} = sup_{\mathbf{h}} sn(\nabla g(\mathbf{h}))$, where $sn(A)$ is the spectral norm of the matrix A, equivalent to its largest singular value. Consequently, SN normalizes the spectral norm of the weights W of each layer to satisfy the soft-Lipschitz constraint $sn(W) = c$ (hard- if the Lipschitz constant $c = 1$): $W_{sn} = W/sn(W)$. Note, that spectral normalization requires residual layers. We refer to (15) for further details.

Runtime. Let N denote the number of parameters of the underlying neural network and B denote the batch size used during training of the underlying discriminative task. Gradient penalty leads to additional runtime/memory cost of $O(NB)$. This originates from backpropagation through the gradients of the input which essentially doubles the computation during backpropagation. Spectral normalization leads to additional runtime/memory cost of $O(N)$ since its complexity equates to applying the affine layers of a model additionally on a single sample.

Overall, we summarize the advantages and disadvantages of each regularization technique enforcing distance aware representations.

- **Distance awareness (general):**
  - **Advantages:** Can be used in combination with GPs and RBF kernels which both assume distance-aware inputs.

  - **Disadvantages:** Assumes an underlying distance metric (e.g. L_2). This can be unsuitable/problematic for some data distributions (e.g. images). Does not correlate with OOD detection performance.
- **Gradient Penalty:**
  - **Advantages:** Architecture-agnostic.
  - **Disadvantages:** High computational and memory costs due to backpropagation through the input's gradients.
- **Spectral Normalization:**
  - **Advantages:** Computationally more efficient compared to gradient penalty.
  - **Disadvantages:** Not architecture-agnostic. Requires the use of residual layers.

6.2.2 INFORMATIVE REPRESENTATIONS

Unlike approaches enforcing distance aware representations, informative representations do not rely on an underlying distance metric. These approaches rather aim at maximizing the mutual information between input data distribution and the distribution of hidden representations heuristically (20) or exactly (42; 17). Subsequently, we discuss the different approaches in greater detail.

Constrastive learning. DCU (42) first pretrains its model using contrastive learning (54). Subsequently, they finetune on the actual classification task by training simultaneously on classification and contrastive learning objective. This approach provably encourages the model to increase the mutual information between the input distribution and the distribution of hidden representations (53). A disadvantage with this approach is the large batch size required for training the contrastive objective. Furthermore, it heavily depends on the underlying data augmentations which need to be tailored to the discriminative task (e.g. classification) at hand.

Reconstruction regularization. The authors of MIR (20) try to heuristically increase the information content about the input in the hidden representations. Therefore, they require the model to be able to reconstruct its input from its hidden representations using a separate decoder module during training. The entire approach can also be viewed as a constrained autoencoder, where the constraint is the objective of the discriminative task at hand (e.g. classification). While this approach is only a heuristic, it is more agnostic of the underlying discriminative task which the model solves.

Entropy regularization. PostNet (20) learns the distribution of hidden representations end-to-end during training of the discriminative model. They parameterize the distribution using one normalizing flow (radial flow) per class. While they do not explicitly mention the problem of feature collapse, their entropy regularization loss fulfills this purpose. In particular, they maximize the entropy of the predicted Dirichlet distribution $D(\alpha^{(i)})$ parameterized by $\alpha^{(i)} = (\alpha_1^{(i)}, \dots, \alpha_c^{(i)})$ for a classification problem with c classes. Each $\alpha_j^{(i)}$ is given by $\alpha_j^{(i)} = \beta^{prior} + N_c P(z^{(i)}|c, \phi)$. Here, β^{prior} denotes a constant prior term shared across classes, N_c denotes the number of occurrences of class c in the training set, $z^{(i)}$ denotes the hidden representation of some input $x^{(i)}$ and $P(z^{(i)}|c, \phi)$ denotes the radial flow associated with class c with parameters ϕ. Importantly, β^{prior} is set to 1 in their experiments which leads to $\alpha_j^{(i)} \geq 1 \forall j \in [1, \dots, c]$. PostNet then encourages large entropies of the Dirichlet distribution during training. The entropy is given by

$$H(D(\alpha^{(i)})) = \log(B(\alpha^{(i)})) + (\alpha_0^{(i)} - c)\psi(\alpha_0^{(i)}) - \sum_{j=0}^{c}(\alpha_j^{(i)} - 1)\psi(\alpha_j^{(i)}) \tag{4}$$

where B is there beta-function, ψ is the Digamma function and $\alpha_0^{(i)} = \sum_{j=1}^{c} \alpha_j^{(i)}$. Importantly, this function has a global maximum for $\alpha_j^{(i)} = 1 \forall j \in [1, \dots c]$. Since $\beta^{prior} = 1$, this term further encourages the normalizing flows to produce likelihoods close to zero. Thus, we encourage large values of the negative log-likelihoods and, consequently, entropies under each radial flow.

Runtime. Let N denote the number of parameters of the underlying neural network and B denote the batch size used during training of the underlying discriminative task. Contrastive learning leads to additional runtime/memory cost of $O(N)$. This originates from the fact that it requires large batch sizes and thus likely increases the batch size compared to the original discriminative task. Reconstruction regularization (20) and entropy regularization (17) lead to additional runtime/memory

cost of $O(B)$, since the size of the decoder model in reconstruction regularization (20), and resp. the normalizing flows in entropy regularization (17), are in principal independent of the size of the original model.

Overall, we summarize the advantages and disadvantages of each regularization technique enforcing informative representations.

- **Informative representations (general):**
  - **Advantages:** Does not rely on an underlying distance metric.
  - **Disadvantages:** When paired with generative modeling of hidden representations, strong regularization is expected to show similar pathologies as explicit generative models trained directly on the data distribution (58). Moreover, it cannot be paired with RBF kernels and GPs approximations based on RBF kernels, since they work under the assumption that also the feature extractor is a distance-preserving function.
- **Contrastive Learning:**
  - **Advantages:** Is shown to simultaneously boost predictive performance (54; 42) - particularly on classification. Architecture-agnostic. Provably maximizes mutual-information between input distribution and distribution of hidden representations (53).
  - **Disadvantages:** High computational and memory costs due to the necessity of large batch sizes. Contrastive learning needs to be tailored (e.g. data augmentations) to the underlying discriminative task.
- **Reconstruction regularization:**
  - **Advantages:** Architecture-agnostic.
  - **Disadvantages:** Only heuristically maximizes mutual information between input distribution and distribution of hidden representations.
- **Entropy regularization:**
  - **Advantages:** Assuming a deterministic neural network, enforcing large entropy in the latent space equates maximizing mutual-information between input distribution and distribution of hidden representations.
  - **Disadvantages:** Not architecture-agnostic, since it requires Batch Normalization prior to entropy regularized hidden representations for training stabilization. Further, requires low-dimensional hidden representations.

6.2.3 Uncertainty Estimation

Training a feature extractor under the regularization constraints imposed by distance awareness (Sec. 3.1.1, Sec. 6.2.1) or representation informativeness (Sec. 3.1.2) allows to leverage intermediate representations to quantify uncertainty over network's predictions. Extending Sec. 3.2, we here distinguish between *generative* and *discriminative* approaches to uncertainty quantification in DUMs and provide a detailed categorization of such techniques.

Generative approaches. Given a model trained under some above-discussed regularization constraint, generative approaches estimate the distribution of hidden representations by fitting density models on the regularized feature space, and use the likelihood as uncertainty metric to detect OOD samples. MIR (20), DDU (19) and DCU (42) learn the density of hidden representations post-training based on the features observed on the training data. In contrast, PostNet (17) learns the density model end-to-end with the underlying discriminative model.

Predominantly, class-conditional GMMs are fitted on the regularized intermediate feature space to estimate its distribution, as done in MIR (20), DDU (19) and DCU (42) - i.e. one multi-variate gaussian per class, to the hidden representations post training. Subsequently, log-likelihood (20) or the log-likelihood of the mixture component associated with the predicted class (19; 42) is used as a proxy for epistemic uncertainty.

On the other hand, PostNet (17) learns the class-conditional distribution of hidden representations end-to-end using normalizing flows (in particular radial flows). They learn one normalizing flow per class. In their work the class-conditional distribution is used to parameterize a Dirichlet distribution. Following PostNet's notation, the parameters $\alpha^{(i)}$ of the Dirichlet distribution associated with a

particular sample $x^{(i)}$ are given by $\alpha_c^{(i)} = \beta^{prior} + \beta^i$ with $\beta^i = N_c P(z^{(i)}|c, \phi)$. Here, c denotes the class, β^{prior} a constant prior term shared across all classes, N_c the number of samples observed in class c and $P(z^{(i)}|c, \phi)$ (ϕ are the parameters of the normalizing flow) is the probability of observing the hidden representation $z^{(i)}$ given the normalizing flow associated with class c. Ultimately, epistemic uncertainty is then quantified as the maximum alpha among all classes. Thus, the epistemic uncertainty is directly derived from the likelihood of the normalizing flow associated with the predicted class of the NN, which mostly corresponds to the normalizing flow with the maximum likelihood assuming a balanced class distribution. We refer to (17) for a more detailed treatment.

Empirically, we find generative approaches to show worse calibration. The underlying assumption of generative approaches to uncertainty estimation is that locations in feature space entail information about the correctness of predictions. While this is arguably true, features also contain additional information that which can render them suboptimal for judging the correctness of predictions due to ambiguities.

Discriminative While generative approaches use the likelihood produced by an explicit generative model fit to the distribution of regularized hidden representations to quantify uncertainty, discriminative methods directly rely on the predictions based on regularized representations.

Centroid-based techniques use distances between points in the latent space to parameterize predictions. Centroids are defined with respect to the distribution of the feature space generated by the training set. Mandelbaum *et al.* (12) propose to use a Distance-based Confidence Score (DCS) to estimate local density at a point as the Euclidean distance in the embedded space between the point and its k nearest neighbors in the training set. Similarly, DUQ (13) builds on Radial Basis Function (RBF) networks (59), which requires the preservation of input distances in the output space which is achieved using the gradient penalty. The class-specific centroids used in the RBF kernel are maintained as a running mean of the features observed for each class.

Other methods are based on the idea that, since GPs with RBF kernels are distance preserving functions (15), they can be combined with regularization techniques that enforce distance awareness of the feature extractor (15; 18) to obtain an end-to-end distance-aware model. The uncertainty can then be computed at the network's output level as the Dempster-Shafer metric (15) or the softmax entropy (18). Based on this intuitive idea, SNGP (15) and DUE (18) simply rely on different approximations of the GP, adopting respectively the Laplace approximation based on the random Fourier feature (RFF) expansion of the GP posterior (60) and the inducing point approximation (61; 62). Another minor difference lies in the spectral normalization algorithm, with DUE providing a SN implementation also for batch normalization layers. While both methods rely on spectral-normalized feature extractors, they could in principle be applied together with any distance-preserving regularization technique. For example, the GPs could be placed on top of a feature extractor trained with gradient penalty (13) to regularize the bi-Lipschtiz constant.

6.2.4 A METHOD-ORIENTED PERSPECTIVE ON INDIVIDUAL DUMS

Here, we discuss each DUMs in our empirical comparison individually. Furthermore, Tab. 4 provides a comparison of DUMs used in our empirical evaluation regarding properties which are interesting for practitioners. We consider four characteristics: Architecture/task constraints, well calibrated uncertainties and computational/memory overhead. We consider the computational/memory overhead of a method not minimal when it scales with at least $O(N)$ where N denotes the number of parameters of the underlying neural network. This is the case for contrastive learning in DCU (42) due to large batch sizes, the gradient penalty in DUQ (13) due to backpropagation through the gradients of a neural network's input and spectral normalization in SNGP (15) and DDU (19).

SNGP (15) uses spectral normalization for regularizing hidden representations (see Sec. 6.2.1). While this denotes an efficient approach to enforcing distance-aware representations, it renders them dependent on an underlying distance metric in the input space. Moreover, they require the underlying model to be composed of residual layers. Furthermore, we find that enforcing distance awareness, does not directly correlate with OOD detection performance. SNGP estimates uncertainty by replacing the softmax layer with a GP based on the RBF kernel. In particular, they use a Laplace approximation of the GP. They estimate epistemic uncertainty using the Dempster-Shafer metric (15). We find that SNGP yields reasonable uncertainty calibration.

Method	Architecture constraints	Task constraints	Well-calibrated	Minimal computational/memory cost
SNGP(15)	residual layers	classification	✓	✗
DUQ(13)	-	classification	✗	✗
DDU(19)	residual layers	-	✗	✗
DCU(42)	-	classification	✗	✗
MIR(20)	-	-	✗	✓
PostNet(17)	batch normalization	classification	✗	✓

Table 4: Qualitative comparison of different DUMs used in our empirical comparison. We are interested in four characteristics that are interested from a practical perspective: Architecture/task constraints, well calibrated uncertainties and computational/memory overhead. We consider the computational/memory overhead of a method not minimal when it scales with at least $O(N)$ where N denotes the number of parameters of the underlying neural network. This is the case for contrastive learning in DCU (42) due to large batch sizes, the gradient penalty in DUQ (13) due to backpropagation through the gradients of a neural network's input and spectral normalization in SNGP (15) and DDU (19).

