DISS. ETH NO. 30083

EPISTEMIC UNCERTAINTY QUANTIFICATION AND 3D POINT CLOUD GENERATION USING EXPLICIT GENERATIVE MODELS

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES
(Dr. sc. ETH Zurich)

presented by

JANIS GERHARD POSTELS
MSc in Robotics, Cognition, Intelligence,
Technical University of Munich

born on 10 June 1990

accepted on the recommendation of

Prof. Dr. Luc Van Gool
Prof. Dr. Stephan Günnemann
Prof. Dr. Hao Wang

2024

To Shimeng

ABSTRACT

This thesis covers two related topics - the application of explicit generative models to epistemic uncertainty quantification and 3D point cloud modeling. Firstly, we target efficient epistemic uncertainty quantification in neural networks. Quantifying the epistemic uncertainty of a neural network is a prerequisite for safety-critical applications. Epistemic uncertainty arises from choosing the parameters of a neural network using limited information or applying an unsuitable model class. Given a recent trend towards larger neural networks, quantifying epistemic uncertainty efficiently becomes increasingly important. Therefore, this thesis investigates epistemic uncertainty quantification using explicit generative models trained on the intermediate representations of a neural network.

Initially, we introduce an approach to quantify the epistemic uncertainty of a neural network based on the distribution of its hidden representations. Specifically, we use the log-likelihood of representations observed at test time as a proxy for uncertainty. Moreover, we introduce a regularization approach that ensures that these representations are suitable for uncertainty quantification. The latter is an essential part of the method.

Then, we empirically investigate the previously introduced method - and similar approaches which we term Deterministic Uncertainty Methods - regarding the quality of their uncertainty estimates and the impact of their regularization techniques. We find that the uncertainty estimates obtained by these methods are often poorly calibrated and the consistency of their regularization techniques varies.

Lastly, the application of explicit generative models to uncertainty quantification leads us to investigate normalizing flows - a family of such models. We specifically focus on their application to 3D point clouds which are an increasingly important data modality. We recognize the challenges arising from the invertibility of normalizing flows

and propose an approach for generating samples with improved perceptual quality.

ZUSAMMENFASSUNG

Diese Arbeit beinhaltet zwei verwandte Themen - die Anwendung expliziter generativer Modelle zur Quantifizierung epistemischer Unsicherheit und zur Modellierung von 3D-Punktwolken. Zunächst konzentrieren wir uns auf die effiziente Quantifizierung epistemischer Unsicherheit in neuronalen Netzwerken. Die Quantifizierung der epistemischen Unsicherheit eines neuronalen Netzwerks ist eine Voraussetzung für sicherheitskritische Anwendungen. Epistemische Unsicherheit entsteht durch die Auswahl der Parameter eines neuronalen Netzwerks mittels begrenzter Informationen oder die Anwendung einer ungeeigneten Modellklasse. Angesichts des aktuellen Trends zu größeren neuronalen Netzwerken wird die effiziente Quantifizierung epistemischer Unsicherheit zunehmend wichtiger. Deshalb untersucht diese Arbeit die Quantifizierung epistemischer Unsicherheit unter Verwendung expliziter generativer Modelle, die auf den Features eines neuronalen Netzwerks trainiert wurden.

Zunächst stellen wir einen Ansatz vor, um die epistemische Unsicherheit eines neuronalen Netzwerks anhand der Verteilung seiner Features zu quantifizieren. Insbesondere verwenden wir die Log-Likelihood der Features zur Testzeit als Indikator für Unsicherheit. Darüber hinaus führen wir einen Ansatz zur Regularisierung ein, der sicherstellt, dass diese Features für die Quantifizierung von Unsicherheit geeignet sind. Letzteres ist ein wesentlicher Bestandteil der Methode.

Dann untersuchen wir empirisch die zuvor vorgestellte Methode - sowie ähnliche Ansätze, die wir Deterministic Uncertainty Methods nennen - hinsichtlich der Qualität ihrer abgeschätzten Unsicherheit und den Auswirkungen ihrer Ansätze zur Regularisierung. Wir stellen fest, dass die abgeschätzte Unsicherheit, die mit diesen Methoden erreicht wird, oft schlecht kalibriert ist und die Konsistenz ihrer Ansätze zur Regularisierung schwankt.

Abschließend führt uns die Anwendung expliziter generativer Modelle zur Quantifizierung von Unsicherheit dazu, Normalizing Flows - eine Kategorie solcher Modelle - zu untersuchen. Wir konzentrieren uns speziell auf ihre Anwendung auf 3D-Punktwolken, die eine zunehmend wichtige Datenmodalität darstellen. Wir erkennen die Herausforderungen, die sich aus der Invertierbarkeit von Normalizing Flows ergeben, und entwickeln einen Ansatz zur Generierung von Samples mit verbesserter visueller Qualität.

PUBLICATIONS

The following publications are included in part or as a whole in this thesis (* denotes equal contribution):

- Janis Postels*, Hermann Blum*, Yannick Strümpler, Cesar Cadena, Roland Siegwart, Luc Van Gool, and Federico Tombari. "The Hidden Uncertainty in a Neural Networks Activations". In: *arXiv e-prints, abs/2012.03082*, 2020

- Janis Postels*, Mattia Segu*, Tao Sun, Luca Sieber, Luc Van Gool, Fisher Yu and Federico Tombari. "On the Practicality of Deterministic Epistemic Uncertainty". In: *International Conference on Machine Learning (ICML)* 2022

- Janis Postels, Martin Danelljan, Luc Van Gool, and Federico Tombari. "ManiFlow: Implicitly Representing Manifolds with Normalizing Flows". In: *International Conference on 3D Vision (3DV)* 2022

Furthermore, the following publications were part of my Ph.D. research, but not included in this thesis.

- Diego Martin Arroyo, Janis Postels, and Federico Tombari. "Variational Transformer Networks for Layout Generation". In: *Conference on Computer Vision and Pattern Recognition (CVPR)* 2021

- Matthäus Heer, Janis Postels, Xiaoran Chen, Ender Konukoglu and Shadi Albarqouni. "The OOD Blind Spot of Unsupervised Anomaly Detection". In: *Medical Imaging with Deep Learning* 2021

- Janis Postels*, Mengya Liu*, Riccardo Spezialetti, Luc Van Gool and Federico Tombari. "Go with the Flows: Mixtures of Normalizing Flows for Point Cloud Generation and Reconstruction". In: *International Conference on 3D Vision (3DV)* 2021

- Tao Sun*, Mattia Segu*, Janis Postels, Yuxuan Wang, Luc Van Gool, Bernt Schiele, Federico Tombari and Fisher Yu. "SHIFT: A Synthetic Driving Dataset for Continuous Multi-Task Domain Adaptation". In: *Conference on Computer Vision and Pattern Recognition (CVPR)* 2022

- Yannick Strümpler*, Janis Postels*, Ren Yang, Luc Van Gool and Federico Tombari. "Implicit Neural Representations for Image Compression". In: *European Conference on Computer Vision (ECCV)* 2022

- Mengya Liu, Ajad Chhatkuli, Janis Postels, Luc Van Gool and Federico Tombari. "Unsupervised Template Warp Consistency for Implicit Surface Correspondences". In: *Computer Graphics Forum* 2023

- Janis Postels, Yannick Strümpler, Klara Reichard, Luc Van Gool and Federico Tombari. "3D Compression Using Neural Fields". In: *arXiv e-prints, abs/2311.13009* 2023

ACKNOWLEDGEMENTS

The PhD journey has been a unique experience that would have been impossible without the support of the many people accompanying me over those last four years. First and foremost, I want to thank my adviser Luc Van Gool for providing me with the unparalleled opportunity to join the Computer Vision Lab.

I am incredibly grateful for the continued support and guidance by Federico Tombari. Federico has been mentoring me from the start of my master thesis in Munich and throughout the PhD adventure. He enabled me to conduct my research at the intersection between ETH and Google, and gave me the freedom and means to succeed on this journey.

Moreover, I want to thank all the amazing collaborators without whom my research would not have been possible. Most notably, I would like to thank Yannick Strümpler who has been collaborating with me ever since his semester project three years ago despite the occasionally harsh realities of academia. Further, I would like to thank Hermann Blum for the frequent exchange of ideas. What our projects lacked in success, I gained in experience from our interaction. I am also grateful to Mattia Segu for the frequent and fruitful discussions and joint projects. My gratitude also extends to all the other collaborators not mentioned so far: Martin Danelljan, Tao Sun, Mengya Liu, Diego Arroyo, Ren Yang, Ajad Chhatkuli, Xiaoran Chen, Cesar Cadena, Shadi Albarqouni, Matthäus Heer, Riccardo Spezialetti, Klara Reichard and everybody else.

I am thankful for the inspiring atmosphere at CVL. I would particularly like to thank Kris, Christina and Christine for their continuous effort to keep the lab running. I would also like to thank Anton, Gustav and Ferjad for the abundant chats about research and beyond. Moverover, I am grateful to everybody else crossing my path and whom I did not mention before.

I also thank the entire Semantic Perception team and everybody else at Google who have been accompanying me throughout this journey. My gratitude especially extends to Alessio, Fabian Mentzer, Eirikur, David, Dawid, Keisuke, Fabian Manhardt, Henri, Liesbeth, Enis, Nathalie, Marta, Petra, Yongqin and everybody else.

I would also like to thank my parents, Gerhard and Kordula, who always have my back. Last but not least, I am grateful for the unconditional support of my wonderful wife Shimeng.

CONTENTS

1

INTRODUCTION

This chapter outlines the research conducted in this thesis. Firstly, we are concerned with quantifying epistemic uncertainty of neural networks (NNs) efficiently. Therefore, we will focus on methods for quantifying the epistemic uncertainty using a point estimate of the parameters of a NN. To this end, we proposed an approach to quantifying the uncertainty of NN based on the distribution of its hidden representations (see Chapter 3). We parameterize the distribution using explicit generative models. Subsequently, we empirically investigate the properties of the uncertainty estimate obtained by our approach and related methods (see Chapter 4).

This leads us to dive deeper into the shortcomings of one family of explicit generative models - namely Normalizing Flows (NFs). Hereby, we particularly focus on their application to 3D point clouds and propose an approach to improve the perceptual quality of generated samples (see Chapter 5). We further demonstrate that our method can be applied to higher dimensional domains, such as images.

1.1 DETERMINISTIC UNCERTAINTY QUANTIFICATION

Quantifying the uncertainty of a NN is both a prerequisite for safety-critical applications and useful for downstream tasks such as active learning [1], reinforcement learning [2, 3], continual learning [4, 5] and out-of-distribution (OOD) detection [6].

Aleatoric/Epistemic Uncertainty. Uncertainty in a model's predictions can arise from two different sources [7, 8]. On the one hand, *aleatoric* uncertainty encompasses the noise inherent in the data and is, consequently, irreducible [7]. Even though aleatoric uncertainty

is irreducible it can be incorporated into the model. On the other hand, *epistemic* uncertainty quantifies the uncertainty associated with choosing the model parameters based on limited information or the underlying model class. It vanishes - theoretically - in the limit of infinite data and is, thus, referred to as reducible. This work is concerned with estimating epistemic uncertainty.

Probabilistic Methods for Estimating Epistemic Uncertainty. The predominant approach for quantifying epistemic uncertainty requires a probabilistic treatment of the parameters - *e.g.*using Bayesian Neural Networks (BNNs) [9, 10] or ensembles [11, 12]. Instead of obtaining a point estimate of the parameters θ of a NN, one obtains the posterior distribution over the parameters $p(\theta|\mathcal{D})$ given a dataset $\mathcal{D}$. Subsequently, epistemic uncertainty can be obtained as the conditional mutual information between the parameters and the predictions y given an input x in the case of classification [1, 13]

$$\mathcal{I}(y, \theta|x) = \mathcal{H}(y|x) - \mathcal{H}(y|x, \theta) \tag{1.1}$$

where $\mathcal{H}(x)$ denotes the entropy of x. $\mathcal{H}(y|x, \theta)$ is the aleatoric uncertainty. In the case of regression, one can use the law of total variance to extract epistemic uncertainty [13]:

$$\underset{\theta}{\text{Var}}\left[\text{E}[y|x, \theta]\right] = \text{Var}[y|x] - \underset{\theta}{\text{E}}\left[\text{Var}[y|x, \theta]\right] \tag{1.2}$$

$\underset{\theta}{\text{E}}\left[\text{Var}[y|x, \theta]\right]$ denotes the aleatoric uncertainty. In practice, this requires sampling, and consequently multiple evaluations of the underlying NN, to quantify epistemic uncertainty which is often prohibitively expensive. Moreover, estimating the posterior distribution $p(\theta|\mathcal{D})$ can be challenging. Thus, extensive research has been conducted towards reducing the computational footprint [12, 14, 15] and combining the estimation of $p(\theta|\mathcal{D})$ with existing approaches to optimizing large NNs [16–18].

A Deterministic Treatment of the Parameters. With an increasing demand for large NNs, it is of practical relevance to investigate

approaches to further reduce the computational requirements. In fact, another line of research investigates the estimation of epistemic uncertainty from a point estimate of the parameters θ [19–22]. The greatest common factor shared by these works is the assumption that the internal representations of appropriately regularized NNs allow quantifying epistemic uncertainty - *e.g.*using an explicit generative model [20, 21, 23, 24] or distances [19, 25, 26] of its hidden representations. Subsequently, the log-likelihood or distance to reference points of novel data points is used as an estimate of epistemic uncertainty. While various approaches that fall in the above categorization have been developed, there remains an essential question that needs to be answered before widespread adoption. Namely, does the distribution of hidden representations entail information about the uncertainty of the predictions of a NN? Intuitively, hidden representations that reside far away from representations observed on the training data are more likely unfamiliar territory for the NN and, thus, lead to uncertain predictions. However, there is a difference between performing OOD detection and quantifying the uncertainty related to a discriminative task. The latter requires information about the expected predictive performance given the input. In other words, the uncertainty is required to be *calibrated*.

In this thesis, we first introduce an approach to quantify epistemic uncertainty based on the distribution of hidden representations of a NN (see Chapter 3). Therefore, we tackle two problems: 1) We propose a simple and practical regularization scheme for obtaining hidden representations that allow quantifying uncertainty and 2) we identify suitable explicit generative models capable of learning the distribution of hidden representations. Subsequently, we empirically investigate the quality of the uncertainty estimate obtained by the method proposed in Chapter 3 and similar approaches in more detail (see Chapter 4). We particularly focus on the calibration of these uncertainty estimates and the impact of their regularization methods.

1.2 NORMALIZING FLOWS FOR 3D POINT CLOUDS

Our study of explicit generative models for the hidden representations of NNs leads us to examine one particular family of models more closely, namely Normalizing Flows (NF). NFs [27, 28] are based on invertible neural networks. Assume we wish to model the distribution p_X on a domain X. Given a parametric distribution p_Z on a domain Z and a NF f_θ comprised of weights θ, likelihood evaluation of a data point $x \in X$ in NFs is based on the change of variable formula

$$p_X(x) = p_Z(f_\theta(x)) \left| det \left(\frac{\partial f_\theta(x)}{\partial x} \right) \right| \tag{1.3}$$

and made feasible by designing invertible layers that allow evaluating the determinant of their Jacobian efficiently. Since one can evaluate p_Z, Eq. 1.3 allows optimizing the parameters of NFs to directly minimize the negative log-likelihood of the data. Samples from p_X can then be generated by sampling from p_Z and applying the inverse f_θ^{-1}.

We here focus on the application of NFs to modeling 3D point clouds. We view a point cloud as a set of samples from a distribution representing the 3D structure. The benefit of an explicit generative model is the ability to evaluate the likelihood of points which allows outlier detection. However, there are two challenges when applying NFs to arbitrary 3D data. Firstly, the 3D structure may be comprised of disconnected components which renders the determinant term in Eq. 1.3 unstable as it has to become very large in order to transform p_Z into p_X. This problem has been tackled by the introduction of mixtures of NFs [29]. Secondly, the fact that 3D structures often reside on a 2D manifold embedded in 3D space leads to similar instabilities resulting from the determinant. This problem is often tackled by training on a noisy version of the original 3D structure [30–32] which leads to reduced perceptual quality. In Chapter 5 we tackle this problem by introducing a strategy for generating samples on the underlying manifold given a NF trained on a noisy version of the

data. Our method assumes knowledge of the noise used to perturb the data.

2

RELATED WORK

This chapter provides an overview of research relevant to the covered topics and relates the research presented in this thesis to it. Sec. 2.1 presents related work on uncertainty quantification with a focus on deterministic approaches. Sec. 2.2 discusses relevant work on modeling 3D point clouds and approaches to modeling manifolds using NFs.

2.1 UNCERTAINTY QUANTIFICATION

2.1.1 *Probabilistic Methods.*

Bayesian Deep Learning (BDL) [33–35] is the predominant way to quantify uncertainty. It treats aleatoric as well as epistemic uncertainty in a principled manner (see Eq. 1.1 and Eq. 1.2). Due to well known difficulties of applying BDL to large neural networks, numerous scalable variants have been proposed [14, 16, 17, 36, 37].

Efficient Probabilistic Methods. Methods based on stochastic regularization are particularly compatible with common optimization methods [16, 17, 36, 38, 39]. By keeping stochasticity at inference time, they estimate uncertainty using multiple forward passes. These methods have been, for instance, developed based on dropout [16], batch normalization [17] or stochastic gradient descent [18]. Another line of work estimates the posterior distribution of BNNs using the Laplace-approximation [40–42]. Moreover, efficient ensemble methods were proposed which capture an ensemble in a single model [12, 43–47]. For instance, these require only an ensemble of the last layers [43], factorize weight matrices into shared and separate components [12] or train ensembles as disjoint subgraphs of a single NN [46, 47].

Despite promising results on large-scale tasks and parameter reduction, these methods still require sampling through the model, which can render them impractical given limited compute. To estimate uncertainty in real-time and resource-demanding tasks, recent work has focused on providing uncertainty estimates with a *single forward pass*. One line of work proposes a principled approach for variance propagation in NNs [14, 15, 48, 49]. While these approaches require only a single forward pass, they differ from the methods discussed in this work since they treat the parameters probabilistically and only approximate sampling by propagating moments.

Notably, a parallel line of work uses the predictive conditional distribution of neural networks to quantify aleatoric uncertainty [50–52]. However, these approaches are problematic outside of the training data distribution. Furthermore, neural networks have been found to be poorly calibrated [53], which poses a challenge to all uncertainty estimates.

2.1.2 *Deterministic Uncertainty Quantification*

It has been suggested that a deterministic NN with suitable regularization can represent uncertainty. [54] uses the distance to the closest class centroid in the feature space of a NN. They outperform the softmax entropy and Monte-Carlo (MC) dropout [16] when regularizing the latent space. DUQ [55] replaces the softmax activation function with a class-wise Gaussian distribution with fixed variance for classification. They further constrain the bi-Lipschitz constant of the NN using a two-sided gradient penalty on the input yielding distance-aware representations. They refer to the distance to the closest class centroid as epistemic uncertainty and evaluate it on OOD detection. SNGP [26] also enforces the representations of a NN to be distance-aware by constraining its bi-Lipschitz constant using spectral normalization, which inspired several extensions [21, 56]. Spectral normalization reduces the large memory and runtime overhead at training time compared to a two-sided gradient penalty

if operated using a single power iteration. However, it also constrains the architecture of the underlying NN. Chapter 3 introduces a new regularization technique that is 1) agnostic to the underlying architecture/task and 2) empirically observed to correlate with OOD detection performance. Moreover, we estimate uncertainty using the negative log-likelihood of hidden representations which is also applicable to any NN architecture/task.

The approach to uncertainty quantification proposed in 3 has been extended with distance-aware hidden representations [21] and applied to explaining the origin of failure modes of NN predictions [24]. Moreover, it has been proposed to use the Mahalanobis distance induced by the covariance matrix of a Gaussian fitted to the hidden representations for uncertainty quantification and OOD detection [23, 57, 58].

Chapter 4 categorizes these methods and investigates the calibration of their predicted uncertainty and the impact of the proposed regularization technique empirically.

2.1.3 *Out-of-Distribution Detection Using the Distribution of Hidden Representations.*

Density estimation is an established technique for OOD detection. While research also developed alternative methods [59, 60], generative models are particularly appealing, since they denote a principled, unsupervised approach to the problem by learning the data distribution [61, 62]. However, concerns have been voiced regarding their effectiveness [63] and, in particular for explicit generative models, vulnerability to OOD samples [64]. Ensembles of generative models [65] have been proposed to solve the latter, as well as considering the distribution of log-likelihoods [66, 67]. Another line of research further applies machine learning approaches to the latent space of neural networks [68–70]. In particular, [70] fits a Gaussian mixture model (GMM) to the representations of a neural network. However, they rely on the availability of OOD data limiting its applicabil-

ity. [71] trains a normalizing flow on the latent space and uses its log-likelihood to detect anomalies. Further, [72] improves OOD detection using NFs on the hidden representations of a NN by mitigating the problem of topological mismatch between the source and the target distribution. In contrast, we demonstrate that this also estimates uncertainty and introduce additional regularization to counteract feature collapse. Moreover, [73] uses the rate term in the Information Bottleneck (VIB) [74] for OOD detection. Further, [75] uses a class-wise Gaussian distribution paired with contrastive pretraining based on SimCLR [76] to detect novel inputs. Unfortunately, contrastive pretraining requires gigantic batch sizes rendering this regularization approach not applicable to problems of practical size.

2.2 NORMALIZING FLOWS FOR 3D POINT CLOUDS

Firstly, we discuss related work on modeling 3D point clouds - in particular using NFs. We then continue by discussing relevant research on representing manifolds with NFs.

2.2.1 *Modeling 3D Point Clouds*

Latent GAN [77] models the latent space of a pretrained autoencoder using a Generative Adversarial Network [78]. AtlasNet [79] learns to map 2D patches onto surfaces in 3D by minimizing the Chamfer distance. They also propose to post-process reconstructions by performing Poisson surface reconstruction. Therefore, they densely sample points from their reconstructed mesh and generate oriented normals by shooting rays from infinity on the point cloud. In contrast, our approach yields oriented normals more efficiently with fewer points, since the direction perpendicular to the surface is readily available as the gradient of the log-likelihood of the NF (see Chapter 5). ShapeGF [80] models 3D point clouds using score matching [81]. Therefore, they train a NN to estimate the gradient of the log-likelihood of the distribution of points. At inference time,

they perform temperature annealed Langevin dynamics [82, 83] to transform a random set of points into realistic 3D shapes. Recently, Diffusion Probabilistic Models (DPMs) [84] have shown strong performance on point clouds [85]. Similar to ShapeGF, they transform an initial set of random points into realistic 3D point clouds. However, instead of relying on an unnormalized density function, DPMs learn to approximately invert a diffusion process. Similar to DPMs and ShapeGF, our method also post-processes an initial estimate of the point cloud by maximizing the log-likelihood of the points. However, we require fewer update steps compared to ShapeGF, since our initial estimate of the point cloud is more accurate. Moreover, our method has the advantage of an explicit generative model. For instance, this is useful when attempting anomaly detection or when post-processing the generated point cloud with Poisson surface reconstruction (Sec. 5.2.3).

Normalizing Flows for 3D Point Clouds. NFs have been applied to 3D point clouds. In pioneering work, PointFlow [31] demonstrated promising performance by applying continuous NFs. Later, Discrete PointFlow (DPF) [32] improved upon PointFlow by using discrete NFs, namely RealNVPs [28], which are computationally more efficient while providing better performance. Concurrently, C-Flow [86] proposed a novel conditioning scheme for conditional NFs which they applied to point clouds. More recently NFs have been applied to a few downstream tasks on point clouds. NFs have been applied to single view reconstruction paired with an additional perceptual loss [87]. Further, they have been applied to upsampling [88] and denoising [89] of 3D point clouds. The authors of [29] and [90] proposed simultaneously to model point clouds using a mixture of NFs which lead to better reconstruction performance.

2.2.2 *Normalizing Flows on Manifolds*

Real world data often resides on lower dimensional manifolds embedded in high dimensional space. This is a well-known problem when

applying NFs to model realistic data distributions [91–96]. In [91], the authors propose an approach to explicitly learn NFs on spheres in n-dimensional space. More recently, Brehmer and Cranmer [93, 97] proposed an alternating optimization approach that simultaneously learns the data distribution and the underlying manifold. Furthermore, [92] propose an end-to-end trainable approach to learning distributions on manifolds. Horvat and Pfister [96] add noise to the data distribution and explicitly split the dimensions of a NF into two sets, where one is assumed independent of the noise. Overall, most methods make explicit assumptions about the underlying structure and dimensionality of the manifold. In contrast, this work does not make any assumptions about the dimensionality of the manifold and only requires the manifold to be sufficiently smooth compared to the magnitude of the noise used to inflate the data distribution. Most relevant to this work is the recently proposed SoftFlow [95]. The authors model manifolds using NFs by training a NF conditioned on the standard deviation σ of the Gaussian noise added to the data distribution. During inference they set $\sigma = 0$ to sample from the original manifold. This work shows that one can achieve similar or better performance without training a model on all possible noise magnitudes. We achieve this by recognizing that a NF trained on noisy data implicitly represents the manifold in regions of maximum likelihood.

3

THE HIDDEN UNCERTAINTY IN A NEURAL NETWORK'S ACTIVATIONS

This work introduces a general method for estimating the predictive uncertainty of a neural network (NN) in a single-forward pass. To this end, we apply a simple regularization technique during training enforcing the hidden representations of a NN to be informative about its input. Post training, we use the negative log-likelihood of such informative representations as a proxy for uncertainty. We demonstrate that our approach excels at out-of-distribution (OOD) detection and, moreover, correlates with epistemic uncertainty obtained from Bayesian neural networks and deep ensembles which, while being computationally more expensive, have strong theoretical support. In contrast to prior work, our method only requires minor changes to the training procedure, is agnostic of the neural network's architecture and can be applied to classification as well as regression. Furthermore, we empirically show that the strength of our regularization technique correlates with the performance on OOD detection which allows practitioners to reason about the regularization strength in the absence of OOD data.

3.1 INTRODUCTION

The recent success of deep NNs in a variety of applications [98, 99] has been primarily driven by improvements in predictive performance, while progress on safety-related issues remained relatively slow [100–102]. Thereby, quantifying the uncertainty associated with the predictions of a NN is a central challenge and a prerequisite for their broader deployment. The predominant framework for estimating uncertainty is BDL [9, 35]. However, scaling BDL to neural networks of practical size remains an open research question. Al-

though recent research proposed various scalable approaches [11, 14, 16, 36, 37, 103], they still fall short of delivering a widely adopted practical solution applicable to compute-limited environments.

To bypass these limitations a parallel line of work estimates uncertainty using a deterministic treatment of a NN's parameters. Importantly, this estimates uncertainty in a single-forward pass through examination of the NN's activations. A major challenge of these approaches is feature collapse [55] where the hidden representations of OOD data become indistinguishable from the ones of in-distribution (ID) data. To prevent this, the hidden representations can be regularized to preserve information about the input data [26, 55]. In addition to such regularization, the final softmax layer in models trained on classification is replaced with a distance-aware function, e.g. RBF kernel or Gaussian process (GP). Independently, another line of work showed that the distribution of hidden representations in a NN can be used to detect OOD data [62, 69, 73, 75]. This work operates at the intersection of both directions yielding an efficient method for estimating uncertainty in a single-forward pass applicable to any NN.

We show that the negative log-likelihood of hidden representations not only enables strong OOD detection performance (Sec. 3.3.2) but also correlates with the uncertainty of a model which allows reasoning about the expected performance on novel data. We demonstrate that this is the case for the negative log-likelihood of deep features (Sec. 3.3.3.3 and Sec. 3.3.5). In contrast to prior work this uncertainty estimate is task-agnostic and can be applied to classification as well as regression (Sec. 3.3.3). While prior work constrains the bi-Lipschitz constant of a NN enforcing hidden representations to be distance-aware with respect to a specific distance metric, we enforce the network to be able to reconstruct the input given its representations. This regularization technique is architecture-agnostic and widely applicable, since it leaves the original model and training procedure unchanged. Moreover, we observe that it trades OOD detection performance off with predictive performance (Sec. 3.3.4). This allows practitioners to reason about the regularization strength.

