

# Latent Discriminant Deterministic Uncertainty

Gianni Franchi<sup>1†</sup>, Xuanlong Yu<sup>1,2†</sup>, Andrei Bursuc<sup>3</sup>, Emanuel Aldea<sup>2</sup>, Severine Dubuisson<sup>4</sup>, and David Filliat<sup>1</sup>

<sup>1</sup> U2IS, ENSTA Paris, Institut polytechnique de Paris

<sup>2</sup> SATIE, Paris-Saclay University

<sup>3</sup> valeo.ai

<sup>4</sup> CNRS, LIS, Aix Marseille University

**Abstract.** Predictive uncertainty estimation is essential for deploying Deep Neural Networks in real-world autonomous systems. However, most successful approaches are computationally intensive. In this work, we attempt to address these challenges in the context of autonomous driving perception tasks. Recently proposed Deterministic Uncertainty Methods (DUM) can only partially meet such requirements as their scalability to complex computer vision tasks is not obvious. In this work we advance a scalable and effective DUM for high-resolution semantic segmentation, that relaxes the Lipschitz constraint typically hindering practicality of such architectures. We learn a discriminant latent space by leveraging a distinction maximization layer over an arbitrarily-sized set of trainable prototypes. Our approach achieves competitive results over Deep Ensembles, the state of the art for uncertainty prediction, on image classification, segmentation and monocular depth estimation tasks. Our code is available at <https://github.com/ENSTA-U2IS/LDU>.

**Keywords:** Deep neural networks · uncertainty estimation · out-of-distribution detection.

## 1 Introduction

Uncertainty estimation and robustness are essential for deploying Deep Neural Networks (DNN) in real-world systems with different levels of autonomy, ranging from simple driving assistance functions to fully autonomous vehicles. In addition to excellent predictive performance, DNNs are also expected to address different types of uncertainty (noisy, ambiguous or out-of-distribution samples, distribution shift, etc.), while ensuring real-time computational performance. These key and challenging requirements have stimulated numerous solutions and research directions leading to significant progress in this area [10, 45, 27, 43, 80, 57]. Yet, the best performing approaches are computationally expensive [45], while faster variants struggle to disentangle different types of uncertainty [23, 52, 56].

We study a promising new line of methods, termed deterministic uncertainty methods (DUMs) [66], that has recently emerged for estimating uncertainty in

---

† Equal contribution.

a computational efficient manner from a single forward pass [56, 77, 3, 65, 48]. In order to quantify uncertainty, these methods rely on some statistical or geometrical properties of the hidden features of the DNNs. While appealing for their good Out-of-Distribution (OOD) uncertainty estimations at low computational cost, they have been used mainly for classification tasks and their specific regularization is often unstable when training deeper DNNs [63]. We then propose a new DUM technique, based on a discriminative latent space that improves both scalability and flexibility. We achieve this by still following the principles of DUMs of learning a sensitive and smooth representation that mirrors well the input distribution, although not by enforcing directly the Lipschitz constraint.

Our DUM, dubbed Latent Discriminant deterministic Uncertainty (LDU), is based on a DNN imbued with a set of prototypes over its latent representations. These prototypes act like a memory that allows to better analyze features from new images in light of the “knowledge” acquired by the DNN from the training data. Various forms of prototypes have been studied for anomaly detection in the past [29] and they often take the shape of a dictionary of representative features. Instead, LDU is trained to learn the optimal prototypes, such that this distance improves the accuracy and the uncertainty prediction. Indeed to train LDU, we introduce a confidence-based loss that learns to predict the error of the DNN given the data. ConfidNet [15] and SLURP [84] have shown that we can train an auxiliary network to predict the uncertainty, at the cost of a more complex training pipeline and more inference steps. Here LDU is lighter, faster and needs only a single forward pass. LDU can be used as a pluggable learning layer on top of DNNs. We demonstrate that LDU avoids feature collapse and can be applied to multiple computer vision tasks. In addition, LDU improves the prediction accuracy of the baseline DNN without LDU.

**Contributions.** To summarize, our contributions are as follows: **(1)** LDU (Latent Discriminant deterministic Uncertainty): an efficient and scalable DUM approach for uncertainty quantification. **(2)** A study of LDU’s properties against feature collapse. **(3)** Evaluations of LDU on a range of computer vision tasks and settings (image classification, semantic segmentation, depth estimation) and the implementation of a set of powerful baselines to further encourage research in this area.

## 2 Related work

In this section, we focus on the related works from two perspectives: uncertainty quantification algorithms applied to computer vision tasks and prototype learning on DNNs. In Table 1, we list various uncertainty quantification algorithms according to different computer vision tasks.

### 2.1 Uncertainty estimation for computer vision tasks

**Uncertainty for image classification and semantic segmentation.** Quantifying uncertainty for classification and semantic segmentation can be done with

Uncertainty estimation methods	Image Classification	Computer vision tasks	
		Semantic Segmentation	1D/2D Regression
Bayesian/Ensemble based methods	Rank-1 BNN [20], PBP [37], Deep Ensembles [45], Bayes by Backprop [10], MultiSWAG [81]	Deep Ensembles [45], Bayes by Backprop [10], MultiSWAG [81]	FlowNetH [40], PBP [37], Deep Ensembles [45], Bayes by Backprop [10], MultiSWAG [81]
Dropout/Sampling based methods	MC-Dropout [27]	MC-Dropout [27], Bayesian SegNet [42]	Infer-perturbations [54], MC-Dropout [27]
Learning distribution/Auxiliary network	ConfidNet [15], Kendall et al. [43]	ConfidNet [15], Kendall et al. [43]	Hu et al. [38], SLURP [84], Mono-Uncertainty [64], Asai et al. [5], Kendall et al. [43], Nix et al. [59]
Deterministic uncertainty methods	SNGP [48], VIB [2], DUM [82], DUE [3], DDU [56], MIR [65], DUQ [77]	MIR [65]	DUE [3], MIR [65]

**Table 1.** Summary of the uncertainty estimation methods applied to the specific computer vision tasks.

Bayesian Neural Networks (BNNs) [10,37,81,20], which estimate the posterior distribution of the DNN weights to marginalize the likelihood distribution at inference time. These approaches achieve good performances on image classification, but they do not scale well to semantic segmentation. Deep Ensembles [45] achieve state-of-the-art performance on various tasks. Yet, this approach is computationally costly in both training and inference. Some techniques learn a confidence score as uncertainty [15], but struggle without sufficient negative samples to learn from. MC-Dropout [27] is a generic and easy to deploy approach, however its uncertainty is not always reliable [60] while requiring multiple forward passes. Deterministic Uncertainty Methods (DUMs) [48,82,2,3,56,65,77] are new strategies that allow to quantify epistemic uncertainty in the DNNs with a single forward pass. Yet, except for MIR [65], to the best of our knowledge none of these techniques work on semantic segmentation.

**Uncertainty for 1D/2D regression.** Regression in computer vision comprises monocular depth estimation [9,46], optical flow estimation [75,74], or pose estimation [12,71]. One solution for quantifying the uncertainty consists in formalizing the output of a DNN as a parametric distribution and training the DNN to estimate its parameters [59,43]. Multi-hypothesis DNNs [40] consider that the output is a Gaussian distribution and focus on optical flow. Some techniques estimate a confidence score for regression thanks to an auxiliary DNN [84,64]. Deep Ensembles [45] for regression, consider that each DNN outputs the parameters of a Gaussian distribution, to form a mixture of Gaussian distributions. Sampling-based methods [27,54] simply apply dropout or perturbations to some layers during test time to quantify the uncertainty. Yet, their computational cost remains important compared to a single forward pass in the network. Some DUMs [3,65] also work on regression tasks. DUE [3] is applied in a 1D regression task and MIR [65] in monocular depth estimation.

## 2.2 Prototype learning in DNNs

Prototype-based learning approaches have been introduced on traditional hand-crafted features [47], and have been recently applied to DNNs as well, for more robust predictions [79,83,29,13]. The center loss [79] can help DNNs to build more discriminative features by compacting intra-class features and dispersing the inter-class ones. Based on this principle, Convolutional Prototype Learning

(CPL) [83] with prototype loss also improves the intra-class compactness of the latent features. Chen et al. [13] try to bound the unknown classes by learning reciprocal points for better open set recognition. Similar to [78,69], MemAE [29] learns a memory slot of the prototypes to strengthen the reconstruction error of anomalies in the process of the reconstruction. These prototype-based methods are well suited for classification tasks but are rarely used in semantic segmentation and regression tasks.

