SLURP: Side Learning Uncertainty for Regression Problems

Xuanlong Yu[1]
xuanlong.yu@universite-paris-saclay.fr

Gianni Franchi[2]
gianni.franchi@ensta-paris.fr

Emanuel Aldea[1]
emanuel.aldea@universite-paris-saclay.fr

[1] SATIE, Paris-Saclay University
Gif-sur-Yvette, France

[2] U2IS, ENSTA Paris
Institut Polytechnique de Paris
Palaiseau, France

Abstract

It has become critical for deep learning algorithms to quantify their output uncertainties to satisfy reliability constraints and provide accurate results. Uncertainty estimation for regression has received less attention than classification due to the more straightforward standardized output of the latter class of tasks and their high importance. However, regression problems are encountered in a wide range of applications in computer vision. We propose SLURP, a generic approach for regression uncertainty estimation via a side learner that exploits the output and the intermediate representations generated by the main task model. We test SLURP on two critical regression tasks in computer vision: monocular depth and optical flow estimation. In addition, we conduct exhaustive benchmarks comprising transfer to different datasets and the addition of aleatoric noise. The results show that our proposal is generic and readily applicable to various regression problems and has a low computational cost with respect to existing solutions.

1 Introduction

The increasing use of Deep Neural Networks (DNNs) fuelled significant advances in the last decade in Computer Vision and fully benefited from the improvements in data availability and computational power. Currently, the main factor impeding a wider adoption of DNNs for critical tasks is their lack of reliability and interpretability. Subsequently, a lot of effort has been dedicated to the uncertainty estimation of model predictions during inference, which aims to alleviate the impact of out-of-distribution and noisy inputs. The most popular approaches are either based on ensembles, or on some simplificatory assumptions applied to Bayesian Neural Networks [12, 32, 47]. In both cases, the predictive uncertainty is related to the output diversity, the quality of which is correlated to a higher computational cost. Moreover, classification models have received more attention since their output is simpler and allows for an easier definition of uncertainty estimation metrics [18, 19]. In a regression setting, DNNs are expected uncertainty-wise to provide a probability distribution over the target domain, but most existing models also lack a default uncertainty output. Following the works addressing classification models, some strategies and metrics for uncertainty estimation and calibration have been proposed more recently for regression [16, 26]. However,

Figure 1: An example for uncertainty estimation on optical flow task. The predicted flow is made by FlowNetS [9]. The prediction error is the end-point-error between ground truth flow and predicted flow. We can see that our uncertainty estimation map can correspond well to the true error, including the semantic loss area (middle left) and the edges in the connected domains of the prediction flow map.

the proposed solutions tend to be more specific due to the higher variability of the output domain associated with diverse input data, *e.g* time series forecasting vs. pixel-wise regression. As a default solution, ensemble-based strategies are relatively effective but incur a significant training and inference computational cost, which is sometimes prohibitive when considered along with the memory requirements for some tasks.

Alternatively, using a side learner [7, 20, 28, 49] may provide a generic solution for predictive uncertainty estimation. The side learner could be regarded as a post-processing applied to the prediction result. Therefore, it is a straightforward approach to provide a precise uncertainty estimation without influencing the performance of the main task model or further refining its hyper-parameters. However, for some pixel-wise regression tasks, this approach may be challenging because of the importance of the semantic context. Compared with the uncertainty for time series data that pays more attention to long short-term information, the uncertainty of image-based tasks should give more consideration to the semantics of the image. The uncertainty map is based on the prediction map because of the similar higher level encoded semantics. However, obviously, the prediction result might miss relevant semantics with respect to the ground truth, which may have a detrimental effect on the uncertainty estimation. Our work intends to fill this gap by introducing SLURP, a general side learning approach for regression problems able to recover semantic information absent from the main task prediction. Our contributions are as follows. **1.** We are the first to solve the uncertainty estimation problem for general pixel-wise regression tasks with an auxiliary network. **2.** We propose a transposable architecture which may be used along with the main task model without modifying/re-training/affecting the performance of the latter, thus greatly improving our proposal's adoption potential. **3.** We demonstrate our side learner's flexibility on two fundamental vision tasks. The extensive experiments validate our algorithm's consistent performance in line with SOTA uncertainty estimation algorithms. The main advantage of our proposed solution is the efficiency and simplicity of its implementation and its competitive uncertainty performance.

2 Related Works

2.1 Major uncertainty estimation approaches

1. Distribution-based approaches By assuming a probability distribution over the model output and by minimizing the associated negative log-likelihood [29, 39], one may obtain the aleatoric uncertainty of the prediction result. Based on this idea, [1] separate the formulation and use it as a two-task learning which could improve the main task and uncertainty

performance on indoor depth estimation. As for the considered distribution, Gundavarapu *et al.* [15] choose to use a multi-variate Gaussian to capture the aleatoric uncertainty in human pose, while in their experiments [13, 26] choose the best performing distribution from an exponential distribution cluster. Apart from supervised methods, [40] learns uncertainty from monocular depth in a self-supervised way with a teacher-student framework using the concept of [29], while being too specific for re-implementation on the other tasks.