DUQ (13) prevents feature collapse by enforcing distance-aware hidden representations. Distance-awareness is enforced using the gradient penalty. While the latter allows DUQ to be model agnostic, it dramatically increases the computational/memory cost at training time. Moreover, following the general drawbacks of enforcing distance-aware representations it depends on an underlying distance metric in the input space. DUQ is only applicable to classification and replaces the softmax output layer using an RBF kernel which compares observed representations to centroids where each class in the classification problem is associated with one centroid. These centroids are updated using a running mean during the training of the model. This renders DUQ sensitive to instabilities at training time in case the mean updates are noisy. For example, the latter case can arise when the number of classes in the classification problem becomes large.

DDU (19) uses spectral normalization for regularizing hidden representations (see Sec. 6.2.1). While this denotes an efficient approach to enforcing distance-aware representations, it renders them dependent on an underlying distance metric in the input space. Moreover, they require the underlying model to be composed of residual layers. In order to estimate uncertainty, DDU estimates the distribution of hidden representations of the penultimate layer using a class-conditional GMM, i.e. they train one multivariate Gaussian per class. Then, epistemic uncertainty is approximated as the negative log-likelihood of the mixture components with the highest probability. This approach to uncertainty estimation allows DDU to be applied across different tasks. However, due to the generative approach to uncertainty estimation it yields poorly calibrated uncertainty.

DCU (42) enforces informative representations at training time to counter feature collapse. To this end DCU uses constrastive learning (see Sec. 6.2.2) as a regularization objective. While this approach has theoretical guarantees for maximizing the information content in the hidden representations and is architecture-agnostic, in practically requires very large batch size to generate hard negative samples at training time (54). Moreover, while the contrastive learning objective boosts performance on classification, it is not directly transferable to other tasks than classification since those may require a different set of data augmentations which are essential to the success of contrastive learning (54). In order to estimate uncertainty, DCU also estimates the distribution of hidden representations of the penultimate layer using a class-conditional GMM, i.e. they train one multivariate Gaussian per class. Then, epistemic uncertainty is approximated as the negative log-likelihood of the mixture components with the highest highest probability. This approach to uncertainty estimation allows DCU to be applied across different tasks. However, due to the generative approach to uncertainty estimation it yields poorly calibrated uncertainty.

MIR (20) regularizes hidden representations using reconstruction regularization. While this approach only heuristically increases the information content in the hidden representations, it is efficient and architecture- and task-agnostic. In order to estimate uncertainty, MIR also estimates the distribution of hidden representations of the penultimate layer using a class-conditional GMM, i.e. they train one multivariate Gaussian per class. Then, epistemic uncertainty is approximated as the negative

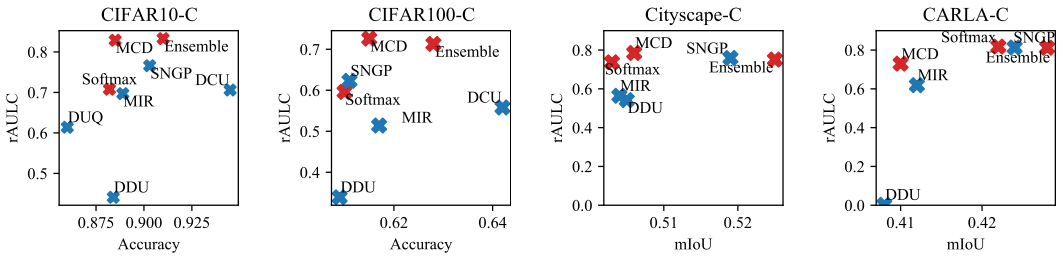


Figure 5: Scatter plots for Accuracy/mIoU versus rAULC on 4 testsets, based on Tab. 2 and 3. The baselines (red) usually occupy the top part of the figure, confirming their effectiveness of uncertainty calibration. Among the DUMs (blue), SNGP and MIR are closer to the region of baselines than others. For realistic shifts (CARLA-C), a significant drop of uncertainty calibration performance can be found for DDU. ("MCD"="MC Dropout")

marginal log-likelihood. This approach to uncertainty estimation allows DCU to be applied across different tasks. However, due to the generative approach to uncertainty estimation it yields poorly calibrated uncertainty.

PostNet (17) learns the distribution of hidden representations end-to-end which allows them to regularize its entropy directly (see Sec. 6.2.2). While this is theoretically guaranteed to lead to high information content in the hidden representations for deterministic models, it leads to some difficulties during training. To ensure stability during training the hidden representations are require to be low dimension. Furthermore, it is required to apply a Batch Normalization layer directly prior to hidden representations which distribution is estimated. While PostNet's approach can be generalized to other predictive distributions, they focus on Dirichlet distributions and thus only classification. PostNet estimates the distribution of hidden representations using one radial flow per class. Uncertainty is estimated using the likelihood of the radial flow with the maximum likelihood. Precisely, they multiply the likelihood with the elements observed for a particular class in the training set which is usually constant in their/our experiments and add one to the result. In accordance with other approaches that use generative modeling of hidden representations for uncertainty estimation, we observe poor calibration of epistemic uncertainty.

6.3 Additional Results

Fig. 5 provides additional visualization of the test accuracy versus calibration performance for the methods compared in Tab. 2 and 3.

6.3.1 Image Classification

OOD Detection. Tab. 5 shows quantitative results on detecting OOD data for DUMs, MC dropout and deep ensembles trained on CIFAR10/100. We observe that DUMs are able to outperform MC dropout and deep ensembles on OOD detection.

Sensitivity to regularization strength. We provide additional ablation studies on the sensitivity to regularization strength for different methods on MNIST (Fig. 6) and FashionMNIST (**??**) These results confirm the findings of the main manuscript, *i.e.* that only MIR and Dropout are sensible to regularization strength, while DUMs based on Lipschitz regularization are not influenced by the regularization strength.

Corruption Severity Analysis. We show the classification accuracy, AUROC and rAULC for each method on the CIFAR10-C (Fig. 11, Fig. 12, Fig. 13) and CIFAR100-C (Fig. 14, Fig. 15, Fig. 16) datasets under different types of corruptions. Each type of corruption is applied at 5 increasing levels of severity. For ease of visualization, we split the 15 different types of corruptions applied on the CIFAR10-C and CIFAR100-C datasets into 3 different figures each. Each figure shows 5 different types of corruptions.

While all methods demonstrate a similar mIoU pattern, DUMs - in particular methods based on generative modeling of hidden representations - yield worse calibration across corruption severities.

	OOD Data	STL10	SVHN	CIFAR100
CIFAR10	MC Dropout (7)	0.686 ± 0.004	0.885 ± 0.002	0.82 ± 0.003
	Ensemble (27)	$\mathbf{0.875 \pm 0.001}$	0.937 ± 0.009	0.758 ± 0.003
	DUQ (13)	0.633 ± 0.008	0.843 ± 0.016	0.766 ± 0.003
	SNGP (15)	0.726 ± 0.007	0.925 ± 0.02	0.861 ± 0.004
	MIR (20)	0.752 ± 0.015	0.916 ± 0.025	0.840 ± 0.007
	DDU (19)	0.737 ± 0.018	0.663 ± 0.073	0.638 ± 0.004
	DCU (42)	0.725 ± 0.027	$\mathbf{0.992 \pm 0.014}$	$\mathbf{0.921 \pm 0.014}$
	OOD Data	STL10	SVHN	CIFAR10
CIFAR100	MC Dropout (7)	0.772 ± 0.004	0.846 ± 0.01	0.735 ± 0.002
	Ensemble (27)	$\mathbf{0.801 \pm 0.014}$	0.741 ± 0.003	0.756 ± 0.007
	DUQ (13)	-	-	-
	SNGP (15)	0.744 ± 0.02	0.795 ± 0.112	0.686 ± 0.007
	MIR (20)	0.789 ± 0.025	0.809 ± 0.031	0.663 ± 0.004
	DDU (19)	0.698 ± 0.021	0.809 ± 0.056	$\mathbf{0.764 \pm 0.019}$
	DCU (42)	0.798 ± 0.019	$\mathbf{0.978 \pm 0.005}$	0.755 ± 0.024

Table 5: OOD detection performance when training on CIFAR10/100 and testing on various other datasets. We report AUROC averaged across 5 independent trainings.

6.3.2 Semantic Segmentation

Examples of segmentation and uncertainty masks. We show qualitative examples of predicted masks, error masks and uncertainty masks for Softmax, MC dropout, SNGP and MIR on semantic segmentation. Fig. 7 illustrates examples under *minimal* distributional shift (*i.e.* Azimuth angle of the sun $= 85^\circ$ and Fig. 8 under *maximal* distributional shift (*i.e.* Azimuth angle of the sun $= -5^\circ$. We show the input image (Input), the segmentation ground truth (GT), the predicted segmentation mask (Prediction), the error mask (Error) and the uncertainty mask (Uncertainty). The error mask is computed as a boolean mask with True values when a pixel is predicted wrongly (yellow) and False (blue) when the prediction is instead correct. The uncertainty mask is preprocessed to facilitate visualization. In particular, we first compute mean μ and standard deviation σ of per-pixel uncertainties over each uncertainty mask. Then, the uncertainty mask is clipped between $[\mu - 2\sigma, \mu + 2\sigma]$. Finally, the uncertainty mask is normalized between 0 and 1 before being visualized.

While the softmax entropy provides decent uncertainty estimates under minimal distributional shift, it tends to be overconfident under severe distributional shift. In particular, Softmax models are only uncertain close to object borders, but they are confident about large portions of the image that are instead predicted wrongly. This can be observed in Fig. 8, where all models tend to predict the entire sky wrongly (assigned to 'building' class), but the Softmax model is the most confident about its predictions of the sky being correct. DUMs and MC dropout do a better job at recognizing wrong predictions under sever domain shift by outputting higher uncertainty values.

Corruption Severity Analysis. We show the mIoU, AUROC and rAULC for each method on the Carla (Fig. 20) and Cityscapes (Fig. 17, Fig. 18, Fig. 19) datasets under different types corruptions. Each type of corruption is applied at 5 increasing levels of severity. For ease of visualization, we split the 15 different types of corruptions applied on the Cityscapes dataset into 3 different figures, showing 5 types of corruptions each.

While all methods demonstrate a similar mIoU pattern, DUMs - in particular methods based on generative modeling of hidden representations - yield worse calibration across corruption severities.

6.4 Training Details

We provide training and optimization details for all evaluated methods. All methods using spectral normalization use 1 power iteration. Hyperparameters were chosen to minimize the validation loss.

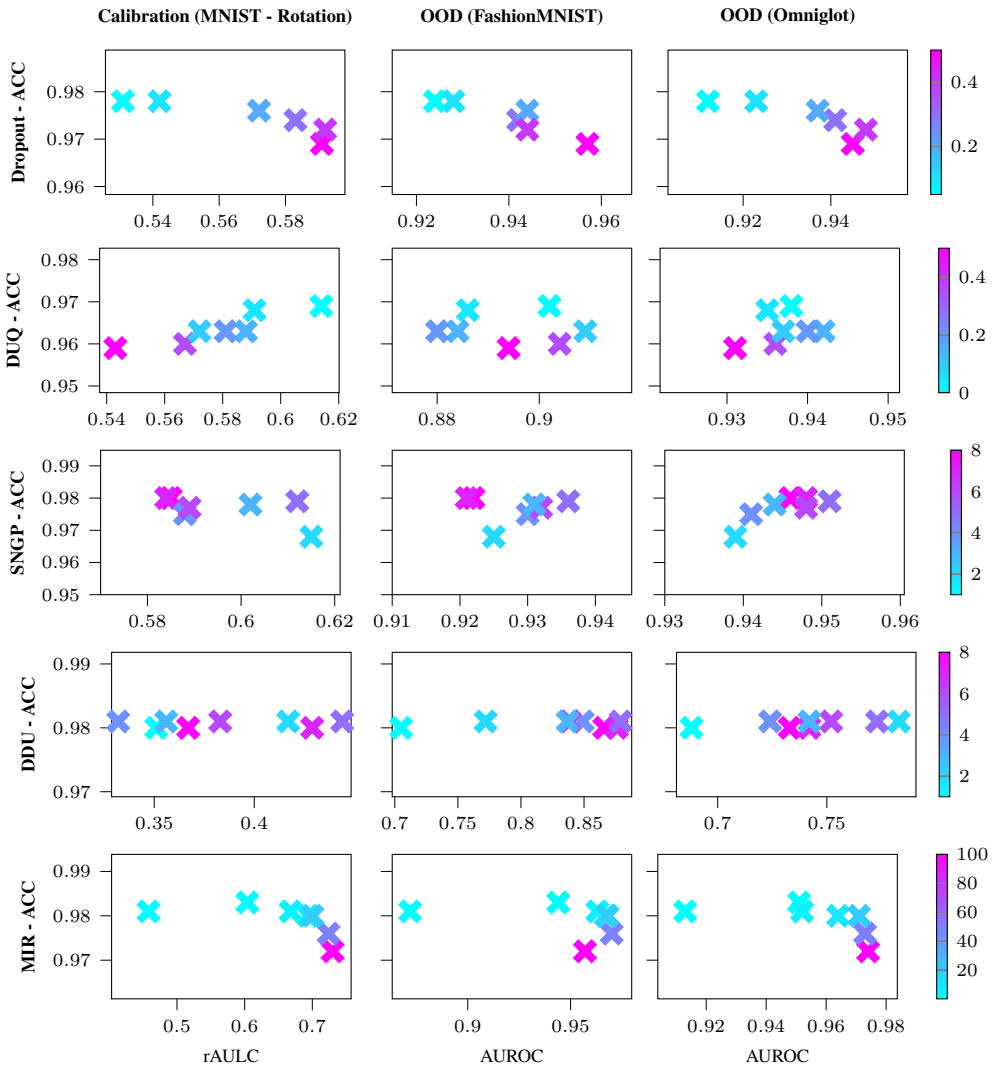


Figure 6: Trained on MNIST. Vertical axis: Test accuracy. Horizontal axis: rAULC (left), AUROC against FashionMNIST (center) and Omniglot (right) for Dropout (1st row), DUQ (2nd row), SNGP (3rd row), DDU (4th row) and MIR (5th row) using different regularization strength. For SNGP a larger hyperparameter corresponds to less regularization. For Dropout and MIR we observe a correlation between regularization strength and performance.