The rest of this work is structured as follows. Sec. 3.2 introduces the proposed framework and discusses its properties and implementation. Then, Sec. 3.3 empirically verifies that the distribution of informative representations excels at OOD detection (Sec. 3.3.2) and correlates with predictive performance of the underlying NN under continuous distributional shifts (Sec. 3.3.3.3). Further, we demonstrate that the proposed method is applicable to regression (Sec. 3.3.3) and investigate the impact of the regularization strength (Sec. 3.3.4) and the depth of the hidden representations (Sec. 3.3.5).

3.2 UNCERTAINTY FROM INFORMATIVE REPRESENTATIONS

This section first introduces our method - i.e. how uncertainty is quantified and hidden representations are regularized (Sec. 3.2.1). Subsequently, we motivate the proposed uncertainty estimate as well as the underlying layer choice and further discuss its relation to aleatoric uncertainty (Sec. 3.2.2). Finally, we discuss how the method can be implemented in practice (Sec. 3.2.3).

3.2.1 *Method*

3.2.1.1 *Uncertainty Quantification*

This section introduces uncertainty quantification based on the hidden representations of a NN. Typically, novel data can lead to the observation of different activation patterns compared to training time. Thus, this work quantifies the uncertainty of a NN using the distribution of hidden representations observed at training time. Therefore, consider a dataset $\mathcal{D} = \{\mathbf{X}, \mathbf{Y}\}$ and a NN which is trained on the discriminative task of estimating the conditional distribution $p(\mathbf{y}|\mathbf{x})$. Further, the NN can be written as a composition of its L layers $f_{NN} = f_{L-1} \circ ... \circ f_0$. After training, we estimate the distribution $p(\mathbf{z}_l)$ of representations at layer $0 \leq l \leq L-1$ and use the negative log-likelihood of samples under this distribution observed at inference

time as a proxy for uncertainty. Formally, we define the uncertainty of a NN u_{NN} as

$$u^l_{NN} = -\log p(\mathbf{z}_l) \tag{3.1}$$

where the superscript l indicates that uncertainty is estimated based on the distribution of representations at the l-th layer. We motivate this definition in a subsequent analysis where we also further discuss the layer choice before providing empirical evidence in the experiment section.

3.2.1.2 *Enforcing Informative Representations*

An obstacle for quantifying uncertainty based on hidden representations is *feature collapse* [55]. This results in hidden representations that are not informative about the input to a NN and can lead to OOD data being indistinguishable from ID data. While related work countered feature collapse by enforcing distance awareness or using contrastive pretraining, this work ensures informative representations by reconstructing the input given the hidden representations. When estimating uncertainty based on the hidden representations $\mathbf{z}_l$ at layer l we augment the original training objective $\mathcal{L}_{orig}$ such that we optimize

$$\mathcal{L} = \mathcal{L}_{orig} + \frac{\lambda}{D}||\mathbf{x} - \hat{\mathbf{x}}||^2_2 \tag{3.2}$$

where $\mathbf{x} \sim \mathbf{X}$ and $\hat{\mathbf{x}} = f_{dec}(f_{l-1} \circ ... \circ f_0(\mathbf{x}))$ using a separate, disposable NN f_{dec} to decode the input given $\mathbf{z}_l$. Further, λ denotes a hyperparameter and D the dimensionality of $\mathbf{x}$.

3.2.2 *Analysis*

3.2.2.1 *Motivating the Log-Likelihood of Hidden Representations as a Proxy for Uncertainty*

There are two sources of uncertainty in the predictions of a NN [10, 104]. Firstly, there exists inherent noise in the training data - aleatoric

uncertainty - which can be incorporated into the model. Secondly, epistemic uncertainty arises from a lack of information or an inadequate model. A common test case for the latter is that it should increase when exposing the NN at inference time to OOD data. However, OOD detection alone is not a sufficient evaluation for uncertainty estimation methods. This becomes apparent when considering that OOD detection represents a model-agnostic task and uncertainty should be aligned with a model's ability to generalise. High uncertainty is only expected for those OOD samples that lead to worse model performance, while such OOD samples to which the model generalises well are expected to have lower uncertainty. All in all, we expect the second type of uncertainty to be related to the training data distribution $p(\mathbf{x})$ as well as the weights θ of the NN since it needs to reflect its generalization ability. With this in mind, consider the distribution of hidden representations $p(\mathbf{z}_l)$ at layer l observed on the training data distribution $p(\mathbf{x})$. Assuming a deterministic NN and given an input $\mathbf{x}'$, we can establish the following relationship between our uncertainty estimate $u_{NN}(\mathbf{x}')$, the data distribution $p(\mathbf{x})$ and a NN's parameters θ:

$$\begin{aligned} u^l_{NN}(\mathbf{x}') &= -\log p(\mathbf{z}_l) \\ &= -\log \left(\int p_\theta(\mathbf{z}'_l|\mathbf{x}) p(\mathbf{x}) d\mathbf{x} \right) \\ &= -\log \left(\int \delta(\mathbf{z}'_l - f_l \circ ... \circ f_0(\mathbf{x})) p(\mathbf{x}) d\mathbf{x} \right) \end{aligned} \tag{3.3}$$

where $p_\theta(\mathbf{z}_l|\mathbf{x})$ denotes the conditional distribution parameterized by the NN evaluated up to layer l. We exploited that for deterministic mappings $p_\theta(\mathbf{z}_l|\mathbf{x})$ represents a delta distribution. Thus, $p(\mathbf{z}_l)$ can be viewed as the training data distribution through the lens of the NN - i.e. the information in $p(\mathbf{x})$ that is regarded as relevant by the learning algorithm for solving the underlying discriminative task. Consequently, $p(\mathbf{z}_l)$ contains the necessary ingredients - dependency on θ and $p(\mathbf{x})$ - for estimating epistemic uncertainty.

3.2.2.2 *Information-theoretic View on the Layer Choice*

Defining the uncertainty of a NN according to Eq. 3.1 raises the problem of choosing a layer. First, consider the data distribution $p(\mathbf{x})$. Using the negative log-likelihood $-\log(p(\mathbf{x}))$ as an uncertainty estimate corresponds to solving the task of OOD detection which can be achieved using the entire information present in the data distribution. This information can be quantified as the entropy $H(p(\mathbf{x})) = E_{x\sim p(\mathbf{x})}\left[-\log(p(\mathbf{x}))\right]$. Assuming an underlying deterministic NN, we can make use of the data processing inequality and write

$$H(p(\mathbf{x})) \geq H(p(\mathbf{z}_0)) \geq ... \geq H(p(\mathbf{z}_{L-1})) \tag{3.4}$$

where $p(\mathbf{z}_l) = p(f_l \circ ... \circ f_0(\mathbf{x}))$ denotes the distribution of representations at layer l when evaluating the NN on $p(\mathbf{x})$. Thus, uncertainty estimates of deeper layers are based on less information and, thus, are expected to perform worse at the sole task of OOD detection. We demonstrate this in Sec. 3.3.5. On the other hand, uncertainty estimates derived from deeper layers depend more on the parameters of the NN. Sec. 3.3.3.3 and Sec. 3.3.5 show that this coincides with better alignment with a model's generalization.

3.2.2.3 *Relation to the Predictive Distribution and a Model's Aleatoric Uncertainty*

We now establish that the distribution of hidden representations can in principle be used to extract aleatoric uncertainty. Note, that we not explicitly incorporate this finding into the proposed method. Typically, the predictive distribution $p(\mathbf{y}|\mathbf{x})$ of a NN can be used to quantify aleatoric uncertainty [8] - e.g. using the softmax entropy. Reusing the assumption of a deterministic neural network of L layers, we can write $\forall l \in [0, ..., L]$

$$\begin{aligned} p(\mathbf{y}|\mathbf{x}) &= \int p(\hat{\mathbf{y}}|\mathbf{z'}_l)p(\mathbf{z'}_l|\mathbf{x})d\mathbf{z'}_l \\ &= \int p(\hat{\mathbf{y}}|\mathbf{z'}_l)\delta(\mathbf{z'}_l - f_l \circ ... \circ f_0(\mathbf{x}))d\mathbf{z'}_l = p(\mathbf{y}|\mathbf{z}_l) \end{aligned}$$

where we used that for deterministic mappings $p(\mathbf{z}|\mathbf{x})$ represents a delta distribution. Further, using Bayes' formula yields $p(\mathbf{y}|\mathbf{x}) = p(\mathbf{y}|\mathbf{z}_l) = \frac{p(\mathbf{z}_l|\mathbf{y})p(\mathbf{y})}{p(\mathbf{z})}$. Thus, by estimating the output conditional distribution $p(\mathbf{z}_l|\mathbf{y})$ - e.g. using a separate GMM for each class - and the marginal distribution $p(\mathbf{y})$ - e.g. by counting class frequencies - one is theoretically able to recover the predictive distribution using the distribution of hidden representations. We provide experimental evidence in the supplement. However, the quality of such uncertainty is upper bounded by the entropy of the softmax in our experiments. Therefore, it is recommended to estimate aleatoric uncertainty directly using the conditional distribution $p(\mathbf{y}|\mathbf{x})$ parameterized by the NN (e.g. softmax entropy) if possible. We note that for some classic tasks, such as regression or bounding box prediction, direct parameterization of $p(\mathbf{y}|\mathbf{x})$ is uncommon and alternative approaches for aleatoric uncertainty estimation can be useful. In summary, this section shows that in addition to the model uncertainty of Eq. 3.1, also aleatoric uncertainty can be estimated from $\mathbf{z}_l$, even tough it is more of theoretical than practical relevance.

3.2.3 *Implementation*

3.2.3.1 *Pseudocode*

Alg. 1 describes the estimation of the distribution of hidden representations. After training with objective Eq. 3.2 and a decoder f_{dec}, we fit a density model $p_\omega(\mathbf{z}_l)$ with parameters ω to the distribution $p(\mathbf{z}_l)$ of representations observed when evaluating on the training data. Note that we normalize representations to unit length prior to estimating the distribution which we observe empirically to improve performance (see Sec. 3.3.2). At inference time we do not use the decoder. Given novel input $\mathbf{x}'$, we rather perform a normal forward pass of the NN yielding predictions $p(\mathbf{y}|\mathbf{x}')$ and the representations $\mathbf{z}'_l$. Subsequently, we normalize $\mathbf{z}'_l$ and uncertainty is estimated by evaluating Eq. 3.1.

Algorithm 1 Learning the probablity density of hidden representations at layer l.

Input: Randomly initialized NN of L layers, training data $\mathcal{D} = \{\mathbf{X}, \mathbf{Y}\}$, layer index $l \in [1, ..., L-]$
Output: Trained NN, density $p_\omega(\mathbf{z}_l)$

1: Initialize decoder f_{dec} for reconstructing $\mathbf{x}$ given $\mathbf{z}_l$
2: **while** not converged **do**
3: Train NN using objective in Eq. 3.2
4: **end while**
5: Initialize density model $p_\omega(\mathbf{z}_l)$ with parameters ω
6: $\mathbf{Z}_l \leftarrow$ Evaluate NN on training data $\mathbf{X}$
7: Normalize entries in $\mathbf{Z}_l$ to unit length
8: Fit $\mathbf{Z}_l$ using density model $p_\omega(\mathbf{z}_l)$
9: **return** NN, $p_\omega(\mathbf{z}_l)$

3.2.3.2 *Density Estimation*

We distinguish between classification and regression. In the case of classification we estimate the class-conditional distribution $p(\mathbf{z}_l|\mathbf{y})$ using a single multivariate Gaussian with full covariance per class. $p(\mathbf{z}_l)$ is obtained by marginilization over $\mathbf{y}$. In line with prior work we find that representations following an affine transformation of each class are well represented by a multivariate Gaussian. We hypthesize that this is, firstly, due to the classification training enforcing clusters in the feature space and, secondly, a byproduct of the central limit theorem. When handling NNs trained on regression we find that multivariate Gaussians are not sufficient for modeling $p(\mathbf{z}_l)$. Thus, for regression models we estimate $p(\mathbf{z}_l)$ using a NF based on coupling layers [28]. For more information regarding NFs we refer to [28] and the supplement.

3.2.3.3 *Normalized Representations*

We find for classification that normalizing hidden representations $\mathbf{z}_l$ to unit length before estimating $p(\mathbf{z}_l)$, improves OOD detection. Thus, the orientation encodes information for distinguishing between ID and OOD data. We demonstrate this empirically in our experiments.

3.3 EXPERIMENTS

We evaluate the proposed approach. Firstly, we measure its performance on OOD detection (Sec. 3.3.2). Moreover, we show that our approach can also be applied to regression problems (Sec. 3.3.3). Finally, we perform two sensitivity analyses - on the impact of the regularization strength (Sec. 3.3.4) and on the depth of the layer used for uncertainty estimation (Sec. 3.3.5). Further, we investigate the behaviour on continuous distributional shifts using the corrupted versions of CIFAR10/100 (CIFAR10/100-C) [106] (Section 3.3.3.3).

3.3.1 *Experimental Setup*

3.3.1.1 *Baselines*

We compare our method with well-established variants of BNNs - Monte-Carlo (MC) dropout [16] and deep ensembles [11]. While these methods yield good uncertainty estimates, they require multiple forward passes for a single inference. Further, we compare our work with two other recent methods that estimate uncertainty using a single-forward pass - DUQ [55] and SNGP [26]. These methods are most relevant to this work since they also treat the weights of the NN deterministically and explicitly regularize the hidden representations of a NN to be informative about its input.

Method	Trained on MNIST				Trained on FashionMNIST			
	Acc.	FashionMNIST	Omniglot	$\frac{T}{T_0}$	Acc.	MNIST	Omniglot	$\frac{T}{T_0}$
MCD	0.981	0.914	0.919	-	0.886	0.881	0.882	-
DE	**0.987**	0.924	0.977	-	**0.904**	0.896	0.917	-
DUQ	0.968	0.879	0.913	180%	0.860	0.930	0.941	188%
SNGP	0.985	0.953	0.957	125%	0.891	0.901	0.902	128%
Ours*	0.982	0.967	0.932	114%	0.883	0.966	**0.964**	125%
Ours	0.982	**0.984**	**0.983**	114%	0.885	**0.985**	0.963	125%

TABLE 3.1: Comparing the OOD detection performance of MC dropout (MCD), deep ensembles (DE), DUQ, SNGP and our method (Ours) in terms of AUROC and relative per sample training runtime ($\frac{T}{T_0}$, T_0: runtime of vanilla softmax model) with prior work when training a multi-layer perceptron on MNIST/FashionMNIST. On MNIST/FashionMNIST we use FashionMNIST/MNIST and Omniglot as OOD data. Our method is evaluated with (Ours) and without (Ours*) normalization of hidden representations.

Method	Trained on SVHN					Trained on CIFAR10				
	Acc.	STL10	CIFAR100	CIFAR10	$\frac{T}{T_0}$	Acc.	STL10	SVHN	CIFAR100	$\frac{T}{T_0}$
MCD	0.954	0.952	0.940	0.949	-	0.883	0.686	0.885	0.82	-
DE	**0.966**	**0.974**	**0.970**	**0.980**	-	**0.916**	**0.875**	**0.937**	0.758	-
DUQ	0.946	0.938	0.917	0.928	320%	0.859	0.633	0.843	0.766	292%
SNGP	0.957	0.936	0.923	0.928	136%	0.61	0.74	0.795	0.686	154%
OUR*	0.953	0.936	0.917	0.927	112%	0.617	0.72	0.509	0.621	117%
OUR	0.953	0.965	0.952	0.959	112%	0.617	0.789	**0.809**	0.663	117%

TABLE 3.2: Comparing the OOD detection performance of MC dropout (MCD), deep ensembles (DE), DUQ, SNGP and our method (Ours) in terms of AUROC with prior work when training a ResNet50 [105] on SVHN/CIFAR10. On SVHN/CIFAR10 we use CIFAR10/SVHN, CIFAR100 and STL10 as OOD data.

Method	Trained on CIFAR100				
	Test Acc.	STL10	SVHN	CIFAR10	$\frac{T}{T_0}$
MCD	0.592	0.772	0.84	0.735	-
DE	**0.622**	**0.801**	0.741	**0.756**	-
DUQ	-	-	-	-	351%
SNGP	0.61	0.74	0.795	0.686	154%
OUR*	0.617	0.72	0.509	0.621	117%
OUR	0.617	0.789	**0.809**	0.663	117%

TABLE 3.3: OOD detection performance of MC dropout (MCD), deep ensembles (DE), DUQ, SNGP and our method (Ours) in terms of AUROC with prior work when training a ResNet50 on CIFAR100. We use CIFAR10, SVHN and STL10 as OOD data.

3.3.1.2 *Neural Network Architectures*

Each method uses the same backbone architecture. When training on MNIST [107], FashionMNIST [108] and the regression tasks (see Sec. 3.3.3) the backbone is a MLP of 4-layers with ReLU activation functions and 100 hidden units. In the case of CIFAR10/100 [109] and SVHN [110] we train a ResNet-50 [105]. Note that for the ablation on the depth of the hidden representations used for uncertainty estimation, we train a ResNet-8. Method-specific final layers (e.g. RBF-kernel (DUQ)) are applied to the backbone's output. Full details are in the supplement.

3.3.1.3 *Decoder Architectures*

For the MLP backbone the decoder is a MLP of 2-layers with ReLU activation functions and 100 hidden units taking the output of the backbone model as an input. The final layer has the dimensionality of the input and a linear activation function. In the case of the ResNet-50

backbone the decoder is a reverse ResNet-8 replacing convolutions with transpose convolutions and reversing the number of channels. Its input is the penultimate layer of the backbone model (i.e. the output of that last layer prior to global average pooling). Full details are in the supplement.

3.3.1.4 *Hyperparameter Optimization*

We choose method-specific hyperparameters using grid search and maximization of performance on the validation set. For our method we optimize the weighting of reconstruction regularization λ_{our} (see Eq. 3.2). In the case of DUQ, resp. SNGP, we optimize the weighting of the gradient penalty λ_{DUQ}, resp. the upper bound c on the spectral norm. Exact values of the grid search and further optimization details are in the supplement. In case of the ResNet backbone we use $\lambda_{our} = 1.0$ and $c = 7$, for the MLP we choose $\lambda_{our} = 100.0$ and $c = 3$. In all cases $\lambda_{DUQ} = 10^{-6}$ led to the best validation performance.

3.3.1.5 *Estimating the Distribution of Hidden Representations*

We use a class-wise multivariate Gaussian in the case of classification and apply NFs based on coupling layers [28] to models trained on regression. All experiments, except Sec. 3.3.5, use the output of the penultimate layer (prior to any applied activation functions), i.e. the output of the backbone model, for uncertainty estimation based on Eq. 3.1. The reason for relying on deep representations for estimating uncertainty is theoretically and empirically motivated (see Sec. 3.2.2 and Sec. 3.3.5)

3.3.1.6 *Statistical Fluctuation of Results*

Results are averaged over 5 independent repetitions. In those cases where the standard deviation is not reported in the main text, it can be found in the supplement.

3.3.2 *Out-of-Distribution Detection*

We evaluate the OOD detection performance of each method by training on MNIST/FashionMNIST/CIFAR10/CIFAR100/SVHN and evaluating on several OOD datasets. We report Area Under the Receiver Operating Characteristic (AUROC) obtained when separating the testset of ID and OOD data. Note that the running mean of class centroids maintained by DUQ [55] throughout training led to training instabilities on CIFAR100 which contains only 600 samples per class. Tab. 3.1, 3.2 and 3.3 shows the results of this experiment. Our method is evaluated with (Ours) and without (Ours*) normalization of hidden representations. While single-forward pass methods, in particular our approach and SNGP, clearly outperform MC dropout, deep ensembles remain a competitive baseline. Moreover, we find that on OOD detection, our approach obtains the best performance among the methods estimating uncertainty using a single forward pass, outperforming DUQ and SNGP in 10 out of 13 scenarios. Moreover, we verify the importance of normalizing hidden representations prior to density estimation (Our) which clearly outperforms our method in the absence of normalization (Ours*) on OOD detection. Lastly, we also observe that our method further reduces the per sample runtime during training ($\frac{T}{T_0}$) compared to SNGP and DUQ which helps scaling it to larger problems.

3.3.3 *Regression*

3.3.3.1 *Regression Toy Example*

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. 3.2. We train on the gray outlined data with a simple four-layer MLP with ReLU activation functions, extract embeddings at the penultimate layer (following our findings from section 3.2 and 4.6 and fit the density with a NF. The same architecture is used for the ensemble. The GP has a RBF kernel with parameters fit to the

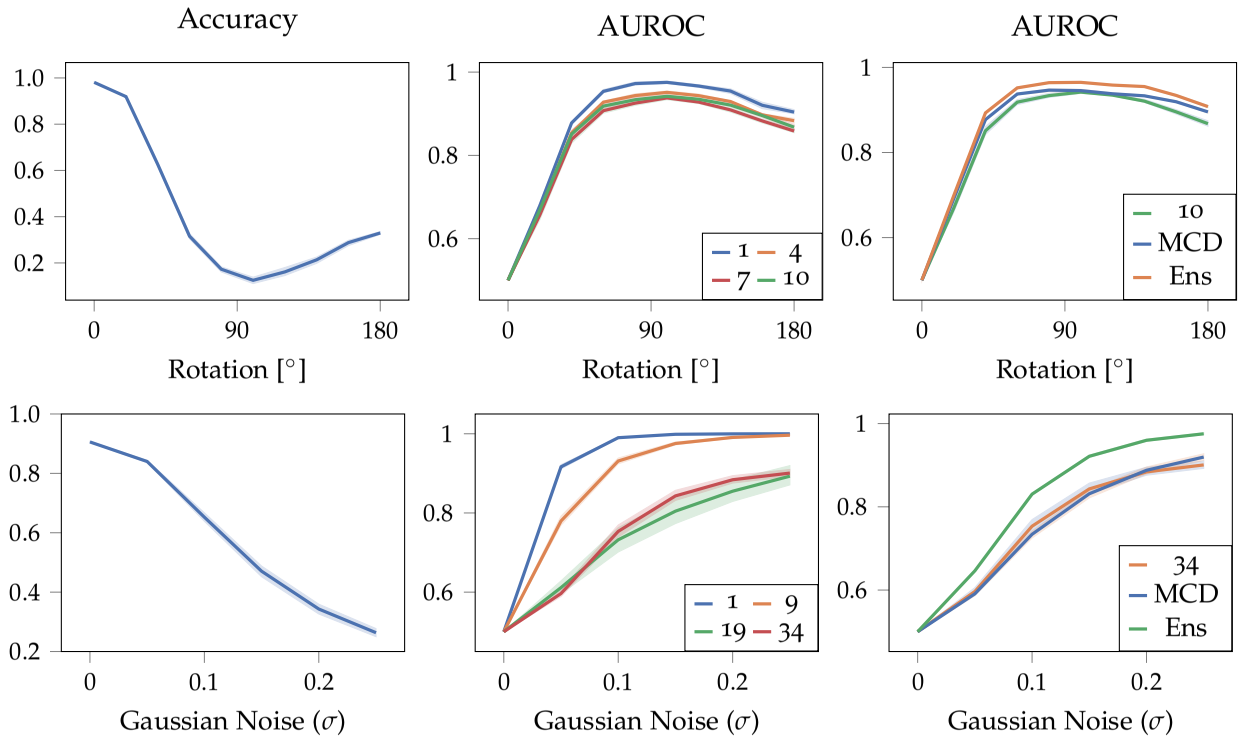


FIGURE 3.1: We plot the classification accuracy (left) and the AUROC against the perturbation magnitude on the test data (center and right). We compare uncertainty obtained from hidden representations of varying depth (center) and other methods (right). We provide the index of each layer as an indicator of its depth and refer to the supplement for index-to-layer mapping. We observe that (i) uncertainty estimates based on deeper layers qualitatively match established - but computationally more expensive - methods (i.e. MC dropout and deep ensembles) and (ii) those based on shallow layers yield a sharper increase in AUROC.

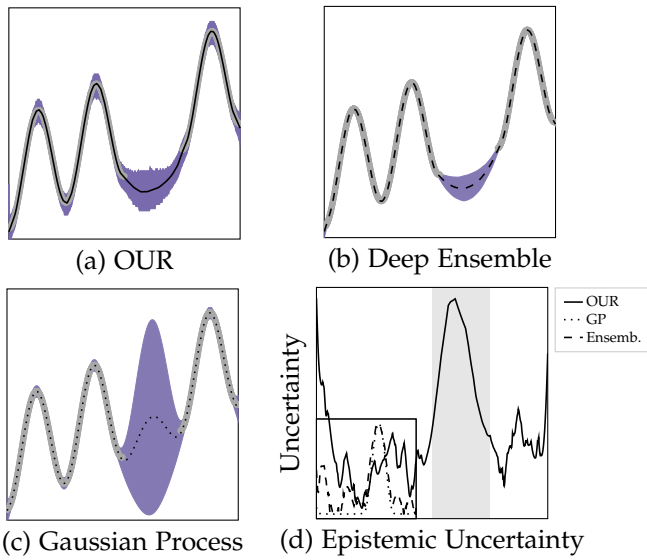


FIGURE 3.2: The models are trained on the data outlined in gray. **(a-c):** Model prediction is given in black and the high-confidence area in blue. For better comparison, **(d)** shows the normalised estimated epistemic uncertainty with the data gap in gray. We observe that all methods follow the expected 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 our method.

Data	OUR	Deep Ensemble
Boston Housing	93.45 ± 1.64	92.11 ± 0.68
SAS Diabetis	65.21 ± 2.87	63.53 ± 1.30
Openml Cholesterol	68.50 ± 1.69	67.67 ± 1.06

TABLE 3.4: OOD detection (AUROC) on several regression datasets for deep ensembles and our method (OUR).

training data. We observe that the uncertainty of deep ensembles, GP and our method demonstrates similar qualitative behaviour.

3.3.3.2 *Real Data*

We evaluate OOD detection on standard regression datasets and report results in Tab. 3.4. Lacking clear OOD splits, we remove the patient's sex from [111, 112] and use it to split the data into two slightly different distributions. Since the (unshuffled) [113] 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 with our method matches it.

3.3.3.3 *Continuous Distributional Shifts*

We have shown that our method detects strong distributional shifts (see Sec. 3.3.2). Another important property is the ability to deduct expected model performance from uncertainty. This experiment evaluates this qualitatively. Therefore, we evaluate models trained on CIFAR10/100 [106] on CIFAR10/100-C and report accuracy and AUROC measured when separating the clean testset from its corrupted counterparts depending on the corruption severity (0: no corruption - 5: strongest corruption). In contrast to similar experiments in [114], we measure AUROC rather than expected calibration error or the Brier score since the uncertainty of our method is not incorporated in a probabilistic forecast. Fig. 3.5 and 3.6 show the results for deep

ensembles [11], DUQ [55], SNGP [26] and our method. Since the optimal AUROC curve given a degradation in model performance is unknown, it is impossible to rank individual methods based on this experiment. However, we can draw a few conclusions. Firstly, the AUROC obtained using SNGP and our method tends to follow a similar trajectory as the one obtained from deep ensembles. Since deep ensembles corresponds to a well-established baseline with vast empirical evidence of the quality of its uncertainty (e.g. [11, 114]), it is promising that we can obtain similar behaviour/calibration using a single-forward pass. Surprisingly the negative log-likelihood of deep features - our uncertainty proxy (see Eq. 3.1) - demonstrates similar characteristics close to the data distribution as deep ensembles. This is a non-trivial outcome which supports the hypothesis made in Sec. 3.2.2 that the weight dependence of the distribution of hidden representations enables uncertainty estimation.