2. Ensemble-based approaches In the statistics community [10, 43] and later in computer vision [26, 51], the most conventional way to capture model uncertainty has been to use bootstrapping in which the uncertainty is linked to the variance of multiple outputs generated by re-sampling. Although it is accessible and well studied, this approach is not well suited to deep learning as it tends to degrade the main task performance. Instead of using part of the data each time, Deep Ensembles [32] uses the full data to train the model under different stochasticity sources and combines the aleatoric uncertainty estimation proposed by [29]. While this method is easily adaptable to different tasks while still achieving an extremely competitive baseline [2], its memory and run-time consumption make it inefficient in real applications due to its dependence on multiple, potentially heavy models. Eddy Ilg *et al.* [26] use a multiple-hypothesis network combining with the concept of [29] to reduce memory consumption of ensembles. However, it is specific for optical flow, it has to modify the main task model and add an extra network in the end for merging the hypotheses, and it requires multiple stages of training.

3. Bayesian approaches Learning the posterior distribution of the weights of the main task neural network given the training dataset is a way to obtain the uncertainty from the network architecture [3]. Gast *et al.* [13] try to weight the uncertainty from network weight by using assumed density filtering [4], but in this case the network has to be modified. MC-Dropout [12] is a widely used method which mimics a BNN by using injected Dropout layers to sample the network weights. It needs to do multiple times forward propagation which is costly especially for pixel-wise regression tasks. At the same time, an additional challenge is raised by properly balancing the main performance and the quality of the generated uncertainty due to the number and position of the injected Dropout layers.

2.2 Side learning for uncertainty estimation

With the development of deep learning, DNNs are becoming more sophisticated, and the training of DNN and the selection of hyperparameters have also become particularly important [45]. As it avoids changing the main task model structure, a side learner is in this respect a convenient approach to estimate uncertainty. Lee *et al.* [34] use a conditional GAN [27] as an auxiliary model to project an image-pair input in optical flow to an uncertainty provided by a MC-Dropout version of main task to reduce the time cost, but its performance will be decreased during dataset shift. Yoo *et al.* [49] design a side learner to learn the loss supplied by the main task model. However, the goal of this work is to provide *a single predicted loss* for the main task, which is not suitable for pixel-wise uncertainty estimation. Instead of predicting the loss from the main task model, based on [32] and [29], Hu *et al.* [20] trains sequentially a network which is identical to the main task model to fit the prediction error. It can improve the stability of uncertainty training but the memory cost of the uncertainty estimator will increase with the increase of the main task model scale. DEUP [28] not only uses a side learner to predict the loss but in order to better obtain the epistemic uncertainty, it enriches the input of the side learner but it needs to train three models including a main task model with aleatoric uncertainty estimator, a data density estimator and a loss side learner.

This has high requirements for model selection and training. Corbière *et al.* [7] introduces ConfidNet which can conveniently provide the uncertainty by adding an auxiliary network after the main task encoder. It shares a more similar approach with our work. It shows a good performance but it works only for classification task and it has to re-tune the encoder part for uncertainty prediction so it structurally depends on the main task network and requires two-stage training.

For pixel-wise regression tasks, the number of data points is huge and data preparation will be costly. SLURP does not need to modify the main task model, does not require data preparation. It faces on general pixel-wise regression tasks, which is different from all the works mentioned above. We use a direct and explicit design, and only train the side learner once to get the uncertainty. This can circumvent the difficulty of modifying the main task model caused by the complexity of the structure, or the lack of the training codes. Using only the in-distribution data, without touching the main task model, we can get better quality and robust uncertainty. Fig. 1 gives an example on uncertainty estimation of our work.

3 Our method

3.1 Problem description and motivation

Our training procedure is a two-step process that first consists of training a DNN f to perform its main regression task. Then in a second stage, the parameters in f are fixed and will not be updated any more. Based on a primary DNN f, we train a second DNN g to predict the uncertainty of the first DNN.

Task 1: given dataset $D = \{(\mathbf{x}_i, y_i)\}_i$, we write $P(Y|\mathbf{x})$ the conditional distribution for the ground truth value given an input value $\mathbf{x}$ (*e.g* an RGB image). Let us denote f, the main task predictor, which is trained by minimizing the objective function $l_f(f(\mathbf{x}), y)$ over the dataset.