6.4.1 Image Classification - MNIST/FashionMNIST

All methods trained on MNIST/FashionMNIST used a MLP as backbone with 3 hidden layers of 100 dimensions each and ReLU activation functions. We used a batch size of 128 samples and trained for 200 epochs. No data augmentation is performed.

Softmax and Deep ensembles. We used for the single softmax model the Adam optimizer with learning rate 0.003, and L_2 weight regularization 0.0001. When using ensembles, 10 models are trained from different random initializations.

MC dropout. We used for all baselines the Adam optimizer with learning rate 0.003, dropout rate 0.4 and L_2 weight regularization 0.0001.

We found the optimal SN coefficient to be 7, with the GP approximation using 10 (number of classes) inducing points initialized using k-means over 10000 samples.

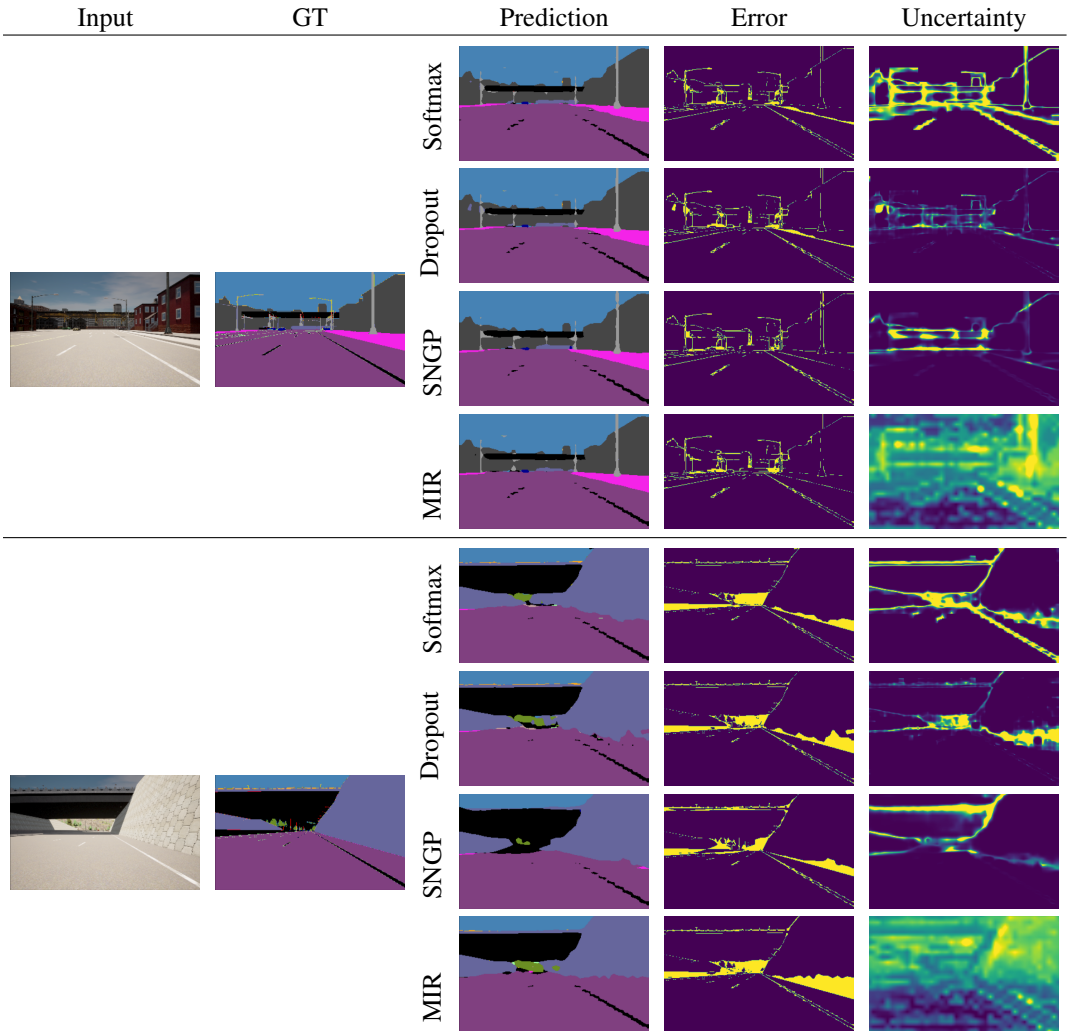


Figure 7: Qualitative comparison of uncertainty from Softmax, MC Dropout, SNGP and MIR under minimal time-of-the-day distribution shift (*i.e.* Azimuth angle of the sun $= 85°$). We show the input image (Input) and the ground truth mask (GT), and we report for each method the predicted segmentation mask (Prediction), the error mask (Error) and the uncertainty mask (Uncertainty).

DUQ We trained DUQ with the SGD optimizer with learning rate 0.01, L_2 weight regularization 0.0001, and a multi-step learning rate decay policy with decay rate 0.3 and decay steps at the epochs 10, 20. Lengthscale for the RBF kernel is 0.1 and optimal gradient penalty loss weight is 0 where we searched along the grid $[0.0, 0.0000001, 0.0000003, 0.000001, 0.000003, 0.00001, 0.00003, 0.0001, 0.0003, 0.001, 0.005, 0.01, 0.025, 0.05, 0.075, 0.1, 0.2, 0.5]$.

DDU. We trained DDU with the Adam optimizer with learning rate 0.001, L_2 weight regularization 0.0001, and a multi-step learning rate decay policy with decay rate 0.2 and decay steps at the epochs 100, 200, 300. We found the optimal SN coefficient to be 6 searching along the grid $[1, 2, 3, 4, 5, 6, 7, 8, 10, 12, 15]$. The GMM is fitted by estimating the empirical mean and covariance matrix of the representations on the training data associated with each class.

SNGP. We trained SNGP with the SGD optimizer with learning rate 0.05, L_2 weight regularization 0.0003, and a multi-step learning rate decay policy with decay rate 0.2 and decay steps at the epochs 60, 120, 160. We found the optimal SN coefficient to be 6 searching along the grid $[1, 2, 3, 4, 5, 6, 7, 8, 10, 12, 15]$., with the GP approximation using 10 hidden dimensions, lengthscale 2 and mean field factor 30.

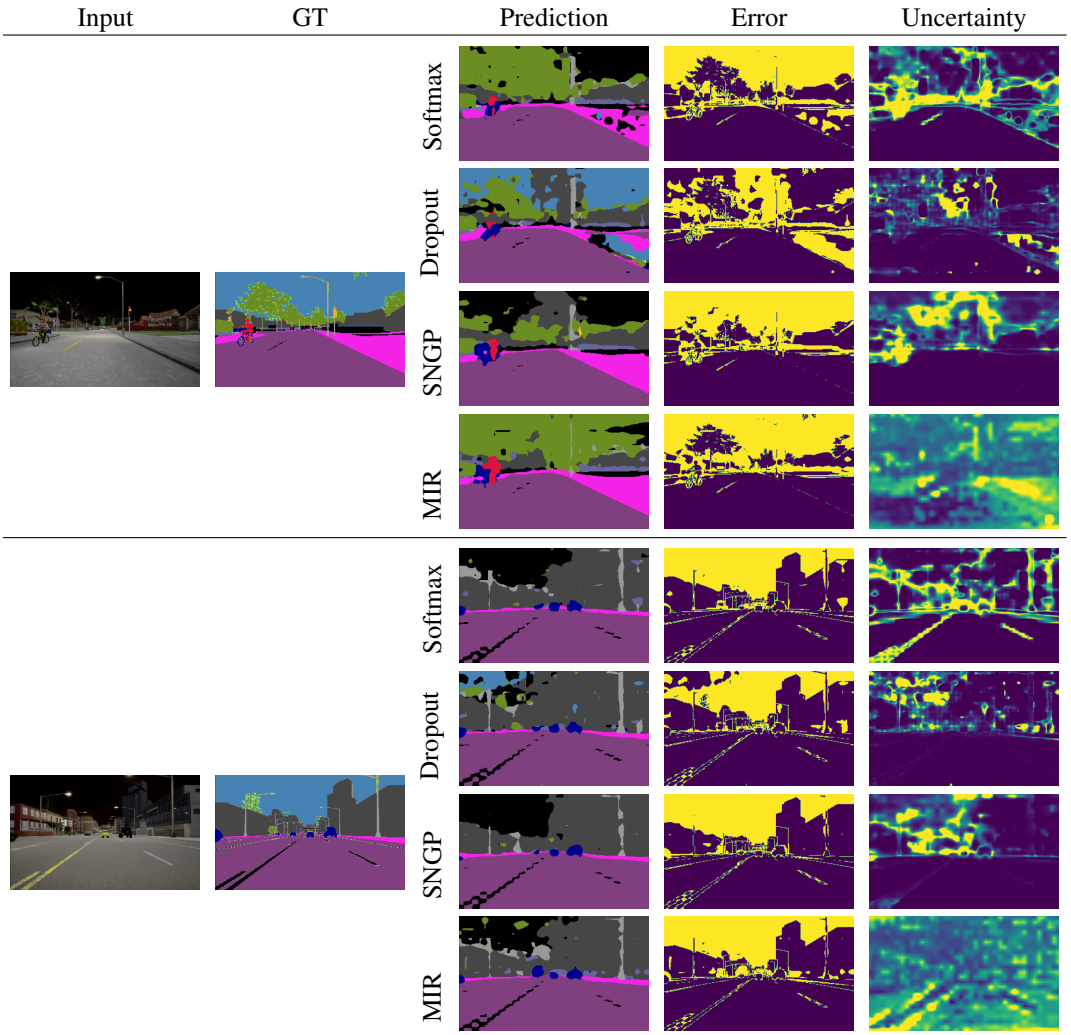


Figure 8: Qualitative comparison of uncertainty from Softmax, MC Dropout, SNGP and MIR under maximal time-of-the-day distribution shift (*i.e.* Azimuth angle of the sun $= -5°$). We show the input image (Input) and the ground truth mask (GT), and we report for each method the predicted segmentation mask (Prediction), the error mask (Error) and the uncertainty mask (Uncertainty).

MIR. We trained MIR with the Adam optimizer with learning rate 0.001, and L_2 weight regularization 0.0001. We found the optimal reconstruction loss weight to be 1 after searching along the grid $[0.0, 0.1, 0.5, 1.0, 2.0, 5.0, 10.0, 20.0, 50.0, 100.0]$.

6.4.2 Image Classification - CIFAR10/SVHN

When training on CIFAR-10/SVHN, we use a ResNet-18 (72) as backbone. The dimensionality of the last feature space encoded with the ResNet backbone is 100 for all methods. We used a batch size of 128 samples and trained for 400 epochs. The training set is augmented with common data augmentation techniques. We apply random horizontal flips, random brightness augmentation with maximum delta 0.2 and random contrast adjustment with multiplier lower bound 0.8 and upper bound 1.2.