3.3.4 *The Impact of Regularization Strength*

We ablate the impact of the regularization strength on our method. Therefore, we fit an MLP on MNIST and a ResNet8 on SVHN and evaluate the models on distributional shifts - rotation (MNIST) and additive Gaussian noise (SVHN). Here, we use various strengths of the reconstruction regularization strength λ. We plot the resulting accuracy and AUROC against the perturbation magnitude. The result can be found in Fig. 3.3. We observe that increasing λ consistently lifts the AUROC curve. This solidifies the insight that our regularization technique consistently increases the sensitivity to OOD data.

3.3.5 *Choice of Layer Depth for Hidden Representations*

In order to focus purely on the impact of depth of the hidden representations used for uncertainty estimation, we set $\lambda_{OUR} = 0$ for this ablation study. We train our method on MNIST and SVHN and, subsequently, evaluate it on continuously shifted data. We rotate

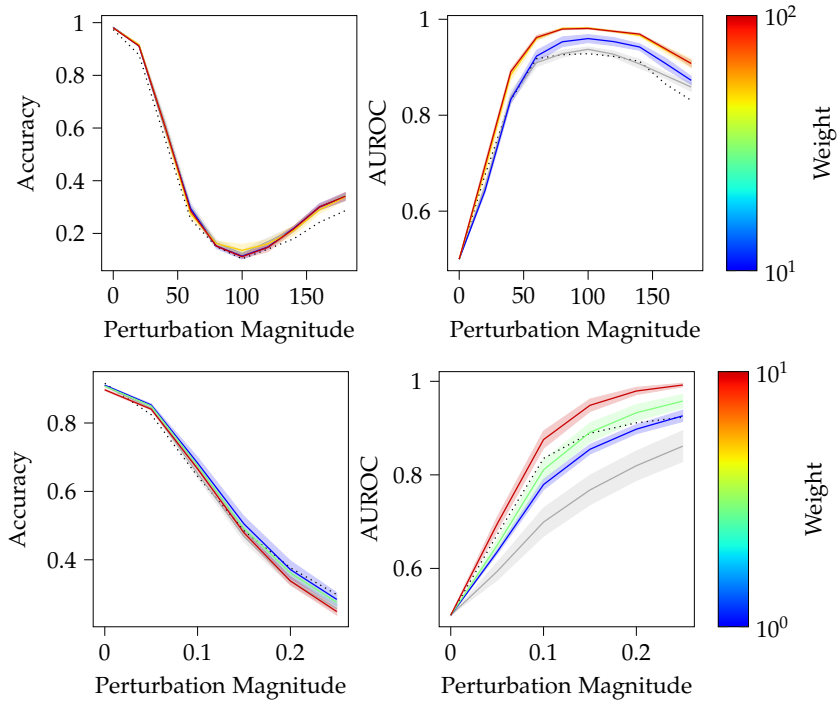


FIGURE 3.3: Accuracy and AUROC on rotated MNIST (top) and SVHN with additive Gaussian noise (bottom) for various perturbation magnitudes and reconstruction weights. We reconstruct and estimate uncertainty from layers 10 (MNIST) and 19 (SVHN). The baseline without reconstruction loss is shown in gray. For comparison, DUQ [55] is plotted as dotted line. The AUROC increases with larger λ while the accuracy does not deviate much from the baseline.

Trained on	OOD Data	L 1	L 4	L 7	L 10
MNIST	FashionMNIST	**0.975**	0.922	0.855	0.811
	omniglot	0.972	0.937	0.892	0.893
FashionMNIST	MNIST	0.985	**0.991**	0.975	0.978
	omniglot	0.971	**0.987**	0.960	0.967
	OOD Data	L 1	L 9	L 19	L 34
SVHN	CIFAR10	**0.991**	0.974	0.934	0.907
	STL10	**0.999**	0.991	0.951	0.912
CIFAR10	SVHN	0.042	0.029	0.091	**0.736**
	STL10	0.790	**0.871**	0.821	0.651

TABLE 3.5: AUROC for OOD Detection Based on Uncertainty from Hidden Representations of a MLP (MNIST/FashionMNIST) and ResNet8 (SVHN/CIFAR10) from Layers (L) of Varying Depth.

MNIST digits between 0°and 180°in steps of 20°. SVHN is perturbed using zero-mean additive Gaussian noise of increasing standard deviation $\sigma \in \{0, 0.05, 0.1, 0.15, 0.2, 0.25\}$. We investigate the accuracy on the testset and the AUROC when separating unperturbed test data from its perturbed counterpart while estimating uncertainty based on the distribution of representation of different depth. Fig. 3.1 depicts the results. While shallow layers facilitate a sharper rise in AUROC, deeper layers behave similar to MC dropout and deep ensembles. We conclude that shallow layers potentially enable better OOD detection. However, deeper layers yield meaningful uncertainty tailored to the underlying model. Moreover, we evaluate models trained on MNIST, FashionMNIST, SVHN and CIFAR10 regarding OOD detection performance depending on the depth of the layer used for uncertainty estimation. Note that since shallow layers yield high-dimensional representations which can lead to difficulties estimating their distribution, we reduce their dimensionality using PCA in this experiment. We refer to the supplement for details. Tab. 3.5 shows the results of

this experiment. In line with prior outcomes, we generally observe that shallow layers tend to enable better OOD detection, except when training on CIFAR10 and evaluating on SVHN. The latter is a known problem of explicit generative models used for OOD detection [64].

3.3.6 *Aleatoric Uncertainty from the Distribution of Hidden Representations at various Depth*

We evaluate models trained in Sec. 3.3.5 on the unperturbed testsets of MNIST and SVHN. Subsequently, we estimate aleatoric uncertainty using the output-conditional distribution of hidden representations based on layers of different depth to verify our analysis in Sec. 3.2.2 and compare it to aleatoric uncertainty obtained using the softmax entropy, Monte-Carlo Dropout and Deep Ensembles. Therefore, we plot the accuracy on the unperturbed testset of all samples with aleatoric uncertainty smaller than some percentile of uncertainty against the percentile. In case the distribution of hidden representations yields correct aleatoric uncertainty, we expect two things: 1) The resulting plot is a monotonically decreasing function and 2) our method yields a similar plot as the baselines. Fig. 3.4 shows the result. We observe that aleatoric uncertainty based on the distribution of deeper layers is indeed a monotonically decreasing function and is similar to the one obtained from the softmax entropy, MC Dropout and Deep Ensembles. However, this is not the case for estimates obtained from shallower layers.

3.4 CONCLUSION

We introduced a general framework for uncertainty estimation based on the distribution of hidden representations and verified its effectiveness empirically for classification (Sec. 3.3.2) and regression (Sec. 3.3.3). Feature collapse is prevented by enforcing informative representations using novel reconstruction regularization (Sec. 3.2.1.2). While this approach demonstrates strong performance on OOD de-

tection (Sec. 3.3.2), it also yields uncertainty that behaves similar epistemic uncertainty predicted by deep ensembles [11] (Sec. 3.3.3.3) at a fraction of the computational cost. Interestingly, we establish that in particular the distribution of deep features in a NN allows estimating uncertainty according to Eq. 3.1 that correlates with the epistemic uncertainty of deep ensembles.

The newly proposed regularization technique - reconstruction regularization - for learning informative representations in Eq. 3.2 is more efficient during training and task/architecture-agnostic compared to enforcing distance-aware representations [26, 55]. Further, we empirically observe that reconstruction regularization, unlike enforcing distance-aware hidden representations, correlates with OOD detection performance (Sec. 3.3.4) making it possible to reason about the regularization strength in practice.

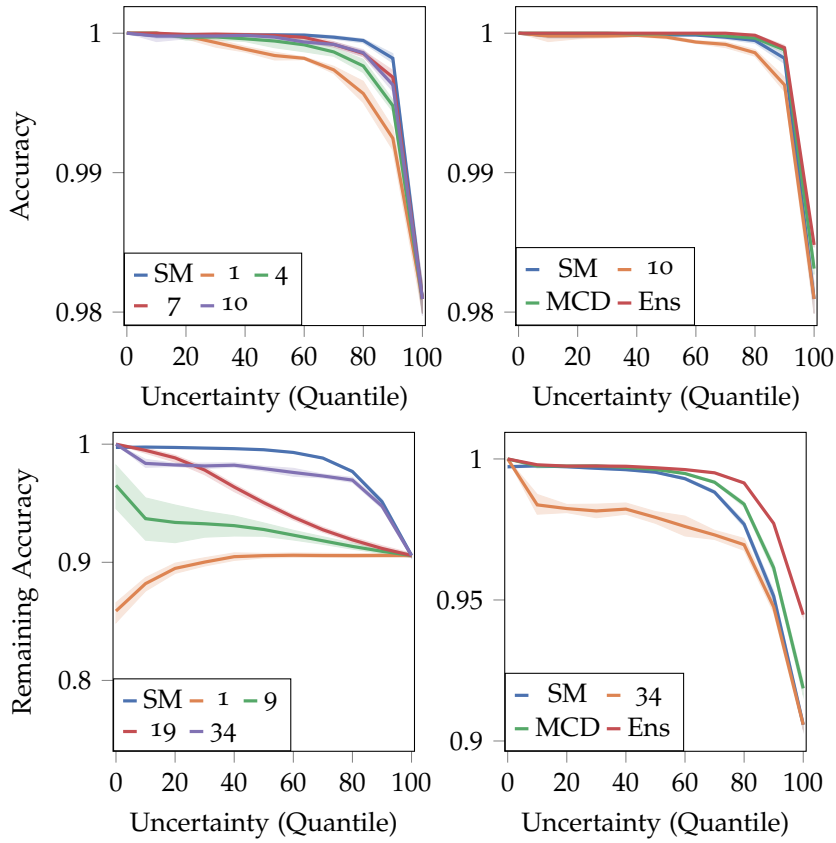


FIGURE 3.4: Calibration of aleatoric uncertainty on MNIST (top) and SVHN (bottom). We plot the remaining accuracy of test samples below an increasing aleatoric uncertainty threshold. We train an MLP on MNIST and compare aleatoric uncertainty from layer 1, 4, 7, 10 and softmax entropy (top left) and from layer 10, softmax entropy, MC dropout and deep ensembles (top right). We train a ResNet8 on SVHN and compare aleatoric uncertainty from layer 1, 9, 19, 34 and softmax entropy (bottom left) and from layer 34, softmax entropy, MC dropout and deep ensembles (bottom right). An index-to-layer mapping is in section 3.5.C. We observe that aleatoric uncertainty based on deeper layers tends to behave similar to the one from other approaches (softmax entropy, deep ensembles, MC dropout).

3.5 APPENDICES

The supplementary material is structured in the following way: Sec. 3.5.A provides additional explanations regarding concepts used in the main paper. Sec. 3.5.B provides additional experimental results as well as standard deviations missing in the main paper. Finally, Sec. 3.5.C describes architectures, optimization and other details used in our experiments.

3.5.A *Background*

3.5.A.1 *Normalizing Flows*

NFs [27, 28] are a family of explicit generative models based on invertible neural networks. Their optimization is based on Eq. 1.3. We apply NFs based on the coupling layer introduced by [28]. 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 NF 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 [115], 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:

$$u_1^{out} = u_1^{in}$$
$$u_2^{out} = \left(u_2^{in} + g_t\left(u_1^{in}; c\right)\right) \odot g_s\left(u_1^{in}; c\right)$$

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 $g_t\left(u_1^{in}\right)$ and $g_s\left(u_1^{in}\right)$.

3.5.A.2 *Estimating Aleatoric and Epistemic Uncertainty with Deep Ensembles and Monte-Carlo Dropout*

We calculate aleatoric and epistemic uncertainty according to Eq. 1.1 and Eq. 1.2.

3.5.B *Additional Experimental Results*

Fig. 3.5 and Fig. 3.6 depict the accuracy and AUROC of deep ensembles, DUQ, SNGP and OUR method on the CIFAR10/100-C.

3.5.C *Experimental Details*

Hardware

All experiments were run on a single Tesla V100 GPU.

3.5.C.1 *Optimization Details: MLP*

We use Adam optimizer [116] ($\beta_1 = 0.9$, $\beta_2 = 0.999$), a batch size of 128 and a dropout rate of 0.1 in every experiment. Furthermore, we train every model for 200 epochs. We keep the model that yields best performance on the validation loss.

MC Dropout, Deep Ensembles & Our Method We use a fix learning rate of 0.003. We apply weight decay with a weight of 0.0001. We use

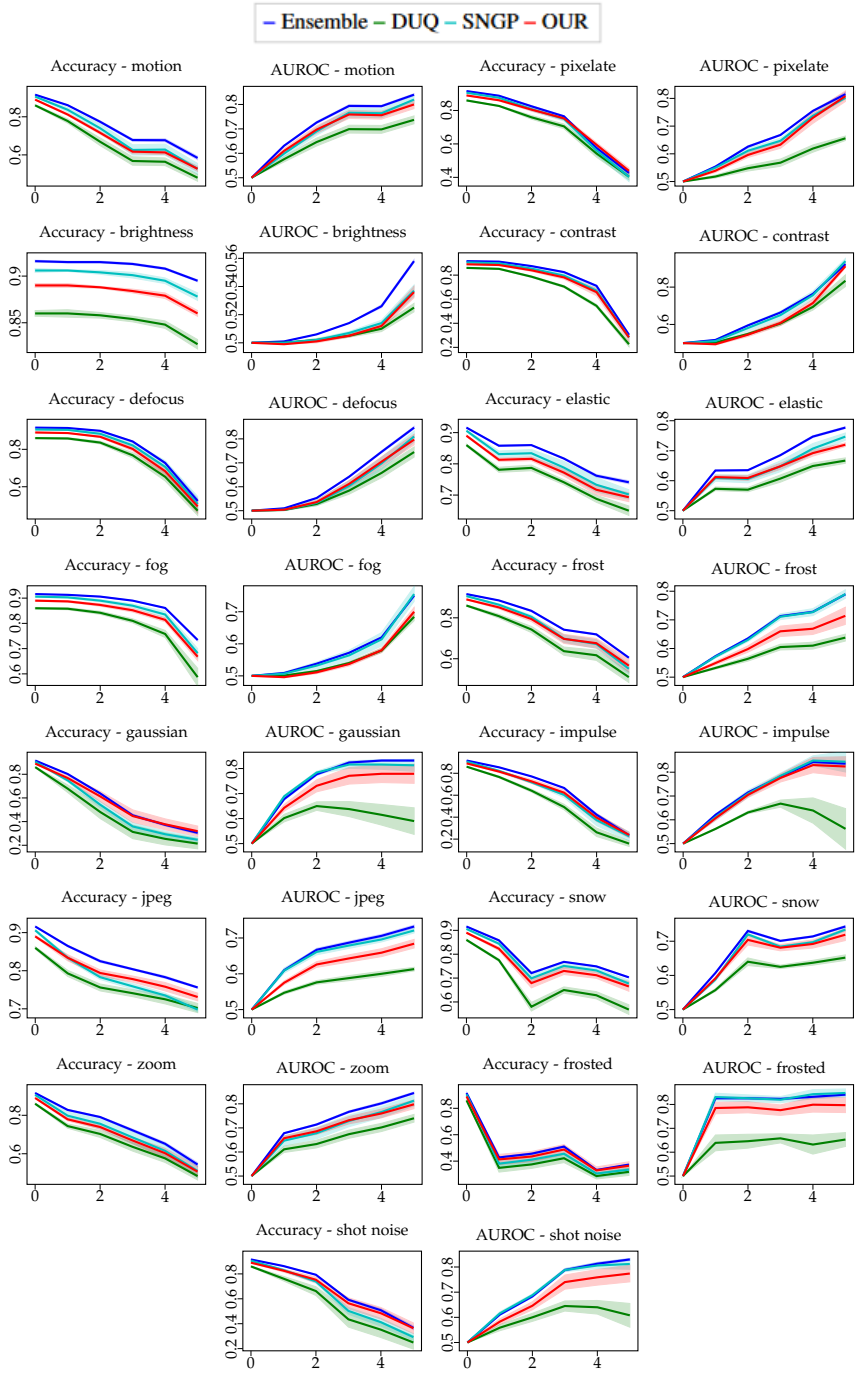


FIGURE 3.5: Accuracy and AUROC on the corruptions of CIFAR10-C.

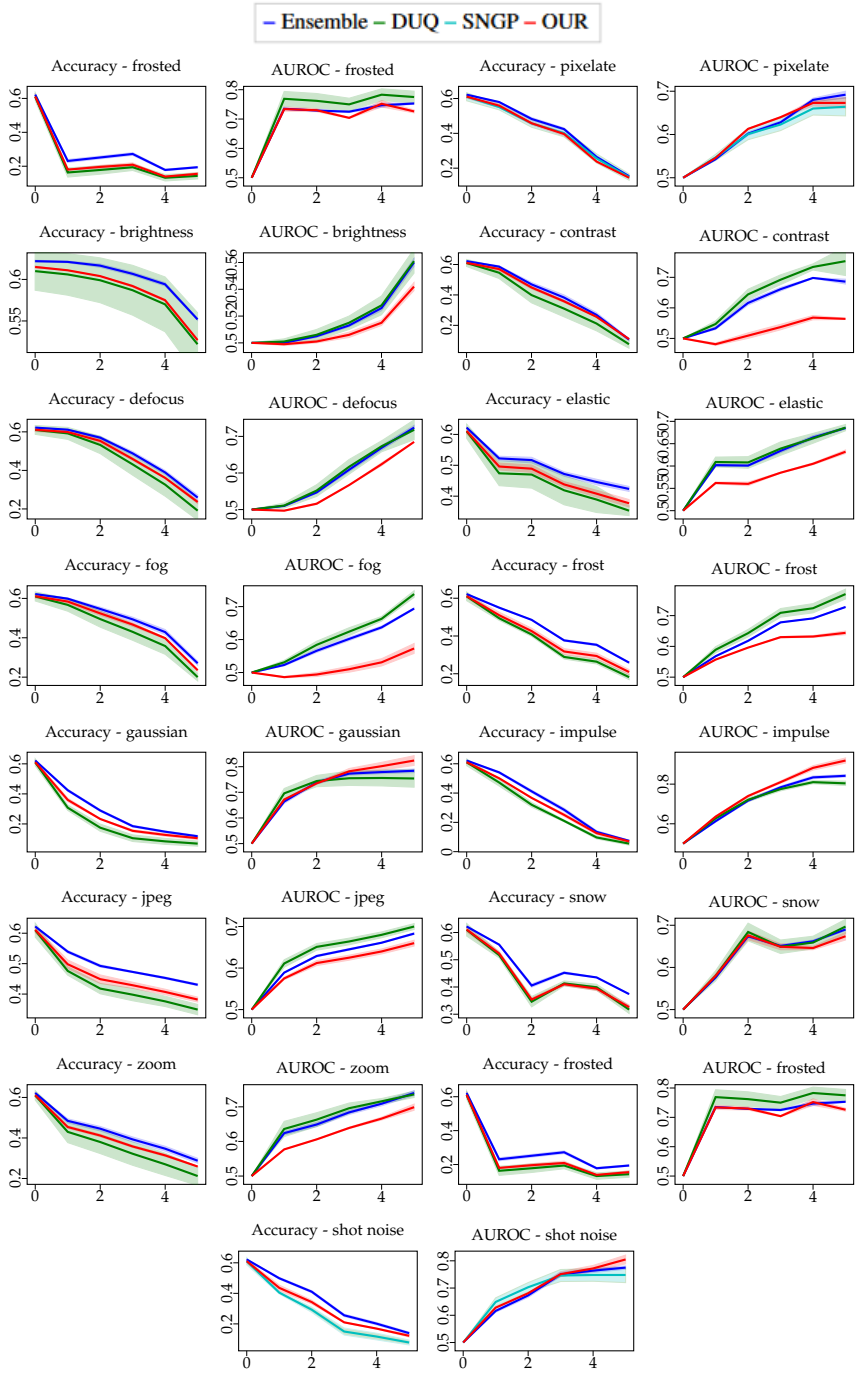


FIGURE 3.6: Accuracy and AUROC on the corruptions of CIFAR100-C.

ensembles with 10 models and run MC dropout with 50 independent forward passes.

SNGP We start with an initial learning rate of 0.05. We decrease the learning rate after 60, 120 and 160 steps using a multiplicative factor $\gamma = 0.2$. Instead of the Laplace-approximated GP we follow the official implementation and use a random feature GP with 128 inducing points and a GP scale of 2.0. Following the official implementation we use soft spectral normalization with 1 power iteration. We apply weight decay with a weight of 0.0004.

DUQ We start with an initial learning rate of 0.01. We decrease the learning rate after 60, 120 and 160 using a multiplicative factor $\gamma = 0.2$. We use a length scale of 0.1. We apply weight decay with a weight of 0.0001.

3.5.c.2 *Optimization Details: ResNet*

We use Adam optimizer [116] ($\beta_1 = 0.9$, $\beta_2 = 0.999$), a batch size of 128 and a dropout rate of 0.05 in every experiment. Furthermore, we train every model for 400 epochs. We keep the model that yields best performance on the validation loss. Further, we apply data augmentation during training on CIFAR10, SVHN and CIFAR100. We apply random flipping along the vertical axis and random brightness/contrast changes.

MC Dropout, Deep Ensembles & Our Method We use an initial learning rate of 0.003. We decrease the learning rate after 100, 200 and 300 steps using a multiplicative factor $\gamma = 0.2$. We apply weight decay with a weight of 0.0001. We use ensembles with 10 models and run MC dropout with 50 independent forward passes.

SNGP We start with an initial learning rate of 0.05. We decrease the learning rate after 100, 200 and 300 steps using a multiplicative factor $\gamma = 0.2$. Instead of the Laplace-approximated GP we follow the official implementation and use a random feature GP with 128 inducing points and a GP scale of 2.0. Following the official implementation

we use soft spectral normalization with 1 power iteration. We apply weight decay with a weight of 0.0003.

DUQ We start with an initial learning rate of 0.03. We decrease the learning rate after 200, 250 and 300 using a multiplicative factor $\gamma = 0.2$. We use a length scale of 0.1. We apply weight decay with a weight of 0.0001.

3.5.C.3 *Optimization Details: Density of Hidden representations*

Classification: Gaussian Mixture Model We estimate the distribution of the hidden representations of each class using a single multi-variate Gaussian. We use the empirical mean and covariance matrix. We also experimented with using NFs for this task. However, we found in the case of classification that a GMM is sufficient to fit the distribution of hidden representations throughout all layers.

Dimensionality Reduction Using PCA in "Choice of Layer Depth for Hidden Representations" We reduce high-dimensional hidden representations to 256 dimensions using PCA prior to estimating the distribution of features of each class using a multi-variate Gaussian.

3.5.C.4 *Decoder Model for Reconstruction Regularization*

We provide a detailed description of the layers of the decoder models used in our experiments in Tab. 3.6 and 3.7 (left).

Index	0	1	2
Layer	Linear	ReLU	Linear

TABLE 3.6: Transposed decoder model when training an MLP on MNIST/FashionMNIST.

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	TransposeConv2d
13	Batchnorm2D
14	ReLU
15	TransposeConv2D
18	Batchnorm2D
19	Add
20	Dropout
21	ReLU
22	TransposeConv2D
23	Batchnorm2D
24	ReLU
25	TransposeConv2D
28	Batchnorm2D
29	Add
30	Dropout
31	ReLU
32	Conv2D

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

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 3.7: Left: Decoder model when training an ResNet50/8 on CIFAR10/CIFAR100/SVHN. Center: Layer index to layer type for the MLP in Sec. 3.3.5. Right: Layer index to Layer type for the ResNet8 in Sec. 3.3.4.

3.5.c.5 *Index-to-Layer Mapping for "Choice of Layer Depth for Hidden Representations"*

We provide a mapping from indices to layer types in Tab. 3.7 (center) and (right).

4

ON THE PRACTICALITY OF DETERMINISTIC EPISTEMIC UNCERTAINTY

A set of novel approaches for estimating epistemic uncertainty in deep neural networks with a single forward pass has recently emerged as a valid alternative to Bayesian Neural Networks. On the premise of informative representations, these deterministic uncertainty methods (DUMs) achieve strong performance on detecting out-of-distribution (OOD) data while adding negligible computational costs at inference time. However, it remains unclear whether DUMs are well calibrated and can seamlessly scale to real-world applications - both prerequisites for their practical deployment. To this end, we first provide a taxonomy of DUMs, and evaluate their calibration under continuous distributional shifts. Then, we extend them to semantic segmentation. We find that, while DUMs scale to realistic vision tasks and perform well on OOD detection, the practicality of current methods is undermined by poor calibration under distributional shifts.

4.1 INTRODUCTION

Despite the dramatic enhancement of predictive performance of deep learning (DL), its adoption remains limited due to unpredictable failure on OOD samples [117, 118] and adversarial attacks [119]. Uncertainty estimation techniques aim at bridging this gap by providing accurate confidence levels on a model's output, allowing for a safe deployment of NNs in safety-critical tasks, *e.g.* autonomous driving or medical applications.

While BNNs represent the predominant holistic solution for quantifying uncertainty [33, 35], exactly modelling their full posterior is often intractable, and scalable versions usually require expensive variational approximations [14–16, 36, 38]. Moreover, it has recently

been shown that true Bayes posterior can also lead to poor uncertainty [120]. Thus, efficient approaches to uncertainty estimation largely remain an open problem, limiting the adoption within real-time applications under strict memory, time and safety requirements.

Recently, a promising line of work emerged for estimating epistemic uncertainty of a NN with a single forward pass while treating its weights deterministically. By regularizing the hidden representations of a model, these methods represent an efficient and scalable solution to epistemic uncertainty estimation and to the related OOD detection problem. In contrast to BNNs, Deterministic Uncertainty Methods (DUMs) quantify epistemic uncertainty using the distribution of latent representations [20, 21, 73, 121–123] or by replacing the final softmax layer with a distance-sensitive function [19, 25, 26, 56]. Further, these methods have been applied to practical problems such as object detection [124]. While OOD detection is a prerequisite for a safe deployment of DL in previously unseen scenarios, the calibration - *i.e.* how well uncertainty correlates with model performance - of such methods under continuous distributional shifts is equally important. Measuring the calibration of an epistemic uncertainty estimate on shifted data investigates whether it entails information about the predictive performance of the model. This an essential requirement for uncertainty and, unlike OOD detection, an evaluation that is not model-agnostic - i.e. one cannot perform well without taking the predictive model into account. Nonetheless, previous work falls short of investigating calibration, and solely focuses on OOD detection [19, 21, 25, 56, 121]. Further, DUMs have thus far only been evaluated on toy datasets for binary classification, small-scale image classification tasks [19, 21] and toy prediction problems in natural language processing [26]. Despite claiming to solve practical issues of traditional uncertainty estimation approaches, the practicality of DUMs remains to be assessed on more challenging tasks.

This work investigates whether recently proposed DUMs are a realistic alternative for practical uncertainty estimation. In particular: (i) we provide the first comprehensive taxonomy of DUMs; (ii) we analyze the calibration of DUMs under synthetic and realistic contin-

uous distributional shifts; (iii) we evaluate the sensitivity of DUMs to their regularization strength; (iv) we scale DUMs to dense prediction tasks, *e.g.* semantic segmentation. Overall, we find that the practicality of many DUMs is undermined by their poor calibration under both synthetic and realistic distributional shifts. Moreover, some techniques for regularizing hidden representations demonstrate only weak correlation with OOD detection and calibration performance.

4.2 TAXONOMY FOR DETERMINISTIC UNCERTAINTY QUANTIFICATION

Since DUMs thus far denote a novel and scattered phenomenon in literature, it is necessary to provide an overview of the research landscape. Note, that we provide more material on existing DUM trends identified in this taxonomy (Sec. 4.5.A). Existing DUMs mostly differentiate along two axis. Firstly, DUMs apply different regularization techniques to equip their representations with the ability to differentiate between ID and OOD data (Sec. 4.2.1). This is important because the primary goal of DUMs is to quantify epistemic uncertainty while treating the weights of a NN deterministically in order to avoid sampling at inference time. Since epistemic uncertainty is expected to increase on OOD data, the representations of a NN need to be sensitive to the input distribution. However, discriminative models suffer from the fundamental problem of feature collapse [19, 21] which has to be counteracted using appropriate regularization. Secondly, DUMs use different methods to estimate uncertainty from such regularized representations (Sec. 4.2.2). Tab. 4.1 shows an overview of the resulting taxonomy.