### 3 Latent Discriminant deterministic Uncertainty (LDU)

#### 3.1 DUM preliminaries

DUMs arise as a promising line of research for estimating epistemic uncertainty in conventional DNNs in a computationally efficient manner and from a single forward pass. DUM approaches generally focus on learning useful and informative hidden representations of a model [48,82,2,3,56,65] by considering that the distribution of the hidden representation should be representative for the input distribution. Most of the conventional models suffer from the *feature collapse* problem [77] when OOD samples are mapped to similar feature representations as in-distribution ones, thus hindering OOD detection from these representations. DUMs address this issue through various regularization strategies for constraining the hidden representations to mimic distances from the input space. In practice this amounts to striking a balance between *sensitivity* (when the input changes, the feature representation should also change) and *smoothness* (a small change in the input cannot generate major shifts in the feature representation) of the model. To this end, most methods enforce constraints over the Lipschitz constant of the DNN [53,77,48].

Formally, we define  $f_\omega(\cdot)$  a DNN with trainable parameters  $\omega$ , and an input sample  $\mathbf{x}$  from a set of images  $\mathcal{X}$ . Our DNN  $f_\omega$  is composed of two main blocks: a feature extractor  $h_\omega$  and a head  $g_\omega$ , such that  $f_\omega(\mathbf{x}) = (g_\omega \circ h_\omega)(\mathbf{x})$ .  $h_\omega(\mathbf{x})$  computes a latent representation from  $\mathbf{x}$ , while  $g_\omega$  is the final layer, that takes  $h_\omega(\mathbf{x})$  as input, and outputs the logits of  $\mathbf{x}$ . The bi-Lipschitz condition implies that for any pair of inputs  $\mathbf{x}_1$  and  $\mathbf{x}_2$  from  $\mathcal{X}$ :

$$L_1 \|\mathbf{x}_1 - \mathbf{x}_2\| \leq \|h_\omega(\mathbf{x}_1) - h_\omega(\mathbf{x}_2)\| \leq L_2 \|\mathbf{x}_1 - \mathbf{x}_2\| \quad (1)$$

where  $L_1$  and  $L_2$  are positive and bounded Lipschitz constants  $0 < L_1 < 1 < L_2$ . The upper Lipschitz bound enforces the smoothness and is an important condition for the robustness of a DNN by preventing over-sensitivity to perturbations in the input space of  $\mathbf{x}$ , i.e., the pixel space. The lower Lipschitz bound deals with the sensitivity and strives to preserve distances in the latent space as mappings of distances from the input space, i.e., preventing representations from being too smooth, thus avoiding feature collapse. Liu et al. [48] argue that for residual DNNs [33], we can ensure  $f_\omega$  to be bi-Lipschitz by forcing its residuals to be Lipschitz and choosing sub-unitary Lipschitz constants.

There are different approaches for imposing the bi-Lipschitz constraint over a DNN, out of which we describe the most commonly used ones in recent

works [4,30,55,7]. Wasserstein GAN [4] enforces the Lipschitz constraint by clipping the weights. However, this turns out to be prone to either vanishing or exploding gradients if the clipping threshold is not carefully tuned [30]. An alternative solution from GAN optimization is gradient penalty [30] which is practically an additional loss term that regularizes the  $L_2$  norm of the Jacobian of weight matrices of the DNN. However this can also lead to high instabilities [48,56] and slower training [56]. Spectral Normalization [55,7] brings better stability and training speed, however, on the downside, it supports only a fixed pre-defined size for the input, in the same manner as fully connected layers. For computer vision tasks, such as semantic segmentation which is typically performed on high resolution images, constraining the input size is a strong limitation. Moreover, Postels et al. [65] argue that in addition to the architectural constraints, these strategies for avoiding feature collapse risk overfitting epistemic uncertainty to the task of OOD detection. This motivates us to seek a new DUM strategy that does not need the network to comply with the Lipschitz constraint. The recent MIR approach [65] advances an alternative regularization strategy that adds a decoder branch to the network, thus forcing the intermediate activations to better cover and represent the input space. However in the case of high resolution images, reconstruction can be a challenging task and the networks can over-focus on potentially useless and uninformative details at the cost of loss of global information. We detail our strategy below.

### 3.2 Discriminant Latent space

An informative latent representation should project similar data samples close and dissimilar ones far away. Yet, it has long been known that in high-dimensional spaces the Euclidean distance and other related p-norms are a very poor indicator of sample similarity as most samples are nearly equally far/close to each other [1,6]. At the same time, the samples of interest are often not uniformly distributed, and may be projected by means of a learned transform on a lower-dimensional manifold, namely the latent representation space.

Instead of focusing on preserving the potentially uninformative distance in the input space, we can rather attempt to better deal with distances in the lower-dimensional latent space. To this end, we propose to use a distinction maximization (DM) layer [49] that has been recently considered as a replacement for the last layer to produce better uncertainty estimates, in particular for OOD detection [61,49]. In a DM layer, the units of the classification layer are seen as representative class prototypes and the classification prediction is computed by analyzing the localization of the input sample w.r.t. all class prototypes as indicated by the negative Euclidean distance. Note that a similar idea has been considered in the few-shot learning literature, where DM layers are known as cosine classifiers [28,67,73]. In contrast to all these approaches that use DM as a last layer for classification predictions, we employ it as hidden layer over latent representations. More specifically, we insert DM in the pre-logit layer. We argue that this allows us to better guide learning and preserve the discriminative properties of the latent representations compared to placing DM as last layers

where the weights are more specialized for classification decision than for feature representation. We can easily integrate this layer in the architecture without impacting the training pipeline.

Formally, we denote  $\mathbf{z} \in \mathbb{R}^n$  the latent representation of dimension  $n$  of  $\mathbf{x}$ , i.e.,  $\mathbf{z}=h_{\omega}(\mathbf{x})$ , that is given as input to the DM layer. Given a set  $\mathbf{p}_{\omega}=\{\mathbf{p}_i\}_{i=1}^m$ , of  $m$  vectors ( $\mathbf{p}_i \in \mathbb{R}^n$ ) that are trainable, we define the DM layer as follows:

$$\text{DM}_p(\mathbf{z}) = [-\|\mathbf{z} - \mathbf{p}_1\|, \dots, -\|\mathbf{z} - \mathbf{p}_m\|]^{\top} \quad (2)$$

The  $L_2$  distance considered in the DM layer is not bounded, thus when DM is used as intermediate layer, relying on the  $L_2$  distance could cause instability during training. In our proposed approach, we use instead the cosine similarity,  $S_c(\cdot, \cdot)$ . Our DM layer reads now:

$$\text{DM}_p(\mathbf{z}) = [S_c(\mathbf{z}, \mathbf{p}_1), \dots, S_c(\mathbf{z}, \mathbf{p}_m)]^{\top} \quad (3)$$

The vectors  $\mathbf{p}_i$  can be seen as a set of prototypes in the latent space that can help in better placing an input sample in the learned representation space using these prototypes as references. This is in contrast to prior works with DM being considered as last layer, where the prototypes represent canonical representations for samples belonging to a class [73,49]. Since hidden layers are used here, we can afford to consider an arbitrary number of prototypes that can define richer latent mapping through a finer coverage of the representation space. DM layers learn the set of weights  $\{\mathbf{p}_i\}_{i=1}^m$  such that the cosine similarity (evaluated between  $\mathbf{z}$  and the prototypes) is optimal for a given task.

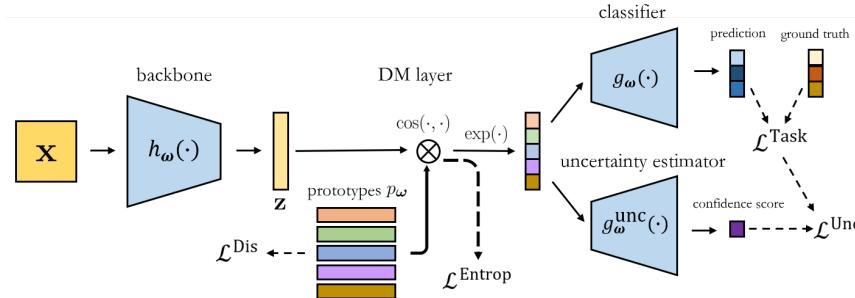
We apply the distinction maximization on this hidden representation, and subsequently use the exponential function as activation function. We consider the exponential function as it can sharpen similarity values and thus facilitates the alignment of the data embedding to the corresponding prototypes in the latent space. Finally, we apply a last fully connected layer for classification on this embedding. Our DNN (see Figure 1) can be written as:

$$f_{\omega}(\mathbf{x}) = [g_{\omega} \circ (\exp(-\text{DM}_p(h_{\omega})))](\mathbf{x}) \quad (4)$$

We can see from Eq. (4) that the vector weights  $\mathbf{p}_i$  are optimized jointly with the other DNN parameters. We argue that  $\mathbf{p}_i$  can work as indicators for analyzing and emphasizing patterns in the latent representation prior to making a classification prediction in the final layers.