Softmax and Deep ensembles. We used for the single softmax model the Adam optimizer with learning rate 0.003, L_2 weight regularization 0.0001, and a multi-step learning rate decay policy with decay rate 0.2 and decay steps at the epochs 250, 300, 400. When using ensembles, 10 models are trained from different random initializations.

MC dropout. We used for all baselines the Adam optimizer with learning rate 0.003, dropout rate 0.3, L_2 weight regularization 0.0001, and a multi-step learning rate decay policy with decay rate 0.2 and decay steps at the epochs 250, 300, 400.

DUE We trained DUE with the SGD optimizer with learning rate 0.01, L_2 weight regularization 0.0005, dropout rate 0.1, and a multi-step learning rate decay policy with decay rate 0.2 and decay steps at the epochs 100, 200, 300. We found the optimal SN coefficient to be 7 for SVHN and 9 for CIFAR-10, with the GP approximation using 10 (number of classes) inducing points initialized using k-means over 10000 samples.

DUQ We trained DUE with the SGD optimizer with learning rate 0.01, L_2 weight regularization 0.0001, dropout rate 0.1, and a multi-step learning rate decay policy with decay rate 0.3 and decay steps at the epochs 200, 250. 300. Lengthscale for the RBF kernel is 0.1 and optimal gradient penalty loss weight is 0

DDU. We trained DDU with the Adam optimizer with learning rate 0.001, L_2 weight regularization 0.0001, dropout rate 0.3, and a multi-step learning rate decay policy with decay rate 0.2 and decay steps at the epochs 80, 120, 180. We found the optimal SN coefficient to be 7. The GMM fit on top of the pretrained feature extractor is trained for 100 epochs and is fit with 64 batches.

SNGP. We trained SNGP with the SGD optimizer with learning rate 0.05, L_2 weight regularization 0.0004, dropout rate 0.1, and a multi-step learning rate decay policy with decay rate 0.2 and decay steps at the epochs 100, 200, 300. We found the optimal SN coefficient to be 7, with the GP approximation using 10 hidden dimensions, lengthscale 2 and mean field factor 30.

MIR. We trained MIR with the Adam optimizer with learning rate 0.003, L_2 weight regularization 0.0001, dropout rate 0.1, and a multi-step learning rate decay policy with decay rate 0.2 and decay steps at the epochs 150, 200, 250, 300. We found the optimal reconstruction loss weight to be 1.

6.4.3 Semantic Segmentation.

When training on semantic segmentation, we use a DRN (75; 76) (DRN-A-50) as backbone. We used a batch size of 4 samples and trained for 200 epochs. Images are rescaled to size 400×640 The training set is augmented with common data augmentation techniques. All training samples are augmented with random cropping with factor 0.8. We apply random horizontal flips, random brightness augmentation with maximum delta 0.2 and random contrast adjustment with multiplier lower bound 0.8 and upper bound 1.2.

Softmax. We used for the single softmax model the Adam optimizer with learning rate 0.0004, L_2 weight regularization 0.0001, and a multi-step learning rate decay policy with decay rate 0.3 and decay steps at the epochs 30, 60, 90, 120.

MC dropout. We used for all baselines the Adam optimizer with learning rate 0.0004, dropout rate 0.4, L_2 weight regularization 0.0001, and a multi-step learning rate decay policy with decay rate 0.3 and decay steps at the epochs 30, 60, 90, 120.

SNGP. We trained SNGP with the SGD optimizer with learning rate 0.0002, L_2 weight regularization 0.0003, dropout rate 0.1, and a multi-step learning rate decay policy with decay rate 0.2 and decay steps at the epochs 20, 40, 60, 80, 100. We found the optimal SN coefficient to be 6, with the GP approximation using 128 hidden dimensions, lengthscale 2 and mean field factor 25.

MIR. We trained MIR with the Adam optimizer with learning rate 0.0002, L_2 weight regularization 0.0001, dropout rate 0.1, and a multi-step learning rate decay policy with decay rate 0.3 and decay steps at the epochs 30, 60, 90, 120. We found the optimal reconstruction loss weight to be 1.

6.5 Implementation Details.

All methods were re-implemented in Tensorflow 2.0. We payed attention to all the details reported in each paper and we run all experiments for each method multiple times to account for stochasticity, *i.e.* 5 times for classification and 3 times for segmentation. When an implementation was publicly available, we relied on it. This is the case for DUQ (`https://github.com/y0ast/deterministic-uncertainty-quantification`), SNGP

(https://github.com/google/uncertainty-baselines/blob/master/baselines/imagenet/sngp.py) and DUE (https://github.com/y0ast/DUE).

SNGP. We follow the publicly available implementation of SNGP, which, compared to the implementation described in the original paper, proposes to further reduce the computational overhead of the GP approximation by replacing the Monte-Carlo averaging with the mean-field approximation (79). This is especially relevant in large-scale tasks like semantic segmentation, were it is important to reduce the computational overload.

6.5.1 IMAGE CLASSIFICATION

DUE. Note that only DUE uses a SN approximation also for the batch normalization layer. All other methods only restrict the Lipschitz constant of convolutional and fully connected layers.

MIR only differs from regular softmax models in its decoder module used for the reconstruction regularization loss (20). When training MLP architectures the decoder is comprised of two fully-connected layer. The first has a ReLU activation function and 200 output neruons. The second has a linear activation function and its output dimensionality equals that of the models' input data. When training convolutional neural networks the decoder is comprised of four blocks of transpose convolutions, batch normalization layers and ReLU activation functions that gradually upscale the hidden representations to the dimensionality of the input data. These four blocks are followed by a 1x1 convolution with linear activation function.

6.5.2 SEMANTIC SEGMENTATION

MIR. Similar to image classification, MIR only differs from regular segmentation models in its decoder module used for the reconstruction regularization loss (20). The decoder module is comprised of a single point-wise feed forward layer that maps the hidden representations $\mathbf{z} \in \mathbb{R}^{W_z \times H_z \times C_z}$ to $\mathbf{z} \in \mathbb{R}^{W_z \times H_z \times 3}$. Subsequently, the result is bilinearly upsampled to the image resolution on which we compute the reconstruction loss.

6.5.3 UNCERTAINTY DERIVATION.

We provide details on the estimation of uncertainty for the baseline methods. For details on the uncertainty derivation in DUMs, please refer to Sec. 3 of the main paper or to the original paper of each amalysed method.

Softmax. In case of the softmax baseline we estimate uncertainty using the entropy of the predictive distribution parameterized by the neural network. Given an input $\mathbf{x}$ the entropy H is given by $H(\mathbf{y}|\mathbf{x}) = \sum -p(\mathbf{y}|\mathbf{x}) \log(p(\mathbf{y}|\mathbf{x}))$ where $p(\mathbf{y}|\mathbf{x})$ are the softmax probabilities.

MC dropout and deep ensembles. We following (80) and compute epistemic uncertainty as the conditional mutual information between the weights $\mathbf{w}$ and the predictions $\hat{y}$ given the input $\mathbf{x}$. Given an input $\mathbf{x}$ and a set of weights $\mathbf{w}$ we observe the predictive distribution $p(\hat{y}|x, w)$. Then epistemic uncertainty u_{ep} is calculated by approximating the mutual information conditioned on the input $\mathbf{x}$:

$$\begin{aligned} u_{ep} &= I\left(\hat{y}, w|x\right) \\ &= H(\hat{y}|x) - H(\hat{y}|w, x) \\ &= E_{y\sim p(\hat{y}|x)}\left[-\log(p(\hat{y}|x))\right] - u_{al} \end{aligned}$$

where u_{al} denotes the aleatoric uncertainty. Here, $p(\hat{y}|x) = \int \mathrm{d}w p(w) p(\hat{y}|x, w)$ is evaluated using a finite set of samples/ensemble members.

6.5.4 UNCERTAINTY DERIVATION FOR SEMANTIC SEGMENTATION.

We derive uncertainty estimates for each method for semantic segmentation. We average pixel-level uncertainties under the assumption that all pixels are represented by i.i.d. variables.

Uncertainty. In our experiments on continuous distributional shifts we want to estimate pixel-level uncertainty for the output map.

MIR estimates epistemic uncertainty using the likelihood of hidden representations $\mathbf{z} \in \mathbb{R}^{W_z \times H_z \times C_z}$. Since $\mathbf{z}$ is high-dimensional in our experiments, we assume that it factorizes along W_z and H_z and

is translation invariant. Formally, $p(\mathbf{z}) = \prod_i^{W_z} \prod_j^{H_z} p_\theta(\mathbf{z}_{ij})$ where $\mathbf{z}_{ij} \in \mathbb{R}^{W_z \times H_z}$ and θ is shared across W_z and H_z.

We parameterize p_θ with a GMM with $n = 10$ components where each component has a full covariance matrix. We fit the GMM on 100000 hidden representations ($\mathbf{z}_{ij} \in \mathbb{R}^{C_z}$) randomly picked from the training dataset post-training. Since $C_z = 1024$ is still high-dimensional, we first apply PCA to reduce its dimensionality to 32.

In the dilated resnet architecture used for semantic segmentation the latent representation $\mathbf{z}$ is passed through a point-wise feedforward layer $f : \mathbb{R}^{W_z \times H_z \times C_z} \mapsto \mathbb{R}^{W_z \times H_z \times 3}$ and, subsequently, bilinearly upsampled to image resolution ($\mathbb{R}^{W \times H \times K}$) where K is the number of classes. We could estimate the global, *i.e.*image-level, uncertainty of an input, by providing the negative log-likelihood of the factorizing distribution. However, in order to also obtain pixel-wise uncertainties using MIR, we first compute the negative log-likelihood (*i.e.*epistemic uncertainty) associated with each latent representation $\mathbf{z}_{ij}$. Then, we bilinearly upsample the negative log-likelihoods and use the result as proxy for pixel-wise epistemic uncertainty. If we wanted to obtain a global, *i.e.*image-level, uncertainty we could average pixel-level uncertainties.

6.6 DATASET

To benchmark our model on data with realistically and continuously changing environment, we collect a synthetic dataset for semantic segmentation. We use the CARLA Simulator (64) for rendering the images and segmentation masks. The classes definition is aligned with the CityScape dataset (73). In order to obtain a fair comparison, all the OOD data are sampled with the same trajectory and the environmental objects, except for the time-of-the-day or weather parameters.

In-domain data The data is collected from 4 towns in CARLA. We produce 32 sequences from each town. The distribution of the vehicles and pedestrians are randomly generated for each sequence. Every sequence has has 500 frames with a sampling rate of 10 FPS. From them we randomly sample the training and validation set.

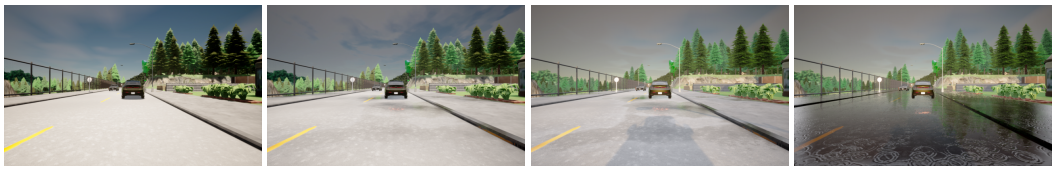
Out-of-domain data Here, we consider the time-of-the-day and the rain strength as the parameters for the continuous changing environment. In practice, these two parameters have major influence for autonomous driving tasks.

The change of the time-of-the-day is illustrated in Fig. 9. The time-of-the-day is parametrized by the Sun's altitude angle, where 90° means the mid-day and the 0° means the dust or dawn. Here, we produce samples with the altitude angle changes from 90° to 15° by step of 5°, and 15° to -5° by step of 1° where the environment changes shapely. From these examples, we can confirm that the change of time-of-the-day leads to the major change in the lightness, color and visibility of the sky, roads and the buildings nearby. The effect of rain strength is demonstrated in Fig. 10. Here the cloudiness and, ground wetness and ground reflection are the main changing parameters.

