# The Hidden Uncertainty in a Neural Network's Activations

**Janis Postels** [*1] **Hermann Blum** [*1] **Yannick Strümpler** [1] **Cesar Cadena** [1] **Roland Siegwart** [1] **Luc Van Gool** [1] **Federico Tombari** [2 3]

## Abstract

The distribution of a neural network's latent representations has been successfully used to detect out-of-distribution (OOD) data. This work investigates whether this distribution moreover correlates with a model's epistemic uncertainty, thus indicates its ability to generalise to novel inputs. We first empirically verify that epistemic uncertainty can be identified with the surprise, thus the negative log-likelihood, of observing a particular latent representation. Moreover, we demonstrate that the output-conditional distribution of hidden representations also allows quantifying aleatoric uncertainty via the entropy of the predictive distribution. We analyse epistemic and aleatoric uncertainty inferred from the representations of different layers and conclude that deeper layers lead to uncertainty with similar behaviour as established - but computationally more expensive - methods (e.g. deep ensembles).

## 1. Uncertainty from Latent Representations

### 1.1. Types of Uncertainty

Uncertainty estimation in machine learning (ML) tries to assign a level of confidence to a model's output. Thus, the true uncertainty correlates with a model's performance. While a correct prediction can have high uncertainty, average model performance degrades when uncertainty increases.

A model's performance fluctuates within the training data distribution. One reason for this is inherent noise in the data, which causes irreducible *aleatoric* uncertainty (Kiureghian & Ditlevsen, 2009). Additionally, the performance depends on the choice of training data, model, and parameters. Uncertainties resulting from these choices are reducible and should be distinguished from their irreducible counterparts. (Kiureghian & Ditlevsen, 2009) call these uncertainties '*epistemic*'. The same terminology was introduced to the ML community (Senge et al., 2014; Gal, 2016).

[*]Equal contribution [1]ETH Zurich [2]Google [3]Technical University Munich. Correspondence to: Janis Postels <jpostels@ethz.ch>, Hermann Blum <blumh@ethz.ch>.

**The Two Faces of Epistemic Uncertainty.** Many related works solely evaluate epistemic uncertainty using OOD detection (Mandelbaum & Weinshall, 2017; Alemi et al., 2018; van Amersfoort et al., 2020; Liu et al., 2020). This wrongly equates epistemic uncertainty estimation with OOD detection. While OOD detection, as an important failure source, is related to epistemic uncertainty estimation and, thus, is a necessary evaluation method, we stress that OOD detection performance alone is not sufficient. This becomes apparent when considering that OOD detection represents a model agnostic task and epistemic uncertainty should be aligned with a model's generalisation. Consequently, high epistemic uncertainty is only expected for those OOD samples that lead to degradation of model performance. In this work we therefore evaluate epistemic uncertainty along two dimensions - its behaviour close to and far away from the training distribution. We expect that a good epistemic uncertainty estimate successfully detects samples that reside far away from the data distribution as well as indicates how well the model generalises to data close to it. We particularly investigate the latter in section 2.1 by applying perturbations of increasing magnitude to the input while tracking estimated uncertainty and model performance.

### 1.2. Uncertainty of Deterministic Neural Networks

We now establish a connection between the density of hidden representations and the information-theoretic surprise of observing a particular prediction under the assumption of a deterministic neural network. Importantly, we only assume the model to be deterministic at inference. Thus, one is free to use any stochastic regularisation methods at training time. Moreover, we relate this surprise to aleatoric and epistemic uncertainty estimates used by prior work.

Given a dataset $\mathcal{D} = \{\mathbf{X}, \mathbf{Y}\}$, the goal of a discriminative task is to model the conditional distribution $p(\mathbf{y}|\mathbf{x})$, where $\mathbf{x} \in \mathbf{X}$ and $\mathbf{y} \in \mathbf{Y}$. Let $\hat{\mathbf{y}}$ correspond to the output of a neural network trained to approximate $p(\mathbf{y}|\mathbf{x})$, which induces the

conditional distribution $p_\theta(\hat{\mathbf{y}}|\mathbf{x})$. The conditional entropy

$$H(\hat{\mathbf{y}}|\mathbf{x}) = E_{p_\theta(\hat{\mathbf{y}},\mathbf{x})}\left[-\log\left(p_\theta(\hat{\mathbf{y}}|\mathbf{x})\right)\right] = E_{p(\mathbf{x})}\left[-\int \mathrm{d}\hat{\mathbf{y}} p_\theta(\hat{\mathbf{y}}|\mathbf{x})\log\left(p_\theta(\hat{\mathbf{y}}|\mathbf{x})\right)\right] \quad (1)$$

corresponds to the total uncertainty about $\hat{\mathbf{y}}$ given knowledge of $\mathbf{x}$ across the dataset $\mathcal{D}$. Both quantities under the expectation have been successfully used to quantify aleatoric uncertainty in deterministic neural networks for classification (Hendrycks & Gimpel, 2016; Alemi et al., 2018)[1] and regression (Kendall & Gal, 2017)[2]. The negative log-likelihood $-\log(p_\theta(\hat{\mathbf{y}}|\mathbf{x}))$ is called surprisal of a particular value $\hat{\mathbf{y}}$ given an input $\mathbf{x}$. Further, $-\int \mathrm{d}\hat{\mathbf{y}} p_\theta(\hat{\mathbf{y}}|\mathbf{x})\log\left(p_\theta(\hat{\mathbf{y}}|\mathbf{x})\right)$ is the expected surprisal for a given input $\mathbf{x}$.

However, eq. 1 only quantifies the uncertainty/surprise about predictions given a specific input $\mathbf{x}$ and not about observing $\mathbf{x}$ in the first place. To account for this, consider the entropy of the joint probability distribution

$$H(\hat{\mathbf{y}},\mathbf{x}) = H(\mathbf{x}) + H(\hat{\mathbf{y}}|\mathbf{x}) = E_{p(\mathbf{x})}\big[-\log(p(\mathbf{x})) - \int \mathrm{d}\hat{\mathbf{y}} p_\theta(\hat{\mathbf{y}}|\mathbf{x})\log\left(p_\theta(\hat{\mathbf{y}}|\mathbf{x})\right)\big]. \quad (2)$$

In eq. 2 the negative log-likelihood $-\log(p(\mathbf{x}))$ quantifies the surprise about observing $\mathbf{x}$ and requires learning the density $p(\mathbf{x})$. Estimating $-\log(p(\mathbf{x}))$ directly would provide an uncertainty estimate independent of the neural network used to parameterize $p_\theta(\hat{\mathbf{y}}|\mathbf{x})$ and its generalisation. It can be considered as a model-agnostic distributional uncertainty. Moreover, this distributional uncertainty incorporates the entire information in the distribution of $\mathbf{X}$ - also information irrelevant for the discriminative task. Additionally, a high-dimensional $\mathbf{X}$ induces further complications, since learning such densities is an active field of research.

Consider a neural network comprised of $L$ layers with $L-1$ latent representations $(\mathbf{z}_0, .., \mathbf{z}_{L-2})$ with a joint distribution factorizing according to $p_\theta(\mathbf{x}, \hat{\mathbf{y}}, \mathbf{z}_0, .., \mathbf{z}_{L-2}) = p_\theta(\hat{\mathbf{y}}|\mathbf{z}_{L-2})p(\mathbf{z}_{L-2}|\mathbf{z}_{L-3})...p_\theta(\mathbf{z}_0|\mathbf{x})p(\mathbf{x})$. Exploiting its deterministic nature and the data processing inequality allows us to make the following statements:

$$H(\hat{\mathbf{y}}|\mathbf{z}_i) = H(\hat{\mathbf{y}}|\mathbf{x}) \quad \forall i \in [0, ..., L-2] \quad (3)$$

$$H(\mathbf{x}) \geq H(\mathbf{z}_0) \geq ... \geq H(\mathbf{z}_{L-2}) \quad (4)$$

A detailed derivation is in the supplement. These inequalities relate the distribution of the latent representations to the uncertainties in eq. 2. Inequality 4 implies that uncertainty estimates based on deeper layers are less conservative. On the other side, since the distribution of the latent space at a particular layer is a functional of all the previous layers, we expect these uncertainty estimates to incorporate some degree of model dependence. We verify this intuition empirically in section 2.1 by detecting OOD samples using density estimate of different layers.

To estimate aleatoric uncertainty, we learn the joint density $p(\mathbf{z}_i, \hat{\mathbf{y}})$ at layer $i$ by estimating the output-conditional distribution $p(\mathbf{z}_i|\hat{\mathbf{y}})$ and $p(\hat{\mathbf{y}})$. Using Bayes' formula and marginalization over $\hat{\mathbf{y}}$, we compute both values under the expectations in eq. 1. In section 2.1 we find empirically that deeper layers enable better aleatoric uncertainty estimates.