Task 2: Once the DNN f is trained, we propose to add a new task, namely the prediction error estimation. Our goal is to predict the error done by the DNN, i.e., to learn to predict $l_f(f(\mathbf{x}), y)$. Note that the loss does not need to be the same as l_f, since in some cases, l_f could follow a specific design such as in focal loss [35], in scale-invariant error [11], etc. Hence we aim to predict the error loss $l_u(f(\mathbf{x}), y)$. As is the case with regression tasks, a sensible choice for l_u is the mean square or absolute error. Our uncertainty DNN g, will have the following training objective:

$$\underset{g}{\operatorname{argmin}} L\Big(l_u(f(\mathbf{x}), y), g(f(\mathbf{x}), \mathbf{x})\Big) \tag{1}$$

where L denotes an objective function which needs to be minimized to obtain a model able to predict the error of f.

Note that the error prediction task of **task 2**, is related to predicting the total uncertainty of the DNN. The total uncertainty $U(f, \mathbf{x})$ may then be interpreted as the sum of prediction errors according to [28]:

$$U(f, \mathbf{x}) = \int l_u(f(\mathbf{x}), y) dP(y|\mathbf{x}) \tag{2}$$

In subsection 3.2, we explain the structure of the side learner g, and in subsection 3.3, we present the design of the loss L to learn to predict the uncertainty.
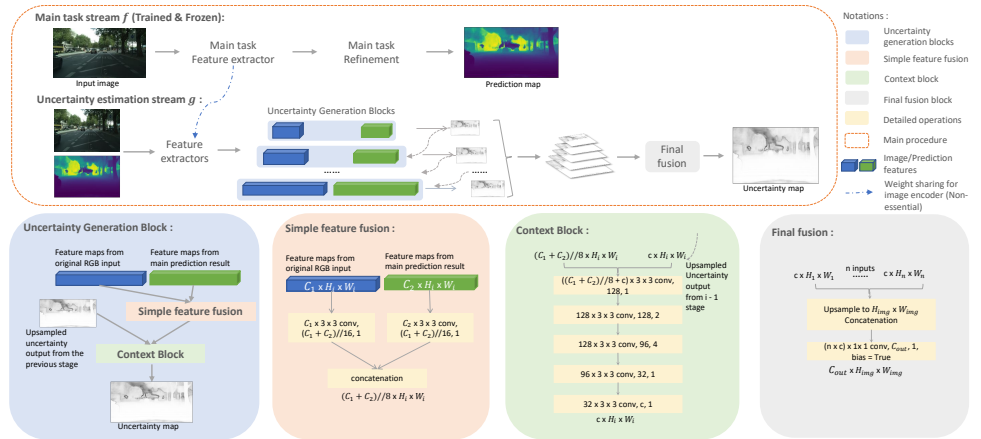
Figure 2: General solution for pixel-wise uncertainty estimation. We take monocular depth estimation as an example. In main procedure, **Main task stream f is first trained and frozen.** Then we use the input and the output of f to train g. An uncertainty generation block is implemented on feature pairs from different stages. The n outputs of context blocks from different stages will be sent to the final fusion block. The conv-layers in detailed operations are described as (shape, number, dilation ratio).

3.2 Side learner's architecture

The side learner g takes two inputs: $\mathbf{x}$ and $f(\mathbf{x})$. The combination of image features and prediction result in uncertainty map that depends on the initial image and on the prediction of f. The design of this architecture is inspired by empirical observations on the prediction error maps. We notice that, for pixel-wise regression tasks, prediction errors are organized 1. along edges of connected domains in the prediction map; 2. as hard predictable areas which are not captured in the prediction map but only exist in the RGB image, e.g. distant or small objects and occlusions. Therefore, the RGB image needs to be combined with the prediction map to recover the semantic information absent from the main task prediction. Meanwhile, we use convolutional features given by the encoders with a final fusion block to better capture the edge information [36]. The concatenated convolutional features are followed by a context block [44] described as follows.

We structure the side learner g in the following three parts, with Figure 2 showing the general design for our pixel-wise uncertainty estimator.

1. Feature encoders: They aim to learn and extract the richer convolutional feature pyramids [36] from raw input (RGB data) and from the final prediction map preparing for the next steps. We choose the widely used backbone DenseNet [23]. Note that the architecture is agnostic, and DenseNet could be replaced by the other backbones like ResNet [17], depending on the context. The image encoder in g can be trained from scratch but can also be replaced by the trained and frozen image encoder from f. It depends on how closely the image input is related to the prediction error. If we have multiple images as a concatenated input, we need to encode the image on which the prediction error is based. We recommend using the same structure for the feature extractors of the image and the prediction map, so as to avoid information asymmetry. The following operations will be done from coarse to fine.