(a) Mid-day ($90°$) (b) Afternoon ($45°$) (c) Afternoon ($15°$) (d) Evening ($10°$)

(e) Dust ($2°$) (f) Sunset ($0°$) (g) Sunset ($-2°$) (h) Night ($\leq -5°$)

Figure 9: Changing of the time-of-the-day



(a) Clear (b) Cloudy (c) Small rain (d) Heavy rain

Figure 10: Changing of the weather

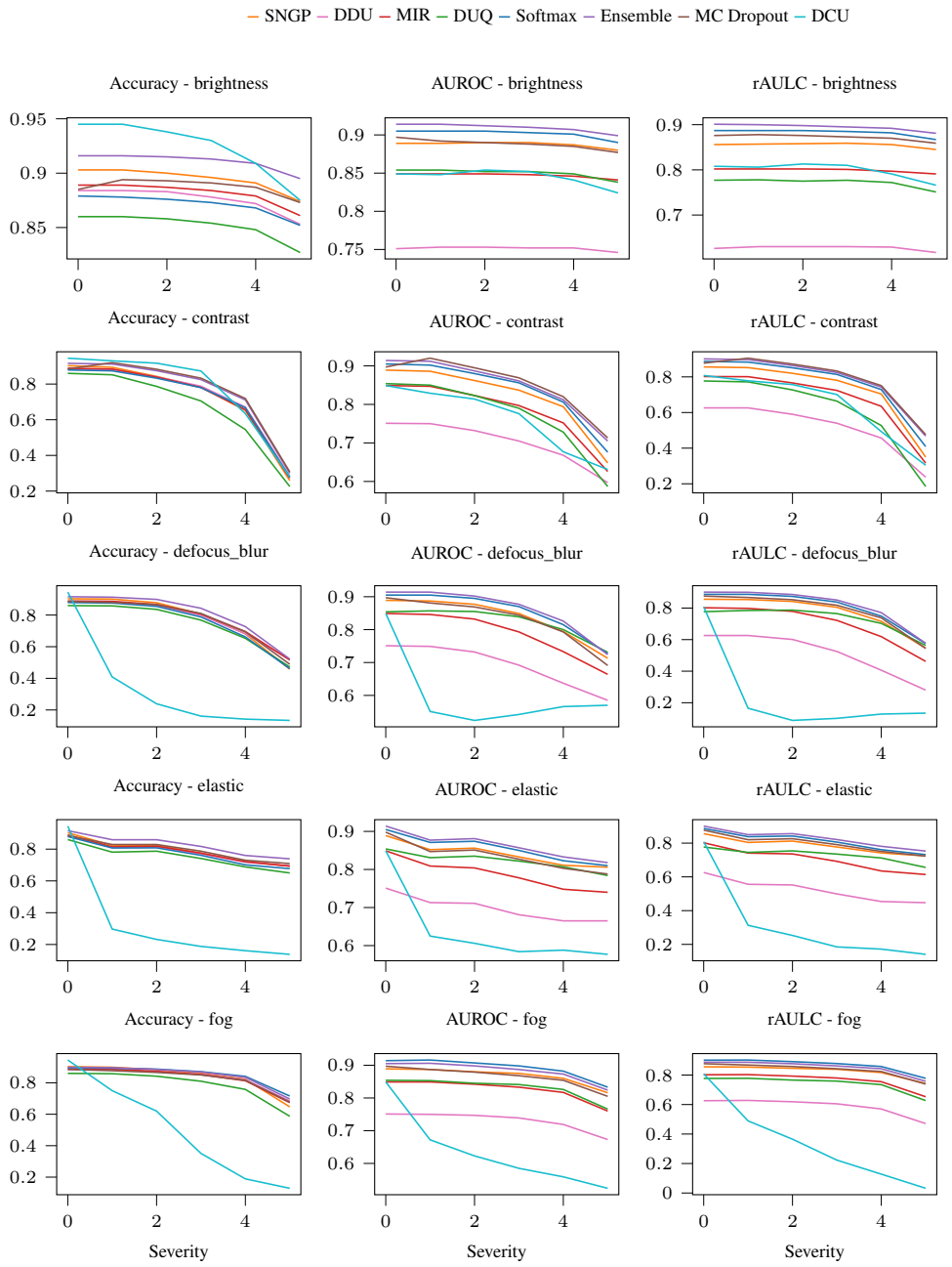


Figure 11: We here compare the performance of DUMs and of the baselines under different corruption types and severities applied on the CIFAR10-C dataset. We show the accuracy, AUROC and rAULC (vertical axis) for each method depending on the corruption severity (horizontal axis) of the following corruption types (listed from top to bottom): brightness, contrast, defocus blur, elastic, fog.

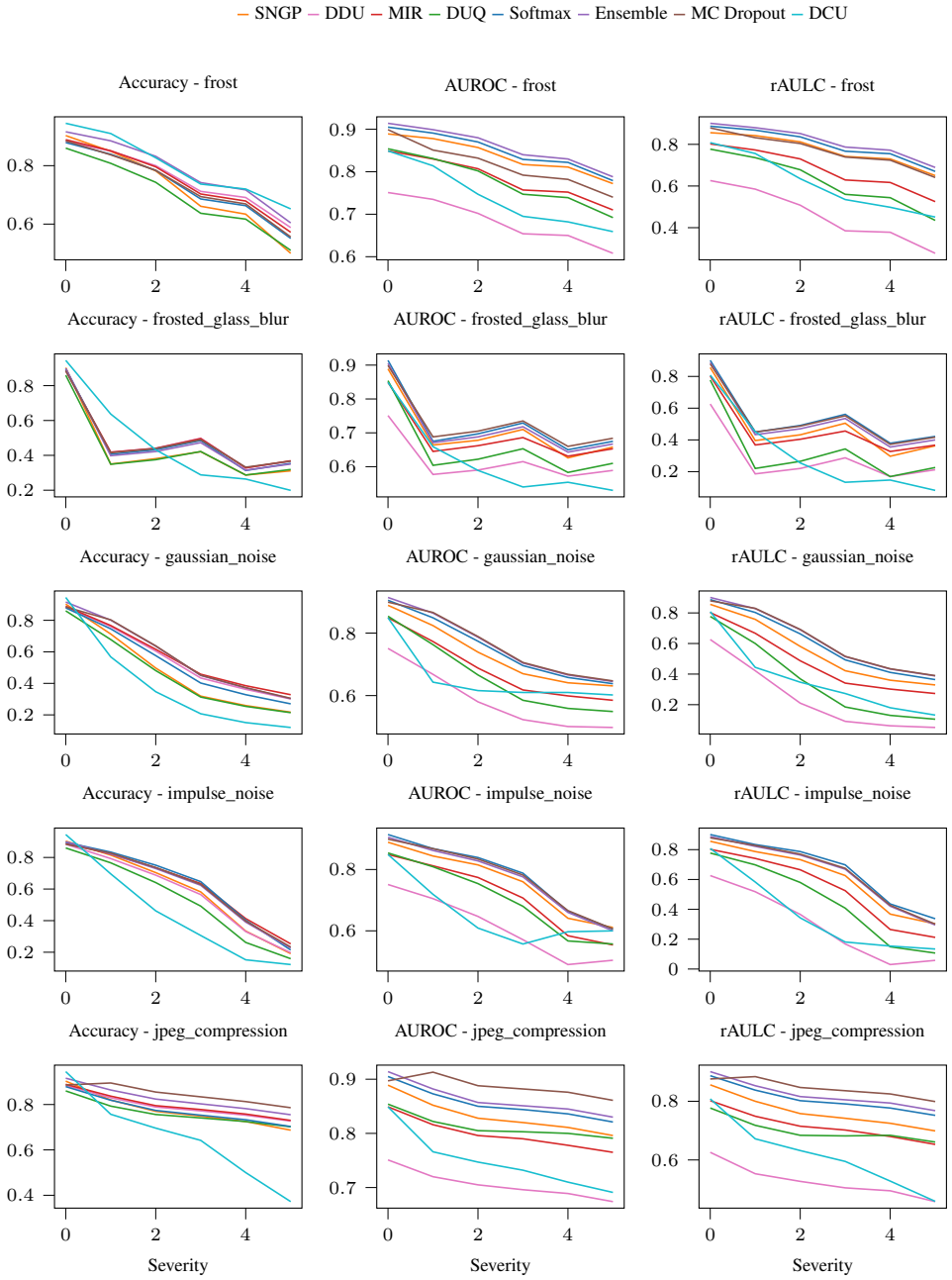


Figure 12: We here compare the performance of DUMs and of the baselines under different corruption types and severities applied on the CIFAR10-C dataset. We show the accuracy, AUROC and rAULC (vertical axis) for each method depending on the corruption severity (horizontal axis) of the following corruption types (listed from top to bottom): frost, frosted glass blur, gaussian noise, impulse noise, jpeg compression.

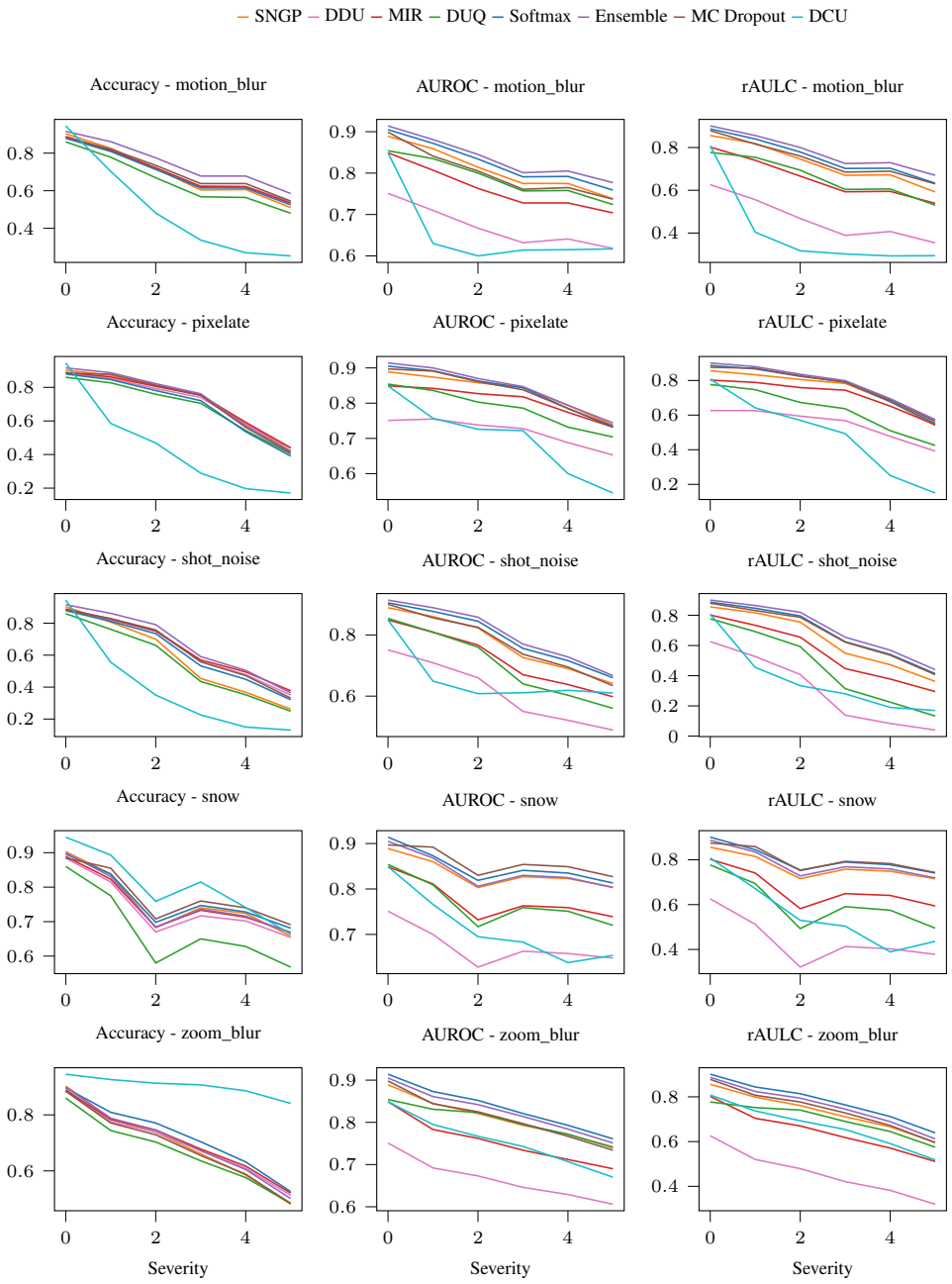


Figure 13: We here compare the performance of DUMs and of the baselines under different corruption types and severities applied on the CIFAR10-C dataset. We show the accuracy, AUROC and rAULC (vertical axis) for each method depending on the corruption severity (horizontal axis) of the following corruption types (listed from top to bottom): motion blur, pixelate, shot noise, snow, zoom blur.