To summarise, given a novel input $\mathbf{x}^\star$ with corresponding latent vector $\mathbf{z}_i^\star$, we identify the epistemic uncertainty with the surprise of observing that latent representation $-\log p(\mathbf{z}_i^\star)$ and the aleatoric uncertainty with the expected surprisal of $\hat{\mathbf{y}} \sim p(\hat{\mathbf{y}}|\mathbf{z}_i^\star)$ which we subsequently denote with $h(\hat{\mathbf{y}}|\mathbf{z}_i^\star)$:

$$-\log p(\mathbf{z}^\star) = -\log\left(\int \mathrm{d}\hat{\mathbf{y}} p(\mathbf{z}^\star|\hat{\mathbf{y}})p(\hat{\mathbf{y}})\right) \quad (5)$$

$$h(\hat{\mathbf{y}}|\mathbf{z}_i^\star) = -\int \mathrm{d}\hat{\mathbf{y}} p_\theta(\hat{\mathbf{y}}|\mathbf{z}_i^\star)\log\left(p_\theta(\hat{\mathbf{y}}|\mathbf{z}_i^\star)\right) \quad (6)$$

We refer to the supplement for an analysis of potential pitfalls and a discussion of implementation details.

## 2. Experiments

While there exists no ground truth for uncertainty, we follow the approach of other works (Gal & Ghahramani, 2016; Lakshminarayanan et al., 2017; Malinin & Gales, 2018) and examine epistemic uncertainty over input transformations and distributional shifts and aleatoric uncertainty via the correlation with the network's predictive performance on in-distribution data. We first investigate the behaviour of uncertainties extracted from the output-conditional density over varying depth of the representations on image classification. The supplementary materials provide additional experiments on regression toy datasets, additional training losses and experimental details and parameters.

### 2.1. Image Classification

We train a multilayer perceptron (FCNet) consisting of fully-connected layers and ReLU activations on MNIST (LeCun et al., 1998) and FashionMNIST (Xiao et al., 2017) and ResNet18 (He et al., 2016) on CIFAR10 (Krizhevsky et al., 2009) and SVHN (Netzer et al., 2011).

#### 2.1.1. Depth Dependant Uncertainty Quality

We estimate the output-conditional density of the hidden representations on the training data of FCNet and ResNet18

[1] In fact, (Hendrycks & Gimpel, 2016) uses the maximum softmax probability. This does not change the ordering of the uncertainty values, since the logarithm is a strictly monotonic function.

[2] The authors parameterize the output with a unimodal Gaussian and use its variance as aleatoric uncertainty. In this case the entropy is proportional to the logarithm of the variance.

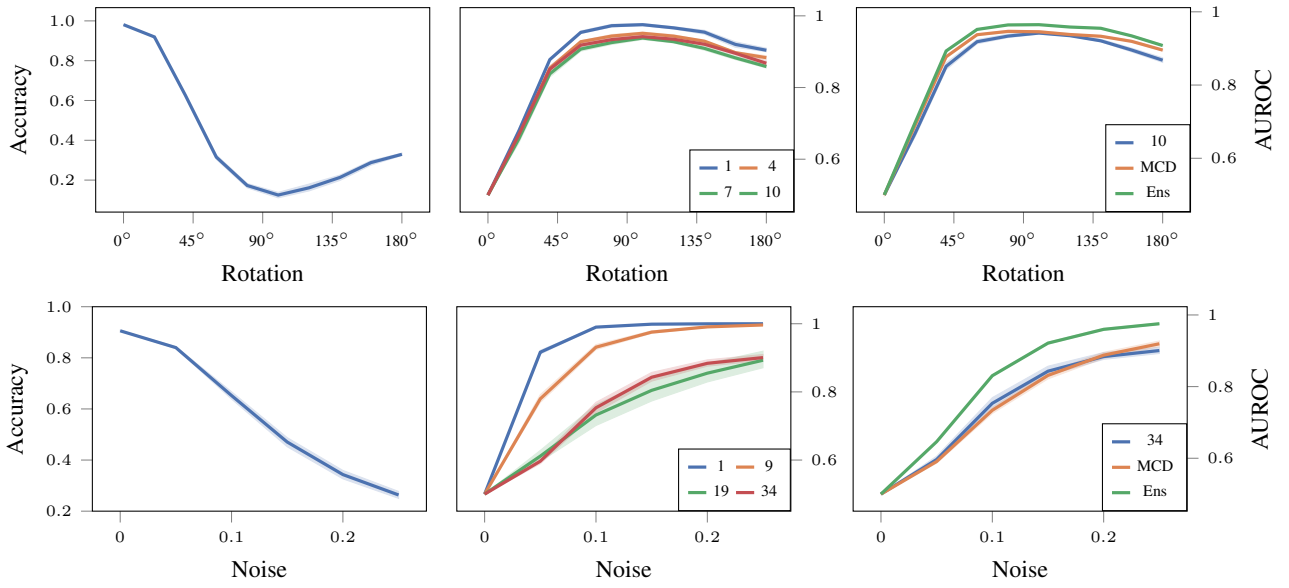



Figure 1: Quality of the estimated epistemic uncertainty over different layers for FCNet on MNIST (top), and ResNet18 on SVHN (bottom). We plot the classification accuracy (**left**) and the area under the ROC curve (AUROC) (obtained using epistemic uncertainty as threshold) against the perturbation magnitude on the test data (**center** and **right**). We compare epistemic uncertainty obtained from hidden representations of varying depth (**center**) and other methods (**right**). We observe that (i) density estimates based on deeper layers match established - but computationally more expensive - methods (i.e. MC Dropout (MCD) and Deep Ensembles (Ens)) and (ii) those based on shallow layers yield more conservative estimates.

using a Gaussian mixture model (GMM) with five components per class. Each training and subsequent density estimation is conducted five times. We quantify epistemic and aleatoric uncertainty according to eq. 5 and eq. 6. We compare our results to MC dropout (Gal & Ghahramani, 2016) and deep ensembles (Lakshminarayanan et al., 2017) for which we compute aleatoric and epistemic uncertainty according to (Gal et al., 2017) (see supplement).

We evaluate the epistemic uncertainty estimate under continuous distributional shifts. We expect epistemic uncertainty to correlate with model generalisation, which is the observed behaviour of established methods such as MC dropout and deep ensembles. We chose to compare with these methods due to their popularity. We compute the area under the ROC curve (AUROC) between the unperturbed testset and each perturbation magnitude. The AUROC measures how well a binary classifier identifies the perturbed test samples based on the estimated epistemic uncertainty. On MNIST we rotate test samples from $0°$ to $180°$ in $20°$ steps and on SVHN we add independent Gaussian noise to the pixel values of input images, while increasing the standard deviation from 0 to 80 with step size 10. Fig. 1 shows the results. We observe that the epistemic uncertainty obtained from density estimates from shallow layers behaves more conservatively in the sense that they are quicker to label perturbed data as OOD. On the other side, epistemic uncertainty estimates from deeper layers behave less conservatively and similar to MC Dropout and Deep Ensembles, while only requiring training one model and a single forward pass.

Fig. 2 analyzes the quality of the estimated aleatoric uncertainty using calibration curves. These show the accuracy of the neural network depending on the uncertainty. We use a threshold sliding through increasing percentiles of the aleatoric uncertainty and plot the remaining accuracy. Ideally, this yields a strictly monotonically decreasing curve. We compare aleatoric uncertainty from various layers, MC dropout, deep ensembles and the softmax entropy. Our experiments show that aleatoric uncertainty from density estimates of deeper layers demonstrates similar calibration curves to established approaches. However, aleatoric uncertainty based on shallow layers can perform poorly (e.g. ResNet18 on SVHN). As discussed in section A.1, we argue that this results from difficulties in estimating the density of very high-dimensional hidden representations.

#### 2.1.2. Epistemic Uncertainty on OOD Detection

We now evaluate the OOD detection performance of epistemic uncertainty. Table 1 reports average performance over 5 independent trainings, with standard deviations in the supplement. We measure the AUROC of the OOD detection using epistemic uncertainty of different layers and compare with deep ensembles. As OOD data we additionally evaluate on Omniglot (Lake et al., 2015), STL10 (Coates et al., 2011), as well as test data perturbed with additive Gaussian

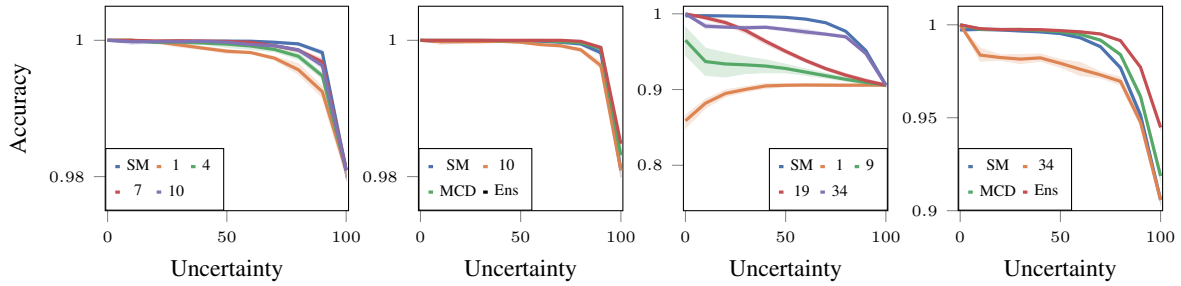



Figure 2: Analysis of the calibration of aleatoric uncertainty on MNIST (left 2) and SVHN (right 2). We plot the remaining accuracy of those test samples above the increasing aleatoric uncertainty threshold. We observe that aleatoric uncertainty based on deeper layers tends to behave similar as other approaches (softmax entropy (SM), deep ensembles (Ens), MC dropout (MCD)).