2. Feature fusion: For each pair of RGB and final prediction features, we concatenate them followed by the convolutional layers in order to reduce the channel number. Dropout layers are implemented before the features are input to the convolutional layer, to reduce the bias
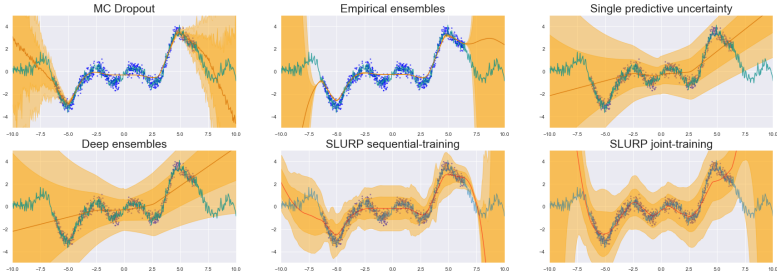
Figure 3: 1D synthetic regression task comparison example. X-axis: spatial coordinate of the Gaussian process. Green curve: ground truth; Blue points: training samples; Red curves: main task prediction; Orange zooms: the uncertainty coverage (1-sigma for inner interval, 2-sigma for outer interval).

caused by focusing on specific patterns. The goal is to find the similarity and the difference between two sources of features to bring out the lost information during main task training.

3. Context block: The architecture of this block follows that of the context network in PWC-net [44]. With various dilation ratios, the convolutional layers can take the features from different size of receptive fields. The output intermediate uncertainty from the previous stage will be up-sampled and concatenated as a guide feature to the input of current stage.

After obtaining the output of the context block with different resolutions, we up-sample these outputs toward the same size as the prediction target and then use a simple convolutional layer to sum them up with different weights as a final fusion output.

3.3 Loss design

1. Natural loss: A straightforward way is to use mean square error loss (MSE) for l_u in eq. 1 and eq. 2. In order to prevent the distribution of training target from being too sparse, we take $l_u(f(\mathbf{x}), y) = |f(\mathbf{x}) - y|$.

2. Target scaling: The uncertainty of the regression task is expressed across a support based on the current value of the variable, with the support being contained in $\mathbb{R}$. The uncertainty of the classification task is related to the confidence of the classifier output, which is set to take values in the range is $[0, 1]$. Hence we process $l_u(f(\mathbf{x}), y)$ to bound it between 0 and 1. This can also avoid the influence of the effect from the outliers which have significant prediction errors thanks to the following equation :

$$\tilde{l}_u(f(\mathbf{x}), y) = \tanh(\lambda * l_u(f(\mathbf{x}), y)) \tag{3}$$

where λ is a stretch hyper-parameter that can spread the training target as much as possible between 0 and 1, and $\tilde{l}_u(f(\mathbf{x}), y)$ is the normalized uncertainty target.

3. Cross-Entropy Loss: After having normalized the uncertainty l_u, we select cross entropy loss to describe the probability distance between training target $\tilde{l}_u$ and our predicted uncertainty. Since $\tilde{l}_u \in [0, 1]$, we choose a binary cross entropy (BCE) loss as L in eq. 1:

$$L(\tilde{l}_u, g) = -\sum_i [\tilde{l}_u(f(\mathbf{x}_i), y_i) \log(\sigma(g(f(\mathbf{x}_i), \mathbf{x}_i))) + (1 - \tilde{l}_u(f(\mathbf{x}_i), y_i)) \log(1 - \sigma(g(f(\mathbf{x}_i), \mathbf{x}_i)))] \tag{4}$$

where $\sigma(.)$ is the sigmoid function. Note that here we use BCE loss not for doing classification, but for regression, and BCE can support a faster convergence for a Sigmoid output value. By calculating the first derivation of eq. 4 with respect to $\sigma(g)$, we can find that the optimum will be $\sigma(g) = \tilde{l}_u$.

4 Experiments

Our main focus is on obtaining high-quality uncertainty maps on pixel-level regression tasks. However, in order to illustrate more comprehensively our approach and show its applicability in a different context, we also apply SLURP on a 1D toy dataset. Overall, our proposed method is illustrated on a synthetic 1D regression dataset, and on two fundamental computer vision tasks: optical flow (OF) and monocular depth (MD). For the former, we just visualize it to give some reference and insights. For the later ones, we evaluate the quality of predicted uncertainty maps, the basic idea is to see whether the predicted uncertainty map matches the prediction error. To this end, we re-implemented the transferable uncertainty estimation approaches MC-Dropout (MC) [12], Single predictive uncertainty (Single-PU) [29], Empirical ensembles (EE), ConfidNet (Confid) [7] and Deep ensembles (DE) [32] to the main task as the comparisons. Due to its particularity, we reproduce the multi-hypothesis prediction network (MHP) [26] only for the optical flow task. Specifically, to transfer Confid solution from the classification task to the regression task, we duplicate the last few layers from the last de-convolutional layer (or up-sample operation) and add three extra 3x3 convolutional layers without changing the resolution as ConfidNet for regression. We keep the training schedule, and the pixel-wise square error maps will replace the original training targets. We use two evaluation criteria: AUSE and AUROC to evaluate uncertainty maps generated by different methods. Additionally, we measure the efficiency of main task - uncertainty estimator system from two perspectives: Runtime and Number of parameters. The specific formulas as well as the implementation details for the methods we compare with are provided in the supp. material.