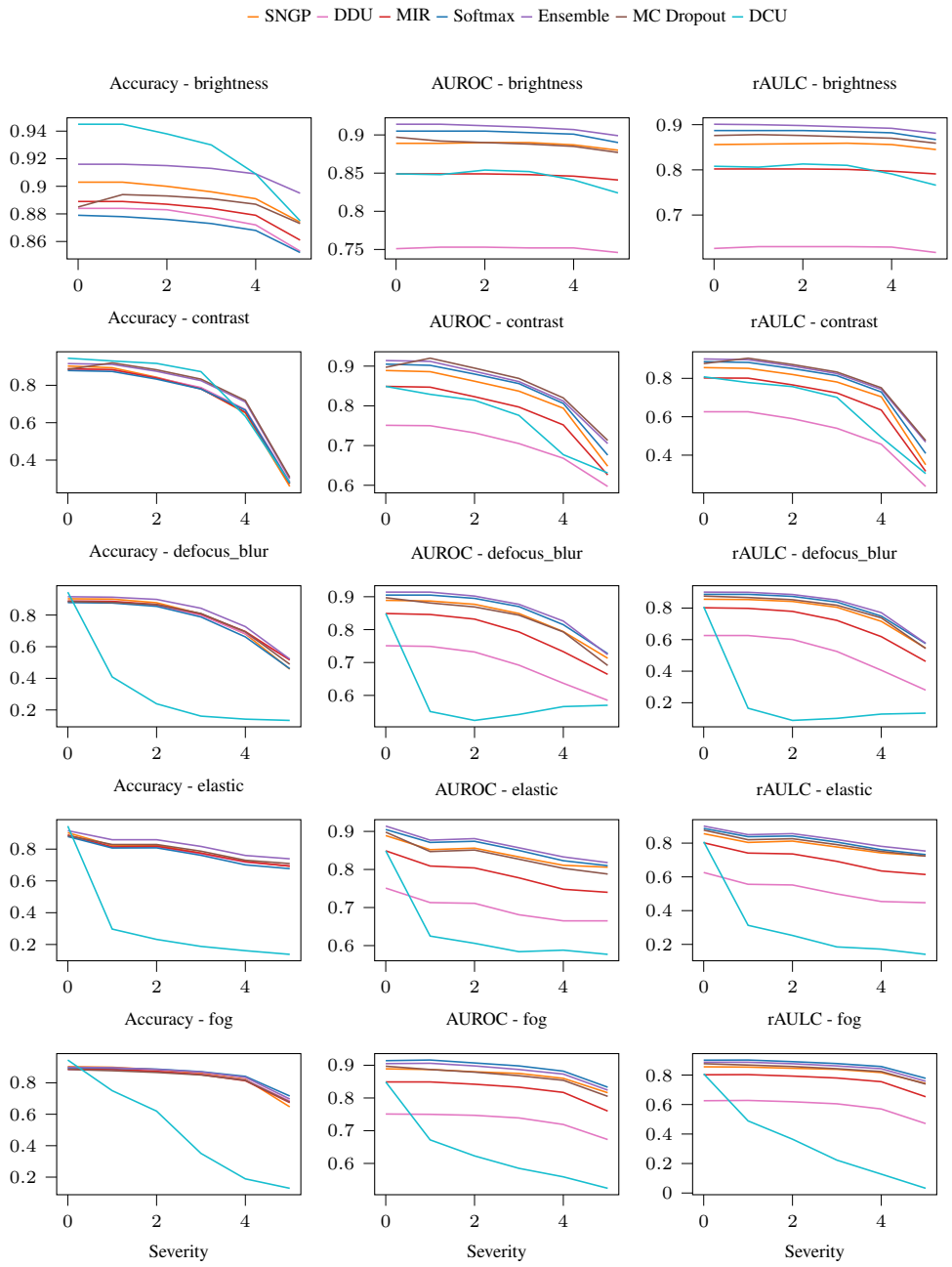


Figure 14: We here compare the performance of DUMs and of the baselines under different corruption types and severities applied on the CIFAR100-C dataset. We show the accuracy, AUROC and rAULC (vertical axis) for each method depending on the corruption severity (horizontal axis) of the following corruption types (listed from top to bottom): brightness, contrast, defocus blur, elastic, fog.

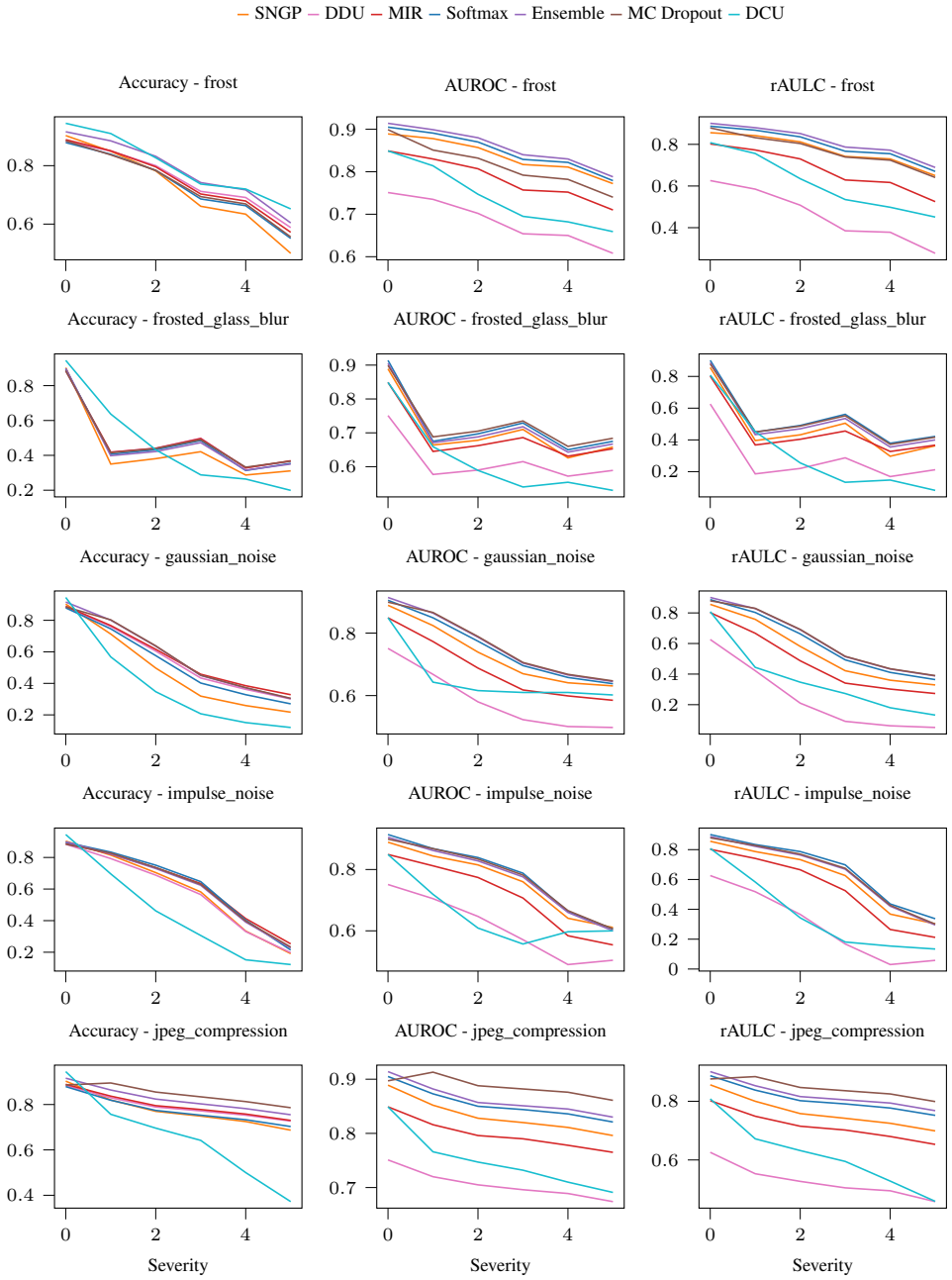


Figure 15: We here compare the performance of DUMs and of the baselines under different corruption types and severities applied on the CIFAR100-C dataset. We show the accuracy, AUROC and rAULC (vertical axis) for each method depending on the corruption severity (horizontal axis) of the following corruption types (listed from top to bottom): frost, frosted glass blur, gaussian noise, impulse noise, jpeg compression.

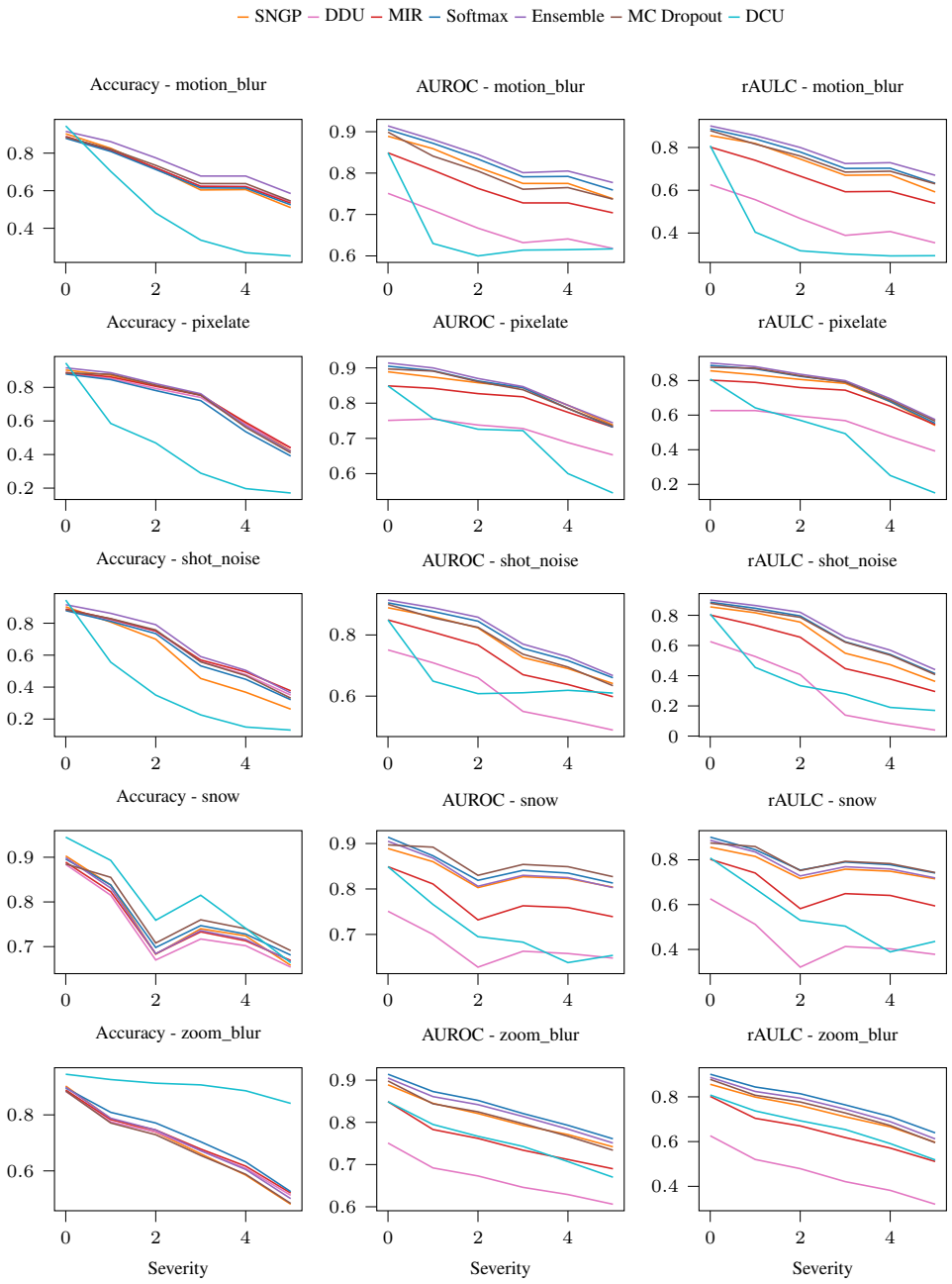


Figure 16: We here compare the performance of DUMs and of the baselines under different corruption types and severities applied on the CIFAR100-C dataset. We show the accuracy, AUROC and rAULC (vertical axis) for each method depending on the corruption severity (horizontal axis) of the following corruption types (listed from top to bottom): motion blur, pixelate, shot noise, snow, zoom blur.