Feature Collapse. Discriminative models can learn to discard a large part of their input information, as exploiting spurious correlations may lead to better performance on the training data distribution [128, 129]. Such invariant representations may be blind to distributional shifts, resulting in a collapse of OOD embeddings to in-distribution features. This problem is known as *feature collapse* [19],

DUMs		Uncertainty Estimation Method			
		Discriminative		Generative	
		Class centroid	Gaussian Processes	Gaussian Mixture Models	Normalizing Flows
Reg.	Distance awareness	DCS, DUQ	SNGP, DUE	DDU	-
	Informative representations	-	-	DCU, MIR	Invertible networks, PostNet

TABLE 4.1: Taxonomy of DUMs. Methods are grouped according to their regularization (**Reg.**) of the hidden representations (rows), and their uncertainty estimation method (columns). For reference: DCS [25], DUQ [19], SNGP [26], DUE [56], DDU [21], DCU [75, 121], MIR [20], Invertible networks [125–127], PostNet [122]

and it makes OOD detection based on high-level representations impossible.

4.2.1 *Regularization of Representations*

We group DUMs according to their approach to mitigating feature collapse. Currently, there are two main paradigms - distance awareness and informative representations - which we discuss in Section 4.2.1.1 and Sec. 4.2.1.2.

4.2.1.1 *Distance Awareness*

Distance-aware representations avoid feature collapse by relating distances between latent representations to distances in the input space. Therefore, one constrains the bi-Lipschitz constant, as it enforces a lower and an upper bound to expansion and contraction performed by a model. A lower bound enforces that different inputs are mapped to distinct representations and, thus, provides a solution to feature collapse. The upper bound enforces smoothness, *i.e.* small changes in the input do not result in large changes in the latent space. While there exist other approaches, *e.g.* [130], recent proposals have primarily adopted two methods to impose the bi-Lipschitz constraint.

The two-sided **Gradient Penalty** relates changes in the input to changes in feature space by directly constraining the gradient of the input [19]. Note, that this leads to large computational overhead as it requires differentiation of the gradients of the input with respect to the NN's parameters. **Spectral Normalization (SN)** [131] is a less computationally-demanding alternative. SN is applicable to residual layers and normalizes the weights W of each layer using their spectral norm $sn(W)$ to constrain the bi-Lipschitz constant. Various DUMs - SNGP [26], DUE [56] and DDU [21] - rely on SN to enforce distance-awareness of hidden representations. More details on gradient penalty and SN can be found in the supplement (Sec. 4.5.A.1).

Notably, the bi-Lipschitz constraint is defined with respect to a fix distance measure, which can be difficult to choose for high-dimensional data distributions. For example, SN [19, 26, 56] corresponds to the L_2 distance. While SN has empirically been found to perform well, it has been suggested [132] that popular SN approximations behave sub-optimally, and their interaction with losses, architecture and optimization is yet to be fully understood [133]. Principled approaches to providing exact singular values in convolutional layers [134] result in prohibitive computational complexity. Further, [135] provides an explanation for effectiveness of SN in convolutional residual NNs.

4.2.1.2 *Informative Representations*

While distance-awareness achieves remarkable performance on OOD detection, it does not explicitly preserve sample-specific information. Thus, depending on the underlying distance metric it may discard useful information about the input or act overly sensitive. An alternative line of work avoids feature collapse by learning informative representations [20, 73, 121, 125–127], thus forcing discriminative models to preserve information in its hidden representations beyond what is required to solve a task independent of the choice of an underlying distance metric. Notably, while representations that are aware of distances in the input space are also informative, both categories remain fundamentally different in their approach to feature collapse. While distance-awareness is based on the choice of a specific distance metric tying together input and latent space, informative representations incentivize a NN to store more information about the input using an auxiliary task [20, 121] or forbid information loss by construction [125–127]. There are currently four distinct approaches.

Contrastive learning [136] has emerged as an approach for learning representations that are both informative and discriminative and provably maximize the mutual information with the data distribution. This is utilized by Wu *et al.* [121] and Winkens *et al.* [75], who apply SimCLR [76] to regularize hidden representations for a discriminative

task by using a contrastive loss for pretraining and fine-tuning to force representations to discriminate between individual instances.

Reconstruction regularization [20] (MIR) instead forces the intermediate activations to fully represent the input. This is achieved by adding a decoder branch fed with the activations of a given layer to reconstruct the input.

Entropy regularization. PostNet [122] learns the class-conditional distribution of hidden representations end-to-end using a NF parameterizing a Dirichlet distribution. This allows them to enforce informative representations by implicitly encouraging large entropy of the NF during training. We refer to the supplement for details and further explanations (Sec. 4.5.A.2).

Invertible Neural Networkss (INNs) [125–127, 137], built via a cascade of invertible layers, cannot discard information except at the final classification stage. Consequently, the mutual information between input and hidden representation is maximized by construction. Interestingly, Behrmann *et al.* [138] showed that a ResNet is invertible if its Lipschitz constant is lower than 1, meaning that invertible ResNets both possess highly-informative representations and satisfy distance-awareness. However, note that this is not a necessary condition for invertibility, and thus information preservation.

4.2.2 *Uncertainty Estimation*

There are two directions regarding uncertainty estimation in DUMs - generative and discriminative approaches. While generative approaches use the likelihood produced by an explicit generative model of the distribution of hidden representations as a proxy for uncertainty, discriminative methods directly use the predictions based on regularized representations to quantify uncertainty.

Generative approaches estimate the distribution of hidden representations post-training or end-to-end, and use the likelihood as an uncertainty proxy. Wu *et al.* [121] propose a method to estimate the distribution in the feature space, where the variance of the dis-

tribution is used as a confidence measure. MIR [20], DDU [21] and DCU [75] fit a class-conditional GMM to their regularized hidden representations and use the log-likelihood as an epistemic uncertainty proxy. DEUP [139] uses the log-likelihood of a normalizing flow in combination with an aleatoric uncertainty estimate to predict the generalization error. A special instance of the generative approaches are INNs as they directly estimate the training data distribution. The likelihood of the input data is used as a proxy for uncertainty. While this idea is appealing, it can lead to training difficulties, imposes strong constraints on the underlying model and still remains susceptible to OOD data [140]. PostNet [122] is a hybrid approach which estimates the distribution of hidden representations of each class using a separate NF which is learned in an end-to-end fashion. Its log-likelihoods parameterize a Dirichlet distribution. We categorize PostNet as a generative approach since their epistemic uncertainty is the log-likelihood of the NF associated with the predicted class.

Discriminative approaches use the predictive distribution to quantify uncertainty. Mandelbaum *et al.* [25] propose to use a Distance-based Confidence Score (DCS) learning a centroid for each class end-to-end. Similarly, DUQ [19] builds on Radial Basis Function (RBF) networks [107] and proposes a novel centroid updating scheme. Both estimate uncertainty as the distance between the model output and the closest centroid. DUMs adopting SN [26, 56] (preserving L_2 distances) typically replace the softmax layer with GPs with RBF kernel, extending distance awareness to the output layer. In particular, SNGP [26] relies on a Laplace approximation of the GP based on the random Fourier feature (RFF) expansion of the GP posterior [141]. DUE [56] uses the inducing point approximation [142, 143], incorporating a large number of inducing points without overfitting [144]. The uncertainty is derived as the Dempster-Shafer metric [26], resp. the softmax entropy [56].

4.3 EXPERIMENTS

We investigate whether a deterministic treatment of the weights of a NN as proposed by DUMs not only detects OOD well but also yields well calibrated epistemic uncertainty, and scales to realistic vision tasks. Therefore, our experiments are comprised of two parts. Firstly, we evaluate DUMs on image classification, where we measure their calibration under synthetic corruptions (Sec. 4.3.1) and sensitivity to their regularization strength. We also evaluate DUMs on OOD detection in Sec. 4.3.1.2. Then, we extend DUMs to a large-scale dense prediction task - semantic segmentation (Sec. 4.3.2) - where we evaluate their calibration on synthetic corruptions (Sec. 4.3.2.1) based on Cityscapes as well as on more realistic distributional shifts (Sec. 4.3.2.2) based on data collected in the simulation environment CARLA [145].

Baselines. We compare DUMs with two baselines for epistemic uncertainty - MC dropout [16] and deep ensembles [11]. Moreover, we report the softmax entropy of a vanilla NN as a simple baseline. We refer to the supplement for details on uncertainty estimation in our baselines. Note, that the softmax entropy is expected to yield suboptimal calibration under distributional shifts since it quantifies aleatoric uncertainty while adding no computational overhead.

Methods. We evaluate DUQ [19], SNGP [26] and DDU [21] as representatives of distance-awareness, since these cover both techniques - SN and gradient penalty - and apply different techniques for uncertainty estimation. We exclude DUE [56] since it provides limited additional insights given SNGP. Moreover, we exclude DCS [25] since it only leads to a marginal improvement in their own experiments and their contrastive loss only operates on class centroids and, thus, is not expected to lead to distance awareness within clusters. Furthermore, we evaluate MIR [20], DCU [75] and PostNet [122] as representatives of informative representations. However, we do not scale DCU and PostNet to semantic segmentation. DCU with its constrastive pretraining based on SimCLR [76] is computationally too demanding due to large batch sizes. PostNet does not scale to se-

mantic segmentation due to instabilities arising from the end-to-end training of the NF for learning the distribution of hidden representations. They require a small hidden dimension ($\leq$10) which already leads to poor testset performance on CIFAR100. Further, we do not evaluate methods based on invertible neural networks [126, 127] since they 1) enforce strict constraints on the underlying architecture (*e.g.* fixed dimensionality of hidden representations) and often lead to training instabilities.

Calibration metrics. Typical calibration metrics are Expected Calibration Error (ECE) [146] and Brier score [147]. However, since most DUMs, except SNGP, do not provide uncertainty in form of a probabilistic forecast, we cannot rely on measuring the calibration of probabilities. Thus, we will exploit another desired property of uncertainty to quantify calibration, namely the ability to distinguish correct from incorrect predictions. In fact, this property is a relaxation of calibrated probabilities, as it is independent of the absolute value of uncertainty estimates. It solely relies on the ability of an uncertainty estimate to sort predictions according to their correctness. We assess the calibration of uncertainty estimates under distributional shifts using two metrics. Firstly, we report the *AUROC* obtained when separating correct and incorrect predictions based on uncertainty. Moreover, we introduce a new metric, *Relative Area Under the Lift Curve (rAULC)*, based on the Area Under the Lift Curve (AULC) [148]. The AULC is obtained by ordering the predictions according to increasing uncertainty and plotting the performance of all samples with an uncertainty value smaller than a certain quantile of the uncertainty against the quantile itself.

Formally, given a set of uncertainty quantiles $q_i \in [0, 1]$, $i \in [1, ..., S]$, with some quantile step width $0 < s < 1$ and the function $F(q_i)$ which returns the accuracy of all samples with uncertainty $u < q_i$, the AULC is defined as $AULC = -1 + \sum_{i \in [1,...,S]} s \frac{F(q_i)}{F_R(q_i)}$. Here, $F_R(\cdot)$ refers to a baseline uncertainty estimate that corresponds to random guessing. We subtract 1 to shift the performance of the random baseline to zero. Note, if an uncertainty estimate is anti-correlated with

a models' performance, this score can also be negative. To alleviate a bias towards better performing models, we further compute the rAULC by dividing the AULC by the AULC of a hypothetical (optimal) uncertainty estimation that perfectly orders samples according to model performance. In classification we measure AUROC and rAULC on the image-level, in semantic segmentation on the pixel-level. In all experiments we set the quantile step width to $s = \frac{1}{N}$, where N is the number of predictions.

We compute AUROC and rAULC on continuous distributional shifts 1) across all severities of distributional shifts (including the clean testset) and 2) for each severity separately. We use the former method to establish a quantitative comparison among the methods and the latter to depict the calibration evolution qualitatively as a function of the distributional shift's severity.

4.3.1 *Image Classification*

Datasets. We train DUMs on CIFAR-10 and CIFAR-100 [149] and evaluate on the corrupted versions of their test set CIFAR10/100-C [106] (Sec. 4.3.1). These include 15 synthetic corruptions, each with 5 levels of severity. Moreover, we explore how sensitive the calibration of DUMs is to the choice of regularization strength on MNIST [150] and FashionMNIST [108]).

Models and optimization. Each method shares the same backbone architecture and uses a method-specific prediction head. When training on CIFAR-10/100, the backbone architecture is a ResNet-50 [105] For the experiments regarding hyperparameter sensitivity on MNIST and Fashion-MNIST, we employ a multilayer perceptron (MLP) as feature extractor with 3 hidden layers of 100 dimensions each and ReLU activation functions. Each DUM has a hyperparameter for the regularization of its hidden representations. We choose the hyperparameter such that it minimizes the validation loss. All results are averaged over 5 independent runs. The standard deviation and opti-

mization details can be found in the supplement where not present in the main paper. Moreover, we provide per-sample training and inference runtimes for each method in the supplement.

4.3.1.1 *Continuous Distributional Shifts*

Method	CIFAR10-C			CIFAR100-C		
	ACC	AUROC	rAULC	ACC	AUROC	rAULC
Softmax entropy	0.882	0.782	0.708	0.610	0.762	0.596
MC Dropout [16]	0.885	**0.866**	0.829	0.615	0.818	**0.726**
Ensemble [11]	0.910	0.85 0	**0.833**	0.628	**0.824**	0.713
SNGP [26]	0.903	0.833	0.766	0.611	0.788	0.623
DDU [21]	0.884	0.673	0.441	0.609	0.635	0.339
MIR [20]	0.889	0.79	0.697	0.617	0.726	0.514
DUQ [19]	0.860	0.773	0.614	-	-	-
DCU [75]	**0.945**	0.794	0.706	**0.642**	0.750	0.558
PostNet [122]	0.882	0.784	0.676	0.520	0.743	0.603

TABLE 4.2: We compare Softmax, MC Dropout [16], Deep Ensembles, SNGP, DDU, MIR, DUQ, DCU and PostNet on CIFAR10/100-C. We evaluate the accuracy (ACC) on the uncorrupted testset, AUROC and rAULC. Ensembles and MC dropout demonstrate better uncertainty calibration than most DUMs. Only SNGP consistently outperforms the softmax entropy. DCU's superior performance is expected since it uses expensive contrastive pretraining. DUQ did not converge on CIFAR100-C due to training instabilities arising from dynamically updated cluster centroids.

Tab. 4.2 reports testset accuracy and calibration for the baselines and DUMs. AUROC and rAULC are computed for each corruption across all severity levels, then averaged over all corruptions. Further,

Metric	DUQ	DDU	SNGP	MIR
Pearson	-0.83	-0.30	0.58	0.76
Spearman	-0.83	-0.43	0.61	0.96

TABLE 4.3: Quantitative correlation between rAULC and regularization strength using Pearson/Spearman's rank correlation coefficient. MIR [20] demonstrates the largest correlation.

Fig. 4.1 depicts our metrics depending on the severity of corruptions. We generally observe that ensembles and MC dropout demonstrate best performance in terms of calibration. Further, SNGP is the only DUM that consistently outperforms the softmax entropy. Overall we observe that DUMs using the distribution of hidden representations for estimating epistemic uncertainty yield worse calibration. Among these we find that regularizing hidden representations by enforcing distance awareness (DDU) yields the worst calibration. The superior performance of DCU [75] in terms of testset accuracy originates from their extensive contrastive pretraining which includes extensive data augmentation and a prolonged training schedule. We note that DUQ [19] did not converge on CIFAR100 due to training instabilities. These arise from maintaining the class centroids, which become very noisy for 100 classes with only 600 samples per class.

Moreover, we report OOD detection performance of models used in Tab. 4.2 and Fig. 4.1 in the supplement. Interestingly, despite competitive performance on OOD detection DUMs fall short in terms of uncertainty calibration compared to MC dropout and deep ensembles.

4.3.1.2 *OOD Detection*

Table 4.4 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.

	OOD Data	STL10	SVHN	CIFAR100
CIFAR10	MC Dropout [16]	0.686 ± 0.004	0.885 ± 0.002	0.82 ± 0.003
	Ensemble [11]	**0.875 ± 0.001**	0.937 ± 0.009	0.758 ± 0.003
	DUQ [19]	0.633 ± 0.008	0.843 ± 0.016	0.766 ± 0.003
	SNGP [26]	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 [21]	0.737 ± 0.018	0.663 ± 0.073	0.638 ± 0.004
	DCU [75]	0.725 ± 0.027	**0.992 ± 0.014**	**0.921 ± 0.014**
	OOD Data	STL10	SVHN	CIFAR10
CIFAR100	MC Dropout [16]	0.772 ± 0.004	0.846 ± 0.01	0.735 ± 0.002
	Ensemble [11]	**0.801 ± 0.014**	0.741 ± 0.003	0.756 ± 0.007
	DUQ [19]	-	-	-
	SNGP [26]	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 [21]	0.698 ± 0.021	0.809 ± 0.056	**0.764 ± 0.019**
	DCU [75]	0.798 ± 0.019	**0.978 ± 0.005**	0.755 ± 0.024

TABLE 4.4: OOD detection performance when training on CIFAR10/100 and testing on various other datasets. We report AUROC averaged across 5 independent trainings.

4.3.1.3 *Sensitivity to Hyperparameters*

We are interested in the impact of the regularization strength on the uncertainty calibration. Therefore, we train DUQ [19], SNGP [26], MIR [20] and DDU [21] using various regularization strengths on MNIST and evaluate on continuously shifted data by rotating from 0 to 180 degrees in steps of 20 degrees. Fig. 4.2 depicts the test accuracy against the rAULC for various regularization strengths. Only for MIR we observe a clear, positive correlation between regularization strength and calibration. Moreover, Tab. 4.3 reports the corresponding Pearson/Spearman correlation coefficients. The supplement depicts

similar results on FashionMNIST [108] as well as for OOD detection performance (see Sec. 4.5.B.1).

4.3.1.4 *Training/Inference Runtime Comparison*

We report the per sample training and inference runtime in Table 4.5. The runtimes were measured on a single V100 using CIFAR10 and a ResNet50 backbone.

Method	Training Runtime [ms]	Inference Runtime [ms]
Softmax	1.14	0.36
MC Dropout [16]	1.13	5.17
DUQ [19]	3.68	0.35
SNGP [26]	2.47	0.44
DUE [21]	2.26	0.49
MIR [20]	1.34	0.55
DDU [21]	2.26	0.49

TABLE 4.5: Per sample runtime during training and inference on CIFAR10. The runtimes of MC dropout were obtained using 10 samples.

4.3.2 *Semantic Segmentation*

This section evaluates whether DUMs seamlessly scale to realistic vision tasks and compares their behaviour under synthetic and realistic continuous distributional shifts with the softmax entropy, MC dropout and ensembles. Therefore, we apply MIR [20], SNGP [26] and DDU [21] to semantic segmentation. Note that DUQ [19] did not converge on this task.

We consider semantic segmentation as a multidimensional classification problem, where each pixel of the output mask represents an independent classification problem. Given an image $\mathbf{x}$ with n pixels $\mathbf{y} = \{y_1, \cdots, y_n\}$, the predictive distribution factorizes according to

$p(\mathbf{y} \mid \mathbf{x}) = p(y_1 \mid \mathbf{x})p(y_2 \mid \mathbf{x}) \cdots p(y_n \mid \mathbf{x})$. We evaluate the calibration of the pixel-level uncertainty in our experiments.

Method	Cityscapes-C			CARLA-C		
	mIoU	AUROC	rAULC	mIoU	AUROC	rAULC
Softmax	0.503	0.815	0.737	0.422	0.854	0.818
MC Dropout [16]	0.506	**0.846**	**0.785**	0.410	0.843	0.730
Ensemble [11]	**0.525**	0.835	0.751	**0.428**	**0.863**	0.812
SNGP [26]	0.519	0.833	0.759	0.424	0.853	**0.813**
DDU [21]	0.505	0.731	0.542	0.408	0.467	-0.038
MIR [20]	0.504	0.729	0.564	0.412	0.744	0.619

TABLE 4.6: We compare semantic segmentation using Softmax, MC Dropout [16], Deep Ensembles, SNGP, DDU and MIR on Cityscapes-C and CARLA-C. We evaluate the mean Intersection over Union (mIoU) on the uncorrupted testset and AUROC/rAULC across all levels of corruption. Again, ensembles and MC dropout yield better calibrated uncertainty than most DUMs. Most notably, only SNGP consistently outperforms the softmax entropy. DUMs using an explicit generative model of hidden representations to estimate uncertainty perform particularly bad on realistic distributional shifts (CARLA-C).

Datasets. We evaluate on synthetic distributional shifts using a corrupted version of Cityscapes [151] (Cityscapes-C [152]) which contains the same corruptions as CIFAR10/100-C. To further benchmark DUMs in a realistically and continuously changing environment, we collect a synthetic dataset for semantic segmentation. We use the CARLA Simulator [145] and leverage the SHIFT dataset [153] toolkit for rendering images and segmentation masks under controlled distributional shifts in a driving scenario. The classes definition is aligned with the CityScape dataset [151]. Training data is collected from four towns in CARLA. We produce 32 sequences from each town. Vehicles and pedestrians are randomly generated for each sequence. Every se-

quence has 500 frames with a sampling rate of 10 FPS. We uniformly sample a validation set. We introduce continuous distributional shifts by varying the time-of-the-day and weather conditions (visual examples and details on data collection are in the supplement). The time-of-the-day is parameterized by the sun's altitude angle, where 90° means mid-day (training data) and the 0° means dust/dawn. We produce samples with altitude angles from 90° to 15° by steps of 5°, and 15° to -5°, where the environment changes sharply, in 1° steps. In order to continuously change the weather conditions, we increase the magnitude of the rain in four steps (see supplement for visual examples). We refer to this dataset as CARLA-C.

Backbone. We adopt Dilated ResNet (DRN) [154, 155] as semantic segmentation backbone since it is based on residual connections allowing the use SN. Using dilated convolutions it improves spatial accuracy, achieving satisfactory results on CityScapes [151]. We adopt the variant DRN-A-50. All results are averaged across 5 independent repetitions.

SNGP. DRN uses 1×1 convolutions at the last layer to map the latest feature map to the predicted segmentation mask. This works under the assumption that all pixels in the output mask are i.i.d. random variables. Following this intuition, we extend SNGP to semantic segmentation by fitting a $GP : \mathbb{R}^Z \to \mathbb{R}^C$ at pixel level that maps from the deep feature dimension z to the number of classes c. By keeping the GP kernel parameters shared across all pixels, we simulate a 1×1 convolutional GP, *i.e.* $\sigma : (H_l \times W_l \times Z) \to (H_l \times W_l \times C)$, where σ convolves the GP, H_l and W_l are, respectively, feature map height and width at layer l, Z is the number of latent features and C is the number of output classes. For details about the GP we refer to [26] or the supplement.

MIR and DDU require fitting the distribution of hidden representations. We fit a GMM with 20 components (*i.e.* number of classes) to each spatial location of the hidden representations using features extracted from the training data independently. This assumes that the distribution is translation invariant and factorizes along the spatial dimensions of the latent space. Pixel-level uncertainties are then com-

puted using bi-cubic interpolation following a similar procedure as the one proposed in [6]. We refer to the supplement for more details.

4.3.2.1 *Cityscapes Corrupted*

We evaluate the softmax entropy, ensembles [11], MC dropout [16], SNGP [26], MIR [20], and DDU [21] on Cityscapes-C [152]. Tab. 4.6 depicts mIoU and calibration performance in terms of AUROC and rAULC. Ensembles and MC dropout yield the best calibration, while among DUMs only SNGP consistently ourperforms the softmax entropy.

4.3.2.2 *Realistic Continuous Distributional Shifts*

Similarly, we evaluate the softmax entropy, ensembles [11], MC dropout [16], SNGP [26], MIR [20], and DDU [21] on CARLA-C. Tab. 4.6 depicts mIoU and calibration performance in terms of AUROC and rAULC. Ensembles yield the best calibration. Among DUMs only SNGP consistently ourperforms the softmax entropy which is in line with the results on image classification (Sec. 4.3.1).

4.4 CONCLUSION

This work investigates the shortcomings of a recent trend in research on deterministic epistemic uncertainty estimation. To this end, we provide the first taxonomy of DUMs. Moreover, we verify that DUMs indeed scale to realistic vision tasks in terms of predictive performance. However we find that (i) the epistemic uncertainty of many DUMs is not well calibrated under distributional shifts and (ii) the regularization strength in methods based on distance awareness does not correlate strongly with OOD detection and calibration.

DUMs recently showed good OOD detection performance and are interesting for practical applications in need of efficient uncertainty quantification. We established that DUMs mainly differ in their

regularization technique for countering feature collapse and their approach to quantifying uncertainty (Sec. 4.2).

Regarding the calibration under continuous distributional shifts of DUMs, we observe that such uncertainty estimates are considerably worse calibrated than scalable Bayesian methods. This is the case for image classification (Sec. 4.3.1) as well as semantic segmentation (Sec. 4.3.2) and for synthetic as well as more realistic distributional shifts. SNGP [26] denotes the only DUM that consistently yields better calibrated uncertainties under continuous distributional shifts than the softmax entropy. Simultaneously, SNGP is the only DUM which derives their uncertainty from its predictive distribution.

In particular, we find that methods relying on the distribution of hidden representations to quantify uncertainty [20, 21, 75] are poorly calibrated. It is understandable that these methods are worse calibrated than SNGP since they do not take into account the predictive distribution. They assume that locations in the feature space entail information about the correctness of predictions. While this is arguably true, features also contain additional information that renders them sub-optimal for judging the correctness of predictions due to ambiguities. Overall, this underlines the necessity to refrain from DUMs that purely rely on distances or log-likelihoods in the feature space when well calibrated uncertainties are required.

Interestingly, one may argue that DUMs simply do not scale to realistic vision tasks and, for that reason, are not well calibration in Sec. 4.3.2. However, note that DUMs in fact demonstrate strong predictive performance in such scenarios (see Tab. 4.6) and have been shown to perform well on OOD detection on realistic vision datasets (*e.g.* pixel-wise anomaly detection [6]). Therefore, we can conclude that this is not a problem of scaling DUMs to realistic scenarios but rather an inherent problem which these types of uncertainty estimates.

Moreover, another desirable property of DUMs would be that the strength of the feature space regularization correlates with the quality of the uncertainty - both in terms of calibration as well as OOD detection. Due to the original purpose of most DUMs, this would be

at least expected for OOD detection. However, we do not observe this for bi-Lipschitz regularization (Sec. 4.3.1). We hypothesize that this originates from the fact that these regularization techniques rely on an underlying distance metric - *i.e.* L_2 distance. Such distance metrics are not meaningful in the case of high-dimensional data, *i.e.* images. We hope that our findings will foster future research on making these promising family of methods better calibrated and more broadly applicable.