4.1 1D regression task toy example

We compare our proposed solution with MC [12], EE, Single-PU [29] and DE [32] on a 1D regression task dataset. The toy dataset is generated by Gaussian process, our spatial coordinate range is $x_i \in [-10,10]$. From $x_i \in [-7,7]$, we cross-select 875 data points as training set and 175 data points as validation set. From $x_i \in [-10,10]$, we randomly select 400 points as test set. The main task predictor f with single output has only one hidden layer, the Single-PU and DE are implemented based on a modified dual-output f. The detailed model and training settings for all the methods are in the supp. material. Moreover, we use two modes to train our SLURP side learner g. One is that we first train one f, then we freeze it and train the g with the prediction result from f and the latent feature extracted from the single hidden layer in f, i.e., sequential-training, which is the original design of our approach. Another one is that we train f and g at the same time by using the negative log-likelihood loss [29], i.e., joint-training. Because of the data dimension, the side learner shown in Fig 2 is modified (see supp. material). As illustrated in Fig 3, the results of MC [12] and EE give a good uncertainty on the unseen area, but little on the training part. While DE [32] and Single-PU [29] can give a sufficient uncertainty to all areas. But in comparison, their main task prediction accuracy has been affected. SLURP can achieve both reasonable main task accuracy and tight uncertainty coverage especially for joint-training one.

4.2 Evaluation protocols for pixel-wise tasks

MD and OF are two fundamental regression tasks which have significant implications in a wide range of applications. In terms of inputs, the main difference is that MD requires a single RGB image, while OF needs an image pair. For each pixel, the MD output is a depth value $y_d \in \mathcal{R}^+$, while the OF output is a 2-channel displacement vector $y_o \in \mathcal{R}^2$. We intro-

duce below the evaluation criteria.

1. Uncertainty ordering: Let us consider that we want to remove the worst pixels based on an uncertainty estimator; we expect that a good uncertainty estimator should allow us to remove the less reliable data. This is evaluated thanks to the **sparsification curve (SC)**, and the **area under sparsification error (AUSE)** [5, 26, 30, 51, 40]. To build the SC, given a set of data and their uncertainty, we iteratively erase $m\%$ (we take $m = 5$ in our experiments) of the data which exhibit the highest uncertainty. Then we calculate the average prediction error for the remaining data. Hence we have the SC. To evaluate the oracle SC, we remove the $m\%$ data with the most significant prediction error and we calculate the average prediction error for the remaining data. We denote the area between the two SC curves as AUSE. The smaller the AUSE, the closer the order of the predicted uncertainty and the order of the Oracle. As a note, AUSE could be changed if we change the error metric. Therefore, we denote AUSE based on different error metrics as AUSE-xxx.

Implementations: Specifically, for monocular depth, we choose square error and absolute relative error [11] .The corresponding AUSEs are **AUSE-RMSE** and **AUSE-Absrel**. For optical flow we use EPE, which is the error map representing the Euclidean distance between the ground truth motion and the predicted one, we denote its AUSE as **AUSE-EPE**.

2. AUROC: Since we have access to a soft evaluation of uncertainty, it is feasible to threshold the dataset into two sets, namely the reliable set and the unreliable set. We propose for MD to set the data as reliable if they check the inlier metrics threshold d1 criterion proposed in [22], and for OF data is reliable if its EPE is below $k = 2$. We scale the predicted uncertainty between 0 and 1 with min-max scaling and evaluate the ROC curve. The larger the AUROC, the more data points are given correct confidence (uncertainty) and the better the uncertainty estimator is.

3. Model efficiency: Due to the different schemes of uncertainty generation design, we will measure the number of model parameters and time consumption of the entire system (**# Param.**), including the main task model and uncertainty generator. We count the running time (**Runtime**) while processing a whole testing dataset in one NVIDIA TITAN RTX GPU and Intel Core i9-10900X CPU then take the average according to the number of samples.

4.3   Monocular depth

In this section, we introduce uncertainty estimations for MD. We choose BTS [33] as the depth estimator, and the uncertainty estimators will be implemented based on this architecture. BTS is one of the state-of-the-art architectures on MD benchmarks [14, 42]. As an encoder-decoder based network it is well suited for the extraction of latent image features as shown in Fig 2, however note that our strategy is agnostic to the main task architecture. In accordance with the default setting, we choose DenseNet161 [23] as encoder for BTS. So for our side learner, we take the trained and fixed image encoder of BTS as our image encoder, and implement a new DenseNet161 encoder for estimated depth map.