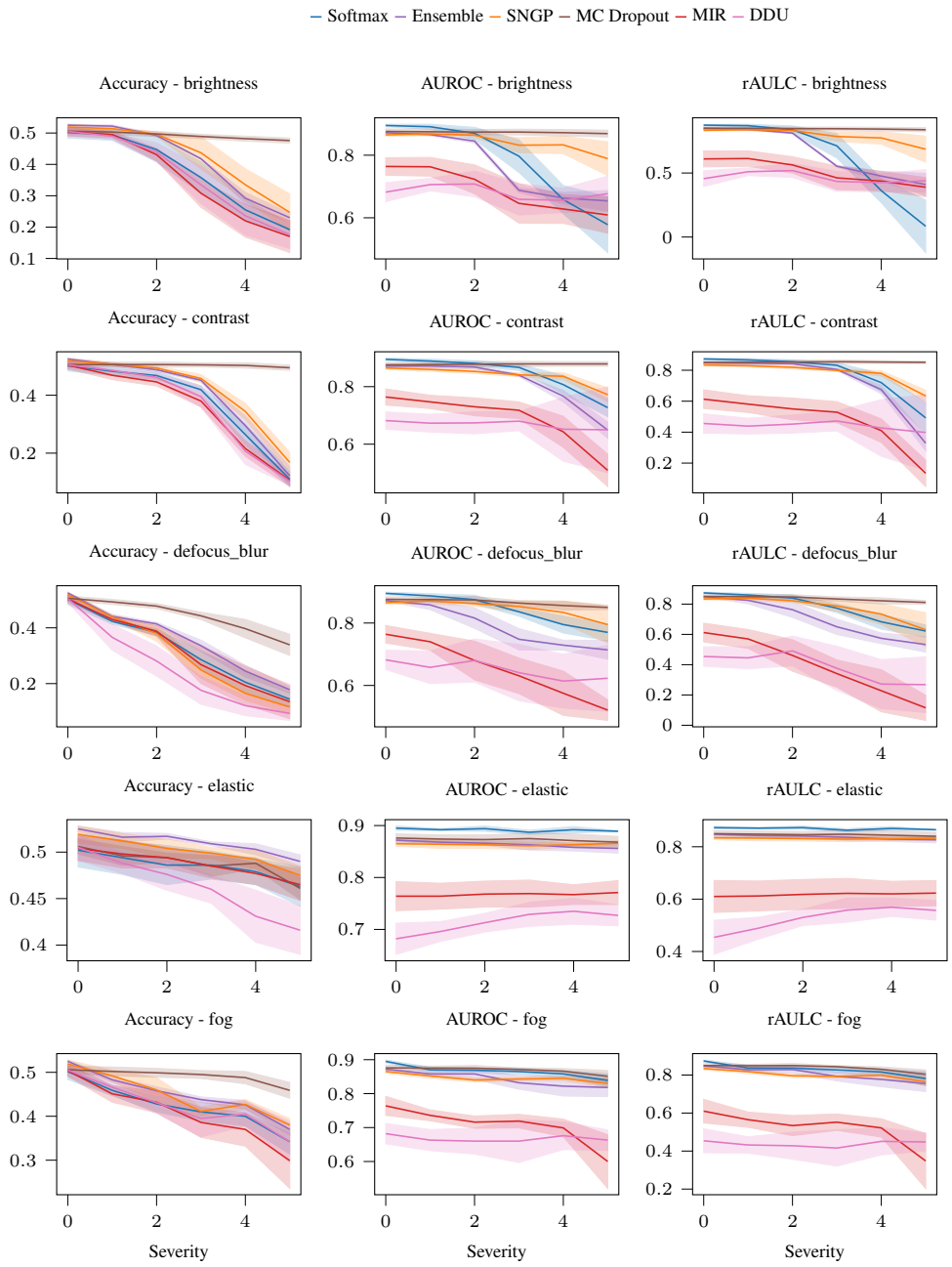


Figure 17: We here compare the performance of DUMs and of the baselines under different corruption types and severities applied on the CIFAR100-C dataset. We show the mIoU, AUROC and rAULC (vertical axis) for each method depending on the corruption severity (horizontal axis) of the following corruption types (listed from top to bottom): brightness, contrast, defocus blur, elastic, fog.

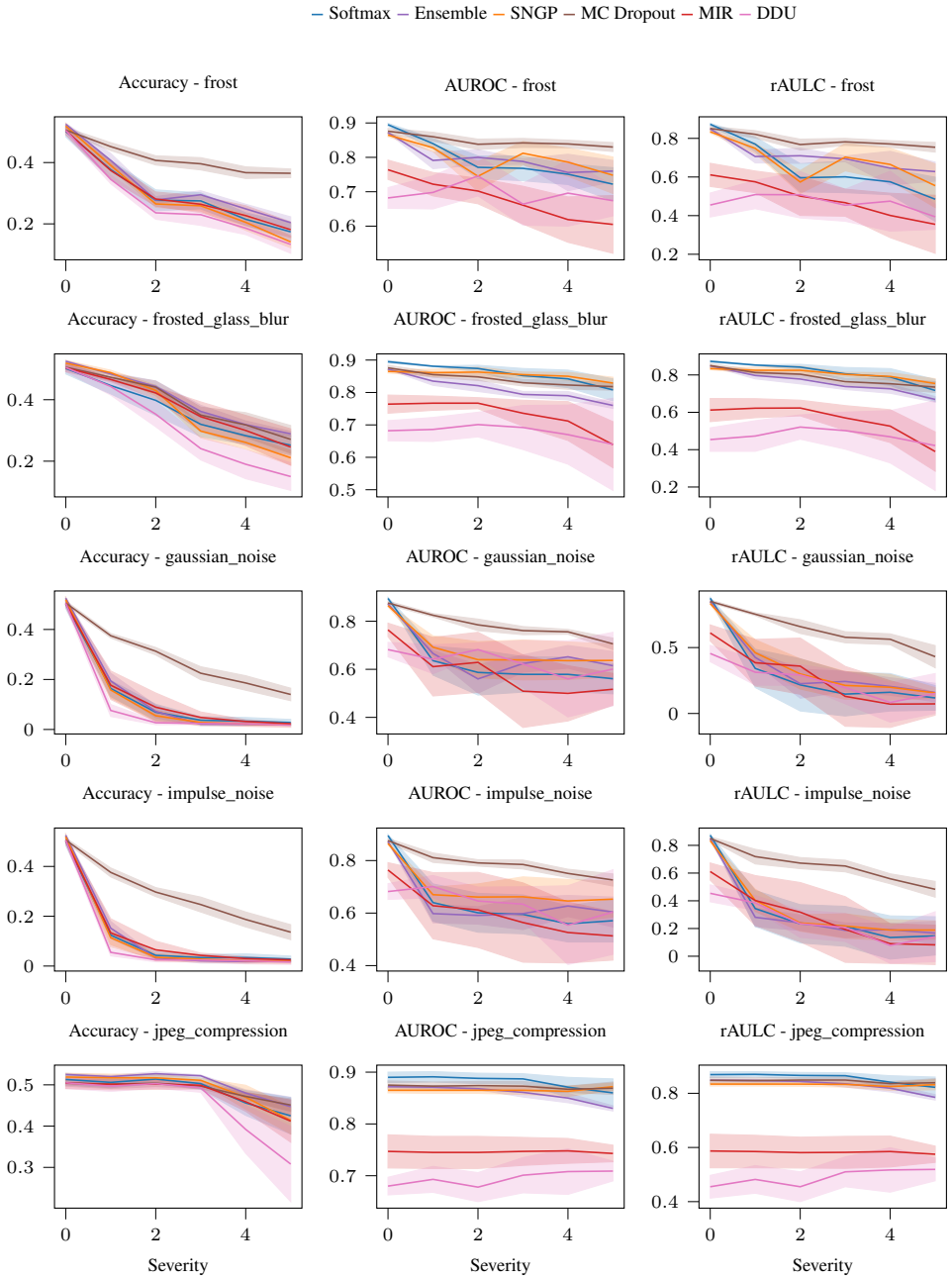


Figure 18: We here compare the performance of DUMs and of the baselines under different corruption types and severities applied on the CIFAR100-C dataset. We show the mIoU, AUROC and rAULC (vertical axis) for each method depending on the corruption severity (horizontal axis) of the following corruption types (listed from top to bottom): frost, frosted glass blur, gaussian noise, impulse noise, jpeg compression.

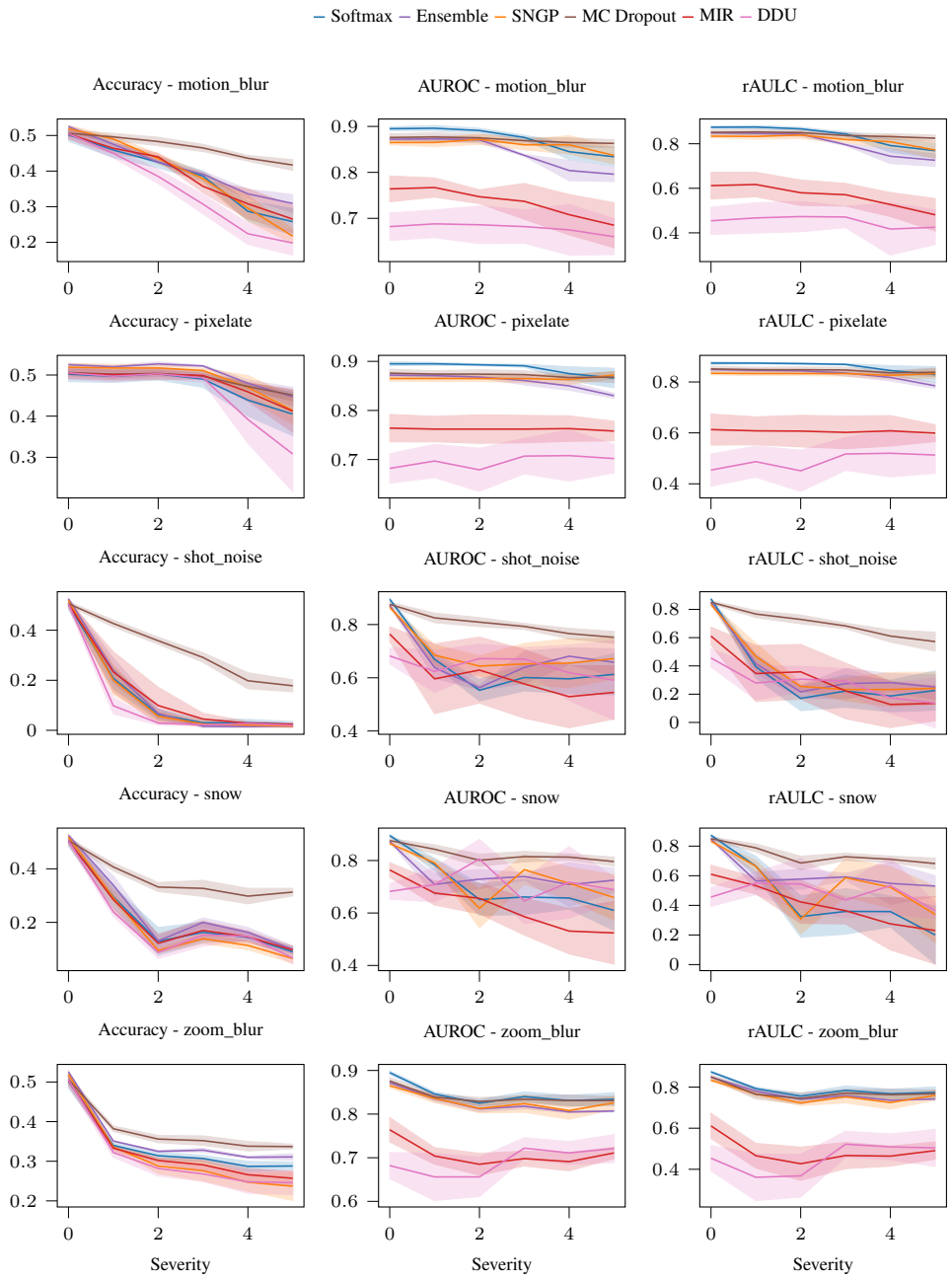


Figure 19: We here compare the performance of DUMs and of the baselines under different corruption types and severities applied on the Cityscapes-C dataset. We show the mIoU, AUROC and rAULC (vertical axis) for each method depending on the corruption severity (horizontal axis) of the following corruption types (listed from top to bottom): motion blur, pixelate, shot noise, snow, zoom blur.

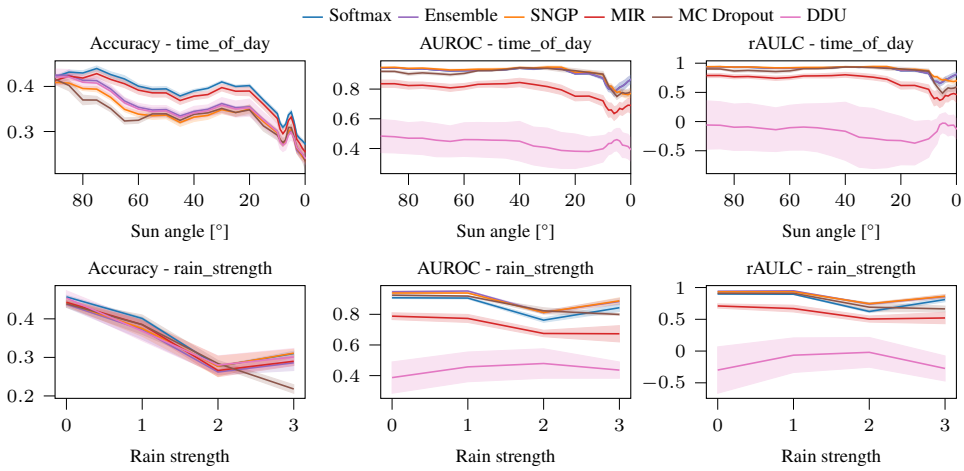


Figure 20: We here compare the performance of DUMs and of the baselines under different corruption types and severities applied on the Carla-C dataset. We show the mIoU, AUROC and rAULC (vertical axis) for each method depending on the corruption severity (horizontal axis) of the following corruption types (listed from top to bottom): time of day, rain.