| | OOD Data | L 1 | L 4 | L 7 | L 10 | Ensemb. |
|---|---|---|---|---|---|---|
| Trained on MNIST | F-MNIST | **0.975** | 0.922 | 0.855 | 0.811 | 0.896 |
| | omniglot | 0.972 | 0.937 | 0.892 | 0.893 | **0.979** |
| | white noise | **1.000** | 0.972 | 0.903 | 0.841 | 0.785 |
| | Rotated 90° | **0.978** | 0.976 | 0.950 | 0.935 | 0.965 |
| | HFlip | 0.902 | **0.907** | 0.883 | 0.864 | 0.905 |
| | VFlip | **0.887** | 0.868 | 0.851 | 0.830 | 0.881 |
| trained on FashionMNIST | MNIST | 0.985 | **0.991** | 0.975 | 0.978 | 0.962 |
| | omniglot | 0.971 | **0.987** | 0.960 | 0.967 | 0.960 |
| | white noise | **1.000** | 0.985 | 0.971 | 0.930 | 0.840 |
| | Rotated 90° | **0.884** | 0.780 | 0.804 | 0.835 | 0.670 |
| | HFlip | **0.719** | 0.696 | 0.701 | 0.693 | 0.657 |
| | VFlip | 0.898 | 0.891 | 0.891 | **0.901** | 0.845 |

| | OOD Data | L 1 | L 9 | L 19 | L 34 | Ensemb. |
|---|---|---|---|---|---|---|
| Trained on SVHN | CIFAR10 | **0.991** | 0.974 | 0.934 | 0.907 | 0.976 |
| | STL10 | **0.999** | 0.991 | 0.951 | 0.912 | 0.982 |
| | white noise | **1.000** | **1.000** | 0.986 | 0.903 | 0.992 |
| | Rotated 90° | 0.615 | 0.646 | 0.689 | 0.918 | **0.957** |
| | HFlip | **0.500** | **0.503** | **0.495** | **0.500** | **0.501** |
| | VFlip | 0.506 | 0.520 | 0.551 | 0.708 | **0.736** |
| Trained on CIFAR10 | SVHN | 0.042 | 0.029 | 0.091 | **0.736** | 0.723 |
| | STL10 | 0.790 | **0.871** | 0.821 | 0.651 | 0.806 |
| | white noise | **1.000** | **1.000** | **1.000** | 0.681 | 0.999 |
| | Rotated 90° | 0.553 | 0.517 | 0.543 | 0.757 | **0.824** |
| | HFlip | **0.500** | **0.500** | **0.499** | **0.500** | **0.501** |
| | VFlip | 0.519 | 0.513 | 0.537 | 0.714 | **0.789** |

Table 1: Measured AUROC for OOD detection based on estimated epistemic uncertainty from hidden activations. We evaluate the epistemic uncertainty computed from the distribution of the output of several affine layers. We evaluate layers (L) 1, 4, 7, 10 for a multilayer perceptron trained on MNIST or FashionMNIST (**left**) and layers 1, 9, 19, 34 for a ResNet18 trained on SVHN or CIFAR10 (**right**). For comparison, we further report the performance of an ensemble of 10 identical networks. We observe that uncertainty estimates based on shallow layers demonstrate strong OOD performance.

noise, rotated by 90° and flipped horizontally/vertically.

We generally observe a strong performance of epistemic uncertainty obtained from shallow layers which coincides with the results of section 2.1.1, where we found that shallow layers yield more conservative estimates. However, we find that, when using a convolutional architecture (ResNet18), epistemic uncertainty obtained from shallow layers fails to detect OOD data generated by globally transforming the test data. We argue that this is expected behaviour since the hidden representations of shallow layers in convolutional networks denote low-level features that are likely to not represent such global transformations. Finally, we note the poor performance of shallow layers when training on CIFAR10 and evaluating on SVHN. These results suggest that recently found difficulties of explicit generative models on the task of OOD detection (Nalisnick et al., 2019) also apply to shallow layers. However, epistemic uncertainty from deeper layers does not suffer from this problem.

## 3. Discussion & Conclusion

This work introduced a holistic framework for uncertainty estimation based on the density of latent representations and verified its effectiveness empirically on the task of classification (section 2.1). Latent representations of shallow layers yield more conservative uncertainty estimates, while epistemic uncertainty obtained from deeper layers behaves similarly to MC Dropout and Deep Ensembles (section 2.1.1). Further, we showed that latent densities denote an effective proxy for epistemic uncertainty in terms of calibration.

Section 2.1.2 concludes that shallow layers have superior OOD performance, even often outperforming Deep Ensembles. This is in line with the results in section 2.1.1 and comes at the cost of worse uncertainty calibration close to the training data distribution.

## References

Alemi, A. A., Fischer, I., and Dillon, J. V. Uncertainty in the variational information bottleneck. *arXiv preprint arXiv:1807.00906*, 2018.

Ardizzone, L., Lüth, C., Kruse, J., Rother, C., and Köthe, U. Guided image generation with conditional invertible neural networks. *arXiv preprint arXiv:1907.02392*, 2019.

Blum, H., Sarlin, P.-E., Nieto, J., Siegwart, R., and Cadena, C. The fishyscapes benchmark: Measuring blind spots in semantic segmentation. *arXiv preprint arXiv:1904.03215*, 2019.

Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., and Adam, H. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 801–818, 2018.

Coates, A., Ng, A., and Lee, H. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 215–223, 2011.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Detrano, R., Janosi, A., Steinbrunn, W., Pfisterer, M., Schmid, J.-J., Sandhu, S., Guppy, K. H., Lee, S., and Froelicher, V. International application of a new probability algorithm for the diagnosis of coronary artery disease. *The American journal of cardiology*, 64(5):304–310, 1989.

Dinh, L., Krueger, D., and Bengio, Y. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using real nvp. *International Conference on Learning Representaions*, 2017.

Efron, B., Hastie, T., Johnstone, I., Tibshirani, R., et al. Least angle regression. *Annals of statistics*, 32(2):407–499, 2004.

Gal, Y. Uncertainty in deep learning. *University of Cambridge*, 1:3, 2016.

Gal, Y. and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pp. 1050–1059, 2016.

Gal, Y., Islam, R., and Ghahramani, Z. Deep bayesian active learning with image data. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1183–1192. JMLR. org, 2017.

Geiger, A., Lenz, P., and Urtasun, R. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

Godard, C., Mac Aodha, O., and Brostow, G. J. Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 270–279, 2017.

Harrison Jr, D. and Rubinfeld, D. L. Hedonic housing prices and the demand for clean air. *Journal of environmental economics and management*, 5(1):81–102, 1978.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hendrycks, D. and Gimpel, K. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

Kendall, A. and Gal, Y. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, pp. 5574–5584, 2017.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, D. P. and Dhariwal, P. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in neural information processing systems*, pp. 10215–10224, 2018.

Kiureghian, A. D. and Ditlevsen, O. Aleatory or epistemic? does it matter? *Structural Safety*, 31(2):105–112, March 2009. ISSN 0167-4730. doi: 10.1016/j.strusafe.2008.06.020.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. *Citeseer*, 2009.

Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems*, pp. 6402–6413, 2017.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Lee, K., Lee, K., Lee, H., and Shin, J. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in Neural Information Processing Systems*, pp. 7167–7177, 2018.

Liu, J. Z., Lin, Z., Padhy, S., Tran, D., Bedrax-Weiss, T., and Lakshminarayanan, B. Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. *arXiv preprint arXiv:2006.10108*, 2020.

Malinin, A. and Gales, M. Predictive uncertainty estimation via prior networks. In *Advances in Neural Information Processing Systems*, pp. 7047–7058, 2018.

Mandelbaum, A. and Weinshall, D. Distance-based confidence score for neural network classifiers. *arXiv preprint arXiv:1709.09844*, 2017.

Nalisnick, E., Matsukawa, A., Teh, Y. W., Gorur, D., and Lakshminarayanan, B. Do deep generative models know what they don't know? *International Conference on Learning Representations*, 2019.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading digits in natural images with unsupervised feature learning. *Citeseer*, 2011.

Papernot, N. and McDaniel, P. Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning. *arXiv preprint arXiv:1803.04765*, 2018.