### 3.3 LDU optimization

Given a DNN  $f_{\omega}$  we usually optimize its parameters to minimize a loss  $\mathcal{L}^{\text{Task}}$ . This can lead to prototypes specialized for solving that task that do not encapsulate uncertainty relevant properties. Hence we propose to enforce the prototypes to be linked to uncertainty first by avoiding the collapse of all prototypes to a single prototype. Second, we constrain the latent representation  $\text{DM}_p(h_{\omega})$  of the DNN to not rely only on a single prototype. Finally, we optimize an MLP  $g_{\omega}^{\text{unc}}$



**Fig. 1.** Overview of LDU: the DNN learns a discriminative latent space thanks to learnable prototypes  $\mathbf{p}_\omega$ . The DNN backbone computes a feature vector  $\mathbf{z}$  for an input  $\mathbf{x}$  and then the DM layer matches it with the prototypes. The computed similarities reflecting the position of  $\mathbf{z}$  in the learned feature space, are subsequently processed by the classification layer and the uncertainty estimation layer. The dashed arrows point to the loss functions that need to be optimized for training LDU.

on the top of the latent representation  $\text{DM}_p(h_\omega)$  such that the output of this MLP provides more meaningful information for uncertainty estimation.

First, we add a loss to force the prototypes to be dissimilar:

$$\mathcal{L}^{\text{Dis}} = - \sum_{i < j} \|\mathbf{p}_i - \mathbf{p}_j\|.$$

Then, we also add one loss to constrain the latent representation to stay close to different prototypes. We achieve this with an entropy-like loss:

$$\mathcal{L}^{\text{Entrop}} = \sum_{i=1}^n \sigma(\text{DM}_p(h_\omega))_i \cdot \log(\sigma(\text{DM}_p(h_\omega))_i),$$

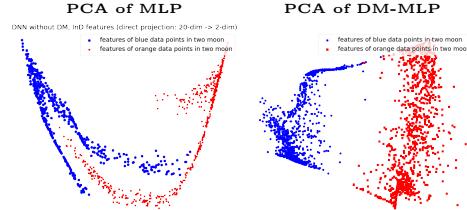
where  $\sigma$  is the softmax layer, and the subscript index  $i$  corresponds to the  $i$ -th coefficient of a tensor. Different from per-class prototypes [79,83,13], we obtain more discriminative features by increasing the distance between prototypes and enlarging the dispersion of features corresponding to different prototypes.

We propose to train  $g_\omega^{\text{unc}}$  to predict the error of the DNN, which helps us relate the prototypes to the uncertainty. Formally, given an input data  $\mathbf{x}$ , its groundtruth  $y$  ( $y$  can be a scalar or a vector if we deal with regression) and, its loss  $\mathcal{L}^{\text{Task}}(g_\omega(\mathbf{x}), y)$ , we train  $g_\omega^{\text{unc}}$  by minimizing:

$$\mathcal{L}^{\text{Unc}} = \text{BCE}([g_\omega^{\text{unc}} \circ (\exp(-\text{DM}_p(h_\omega)))](\mathbf{x}), \mathcal{L}^{\text{Task}}(g_\omega(\mathbf{x}), y)),$$

after normalizing  $\mathcal{L}^{\text{Task}}(g_\omega(\mathbf{x}), y)$  over the mini-batch such that its maximum value is equal to one and its minimum is equal to zero. BCE stands for the binary cross entropy, which was empirically validated to perform better than common alternatives such as the mean square error and the absolute error.

All these losses combined allow us to have a DNN which can predict uncertainty, avoid feature collapse and have the potential to improve the accuracy of



**Fig. 2.** PCA 2D projection on the left of a standard MLP and on the right of a DM-MLP trained on the two moons dataset. Blue and red points indicate the features of data points of the two classes respectively. As we can see, the representations on the MLP are overlapping between the two classes, leading to a network that will be prone to feature collapse, unlike the DM-MLP.

the prediction. To summarize, the following loss function  $\mathcal{L}^{\text{total}}$  will be optimized to train a DNN containing a DM layer:

$$\mathcal{L}^{\text{total}} = \mathcal{L}^{\text{Task}} + \lambda(\mathcal{L}^{\text{Entrop}} + \mathcal{L}^{\text{Dis}} + \mathcal{L}^{\text{Unc}}) \quad (5)$$

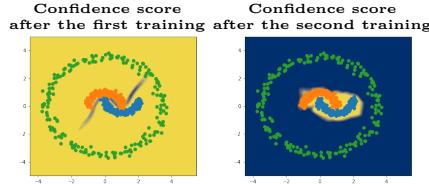
where  $\lambda$  is a hyper-parameter for the auxiliary losses.

### 3.4 Addressing feature collapse

In order to illustrate the feature collapse problem, we consider a toy example on the two moon dataset. We train on it two MLPs with two hidden layers, each containing 17 neurons. One of the MLP additionally integrates our proposed DM layer and is denoted as DM-MLP, while the standard architecture is called MLP. The two networks reach the same classification performance, about 99% of accuracy. We perform PCA on the pre-logit latent space of both networks after training and visualize PCA projections in Figure 2. We can observe the feature collapse as the MLP assigns strongly correlated feature representation to both classes which can lead to unreliable uncertainty prediction. However, our DM layer allows a better disentangling of the latent space. Note that, as the networks have the same performance, it is impossible to detect the feature collapse based on the test accuracy alone.

We note that our LDU layer is a Lipschitz function, hence:  $\|\exp(-\text{DM}_p(z_1)) - \exp(-\text{DM}_p(z_2))\| \leq k\|z_1 - z_2\|$  with  $k \in \mathbb{R}^+$ . However,  $h_{\omega}$  is not necessarily a Lipschitz function, and we cannot thus guarantee that its features do not entangle ID and OOD data. Yet, using a distance function in the DNN [53,50] can allow it to learn to separate the two data distributions better as illustrated in Figure 2.

Most DUM methods aim for bi-Lipschitz DNNs with small Lipschitz constants. Yet, this is sub-optimal according to the concentration theory. Indeed, let  $\mathbf{X}$  be a set of random vectors of size  $d$  i.i.d. from a normal distribution  $\mathcal{N}(0, \sigma^2 I_d)$ .  $I_d$  is the identity matrix of size  $d$ . Let  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  be a Lipschitz



**Fig. 3.** Illustration of confidence score results on the two moons dataset after the first training (on original data) on the left and with second training (on synthesized outliers) on the right. Orange and blue data points are sampled from two classes in two moons, and the green points are OOD data points. Yellow area indicates high confidence, blue area indicates uncertainty. The left image shows that the uncertain area is between the two classes leading to a confidence score related to aleatoric uncertainty. In the right one, the uncertain area is around the dataset leading to a confidence score related to epistemic uncertainty.

function with Lipschitz constant  $K$ . The concentration theory ([11], p. 125) stipulates that :  $\mathcal{P}(|f(\mathbf{X}) - \mathbb{E}(f(\mathbf{X}))| > t) \leq 2\exp(-\frac{t^2}{2K^2\sigma^2})$  for all  $t > 0$ . This means that the smaller  $K$  is, the more the concentration of the data around their mean increases, leading to increased feature collapse. Hence, it is desirable to have a Lipschitz function that will bring similar data close, but it is at the same time essential to put dissimilar data apart.

### 3.5 LDU and Epistemic/Aleatoric Uncertainty

We are interested in capturing two types of uncertainty with our DNN: aleatoric and epistemic uncertainty [17,43]. Aleatoric uncertainty is related to the inherent noise and ambiguity in data or in annotations and is often called irreducible uncertainty as it does not fade away with more data in the training set. Epistemic uncertainty is related to the model, the learning process and the amount of training data. Since it can be reduced with more training data, it is also called reducible uncertainty. Disentangling these two sources of uncertainty is generally non-trivial [56] and ensemble methods are usually superior [23,52].

**Optional training with synthesized outliers.** Due to limited training data and to the penalty enforced by  $\mathcal{L}^{\text{Task}}$  being too small, the loss term  $\mathcal{L}^{\text{Unc}}$  may potentially force the DNN in some circumstances to overfit the aleatoric uncertainty. Although we did not encounter this behavior on the computer vision tasks given the dataset size, it might occur on more specific data, and among other potential solutions, we propose one relying on synthesized outliers that we illustrate on the two moons dataset as follows. More specifically, we propose to add noise to the data similarly to [51,19], and to introduce an optional step for training  $g_{\omega}^{\text{unc}}$  on these new samples. We consider a two-stage training scheme. In the first stage we train over data without noise and in the second we optimize only the parameters of  $g_{\omega}^{\text{unc}}$  over the synthesized outliers. Note that this optional stage would require for vision tasks an adequate OOD synthesizer [8,19] which

is beyond the scope of this paper, and that we applied it on the toy dataset. In Figure 3 we assess the uncertainty estimation performance of this model on the two moons dataset. We can see that the confidence score relates to the aleatoric uncertainty after the first training stage. After the second one, it is linked to the epistemic uncertainty of the model.