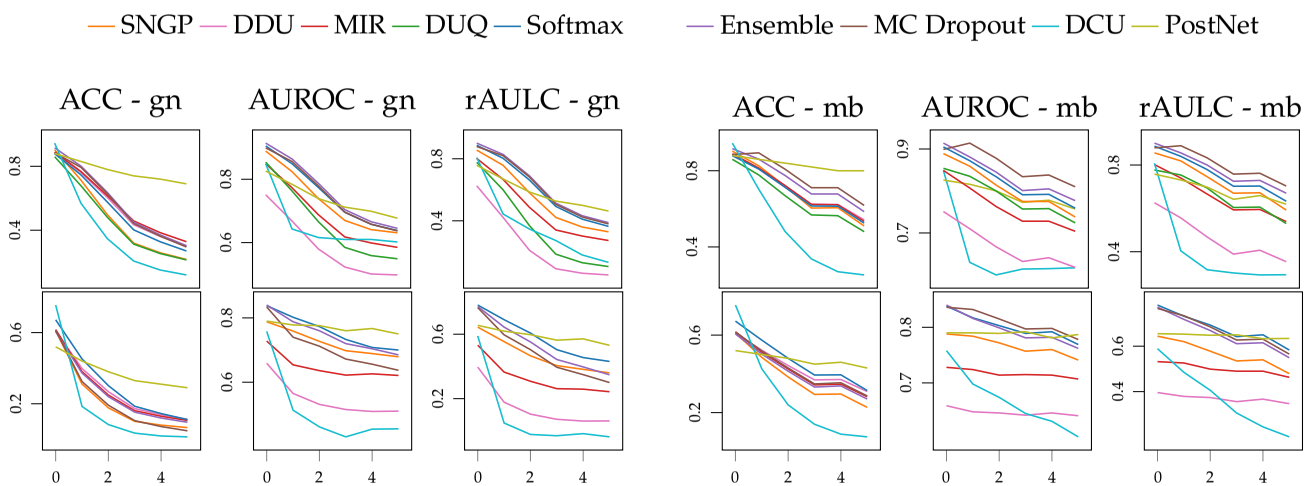


FIGURE 4.1: Softmax entropy, ensembles [11], MC dropout [16], DUQ [19], SNGP [26], MIR [20], DDU [21] and DCU [75] on CIFAR10-C (upper row) and CIFAR100-C (lower row) [106]. We show the accuracy, AUROC and rAULC on the corruptions gaussian_noise (gn) and motion_blur (mb) against the corruption severity. While all methods, except DCU, demonstrate a similar accuracy, DUMs - in particular methods based on generative modeling of hidden representations - yield worse calibration. DUQ did not converge on CIFAR100. Other corruptions are included in the supplement.

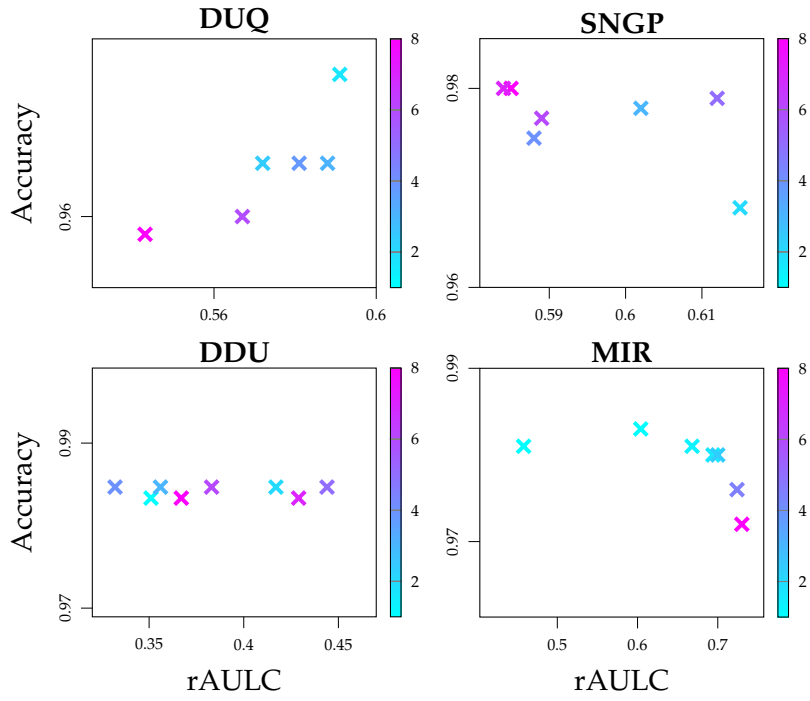


FIGURE 4.2: We analyze the sensitivity of DUQ [19], SNGP [26], MIR [20] and DDU [21] to their regularization strength. Therefore, we train models on MNIST and evaluate on continuously shifted data by rotating from 0 to 180 degrees in steps of 20 degrees. We plot the accuracy on the unperturbed testset against the rAULC computed using data from all levels of perturbation. Only for MIR we observe a clear, positive correlation between regularization strength and calibration. For DDU and SNGP, smaller regularization parameter denotes stronger regularization.

4.5 APPENDICES

We here provide additional theoretical background, implementation and optimization details, additional results, comparisons and ablation studies. In particular, we report additional details on the theoretical background necessary to understand DUMs' design choices in Sec. 4.5.A. We describe techniques used for enforcing Lipschitz constraints Sec. 4.5.A.1 and informative representations Sec. 4.5.A.2 in more detail. Moreover, Sec. 4.5.A.3 summarizes common choices of uncertainty estimation techniques for discriminative and generative DUMs.

Sec. 4.5.B.1/Sec. 4.5.B.2 show additional results for image classification/semantic segmentation. We report optimization/implementation details in Sec. 4.5.C/Sec. 4.5.D. Furthermore, we describe the quantification of uncertainty for image classification Sec. 4.5.D.3 and semantic segmentation Sec. 4.5.D.4. Finally, we provide details on the data collection process in CARLA and examples from the sequences collected for semantic segmentation Sec. 4.5.E.

4.5.A *DUMs - Fundamentals*

Subsequently, we provide a more detailed introduction of the most common concepts applied by DUMs. In particular, We describe regularization techniques, including Lipschitz regularization (Sec. 4.5.A.1) for enforcing distance awareness and informative representations (Sec. 4.5.A.2). Sec. 4.5.A.3 summarizes discriminative and generative approaches to uncertainty estimation in DUMs.

4.5.A.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 un-

derlying 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 this constraint.

Gradient Penalty. First introduced to regularize the Lipschitz constant in GAN training [156], a two-sided gradient penalty is used as an additional loss term to enforce sensitivity of the feature space to changes in the input by DUQ [19]. 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 [19].

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 [131]. For each layer $g : \mathbf{h}_{in} \to \mathbf{h}_{out}$, 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 constrains 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, which is 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 [26] 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.

4.5.A.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 [75, 122]. Subsequently, we discuss the different approaches in greater detail.

Constrastive learning. DCU [75] first pretrains its model using contrastive learning [76]. Subsequently, they finetune on the actual classification task by training simultaneously on a classification and a contrastive learning objective. This approach provably encourages the model to increase the mutual information between the input distribution and the distribution of hidden representations [136]. A disadvantage of 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 at hand (e.g. classification).

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.

Entropy regularization. PostNet [122] 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 $\alpha^{(i)} = (\alpha_1^{(i)}, \ldots, \alpha_c^{(i)})$ 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

$$H(D(\alpha^{(i)})) = \log(B(\alpha^{(i)})) + (\alpha_0^{(i)} - c)\psi(\alpha_0^{(i)}) - \sum_{j=0}^{c}(\alpha_j^{(i)} - 1)\psi(\alpha_j^{(i)}) \tag{4.1}$$

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, it encourages 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 [122] 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 [122], 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 [140]. 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 [75, 76] - particularly on classification. Architecture-agnostic. Provably maximizes mutual-information between input distribution and distribution of hidden representations [136].
  - **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.

4.5.A.3 *Uncertainty Estimation*

Training a feature extractor under the regularization constraints imposed by distance awareness (Section 4.2.1.1, Section 4.5.A.1) or representation informativeness (Section 4.2.1.2) allows to leverage intermediate representations to quantify uncertainty over network's predictions. Extending Section 4.2.2, 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 [21] and DCU [75] learn the density of hidden representations post-training based on the features observed on the training data. In contrast, PostNet [122] 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 [21] and DCU [75] - 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 [21, 75] is used as a proxy for epistemic uncertainty.

On the other hand, PostNet [122] 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 [122] 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.* [25] 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 [19] builds on Radial Basis Function (RBF) networks [107], 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 [26], they can be combined with regularization techniques that enforce distance awareness of

the feature extractor [26, 56] 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 [26] or the softmax entropy [56]. Based on this intuitive idea, SNGP [26] and DUE [56] 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 [141] and the inducing point approximation [142, 143]. 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 [19] to regularize the bi-Lipschtiz constant.

4.5.B *Additional Results*

4.5.B.1 *Image Classification - Sensitivity to regularization strength*

We provide additional ablation studies on the sensitivity to regularization strength for different methods on MNIST (Fig. 4.3) and FashionMNIST (Fig. 4.4)

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.

4.5.B.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. 4.5 illustrates examples under *minimal* distributional shift (*i.e.* Azimuth angle of the sun $= 85°$ and Fig. 4.6 under *maximal*

distributional shift (*i.e.* Azimuth angle of the sun $= -5°$. 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. 4.6, 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.

4.5.C *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.

4.5.C.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.

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.

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 $[0.0, 0.1, 0.5, 1.0, 2.0, 5.0, 10.0, 20.0, 50.0, 100.0]$.

4.5.c.2 *Image Classification - CIFAR10/SVHN*

When training on CIFAR-10/SVHN, we use a ResNet-18 [105] 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.

4.5.C.3 *Semantic Segmentation.*

When training on semantic segmentation, we use a DRN [154, 155] (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.

4.5.D *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[1], SNGP[2] and DUE[3].

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 [157]. This is especially relevant in large-scale tasks like semantic segmentation, were it is important to reduce the computational overload.

4.5.D.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.

1 https://github.com/y0ast/deterministic-uncertainty-quantification
2 https://github.com/google/uncertainty-baselines/blob/master/baselines/imagenet/sngp.py
3 https://github.com/y0ast/DUE

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.

4.5.D.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.

4.5.D.3 *Uncertainty Estimation*

We provide details on the estimation of uncertainty for the baseline methods. For details on the uncertainty derivation in DUMs, please refer to Section 4.2 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 estimate epistemic uncertainty according to Eq. 1.1.

4.5.D.4 *Uncertainty Estimation 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.

4.5.E *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 [145] for rendering the images and segmentation masks. The classes definition is aligned with the CityScape dataset [151]. 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. 4.7 (first and second row). The time-of-the-day is parametrized by the Sun's altitude angle, where 90° means the mid-day and the 0° means the dusk 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. 4.7 (bottom). Here the cloudiness and, ground wetness and ground reflection are the main changing parameters.

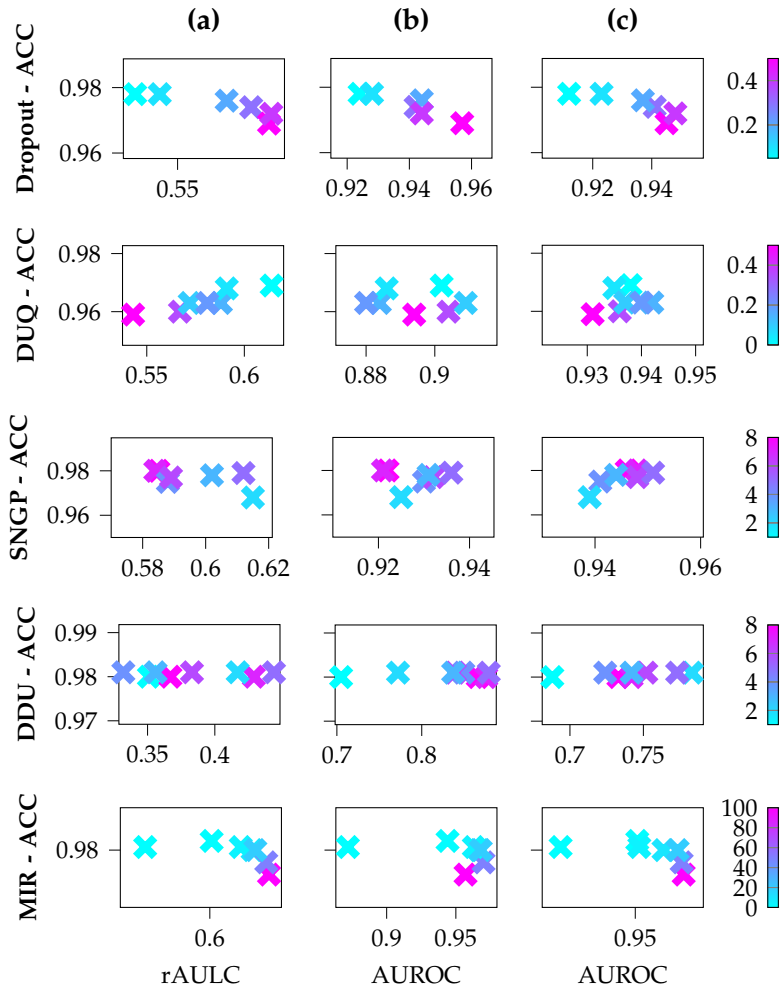


FIGURE 4.3: Trained on MNIST. Vertical axis: Test accuracy. (a) depicts calibration (MNIST - Rotation), (b) OOD detection performance on FashionMNIST and (c) OOD detection performance on Omniglot. Horizontal axis: rAULC (left), AUROC against FashionMNIST (center) and Omniglot (right) for Dropout (1st 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.

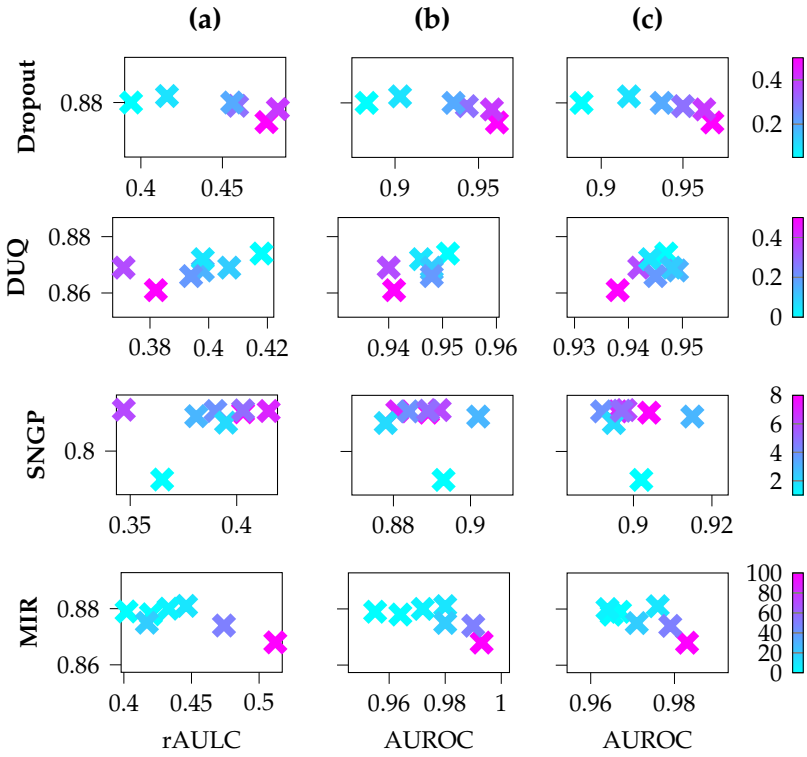


FIGURE 4.4: Trained on FashionMNIST. Vertical axis: Test accuracy. (a) depicts calibration (FashionMNIST - Rotation), (b) OOD detection performance on MNIST and (c) OOD detection performance on Omniglot. Horizontal axis: rAULC (left), AUROC against MNIST (center) and Omniglot (right) for Dropout (1st row), DUQ (2nd row), SNGP (3rd row) and MIR (4th 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.

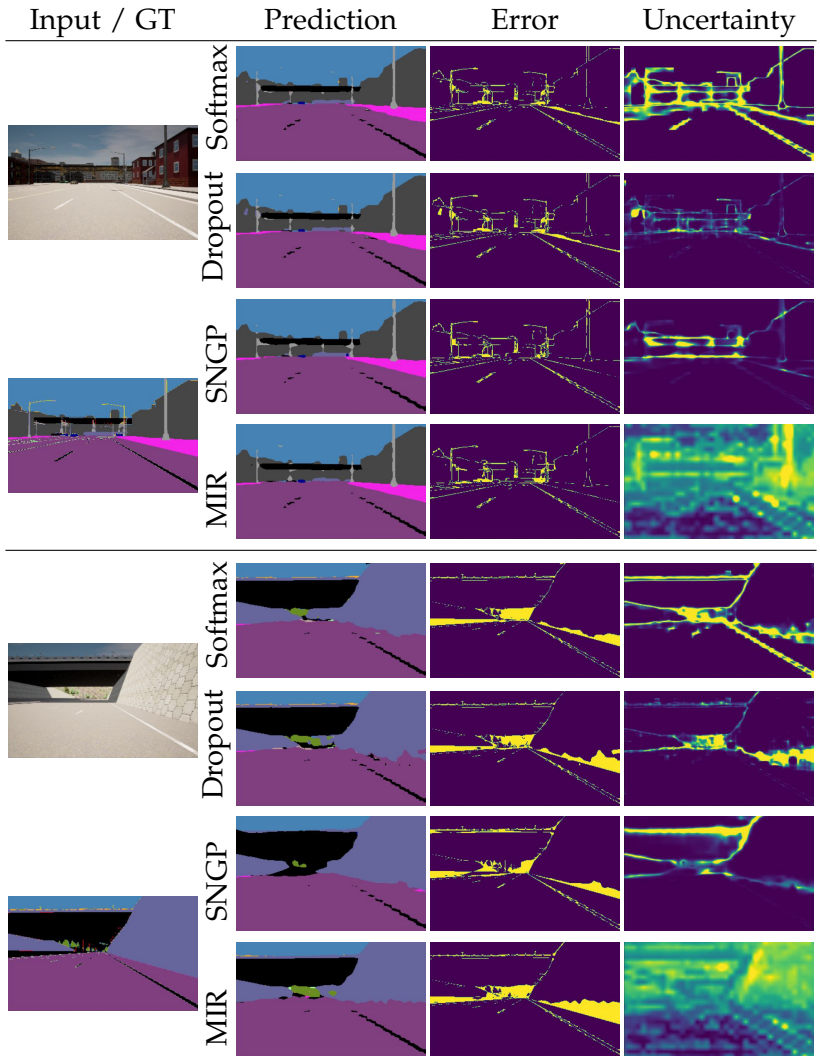


FIGURE 4.5: Qualitative comparison of uncertainty from Softmax, MC Dropout, SNGP and MIR under minimal time-of-the-day distribution shift (*i.e.* Azimuth angle of the sun = 85°). We show the input image (Input) and the ground truth mask (GT), and we report for each method the predicted segmentation mask (Prediction), the error mask (Error) and the uncertainty mask (Uncertainty).

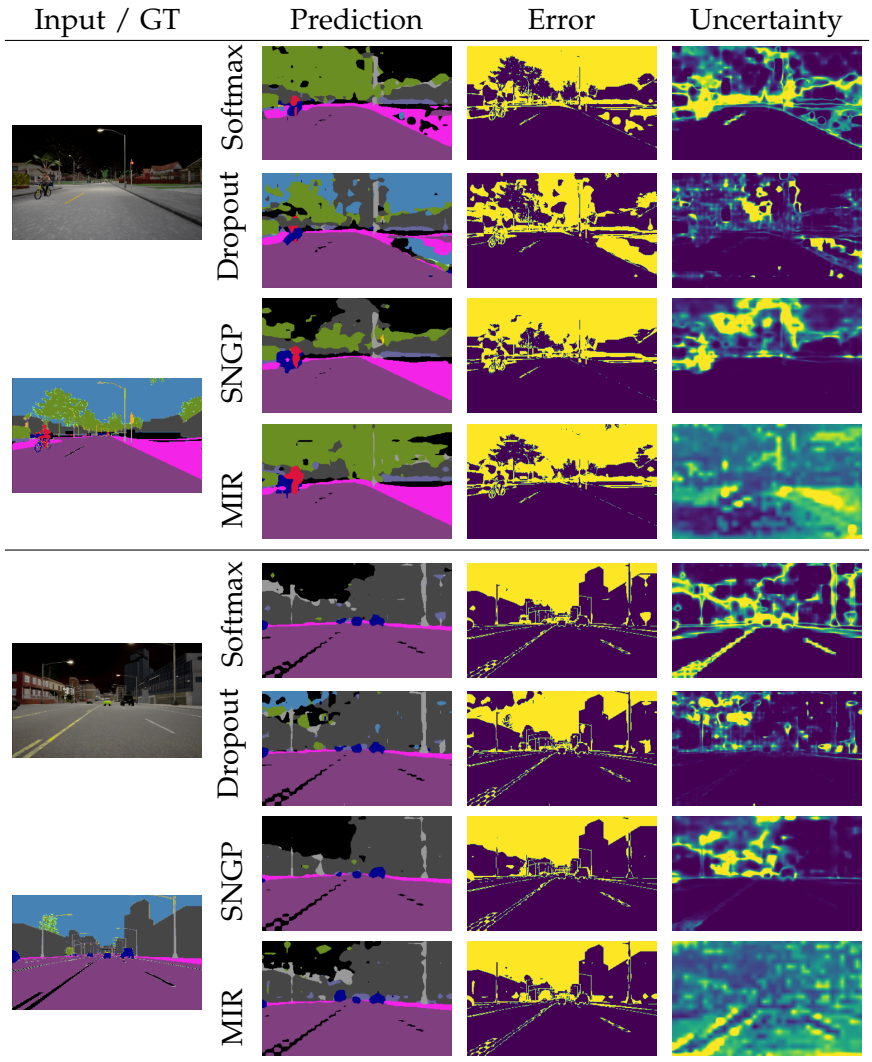
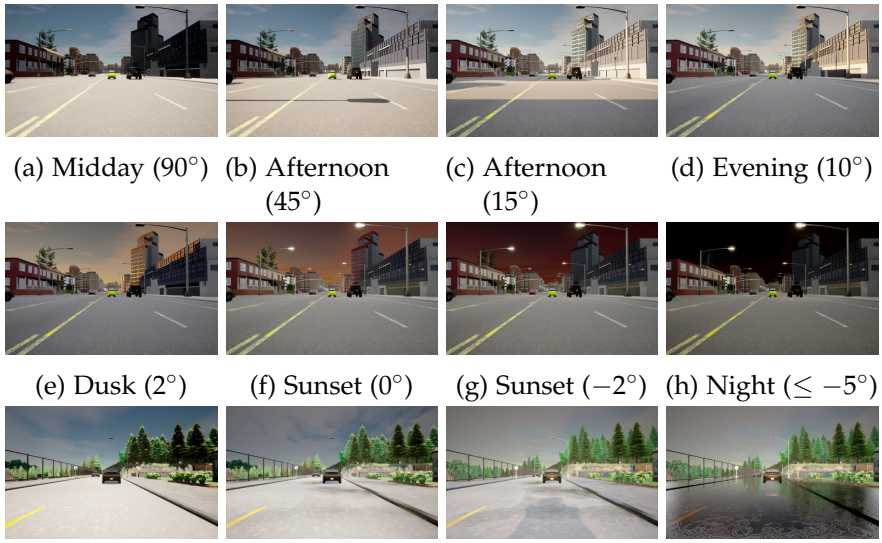


FIGURE 4.6: Qualitative comparison of uncertainty from Softmax, MC Dropout, SNGP and MIR under maximal time-of-the-day distribution shift (*i.e.* Azimuth angle of the sun $= -5°$). We show the input image (Input) and the ground truth mask (GT), and we report for each method the predicted segmentation mask (Prediction), the error mask (Error) and the uncertainty mask (Uncertainty).



(a) Midday (90°) (b) Afternoon (45°) (c) Afternoon (15°) (d) Evening (10°)

(e) Dusk (2°) (f) Sunset (0°) (g) Sunset (−2°) (h) Night (≤ −5°)

(i) Clear (j) Cloudy (k) Small rain (l) Heavy rain

FIGURE 4.7: Changing of the time-of-the-day and the weather

5

MANIFLOW: IMPLICITLY REPRESENTING MANIFOLDS WITH NORMALIZING FLOWS

Normalizing Flows (NFs) are flexible explicit generative models that have been shown to accurately model complex real-world data distributions. However, their invertibility constraint imposes limitations on data distributions that reside on lower dimensional manifolds embedded in higher dimensional space. Practically, this shortcoming is often bypassed by adding noise to the data which impacts the quality of the generated samples. In contrast to prior work, we approach this problem by generating samples from the original data distribution given full knowledge about the perturbed distribution and the noise model. To this end, we establish that NFs trained on perturbed data implicitly represent the manifold in regions of maximum likelihood. Then, we propose an optimization objective that recovers the most likely point on the manifold given a sample from the perturbed distribution. Finally, we focus on 3D point clouds for which we utilize the explicit nature of NFs, i.e. surface normals extracted from the gradient of the log-likelihood and the log-likelihood itself, to apply Poisson surface reconstruction to refine generated point sets.

5.1 INTRODUCTION

The goal of generative modeling is to grasp the fundamental laws governing a data distribution in order to autonomously create alternative realizations. As such, it represents a core element of intelligence and, thus, also machine learning research. Beyond fundamental research, recent advances in generative modeling [28, 78, 83, 84, 158] have lead to a variety of real-world applications [159–161]. Existing generative models can be subdivided into implicit and explicit models. Both types aim to generate novel realizations consistent with

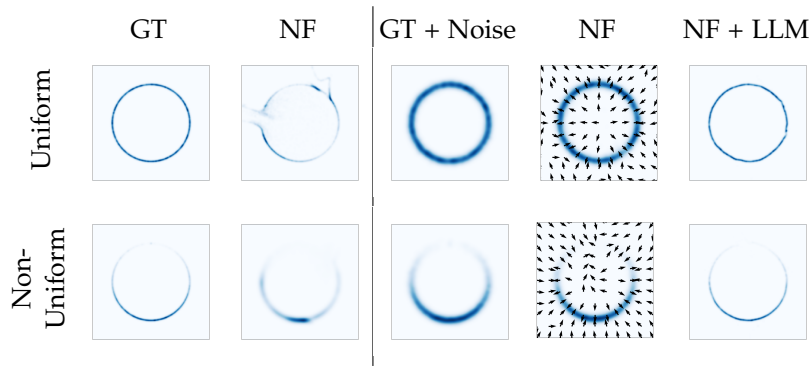


FIGURE 5.1: In this toy example, we learn a distribution, both uniform (GT) and non-uniform (GT + Noise), that exists on a sphere using a NF. Training a vanilla RealNVP [28] on uncorrupted data fails (left). The NF is able to learn the noise-perturbed distribution. Applying our proposed log-likelihood maximization (LLM) according to Eq. 5.3, we are able to recover the original distribution - in the uniform as well as non-uniform scenario. We also visualize the direction of the gradients of the log-likelihood. We observe that they are perpendicular to the manifold in both cases.

prior experience. However, unlike their implicit counterparts, explicit models can also attach a likelihood to new observations. This directly gives rise to more applications, such as anomaly detection [159]. Among explicit generative models, NFs [28] have recently emerged as a family of parametric models of exceptional flexibility. Based on invertible NNs, they allow modeling high-dimensional data while maintaining the advantage of explicit likelihood evaluation [30, 162, 163] and have been successfully applied in a variety of contexts - e.g. image super-resolution [160], modeling 3D point clouds [31] and physical simulation [164].

Despite recent successes, NFs still possess a fundamental drawback rooted in their core idea - their invertibility. Real-world data often resides on a lower dimensional manifold embedded in higher

dimensional space. While apparent for surfaces of 3D shapes, this also holds for other data modalities, e.g. images. NFs rely on invertible NNs to quantify the likelihood of data instances and are trained by directly maximizing the likelihood of the data. Since a lower dimensional manifold has no volume, NFs can predict arbitrarily large density values in such cases and their training objective becomes ill-conditioned on data distributions that lie on manifolds. This well-known problem [91, 93] is typically tackled by artificially increasing the dimensionality of the data distribution. Practically, this is achieved by adding noise to the data [30–32]. While this makes the optimization of NFs feasible, it produces the undesirable side effect of reduced quality of generated data. Most prior works tackling this problem make strong assumptions about the underlying manifold [91–94, 96] - i.e. regarding its structure or dimensionality. However, such assumptions are typically impractical in the case of real-world applications. Notably, SoftFlow [95] proposed a practical framework for learning distributions on lower dimensional manifolds using NFs and evaluated its benefit on 3D point clouds.