Pinggera, P., Ramos, S., Gehrig, S., Franke, U., Rother, C., and Mester, R. Lost and found: detecting small road hazards for self-driving vehicles. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

Senge, R., Bösner, S., Dembczyński, K., Haasenritter, J., Hirsch, O., Donner-Banzhoff, N., and Hüllermeier, E. Reliable classification: Learning classifiers that distinguish aleatoric and epistemic uncertainty. *Information Sciences*, 255:16–29, 2014.

van Amersfoort, J., Smith, L., Teh, Y. W., and Gal, Y. Simple and scalable epistemic uncertainty estimation using a single deep deterministic neural network. *arXiv preprint arXiv:2003.02037*, 2020.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

# Appendix

The supplementary material is structured in the following way: Section A provides additional explanations and proofs regarding the proposed method. Section B lists all experimental details and additional results to facilitate reproduction and reimplementaion of our experiments. Finally, section C shows two case-studies on deployment of latent activation based uncertainty estimation to real-world autonomous driving scenarios.

## A. Background

### A.1. Pitfalls

**Feature Collapse.** One of the strengths of discriminative neural networks is to ignore task-irrelevant features of the data. However, this can lead to activations of OOD data becoming indistinguishable from activations of the training data. This phenomenon is also called feature collapse (van Amersfoort et al., 2020) and poses a practical challenge for uncertainty quantification based on hidden activations. Different prior works have addressed this problem by tuning additional hyperparameters enforcing adequately structured latent spaces. Hereby, the network is made more sensitive to changes in the input by constraining its Lipschitz constant. This is achieved by using a distance-based loss on the hidden representations (Mandelbaum & Weinshall, 2017), a gradient penalty (van Amersfoort et al., 2020) or spectral normalization (Liu et al., 2020). Also training with the VIB framework (Alemi et al., 2018) regularises the distribution of hidden representations according to a prior distribution.

In our main contribution (section 2.1 & B.2) we do not explicitly address feature collapse, since it requires altering the training procedure, adds architectural constraints and, most importantly, risks overfitting epistemic uncertainty to the task of OOD detection. All mentioned prior works focus on OOD detection and it remains unclear whether such approaches translate into good epistemic uncertainty, especially close to the training data distribution. We demonstrated that, *without additional regularisation*, $\log p(\mathbf{z}_l)$ of deeper layers as a proxy for epistemic uncertainty is similarly calibrated as well established methods, while maintaining good OOD detection performance.

To link these results to prior works above and illustrate the risk of overfitting epistemic uncertainty to OOD detection, we explore within the framework of eq. 4 how regularisation techniques that yield more informative $\mathbf{z'}_l$ ($H(\mathbf{x}) \geq H(\mathbf{z'}_l) \geq H(\mathbf{z}_l)$) influence the estimated epistemic uncertainty. Therefore, we augment the original loss by a weighted reconstruction objective:

$$\tilde{\mathcal{L}} = \mathcal{L}_{orig} + \lambda \, \text{MSE}(\boldsymbol{x}, \hat{\boldsymbol{x}}) \tag{7}$$

where $\boldsymbol{x}$ is the input and $\hat{\boldsymbol{x}} = D(\mathbf{z'}_l)$ is the output of a reconstruction network $D$ (see supplement for details). We evaluate the impact of this regularisation on epistemic uncertainty in section B.1.

**High-Dimensional Densities.** The hidden representations are high-dimensional for large neural networks or shallow layers. This renders estimating their density impractical. Prior work has also faced the curse of dimensionality. For example, (Lee et al., 2018) reduced the dimensionality of hidden representations by performing principal component analysis (PCA) on the training data and, subsequently, reusing the transformation at inference time. Moreover, (Papernot & McDaniel, 2018) uses locality-sensitive hashing in combination with random projections for nearest neighbour search. Here, we follow (Lee et al., 2018) applying PCA and find that it yields satisfying results in practice. However, we note that PCA as well as random projections are not optimal solutions in the case of convolutional neural networks whose hidden representations entail translational invariances that get lost by applying dense projection matrices.

### A.2. Implementation

In order to then estimate epistemic (eq. 5) and aleatoric uncertainty (eq. 6), we extract the latent representations of a particular layer $i$ and learn the output-conditional density $p(\mathbf{z}_i|\hat{\mathbf{y}})$ and the marginal distribution of the neural network's outputs $p(\hat{\mathbf{y}})$ on the training data. We learn $p(\mathbf{z}_i|\hat{\mathbf{y}})$ using an explicit generative model. For image classification (section 2.1), we use one GMM with $k \in \mathbb{N}_{\setminus\{0\}}$ components for each class. For regression (section B.2) which entails continuous predictions, we found that conditional normalizing flows (CNFs) and in particular the conditioning scheme used in (Ardizzone et al., 2019) yielded good results. We refer the reader to the supplementary material for details.

Estimating $p(\hat{\mathbf{y}})$ for classification simply involves counting the predicted labels on the training data. For regression, however, it is necessary to estimate the density of the predictions in the output space. This is relatively simple, since the output space is low dimensional for most common discriminative tasks. We found it sufficient to approximate it with common parametric distributions - a uniform distribution in section B.2 and a betaprime distribution for the predicted depth in the supplement.

We then obtain the epistemic uncertainty in eq. 5 by marginalizing $\hat{\mathbf{y}}$. To solve the integral, we use summation (classification) or numerical integration (regression). We found the latter to be sufficient for the one dimensional output spaces in our regression experiments (see section B.2). Then, we calculate the aleatoric uncertainty in eq. 6 by applying Bayes' formula.

### A.3. Process of Estimating the Output-Conditional Distribution of Hidden Representations

Algorithm 1 show schematically how to estimate the output-conditional density of hidden representations which is needed to quantify uncertainty with the proposed approach.

**Algorithm 1** Estimating the output-conditional distribution of hidden representations.

**Input:** Dataset {X, Y}, Index l of layer used for uncertainty estimation, d maximum dimension of hidden representations.

**Result:** Trained model $M$, output-conditional density model $M_{gen}$ of hidden representations at layer l, estimate of P($\hat{Y}$)

Train model $M$ on {X, Y}

$Z_l \leftarrow$ activations at layer l on {X, Y}

$\hat{Y} \leftarrow$ M(X)

**if** $dim(Z_l) > d$ **then**
  $Z_l \leftarrow$ reduce dimension of $Z_l$ using PCA
**end**

Initialize generative model $M_{gen}$

Train $M_{gen}$ on $Z_l$ given $\hat{Y}$ to estimate P($Z_l|\hat{Y}$)

**if** *classification* **then**
  Estimate marginal categorical distribution of network predictions P($\hat{Y}$) by counting frequencies
**else**
  Estimate marginal distribution of network predictions P($\hat{Y}$) with univariate parametric distribution (e.g. Gaussian, beta prime)
**end**

### A.4. Proof of Equation 3 and 4

Consider a deterministic neural network comprised of $L$ layers with $L-1$ latent representations $(\mathbf{z}_0, .., \mathbf{z}_{L-2})$ and some input $\mathbf{x}$. Since the network is deterministic we know that $p(\mathbf{z'}_i|\mathbf{x}) = \delta(\mathbf{z'}_i - f_i(\mathbf{x}))$ and consequently:

$$\begin{aligned} p(\hat{\mathbf{y}}|\mathbf{x}) &= \int p(\hat{\mathbf{y}}|\mathbf{z'}_i) p(\mathbf{z'}_i|\mathbf{x}) d\mathbf{z'}_i \\ &= \int p(\hat{\mathbf{y}}|\mathbf{z'}_i) \delta(\mathbf{z'}_i - f_i(\mathbf{x})) d\mathbf{z'}_i \\ &= p(\hat{\mathbf{y}}|\mathbf{z}_i) \end{aligned} \tag{8}$$

where $\mathbf{z}_i = f_i(\mathbf{x})$ denotes the evaluation of the neural network until layer $i$. Eq. (3), $H(\hat{\mathbf{y}}|\mathbf{z}_i) = H(\hat{\mathbf{y}}|\mathbf{x})\ \forall i \in [0, L-1]$, follows directly from that. Furthermore, according to the data processing inequality, any deterministic function $f$ applied to a random variable $x$ can only decrease its (differential) entropy: $H(x) \geq H(f(x))$. Considering that the $z_i$ are deterministic functions of the $z_{i-1}$ proves eq. (4): $H(\mathbf{x}) \geq H(\mathbf{z}_0) \geq ... \geq H(\mathbf{z}_{L-2})\ \forall i \in [0, L-1]$.

### A.5. Conditional Normalizing Flows

Normalizing Flows (NFs) (Dinh et al., 2014; 2017) are a family of explicit generative models based on invertible neural networks. Assume we wish to model the distribution $p_X$ of some data $X$. Given some parametric distribution $p_Z$ and a NF $f_\theta$ comprised of weights $\theta$, likelihood evaluation of a data point $x \in X$ in NFs is based on the change of variable formula

$$p_X(x) = p_Z(f_\theta(x)) \left| det \left( \frac{\partial f_\theta(x)}{\partial x} \right) \right| \tag{9}$$

and made feasible by designing invertible layers that allow evaluating the determinant of their Jacobian efficiently. One such layer is the coupling layer introduced by (Dinh et al., 2017). Each coupling layer splits its input activations into two sets $u^{in}_{1/2}$ and uses one set $u^{in}_1$ to compute scaling and translation of the other set $u^{in}_2$ using trainable, non-invertible functions $g_s$ and $g_t$, while it applies the identity function to $u^{in}_1$.

$$\begin{aligned} u^{out}_1 &= u^{in}_1 \\ u^{out}_2 &= \left(u^{in}_2 + g_t\left(u^{in}_1\right)\right) \odot g_s\left(u^{in}_1\right) \end{aligned}$$

This transformation renders the Jacobian a lower triangular matrix where the determinant is simply the product of the main diagonal.