Distinguishing between the two sources of uncertainty is essential for various practical applications, such as active learning, monitoring, OOD detection. In the following, we propose two strategies for computing each type of uncertainty.

**Aleatoric uncertainty.** For estimating aleatoric uncertainty in classification, maximum class probability (MCP) [36] is a common strategy. The intuition is that a lower MCP can mean a higher entropy, i.e., a potential confusion on the classifier regarding the most likely class of the image. We use this criterion for the aleatoric uncertainty for classification and for semantic segmentation, while for the regression task we use  $g_\omega^{\text{unc}}$  as confidence score.

**Epistemic uncertainty.** To estimate epistemic uncertainty, we analyzed the latent representations of the DM layers followed by an exponential activation and found that the maximum value can model well uncertainty. The position of the feature w.r.t. the learned prototypes carries information about the proximity of the current sample with the in-distribution features. Yet, we propose to use the output of  $g_\omega^{\text{unc}}$  as confidence score since we train this criterion for this purpose.

## 4 Experiments

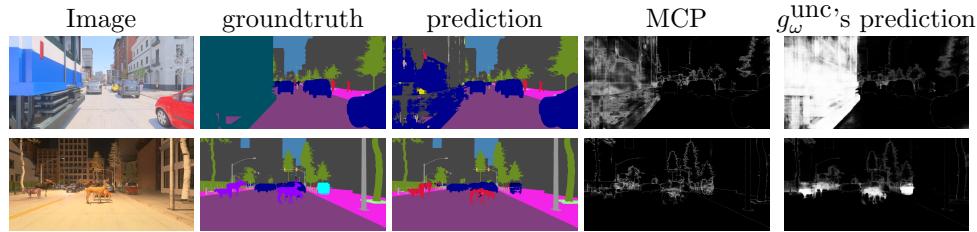
One major interest of our technique is that it may be seamlessly applied to any computer vision task, be it classification or regression. Thus, we propose to evaluate the quality of uncertainty quantification of different techniques on three major tasks, namely image classification, semantic segmentation and monocular depth estimation. For all the three tasks, we compare our technique against MC Dropout [27] and Deep Ensembles [45]. For image classification, we also compare our technique to relevant DUM techniques for image classification, namely DDU [56], DUQ [77], DUE [3], MIR [65] and SNGP [48].

We evaluate the predictive performance in terms of accuracy for image classification, mIoU [22] for semantic segmentation, and the metrics first introduced in [21] and used in many subsequent works for monocular depth estimation. For image classification and semantic segmentation, we also evaluate the quality of the confidence scores provided by the DNNs via the following metrics: Expected Calibration Error (ECE) [31], AUROC [36,34] and AUPR [36,34]. Note that ECE we use is the confidence ECE defined in [31]. We use the Area Under the Sparsification Error: AUSE-RMSE and AUSE-Absrel similarly to [64,32,84] to better evaluate the uncertainty quantification on monocular depth estimation.

We run all methods ourselves in similar settings using publicly available codes and hyper-parameters for related methods. In the following tables, Top-2 results are highlighted in color.

Method	CIFAR-10				
	Acc ↑	AUC ↑	AUPR ↑	ECE ↓	
Baseline (MCP) [36]	88.02	0.8032	0.8713	0.5126	
MCP lipz. [7]	88.50	0.8403	0.9058	<b>0.3820</b>	
Deep Ensembles [45]	<b>89.96</b>	0.8513	0.9087	0.4249	
SNGP [48]	88.45	0.8447	0.9139	0.4254	
DUQ [77]	89.9	0.8446	0.9144	0.5695	
DUE [3]	87.54	0.8434	0.9082	0.4313	
DDU [56]	87.87	0.8199	0.8754	<b>0.3820</b>	
MIR [65]	87.95	0.7574	0.8556	0.4004	
LDU #p = 128	87.95	<b>0.8721</b>	<b>0.9147</b>	0.4933	
LDU #p = 64	88.06	0.8625	0.9070	0.5010	
LDU #p = 32	87.83	0.8129	0.8900	0.5264	
LDU #p = 16	88.33	0.8479	0.9094	0.4975	

**Table 2.** Comparative results for **image classification tasks**. We evaluate on CIFAR-10 for the tasks: in-domain classification, out-of-distribution detection with SVHN. Results are averaged over three seeds.



**Fig. 4.** Illustration of the different confidence scores on one image of MUAD. Note that the class **train**, **bicycle**, **Stand food** and the **animals** are OOD.

#### 4.1 Classification experiments

To evaluate uncertainty quantification for image classification, we adopt a standard approach based on training on CIFAR-10 [44], and using SVHN [58] as OOD data [66,25,48]. We use ResNet18 [33] as architecture for all methods. Note that for all DNNs, even for DE, we average results over three random seeds for statistical relevance. We follow the corresponding protocol for all DUM techniques (except LDU). For Deep Ensembles, MCP, and LDU, we use the same protocol. Please refer to the appendix for implementation details of LDU. The performances of the different algorithms are shown in Table 2. We can see that LDU has state-of-the-art performances on CIFAR-10. We note that LDU’s OOD detection performance improves with the number of prototypes. This can be linked with the fact that the more prototypes we have, the better we can model complex distributions. The ablation studies on sensitivity of the choice of  $\lambda$  and the impact of different losses are provided in the appendix.

#### 4.2 Semantic segmentation experiments

Our semantic segmentation study consists of three experiments. The first one is on a new synthetic dataset: MUAD [26]. It comprises a training set and a test set without OOD classes and adverse weather conditions. We denote this set **normal**

Evaluation data	normal set		OOD set		low adv. set		high adv. set	
	mIoU ↑ ECE ↓	mIoU ↑ ECE ↓	mIoU ↑ AUROC ↑ AUPR ↑ FPR ↓	mIoU ↑ ECE ↓ AUROC ↑ AUPR ↑ FPR ↓	mIoU ↑ ECE ↓ AUROC ↑ AUPR ↑ FPR ↓	mIoU ↑ ECE ↓ AUROC ↑ AUPR ↑ FPR ↓	mIoU ↑ ECE ↓ AUROC ↑ AUPR ↑ FPR ↓	mIoU ↑ ECE ↓ AUROC ↑ AUPR ↑ FPR ↓
Baseline (MCP) [36]	68.90%	0.0138	57.32%	0.0607	0.8624	0.2604	0.3943	31.84%
Baseline (MCP) lipz. [7]	53.96%	0.01398	45.97%	0.0601	0.8419	0.2035	0.3940	16.79%
MIR [65]	53.96%	0.01398	45.97%	0.0601	0.6223	0.1469	0.8406	16.79%
MC-Dropout [27]	65.33%	0.0173	55.62%	0.0645	0.8439	0.2225	0.4575	33.38%
Deep Ensembles [45]	<b>69.80%</b>	<b>0.01296</b>	<b>58.29%</b>	<b>0.0588</b>	0.871	0.2802	0.3760	34.91%
LDU (ours)	69.32%	0.01356	<b>58.29%</b>	0.0594	<b>0.8816</b>	<b>0.4418</b>	<b>0.3548</b>	<b>36.12%</b>

**Table 3.** Comparative results for semantic segmentation on MUAD.

Evaluation data	Cityscapes		Cityscapes-C lvl 1		Cityscapes-C lvl 2		Cityscapes-C lvl 3		Cityscapes-C lvl 4		Cityscapes-C lvl 5	
	mIoU ↑ ECE ↓	mIoU ↑	mIoU ↑ ECE ↓	mIoU ↑	ECE ↓	mIoU ↑	ECE ↓	mIoU ↑ ECE ↓	mIoU ↑	ECE ↓	mIoU ↑	ECE ↓
Baseline (MCP) [36]	76.84%	0.1180	51.59%	0.1793	41.45%	0.2291	35.67%	0.2136	30.12%	0.1970	24.84%	0.2131
Baseline (MCP) lipz. [7]	58.38%	0.1037	44.70%	<b>0.1211</b>	38.04%	0.1475	32.70%	0.1802	25.35%	0.2047	18.36%	0.2948
MC-Dropout [27]	71.88%	0.1157	53.61%	0.1501	42.02%	0.2531	35.91%	0.1718	29.52%	0.1947	25.61%	0.2184
Deep Ensembles [45]	<b>77.23%</b>	0.1139	<b>54.98%</b>	0.1422	<b>44.63%</b>	0.1902	<b>38.00%</b>	0.1851	32.14%	0.1602	<b>28.74%</b>	0.1729
LDU (ours)	76.62%	<b>0.0893</b>	52.00%	0.1371	43.02%	<b>0.1314</b>	37.17%	<b>0.1702</b>	32.27%	0.1314	27.30%	<b>0.1712</b>

**Table 4.** Comparative results for semantic segmentation on Cityscapes and Cityscapes-C.

OOD technique	mIoU ↑ AUC ↑ AUPR ↑ FPR-95%-TPR ↓
Baseline (MCP) [36]	52.8 86.0 5.4 27.7
MC-Dropout [27]	49.5 85.2 5.0 29.3
Deep Ensembles [45]	<b>57.6</b> 87.0 <b>6.0</b> <b>25.0</b>
TRADI [25]	52.1 86.1 5.6 26.9
ConfidNET [15]	52.8 85.4 5.1 29.1
LDU (ours)	55.1 <b>87.1</b> 5.8 26.2

**Table 5.** Comparative results obtained on the OOD detection task on BDD Anomaly [34] with PSPNet (ResNet50).

**set.** MUAD contains three more test sets that we denote **OOD set**, **low adv. set** and **high adv. set** which contain respectively images with OOD pixels but without adverse weather conditions, images with OOD pixels and weak adverse weather conditions, and for the last set, images with OOD pixels and strong adverse weather conditions. The second experiment evaluates the segmentation precision and uncertainty quality on the Cityscapes [16] and the Cityscapes-C [41,70,24] datasets to assess performance under distribution shift. Finally we analyze OOD detection performance on BDD Anomaly dataset [34] whose test set contains objects unseen during training. We detail the experimental protocol of all datasets in the appendix.