This work approaches this problem from a novel perspective. Given a NF trained on a data distribution with added noise, we generate samples from the original distribution which resides on the lower dimensional manifold. We achieve this by recognizing that the regions of maximum likelihood locally present an implicit representation of the manifold if the added noise is sufficiently small. Using this observation, we propose an optimization objective that recovers the most likely point on the manifold, given a sample from the distribution with added noise. We show that our approach consistently improves performance when modeling 3D point clouds as well as images. Furthermore, we leverage the benefits of an explicit generative model to derive an approach that improves our method and reduces the overhead of test time optimization for the specific case of 3D point clouds. To this end, we use the normal direction extracted from the gradient of the log-likelihood and log-likelihood itself to perform Poisson surface reconstruction [165] to efficiently refine the generated 3D point clouds.

5.2 METHOD

This section presents a new approach to sample from distributions that reside on manifolds using NFs. One can regard those as delta distributions in a higher dimensional space. We first revisit NFs and discuss their challenges (Sec. 5.2.1). We then demonstrate how NFs implicitly represent manifolds and introduce a general framework for sampling from this implicit representation (Sec. 5.2.2)). Finally, we propose an efficient algorithm based on Poisson surface reconstruction [80] for 3D point clouds (Sec. 5.2.3).

Therefore, let Ω denote a m-dimensional compact manifold embedded in n-dimensional Euclidean space, $\Omega \in \mathbb{R}^n$ where $m < n$. Capital letters X refer to random variables with their corresponding probability density P_X and lower case letters refer to their realizations $x \sim P_X(x)$.

5.2.1 *The Curse of Invertibility - Normalizing Flows on Manifolds*

NFs are a family of flexible parametric models for estimating probability distributions based on invertible NNs. Consider an invertible transformation $F_\theta : \mathbb{R}^n \to \mathbb{R}^n$ with parameters θ - parameterized by a NN. In order to learn a probability distribution $P_X(x)$ with $x \in \mathbb{R}^n$, NFs parameterize the latter using the change of variable formula

$$P_X(x) = P_Y(F_\theta^{-1}(x))\det\left|J_{F_\theta^{-1}}(x)\right|$$

where $J_{F_\theta^{-1}}(x)$ is the Jacobian of F_θ^{-1} at x and Y is a random variable with known parametric distribution P_Y. The parameters θ are learned by maximizing the log-likelihood:

$$\begin{aligned}\log(P_X(x)) = \log(P_Y(F_\theta^{-1}(x))) \\ + \log\left(\det\left|J_{F_\theta^{-1}}(x)\right|\right)\end{aligned} \tag{5.1}$$

From Eq. 5.1 we observe the main challenge of implementing NFs is the design of invertible NN layers allowing to evaluate the logarithm of their Jacobian determinant efficiently.

Consider a random variable S that exists on a m-dimensional manifold Ω which is distributed according to $P_S(x) = 0 \; \forall x \notin \Omega$. Assuming $P_Y(x)$ has full support in $\mathbb{R}^n$, Eq. 5.1 yields an ill-posed optimization problem since $\log\left(\det\left|J_{F_\theta^{-1}}(x)\right|\right)$ becomes infinitely large when squeezing $\mathbb{R}^n$ in its entirety onto Ω. Likewise, choosing $P_Y(x)$ without full support in $\mathbb{R}^n$ renders the optimization problem equally ill-posed. The initial F_θ^{-1} is required to be an injective transformation from Ω and the support of Y. Otherwise the first term in Eq. 5.1, $\log(P_Y(F_\theta^{-1}(x)))$, would tend to $-\infty$. This can also be observed in the toy example depicted in Fig. 5.1 where the vanilla NF is not able to successfully learn the distribution which lies on the 1-sphere.

In practice these optimization difficulties are bypassed by adding noise to S [30–32, 95]. We refer to this as inflating the distribution $P_S(x)$. Formally, this corresponds to convolving P_S with a noise distribution P_N:

$$P_X(x) = \int_{\mathbb{R}^n} P_N(x-s)P_S(s)ds \tag{5.2}$$

A common choice for P_N is the Gaussian distribution. Learning the inflated distribution P_X instead of P_S stabilizes training with Eq. 5.1. However, it inevitably leads to a loss of information about S since the learned distribution P_X deviates from the distribution of interest P_S.

5.2.2 *Sampling from the Implicit Manifold Representation*

The goal of this work is to generate samples from P_S given knowledge of P_X and P_N. We achieve this by choosing the form of P_N such that it allows us to generate samples from P_S on S given samples from P_X. We note that this is an ill-posed problem for general P_N. We first assume that S is uniformly distributed on Ω. Further, we ensure that P_N is unimodal and zero-centered. We achieve this by construction, e.g. by choosing $P_N = \mathcal{N}(x; 0, \Sigma)$. *This allows us to identify the maximum of P_X as an implicit representation of S*. Subsequently, we here set P_N to be a n-dimensional Gaussian - $P_N(x) = \mathcal{N}(x; 0, \Sigma)$ with

$\Sigma = \sigma^2 \mathrm{I}$. Given this insight, our primary concern is subsequently to develop an approach to project a sample $x \sim P_X(x)$ back to S.

Naturally, the quality of this representation deteriorates for larger noise magnitudes σ. In order to represent S well, the noise magnitude σ needs to be sufficiently small. In particular, consider the typical length scale l of the manifold S. We define l as the maximum side length of the m-dimensional hypercube that approximates S well everywhere locally. In this case, the gradient of the log-likelihood only has a significant component that is perpendicular to Ω since P_S is uniformly distributed on Ω. This allows us to decode S locally for a given point sampled from P_X by following its gradient of the log-likelihood to its maximum.

Thus, assuming a uniformly distributed S and that F_θ sufficiently well estimates P_X, we can sample $s \sim P_S(x)$ on S by: 1.) drawing a sample $x' \sim P_X(x)$ and 2.) solving the following optimization problem that equates to finding the maximum of $P_X(x)$ closest to x:

$$s = \underset{x \in \mathbb{R}^n}{\operatorname{argmin}} \left(||x - x'||_2^2 - \log(P_X(x)) \right) \tag{5.3}$$

Note that even uniformly distributed P_S are practically relevant as they are often assumed for point clouds of 3D shapes. However, it is also interesting to consider the case of non-uniformly distributed P_S. For such P_S, applying Eq. 5.3 would still generate samples that lie on Ω. However, the resulting distribution may not resemble P_S since for non-uniform P_S we do not have $\nabla P_X(x) = 0$ everywhere on S. Nevertheless, for sufficiently smooth P_S and small σ Eq. 5.3 still yields a good estimate of P_S. Specifically, we consider σ such that $\sigma \ll \frac{1}{\max_{s \in \Omega}(|\nabla_\Omega P_S(x)|)}$, where ∇_Ω denotes the derivative along Ω. In this case we can still approximate P_S locally as a uniform distribution on an m-dimensional hypercube and, thus, obtain samples from P_S.

We optimize Eq. 5.3 by minimizing the objective:

$$\mathcal{L}(x) = -\log(P_X(x)) + \lambda ||x - x'||_2^2 \tag{5.4}$$

λ denotes a hyperparameter that is set by optimizing performance on the validation set. Interestingly, $\frac{\mathrm{d}\lambda \|x-x'\|_2^2}{\mathrm{d}x} = -\frac{\mathrm{d}\log\left(\mathcal{N}(x;x',\frac{1}{\lambda}I)\right)}{\mathrm{d}x} + C$

with C constant in x. This is the kernel used to inflate P_S. Thus, from a Bayesian perspective we can also interpret it as the negative logarithm of a prior distribution on the location of the most likely point s on Ω that generated x'. In this light, $P_X(x)$ resembles the likelihood function learned from the data and we understand the optimum of Eq. 5.4 as the MAP estimate of s given x'. Subsequently, we refer to Log-Likelihood Maximization according to Eq. 5.3 as LLM.

Choice of λ. We implement λ as a global hyperparameter which is optimized using the validation set. However, a locally varying $\lambda = \lambda(x)$ may improve upon this restrictive assumption. While we find empirically that a global λ already yields significant advantages, we consider a locally varying λ as a promising future research avenue.

5.2.3 *An Adjusted Algorithm for 3D Point Clouds*

While Eq. 5.4 gives us a practical and general framework for generating samples from P_S using P_X, it can be computationally demanding since it requires multiple gradient updates. This is particularly challenging when using large neural networks to parameterize $P_X(x)$ as it is often the case for real-world data distributions such as images [30, 162, 163] and 3D point clouds [29, 31, 32].

We here view 3D point clouds as distributions on 2D manifolds in $\mathbb{R}^3$ and derive an approach that circumvents multiple gradient updates for point clouds. In contrast to other data modalities such as images, generating a 3D point cloud requires sampling a large set of points which accurately represents the entire distribution at once. Thus, we can generate samples on the manifold given global information about the entire set of points. Hereafter, we focus on the case where Ω refers to a 2D manifold embedded in $\mathbb{R}^3$ and P_S refers to a uniform distribution on Ω.

We consider the gradient of the log-likelihood parameterized by the NF $\nabla \log(P_X(x))$. Given sufficiently small σ, $\nabla \log(P_X(x))$ is parallel to a surface normal on Ω for uniform P_S in the vicinity of Ω.

We illustrate this in Fig. 5.1. Such normal vectors can be used to fuel off-the-shelf surface reconstruction algorithms such as Poisson surface reconstruction [165]. Poisson surface reconstruction uses point sets and oriented surface normals to construct an implicit function of a watertight 3D shape. The surface can then be reconstructed by locating the iso-surface of the implicit function. While we here work with Poisson surface reconstruction, our approach naturally transfers to any surface reconstruction algorithm that relies on surface normals [166]. Subsequently, we term this this approach LL-Poisson.

The proposed algorithm is depicted in Alg. 2. Given a set of K_1 points $\mathcal{X}$ sampled from P_X, we apply the following routine. We first compute the gradient $\nabla \log(P_X(x))$ at each point and normalize it to unit length. Then, we create consistent normals by propagating the orientation of these vectors using a graph created by considering the k nearest neighbors of each point. These K vector-point pairs are then used to obtain a mesh representing Ω. We refine this mesh using $\log(P_X(x))$ by removing vertices that have lower likelihood than the α-percentile of the original set of points $\mathcal{X}$. The latter is particularly crucial for removing vertices that reside far away from Ω and when dealing with non-watertight 3D shapes since Poisson surface reconstruction generates watertight meshes. Finally, we uniformly sample K_2 points from the resulting mesh.

5.3 EXPERIMENTS

We first present a toy examples to foster an intuitive understanding of the effect of optimizing Eq. 5.3 (Sec. 5.1). Then, we compare ManiFlow based on log-likelihood maximization (LLM) and LL-Poisson with SoftFlow [95] (Sec. 5.3.2.1) and other point cloud representations (Sec. 5.3.2.2) for point cloud autoencoding on ShapeNet [167]. Finally, we demonstrate that ManiFlow scales to high-dimensional data distributions on image generation (Sec. 5.3.4).

Algorithm 2 Generating Samples on Ω

Require: F_θ, initial point set $\mathcal{X}$, k, K_2, α
$G \leftarrow \nabla \log(P_X(\mathcal{X}))$
$G \leftarrow \text{normalize}(G)$ ▷ Normalized to unit length
$G \leftarrow \text{propagate\_orientation}(G, k)$ ▷ Propagates orientation according to k nearest neighbors
$\text{mesh} \leftarrow \text{poisson\_reconstruction}(\mathcal{X}, G)$
$\text{perc}_\alpha \leftarrow \text{percentile}(\alpha, P_X(\mathcal{X}))$
$\text{mesh} \leftarrow \text{remove\_vertices}(\text{mesh}, \text{perc}_\alpha)$ ▷ Remove unlikely vertices
$\mathcal{X}_\Omega \leftarrow \text{sample\_uniformly}(\text{mesh}, K_2)$ ▷ Generate K_2 uniform samples from the mesh **return** $\mathcal{X}_\Omega$

5.3.1 *Synthetic Data*

We perform experiments on artificial distributions in 2D. We consider two cases: a uniform and a non-uniform distribution on a 1-sphere. We train a RealNVP [28] with 5 coupling layers for 5000 epochs on both distributions. More training details can be found in the supplement.

The distributions and results are depicted in Fig. 5.1. As expected we observe that a standard NF directly trained on the clean data fails to generate samples that match the target distribution. We further verify that the problem can be mitigated by adding Gaussian noise to the data at training time. We find that performing post-training LLM according to Eq. 5.3 allows us to sample from the original manifold on the 1-sphere in accordance with the ground truth distribution. Fig. 5.1 also visualizes the direction of the gradients of the log-likelihood. The gradients are approximately perpendicular to the 1-sphere in the vicinity of the manifold in case of the uniform as well as the non-uniform distribution. The direction only starts to deviate in regions of very low probability density. This observation underlines our discussion in Sec. 5.2.2 and motivates us to use NFs in combination with Poisson surface reconstruction (see Sec. 5.2.3).

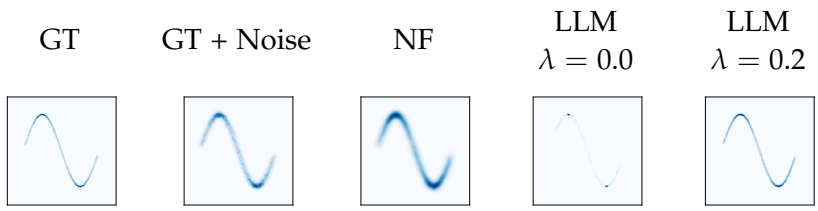


FIGURE 5.2: In this experiment, we learn a non-uniform distribution (GT, GT + Noise) that exists on a 1D sine curve in 2D using a NF. We demonstrate the importance of the regularization parameter λ. In absence of regularization ($\lambda = 0$) LLM is not able to recover the correct distribution on the sine curve. However, using $\lambda = 0.2$ recovers the correct distribution.

Impact of λ. We provide an additional experiment on artificial data showing the importance of setting $\lambda > 0$ in Eq. 5.3. To this end, we train a NF on a non-uniform distribution on a 1D sine curve embedded in 2D space. After training, we recover the manifold using ManiFlow - LLM using $\lambda = 0.0$ and $\lambda = 0.2$. The results are in Fig. 5.2. We observe that $\lambda = 0.0$ fails to recover the true distribution on the manifold since generated points collapse towards global peaks of the distribution parameterized by the NF.

5.3.2 *3D Point Cloud Autoencoding*

3D point clouds are a widespread data modality. Since they are often exclusively comprised of surface points, they denote an important testbed for methods that generate data on lower dimensional manifolds. We evaluate on point cloud autoencoding since we are primarily interested in the ability of ManiFlow to represent 3D point clouds.

Dataset. We evaluate on the ShapeNet dataset [167]. We use the categories: airplanes, cars, and chairs. We train on the presampled point clouds provided by PointFlow [31], which are also used by Soft-

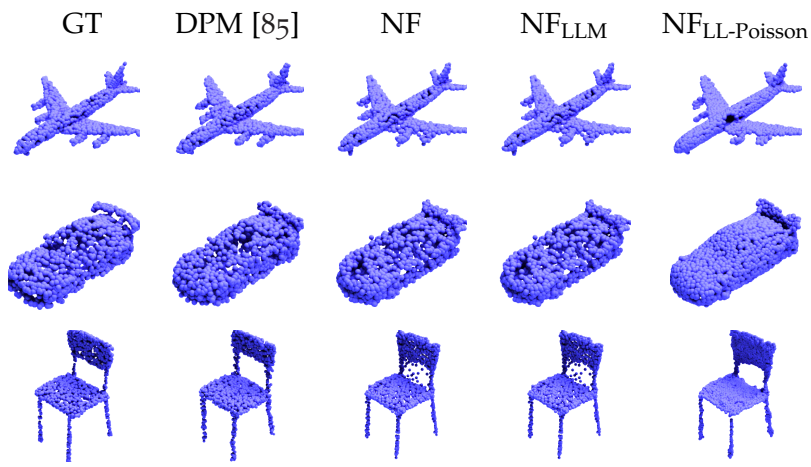


FIGURE 5.3: Qualitative comparison of the ground truth (GT), DPM [85], a RealNVP (NF) and a RealNVP with LLM/LL-Poisson (NF$_{\text{LLM}}$/NF$_{\text{LL-Poisson}}$). We observe that ManiFlow paired with LL-Poisson yields smoother surfaces.

Flow [95], DPM [85] and ShapeGF [80]. Each shape consists of 15000 points which we randomly subsample to 2048 points during training and test time. Prior to subsampling each shape is individually normalized such that it has zero mean and unit variance.

Evaluation. At test time we decode 2048 points from the trained models and compare them to 2048 points uniformly sampled from the ground truth. We repeat each evaluation 5 times and report the mean result. We use three metrics for measuring the quality of reconstructed point clouds. We follow prior work and compute the Chamfer Distance (CD) and Earth Mover Distance (EMD) [77]. If not stated otherwise, we follow prior work [31] and report results on CD multiplied by 10^4 and results on EMD by 10^2. We further report the F1-score (F1) [168]. We use a threshold of $\tau = 10^{-4}$ for the F1-score. We also report the performance of an oracle as an upper bound to the performance. The oracle performance is computed by comparing two independently sampled subsets of the ground truth point cloud.

Encoder. In an autoencoder an encoder model computes a latent representation of a point cloud that, subsequently, a decoder model uses to reconstruct the original point cloud. We follow PointFlow [31] and parameterize the encoder with a PointNet [169], which produces permutation-invariant latent representations. Specifically, we use convolutions with filter sizes of 128, 256 and 512 followed by a max pooling layer. The resulting 512-dimensional representation is transformed into a 256-dimensional latent representation using a single fully-connected layer. As a decoder model we use a conditional NF. In particular, when comparing with SoftFlow [95], we follow their experimental setup and use a NF based on autoregressive layers and invertible 1×1 convolutions (Sec. 5.3.2.1). When comparing against methods for modeling point clouds (Sec. 5.3.2.2), we adopt a RealNVP similar to the one used in DPF [32]. The RealNVP consists of 64 coupling layers where each coupling layer consists of two fully-connected layers with a hidden dimensionality of 64. The first fully-connected layer is followed by a Batch Normalization layer and the swish activation function [170]. Further details are in the supplement.

Optimization. In Sec. 5.3.2.1 we precisely follow the setup used by SoftFlow [95]. We only deviate from it when training a basic NF in absence of the SoftFlow framework. In this case we simply use Gaussian noise of a fixed magnitude during training instead of randomly sampling the noise variance from a range of values at each iteration. We set the standard deviation of the noise to 0.02. In Sec. 5.3.2.2 we train the autoencoder for 1500 epochs using a batch size of 128 using the Adam optimizer [116]. We set the initial learning rate to 6.4×10^{-4} and divide it by 4 after 1200 and 1400 steps. We exponentially decay the standard deviation of the Gaussian noise added to the data from 0.25 to 0.02 between the epochs 100 and 1200. Each model is optimized on 4 V100 GPUs. For more details we refer to the supplement.

ManiFlow - LLM. When performing LLM (Eq. 5.3) we use the Adam optimizer and a fixed learning rate of 10^{-3}. We found 25 gradient updates to be sufficient. We choose a regularization parameter λ in

	Method	CD↓	EMD↓	F1 ↑
Airplane	NF	1.262	2.58	84.22
	NF + SF	1.179	2.69	85.63
	NF + SF + LLM	1.161	2.68	85.96
	NF + LLM	1.151	**2.56**	86.25
	NF + LL-Poisson	**1.132**	3.06	**87.26**
Car	NF	6.978	5.18	22.36
	NF + SF	6.876	5.23	23.19
	NF + SF + LLM	6.850	5.22	23.44
	NF + LLM	6.887	5.18	**24.05**
	NF + LL-Poisson	**6.830**	**5.01**	23.73
Chair	NF	11.76	6.86	19.77
	NF + SF	10.84	**6.43**	20.86
	NF + SF + LLM	**10.83**	**6.43**	21.13
	NF + LLM	11.57	6.78	20.85
	NF + LL-Poisson	11.60	6.69	**21.91**

TABLE 5.1: Quantitative comparison of ManiFlow on point cloud autoencoding with SoftFlow [95] (SF) on the categories airplane, car and chair of ShapeNet. We train a vanilla NF trained on noisy data with and without the SoftFlow framework. Post-training we apply either log-likelihood maximization (LLM) (Eq. 5.3) or Poisson surface reconstruction (LL-Poisson). ManiFlow consistently outperforms SoftFlow and is even able to improve SoftFlow. CD is multiplied by 10^4 and EMD by 10^4.

(5.4) such that performance on validation set is optimized. Note that λ is sensitive to the expressiveness of the NF. More expressive NF lead to larger log-likelihoods, which requires larger λ. In Sec. 5.3.2.1 we use $\lambda = 2.0$ and in Sec. 5.3.2.2 we use $\lambda = 4000$.

ManiFlow - LL-Poisson. We first create an initial set of K_1=10000 points using the NF and compute the gradient of the log-likelihood for each of them. Then, we ensure consistent orientation of the resulting vectors by propagating their orientation based on $k = 20$ nearest neighbors. We perform Poisson surface reconstruction using maximum depth of 9 provided by Open3D [171]. Finally, we set $\alpha = 0.05$ and sample K_2=2048 points uniformly from the resulting mesh.

5.3.2.1 *Comparison with SoftFlow*

Here we compare ManiFlow with SoftFlow [95]. We report autoencoding performance of NF with and without the SoftFlow framework and apply ManiFlow based on LLM/LL-Poisson to the NF in absence of SoftFlow. We also perform LLM on the NF trained with SoftFlow.

The results are in Tab. 5.1. We make several observations. ManiFlow LLM/LL-Poisson consistently improves reconstruction performance. Secondly, despite the absence of the SoftFlow framework at training time, a vanilla NF is able to outperform a NF trained with SoftFlow when applying ManiFlow. Further, the performance of SoftFlow can be reliably improved when applying LLM to it, indicating that SoftFlow can be enhanced by applying ManiFlow. Lastly, we note that LL-Poisson tends to outperform LLM despite of being twice as fast. This demonstrates that refining the entire point set as a whole based on the NF's log-likelihood is superior over projecting each point individually.

5.3.2.2 *Further Comparison on Autoencoding.*

We compare ManiFlow with recent methods proposed for representing point clouds - AtlasNet [79], PointFlow [31], ShapeGF [80] and DPM [85]. The results are in Tab. 5.2. ManiFlow LLM/LL-Poisson outperforms standard NFs, which is consistent with previous results. Further, NFs equipped with ManiFlow are able to clearly outperform AtlasNet [79] which was not possible before [29, 32] while allowing

	Method	CD↓	EMD↓	F1 ↑
Airplane	AtlasNet - Sphere [79]	1.178	3.36	84.66
	AtlasNet - Square [79]	1.211	3.42	84.30
	PointFlow [31]	1.213	2.76	85.31
	ShapeGF [80]	0.949	2.49	89.14
	DPM [85]	**0.947**	**2.19**	**89.27**
	RealNVPs	1.139	2.77	85.69
	RealNVPs + LLM	1.063	2.76	86.98
	RealNVPs + Poisson	1.045	2.86	88.13
	Oracle	0.722	1.95	92.31
Car	AtlasNet - Sphere [79]	6.231	5.09	25.96
	AtlasNet - Square [79]	6.221	5.19	25.79
	PointFlow [31]	6.540	5.16	24.02
	ShapeGF [80]	5.508	4.23	**28.91**
	DPM [85]	**5.462**	**3.96**	28.77
	RealNVPs	6.134	4.89	24.96
	RealNVPs + LLM	5.976	4.90	26.99
	RealNVPs + Poisson	5.928	4.30	26.44
	Oracle	3.368	3.04	49.76
Chair	AtlasNet - Sphere [79]	7.643	6.30	27.51
	AtlasNet - Square [79]	7.690	6.74	27.13
	PointFlow [31]	10.094	6.46	21.33
	ShapeGF [80]	6.297	5.03	32.68
	DPM [85]	**6.293**	**4.27**	**32.59**
	RealNVPs	8.071	5.31	25.59
	RealNVPs + LLM	8.003	5.34	27.09
	RealNVPs + Poisson	8.013	5.04	27.34
	Oracle	2.762	3.07	56.02

TABLE 5.2: Quantitative comparison of autoencoding performance with AtlasNet [79], PointFlow [31], ShapeGF [80] and DPM [85] on airplane/chair/car category of ShapeNet. ManiFlow reliably improves performance and enables NFs to outperform AtlasNet. Poisson slightly outperforms LLM. ManiFlow consistently outperforms SoftFlow and is even able to enhance the performance of SoftFlow. CD is multiplied by 10^4 and EMD by 10^4.

Poisson	CD↓	EMD↓	F1↑
✗	**0.947**	**2.19**	**89.27**
✓	0.975	2.99	88.91

TABLE 5.3: Poisson surface reconstruction on point clouds generated by DPM [85] on the airplane category. Unlike for NFs, we observe that the reconstruction quality degrades

Use LL	CD↓	EMD↓	F1↑
✗	1.077	3.01	87.50
✓	**1.045**	**2.86**	**88.13**

TABLE 5.4: ManiFlow LL-Poisson with/without the log-likelihood (LL) of the NF on the airplane category. The reconstruction quality degrades without LL of the NF

Use LL	CD↓	EMD↓	F1↑
✗	24.256	7.73	41.53
✓	**9.427**	**6.91**	**54.83**

TABLE 5.5: ManiFlow LL-Poisson with/without the log-likelihood (LL) of the NF on sparse point clouds (256 points) of the the airplane category. The reconstruction quality degrades when not relying on the LL of the NF more significantly on sparse point clouds

likelihood evaluation. While recent implicit generative models, such DPM and ShapeGF, demonstrate strong performance, ManiFlow is able to reduce the gap. Moreover, Fig. 5.3 visualizes reconstructed point clouds from DPM and ManiFlow. LL-Poisson tends to create smooth surfaces.

For ManiFlow LL-Poisson, it is important to use large number of points K_1 for obtaining competitive performance. To verify that this does not equally help other models, we also post-process point clouds generated by DPM using Poisson surface reconstruction. We apply

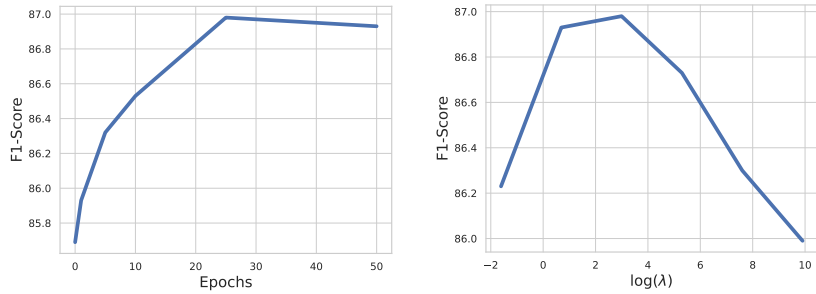
the same hyperparameters as for ManiFlow with LL-Poisson. The only difference is the inability to evaluate the likelihood of each point. Therefore, we estimate normals based on Principal Components Analysis (PCA). We report the result in Tab. 5.3. Although using a large number of points for creating the mesh, the final autoencoding performance deteriorates in absence of the ability to evaluate the log-likelihood.

Similarly, we investigate the impact of using the log-likelihood provided by the NF when using LL-Poisson. Therefore, we evaluate our model using LL-Poisson and with Poisson surface reconstruction with normals estimated via PCA on the airplane category of ShapeNet. The results are reported in Tab. 5.4. LL-Poisson outperforms the baseline on every metric. We conclude that the log-likelihood is an important ingredient for LL-Poisson.

Given the importance of the information stored in the log-likelihood, we are interested in determining whether LL-Poisson is beneficial when reconstructing a mesh from a sparse point cloud. Therefore, we set K_1=256 and perform again surface reconstruction with and without information from the log-likelihood. We expect that for a good mesh uniformly sampled points from it demonstrate good autoencoding performance. We show the results in Tab. 5.5. We observe an even larger performance improvement when using LL-Poisson over the baseline in the case of sparse point clouds. We conclude that LL-Poisson can be useful when reconstructing meshes from sparse point clouds.