Several ways have been proposed to use NFs for modeling the conditional distribution $p_{X|C}$ of $x \in X$ conditioned on some random variable $c \in C$. We adapt the conditioning scheme used in (Ardizzone et al., 2019), where the authors use a conditional version of the above coupling layer. Consequently, the coupling layers in the conditional normalizing flow $f_\theta(x|c)$ become:

$$\begin{aligned} u^{out}_1 &= u^{in}_1 \\ u^{out}_2 &= \left(u^{in}_2 + g_t\left(u^{in}_1; c\right)\right) \odot g_s\left(u^{in}_1; c\right) \end{aligned}$$

$g_s$ and $g_t$ are implemented as multi-layer perceptrons. To feed the conditional information into $g_s$ and $g_t$, we apply a separate multi-layer perceptron to $c$ and add the result to $g_t\left(u^{in}_1\right)$ and $g_s\left(u^{in}_1\right)$.

### A.6. Estimating Aleatoric and Epistemic Uncertainty with Deep Ensembles and Monte-Carlo Dropout

In the case of MC dropout and deep ensembles, following (Gal et al., 2017) aleatoric uncertainty $u_{al}$ is computed as

the average entropy of the predicted distribution (for several samples from the weight distribution):

$$u_{al} = E_{w\sim p(w)}\left[E_{y\sim p(\hat{y}|x,w)}\left[-\log(p(\hat{y}|x,w))\right]\right]$$

Here, $p(\hat{y}|x,w)$ is the predicted distribution given an input $x$ and a set of weights $w$. $p(w)$ is the weight distribution (i.e. approximated posterior distribution). Here, we omit the conditioning of the weight distribution on the data. The epistemic uncertainty $u_{ep}$ is then estimated by approximating the mutual information between the weights $w$ and the predictions $\hat{y}$ conditioned on the input $x$:

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

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.

## B. Further Experiments and Experimental Details

### B.1. Image Classification

Effects of Regularisation To confirm that more informative hidden representations translate into better OOD detection, we apply an additional reconstruction loss (section A.1). We show this on FCNet trained on MNIST and evaluated FashionMNIST and on ResNet18 trained on SVHN and evaluated on STL10. We reconstruct the input from layer 10/19 (FCNet/ResNet18) [3].

Fig. 3 shows that in both cases the AUROC increases with the weight of the reconstruction loss and the reconstruction quality (PSNR). Most importantly, the improvement in AUROC is already achieved with low-weighted reconstruction losses that do not cause a noticeable reduction in accuracy. Fig. 4 shows how the accuracy and AUROC evolve when the original in-distribution dataset is perturbed with increasing magnitude[4]. Following section 2.1 we apply rotations to MNIST and additive Gaussian noise to SVHN. We can see that the reconstruction weight has no impact on the shape of the accuracy curve, however we generally observe larger AUROCs with increasing reconstruction weights. Note, that this ambiguity complicates the hyperparamter choice since

[3]Unlike for FCNet we do not choose the last layer in tab. 1 for ResNet18 since layer 34 has only 10 dimensions which we consider too small for reconstructing the input image.

[4]We choose the max reconstruction weight such it does not negatively impact accuracy on the validation set.

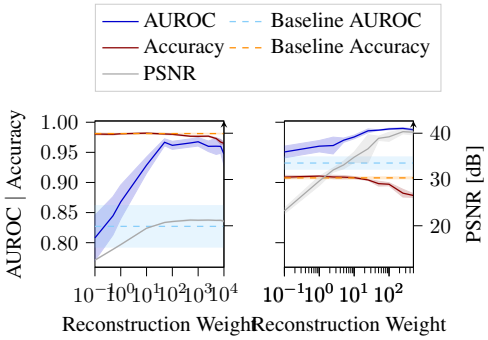



Figure 3: Detecting FashionMNIST/STL10 (left/right) when training on MNIST/SVHN with varying weight of the reconstruction loss. We use layer 10/19 for reconstruction and AUROC computation. Baseline Accuracy & AUROC show the performance of a model trained without additional reconstruction loss. More informative representations boost OOD detection significantly.

it is difficult to determine which AUROC behavior close to the training data distribution is optimal. Training without the reconstruction loss tends induce strong correlation with the behavior MC dropout while higher weights tend to be close to deep ensembles (compare with fig. 1).
We further compare to DUQ (van Amersfoort et al., 2020), which behaves similar to low to medium reconstruction weights.

#### B.1.1. Architectures

The architectures of the neural networks used in our image classification experiments can be found in fig. 5. We applied the multi-layer perceptron (fig. 5) to MNIST and FashionMNIST and ResNet18 (He et al., 2016) to CIFAR10 and SVHN. Note that the dropout layer in FCNet only has $p > 0$ when using MC dropout.

Furthermore, table 2 and 3 show the correspondences between layer indices used in the main work and the layer type. Moreover, in the main work we primarily estimate the output-conditional distribution at the "Add" layers of ResNet18. This is done for convenience since it allows estimating the density from a single layer output instead of taking to parallel branches of the computational graph into account.

#### B.1.2. Deep Ensemble and MC Dropout

All deep ensembles in section 2.1.1 consist of ten neural networks of the same architecture as used by our approach. When training MC Dropout, we place an additional dropout layer after every dense layer (FCNet) or after every ResBlock (resnet).

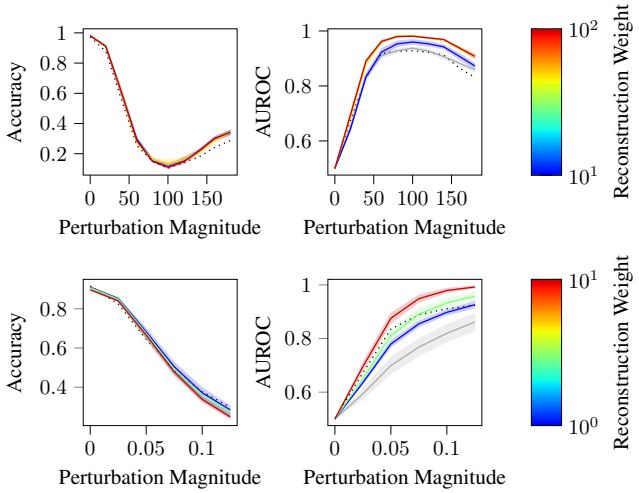
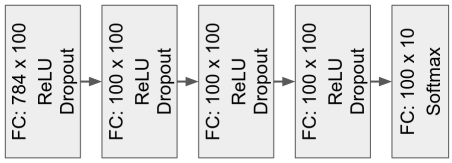



Figure 4: OOD detection performance on rotated MNIST (top) and SVHN with additive Gaussian noise (bottom) for various perturbation magnitudes and reconstruction weights. We reconstruct and estimate uncertainty from layers 10 (MNIST) and 19 (SVHN). The baseline without reconstruction loss is shown in gray. For comparison, DUQ (van Amersfoort et al., 2020) is plotted as dotted line. The AUROC increases with higher reconstruction losses while the Accuracy does not deviate much from the baseline.




(a)

Figure 5: Architecture used in our image classification experiments. Multi-layer perceptron. FC denotes a fully-connected layer.

| Index | Layer |
|---|---|
| 0 | Flatten |
| 1 | Linear |
| 3 | Dropout |
| 4 | Linear |
| 5 | ReLU |
| 6 | Dropout |
| 7 | Linear |
| 8 | ReLU |
| 9 | Dropout |
| 10 | Linear |
| 11 | ReLU |
| 12 | Dropout |
| 13 | Linear |
| 14 | Softmax |

Table 2: Layer index to layer type for FCNet.

| Index | Layer |
|---|---|
| 0 | Input |
| 1 | Conv2D |
| 2 | Batchnorm2D |
| 3 | ReLU |
| 4 | Conv2D |
| 5 | Batchnorm2D |
| 6 | ReLU |
| 7 | Conv2D |
| 8 | Batchnorm2D |
| 9 | Add |
| 10 | Dropout |
| 11 | ReLU |
| 12 | Conv2D |
| 13 | Batchnorm2D |
| 14 | ReLU |
| 15 | Conv2D |
| 16 | Conv2D |
| 17 | Batchnorm2D |
| 18 | Batchnorm2D |
| 19 | Add |
| 20 | Dropout |
| 21 | ReLU |
| 22 | Conv2D |
| 23 | Batchnorm2D |
| 24 | ReLU |
| 25 | Conv2D |
| 26 | Conv2D |
| 27 | Batchnorm2D |
| 28 | Batchnorm2D |
| 29 | Add |
| 30 | Dropout |
| 31 | ReLU |
| 32 | AveragePooling2D |
| 33 | Flatten |
| 34 | Linear |
| 35 | Softmax |

Table 3: Layer index to layer type for ResNet18.