We train a DeepLabV3+ [14] network with ResNet50 encoder [33] on MUAD. Table 3 lists the results from different uncertainty techniques. For this task, we found that enforcing the Lipschitz constraint (see Baseline (MCP) lipz.) has a significant impact. Figure 4 shows a qualitative example of typical uncertainty maps computed on MUAD images.

Similarly to [24,41] we assess predictive uncertainty and robustness under distribution shift using Cityscapes-C, a corrupted version of Cityscapes images with perturbations of varying intensity. We generate Cityscapes-C ourselves from the original Cityscapes images using the code of Hendrycks et al. [35] to apply the different corruptions on the images. Following [35], we apply the following perturbations: Gaussian noise, shot noise, impulse noise, defocus blur, frosted, glass blur, motion blur, zoom blur, snow, frost, fog, brightness, contrast, elastic,

Method	Depth performance						Uncertainty performance					
	d1↑	d2↑	d3↑	Abs Rel↓	Sq Rel↓	RMSE↓	RMSE log↓	log10↓	AUSE	RMSE↓	AUSE	Absrel↓
Baseline	0.955	<b>0.993</b>	0.998	<b>0.060</b>	0.249	2.798	0.096	0.027	-	-	-	-
Deep Ensembles [45]	<b>0.956</b>	<b>0.993</b>	<b>0.999</b>	<b>0.060</b>	<b>0.236</b>	<b>2.700</b>	<b>0.094</b>	<b>0.026</b>	<b>0.08</b>	<b>0.21</b>	-	-
MC-Dropout [27]	0.945	0.992	0.998	0.072	0.287	2.902	0.107	0.031	0.46	0.50	-	-
Single-PU [43]	0.949	0.991	0.998	0.064	0.263	2.796	0.101	0.029	<b>0.08</b>	<b>0.21</b>	-	-
Infer-noise [54]	0.955	<b>0.993</b>	0.998	<b>0.060</b>	0.249	2.798	0.096	0.027	0.33	0.48	-	-
LDU #p = 5, $\lambda = 1.0$	0.954	<b>0.993</b>	0.998	0.063	0.253	2.768	0.098	0.027	<b>0.08</b>	<b>0.21</b>	-	-
LDU #p = 15, $\lambda = 0.1$	0.954	<b>0.993</b>	0.998	0.062	0.249	2.769	0.098	0.027	0.10	0.28	-	-
LDU #p = 30, $\lambda = 0.1$	0.955	0.992	0.998	0.061	0.248	2.757	0.097	0.027	0.09	0.26	-	-

**Table 6.** Comparative results for **monocular depth estimation on KITTI eigen-split validation set**.

pixelate, JPEG. Each perturbation is scaled with five levels of strength. We train a DeepLabV3+ [14] with ResNet50 encoder [33] on Cityscapes. Results in Table 4 show that LDU is closely trailing in accuracy (mIoU score) the much more costly Deep Ensembles [45], while making better calibrated predictions (ECE score).

In order to assess the epistemic uncertainty quantification on real data we used PSPNet [85] with ResNet50 backbone using the experimental protocol in [34]. BDD Anomaly is a subset of BDD dataset, composed of 6688 street scenes for the training set and 361 for the testing set. The training set contains 17 classes, and the test set is composed of the 17 training classes and 2 OOD classes. Results in Table 5 show again that the performances of LDU are close to the ones of Deep Ensembles.

### 4.3 Monocular depth experiments

We set up our experiments on KITTI dataset [76] with Eigen split training and validation set [21] to evaluate and compare the predicted depth accuracy and uncertainty quality. We train BTS [46] with DenseNet161 [39], and we use the default training setting of BTS (number of epochs, weight decay, batch size) to train DNNs for all uncertainty estimation techniques applied on this backbone.

By default, the BTS baseline does not output uncertainty. Similarly to [43,40], we can consider that a DNN may be constructed to find and output the parameters of a parametric distribution (e.g., the mean and variance for a Gaussian distribution). Such networks can be optimized by maximizing their log-likelihood. We denote the result as single predictive uncertainty (Single-PU). We also train a Deep Ensembles [45] by ensembling 3 DNNs, as well as a MC-Dropout [27] with eight forward passes. Without the extra DNNs or training procedures, we also applied Infer-noise [54], which injects Gaussian noise layers to the trained BTS baseline model and propagate eight times to predict the uncertainty.

We have also implemented LDU with the BTS model, but we note however that, in the monocular depth estimation setting and in agreement with previous works [18], the definition of OOD is fundamentally different with respect to the tasks introduced in the prior experiments. Thus, our objective is to investigate whether LDU is robust, can improve the prediction accuracy and still perform well for aleatoric uncertainty estimation. Table 6 lists the depth and uncertainty estimation results on KITTI dataset. Using different settings of #p and  $\lambda$ , the

Method	Semantic segmentation			Monocular depth		
	Runtime (ms)	Training time (ms)	#param.	Runtime (ms)	Training time (ms)	#param.
Baseline	14	166.4	39.76	45	92.8	47.00
Deep Ensembles [45]	56	499.2	119.28	133	287.8	141.03
MC-Dropout [27]	199	166.4	39.76	370	92.3	47.00
Single-PU [43]	-	-	-	45	95.6	47.01
LDU (ours)	14	177.8	39.76	45	104.0	47.00

**Table 7. Comparative results for training (forward+backward) and inference wall-clock timings and number of parameters for evaluated methods.** Timings are computed per image and averaged over 100 images.

proposed LDU is virtually aligned with the current state-of-the-art, while being significantly lighter computationally (see also Table. 7). More ablation results on the influence of  $\#p$  and  $\lambda$  can be found in the supplementary materials.

## 5 Discussions and Conclusions

**Discussions.** In Table 7 we compare the computational cost of LDU and related methods. For each approach we measure the training (forward+backward) and inference time per image on a NVIDIA RTX 3090Ti and report the corresponding number of parameters. We report training and inference wall-clock timings averaged over 100 training and validation. We use the same backbones as mentioned in Section 4.2 and Section 4.3 for semantic segmentation and monocular depth estimation respectively. We note that the runtime of LDU is almost the same as that of the baseline model (standard single forward model). This underpins the efficiency of our approach during inference, a particularly important requirement for practical applications.

**Conclusions.** In this work, we propose a simple way to modify a DNN to better estimate its predictive uncertainty. These minimal changes consist in optimizing a set of latent prototypes to learn to quantify the uncertainty by analyzing the position of an input sample in this prototype space. We perform extensive experiments and show that LDU can outperform state-of-the-art DUMs in most tasks and reach results comparable to Deep Ensembles with a significant advantage in terms of computational efficiency and memory requirements.

Along with the current state of the art methods, a limitation of our proposed LDU is that despite the empirical improvements in uncertainty quantification, it does not provide theoretical guarantees on the correctness of the predicted uncertainty. Our perspectives concern further exploration and improvements of the regularization strategies introduced in LDU on the latent feature representation that would allow us to bound the model error while still preserving its main task high performance.

## Acknowledgement

This work was performed using HPC resources from GENCI-IDRIS (Grant 2020-AD011011970) and (Grant 2021-AD011011970R1) and Saclay-IA computing platform.