5.3.3 *Impact of Hyperparameters*

We analyse the sensitivity of LLM/LL-Poisson to its hyperparameters. We perform this anaysis on the task of point cloud autoencoding and use the RealNVP trained on the airplane category in Sec. 5.3.2. For LLM we plot the F1-score against the number of epochs used to optimize Eq. 5.3 and the logarithm of the magnitude of the weight λ. For LL-Poisson we plot the F1-score against the depth used for



(a) F1-Score depending on the number of epochs used for ManiFlow-LLM.

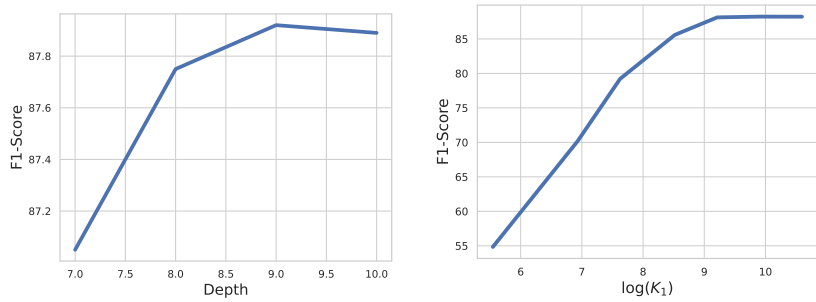
(b) F1-Score depending on the on the logarithm of the weight λ used for ManiFlow-LLM.

FIGURE 5.4: Analysis of the sensitivity of ManiFlow-LLM to the number of epochs (a) and the magnitude of λ (b) on the airplane category.

LLM	MNIST	CIFAR10	CelebA
✗	30.0	72.7	51.7
✓	**21.7**	**64.7**	**43.9**

TABLE 5.6: Quantitative evaluation of LLM on image generation. We report the FID score when training a GLOW model [30] on MNIST/CIFAR10/CelebA with/without LLM. During training Gaussian noise is added to the images. LLM improves generation performance

Poisson surface reconstruction and the logarithm of the number of points K_1 sampled prior to surface reconstruction. Fig. 5.4 and Fig. 5.5 depict the results. While there exists an optimal λ, we find that the other hyperparameters typically lead to saturating performance given a sufficient magnitude.



(a) F1-Score depending on the depth used in Poisson surface reconstruction.

(b) F1-Score depending on the on the logarithm of the number of points K_1.

FIGURE 5.5: Analysis of the sensitivity of ManiFlow-LL-Poisson to the depth used for Poisson surface reconstruction (a) and the number of points K_1 (b) on the airplane category.

5.3.4 *Generative Modelling on Images*

So far we have conducted experiments low-dimensional data. In the following we further evaluate ManiFlow based on LLM on image generation to investigate whether it scales to high-dimensional data distributions. To this end, we train a GLOW [30] on MNIST [107], CIFAR10 [109] and CelebA [172]. We train each model for 400 epochs using a batch size of 256 on MNIST/CIFAR10 and 128 on CelebA. We use the Adam [116] optimizer and a fixed learning rate of 10^{-4}. When training NFs on images one typically uses additive uniform noise [30]. Since uniform noise violates our assumption of unimodal noise, we replace it with additive Gaussian noise. We choose the standard deviation of the Gaussian noise such that it has the same entropy as the uniform distribution used in the original work, i.e. $\sigma = 0.00756$. After training we perform 50 steps of LLM using $\lambda = 0.2$ and a learning rate of 10^{-3}. We report the Fréchet Inception Distance (FID) [173] which is widely used to quantify the perceptual quality of generated images.

Tab. 5.6 shows quantitative results on image generation. We observe a reliable and clear improvement of the FID-score when applying ManiFlow. Thus, ManiFlow paired with LLM based on Eq. 5.3 scales to high-dimensional data.

5.4 CONCLUSION

This work introduced ManiFlow - a practical framework for sampling from lower dimensional manifolds using NFs. To this end, we recognised that regions of maximum likelihood in NFs trained with sufficiently small noise implicitly represent the underlying manifold. Thus, given a trained NF we can generate samples on manifold. We proposed two strategies to achieve this.

Firstly, we introduced a general framework based on maximizing the log-likelihood using Eq. 5.3 (LLM). ManiFlow based on LLM is applicable to low-dimensional as well as high-dimensional real world data distributions. It consistently improves 3D point cloud reconstruction (Sec. 5.3.2) and image generation (Sec. 5.3.4). In particular, on 3D point cloud autoencoding we find that post-processing samples from NFs with ManiFlow LLM performs considerably better than AtlasNet while being able to evaluate the likelihood. This was previously not possible [29, 32].

Moreover, we proposed LL-Poisson as a specialized version for 3D point clouds (Sec. 5.2.3). LL-Poisson utilizes the NF's log-likelihood and its normalized gradients to perform Poisson surface reconstruction. LLM post-processes samples individually. In contrast, LL-Poisson relocates points also based on local information about adjacent points and information about surface normals extracted from the gradient field. Thus, LL-Poisson outperforms LLM on point cloud autoencoding (Sec. 5.3.2). We showed that the log-likelihood of the NF and its gradient improve Poisson surface reconstruction (see Tab. 5.3, Tab. 5.4 and Tab. 5.5). We particularly found that the benefit the NF's log-likelihood for LL-Poisson magnifies when operating on

sparse point clouds. We conclude that LL-Poisson can also be used to improve mesh creation from sparse point clouds.

In the realm of 3D shapes, it is also interesting to interpret ManiFlow in the light of implicit neural representations (INRs). INRs represent shapes as the level set of an implicit function parameterized by a NN and are typically learned as signed distance functions [174]. Similarly, ManiFlow naturally encodes the manifold implicitly in its regions of maximum likelihood. Practically the main difference remains that the underlying NF in ManiFlow allows direct sampling in the vicinity of the surface whereas an INR requires many evaluations far away from the surface.

While ManiFlow already yields strong improvements over standard NFs, it simultaneously gives rise to interesting future research avenues. For example, since we have seen that the log-likelihood of NFs contains geometric information, it is natural to ask whether we can adjust the training of NFs to improve this property. A resulting approach could lead to the intersection of score-matching and direct likelihood maximization. Furthermore, score matching has recently demonstrated strong performance on point cloud denoising [175]. Similarly, it worth investigating whether NFs equipped with ManiFlow are useful for this task. In fact, such an approach could extend traditional denoising approaches based on kernel density estimates [176].

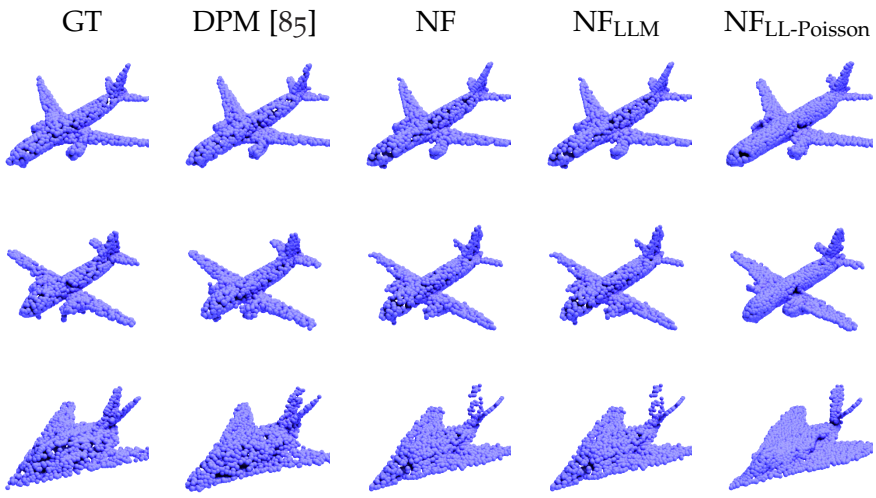


FIGURE 5.6: Further qualitative results on the airplane category of the ground truth (GT), DPM [85], a RealNVP (NF) and a RealNVP with LLM/LL-Poisson (NF$_{\text{LLM}}$/NF$_{\text{LL-Poisson}}$).

5.5 APPENDICES

In Sec. 5.5.A, Sec. 5.5.B and Sec. 5.5.C, we depict additional qualitative examples of reconstructed point clouds, visualize the meshes obtained using LL-Poisson and show qualitative examples of generated image. Finally, we provide additional details about the architecture and optimization procedure used in the experiments in Sec. 5.5.D.

5.5.A *Further Visualization of Point Clouds*

We provide additional qualitative examples of generated point clouds in Fig. 5.6, Fig. 5.7 and Fig. 5.8.

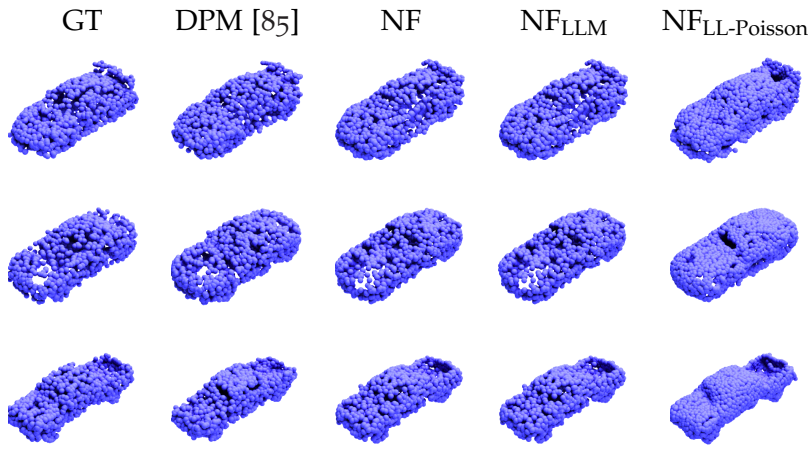


FIGURE 5.7: Further qualitative results on the car category of the ground truth (GT), DPM [85], a RealNVP (NF) and a RealNVP with LLM/LL-Poisson (NF$_{\text{LLM}}$/NF$_{\text{LL-Poisson}}$).

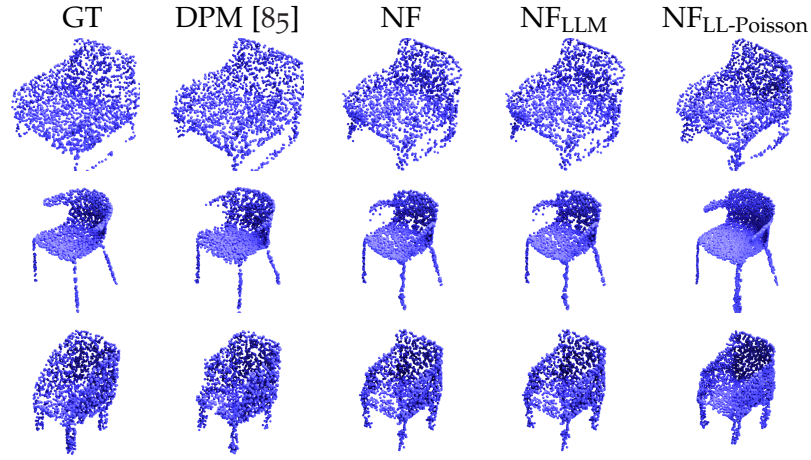


FIGURE 5.8: Further qualitative results on the chair category of the ground truth (GT), DPM [85], a RealNVP (NF) and a RealNVP with LLM/LL-Poisson (NF$_{\text{LLM}}$/NF$_{\text{LL-Poisson}}$).

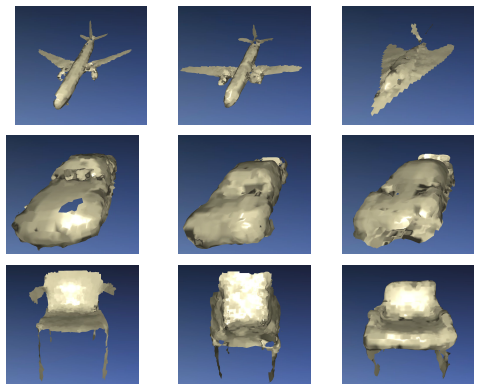


FIGURE 5.9: Qualitative examples of meshes created with LL-Poisson on the categories airplane, car and chair.

5.5.B *Visualization of Meshes*

Poisson surface reconstruction in LL-Poisson extracts a mesh of the 3D shape from the NF's log-likelihood. We provide qualitative examples of these meshes in Fig. 5.9. Note that these meshes correspond to the point clouds visualized in Fig. 5.6, Fig. 5.7 and Fig. 5.8.

5.5.C *Visualization of Generated Images*

We provide qualitative examples of applying LLM to a GLOW [30] pretrained of CelebA [172]. We further visualize the difference between the images generated by GLOW before and after post-processing using LLM.

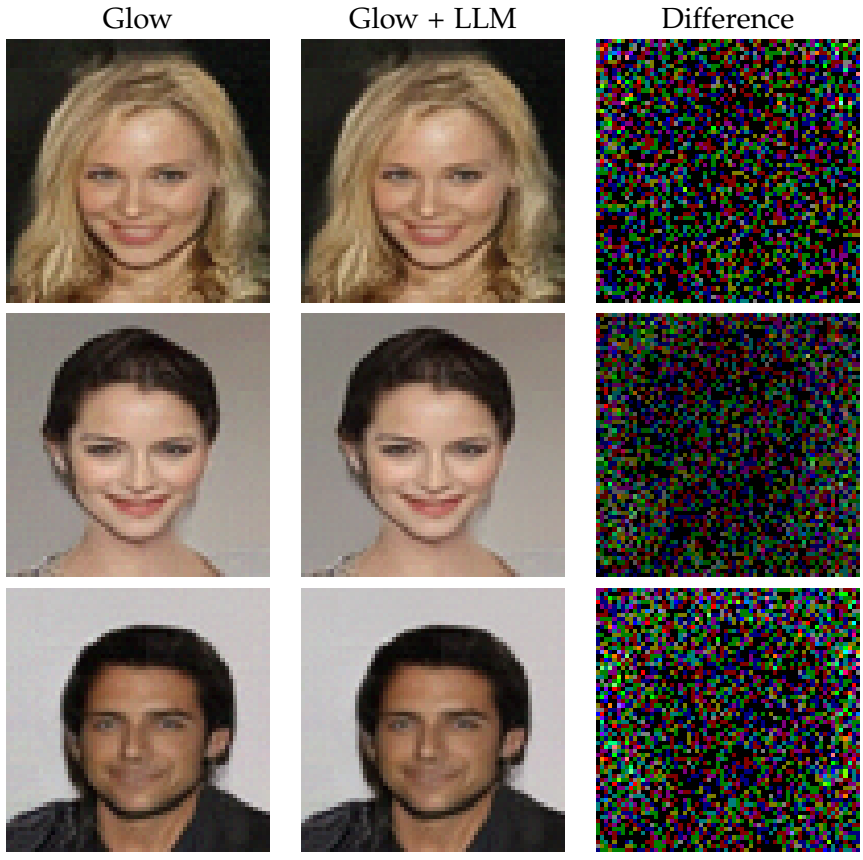


FIGURE 5.10: Samples from Glow (+LLM) and the corresponding difference. The difference magnified such that its largest value is 255 for visualization purpose. The difference resembles white noise.

5.5.D *Experimental Details*

5.5.D.1 *Architecture of Normalizing Flows*

Experiments on Artificial Data. We train a RealNVP [28] comprised of 5 coupling layers. Each coupling layer is comprised of two linear

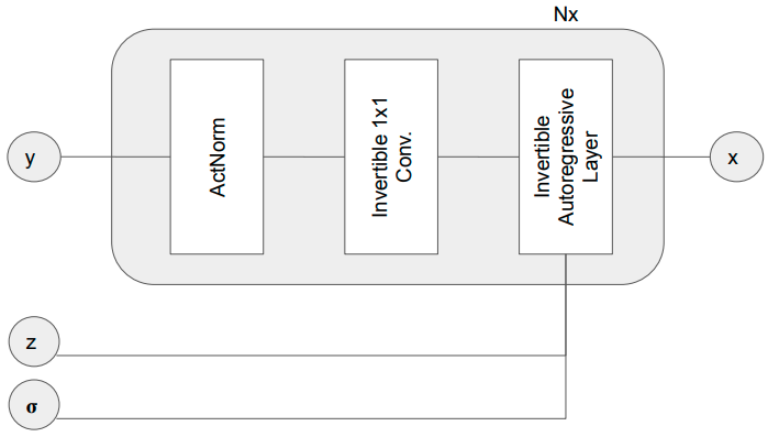
layers with a relu activation function and 128 hidden dimensions between them. We mask dimensions of the 2D data alternatingly.

Comparison with SoftFlow [95] in Sec. 5.3.2. We apply the architecture proposed in the original work [95]. The decoder NF is comprised of a sequence of 8 identical modules. This module is depicted in Fig. 5.11a. Here, activation normalization (ActNorm) and invertible 1x1 convolutions are implemented according to GLOW [30]. 1x1 convolutions are enforced to be invertible by parameterizing the kernel using the LU-decomposition [30]. The invertible autoregressive layer is implemented according to [177]. In invertible autoregressive layers scaling and translation of each dimension are calculated autoregressively. Each autoregressive step is parameterized by a separate NN consisting of of 4 linear layers with tanh activation functions and 256 hidden dimensions.

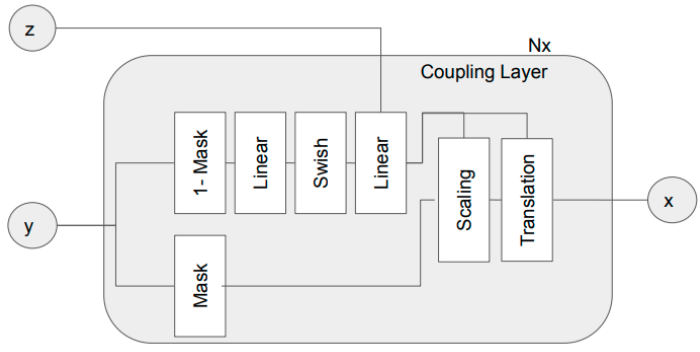
General experiments on autoencoding in Sec. 5.3.2. Here the decoder model is implemented as a ReaLNVP [28] consisting of 63 coupling layers. Such coupling layers are schematically depicted in Fig. 5.11b. A coupling layer splits the dimensions into two sets and applies the identity mapping to one set of dimensions. Further, the set of dimensions, which is mapped by the identity, is used to compute scaling and translation factors for the other set of dimensions. The computation of scaling and translation is parameterized by a NN which consists of 2 linear layers with an intermediate Swish activation function and 64 hidden dimensions. The conditioning on the latent shape representation z is implemented by FiLM conditioning [178].

5.5.D.2 *Training Details*

Artificial Data. We train the RealNVP for 5000 iterations with a batch size of 128 on randomly drawn samples from the aforementioned distributions. We use the Adam [116] optimizer and an initial learning rate of 10^{-3} which we half after 2500 iterations.



(a) Basic module of architecture used in comparison with SoftFlow [95].



(b) Basic module of RealNVP architecture used in general comparison on point cloud autoencoding.

FIGURE 5.11: Architectures used in comparison in Sec. 5.3.2.1 (a) and Sec. 5.3.2. We use N=8 (a) and resp. N=63 (b) blocks.

Comparison with SoftFlow [95] in Sec. 5.3.2. We train autoencoder for 15000 epochs using a batch size of 128 and the Adam optimizer with an initial learning rate of 10^{-3}. We multiply the learning rate by a factor of 0.24 after 5000 and 10000 epochs. When training a NF absent of the SoftFlow framework we add noise of the magnitude 0.02 to the data. When training with the SoftFlow framework, we follow the original implementation [95] and use a maximum noise standard deviation of 0.075 and a minimum noise standard deviation of 0.0.

General experiments on autoencoding in Sec. 5.3.2. We train the RealNVP for 1500 epochs using a batch size of 128 and the Adam optimizer with an initial learning rate of 6.4×10^{-4} which we multiply with 0.25 after 1200 and 1400 epochs. Additionally, we vary the magnitude of noise added to the point clouds over the process of the training. Initially, we train with a Gaussian noise of standard deviation $\sigma = 0.25$ for the first 100 epochs. Then, we exponentially decay the standard deviation of the noise to $\sigma = 0.02$ between epoch 100 and epoch 1200. Afterwards we fix $\sigma = 0.02$.

6

CONCLUSION

We subsequently summarize our contributions (see Sec. 6.1) and discuss promising future research directions (see Sec. 6.2).

6.1 CONTRIBUTIONS

Uncertainty quantification based on hidden representations. Chapter 3 introduces an approach to quantify the epistemic uncertainty of a NN based on the distribution of its hidden representations (see Sec. 3.2.1). We demonstrate that this can yield epistemic uncertainty efficiently and agnostic of the underlying architecture. Moreover, we propose a simple approach to regularize hidden representations that trades OOD detection performance off with predictive performance (see Sec. 3.3.4).

Deterministic Uncertainty Methods. Chapter 4 provides the first overview of DUMs (see Sec. 4.2). We demonstrate that the uncertainty obtained by DUMs is often not well calibrated - especially in the case of methods relying on explicit generative models trained on the hidden representations of a NN (see Sec. 4.3.1.1 and 4.3.2.2). Furthermore, we investigate the impact of the regularization strength of various approaches and observe, surprisingly, that the regularization strength when enforcing distance-aware representations does not correlate well with OOD detection performance (see Sec. 4.3.1.3 and 4.5.B.1). Lastly, we also scale DUMs to large dense prediction tasks (see Sec. 4.3.2)

Improving the perceptual quality of samples from NFs. Chapter 5 introduces an approach to sample from the underlying data manifold given a NF trained on the data perturbed by noise and knowledge

of the noise itself (see Sec. 5.2.2). We focus on the application to 3D point clouds but further demonstrate that our method also yields promising results on images. Moreover, we further propose a refined version of our method taylored to 3D point clouds based on Poisson surface reconstruction (see Sec. 5.2.3). Hereby, we show that the likelihood, and its gradient, learned by the NF benefit Poisson surface reconstruction from point clouds (see Sec. 5.3.2.2).

6.2 OUTLOOK

Regularization of DUMs. Chapter 3 and 4 demonstrate the importance of regularizing NNs when estimating uncertainty based on a point estimate of the parameters of a NN. In particular, we note that both, DDU [21] and MIR [20], estimate epistemic uncertainty of test data using the log-likelihood of hidden representations under a GMM trained on representations of the training dataset. They only differ in their regularization technique. However, MIR demonstrates significantly stronger calibration (see Tab. 4.3) and mildly better OOD detection performance (see Tab. 4.4). Thus, the investigation of novel methods for regularizing DUMs aimed at improving the calibration denotes a promising future research direction.

Large pretrained NNs. It has recently been shown that very large NNs pretrained on massive amounts of data are able to predict high quality uncertainty simply by using the entropy of the predicted softmax values [179]. In fact, the authors even used such a pretrained model in combination with SNGP and demonstrated its scaling behavior. It would be interesting to investigate how such pretraining impacts calibration on OOD data for various DUMs.

Log-likelihood of NFs. Chapter 5 shows that the likelihood obtained from NFs can improve Poisson surface reconstruction. It would be interesting to investigate whether this is also the case for other tasks which require normals or the removal of outliers. In fact, NFs [89]

and score matching [175] have been recently successfully applied to point cloud denoising.

BIBLIOGRAPHY

1. Gal, Y., Islam, R. & Ghahramani, Z. *Deep bayesian active learning with image data* in *Proceedings of the 34th International Conference on Machine Learning-Volume 70* (2017), 1183.
2. Da Silva, F. L., Hernandez-Leal, P., Kartal, B. & Taylor, M. E. *Uncertainty-aware action advising for deep reinforcement learning agents* in *Proceedings of the AAAI conference on artificial intelligence* **34** (2020), 5792.
3. Lütjens, B., Everett, M. & How, J. P. *Safe reinforcement learning with model uncertainty estimates* in *2019 International Conference on Robotics and Automation (ICRA)* (2019), 8662.
4. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., *et al.* Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences* **114**, 3521 (2017).
5. Ebrahimi, S., Elhoseiny, M., Darrell, T. & Rohrbach, M. *Uncertainty-guided Continual Learning with Bayesian Neural Networks* in *International Conference on Learning Representations* (2019).
6. Blum, H., Sarlin, P.-E., Nieto, J., Siegwart, R. & Cadena, C. The fishyscapes benchmark: Measuring blind spots in semantic segmentation. *arXiv preprint arXiv:1904.03215* (2019).
7. Der Kiureghian, A. & Ditlevsen, O. Aleatory or epistemic? Does it matter? *Structural safety* **31**, 105 (2009).
8. Kendall, A. & Gal, Y. *What uncertainties do we need in bayesian deep learning for computer vision?* in *Advances in neural information processing systems* (2017), 5574.

9. MacKay, D. J. A practical Bayesian framework for backpropagation networks. *Neural computation* **4**, 448 (1992).

10. Gal, Y. Uncertainty in deep learning. *University of Cambridge* **1**, 3 (2016).

11. Lakshminarayanan, B., Pritzel, A. & Blundell, C. *Simple and scalable predictive uncertainty estimation using deep ensembles* in *Advances in neural information processing systems* (2017), 6402.

12. Wen, Y., Tran, D. & Ba, J. Batchensemble: an alternative approach to efficient ensemble and lifelong learning. *arXiv preprint arXiv:2002.06715* (2020).

13. Depeweg, S., Hernandez-Lobato, J.-M., Doshi-Velez, F. & Udluft, S. *Decomposition of uncertainty in Bayesian deep learning for efficient and risk-sensitive learning* in *International Conference on Machine Learning* (2018), 1184.

14. Postels, J., Ferroni, F., Coskun, H., Navab, N. & Tombari, F. *Sampling-free Epistemic Uncertainty Estimation Using Approximated Variance Propagation* in *Proceedings of the IEEE International Conference on Computer Vision* (2019), 2931.

15. Loquercio, A., Segu, M. & Scaramuzza, D. A general framework for uncertainty estimation in deep learning. *IEEE Robotics and Automation Letters* **5**, 3153 (2020).

16. Gal, Y. & Ghahramani, Z. *Dropout as a bayesian approximation: Representing model uncertainty in deep learning* in *international conference on machine learning* (2016), 1050.

17. Teye, M., Azizpour, H. & Smith, K. *Bayesian Uncertainty Estimation for Batch Normalized Deep Networks* in *International Conference on Machine Learning* (2018), 4907.

18. Maddox, W. J., Izmailov, P., Garipov, T., Vetrov, D. P. & Wilson, A. G. A simple baseline for bayesian uncertainty in deep learning. *Advances in neural information processing systems* **32** (2019).

19. Van Amersfoort, J., Smith, L., Teh, Y. W. & Gal, Y. *Uncertainty estimation using a single deep deterministic neural network* in *International Conference on Machine Learning* (2020), 9690.

20. Postels, J., Blum, H., Strümpler, Y., Cadena, C., Siegwart, R., Van Gool, L. & Tombari, F. The Hidden Uncertainty in a Neural Networks Activations. *arXiv preprint arXiv:2012.03082* (2020).

21. Mukhoti, J., Kirsch, A., van Amersfoort, J., Torr, P. H. & Gal, Y. Deterministic Neural Networks with Appropriate Inductive Biases Capture Epistemic and Aleatoric Uncertainty. *arXiv preprint arXiv:2102.11582* (2021).

22. Postels, J., Segu, M., Sun, T., Sieber, L. D., Van Gool, L., Yu, F. & Tombari, F. *On the practicality of deterministic epistemic uncertainty* in *Proceedings of the 39th International Conference on Machine Learning* **162** (2022), 17870.

23. Venkataramanan, A., Benbihi, A., Laviale, M. & Pradalier, C. *Gaussian Latent Representations for Uncertainty Estimation using Mahalanobis Distance in Deep Classifiers* in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2023), 4488.

24. Shin, S., Bae, W., Noh, J. & Choi, S. *Towards Explainable Computer Vision Methods via Uncertainty Activation Map* in *Asian Conference on Pattern Recognition* (2023), 1.

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

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

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

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

29. Postels, J., Liu, M., Spezialetti, R., Van Gool, L. & Tombari, F. *Go with the flows: Mixtures of normalizing flows for point cloud generation and reconstruction* in *2021 International Conference on 3D Vision (3DV)* (2021), 1249.