### B.1.3. Training

We describe the training of the neural networks on image classification and the output-conditional densities on the latent representations separately.

**Neural Network Training**

We train both, FCNet and ResNet18, using Adam (Kingma & Ba, 2014) ($\beta_1 = 0.9$, $\beta_2 = 0.999$) and a batch size of 32. We use a learning rate of $0.001$ and weight decay of $0.0001$. We train the networks for 200 epochs, while performing early stopping with a patience parameter of 20. For all datasets, CIFAR10, SVHN, MNIST and FashionMNIST, we use fixed validation set containing $20\%$ of the training data. All reported results are on the test set of the corresponding dataset.

**Output-conditional Density Training**

After training the neural networks, we extract the activations of the training data at layers of varying depth with the corresponding prediction of the neural network. Subsequently, we estimate the distribution of the latent representations of each class prediction with a separate GMM of five components. We use the sklearn GMM implementation[5]. Since the shallow layers of ResNet18 are high-dimensional ($\gg 100$), density estimation - especially in the output-conditional case - becomes challenging. We counter this by reducing the di-

[5]https://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html

mensionality of the activations of each layer to 512 using a principal component analysis. Our multi-layer perceptron does not require dimensionality reduction, since each hidden layer has only 100 hidden units.

We also experimented with using NFs for this task. However, we found that a GMM is sufficient to fit the output-conditional distribution of latent representations throughout all layers.

#### B.1.4. Error Margins

We report the error margins over multiple runs on the OOD experiments in the tables 4 and 5.

### B.2. Regression

#### B.2.1. Experiment

As an illustration, we train regressors on data sampled from $f(x) = \frac{1}{2}\left(\sin(4\pi x - \frac{\pi}{2}) + x\right)$ for $x \in [-1, 1]$. The training data and methods are shown in Fig. 6. We train on the gray outlined data with a simple four-layer perceptron, extract embeddings at the penultimate layer (following our findings from section 2.1) and fit the density with a CNF. For inference, we perform numerical integration to evaluate the likelihood of the latent vectors. The same architecture is used for the ensemble. The Gaussian process (GP) has a RBF kernel with parameters fit to the training data.

We further evaluate OOD detection on standard regression datasets and report results in Table 6. Lacking clear OOD splits, we remove the patient's sex from the features of (Efron et al., 2004; Detrano et al., 1989) and use it to split the data into two slightly different distributions. Since the (unshuffled) (Harrison Jr & Rubinfeld, 1978) is ordered by neighborhoods, we split it into two parts before shuffling. While neither split guarantees non-overlapping data distributions, we take the ensemble as a reference for good uncertainty estimation and observe that the uncertainty estimated from latent densities matches it.

#### B.2.2. Model Training

For the regression prediction, we train a multilayer perceptron with 100 hidden units and 4 hidden layers until convergence. We then extract latent activations $z$ from the penultimate layer and train a CNF conditioned on the predictions of the multilayer perceptron over the whole training data. The CNF is based on GLOW (Kingma & Dhariwal, 2018) and consists of 4 layers with GLOW coupling blocks, each conditioned on the regression preduction, and random permutations. Since our training data is relatively small (750 samples), we do not use minibatches in both trainings.

For the ensemble, we take the same perceptron architecture and train 10 models with different random seeds.

The GP has a simple RBF kernel and is fit (with the sklearn implementation) to the same training data as the perceptron. The fit parameters are `0.601**2 * RBF(length_scale=0.164)`.

#### B.2.3. Evaluation

To evaluate eq. 5, we take a range of 1000 support points of $y \in [-10, 10]$ and estimate $p(\boldsymbol{z}|y)$ with the CNF. We further assume a uniform prior $p(y)$ and numerically integrate over the support points to yield $\log p(\boldsymbol{z})$.

For the visualised confidence regions, we start the integration at the prediction of the regression network and integrate separately in positive and negative direction until the integral reaches the set probability mass. In fig. 6, we visualize the integration boundaries to reach 20% of the probability mass from the CNF.

### B.3. Effects of Regularisation

#### B.3.1. Model Architecture

To explore the effects of regularization, we augment the model with a reconstruction network. The input to the reconstruction network is the latent vector extracted from the layer of our choice. The mapping from the input to the latent vector can be seen as the encoder. We choose the decoder for reconstruction to be symmetrical to the encoder such that their combination forms a symmetrical autoencoder. In the case of the fully connected network, we reconstruction the latent vector from layer $k$ using another $k$ fully connected layers as shown in fig. 7. The reconstruction network for the ResNet18 architecture follows a similar design. The difference is that the output of the convolutional layers has spatial dimensions, which we retain when feeding the latent tensor into the reconstruction network. The decoder is a ResNet18 model itself where the strided convolutions are replaced with a transpose convolution. We build the decoder with stacks of residual units where each stack upsamples the spatial resolution by a factor of two. The number of stacks can be directly inferred from the spatial resolution of the latent tensor.

#### B.3.2. Model Training

The combined model consisting of forward and reconstruction network are trained with the same parameters as described in section B.1.3. For fitting the output-conditional density after training, we only use the forward network. The reconstruction network is discarded after training as its sole purpose is regularization.

| | OOD Dataset | Layer 1 | Layer 9 | Layer 19 | Layer 34 | Ensemble |
|---|---|---|---|---|---|---|
| Trained on SVHN | CIFAR10 | **0.991 ± 0.001** | 0.974 ± 0.004 | 0.934 ± 0.008 | 0.907 ± 0.003 | 0.976 ± 0.002 |
| | STL10 | **0.999 ± 0.000** | 0.991 ± 0.002 | 0.951 ± 0.009 | 0.912 ± 0.003 | 0.982 ± 0.002 |
| | Gaussian noise | **1.000 ± 0.000** | **1.000 ± 0.000** | 0.986 ± 0.011 | 0.903 ± 0.016 | 0.992 ± 0.002 |
| | Rotated (90°) | 0.615 ± 0.001 | 0.646 ± 0.010 | 0.689 ± 0.024 | 0.918 ± 0.002 | **0.957 ± 0.002** |
| | HFlip | **0.500 ± 0.000** | **0.503 ± 0.005** | **0.495 ± 0.010** | **0.500 ± 0.002** | **0.501 ± 0.002** |
| | VFlip | 0.506 ± 0.000 | 0.520 ± 0.004 | 0.551 ± 0.010 | 0.708 ± 0.006 | **0.736 ± 0.001** |
| Trained on CIFAR10 | SVHN | 0.042 ± 0.001 | 0.029 ± 0.004 | 0.091 ± 0.016 | **0.736 ± 0.055** | 0.723 ± 0.048 |
| | STL10 | 0.790 ± 0.010 | **0.871 ± 0.019** | 0.821 ± 0.038 | 0.651 ± 0.008 | 0.806 ± 0.004 |
| | Gaussian noise | **1.000 ± 0.000** | **1.000 ± 0.000** | **1.000 ± 0.000** | 0.681 ± 0.057 | 0.999 ± 0.002 |
| | Rotated (90°) | 0.553 ± 0.003 | 0.517 ± 0.020 | 0.543 ± 0.005 | 0.757 ± 0.001 | **0.824 ± 0.003** |
| | HFlip | **0.500 ± 0.000** | **0.500 ± 0.000** | **0.499 ± 0.001** | **0.500 ± 0.002** | **0.501 ± 0.001** |
| | VFlip | 0.519 ± 0.003 | 0.513 ± 0.006 | 0.537 ± 0.006 | 0.714 ± 0.001 | **0.789 ± 0.004** |

Table 4: OOD performance in terms of AUROC on various datasets when estimating epistemic uncertainty using the hidden activations of a resnet 18 on SVHN and CIFAR10. We evaluate AUROC using the epistemic uncertainty computed from the distribution of the output of several affine layers (1, 9, 19, 34). For comparison, we further report the performance of an ensemble containing 10 identical networks. We observe that uncertainty estimates based on shallow layers demonstrate strong OOD performance.