## References

1. Aggarwal, C.C., Hinneburg, A., Keim, D.A.: On the surprising behavior of distance metrics in high dimensional space. In: ICDT (2001) [5](#)
2. Alemi, A.A., Fischer, I., Dillon, J.V.: Uncertainty in the variational information bottleneck. In: UAI (2018) [3](#), [4](#)
3. 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) [2](#), [3](#), [4](#), [10](#), [11](#)
4. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein generative adversarial networks. In: ICLR (2017) [5](#)
5. Asai, A., Ikami, D., Aizawa, K.: Multi-task learning based on separable formulation of depth estimation and its uncertainty. In: CVPR Workshops (2019) [3](#)
6. Assent, I.: Clustering high dimensional data. KDD (2012) [5](#)
7. Behrmann, J., Grathwohl, W., Chen, R.T., Duvenaud, D., Jacobsen, J.H.: Invertible residual networks. In: ICML (2019) [5](#), [11](#), [12](#)
8. Besnier, V., Bursuc, A., Picard, D., Briot, A.: Triggering failures: Out-of-distribution detection by learning from local adversarial attacks in semantic segmentation. In: ICCV (2021) [9](#)
9. Bhat, S.F., Alhashim, I., Wonka, P.: Adabins: Depth estimation using adaptive bins. In: CVPR (2021) [3](#)
10. Blundell, C., Cornebise, J., Kavukcuoglu, K., Wierstra, D.: Weight uncertainty in neural network. In: ICML (2015) [1](#), [3](#)
11. Boucheron, S., Lugosi, G., Massart, P.: Concentration inequalities: A nonasymptotic theory of independence. Oxford university press (2013) [9](#)
12. Cao, Z., Hidalgo Martinez, G., Simon, T., Wei, S., Sheikh, Y.A.: Openpose: Realtime multi-person 2d pose estimation using part affinity fields. TPAMI (2019) [3](#)
13. Chen, G., Qiao, L., Shi, Y., Peng, P., Li, J., Huang, T., Pu, S., Tian, Y.: Learning open set network with discriminative reciprocal points. In: ECCV (2020) [3](#), [4](#), [7](#)
14. Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H.: Encoder-decoder with atrous separable convolution for semantic image segmentation. In: ECCV (2018) [12](#), [13](#)
15. Corbière, C., Thome, N., Bar-Hen, A., Cord, M., Pérez, P.: Addressing failure prediction by learning model confidence. In: NeurIPS (2019) [2](#), [3](#), [12](#)
16. 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: CVPR (2016) [12](#)
17. Der Kiureghian, A., Ditlevsen, O.: Aleatory or epistemic? does it matter? Structural safety (2009) [9](#)
18. Dijk, T.v., Croon, G.d.: How do neural networks see depth in single images? In: ICCV (2019) [13](#)
19. Du, X., Wang, Z., Cai, M., Li, Y.: Vos: Learning what you don't know by virtual outlier synthesis. In: ICLR (2022) [9](#)
20. 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: ICML (2020) [3](#)
21. Eigen, D., Puhrsch, C., Fergus, R.: Depth map prediction from a single image using a multi-scale deep network. In: NeurIPS (2014) [10](#), [13](#)

22. Everingham, M., Eslami, S.A., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes challenge: A retrospective. IJCV (2015) **10**
23. Fort, S., Hu, H., Lakshminarayanan, B.: Deep ensembles: A loss landscape perspective. arXiv preprint arXiv:1912.02757 (2019) **1, 9**
24. Franchi, G., Belkhir, N., Ha, M.L., Hu, Y., Bursuc, A., Blanz, V., Yao, A.: Robust semantic segmentation with superpixel-mix. In: BMVC (2021) **12**
25. Franchi, G., Bursuc, A., Aldea, E., Dubuisson, S., Bloch, I.: TRADI: Tracking deep neural network weight distributions. In: ECCV (2020) **11, 12**
26. Franchi, G., Yu, X., Bursuc, A., Kazmierczak, R., Dubuisson, S., Aldea, E., Filliat, D.: MUAD: Multiple uncertainties for autonomous driving benchmark for multiple uncertainty types and tasks. arXiv preprint arXiv:2203.01437 (2022) **11**
27. Gal, Y., Ghahramani, Z.: Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In: ICML (2016) **1, 3, 10, 12, 13, 14**
28. Gidaris, S., Komodakis, N.: Dynamic few-shot visual learning without forgetting. In: CVPR (2018) **5**
29. Gong, D., Liu, L., Le, V., Saha, B., Mansour, M.R., Venkatesh, S., Hengel, A.v.d.: Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection. In: ICCV (2019) **2, 3, 4**
30. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.: Improved training of wasserstein gans. In: NeurIPS (2017) **5**
31. Guo, C., Pleiss, G., Sun, Y., Weinberger, K.Q.: On calibration of modern neural networks. In: ICML (2017) **10**
32. Gustafsson, F.K., Danelljan, M., Schon, T.B.: Evaluating scalable bayesian deep learning methods for robust computer vision. In: CVPR Workshops (2020) **10**
33. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016) **4, 11, 12, 13**
34. Hendrycks, D., Basart, S., Mazeika, M., Mostajabi, M., Steinhardt, J., Song, D.: A benchmark for anomaly segmentation. arXiv preprint arXiv:1911.11132 (2019) **10, 12, 13**
35. Hendrycks, D., Dietterich, T.: Benchmarking neural network robustness to common corruptions and perturbations. In: ICLR (2019) **12**
36. Hendrycks, D., Gimpel, K.: A baseline for detecting misclassified and out-of-distribution examples in neural networks. In: ICLR (2017) **10, 11, 12**
37. Hernández-Lobato, J.M., Adams, R.: Probabilistic backpropagation for scalable learning of bayesian neural networks. In: ICML (2015) **3**
38. Hu, S., Pezzotti, N., Welling, M.: Learning to predict error for mri reconstruction. In: MICCAI (2021) **3**
39. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: CVPR (2017) **13, 21**
40. Ilg, E., Cicek, O., Galessio, S., Klein, A., Makansi, O., Hutter, F., Brox, T.: Uncertainty estimates and multi-hypotheses networks for optical flow. In: ECCV (2018) **3, 13**
41. Kamann, C., Rother, C.: Benchmarking the robustness of semantic segmentation models with respect to common corruptions. IJCV (2021) **12**
42. Kendall, A., Badrinarayanan, V., Cipolla, R.: Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. arXiv preprint arXiv:1511.02680 (2015) **3**
43. Kendall, A., Gal, Y.: What uncertainties do we need in bayesian deep learning for computer vision? In: NeurIPS (2017) **1, 3, 9, 13, 14**
44. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images. Tech. rep. (2009) **11**

45. Lakshminarayanan, B., Pritzel, A., Blundell, C.: Simple and scalable predictive uncertainty estimation using deep ensembles. In: NeurIPS (2017) [1](#), [3](#), [10](#), [11](#), [12](#), [13](#), [14](#)
46. Lee, J.H., Han, M.K., Ko, D.W., Suh, I.H.: From big to small: Multi-scale local planar guidance for monocular depth estimation. arXiv preprint arXiv:1907.10326 (2019) [3](#), [13](#), [21](#), [23](#)
47. Liu, C.L., Nakagawa, M.: Evaluation of prototype learning algorithms for nearest-neighbor classifier in application to handwritten character recognition. PR (2001) [3](#)
48. 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. In: NeurIPS (2020) [2](#), [3](#), [4](#), [5](#), [10](#), [11](#), [19](#)
49. Macêdo, D., Ren, T.I., Zanchettin, C., Oliveira, A.L., Ludermir, T.: Entropic out-of-distribution detection. In: IJCNN (2021) [5](#), [6](#)
50. Macêdo, D., Zanchettin, C., Ludermir, T.: Distinction maximization loss: Efficiently improving classification accuracy, uncertainty estimation, and out-of-distribution detection simply replacing the loss and calibrating. arXiv preprint arXiv:2205.05874 (2022) [8](#)
51. Malinin, A., Gales, M.: Predictive uncertainty estimation via prior networks (2018) [9](#)
52. Malinin, A., Mlodzieniec, B., Gales, M.: Ensemble distribution distillation. In: ICLR (2020) [1](#), [9](#)
53. Mandelbaum, A., Weinshall, D.: Distance-based confidence score for neural network classifiers. arXiv preprint arXiv:1709.09844 (2017) [4](#), [8](#)
54. Mi, L., Wang, H., Tian, Y., Shavit, N.: Training-free uncertainty estimation for dense regression: Sensitivity as a surrogate. In: AAAI (2022) [3](#), [13](#)
55. Miyato, T., Kataoka, T., Koyama, M., Yoshida, Y.: Spectral normalization for generative adversarial networks. In: ICLR (2018) [5](#)
56. Mukhoti, J., Kirsch, A., van Amersfoort, J., Torr, P.H., Gal, Y.: Deterministic neural networks with appropriate inductive biases capture epistemic and aleatoric uncertainty. In: ICML Workshops (2021) [1](#), [2](#), [3](#), [4](#), [5](#), [9](#), [10](#), [11](#)
57. Mukhoti, J., Kulharia, V., Sanyal, A., Golodetz, S., Torr, P.H., Dokania, P.K.: Calibrating deep neural networks using focal loss. In: NeurIPS (2020) [1](#)
58. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning. In: NeurIPS (2011) [11](#)
59. Nix, D., Weigend, A.: Estimating the mean and variance of the target probability distribution. In: ICNN (1994) [3](#)
60. Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., Nowozin, S., Dillon, J., Lakshminarayanan, B., Snoek, J.: Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In: NeurIPS (2019) [3](#)
61. Padhy, S., Nado, Z., Ren, J., Liu, J., Snoek, J., Lakshminarayanan, B.: Revisiting one-vs-all classifiers for predictive uncertainty and out-of-distribution detection in neural networks. In: ICML Workshops (2020) [5](#)
62. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: NeurIPS (2019) [19](#)
63. Pinto, F., Yang, H., Lim, S.N., Torr, P., Dokania, P.K.: Mix-maxent: Improving accuracy and uncertainty estimates of deterministic neural networks. In: NeurIPS Workshops (2021) [2](#)