30. Kingma, D. P. & Dhariwal, P. *Glow: Generative flow with invertible 1x1 convolutions* in *Advances in neural information processing systems* (2018), 10215.

31. Yang, G., Huang, X., Hao, Z., Liu, M.-Y., Belongie, S. & Hariharan, B. *Pointflow: 3d point cloud generation with continuous normalizing flows* in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019), 4541.

32. Klokov, R., Boyer, E. & Verbeek, J. *Discrete point flow networks for efficient point cloud generation* in *European Conference on Computer Vision* (2020), 694.

33. Neal, R. M. *Bayesian learning for neural networks* (Springer Science & Business Media, 2012).

34. Neal, R. M. *et al.* MCMC using Hamiltonian dynamics. *Handbook of markov chain monte carlo* **2**, 2 (2011).

35. Hinton, G. E. & Van Camp, D. *Keeping the neural networks simple by minimizing the description length of the weights* in *Proceedings of the sixth annual conference on Computational learning theory* (1993), 5.

36. Kingma, D. P., Salimans, T. & Welling, M. *Variational dropout and the local reparameterization trick* in *Advances in neural information processing systems* (2015), 2575.

37. Zhang, G., Sun, S., Duvenaud, D. & Grosse, R. Noisy natural gradient as variational inference. *arXiv preprint arXiv:1712.02390* (2017).

38. Zhang, G., Sun, S., Duvenaud, D. & Grosse, R. *Noisy natural gradient as variational inference* in *International Conference on Machine Learning* (2018), 5852.

39. Osband, I., Wen, Z., Asghari, M., Ibrahimi, M., Lu, X. & Van Roy, B. Epistemic neural networks. *arXiv preprint arXiv:2107.08924* (2021).

40. Ritter, H., Botev, A. & Barber, D. *A scalable laplace approximation for neural networks* in *6th International Conference on Learning Representations, ICLR 2018-Conference Track Proceedings* **6** (2018).

41. Lee, J., Humt, M., Feng, J. & Triebel, R. *Estimating model uncertainty of neural networks in sparse information form* in *International Conference on Machine Learning* (2020), 5702.

42. Sharma, A., Azizan, N. & Pavone, M. Sketching Curvature for Efficient Out-of-Distribution Detection for Deep Neural Networks. *arXiv preprint arXiv:2102.12567* (2021).

43. Rupprecht, C., Laina, I., DiPietro, R., Baust, M., Tombari, F., Navab, N. & Hager, G. D. *Learning in an uncertain world: Representing ambiguity through multiple hypotheses* in *Proceedings of the IEEE International Conference on Computer Vision* (2017), 3591.

44. Dusenberry, M., Jerfel, G., Wen, Y., Ma, Y., Snoek, J., Heller, K., Lakshminarayanan, B. & Tran, D. *Efficient and scalable bayesian neural nets with rank-1 factors* in *International conference on machine learning* (2020), 2782.

45. Havasi, M., Jenatton, R., Fort, S., Liu, J. Z., Snoek, J., Lakshminarayanan, B., Dai, A. M. & Tran, D. *Training independent subnetworks for robust prediction* in *International Conference on Learning Representations* (2020).

46. Rame, A., Sun, R. & Cord, M. MixMo: Mixing Multiple Inputs for Multiple Outputs via Deep Subnetworks. *Proceedings of the IEEE International Conference on Computer Vision* (2021).

47. Durasov, N., Bagautdinov, T., Baque, P. & Fua, P. *Masksembles for uncertainty estimation* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), 13539.

48. Wang, H., Shi, X. & Yeung, D.-Y. Natural-parameter networks: A class of probabilistic neural networks. *Advances in neural information processing systems* **29** (2016).

49. Haußmann, M., Hamprecht, F. A. & Kandemir, M. *Sampling-free variational inference of bayesian neural networks by variance backpropagation* in *Uncertainty in Artificial Intelligence* (2020), 563.

50. Bishop, C. M. Mixture density networks. *Aston University* (1994).

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

52. Lee, K., Lee, H., Lee, K. & Shin, J. Training confidence-calibrated classifiers for detecting out-of-distribution samples. *arXiv preprint arXiv:1711.09325* (2017).

53. Guo, C., Pleiss, G., Sun, Y. & Weinberger, K. Q. *On calibration of modern neural networks* in *Proceedings of the 34th International Conference on Machine Learning-Volume 70* (2017), 1321.

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

55. Van Amersfoort, J., Smith, L., Teh, Y. W. & Gal, Y. Simple and Scalable Epistemic Uncertainty Estimation Using a Single Deep Deterministic Neural Network. *arXiv preprint arXiv:2003.02037* (2020).

56. Van Amersfoort, J., Smith, L., Jesson, A., Key, O. & Gal, Y. Improving Deterministic Uncertainty Estimation in Deep Learning for Classification and Regression. *arXiv preprint arXiv:2102.11409* (2021).

57. Huang, H., van Amersfoort, J. & Gal, Y. Decomposing Representations for Deterministic Uncertainty Estimation. *arXiv preprint arXiv:2112.00856* (2021).

58. Ren, J., Fort, S., Liu, J., Roy, A. G., Padhy, S. & Lakshminarayanan, B. A simple fix to mahalanobis distance for improving near-ood detection. *arXiv preprint arXiv:2106.09022* (2021).

59. Pimentel, M. A., Clifton, D. A., Clifton, L. & Tarassenko, L. A review of novelty detection. *Signal Processing* **99**, 215 (2014).

60. Chalapathy, R. & Chawla, S. Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407* (2019).

61. Schlegl, T., Seeböck, P., Waldstein, S. M., Schmidt-Erfurth, U. & Langs, G. *Unsupervised anomaly detection with generative adversarial networks to guide marker discovery* in *International conference on information processing in medical imaging* (2017), 146.

62. Li, D., Chen, D., Goh, J. & Ng, S.-K. Anomaly detection with generative adversarial networks for multivariate time series. *arXiv preprint arXiv:1809.04758* (2018).

63. Škvára, V., Pevnỳ, T. & Šmídl, V. Are generative deep models for novelty detection truly better? *arXiv preprint arXiv:1807.05027* (2018).

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

65. Choi, H., Jang, E. & Alemi, A. A. Waic, but why? generative ensembles for robust anomaly detection. *arXiv preprint arXiv:1810.01392* (2018).

66. Nalisnick, E., Matsukawa, A., Teh, Y. W. & Lakshminarayanan, B. Detecting Out-of-Distribution Inputs to Deep Generative Models Using Typicality. *arXiv preprint arXiv:1906.02994* (2019).

67. Morningstar, W. R., Ham, C., Gallagher, A. G., Lakshminarayanan, B., Alemi, A. A. & Dillon, J. V. Density of States Estimation for Out-of-Distribution Detection. *arXiv preprint arXiv:2006.09273* (2020).

68. Ruff, L., Vandermeulen, R., Goernitz, N., Deecke, L., Siddiqui, S. A., Binder, A., Müller, E. & Kloft, M. *Deep one-class classification* in *International conference on machine learning* (2018), 4393.

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

70. Lee, K., Lee, K., Lee, H. & Shin, J. *A simple unified framework for detecting out-of-distribution samples and adversarial attacks* in *Advances in Neural Information Processing Systems* (2018), 7167.

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

72. Feng, J., Lee, J., Geisler, S., Gunnemann, S. & Triebel, R. Topology-Matching Normalizing Flows for Out-of-Distribution Detection in Robot Learning. *arXiv preprint arXiv:2311.06481* (2023).

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

74. Alemi, A. A., Fischer, I., Dillon, J. V. & Murphy, K. Deep variational information bottleneck. *Proceedings of the International Conference on Learning Representations (ICLR) 2017* (2016).

75. Winkens, J., Bunel, R., Roy, A. G., Stanforth, R., Natarajan, V., Ledsam, J. R., MacWilliams, P., Kohli, P., Karthikesalingam, A., Kohl, S., *et al.* Contrastive training for improved out-of-distribution detection. *arXiv preprint arXiv:2007.05566* (2020).

76. Chen, T., Kornblith, S., Norouzi, M. & Hinton, G. *A simple framework for contrastive learning of visual representations* in *International conference on machine learning* (2020), 1597.

77. Achlioptas, P., Diamanti, O., Mitliagkas, I. & Guibas, L. *Learning representations and generative models for 3d point clouds* in *International conference on machine learning* (2018), 40.

78. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. & Bengio, Y. *Generative adversarial nets* in *Advances in neural information processing systems* (2014), 2672.

79. Groueix, T., Fisher, M., Kim, V. G., Russell, B. C. & Aubry, M. *A papier-mâché approach to learning 3d surface generation* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), 216.

80. Cai, R., Yang, G., Averbuch-Elor, H., Hao, Z., Belongie, S., Snavely, N. & Hariharan, B. *Learning gradient fields for shape generation* in *European Conference on Computer Vision* (2020), 364.

81. Hyvärinen, A. & Dayan, P. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research* **6** (2005).

82. Welling, M. & Teh, Y. W. *Bayesian learning via stochastic gradient Langevin dynamics* in *Proceedings of the 28th international conference on machine learning (ICML-11)* (2011), 681.

83. Song, Y. & Ermon, S. Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems* **32** (2019).

84. Ho, J., Jain, A. & Abbeel, P. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems* **33**, 6840 (2020).

85. Luo, S. & Hu, W. *Diffusion probabilistic models for 3d point cloud generation* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), 2837.

86. Pumarola, A., Popov, S., Moreno-Noguer, F. & Ferrari, V. *C-flow: Conditional generative flow models for images and 3d point clouds* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), 7949.

87. Wei, Y., Vosselman, G. & Yang, M. Y. Flow-based GAN for 3D point cloud generation from a single image. *Proceedings of the The British Machine Vision Conference* (2022).

88. Mao, A., Du, Z., Hou, J., Duan, Y., Liu, Y.-j. & He, Y. PU-Flow: A point cloud upsampling network with normalizing flows. *IEEE Transactions on Visualization and Computer Graphics* (2022).

89. Mao, A., Du, Z., Wen, Y.-H., Xuan, J. & Liu, Y.-J. *Pd-flow: A point cloud denoising framework with normalizing flows* in *European Conference on Computer Vision* (2022), 398.

90. Kimura, T., Matsubara, T. & Uehara, K. *Chartpointflow for topology-aware 3d point cloud generation* in *Proceedings of the 29th ACM International Conference on Multimedia* (2021), 1396.

91. Gemici, M. C., Rezende, D. & Mohamed, S. Normalizing flows on riemannian manifolds. *arXiv preprint arXiv:1611.02304* (2016).

92. Cunningham, E., Zabounidis, R., Agrawal, A., Fiterau, I. & Sheldon, D. Normalizing flows across dimensions. *arXiv preprint arXiv:2006.13070* (2020).

93. Brehmer, J. & Cranmer, K. Flows for simultaneous manifold learning and density estimation. *Advances in Neural Information Processing Systems* **33**, 442 (2020).

94. Mathieu, E. & Nickel, M. Riemannian continuous normalizing flows. *Advances in Neural Information Processing Systems* **33**, 2503 (2020).

95. Kim, H., Lee, H., Kang, W. H., Lee, J. Y. & Kim, N. S. Softflow: Probabilistic framework for normalizing flow on manifolds. *Advances in Neural Information Processing Systems* **33**, 16388 (2020).

96. Horvat, C. & Pfister, J.-P. Denoising Normalizing Flow. *Advances in Neural Information Processing Systems* **34** (2021).

97. Cramer, E., Rauh, F., Mitsos, A., Tempone, R. & Dahmen, M. Nonlinear isometric manifold learning for injective normalizing flows. *arXiv preprint arXiv:2203.03934* (2022).

98. Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., *et al.* Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine* **29**, 82 (2012).

99. Redmon, J., Divvala, S., Girshick, R. & Farhadi, A. *You Only Look Once: Unified, Real-Time Object Detection* in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, Las Vegas, NV, USA, 2016), 779.

100. Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J. & Mané, D. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565* (2016).

101. Bozhinoski, D., Di Ruscio, D., Malavolta, I., Pelliccione, P. & Crnkovic, I. Safety for mobile robotic systems: A systematic mapping study from a software engineering perspective. *Journal of Systems and Software* **151**, 150 (2019).

102. Janai, J., Güney, F., Behl, A. & Geiger, A. Computer vision for autonomous vehicles: Problems, datasets and state-of-the-art. *arXiv preprint arXiv:1704.05519* (2017).

103. Wen, Y., Tran, D. & Ba, J. BatchEnsemble: An Alternative Approach to Efficient Ensemble and Lifelong Learning. *ICLR* (2020).

104. Kiureghian, A. D. & Ditlevsen, O. Aleatory or epistemic? Does it matter? *Structural Safety* **31**, 105 (2009).

105. He, K., Zhang, X., Ren, S. & Sun, J. *Deep residual learning for image recognition* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), 770.

106. Hendrycks, D. & Dietterich, T. Benchmarking Neural Network Robustness to Common Corruptions and Perturbations. *Proceedings of the International Conference on Learning Representations* (2019).

107. LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* **86**, 2278 (1998).

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

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

110. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B. & Ng, A. Y. Reading Digits in Natural Images with Unsupervised Feature Learning. *Citeseer* (2011).

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

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

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

114. Snoek, J., Ovadia, Y., Fertig, E., Lakshminarayanan, B., Nowozin, S., Sculley, D., Dillon, J., Ren, J. & Nado, Z. *Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift* in *Advances in Neural Information Processing Systems* (2019), 13969.

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

116. Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

117. Blanchard, G., Lee, G. & Scott, C. Generalizing from several related classification tasks to a new unlabeled sample. *Advances in neural information processing systems* **24**, 2178 (2011).

118. Muandet, K., Balduzzi, D. & Schölkopf, B. *Domain generalization via invariant feature representation* in *International Conference on Machine Learning* (2013), 10.

119. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. & Fergus, R. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199* (2013).

120. Wenzel, F., Roth, K., Veeling, B. S., Światkowski, J., Tran, L., Mandt, S., Snoek, J., Salimans, T., Jenatton, R. & Nowozin, S. How good is the bayes posterior in deep neural networks really? *arXiv preprint arXiv:2002.02405* (2020).

121. Wu, M. & Goodman, N. A Simple Framework for Uncertainty in Contrastive Learning. *arXiv preprint arXiv:2010.02038* (2020).

122. Charpentier, B., Zügner, D. & Günnemann, S. Posterior Network: Uncertainty Estimation without OOD Samples via Density-Based Pseudo-Counts. *Advances in Neural Information Processing Systems* **33**, 1356 (2020).

123. Charpentier, B., Borchert, O., Zügner, D., Geisler, S. & Günnemann, S. Natural Posterior Network: Deep Bayesian Predictive Uncertainty for Exponential Family Distributions. *arXiv preprint arXiv:2105.04471* (2021).

124. Gasperini, S., Haug, J., Mahani, M.-A. N., Marcos-Ramiro, A., Navab, N., Busam, B. & Tombari, F. CertainNet: Sampling-free uncertainty estimation for object detection. *IEEE Robotics and Automation Letters* **7**, 698 (2021).

125. Ardizzone, L., Kruse, J., Wirkert, S., Rahner, D., Pellegrini, E. W., Klessen, R. S., Maier-Hein, L., Rother, C. & Köthe, U. Analyzing inverse problems with invertible neural networks. *arXiv preprint arXiv:1808.04730* (2018).

126. Nalisnick, E., Matsukawa, A., Teh, Y. W., Gorur, D. & Lakshminarayanan, B. *Hybrid models with deep and invertible features* in *International Conference on Machine Learning* (2019), 4723.

127. Ardizzone, L., Mackowiak, R., Rother, C. & Köthe, U. Training Normalizing Flows with the Information Bottleneck for Competitive Generative Classification. *Advances in Neural Information Processing Systems* **33** (2020).

128. Peters, J., Janzing, D. & Schölkopf, B. *Elements of causal inference: foundations and learning algorithms* (The MIT Press, 2017).

129. Segù, M., Tonioni, A. & Tombari, F. Batch Normalization Embeddings for Deep Domain Generalization. *arXiv preprint arXiv:2011.12672* (2020).

130. Obukhov, A., Rakhuba, M., Liniger, A., Huang, Z., Georgoulis, S., Dai, D. & Van Gool, L. *Spectral Tensor Train Parameterization of Deep Learning Layers* in *International Conference on Artificial Intelligence and Statistics* (2021), 3547.

131. Miyato, T., Kataoka, T., Koyama, M. & Yoshida, Y. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957* (2018).

132. Singla, S. & Feizi, S. Bounding singular values of convolution layers. *arXiv preprint arXiv:1911.10258* (2019).

133. Rosca, M., Weber, T., Gretton, A. & Mohamed, S. A case for new neural network smoothness constraints. *arXiv preprint arXiv:2012.07969* (2020).

134. Sedghi, H., Gupta, V. & Long, P. M. The singular values of convolutional layers. *arXiv preprint arXiv:1805.10408* (2018).

135. Smith, L., van Amersfoort, J., Huang, H., Roberts, S. & Gal, Y. Can convolutional ResNets approximately preserve input distances? A frequency analysis perspective. *arXiv preprint arXiv:2106.02469* (2021).

136. Oord, A. v. d., Li, Y. & Vinyals, O. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748* (2018).

137. Jacobsen, J.-H., Smeulders, A. & Oyallon, E. i-revnet: Deep invertible networks. *arXiv preprint arXiv:1802.07088* (2018).

138. Behrmann, J., Grathwohl, W., Chen, R., Duvenaud, D. & Jacobsen, J. Invertible residual networks. arXiv e-prints. *arXiv preprint arXiv:1811.00995* (2018).

139. Jain, M., Lahlou, S., Nekoei, H., Butoi, V., Bertin, P., Rector-Brooks, J., Korablyov, M. & Bengio, Y. DEUP: Direct Epistemic Uncertainty Prediction. *arXiv preprint arXiv:2102.08501* (2021).

140. Nalisnick, E., Matsukawa, A., Teh, Y. W., Gorur, D. & Lakshminarayanan, B. *Do Deep Generative Models Know What They Don't Know?* in *International Conference on Learning Representations* (2018).

141. Rasmussen, C. E. *Gaussian processes in machine learning* in *Summer school on machine learning* (2003), 63.

142. Titsias, M. *Variational learning of inducing variables in sparse Gaussian processes* in *Artificial intelligence and statistics* (2009), 567.

143. Hensman, J., Matthews, A. & Ghahramani, Z. *Scalable variational Gaussian process classification* in *Artificial Intelligence and Statistics* (2015), 351.

144. Burt, D., Rasmussen, C. E. & Van Der Wilk, M. *Rates of convergence for sparse variational Gaussian process regression* in *International Conference on Machine Learning* (2019), 862.

145. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A. & Koltun, V. *CARLA: An Open Urban Driving Simulator* in *Proceedings of the 1st Annual Conference on Robot Learning* (2017), 1.

146. Naeini, M. P., Cooper, G. & Hauskrecht, M. *Obtaining well calibrated probabilities using bayesian binning* in *Proceedings of the AAAI Conference on Artificial Intelligence* **29** (2015).

147. BRIER, G. W. VERIFICATION OF FORECASTS EXPRESSED IN TERMS OF PROBABILITY. *Monthly Weather Review* **78**, 1 (1950).

148. Vuk, M. & Curk, T. ROC curve, lift chart and calibration plot. *Metodoloski zvezki* **3**, 89 (2006).

149. Krizhevsky, A., Nair, V. & Hinton, G. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html* **55**, 5 (2014).

150. LeCun, Y. The MNIST database of handwritten digits. *http://yann. lecun. com/exdb/mnist/* (1998).

151. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S. & Schiele, B. *The Cityscapes Dataset for Semantic Urban Scene Understanding* in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016).

152. Michaelis, C., Mitzkus, B., Geirhos, R., Rusak, E., Bringmann, O., Ecker, A. S., Bethge, M. & Brendel, W. Benchmarking robustness in object detection: Autonomous driving when winter is coming. *arXiv preprint arXiv:1907.07484* (2019).

153. Sun, T., Segù, M., Postels, J., Wang, Y., Van Gool, L., Schiele, B., Tombari, F. & Yu, F. *SHIFT: A Synthetic Driving Dataset for Continuous Multi-Task Domain Adaptation* in *Computer Vision and Pattern Recognition* (2022).

154. Yu, F. & Koltun, V. *Multi-scale context aggregation by dilated convolutions* in *International Conference on Learning Representations (ICLR)* (2016).

155. Yu, F., Koltun, V. & Funkhouser, T. *Dilated Residual Networks* in *Computer Vision and Pattern Recognition (CVPR)* (2017).

156. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V. & Courville, A. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028* (2017).

157. Daunizeau, J. Semi-analytical approximations to statistical moments of sigmoid and softmax mappings of normal variables. *arXiv preprint arXiv:1703.00091* (2017).

158. Kingma, D. P. & Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

159. Chen, X. & Konukoglu, E. Unsupervised detection of lesions in brain MRI using constrained adversarial auto-encoders. *arXiv preprint arXiv:1806.04972* (2018).

160. Lugmayr, A., Danelljan, M., Gool, L. V. & Timofte, R. *Srflow: Learning the super-resolution space with normalizing flow* in *European conference on computer vision* (2020), 715.

161. Arroyo, D. M., Postels, J. & Tombari, F. *Variational transformer networks for layout generation* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), 13642.

162. Grathwohl, W., Chen, R. T., Bettencourt, J., Sutskever, I. & Duvenaud, D. *FFJORD: Free-Form Continuous Dynamics for Scalable Reversible Generative Models* in *International Conference on Learning Representations* (2018).

163. Behrmann, J., Grathwohl, W., Chen, R. T., Duvenaud, D. & Jacobsen, J.-H. *Invertible residual networks* in *International Conference on Machine Learning* (2019), 573.

164. Kanwar, G., Albergo, M. S., Boyda, D., Cranmer, K., Hackett, D. C., Racaniere, S., Rezende, D. J. & Shanahan, P. E. Equivariant flow-based sampling for lattice gauge theory. *Physical Review Letters* **125**, 121601 (2020).

165. Kazhdan, M., Bolitho, M. & Hoppe, H. *Poisson surface reconstruction* in *Proceedings of the fourth Eurographics symposium on Geometry processing* **7** (2006).

166. Berger, M., Tagliasacchi, A., Seversky, L. M., Alliez, P., Guennebaud, G., Levine, J. A., Sharf, A. & Silva, C. T. *A survey of surface reconstruction from point clouds* in *Computer Graphics Forum* **36** (2017), 301.

167. Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L. & Yu, F. *ShapeNet: An Information-Rich 3D Model Repository* tech. rep. arXiv:1512.03012 [cs.GR] (Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015).

168. Knapitsch, A., Park, J., Zhou, Q.-Y. & Koltun, V. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)* **36**, 1 (2017).

169. Qi, C. R., Su, H., Mo, K. & Guibas, L. J. *Pointnet: Deep learning on point sets for 3d classification and segmentation* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), 652.

170. Ramachandran, P., Zoph, B. & Le, Q. V. Searching for activation functions. *arXiv preprint arXiv:1710.05941* (2017).

171. Zhou, Q.-Y., Park, J. & Koltun, V. Open3D: A Modern Library for 3D Data Processing. *arXiv:1801.09847* (2018).

172. Liu, Z., Luo, P., Wang, X. & Tang, X. *Deep Learning Face Attributes in the Wild* in *Proceedings of International Conference on Computer Vision (ICCV)* (2015).

173. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B. & Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems* **30** (2017).

174. Park, J. J., Florence, P., Straub, J., Newcombe, R. & Lovegrove, S. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. *Conference on Computer Vision and Pattern Recognition (CVPR)* (2019).

175. Luo, S. & Hu, W. *Score-based point cloud denoising* in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), 4583.

176. Schall, O., Belyaev, A. & Seidel, H.-P. *Robust filtering of noisy scattered point data* in *Proceedings Eurographics/IEEE VGTC Symposium Point-Based Graphics, 2005.* (2005), 71.

177. Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I. & Welling, M. Improved variational inference with inverse autoregressive flow. *Advances in neural information processing systems* **29** (2016).

178. Perez, E., Strub, F., De Vries, H., Dumoulin, V. & Courville, A. *Film: Visual reasoning with a general conditioning layer* in *Proceedings of the AAAI Conference on Artificial Intelligence* **32** (2018).

179. Tran, D., Liu, J., Dusenberry, M. W., Phan, D., Collier, M., Ren, J., Han, K., Wang, Z., Mariet, Z., Hu, H., *et al.* Plex: Towards reliability using pretrained large model extensions. *arXiv preprint arXiv:2207.07411* (2022).

CURRICULUM VITAE

PERSONAL DATA

Name	Janis Gerhard Postels
Date of Birth	June 10, 1990
Place of Birth	Wilhelmshaven, Germany
Citizen of	Germany

EDUCATION

2020 – 2024	ETH Zürich, Zürich, Switzerland *Final degree:* Doctor of Sciences
2017 – 2019	TU München, München, Germany *Final degree:* Master of Science
2011 – 2016	Universität Heidelberg, Heidelberg, Germany *Final degree:* Bachelor of Science

PROFESSIONAL EXPERIENCE

2023	NLP Quantitative Research Intern, *G-Research*, London, United Kingdom
2019 – 2020	Research Scientist, *Qualcomm AI Research*, Amsterdam, Netherlands
2018 – 2019	Master Thesis, *Autonomous Intelligent Driving*, München, Germany
2018	Working Student, *E-bot7*, München, Germany
2018	Working Student, *Allianz*, München, Germany
2016-2017	IT Consultant, *SPECTRUM AG*, Stuttgart, Germany

PUBLICATIONS

Janis Postels, Yannick Strümpler, Klara Reichard, Luc Van Gool and Federico Tombari. "3D Compression Using Neural Fields". In: *arXiv e-prints, abs/2311.13009* 2023

Mengya Liu, Ajad Chhatkuli, Janis Postels, Luc Van Gool and Federico Tombari. "Unsupervised Template Warp Consistency for Implicit Surface Correspondences". In: *Computer Graphics Forum* 2023

Yannick Strümpler*, Janis Postels*, Ren Yang, Luc Van Gool and Federico Tombari. "Implicit Neural Representations for Image Compression". In: *ECCV* 2022

Janis Postels, Martin Danelljan, Luc Van Gool, and Federico Tombari. "ManiFlow: Implicitly Representing Manifolds with Normalizing Flows". In: *3DV* 2022

Janis Postels*, Mattia Segu*, Tao Sun, Luca Sieber, Luc Van Gool, Fisher Yu and Federico Tombari. "On the Practicality of Deterministic Epistemic Uncertainty". In: *ICML* 2022

Tao Sun*, Mattia Segu*, Janis Postels, Yuxuan Wang, Luc Van Gool, Bernt Schiele, Federico Tombari and Fisher Yu. "SHIFT: A Synthetic Driving Dataset for Continuous Multi-Task Domain Adaptation". In: *CVPR* 2022

Janis Postels*, Mengya Liu*, Riccardo Spezialetti, Luc Van Gool and Federico Tombari. "Go with the Flows: Mixtures of Normalizing Flows for Point Cloud Generation and Reconstruction". In: *3DV* 2021

Diego Martin Arroyo, Janis Postels, and Federico Tombari. "Variational Transformer Networks for Layout Generation". In: *CVPR* 2021

Matthäus Heer, Janis Postels, Xiaoran Chen, Ender Konukoglu and Shadi Albarqouni. "The OOD Blind Spot of Unsupervised Anomaly Detection". In: *MIDL* 2021

Janis Postels*, Hermann Blum*, Yannick Strümpler, Cesar Cadena, Roland Siegwart, Luc Van Gool, and Federico Tombari. "The Hidden Uncertainty in a Neural Networks Activations". In: *arXiv e-prints, abs/2012.03082*, 2020

Janis Postels, Francesco Ferroni, Huseyin Coskun, Nassir Navab and Federico Tombari. "Sampling-free Epistemic Uncertainty Estimation Using Approximated Variance Propagation". In: *ICCV* 2019