| | OOD Dataset | Layer 1 | Layer 4 | Layer 7 | Layer 10 | Ensemble |
|---|---|---|---|---|---|---|
| Trained on MNIST | FashionMNIST | **0.975 ± 0.001** | 0.922 ± 0.013 | 0.855 ± 0.032 | 0.811 ± 0.035 | 0.896 ± 0.018 |
| | OMNIGLOT | 0.972 ± 0.002 | 0.937 ± 0.006 | 0.892 ± 0.006 | 0.893 ± 0.011 | **0.979 ± 0.001** |
| | Gaussian noise | **1.000 ± 0.000** | 0.972 ± 0.005 | 0.903 ± 0.005 | 0.841 ± 0.010 | 0.785 ± 0.005 |
| | Rotated (90°) | **0.978 ± 0.001** | 0.976 ± 0.004 | 0.950 ± 0.005 | 0.935 ± 0.006 | 0.965 ± 0.002 |
| | HFlip | 0.902 ± 0.002 | **0.907 ± 0.005** | 0.883 ± 0.007 | 0.864 ± 0.007 | 0.905 ± 0.001 |
| | VFlip | **0.887 ± 0.001** | 0.868 ± 0.002 | 0.851 ± 0.004 | 0.830 ± 0.012 | 0.881 ± 0.002 |
| Trained on FashionMNIST | MNIST | 0.985 ± 0.001 | **0.991 ± 0.003** | 0.975 ± 0.004 | 0.978 ± 0.005 | 0.962 ± 0.004 |
| | OMNIGLOT | 0.971 ± 0.001 | **0.987 ± 0.002** | 0.960 ± 0.012 | 0.967 ± 0.008 | 0.960 ± 0.003 |
| | Gaussian noise | **1.000 ± 0.000** | 0.985 ± 0.002 | 0.971 ± 0.006 | 0.930 ± 0.013 | 0.840 ± 0.004 |
| | Rotated (90°) | **0.884 ± 0.002** | 0.780 ± 0.045 | 0.804 ± 0.030 | 0.835 ± 0.013 | 0.670 ± 0.013 |
| | HFlip | **0.719 ± 0.003** | 0.696 ± 0.007 | 0.701 ± 0.010 | 0.693 ± 0.008 | 0.657 ± 0.010 |
| | VFlip | 0.898 ± 0.006 | 0.891 ± 0.014 | 0.891 ± 0.006 | **0.901 ± 0.013** | 0.845 ± 0.014 |

Table 5: OOD performance in terms of AUROC on various datasets when estimating epistemic uncertainty using the hidden activations of a multilayer perceptron on MNIST and FashionMNIST. We evaluate AUROC using the epistemic uncertainty computed from the distribution of the output of several affine layers (1, 4, 7, 10). For comparison, we further report the performance of an ensemble of 10 neural networks of identical architecture. We observe that uncertainty estimates based on shallow layers demonstrate strong OOD performance.

## C. Additional Experiments on Real-World Models

In the following, we present two case studies on larger models that are trained to perform pixelwise predictions. Such models are often used in safety critical contexts like semantic segmentation for autonomous driving or segmentation of medical images. They also pose a specific implementation problem for density based models since the convolutional latent features cannot be accumulated accross the spatial dimension.

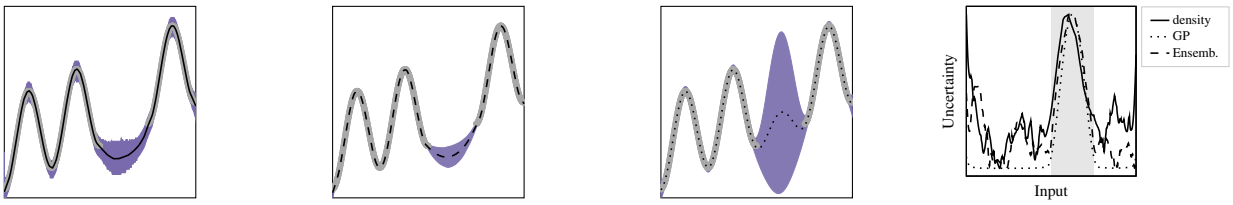
### C.1. Semantic Segmentation

As a large-scale classification experiment, we evaluate our method on semantic segmentation for urban driving. We use the state-of-the-art DeepLabv3+ (Chen et al., 2018) network trained on the Cityscapes urban driving dataset (Cordts et al., 2016) and estimate the feature density in the last layer before the logits.

#### C.1.1. Experimental Setup

We use the pre-trained DeepLabv3+ (Chen et al., 2018) provided by the authors[6] as the base network for the experiments with semantic segmentation. To train the output-conditional GMMs, we extract latent embeddings from the last layer before the logits (`decoder_conv1_0`) before the relu activation. We found the embedding dimensionality of 256 to be low enough to fit densities directly, without furhter dimensionality reduction. To associate latent vectors with predicted classes, we interpolate them bilinearly to the

[6] https://github.com/tensorflow/models/blob/master/research/deeplab/g3doc/model_zoo.md#deeplab-models-trained-on-cityscapes




(a) Latent Density (b) Ensemble (c) Gaussian Process (d) Epistemic Uncertainty

Figure 6: The models are trained on the data outlined in gray. **(a-c):** Model prediction is given in black and the high-confidence area in blue. For better comparison, **(d)** shows the normalised estimated epistemic uncertainty with the data gap in gray. We observe that all methods follow the expected behaviour of high uncertainty in the data gap and low uncertainty on the training data, with slightly increasing uncertainty towards the boundaries for the ensemble and the latent density.

| Data (train/OOD split) | Density $\log p(z)$ | Ensemble |
|---|---|---|
| Boston Housing (Harrison Jr & Rubinfeld, 1978) (70%/30%) | $93.45 \pm 1.64$ | $92.11 \pm 0.68$ |
| SAS Diabetis (Efron et al., 2004) (male/female) | $65.21 \pm 2.87$ | $63.53 \pm 1.30$ |
| Openml Cholesterol (Detrano et al., 1989) (male/female) | $68.50 \pm 1.69$ | $67.67 \pm 1.06$ |

Table 6: We split several regression datasets in 2 parts. We train on the first part and measure AUROC of detecting the second part as outlier (i.e. part 2 should be assigned higher epistemic uncertainty). Since part 2 is not fully OOD, we expect the ensemble to represent desirable AUROC values. We observe that latent density and ensemble have matching

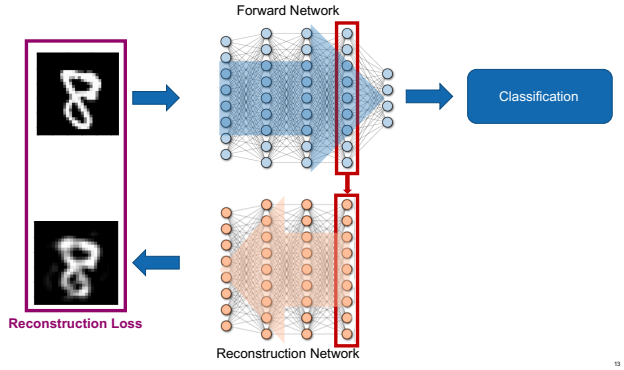



Figure 7: Overview of the model architecture with additional reconstruction network during training.

output resolution of 1024x2048. To counter the correlations of latent vectors within the same image, we subsample a maximum of 500 vectors for every class and image. We then train the GMMs on a shuffled set of 100000 such vectors. We find suitable hyperparameters for the GMMs on the Fishyscapes Lost & Found validation set and use the same hyperparameters for all classes. The best average precision on the validation set was achieved with 2 mixture components per class, tied covariance matrices, 1000 iterations and covariance regularisation factor $0.0001$. We show qualitative examples in figure 9.

For estimating aleatoric uncertainty, we experienced numerical issues when evaluating $p(\hat{\mathbf{y}}|\mathbf{z}^\star)$. This is due to the normalization $\sum_y p(\hat{\mathbf{y}}|\mathbf{z}^\star) = 1$ when some classes are much more likely than others, i.e. when the entropy is low. We counteract these numerical issues by evaluating $h(\hat{\mathbf{y}}|\mathbf{z}^\star)$ on

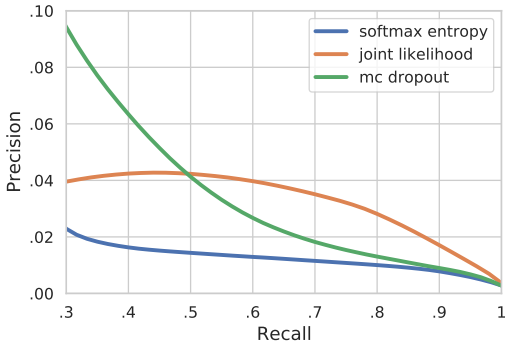



Figure 8: Precision-Recall curve for OOD detection in semantic segmentation on the Fishyscapes benchmark. The shown high-recall region is especially important for safety-critical applications. The curve illustrates the different characteristics that mc dropout and density based uncertainty exhibit on this task.