Datasets and procedures We choose two widely used datasets and their variations to train and evaluate uncertainty estimators as follows. *Training set:* KITTI [14, 46] Eigen-split training set [11]; *Fine-tuning set:* Cityscapes training set [8]; *Test set:* KITTI Eigen-split test set, Cityscapes test set, Foggy Cityscapes-DBF test set [41] with three severity levels and Rainy Cityscapes test set [21] with three parameter sets which indicate three severities. KITTI depth dataset and Cityscapes dataset are both outdoor datasets. However, since the ground truth on KITTI is sparse and has only half the content of the image, the model performance is limited if training on KITTI and testing on Cityscapes. Therefore, we first train

Datasets	Criteria	MC	EE	Single PU	DE	Confid	Ours
	AUSE-RMSE	8.14	3.17	1.89	**1.68**	1.76	**1.68**
KITTI	AUSE-Absrel	9.48	5.02	4.59	4.32	**4.24**	4.36
	AUROC	0.686	0.882	0.882	**0.897**	0.892	0.895
	AUSE-RMSE	**9.42**	11.56	9.91	11.47	10.48	9.48
CityScapes	AUSE-Absrel	9.52	13.14	9.96	9.36	**5.75**	10.90
	AUROC	0.420	0.504	0.386	0.501	**0.519**	0.400
After fine-tuning on CityScapes							
	AUSE-RMSE	7.72	8.20	4.35	**3.03**	4.05	3.05
CityScapes	AUSE-Absrel	8.13	7.50	6.44	6.81	6.34	6.55
	AUROC	0.705	0.786	0.741	**0.856**	0.821	0.849
CityScapes Rainy s=1	AUSE-RMSE	7.06	7.29	4.17	3.42	4.89	**3.39**
	AUSE-Absrel	8.73	6.92	6.55	6.68	7.26	**5.62**
	AUROC	0.659	0.757	0.731	0.746	0.697	**0.788**
CityScapes Rainy s=2	AUSE-RMSE	7.14	6.9	4.27	**3.35**	4.68	3.36
	AUSE-Absrel	8.36	6.48	6.79	6.24	6.86	**5.28**
	AUROC	0.667	0.767	0.731	0.756	0.714	**0.794**
CityScapes Rainy s=3	AUSE-RMSE	7.30	6.66	4.35	**3.28**	4.59	3.41
	AUSE-Absrel	8.27	6.03	6.44	5.85	6.64	**5.05**
	AUROC	0.665	0.778	0.742	0.767	0.729	**0.801**
CityScapes Foggy s=1	AUSE-RMSE	7.80	7.82	3.42	3.05	3.98	**3.04**
	AUSE-Absrel	8.36	7.33	6.78	6.58	**6.21**	6.25
	AUROC	0.700	0.783	0.842	**0.852**	0.824	0.847
CityScapes Foggy s=2	AUSE-RMSE	7.82	7.53	3.42	**2.98**	3.86	3.01
	AUSE-Absrel	8.20	7.09	6.55	6.35	**6.02**	6.06
	AUROC	0.704	0.791	0.847	**0.857**	0.833	0.852
CityScapes Foggy s=3	AUSE-RMSE	7.84	7.28	3.48	**2.93**	3.70	3.08
	AUSE-Absrel	7.87	6.80	6.19	6.01	**5.78**	5.80
	AUROC	0.715	0.801	0.851	**0.863**	0.846	0.857

Table 1: MD uncertainty performance. Bold value: result with the best performance, Blue value: second performance. s (*e.g* s=1) indicates severity, higher the s value, higher the severity.

models on KITTI Eigen-split training set then evaluate the uncertainty on KITTI Eigen-split test set and Cityscapes test set. After that, we fine-tune the models on Cityscapes training set and evaluate the uncertainty on all Cityscapes test sets listed above. **Results** Table 1 presents the performance of different methods in normal circumstances and against gradual input perturbations. The top performing method is highlighted in bold, while the second one is highlighted in blue. SLURP, ConfidNet and Deep ensembles exhibit competitive performance being both ranked close in terms of uncertainty ordering on the different metrics, with our proposal being clearly more robust against strong input perturbations.

4.4 Optical flow

FlowNetS [9] is a popular OF architecture, as the first learning-based optical flow estimation method. We apply it as our main task model and implement the uncertainty estimators based on it, since it can be regarded as a good example for uncertainty estimation on early structures with special inputs. FlowNetS is also an encoder-decoder network, however - differently from the more recent architectures with a separate feature extractor implemented for the first image [24, 25, 40, 44, 48], FlowNetS takes directly image pairs as encoder input, so the features from different encoder stages will be mixture feature maps of two RGB images. Since the ground truth motion is based on the first image, the true error will be based only on the first image, we use a new encoder to extract the RGB features only for the first image. In this case, in our side learner, we use two trainable encoders respectively for the first image and the predicted flow. We choose both DenseNet161 and DenseNet121 [23] as the encoders in our experiments and we denote the latter Ours-light as a lightweight version of the former. **Datasets** *Training set:* synthetic FlyingChairs training set [9]; *Testing set:* FlyingChairs test set, synthetic Sintel training set [6] and real-world KITTI 2015 training set [14, 37, 38]. The train-test split file for FlyingChairs dataset is provided officially. Sintel has more complex moving objects and movements than FlyingChairs, while KITTI is taken from real-world,