64. Poggi, M., Aleotti, F., Tosi, F., Mattoccia, S.: On the uncertainty of self-supervised monocular depth estimation. In: CVPR (2020) [3](#), [10](#)
65. Postels, J., Blum, H., Strümpler, Y., Cadena, C., Siegwart, R., Van Gool, L., Tombari, F.: The hidden uncertainty in a neural networks activations. In: ICML (2021) [2](#), [3](#), [4](#), [5](#), [10](#), [11](#), [12](#)
66. Postels, J., Segu, M., Sun, T., Van Gool, L., Yu, F., Tombari, F.: On the practicality of deterministic epistemic uncertainty. arXiv preprint arXiv:2107.00649 (2021) [1](#), [11](#)
67. Qi, H., Brown, M., Lowe, D.G.: Low-shot learning with imprinted weights. In: CVPR (2018) [5](#)
68. Rahimi, A., Recht, B., et al.: Random features for large-scale kernel machines. In: NeurIPS (2007) [19](#)
69. Razavi, A., Van den Oord, A., Vinyals, O.: Generating diverse high-fidelity images with vq-vae-2. In: NeurIPS (2019) [4](#)
70. Rebut, J., Bursuc, A., Pérez, P.: Styleless layer: Improving robustness for real-world driving. In: IROS (2021) [12](#)
71. Rogez, G., Weinzaepfel, P., Schmid, C.: LCR-Net++: Multi-person 2D and 3D pose detection in natural images. TPAMI (2019) [3](#)
72. Smola, A.J., Schölkopf, B.: Learning with kernels. Citeseer (1998) [19](#)
73. Snell, J., Swersky, K., Zemel, R.S.: Prototypical networks for few-shot learning. In: NeurIPS (2017) [5](#), [6](#)
74. Sun, D., Yang, X., Liu, M.Y., Kautz, J.: PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In: CVPR (2018) [3](#)
75. Teed, Z., Deng, J.: Raft: Recurrent all-pairs field transforms for optical flow. In: ECCV (2020) [3](#)
76. Uhrig, J., Schneider, N., Schneider, L., Franke, U., Brox, T., Geiger, A.: Sparsity invariant CNNs. In: 3DV (2017) [13](#)
77. Van Amersfoort, J., Smith, L., Teh, Y.W., Gal, Y.: Uncertainty estimation using a single deep deterministic neural network. In: ICML (2020) [2](#), [3](#), [4](#), [10](#), [11](#), [19](#)
78. Van Den Oord, A., Vinyals, O., et al.: Neural discrete representation learning. In: NeurIPS (2017) [4](#)
79. Wen, Y., Zhang, K., Li, Z., Qiao, Y.: A discriminative feature learning approach for deep face recognition. In: ECCV (2016) [3](#), [7](#)
80. Wen, Y., Tran, D., Ba, J.: BatchEnsemble: an alternative approach to efficient ensemble and lifelong learning. In: ICLR (2020) [1](#)
81. Wilson, A.G., Izmailov, P.: Bayesian deep learning and a probabilistic perspective of generalization. arXiv preprint arXiv:2002.08791 (2020) [3](#)
82. Wu, M., Goodman, N.: A simple framework for uncertainty in contrastive learning. arXiv preprint arXiv:2010.02038 (2020) [3](#), [4](#)
83. Yang, H.M., Zhang, X.Y., Yin, F., Liu, C.L.: Robust classification with convolutional prototype learning. In: CVPR (2018) [3](#), [4](#), [7](#)
84. Yu, X., Franchi, G., Aldea, E.: SLURP: Side learning uncertainty for regression problems. In: BMVC (2021) [2](#), [3](#), [10](#)
85. Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.: Pyramid scene parsing network. In: CVPR (2017) [13](#)

# Latent Discriminant deterministic Uncertainty

## Supplementary Material

Gianni Franchi<sup>1†</sup>, Xuanlong Yu<sup>1,2†</sup>, Andrei Bursuc<sup>3</sup>, Emanuel Aldea<sup>2</sup>, Severine Dubuisson<sup>4</sup>, and David Filliat<sup>1</sup>

<sup>1</sup> U2IS, ENSTA Paris, Institut polytechnique de Paris

<sup>2</sup> SATIE, Paris-Saclay University

<sup>3</sup> valeo.ai

<sup>4</sup> CNRS, LIS, Aix Marseille University

## Table of Contents

A	Connection with kernel methods .....	19
B	Implementation details .....	19
	B.1 Classification and semantic segmentation experiments .....	19
	B.2 Monocular depth experiments .....	21
C	Ablation study .....	21
D	Training dynamics .....	21
E	More visualizations .....	22

## A Connection with kernel methods

This section provides a connection between LDU (section 3.2) and kernel ridge regression. Let us consider the generic training of a kernel ridge regressor model [72]  $\mathbf{f}$  on the training set  $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ , where  $n$  is the number of training samples. Let us denote by  $k$  the kernel. The Representer Theorem [72] states that the solution to this problem is of the form  $\mathbf{f}(\mathbf{x}) = \sum_{i=1}^n \alpha_i k(\mathbf{x}_i, \mathbf{x})$ , where  $\alpha_i$  are the parameters to optimize.

Therefore, one should evaluate the kernel centered in each training sample. If one relates the proposed LDU model to kernel ridge regression, the main distinction is that we do not evaluate it across the entire training set, but across a smaller set composed of the prototypes. Hence  $\mathbf{f}(\mathbf{x}) = \sum_{i=1}^m \alpha_i k(p_i, h_\omega(\mathbf{x}_i))$  with  $k(\cdot, \cdot) = \exp(-S_c(\cdot, \cdot))$ , which is a composition of kernel positive definite functions. In contrast with DUQ [77] and similarly to SNGP [48], we approximate the kernel ridge regression based on the prototype set which allows us to simplify the training. However for SNGP, the authors choose to approximate the kernel differently, by relying on the random projection trick [68].

## B Implementation details

### B.1 Classification and semantic segmentation experiments

This section provides the hyper-parameters used in the classification and semantic-segmentation experiments. Our code is implemented in PyTorch [62]. We used

Hyper-parameter	CIFAR-10
backbone	ResNet18
initial learning rate	0.1
batch size	128
lr decay ratio	0.1
lr decay epochs	[80, 160, 200]
number of train epochs	250
weight decay	0.0001
cutout	False
SyncEnsemble BN	False

**Table A1.** Hyper-parameter configuration used in the classification experiments (§4.1).

Hyper-parameter	MUAD	Cityscapes	BDD-Anomaly
Architecture	Deeplab v3+	Deeplab v3+	PSPNet
backbone	ResNet50	ResNet50	ResNet50
output stride	8	8	None
learning rate	0.1	0.1	0.02
batch size	16	16	4
number of train epochs	50	50	30
nb Prototypes	22	22	30
weight decay	0.0001	0.0001	0.0001
SyncEnsemble BN	False	False	False
Cutout	False	False	False
random crop of training images	768	768	None

**Table A2.** Hyper-parameter configuration used in the semantic segmentation experiments (§4.2).

30 prototypes for BDD-Anomaly and 22 for the other semantic segmentation dataset.  $g_\omega^{\text{unc}}$  is for all the semantic segmentation dataset an MLP with one hidden layer, and the number of neurons in the hidden layer is equal to the number of prototypes. For classification,  $g_\omega^{\text{unc}}$  is a single fully connected layer. All the parameters are introduced in Table A1 for the classification and Table A2 for the semantic segmentation.

Hyper-parameter	KITTI
Architecture	BTS [46]
backbone	DenseNet161 [39]
initial learning rate	0.0001
batch size	4
number of train epochs	50
weight decay	0.01
random crop of training images	(352, 704)
nb Prototypes	30

**Table A3.** Hyper-parameter configuration used in the monocular depth estimation experiments (§4.3).

## B.2 Monocular depth experiments

As mentioned in the main paper, we train all models using the same hyper-parameters as in the original BTS code [46]. We use 30 prototypes in the DM layer for our LDU model. For Single-PU, we duplicate the top layer and double the number of output channels of the pre-logit layer. For the last layer, we have two one-channel-map outputs: one for depth estimation, one for uncertainty estimation. For the Deep Ensembles baseline, we train Single-PU models. We provide all our hyper-parameters in Table A3. We will make the code publicly available after the anonymity period.