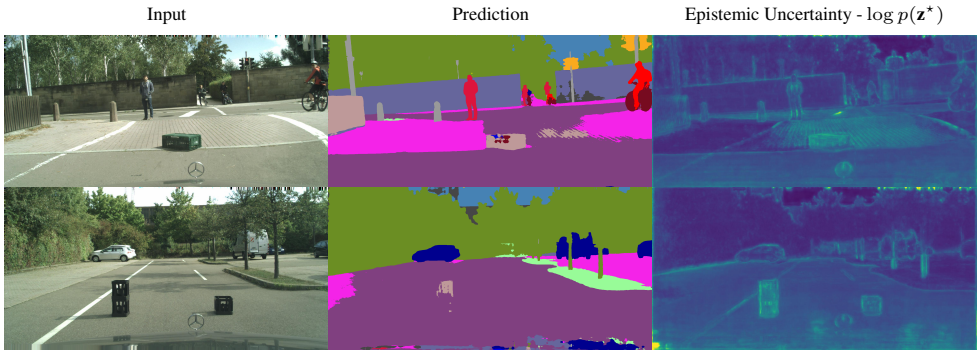



Figure 9: Qualitative examples of the estimated epistemic uncertainty in semantic segmentation. The examples are from the Lost & Found dataset that features images similar to the training set, but with OOD objects (Pinggera et al., 2016). We observe that the estimated epistemic uncertainty is high for anomalous objects and the uncommon street texture in the first row and low where the model correctly generalised.

| | Misclass. Detection | OOD Detection | |
|---|---|---|---|
| method | AUROC | AP | $FPR_{95}$ |
| softmax entropy | **94** | 2.9 | 45 |
| mc dropout | - | **9.8** | 38 |
| joint likelihood | 70 | 3.1 | **24** |
| conditional entropy | 92 | 0.8 | 100 |

Table 7: Comparison against softmax entropy and monte-carlo dropout on the two downstream tasks of misclassification and OOD detection in semantic segmentation. All values in [%]. The aleatoric uncertainty estimated with the conditional entropy can detect misclassified pixel with high AUROC, matching the performance of the softmax entropy. For detecting anomalous objects, the epistemic uncertainty from mc dropout and the joint likelihood have very different performances in the two benchmark metrics. We conclude that aleatoric and epistemic uncertainty can be successfully extracted from latent activations and are separated as expected to indicate different uncertainty aspects.

underfit GMMs[7] and on embedding vectors that were interpolated to output resolution (which was not found necessary for epistemic uncertainty and significantly increases memory usage).

#### C.1.2. Results

We test the estimation of aleatoric uncertainty by measuring the performance for misclassification detection on the Cityscapes validation data. Due to the nature of misclassifications, which are caused by the network itself, we can only directly compare against other methods that work on the exact same network. For OOD detection, we evaluate on the Fishyscapes anomaly detection benchmark (Blum et al., 2019). We report the results in table 7.

We find that the aleatoric uncertainty estimated from output-conditional latent densities closely matches that of the softmax entropy. On the task of OOD detection the estimated epistemic uncertainty based on the log probability of the latent features does however not directly match the behaviour of the MC dropout. In particular, figure 8 shows a very bad performance in low-recall areas and a good performance for the high-recall area, contrary to MC dropout. The qualitative examples in figure 9 show that outlier objects have a lower density, but there are also many other parts of the images with similar values. Further tests over different layers and architectures would be necessary to conclude whether this is an issue of e.g. feature collapse or poorly fitted densities.

Overall, we find that the holistic framework of aleatoric and epistemic uncertainty from latent densities scales with some adaptations to this complex task.

### C.2. Depth Regression

As a large scale application in regression, we apply our framework to monocular depth regression on KITTI (Geiger et al., 2012).

#### C.2.1. Experimental Setup

We use a pretrained model from (Godard et al., 2017)[8] trained on KITTI and extract the activations $z$ from the penultimate layer (output of the penultimate convolution) on the training data and the corresponding depth prediction. We need to estimate two densities: the output-conditional

[7] 1 component, only 50 iterations, covariance regularisation with 0.1

[8] https://github.com/mrharicot/monodepth

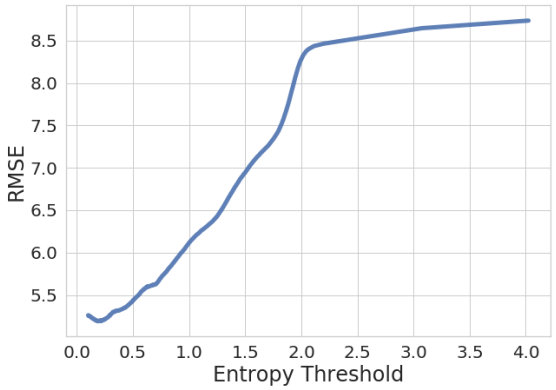



Figure 10: Average RMSE for a monocular depth prediction network (Godard et al., 2017) trained on KITTI (Geiger et al., 2012), accumulated below the given threshold of aleatoric uncertainty $h(\hat{\mathbf{y}}|\mathbf{z}^\star)$. As expected of aleatoric uncertainty, it increases monotonically for larger errors.

density $p(\mathbf{z}^\star|\hat{\mathbf{y}})$ and the density of the predictions $p(\hat{\mathbf{y}})$.

**Output-conditional Density of Latent Representations**

The conditional density $p(\mathbf{z}^\star|\hat{\mathbf{y}})$ is shared across all pixels. To assemble a dataset for training the density model on the training data, we extract the depth predictions pixel-wise and all activations that are in the corresponding receptive field. This leads to a 144 dimensional distribution conditioned on a scalar depth prediction. We randomly extract 12800 samples from the $256 \times 512$ dimensional depth maps and shuffle the resulting dataset to ensure that correlations between adjacent pixels have a minimal impact on the i.i.d. assumption when training the CNF. Subsequently, we train a conditional version of the RealNVP (Dinh et al., 2017) on the resulting dataset, where we use a conditioning scheme similar to the one used by (Ardizzone et al., 2019) (see section A.5). Our RealNVP consists of 3 coupling layers where scaling and translation are each computed using a separate multi-layer perceptron with 2 hidden layers of each 100 dimensions. We refer to the original work (Dinh et al., 2017) for a general description of the RealNVP. We train the CNF using Adam optimizer with learning rate $1e-6$ and weight decay $1e-5$. We further use a batch size of 128 and use early stopping with a patience parameter of 20.

**Parametric Distribution of Depth Predictions**

We empirically found that the distribution of depth predictions on the training data is well represented by a univariate beta prime distribution with $\alpha = 31.76$ and $\beta = 3.07$. Fig. 11 shows the quality of this estimate.

**Evaluation of Aleatoric Uncertainty**

Evaluating aleatoric uncertainty requires computing $h(\hat{\mathbf{y}}|\mathbf{z}^\star) = -\int d\hat{\mathbf{y}} p(\hat{\mathbf{y}}|\mathbf{z}^\star) log\left(p(\hat{\mathbf{y}}|\mathbf{z}^\star)\right)$ where $p(\hat{\mathbf{y}}|\mathbf{z}^\star) = \frac{p(\mathbf{z}^\star|\hat{\mathbf{y}})p(\hat{\mathbf{y}})}{p(\mathbf{z}^\star)}$ and $p(\mathbf{z}^\star)$ is obtained by marginalizing over $\hat{\mathbf{y}}$. Since the depth predictions in our experiment are only one-dimensional, computing $h(\hat{\mathbf{y}}|\mathbf{z}^\star)$ is achieved in a straightforward manner using numerical integration. We discretize the interval of observed depth values $[3, 80]$ with 154 supporting points with equidistant spacing of $0.5$. Note that for higher-dimensional $\hat{\mathbf{y}}$ above integral can be solved more efficiently using importance sampling.

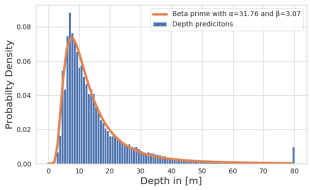



Figure 11: Fitting distribution of depth predictions with univariate beta prime distribution. We visualize normalized histogram of predicted depth values using a pretrained model from (Godard et al., 2017) and a beta prime distribution with parameters $\alpha = 31.76$ and $\beta = 3.07$. The small peak of the predicted depth values at 80m is an artifact of the code on which we base this experiment, since they clip depth predictions at 80m.

### C.2.2. Results

We evaluate the quality of the aleatoric uncertainty. We compute $h(\hat{\mathbf{y}}|\mathbf{z}^\star)$ according to eq. 6 and evaluate its correlation with the RMSE in fig. 10. As expected, larger errors have larger aleatoric uncertainty. We conclude that the aleatoric uncertainty estimate scales to large regression networks and allows extraction of aleatoric uncertainty post-training.

## D. Data Licenses

The data used in our experiments is licensed in the following way:

- MNIST: Creative Commons Attribution-Share Alike 3.0
- FashionMNIST: MIT
- OMNIGLOT: MIT
- SVHN: CC0 1.0 Universal (CC0 1.0) Public Domain according to https://www.kaggle.com/stanfordu/street-view-house-numbers/metadata
- CIFAR10: unknown, but established for scientific research
- STL10: custom ImageNet License (https://image-net.org/download) + Attribution

- Boston Housing: Public Domain Mark 1.0
- SAS Diabetis: public domain
- Openml Cholesterol: Public Domain Mark 1.0
- Cityscapes: https://www.cityscapes-dataset.com/license/
- Lost & Found: http://www.6d-vision.com/lostandfounddataset
- KITTI: Creative Commons Attribution-NonCommercial-ShareAlike 3.0