Datasets	Criteria	MC	EE	Single PU	DE	Confid	MHP	Ours-light	Ours
FlyingChairs	AUSE-EPE	2.75	1.97	1.28	1.26	1.92	1.88	**1.16**	1.20
	AUROC	0.896	0.900	**0.977**	0.959	0.945	0.936	**0.977**	0.974
KITTI	AUSE-EPE	3.57	4.41	3.71	**3.45**	6.56	5.48	5.20	4.69
	AUROC	0.870	**0.904**	0.848	0.866	0.687	0.854	0.791	0.800
Sintel Clean	AUSE-EPE	3.33	2.89	2.74	3.02	5.28	**2.61**	3.02	2.91
	AUROC	0.861	0.825	**0.925**	0.895	0.767	0.886	0.889	0.896
Sintel Final	AUSE-EPE	3.30	3.02	3.09	3.05	6.06	**2.71**	2.95	2.86
	AUROC	0.858	0.814	**0.916**	0.899	0.728	0.878	0.901	0.906

Table 2: OF uncertainty performance. Bold value: result with the best performance. Blue value: second performance.

Task	Criteria	MC	EE	Single-PU	DE	Confid	MHP	Ours-light	Ours
MD	Runtime (ms)	386	144	98	286	106	-	-	**88**
	# Param. (M)	**47.0**	141.0	94.0	282.0	94.7	-	-	87.2
OF	Runtime (ms)	79	65	**64**	66	65	67	65	76
	# Param. (M)	**38.7**	116.1	**38.7**	116.3	78.6	78.8	57.0	105.3

Table 3: Average time cost for processing one image and number of parameters of the model(s) (main task + uncertainty task). Bold value: result with the best performance. Blue value: second performance.

and exhibits larger movement magnitude.

Results Table 2 shows the uncertainty estimation performance on the various OF datasets. Even though all uncertainty estimators are not fine-tuned on datasets other than the training set, they maintain a good estimation level. On this very different scenario, MHP did a good job due to its pertinence, but our side learner performs competitively and robustly across all tests/metrics.

4.5 Model efficiency

Datasets and settings We select to use Sintel training set [6] and KITTI [14] Eigen-split validation set [11] as the testing sets for Runtime evaluation. We use three models to form EE and DE and eight forward propagations for MC. In MD, due to the instability of training, we use sequential-training for Single-PU and DE. More details are in the supp. material.

Results Table 3 shows the model efficiency in two tasks. In OF task, due to the convenience of modifying the main task model, we confirm that Single-PU is more efficient than the other methods, while our lightweight version can achieve comparable performance. For MD, due to the complexity in implementation for joint uncertainty generation, the side-learner-based methods have a fixed computational footprint and take significant advantage in the cost. In addition, our solution can reuse without re-tuning the main task encoder, so it is lighter and faster than the other auxiliary networks.

5 Conclusion

In this work we proposed a novel solution for estimating the total uncertainty of a DNN. SLURP is a side learner that does not affect the main prediction estimator. Given a trained main task model, we use the original input and main task prediction results as input, and the prediction error as the target. Following feature extraction, fusion and content reconstruction, we train for the total uncertainty of the main task prediction. SLURP can also exploit latent features of the original input provided by a frozen feature extractor of the main task model, such as an image encoder. We compared our proposal with popular uncertainty estimation approaches, and performed detailed experiments on 2D pixel-wise regression tasks to prove that our method is feasible and robust. Future works involve further optimizing the SLURP structure to decrease its footprint and make it accessible for embedded processing, as well as applying this side learning concept to other types of input signals.

References

[1] Akari Asai, Daiki Ikami, and Kiyoharu Aizawa. Multi-task learning based on separable formulation of depth estimation and its uncertainty. In *CVPR Workshops*, pages 21–24, 2019.

[2] Arsenii Ashukha, Alexander Lyzhov, Dmitry Molchanov, and Dmitry Vetrov. Pitfalls of in-domain uncertainty estimation and ensembling in deep learning. *arXiv preprint arXiv:2002.06470*, 2020.

[3] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International Conference on Machine Learning*, pages 1613–1622. PMLR, 2015.

[4] Xavier Boyen and Daphne Koller. Tractable inference for complex stochastic processes. *arXiv preprint arXiv:1301.7362*, 2013.

[5] Andrés Bruhn and Joachim Weickert. A confidence measure for variational optic flow methods. In *Geometric Properties for Incomplete Data*, pages 283–298. Springer, 2006.

[6] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In A. Fitzgibbon et al. (Eds.), editor, *European Conf. on Computer Vision (ECCV)*, Part IV, LNCS 7577, pages 611–625. Springer-Verlag, October 2012.