REFERENCES

[1] Gilles Blanchard, Gyemin Lee, and Clayton Scott. Generalizing from several related classification tasks to a new unlabeled sample. *Advances in neural information processing systems*, 24:2178–2186, 2011.

[2] Krikamol Muandet, David Balduzzi, and Bernhard Schölkopf. Domain generalization via invariant feature representation. In *International Conference on Machine Learning*, pages 10–18. PMLR, 2013.

[3] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[4] Geoffrey E Hinton and Drew Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 5–13, 1993.

[5] Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

[6] Durk P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In *Advances in neural information processing systems*, pages 2575–2583, 2015.

[7] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016.

[8] Guodong Zhang, Shengyang Sun, David Duvenaud, and Roger Grosse. Noisy natural gradient as variational inference. In *International Conference on Machine Learning*, pages 5852–5861. PMLR, 2018.

[9] Janis Postels, Francesco Ferroni, Huseyin Coskun, Nassir Navab, and Federico Tombari. Sampling-free epistemic uncertainty estimation using approximated variance propagation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2931–2940, 2019.

[10] Antonio Loquercio, Mattia Segu, and Davide Scaramuzza. A general framework for uncertainty estimation in deep learning. *IEEE Robotics and Automation Letters*, 5(2):3153–3160, 2020.

[11] Florian Wenzel, Kevin Roth, Bastiaan 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 *International Conference on Machine Learning*, pages 10248–10259. PMLR, 2020.

[12] Amit Mandelbaum and Daphna Weinshall. Distance-based confidence score for neural network classifiers. *arXiv preprint arXiv:1709.09844*, 2017.

[13] Joost Van Amersfoort, Lewis Smith, Yee Whye Teh, and Yarin Gal. Uncertainty estimation using a single deep deterministic neural network. In *International Conference on Machine Learning*, pages 9690–9700. PMLR, 2020.

[14] Alexander A Alemi, Ian Fischer, and Joshua V Dillon. Uncertainty in the variational information bottleneck. *arXiv preprint arXiv:1807.00906*, 2018.

[15] Jeremiah Zhe 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. *Conference on Neural Information Processing Systems*, 2020.

[16] Mike Wu and Noah Goodman. A simple framework for uncertainty in contrastive learning. *arXiv preprint arXiv:2010.02038*, 2020.

[17] Bertrand Charpentier, Daniel Zügner, and Stephan Günnemann. Posterior network: Uncertainty estimation without ood samples via density-based pseudo-counts. *Advances in Neural Information Processing Systems*, 33:1356–1367, 2020.

[18] Joost van Amersfoort, Lewis Smith, Andrew Jesson, Oscar Key, and Yarin Gal. Improving deterministic uncertainty estimation in deep learning for classification and regression. *arXiv preprint arXiv:2102.11409*, 2021.

[19] Jishnu Mukhoti, Andreas Kirsch, Joost van Amersfoort, Philip HS Torr, and Yarin Gal. Deterministic neural networks with appropriate inductive biases capture epistemic and aleatoric uncertainty. *arXiv preprint arXiv:2102.11582*, 2021.

[20] Janis Postels, Hermann Blum, Yannick Strümpler, Cesar Cadena, Roland Siegwart, Luc Van Gool, and Federico Tombari. The hidden uncertainty in a neural networks activations. *arXiv preprint arXiv:2012.03082*, 2020.

[21] Bertrand Charpentier, Oliver Borchert, Daniel Zügner, Simon Geisler, and Stephan Günnemann. Natural posterior network: Deep bayesian predictive uncertainty for exponential family distributions. *arXiv preprint arXiv:2105.04471*, 2021.

[22] Armen Der Kiureghian and Ove Ditlevsen. Aleatory or epistemic? does it matter? *Structural safety*, 31(2):105–112, 2009.

[23] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 5580–5590, 2017.

[24] Jasper Snoek, Yaniv Ovadia, Emily Fertig, Balaji Lakshminarayanan, Sebastian Nowozin, D Sculley, Joshua Dillon, Jie Ren, and Zachary Nado. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems*, pages 13969–13980, 2019.

[25] Fredrik K Gustafsson, Martin Danelljan, and Thomas B Schon. Evaluating scalable bayesian deep learning methods for robust computer vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 318–319, 2020.

[26] Radford M Neal et al. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011.

[27] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems*, pages 6402–6413, 2017.

[28] Yeming Wen, Dustin Tran, and Jimmy Ba. Batchensemble: an alternative approach to efficient ensemble and lifelong learning. *arXiv preprint arXiv:2002.06715*, 2020.

[29] Michael Dusenberry, Ghassen Jerfel, Yeming Wen, Yian Ma, Jasper Snoek, Katherine Heller, Balaji Lakshminarayanan, and Dustin Tran. Efficient and scalable bayesian neural nets with rank-1 factors. In *International conference on machine learning*, pages 2782–2792. PMLR, 2020.

[30] Mattias Teye, Hossein Azizpour, and Kevin Smith. Bayesian uncertainty estimation for batch normalized deep networks. In *International Conference on Machine Learning*, pages 4907–4916, 2018.

[31] Ian Osband, Zheng Wen, Mohammad Asghari, Morteza Ibrahimi, Xiyuan Lu, and Benjamin Van Roy. Epistemic neural networks. *arXiv preprint arXiv:2107.08924*, 2021.

[32] Hippolyt Ritter, Aleksandar Botev, and David Barber. A scalable laplace approximation for neural networks. In *6th International Conference on Learning Representations, ICLR 2018-Conference Track Proceedings*, volume 6. International Conference on Representation Learning, 2018.

[33] Jongseok Lee, Matthias Humt, Jianxiang Feng, and Rudolph Triebel. Estimating model uncertainty of neural networks in sparse information form. In *International Conference on Machine Learning*, pages 5702–5713. PMLR, 2020.

[34] Apoorva Sharma, Navid Azizan, and Marco Pavone. Sketching curvature for efficient out-of-distribution detection for deep neural networks. *arXiv preprint arXiv:2102.12567*, 2021.

[35] Christian Rupprecht, Iro Laina, Robert DiPietro, Maximilian Baust, Federico Tombari, Nassir Navab, and Gregory D Hager. Learning in an uncertain world: Representing ambiguity through multiple hypotheses. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3591–3600, 2017.

[36] Marton Havasi, Rodolphe Jenatton, Stanislav Fort, Jeremiah Zhe Liu, Jasper Snoek, Balaji Lakshminarayanan, Andrew Mingbo Dai, and Dustin Tran. Training independent subnetworks for robust prediction. In *International Conference on Learning Representations*, 2020.

[37] Alexandre Rame, Remy Sun, and Matthieu Cord. Mixmo: Mixing multiple inputs for multiple outputs via deep subnetworks. *Proceedings of the IEEE International Conference on Computer Vision*, 2021.

[38] Manuel Haußmann, Fred A Hamprecht, and Melih Kandemir. Sampling-free variational inference of bayesian neural networks by variance backpropagation. In *Uncertainty in Artificial Intelligence*, pages 563–573. PMLR, 2020.

[39] Christopher M Bishop. Mixture density networks. 1994.

[40] Philipp Oberdiek, Matthias Rottmann, and Hanno Gottschalk. Classification uncertainty of deep neural networks based on gradient information. In *IAPR Workshop on Artificial Neural Networks in Pattern Recognition*, pages 113–125. Springer, 2018.

[41] Seong Joon Oh, Kevin Murphy, Jiyan Pan, Joseph Roth, Florian Schroff, and Andrew Gallagher. Modeling uncertainty with hedged instance embedding. *Proceedings of the International Conference on Learning Representations*, 2019.

[42] Jim Winkens, Rudy Bunel, Abhijit Guha Roy, Robert Stanforth, Vivek Natarajan, Joseph R Ledsam, Patricia MacWilliams, Pushmeet Kohli, Alan Karthikesalingam, Simon Kohl, et al. Contrastive training for improved out-of-distribution detection. *arXiv preprint arXiv:2007.05566*, 2020.

[43] Lynton Ardizzone, Jakob Kruse, Sebastian Wirkert, Daniel Rahner, Eric W Pellegrini, Ralf S Klessen, Lena Maier-Hein, Carsten Rother, and Ullrich Köthe. Analyzing inverse problems with invertible neural networks. *arXiv preprint arXiv:1808.04730*, 2018.

[44] Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Hybrid models with deep and invertible features. In *International Conference on Machine Learning*, pages 4723–4732. PMLR, 2019.

[45] Lynton Ardizzone, Radek Mackowiak, Carsten Rother, and Ullrich Köthe. Training normalizing flows with the information bottleneck for competitive generative classification. *Advances in Neural Information Processing Systems*, 33, 2020.

[46] Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of causal inference: foundations and learning algorithms*. The MIT Press, 2017.

[47] Mattia Segù, Alessio Tonioni, and Federico Tombari. Batch normalization embeddings for deep domain generalization. *arXiv preprint arXiv:2011.12672*, 2020.

[48] Anton Obukhov, Maxim Rakhuba, Alexander Liniger, Zhiwu Huang, Stamatios Georgoulis, Dengxin Dai, and Luc Van Gool. Spectral tensor train parameterization of deep learning layers. In *International Conference on Artificial Intelligence and Statistics*, pages 3547–3555. PMLR, 2021.

[49] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

[50] Hanie Sedghi, Vineet Gupta, and Philip M Long. The singular values of convolutional layers. *arXiv preprint arXiv:1805.10408*, 2018.

[51] Sahil Singla and Soheil Feizi. Bounding singular values of convolution layers. *arXiv preprint arXiv:1911.10258*, 2019.

[52] Mihaela Rosca, Theophane Weber, Arthur Gretton, and Shakir Mohamed. A case for new neural network smoothness constraints. *arXiv preprint arXiv:2012.07969*, 2020.

[53] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[54] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[55] Jörn-Henrik Jacobsen, Arnold Smeulders, and Edouard Oyallon. i-revnet: Deep invertible networks. *arXiv preprint arXiv:1802.07088*, 2018.

[56] J Behrmann, W Grathwohl, RTQ Chen, D Duvenaud, and JH Jacobsen. Invertible residual networks. arxiv e-prints. *arXiv preprint arXiv:1811.00995*, 2018.

[57] Moksh Jain, Salem Lahlou, Hadi Nekoei, Victor Butoi, Paul Bertin, Jarrid Rector-Brooks, Maksym Korablyov, and Yoshua Bengio. Deup: Direct epistemic uncertainty prediction. *arXiv preprint arXiv:2102.08501*, 2021.

[58] Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Do deep generative models know what they don't know? In *International Conference on Learning Representations*, 2018.

[59] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[60] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer school on machine learning*, pages 63–71. Springer, 2003.

[61] Michalis Titsias. Variational learning of inducing variables in sparse gaussian processes. In *Artificial intelligence and statistics*, pages 567–574. PMLR, 2009.

[62] James Hensman, Alexander Matthews, and Zoubin Ghahramani. Scalable variational gaussian process classification. In *Artificial Intelligence and Statistics*, pages 351–360. PMLR, 2015.

[63] David Burt, Carl Edward Rasmussen, and Mark Van Der Wilk. Rates of convergence for sparse variational gaussian process regression. In *International Conference on Machine Learning*, pages 862–871. PMLR, 2019.

[64] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[65] Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

[66] GLENN W BRIER. Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, 78(1):1–3, 1950.

[67] Miha Vuk and Tomaz Curk. Roc curve, lift chart and calibration plot. *Metodoloski zvezki*, 3(1):89, 2006.

[68] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 55:5, 2014.

[69] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *Proceedings of the International Conference on Learning Representations*, 2019.

[70] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[71] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[72] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[73] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[74] Claudio Michaelis, Benjamin Mitzkus, Robert Geirhos, Evgenia Rusak, Oliver Bringmann, Alexander S Ecker, Matthias Bethge, and Wieland Brendel. Benchmarking robustness in object detection: Autonomous driving when winter is coming. *arXiv preprint arXiv:1907.07484*, 2019.

[75] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In *International Conference on Learning Representations (ICLR)*, 2016.

[76] Fisher Yu, Vladlen Koltun, and Thomas Funkhouser. Dilated residual networks. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[77] Hermann Blum, Paul-Edouard Sarlin, Juan Nieto, Roland Siegwart, and Cesar Cadena. The fishyscapes benchmark: Measuring blind spots in semantic segmentation. *arXiv preprint arXiv:1904.03215*, 2019.

[78] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*, 2017.

[79] Jean Daunizeau. Semi-analytical approximations to statistical moments of sigmoid and softmax mappings of normal variables. *arXiv preprint arXiv:1703.00091*, 2017.

[80] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1183–1192. JMLR. org, 2017.