## C Ablation study

In this section, we provide various ablation studies on evaluating the sensitivity of hyper-parameter choices for classification and regression tasks.

In Table A4 and Table A6 we report the results for different values of  $\lambda$ , the weight of the extra losses for classification on CIRFAR-10 and for monocular depth estimation on KITTI respectively. The performance across metrics is relatively stable with respect to the choice of  $\lambda$  with a good compromise at 0.1.

Concerning the influence of prototype number, according to Table 2 in the main paper and Table A6, the predictive performance is higher with more prototypes, while fewer prototypes make lighter and faster models.

We also study the impact of different losses in the classification task in Table A5. The three losses bring consistent improvements individually and together.

## D Training dynamics

In this section we illustrate the training curves when we train our models with/without applying LDU modifications. We take our regression experiment as an example,

$\lambda$	Acc $\uparrow$	AUC $\uparrow$	AUPR $\uparrow$	ECE $\downarrow$
0.01	87.93	0.8425	0.9079	0.5319
0.1	87.95	<b>0.8721</b>	<b>0.9147</b>	<b>0.4933</b>
0.5	87.99	0.8526	0.9109	0.5184
1.0	87.88	0.8399	0.9035	0.5005
2.0	<b>88.07</b>	0.8339	0.9008	0.4954

**Table A4.** Ablation studies for image classification on CIFAR-10. Sensitivity of  $\lambda$  values.

$\mathcal{L}^{\text{Unc}}$	$\mathcal{L}^{\text{Entrop}}$	$\mathcal{L}^{\text{Dis}}$	Acc $\uparrow$	AUC $\uparrow$	AUPR $\uparrow$	ECE $\downarrow$
✓			87.69	0.8195	0.8600	0.5256
✓	✓		87.87	0.8362	0.9090	0.5386
✓		✓	<b>88.04</b>	0.8613	0.8897	0.4973
✓	✓	✓	87.95	<b>0.8721</b>	<b>0.9147</b>	<b>0.4933</b>

**Table A5.** Ablation studies for image classification on CIFAR-10. Impact of proposed losses ( $\lambda=0.1$ ).

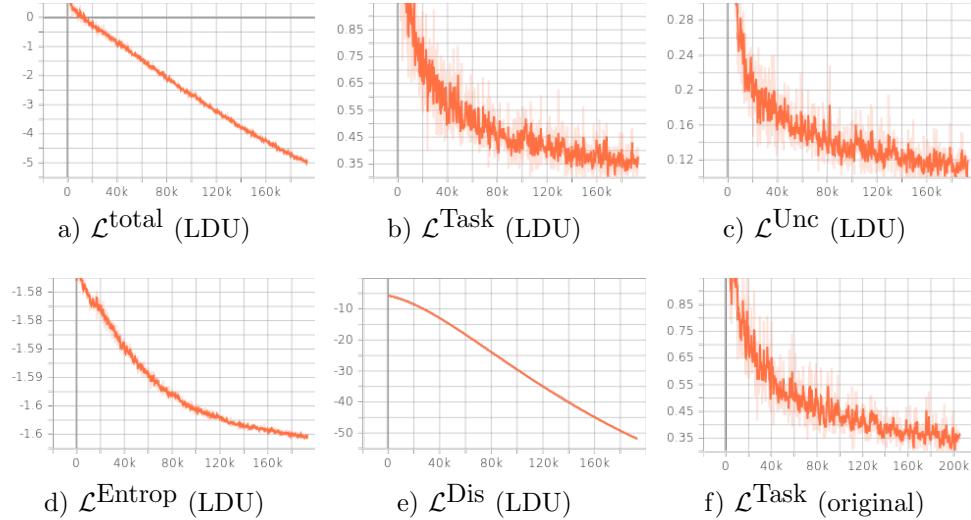
and plot the training curves in Fig. A1. Compared to the original setting, the inserted DM layers and additional losses did not affect the stability of training, all losses decrease smoothly.

## E More visualizations

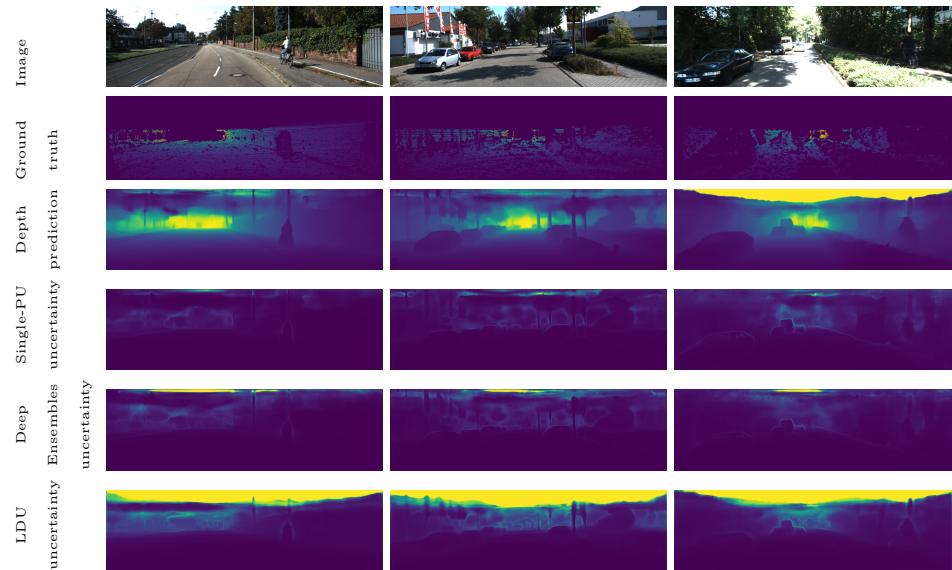
In Fig. A2 we present qualitative results depicting uncertainty for monocular depth estimation. We show side-by-side depth predictions and uncertainty maps generated by Single-PU, Deep Ensembles and LDU respectively. We observe that for the areas with valid ground truth, all uncertainty estimation strategies highlight the edges of the objects, where the aleatoric uncertainty is frequently prominent. Concerning the areas without valid ground truth, Deep Ensembles does a better job than Single-PU in highlighting them since it can capture more epistemic uncertainty due to the ensembling of multiple predictions from the individual models. Our proposed LDU highlights even better some distant areas, especially the upper part of the image where LiDAR beams do not go. We consider that this result stems from LDU regarding this region as OOD regions after training on the entire dataset.

#p	$\lambda$	d1↑	d2↑	d3↑	Abs Rel↓	Sq Rel↓	RMSE↓	RMSE log↓	log10↓	AUSE	AUSE
										RMSE↓	Absrel↓
30	0.01	<b>0.957</b>	<b>0.993</b>	0.998	<b>0.061</b>	<b>0.246</b>	2.768	<b>0.097</b>	<b>0.027</b>	0.12	0.29
	0.1	0.955	0.992	0.998	<b>0.061</b>	0.248	<b>2.757</b>	<b>0.097</b>	<b>0.027</b>	0.09	0.26
	0.5	0.952	<b>0.993</b>	0.998	0.064	0.256	2.789	0.100	0.028	0.09	0.28
	1.0	0.952	0.992	0.998	0.064	0.257	2.767	0.099	0.028	<b>0.08</b>	0.23
5	0.01	0.953	<b>0.993</b>	0.998	0.064	0.264	2.777	0.099	0.028	0.15	0.39
	0.1	0.953	0.992	0.998	0.064	0.256	2.773	0.100	0.028	0.12	0.33
	0.5	0.953	<b>0.993</b>	0.998	0.063	0.253	2.776	0.099	0.028	0.09	0.26
	1.0	0.954	<b>0.993</b>	0.998	0.063	0.253	2.768	0.098	<b>0.027</b>	<b>0.08</b>	<b>0.21</b>
15	0.1	0.954	<b>0.993</b>	0.998	0.062	0.249	2.769	0.098	<b>0.027</b>	0.10	0.28

**Table A6. Ablation studies for monocular depth estimation on KITTI.**  
Sensitivity of  $\lambda$  and the number of prototypes.



**Fig. A1.** Illustrations on training curves for different losses. Figure a) - e): training curves for the model with LDU modifications; Figure f): training curve for the original model. The  $\mathcal{L}^{\text{Task}}$  is silog loss for depth regression as defined in BTS [46].



**Fig. A2.** Illustration of different uncertainty maps (LDU, Deep Ensembles, Single-PU) on KITTI images for the monocular depth estimation task (§4.3). For both depth and uncertainty maps, the brighter the color is, the bigger the value the pixel has.