[7] Charles Corbière, Nicolas Thome, Avner Bar-Hen, Matthieu Cord, and Patrick Pérez. Addressing failure prediction by learning model confidence. *arXiv preprint arXiv:1910.04851*, 2019.

[8] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[9] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766, 2015.

[10] Bradley Efron and Robert J Tibshirani. *An introduction to the bootstrap*. CRC press, 1994.

[11] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. *arXiv preprint arXiv:1406.2283*, 2014.

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

[13] Jochen Gast and Stefan Roth. Lightweight probabilistic deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3369–3378, 2018.

[14] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11): 1231–1237, 2013.

[15] Nitesh B Gundavarapu, Divyansh Srivastava, Rahul Mitra, Abhishek Sharma, and Arjun Jain. Structured aleatoric uncertainty in human pose estimation. In *CVPR Workshops*, volume 2, page 2, 2019.

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

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

[18] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

[19] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

[20] Shi Hu, Nicola Pezzotti, and Max Welling. A new perspective on uncertainty quantification of deep ensembles. *arXiv e-prints*, pages arXiv–2002, 2020.

[21] Xiaowei Hu, Chi-Wing Fu, Lei Zhu, and Pheng-Ann Heng. Depth-attentional features for single-image rain removal. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8022–8031, 2019.

[22] Yinlin Hu, Yunsong Li, and Rui Song. Robust interpolation of correspondences for large displacement optical flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 481–489, 2017.

[23] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[24] Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. Liteflownet: A lightweight convolutional neural network for optical flow estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8981–8989, 2018.

[25] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2462–2470, 2017.

[26] Eddy Ilg, Ozgun Cicek, Silvio Galesso, Aaron Klein, Osama Makansi, Frank Hutter, and Thomas Brox. Uncertainty estimates and multi-hypotheses networks for optical flow. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 652–667, 2018.

[27] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

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

[29] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[30] Claudia Kondermann, Rudolf Mester, and Christoph Garbe. A statistical confidence measure for optical flows. In *European Conference on Computer Vision*, pages 290–301. Springer, 2008.

[31] Jan Kybic and Claudia Nieuwenhuis. Bootstrap optical flow confidence and uncertainty measure. *Computer Vision and Image Understanding*, 115(10):1449–1462, 2011.

[32] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *arXiv preprint arXiv:1612.01474*, 2016.

[33] Jin Han Lee, Myung-Kyu Han, Dong Wook Ko, and Il Hong Suh. From big to small: Multi-scale local planar guidance for monocular depth estimation. *arXiv preprint arXiv:1907.10326*, 2019.

[34] Serin Lee, Vincenzo Capuano, Alexei Harvard, and Soon-Jo Chung. Fast uncertainty estimation for deep learning based optical flow. 2020.

[35] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[36] Yun Liu, Ming-Ming Cheng, Xiaowei Hu, Kai Wang, and Xiang Bai. Richer convolutional features for edge detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3000–3009, 2017.

[37] Moritz Menze, Christian Heipke, and Andreas Geiger. Joint 3d estimation of vehicles and scene flow. In *ISPRS Workshop on Image Sequence Analysis (ISA)*, 2015.

[38] Moritz Menze, Christian Heipke, and Andreas Geiger. Object scene flow. *ISPRS Journal of Photogrammetry and Remote Sensing (JPRS)*, 2018.

[39] D. Nix and A. Weigend. Estimating the mean and variance of the target probability distribution. *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, 1:55–60 vol.1, 1994.

[40] Matteo Poggi, Filippo Aleotti, Fabio Tosi, and Stefano Mattoccia. On the uncertainty of self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3227–3237, 2020.

[41] Christos Sakaridis, Dengxin Dai, Simon Hecker, and Luc Van Gool. Model adaptation with synthetic and real data for semantic dense foggy scene understanding. In *European Conference on Computer Vision (ECCV)*, pages 707–724, 2018.

[42] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *European conference on computer vision*, pages 746–760. Springer, 2012.

[43] Harald Steck and Tommi S Jaakkola. Bias-corrected bootstrap and model uncertainty. In *NIPS*, pages 521–528. Citeseer, 2003.

[44] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8934–8943, 2018.

[45] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Models matter, so does training: An empirical study of cnns for optical flow estimation. *IEEE transactions on pattern analysis and machine intelligence*, 42(6):1408–1423, 2019.

[46] Jonas Uhrig, Nick Schneider, Lukas Schneider, Uwe Franke, Thomas Brox, and Andreas Geiger. Sparsity invariant cnns. In *International Conference on 3D Vision (3DV)*, 2017.

[47] Yeming Wen, Dustin Tran, and Jimmy Ba. Batchensemble: an alternative approach to efficient ensemble and lifelong learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[48] Gengshan Yang and Deva Ramanan. Volumetric correspondence networks for optical flow. *NeurIPS*, 5:12, 2019.

[49] Donggeun Yoo and In So Kweon. Learning loss for active learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 93–102, 2019.